

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
25 February 2010 (25.02.2010)

PCT

(10) International Publication Number
WO 2010/022400 A2

(51) International Patent Classification:
G01N 33/68 (2006.01) **G06F 19/00** (2006.01)

(21) International Application Number:
PCT/US2009/054796

(22) International Filing Date:
24 August 2009 (24.08.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/091,039 22 August 2008 (22.08.2008) US
61/191,609 9 September 2008 (09.09.2008) US
12/415,741 31 March 2009 (31.03.2009) US

(71) Applicant (for all designated States except US): **THE UNITED STATES OF AMERICA, AS REPRESENTED BY THE SECRETARY OF AGRICULTURE** [US/US]; 1400 Independence Ave. SW, Washington, DC 20250-0302 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **FAGERQUIST, Clifton, K.** [US/US]; 800 Buchanan St., Albany, CA 94710 (US). **HARDEN, Leslie, A.** [US/US]; 800 Buchanan St., Albany, CA 94710 (US). **GARBUS, Brandon, R.** [US/US]; 6102 Chene Ct., Lutz, FL 33558 (US).

(74) Agents: **SMITH, Sarah E.** et al.; Akerman Senterfitt, P.O. Box 3188, West Palm Beach, FL 33402-3188 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report (Rule 48.2(g))

— with sequence listing part of description (Rule 5.2(a))

(54) Title: RAPID IDENTIFICATION OF PROTEINS AND THEIR CORRESPONDING SOURCE ORGANISMS BY GAS PHASE FRAGMENTATION AND IDENTIFICATION OF PROTEIN BIOMARKERS

(57) Abstract: Embodiments of the present invention relate to the identification of proteins using laser desorption ionization mass spectrometry, the identification of source organisms comprising the identified proteins and a computer readable storage medium storing instructions that, when executed by a computer cause the computer to perform a method for the identification of proteins using mass spectra generated through the application of laser desorption ionization mass spectrometry of the proteins.



WO 2010/022400 A2

RAPID IDENTIFICATION OF PROTEINS AND THEIR CORRESPONDING SOURCE
ORGANISMS BY GAS PHASE FRAGMENTATION AND IDENTIFICATION OF
PROTEIN BIOMARKERS

FIELD OF THE INVENTION

[0001] The invention relates to the use of laser desorption/ionization mass spectrometry techniques for protein identification, and the detection, identification, and taxonomic classification of the corresponding source organisms from which the proteins are derived.

BACKGROUND OF THE INVENTION

[0002] Rapid, accurate identification of microorganisms is an increasingly important area of research. Indeed, safeguarding of the environment, public health, food production and safety, transportation and national defense all depend on the ability to rapidly and accurately identify pathogens, microbial and otherwise.

[0003] For example, foodborne illness is a serious and continuing problem that causes great suffering, death and otherwise exacts enormous societal costs *e.g.* losses in worker productivity due to illness, recall of food products determined (or suspected) to be contaminated, etc. An estimated 76 million cases of foodborne illness occur each year in the US alone (*see e.g.*, Mead, P.S. *et al.* (1999) *Emerging Infectious Diseases* 5(5): 607-625) Frigheningly, foodborne disease caused by bacterial microorganisms is likely to increase with the rise in global temperatures.

[0004] Thus, there is a critical need for methods that sensitively detect and rapidly and accurately identify foodborne pathogens before, during and after an outbreak of foodborne illness.

[0005] A number of techniques have been developed for detection and identification of microorganisms and other disease agents (*e.g.*, viruses, *e.g.*, Avian influenza virus; protein toxins, prions etc). Pre-eminent among these techniques is mass spectrometry. Mass spectrometry is the science of "weighing" atoms and molecules. Because of its high sensitivity and specificity, mass spectrometry has become a popular technique for detection and taxonomic classification of microorganisms (*see e.g.*, Fenselau C., and Demirev P.A. *Mass Spectrom. Rev.* 2001; **20**: 157; Lay Jr. JO. *Mass Spectrom. Rev.* 2001; **20**: 172).

[0006] As is well known in the art, the use of mass spectrometry (MS) in the analysis of microorganisms is a relatively recent application that was facilitated by the development of two ionization techniques in the late 1980s and early 1990s: electrospray ionization (ESI)

(see e.g., Fenn, J. B., et al. (1989). *Science*, 246, 64-71) and matrix-assisted laser desorption/ionization (MALDI) (see e.g., Tanaka, K., et al. (1987) *Second Japan-China Joint Symposium on Mass Spectrometry* (abstract), Osaka, Japan and Karas, M., et al. (1987) *International Journal of Mass Spectrometry and Ion Processes.*, 78, 53-68). MALDI, coupled with time-of-flight (TOF) mass spectrometry, is a powerful tool in "fingerprinting" microorganisms using either cell extracts or lysis of intact cells (see e.g., Cain, T.C., et al. (1994) *Rapid Commun. Mass Spectrom.* 8: 1026; Krishnamurthy, T., et al. (1996) *Rapid Commun. Mass Spectrom.* 10: 883; Krishnamurthy, T., and Ross PL. *Rapid Commun. Mass Spectrom.* 1996; 10:1992; Holland, R.D., et al.(1996) *Rapid Commun. Mass Spectrom.* 10:1227; Arnold, R., and Reilly, J. (1998) *Rapid Commun. Mass Spectrom.* 12: 630; Welham, K., et al. (1998) *Rapid Commun. Mass Spectrom.* 12: 176; Haag, A., et al. (1998) *J. Mass Spectrom.* 33: 750; Wang Z, et al. (1998) *Rapid Commun. Mass Spectrom.* 1998; 12: 456; Dai Y, et al. *Rapid Commun. Mass Spectrom.* 1999; 13: 73; Ramirez, J., and Fenselau C. (2001) *J. Mass Spectrom.* 36: 929; Mandrell, R.E., et al. (2005) *Applied Environ. Microbio.* 71: 6292; Fagerquist, C.K., et al. (2005) *Anal. Chem.* 77: 4897; and Fagerquist, C.K., et al. (2006) *J. Proteome Research.* 5: 2527). As is known in the art, MALDI-TOF-MS fingerprinting of microorganisms, viruses, proteins and peptides is typically achieved using either pattern recognition or bioinformatic algorithms.

[0007] A pattern recognition approach to data analysis typically involves comparison of a MALDI-TOF-MS spectrum from an unknown sample to MALDI-TOF-MS spectra from known, identified entities and searches for similarities in prominent peaks (see e.g., Jarmon, K.H., et al. (2000) *Anal. Chem.* 72: 1217; and Wahl, K.L., et al. (2002) *Anal. Chem.* 2002; 74: 6191). A high similarity between an unknown MS spectrum and a known MS spectrum suggests the identity of the unknown. Unfortunately however, this pattern recognition approach is limited by the fact that it does not rely on actual identification of the peaks in an MS spectrum, but rather on the spectral pattern. Thus, because a particular mass-to-charge ratio (m/z) could be associated with any of a number of biomolecules generated by the microorganism: proteins, nucleic acids, lipids, etc, accuracy of identification using the pattern recognition approach is best achieved with purified molecules.

[0008] In addition to the pattern recognition approach, microorganisms may be identified using a bioinformatic approach. The bioinformatic approach to identification of microorganisms by MALDI-TOF-MS, typically involves the use of protein molecular weight (MW) information contained in genomic databases to tentatively assign peaks in a spectrum to specific proteins (see e.g., Demirev, P.A., et al. (1999) *Anal. Chem.* 71: 2732; Peneda,

F.J., et al. (2000) *Anal. Chem.* 72: 3739; Demirev, P.A., et al. (2001) *Anal. Chem.* 73: 4566; Yao Z-P, et al. (2002) *Anal. Chem.*; 74: 2529; and Peneda, F.J., et al. (2003) *Anal. Chem.* 75: 3817). As is known in the art, a protein, and by implication the microorganism, are identified when a significant number of peaks in a MS spectrum correspond to the protein MWs derived from the open reading frames (ORFs) of a particular microorganism genome.

[0009] The bioinformatic approach may account for simple post-translational modifications, e.g. N-terminal methionine cleavage (*see e.g.*, Demirev et al. (2001) *supra*), but many proteins of interest have extensive post translational modifications. Thus, since protein biomarker identification relies solely on protein MW, identification of any individual biomarker may only at best, be considered as a tentative assignment.

[0010] Pattern recognition and bioinformatic MALDI-TOF-MS analysis work well for the identification of pure strains. Unfortunately however, data analysis using either of these methods is complicated when multiple bacterial microorganisms are present in a sample. Indeed, MALDI-TOF-MS analysis of bacterial mixtures typically results in the detection of protein biomarkers from multiple bacteria, thus making the pattern of m/z peaks presented in an MS spectrum difficult to interpret. Similarly, the bioinformatic approach may tentatively assign spectral peaks as being proteins on the basis of predicted protein MWs from the ORFs found in multiple bacterial genomes. Furthermore, the presence (and/or relative abundance) of proteins from multiple microorganisms affects on protein ionization efficiency resulting in changes in the proteins ionized and detected. Thus, analysis of samples containing multiple bacterial (or other microorganisms) presents increased challenges for MALDI-TOF-MS.

[0011] To overcome the increased challenges for MALDI-TOF-MS when multiple organisms are present, protein biomarkers are either enzymatically digested and the resulting tryptic peptides are analyzed by MS or by MS/MS ("bottom-up proteomics). Alternatively, the intact protein is fragmented in the gas phase to obtain sequence-specific information from fragment ions ("top-down" proteomics).

[0012] Until recently, gas phase fragmentation was only accomplished using very expensive mass spectrometric instrumentation and/or complicated gas phase ion dissociation techniques. However, the development of MALDI tandem mass spectrometry (MALDI-TOF-TOF) instruments has made gas phase fragmentation less cumbersome to undertake (*see e.g.*, Medzihradszky KF, et al. *Anal. Chem.* 2000: 72: 552). MALDI-TOF-TOF instruments may be used to fragment small and modest-sized intact proteins (> 5 kDa) (*see e.g.*, Lin M, et al. *Rapid Comm. Mass Spectrom.* 2003; 17: 1809). Thus, a MALDI-TOF-TOF instrument may provide a "fingerprint" of a bacterial microorganism in MS mode, and also sequence-

specific information of protein biomarkers by MS/MS (*see e.g.*, Demirev, P.A., *et al.* (2005) *Anal. Chem.* 2005; 77: 7455).

[0013] Because the amino acid sequence of a protein is often unique to a microbial strain, identification of a single protein biomarker is often sufficient for identification of the microorganism. In addition, whereas multiple microorganisms may complicate MALDI-TOF-MS analysis, MS/MS of specific protein biomarkers allows identification of unique protein biomarkers that are specific to a particular microorganism and thus facilitates the identification of a microorganism in a mixed sample.

[0014] The potential of tandem mass spectrometry techniques such as *e.g.*, MALDI-TOF-TOF, for the rapid and accurate identification of proteins and their source organisms is only beginning to be realized. Indeed, given the proliferation of biological threats from diseases, pests, and bioterrorism, the increasing numbers of proteomes in public databases, (which continues to grow at an exponential rate) and the potential forensic applications (*see e.g.*, Goldsmith, J. (1990) *A crime lab for animals - National Fish and Wildlife Forensics Laboratory, Ashland, Oregon - Special Issue: Environmental Restoration* Whole Earth Review, Spring, 1990) one can rest assured that the value and importance of tandem mass spectrometry techniques for the identification of proteins will continue to grow.

[0015] Thus, what is needed in the art, are better matching algorithms and improved scoring schemes which facilitate ease of use and which will improve the accuracy, and rapidity of tandem mass spectrometry techniques thereby facilitating their utility.

[0016] Fortunately, as will be clear from the following disclosure, the present invention provides for these and other needs.

SUMMARY OF THE INVENTION

[0017] One embodiment of the present invention provides a method for identification of an unknown protein comprising: (i) comparing mass-to-charge (m/z) fragment ion peaks from a mass spectrum of the unknown protein to that of m/z fragment ion peaks from mass spectra of known protein sequences; and (ii) matching the m/z fragment ion peaks from the mass spectrum of the unknown protein to the m/z fragment ion peaks from the mass spectra of known protein sequences to provide mass spectrometry peak matches; (iii) scoring the mass spectrometry peak matches using a USDA peak matching algorithm thereby providing a USDA peak matching score; and (iv) ranking the USDA peak matching score from highest to lowest; wherein the highest USDA peak matching score constitutes an identification of the unknown protein.

[0018] In one exemplary embodiment the USDA peak matching algorithm is USDA peak matching algorithm α , which has the formula:

$$USDA\alpha\text{ Score} = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}$$

[0019] In another exemplary embodiment the USDA peak matching algorithm is USDA peak matching algorithm β , which has the formula:

$$USDA\beta\text{ Score} = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}$$

[0020] In one exemplary embodiment, the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than 2.

[0021] In another exemplary embodiment, the mass spectra are members selected from the group consisting of MS spectra and MS/MS spectra.

[0022] In one exemplary embodiment, the mass spectrum of the unknown protein is an MS spectrum and the mass spectra of known protein sequences are MS spectra of known proteins.

[0023] In another exemplary embodiment, the mass spectrum of the unknown protein is an MS/MS spectrum and the mass spectra of known protein sequences are members of the group consisting of *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences and MS/MS spectra of known proteins.

[0024] In one exemplary embodiment, the mass spectra of known protein sequences are *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences.

[0025] In one exemplary embodiment, the method further comprises: (v) confirming that the highest USDA peak matching score constitutes an identification of the unknown protein. The confirming comprises: plotting the difference in m/z ($\Delta m/z$) between an MS/MS fragment ion of the unknown protein and its matched fragment ion from the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences as a function of MS/MS fragment ion m/z , thereby providing a plot of $\Delta m/z$ vs MS/MS fragment ion

m/z , wherein a correct protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals a characteristic systematic error related to the inherent calibration error of the mass analyzer, and wherein an incorrect protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals statistically random error.

[0026] In another exemplary embodiment, the method further comprises: (v) confirming that the highest USDA peak matching score constitutes an identification of the unknown protein. In one exemplary embodiment, the confirming comprises: performing steps (i) – (iv), wherein the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences are limited to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue or proline (P) residue, and (vi) observing that the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when steps (i) – (iv) are performed using *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences that are not limited spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue or proline (P) residue, thereby confirming that the highest USDA peak matching score constitutes an identification of the unknown protein.

[0027] In another exemplary embodiment, the confirming comprises: performing steps (i) – (iv), wherein the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences are limited to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, and (vi) observing that the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when steps (i) – (iv) are performed using *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences that are not limited spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, thereby confirming that the highest USDA peak matching score constitutes an identification of the unknown protein.

[0028] In one exemplary embodiment, the unknown protein is a protein toxin.

[0029] In another exemplary embodiment, identification of the unknown protein is used to identify the source organism from which the unknown protein derived. In one exemplary

embodiment, the source organism is a member selected from the group consisting of: viruses, microorganisms, insects, plants, and animals. In another exemplary embodiment, the source organism is a microorganism selected from the group consisting of: bacteria and fungi. In another exemplary embodiment, the source organism is a bacterium that is a member selected from the group consisting of *Campylobacter*, *Escherichia coli*, *Listeria* and *Salmonella*.

[0030] Another embodiment of the invention provides a computer readable storage medium storing instructions that, when executed by a computer, causes the computer to perform a method for identification of an unknown protein comprising: (i) comparing mass-to-charge (m/z) fragment ion peaks from a mass spectrum of the unknown protein to that of m/z fragment ion peaks from mass spectra of known protein sequences; and (ii) matching the m/z fragment ion peaks from the mass spectrum of the unknown protein to the m/z fragment ion peaks from the mass spectra of known protein sequences to provide mass spectrometry peak matches; (iii) scoring the mass spectrometry peak matches using a USDA peak matching algorithm thereby providing a USDA peak matching score; and (iv) ranking the USDA peak matching score from highest to lowest; wherein the highest USDA peak matching score constitutes an identification of the unknown protein.

[0031] In one exemplary embodiment the USDA peak matching algorithm is USDA peak matching algorithm α , which has the formula:

$$USDA\alpha\text{ Score} = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}$$

[0032] In another exemplary embodiment the USDA peak matching algorithm is USDA peak matching algorithm β , which has the formula:

$$USDA\beta\text{ Score} = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}$$

[0033] In one exemplary embodiment, the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than 2.

[0034] In another exemplary embodiment, the mass spectra are members selected from the group consisting of MS spectra and MS/MS spectra.

[0035] In one exemplary embodiment, the mass spectrum of the unknown protein is an MS spectrum and wherein the mass spectra of known protein sequences are MS spectra of known proteins.

[0036] In one exemplary embodiment, the mass spectrum of the unknown protein is an MS/MS spectrum and the mass spectra of known protein sequences are members of the group consisting of *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences and MS/MS spectra of known proteins. In another exemplary embodiment, the mass spectra of known protein sequences are *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences.

[0037] In another exemplary embodiment, the method further comprises: (v) confirming that the highest USDA peak matching score constitutes an identification of the unknown protein, wherein the confirming comprises: plotting the difference in m/z ($\Delta m/z$) between an MS/MS fragment ion of the unknown protein and its matched fragment ion from the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences as a function of MS/MS fragment ion m/z , thereby providing a plot of $\Delta m/z$ vs MS/MS fragment ion m/z , wherein a correct protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals a characteristic systematic error related to the inherent calibration error of the mass analyzer, and wherein an incorrect protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals statistically random error.

[0038] In another exemplary embodiment, the method further comprises: (v) confirming that the highest USDA peak matching score constitutes an identification of the unknown protein, wherein the confirming comprises: performing steps (i) – (iv), wherein the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences are limited to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue or proline (P) residue, and (vi) observing that the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when steps (i) – (iv) are performed using *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences that are not limited spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue or proline (P) residue, thereby that the highest USDA peak matching score constitutes an identification of the unknown protein.

[0039] In another exemplary embodiment, the confirming comprises: performing steps (i) – (iv), wherein the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences are limited to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, and (vi) observing that the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when steps (i) – (iv) are performed using *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences that are not limited spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, thereby confirming that the highest USDA peak matching score constitutes an identification of the unknown protein.

[0040] In one exemplary embodiment, the unknown protein is a protein toxin.

[0041] In another exemplary embodiment, identification of the unknown protein is used to identify the source organism from which the unknown protein derived. In another exemplary embodiment, source organism is a member selected from the group consisting of: viruses, microorganisms, insects, plants, and animals. In another exemplary embodiment, the source organism is a microorganism selected from the group consisting of: bacteria and fungi. In another exemplary embodiment, the source organism is a bacterium that is a member selected from the group consisting of *Campylobacter*, *Escherichia coli*, *Listeria* and *Salmonella*.

[0042] Other features, objects and advantages of the invention will be apparent from the detailed description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0043] FIG. 1 provides a general overview of the framework for the USDA mass spectrometry comparison software.

[0044] FIG. 2 shows the database model describing how the data is stored.

[0045] FIG. 3 describes the USDA mass spectrometry comparison software process used to compare each dataset.

[0046] FIG. 4 shows an MS spectrum (as displayed in the MALDI-TOF-TOF instrument software) of the extracted cell lysate of *Campylobacter upsaliensis* (strain RM3195) analyzed by MALDI-TOF-MS using sinapinic acid as the MALDI matrix.

[0047] FIG. 5 shows the MS/MS spectrum (as displayed in the MALDI-TOF-TOF instrument software) of the protein biomarker ion of *Campylobacter upsaliensis* (strain

RM3195) at m/z 11138.9 analyzed on a MALDI-TOF-TOF-MS using sinapinic acid as the MALDI matrix.

[0048] FIG. 6 shows an ASCII MS/MS spectrum (m/z vs. relative intensity as displayed in the USDA software window) of thioredoxin from *Campylobacter upsaliensis* (strain RM3195) previously displayed in FIG. 5. The MS/MS data in FIG. 5 was centroided (i.e. converted to ASCII format: m/z vs. absolute intensity), exported from the MALDI-TOF-TOF instrument software and imported to the MS/MS databases of the USDA software. The m/z and absolute intensity of MS/MS fragment ions are shown in the sidebar.

[0049] FIG. 7 shows the "MS/MS To *In Silico* Comparison" search parameter table as displayed in the USDA software window.

[0050] FIG. 8 shows the *in silico* MS/MS spectrum (as displayed in the USDA software window) of the protein thioredoxin of *Campylobacter upsaliensis* (strain RM3195).

[0051] FIG. 9 shows a plot of fragment ion error (as displayed in the USDA software) as a function of the MS/MS fragment ion mass-to-charge (m/z) of the protein biomarker thioredoxin (from *Campylobacter upsaliensis* strain RM3195) and its "match" to the *in silico* fragment ions of thioredoxin of *Campylobacter upsaliensis* strain RM3195 (the highest scoring identification and the correct identification). Fragment ion error ($\Delta m/z$) is the difference between m/z of an MS/MS fragment ion and the m/z of its putative *in silico* fragment ion "match". A red square represents a "match" whose *in silico* fragment ion was generated from a protein backbone cleavage site that is adjacent to a D, E or P residue. A black square represents a "match" whose *in silico* fragment ion was generated from a protein backbone cleavage site that is adjacent to *only* non-D,E,P residues.

[0052] FIG. 10 shows a plot of fragment ion error (as displayed in the USDA software) as a function of the MS/MS fragment ion mass-to-charge (m/z) of the protein biomarker thioredoxin (from *Campylobacter upsaliensis* strain RM3195) and its "match" to the *in silico* fragment ions of carbon dioxide concentrating mechanism protein of *Cyanothece* strain ATCC 51142 (the second highest scoring identification and an incorrect identification). Fragment ion error ($\Delta m/z$) is defined here as the difference between m/z of an MS/MS fragment ion and the m/z of its putative *in silico* fragment ion "match". A red square represents a "match" whose *in silico* fragment ion was generated from a protein backbone cleavage site that is adjacent to a D, E or P residue. A black square represents a "match" whose *in silico* fragment ion was generated from a protein backbone cleavage site that is adjacent to *only* non-D,E,P residues.

[0053] **FIG. 11** An MS spectrum of the extracted cell lysate of *E. coli* O157:H7 strain EDL-933 analyzed on the MALDI-TOF-TOF-MS instrument (in linear mode) using HCCA matrix. Data acquisition: 1000 laser shots.

[0054] **FIG. 12** An MS/MS spectrum of the protein biomarker ion $[M + H]^+$ at m/z 7705.6 (**FIG 11**) from *E. coli* O157:H7 strain EDL-933 analyzed on the MALDI-TOF-TOF-MS instrument using HCCA matrix. Data acquisition: 32,000 laser shots. Fragmentation by post-source dissociation (PSD). Prominent fragment ions are identified by their m/z and ion type/number and amino acid residues adjacent to the site of polypeptide backbone cleavage for the amino acid sequence of the top identification: putative uncharacterized protein yahO from *E. coli* O157:H7 strain EDL-933 (Table 4). Many of the fragment ions are the result of a polypeptide cleavage site adjacent to an aspartic acid (D) or glutamic acid (E) residues. The MS/MS data was centroided, exported as an ASCII formatted file and uploaded to the USDA software for analysis.

[0055] **FIG. 13** Primary amino acid sequence of the putative uncharacterized protein yahO of the pathogenic *E. coli* O157:H7 (EDL-933 strain)
 MKIISKMLVGALAFVNTNVYAAELMTKAEFEKVESQYEEKIGDISTSNEMSTADAKED
 LIKKADEKGADVLVLTSGQTDNKHGTADIYKKK (SEQ ID NO:1) and the UPF0379
 protein yahO of the non-pathogenic *E. coli* (K-12 strain)
 MKIISKMLVGALALAVTNTNVYAAELMTKAEFEKVESQYEEKIGDISTSNEMSTADAKED
 LIKKADEKGADVLVLTSGQTDNKHGTANIYKKK (SEQ ID NO:2). Both proteins are
 post-translationally modified with a 21 residue signal peptide (boxed). Differences in amino
 acids between the two proteins are highlighted in bold. The D ↔ N amino acid substitution
 results in a protein MW difference of 1 Da in the mature protein. Such slight differences in
 protein biomarker MW are nearly impossible to detect by MALDI-TOF-MS, however
 sequence-specific fragmentation by MS/MS allows these two proteins (and their source
 microorganisms) to be differentiated.

DETAILED DESCRIPTION OF THE INVENTION

Definitions

[0056] The expression “mass spectrum” as used herein, refers in general, to an intensity vs. m/z (mass-to-charge ratio) plot that represents the result of a chemical analysis. Hence, the mass spectrum of a sample is a pattern representing the distribution of charged molecules by mass-to-charge ratio in a sample. Thus, an “MS spectrum” as used herein, refers to a mass

spectrum produced by mass spectrometer having a single mass analyser and single detector. In contrast, an “MS/MS spectrum” as used herein, refers to a mass spectrum produced by a tandem mass spectrometer.

[0057] The expression “matching the m/z fragment ion peaks from a mass spectrum of the unknown protein to the m/z fragment ion peaks from the mass spectra of known protein sequences” as used herein refers to a procedural step used by a person having ordinary skill in the art to make an identification of an unknown protein. Typically, “matching the m/z fragment ion peaks from a mass spectrum of the unknown protein to the m/z fragment ion peaks from the mass spectra of known protein sequences” refers to comparing the m/z of fragment ions that result from ionization and fragmentation of an intact unknown protein to the theoretical/calculated m/z of *in silico* generated fragment ions of known protein sequences. The theoretical/calculated *in silico* molecular masses are based on the known atomic masses of the atoms which make up the known amino acid sequence, and as such, the calculated masses are considered to be “exact”. The measured m/z of fragment ions from the ionized and fragmented unknown protein is not “exact” but, rather, is subject to the mass accuracy of a given mass analyzer. The mass accuracy of the mass analyzer may be determined and used to compare measured m/z values to those calculated “exact” masses. If the measured m/z value is within the analyzer’s known uncertainty (mass accuracy and precision), then the comparison is considered to be a match (*i.e.* a “peak match”). If the measured m/z value differs from the calculated *in silico* m/z value by more than the instrument’s precision, then the comparison results in no match being found.

[0058] Thus, “identification of an unknown protein” occurs when the m/z fragment ion peaks from a mass spectrum of the unknown protein “match” the m/z fragment ion peaks from the mass spectra of a known protein sequence.

[0059] The expression “USDA peak matching score” as used herein, refers to the number of measured m/z values which match calculated “exact” m/z values (within the instrument’s measurement precision). In an exemplary embodiment, a “USDA peak matching score” is obtained by applying a “USDA peak matching algorithm”. In an exemplary embodiment “USDA peak matching algorithm” is the “USDA peak matching algorithm alpha (α)”, which has the following formula:

$$USDA\alpha \text{ Score} = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}$$

[0060] In another exemplary embodiment, the “USDA peak matching algorithm” is the “USDA peak matching algorithm beta (β)” which has the following formula:

$$USDA\beta \text{ Score} = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}$$

[0061] Thus, unless specifically referred to otherwise, the expression “USDA peak matching score” refers inclusively to USDA peak matching scores obtained using the “USDA peak matching algorithm alpha (α)” and/or the “USDA peak matching algorithm beta (β)”.

[0062] The expression “ranking the USDA peak matching score from highest to lowest” as used herein, refers to ordering of scores from highest to lowest where the highest reflects the greatest number of matches between measured m/z values and calculated masses from a known sequence.

[0063] The expression “difference between highest USDA peak matching score and second highest USDA peak matching score” as used herein, refers to the result obtained by subtracting the second highest USDA peak matching score from the highest USDA peak matching score.

[0064] The term “*in silico*” as used herein refers generally, to processes taking place via computer calculations. In recent years, the DNA sequences of hundreds of organisms have been decoded and stored in databases. Indeed, the amount of information stored in “*in silico* databases” continues to grow exponentially as more genomes are sequenced and more proteomes are characterized. Thus, a large body of “*in silico*” genomic information has been obtained and stored and the amount continues to grow. The information stored in an “*in silico* database” may be analyzed to determine *inter alia* genes that encode polypeptides and/or the sequence of the polypeptides encoded by the genes. Such protein sequences are referred to herein as “genomically-derived”. Thus, in an exemplary embodiment, “*in silico*” protein sequences comprise a “genomically-derived” protein sequence deduced by computer calculations based on a known DNA sequence. Alternatively, an “*in silico*” protein sequence may comprise the actual protein sequence as determined by a physical protein sequencing method *e.g.*, Edman degradation, *de novo* sequencing by mass spectrometry, etc.

[0065] The expression “*in silico* MS/MS spectrum” as used herein refers to a theoretical or hypothetical tandem mass spectrum generated by a computer based on known protein sequences. The term “*in silico*” fragment ion as used herein, takes on the meaning as typically used in the art and thus, refers to theoretical protein fragments generated by

fragmentation performed *in silico* on known protein sequences. In some exemplary embodiments an *in silico* MS/MS spectrum shows theoretical fragment ions generated at every residue. In other exemplary embodiments, an *in silico* MS/MS spectrum shows theoretical fragment ions generated at selected residues.

[0066] The expression “mass spectra of known protein sequences” as used herein, refers to theoretical mass spectra generated *in silico* by calculated fragmentation of genomically-derived protein sequences and/or to mass spectra that are physically generated from known proteins.

[0067] The expression “mass spectrum of an unknown protein” as used herein, refers to the experimentally obtained mass spectrum of a protein whose identity has not been determined with certainty. Typically, an experimentally acquired MS/MS spectrum of a protein rarely shows fragment ions at every residue because the fragmentation efficiency at every residue is not the same.

[0068] The expression “source organism” as used herein, refers to an organism that has provided or that is used to provide a sample and hence proteins for analysis by laser desorption ionization mass spectrometry (*e.g.*, Matrix-Assisted Laser Desorption/Ionization MALDI) as disclosed herein. In an exemplary embodiment a source organism is a microorganism. In another exemplary embodiment a source organism is a bacterium. In still other exemplary embodiments a source organism is a protein toxin, a plant, an animal, an insect, a virus or a human.

[0069] The expression “providing a plot of $\Delta m/z$ vs fragment ion m/z ” as used herein, refers to plotting of the difference between the measured m/z and the calculated exact m/z of an analyte (*e.g.*, a protein) as a function of analyte ion m/z . Instrument calibration involves comparing the observed masses of a mixture of known standards (*e.g.*, protein standards) to their calculated exact masses. Any discrepancy between a standard's measured m/z and its known m/z is considered error. The m/z error ($\Delta m/z$, i.e. the difference between the measured m/z and the actual m/z) is plotted as a function of analyte ion m/z .

[0070] The expression “characteristic systematic error related to the inherent calibration of the instrument” as used herein, refers to systematic error as revealed by a linear regression analysis fit of the relevant data (*e.g.*, $\Delta m/z$ vs fragment ion m/z). Similarly, the expression “statistically random error” as used herein, refers to statistically random error as defined by a linear regression analysis fit of data. Regression analysis is well known in the art (*see e.g.*, Draper, N.R. and Smith, H. *Applied Regression Analysis*, Wiley Series in Probability and Statistics (1998); Richard A. Berk, *Regression Analysis: A Constructive Critique*, Sage

Publications (2004); David A. Freedman, Statistical Models: Theory and Practice, Cambridge University Press (2005))

[0071] The terms "polypeptide," and "protein" are used interchangeably herein to refer to a functional polymer of amino acid residues. Typically a functional protein folds cooperatively and comprises multiple secondary structural elements *e.g.*, α -helices or β -pleated sheets, that interact in the final structure. The smallest known protein is TRP-Cage, a functional protein of only 20 amino acids (*see e.g.*, Jonathan W. Neidigh et al. (2002) *Nature Structural Biology* 9:425 – 430). However, as used herein, the term "protein" typically refers to a polymer of amino acid residues that is above about 5 kDa molecular weight. A polymer of amino acid residues that is below about 5 kDa molecular weight is typically referred to as a "peptide". Thus, as used herein, a "peptide" is not a "protein", and a "peptide" is not a "polypeptide".

[0072] The term "amino acid" as used herein, refers to naturally occurring and synthetic amino acids, as well as amino acid analogs and amino acid mimetics that function in a manner similar to the naturally occurring amino acids. Naturally occurring amino acids are those encoded by the genetic code, as well as those amino acids that are later modified, *e.g.*, hydroxyproline, γ -carboxyglutamate, and O-phosphoserine. Amino acid analogs refers to compounds that have the same basic chemical structure as a naturally occurring amino acid, *i.e.*, an α carbon that is bound to a hydrogen, a carboxyl group, an amino group, and an R group, *e.g.*, homoserine, norleucine, methionine sulfoxide, methionine methyl sulfonium. Such analogs have modified R groups (*e.g.*, norleucine) or modified peptide backbones, but retain the same basic chemical structure as a naturally occurring amino acid. Amino acid mimetics refers to chemical compounds that have a structure that is different from the general chemical structure of an amino acid, but that functions in a manner similar to a naturally occurring amino acid. Amino acids are referred to herein by either their commonly known three letter symbols or by the one-letter symbols recommended by the IUPAC-IUB Biochemical Nomenclature Commission.

[0073] The term "microorganism" is used herein, refers to is an organism that is microscopic. In its broadest sense the term "microorganism" refers to a diverse array of organisms including, but not limited to bacteria, fungi, archaea, and protists, as well as some microscopic plants and animals *e.g.*, plankton.

[0074] The term "virus" as used herein, refers to sub-microscopic infectious agents that are typically unable to grow or reproduce outside a host cell. Typically a viral particle, or

virion, comprises genetic material *e.g.*, DNA or RNA, within a protective protein coat called a capsid. Viruses infect all cellular life forms and are grouped into animal, plant and bacterial types, according to the type of host infected.

[0075] The term "protein toxin" as used herein, refers to refers to any polypeptide that is deleterious to humans or other mammals when it is injected, ingested, inhaled or when contacted with the skin.

I. Introduction:

[0076] Rapid, accurate identification of proteins and the source organisms from which they are derived is increasingly valuable as threats to biosecurity of the environment, public health, food production and safety, transportation and national defense all continue to increase. Rapid, accurate identification of proteins also facilitates the work of forensic scientists concerned with the protection of endangered species and/or with solving of personal crime.

[0077] Since the identity of proteins connects directly to the genetic identity of the source organism (*e.g.*, a pathogenic organism; protein toxin *e.g.*, prion; blood sample; etc).

Identification of a protein may be used to successfully identify the source organism (*e.g.*, a bacterium, an endangered species, etc) from which it is derived. Fortunately, variations in protein biomarker amino acid composition due to non-synonymous mutations of the biomarker gene result in variations in protein molecular weight (MW) across genus, species and strains. The ability to tap into this information on a precise level is thus invaluable for the accurate identification of source organisms.

[0078] Improvements in the science of protein identification are always needed and welcome. Thus, an exemplary embodiment of the invention provides a method for rapid and accurate identification of proteins using laser desorption ionization mass spectrometry methods *e.g.*, MALDI. In one exemplary embodiment, identification of the protein permits identification of the source organism from which the identified protein is derived. In another exemplary embodiment, the method comprises a peak-matching/scoring mathematical algorithm (the USDA peak matching algorithm) for comparison of MS/MS spectra of protein biomarkers and *in silico* MS/MS spectra of protein sequences contained in a database.

[0079] In one exemplary embodiment, a USDA peak matching algorithm is the USDA α peak matching algorithm and has the formula:

$$USDA\alpha\text{ Score} = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}$$

[0080] In another exemplary embodiment, a USDA peak matching algorithm is the USDA β peak matching algorithm and has the formula:

$$USDA\beta\text{ Score} = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}$$

[0081] In one exemplary embodiment, the USDA peak matching algorithm (either or α or β) is incorporated into a computer software program such that a MS/MS spectrum can be compared to thousands of *in silico* MS/MS spectra in seconds thereby allowing rapid identification of a protein and its source organism.

[0082] In another exemplary embodiment, the method further comprises steps for confirming the correctness of the protein identification, thereby ensuring accuracy. In one exemplary embodiment, confirming the correctness of the protein identification is achieved through error analysis. In this embodiment, the error analysis comprises analyzing the difference in mass-to-charge (m/z) between the MS/MS fragment ions and their putative *in silico* fragment ion "matches". Correct identifications are characterized in that the error is typically limited to systematic error caused by the inherent calibration error of the mass analyzer. In contrast, incorrect identifications are characterized by random error caused by false "matches".

[0083] In another exemplary embodiment, confirming the correctness of the protein identification is achieved by performing an identification of an unknown protein by comparing the MS/MS spectrum of the unknown protein to a full array of MS/MS *in silico* spectra of protein sequences contained in a database, and then performing the identification by comparing the MS/MS spectrum of the unknown protein to a set of MS/MS *in silico* spectra of protein sequences contained in the same database which are limited to those spectra which arise when the fragmentation occurs only at aspartic acid (D) glutamic acid (E) and proline (P) residues; and observing that the difference between the highest and the second highest USDA peak matching scores is the same or greater than the difference between the highest and the second highest USDA peak matching scores obtained in the identification using the full array of MS/MS *in silico* spectra of protein sequences contained in the

database.

[0084] In still another exemplary embodiment, confirming the correctness of the protein identification is achieved by performing an identification of an unknown protein by comparing the MS/MS spectrum of the unknown protein to a full array of MS/MS *in silico* spectra of protein sequences contained in a database, and then performing the identification by comparing the MS/MS spectrum of the unknown protein to a set of MS/MS *in silico* spectra of protein sequences contained in the same database which are limited to those spectra which arise when the fragmentation occurs only at aspartic acid (D) residues; and observing that the difference between the highest and the second highest USDA peak matching scores is the same or greater than the difference between the highest and the second highest USDA peak matching scores obtained in the identification using the full array of MS/MS *in silico* spectra of protein sequences contained in the database.

[0085] Enzymatic digestion of a protein is not required for analysis and identification of proteins using the disclosed methods. Thus, in one exemplary embodiment, proteins are analyzed without prior enzymatic digestion. However, in another exemplary embodiment, proteins are analyzed following enzymatic digestion.

[0086] In one embodiment of the invention, the methods and algorithms disclosed herein are contained in a computer readable storage medium thereby providing instructions that when executed by a computer, causes the computer to perform a method for identification of an unknown protein.

II. Identification of Proteins Using Laser Desorption Ionization Mass Spectrometry.

A. General Laser Desorption Ionization Mass Spectrometry Methods

[0087] Mass spectrometry is a sensitive and specific analytical technique that is increasingly being developed for detection, identification and characterization of proteins. Thus, mass spectrometry is also becoming a powerful tool for the identification of pathogenic bacterial microorganisms, viruses, and protein toxins and also finds use in proteomics and forensic science.

[0088] The methods disclosed herein utilize routine techniques in the field of mass spectrometry. Basic texts disclosing general mass spectrometry methods include *e.g.*, *Mass Spectrometry: A Textbook*, by Jürgen H. Gross, (2002, 2004, 2006) Springer, and *Mass Spectrometry: Principles and Applications* by Edmond de Hoffmann, and Vincent Stroobant,

(2002) John Wiley and Sons Ltd, (each of which is incorporated herein by reference). Laser Desorption/Ionization is disclosed *e.g.*, in *Laser Desorption Ionization Mass Spectrometry of Bioorganic Molecules* by: Michael Karas, and Ute Bahr (1993) *Methods in Molecular Biology* Vol.17 pages 215-228 Springer.

[0089] Laser desorption/ionization mass spectrometry methods which produce singly, or occasionally doubly charge ions are suitable for use in the protein identification methods disclosed herein. In addition to MALDI, any method that is capable of mass isolating and fragmenting singly charged protein ions in the gas phase is suitable for use in the protein identification methods disclosed herein.

[0090] Laser desorption/ionization methods known in the art which have the ability to produce singly charged and sometimes doubly charged ions include, but are not limited to SELDI (*see e.g.* Surface-enhanced laser desorption/ionization (SELDI), Ning Tang, *et al.* (2004) *Mass Spectrometry Reviews*, 23:34–44); Infrared laser desorption/ionization on silicon (*see e.g.*, Bhattacharya SH *et al.* (2002) *Anal Chem.* May 1;74(9):2228-31); Laser desorption/ionization mass spectrometry on porous silicon (*see e.g.*, Seetharaman Vaidyanathan, *et al.* (2007) *Rapid Communications in Mass Spectrometry* 21(13): 2157-2166); MALDI (*see e.g.*, Tanaka, K., *et al.* (1987) *supra* and Karas, M., *et al.* (1987) *supra*) etc.

[0091] The methods disclosed herein for identification of an unknown protein are not suitable for use with ionization methods that produce multiply charged ions. Such methods include, but are not limited to *e.g.*, electrospray ionization (ESI) (*see e.g.*, Fenn, J. B., *et al.* (1989) *Science*, 246, 64-71).

B. Matrix-Assisted Laser Desorption/Ionization (MALDI)

[0092] In an exemplary embodiment, the laser desorption/ionization mass spectrometry method is matrix-assisted laser desorption/ionization (MALDI). MALDI is a soft ionization technique, that is useful for, *inter alia* the analysis of biomolecules (*see e.g.*, Cain, T.C., *et al.* (1994) *supra*; Krishnamurthy, T., *et al.* (1996) *supra*; Krishnamurthy, T., and Ross PL. (1996) **10**:1992 *supra*; Karas, M. & Hillenkamp, F. (1988) *Analytical Chemistry*, 60, 2299-2301; Karas, M, *et al.* (1987). *Int. J. Mass Spectrom. Ion Proc.* 78:53–68; Lancaster, K. S., *et al.* (2006) *Anal Chem.*, 78, 4990-4997) and large organic molecules generally (*see e.g.*, Karas, M.; Bahr, U. (1990). "Laser Desorption Ionization Mass Spectrometry of Large Biomolecules". *Trends Anal. Chem.* 9: 321–5).

[0093] MALDI involves embedding analyte molecules in a matrix of a weak organic acid

that are solids at room temperature (*e.g.* α -cyano-4-hydroxycinnamic acid). MALDI matrices typically absorb strongly in the ultraviolet electromagnetic spectrum (*e.g.* 337 nm or 355 nm, etc.). Typically, a pulsed nitrogen (at 337 nm) or pulsed solid state YAG laser (at 355 nm) is used to irradiate a dried spot of organic acid matrix that is embedded with analyte. These lasers have repetition rates from 2-100 Hz with a pulse width of 10 nanoseconds. The strong absorption of the matrix at the laser wavelength results in an explosive desorption from the target plate of matrix and analyte molecules into the gas phase.

[0094] In a process that is still not well understood, analyte molecules are protonated (or deprotonated). The MALDI ionization process typically results in singly charged protonated analytes (in positive ion mode). Higher charge states are possible but these typically have less signal intensity. In the case of peptides and proteins, protonation typically occurs at the side-chains of basic residues (*e.g.*, arginine, lysine, histidine).

[0095] Because relatively low charge states are produced by MALDI the mass-to-charge (m/z) ratio of proteins is typically beyond the mass range of most mass analyzers. Thus, the type of a mass spectrometer most widely used with MALDI is the TOF (time-of-flight mass spectrometer) since this mass analyzer has a theoretically unlimited mass range.

[0096] MALDI has been used for the identification of proteins isolated through gel electrophoresis or other purification methods (*see e.g.*, Demirev, P.A., *et al. Anal. Chem.* 1999; 71: 2732; Peneda, F.J., *et al. Anal. Chem.* 2000; 72: 3739; Demirev, P.A., *et al. Anal. Chem.* 2001; 73: 4566; Yao Z-P, *et al. Anal. Chem.* 2002; 74: 2529; Peneda, F.J., *et al. Anal. Chem.* 2003; 75: 3817). MALDI is also used for peptide mass fingerprinting by methods known in the art (*see e.g.*, Pappin DJ, *et al.* (1993). *Curr. Biol.* 3 (6): 327–32; Henzel WJ, *et al.* (1993). *Proc. Natl. Acad. Sci. U.S.A.* 90 (11): 5011–5; Mann M, *et al.* (1993). *Biol. Mass Spectrom.* 22 (6): 338–45; James P, *et al.* (1993). *Biochem. Biophys. Res. Commun.* 195 (1): 58–64; Yates JR, *et al.* (1993) *Anal. Biochem.* 214 (2): 397–408).

[0097] Tandem MALDI *e.g.*, MALDI-TOF-TOF instruments have been used to fragment small to medium-sized proteins (*see e.g.*, Lin M, *et al.* (2003) *supra*). MALDI has also been used for identification of *Bacillus* spores by fragmenting their protein biomarkers using a tandem MALDI mass spectrometer and analyzing the sequence-specific fragment ions generated by comparison to *in silico* fragment ions derived from protein amino acid sequences from open reading frames of genomic databases (*see e.g.*, Demerev *et al.* (2005) *Anal Chem* 77:7455). Identifications were tested for significance by a p-value calculation, which provides a probability that an identification occurred randomly (*see e.g.*, Demerev *et*

al. (2005) *supra*).

[0098] At one time, mass spectrometry was only practiced by relatively small number of highly trained specialists. However, a revolution in mass spectrometry was triggered by the development of two ionization techniques developed in the late 1980s and early 1990s: matrix-assisted laser desorption/ionization or MALDI (Tanaka, Ido, Akita, Yoshida & Yoshida, 1987), (Karas, Bachmann, Bahr & Hillenkamp, 1987), (Karas & Hillenkamp, 1988), and electrospray ionization or ESI (Fenn, Mann, Meng, Wong, & Whitehouse, 1989). The impact that these two inventions dramatically extended the development of mass spectrometry and its application to the analysis of large biomolecules including peptides, proteins, nucleic acids and complex carbohydrates.

[0099] With the rapid development of genomic sequencing (i.e. "next-generation"), the speed of genomic sequencing, and the ability to acquire longer DNA sequence "reads", will allow almost any strain of microorganism to be completely sequenced in a relatively short period of time. This capability means that the number of strains of a genus, species, sub-species, sub-type of microorganism that will be genomically sequenced will increase significantly beyond what is currently possible. With this ever increasing amount of genomic data being made available in public and private databases, the number of unique protein amino acid sequences (open reading frames or ORFs of genomic data) is expected to increase together with the genomic sequencing efforts. In consequence, it is critical that, as the volume of genomic and proteomic data increases, the mathematical algorithms used to analyze and exploit such data be as computationally efficient as possible.

[0100] Fortunately, the USDA peak matching algorithms disclosed herein are demonstrably faster in computation speed than other method currently used in the art *e.g.*, p-value calculation (*see e.g.*, Demirev *et al.* (2005) *Anal. Chem.* 77:7455). Fortunately, in using the faster USDA peak matching algorithms, accuracy is not sacrificed for speed. Indeed, the USDA peak matching algorithms are faster and give comparable results in scoring/ranking of protein/organism identifications as that obtained by other methods more complicated scoring algorithms known in the art *e.g.*, p-value calculation (*see e.g.*, Tables 1A, 2A, 3A).

[0101] The difference in algorithm computation speeds is further enhanced (in favor of the USDA peak matching algorithms) when only residue-specific *in silico* fragment ions (*e.g.* D,E,P-specific fragment ions) are used for comparison to MS/MS fragment ions (*see e.g.*, Tables 1B, 2B, 3B). Under these circumstances, the USDA algorithm computation speed is cut by at least about half whereas the p-value computation speed is reduced by only about

~20% at best. The increased speed and accuracy of protein identification provided by residue-specific analysis has not heretofore been appreciated by others (*see e.g.*, Demirev *et al.* (2005) *Anal. Chem.* 77:7455). Indeed, in the past it has typically been assumed, that fragment ions could be generated, theoretically, at *every* residue of a protein.

[0102] In some exemplary embodiments, the USDA mass spectrometry comparison software uses plots and "fits" fragment ion m/z error as a function of fragment ion m/z to confirm the correctness of the USDA peak matching algorithm identification. Such a confirmatory analysis has heretofore not been utilized or demonstrated. Thus, the methods disclosed herein are simpler, faster, more accurate and more complete than other currently available methods.

1. Sample Preparation for MALDI

[0103] Samples may be prepared for MALDI ionization by any method known in the art (*see e.g.*, Onnerfjord, P., Ekström, S., Bergquist, J., Nilsson, J., Laurell, T., Marko-Varga, G. (1999) *Rapid Commun Mass Spectrom.*, 13, 315-322).

[0104] Typically, a purified sample of the analyte compound (*e.g.* peptide, protein, small molecule) is dissolved in an aqueous solution at a concentration of picomole (10^{-12} mole) per μL (10^{-6} L) or lower.

[0105] The MALDI matrix of choice (see below) is dissolved in a separate aqueous solution (*e.g.* 2:1 water/acetonitrile) to make a saturated (or sub-saturated) solution of matrix. Typically, this solution is acidified with an acid such as *e.g.*, 0.1% of trifluoroacetic acid (TFA) however, any suitable acidifier can be used in an effective concentration. As is known in the art, acidification facilitates protonation of analyte molecules during the desorption/ionization process.

[0106] In one exemplary embodiment, roughly equal aliquots of matrix solution and analyte solution are mixed together and then spotted onto a stainless steel MALDI target. The spot is allowed to dry and the target is inserted into the ionization source of the mass spectrometer.

[0107] In another exemplary embodiment, a one μL aliquot of the matrix is spotted onto the stainless steel MALDI target and allowed to dry before a one μL aliquot of the sample is overlayed on top of the spot of dried matrix. The re-dissolved spot is allowed to dry before inserting the target into the mass spectrometer.

[0108] In still another exemplary embodiment, the sample is spotted the on the stainless

steel target, and allowed to dry and then spotting the matrix on top of the dried spot of sample (see e.g., Onnerfjord, Ekström, Bergquist, Nilsson, Laurell, Marko-Varga, 1999).

[0109] MALDI works best when contaminants (e.g. salts, small molecules, etc.) are removed from the sample before MALDI preparation. Salts, in particular, interfere with ionization by MALDI. Sample clean-up using high performance liquid chromatography (HPLC) or dialysis helps to reduce salts and small molecule contaminants that may interfere with MALDI ionization of the analyte of interest.

2. Protein Extraction

[0110] Protein extraction may be accomplished using any suitable technique known in the art. Typically the technique chosen depends on the nature of the protein one wishes to extract. The skilled artisan will appreciate those factors that influence extraction of their protein of interest and will choose their extraction techniques accordingly. For example, proteins may be hydrophilic or hydrophobic or have regions that display both properties. Cytosolic proteins (because they reside in the aqueous cytosol of the cell) are often the easiest proteins to extract because they are hydrophilic and thus dissolve easily in aqueous solutions. Cytosolic proteins are also often high copy proteins so their relative abundance in the cellular proteome is high. In consequence, protein biomarkers of bacterial cells are often high copy cytosolic proteins. Extraction of cytosolic proteins typically involves nothing more difficult than rupturing the cellular membrane (or cell wall) to release the cytosolic proteins into the surrounding protein extraction medium (e.g. water/acetonitrile).

[0111] Cell membrane or cell wall disruption is accomplished by any technique known in the art, e.g. bead-beating (see e.g., Mandrell, R. E., Harden, L. A., Bates, A., Miller, W. G., Haddon, W. F., & Fagerquist, C. K. (2005). *Appl Environ Microbiol.*, 71, 6292-6307; high pressure disruption (see e.g., Smejkal, G.B., Robinson, M.H., Lawrence, N.P., Tao, F., Saravis, C.A., Schumacher, R.T. (2006) *J. Biomol. Tech.*, 17, 173-175; or sonication (see e.g., Balasundaram, B., Pandit, A.B. (2001) *Biotechnol Bioeng.*;75, 607-614).

3. MALDI Matrices

[0112] In general, the identity of suitable matrix compounds is determined to some extent by trial and error. However, some considerations are helpful in guiding the skilled practitioner (see e.g., K. Strupat, et al. (1991) *International Journal of Mass Spectrometry and Ion Processes* Vol. 111, 1991, Pages 89-102; Beavis RC, and Chait BT (1989). *Rapid Commun. Mass Spectrom.* 3 (12): 432-5; Karas, M. et al. (1987). *Int J Mass Spectrom Ion Proc* 78:

53–68.).

[0113] In general, exemplary MALDI matrices are low molecular weight (*e.g.*, 100–200 Da) weak organic acids which are solid at room temperature but absorb strongly at ultraviolet light at wavelengths, *e.g.* 337 nm (nitrogen laser) or 355 nm (3rd-harmonic of a solid state YAG laser).

[0114] Exemplary MALDI matrices are also typically are functionalized with polar groups, allowing their use in aqueous solutions. The most popular MALDI matrices for ionizing peptides and proteins are α -cyano-4-hydroxycinnamic acid (abbreviated CHCA or CCA), 2,5-dihydroxybenzoic acid (abbreviated DHB), trans-4-hydroxy-3-methoxy-cinnamic acid (common name: ferulic acid) and 3,5 dimethoxy-4-hydroxy-cinnamic acid (common name: sinapinic acid). CHCA is considered a "hot" matrix because during laser vaporization/desorption this matrix appears deposit significant amounts of energy into the peptide or protein which can result in dissociation of the peptide or protein ion after it exits the ionization chamber. CHCA also generates doubly and triply protonated (charged) analytes in addition to singly protonated (charged) analytes, thus, it should be used with care in order to be suitable for use with the method disclosed herein for identification of an unknown protein. At the other end of the MALDI matrix spectrum is sinapinic acid which is considered to be a "cold" matrix. Use of sinapinic acid as a MALDI matrix tends to generate only singly protonated ions as well as high molecular weight proteins. In consequence, sinapinic acid is particularly suited to ionization of higher molecular weight proteins, *e.g.*, above about 10 kDa.

C. Mass Spectrometry

[0115] Mass spectrometry is the science of "weighing" (or measuring the mass) of atoms and molecules. As is known in the art, the applications of mass spectrometry are numerous, ranging from measuring very precisely the abundances of atomic isotopes of elements found in the periodic table, to measuring with high resolution and mass accuracy the mass of large biomolecules, *e.g.* proteins (*see e.g.*, Grayson, M.A. (2002) *Measuring Mass*, Chemical Heritage Foundation).

I. Tandem Mass Spectrometry

[0116] In an exemplary embodiment, the methods disclosed herein utilize tandem MALDI mass spectrometry. As is well known in the art, tandem mass spectrometry (or MS/MS) involves coupling two (or more) mass analyzers in tandem in order to selectively interrogate individual analyte ions observed in MS analysis.

[0117] A popular tandem mass spectrometer is the triple quadrupole (*see e.g.*, Yost, R.A., Enke, C.G. (1978) *J. Amer. Chem. Soc.*, 100:2274-2275). When used in MS/MS mode, the first quadrupole of a triple quadrupole instrument is operated so as to allow only ions of a particular m/z to transit into the second quadrupole mass analyzer. This is called mass isolation or mass selection. The mass-selected ions are fragmented in a second quadrupole typically by collision-activated dissociation (CAD). The fragment ions generated are then analyzed and detected by the third quadrupole mass analyzer. Because of mass-selection/isolation, the fragment ions generated in the second quadrupole are linked to the analyte precursor ions selected by the first quadrupole.

[0118] The principles of MS/MS have been extended to a number of instrument platforms *e.g.*, time of flight (TOF). TOF/TOF mass spectrometry is known in the art (*see e.g.*, U.S. Pat. No. 6,300,627).

[0119] In an exemplary embodiment, a method of identifying an unknown protein employs MALDI TOF/TOF mass spectrometry. Briefly, MS/MS using a MALDI-TOF-TOF instruments involves using the first TOF mass analyzer to separate ions in space and time. Because ions are separated on TOF instruments on the basis of their relative velocities and this is related to an ion's m/z , a timed ion "gate" is used to mass-select/isolate ions of a specific m/z . Only ions of a specific m/z (with a specific velocity and thus arrival time at the "gate") are allowed to pass from the first TOF mass analyzer to a collision cell. All other ions arriving either "too early" or "too late" are deflected from entering the collision cell. Thus, mass-selection/isolation is attained.

[0120] The mass selected/isolated ions are dissociated in the collision cell by high energy collision-activated dissociation (CAD) or by post-source dissociation (PSD) or by laser-induced dissociation (LID). On TOF-TOF instruments, collision energies of 1-2 keV are possible. PSD involves the dissociation of ions as a result of the energy deposited into the analyte ion at the ionization source. After precursor ion fragmentation, fragment ions are analyzed by the second TOF mass analyzer (*see e.g.*, Medzihradszky, K.F., *et al.* (2000). *Anal. Chem.*, 72, 552-558).

a. Error Analysis

[0121] In an exemplary embodiment, the method comprises confirming that the highest USDA peak matching score constitutes an identification of the unknown protein. In this embodiment the correctness of the identification is confirmed by plotting the difference in m/z ($\Delta m/z$) between an MS/MS fragment ion of the unknown protein and its matched

fragment ion from the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences as a function of MS/MS fragment ion m/z . The resulting plot of $\Delta m/z$ vs MS/MS fragment ion m/z is subjected to a linear regression analysis fit of the data. Linear regression analysis is known in the art (*see e.g.*, Skoog, D.A. & West, D.M. (1986). *Analytical Chemistry*, 4th edition.)

[0122] Briefly, linear regression analysis is the process by which a "best" fit straight line is fitted to data plotted in two dimensions (*e.g.* x-dimension and y-dimension). In an exemplary embodiment, such an analysis is used in creating a calibration curve for a mass spectrometer. The simplest linear regression is the method of least squares. The method of least squares is a regression analysis that minimizes the square of the vertical displacement (residuals) of the data points from the line that is being fitted to the data. This process generates the "best" fit of the line to the data points (Skoog & West, 1986, *supra*).

[0123] A correct protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals a characteristic systematic error related to the inherent calibration error of the mass analyzer.

[0124] An incorrect protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals statistically random error.

[0125] In an exemplary embodiment, prior to running a sample, the mass spectrometer is tuned to maximize sensitivity, resolution and mass accuracy. Typically, this involves either manually or automatically adjusting the instrument's ion optics using a known standard. This process ensures that the ionization source, mass analyzer and detector are all functioning properly. As part of this process, the mass analyzer is also calibrated.

[0126] Calibrating the mass analyzer ensures that mass measurement is within the operating specifications of the instrument. As the skilled artisan will appreciate, instrument calibration in MS mode involves comparing the observed masses of a mixture of known standards to their actual masses. Any discrepancy between a standard's measured m/z and its known m/z is considered error. The m/z error or $\Delta m/z$ (i.e. difference between the measured m/z and the actual m/z) is plotted as a function of analyte ion m/z . A linear regression analysis (*e.g.*, higher order curve fitting) is used to generate a calibration curve that is then applied to the mass assignment of any subsequent data collected to correct for inaccuracies in the mass analyzer's performance.

[0127] As the skilled artisan also appreciates, it is not unusual, even after calibration, for a mass analyzer to experience "drift", i.e. the instrument calibration is no longer correcting

for a mass analyzer's slight inaccuracies in mass measurement. In an exemplary embodiment, mass analyzer "drift" is due to changes in the temperature of the room in which the instrument is housed, however other causes could be in play. TOF, TOF-TOF and hybrid quadrupole-TOF or ion trap-TOF instruments, are particularly susceptible to calibration "drift" due to changes in room temperature. In consequence, the instrument may require re-calibration daily (or more often) if there is a significant change in room temperature.

[0128] In an exemplary embodiment, the "drift" in calibration of TOF-based mass spectrometry instrumentation is exploited with respect to peptide and protein identification that involves comparison of an MS/MS spectrum to *in silico* MS/MS spectra.

[0129] It has been shown that a tryptic peptide (with a known amino acid sequence) analyzed using a hybrid quadrupole-TOF mass spectrometer in MS/MS mode generates an MS/MS spectrum that, when "matched" to its correctly identified *in silico* MS/MS peptide spectrum, results in a linear correlation (using linear regression analysis) between the fragment ion m/z error as a function of fragment ion m/z , and that this linear correlation was not observed when the MS/MS spectrum was "matched" to an incorrect *in silico* MS/MS peptide spectrum.

[0130] However, since it was concluded experimentally that the "drift" in the instrument calibration of the TOF mass analyzer of the quadrupole-TOF instrument was caused by changes in the temperature of the instrumentation room (*see e.g.*, Taylor, J.A. & Johnson, R.S (2001) *Analytical Chemistry*, 73, 2594-2604), this linear correlation for the correct *in silico* peptide identification was simply incorporated into *de novo* peptide sequencing software (as an internal standard, i.e. trypsin autolysis peptides) to correct for the calibration "drift" of the TOF mass analyzer on the quadrupole-TOF instrument due to room temperature changes.

[0131] Thus, although a linear "fit" of the fragment ion m/z error has been used to correct for calibration error of a TOF mass analyzer, it has not been appreciated before the instant disclosure that a linear "fit" of the fragment ion m/z error could be used to "identify" or "confirm the identity" of a protein and/or peptide sequence to that of an *in silico* sequence in a database.

[0132] Thus an exemplary embodiment of the invention provides a comparison of fragment ions generated from MS/MS of mature, intact proteins from cell lysates analyzed using a tandem TOF instrument to *in silico* fragment ions from protein amino acid sequence in a database. The difference in m/z between the MS/MS fragment ions and their putative *in silico* fragment ion "matches" are plotted as a function of the MS/MS fragment ion m/z . A linear or least-squares fit of such data is used to confirm the correctness of the USDA peak

matching algorithm identification. In an exemplary embodiment, a correct identification displays evidence of a linear correlation between m/z error vs. fragment ion m/z whereas an incorrect identification will display poor correlation (i.e. random error) between m/z error vs. fragment ion m/z .

b. *Residue Specific Analysis*

[0133] In an exemplary embodiment, the method comprises confirming that the highest USDA peak matching score constitutes an identification of the unknown protein. In this embodiment the correctness of the identification is confirmed by performing the method while limiting the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue, or proline (P) residue. If the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when the method is performed without the limitation to D, E, and/or P residues this confirms the identification of the unknown protein.

[0134] As discussed more extensively below, in an exemplary embodiment, the method for identifying an unknown protein is contained within computer readable storage medium storing instructions that, when executed by a computer, causes the computer to perform a method for identification of an unknown protein.

[0135] In one exemplary embodiment, the software allows an "MS/MS to *In Silico* Comparison" to compare MS/MS fragment ions to *all* possible *in silico* MS/MS fragment ions or to only residue-specific *in silico* MS/MS fragment ions.

[0136] In another exemplary embodiment, the software allows an "MS/MS to *In Silico* Comparison" to compare MS/MS fragment ions to a subset of all possible *in silico* MS/MS comprising certain selected residue-specific *in silico* MS/MS fragment ions. In one exemplary embodiment, the comparison is executed by the software that compares the subset of *in silico* MS/MS residue-specific *in silico* MS/MS fragment ions comprising D,E,P-specific *in silico* MS/MS fragment ions to the MS/MS fragment ions. In another exemplary embodiment, the comparison is executed by the software that compares the subset of *in silico* MS/MS residue-specific *in silico* MS/MS fragment ions comprising D-specific *in silico* MS/MS fragment ions to the MS/MS fragment ions.

[0137] Although it has been shown experimentally that singly protonated proteins are

more likely to fragment at the polypeptide backbone at sites that are adjacent to an aspartic acid (D), glutamic acid (E) or proline (P) (*see e.g.*, Lin, M., et al. (2003) *Rapid Commun. Mass Spectrom.* 17:1809-1814.), until the discovery by the present inventors, it was not heretofore appreciated that this experimental observation could be exploited into a method or software algorithm to specifically identify a protein (and its source organism) on the basis of a comparison of MS/MS fragment ions to residue-specific *in silico* MS/MS fragment ions.

[0138] Incorporating residue specific analysis into the methods for identifying an unknown protein, as disclosed herein, significantly increases the ability to correctly and accurately identify an unknown protein as reflected in a USDA peak matching score (*i.e.*, USDA α Score and/or USDA β Score). Indeed, the absolute value of the USDA peak matching score is increased as well as the relative value of the highest USDA peak matching score relative to the second highest USDA peak matching score(s) and the USDA peak matching scores of other incorrect identifications.

[0139] Without being bound by theory, it is believed that the reason for this increase is that, if it is known experimentally that a particular protein (or proteins) will preferentially fragment at particular sites along the polypeptide backbone due to specific residues adjacent to such cleavage sites, then the software can be directed to include only those residue-specific *in silico* MS/MS fragment ions for comparison to the experimentally measured MS/MS fragment ions, *i.e.* they are fewer *in silico* MS/MS fragment ions that are involved in the comparison to MS/MS fragment ions, thus increasing the numerical value of the "USDA Score". Such selectivity not only increases the identification score of the correct identification, but also reduces the computation time of the analysis because fewer *in silico* MS/MS fragment ions are being compared.

D. Databases

I. *In Silico* Databases

Constructing an *In Silico* MS/MS Database

[0140] In an exemplary embodiment, an *in silico* MS/MS database is constructed to retrieve the amino acid sequences of all bacterial proteins having a protein molecular weight the same as the protein biomarker molecular weight within a specified mass tolerance (*e.g.* 10,000 $\pm$ 5 Da). Bacterial protein amino acid sequences can be downloaded from public web-based database servers (*e.g.* ExPASy proteomics server, available through Swiss Institute of Bioinformatics). In an exemplary embodiment, the *TagIdent* software tool at the *Expasy* website allows the retrieval of proteins on the basis of taxonomic identification of the

source organism as well as protein *pI*, protein molecular weight and any sequence tag information (*e.g.*, a short sequence of adjacent amino acid residues in the protein) if it available. *TagIdent* searches the public databases of UniProtKB/Swiss-Prot and/or UniProtKB/TrEMBL for all proteins which correspond to the specified search parameters.

[0141] In an exemplary embodiment, all of the proteins that fit the specified search criteria are compiled into a single large FASTA file. This multi-protein sequence FASTA file is then imported into a mass spectrometry-based proteomics software (*e.g.* the a commercially available GPMW, ©Lighthouse Data). Mass spectrometry-based proteomics software *e.g.*, GPMW, converts each protein amino acid sequence contained in the FASTA file into a series of *in silico* MS/MS fragment ions. The fragment ions are identified by their *m/z*, ion type/number (*e.g.* *b*₅₂, *y*₄₄, *a*₂₇, *y*₃₂₋₁₇, *b*₉₂₋₁₈, etc.) and the two amino acid residues immediately adjacent to the polypeptide cleavage site that generated the fragment ion.

[0142] Each protein sequence (and its associated *in silico* MS/MS fragment ion information) is exported by the mass spectrometry-based proteomics software *e.g.*, GPMW, software as a separate datafile. Thus, a single multi-protein sequence FASTA file containing 700 protein sequences will result in 700 *in silico* MS/MS datafiles when processed by GPMW. These *in silico* MS/MS datafiles are then imported into the *in silico* MS/MS database of the USDA mass spectrometry comparison software using Java client ("InSilicoUpload.jar", *i.e.* a binary file [jar file] that is interpreted by the Java Runtime Environment or JRE) which was created to import/upload automatically multiple *in silico* MS/MS data files to the MySQL database of the USDA mass spectrometry comparison software. This software program is operated independently of the Tomcat Apache (TA)/Java Server Pages (JSP) of the USDA mass spectrometry comparison software. Once the *in silico* files are imported into the *In silico* MS/MS Database of the USDA mass spectrometry comparison software, the data can be displayed or compared against MS/MS acquired experimentally.

[0143] Display of an *in silico* MS/MS spectrum looks very much like a "barcode" because each *in silico* MS/MS fragment ion has a 100% relative intensity as shown in FIG. 8. In contrast, an experimentally acquired MS/MS spectrum will have fragment ions whose intensity can vary from 0 to 100% as that shown in FIG. 6.

2. Mass Spectra of Known Proteins

[0144] In an exemplary embodiment, the amino acid sequence of a protein is determined by genetic or genomic sequencing of the gene whose product is protein. In another

exemplary embodiment, the genomically-derived amino acid sequence of protein is determined and/or confirmed by proteomic techniques such as Edman sequencing or "bottom-up" or "top-down" mass spectrometry-based proteomic techniques.

[0145] An MS and/or MS/MS spectrum of a protein whose identity was previously determined based on genomic sequencing of the protein's gene or Edman sequencing of the protein's N-terminal sequence, is an exemplary mass spectrum of a known protein.

3. *Temporary Databases*

[0146] A "temporary database" is simply tabulated data that is stored temporarily in random access memory (RAM). Processing such tabulated data when it is stored in RAM is significantly faster than processing the same data when it is stored on a hard drive.

E. *Computer Software*

1. *Overview*

[0147] In an exemplary embodiment, the invention provides a computer readable storage medium storing instructions that, when executed by a computer, causes the computer to perform a method for identification of an unknown protein. The method for identifying the unknown protein comprises: (i) comparing mass-to-charge (m/z) fragment ion peaks from an mass spectrum of the unknown protein to that of m/z fragment ion peaks from mass spectra of known protein sequences; and (ii) matching the m/z fragment ion peaks from the mass spectrometry spectrum of the unknown protein to the m/z fragment ion peaks from the mass spectrometry spectra of known protein sequences to provide mass spectrometry peak matches; (iii) scoring the mass spectrometry peak matches using the formula:

$$USDA\alpha \text{ Score} = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}$$

thereby providing a USDA α peak matching score; and (iv) ranking the USDA α peak matching score from highest to lowest; wherein the highest USDA α peak matching score constitutes an identification of the unknown protein.

[0148] In another exemplary embodiment, the invention provides a computer readable storage medium storing instructions that, when executed by a computer, causes the computer to perform a method for identification of an unknown protein. The method for identifying the unknown protein comprises: (i) comparing mass-to-charge (m/z) fragment ion peaks from an

mass spectrum of the unknown protein to that of m/z fragment ion peaks from mass spectra of known protein sequences; and (ii) matching the m/z fragment ion peaks from the mass spectrometry spectrum of the unknown protein to the m/z fragment ion peaks from the mass spectrometry spectra of known protein sequences to provide mass spectrometry peak matches; (iii) scoring the mass spectrometry peak matches using the formula:

$$USDA\beta \text{ Score} = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}$$

thereby providing a USDA β peak matching score; and (iv) ranking the USDA β peak matching score from highest to lowest; wherein the highest USDA β peak matching score constitutes an identification of the unknown protein.

[0149] In one exemplary embodiment, the instructions on the computer readable storage medium comprise a computer software program for the identification of proteins by analysis of fragment ions generated from gas phase fragmentation of singly charged protein ions generated by matrix-assisted laser desorption/ionization (MALDI) and analyzed by tandem time-of-flight (TOF) mass spectrometry (MS/MS). Prior enzymatic digestion of the proteins is not required. The software also provides instructions for performing a pair-wise comparison of an unknown MALDI-TOF-MS spectrum to that of known MALDI-TOF-MS spectra and rank spectra according to spectral similarity thereby enabling a user to identify an unknown spectrum.

[0150] In one exemplary embodiment, the USDA mass spectrometry comparison software (which in exemplary embodiments may utilize either the USDA α or the USDA β peak matching algorithms) is written in JAVA and JSP computer language. However, in other exemplary embodiments the software is written in C, C++, PHP, Assembly or any other web-based software languages known in the art. To ensure that the web based/cross platform component is available a web server based language is utilized *e.g.*, ASP, JSP, PHP, etc.

[0151] The code for software embodiments is disclosed herein in the computer program listing appendix. The USDA mass spectrometry comparison software (which in exemplary embodiments may utilize either the USDA α or the USDA β peak matching algorithms) comprises the following features, functionalities and attributes:

[0152] A web browser interface and associated databases that utilize MySQL for database management.

[0153] The USDA mass spectrometry comparison software enables efficient entry, deletion, retrieval and searching of organism (*e.g.*, microorganism) isolate information, *e.g.* in

the case of a microorganism taxonomy, origin, source, etc.

[0154] The USDA mass spectrometry comparison software enables the entry of ASCII formatted m/z and intensity MS and MS/MS data files to databases for subsequent searching, retrieval or deletion.

[0155] The USDA mass spectrometry comparison software enables the entry of single data files (or multiple data files) of ASCII formatted m/z MS/MS *in silico* data files to a database for subsequent searching, retrieval or deletion.

[0156] The USDA peak matching algorithms enable comparison of the mass-to-charge (m/z) of fragment ion peaks from a MS/MS spectrum to the m/z of fragment ions of an *in silico* MS/MS spectra of proteins. The software compares fragment ions of *in silico* MS/MS spectra to fragment ions of an MS/MS spectrum of an unknown protein when the mass of the protein falls within a pre-specified mass tolerance (e.g. ± 1.0 Th; ± 2.0 Th; ± 3.0 Th; ± 4.0 Th; ± 5.0 Th; wherein Th is Thomson the unit of m/z) of the precursor protein ion. When the difference between the m/z of a fragment ion in a MS/MS spectrum of an unknown protein and the m/z of a fragment ion in an *in silico* MS/MS spectrum fall within the set pre-specified mass tolerance (e.g. ± 2.5 Th) it is a "match". The USDA α peak matching algorithm has the formula:

$$USDA\alpha\ Score = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}$$

The USDA β peak matching algorithm has the formula:

$$USDA\beta\ Score = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}$$

[0157] The highest scoring protein/microorganism identification obtained using either the USDA α or USDA β peak matching algorithms, that is significantly higher than the second highest scoring protein/microorganism identification constitutes a presumptive correct identification. In an exemplary embodiment, significantly higher is a difference in score between the highest scoring identification and the second highest scoring identification that is equal to or greater than 2. In another exemplary embodiment, significantly higher is a difference in score between the highest scoring identification and the second highest scoring identification that is equal to or greater than 3. In another exemplary embodiment, significantly higher is a difference in score between the highest scoring identification and the

second highest scoring identification that is equal to or greater than 4 or more.

[0158] For convenience, in an exemplary embodiment, a probability-based significance testing algorithm (*e.g.*, p-value calculation *see* Demirev *et al.* (2005) *supra*) is incorporated into the USDA mass spectrometry comparison software. The USDA peak matching score and the p-value calculation are completely independent. In this embodiment, the results of each calculation are displayed side-by-side in the identification results window. In this embodiment, the USDA mass spectrometry comparison software enables a user to rank the protein/microorganism identifications according to either the USDA peak matching score or the probability-based significance testing algorithm.

[0159] Thus, the USDA mass spectrometry comparison software enables a user to enter and retrieve data, and search or delete a database containing ASCII formatted m/z and intensity MALDI-TOF-MS data. In an exemplary embodiment, the USDA mass spectrometry comparison software performs a pair-wise comparison of a MALDI-TOF-MS spectrum from an unknown protein to that of MALDI-TOF-MS spectra from samples of known proteins and to rank the spectra according to spectral similarity thereby providing a taxonomic classification of the unknown protein and hence of the source organism from which the unknown protein is derived. The algorithms used for such a pairwise comparison are disclosed above. In another exemplary embodiment, the USDA mass spectrometry comparison software assigns the identity of m/z peaks in a MALDI-TOF-MS spectrum by comparison to another database of theoretical protein molecular weights derived from genomic data (*e.g.*, open-reading frames or ORFs).

[0160] In one exemplary embodiment, for each unknown protein "identification", the USDA mass spectrometry comparison software displays a side-by-side comparison of the measured fragment ion m/z and the m/z of an *in silico* fragment ions to which it matches. When the identification is correct, a plot of the m/z error ($\Delta m/z$), *i.e.* the difference in m/z between the measured fragment ions and their *in silico* fragment ion matches, reveals a characteristic systematic error caused by the inherent calibration error of the instrument. Incorrect protein identifications, in contrast, are seen to display only statistically random error caused by false "matches". The confirmation of the correctness of the unknown protein identification by systematic error is independent of the mathematical algorithm used for unknown protein identification.

2. Software/database Structure and Functionality

[0161] FIG. 1 provides a general overview of the framework for the USDA mass

spectrometry comparison software. The software stored on a computer readable storage medium provides instructions that when executed by a computer, causes the computer to perform a method for identification of an unknown protein. The instructions comprising the USDA mass spectrometry comparison software utilizing either the USDA α or USDA β peak matching algorithms are provided herein in the computer program listing appendix.

[0162] FIG. 2 shows the database model describing how the data is stored. A *MySQL* Database Management System (*MySQL* DBMS) is used as the back-end (i.e. software that indirectly provides support for a client) for the archival/retrieval/analysis software developed for protein/microorganism identification by mass spectrometry (MS) and tandem mass spectrometry (MS/MS). A *Tomcat Apache* (TA) web server delivers the user/operator interface to any computer platform (e.g. PCs, Macintosh, Unix). The front-end (i.e. the software with which the user/operator interacts) was created using *Java Server Pages* (JSP). A second front-end is a *Java* client (i.e. a binary file [jar file] that is interpreted by the Java Runtime Environment or JRE) which was created to upload automatically multiple *in silico* MS/MS data files to the *MySQL* DB and is operated independently of the TA/JSP server. The web client allows the user/operator to search, archive, retrieve and edit microorganism strain information as well as perform spectral similarity analysis of MS and MS/MS data of microorganism proteins or toxin proteins, e.g. comparison of MS $_a \leftrightarrow$ MS $_b$ or MS/MS $_a \leftrightarrow$ MS/MS $_b$ or MS/MS $\leftrightarrow$ *In Silico* MS/MS. The web client also allows a user/operator to display plots of archived MS, MS/MS and *in silico* MS/MS data. The web client also facilitates visual comparison of putative "matched" data by displaying stacked or mirrored plots of MS $\leftrightarrow$ MS, MS/MS $\leftrightarrow$ MS/MS or MS/MS $\leftrightarrow$ *In Silico* MS/MS data.

3. Spectral Similarity Analysis

[0163] FIG. 3 describes the USDA mass spectrometry comparison software process used to compare each dataset. Spectral similarity analysis of MS or MS/MS data involves multiple steps. First, the user/operator selects the parameters under which the data analysis is to be conducted. Exemplary parameters include, but are not limited to: 1. the specific microorganisms MS/MS data as determined by a primary key (i.e. an integer that uniquely identifies a dataset); 2. the mass range to be compared; 3. the minimum relative intensity threshold of MS peaks to be compared; 4. the tolerance of the mass-to-charge ($\Delta m/z$) when comparing the m/z of two fragment ion peaks; 5. the scoring algorithm to be used in the analysis; 6. the sorting the identifications by their scores. The user/operator may also choose

search parameters that affect the list of reference data sets, for example, 1. taxonomical identification of the microorganism; 2. identification number; 3. protein name; 4. text notes of the references.

[0164] Second, the primary keys (an integer that uniquely identifies a dataset) of the datasets are stored in random access memory (RAM) and looped through and their respective data to be individually compared against the user/operator selected dataset iteratively. Both datasets, the one chosen by the user/operator to run for a comparison and the current iteration for comparison against the whole database, are "filtered" by the parameters selected by the user/operator as described above. This is then placed into two separate memory tables to be compared against one another. Memory table is a term being used to describe a *MySQL* DB table that utilizes the memory based storage engine (e.g. RAM). This increases the speed of the comparison.

4. *USDA α Peak Matching Algorithm*

[0165] The USDA peak matching algorithm counts the total number of *in silico* fragment ions and adds this number to the total number of MS/MS fragment ions whose intensity is equal to or greater than the minimum intensity threshold.

[0166] The algorithm then counts the total number of *in silico* fragment ions that are within $\Delta m/z$ of MS/MS fragment ions, i.e. the number of "matches" between two datasets. The number of counted "matches" is multiplied by two and divided by the previously mentioned summation of *in silico* MS/MS and MS/MS fragment ions. This number is then multiplied by 100% to give the USDA peak matching score. The USDA peak matching score has a theoretical range from 0 to 100% with 0% meaning no matches were identified and 100% meaning every single peak was matched between the two datasets. A non-zero $\Delta m/z$ means that it is possible for an MS/MS fragment ion peak to "match" to two (or more) *in silico* fragment ion peaks (or vice versa). Such "multiple" matches are counted only once by the USDA peak matching algorithm, otherwise a USDA peak matching score greater than 100% may occur. This counting/matching/scoring process is identical to the method used for comparing two MS spectra of MALDI-TOF MS data.

5. *USDA β Peak Matching Algorithm*

[0167] The peak matching algorithm involves counting the number of MS/MS fragment ions whose intensity is equal to or greater than a relative intensity threshold (e.g. 2%). The algorithm then counts the number of *in silico* fragment ions whose m/z falls within a specified

m/z tolerance (e.g. ± 2.5 Th) to that of the m/z of MS/MS fragment ions, i.e. counting the number of "peak matches" between MS/MS fragment ions and *in silico* fragment ions of the two datasets. The number of "peak matches" is then divided by the total number of MS/MS fragment ions above the specified intensity threshold. This number is then multiplied by 100% to give the peak matching score shown below:

$$USDA\beta \text{ Score} = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}$$

[0168] The USDA β peak matching score has a theoretical range from 0 to 100%. Zero percent indicates no peak matches were identified and 100% indicates every single MS/MS fragment ion was matched to an *in silico* fragment ion of an identification. A non-zero fragment ion m/z tolerance indicates that it is possible for an MS/MS fragment ion m/z to "match" to the m/z of two (or more) *in silico* fragment ions (or vice versa). Such multiple matches are counted only once by the algorithm, otherwise a score greater than 100% may result. The highest scoring protein/microorganism identification that is *significantly* higher than the second highest scoring protein/microorganism identification constitutes a presumptive correct identification. "Significantly" as used herein refers a relative difference in score between the top identification and the second highest scoring identification, typically of about 15-20% (or greater).

[0169] Software functionality allows selective operation of one or the other algorithms (or both). Algorithm computation time is provided by the software. In addition, the protein/microorganism identifications can be "ranked" by either the peak matching scores or p-values.

6. *Computation speed of USDA algorithms*

[0170] The USDA peak matching algorithms are less computationally complex than other protein identification algorithms known in the art e.g., a p-value calculation (*see e.g.*, Demirev et al. (2005) *supra*). Thus, the computation time of the USDA peak matching algorithms are rapid as well as accurate. Indeed, in an exemplary embodiment, computation time is as much as 50% faster than the same p-value calculation. As the number of *in silico* bacterial proteins increase due to the increasing number of bacterial genomes, algorithm computation time plays an increasingly important factor in rapid and efficient protein/microorganism identification.

[0171] The following examples are offered to illustrate, but not to limit the invention.

EXAMPLES

Example 1

[0172] The following example illustrates an exemplary microorganism identification protocol for the identification of a microorganism using matrix assisted laser desorption ionization (MALDI) mass spectrometry to identify an unknown protein and its corresponding source organism.

Flow Chart of Protein/Microorganism Identification

[0173] A 1 μ L loop of bacterial cells suspended in 0.5 mL of solution (33% CH₃CN, 67% H₂O and 0.1% TFA). Bead-beating 60 seconds, centrifugation 5 minutes @ 10,000 rpm.



[0174] A 0.5 μ L aliquot of supernatant is deposited onto stainless steel target pre-spotted with MALDI matrix.



[0175] Collect MS and MS/MS data of bacterial strain using MALDI tandem mass spectrometer (e.g. MALDI-TOF-TOF).



[0176] Raw MS and MS/MS data is processed and converted to ASCII formatted mass-to-charge (m/z) vs. intensity data. The ASCII formatted MS and MS/MS data are uploaded to their respective MySQL MS database (DB) and MySQL MS/MS DB of the USDA software.



[0177] From the *ExPASy TagIdent* web server, a search/retrieve/download was conducted of all bacterial proteins with a molecular weight (MW) equivalent in mass to the protein precursor ion m/z (PPI m/z) after removal of its proton (-1 Th) and within a pre-set mass tolerance, e.g. $MW = (PPI\ m/z - 1) \pm 5\ Da$.



[0178] From the *ExPASy TagIdent* web server, a search/retrieve/download was conducted of all bacterial proteins with a MW equivalent in mass to the PPI m/z after removal of its proton (-1 Th) and addition of the mass of a methionine residue (+131 Da) and within a pre-set mass tolerance, e.g. $MW = [(PPI\ m/z - 1) + 131] \pm 5\ Da$.



[0179] The $MW = (PPI\ m/z - 1) \pm 5$ Da and the $MW = [(PPI\ m/z - 1) + 131] \pm 5$ Da searches generate two large Fasta files each containing ~ 600-700 bacterial protein sequences.



[0180] The $MW = [(PPI\ m/z - 1) + 131] \pm 5$ Da FASTA file is edited to remove N-terminal methionine from all protein amino acid sequences. The protein identification is also modified to highlight this *in silico* post-translational modification (PTM).



[0181] Each multi-sequence Fasta file is processed using a beta-version of the commercial software GPMW. The beta-version (8.01a5) has enhanced features developed by ©Lighthouse Data at USDA's request.



[0182] Each processed FASTA file generates ~ 600-700 text files. Each text file contains the taxonomic identification of the microorganism, the protein identification and its MW (average) and its *in silico* fragment ions as identified by their m/z (average), their ion type (a, b, b-18, y, y-17, y-18) and the two amino acid residues immediately adjacent to the site of polypeptide backbone cleavage which generated the specific fragment ion.



[0183] The $MW = (PPI\ m/z - 1) \pm 5$ Da text files are sorted by file size in order to identify any proteins having a putative signal peptide (SP). These SP-associated text files are excluded from upload to the MySQL *In Silico* DB. Instead the SP-identified proteins are individually downloaded from the *ExPASy* web server as single sequence FASTA files. These single sequence FASTA files are edited to remove the SP, and the protein identification is modified to highlight this *in silico* post-translational modification (PTM). These new sequences are compiled into a single FASTA file and processed by GPMW. The text files generated are then ready for upload to the MySQL *In Silico* DB.



[0184] The $MW = [(PPI\ m/z - 1) + 131] \pm 5$ Da text files are sorted by file size in order to identify a small number of proteins having a putative SP. These SP-associated text files are excluded from upload to the MySQL *In Silico* DB. The non-SP-associated text files are ready for upload to the MySQL *In Silico* DB.



[0185] *In silico* text files are batch uploaded to the MySQL *In Silico* DB using a *Java*

client (i.e. a binary file [jar file] that is interpreted by the Java Runtime Environment or JRE) that is part of, but separate from, the USDA software.



[0186] A "MS/MS To *In Silico* Comparison" functionality of the USDA software compares the MS/MS spectrum against all *in silico* MS/MS spectra whose calculated protein MW falls within the pre-specified protein MW tolerance of the PPI m/z , e.g. $MW = (PPI\ m/z - 1) \pm 5\ Da$.



[0187] Protein/bacteria IDs are sorted and displayed on the basis of their USDA score (α or β) and/or p-value.



[0188] Protein/bacteria IDs are also confirmed by plots of fragment ion error analysis: systematic error (i.e. instrument calibration error) *versus* random error.



[0189] Protein/bacteria IDs are also confirmed by residue-specific fragment ions, e.g. the number and percentage of D,E,P-specific fragment ions.

Example 2:

[0190] The following example illustrates an exemplary method for bacterial protein extraction for MALDI-TOF/TOF analysis.

[0191] Bacterial proteins were extracted from *Campylobacter* cells using a technique that has been previously reported (*see e.g.*, Mandrell RE, et al. *Applied Environ. Microbio.* 2005; **71**: 6292; Fagerquist CK, et al. *Anal. Chem.* 2005; **77**: 4897; Fagerquist CK, et al. *J. Proteome Research.* 2006; **5**: 2527. Briefly, *C. jejuni* (strain RM1221), *C. coli* (strain RM2228), *C. upsaliensis* (strain RM3195) and *C. lari* (strain RM2100) were each cultured on non-selective growth media for 24-48 hours. One μL of cells were transferred to a microcentrifuge tube containing 0.5 mL of extraction solvents (67% water, 33 % acetonitrile and 0.1 % TFA) and 40 mg of 0.1 mm zirconia/silica beads (BioSpec Products Inc., Bartlesville, OK). The tube was capped and agitated for 60 seconds with a bead-beater. The tube was then centrifuged at 10,000 rpm for 4-5 minutes. An equal volume of sample supernatant was combined with an equal volume of either α -cyano-4-hydroxycinnamic acid (CCA) or sinapinic acid, mixed and spotted onto a 384-spot stainless steel plate. Samples were allowed to air-dry prior to analysis.

Example 3:

[0192] The following example illustrates an exemplary mass analysis using high energy collision-induced dissociation as well as post-source dissociation on a MALDI tandem-TOF mass spectrometer to generate MS/MS spectra in reflectron-mode in positive ion mode.

[0193] Samples were analyzed using an Applied Biosystems 4800 TOF-TOF™ Analyzer mass spectrometer (Foster City, CA). Spectra were analyzed in both MS linear-mode and MS/MS in reflectron-mode in positive ion mode. Source acceleration in the first source in linear-mode was 20.00 kV. In MS/MS reflectron-mode, ions were initially accelerated at 8.00 kV in the first source, mass-selected with a timed ion selector, the mass-selected ions were then decelerated to 1.70 kV, fragmented with the target gas (air) in the floating collision cell at 2 kV and the fragment ions formed were re-accelerated to 15 kV in the second source. Data was collected with the metastable (precursor ion) suppressor, located after the 2nd source, in both "on" and "off" mode. A two-stage reflectron mirror assembly was operated at 10.910 kV (mirror 1) at 18.750 kV (mirror 2). Both linear and reflectron detectors were operated at 2.190 kV. Linear data were externally calibrated using insulin (MW = 5734.59 Da), thioredoxin (MW = 111674.48 Da) and apomyoglobin (MW = 16,952.56 Da). Reflectron-TOF MS/MS spectra were calibrated using updated instrument default calibration.

[0194] MS and MS/MS data were analyzed using the commercially available instrument software (Data Explorer Software®, Version 4.9). MS data were processed using a noise filter (correlation factor = 0.7). MS/MS data were processed first with an advanced baseline correction (peak width = 32, flexibility = 0.5, degree = 0.1), then with noise removal (standard deviations = 2) and finally with a Gaussian smooth (filter width = 31).

Example 4:

[0195] The intact protein biomarker, thioredoxin from *Campylobacter upsaliensis* strain RM3195 was mass-selected and fragmented in the gas phase using a MALDI-TOF-TOF mass spectrometer by methods known in the art (*see e.g.*, Example 3 above). The protein was identified previously by "bottom-up" proteomics techniques (*see e.g.*, Fagerquist, C.K. *J. Proteome Research*. 2007; 6: 2539-2549). Since the identity of the proteins had been determined previously, it served as a control protein for confirmation of the accuracy and rapidity of identification achievable by the method for identification of an unknown protein disclosed herein.

[0196] The sequence-specific fragment ions generated were used to identify the protein

(and thus the microorganism) by comparison against a database containing *in silico* fragment ions derived from protein sequences from bacterial genomic databases.

[0197] The USDA peak-matching algorithm alpha (α) which has the formula:

$$USDA\alpha\ Score = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}$$

was incorporated into the data analysis software as disclosed herein and was used to score/rank protein/microorganism identifications. The algorithm correctly identified the thioredoxin protein biomarker from nearly ~ 1400 possible bacterial proteins. The algorithm also gave comparable rankings in protein identifications as that obtained with a much more complicated p-value calculation (*see e.g.*, Demirev *et al.* (2005) *Anal Chem* 77:7455).

[0198] Protein identifications made using the USDA peak-matching mathematical algorithm were confirmed by error analysis as disclosed herein and also by residue specific analysis. As is illustrated below, the correctly identified protein exhibits only systematic error that is due to the inherent calibration error of the mass analyzer, whereas incorrect identifications exhibit random error caused by false "matches". Furthermore, correct identification is confirmed by residue specific analysis. Correct identifications as confirmed by residue specific analysis reveal an increase in the absolute value of the highest USDA peak matching score, as well as an increase in the difference between the highest USDA peak matching score and the second highest USDA peak matching score.

Example 5:

[0199] The following example illustrates that a correct protein identification made using the peak-matching mathematical algorithm disclosed herein is confirmed by analysis of the difference in *m/z* between the MS/MS fragment ions of the protein and their putative "matches" from a MS/MS *in silico* spectra. For a correct identification, of thioredoxin protein, a linear regression analysis fit of the data reveals only systematic error which is due to the inherent calibration error of the mass analyzer (*see e.g.*, FIG. 9). In contrast, a linear regression analysis fit of the data for an incorrect identification of the protein thioredoxin exhibits random error caused by false "matches" (*see e.g.*, FIG. 10).

[0200] FIG. 4 shows an MS spectrum (as displayed in the MALDI-TOF-TOF instrument software) of the extracted cell lysate of *Campylobacter upsaliensis* (strain RM3195) analyzed by MALDI-TOF-MS using sinapinic acid as the MALDI matrix. The protein biomarker ion

at m/z 11138.9 was analyzed by tandem mass spectrometry. The MS/MS spectrum of protein biomarker ion at m/z 11138.9 is shown in FIG. 5.

[0201] FIG. 5 shows the MS/MS spectrum (as displayed in the MALDI-TOF-TOF instrument software) of the protein biomarker ion of *Campylobacter upsaliensis* (strain RM3195) at m/z 11138.9 analyzed on a MALDI-TOF-TOF-MS using sinapinic acid as the MALDI matrix. Prominent fragment ions are identified by their m/z , ion type and number and amino acid residues immediately adjacent to the site of polypeptide backbone cleavage giving rise to the fragment ion. As can be seen, most (if not all) of the fragment ions are the result of polypeptide backbone cleavage immediately adjacent to a D (aspartic acid) or E (glutamic acid) residues. This MS/MS data is then centroided (i.e. converted to ASCII format: m/z vs. absolute intensity) and exported for uploading to the USDA software.

[0202] FIG. 6 shows an ASCII MS/MS spectrum (m/z vs. relative intensity as displayed in the USDA software window) of thioredoxin from *Campylobacter upsaliensis* (strain RM3195) previously displayed in FIG. 5. The MS/MS data in FIG. 5 was centroided (i.e. converted to ASCII format: m/z vs. absolute intensity), exported from the MALDI-TOF-TOF instrument software and imported to the MS/MS databases of the USDA software. The m/z and absolute intensity of MS/MS fragment ions are shown in the sidebar.

[0203] FIG. 7 shows The "MS/MS To *In Silico* Comparison" search parameter table as displayed in the USDA mass spectrometry comparison software window. The user/operator defines the m/z range to be compared, the number of results to be displayed, the average molecular weight of the protein to be searched (with an associated molecular weight tolerance, e.g. ± 10 Da), the fragment ion m/z tolerance (in Th) and restrictions (if any) on residue-specific fragment ions, i.e. fragment ions generated from polypeptide cleavage immediately adjacent to specific amino acid residues. The absence of an entry in the "Residues" field indicates that *all* fragment ions generated from the polypeptide (protein) are to be included in the search. The user/operator can also select which scoring algorithm to use for analysis, i.e. USDA peak matching algorithm or the p-value algorithm (or both). The scores/identifications can also be ranked/sorted.

[0204] FIG. 8 shows the *in silico* MS/MS spectrum (as displayed in the USDA software window) of the protein thioredoxin of *Campylobacter upsaliensis* (strain RM3195). The *in silico* sequence was post-translationally modified (PTM) to remove the N-terminal methionine. *In silico* fragment ions m/z are shown in the sidebar.

[0205] Table 1A (below) displays the top five identifications of a protein biomarker of

Campylobacter upsaliensis (strain RM3195) at m/z 11138.9 (as shown in FIG. 4) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry (as shown in FIG. 5 and FIG. 6) and compared to all *in silico* MS/MS fragment ions of bacterial protein sequences having the same molecular weight as the biomarker (within ± 5 Da. This corresponded to 1409 *in silico* protein sequences). Table 1A compares peak matching scores obtained using the USDA α peak matching algorithm to the p-value calculation known in the art. The protein biomarker had been identified previously by "bottom-up" proteomics techniques as thioredoxin (Fagerquist, C. K., 2007, *J. Proteome Res.*, 6, 2539-2549). The top identification correctly identifies the protein biomarker and its source microorganism.

Table 1A					
In Silico ID	Identifier	Sample Name	Protein	USDA Score- α	P-value
7653	>Q4HNM5 Q4HNM5_CAMUP	<i>Campylobacter upsaliensis</i> RM3195.	Thioredoxin PTM_Met 11136.76	12.95	7.7E-19
6714	>B1WRD3 B1WRD3_CYAA5	<i>Cyanotheca</i> (strain ATCC 51142).	Carbon dioxide concentrating mechanism protein 11133.85	8.43	3.6E-6
6462	>A4T659 A4T659_MYCGI	<i>Mycobacterium gilvum</i> (strain PYR_GCK) (<i>Mycobacterium flavescens</i> (strain ATCC 700033 / PYR_GCK)).	Putative uncharacterized protein 11142.24	7.36	3.1E-4
7088	>Q1LMQ8 Q1LMQ8_RALME	<i>Ralstonia metallidurans</i> (strain CH34 / ATCC 43123 / DSM 2839).	Putative uncharacterized protein PTM_23SigPep 11128.81	6.93	1.1E-3
7545	>Q18XI2 Q18XI2_DESHD	<i>Desulfitobacterium hafniense</i> (strain DCB_2).	Small multidrug resistance protein PTM_Met 11138.67	6.74	1.6E-3
<u>MS/MS to <i>In silico</i> Comparison Parameters</u> Intensity threshold: 2% m/z range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 11138 $\pm$ 10 Da. ALL <i>in silico</i> fragment ions compared. "PTM_Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM_#SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm Computation Speeds</u> USDA peak matching algorithm alpha: 33.8 seconds. P-value calculation: 47.1 seconds.					

[0206] Table 1B (below) displays the top five identifications of the same protein biomarker of *Campylobacter upsaliensis* (strain RM3195) at m/z 11138.9 (as shown in FIG. 4) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry (as shown in FIG. 5 and FIG. 6) and compared to all 1409 *in silico* MS/MS fragment ions of bacterial protein sequences having the same molecular weight as the biomarker (within ± 5 Da). Table 1B compares peak matching scores obtained using the USDA β peak matching algorithm to the

p-value calculation known in the art.

Table 1B					
<i>In Silico</i> ID	Identifier	Sample Name	Protein	USDA Score-β	P-value
1262	>Q4HNM5 Q4HNM5_CAMUP	<i>Campylobacter upsaliensis</i> RM3195.	Thioredoxin PTM_Met 11136.76	82.69	7.7E-19
323	>B1WRD3 B1WRD3_CYAA5	<i>Cyanotheca</i> (strain ATCC 51142).	Carbon dioxide concentrating mechanism protein 11133.85	53.85	3.6E-6
71	>A4T659 A4T659_MYCGI	<i>Mycobacterium gilvum</i> (strain PYR-GCK) (<i>Mycobacterium flavescens</i> (strain ATCC 700033 / PYR-GCK)).	Putative uncharacterized protein 11142.24	46.15	3.1E-4
697	>Q1LMQ8 Q1LMQ8_RALME	<i>Ralstonia metallidurans</i> (strain CH34 / ATCC 43123 / DSM 2839).	Putative uncharacterized protein PTM_23SigPep 11128.81	44.23	1.1E-3
1154	>Q18XI2 Q18XI2_DESHD	<i>Desulfitobacterium hafniense</i> (strain DCB-2).	Small multidrug resistance protein PTM_Met 11138.67	44.23	1.6E-3
<u>MS/MS to <i>In silico</i> Comparison Parameters</u> Intensity threshold: 2% m/z range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 11138 ± 10 Da. ALL <i>in silico</i> fragment ions compared. "PTM_Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM_#SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm Computation Speeds</u> USDA peak matching algorithm beta: 35.0 seconds. P-value calculation: 48.4 seconds.					

[0207] FIG. 9 shows a plot of fragment ion error (as displayed in the USDA software) as a function of the MS/MS fragment ion mass-to-charge (m/z) of the protein biomarker thioredoxin (from *Campylobacter upsaliensis* strain RM3195) and its "match" to the *in silico* fragment ions of thioredoxin of *Campylobacter upsaliensis* strain RM3195 (the highest scoring identification and the correct identification). Fragment ion error ($\Delta m/z$) is defined here as the difference between m/z of an MS/MS fragment ion and the m/z of its putative *in silico* fragment ion "match". A red square represents a "match" whose *in silico* fragment ion was generated from a protein backbone cleavage site that is adjacent to a D, E or P residue. A black square represents a "match" whose *in silico* fragment ion was generated from a protein backbone cleavage site that is adjacent to *only* non-D,E,P residues. Note the high number (27) and percentage (87.1%) of "matches" that are associated with D,E,P residues supporting the correctness of the identification. A linear regression fit of the data indicates a correlation ($R^2 = 0.6438$) between $\Delta m/z$ and MS/MS fragment ion m/z due to systematic error caused by calibration error of the mass analyzer. This systematic error confirms that this is a correct identification.

[0208] FIG. 10 shows plot of fragment ion error (as displayed in the USDA software) as a function of the MS/MS fragment ion mass-to-charge (m/z) of the protein biomarker thioredoxin (from *Campylobacter upsaliensis* strain RM3195) and its "match" to the *in silico* fragment ions of carbon dioxide concentrating mechanism protein of *Cyanothece* strain ATCC 51142 (the second highest scoring identification and an incorrect identification). Fragment ion error ($\Delta m/z$) is defined here as the difference between m/z of an MS/MS fragment ion and the m/z of its putative *in silico* fragment ion "match". A red square represents a "match" whose *in silico* fragment ion was generated from a protein backbone cleavage site that is adjacent to a D, E or P residue. A black square represents a "match" whose *in silico* fragment ion was generated from a protein backbone cleavage site that is adjacent to *only* non-D,E,P residues. Note the low number (5) and percentage (26.3%) of "matches" that are D,E,P-associated supporting that this is not the correct identification. A linear regression fit of the data indicates a very poor correlation ($R^2 = 0.0206$) between $\Delta m/z$ and MS/MS fragment ion m/z , i.e. random statistical error. This random error supports that this is an incorrect identification.

[0209] Table 1C (below) displays the top five identifications of a protein biomarker of *Campylobacter upsaliensis* (strain RM3195) at m/z 11138.9 (as shown in FIG. 4) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry (as shown in FIG. 5 and FIG. 6) compared to D,E,P-specific *in silico* MS/MS fragment ions of bacterial protein sequences having the same molecular weight as the biomarker (within ± 5 Da). This corresponded to 1409 *in silico* protein sequences. The protein biomarker had been identified previously by "bottom-up" proteomics techniques as thioredoxin (Fagerquist, C. K., 2007, *J. Proteome Res.*, 6, 2539-2549). The top identification correctly identifies the protein biomarker and its source microorganism. Table 1C compares peak matching scores obtained using the USDA α peak matching algorithm to the p-value calculation known in the art.

Table 1C					
<i>In Silico</i> ID	Identifier	Sample Name	Protein	USDA Score _{US}	P-value
7653	>Q4HNM5 Q4HNM5_CAMUP	<i>Campylobacter upsaliensis</i> RM3195.	Thioredoxin PTM_Met 11136.76	25.00	3.0E-21
7333	>A8LX75 A8LX75_SALAI	<i>Salinira arenicola</i> (strain CNS_205).	Putative uncharacterized protein PTM_Met 11140.56	12.99	9.0E-5
6534	>A7A905 A7A905_BIFAD	<i>Bifidobacterium adolescentis</i> L2_32.	Putative uncharacterized protein 11134.94	11.50	9.4E-5
6527	>A6QD71 A6QD71_STAAE	<i>Staphylococcus aureus</i> (strain Newman).	Putative uncharacterized protein 11138.98	11.54	1.5E-4
6840	>Q2G1R1 Q2G1R1_STAA8	<i>Staphylococcus aureus</i> (strain NCTC 8325).	Putative uncharacterized protein 11138.98	11.54	1.5E-4
<u>MS/MS to <i>In silico</i> Comparison Parameters</u> Intensity threshold: 2% <i>m/z</i> range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 11138 ± 10 Da. D,E,P-specific <i>in silico</i> fragment ions compared. "PTM_Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM_#SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm computation speeds</u> USDA peak matching algorithm alpha: 16.1 seconds. P-value calculation: 37.1 seconds.					

[0210] Table 1D (below) displays the top five identifications of the same protein biomarker as in Table 1C of *Campylobacter upsaliensis* (strain RM3195) at *m/z* 11138.9 (as shown in FIG. 4) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry (as shown in FIG. 5 and FIG. 6) compared to D,E,P-specific *in silico* MS/MS fragment ions of bacterial protein sequences having the same molecular weight as the biomarker (within ± 5 Da). This corresponded to 1409 *in silico* protein sequences. The protein biomarker had been identified previously by "bottom-up" proteomics techniques as thioredoxin (Fagerquist, C. K., 2007, *J. Proteome Res.*, 6, 2539-2549). The top identification correctly identifies the protein biomarker and its source microorganism. Table 1D compares peak matching scores obtained using the USDA β peak matching algorithm to the p-value calculation known in the art.

Table 1D					
In Silico ID	Identifier	Sample Name	Protein	USDA Score-β	P-value
1262	>Q4HNM5 Q4HNM5_CAMUP	<i>Campylobacter upsaliensis</i> RM3195.	Thioredoxin PTM_Met 11136.76	61.54	3.0E-21
143	>A7A905 A7A905_BIFAD	<i>Bifidobacterium adolescentis</i> L2-32.	Putative uncharacterized protein 11134.94	25.00	
942	>A8LX75 A8LX75_SALAI	<i>Salinira arenicola</i> (strain CNS-205).	Putative uncharacterized protein PTM_Met 11140.56		9.0E-5
136	>A6QD71 A6QD71_STAAE	<i>Staphylococcus aureus</i> (strain Newman).	Putative uncharacterized protein 11138.98	23.08	
143	>A7A905 A7A905_BIFAD	<i>Bifidobacterium adolescentis</i> L2-32.	Putative uncharacterized protein 11134.94		9.4E-5
145	>A7BQP4 A7BQP4_9GAMM	<i>Beggiatoa</i> PS.	Putative uncharacterized protein 11132.6	23.08	
136	>A6QD71 A6QD71_STAAE	<i>Staphylococcus aureus</i> (strain Newman).	Putative uncharacterized protein 11138.98		1.5E-4
271	>B1BI09 B1BI09_CLOPE	<i>Clostridium perfringens</i> C str. JGS1495.	Putative uncharacterized protein 11140.05	23.08	
449	>Q2G1R1 Q2G1R1_STAA8	<i>Staphylococcus aureus</i> (strain NCTC 8325).	Putative uncharacterized protein 11138.98		1.5E-4
<u>MS/MS to In silico Comparison Parameters</u> Intensity threshold: 2% m/z range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 11138 ± 10 Da. D,E,P-specific <i>in silico</i> fragment ions compared. "PTM_Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM_#SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm computation speeds</u> USDA peak matching algorithm beta: 18.1 seconds. P-value calculation: 39.4 seconds.					

[0211] Table 2A (below) displays the top five identifications of a protein biomarker of *Campylobacter upsaliensis* (strain RM3195) at m/z 12855.3 (as shown in FIG. 4) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry and compared to ALL *in silico* MS/MS fragment ions of bacterial protein sequences having the same molecular weight as the biomarker (within ± 5 Da). This corresponded to 1315 *in silico* protein sequences. The protein biomarker had been identified previously by "bottom-up" proteomics techniques as ribosomal protein 50S L7/L12 (Fagerquist, C. K., 2007, *J. Proteome Res.*, 6, 2539-2549). The top identification correctly identifies the protein biomarker and its source microorganism.

Table 2A					
<i>In Silico</i> ID	Identifier	Sample Name	Protein	USDA Score-8	P-value
8942	>Q4HS60 Q4HS60_CAMUP	<i>Campylobacter upsaliensis</i> RM3195.	Ribosomal protein L7/L12 PTM_Met 12855.65	9.32	1.1E-10
8941	>Q4HDZ9 Q4HDZ9_CAMCO	<i>Campylobacter coli</i> RM2228.	Ribosomal protein L7/L12 PTM_Met 12853.74	7.30	1.5E-5
8543	>A5ZFY2 A5ZFY2_9BACE	<i>Bacteroides caccae</i> ATCC 43185.	Putative uncharacterized protein PTM_Met, 12852.15	6.54	8.7E-4
8759	>B2HXJ0 B2HXJ0_ACIBA	<i>Acinetobacter baumannii</i> ACICU.	Putative transcriptional regulator, TetR family, PTM_Met 12853.04	6.30	2.8E-3
8923	>Q3VUB8 Q3VUB8_PROAE	<i>Prosthecochloris aestuarii</i> DSM 271.	Ribosomal protein L7/L12 PTM_Met 12855.76	6.00	2.8E-3
<u>MS/MS to <i>In silico</i> Comparison Parameters</u> Intensity threshold: 2% m/z range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 12854 $\pm$ 10 Da. ALL <i>in silico</i> fragment ions compared. "PTM_Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM_#SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm Computation Speeds</u> USDA peak matching algorithm alpha: 34.7 seconds. P-value calculation: 50.3 seconds.					

[0212] **Table 2B** (below) displays the top five identifications of a protein biomarker of *Campylobacter upsaliensis* (strain RM3195) at m/z 12855.3 (as shown in FIG. 4) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry and compared to D,E,P-specific *in silico* MS/MS fragment ions of bacterial protein sequences having the same molecular weight as the biomarker (within ± 5 Da). This corresponded to 1315 *in silico* protein sequences. The protein biomarker had been identified previously by "bottom-up" proteomics techniques as ribosomal protein 50S L7/L12 (Fagerquist, C. K., 2007, *J. Proteome Res.*, 6, 2539-2549). The top identification correctly identifies the protein biomarker and its source microorganism.

Table 2B					
In Silico ID	Identifier	Sample Name	Protein	USDA Score-S	P-value
8942	>Q4HS60 Q4HS60_CAMUP	<i>Campylobacter upsaliensis</i> RM3195.	Ribosomal protein L7/L12 PTM_Met 12855.65	17.75	1.3E-15
8941	>Q4HDZ9 Q4HDZ9_CAMCO	<i>Campylobacter coli</i> RM2228.	Ribosomal protein L7/L12 PTM_Met 12853.74	13.02	1.4E-8
8067	>B0ACQ4 B0ACQ4_9CLOT	<i>Clostridium bartlettii</i> DSM 16795.	Putative uncharacterized protein 12849.21	9.09	1.5E-3
8983	>Q63UK3 Q63UK3_BURPS	<i>Burkholderia pseudomallei</i> (<i>Pseudomonas pseudomallei</i>).	Putative membrane protein PTM_Met 12847.95	8.28	2.8E-3
8008	>A0HKV3 A0HKV3_COMTE	<i>Comamonas testosteroni</i> KF_1.	Iron_sulfur cluster assembly accessory protein 12856.3	8.09	4.6E-3
<u>MS/MS to In silico Comparison Parameters</u> Intensity threshold: 2% m/z range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 12854 ± 10 Da. D,E,P-specific <i>in silico</i> fragment ions compared. "PTM_Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM_#SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm Computation Speeds</u> USDA peak matching algorithm alpha: 16.5 seconds. P-value calculation: 41.2 seconds.					

[0213] Table 3A (below) displays the top five identifications of a protein biomarker of *Campylobacter upsaliensis* (strain RM3195) at m/z 9298.3 (as shown in FIG. 4) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry and compared to all *in silico* MS/MS fragment ions of bacterial protein sequences having the same molecular weight as the biomarker (within ± 5 Da). This corresponded to 1404 *in silico* protein sequences. The protein biomarker had been identified previously by "bottom-up" proteomics techniques as 10 kD chaperonin (Fagerquist, C. K., 2007, *J. Proteome Res.*, 6, 2539-2549). The top identification correctly identifies the protein biomarker and its source microorganism.

Table 3A					
<i>In Silico</i> ID	Identifier	Sample Name	Protein	USDA Score-s	P-value
10519	>Q4HRH0 Q4HRH0_CAMUP	<i>Campylobacter upsaliensis</i> RM3195.	Chaperonin, 10 kDa 9299.82	15.28	3.7E-10
9741	>B2JW66 B2JW66_9BURK	<i>Burkholderia phymatum</i> STM815.	PAAR repeat containing protein 9296.43	10.68	7.1E-4
9886	>Q7U8T4 Q7U8T4_SYNPX	<i>Synechococcus</i> (strain WH8102).	Putative uncharacterized protein 9294.61	10.97	9.0E-4
9141	>A4BQR2 A4BQR2_9GAMM	<i>Nitrococcus mobilis</i> Nb_231.	SirA like protein 9296.79	10.49	3.0E-3
10507	>A2P9Y8 A2P9Y8_VIBCH	<i>Vibrio cholerae</i> 1587.	Putative uncharacterized protein PTM_Met 9294.56	10.38	3.5E-3
<u>MS/MS to <i>In silico</i> Comparison Parameters</u> Intensity threshold: 2% m/z range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 9297 $\pm$ 10 Da. All <i>in silico</i> fragment ions compared. "PTM_Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM_#SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm Computation Speeds</u> USDA peak matching algorithm alpha: 37.9 seconds. P-value calculation: 80.8 seconds.					

[0214] Table 3B (below) displays the top five identifications of a protein biomarker of *Campylobacter upsaliensis* (strain RM3195) at m/z 9298.3 (as shown in FIG. 4) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry and compared to D,E,P-specific *in silico* MS/MS fragment ions of bacterial protein sequences having the same molecular weight as the biomarker (within ± 5 Da). This corresponded to 1404 *in silico* protein sequences. The protein biomarker had been identified previously by "bottom-up" proteomics techniques as 10 kD chaperonin (Fagerquist, C. K., 2007, *J. Proteome Res.*, 6, 2539-2549). The top identification correctly identifies the protein biomarker and its source microorganism.

Table 3B					
In Silico ID	Identifier	Sample Name	Protein	USDA Score-s	P-value
10519	>Q4HRH0 Q4HRH0_CAMUP	<i>Campylobacter upsaliensis</i> RM3195.	Chaperonin, 10 kDa 9299.82	26.88	9.3E-16
10272	>Q2JWH0 Q2JWH0_SYNJA	<i>Synechococcus</i> (strain JA_3_3Ab) (Cyanobacteria bacterium Yellowstone A Prime).	Putative uncharacterized protein PTM_Met 9293.46	12.78	7.1E-4
9290	>A8S0J6 A8S0J6_9CLOT	<i>Clostridium botetiae</i> ATCC BAA 613.	Putative uncharacterized protein 9301.58	11.62	5.5E-3
9445	>B1UDC6 B1UDC6_SYNP8	<i>Synechococcus</i> (strain PCC 8801 / RF 1) (Cyanothecae PCC 8801).	Putative uncharacterized protein 9299.41	11.08	6.5E-3
10361	>Q5FIU9 Q5FIU9_LACAC	<i>Lactobacillus acidophilus</i> .	Putative phosphoribosylformylglycinamide PTM_Met 9293.55	11.34	7.4E-3
<u>MS/MS to In silico Comparison Parameters:</u> Intensity threshold: 2%. <i>m/z</i> range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 9297 ± 10 Da. D,E,P-specific <i>in silico</i> fragment ions compared. "PTM_Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM_#SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm Computation Speeds</u> USDA peak matching algorithm alpha: 17.0 seconds. P-value calculation: 89.7 seconds.					

Example 6

[0215] The following Example illustrates a "top-down" proteomic approach for protein/microorganism identification of foodborne pathogens. In particular the Example illustrates the identification of a protein biomarker (putative uncharacterized protein yahO) whose amino acid sequence is unique to *E. coli* O157:H7 strains and thus can be used to distinguish O157:H7 strains from non-O157:H7 strains. Further, the Example illustrates that if multiple microorganisms are present in a culture, it is possible to identify multiple microorganisms on the basis of identification of at least one unique protein biomarker from each microorganism. Indeed, even though a non-genomically sequenced bacterial strain may have extensive protein sequence homology to a closely-related genomically sequenced bacterial strain the bacterial strains may nonetheless have slight differences in protein amino acid sequence. These amino acid variations can change the overall MW of the protein as well as affect the *m/z* of fragment ions generated from protein fragmentation.

Materials and Methods

Origin and cultivation of *E. coli* strains

[0216] *E. coli* strains were preserved at -80 °C in Microbank™ Microbial Preservation System beads (Pro-Lab Diagnostics, Richmond Hill, Ontario, Canada). The strains were propagated using a bead and streaking for isolation on LB agar (Becton Dickinson, Sparks MD). The inoculated agar plates were incubated overnight at 37 °C in normal atmosphere. Three *E. coli* strains were used in this study: two O157:H7 strains (EDL-933 and a Salinas strain) and one non-O157:H7, non-pathogenic *E. coli* strain. The genomically sequenced strain *E. coli* O157:H7 strain EDL-933 (designated **RM1272** in our strain collection) was originally isolated from raw hamburger associated with hemorrhagic colitis outbreak and binds to anti-O157 and anti-H7 MAbs. The EDL-933 strain was generously provided by Jim Keen, Clay Center, Nebraska. The other O157:H7 strain (designated **RM5603** in our strain collection) was isolated from a water sample taken from the Salinas River (Salinas, CA) in 2006.

[0217] **RM5603** is a negative for shiga toxin 1, positive for shiga toxin 2, positive for the intimin gene (*eae*) and highly positive by PCR. In addition, **RM5603** was analyzed by multi-locus variable-number tandem repeat analysis (MLVA). The non-pathogenic *E. coli* strain **RM3061** was isolated from Romaine lettuce as part of a USDA microbiological survey of produce conducted in 2002. This sample was analyzed by Biolog microbial ID (Hayward, CA) and identified as a non-pathogenic *E. coli*.

Protein biomarker extraction and analysis

[0218] Bacterial cells were harvested and their proteins extracted as described in detail previously. Briefly, a 1 µL loop of bacterial cells were bead-beat for 1 minute in 0.5 mL of extraction solution (67% water, 33 % acetonitrile and 0.1 % TFA) with 40 mg of 0.1 mm zirconia/silica beads (BioSpec Products Inc., Bartlesville, OK). After bead-beating, the sample was centrifuged at 10,000 rpm for 5 minutes.

MALDI-TOF-TOF-MS analysis

[0219] Cell lysate/protein extract samples were analyzed using a 4800 TOF-TOF™ proteomics analyzer (Applied Biosystems, Foster City, CA) as described in detail in a previous report. Briefly, equal aliquot volumes of sample extraction supernatant and a saturated solution of MALDI matrix were mixed. A 0.5 µL aliquot of this mixture was

deposited onto a stainless steel target having 384 target spots. Two MALDI matrices were utilized for this study: 3,5-dimethoxy-4-hydroxycinnamic acid (sinapinic acid, $M_r = 224.21$ Da) and α -cyano-4-hydroxycinnamic acid (HCCA, $M_r = 189.17$ Da). The mass spectrometer is equipped with a pulsed solid-state YAG laser (repetition rate: 200 Hz, $\lambda = 355$ nm, pulse width: 5 nsec). Samples were analyzed in both linear (MS analysis) and reflectron-modes (MS/MS analysis). The instrument was operated for detection of only positively charged ions. The instrument was externally calibrated in linear mode using bovine insulin (MW = 5733.58 Da), *E. coli* thioredoxin (MW = 11673.47 Da) and horse heart apomyoglobin (MW = 16,951.55 Da). The instrument was externally calibrated in reflectron-mode using the y-type fragment ions of glu¹-fibrino-peptide B (MW = 1570.60) at m/z 175.120 and 1441.635.

[0220] For MS analysis, laser desorbed ions were accelerated from the source at 20 kV after delayed ion extraction for improved ion focussing. Ions were separated over an effective field free length of 1.5 m before striking a multichannel plate detector operated at 2.190 kV. For MS/MS analysis, ions were accelerated from the source at 8.0 kV after delayed extraction. Ions were separated in a field free region between the source and the collision cell. A timed ion selector (TIS) before the collision cell deflects (mass-selects) ions on the basis of arrival time so that only ions with a pre-selected m/z (i.e. arrival time) pass through the TIS. The TIS window was typically operated at ± 100 Da for MS/MS analysis. Ions passing through the TIS were decelerated to 1.70 kV prior to entry into a floating collision cell at 2.0 kV. Ions were fragmented by post-source dissociation (PSD) at higher than normal laser fluence. Ions exiting the collision cell were re-accelerated to 15 kV. An ion gate, after the second acceleration region, was used to suppress the un-fragmented protein ion signal. A two-stage reflectron mirror assembly (mirror 1: 10.910 kV, mirror 2: 18.750 kV) separates and deflects the fragment ions toward the reflectron multichannel plate detector also operated at 2.190 kV.

[0221] MS data was collected and summed from 1000-2000 laser shots which provided excellent signal-to-noise (S/N). For MS/MS data, because of low the fragmentation efficiency of singly charged protein ions, 30,000 to 40,000 laser shots were collected and summed in order to improve S/N. At 200 Hz laser repetition rate this required 2-3 minutes data acquisition from a single spot. The ultrafast MALDI target rastering allowed higher than normal laser fluence, required for protein ion fragmentation by PSD, without exhausting the sample spot.

[0222] Raw MS and MS/MS data were processed using the instrument software (Data Explorer® Software, Version 4.9). MS data were subjected to noise filtering (correlation factor = 0.7). Raw MS/MS data were subjected to an advanced baseline correction (peak width = 32, flexibility = 0.5, degree = 0.1) followed by noise removal (std. dev. = 2) followed by a Gaussian smooth (filter width = 31 points). Processed MS and MS/MS spectra were centroided and exported as an ASCII file (m/z vs. absolute intensity). MS and MS/MS ASCII files were uploaded to their respective databases in the USDA software.

Results

E. coli O157:H7 strain EDL-933

[0223] FIG. 11 shows the spectrum from MALDI-TOF-TOF-MS analysis of the extracted cell lysate of *E. coli* O157:H7 strain EDL-933 (genomically sequenced by Perna and coworkers in 2001, Perna, N.T. et al. (2001) *Nature*. 409(6819):529-533). The MS data was collected in linear mode with HCCA MALDI matrix.

[0224] FIG. 12 shows the MS/MS spectrum, as viewed in the instrument software, of the protein biomarker ion a m/z 7705.6 shown in FIG. 11. Prominent fragment ions are identified by their m/z and ion type/number and amino acid residues adjacent to the site of polypeptide backbone cleavage for the protein amino acid sequence of the top identification shown in Table 4: putative uncharacterized protein yahO of the pathogenic *E. coli* O157:H7 strain EDL-933. As can be seen, many of the fragment ions are the result of polypeptide cleavage adjacent to aspartic (D) and/or glutamic acid (E) residues. The MS/MS data in FIG. 12 was centroided, exported from the instrument software as an ASCII formatted file (m/z vs. absolute intensity) and uploaded to the USDA software for analysis.

[0225] Table 4 (below) shows the top five identifications of the protein biomarker of *E. coli* O157:H7 strain EDL-933 at m/z 7705.6 (shown in FIG. 11) and analyzed by MS/MS using MALDI-TOF-TOF mass spectrometry and compared to all *in silico* fragment ions of bacterial protein sequences having the same MW as the biomarker within a tolerance of ± 5 Da. The top identification of both the USDA peak matching algorithm β and the p-value calculation correctly identified the source microorganism. The protein biomarker was identified as the putative uncharacterized protein yahO. The scores of the 1st and 2nd identification are significantly different compared to the other runner-up identifications. This is reflective of the very high quality of this MS/MS spectrum. In addition, it was possible to distinguish between the pathogenic *E. coli* O157:H7 strain EDL-933 and the non-pathogenic *E. coli* strain K-12 with this protein biomarker. As can be seen, the USDA peak

matching algorithm β was approximately three times faster than the p-value calculation in terms of computation speed.

Table 4					
In Silico ID	Identifier	Sample Name	Protein	USDA Score- β	P-value
26947	>tr Q8X699 Q8X699_ECO57	<i>Escherichia coli</i> O157:H7 (strain EDL-933)	Putative uncharacterized protein yahO (Putative uncharacterized protein ECs0383) PTM-21SigPep 7707.62	66.30	4.3E-18
26281	>sp P75694 YAH0_ECOLI	<i>Escherichia coli</i> (strain K-12)	UPF0379 protein yahO PTM-21SigPep 7706.64	63.04	9.0E-16
25925	tr B3G283 B3G283_PSEAE	<i>Pseudomonas aeruginosa</i>	Putative uncharacterized protein 7709.01	44.57	4.8E-5
25880	>tr B1SUG8 B1SUG8_9BACI	<i>Geobacillus</i> WCH70	Putative uncharacterized protein 7707.91	38.04	4.9E-4
26056	>tr Q20ZY6 Q20ZY6_RHOPB	<i>Rhodopseudomonas palustris</i> (strain BisB18)	Rubredoxin type Fe(Cys)4 protein 7695.44	38.04	1.3E-3
<u>MS/MS to in silico comparison parameters</u> Intensity threshold: 2% Number of MS/MS peaks with intensity $\geq$ 2%: 92. <i>m/z</i> range for comparison: 0-14,000 Th. Fragment ion tolerance: 2.5 Th. Protein MW 7705 $\pm$ 10 Da. Number of bacterial proteins 1320. All <i>in silico</i> fragment ions compared. "PTM N-Met" indicates that the <i>in silico</i> protein sequence was modified to remove the N-terminal methionine. "PTM #SigPep" indicates that the <i>in silico</i> protein sequence was modified to remove a signal peptide. <u>Algorithm computation times</u> USDA peak matching algorithm beta: 35.7 seconds. P-value: 101.0 seconds.					

[0226] FIG. 13 shows the primary amino acid sequence of the putative uncharacterized protein yahO of the pathogenic *E. coli* O157:H7 strain EDL-933 strain (SEQ ID NO:1) and the UPF0379 protein yahO of the non-pathogenic *E. coli* strain K-12 strain (SEQ ID NO:2). Both proteins have a 21 residue N-terminal signal peptide (boxed). The two proteins are different by two amino acid substitutions. A F $\leftrightarrow$ L substitution at residue in the signal peptide and a D $\leftrightarrow$ N substitution that results in a protein MW difference of 1 Da in the mature protein. Such slight differences in protein biomarker MW are difficult to detect by MALDI-TOF-MS, however sequence-specific fragmentation by MS/MS allows these two proteins (and their source microorganisms) to be differentiated.

[0227] Identification of the putative uncharacterized protein yahO of *E. coli* O157:H7 strain EDL-933 was facilitated due to the high sequence homology to UPF0379 protein yahO of *E. coli* strain K-12. The putative uncharacterized protein yahO of *E. coli* O157:H7 strain EDL-933 currently in ExPASy databases (>tr|Q8X699|Q8X699_ECO57) is displayed as the

unprocessed protein sequence with no indication that this protein might possess a signal peptide. In consequence, this protein was not retrieved during the initial TagIdent protein MW search. However, the search did retrieve UPF0379 protein yahO of *E. coli* K-12 (>sp|P75694|YAH0_ECOLI) which possessed a 21-residue N-terminal signal peptide. The signal peptide-containing yahO sequence of *E. coli* K-12 (and other signal-peptide containing proteins) was identified from its larger file size. The signal peptides were removed from the *in silico* sequences, processed and uploaded by methods known in the art. Initially, the top identification of MS/MS of the *m/z* 7705.6 biomarker ion (**FIG. 12**) of *E. coli* O157:H7 strain EDL-933 was UPF0379 protein yahO of *E. coli* strain K-12 with a 21-residue N-terminal signal peptide and no other "runner up" identifications of *E. coli* among the top 5 identifications. Working on the assumption that the Expasy database may not indicate that yahO protein of *E. coli* O157:H7 strain EDL-933 possessed a signal peptide, a targeted search was conducted for the yahO protein and *E. coli* O157:H7 strain EDL-933 which resulted in the identification of an unprocessed sequence of putative uncharacterized protein yahO *E. coli* O157:H7 strain EDL-933. This sequence was processed to remove twenty-one residues from the N-terminus. The FastA file was then processed by methods known in the art and uploaded to the USDA software that in this instance utilized the USDA peak matching algorithm β . The MS/MS data was re-analyzed and the results are displayed in Table 4. Further inspection of Expasy database revealed nine other strains of *E. coli* O157:H7 whose yahO unprocessed sequence is identical to the yahO sequence of strain EDL-933. They are: EC4196, EC4113, EC4076, EC4401, EC4486, EC4501, EC869, EC508 and EC4115/EHEC. Thus, it would not be possible to distinguish between different strains of *E. coli* O157:H7 using only the yahO protein biomarker.

[0228] Although there is only a one residue difference (N $\leftrightarrow$ D) between the amino acid sequences of the *mature* yahO protein of *E. coli* strain K-12 and *E. coli* O157:H7 strain EDL-933 (SEQ ID NO:1) (**FIG. 13**), this difference involves an aspartic acid residue substitution. Aspartic acid residues facilitate fragmentation of singly protonated protein ions. Without being bound by theory it is believed that fragmentation is facilitated because the short acidic side chain allows transfer of its acidic proton to the polypeptide backbone thereby increasing the fragmentation efficiency of the polypeptide backbone. The other consequence of the N $\leftrightarrow$ D substitution is that the difference in mass of the mature yahO proteins is only 1 Da. Such slight differences in protein biomarker MW are nearly impossible to detect by MALDI-TOF-MS, however sequence-specific fragmentation by MS/MS allows these two yahO

proteins (and their source microorganisms) to be differentiated.

[0229] It is understood that the examples and embodiments described herein are for illustrative purposes only and that various modifications or changes in light thereof will be suggested to persons skilled in the art and are to be included within the spirit and purview of this application and scope of the appended claims.

Computer Program Listing Appendix

```

<?xml version="1.0" encoding="UTF-8" ?>
<Form version="1.3" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <Properties>
    <Property name="defaultCloseOperation" type="int" value="3"/>
    <Property name="title" type="java.lang.String" value="Insilico Data Uploader"/>
  </Properties>
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
  </SyntheticProperties>
  <AuxValues>
    <AuxValue name="FormSettings_generateMnemonicsCode" type="java.lang.Boolean"
value="false"/>
    <AuxValue name="FormSettings_layoutCodeTarget" type="java.lang.Integer"
value="1"/>
    <AuxValue name="FormSettings_listenerGenerationStyle" type="java.lang.Integer"
value="0"/>
    <AuxValue name="FormSettings_variablesLocal" type="java.lang.Boolean"
value="false"/>
    <AuxValue name="FormSettings_variablesModifier" type="java.lang.Integer"
value="2"/>
  </AuxValues>

  <Layout>
    <DimensionLayout dim="0">
      <Group type="103" groupAlignment="0" attributes="0">
        <Group type="102" alignment="0" attributes="0">
          <EmptySpace max="-2" attributes="0"/>
          <Group type="103" groupAlignment="0" attributes="0">
            <Component id="jScrollPane1" alignment="1" pref="518" max="32767"
attributes="0"/>
            <Group type="102" alignment="0" attributes="0">
              <Component id="jLabel1" min="-2" max="-2" attributes="0"/>
              <EmptySpace max="-2" attributes="0"/>
              <Component id="txt_Dir" pref="428" max="32767"
attributes="0"/>
            </Group>
            <Group type="102" alignment="1" attributes="0">
              <Component id="pb_upload" pref="421" max="32767"
attributes="0"/>
              <EmptySpace max="-2" attributes="0"/>
              <Component id="btn_Upload" min="-2" max="-2" attributes="0"/>
            </Group>
          </Group>
          <EmptySpace max="-2" attributes="0"/>
        </Group>
      </Group>
    </DimensionLayout>
    <DimensionLayout dim="1">
      <Group type="103" groupAlignment="0" attributes="0">
        <Group type="102" alignment="0" attributes="0">
          <EmptySpace max="-2" attributes="0"/>
          <Group type="103" groupAlignment="3" attributes="0">
            <Component id="jLabel1" alignment="3" min="-2" max="-2"
attributes="0"/>
            <Component id="txt_Dir" alignment="3" min="-2" max="-2"
attributes="0"/>
          </Group>
          <EmptySpace max="-2" attributes="0"/>
          <Group type="103" groupAlignment="0" max="-2" attributes="0">
            <Component id="pb_upload" max="32767" attributes="1"/>
            <Component id="btn_Upload" alignment="0" max="32767"
attributes="1"/>
          </Group>
          <EmptySpace min="-2" pref="12" max="-2" attributes="0"/>
        </Group>
      </Group>
    </DimensionLayout>
  </Layout>
</Form>

```



```

        <Component id="jScrollPane1" pref="251" max="32767" attributes="0"/>
        <EmptySpace max="-2" attributes="0"/>
    </Group>
</Group>
</DimensionLayout>
</Layout>
<SubComponents>
    <Component class="javax.swing.JLabel" name="jLabel1">
        <Properties>
            <Property name="text" type="java.lang.String" value="Enter a Directory:"/>
        </Properties>
    </Component>
    <Component class="javax.swing.JTextField" name="txt_Dir">
        <Properties>
            <Property name="text" type="java.lang.String" value="C:\"/>
        </Properties>
    </Component>
    <Container class="javax.swing.JScrollPane" name="jScrollPane1">
        <AuxValues>
            <AuxValue name="autoScrollPane" type="java.lang.Boolean" value="true"/>
        </AuxValues>

        <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayout"/>
        <SubComponents>
            <Component class="javax.swing.JTextArea" name="txt_Status">
                <Properties>
                    <Property name="columns" type="int" value="20"/>
                    <Property name="rows" type="int" value="5"/>
                </Properties>
            </Component>
        </SubComponents>
    </Container>
    <Component class="javax.swing.JButton" name="btn_Upload">
        <Properties>
            <Property name="text" type="java.lang.String" value="Upload Files"/>
        </Properties>
        <Events>
            <EventHandler event="mouseClicked" listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent" handler="btn_UploadMouseClicked"/>
        </Events>
    </Component>
    <Component class="javax.swing.JProgressBar" name="pb_upload">
    </Component>
</SubComponents>
</Form>

```

```

/*
 * InSilicoUpload.java
 *
 * Created on June 10, 2008, 7:37 AM
 */

/**
 *
 * @author Brandon Garbus
 */

import com.sun.jmx.snmp.tasks.Task;
import javax.swing.*;
import com.mysql.jdbc.Statement;
import java.sql.*;
import java.io.File;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.*;

public class InSilicoUpload extends JFrame {

    public Connection myConn;
    public java.sql.Statement stmt;
    private Task runCompare;

    /** Creates new form InSilicoUpload */
    public InSilicoUpload() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code
">//GEN-BEGIN: initComponents
    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        txt_Dir = new javax.swing.JTextField();
        jScrollPane1 = new javax.swing.JScrollPane();
        txt_Status = new javax.swing.JTextArea();
        btn_Upload = new javax.swing.JButton();
        pb_upload = new javax.swing.JProgressBar();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Insilico Data Uploader");
        jLabel1.setText("Enter a Directory:");

        txt_Dir.setText("C:\\");

        txt_Status.setColumns(20);
        txt_Status.setRows(5);
        jScrollPane1.setViewportView(txt_Status);

        btn_Upload.setText("Upload Files");
        btn_Upload.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                btn_UploadMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane());

```

```

        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jScrollPane1,
                        javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
                        518, Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel1)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(txt_Dir,
                            javax.swing.GroupLayout.DEFAULT_SIZE, 428, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(pb_upload,
                            javax.swing.GroupLayout.DEFAULT_SIZE, 421, Short.MAX_VALUE)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(btn_upload)))
                .addContainerGap()
        );
        layout.setVerticalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jLabel1)
                        .addComponent(txt_Dir,
                            javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
                        false)
                        .addComponent(pb_upload,
                            javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                            Short.MAX_VALUE)
                        .addComponent(btn_upload,
                            javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                            Short.MAX_VALUE))
                    .addGap(12, 12, 12)
                    .addComponent(jScrollPane1,
                        javax.swing.GroupLayout.DEFAULT_SIZE, 251, Short.MAX_VALUE)
                    .addContainerGap())
                .addContainerGap()
        );
        pack();
    } // </editor-fold> // GEN-END: initComponents

    private void btn_uploadMouseClicked(java.awt.event.MouseEvent evt)
    { // GEN-FIRST:event_btn_uploadMouseClicked
        btn_upload.setEnabled(false);
        runCompare = new Task();
        runCompare.execute();
    } // GEN-LAST:event_btn_uploadMouseClicked

    /**
     * @param args the command line arguments

```

```

    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new InSilicoUpload().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btn_Upload;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JProgressBar pb_upload;
    private javax.swing.JTextField txt_Dir;
    private javax.swing.JTextArea txt_Status;
    // End of variables declaration//GEN-END:variables

    class Task extends SwingWorker<Void, Void> {

        @Override
        public Void doInBackground() {

            //First get the list of files
            File mfile = new File(txt_Dir.getText().toLowerCase());
            File[] mfiles;
            mfiles = mfile.listFiles();

            if (mfiles.length == 0) {
                txt_Status.setText("There are no files in this directory. Please
choose a valid directory.");
                return null;
            }

            //Connect to the database
            dbConnect();

            pb_upload.setStringPainted(true);
            pb_upload.setMaximum(mfiles.length);
            pb_upload.setMinimum(0);
            pb_upload.setValue(0);

            txt_Status.setText(txt_Status.getText() + "\nUploading Files:");
            for(int i = 0; i < mfiles.length; i++) {

                try {

                    txt_Status.setText(txt_Status.getText() + "\n" +
mfiles[i].toString());
                    //Now we want to open the file
                    BufferedReader in = new BufferedReader( new
FileReader(mfiles[i]));
                    String ln;
                    double mw;
                    String aa;

                    //Read the first line
                    ln = in.readLine();
                    aa = in.readLine().trim();
                    mw = Double.valueOf(in.readLine());

                    //Parse out the first line with ACCSSION Number, Protein,
Strain, Notes
                    String[] idat = ln.split(" ");

```

```

//Parse the strain data
//idat[1] contains the strain. This removes the sp. Identifier
from the strain.
//It means species and is redundant and annoying that people put
that in the FASTA files
idat[1] = idat[1].replaceAll("sp.", "");

//Parse the strain data. Seperate everythign by a space. Nothing
too complicated
String[] strain = idat[1].split(" ");

//Seperate the protein and accession number
idat = idat[0].split(" ");

String prot = "";

for (int k = 1; k < idat.length; k++) {
    prot += idat[k] + " ";
}

prot = prot.trim();

//Array to store the mass list from the file
ArrayList mass = new ArrayList();

int slen = strain.length;

//If the strain is too long add extra stuff to subtype/other
if (slen > 4) {
    for(int z = 4; z < strain.length; z++) {
        strain[3] = strain[3] + " " + strain[z];
    }
    slen = 4;
}

String genus = "", species = "", subspecies = "", subtype = "";

switch(slen) {
    case 4:
        subtype = strain[3];
    case 3:
        subspecies = strain[2];
    case 2:
        species = strain[1];
    case 1:
        genus = strain[0];
}

int igenus, ispecies, isubspecies, isubtype;
//Check and see if a strain exists
igenus = insertNewGenus(genus);
ispecies = insertNewSpecies(species);
isubspecies = insertNewSubspecies(subspecies);
isubtype = insertNewSubtype(subtype);

String slctStr = "SELECT strain_id FROM strain WHERE genus_id="
+ igenus + " AND species_id=" + ispecies + " AND subspecies_id=" + isubspecies + "
AND subtype_id=" + isubtype;
int strainid, msmsid;

ResultSet myRS = getRS(slctStr);

if (myRS.next()) {
    strainid = myRS.getInt("strain_id");
} else {

```

```

        //the strain does not exist, make a new one and use the new
id
        myRS.close();
        //Insert the strain ID, now get it the same way as above
        String insrtStrain;
        insrtStrain = "INSERT INTO strain (genus_id, species_id,
subspecies_id, subtype_id) VALUES(" + igenus + "," + ispecies + "," + isubspecies +
", " + isubtype + ")";
        stmt.executeUpdate(insrtStrain);
        stmt.close();

        myRS = getRS(slctStr);

        myRS.next();
        strainid = myRS.getInt("strain_id");
    }
    myRS.close();

    String insStr = "INSERT INTO in_silico(date, identification,
strain_id, average_mw, protein, notes, aa_seq) VALUES(CURDATE(),'" + idat[0] + "','" +
+ strainid + "','" + mw + "','" + prot + "','" + 'Filename: ' + mfiles[i].getName() + "','" +
aa + "')";
    stmt.executeUpdate(insStr,
java.sql.Statement.RETURN_GENERATED_KEYS);
    myRS = stmt.getGeneratedKeys();
    myRS.next();
    msmsid = myRS.getInt(1);
    myRS.close();

    while ((ln = in.readLine()) != null) {
        //Splits the line using comma delimiter. First is m to z,
second is ion type, third is residue
        String[] mzln = ln.split(",");

        String templnStr = "INSERT INTO in_silico_data
(in_silico_id, mass,intensity, iontype, residue) VALUES(";
        templnStr += msmsid + "," + mzln[0] + ",100,'" +
mzln[1].trim() + "','" + mzln[2] + "')";
        stmt.executeUpdate(templnStr);
    }

    //Update Progress Bar
    pb_upload.setValue( i + 1 );
} catch(SQLException sqle) {
    txt_Status.setText(txt_Status.getText() + "\nAn Error
Occurred!!");
    txt_Status.setText(txt_Status.getText() + sqle.toString());
    return null;
} catch(Exception e) {
    txt_Status.setText(txt_Status.getText() + "\nAn Error
Occurred!!");
    txt_Status.setText(txt_Status.getText() + e.toString());
    return null;
}

}

return null;

```

```

    }

    @Override
    public void done() {
        dbClose();
        btn_Upload.setEnabled(true);
    }
}

public void dbConnect() {
try {
    txt_Status.setText("Connecting...");
    String url = "jdbc:mysql://10.200.38.109:3306/psm";
    //String url = "jdbc:mysql://localhost/psm";
    Class.forName("com.mysql.jdbc.Driver");

    DriverManager.setLoginTimeout(120);
    myConn = DriverManager.getConnection(url, "maldiuser", "?!tOF2");

    txt_Status.setText(txt_Status.getText() + "\nConnected Successfully!");
} catch (SQLException e) {
    return;
} catch (Exception e) {
    return;
}
}

public java.sql.ResultSet getRS(String myStmt) {
//Handles all necessary code for creating a record set
java.sql.ResultSet myRS;
myRS = null;
//Exit out if there is no connection
if (myConn == null) {return null;}

try {
    stmt = myConn.createStatement(ResultSet.FETCH_FORWARD,
ResultSet.CONCUR_UPDATABLE);
    myRS = stmt.executeQuery(myStmt);

} catch (SQLException e) {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    if (myRS!=null) {
        return myRS;
    } else {
        return null;
    }
}
}

public int dbClose() {
    txt_Status.setText(txt_Status.getText() + "\nConnection Closed.");
    if (myConn != null) {
        try {myConn.close();} catch (Exception e) {e.printStackTrace();}
    }
    return 0;
}

public int insertNewGenus(String genusname) {
try {

```

```

        //First check to see if the name is already in the db
        ResultSet myRS = getRS("SELECT * FROM genus WHERE name = '" + genusname
+ "'");

        //If it is return the id
        if (myRS.next()) {
            int id = myRS.getInt("genus_id");
            myRS.close();
            stmt.close();
            return id;
        }
        myRS.close();
        stmt.close();
        String instStmt;
        instStmt = "INSERT INTO genus (name) VALUES('" + genusname + "')";
        stmt = myConn.createStatement();
        stmt.executeUpdate(instStmt);
        ResultSet myrRS = stmt.getGeneratedKeys();
        myrRS.next();
        int id = myrRS.getInt(1);
        myrRS.close();
        stmt.close();
        return id;
    } catch (SQLException e) {
        return -1;
    } catch (Exception e) {
        return -1;
    }
}

public int insertNewSpecies(String speciesname) {
    try {
        //First check to see if the name is already in the db
        ResultSet myRS = getRS("SELECT * FROM species WHERE name = '" +
speciesname + "'");
        //If it is return the id
        if (myRS.next()) {
            int id = myRS.getInt("species_id");
            myRS.close();
            stmt.close();
            return id;
        }
        myRS.close();
        stmt.close();
        String instStmt;
        instStmt = "INSERT INTO species (name) VALUES('" + speciesname + "')";
        stmt = myConn.createStatement();
        stmt.executeUpdate(instStmt);
        ResultSet myrRS = stmt.getGeneratedKeys();
        myrRS.next();
        int id = myrRS.getInt(1);
        myrRS.close();
        stmt.close();
        return id;
    } catch (SQLException e) {
        return -1;
    } catch (Exception e) {
        return -1;
    }
}

public int insertNewSubspecies(String subspeciesname) {
    try {
        //First check to see if the name is already in the db

```



```

        ResultSet myRS = getRS("SELECT * FROM subspecies WHERE name = '" +
subspeciesname + "'");
        //If it is return the id
        if (myRS.next()) {
            int id = myRS.getInt("subspecies_id");
            myRS.close();
            stmt.close();
            return id;
        }
        myRS.close();
        stmt.close();
        String instStmt;
        instStmt = "INSERT INTO subspecies (name) VALUES('" + subspeciesname +
        "')";

        stmt = myConn.createStatement();
        stmt.executeUpdate(instStmt);
        ResultSet myrRS = stmt.getGeneratedKeys();
        myrRS.next();
        int id = myrRS.getInt(1);
        myrRS.close();
        stmt.close();
        return id;
    } catch (SQLException e) {
        return -1;
    } catch (Exception e) {
        return -1;
    }
}

public int insertNewSubtype(String othername) {
    try {
        //First check to see if the name is already in the db
        ResultSet myRS = getRS("SELECT * FROM subtype WHERE name = '" +
othername + "'");
        //If it is return the id
        if (myRS.next()) {
            int id = myRS.getInt("subtype_id");
            myRS.close();
            stmt.close();
            return id;
        }
        myRS.close();
        stmt.close();
        String instStmt;
        instStmt = "INSERT INTO subtype (name) VALUES('" + othername + "')";
        stmt = myConn.createStatement();
        stmt.executeUpdate(instStmt);
        ResultSet myrRS = stmt.getGeneratedKeys();
        myrRS.next();
        int id = myrRS.getInt(1);
        myrRS.close();
        stmt.close();
        return id;
    } catch (SQLException e) {
        return -1;
    } catch (Exception e) {
        return -1;
    }
}
}

```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- You may freely edit this file. See commented blocks below for -->
<!-- some examples of how to customize the build. -->
<!-- (If you delete it and reopen the project it will be recreated.) -->
<project name="InSilicoUpload" default="default" basedir=".">
  <description>Builds, tests, and runs the project InSilicoUpload.</description>
  <import file="nbproject/build-impl.xml"/>
  <!--
```

There exist several targets which are by default empty and which can be used for execution of your tasks. These targets are usually executed before and after some main targets. They are:

-pre-init:	called before initialization of project properties
-post-init:	called after initialization of project properties
-pre-compile:	called before javac compilation
-post-compile:	called after javac compilation
-pre-compile-single:	called before javac compilation of single file
-post-compile-single:	called after javac compilation of single file
-pre-compile-test:	called before javac compilation of JUnit tests
-post-compile-test:	called after javac compilation of JUnit tests
-pre-compile-test-single:	called before javac compilation of single JUnit
test	
-post-compile-test-single:	called after javac compilation of single JUnit
test	
-pre-jar:	called before JAR building
-post-jar:	called after JAR building
-post-clean:	called after cleaning build products

(Targets beginning with '-' are not intended to be called on their own.)

Example of inserting an obfuscator after compilation could look like this:

```
<target name="-post-compile">
  <obfuscate>
    <fileset dir="${build.classes.dir}"/>
  </obfuscate>
</target>
```

For list of available properties check the imported nbproject/build-impl.xml file.

Another way to customize the build is by overriding existing main targets. The targets of interest are:

-init-macrodef-javac:	defines macro for javac compilation
-init-macrodef-junit:	defines macro for junit execution
-init-macrodef-debug:	defines macro for class debugging
-init-macrodef-java:	defines macro for class execution
-do-jar-with-manifest:	JAR building (if you are using a manifest)
-do-jar-without-manifest:	JAR building (if you are not using a manifest)
run:	execution of project
-javadoc-build:	Javadoc generation
test-report:	JUnit report generation

An example of overriding the target for project execution could look like this:

```
<target name="run" depends="InSilicoUpload-impl.jar">
  <exec dir="bin" executable="launcher.exe">
    <arg file="${dist.jar}"/>
  </exec>
</target>
```

Notice that the overridden target depends on the jar target and not only on

the compile target as the regular run target does. Again, for a list of available properties which you can use, check the target you are overriding in the nbproject/build-impl.xml file.

-->
</project>

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%--
The taglib directive below imports the JSTL library. If you uncomment it,
you must also add the JSTL library to the project. The Add Library... action
on Libraries node in Projects view can be used to add the JSTL 1.1 library.
--%>
<%--
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <jsp:useBean id="nIn" class="psm.gene.nucleotideinsert" scope="session"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
    <%
    dbBean.dbConnect("genelist");
    if (nIn.insertNucleotide(request, request.getParameter("filetype"), dbBean) ==
0) {
      out.write("Insert successfull!");
    } else {
      out.write("Insert failed, please check and make sure the file extension
matches what you selected.<br>You may also want to make sure that the locus has
already been inserted into the database.")
    }
    dbBean.dbClose();
    %>

  </body>
</html>

```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>

    <h1>JSP Page</h1>

    <form method=post action="insertnucleotidedataconfirm.jsp"
    enctype="MULTIPART/FORM-DATA">
      <table>
        <tr>
          <td>
            <input type=file name="aantfile">
          </td>
        </tr>
        <tr>
          <td>
            <input type="radio" name="filetype" value="AA">AA
            <input type="radio" name="filetype" value="NT">NT
          </td>
        </tr>
        <tr>
          <td>
            <input type="submit" value="Send">
          </td>
        </tr>
      </table>
    </form>

  </body>
</html>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%--
The taglib directive below imports the JSTL library. If you uncomment it,
you must also add the JSTL library to the project. The Add Library... action
on Libraries node in Projects view can be used to add the JSTL 1.1 library.
--%>
<%--
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>

    <form>
    <table>
      <tr>
        <td>
          Locus:
        </td>
        <td>
          <input type="text" name="locus">
        </td>
      </tr>
      <tr>
        <td>
          Strain:
        </td>
        <td>
          <input type="text" name="strain">
        </td>
      </tr>
      <tr>
        <td>
          Genus:
        </td>
        <td>
          <input type="text" name="genus">
        </td>
      </tr>
      <tr>
        <td>
          Species:
        </td>
        <td>
          <input type="text" name="species">
        </td>
      </tr>
      <tr>
        <td>
          Location:
        </td>
        <td>
          <input type="text" name="location">
        </td>
      </tr>
      <tr>
        <td>

```

```
        Location 2:
      </td>
      <td>
        <input type="text" name="location2">
      </td>
    </tr>
    <tr>
      <td>
        Product:
      </td>
      <td>
        <input type="text" name="product">
      </td>
    </tr>
    <tr>
      <td>
        Notes:
      </td>
      <td>
        <input type="text" name="notes">
      </td>
    </tr>
  </table>
</form>

</body>
</html>
```

[illegible]

76

[illegible]

```
|  |  |
| --- | --- |
| |  | | --- | | <a href="newspecies.jsp">Insert Subspecies</a></td> | |
| |  | | --- | | <a href="newsbtype.jsp">Insert Subtype</a></td> | |
| |  | | --- | | <a href="insertmaldidatab.jsp">Insert MALDI Data</a></td> | |
| |  | | --- | | <a href="insilicoadd.jsp">Insert In Silico Data</a></td> | |
| |  | | --- | | <a href="msmsadd.jsp">Insert MS/MS Data</a></td> | |
| |  | | --- | | <a href="importcsv.jsp">Import New Strains</a></td> | |
| |  | | --- | | <a href="importnotestabdel.jsp">Import Notes</a></td> | |
| |  | | --- | | <a href="downloadstrainfile.jsp?rmonly=yes">Download Strains</a></td> | |

```

<!--}adminonly-->

```





```

<tr style="font-family: Century Gothic;">

<td class="nav" align="center" valign="top">

</td>

</tr>

<tr>

<td align="left" valign="top"></td>

</tr>

```

        <tr>
            <td align="center" valign="top">
                <table
style="width: 90%; text-align: left; margin-left: auto; margin-right: 0px;"
border="0" cellpadding="0" cellspacing="0">
                    <tbody>
<!--adminonly{--><tr>
style="font-weight: bold; font-family: Century Gothic;"
align="left">
                        <td style="vertical-align: top;"
class="nav"><a href="importtabgenelist.jsp">Insert Genelist</a></td>
                    </tr>
<!--}adminonly--><tr>
style="font-weight: bold; font-family: Century Gothic;"
align="left">
                        <td style="vertical-align: top;"
class="nav"><a href="searchgenelist.jsp">Search Loci</a></td>
                    </tr>
<!--adminonly{--><tr>
style="font-weight: bold; font-family: Century Gothic;"
align="left">
                        <td style="vertical-align: top;"
class="nav"><a href="insertnucleotidedata.jsp">Insert AA/NT</a></td>
                    </tr>
<!--}adminonly-->
                    </tbody>
                </table>
            </td>
        </tr>
<!--adminonly{--> <tr>
style="font-family: Century Gothic; font-weight: bold;"
            <td style="text-align: center;" valign="top"><a
href="insertnucleotidedata.jsp"><span
style="color: rgb(204, 255, 255);"></span><br>
            </a></td>
        </tr>
<!--}adminonly-->
        </tbody>
    </table>
    </td>
    <td style="text-align: center; vertical-align: top;">
        <table
style="background-color: rgb(255, 255, 255); width: 90%; text-align: left;
margin-left: auto; margin-right: auto;"
bordercolorlight="#000000" bordercolordark="#000000"
border="2" bordercolor="#000000" cellpadding="0"
cellspacing="0">
        <tbody>
            <tr>
                <td style="vertical-align: top; text-align: left;">&nbsp;
$MIDDLE$ </td>
            </tr>
        </tbody>
        </table>
    </td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
    <td colspan="2" rowspan="1"
style="vertical-align: top; text-align: left; width: 100px;">
        <table style="text-align: left; width: 100%;" border="0"
cellpadding="0" cellspacing="0">

```

```
<tbody>
  <tr>
    <td
style="vertical-align: top; text-align: left; width: 200px;"></td>
    <td style="text-align: center; vertical-align: middle;"></td>
  </tr>
</tbody>
</table>
</td>
</tr>
<tr>
  <td colspan="2" rowspan="1"
style="vertical-align: top; text-align: left; width: 100px;"></td>
</tr>
</tbody>
</table>
<br>
</body>
</html>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Change Password",
"ChangePassword", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        String cpwd = dbBean.getPassword();
        String newpwd = request.getParameter("newpwd");
        if (cpwd.compareTo(request.getParameter("oldpwd")) == 0) {
            if (newpwd.compareTo(request.getParameter("retype")) == 0) {
                if (dbBean.changePassowrd(request.getParameter("newpwd"))) {
                    out.print("Password Change Successfull!");
                } else {
                    out.print ("A Database error occurred, your password may not
have been changed, if you experience trouble contact the system administrator.");
                }
            } else {
                out.print("Your passwords do not match, press back on your browser
and try again.");
            }
        } else {
            out.print("Your old password is incorrect, press back on your browser to try
again.");
        }
        dbBean.dbClose();
        out.write(pageSource.getPageFooter());
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Change Password", "Change
Password", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
    %>

        <center><h1>Change Password</h1></center><br>

        <form action="changeconfirmation.jsp">
        <table border="0" cellpadding="0" cellspacing="2" align="center">
        <tr>
            <td>Old Password: </td><td><input type="password" name="oldpwd"></td>
        </tr>
        <tr>
            <td>New Password: </td><td><input type="password" name="newpwd"></td>
        </tr>
        <tr>
            <td>Retype New Password:</td><td><input type="password"
name="retype"> </td>
        </tr>
        <tr>
            <td colspan="2" align="right"><input type="submit" value="OK" </td>
        </tr>
        </table>
        </form>

    <%
        out.write(pageSource.getPageFooter());
        dbBean.dbClose();
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>
<%@page import="java.sql.*" %>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <jsp:useBean id="frInsilico" class="psm.mformat.formatinsilico"
scope="session"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert <i>In Silico</i>
Data", "InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        frInsilico.formatMALDIFile(request);
        ResultSet myRS;
    %>

        <form action="insilicoconfirmation.jsp">
        <table align="center">
            <tr>
                <td colspan="2">
                    <center>Please check and make sure the data from the
file was pased correctly<br>and make any changes you wish before hitting
submit.<br><br></center>
                </td>
            </tr>
            <tr>
                <td>
                    Identification Number:&nbsp;
                </td>
                <td>
                    <input type="text" name="ident"
value="<%=frInsilico.getNumber()%>">
                </td>
            </tr>
            <tr>
                <td>
                    Molecular weight:&nbsp;
                </td>
                <td colspan="2">
                    <input type="text" name="mw"
value="<%=Double.toString(frInsilico.getMW()%>">
                </td>
            </tr>
            <tr>
                <td>
                    Genus:
                </td>
                <td>
                    <input type="text" name="genus"
value="<%=frInsilico.getGenus()%>">
                </td>
            </tr>
            <tr>
                <td>
                    Species:
                </td>
                <td>
                    <input type="text" name="species"
value="<%=frInsilico.getSpecies()%>">
                </td>
            </tr>
            <tr>
                <td>
                    Subspecies:
                </td>
            </tr>
        </table>
    </form>

```



```

        <td>
            <input type="text" name="subspecies"
value="<%=frInsilico.getSubspecies()%>">
        </td>
    </tr>
    <tr>
        <td>
            Subtype:
        </td>
        <td>
            <input type="text" name="subtype"
value="<%=frInsilico.getSubtype()%>">
        </td>
    </tr>
    <tr>
        <td>Protein:
        </td>
        <td><input type="text" name="protein" value="<%=if
(frInsilico.getProtein() != null) {out.write((frInsilico.getProtein()));}%>"
        </td>
    </tr>
    <tr>
        <td>
            Notes:
        </td>
        <td>
            <input type="text" name="notes"
value="<%=frInsilico.getNotes()%>">
        </td>
    </tr>
    <tr>
        <td>
            Mass to Charge:
        </td>
        <td height=100px >
            <textarea readonly><%=
frInsilico.getMaldiMZ();
i++) {
                                ArrayList mz =
                                for (int i= 0; i < mz.size();
                                out.write(mz.get(i) + "\n" );
                                }
                                %></textarea>
        </td>
    </tr>
    <tr>
        <td colspan="3" align="center">
            <br>
            <input type="submit">
        </td>
    </tr>
</table>
</form>

<%
dbBean.dbClose();
out.write(pageSource.getPageFooter());
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <jsp:useBean id="frMaldi" class="psm.mformat.formatMALDI" scope="session"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MALDI Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        frMaldi.formatMALDIFile(request);
    %>

        <form action="maldiconfirmation.jsp">
        <table align="center">
            <tr>
                <td colspan="2">
                    <center>Please check and make sure the data from the
file was pased correctly<br>and make any changes you wish before hitting
submit.<br><br></center>
                </td>
            </tr>
            <tr>
                <td>
                    Identification Number:&nbsp;
                </td>
                <td>
                    <input type="text" name="ident" size=33
value="<%=frMaldi.getNumber()%>">
                </td>
            </tr>
            <tr>
                <td>
                    Date:
                </td>
                <td>
                    <input type="text" name="date" size=33 value="<%if
(frMaldi.getMdate() != null) {out.write((frMaldi.getMdate().getYear() + 1900) + "- "
+ (Integer.toString(frMaldi.getMdate().getMonth() + 1)) + "- " +
frMaldi.getMdate().getDate());}%>">
                </td>
            </tr>
            <tr>
                <td>
                    Target Spot Number:&nbsp;
                </td>
                <td colspan="2">
                    <input type="text" name="target" size=33
value="<%=frMaldi.getTargetNum()%>">
                </td>
            </tr>
            <tr>
                <td>
                    Notes:
                </td>
                <td colspan="2">
                    <input type="text" name="notes" size=33
value="<%=frMaldi.getNotes()%>">
                </td>
            </tr>
            <tr>
                <td colspan="3">
                    Mass to Charge/Intensity:
                </td>
            </tr>
            <td colspan="3" height=100px>

```

```

                                <textarea readonly name="mz" cols=25><%
                                ArrayList mz = frMaldi.getMaldiMZ();
                                ArrayList intensity =
frMaldi.getMaldiIntensity();
                                for (int i= 0; i < mz.size(); i++) {
+intensity.get(i) + "\n" );
                                out.write(mz.get(i) + "\t\t"
                                }
                                %></textarea>
                                </td>
                                </tr>
                                <tr>
                                <td colspan="3" align=center>
                                <br>
                                <input type="submit">
                                </td>
                                </tr>
                                </table>
                                </form>
<%
dbBean.dbClose();
out.write(pageSource.getPageFooter());
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <jsp:useBean id="frMSMS" class="psm.mformat.formatMSMSData" scope="session"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MS/MS Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    frMSMS.formatMALDIFile(request);
%>
        <form action="msmsconfirmation.jsp">
        <table align="center">
            <tr>
                <td colspan="2">
                    <center>Please check and make sure the data from the
file was pased correctly<br>and make any changes you wish before hitting
submit.<br><br></center>
                </td>
            </tr>
            <tr>
                <td>
                    Identification Number:&nbsp;
                </td>
                <td>
                    <input type="text" name="ident" size="46"
value="<%=frMSMS.getNumber()%>">
                </td>
            </tr>
            <tr>
                <td>
                    Date:
                </td>
                <td>
                    <input type="text" name="date" size="46" value="<%=if
(frMSMS.getMdate() != null) {out.write((frMSMS.getMdate().getYear() + 1900) + "-" +
(Integer.toString(frMSMS.getMdate().getMonth() + 1)) + "-" +
frMSMS.getMdate().getDate());}%>">
                </td>
            </tr>
            <tr>
                <td>
                    Molecular weight:&nbsp;
                </td>
                <td colspan="2">
                    <input type="text" name="precionmass" size="46"
value="<%=Double.toString(frMSMS.getMW()%>">
                </td>
            </tr>
            <tr>
                <td>Protein:
                </td>
                <td><input type="text" name="protein" size="46"
value="<%=if (frMSMS.getProtein() != null) {out.write((frMSMS.getProtein()));}%>"
                </td>
            </tr>
            <tr>
                <td>
                    Notes:
                </td>
                <td>
                    <input type="text" name="notes" size="46"
value="<%=frMSMS.getNotes()%>">
                </td>
            </tr>
        </table>
        </form>
    
```

```

        </td>
    </tr>
    <tr>
        <td>
            Mass to Charge/Intensity:
        </td>
        <td height=100px >
            <textarea readonly cols="35"><%
                ArrayList mz = frMSMS.getMaldiMZ();
                ArrayList intensity = frMSMS.getMaldiIntensity();
                for (int i= 0; i < mz.size(); i++) {
                    out.write(mz.get(i) + "\t\t" +intensity.get(i) +
"\n" );
                }
            %></textarea>
        </td>
    </tr>
    <tr>
        <td colspan="3" align=center>
            <br>
            <input type="submit">
        </td>
    </tr>
</table>
</form>

<%
dbBean.dbClose();
out.write(pageSource.getPageFooter());
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>MALDI-TOF Graph</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="comparefilegraphs.jsp">
      <%
        int rangeA, rangeB, topnum;
        if (request.getParameter("rangeA") != "" &
request.getParameter("rangeA") != null) {
          rangeA = Integer.parseInt(request.getParameter("rangeA"));
        } else {
          rangeA = 4000;
        }
        if (request.getParameter("rangeB") != "" &
request.getParameter("rangeB") != null) {
          rangeB = Integer.parseInt(request.getParameter("rangeB"));
        } else {
          rangeB = 14000;
        }
        if (request.getParameter("topnum") != "" &
request.getParameter("topnum") != null) {
          topnum = Integer.parseInt(request.getParameter("topnum"));
        } else {
          topnum = 5;
        }
      %>
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &rangeA=<%=rang
eA%>&rangeB=<%=rangeB%>&topnum=<%=topnum%>&title=<%=request.getParameter("title")%>"
>
            </td>
          </tr>
          <tr>
            <td colspan="3">
              &nbsp;
            </td>
          </tr>
          <tr>
            <td colspan="3">
              &rangeA=<%=rangeA%>
&rangeB=<%=rangeB%>&topnum=<%=topnum%>">
            </td>
          </tr>
          <tr>
            <td>
              Low Value:&nbsp;<input type="text" name="rangeA" value="<%=rangeA%>">
            </td>
            <td>
              High Value:&nbsp;<input type="text" name="rangeB"
value="<%=rangeB%>">

```

```
</td>
Number of Values Labeled: <input type="text" name="topnum"
value="<%=topnum%>">
</td>
</tr>
<tr>
<td> <br>
Maldi ID 1:&nbsp;   <input type="text" name="id1" value=
<%=request.getParameter("id1")%> readonly>
</td>
<td colspan = "2"> <br>
Maldi ID 2:&nbsp;   <input type="text" name="id2" value=
<%=request.getParameter("id2")%> readonly>
</td>
</tr>
<tr>
<td>
<br>
Title:&nbsp;   &nbsp;  <input type="text" name="title"
value="<%=request.getParameter("title")%>">
</td>
</tr>
<tr>
<td colspan = 3 align = right>
<input type=submit>
</td>
</tr>
</table>
</form>
</body>
</html>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>MALDI-TOF Graph</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="comparegraphs.jsp">
      <%
        int rangeA, rangeB, topnum;
        if (request.getParameter("rangeA") != "" &&
request.getParameter("rangeA") != null) {
          rangeA = Integer.parseInt(request.getParameter("rangeA"));
        } else {
          rangeA = 4000;
        }
        if (request.getParameter("rangeB") != "" &&
request.getParameter("rangeB") != null) {
          rangeB = Integer.parseInt(request.getParameter("rangeB"));
        } else {
          rangeB = 14000;
        }
        if (request.getParameter("topnum") != "" &&
request.getParameter("topnum") != null) {
          topnum = Integer.parseInt(request.getParameter("topnum"));
        } else {
          topnum = 5;
        }
      %>
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &rangeA=<%=rangeA%>
&rangeB=<%=rangeB%>&topnum=<%=topnum%>">
            </td>
          </tr>
          <tr>
            <td colspan="3">
              &nbsp;
            </td>
          </tr>
          <tr>
            <td colspan="3">
              &rangeA=<%=rangeA%>
&rangeB=<%=rangeB%>&topnum=<%=topnum%>">
            </td>
          </tr>
          <tr>
            <td>
              Low Value:&nbsp;<input type="text" name="rangeA" value="<%=rangeA%>">
            </td>
            <td>
              High Value:&nbsp;<input type="text" name="rangeB"
value="<%=rangeB%>">
            </td>
          </tr>
        </table>
      </form>
    </body>
  </html>

```


[illegible]

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "MALDI Comparison",
    "MaldiComparisonTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    %>

    <form method="post" action="comparemaldifilesetup.jsp"
    enctype="MULTIPART/FORM-DATA">
        <table align="center">
            <tr>
                <td colspan="2">
                    <center><h1>Upload MALDI-TOF MS Data<br>for Database
comparison</h1></center>
                    This accepts a Bruker type MALDI-TOF data file for comparison of data.
                </td>
            </tr>
            <tr>
                <td>
                    Find File:
                </td>
                <td>
                    <input type="file" name="maldifile">
                </td>
            </tr>
            <tr>
                <td></td>
                <td colspan="1" align="left">
                    <input type="submit" value="Send">
                </td>
            </tr>
        </table>
    </form>
    <%
    dbBean.dbClose();
    out.write(pageSource.getPageFooter());
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>

    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
    <%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "MALDI Comparison",
    "MaldiComparisonTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    %>

    <jsp:useBean id="frMaldi" class="psm.mformat.formatMALDI" scope="session"/>
    <%
    frMaldi.formatMALDIFile(request);
    %>

    <form action="maldifilecompareresults.jsp">
    <table align="center">
        <tr>
            <td colspan="2">
                <center>Please check and make sure the data from the
file was pased correctly<br>and make any changes you wish before hitting
submit.<br><br></center>
            </td>
        </tr>
        <tr>
            <td>
                Title:
            </td>
            <td>
                <input type="text"
value="<%=frMaldi.getFileName()%>" name="title">
            </td>
        </tr>
        <tr>
            <td>
                Mass to Charge - Intensity:
            </td>
            <td height=100px>
                <textarea readonly><%
                ArrayList mz = frMaldi.getMaldiMZ();
                ArrayList intensity = frMaldi.getMaldiIntensity();
                for (int i= 0; i < mz.size(); i++) {
                    out.write(mz.get(i) + "\t\t" +intensity.get(i) +
"\n" );
                }
                %></textarea>
            </td>
        </tr>
        <tr>
            <td>
                Error Threshold (ppm):
            </td>
            <td>
                <input type="text" name="ppm" value="500">
            </td>
        </tr>
        <tr>
            <td>
                Intensity Threshold (%):
            </td>
            <td>
                <input type="text" name="intensity" value=10>
            </td>
        </tr>
    </table>
    </form>

```

```
</tr>
<tr>
    <td>
        Lower Mass/Charge Bound:
    </td>
    <td>
        <input type="text" name="lMass" value="4000">
    </td>
</tr>
<tr>
    <td>
        Upper Mass/Charge Bound:
    </td>
    <td>
        <input type="text" name="hMass" value="14000">
    </td>
</tr>
<tr>
    <td>
        Maximum Number of Results:
    </td>
    <td>
        <input type="text" name="maxresults" value="25">
    </td>
</tr>
<tr>
    <td colspan=
```

```

        </td>
        <td>
            <input type="text" name="species">
        </td>
    </tr>
    <tr>
        <td>
            Search by Subspecies (e.g. Fetus):
        </td>
        <td>
            <input type="text" name="subspecies">
        </td>
    </tr>
    <tr>
        <td>
            Search by Other Subtype:
        </td>
        <td>
            <input type="text" name="other">
        </td>
    </tr>
    <tr>
        <td>
            Search Inside Notes:
        </td>
        <td>
            <input type="text" name="notes">
        </td>
    </tr>
    <tr>
        <td>
            Search Inside History:
        </td>
        <td>
            <input type="text" name="history">
        </td>
    </tr>
    <tr>
        <td colspan=2>&nbsp;
    </td>
    </tr>
    <tr>
        <td colspan=2>&nbsp;
    </td>
    </tr>
    <tr>
        <td>
            <input type="radio" name="searchtype" value="intensity"
checked>Intensity Comparison
        </td>
        <td>
            <input type="radio" name="searchtype" value="mass">Mass only
Comparison
        </td>
    </tr>
    <tr>
        <td colspan=2 align=center><br>
        <input type="reset">&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit">
        </td>
    </tr>
    <td colspan=2>
        <center><br>Using the default mass range searches will be much
faster.<br>Editing, or changing the range from 4000 to 14000 requires
re-normalization of the data<br>and can cause search times in excess of 2

```

```
minutes.<br>Please be patient.</center>
      </td>
    </table>
  </form>
<%
dbBean.dbClose();
out.write(pageSource.getPageFooter());
%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<%@page import="java.text.*" %>
<%@page import="psm.db.box.*" %>

<style>
table.notes
{
font-size:100%;
position:absolute;
visibility:hidden;
background:white;
}
</style>

<script language="JavaScript">
function selectAll(c){
for(i=1; i < c.length; i++) c[i].checked=c[0].checked;
}
</script>

<script type="text/javascript">
function showmenu(elmnt)
{
document.getElementById(elmnt).style.visibility="visible"
}
function hidemenu(elmnt)
{
document.getElementById(elmnt).style.visibility="hidden"
}
function switchmnu(elmnt) {
if (document.getElementById(elmnt).style.visibility=="hidden") {
document.getElementById(elmnt).style.visibility="visible"
} else {
document.getElementById(elmnt).style.visibility="hidden"
}
}
</script>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
dbBean.dbConnect();
pageSource.splitTemplatePage(dbBean.getUsername(), "MALDI Comparison",
"MalDiComparisonTitle", application.getRealPath("/"), dbBean);
out.write(pageSource.getPageHeader());
dbBean.dbClose();
%>
<form name="myForm">
<table align=center cellpadding="0" cellspacing="0"
style="horizontal-align: center; text-align:center">
<tr><td colspan=9>
<table align=center cellpadding="0" cellspacing="0"
style="horizontal-align: center; text-align:center">
<tr>
<td colspan=9>
<a
href="comparemaldisampletexonly.jsp?rm=<%=request.getParameter("rm")%>&id=<%=request.getParameter("id")%>&ppm=<%=request.getParameter("ppm")%>&intensity=<%=request.getParameter("intensity")%>&lMass=<%=request.getParameter("lMass")%>&hMass=<%=request.getParameter("hMass")%>&maxresults=<%=request.getParameter("maxresults")%>&date=<%=re

```

```

quest.getParameter("date")%&andor=<%=request.getParameter("andor")%&identifier=<%=
request.getParameter("identifier")%&lowRM=<%=request.getParameter("lowRM")%&highRM
=<%=request.getParameter("highRM")%&genus=<%=request.getParameter("genus")%&specie
s=<%=request.getParameter("species")%&subspecies=<%=request.getParameter("subspecie
s")%&other=<%=request.getParameter("other")%&notes=<%=request.getParameter("notes"
)%&history=<%=request.getParameter("history")%>">Downloadable Text Format</a>
</td>
</tr>
<tr><td colspan=9>&nbsp;</td></tr>
<tr>
<td>Low Value:&nbsp;</td>
<td>
<input type="text" name="rangeA"
value="<%=request.getParameter("lMass") %>" size="5">
</td>
<td>High Value:&nbsp;</td>
<td>
<input type="text" name="rangeB"
value="<%=request.getParameter("hMass")%>" size="5">
</td>
<td>Labeled Values:&nbsp;</td>
<td colspan=2 align=left>
<input type="text" name="topnum" value="5" size="2">
</td>
</tr>
<tr>
<td colspan=9>&nbsp;</td>
</tr>
<tr align=left>
<td colspan=2>Intensity Threshold (%):&nbsp;<
<input type="text" name="intensity"
value="<%=request.getParameter("intensity")%>" size="2">
</td>
<td colspan=2>&nbsp;<Error (ppm):&nbsp;<
<input type="text" name="ppm"
value="<%=request.getParameter("ppm")%>" size="2">
</td>
<td colspan=2>Size (pixels):&nbsp;<
<input type="text" name="x" value="1200"
size="4">&nbsp;<x&nbsp;<
<input type="text" name="y" value="900" size="4">
</td>
</tr>
<tr>
<td colspan=9>&nbsp;</td>
</tr>
<tr>
<td colspan=2>&nbsp;<input type="button" value="Subtraction"
onclick="document.myForm.action='subtractiongraph.jsp';document.myForm.target='_sgra
ph';document.myForm.submit();">
</td>
<td colspan=1>
&nbsp;<input type="button" value="Mirrored"
onclick="document.myForm.action='mirroredgraphs.jsp';document.myForm.target='_mirgra
ph';document.myForm.submit();">
</td>
<td colspan=2>
&nbsp;<input type="button" value="Show Graphs"
onclick="document.myForm.action='multiplegraphdisplay.jsp';document.myForm.target='_
mgraph';document.myForm.submit();">
</td>
<td colspan=2>
&nbsp;<input type="button" value="Scatterplot"
onclick="document.myForm.action='scatterplot.jsp';document.myForm.target='_scatter';
document.myForm.submit();">

```


[illegible]

```

        database        //This deletes any temporary data that is two days old in the
                        dbBean.executeUpdate("DELETE maldi_tof, maldi_tof_data, maldirat1
FROM maldirat1, maldi_tof, sample, identifier, maldi_tof_data where ident =
'Temporary' AND sample.identifier_id = identifier.identifier_id AND
mal di_tof.sample_id = sample.sample_id AND maldi_tof.DATE <
DATE_SUB(Current_timestamp, INTERVAL 1 day) AND maldi_tof.maldi_tof_id =
mal di_tof_data.maldi_tof_id AND maldirat1.maldi_tof_id = maldi_tof.maldi_tof_id");

                        //Loop through each ID and compare it
                        sqlSel = sqlSel.replace("$cmpr$", Integer.toString(maldiTempId));

                        //If the mass range does not fall within 4 to 14 thousand then we
need to normalize the data before comparing it
                        //If it does fall within this mass range then the data is already
normalized and we can use that instead
                        double nstar = (hmass - lmass) /
((Double.parseDouble(request.getParameter("ppm"))/ Math.pow(10, 6)) * hmass * 2);
                        if (lmass != 4000 | hmass != 14000) {
                            for (int i = 0; i < mID.size(); i++) {
                                String l = "";
                                l += mID.get(i);
                                //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}

                                int cmprToId;
                                cmprToId = dbBean.dumpMALDIDataToTemp(Integer.parseInt(l),
intensity, lmass,hmass, "malditemp2");

                                String tmp = sqlSel.replace("$cmprTo$",
Integer.toString(cmprToId) );
                                ResultSet myRS = dbBean.getRS(tmp);
                                myRS.next();

                                rslts.add(new scorecontainer(l, myRS.getString("score"),
myRS.getInt("lc_k"), myRS.getInt("uc_k"), myRS.getInt("n"), nstar));
                                myRS.close();
                                dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
                            }
                        } else {
                            for (int i = 0; i < mID.size(); i++) {
                                String l = "";
                                l += mID.get(i);
                                //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}

                                String tmp = sqlSel.replace("$cmprTo$", l);

                                ResultSet myRS = dbBean.getRS(tmp);
                                myRS.next();

                                rslts.add(new scorecontainer(l, myRS.getString("score"),
myRS.getInt("lc_k"), myRS.getInt("uc_k"), myRS.getInt("n"), nstar));
                                myRS.close();
                            }
                        }
                    }
                }
                //now delete the maldiTemp data to free up the memory
                dbBean.deleteMaldiTempData(maldiTempId, "malditemp");

                mID = null;
                //Sort the array, this is a simple bubble sort algorithm
                for (int i = 0 ; i< rslts.size(); i++) {
                    for (int j=0; j < (rslts.size() - 1 - i) ; j++) {
                        scorecontainer x,y;
                        x = (scorecontainer) rslts.get(j + 1);
                        y = (scorecontainer) rslts.get(j);
                        if (x.getScore() == null | y.getScore() == null) {

```

```

bottom
returned
higher than
once the data

//This was a bad score, if it is null move it to the
//A null value for a score means no data set was
//This is probably because the intenisty threshold was
//what is present in the DB, this problem should go away
//is normalized.
if (x.getScore() == null) {
    rslts.set(j, y);
    rslts.set(j+1, x);
} else {
    rslts.set(j, x);
    rslts.set(j+1, y);
}
} else {
    int mInt1 = Integer.parseInt(x.getScore());
    int mInt2 = Integer.parseInt(y.getScore());
    if ( mInt1 > mInt2 ) {
        rslts.set(j, x);
        rslts.set(j+1, y);
    }
}
}
}

//Now we want to determine the best guess case for identification
//This works by Taking all Results that scored above 60% and finding
the most occuring strain in the set

//First lets convert all the MALDIIDS with a score above 60 to a
strain ID and store them in an arraylist
ArrayList strains = new ArrayList();

//Since rslts is already sorted from bottom to top (item 0 being
the highest score) we can do a while loop until the score is below 60
int dex=0;
String strainSQL = "SELECT s.strain_id FROM sample s, maldi_tof m,
identifier WHERE m.sample_id = s.sample_id AND ("
//This loop will develop a sql statement to grab a list of all the
strain ids
scorecontainer b = (scorecontainer) rslts.get(0);
int sc = Integer.parseInt(b.getScore());
while ( sc >= 60) {
    strainSQL += "m.maldi_tof_id = " + b.getMid() + " OR ";
    dex++;
    b = (scorecontainer) rslts.get(dex);
    sc = Integer.parseInt(b.getScore());
}

//Format the string, get rid of the last OR and add a )
strainSQL = strainSQL.substring(0, strainSQL.length() - 3) + ") AND
identifier.identifier_id = s.identifier_id AND CONCAT_WS(' ', identifier.ident,
s.number) != '" + request.getParameter("rm") + "'";

//now we want to use the SQL statement and make a list of the count
of each strain
ResultSet strainRS = dbBean.getRS(strainSQL);
if (strainRS != null) {
    while (strainRS.next()) {
        //loop through the arraylist to find the strain
        boolean found = false;
        int indx = 0;
        for (int i = 0; i < strains.size(); i++) {

```

```

        box z = (box) strains.get(i);
        if (strainRS.getString(1).compareTo(z.getO1()) == 0 ) {
            found = true;
            indx = i;
            i = strains.size();
        }
    }
    if (found) {
        box z = (box) strains.get(indx);
        z.setO2(Integer.toString(Integer.parseInt(z.getO2()) + 1
));
    } else {
        strains.add(new box(strainRS.getString(1), "1"));
    }
}
strainRS.close();
//Now we want to loop through this list, get the largest score,
and display this is the identified strain
box ab;
if (strains.size() == 0) {
    //nothing above a score of 60
    out.write("<tr><td colspan = 9
style='text-align:center;background-color: rgb(45, 190, 230);font-weight: bold;'>No
identification made</td></tr>");
} else {
    ab = (box) strains.get(0);
    String topid = ab.getO1();
    String score = ab.getO2();
    for (int i =0; i < strains.size(); i ++) {
        ab = (box) strains.get(i);
        if (Integer.parseInt(ab.getO2()) >
Integer.parseInt(score)) {
            topid = ab.getO1();
            score = ab.getO2();
        }
    }
    //Now lets get the strain
    strainRS = dbBean.getRS("SELECT CONCAT_WS(' ', g.name,
s.name, sp.name, st.name) FROM strain sn, genus g, species s, subspecies sp, subtype
st WHERE g.genus_id = sn.genus_id AND s.species_id = sn.species_id AND
sp.subspecies_id = sn.subspecies_id AND st.subtype_id = sn.subtype_id AND
sn.strain_id = " + topid);
    strainRS.next();
    out.write("<tr><td colspan = 9
style='text-align:center;background-color: rgb(45, 190, 230);font-weight:
bold;'>Identified as:<br>" + strainRS.getString(1) + "</td></tr>");
    strainRS.close();
}
} else {
    //nothing above a score of 60
    out.write("<tr><td colspan = 8
style='text-align:center;background-color: rgb(45, 190, 230);font-weight: bold;'>No
identification made</td></tr>");
}

%>
<tr>
<td colspan=8>&nbsp;   </td>
</tr>

<tr>
<td>
<input type=checkbox name="checkid"
onclick="selectAll(this.form.checkid);">
</td>

```

```
 &nbsp;MALDI ID&nbsp; | &nbsp;Date&nbsp; | &nbsp;Sample&nbsp; | &nbsp;Sample Name&nbsp; | &nbsp;Notes&nbsp; | &nbsp;Score&nbsp; | &nbsp;<i>P</i>&nbsp; | &nbsp;Compare&nbsp; || &nbsp; | | | | | | | | |


<%
DecimalFormat df = new DecimalFormat("0.0E0");
for (int i = 0; i < maxresults & i < rsIts.size() ; i++) {
    %><%
    scorecontainer x = (scorecontainer) rsIts.get(i);
    ResultSet rsItData = dbBean.getMaldiAndSampleInfo(x.getMid());
    rsItData.next();
    if (rsItData.getString(1).compareTo(request.getParameter("rm"))
== 0) {
        String notes = rsItData.getString("notes");
        String notesfull = notes;
        if (notes == null) {
            notes= "";
        } else {
            notes = "&nbsp;" + notes.substring(0,10);
        }
        String popuptbl;
        popuptbl = "<table cellpadding='0' cellspacing='0'
border='1' class='notes' id='" + x.getMid() + "' width='400'><tr><td>" + notesfull +
"</td></tr></table>";
        String popuptbl2 = "<table bordercolor=black cellpadding='0'
cellpadding='0' border='1' class='notes' id='" + x.getMid() + "'
width='250'><tr><td style='vertical-align: top; text-align: Left; background-color:
rgb(36, 147, 184);'><i>P</i> Value: " + x.getPvalue() + "<br>K: " + x.getUC_k() +
"<br>k: " + x.getLC_k() + "<br>n: " + x.getN() + "<br>n*: " + x.getNstar() + "<br>s:
" + x.getS() + "</td></tr></table>";
        out.write("<tr><td style='background-color: rgb(251,
175, 55);'><input type='checkbox' name='checkid' value='" + x.getMid() + "'></td><td
style='background-color: rgb(251, 175, 55);'><a
href='maldidatawithgraph.jsp?malidiid=" + x.getMid() + "'>" + x.getMid() +
"</a></td><td style='background-color: rgb(251, 175, 55);'><a
href='maldidatawithgraph.jsp?malidiid=" + x.getMid() + "'>" + rsItData.getString(2) +
"</a></td><td style='background-color: rgb(251, 175, 55);'><a
href='strainDetail.jsp?sampleid=" + rsItData.getString(5) + "'>" +
rsItData.getString(1) + "</a></td><td style='background-color: rgb(251, 175,

```

```

55);'><a href='strainDetail.jsp?sampleid=' + rsltData.getString(5) + '>' +
rsltData.getString(4) + "</a></td><td style='background-color: rgb(251, 175, 55);'
onmouseout='\"hidemenu('\" + x.getMid() + '\"')\"><a href='javascript:void(0);'
onclick='\"showmenu('\" + x.getMid() + '\"')\">'\" + notes + \"</a>\" + popuptbl + \"</td><td
align=center style='background-color: rgb(251, 175, 55);'>\" + x.getScore() +
\"</td><td align=left style='background-color: rgb(251, 175, 55);'><a
href='javascript:void(0);' onclick='\"switchmnu('\" + x.getMid() + \"b')\">\" +
df.format(x.getPvalue()) + \"</a>\" + popuptbl2 + \"</td><td style='background-color:
rgb(251, 175, 55);'><a href='comparegraphs.jsp?id1=\" + request.getParameter(\"id\") +
\"&id2=\" + x.getMid() + \"&rangeA=\" + lmass + \"&rangeB=\" + hmass + '\">Graphs</a>\" +
\"</td></tr>\"");

    } else {
        String notes = rsltData.getString("notes");
        String notesfull = notes;
        String popuptbl;
        popuptbl = "<table cellpadding='0' cellspacing='0'
border='1' class='notes' id='\" + x.getMid() + \"' width='400'><tr><td>\" + notesfull +
\"</td></tr></table>\";
        String popuptbl2 = "<table bordercolor=black cellpadding='0'
cellpadding='0' border='1' class='notes' id='\" + x.getMid() + \"b'
width='250'><tr><td style='vertical-align: top; text-align: Left; background-color:
rgb(36, 147, 184);'><i>P</i> Value: \" + x.getPvalue() + \"<br>K: \" + x.getUc_k() +
\"<br>k: \" + x.getLc_k() + \"<br>n: \" + x.getN() + \"<br>n*: \" + x.getNstar() + \"<br>s:
\" + x.getS() + \"</td></tr></table>\";
        if (notes == null) {
            notes = \"\";
        } else {
            if (notes.length() > 10) {
                notes = "&nbsp;\" + notes.substring(0,10);
            } else {
                notes = "&nbsp;\" + notes;
            }
        }
        out.write("<tr><td><input type='checkbox' name='checkid'
value='\" + x.getMid() + \"'></td><td><a href='maldidatawithgraph.jsp?maldiid=\" +
x.getMid() + '\">\" + x.getMid() + \"</a></td><td><a
href='maldidatawithgraph.jsp?maldiid=\" + x.getMid() + '\">\" + rsltData.getString(2) +
\"</a></td><td><a href='strainDetail.jsp?sampleid=\" + rsltData.getString(5) + '\">\" +
rsltData.getString(1) + \"</a></td><td><a href='strainDetail.jsp?sampleid=\" +
rsltData.getString(5) + '\">\" + rsltData.getString(4) + \"</a></td><td
onmouseout='\"hidemenu('\" + x.getMid() + '\"')\"><a href='javascript:void(0);'
onclick='\"showmenu('\" + x.getMid() + '\"')\">'\" + notes + \"</a>\" + popuptbl +
\"</td><td align=center>\" + x.getScore() + \"</td><td align=left><a
href='javascript:void(0);' onclick='\"switchmnu('\" + x.getMid() + \"b')\">\" +
df.format(x.getPvalue()) + \"</a>\" + popuptbl2 + \"</td><td><a
href='comparegraphs.jsp?id1=\" + request.getParameter(\"id\") + \"&id2=\" + x.getMid() +
\"&rangeA=\" + lmass + \"&rangeB=\" + hmass + '\">Graphs</a>\" + \"</td></tr>\"");
    }
    rsltData.close();
}
dbBean.dbClose();
%>
</table>
</form>
<%out.write(pageSource.getPageFooter()); %>

```

```

<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<%@page import="psm.db.box.*" %>
<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>

<%
int intensity, lmass, hmass, ppm, maxresults;
boolean massSearch = false;
if (request.getParameter("searchtype").compareTo("mass") == 0) {
    massSearch = true;
}
if (request.getParameter("intensity")!="") {
    intensity = Integer.parseInt(request.getParameter("intensity"));
} else {
    intensity = 10;
}
if (request.getParameter("lMass") != "") {
    lmass = Integer.parseInt(request.getParameter("lMass"));
} else {
    lmass = 4000;
}
if (request.getParameter("hMass") != "") {
    hmass = Integer.parseInt(request.getParameter("hMass"));
} else {
    hmass = 14000;
}
if (request.getParameter("ppm") != "") {
    ppm = Integer.parseInt(request.getParameter("ppm"));
} else {
    ppm=500;
}
if (request.getParameter("maxresults") != "") {
    maxresults = Integer.parseInt(request.getParameter("maxresults"));
} else {
    maxresults = 10;
}

//Get the sql statement
String sqlSel = "";
if (massSearch) {
    sqlSel = dbBean.buildCompareStringForTempMassOnly(intensity, lmass,hmass, ppm);
} else {
    sqlSel = dbBean.buildCompareStringForTemp(intensity, lmass,hmass, ppm);
}
//Get the array maldi ID list

ArrayList mID = dbBean.getMALDILISTwithSQLSel(request.getParameter("date"),
request.getParameter("sqlone"));
ArrayList rsIts = new ArrayList();
//connect to the database
dbBean.dbConnect();

//To make the search faster we are going to dump all the data into the malditemp
table
//this table is stored in memory which makes it a lot faster and it also normalizes
teh data
int maldiTempId =
dbBean.dumpMALDIDataToTemp(Integer.parseInt(request.getParameter("id")), intensity,
lmass, hmass, "malditemp");

```

```

dbBean.dbClose();
dbBean.dbConnect();

//Loop through each ID and compare it
sqlSel = sqlSel.replace("$cmpr$", Integer.toString(malDiTempId));
%><%
//If the mass range does not fall within 4 to 14 thousand then we need to normalize
the data before comparing it
//If it does fall within this mass range then the data is already normalized and we
can use that instead
if (lmass != 4000 | hmass != 14000) {
    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) { continue;}
        int cmprToId;
        cmprToId = dbBean.dumpMALDIDataToTemp(Integer.parseInt(l), intensity,
lmass, hmass, "malDiTemp2");

        String tmp = sqlSel.replace("$cmprTo$", Integer.toString(cmprToId) );
        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rsIts.add(new box(l, myRS.getString("score")));
        myRS.close();
        dbBean.deleteMalDiTempData(cmprToId, "malDiTemp2");
    }
} else {
    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) { continue;}
        String tmp = sqlSel.replace("$cmprTo$", l);

        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rsIts.add(new box(l, myRS.getString("score")));
        myRS.close();
    }
}
%><%
dbBean.dbClose();

//Now we need to take our results, and count the occurrences of scores 0-100
int[] scoreCount = new int[101];

for(int a=0;a < rsIts.size(); a++) {
    box b = (box) rsIts.get(a);
    scoreCount[Integer.parseInt(b.getO2())]++;
}
//now we want to find the most frequent occurring score.
int maxCount=0;
for (int i=1;i<=100;i++) {
    if (scoreCount[i] > maxCount) {
        maxCount = scoreCount[i];
    }
}
%><%
//Define vars for the size
int height = 500, width = 600;
//Start Drawing the graph

```



```

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
Graphics2D biContext = bImg.createGraphics();

biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,width, height);

//draw the x axis
biContext.setColor(Color.BLACK);
biContext.drawLine(50, height - 50, width - 50, height - 50);

//draw y axis
biContext.drawLine(50, height - 50, 50, 50);

//Draw the iteration marks on the x axis and labels
int xIteration = (width-100)/100;

for (int i = 0; i <= 100; i++) {
    if (i % 10 == 0) {
        biContext.drawLine(50 + ((width - 100)/100) * i, height - 45, 50 + ((width - 100)/100) * i, height - 55);
        biContext.drawString(Integer.toString(i), 50 + ((width - 100)/100) * i - 2, height - 30);
    }
}

//Draw the y axis iteration marks and labels

//First we need to take maxCount and get a usable number this will give us a number
slightly bigger than the maximum count number but divisible by ten
int hiNum = ((maxCount / 10) + 1) * 10;
float yIter = (float) (height - 100.0) / (float) hiNum;

for (int i=0; i<=hiNum;i++) {
    if ( i % (hiNum/ 10) == 0) {
        float yd = (float) height - (float) 50.0 - (float) i * yIter;
        int y = Math.round(yd);
        biContext.drawLine(45, y, 55,y);
        biContext.drawString(Integer.toString(i), 15, y + 4);
    }
}

//now draw the points on the screen
biContext.setColor(Color.RED);

for (int i=0;i<=100;i++) {
    if (scoreCount[i] != 0) {
        float yd = (float) height - (float) 50.0 - (float) scoreCount[i] * yIter;
        int y = Math.round(yd);
        biContext.drawOval(50 + i*xIteration, y , 2,2);
        biContext.fillOval(50 + i*xIteration, y , 2,2);
    }
}

//Now Draw the second plots
mID = dbBean.getMALDILISTwithSQLSel(request.getParameter("date"),
request.getParameter("sqltwo"));
rsIts = new ArrayList();
//connect to the database
dbBean.dbConnect();

//To make the search faster we are going to dump all the data into the malditemp
table
//this table is stored in memory which makes it a lot faster and it also normalizes

```

teh data

```
dbBean.dbClose();
dbBean.dbConnect();
```

```
//Loop through each ID and compare it
sqlSel = sqlSel.replace("$cmpr$", Integer.toString(maldiTempId));
%><%
//If the mass range does not fall within 4 to 14 thousand then we need to normalize
the data before comparing it
//If it does fall within this mass range then the data is already normalized and we
can use that instead
if (lmass != 4000 | hmass != 14000) {
    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) { continue;}
        int cmprToId;
        cmprToId = dbBean.dumpMALDIDataToTemp(Integer.parseInt(l), intensity,
lmass, hmass, "malditemp2");

        String tmp = sqlSel.replace("$cmprTo$", Integer.toString(cmprToId) );
        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rsIts.add(new box(l, myRS.getString("score")));
        myRS.close();
        dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
    }
} else {
    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) { continue;}
        String tmp = sqlSel.replace("$cmprTo$", l);

        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rsIts.add(new box(l, myRS.getString("score")));
        myRS.close();
    }
}
//now delete the maldiTemp data to free up the memory
dbBean.deleteMaldiTempData(maldiTempId, "malditemp");
mID = null;
%><%
dbBean.dbClose();

//Now we need to take our results, and count the occurrences of scores 0-100
int[] scoreCount2 = new int[101];

for(int a=0;a < rsIts.size(); a++) {
    box b = (box) rsIts.get(a);
    scoreCount2[Integer.parseInt(b.getO2())]++;
}
//now we want to find the most frequent occurring score.
maxCount=0;
for (int i=1;i<=100;i++) {
    if (scoreCount2[i] > maxCount) {
        maxCount = scoreCount[i];
    }
}
}
```

```

biContext.setColor(Color.BLUE);
for (int i=0;i<=100;i++) {
    if (scoreCount2[i] != 0) {
        float yd = (float) height - (float) 50.0 - (float) scoreCount2[i] * yIter;
        int y = Math.round(yd);
        biContext.drawOval(50 + i*xIteration, y , 2,2);
        biContext.fillOval(50 + i*xIteration, y , 2,2);
    }
}

//Draw the labels
biContext.setColor(Color.BLACK);
biContext.drawString("Percent Similarity (Score)", width/2 - 30, height-5);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 2, 280 - (fm.getHeight()/2));
biContext.drawString("Number of Results", 2,280);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();

%>

```

```

<%@page contentType="text"%>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<%@page import="psm.db.box.*" %>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
    <%
        int intensity, lmass, hmass, ppm, maxresults;
        boolean massSearch = false;
        if (request.getParameter("searchtype").compareTo("mass") == 0) {
            massSearch = true;
        }
        if (request.getParameter("intensity") != "") {
            intensity = Integer.parseInt(request.getParameter("intensity"));
        } else {
            intensity = 10;
        }

        if (request.getParameter("lMass") != "") {
            lmass = Integer.parseInt(request.getParameter("lMass"));
        } else {
            lmass = 4000;
        }

        if (request.getParameter("hMass") != "") {
            hmass = Integer.parseInt(request.getParameter("hMass"));
        } else {
            hmass = 14000;
        }

        if (request.getParameter("ppm") != "") {
            ppm = Integer.parseInt(request.getParameter("ppm"));
        } else {
            ppm=500;
        }

        if (request.getParameter("maxresults") != "") {
            maxresults =
Integer.parseInt(request.getParameter("maxresults"));
        } else {
            maxresults = 10;
        }
        //Get the sql statement
        String sqlSel = "";
        if (massSearch) {
            sqlSel = dbBean.buildCompareStringForTempMassOnly(intensity,
lmass,hmass, ppm);
        } else {
            sqlSel = dbBean.buildCompareStringForTemp(intensity,
lmass,hmass, ppm);
        }
        //Get the array maldi ID list

        ArrayList mID =
dbBean.getAllMaldiDataID(request.getParameter("andor"),request.getParameter("identif
ier"),request.getParameter("lowRM"),
request.getParameter("highRM"),request.getParameter("genus"),request.getParameter("s
pecies"),request.getParameter("subspecies"),request.getParameter("other"),request.ge
tParameter("notes"),request.getParameter("history"), request.getParameter("date"));
        ArrayList rsIts = new ArrayList();
        //connect to the database
        dbBean.dbConnect();

        //To make the search faster we are going to dump all the data into
the malditemp table
        //this table is stored in memory which makes it a lot faster and it
    %>

```

```

also normalizes teh data
    int maldiTempId =
dbBean.dumpMALDIDataToTemp(Integer.parseInt(request.getParameter("id")), intensity,
lmass, hmass, "malditemp");
    dbBean.dbClose();
    dbBean.dbConnect();

    //Loop through each ID and compare it
    sqlSel = sqlSel.replace("$cmpr$", Integer.toString(maldiTempId));

    //If the mass range does not fall within 4 to 14 thousand then we
need to normalize the data before comparing it
    //If it does fall within this mass range then the data is already
normalized and we can use that instead
    if (lmass != 4000 | hmass != 14000) {
        for (int i = 0; i < mID.size(); i++) {
            String l = "";
            l += mID.get(i);
            //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}

            int cmprToId;
            cmprToId = dbBean.dumpMALDIDataToTemp(Integer.parseInt(l),
intensity, lmass, hmass, "malditemp2");

            String tmp = sqlSel.replace("$cmprTo$",
Integer.toString(cmprToId) );
            ResultSet myRS = dbBean.getRS(tmp);
            myRS.next();

            rslts.add(new box(l, myRS.getString("score")));
            myRS.close();
            dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
        }
    } else {
        for (int i = 0; i < mID.size(); i++) {
            String l = "";
            l += mID.get(i);
            //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}

            String tmp = sqlSel.replace("$cmprTo$", l);

            ResultSet myRS = dbBean.getRS(tmp);
            myRS.next();

            rslts.add(new box(l, myRS.getString("score")));
            myRS.close();
        }
    }
}
//now delete the maldiTemp data to free up the memory
dbBean.deleteMaldiTempData(maldiTempId, "malditemp");

mID = null;
//Sort the array, this is a simple bubble sort algorithm
for (int i = 0 ; i< rslts.size(); i++) {
    for (int j=0; j < (rslts.size() - 1 - i) ; j++) {
        box x,y;
        x = (box) rslts.get(j + 1);
        y = (box) rslts.get(j);
        if (x.getO2() == null | y.getO2() == null) {
            //This was a bad score, if it is null move it to the
bottom

            //A null value for a score means no data set was
returned

            //This is probably because the intenisty threshold was

```

higher than
once the data

```

//What is present in the DB, this problem should go away
//is normalized.
if (x.getO2() == null) {
    rslts.set(j, y);
    rslts.set(j+1, x);
} else {
    rslts.set(j, x);
    rslts.set(j+1, y);
}
} else {
    int mInt1 = Integer.parseInt(x.getO2());
    int mInt2 = Integer.parseInt(y.getO2());
    if ( mInt1 > mInt2 ) {
        rslts.set(j, x);
        rslts.set(j+1, y);
    }
}
}
}
for (int i = 0; i < maxresults & i < rslts.size() ; i++) {
    box x = (box) rslts.get(i);
    out.write(x.getO1() + "\t" + x.getO2() + "\n");
}
dbBean.dbClose();
%>

```

```

<%@page contentType="text"%><%@page pageEncoding="UTF-8"%><%@page
import="java.sql.*" %><%@page import="java.util.*" %><%@page import="java.text.*"
%><%@page import="psm.db.box.*" %><jsp:useBean id="dbBean" class="psm.db.dbAccess"
scope="session"/><%
    int intensity, lmass, hmass, ppm, maxresults;
    if (request.getParameter("intensity")!="") {
        intensity = Integer.parseInt(request.getParameter("intensity"));
    } else {
        intensity = 10;
    }

    if (request.getParameter("lMass") != "") {
        lmass = Integer.parseInt(request.getParameter("lMass"));
    } else {
        lmass = 4000;
    }

    if (request.getParameter("hMass") != "") {
        hmass = Integer.parseInt(request.getParameter("hMass"));
    } else {
        hmass = 14000;
    }

    if (request.getParameter("ppm") != "") {
        ppm = Integer.parseInt(request.getParameter("ppm"));
    } else {
        ppm=500;
    }

    if (request.getParameter("maxresults") != "") {
        maxresults =
Integer.parseInt(request.getParameter("maxresults"));
    } else {
        maxresults = 10;
    }
    //Get the sql statement
    String sqlSel = "";
    sqlSel = dbBean.buildCompareStringForTempMassOnly(intensity,
lmass,hmass, ppm);
    //Get the array maldi ID list
    ArrayList mID =
dbBean.getAllMaldiDataID(request.getParameter("andor"),request.getParameter("identif
ier"),request.getParameter("lowRM"),
request.getParameter("highRM"),request.getParameter("genus"),request.getParameter("s
pecies"),request.getParameter("subspecies"),request.getParameter("other"),request.ge
tParameter("notes"),request.getParameter("history"), request.getParameter("date"));
    ArrayList rsIts = new ArrayList();
    //connect to the database
    dbBean.dbConnect();

    //To make the search faster we are going to dump all the data into
the malditemp table
    //this table is stored in memory which makes it a lot faster and it
also normalizes teh data
    int maldiTempId =
dbBean.dumpMALDIDataToTemp(Integer.parseInt(request.getParameter("id")), intensity,
lmass, hmass, "malditemp");
    dbBean.dbClose();
    dbBean.dbConnect();

    //Loop through each ID and compare it
    sqlSel = sqlSel.replace("$cmpr$", Integer.toString(maldiTempId));
    //If the mass range does not fall within 4 to 14 thousand then we

```

```

need to normalize the data before comparing it
//If it does fall within this mass range then the data is already
normalized and we can use that instead
double nstar = (hmass - lmass) /
((Double.parseDouble(request.getParameter("ppm"))/ Math.pow(10, 6)) * hmass * 2);
if (lmass != 4000 | hmass != 14000) {
    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}

        int cmprToId;
        cmprToId = dbBean.dumpMALDIDataToTemp(Integer.parseInt(l),
intensity, lmass,hmass, "malditemp2");

        String tmp = sqlSel.replace("$cmprTo$",
Integer.toString(cmprToId) );
        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rslts.add(new scorecontainer(l, myRS.getString("score"),
myRS.getInt("lc_k"), myRS.getInt("uc_k"), myRS.getInt("n"), nstar));
        myRS.close();
        dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
    }
} else {
    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}

        String tmp = sqlSel.replace("$cmprTo$", l);

        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rslts.add(new scorecontainer(l, myRS.getString("score"),
myRS.getInt("lc_k"), myRS.getInt("uc_k"), myRS.getInt("n"), nstar));
        myRS.close();
    }
}
//now delete the maldiTemp data to free up the memory
dbBean.deleteMaldiTempData(maldiTempId, "malditemp");

mID = null;
//Sort the array, this is a simple bubble sort algorithm
for (int i = 0 ; i< rslts.size(); i++) {
    for (int j=0; j < (rslts.size() - 1 - i) ; j++) {
        scorecontainer x,y;
        x = (scorecontainer) rslts.get(j + 1);
        y = (scorecontainer) rslts.get(j);
        if (x.getScore() == null | y.getScore() == null) {
            //This was a bad score, if it is null move it to the
bottom
            //A null value for a score means no data set was
returned
            //This is probably because the intenisty threshold was
higher than
            //What is present in the DB, this problem should go away
once the data
            //is normalized.
            if (x.getScore() == null) {
                rslts.set(j, y);
                rslts.set(j+1, x);
            } else {

```



```

        rslts.set(j, x);
        rslts.set(j+1, y);
    }
} else {
    int mInt1 = Integer.parseInt(x.getScore());
    int mInt2 = Integer.parseInt(y.getScore());
    if ( mInt1 > mInt2 ) {
        rslts.set(j, x);
        rslts.set(j+1, y);
    }
}
}
}

//Now we want to determine the best guess case for identification
//This works by Taking all Results that scored above 60% and finding
the most occuring strain in the set

//First lets convert all the MALDIIDS with a score above 60 to a
strain ID and store them in an arraylist
ArrayList strains = new ArrayList();

//Since rslts is already sorted from bottom to top (item 0 being
the highest score) we can do a while loop until the score is below 60
int dex=0;
String strainsSQL = "SELECT s.strain_id FROM sample s, maldi_tof m,
identifier WHERE m.sample_id = s.sample_id AND (";
//This loop will develop a sql statement to grab a list of all the
strain ids
scorecontainer b = (scorecontainer) rslts.get(0);
int sc = Integer.parseInt(b.getScore());
while ( sc >= 60) {
    strainsSQL += "m.maldi_tof_id = " + b.getMid() + " OR ";
    dex++;
    b = (scorecontainer) rslts.get(dex);
    sc = Integer.parseInt(b.getScore());
}
out.write("Comparison of: " + request.getParameter("id") + "\tError
(ppm): " + request.getParameter("ppm") + "\tMass Range: " + Integer.toString(lmass)
+ " - " + Integer.toString(hmass));
out.write("\n");
out.write("MALDI ID\tDate\tSample\tSample Name\tNotes\tScore\tP
value\tk\tk\ttn\ttn*\tS");
out.write("\n");

for (int i = 0; i < maxresults & i < rslts.size() ; i++) {
    %><%
    scorecontainer x = (scorecontainer) rslts.get(i);
    ResultSet rsltData = dbBean.getMaldiAndSampleInfo(x.getMid());
    rsltData.next();
    String notes = rsltData.getString("notes");
    out.write(x.getMid() + "\t" + rsltData.getString(2) + "\t" +
rsltData.getString(1) + "\t" + rsltData.getString(4) + "\t\t" + notes + "\t\t" +
x.getScore() + "\t" + x.getPvalue() + "\t" + x.getUc_k() + "\t" + x.getLc_k() + "\t"
+ x.getN() + "\t" + x.getNstar() + "\t" + x.getS());
    out.write("\n");
    rsltData.close();
}
dbBean.dbClose();
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>MALDI-TOF Graph</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="comparemsinsilicographs.jsp">
      <%
        int rangeA, rangeB, topnum;
        if (request.getParameter("rangeA") != "" &&
request.getParameter("rangeA") != null) {
          rangeA = Integer.parseInt(request.getParameter("rangeA"));
        } else {
          rangeA = 4000;
        }
        if (request.getParameter("rangeB") != "" &&
request.getParameter("rangeB") != null) {
          rangeB = Integer.parseInt(request.getParameter("rangeB"));
        } else {
          rangeB = 14000;
        }
        if (request.getParameter("topnum") != "" &&
request.getParameter("topnum") != null) {
          topnum = Integer.parseInt(request.getParameter("topnum"));
        } else {
          topnum = 5;
        }
      %>
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &rangeA=<%=rangeA%>&r
angeB=<%=rangeB%>&topnum=<%=topnum%>">
            </td>
          </tr>
          <tr>
            <td colspan="3">
              &nbsp;
            </td>
          </tr>
          <tr>
            <td colspan="3">
              &rangeA=<%=ra
ngeA%>&rangeB=<%=rangeB%>&topnum=<%=topnum%>&residue=<%=request.getParameter("residu
e")%>">
              </td>
            </tr>
            <tr>
              <td>
                Low Value:&nbsp;<input type="text" name="rangeA" value="<%=rangeA%>">
              </td>
              <td>
                High Value:&nbsp;<input type="text" name="rangeB"
value="<%=rangeB%>">

```

```

        </td>
        <td>
            Number of values Labeled: <input type="text" name="topnum"
value="<%=topnum%>">
        </td>
    </tr>
    <tr>
        <td>
            Residue:&nbsp;<input type="text" name="residue"
value="<%=request.getParameter("residue")%>" readonly>
        </td>
        <td>
            Mass Tolerance:<input type="text" name="masserror"
value="<%=request.getParameter("masserror")%>" readonly>
        </td>
        <td>
            Intensity:<input type="text" name="intensity"
value="<%=request.getParameter("intensity")%>" readonly>
        </td>
    </tr>
    <tr>
        <td colspan="1"> <br>
            In Silico ID:&nbsp;<input type="text" name="id2"
value="<%=request.getParameter("id2")%>" readonly>
        </td>
        <td colspan="2">
            MS/MS ID:&nbsp;<input type="text" name="id1" value=
<%=request.getParameter("id1")%> readonly>
        </td>
    </tr>
    <tr>
        <td colspan="3" align="right">
            <input type="submit" value="Reload">
        </td>
    </tr>
    <tr>
        <td colspan="3" align="center">
            <a
href=insilicoerrorgraphsallmatchandopt.jsp?id1=<%=request.getParameter("id1")%>&id2=
<%=request.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<
%=request.getParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")%>&
masserror=<%=request.getParameter("masserror")%>&residue=<%=request.getParameter("re
sidue")%> target="_top">Error Graphs All Matches</a>
        </td>
    </tr>
    <tr>
        <td colspan="3" align="center">
            <a
href=insilicoerrorgraphssinglematchandopt.jsp?id1=<%=request.getParameter("id1")%>&i
d2=<%=request.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&range
B=<%=request.getParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")
%>&masserror=<%=request.getParameter("masserror")%>&residue=<%=request.getParameter(
"residue")%> target="_top">Error Graphs Single Matches</a>
        </td>
    </tr>
    <tr>
        <td colspan="3" align="center">
            <a
href=comparemstomsAllMatches.jsp?id1=<%=request.getParameter("id1")%>&id2=<%=request
.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<%=request.
getParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")%>&masserror=
<%=request.getParameter("masserror")%>&residue=<%=request.getParameter("residue")%>&
topnum=<%=request.getParameter("topnum")%> target="_top">MSMS All Matches</a>
        </td>
    </tr>

```

```

        <tr>
        <td colspan = "3" align = center>
        <a
href=comparemstomssSingleMatches.jsp?id1=<%=request.getParameter("id1")%>&id2=<%=request.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<%=request.getParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")%>&masserror=<%=request.getParameter("masserror")%>&residue=<%=request.getParameter("residue")%>&topnum=<%=request.getParameter("topnum")%> target="_top">MSMS Single Matches</a>
        </td>
        </tr>
    </table>
</form>
</body>
</html>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<%@page import="java.text.*" %>
<%@page import="psm.db.box.*" %>
<%@page import="java.util.Date" %>
<%@page import="java.text.DateFormat" %>
<%@page import="java.text.SimpleDateFormat"%>

<style>
table.notes
{
font-size:100%;
position:absolute;
visibility:hidden;
background:white;
}
</style>

<script type="text/javascript">
function showmenu(elmnt)
{
document.getElementById(elmnt).style.visibility="visible"
}
function hidemenu(elmnt)
{
document.getElementById(elmnt).style.visibility="hidden"
}
function switchmnu(elmnt) {
if (document.getElementById(elmnt).style.visibility=="hidden") {
document.getElementById(elmnt).style.visibility="visible"
} else {
document.getElementById(elmnt).style.visibility="hidden"
}
}
}

</script>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
boolean calcp = false, calcsc = false, calcscb = false;
if(request.getParameter("calcpval") != null &&
request.getParameter("calcpval").compareTo("on") == 0) {
calcp = true;
}
if(request.getParameter("calcscorea") != null &&
request.getParameter("calcscorea").compareTo("on") == 0) {
calcsc = true;
}
if(request.getParameter("calcscoreb") != null &&
request.getParameter("calcscoreb").compareTo("on") == 0) {
calcscb = true;
}
if(calcp == false && calcsc == false && calcscb == false) {
calcsc = true;
}
long cstart = System.currentTimeMillis();
dbBean.dbConnect();
pageSource.splitTemplatePage(dbBean.getUsername(), "MS/MS to <center><i>In
Silico</i> Comparison</center>", "MaldiComparisonTitle",
application.getRealPath("/"), dbBean);

```

```

out.write(pageSource.getPageHeader());
dbBean.dbClose();
%>
<table align=center cellpadding="0" cellspacing="0"
style="horizontal-align: center; text-align:center"><%
intensity=2;
int mass = 4000, hmass = 14000, maxresults = 25, searchtype = 2,
String fierror = "2.5", masserror = "5", residue = "All";
/*if (request.getParameter("searchtype").compareTo("insilico") == 0)
{
    searchtype = 0;
} else if (request.getParameter("searchtype").compareTo("sms") ==
0){
    searchtype = 1;
}*/
if (request.getParameter("residue") != "") {
    residue = request.getParameter("residue");
}
if (request.getParameter("lMass") != "") {
    lmass = Integer.parseInt(request.getParameter("lMass"));
}
if (request.getParameter("hMass") != "") {
    hmass = Integer.parseInt(request.getParameter("hMass"));
}
if (request.getParameter("masserror") != "") {
    masserror = request.getParameter("masserror");
}
if (request.getParameter("maxresults") != "") {
    maxresults =
Integer.parseInt(request.getParameter("maxresults"));
}
if (request.getParameter("intensity") != "") {
    intensity = Integer.parseInt(request.getParameter("intensity"));
}
if (request.getParameter("fierror") != "") {
    fierror = request.getParameter("fierror");
}

//Get the sql statement
//String sqlSel = "SELECT ROUND(((SELECT COUNT(*) FROM (select
(a.mass + b.mass) / 2 mass from maldiTemp a, maldiTemp2 b where a.intensity >
$minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between (b.mass -
$error$) and (b.mass + $error$) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) m)) * 2 /((select COUNT(*)from maldiTemp a where
a.intensity > $minIntensity$ AND a.mass between $lowMass$ and $highMass$ AND
a.maldi_tof_id = $cmpr$) +(Select COUNT(*)from maldiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$)) * 100, 0) score";
String sqlSel = "";
if (calcs && calcsb) {
    sqlSel = "SELECT lc_k, uc_k, n, Round(lc_k * 2 / (n + uc_k) *
100, 2) as score, Round(lc_k/ uc_k * 100, 2) as scorebeta FROM (SELECT (SELECT
COUNT(*) FROM (select DISTINCT a.mass from maldiTemp a, maldiTemp2 b where
a.intensity > $minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between
(b.mass - $error$) and (b.mass + $error$) and a.mass between $lowMass$ and
$highMass$ and b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$
and b.maldi_tof_id = $cmprTo$) as m) as lc_k, (Select COUNT(*) as
number_in_silico_ions FROM maldiTemp2 b where b.intensity > $minIntensity$ AND

```

```

b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id = $cmprTo$) as n, (select
COUNT(*) FROM maldiTemp a where a.intensity > $minIntensity$ AND a.mass between
$lowMass$ and $highMass$ AND a.maldi_tof_id = $cmpr$) as uc_k) as tbl";
    } else if (calcs) {
        sqlSel = "SELECT lc_k, uc_k, n, Round(lc_k * 2 / (n + uc_k) *
100, 2) as score FROM (SELECT (SELECT COUNT(*) FROM (select DISTINCT a.mass from
malDiTemp a, maldiTemp2 b where a.intensity > $minIntensity$ AND b.intensity >
$minIntensity$ AND a.mass between (b.mass - $error$) and (b.mass + $error$) and
a.mass between $lowMass$ and $highMass$ and b.mass between $lowMass$ and $highMass$
and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) as m) as lc_k, (select
COUNT(*) as number_in_silico_ions FROM maldiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$) as n, (select COUNT(*) FROM maldiTemp a where a.intensity > $minIntensity$
AND a.mass between $lowMass$ and $highMass$ AND a.maldi_tof_id = $cmpr$) as uc_k)
as tbl";
    } else if (calcsb) {
        sqlSel = "SELECT lc_k, uc_k, n, Round(lc_k/ uc_k * 100, 2) as
scorebeta FROM (SELECT (SELECT COUNT(*) FROM (select DISTINCT a.mass from maldiTemp
a, maldiTemp2 b where a.intensity > $minIntensity$ AND b.intensity > $minIntensity$
AND a.mass between (b.mass - $error$) and (b.mass + $error$) and a.mass between
$lowMass$ and $highMass$ and b.mass between $lowMass$ and $highMass$ and
a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) as m) as lc_k, (select
COUNT(*) as number_in_silico_ions FROM maldiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$) as n, (select COUNT(*) FROM maldiTemp a where a.intensity > $minIntensity$
AND a.mass between $lowMass$ and $highMass$ AND a.maldi_tof_id = $cmpr$) as uc_k)
as tbl";
    } else if (calcp) {
        sqlSel = "SELECT lc_k, uc_k, n FROM (SELECT (SELECT COUNT(*)
FROM (select DISTINCT a.mass from maldiTemp a, maldiTemp2 b where a.intensity >
$minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between (b.mass -
$error$) and (b.mass + $error$) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) as m) as lc_k, (select COUNT(*) as number_in_silico_ions
FROM maldiTemp2 b where b.intensity > $minIntensity$ AND b.mass between $lowMass$
and $highMass$ AND b.maldi_tof_id = $cmprTo$) as n, (select COUNT(*) FROM maldiTemp
a where a.intensity > $minIntensity$ AND a.mass between $lowMass$ and $highMass$
AND a.maldi_tof_id = $cmpr$) as uc_k) as tbl";
    }
    //Get the array maldi ID list
    sqlSel = sqlSel.replace("$minIntensity$",
Integer.toString(intensity));
    sqlSel = sqlSel.replace("$error$", fierror);
    sqlSel = sqlSel.replace("$lowMass$", Integer.toString(lmass));
    sqlSel = sqlSel.replace("$highMass$", Integer.toString(hmass));

    ArrayList mID =
dbBean.getAllInSilicoDataID(request.getParameter("andor"), request.getParameter("iden
tifier"), request.getParameter("genus"), request.getParameter("species"), request.getPa
rameter("subspecies"), request.getParameter("other"), request.getParameter("protein"),
request.getParameter("notes"), request.getParameter("mass"),
request.getParameter("masserror"));
    ArrayList rsIts = new ArrayList();
    //connect to the database
    dbBean.dbConnect();

    //To make the search faster we are going to dump all the data into
the maldiTemp table
    //this table is stored in memory which makes it a lot faster and it
also normalizes teh data
    int maldiTempId =
dbBean.dumpMSMSDataToTemp(Integer.parseInt(request.getParameter("id")), intensity,
lmass, hmass, "maldiTemp");
    dbBean.dbClose();

```

```

        dbBean.dbConnect();
        //Loop through each ID and compare it
        sqlSel = sqlSel.replace("$cmpr$", Integer.toString(maldiTempId));

        //Used in the pvalue calculation
        double nstar = (Double.parseDouble(request.getParameter("mass")) -
1mass) / (Double.parseDouble(request.getParameter("fierror"))*2.0);

        int lstProteinSize = mID.size();
        for (int i = 0; i < lstProteinSize; i++) {
            String l = "";
            l += mID.get(i);
            //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}
            int cmprToId;
            cmprToId = dbBean.dumpINSILICODataToTemp(Integer.parseInt(l),
intensity, lmass,hmass, "malditemp2", request.getParameter("residue"));

            String tmp = sqlSel.replace("$cmprTo$",
Integer.toString(cmprToId) );
            ResultSet myRS = dbBean.getRS(tmp);
            myRS.next();

            String USDABeta = "0";
            String USDAAAlpha = "0";

            if (calcs) {
                USDAAAlpha = myRS.getString("score");
            }

            if(calcsb) {
                USDABeta = myRS.getString("scorebeta");
            }

            if(calcp) {
                rslts.add(new scorecontainer(l, USDAAAlpha, USDABeta,
myRS.getInt("lc_k"), myRS.getInt("uc_k"), myRS.getInt("n"), nstar));
            } else {
                rslts.add(new scorecontainer(l, USDAAAlpha, USDABeta));
            }
            myRS.close();
            dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
        }

        //now delete the maldiTemp data to free up the memory
        dbBean.deleteMaldiTempData(maldiTempId, "malditemp");

        mID = null;
        //Sort the array, this is a simple bubble sort algorithm
        for (int i = 0 ; i< rslts.size(); i++) {
            for (int j=0; j < (rslts.size() - 1 - i) ; j++) {
                scorecontainer x,y;
                x = (scorecontainer) rslts.get(j + 1);
                y = (scorecontainer) rslts.get(j);

                double mInt1, mInt2;

                if ((calcs == false && calcsb == false) || (calcp &&
request.getParameter("sort").compareTo("pval") == 0)) {
                    mInt2 = x.getPvalue();
                    mInt1 = y.getPvalue();
                } else if((calcs == false && calcp == false) || calcsb &&
request.getParameter("sort").compareTo("scoreb") == 0) {
                    mInt1 = Double.parseDouble(x.getScorebeta());

```



```

        mInt2 = Double.parseDouble(y.getScorebeta());
    } else {
        mInt1 = Double.parseDouble(x.getScore());
        mInt2 = Double.parseDouble(y.getScore());
    }
    if ( mInt1 > mInt2 ) {
        rsIts.set(j, x);
        rsIts.set(j+1, y);
    }
}
}
long cfinish = System.currentTimeMillis();

long timeToExec = cfinish - cstart;

%>
<tr>
    <td colspan=1 align=center>
        Intensity: <%=request.getParameter("intensity")%>
    </td>
    <td colspan=1 align=center>
        FI Tolerance: <%=request.getParameter("fierror")%>
    </td>
    <td colspan=1 align=left>
        Mass Tolerance: <%=request.getParameter("masserror")%>
    </td>
    <td colspan = 1>
        Residue: <%=residue%>
    </td>
    <td colspan = 1>
        # of Proteins: <%=lstProteinSize%>
    </td>
    <td colspan = 3>
        Execution Speed (ms): <%=Long.toString(timeToExec)%>
    </td>
</tr>
<tr><td colspan=9>
    <a
href="comparemsssampletextonly.jsp?rm=<%=request.getParameter("rm")%>&id=<%=request
.getParameter("id")%>&intensity=<%=request.getParameter("intensity")%>&lMass=<%=requ
est.getParameter("lMass")%>&hMass=<%=request.getParameter("hMass")%>&maxresults=<%=r
quest.getParameter("maxresults")%>&date=<%=request.getParameter("date")%>&mass=<%=r
quest.getParameter("mass")%>&masserror=<%=request.getParameter("masserror")%>&fierr
or=<%=request.getParameter("fierror")%>&sort=<%=request.getParameter("sort")%>&andor
=<%=request.getParameter("andor")%>&identifier=<%=request.getParameter("identifier")
%>&genus=<%=request.getParameter("genus")%>&species=<%=request.getParameter("species
")%>&subspecies=<%=request.getParameter("subspecies")%>&other=<%=request.getParame
ter("other")%>&protein=<%=request.getParameter("protein")%>&notes=<%=request.getParame
ter("notes")%>">Downloadable Text Format</a>
    </td></tr>
<tr>
    <td>
        <span style="font-weight: bold;">&nbsp;<i>In Silico</i>
ID&nbsp;</span>
    </td>
    <td>
        <span style="font-weight: bold;" >&nbsp;<Identifier:&nbsp;</span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;<Sample Name:&nbsp;</span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;<Protein:&nbsp;</span>
    </td>
    <td>

```

```

        <span style="font-weight: bold;">&nbsp;  ALPHA:&nbsp;  </span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;  BETA:&nbsp;  </span>
    </td>
    <td>
        <span style="font-weight:
bold;">&nbsp;  <i>P</i>:&nbsp;  </span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;  Compare:&nbsp;  </span>
    </td>
</tr>
<%
DecimalFormat df = new DecimalFormat("0.0E0");
for (int i = 0; i < maxresults & i < rslts.size() ; i++) {
    %><%
        scorecontainer x = (scorecontainer) rslts.get(i);
        ResultSet rsltData =
dbBean.getInSilicoAndSampleInfo(x.getMid());
        rsltData.next();
        String notes = rsltData.getString("protein") + "<br>" +
rsltData.getString("average_mw");
        String notesfull = notes;
        String popuptbl;
        popuptbl = "<table bordercolor=black cellspacing='0'
cellpadding='0' border='1' class='notes' id='" + x.getMid() + "' width='300'><tr><td
style='vertical-align: top; text-align: center; background-color: rgb(36, 147,
184);'\>" + notesfull + "</td></tr></table>";
        String popuptbl2 = "<table bordercolor=black cellspacing='0'
cellpadding='0' border='1' class='notes' id='" + x.getMid() + "' width='250'><tr><td style='vertical-align: top; text-align: Left; background-color:
rgb(36, 147, 184);'\><i>P</i> Value: " + x.getPvalue() + "<br>K: " + x.getUc_k() +
"<br>k: " + x.getLc_k() + "<br>n: " + x.getN() + "<br>n*: " + x.getNstar() + "<br>s:
" + x.getS() + "</td></tr></table>";
        if (notes == null || notes.compareTo("") == 0) {
            notes = "";
        } else {
            if (notes.length() > 8) {
                notes = "&nbsp;  " + notes.substring(0,8);
            } else {
                notes = "&nbsp;  " + notes;
            }
        }
        out.write("<tr><td><a
href='insilicodatawithgraph.jsp?insilicoid=" + x.getMid() + "'>" + x.getMid() +
"</a></td><td>" + rsltData.getString(1) + "</td><td>" + rsltData.getString(3) +
"</td><td onmouseout='hidemenu('" + x.getMid() + "')'\><a
href='javascript:void(0);' onmouseover='showmenu('" + x.getMid() + "')'\>" + notes
+ "</a>" + popuptbl + "</td><td align=center>" + x.getScore() + "</td><td
align=center>" + x.getScorebeta() + "</td><td align=left><a
href='javascript:void(0);' onclick='switchmnu('" + x.getMid() + "'>" +
df.format(x.getPvalue()) + "</a>" + popuptbl2 + "</td><td><a
href='msinsilicodatawithgraphcompare.jsp?id1=" + request.getParameter("id") +
"&id2=" + x.getMid() + "&rangeA=" + lmass + "&rangeB=" + hmass + "&intensity=" +
intensity + "&masserror=" + fierror + "&residue=" + request.getParameter("residue")
+ "'>Graphs</a>" + "</td></tr>");
        rsltData.close();
    }
}
dbBean.dbClose();
%>
</table>
<%out.write(pageSource.getPageFooter()); %>

```

```

<%@page contentType="text"%><%@page pageEncoding="UTF-8"%><%@page
import="java.sql.*" %><%@page import="java.util.*" %><%@page import="java.text.*"
%><%@page import="psm.db.box.*" %><jsp:useBean id="dbBean" class="psm.db.dbAccess"
scope="session"/><%
intensity=2;    int lmass = 4000, hmass = 14000, maxresults = 25, searchtype = 2,
String fierror = "2.5", masserror = "5";
/*if (request.getParameter("searchtype").compareTo("insilico") == 0)
{
    searchtype = 0;
} else if (request.getParameter("searchtype").compareTo("msms") ==
0){
    searchtype = 1;
}*/

if (request.getParameter("lMass") != "") {
    lmass = Integer.parseInt(request.getParameter("lMass"));
}

if (request.getParameter("hMass") != "") {
    hmass = Integer.parseInt(request.getParameter("hMass"));
}

if (request.getParameter("masserror") != "") {
    masserror = request.getParameter("masserror");
}

if (request.getParameter("maxresults") != "") {
    maxresults =
Integer.parseInt(request.getParameter("maxresults"));
}

if (request.getParameter("intensity") != "") {
    intensity = Integer.parseInt(request.getParameter("intensity"));
}

if (request.getParameter("fierror") != "") {
    fierror = request.getParameter("fierror");
}

//Get the sql statement
//String sqlSel = "SELECT ROUND(((SELECT COUNT(*) FROM (select
(a.mass + b.mass) / 2 mass from maldiTemp a, maldiTemp2 b where a.intensity >
$minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between (b.mass -
$error$) and (b.mass + $error$) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) m)) * 2 / ((select COUNT(*) from maldiTemp a where
a.intensity > $minIntensity$ AND a.mass between $lowMass$ and $highMass$ AND
a.maldi_tof_id = $cmpr$) + (select COUNT(*) from maldiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$)) * 100, 0) score";
String sqlSel = "SELECT lc_k, uc_k, n, Round(lc_k * 2 / (n + uc_k) *
100, 0) as score FROM (SELECT (SELECT COUNT(*) FROM (select DISTINCT a.mass from
malDiTemp a, malDiTemp2 b where a.intensity > $minIntensity$ AND b.intensity >
$minIntensity$ AND a.mass between (b.mass - $error$) and (b.mass + $error$) and
a.mass between $lowMass$ and $highMass$ and b.mass between $lowMass$ and $highMass$
and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) as m) as lc_k, (select
COUNT(*) as number_in_silico_ions FROM malDiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$) as n, (select COUNT(*) FROM malDiTemp a where a.intensity > $minIntensity$
AND a.mass between $lowMass$ and $highMass$ AND a.maldi_tof_id = $cmpr$) as uc_k)
as tbl";

//Get the array maldi ID list
sqlSel = sqlSel.replace("$minIntensity$",

```

```

Integer.toString(intensity));
    sqlSel = sqlSel.replace("$error$", fierror);
    sqlSel = sqlSel.replace("$lowMass$", Integer.toString(lmass));
    sqlSel = sqlSel.replace("$highMass$", Integer.toString(hmass));

    ArrayList mID =
dbBean.getAllInSilicoDataID(request.getParameter("andor"), request.getParameter("iden
tifier"), request.getParameter("genus"), request.getParameter("species"), request.getPa
rameter("subspecies"), request.getParameter("other"), request.getParameter("protein"),
request.getParameter("notes"), request.getParameter("mass"),
request.getParameter("masserror"));
    ArrayList rslts = new ArrayList();
    //connect to the database
    dbBean.dbConnect();

    //To make the search faster we are going to dump all the data into
the malditemp table
    //this table is stored in memory which makes it a lot faster and it
also normalizes teh data
    int maldiTempId =
dbBean.dumpMSMSDataToTemp(Integer.parseInt(request.getParameter("id")), intensity,
lmass, hmass, "malditemp");
    dbBean.dbClose();
    dbBean.dbConnect();
    //Loop through each ID and compare it
    sqlSel = sqlSel.replace("$cmpr$", Integer.toString(maldiTempId));

    //Used in the pvalue calculation
    double nstar = (Double.parseDouble(request.getParameter("mass")) -
lmass) / (Double.parseDouble(request.getParameter("fierror"))*2.0);

    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}
        int cmprToId;
        cmprToId = dbBean.dumpINSILICODataToTemp(Integer.parseInt(l),
intensity, lmass, hmass, "malditemp2");

        String tmp = sqlSel.replace("$cmprTo$",
Integer.toString(cmprToId) );
        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rslts.add(new scorecontainer(l, myRS.getString("score"),
myRS.getInt("lc_k"), myRS.getInt("uc_k"), myRS.getInt("n"), nstar));
        myRS.close();
        dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
    }

    //now delete the maldiTemp data to free up the memory
    dbBean.deleteMaldiTempData(maldiTempId, "malditemp");

    mID = null;
    //Sort the array, this is a simple bubble sort algorithm
    for (int i = 0 ; i< rslts.size(); i++) {
        for (int j=0; j < (rslts.size() - 1 - i) ; j++) {
            scorecontainer x,y;
            x = (scorecontainer) rslts.get(j + 1);
            y = (scorecontainer) rslts.get(j);

            double mInt1, mInt2;

            if (request.getParameter("sort").compareTo("pval") == 0) {

```

```

        mInt2 = x.getPvalue();
        mInt1 = y.getPvalue();
    } else {
        mInt1 = Integer.parseInt(x.getScore());
        mInt2 = Integer.parseInt(y.getScore());
    }
    if ( mInt1 > mInt2 ) {
        rslts.set(j, x);
        rslts.set(j+1, y);
    }
}

    out.write("Comparison of: " + request.getParameter("id") + "\tError
(daltons): " + request.getParameter("masserror") + "\tMass Range: " +
Integer.toString(lmass) + " - " + Integer.toString(hmass));
    out.write("\n");
    out.write("In Silico ID\tIdentifier\tSample Name\tProtein\tAverage
MW\tScore\tP Value\tK\tk\tn\tn*\tS");
    out.write("\n");

    for (int i = 0; i < maxresults & i < rslts.size() ; i++) {
        %><%
        scorecontainer x = (scorecontainer) rslts.get(i);
        ResultSet rsltData =
dbBean.getInSilicoAndSampleInfo(x.getMid());
        rsltData.next();
        String notes = rsltData.getString("protein");

        out.write(x.getMid() + "\t" + rsltData.getString(1) + "\t" +
rsltData.getString(3) + "\t" + notes + "\t" + rsltData.getString("average_mw") +
"\t" + x.getScore() + "\t" + x.getPvalue() + "\t" + x.getUc_k() + "\t" + x.getLc_k()
+ "\t" + x.getN() + "\t" + x.getNstar() + "\t" + x.getS());
        out.write("\n");
        rsltData.close();
    }
    dbBean.dbClose();
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>MALDI-TOF Graph</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="comparemsmstomsgsgraphs.jsp">
      <%
        int rangeA, rangeB, topnum;
        if (request.getParameter("rangeA") != "" &&
request.getParameter("rangeA") != null) {
          rangeA = Integer.parseInt(request.getParameter("rangeA"));
        } else {
          rangeA = 4000;
        }
        if (request.getParameter("rangeB") != "" &&
request.getParameter("rangeB") != null) {
          rangeB = Integer.parseInt(request.getParameter("rangeB"));
        } else {
          rangeB = 14000;
        }
        if (request.getParameter("topnum") != "" &&
request.getParameter("topnum") != null) {
          topnum = Integer.parseInt(request.getParameter("topnum"));
        } else {
          topnum = 5;
        }
      %>
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &rangeA=<%=rangeA%>&r
angeB=<%=rangeB%>&topnum=<%=topnum%>">
            </td>
          </tr>
          <tr>
            <td colspan="3">
              &nbsp;
            </td>
          </tr>
          <tr>
            <td colspan="3">
              &rangeA=<%=rangeA%>&r
angeB=<%=rangeB%>&topnum=<%=topnum%>">
            </td>
          </tr>
          <tr>
            <td>
              Low Value:&nbsp;<input type="text" name="rangeA" value="<%=rangeA%>">
            </td>
            <td>
              High Value:&nbsp;<input type="text" name="rangeB"
value="<%=rangeB%>">
            </td>
          </tr>
        </table>
      </form>
    </body>
  </html>

```

[illegible]

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<%@page import="java.text.*" %>
<%@page import="psm.db.box.*" %>

<style>
table.notes
{
font-size:100%;
position:absolute;
visibility:hidden;
background:white;
}
</style>
<script language="JavaScript">
function selectAll(c){
for(i=1; i < c.length; i++) c[i].checked=c[0].checked;
}
</script>
<script type="text/javascript">
function showmenu(elmnt)
{
document.getElementById(elmnt).style.visibility="visible"
}
function hidemenu(elmnt)
{
document.getElementById(elmnt).style.visibility="hidden"
}
function switchmnu(elmnt) {
if (document.getElementById(elmnt).style.visibility=="hidden") {
document.getElementById(elmnt).style.visibility="visible"
} else {
document.getElementById(elmnt).style.visibility="hidden"
}
}
</script>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
dbBean.dbConnect();
pageSource.splitTemplatePage(dbBean.getUsername(), "MS/MS-MS/MS Comparison",
"MalDiComparisonTitle", application.getRealPath("/"), dbBean);
out.write(pageSource.getPageHeader());
dbBean.dbClose();
%>
<table align=center cellpadding="0" cellspacing="0"
style="horizontal-align: center; text-align:center" width=100%>
<form name="myForm">
<tr><td colspan=8>
<table align=center cellpadding="0" cellspacing="0"
style="horizontal-align: center; text-align:center">
<tr>
<td>Low value:&nbsp;  </td>
<td>
<input type="text" name="rangeA"
value="<%=request.getParameter("lMass") %>" size="5">
</td>
<td>High value:&nbsp;  </td>
<td>

```



```

        <input type="text" name="rangeB"
value="<%=request.getParameter("hMass")%>" size="5">
        </td>
        <td>Labeled values:&nbsp;&nbsp;&nbsp;</td>
        <td colspan=2 align=left>
            <input type="text" name="topnum" value="5"
size="2">
        </td>
    </tr>
    <tr>
        <td colspan=9>&nbsp;&nbsp;&nbsp;</td>
    </tr>
    <tr align=left>
        <td colspan=2>Intensity Threshold (%):&nbsp;&nbsp;&
            <input type="text" name="intensity"
value="<%=request.getParameter("intensity")%>" size="2">
        </td>
        <td colspan=2>&nbsp;&nbsp;&Error (daltons):&nbsp;&nbsp;&
            <input type="text" name="masserror"
value="<%=request.getParameter("fierror")%>" size="2">
        </td>
        <td colspan=2>Size (pixels):&nbsp;&nbsp;&
            <input type="text" name="x" value="1200"
size="4">&nbsp;&nbsp;&x&nbsp;&nbsp;&
            <input type="text" name="y" value="900"
size="4">
        </td>
    </tr>
    <tr>
        <td colspan=9>&nbsp;&nbsp;&</td>
    </tr>
    <tr>
        <td colspan=2>&nbsp;&nbsp;&<input type="button"
value="Subtraction"
onclick="document.myForm.action='subtractionmsmsgraph.jsp';document.myForm.target='_
sgraph';document.myForm.submit();">
        </td>
        <td colspan=1>
            &nbsp;&nbsp;&<input type="button" value="Mirrored"
onclick="document.myForm.action='mirroredmsmsgraphs.jsp';document.myForm.target='_mi
rgraph';document.myForm.submit();">
        </td>
        <td colspan=2>
            &nbsp;&nbsp;&<input type="button" value="Show Graphs"
onclick="document.myForm.action='multiplemsmsgraphdisplay.jsp';document.myForm.targe
t='_mgraph';document.myForm.submit();">
        </td>
        <td colspan=2>
            &nbsp;&nbsp;&<input type="button" value="Scatterplot"
onclick="document.myForm.action='scatterplotmsms.jsp';document.myForm.target='_scatt
er';document.myForm.submit();">
        </td>
    </tr>
    <tr>
        <td colspan=9>&nbsp;&nbsp;&</td>
    </tr>
</table>
</td></tr>
<%
int lmass = 4000, hmass = 14000, maxresults = 25, searchtype = 2,
intensity=2;
String fierror = "2.5", masserror = "5";
/*if (request.getParameter("searchtype").compareTo("insilico") == 0)
{
    searchtype = 0;

```

```

    } else if (request.getParameter("searchtype").compareTo("msms") ==
0){
        searchtype = 1;
    }*/
    if (request.getParameter("lMass") != "") {
        lmass = Integer.parseInt(request.getParameter("lMass"));
    }
    if (request.getParameter("hMass") != "") {
        hmass = Integer.parseInt(request.getParameter("hMass"));
    }
    if (request.getParameter("masserror") != "") {
        masserror = request.getParameter("masserror");
    }
    if (request.getParameter("maxresults") != "") {
        maxresults =
Integer.parseInt(request.getParameter("maxresults"));
    }
    if (request.getParameter("intensity") != "") {
        intensity = Integer.parseInt(request.getParameter("intensity"));
    }
    if (request.getParameter("fierror") != "") {
        fierror = request.getParameter("fierror");
    }

    //Get the sql statement
    //String sqlSel = "SELECT ROUND(((SELECT COUNT(*) FROM (select
(a.mass + b.mass) / 2 mass from maldiTemp a, maldiTemp2 b where a.intensity >
$minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between (b.mass -
$error$) and (b.mass + $error$) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) m)) * 2 /((select COUNT(*)from maldiTemp a where
a.intensity > $minIntensity$ AND a.mass between $lowMass$ and $highMass$ AND
a.maldi_tof_id = $cmpr$) +(Select COUNT(*)from maldiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$)) * 100, 0) score";
    String sqlSel = "SELECT lc_k, uc_k, n, Round(lc_k * 2 / (n + uc_k) *
100, 0) as score FROM (SELECT (SELECT COUNT(*) FROM (select DISTINCT a.mass from
malDiTemp a, malDiTemp2 b where a.intensity > $minIntensity$ AND b.intensity >
$minIntensity$ AND a.mass between (b.mass - $error$) and (b.mass + $error$) and
a.mass between $lowMass$ and $highMass$ and b.mass between $lowMass$ and $highMass$
and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) as m) as lc_k, (Select
COUNT(*) as number_in_silico_ions FROM malDiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$) as n, (select COUNT(*) FROM malDiTemp a where a.intensity > $minIntensity$
AND a.mass between $lowMass$ and $highMass$ AND a.maldi_tof_id = $cmpr$) as uc_k)
as tbl";
    //Get the array maldi ID list

    sqlSel = sqlSel.replace("$minIntensity$",
Integer.toString(intensity));
    sqlSel = sqlSel.replace("$error$", fierror);
    sqlSel = sqlSel.replace("$lowMass$", Integer.toString(lmass));
    sqlSel = sqlSel.replace("$highMass$", Integer.toString(hmass));

    ArrayList mID =
dbBean.getAllMSMSDataID(request.getParameter("andor"), request.getParameter("identifi
er"), request.getParameter("genus"), request.getParameter("species"), request.getParame
ter("subspecies"), request.getParameter("other"), request.getParameter("protein"), requ
est.getParameter("notes"), request.getParameter("mass"),

```

```

request.getParameter("masserror"));
    ArrayList rslts = new ArrayList();
    //connect to the database
    dbBean.dbConnect();

    //To make the search faster we are going to dump all the data into
the malditemp table
    //this table is stored in memory which makes it a lot faster and it
also normalizes teh data
    int maldiTempId =
dbBean.dumpMSMSDataToTemp(Integer.parseInt(request.getParameter("id")), intensity,
lmass, hmass, "malditemp");
    dbBean.dbClose();
    dbBean.dbConnect();
    //Loop through each ID and compare it
    sqlSel = sqlSel.replace("$cmpr$", Integer.toString(maldiTempId));

    //Used in the pvalue calculation
    double nstar = (Double.parseDouble(request.getParameter("mass")) -
lmass) / (Double.parseDouble(request.getParameter("fierror"))*2.0);

    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}
        int cmprToId;
        cmprToId = dbBean.dumpMSMSDataToTemp(Integer.parseInt(l),
intensity, lmass, hmass, "malditemp2");

        String tmp = sqlSel.replace("$cmprTo$",
Integer.toString(cmprToId) );
        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rslts.add(new scorecontainer(l, myRS.getString("score"),
myRS.getInt("lc_k"), myRS.getInt("uc_k"), myRS.getInt("n"), nstar));
        myRS.close();
        dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
    }

    //now delete the maldiTemp data to free up the memory
    dbBean.deleteMaldiTempData(maldiTempId, "malditemp");

    mID = null;
    //Sort the array, this is a simple bubble sort algorithm
    for (int i = 0 ; i< rslts.size(); i++) {
        for (int j=0; j < (rslts.size() - 1 - i) ; j++) {
            scorecontainer x,y;
            x = (scorecontainer) rslts.get(j + 1);
            y = (scorecontainer) rslts.get(j);

            double mInt1, mInt2;

            if (request.getParameter("sort").compareTo("pval") == 0) {
                mInt2 = x.getPvalue();
                mInt1 = y.getPvalue();
            } else {
                mInt1 = Integer.parseInt(x.getScore());
                mInt2 = Integer.parseInt(y.getScore());
            }
            if ( mInt1 > mInt2 ) {
                rslts.set(j, x);
                rslts.set(j+1, y);
            }
        }
    }

```

```

    }
}

%>
<tr><td colspan=9>
    <a
href="comparemsmstomsmssampletextonly.jsp?rm=<%=request.getParameter("rm")%>&id=<%=r
equest.getParameter("id")%>&intensity=<%=request.getParameter("intensity")%>&lMass=<
%=request.getParameter("lMass")%>&hMass=<%=request.getParameter("hMass")%>&maxresult
s=<%=request.getParameter("maxresults")%>&date=<%=request.getParameter("date")%>&mas
s=<%=request.getParameter("mass")%>&masserror=<%=request.getParameter("masserror")%>
&fierror=<%=request.getParameter("fierror")%>&sort=<%=request.getParameter("sort")%>
&andor=<%=request.getParameter("andor")%>&identifier=<%=request.getParameter("identi
fier")%>&genus=<%=request.getParameter("genus")%>&species=<%=request.getParameter("s
pecies")%>&subspecies=<%=request.getParameter("subspecies")%>&other=<%=request.getPa
rameter("other")%>&protein=<%=request.getParameter("protein")%>&notes=<%=request.get
Parameter("notes")%>">Downloadable Text Format</a>
    </td></tr>
<tr>
    <td>
        <input type=checkbox name="checkid"
onclick="selectAll(this.form.checkid);">
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;MS/MS ID&nbsp;</span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;Identifier:&nbsp;</span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;Sample Name:&nbsp;</span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;Protein:&nbsp;</span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;Score:&nbsp;</span>
    </td>
    <td>
        <span style="font-weight:
bold;">&nbsp;<i>P</i>:&nbsp;</span>
    </td>
    <td>
        <span style="font-weight: bold;">&nbsp;Compare:&nbsp;</span>
    </td>
</tr>
<%
DecimalFormat df= new DecimalFormat("0.0E0");
for (int i = 0; i < maxresults & i < rslts.size() ; i++) {
    %><%
    scorecontainer x = (scorecontainer) rslts.get(i);
    ResultSet rsltData = dbBean.getMSSMSAndSampleInfo(x.getMid());
    rsltData.next();
    String notes = rsltData.getString("protein") + "<br>" +
rsltData.getString("precursor_ion_mtoz");
    String notesfull = notes;
    String popuptbl;
    popuptbl = "<table bordercolor=black cellspacing='0'
cellpadding='0' border='1' class='notes' id='" + x.getMid() + "' width='300'><tr><td
style='vertical-align: top; text-align: center; background-color: rgb(36, 147,
184);'\>" + notesfull + "</td></tr></table>";
    String popuptbl2 = "<table bordercolor=black cellspacing='0'
cellpadding='0' border='1' class='notes' id='" + x.getMid() + "'><tr><td style='
width='250'><tr><td style='vertical-align: top; text-align: Left; background-color:
rgb(36, 147, 184);'\><i>P</i> Value: " + x.getPvalue() + "<br>K: " + x.getUc_k() +

```

```

"<br>k: " + x.getLc_k() + "<br>n: " + x.getN() + "<br>n*: " + x.getNstar() + "<br>s:
" + x.getS() + "</td></tr></table>";
        if (notes == null || notes.compareTo("") == 0) {
            notes= "";
        } else {
            if (notes.length() > 8) {
                notes = "&nbsp;" + notes.substring(0,8);
            } else {
                notes = "&nbsp;" + notes;
            }
        }
        String outstr = "<tr><td $color$><input type=checkbox
name='checkid' value='" + x.getMid() + "'></td><td $color$><a
href='msmsdatawithgraph.jsp?msmsid='" + x.getMid() + "'>" + x.getMid() +
"</a></td><td $color$>" + rsldata.getString(1) + "</td><td $color$>" +
rsldata.getString(4) + "</td><td $color$ onmouseover=\"hidemenu('" + x.getMid() +
"')\"><a href='javascript:void(0);' onmouseover=\"showmenu('" + x.getMid() +
"')\">" + notes + "</a>" + popuptbl + "</td><td align=center $color$>" + x.getScore() +
"</td><td align=left $color$><a href='javascript:void(0);' onclick=\"switchmnu('" +
x.getMid() + "b')\">" + df.format(x.getPvalue()) + "</a>" + popuptbl2 + "</td><td
$color$><a href='comparemsmstomsmsgraphs.jsp?id1='" + request.getParameter("id") +
"&id2='" + x.getMid() + "&rangeA='" + lmass + "&rangeB='" + hmass + "'>Graphs</a>" +
"</td></tr>";
        if (request.getParameter("id").compareTo(x.getMid()) == 0) {
            outstr = outstr.replace("$color$", "style='background-color:
rgb(251, 175, 55);'");
        } else {
            outstr = outstr.replace("$color$", "");
        }
        out.write(outstr);
        rsldata.close();
    }
    dbBean.dbClose();
    %>
</table>
</form>
<%out.write(pageSource.getPageFooter()); %>

```

```

<%@page contentType="text"%><%@page pageEncoding="UTF-8"%><%@page
import="java.sql.*" %><%@page import="java.util.*" %><%@page import="java.text.*"
%><%@page import="psm.db.box.*" %><jsp:useBean id="dbBean" class="psm.db.dbAccess"
scope="session"/><%
intensity=2;    int lmass = 4000, hmass = 14000, maxresults = 25, searchtype = 2,
String fierror = "2.5", masserror = "5";
/*if (request.getParameter("searchtype").compareTo("insilico") == 0)
{
    searchtype = 0;
} else if (request.getParameter("searchtype").compareTo("msms") ==
0){
    searchtype = 1;
}*/

if (request.getParameter("lMass") != "") {
    lmass = Integer.parseInt(request.getParameter("lMass"));
}

if (request.getParameter("hMass") != "") {
    hmass = Integer.parseInt(request.getParameter("hMass"));
}

if (request.getParameter("masserror") != "") {
    masserror = request.getParameter("masserror");
}

if (request.getParameter("maxresults") != "") {
    maxresults =
Integer.parseInt(request.getParameter("maxresults"));
}

if (request.getParameter("intensity") != "") {
    intensity = Integer.parseInt(request.getParameter("intensity"));
}

if (request.getParameter("fierror") != "") {
    fierror = request.getParameter("fierror");
}

//Get the sql statement
//String sqlSel = "SELECT ROUND(((SELECT COUNT(*) FROM (select
(a.mass + b.mass) / 2 mass from maldiTemp a, maldiTemp2 b where a.intensity >
$minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between (b.mass -
$error$) and (b.mass + $error$) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) m)) * 2 / ((select COUNT(*) from maldiTemp a where
a.intensity > $minIntensity$ AND a.mass between $lowMass$ and $highMass$ AND
a.maldi_tof_id = $cmpr$) + (select COUNT(*) from maldiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$)) * 100, 0) score";
String sqlSel = "SELECT lc_k, uc_k, n, Round(lc_k * 2 / (n + uc_k) *
100, 0) as score FROM (SELECT (SELECT COUNT(*) FROM (select DISTINCT a.mass from
malDiTemp a, malDiTemp2 b where a.intensity > $minIntensity$ AND b.intensity >
$minIntensity$ AND a.mass between (b.mass - $error$) and (b.mass + $error$) and
a.mass between $lowMass$ and $highMass$ and b.mass between $lowMass$ and $highMass$
and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) as m) as lc_k, (select
COUNT(*) as number_in_silico_ions FROM malDiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$) as n, (select COUNT(*) FROM malDiTemp a where a.intensity > $minIntensity$
AND a.mass between $lowMass$ and $highMass$ AND a.maldi_tof_id = $cmpr$) as uc_k)
as tbl";

//Get the array maldi ID list
sqlSel = sqlSel.replace("$minIntensity$",

```

```

Integer.toString(intensity));
    sqlSel = sqlSel.replace("$error$", fierror);
    sqlSel = sqlSel.replace("$lowMass$", Integer.toString(lmass));
    sqlSel = sqlSel.replace("$highMass$", Integer.toString(hmass));

    ArrayList mID =
dbBean.getAllMSMSDataID(request.getParameter("andor"), request.getParameter("identi-
er"), request.getParameter("genus"), request.getParameter("species"), request.getParame-
ter("subspecies"), request.getParameter("other"), request.getParameter("protein"), requ-
est.getParameter("notes"), request.getParameter("mass"),
request.getParameter("masserror"));
    ArrayList rslts = new ArrayList();
    //connect to the database
    dbBean.dbConnect();

    //To make the search faster we are going to dump all the data into
the malditemp table
    //this table is stored in memory which makes it a lot faster and it
also normalizes teh data
    int maldiTempId =
dbBean.dumpMSMSDataToTemp(Integer.parseInt(request.getParameter("id")), intensity,
lmass, hmass, "malditemp");
    dbBean.dbClose();
    dbBean.dbConnect();
    //Loop through each ID and compare it
    sqlSel = sqlSel.replace("$cmpr$", Integer.toString(maldiTempId));

    //Used in the pvalue calculation
    double nstar = (Double.parseDouble(request.getParameter("mass")) -
lmass) / (Double.parseDouble(request.getParameter("fierror"))*2.0);

    for (int i = 0; i < mID.size(); i++) {
        String l = "";
        l += mID.get(i);
        //if (l.compareTo(request.getParameter("id")) == 0 ) {
continue;}
        int cmprToId;
        cmprToId = dbBean.dumpMSMSDataToTemp(Integer.parseInt(l),
intensity, lmass, hmass, "malditemp2");

        String tmp = sqlSel.replace("$cmprTo$",
Integer.toString(cmprToId) );
        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rslts.add(new scorecontainer(l, myRS.getString("score"),
myRS.getInt("lc_k"), myRS.getInt("uc_k"), myRS.getInt("n"), nstar));
        myRS.close();
        dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
    }

    //now delete the maldiTemp data to free up the memory
    dbBean.deleteMaldiTempData(maldiTempId, "malditemp");

    mID = null;
    //Sort the array, this is a simple bubble sort algorithm
    for (int i = 0 ; i< rslts.size(); i++) {
        for (int j=0; j < (rslts.size() - 1 - i) ; j++) {
            scorecontainer x,y;
            x = (scorecontainer) rslts.get(j + 1);
            y = (scorecontainer) rslts.get(j);

            double mInt1, mInt2;

            if (request.getParameter("sort").compareTo("pval") == 0) {

```

```

        mInt2 = x.getPvalue();
        mInt1 = y.getPvalue();
    } else {
        mInt1 = Integer.parseInt(x.getScore());
        mInt2 = Integer.parseInt(y.getScore());
    }
    if ( mInt1 > mInt2 ) {
        rslts.set(j, x);
        rslts.set(j+1, y);
    }
}

        out.write("Comparison of: " + request.getParameter("id") + "\tError
(daltons): " + request.getParameter("masserror") + "\tMass Range: " +
Integer.toString(lmass) + " - " + Integer.toString(hmass));
        out.write("\n");
        out.write("MS/MS ID\tIdentifier\tSample Name\tProtein\tAverage
MW\tScore\tP Value\tK\tk\tN\tN*\tS");
        out.write("\n");

        for (int i = 0; i < maxresults & i < rslts.size() ; i++) {
            %><%
            scorecontainer x = (scorecontainer) rslts.get(i);
            ResultSet rsltData = dbBean.getMSMSAndSampleInfo(x.getMid());
            rsltData.next();
            String notes = rsltData.getString("protein");

            out.write(x.getMid() + "\t" + rsltData.getString(1) + "\t" +
rsltData.getString(3) + "\t" + notes + "\t" +
rsltData.getString("precursor_ion_mtoz") + "\t" + x.getScore() + "\t" +
x.getPvalue() + "\t" + x.getUc_k() + "\t" + x.getLc_k() + "\t" + x.getN() + "\t" +
x.getNstar() + "\t" + x.getS());
            out.write("\n");
            rsltData.close();
        }
        dbBean.dbClose();
    %>

```



```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>MS/MS Graph All Matches</title>
</head>
<body>
<center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
<form action="comparemstomsAllMatches.jsp">
<%
int rangeA, rangeB, topnum;
if (request.getParameter("rangeA") != "" &&
request.getParameter("rangeA") != null) {
rangeA = Integer.parseInt(request.getParameter("rangeA"));
} else {
rangeA = 4000;
}
if (request.getParameter("rangeB") != "" &&
request.getParameter("rangeB") != null) {
rangeB = Integer.parseInt(request.getParameter("rangeB"));
} else {
rangeB = 14000;
}
if (request.getParameter("topnum") != "" &&
request.getParameter("topnum") != null) {
topnum = Integer.parseInt(request.getParameter("topnum"));
} else {
topnum = 5;
}
%>
<table border="0" cellpadding = "0" cellspacing = "0" align=center >
<tr>
<td colspan="3">
&rangeA=<%=rangeA%>&r
angeB=<%=rangeB%>&topnum=<%=topnum%>">
</td>
</tr>
<tr>
<td colspan="3">
<%
dbBean.dbConnect();
String selstr = "";
selstr = "SELECT i.aa_seq, i.protein, CONCAT_WS(' ', g.name, sp.name,
sb.name, st.name) as org FROM in_silico i, strain s, genus g, species sp, subspecies
sb, subtype st WHERE s.strain_id = i.strain_id AND g.genus_id = s.genus_id AND
sp.species_id = s.species_id AND sb.subspecies_id = s.subspecies_id AND
st.subtype_id = s.subtype_id AND i.in_silico_id = " + request.getParameter("id2");
ResultSet myRS = dbBean.getRS(selstr);
if (myRS.next()) {
%>
<b>AA Seq:&nbsp;&nbsp;&nbsp;</b><%=myRS.getString("aa_seq")%><br>
<b>Protein:&nbsp;&nbsp;&nbsp;</b><%=myRS.getString("protein")%><br>
<b>Organism:&nbsp;&nbsp;&nbsp;</b><%=myRS.getString("org")%><br><br>
<%

```

```

    }
    myRS.close();
    dbBean.dbClose();
%>
</td>
</tr>
<tr>
<td colspan="3">
    &id2=<%=request.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<%=request.getParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")%>&masserror=<%=request.getParameter("masserror")%>&residue=<%=request.getParameter("residue")%>&topnum=<%=request.getParameter("topnum")%>">
    </td>
</tr>
<tr>
    <td>
        Low Value:&nbsp;<input type="text" name="rangeA" value="<%=rangeA%>">
    </td>
    <td>
        High Value:&nbsp;<input type="text" name="rangeB"
value="<%=rangeB%>">
    </td>
    <td>
        Number of Values Labeled: <input type="text" name="topnum"
value="<%=topnum%>">
    </td>
</tr>
<tr>
    <td>
        Residue:&nbsp;<input type="text" name="residue"
value="<%=request.getParameter("residue")%>" readonly>
    </td>
    <td>
        Mass Tolerance:<input type="text" name="masserror"
value="<%=request.getParameter("masserror")%>" readonly>
    </td>
    <td>
        Intensity:<input type="text" name="intensity"
value="<%=request.getParameter("intensity")%>" readonly>
    </td>
</tr>
<tr>
    <td colspan="3">
        <td colspan="1"> <br>
            In Silico ID:&nbsp;<input type="text" name="id2"
value="<%=request.getParameter("id2")%>" readonly>
        </td>
        <td colspan="2">
            MS/MS ID:&nbsp;<input type="text" name="id1" value=
<%=request.getParameter("id1")%> readonly>
        </td>
    </tr>
<tr>
    <td colspan="3">
        <td colspan="3" align = right>
            <input type=submit value="Reload">
        </td>
    </tr>
</table>
</form>
</body>
</html>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>MS/MS Graph Single Matches</title>
</head>
<body>
<center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
<form action="comparemstomssingleMatches.jsp">
<%
int rangeA, rangeB, topnum;
if (request.getParameter("rangeA") != "" &&
request.getParameter("rangeA") != null) {
rangeA = Integer.parseInt(request.getParameter("rangeA"));
} else {
rangeA = 4000;
}
if (request.getParameter("rangeB") != "" &&
request.getParameter("rangeB") != null) {
rangeB = Integer.parseInt(request.getParameter("rangeB"));
} else {
rangeB = 14000;
}
if (request.getParameter("topnum") != "" &&
request.getParameter("topnum") != null) {
topnum = Integer.parseInt(request.getParameter("topnum"));
} else {
topnum = 5;
}
%>
<table border="0" cellpadding = "0" cellspacing = "0" align=center >
<tr>
<td colspan="3">
&rangeA=<%=rangeA%>&r
angeB=<%=rangeB%>&topnum=<%=topnum%>">
</td>
</tr>
<tr>
<td colspan="3">
<%
dbBean.dbConnect();
String selstr = "";
selstr = "SELECT i.aa_seq, i.protein, CONCAT_WS(' ', g.name, sp.name,
sb.name, st.name) as org FROM in_silico i, strain s, genus g, species sp, subspecies
sb, subtype st WHERE s.strain_id = i.strain_id AND g.genus_id = s.genus_id AND
sp.species_id = s.species_id AND sb.subspecies_id = s.subspecies_id AND
st.subtype_id = s.subtype_id AND i.in_silico_id = " + request.getParameter("id2");
ResultSet myRS = dbBean.getRS(selstr);
if (myRS.next()) {
%>
<b>AA Seq:&nbsp;&nbsp;&nbsp;</b><%=myRS.getString("aa_seq")%><br>
<b>Protein:&nbsp;&nbsp;&nbsp;</b><%=myRS.getString("protein")%><br>
<b>Organism:&nbsp;&nbsp;&nbsp;</b><%=myRS.getString("org")%><br><br>
<%

```

```

    }
    myRS.close();
    dbBean.dbClose();
%>
</td>
</tr>
<tr>
<td colspan="3">
&id2=<%=request.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<%=request.getParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")%>&masserror=<%=request.getParameter("masserror")%>&residue=<%=request.getParameter("residue")%>&topnum=<%=request.getParameter("topnum")%>">
</td>
</tr>
<tr>
<td>
Low Value:&nbsp;<input type="text" name="rangeA" value="<%=rangeA%>">
</td>
<td>
High Value:&nbsp;<input type="text" name="rangeB"
value="<%=rangeB%>">
</td>
<td>
Number of Values Labeled: <input type="text" name="topnum"
value="<%=topnum%>">
</td>
</tr>
<tr>
<td>
Residue:&nbsp;<input type="text" name="residue"
value="<%=request.getParameter("residue")%>" readonly>
</td>
<td>
Mass Tolerance:<input type="text" name="masserror"
value="<%=request.getParameter("masserror")%>" readonly>
</td>
<td>
Intensity:<input type="text" name="intensity"
value="<%=request.getParameter("intensity")%>" readonly>
</td>
</tr>
<tr>
<td colspan="3">
<td colspan="1"> <br>
In Silico ID:&nbsp;<input type="text" name="id2"
value="<%=request.getParameter("id2")%>" readonly>
</td>
<td colspan="2">
MS/MS ID:&nbsp;<input type="text" name="id1" value=
<%=request.getParameter("id1")%> readonly>
</td>
</tr>
<tr>
<td colspan="3">
<td colspan="3" align = right>
<input type="submit" value="Reload">
</td>
</tr>
</table>
</form>
</body>
</html>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
  <meta http-equiv="content-type"
    content="text/html; charset=ISO-8859-1">
  <title>USDA: Produce Safety and Microbiology Strain Database</title>
</head>
<body alink="#000099" vlink="#990099" link="#000099"
  style="color: rgb(0, 0, 0); background-color: rgb(0, 0, 153);">
<table cellpadding="0" cellspacing="0" border="0"
  style="width: 800px; text-align: left; background-color: rgb(255, 255, 255);
margin-left: auto; margin-right: auto;">
  <tbody>
    <tr align="left">
      <td style="vertical-align: top; width: 70px;" rowspan="1"
        colspan="1">
        <table cellpadding="0" cellspacing="0" border="1"
          style="width: 100%; text-align: left;">
          <tbody>
            <tr>
              <td style="vertical-align: top;">
                <table cellpadding="0" cellspacing="0" border="0"
                  style="width: 800px; text-align: left; background-color: rgb(255, 255, 255);
margin-left: auto; margin-right: auto;">
                  <tbody>
                    <tr align="left">
                      <td style="vertical-align: top; width: 70px;"></td>
                      <td
style="vertical-align: middle; text-align: center;"><big><big><big><span
style="font-weight: bold;">Produce Safety and Microbiology<br>
<small><small>MALDI-TOF Strain Database</small></small><br>
</span></big></big></big> </td>
                    </tr>
                    <tr>
                      <td colspan="2">
                        <a href="home.jsp">New Search</a>
                        <a href="insertstrain.jsp">Insert New Strain</a>
                        <a href="./changepassword.jsp" >Change Password</a>
                        <a href="insertmaldidata.jsp">Add New MALDI-TOF Data</a>
                        <a href="comparemaldifile.jsp">Compare MALDI File</a>
                      </td>
                    </tr>
                  </tbody>
                </table>
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </tbody>
</table>
  <td style="vertical-align: top; background-color: rgb(0, 0, 153);"
    rowspan="1" colspan="1"><br>
  </td>
</tr>
</tbody>
</table>

```

```

        <td style="vertical-align: top;" rowspan="1" colspan="1">
        <table cellpadding="0" cellspacing="0" border="1"
style="width: 100%; text-align: left;">
        <tbody>
        <tr>
        <td style="vertical-align: top;">
        <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope= "session"/>

        <%

        int gtent, sptent, subtent, othertent;
        String cmpr = "1";

        if (request.getParameter("gtent") == null) {
            gtent = 0;
        } else {
            gtent = 1;
        }
        if (request.getParameter("sptent") == null) {
            sptent = 0;
        } else {
            sptent = 1;
        }
        if (request.getParameter("subtent") == null) {
            subtent = 0;
        } else {
            subtent = 1;
        }
        if (request.getParameter("othertent") == null) {
            othertent = 0;
        } else {
            othertent = 1;
        }
        dbBean.dbConnect();
        for (int i =0; i < Integer.parseInt(request.getParameter("numberofentries"));
i++) {
            int dbRtrn =
            dbBean.insertNewStrain(Integer.parseInt(request.getParameter("ident")), (Integer.pars
eInt(request.getParameter("number")) + i),
            request.getParameter("date"), Integer.parseInt(request.getParameter("genus")), Integer
.parseInt(request.getParameter("species")), Integer.parseInt(request.getParameter("su
bspecies")), Integer.parseInt(request.getParameter("subtype")), request.getParameter("
notes"), request.getParameter("history"), gtent ,sptent, subtent, othertent);
            %> <center> <%
            if ( dbRtrn ==0) {
                out.write(Integer.toString(i + 1) + " strain(s) successfully inserted.")
                out.write("Insert Successfull!");
            } else if ( dbRtrn == 3) {
                out.write(Integer.toString(i + 1) + " strain(s) successfully inserted.")
                out.write ("This identifier and number combination already exists,
please check your number again or use the update page to change this data.");
            } else{
                out.write(Integer.toString(i + 1) + " strain(s) successfully inserted.")
                out.write("Insert Failed, please press back and make sure that your data
is correct and that your login has not timed out. You may also not have permission
to insert new data.");
            }
            }
            dbBean.dbClose();
            %>
        </center>
    </td>
    </tr>
</tbody>
</table>

```

```
        </td>
      </tr>
    </tbody>
  </table>
  <br>
</body>
</html>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Update Strain",
"UpdateStrainTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
    %>
    <%

    int gtent, sptent, subtent, othertent;
    String cmp = "1";

    if (request.getParameter("gtent") == null) {
        gtent = 0;
    } else {
        gtent = 1;
    }
    if (request.getParameter("sptent") == null) {
        sptent = 0;
    } else {
        sptent = 1;
    }
    if (request.getParameter("subtent") == null) {
        subtent = 0;
    } else {
        subtent = 1;
    }
    if (request.getParameter("othertent") == null) {
        othertent = 0;
    } else {
        othertent = 1;
    }
    dbBean.dbConnect();

    if (
    dbBean.updateStrain(Integer.parseInt(request.getParameter("sampleid")), Integer.parse
Int(request.getParameter("ident")), Integer.parseInt(request.getParameter("number")),
request.getParameter("date"), Integer.parseInt(request.getParameter("genus")), Integer
.parseInt(request.getParameter("species")), Integer.parseInt(request.getParameter("su
bspecies")), Integer.parseInt(request.getParameter("subtype")), request.getParameter("
notes"), request.getParameter("history"), gtent, sptent, subtent, othertent) == 0) {
        out.write("Update Successful!");
    } else {
        out.write("Update Failed, please press back and make sure that your data is
correct and that your login has not timed out.");
    }
    dbBean.dbClose();
    out.write(pageSource.getPageFooter());
    %>

```



```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove In Silico Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE</i> In Silico Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                <%
                    if (request.getParameter("confirmationnum").compareTo("5105595691")
== 0 &&
request.getParameter("insilicoid").compareTo(request.getParameter("insilicoconfirm")
) == 0) {
                        dbBean.dbConnect();
                        dbBean.executeString("DELETE FROM in_silico_data WHERE in_silico_id
= " + request.getParameter("insilicoid"));
                        dbBean.executeString("DELETE FROM in_silico WHERE in_silico_id = " +
request.getParameter("insilicoid"));
                        out.write("Data Deleted");
                    } else {
                        out.write("Either the confirmation number, or the msms id entered
are incorrect.");
                    }
                %>
                dbBean.dbClose();
            <%>
            </td>
        </tr>
    </table>
    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove <i>In Silico</i>
Data", "InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form action="deleteinsilico.jsp">
    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE In Silico</i> Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                Are you sure you want to delete the In Silico data with the id:
            </td>
        </tr>
        <tr>
            <td colspan = 2 align=center>
                <input align=center type=text name=insilicoid
value='<%=request.getParameter("insilicoid")%>'>
            </td>
        </tr>
        <tr>
            <td>
                Enter Confirmation Number:
            </td>
            <td>
                <input type="text" name="confirmationnum">
            </td>
        </tr>
        <tr>
            <td>
                Enter MS/MS ID Number:
            </td>
            <td>
                <input type="text" name="insilicoconfirm">
            </td>
        </tr>
        <tr>
            <td colspan="2" align=center>
                <input type="submit" value="YES! DELETE This Data">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove MALDI-TOF Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE</i> MALDI-TOF- Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                <%
                    if (request.getParameter("confirmationnum").compareTo("5105595691")
== 0 &&
request.getParameter("maldiid").compareTo(request.getParameter("maldiconfirm")) ==
0) {
                        dbBean.dbConnect();
                        dbBean.executeString("DELETE FROM maldi_tof_data WHERE maldi_tof_id
= " + request.getParameter("maldiid"));
                        dbBean.executeString("DELETE FROM maldi_tof WHERE maldi_tof_id = " +
request.getParameter("maldiid"));
                        out.write("Data Deleted");
                    } else {
                        out.write("Either the confirmation number, or the maldi id entered
are incorrect.");
                    }
                    dbBean.dbClose();
                %>
            </td>
        </tr>
    </table>
    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove MALDI-TOF Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form action="deletemaldi.jsp">
    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE</i> MALDI-TOF Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                Are you sure you want to delete the MALDI-TOF data with the id:
            </td>
        </tr>
        <tr>
            <td colspan = 2 align=center>
                <input align=center type=text name=maldiid
value='<%=request.getParameter("maldiid")%>'>
            </td>
        </tr>
        <tr>
            <td>
                Enter Confirmation Number:
            </td>
            <td>
                <input type="text" name="confirmationnum">
            </td>
        </tr>
        <tr>
            <td>
                Enter MALDI-TOF ID Number:
            </td>
            <td>
                <input type="text" name="maldiconfirm">
            </td>
        </tr>
        <tr>
            <td colspan="2" align=center>
                <input type="submit" value="YES! DELETE This Data">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove MS/MS Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE</i> MS/MS Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                <%
                    if (request.getParameter("confirmationnum").compareTo("5105595691")
== 0 &&
request.getParameter("msmsid").compareTo(request.getParameter("msmsconfirm")) == 0)
{
                        dbBean.dbConnect();
                        dbBean.executeUpdate("DELETE FROM ms_ms_data WHERE ms_ms_id = " +
request.getParameter("msmsid"));
                        dbBean.executeUpdate("DELETE FROM ms_ms WHERE ms_ms_id = " +
request.getParameter("msmsid"));
                        out.write("Data Deleted");
                    } else {
                        out.write("Either the confirmation number, or the msms id entered
are incorrect.");
                    }
                }
                dbBean.dbClose();
            %>
        </td>
    </tr>
</table>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove MS/MS Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form action="deletemsms.jsp">
    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE</i> MS/MS Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                Are you sure you want to delete the MS/MS data with the id:
            </td>
        </tr>
        <tr>
            <td colspan = 2 align=center>
                <input align=center type=text name=msmsid
value='<%=request.getParameter("msmsid")%>'>
            </td>
        </tr>
        <tr>
            <td>
                Enter Confirmation Number:
            </td>
            <td>
                <input type="text" name="confirmationnum">
            </td>
        </tr>
        <tr>
            <td>
                Enter MS/MS ID Number:
            </td>
            <td>
                <input type="text" name="msmsconfirm">
            </td>
        </tr>
        <tr>
            <td colspan="2" align=center>
                <input type="submit" value="YES! DELETE This Data">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove Sample Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE</i> Strain Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                <%
                    if
(request.getParameter("confirmationnum").compareToIgnoreCase("YES") == 0 &&
request.getParameter("sampleid").compareTo(request.getParameter("sampleconfirm")) ==
0) {
                        dbBean.dbConnect();
                        dbBean.executeUpdate("DELETE FROM sample WHERE sample.sample_id = "
+ request.getParameter("sampleid"));
                        dbBean.executeUpdate("DELETE FROM maldi_tof, maldi_tof_data WHERE
maldi_tof_data.maldi_tof_id = maldi_tof.maldi_tof_id AND maldi_tof.sample_id = " +
request.getParameter("sampleid"));
                        out.write("Data Deleted");
                    } else {
                        out.write("Either the confirmation, or the maldi id entered are
incorrect.");
                    }
                    dbBean.dbClose();
                %>
            </td>
        </tr>
    </table>
    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove Sample Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form action="deletesample.jsp">
    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE</i> Sample Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                Are you sure you want to delete the sample data with the id:
            </td>
        </tr>
        <tr>
            <td colspan = 2 align=center>
                <input align=center type=text name=sampleid
value='<%=request.getParameter("sampleid")%>'>
            </td>
        </tr>
        <tr>
            <td>
                Enter Yes to confirm you want to delete this data:
            </td>
            <td>
                <input type="text" name="confirmationnum">
            </td>
        </tr>
        <tr>
            <td>
                Enter sample ID Number:
            </td>
            <td>
                <input type="text" name="sampleconfirm">
            </td>
        </tr>
        <tr>
            <td colspan="2" align=center>
                <input type="submit" value="YES! DELETE This Data">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```



```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
dbBean.dbConnect();
pageSource.splitTemplatePage(dbBean.getUsername(), "Remove MALDI-TOF Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
out.write(pageSource.getPageHeader());
dbBean.dbClose();
%>

<form action="deletesamplerange.jsp">
<table align = center>
<tr>
<td colspan = 2>
<center><h1><i>DELETE</i> Sample Data</h1></center>
</td>
</tr>
<tr>
<td colspan=2>
Select The Range of RM Numbers to Delete
</td>
</tr>
<tr>
<td colspan = 2 align=center>
<input align=right type=text name=lowrange
value=''>&nbsp;To:&nbsp;
<input align=left type=text name=highbound value=''>
</td>
</tr>
<tr>
<td>
Do you want to delete the above sample data? (Type yes in the
box):
</td>
<td>
<input type="text" name="confirmationnum">
</td>
</tr>
<tr>
<td colspan="2" align=center>
<input type="submit" value="YES! DELETE This Data">
</td>
</tr>
</table>
</form>
<%out.write(pageSource.getPageFooter());%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Remove Sample Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1><i>DELETE</i> Strain Data</h1></center>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                <%
                    if
(request.getParameter("confirmationnum").compareToIgnoreCase("YES") == 0) {
                        dbBean.dbConnect();
                        dbBean.executeUpdateString("DELETE FROM sample USING sample,
identifier WHERE sample.identifier_id = identifier.identifier_id AND
identifier.ident='RM' AND sample.number BETWEEN " + request.getParameter("lowrange")
+ " AND " + request.getParameter("highrange"));
                        out.write("Data Deleted");
                    } else {
                        out.write("You did not enter Yes in the confirmation text box.");
                    }
                    dbBean.dbClose();
                %>
            </td>
        </tr>
    </table>
    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text"%><%@page import="java.sql.*" %><%@page
import="java.util.*" %><jsp:useBean id="dbBean" class="psm.db.dbAccess"
scope="session"/>
<%int validlogin = dbBean.dbConnect(); %>
<% if (validlogin ==0 & (dbBean.getUsername().compareTo("rmandrell") == 0 ||
dbBean.getUsername().compareTo("psmlogin") == 0) == true) {
    ResultSet myRS;
    boolean rmonly;
    if (request.getParameter("rmonly") != null) {
        if (request.getParameter("rmonly").compareTo("yes") == 0) {
            rmonly = true;
        } else {
            rmonly = false;
        }
    } else {
        rmonly = false;
    }
    if (rmonly) {
        myRS = dbBean.getRS("SELECT sample.sample_id, CONCAT(identifier.ident,
sample.number) ident, sample.date, genus.name genus, species.name species,
subspecies.name subspecies, subtype.name subtype, sample.notes, sample.history,
sample.tentative_genus, sample.tentative_species, sample.tentative_subspecies,
sample.tentative_subtype FROM identifier, sample, strain, genus, species,
subspecies, subtype WHERE strain.genus_id = genus.genus_id AND strain.species_id =
species.species_id AND subspecies.subspecies_id = strain.subspecies_id AND
subtype.subtype_id = strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id and identifier.identifier_id = 1");
    } else {
        String strandor = request.getParameter("andor");
        int maldionly;
        if (request.getParameter("maldionly") == null) {
            maldionly = 0;
        } else {
            maldionly = 1;
        }
        if (strandor.compareTo("Or") == 0 ) {
            myRS = dbBean.runORSearch(request.getParameter("sampleid"),
request.getParameter("identifier"), request.getParameter("genus"),
request.getParameter("species"), request.getParameter("subspecies"),
request.getParameter("other"), request.getParameter("notes"),
request.getParameter("history"), request.getParameter("gtent"),
request.getParameter("sptent"), request.getParameter("subtent"),
request.getParameter("othertent"), maldionly, request.getParameter("lowRM"),
request.getParameter("highRM"));
            //java.sql.Statement temps = myRS.getStatement();
            //String s;
        } else {
            myRS = dbBean.runANDSearch(request.getParameter("sampleid"),
request.getParameter("identifier"), request.getParameter("genus"),
request.getParameter("species"), request.getParameter("subspecies"),
request.getParameter("other"), request.getParameter("notes"),
request.getParameter("history"), request.getParameter("gtent"),
request.getParameter("sptent"), request.getParameter("subtent"),
request.getParameter("othertent"), maldionly, request.getParameter("mass"),
Integer.parseInt(request.getParameter("masserror")),
request.getParameter("minintensity"), request.getParameter("lowRM"),
request.getParameter("highRM"));
        }
    }
    //write the header/column names
    out.write("Sample Number\tDate\tName\tNotes\tHistory\n");
    while (myRS.next()) {
        String notes = myRS.getString("notes");
        String hist = myRS.getString("history");
        if (notes != null) {

```

```

notes = notes.replace("\\"", "\\\"");
notes = notes.replaceAll("\r\n|\n|\t", "");
}else {
    notes="";
}
if (hist != null) {
hist = hist.replace("\\"", "\\\"");
hist = hist.replaceAll("\r\n|\n", "");

} else {
    hist = "";
}
String qG = "", qS = "", qSS = "", qO = "";
if (myRS.getInt("tentative_genus") == 1) {
    qG = "?";
}
if (myRS.getInt("tentative_species") == 1) {
    qS = "?";
}
if (myRS.getInt("tentative_subspecies") == 1) {
    qSS = "?";
}
if (myRS.getInt("tentative_subtype") == 1) {
    qO = "?";
}

String lname = myRS.getString("genus") + qG + " " +
myRS.getString("species") + qS + " " + myRS.getString("subspecies") + qSS + " " +
myRS.getString("subtype") + qO;

lname = lname.replace("null", "");
lname = lname.trim();
String line;
//RUMm number
line = myRS.getString("ident") + "\t";
//date
line += "\"" + myRS.getString("date") + "\"\t";
//name
line += "\"" + lname + "\"\t";
line += "\"" + notes + "\"\t";
line += "\"" + hist + "\"\n";
out.write(line);
}
myRS.close();
}
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Find And Replace",
"FindAndReplaceTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>

    <form action="findandreplace.jsp">

    <%
        int genus = 0, species=0, subspecies=0, other=0;
        String rdate = "";
        if (request.getParameter("genusr") != null) {
            genus = Integer.parseInt(request.getParameter("genus"));
        }
        if (request.getParameter("spr") != null) {
            species = Integer.parseInt(request.getParameter("species"));
        }
        if (request.getParameter("subspr") != null) {
            subspecies = Integer.parseInt(request.getParameter("subspecies"));
        }
        if (request.getParameter("other") != null) {
            other = Integer.parseInt(request.getParameter("subtype"));
        }
        if (request.getParameter("dater") != null) {
            rdate = request.getParameter("date");
        }
        if (request.getParameter("submit") != null) {
            boolean noerror = true;
            if (request.getParameter("lRange") == null) {
                out.write("Please enter a low range value.");
                noerror = false;
            }

            if (request.getParameter("hRange") == null) {
                out.write("Please enter a high range value.");
                noerror = false;
            }

            if (request.getParameter("replace") == null) {
                out.write("Please enter a search/replace value");
                noerror = false;
            }

            if (request.getParameter("with") == null) {
                out.write("Please enter a replacement value.");
                noerror = false;
            }
            String mChar = "";
            int mSInt = 0;

            if (noerror) {
                dbBean.dbConnect();
                if (request.getParameter("inChar") != null &&
request.getParameter("inChar") != "") {
                    mChar = request.getParameter("inChar");
                }
                if (request.getParameter("inStart") != null &&

```

```

request.getParameter("inStart") != "") {
    mSInt = Integer.parseInt(request.getParameter("inStart"));
    }
    int num =
dbBean.findAndReplace(Integer.parseInt(request.getParameter("lRange")),
Integer.parseInt(request.getParameter("hRange")), request.getParameter("replace"),
request.getParameter("with"), Integer.parseInt(request.getParameter("ident")),mChar,
mSInt, genus,species,subspecies,other, rdate);
    out.write(num + " rows have been updated.");
    dbBean.dbClose();
}

} else if(request.getParameter("Prepend") != null |
request.getParameter("Append") != null) {
    boolean noerror=true;
    if (request.getParameter("with") == null) {
        out.write("Please enter a replacement value.");
        noerror = false;
    }
    if (request.getParameter("lRange") == null) {
        out.write("Please enter a low range value.");
        noerror = false;
    }
    if (request.getParameter("hRange") == null) {
        out.write("Please enter a high range value.");
        noerror = false;
    }
    if (noerror) {
        dbBean.dbConnect();

        String mChar = "";
        int mSInt = 0;

        if (request.getParameter("inChar") != null &&
request.getParameter("inChar") != "") {
            mChar = request.getParameter("inChar");
        }
        if (request.getParameter("inStart") != null &&
request.getParameter("inStart") != "") {
            mSInt = Integer.parseInt(request.getParameter("inStart"));
        }
        if (request.getParameter("Append") !=null) {
            int num =
dbBean.findAndReplaceAppend(Integer.parseInt(request.getParameter("lRange")),
Integer.parseInt(request.getParameter("hRange")), request.getParameter("with"),
Integer.parseInt(request.getParameter("ident")),mChar, mSInt, 0, genus, species,
subspecies, other, rdate);
            out.write(num + " rows have been updated.");
        } else {
            int num =
dbBean.findAndReplaceAppend(Integer.parseInt(request.getParameter("lRange")),
Integer.parseInt(request.getParameter("hRange")), request.getParameter("with"),
Integer.parseInt(request.getParameter("ident")),mChar, mSInt, 1, genus, species,
subspecies, other, rdate);
            out.write(num + " rows have been updated.");
        }
        dbBean.dbClose();
    }
}
}
%>

<table align=center>
<tr>
<td colspan=2>

```

```

        <center><h1>Find and Replace</h1></center>
    </td>
</tr>
<tr>
<td>
    Identifier (RM):&nbsp;
</td>
<td>
        <select name="ident">
            <%
                dbBean.dbConnect();
                ResultSet myRS = dbBean.getIdentPrefix();
                if (myRS.next()) {
                    myRS.first();
                    do {
                        out.write("<option value=" +
myRS.getInt("identifier_id") + ">" + myRS.getString("ident"));
                    } while (myRS.next());
                }
                myRS.close();
            %>
        </select>
    </td>
</tr>
<tr>
    <td>
        Range:
    </td>
    <td>
        <input type="text" name="lRange" size=5>&nbsp;To:
&nbsp;<input type="text" name="hRange" size= 5>
    </td>
</tr>
<tr>
    <td>
        Replace:
    </td>
    <td>
        <input type="text" name="replace">
    </td>
</tr>
<tr>
    <td>
        Genus:
    </td>
    <td>
        <select name="genus">
            <%
                myRS = dbBean.getGenus();
                if (myRS.next()) {
                    myRS.first();
                    do {
                        String cmpr = request.getParameter("genus");
                        if (cmpr == null) { cmpr = ""; }
                        if (myRS.getString("name")!=null) {
                            if
(myRS.compareTo(myRS.getString("genus_id")) == 0) {
                                out.write("<option value='" +
myRS.getInt("genus_id") + "' selected>" + myRS.getString("name"));
                            } else {
                                out.write("<option value='" +
myRS.getInt("genus_id") + "'>" + myRS.getString("name"));
                            }
                        } else {
                            if (cmpr.compareTo("")==0) {

```

```

                                out.write("<option value='" +
myRS.getInt("genus_id") + "' selected>No Genus");
                                } else {
                                out.write("<option value='" +
myRS.getInt("genus_id") + "'>No Genus");
                                }
                                } while (myRS.next());
                                }
                                myRS.close();
                                %>
                                </select><input type='checkbox' name='genusr' value='1'
unchecked>Replace
                                </td>
                                </tr>
                                <tr>
                                <td>
                                Species:
                                </td>
                                <td>
                                <select name="species">
                                <%
                                myRS = dbBean.getSpecies();
                                if (myRS.next()) {
                                myRS.first();
                                do {
                                String cmpr = request.getParameter("species");
                                if (cmpr == null) { cmpr = ""; }
                                if (myRS.getString("name")!=null) {
                                if (cmpr.compareTo(myRS.getString("species_id")) ==
0) {
                                out.write("<option value='" +
myRS.getInt("species_id") + "' selected>" + myRS.getString("name"));
                                } else {
                                out.write("<option value='" +
myRS.getInt("species_id") + "'>" + myRS.getString("name"));
                                }
                                } else {
                                if (cmpr.compareTo("")==0) {
                                out.write("<option value='" +
myRS.getInt("species_id") + "' selected>No Species");
                                } else {
                                out.write("<option value='" +
myRS.getInt("species_id") + "'>No Species");
                                }
                                }
                                } while (myRS.next());
                                }
                                myRS.close();
                                %>
                                </select><input type='checkbox' name='spr' value='1'
unchecked>Replace
                                </td>
                                </tr>
                                <tr>
                                <td>
                                Subspecies:
                                </td>
                                <td>
                                <select name="subspecies">
                                <%
                                myRS = dbBean.getSubSpecies();
                                if (myRS.next()) {
                                myRS.first();
                                do {

```



```

String cmpr = request.getParameter("subspecies");
if (cmpr == null) { cmpr = "";}
if (myRS.getString("name")!=null) {
    if (cmpr.compareTo(myRS.getString("subspecies_id"))
== 0) {
        out.write("<option value='" +
myRS.getInt("subspecies_id") + "' selected>" + myRS.getString("name"));
    } else {
        out.write("<option value='" +
myRS.getInt("subspecies_id") + "'>" + myRS.getString("name"));
    }
} else {
    if (cmpr.compareTo("")==0 ) {
        out.write("<option value='" +
myRS.getInt("subspecies_id") + "' selected>No Subspecies");
    } else {
        out.write("<option value='" +
myRS.getInt("subspecies_id") + "'>No Subspecies");
    }
}
} while (myRS.next());
}
myRS.close();
%>
</select><input type='checkbox' name='subspr' value='1'
unchecked>Replace
</td>
</tr>
<tr>
<td>
Serovar/Subtype/Genotype:
</td>
<td>
<select name="subtype">
<%
myRS = dbBean.getSubType();
if (myRS.next()) {
    myRS.first();
    do {
        String cmpr = request.getParameter("subtype");
        if (cmpr == null) { cmpr = "";}
        if (myRS.getString("name")!=null) {
            if (cmpr.compareTo(myRS.getString("subtype_id")) ==
0) {
                out.write("<option value='" +
myRS.getInt("subtype_id") + "' selected>" + myRS.getString("name"));
            } else {
                out.write("<option value='" +
myRS.getInt("subtype_id") + "'>" + myRS.getString("name"));
            }
        } else {
            if (cmpr.compareTo("")==0) {
                out.write("<option value='" +
myRS.getInt("subtype_id") + "' selected>No Subtype");
            } else {
                out.write("<option value='" +
myRS.getInt("subtype_id") + "'>No Subtype");
            }
        }
    } while (myRS.next());
}
myRS.close();
%>
</select><input type='checkbox' name='other' value='1'
unchecked>Replace

```

```

        </td>
    </tr>
    <tr>
        <td>
            Date (2005-8-24):
        </td>
        <td>
            <input type="text" name="date"><input type='checkbox'
name='dater' value='1' unchecked>Replace
        </td>
    </tr>
    <tr>
        <td>
            With (comma delimited):
        </td>
        <td>
            <input type="text" name="with">
        </td>
    </tr>
    <tr>
        <td>
            Auto-Increment Character:
        </td>
        <td>
            <input type="text" name="inChar" size=5>
            Start:
            <input type="text" name="inStart" size=5>
        </td>
    </tr>
    <tr>
        <td colspan="2" style="text-align: center;"><br>
        <input type="submit" value="Prepend" name="Prepend">
        <input type="submit" value="Append" name="Append">
        <input type="submit" value="Replace" name="submit">
        </td>
    </tr>
</table>
</form>

<% out.write(pageSource.getPageFooter());
    dbBean.dbClose(); %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<%@page import="java.sql.*" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<%
//Becuase of the nature of this search the majority of the code will be processed
here instead of in a bean
String searchSQL = "SELECT locus_id";
if (request.getParameter("locus") != null &&
request.getParameter("locus").compareTo("on") == 0) {
    searchSQL += ", locus.name 'Locus'";
}
if (request.getParameter("strain") != null &&
request.getParameter("strain").compareTo("on") == 0) {
    searchSQL += ", CONCAT(identifider.name, sample.number) 'Strain'";
}
if (request.getParameter("genusspecies") != null &&
request.getParameter("genusspecies").compareTo("on") == 0) {
    searchSQL += ", CONCAT(genus.name, ' ', species.name) 'genus-species'";
}
if (request.getParameter("location1") != null &&
request.getParameter("location1").compareTo("on") == 0) {
    searchSQL += ", a.name 'Location1'";
}
if (request.getParameter("location2") != null &&
request.getParameter("location2").compareTo("on") == 0) {
    searchSQL += ", b.name 'Location2'";
}
if (request.getParameter("start") != null &&
request.getParameter("start").compareTo("on") == 0) {
    searchSQL += ", locus.start 'Start'";
}
if (request.getParameter("end") != null &&
request.getParameter("end").compareTo("on") == 0) {
    searchSQL += ", locus.end 'End'";
}
if (request.getParameter("GBK") != null &&
request.getParameter("GBK").compareTo("on") == 0) {
    searchSQL += ", CASE locus.FR WHEN 'F' THEN CONCAT(locus.start, '..', locus.end) WHEN 'R' THEN CONCAT('complement(', locus.start, '..', locus.end, ')') End 'GBK'";
}
if (request.getParameter("fr") != null &&
request.getParameter("fr").compareTo("on") == 0) {
    searchSQL += ", locus.FR";
}
if (request.getParameter("NT") != null &&
request.getParameter("NT").compareTo("on") == 0) {
    searchSQL += ", locus.NT";
}
if (request.getParameter("AA") != null &&
request.getParameter("AA").compareTo("on") == 0) {
    searchSQL += ", locus.AA";
}
if (request.getParameter("MW") != null &&
request.getParameter("MW").compareTo("on") == 0) {
    searchSQL += ", locus.MW";
}
if (request.getParameter("iep") != null &&
request.getParameter("iep").compareTo("on") == 0) {
    searchSQL += ", locus.IEP";
}

```

```

    }
    if (request.getParameter("gene") != null &&
request.getParameter("gene").compareTo("on") == 0) {
        searchSQL += ", locus.gene 'Gene'";
    }
    if (request.getParameter("code") != null &&
request.getParameter("code").compareTo("on") == 0) {
        searchSQL += ", locus.code 'Code'";
    }
    if (request.getParameter("COG") != null &&
request.getParameter("COG").compareTo("on") == 0) {
        searchSQL += ", locus.cog 'COG'";
    }
    if (request.getParameter("ec1") != null &&
request.getParameter("ec1").compareTo("on") == 0) {
        searchSQL += ", locus.EC1 'EC#1'";
    }
    if (request.getParameter("ec2") != null &&
request.getParameter("ec2").compareTo("on") == 0) {
        searchSQL += ", locus.EC2 'EC#2'";
    }
    if (request.getParameter("product") != null &&
request.getParameter("product").compareTo("on") == 0) {
        searchSQL += ", locus.product 'Product'";
    }
    if (request.getParameter("notes") != null &&
request.getParameter("notes").compareTo("on") == 0) {
        searchSQL += ", locus.notes 'Notes'";
    }
}

//Add the default WHERE clause necessary to retrieve the real data not just the
IDS
searchSQL += " FROM locus,location a, location b, sample, identifier, strain,
genus, species WHERE locus.location_id = a.location_id AND locus.location2_id =
b.location_id AND locus.sample_id = sample.sample_id AND sample.identifier_id =
identifier.identifier_id AND sample.strain_id = strain.strain_id AND strain.genus_id
= genus.genus_id AND strain.species_id = species.species_id";

//Now add to the where clause the search criteria
if (request.getParameter("locusstr") != null && request.getParameter("locusstr")
!= "") {
    searchSQL += " AND locus.name LIKE '%" + request.getParameter("locusstr") +
"%'";
}
if (request.getParameter("strainstr") != null &&
request.getParameter("strainstr") != "") {
    searchSQL += " AND CONCAT(identifier.name, sample.number) LIKE '%" +
request.getParameter("strainstr") + "%'";
}
if (request.getParameter("genusstr") != null && request.getParameter("genusstr")
!= "") {
    searchSQL += " AND genus.name LIKE '%" + request.getParameter("genusstr") +
"%'";
}
if (request.getParameter("speciesstr") != null &&
request.getParameter("speciesstr") != "") {
    searchSQL += " AND species.name LIKE '%" + request.getParameter("speciesstr")
+ "%'";
}
if (request.getParameter("locationstr") != null &&
request.getParameter("locationstr") != "") {
    searchSQL += " AND a.name LIKE '%" + request.getParameter("locationstr") +
"%'";
}
if (request.getParameter("location2str") != null &&

```

```

request.getParameter("location2str") != "") {
    searchSQL += " AND b.name LIKE '%" + request.getParameter("location2str") +
    "%'";
}
if (request.getParameter("productstr") != null &&
request.getParameter("productstr") != "") {
    searchSQL += " AND locus.Product LIKE '%" +
request.getParameter("productstr") + "%'";
}
if (request.getParameter("notesstr") != null && request.getParameter("notesstr")
!= "") {
    searchSQL += " AND locus.Notes LIKE '%" + request.getParameter("notestr") +
    "%'";
}
if (request.getParameter("mw") != null && request.getParameter("mw") != "") {
    searchSQL += " AND locus.MW between " + request.getParameter("mw") + " - (" +
request.getParameter("mw") + " * " + request.getParameter("ppm") + " / 1000000.0)
AND " + request.getParameter("mw") + " + (" + request.getParameter("mw") + " * " +
request.getParameter("ppm") + " / 1000000.0)";
}
if (request.getParameter("genestr") != null && request.getParameter("genestr") !=
"") {
    searchSQL += " AND locus.gene LIKE '%" + request.getParameter("genestr") +
    "%'";
}
dbBean.dbConnect("genelist");
ResultSet myRS = dbBean.getRS(searchSQL);
ResultSetMetaData myMD = myRS.getMetaData();
%>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>USDA Produce Safety and Microbiology Database</title>
</head>
<body style="background-color: rgb(54, 171, 54);">
<table cellpadding=0 cellspacing=0 align=center>
    <tr>
    <%
        for (int i=1; i <= myMD.getColumnCount() -1; i++) {
            out.write("<th bgcolor = #008700><span style='color: rgb(231,229, 38);'>"
+ myMD.getColumnName(i + 1) + "</th></span>");
        }
    %>
    <%
        String mColor;
        int count = 0;
        while (myRS.next()) {
            out.write("<tr>");
            if (count % 2 == 0) {
                mColor= "#ffd0ae";
            }else {
                mColor = "#ecac7d";
            }
            for (int i=1; i <= myMD.getColumnCount() -1; i++) {
                if (myRS.getString(i+1) != null) {
                    if (myMD.getColumnName(i+1).compareTo("Locus") == 0) {
                        out.write("<td style='vertical-align: top; background-color:
" + mColor + "'><span style='color: rgb(0,0,0);font-weight: bold;'>&nbsp;<a
href='locusdetails.jsp?locusid=" + myRS.getString(1) + "'>" + myRS.getString(i+ 1) +
"</a>&nbsp;</td></span>");
                    } else if (myMD.getColumnName(i+1).compareTo("Strain") == 0) {
                        out.write("<td style='vertical-align: top; background-color:
" + mColor + "'><span style='color: rgb(0,0,0);font-weight: bold;'>&nbsp;<a
href='searchresults.jsp?andor=Or&lowRM=&highRM=&genus=&species=&subspecies=&other=&

```

```

otes=" + myRS.getString(i + 1) +
"&history=&gtent=Any&sptent=Any&subtent=Any&othertent=Any&sampleid=&mass=&identifier
=" + myRS.getString(i + 1) + "'>" + myRS.getString(i+ 1) +
"</a>&nbsp;</td></span>");
    } else{
        out.write("<td style='vertical-align: top; background-color:
" + mColor + "'><span style='color: rgb(0,0,0);font-weight: bold;'>&nbsp;</span>" +
myRS.getString(i+ 1) + "&nbsp;</td></span>");
    }
    } else {
        out.write("<td style='vertical-align: top; background-color: " +
mColor + "'><span style='color: rgb(0,0,0);font-weight: bold;'></td></span>");
    }
    }
    count++;
    if (count == 2000) {myRS = null;}
    out.write("</tr>");
}
}%>
</table>
</body>
</html>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Genus",
"InsertGenusTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
    %>
    <%
    dbBean.dbConnect();
    int rtn;
    rtn = dbBean.insertNewGenus(request.getParameter("name"));
    if (rtn > 0) {
        out.write("Insert successfull!");
    } else if (rtn == -1) {
        out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
    }
    dbBean.dbClose();
    %>
    <br><center><a href="insertstrain.jsp">Insert New Strain</a></center>
    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:setProperty name="dbBean" property="*" />
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%int validlogin = dbBean.dbConnect(); %>

<%
if (validlogin ==0) {
    pageSource.splitTemplatePage(dbBean.getUsername(), "Search Strain Data",
    "StrainSearchTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
}
    if (validlogin != 0) {%>
        <center>The username/password combination you have entered does not
exist or is no longer valid. Please press back on your browser and try
again.</center>
        <%} else { dbBean.dbClose(); %>

        <table border="0" cellpadding=0 cellspacing=0>
            <tr>

                <form action="searchresults.jsp">
                <br>
                <table align=center>
                    <tr>
                        <td colspan="2">
                            <center><b>Enter the parameters for your
search:</b></center><br>
                        </td>
                    </tr>
                    <tr>
                        <td colspan=2 align = center>
                            <input type="radio" name="andor" value="and" checked>AND
                            <input type="radio" name="andor" value="Or">OR
                        </td>
                    </tr>
                    <tr>
                        <td>
                            Search by Identification Number (e.g. RM1203 or 1203):
                        </td>
                        <td>
                            <input type="text" name="identifier">
                        </td>
                    </tr>
                    <tr>
                        <td>
                            Within Range (e.g. 1203 To: 1403)
                        </td>
                        <td>
                            <input type="text" name="lowRM" size="5">&nbsp; To:
                            &nbsp;<input type="text" name="highRM" size="5">
                        </td>
                    </tr>
                    <tr>
                        <td>
                            Search by Genus (e.g. Campylobacter):
                        </td>
                        <td>
                            <input type="text" name="genus">
                        </td>
                    </tr>
                    <tr>
                        <td>

```


Search by Species (e.g. Coli):
<input name="species" type="text"/>

Search by Subspecies (e.g. Fetus):
<input name="subspecies" type="text"/>

Search by Other Subtype:
<input name="other" type="text"/>

Search Inside Notes:
<input name="notes" type="text"/>

Search Inside History:
<input name="history" type="text"/>

Search Genus Tentativity:
<input checked="" name="gtent" type="radio" value="Any"/> <input name="gtent" type="radio" value="true"/> True <input name="gtent" type="radio" value="false"/> False

Search Species Tentativity:
<input checked="" name="sptent" type="radio" value="Any"/> <input name="sptent" type="radio" value="true"/> True <input name="sptent" type="radio" value="false"/> False

Search Subspecies Tentativity:
<input checked="" name="subtent" type="radio" value="Any"/> <input name="subtent" type="radio" value="true"/> True

```

        <input type=radio name="subtent" value="false">False
    </td>
</tr>
    <tr>
        <td>
            Search Serovar/Subtype/Genotype Tentativity:
        </td>
        <td>
            <input type=radio name="othertent" value="Any" checked>
            <input type=radio name="othertent" value="true">True
            <input type=radio name="othertent" value="false">False
        </td>
    </tr>
</tr>
<tr>
    <td>
        Search by Sample ID (e.g. 321):
    </td>
    <td>
        <input type=text name="sampleid">
    </td>
</tr>
<tr>
    <td colspan=2 align=center>
        <span style="Font-Weight: Bold;">For AND seach only:</span>
    </td>
</tr>
<tr>
    <td>
        Search by mass:
    </td>
    <td>
        <input type=text" name="mass">
    </td>
</tr>
<tr>
    <td>
        Mass Error (ppm):
    </td>
    <td>
        <input type=text" name="masserror" value="500">
    </td>
</tr>
<tr>
    <td>
        Minimum Intensity:
    </td>
    <td>
        <input type=text" name="minintensity" value="10">
    </td>
</tr>
<tr>
    <td>
        Only Display Results With MALDI-TOF Data:<br>(This will
        Lengthen Search Time, Please Be Patient)
    </td>
    <td>
        <input type=checkbox name="maldionly" value="1">Yes
    </td>
</tr>
<tr>
    <td colspan = "2" align=right>
        <input type="reset">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input
        type=submit>
    </td>
</tr>

```

```
                <tr>
                </tr>
                <tr><td>&nbsp;</td></tr>
            </table>
            <br>
        </form>
    <%
    }
    if (validlogin ==0) {
        out.write(pageSource.getPageFooter());
    }
    %> <%-- Close out the else statement --%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Identifier",
"InsertIdentifierTitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <%
        dbBean.dbConnect();
        int rtn;
        rtn = dbBean.insertNewIdentifier(request.getParameter("name"),
request.getParameter("notes"));
        if (rtn == 3) {
            out.write("This identifier is already present in the database.");
        } else if (rtn !=0) {
            out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
        } else if (rtn ==0 ) {
            out.write("Insert successfull!");
        }
        dbBean.dbClose();
    %>
    <br><center><a href="insertstrain.jsp">Insert New Strain</a></center>

    <%out.write(pageSource.getPageFooter());%>

```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<jsp:useBean id="importCSV" class="psm.mformat.importStrain"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subtype",
"InsertSubtypeTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());

    if (importCSV.parseandImportStrain(request,dbBean) == 0) {
        out.write("Update Successful!");
    } else {
        out.write("An error occurred please try again.");
    }
%>
<%
dbBean.dbClose();
out.write(pageSource.getPageFooter());%>
```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<jsp:useBean id="importCSV" class="psm.mformat.importStrain"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subtype",
"InsertSubtypeTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());

    if (importCSV.parseandImportNoteData(request,dbBean) == 0) {
        out.write("Update Successful!");
    } else {
        out.write("An error occurred please try again.");
    }
%>
<%
dbBean.dbClose();
out.write(pageSource.getPageFooter());%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subtype",
"InsertSubtypeTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
    <center><h1>Import Comma Delimited File</h1></center>
    <form method=post action="importconfirm.jsp" enctype="MULTIPART/FORM-DATA">
        <table align=center>
            <tr>
                <td colspan=2>
                    <center>You can upload a tab delimited file with the following
format:<br>
                    StrainNumber    Date    Genus    Species    Subspecies
Subtype    Notes    History<br>
                    Make sure to strip out any header information, the easiest way
to make a tab delimited file is to use excel and save as text tab delimited
file.</center><br>
                </td>
            </tr>
            <tr>
                <td colspan=2>
                    <center>Choose which fields you want changed/updated from this
file</center>
                </td>
            </tr>
            <tr>
                <td>
                    <input type=checkbox name="date">
                </td>
                <td>
                    Date
                </td>
            </tr>
            <tr>
                <td>
                    <input type=checkbox name="Genus">
                </td>
                <td>
                    Genus
                </td>
            </tr>
            <tr>
                <td>
                    <input type=checkbox name="Species">
                </td>
                <td>
                    Species
                </td>
            </tr>
            <tr>
                <td>
                    <input type=checkbox name="Subspecies">
                </td>
                <td>
                    Subspecies
                </td>
            </tr>
        </table>
    </form>

```

```

        <td>
        <input type=checkbox name="Subtype">
        </td>
        <td>
        Subtype
        </td>
    </tr>
    <tr>
        <td>
        <input type=checkbox name="notes" checked>
        </td>
        <td>
        Notes
        </td>
    </tr>
    <tr>
        <td>
        <input type=checkbox name="History" checked>
        </td>
        <td>
        History
        </td>
    </tr>
    <tr>
        <td>
        <input type=checkbox name="append" checked>
        </td>
        <td>
        Append notes and history Data to existing data<br>(Note otherwise
all fields will be replaced)
        </td>
    </tr>
    <tr>
        <td colspan=2>
        &nbsp;
        </td>
    </tr>
    <tr>
        <td>
        File:
        </td>
        <td>
        <input type="file" name="csvfile">
        </td>
    </tr>
    <tr>
        <td colspan="2" align="right">
        <input type="submit" value="Upload File">
        </td>
    </tr>
</table>
</form>

<%out.write(pageSource.getPageFooter());%>

```



```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subtype",
"InsertSubtypeTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
    %>
    <center><h1>Import Tab Delimited File</h1></center>
    <form method=post action="importconfirmnotes.jsp" enctype="MULTIPART/FORM-DATA">
        <table align=center>
            <tr>
                <td colspan=2>
                    <center>Import notes data with RM Number first and Notes in the
second column seperated by a tab.<br>This will append notes to existing
notes.</center>
                </td>
            </tr>

            <tr>
                <td colspan=2>
                    &nbsp;
                </td>
            </tr>
            <tr>
                <td>
                    File:
                </td>
                <td>
                    <input type="file" name="csvfile">
                </td>
            </tr>
            <tr>
                <td colspan="2" align="right">
                    <input type="submit" value="Upload File">
                </td>
            </tr>
        </table>
    </form>

<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subtype",
"InsertSubtypeTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
    %>
    <center><h1>Import Comma Delimited File</h1></center>
    <form method=post action="locusimportconfirm.jsp" enctype="MULTIPART/FORM-DATA">
        <table align=center>
            <tr>
                <td colspan=2>
                    <center>You can upload a tab delimited file with the following
format:<br>
                    Locus Genus Species Subspecies Subtype Strain Location
Location2(Locus) START END GBK F/R NT AA MW IEP GENE CODE COG EC#1 EC#2 Product
Notes<br>
                    Make sure to strip out any header information, the easiest way
to make a tab delimited file is to use excel and save as text tab delimited
file.</center><br>
                </td>
            </tr>
            <tr>
                <td colspan=2>
                    &nbsp;
                </td>
            </tr>
            <tr>
                <td>
                    File:
                </td>
                <td>
                    <input type="file" name="csvfile">
                </td>
            </tr>
            <tr>
                <td colspan="2" align="right">
                    <input type="submit" value="Upload File">
                </td>
            </tr>
        </table>
    </form>

<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%
response.setHeader("Cache-Control", "no-cache, post-check=0, pre-check=0");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 01 Dec 1994 16:00:00 GMT");
%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>PSM Strain Database</title>
  </head>
  <body style="background-color: rgb(54, 171, 54);">
    <br>
    <center><span style="color: rgb(231, 229, 38);font-weight: bold;"><h3>welcome to
the Produce Safety and Microbiology Databases<br>Please login
below</h3></span></center>
    <br>
    <table border=0 align=center>
      <tr>
        <td>
          <table align=left cellpadding=0 >
            <form method="post" action="home.jsp">
              <tr border=0>
                <td><span style="color: rgb(253, 161, 17);font-weight:
bold;"><b>Username:&nbsp;</b></span></td>
                <td><input type="text" name = "username" size=25></td>
              </tr>
              <tr>
                <td><span style="color: rgb(253, 161, 17);font-weight:
bold;"><b>Password:&nbsp;</b></span></td>
                <td><input type = "password" name = "password" size=25></td>
              </tr>
              <tr border=0>
                <td></td>
                <td align=right><input type=submit value="Login"
style="font-weight: bold;"></td>
              </tr>
            </form>
          </table>
        </td>
      </tr>
    </table>
  </body>
</html>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="nIn" class="psm.gene.nucleotideinsert" scope="session"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert AA/NT",
"InsertAANTTitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <%
        dbBean.dbConnect("genelist");
        if (nIn.insertNucleotide(request, dbBean) == 0) {
            out.write("Insert successfull!");
        } else {
            out.write("Insert failed, please check and make sure the file extension
matches what you selected.<br>You may also want to make sure that the locus has
already been inserted into the database.");
        }
        dbBean.dbClose();

        out.write(pageSource.getPageFooter());
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MALDI Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form method=post action="checkmaldiparse.jsp" enctype="MULTIPART/FORM-DATA">
    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1>Upload MALDI-TOF MS Data</h1></center>
            </td>
        </tr>
        <tr>
            <td>
                Find File:
            </td>
            <td>
                <input type="file" name="maldifile">
            </td>
        </tr>
        <tr>
            <td></td>
            <td colspan="1" align=left>
                <input type="submit" value="Send">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert AA/NT",
"InsertAANTTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
    %>

    <center><h1>Insert AA/NT</h1></center>

    <form method=post action="insertnucleotidedataconfirm.jsp"
enctype="MULTIPART/FORM-DATA">
    <table align=center>
        <tr>
            <td>
                <input type=file name="aantfile">
            </td>
        </tr>
        <tr>
            <td style="text-align: right;">
                <input type="submit" value="Send" >
            </td>
        </tr>
    </table>
</form>

    <%
    out.write(pageSource.getPageFooter());
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
dbBean.dbConnect();
pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Strain",
"InsertStrainTitle", application.getRealPath("/"), dbBean);
out.write(pageSource.getPageHeader());
dbBean.dbClose();
%>

    <%dbBean.dbConnect();%>

    <%
    int gtent, sptent, subtent, othertent;
    //String cmpr = "1";

    if (request.getParameter("gtent") == null) {
        gtent = 0;
    } else {
        gtent = 1;
    }
    if (request.getParameter("sptent") == null) {
        sptent = 0;
    } else {
        sptent = 1;
    }
    if (request.getParameter("subtent") == null) {
        subtent = 0;
    } else {
        subtent = 1;
    }
    if (request.getParameter("othertent") == null) {
        othertent = 0;
    } else {
        othertent = 1;
    }
    %>

    <%
    //Test and see if this page loaded from the submit, or submit and increment
    button, or from neither,
    if (request.getParameter("submit") != null) {
        int success = 0;
        for (int i = 0; i <
Integer.parseInt(request.getParameter("numberofentries")); i++) {
            //Page Reloaded from submit
            int dbRtrn =
dbBean.insertNewStrain(Integer.parseInt(request.getParameter("ident")), Integer.parse
Int(request.getParameter("number")) + i,
request.getParameter("date"), Integer.parseInt(request.getParameter("genus")), Integer
.parseInt(request.getParameter("species")), Integer.parseInt(request.getParameter("su
bspecies")), Integer.parseInt(request.getParameter("subtype")), request.getParameter("
notes"), request.getParameter("history"), gtent, sptent, subtent, othertent);
            if (dbRtrn == 0) {
                success++;
                if(i == (Integer.parseInt(request.getParameter("numberofentries"))
-1)) {
                    out.write(Integer.toString(i+1) + " strain(s) successfully
inserted.<br>");
                }
            }
        }
    }
    %>

```

```

        } else if ( dbRtrn == 3) {
            out.write ("This identifier and number combination already exists,
please check your number again or use the update page to change this data.<br>");
        } else{
            out.write("Insert Failed, please press back and make sure that your
data is correct and that your login has not timed out.<br>");
        }
        dbRtrn = 0;
    }

    } else {
        int success = 0;
        if (request.getParameter("increment") != null) {
            for (int i =0; i <
Integer.parseInt(request.getParameter("numberofentries")); i++) {
                //Page Reloaded from submit
                int dbRtrn =
dbBean.insertNewStrain(Integer.parseInt(request.getParameter("ident")),Integer.parse
Int(request.getParameter("number"))+ i,
request.getParameter("date"),Integer.parseInt(request.getParameter("genus")),Integer
.parseInt(request.getParameter("species")),Integer.parseInt(request.getParameter("su
bspecies")),Integer.parseInt(request.getParameter("subtype")),request.getParameter("
notes"),request.getParameter("history"),gtent ,sptent,subtent,othertent);
                if ( dbRtrn ==0) {
                    success++;
                    if(i ==
(Integer.parseInt(request.getParameter("numberofentries")) -1)) {
                        out.write(Integer.toString(i+1) + " strain(s) successfully
inserted.<br>");
                    }
                } else if ( dbRtrn == 3) {
                    out.write ("This identifier and number combination already
exists, please check your number again or use the update page to change this
data.<br>");
                } else{
                    out.write("Insert Failed, please press back and make sure that
your data is correct and that your login has not timed out.<br>");
                }
                dbRtrn = 0;
            }
        }

    }

    %>
    <form action="insertstrain.jsp">
    <table border="0" cellpadding = "0" cellspacing = "1" align="center">
    <tr>
    <td colspan = "3">
    <center><h1>Insert New Strain</h1></center>
    </td>
    </tr>
    <tr>
    <td colspan="3">
    <center>All fields are nullable.</center>
    </td>
    </tr>
    <tr>
    <td>
        Identifier (RM):&nbsp;
    </td>
    <td>
        <select name="ident">
            <%
                ResultSet myRS = dbBean.getIdentPrefix();
                if (myRS.next()) {
                    myRS.first();
                    do {

```



```

        if (request.getParameter("ident") != null &&
request.getParameter("ident").compareTo(myRS.getString("identifier_id")) == 0) {
        out.write("<option value=" +
myRS.getInt("identifier_id") + " selected>" + myRS.getString("ident"));
        } else {
        out.write("<option value=" +
myRS.getInt("identifier_id") + ">" + myRS.getString("ident"));
        }
    } while (myRS.next());
    }
    myRS.close();
    %>
</select>&nbsp;<a href="newidentifier.jsp">New Identifier</a>&nbsp; 
</td>
</tr>
<tr>
    <td>
        Number (1222):&nbsp; 
    </td>
    <td>
        <input type="text" name="number" value ="<% if
(request.getParameter("number") != null) {
Integer.parseInt(request.getParameter("number"));
Integer.parseInt(request.getParameter("numberofentries"));
out.write(Integer.toString(mInt));}
int mInt =
mInt = mInt +
else {
out.write(Integer.toString(dbBean.getLastUsedStrainNumber()+1));
}
%>">
    </td>
</tr>
<tr>
    <td>
        Date (2005-8-24):
    </td>
    <td>
        <input type="text" name="date" value="<% if
(request.getParameter("date") != null) {
out.write(request.getParameter("date"));}%>">
    </td>
</tr>
<tr>
    <td>
        Genus:
    </td>
    <td>
        <select name="genus">
            <%
myRS = dbBean.getGenus();
if (myRS.next()) {
myRS.first();
do {
String cmpr = request.getParameter("genus");
if (cmpr == null) { cmpr = "";}
if (myRS.getString("name")!=null) {
if (cmpr.compareTo(myRS.getString("genus_id")) == 0)
{
out.write("<option value='" +
myRS.getInt("genus_id") + "' selected>" + myRS.getString("name"));
} else {
out.write("<option value='" +

```

```

myRS.getInt("genus_id") + "'>" + myRS.getString("name"));
    } else {
        if (cmpr.compareTo("")==0) {
            out.write("<option value='" +
myRS.getInt("genus_id") + "' selected>No Genus");
        } else {
            out.write("<option value='" +
myRS.getInt("genus_id") + "'>No Genus");
        }
    }
} while (myRS.next());
}
myRS.close();
%>
</select>
</td>
<td>
    <%
    if (request.getParameter("gtent") == null ||
Integer.parseInt(request.getParameter("gtent")) == 0) {
        out.write("<input type='checkbox' name='gtent'
value='1'>Tentative");
    } else {
        out.write("<input type='checkbox' name='gtent' value='1'
checked>Tentative");
    }
    %>
</td>
</tr>
<tr>
<td>
    Species:
</td>
<td>
        <select name="species">
            <%
            myRS = dbBean.getSpecies();
            if (myRS.next()) {
                myRS.first();
                do {
                    String cmpr = request.getParameter("species");
                    if (cmpr == null) { cmpr = ""; }
                    if (myRS.getString("name")!=null) {
                        if (cmpr.compareTo(myRS.getString("species_id")) ==
0) {
                            out.write("<option value='" +
myRS.getInt("species_id") + "' selected>" + myRS.getString("name"));
                        } else {
                            out.write("<option value='" +
myRS.getInt("species_id") + "'>" + myRS.getString("name"));
                        }
                    } else {
                        if (cmpr.compareTo("")==0) {
                            out.write("<option value='" +
myRS.getInt("species_id") + "' selected>No Species");
                        } else {
                            out.write("<option value='" +
myRS.getInt("species_id") + "'>No Species");
                        }
                    }
                } while (myRS.next());
            }
            myRS.close();
            %>

```

```

        </select>
    </td>
    <td>
        <%
            if (request.getParameter("sptent") == null ||
Integer.parseInt(request.getParameter("sptent")) == 0) {
                out.write("<input type='checkbox' name='sptent'
value='1'>Tentative");
            } else {
                out.write("<input type='checkbox' name='sptent' value='1'
checked>Tentative");
            }
        %>
    </td>
</tr>
<tr>
    <td>
        Subspecies:
    </td>
    <td>
        <select name="subspecies">
            <%
                myRS = dbBean.getSubSpecies();
                if (myRS.next()) {
                    myRS.first();
                    do {
                        String cmpr = request.getParameter("subspecies");
                        if (cmpr == null) { cmpr = ""; }
                        if (myRS.getString("name")!=null) {
                            if (cmpr.compareTo(myRS.getString("subspecies_id"))
== 0) {
                                out.write("<option value='" +
myRS.getInt("subspecies_id") + "' selected>" + myRS.getString("name"));
                            } else {
                                out.write("<option value='" +
myRS.getInt("subspecies_id") + "'>" + myRS.getString("name"));
                            }
                        } else {
                            if (cmpr.compareTo("")==0 ) {
                                out.write("<option value='" +
myRS.getInt("subspecies_id") + "' selected>No Subspecies");
                            } else {
                                out.write("<option value='" +
myRS.getInt("subspecies_id") + "'>No Subspecies");
                            }
                        }
                    } while (myRS.next());
                }
                myRS.close();
            %>
        </select>
    </td>
    <td>
        <%
            if (request.getParameter("subtent") == null ||
Integer.parseInt(request.getParameter("subtent")) == 0) {
                out.write("<input type='checkbox' name='subtent'
value='1'>Tentative");
            } else {
                out.write("<input type='checkbox' name='subtent' value='1'
checked>Tentative");
            }
        %>
    </td>
</tr>

```

```
|  |  |
| --- | --- |
| Serovar/Subtype/Genotype: | <select name="subtype">           <%             myRS = dbBean.getSubType();             if (myRS.next()) {               myRS.first();               do {                 String cmpr = request.getParameter("subtype");                 if (cmpr == null) { cmpr = ""; }                 if (myRS.getString("name") != null) {                   if (cmpr.compareTo(myRS.getString("subtype_id")) == 0) {                     out.write("<option value='" + myRS.getInt("subtype_id") + "' selected>" + myRS.getString("name"));                   } else {                     out.write("<option value='" + myRS.getInt("subtype_id") + "'>" + myRS.getString("name"));                   }                 } else {                   if (cmpr.compareTo("") == 0) {                     out.write("<option value='" + myRS.getInt("subtype_id") + "' selected>No Subtype");                   } else {                     out.write("<option value='" + myRS.getInt("subtype_id") + "'>No Subtype");                   }                 }               } while (myRS.next());             }             myRS.close();           %>         </select> |
| <%           if (request.getParameter("othertent") == null || Integer.parseInt(request.getParameter("othertent")) == 0) {             out.write("<input type='checkbox' name='othertent' value='1'>Tentative");           } else {             out.write("<input type='checkbox' name='othertent' value='1' checked>Tentative");           }         %> | </td> |
| <td style="vertical-align: top">           Notes:         </td>         <td colspan="2">           <textarea name="notes" cols="60" rows="10" ><%if (request.getParameter("notes") != null) {out.write(request.getParameter("notes"));}%></textarea>         </td> | </tr> |
| <td style="vertical-align: top">           History:         </td>         <td colspan="2"> |

```

```

        <textarea name="history" cols="60" rows="5"><%if
(request.getParameter("history") != null)
{out.write(request.getParameter("history"));}%></textarea>
    </td>
    <tr>
    <tr>
        <td>
            Number of entries to insert:
        </td>
        <td>
            <input name="numberofentries" type="text" value="1" size="5"><br>*Press
Submit New Strain for this to be applicable, not Submit and Increment
        </td>
    </tr>
    <tr>
        <td colspan="3" align="right">
            <input type="submit" name="increment" value="Submit and
Increment">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit" value="Submit New Strain"
name="submit">
        </td>
    </tr>
</table>
</form>
<%}dbBean.dbClose();%>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert <i>In Silico</i>
Data", "InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form method=post action="checkinsilicoparse.jsp" enctype="MULTIPART/FORM-DATA">
    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1>Upload <i>In Silico</i> Data</h1></center>
            </td>
        </tr>
        <tr>
            <td>
                Find File:
            </td>
            <td>
                <input type="file" name="maldifile">
            </td>
        </tr>
        <tr>
            <td></td>
            <td colspan="1" align=left>
                <input type="submit" value="Send">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="frInsilico" class="psm.mformat.formatinsilico"
scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
dbBean.dbConnect();
pageSource.splitTemplatePage(dbBean.getUsername(), "Insert <i>In Silico</i>
Data", "InsertMALDITitle", application.getRealPath("/"), dbBean);
out.write(pageSource.getPageHeader());
dbBean.dbClose();
%>
<%
dbBean.dbConnect();
int rtn;
int genus = dbBean.insertNewGenus(request.getParameter("genus"));
int species = dbBean.insertNewSpecies(request.getParameter("species"));
int subspecies = dbBean.insertNewSubspecies(request.getParameter("subspecies"));
int subtype = dbBean.insertNewSubtype( request.getParameter("subtype"));

rtn =
dbBean.insertINSILICO(request.getParameter("ident"), Integer.toString(genus),
Integer.toString(species), Integer.toString(subspecies) ,Integer.toString(subtype),
request.getParameter("mw"), request.getParameter("protein"),
request.getParameter("notes"), frInsilico.getMaldiMZ());
if (rtn == 1) {
    out.write("This identifier does not exist, please press back and choose an
identifier already in the database.");
} else if (rtn !=0) {
    out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
} else if (rtn ==0 ) {
    out.write("Insert successfull!");
}
dbBean.dbClose();
%>
<%out.write(pageSource.getPageFooter()); %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>In Silico Data</title>
</head>
<body>

<h1><i>In Silico</i> Data</h1>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session" />

<b><i>In Silico</i> ID:</b> <%=request.getParameter("insilicoid")%><br>

<%
dbBean.dbConnect();
ResultSet myRS;

myRS = dbBean.getRS("SELECT CONCAT_WS(' ', genus.name, species.name,
subspecies.name, subtype.name) FROM genus,species,subspecies, subtype, in_silico,
strain WHERE strain.strain_id = in_silico.strain_id AND strain.genus_id =
genus.genus_id AND strain.species_id = species.species_id AND strain.subspecies_id =
subspecies.subspecies_id AND strain.subtype_id = subtype.subtype_id AND
in_silico.in_silico_id = " + request.getParameter("insilicoid"));
myRS.next();
out.write(myRS.getString(1));
myRS.close();
myRS = dbBean.getINSILICOData(request.getParameter("insilicoid"));
if (myRS.next()) {
myRS.first();
%>
<br>
<table>
<th>Mass to Charge</th>

<%
do {
%>

<tr>
<td>
<%=myRS.getString("mass")%>
</td>

<%} while (myRS.next());
} else {%>
There is no MS/MS data for this sample.
<%}

myRS.close();
dbBean.dbClose();
%>

</table>
<br>
<a
href="/deleteinsilicoconfirm.jsp?insilicoid=<%=request.getParameter("insilicoid")%>
" target="_top">DELETE Data</a><br>
</body>
</body>

```


</html>

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

    <title>MALDI-TOF DATA and Stick Graph</title>

    <frameset cols="75%,25%">
    <frame
src="insilicographandoptions.jsp?insilicoid=<%=request.getParameter("insilicoid")%>&
rangeA=0&rangeB=14000&topnum=1">
    <frame
src="insilicodata.jsp?insilicoid=<%=request.getParameter("insilicoid")%>">
    </frameset>

</html>
```

```

<%@ page language="java" %>
<%@ page import ="java.util.*" %>
<%@ page import ="java.text.NumberFormat" %>
<%@ page import ="java.text.DecimalFormat" %>
<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%

int id1, id2, intA = 0, intB = 14000, intensity=2, ybound;
String masserror = "2.5";

id1 = Integer.parseInt(request.getParameter("id1"));
id2 = Integer.parseInt(request.getParameter("id2"));

if (request.getParameter("rangeA") != "") {
intA = Integer.parseInt(request.getParameter("rangeA"));
}

if (request.getParameter("rangeB") != "") {
intB = Integer.parseInt(request.getParameter("rangeB"));
}

if (request.getParameter("masserror") != "") {
masserror = request.getParameter("masserror");
}

if (request.getParameter("intensity") != "") {
intensity = Integer.parseInt(request.getParameter("intensity"));
}

ybound = (int) (Double.parseDouble(masserror) + 0.5);
ybound = ybound * -1;

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,580,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,580);

biContext.setColor(Color.BLACK);
//Vertical Line
//left
biContext.drawLine(60,40,60,540);
//right
biContext.drawLine(810,40,810,540);
//Horizontal Line
//top
biContext.drawLine(60,40,810,40);
//bottom
biContext.drawLine(60,540,810,540);
biContext.setFont(font);

//Horizontal Line
//middle
biContext.drawLine(60,290,810,290);

```

```

//Draws the Y Numbers
//biContext.drawString(Integer.toString(ybound),35,545);
//biContext.drawLine(53, 540, 60, 540);

double interval = 500.0 / (ybound*-2);

for (int j = 0; j <= (ybound * -2); j++) {
    int increment = (int) ((double) j *interval + .5);
    biContext.drawString(Integer.toString(ybound + j), 30, 545 -increment);
    biContext.drawLine(53, 540-increment, 60, 540-increment);
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 315);

//Draws the X numbers
for (l=1; l <=5; l++) {

    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

    if (num.length()%2 !=0) {
        centerint+= 3;
    }

    biContext.drawString(num, (60 + 150*(l) - centerint) , 315);

    biContext.drawLine(60+150*l, 280, 60+150*l, 300);
}

dbBean.dbConnect();

ResultSet myRS =
dbBean.getInSilicoErrorData(Integer.parseInt(request.getParameter("id1")),
Integer.parseInt(request.getParameter("id2")), Integer.toString(intensity),
masserror, Integer.toString(intA), Integer.toString(intB),
request.getParameter("residue"));

ArrayList x, y, r;
x = new ArrayList();
y = new ArrayList();
r = new ArrayList();

int n = 0;
double xavg = 0;
double yavg = 0;
double xmin = intB;
double xmax = 0;

Double tempx;
Double tempy;
String tempr;

//Grab the data from the database and calculate the average of x and y
//also determine maximum and minimum value for y

```

```

while(myRS.next()) {
    x.add(new Double(myRS.getDouble("x")));
    y.add(new Double(myRS.getDouble("Error")));
    r.add(new String(myRS.getString("residue")));

    tempX = (Double)x.get(n);

    xavg += tempX.doubleValue();
    tempY = (Double)y.get(n);
    yavg += tempY.doubleValue();
    if (xmin > tempX.doubleValue()) {
        xmin = tempX.doubleValue();
    }
    if(xmax < tempX.doubleValue()) {
        xmax = tempX.doubleValue();
    }
    n++;
}

xavg = xavg / (n);
yavg = yavg / (n);
double Sxx = 0.0;
double Syy = 0.0;
double Sxy = 0.0;
int countDEP = 0;

double xinterval = 750.0 / (intB - intA);
int radius = 5;
for(int t = 0; t < x.size(); t++) {
    tempX = (Double)x.get(t);
    tempY = (Double)y.get(t);
    tempr = (String)r.get(t);

    //Calculate Sxx and Syy and Sxy
    Sxx += Math.pow((tempX.doubleValue() - xavg),2);
    Syy += Math.pow((tempY.doubleValue() - yavg),2);
    Sxy += (tempX.doubleValue() - xavg)*(tempY.doubleValue() - yavg);

    //Choose which color to plot the point as. DEP residues are labeled red
    //all others are black

    if(tempr.contains("D") || tempr.contains("E") || tempr.contains("P")) {
        biContext.setColor(Color.red);
        countDEP++;
    } else {
        biContext.setColor(Color.black);
    }

    //Plot the points

    double x1 = (int) 60.5 + xinterval * (tempX.doubleValue() - intA);
    double y1 = (int) 290.5 - interval*(tempY.doubleValue());

    biContext.drawRect((int) x1,(int) y1 - (radius/2), radius, radius);
    biContext.fillRect((int) x1, (int) y1 - (radius/2), radius, radius);
}

double ratioDEP = 100 * (double)countDEP / (double) x.size();

biContext.setColor(Color.black);
//calculate the slope
double m = Sxy/Sxx;
//calculate the intercept
double b = yavg - m*xavg;

```

```

//calculate teh r2 value
double r2 = Math.pow(Sxy,2) / (Sxx * Syy);
xmin -= xmin * .15;
xmax += xmax * .15;

if (xmin < intA) {
    xmin = intA;
}
if (xmax > intB) {
    xmax = intB;
}

int tx1 = (int) (((xmin - intA) * xinterval) + 60.5);
int tx2 = (int) (((xmax - intA) * xinterval) + 60.5);
int yx1 = (int) (290.5 - ((xmin*m + b)) * interval);
int yx2 = (int) (290.5 - ((xmax*m + b)) * interval);
biContext.setStroke(new BasicStroke(2));
biContext.drawLine(tx1,yx1,tx2,yx2);

NumberFormat outnum = new DecimalFormat("#0.0000");

biContext.drawString("R^2 = " + outnum.format(r2),650, 55 );
biContext.drawString("y = " + outnum.format(m) + "x + " + outnum.format(b), 650,
70);
biContext.drawString("DEP Count: " + Integer.toString(countDEP), 650, 85);
biContext.drawString("DEP Ratio: " + outnum.format(ratioDEP) + "%", 650, 100);
String residue = "All";
if (request.getParameter("residue") != null) {
    residue = request.getParameter("residue");
}
biContext.drawString("All Matches, " + Integer.toString(intensity) + "% Intensity
Cutoff, " + masserror + " Da Mass Tolerance" + ", Residue: " + residue, 275, 30);

Font font2 = new Font("ARIAL",Font.ITALIC + Font.BOLD, 12);

biContext.setFont(font2);
biContext.drawString("m/z", 350, 560);
biContext.setFont(font);
biContext.drawString(" of MS/MS Fragment Ions", 370, 560);

FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 325 - (fm.getHeight()/2));
biContext.drawString("Error", 25,325);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Error Graphs</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="insilicoerrorgraphsallmatchandopt.jsp">
      <%
        int rangeA, rangeB, topnum;
        if (request.getParameter("rangeA") != "" &&
request.getParameter("rangeA") != null) {
          rangeA = Integer.parseInt(request.getParameter("rangeA"));
        } else {
          rangeA = 4000;
        }
        if (request.getParameter("rangeB") != "" &&
request.getParameter("rangeB") != null) {
          rangeB = Integer.parseInt(request.getParameter("rangeB"));
        } else {
          rangeB = 14000;
        }
        if (request.getParameter("topnum") != "" &&
request.getParameter("topnum") != null) {
          topnum = Integer.parseInt(request.getParameter("topnum"));
        } else {
          topnum = 5;
        }
      %>
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &id2=<%=req
uest.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<%=requ
est.getParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")%>&masser
ror=<%=request.getParameter("masserror")%>&residue=<%=request.getParameter("residue"
)%>">
            </td>
          </tr>
          <tr>
            <td>
              Low Value:&nbsp;<input type="text" name="rangeA" value="<%=rangeA%>">
            </td>
            <td>
              High Value:&nbsp;<input type="text" name="rangeB"
value="<%=rangeB%>">
            </td>
            <td>
              Number of values Labeled: <input type="text" name="topnum"
value="<%=topnum%>">
            </td>
          </tr>
          <tr>
            <td>
              Residue:&nbsp;<input type="text" name="residue"
value="<%=request.getParameter("residue")%>" readonly>

```

```
</td>
      Mass Tolerance:<input type="text" name="masserror"
value="<%=request.getParameter("masserror")%>" readonly>
    </td>
      Intensity:<input type="text" name="intensity"
value="<%=request.getParameter("intensity")%>" readonly>
    </td>
  </tr>
  <tr>
    <td colspan = "1"> <br>
      In Silico ID:&nbsp;  <input type="text" name="id2"
value="<%=request.getParameter("id2")%>" readonly>
    </td>
    <td colspan ="2">
      MS/MS ID:&nbsp;  <input type="text" name="id1" value=
<%=request.getParameter("id1")%> readonly>
    </td>
  </tr>
  <tr>
    <td colspan = 3 align = right>
      <input type=submit value="Reload">
    </td>
  </tr>
</table>
</form>
</body>
</html>
```



```

<%@ page language="java" %>
<%@ page import ="java.util.*" %>
<%@ page import ="java.text.NumberFormat" %>
<%@ page import ="java.text.DecimalFormat" %>
<%@ page import ="java.io.OutputStream" %>
<%@ page import ="java.awt.image.BufferedImage" %>
<%@ page import ="java.awt.*" %>
<%@ page import ="com.sun.image.codec.jpeg.*" %>
<%@page import ="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%

int id1, id2, intA = 0, intB = 14000, intensity=2, ybound;
String masserror = "2.5";

id1 = Integer.parseInt(request.getParameter("id1"));
id2 = Integer.parseInt(request.getParameter("id2"));

if (request.getParameter("rangeA") != "") {
intA = Integer.parseInt(request.getParameter("rangeA"));
}

if (request.getParameter("rangeB") != "") {
intB = Integer.parseInt(request.getParameter("rangeB"));
}

if (request.getParameter("masserror") != "") {
masserror = request.getParameter("masserror");
}

if (request.getParameter("intensity") != "") {
intensity = Integer.parseInt(request.getParameter("intensity"));
}

ybound = (int) (Double.parseDouble(masserror) + 0.5);
ybound = ybound * -1;

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,580,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,580);

biContext.setColor(Color.BLACK);
//Vertical Line
//left
biContext.drawLine(60,40,60,540);
//right
biContext.drawLine(810,40,810,540);
//Horizontal Line
//top
biContext.drawLine(60,40,810,40);
//bottom
biContext.drawLine(60,540,810,540);
biContext.setFont(font);

//Horizontal Line
//middle
biContext.drawLine(60,290,810,290);

```

```

//Draws the Y Numbers
//biContext.drawString(Integer.toString(ybound),35,545);
//biContext.drawLine(53, 540, 60, 540);

double interval = 500.0 / (ybound*-2);

for (int j = 0; j <= (ybound * -2); j++) {
    int increment = (int) ((double) j *interval + .5);
    biContext.drawString(Integer.toString(ybound + j), 30, 545 -increment);
    biContext.drawLine(53, 540-increment, 60, 540-increment);
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 315);

//Draws the X numbers
for (l=1; l <=5; l++) {

    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

    if (num.length()%2 !=0) {
        centerint+= 3;
    }

    biContext.drawString(num, (60 + 150*(l) - centerint) , 315);

    biContext.drawLine(60+150*l, 280, 60+150*l, 300);
}

dbBean.dbConnect();

ResultSet myRS =
dbBean.getInSilicoErrorData(Integer.parseInt(request.getParameter("id1")),
Integer.parseInt(request.getParameter("id2")), Integer.toString(intensity),
masserror, Integer.toString(intA), Integer.toString(intB),
request.getParameter("residue"));

ArrayList x, y, z, r, tossx, tossy, tossi;

z = new ArrayList();
x = new ArrayList();
y = new ArrayList();
r = new ArrayList();
tossx = new ArrayList();
tossy = new ArrayList();
tossi = new ArrayList();
int n = 0;
double xavg = 0;
double yavg = 0;
double xmin = intB;
double xmax = 0;

Double tempx = new Double(0.0);
Double ttempx = new Double(0.0);

```

```

Double tempi = new Double(0.0);
Double tempy = new Double(0.0);
Double tempz = new Double(0.0);
String tempr = new String("");

//Grab the data from the database and calculate the average of x and y
//also determine maximum and minimum value for y
while(myRS.next()) {
    double insilico = myRS.getDouble("insilico");
    double nx = myRS.getDouble("x");
    double nerr = myRS.getDouble("Error");
    String nR = myRS.getString("residue");
    if (n > 0) {
        tempx = (Double)x.get(n-1);
        if (tossx.size() > 0) {
            ttempx = (Double)tossx.get(tossx.size()-1);
        }
        if (tossi.size() > 0) {
            tempi = (Double)tossi.get(tossi.size()-1);
        }
        tempy = (Double)y.get(n-1);
        tempz = (Double)z.get(n-1);
        tempr = (String)r.get(n-1);

        if ((tossi.size() > 0 && insilico == tempi.doubleValue()) || (tossx.size() >
0 && nx == ttempx.doubleValue())) {
            //Dont follow through ignore this one
        } else if (tempx.doubleValue() == nx) {
            xavg -= tempx.doubleValue();
            yavg -= tempy.doubleValue();
            x.remove(n-1);
            y.remove(n-1);
            z.remove(n-1);
            r.remove(n-1);
            tossx.add(new Double(nx));
            n--;
        } else if (tempz.doubleValue() == insilico) {
            xavg -= tempx.doubleValue();
            yavg -= tempy.doubleValue();
            x.remove(n-1);
            y.remove(n-1);
            z.remove(n-1);
            r.remove(n-1);
            tossi.add(new Double(nerr));
            n--;
        } else {
            x.add(new Double(nx));
            y.add(new Double(nerr));
            z.add(new Double(insilico));
            r.add(new String(nR));
            tempx = (Double)x.get(n);
            tempy = (Double)y.get(n);
            xavg += tempx.doubleValue();
            yavg += tempy.doubleValue();

            n++;
        }
    } else {
        x.add(new Double(nx));
        y.add(new Double(nerr));
        z.add(new Double(insilico));
        r.add(new String(nR));
        tempx = (Double)x.get(n);
        tempy = (Double)y.get(n);
        xavg += tempx.doubleValue();
    }
}

```

```

        yavg += tempy.doubleValue();
        n++;
    }
}

xavg = xavg / (n);
yavg = yavg / (n);
double Sxx = 0.0;
double Syy = 0.0;
double Sxy = 0.0;
int countDEP = 0;
double xinterval = 750.0 / (intB - intA);
int radius = 5;
for(int t = 0; t < x.size(); t++) {

    tempx = (Double)x.get(t);
    tempy = (Double)y.get(t);
    tempr = (String)r.get(t);
    //Calculate Sxx and Syy and Sxy
    Sxx += Math.pow((tempx.doubleValue() - xavg),2);
    Syy += Math.pow((tempy.doubleValue() - yavg),2);
    Sxy += (tempx.doubleValue() - xavg)*(tempy.doubleValue() - yavg);

    //Choose color
    if(tempr.contains("D") || tempr.contains("E") || tempr.contains("P")) {
        biContext.setColor(Color.red);
        countDEP++;
    } else {
        biContext.setColor(Color.black);
    }

    //Plot the points
    double x1 = (int) 60.5 + xinterval * (tempx.doubleValue() - intA);
    double y1 = (int) 290.5 - interval*(tempy.doubleValue());
    biContext.drawRect((int) x1 - (radius/2),(int) y1 - (radius/2), radius, radius);
    biContext.fillRect((int) x1 - (radius/2), (int) y1 - (radius/2), radius,
radius);

    if (xmin > tempx.doubleValue()) {
        xmin = tempx.doubleValue();
    }
    if(xmax < tempx.doubleValue()) {
        xmax = tempx.doubleValue();
    }
}

biContext.setColor(Color.black);

double ratioDEP = 100 * (double)countDEP / (double) x.size();

//calculate the slope
double m = Sxy/Sxx;
//calculate the intercept
double b = yavg - m*xavg;
//calculate teh r2 value
double r2 = Math.pow(Sxy,2) / (Sxx * Syy);
xmin -= xmin * .15;
xmax += xmax * .15;

if (xmin < intA) {
    xmin = intA;
}

```

```

}
if (xmax > intB) {
    xmax = intB;
}

int tx1 = (int) (((xmin - intA) * xinterval) + 60.5);
int tx2 = (int) (((xmax - intA) * xinterval) + 60.5);
int yx1 = (int) (290.5 - ((xmin*m + b)) * interval);
int yx2 = (int) (290.5 - ((xmax*m + b)) * interval);
biContext.setStroke(new BasicStroke(2));
biContext.drawLine(tx1,yx1,tx2,yx2);

NumberFormat outnum = new DecimalFormat("#0.0000");

biContext.drawString("R^2 = " + outnum.format(r2),650, 55 );
biContext.drawString("y = " + outnum.format(m) + "x + " + outnum.format(b), 650,
70);
biContext.drawString("DEP Count: " + Integer.toString(countDEP), 650, 85);
biContext.drawString("DEP Ratio: " + outnum.format(ratioDEP) + "%", 650, 100);
String residue = "All";
if (request.getParameter("residue") != null) {
    residue = request.getParameter("residue");
}
biContext.drawString("Single Matches, " + Integer.toString(intensity) + "% Intensity
Cutoff, " + masserror + " Da Mass Tolerance" + ", Residue: " + residue, 275, 30);

Font font2 = new Font("ARIAL",Font.ITALIC + Font.BOLD, 12);

biContext.setFont(font2);
biContext.drawString("m/z", 350, 560);
biContext.setFont(font);
biContext.drawString(" of MS/MS Fragment Ions", 370, 560);

FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 325 - (fm.getHeight()/2));
biContext.drawString("Error", 25,325);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Error Graphs</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="insilicoerrorgraphssinglematchandopt.jsp">
      <%
        int rangeA, rangeB, topnum;
        if (request.getParameter("rangeA") != "" &&
request.getParameter("rangeA") != null) {
          rangeA = Integer.parseInt(request.getParameter("rangeA"));
        } else {
          rangeA = 4000;
        }
        if (request.getParameter("rangeB") != "" &&
request.getParameter("rangeB") != null) {
          rangeB = Integer.parseInt(request.getParameter("rangeB"));
        } else {
          rangeB = 14000;
        }
        if (request.getParameter("topnum") != "" &&
request.getParameter("topnum") != null) {
          topnum = Integer.parseInt(request.getParameter("topnum"));
        } else {
          topnum = 5;
        }
      %>
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &id2=<%=
request.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<%=r
equest.getParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")%>&mas
serror=<%=request.getParameter("masserror")%>&residue=<%=request.getParameter("resid
ue")%>">
            </td>
          </tr>
        </tr>
        <tr>
          <td>
            Low Value:&nbsp;<input type="text" name="rangeA" value="<%=rangeA%>">
          </td>
          <td>
            High Value:&nbsp;<input type="text" name="rangeB"
value="<%=rangeB%>">
          </td>
          <td>
            Number of values Labeled: <input type="text" name="topnum"
value="<%=topnum%>">
          </td>
        </tr>
        <tr>
          <td>
            Residue:&nbsp;<input type="text" name="residue"

```

```
value="<%=request.getParameter("residue")%>" readonly>
    </td>
    <td>
        Mass Tolerance:<input type="text" name="masserror"
value="<%=request.getParameter("masserror")%>" readonly>
        </td>
        Intensity:<input type="text" name="intensity"
value="<%=request.getParameter("intensity")%>" readonly>
    </td>
</tr>
<tr>
    <td colspan = "1"> <br>
        In Silico ID:&nbsp;  <input type="text" name="id2"
value="<%=request.getParameter("id2")%>" readonly>
    </td>
    <td colspan = "2">
        MS/MS ID:&nbsp;  <input type="text" name="id1" value=
<%=request.getParameter("id1")%> readonly>
    </td>
</tr>
<tr>
    <td colspan = 3 align = right>
        <input type=submit value="Reload">
    </td>
</tr>
</table>
</form>
</body>
</html>
```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("insilicoid");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,620,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,620);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,540);
biContext.drawLine(60,540,810,540);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,545);
biContext.drawLine(53, 540, 60, 540);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 545 - j*5);
        biContext.drawLine(53, 540-j*5, 60, 540-j*5);
    } else {
        biContext.drawLine(58 , 540 -j *5, 60, 540-j*5 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 570);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```



```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 570);

        biContext.drawLine(60+150*l, 540, 60+150*l, 555);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 540, 60+(75*l), 550);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0 ) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 540, 60+lshift, 545);
        }
    }
}

dbBean.dbConnect();

//draw the lines
biContext = gBean.gDraw(biContext, dbBean.getINSILICOGraphData(id, intA, intB),
intA, intB, topnum, 540, 5, 750);
biContext.setColor(Color.BLACK);
biContext = gBean.gDrawMSMSTitleAA(biContext, dbBean.getINSILICOAndSampleInfo(id),
870);

biContext.drawString("Mass to Charge (M/Z)", 365, 610);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 325 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,325);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title><i>In Silico</i> Graph</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="insilicographandoptions.jsp">
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &rangeA=<
%=request.getParameter("rangeA")%>&rangeB=<%=request.getParameter("rangeB")%>&topnum
=<%=request.getParameter("topnum")%>">
            </td>
          </tr>
          <tr>
            <td>
              <div>
                Low Value:&nbsp;<input type="text" name="rangeA"
value="<%=request.getParameter("rangeA")%>">
              </div>
            </td>
            <td>
              <div>
                High Value:&nbsp;<input type="text" name="rangeB"
value="<%=request.getParameter("rangeB")%>">
              </div>
            </td>
            <td>
              <div>
                Number of Values Labeled: <input type="text" name="topnum"
value="<%=request.getParameter("topnum")%>">
              </div>
            </td>
          </tr>
          <tr>
            <td colspan = "3"> <br>
              In Silico ID:&nbsp;<input type="text" name="insilicoid" value=
<%=request.getParameter("insilicoid")%> readonly>
            </td>
          </tr>
          <tr>
            <td colspan = 3 align = right>
              <input type=submit>
            </td>
          </tr>
        </table>
      </form>
    </body>
  </html>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("insilicoid");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,400,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,600);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,340);
biContext.drawLine(60,340,810,340);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,345);
biContext.drawLine(53, 340, 60, 340);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 345 - j*3);
        biContext.drawLine(53, 340-j*3, 60, 340-j*3);
    } else {
        biContext.drawLine(58 , 340 -j *3, 60, 340-j*3 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 370);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 370);

        biContext.drawLine(60+150*l, 340, 60+150*l, 355);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 340, 60+(75*l), 350);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0 ) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 340, 60+lshift, 345);
        }
    }
}

dbBean.dbConnect();

//draw the lines
biContext = gBean.gDraw(biContext, dbBean.getINSILICOGraphData(id, intA, intB,
request.getParameter("residue")), intA, intB, topnum, 340, 3,750);
biContext.setColor(Color.BLACK);
biContext = gBean.gDrawMSMSTitleAA(biContext, dbBean.getINSILICOAndSampleInfo(id),
870);

biContext.drawString("Mass to Charge (M/Z)", 375, 390);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 230 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,230);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>

<%
dbBean.dbConnect();
pageSource.splitTemplatePage(dbBean.getUsername(), "Strain", "StrainSearchTitle",
application.getRealPath("/"), dbBean);
out.write(pageSource.getPageHeader());
dbBean.dbClose();
%>

<%
//Connect to the genelist db
dbBean.dbConnect("genelist");
//grab the pertinent information for the locus
ResultSet myRS = dbBean.getRS("SELECT locus.locus_id, locus.name 'Locus',
CONCAT(identifier.name, sample.number) 'Strain', CONCAT(genus.name, ' ',
species.name) 'Name', a.name 'Location 1', b.name 'Location 2', locus.start 'Start',
locus.end 'End', CASE locus.FR WHEN 'F' THEN CONCAT(locus.start, '..', locus.end)
WHEN 'R' THEN CONCAT('complement(', locus.start, '..', locus.end, ')') End 'GBK',
locus.FR, locus.NT, locus.AA, locus.MW, locus.IEP, locus.gene 'Gene', locus.code
'Code', locus.COG, locus.EC1 'EC#1', locus.EC2 'EC#2', locus.product 'Product',
locus.notes 'Notes' FROM locus, location a, location b, sample, identifier, strain,
genus, species WHERE locus.location_id = a.location_id AND locus.location2_id =
b.location_id AND locus.sample_id = sample.sample_id AND sample.identifier_id =
identifier.identifier_id AND sample.strain_id = strain.strain_id AND strain.genus_id
= genus.genus_id AND strain.species_id = species.species_id AND locus_id = " +
request.getParameter("locusid"));
ResultSetMetaData myMD = myRS.getMetaData();
myRS.next();
%>

<table cellpadding=0 cellspacing=0 align=center>
<%
    for (int i=1; i <= myMD.getColumnCount() -1; i++) {
        out.write("<tr>");
        out.write("<td style='font-weight: bold;'>&nbsp;" + myMD.getColumnName(i + 1)
+ "</td>");
        out.write("<td>");
        if (myRS.getString(i+1) != null) {
            out.write("&nbsp;" + myRS.getString(i + 1));
        } else {
            out.write("-");
        }
        out.write("</td>");
        out.write("</tr>");
    }
    myRS.close();
    //Now we want the nucleotide data
    myRS = dbBean.getRS("SELECT nucleotide FROM nucleotide WHERE locus_id = " +
request.getParameter("locusid"));
    if (myRS.next()) {
%>
        <tr>
            <td style="vertical-align: top;font-weight: bold;">
                &nbsp;NT Sequence:&nbsp;
            </td>
            <td style="vertical-align: top;">
                <small><small><%=myRS.getString("nucleotide")%></small></small>
            </td>
        </tr>
    <%
%>

```

```
    }
%>
<% myRS.close(); %>
<%
myRS = dbBean.getRS("SELECT sequence FROM nucleotidesequence WHERE locus_id = " +
request.getParameter("locusid"));
if (myRS.next()) {
%>
    <tr>
    <td colspan="2">&nbsp;</td>
    <tr>
    <tr>
        <td style="vertical-align: top;font-weight: bold;">
        &nbsp;&nbsp;&nbsp;&AA Sequence:
        </td>
        <td style="vertical-align: top;">
        <small><small><%=myRS.getString("sequence")%></small></small>
        </td>
    </tr>
<%
    }
%>
</table>

<%
myRS.close();
dbBean.dbClose();
%>

<%
out.write(pageSource.getPageFooter());
%>
```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<jsp:useBean id="importCSV" class="psm.mformat.importGenelist"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subtype",
"InsertSubtypeTitle", application.getRealPath("/"), dbBean);
    dbBean.dbClose();
    out.write(pageSource.getPageHeader());

    dbBean.dbConnect("genelist");
    if (importCSV.uploadGenelist(request,dbBean) == 0) {
        out.write("Update Successfull!");
    } else {
        out.write("An error occurred please try again.");
    }
%>
<%
dbBean.dbClose();
out.write(pageSource.getPageFooter());%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "MALDI Comparison",
    "MaldiComparisonTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
    <form action="comparemaldisample.jsp">
    <table align="center">
    <tr>
    <td colspan=2>
    <center><h1>MALDI-TOF Compare</h1></center>
    </td>
    </tr>
    <tr>
    <td colspan=2>
    <center>
    <%
        dbBean.dbConnect();
        ResultSet myRS = dbBean.getRS("SELECT sample_id FROM maldi_tof WHERE
maldi_tof_id = " + request.getParameter("maldiid"));
        myRS.next();
        String smpl = myRS.getString("sample_id");
        myRS.close();
        myRS = dbBean.getRS("SELECT concat(identifier.ident, sample.number)
as 'ident' FROM sample, identifier WHERE sample.identifier_id =
identifier.identifier_id AND sample.sample_id = " + smpl );
        myRS.next();
        smpl = myRS.getString("ident");
        myRS.close();
        dbBean.dbClose();
    %>
    </center>
    </td>
    </tr>
    <tr>
    <td colspan=2>
    &nbsp;
    </td>
    </tr>
    <tr>
    <td>
        Identifier:
    </td>
    <td>
        <input type="text" name="rm" value="<%=smpl%>">
    </td>
    </tr>
    <tr>
    <td>
        MALDI-TOF ID:
    </td>
    <td>
        <input type="text" name="id"
value="<%=request.getParameter("maldiid")%>">
    </td>
    </tr>
    <tr>
    <td>
        Error Threshold (ppm):
    </td>

```



```

        <td>
        <input type="text" name="ppm" value="500">
        </td>
    </tr>
    <tr>
        <td>
        Intensity Threshold (%):
        </td>
        <td>
        <input type="text" name="intensity" value=5>
        </td>
    </tr>
    <tr>
        <td>
        Lower Mass/Charge Bound:
        </td>
        <td>
        <input type="text" name="lMass" value="4000">
        </td>
    </tr>
    <tr>
        <td>
        Upper Mass/Charge Bound:
        </td>
        <td>
        <input type="text" name="hMass" value="14000">
        </td>
    </tr>
    <tr>
        <td>
        Maximum Number of Results:
        </td>
        <td>
        <input type="text" name="maxresults" value="25">
        </td>
    </tr>
    <tr>
        <td>
        Ran After (2005-8-15):
        </td>
        <td>
        <input type="text" name="date">
        </td>
    </tr>
    <tr>
        <td colspan=2>&nbsp;
        </td>
    </tr>
    <tr>
        <td colspan=2 align = center>
        Compare Against Only Those That Meet This Search Criteria<br>(Leave
Blank for full search)<br>
        <input type="radio" name="andor" value="and" checked>AND
        <input type="radio" name="andor" value="Or">OR
        </td>
    </tr>
    <tr>
        <td>
        Search by Identification Number (e.g. RM1203 or 1203):
        </td>
        <td>
        <input type="text" name="identifier">
        </td>
    </tr>
    <tr>

```

```

        <td>
            Within Range (e.g. 1203 To: 1403)
        </td>
        <td>
            <input type="text" name="lowRM" size="5">&nbsp; To: &nbsp;<input
type="text" name="highRM" size="5">
        </td>
    </tr>
    <tr>
        <td>
            Search by Genus (e.g. Campylobacter):
        </td>
        <td>
            <input type="text" name="genus">
        </td>
    </tr>
    <tr>
        <td>
            Search by Species (e.g. Coli):
        </td>
        <td>
            <input type="text" name="species">
        </td>
    </tr>
    <tr>
        <td>
            Search by Subspecies (e.g. Fetus):
        </td>
        <td>
            <input type="text" name="subspecies">
        </td>
    </tr>
    <tr>
        <td>
            Search by Other Subtype:
        </td>
        <td>
            <input type="text" name="other">
        </td>
    </tr>
    <tr>
        <td>
            Search Inside Notes:
        </td>
        <td>
            <input type="text" name="notes">
        </td>
    </tr>
    <tr>
        <td>
            Search Inside History:
        </td>
        <td>
            <input type="text" name="history">
        </td>
    </tr>
    <tr>
        <td colspan="2">&nbsp;
    </td>
    </tr>
    <tr>
        <td colspan="2" align="center"><br>
            <input type="reset">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit">
        </td>
    </tr>

```

```
<tr>
  <td colspan=2>
    <center><br>Using the default mass range searches will be much
faster.<br>Editing, or changing the range from 4000 to 14000 requires
re-normalization of the data<br>and can cause search times in excess of 2
minutes.<br>Please be patient.</center>
  </td>
</tr>
</table>
</form>

<%dbBean.dbClose();
out.write(pageSource.getPageFooter());
%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "MALDI Comparison",
"MalDiComparisonTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
        <form action="comparemaldisamplestatanalysis.jsp">
        <table align="center">
        <tr>
        <td colspan=2>
        <center><h1>MALDI-TOF Compare</h1></center>
        </td>
        </tr>
        <tr>
        <td colspan=2>
        <center>
        <%
            dbBean.dbConnect();
            ResultSet myRS = dbBean.getRS("SELECT sample_id FROM maldi_tof WHERE
maldi_tof_id = " + request.getParameter("maldiid"));
            myRS.next();
            String smpl = myRS.getString("sample_id");
            myRS.close();
            myRS = dbBean.getRS("SELECT concat(identifier.ident, sample.number)
as 'ident' FROM sample, identifier WHERE sample.identifier_id =
identifier.identifier_id AND sample.sample_id = " + smpl );
            myRS.next();
            smpl = myRS.getString("ident");
            myRS.close();
            dbBean.dbClose();
        %>
        </center>
        </td>
        </tr>
        <tr>
        <td colspan=2>
        &nbsp;
        </td>
        </tr>
        <tr>
        <td>
            Identifier:
        </td>
        <td>
            <input type="text" name="rm" value="<%=smpl%>">
        </td>
        </tr>
        <tr>
        <td>
            MALDI-TOF ID:
        </td>
        <td>
            <input type="text" name="id"
value="<%=request.getParameter("maldiid")%>">
        </td>
        </tr>
        <tr>
        <td>
            Error Threshold (ppm):
        </td>

```

```

        <td>
        <input type="text" name="ppm" value="500">
        </td>
    </tr>
    <tr>
        <td>
        Intensity Threshold (%):
        </td>
        <td>
        <input type="text" name="intensity" value=10>
        </td>
    </tr>
    <tr>
        <td>
        Lower Mass/Charge Bound:
        </td>
        <td>
        <input type="text" name="lMass" value="4000">
        </td>
    </tr>
    <tr>
        <td>
        Upper Mass/Charge Bound:
        </td>
        <td>
        <input type="text" name="hMass" value="14000">
        </td>
    </tr>
    <tr>
        <td>
        Maximum Number of Results:
        </td>
        <td>
        <input type="text" name="maxresults" value="25">
        </td>
    </tr>
    <tr>
        <td>
        Ran After (2005-8-15):
        </td>
        <td>
        <input type="text" name="date">
        </td>
    </tr>
    <tr>
        <td colspan=2>&nbsp;
        </td>
    </tr>
    <tr>
        <td colspan=2 align = center>
        Use the below text areas to write SQL statements that will be used
        to determine what data to process.
        </td>
    </tr>
    <tr>
        <td colspan=2>
        Select Statement 1:
        </td>
    </tr>
    <tr>
        <td colspan=2>
        <textarea cols="60" name="sqlone"></textarea>
        </td>
    </tr>
    <tr>

```

```

        <td colspan=2>
        Select Statement 2:
        </td>
    </tr>
    <tr>
        <td colspan=2>
        <textarea cols="60" name="sqltwo"></textarea>
        </td>
    </tr>
    <tr>
        <td colspan=2 align = center>
        <input type="radio" name="searchtype" value="intensity"
checked>Intensity Comparison
        <input type="radio" name="searchtype" value="mass">Mass only
        Comparison
        </td>
    </tr>
    <tr>
        <td colspan=2 align=center><br>
        <input type=reset>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit">
        </td>
    </tr>
</table>
</form>

<%dbBean.close();
out.write(pageSource.getPageFooter());
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="frMaldi" class="psm.mformat.formatMALDI" scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MALDI Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <%
        dbBean.dbConnect();
        int rtn;
        rtn =
dbBean.insertMALDI(request.getParameter("ident"),request.getParameter("date"),
request.getParameter("target"),request.getParameter("notes"),frMaldi.getMaldiMZ(),
frMaldi.getMaldiIntensity());
        if (rtn == 1) {
            out.write("This identifier does not exist, please press back and choose an
identifier already in the database.");
        } else if (rtn ==9) {
            out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
        } else if (rtn !=9 ) {
            out.write("Insert successfull!");
        }
        dbBean.dbClose();
    %>
    <%out.write(pageSource.getPageFooter()); %>

```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

    <title>MALDI-TOF DATA and Stick Graph</title>

    <frameset cols="75%,25%">
    <frame
src="maldigraphandoptions.jsp?maldiid=<%=request.getParameter("maldiid")%>&rangeA=40
00&rangeB=14000&topnum=5">
    <frame src="malditofdata.jsp?maldiid=<%=request.getParameter("maldiid")%>">
    </frameset>

</html>
```



```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<%@page import="psm.db.box.*" %>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
    <jsp:useBean id="frMaldi" class="psm.mformat.formatMALDI"
scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "MALDI Comparison",
"MaldiComparisonTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
    %>
    <%
    //SLightly different from the internal database
    //First insert the maldi file sample into a temporary table, which is
terminated at the end of a session
    //Then compare teh results

    //connect to the database
    dbBean.dbConnect();
    int id;
    id = dbBean.insertTempMALDI(frMaldi.getMaldiMZ(),
frMaldi.getMaldiIntensity(), Integer.parseInt(request.getParameter("lMass")),
Integer.parseInt(request.getParameter("hMass")));
    %>

    <table align=center cellpadding="1" cellspacing="4"
style="horizontal-align: center">
        <tr>
            <td colspan="6" align="center">
                <%=request.getParameter("title")%>
            </td>
        </tr>
        <tr>
            <td colspan="6" align=center>
                <a
href="maldifiledatawithgraph.jsp?maldiid=<%=id%>&title=<%=request.getParameter("titl
e")%>">See Stick Graph of MALDI-TOF Data</a>
            </td>
        </tr>
        <tr>
            <td>
                &nbsp;
            </td>
            <td>
                <span style="font-weight: bold;">MALDI-TOF ID</span>
            </td>
            <td>
                <span style="font-weight: bold;">MALDI-TOF Date:</span>
            </td>
            <td>
                <span style="font-weight: bold;">Sample Identifier:</span>
            </td>
            <td>
                <span style="font-weight: bold;">Sample Name:</span>
            </td>
            <td>
                <span style="font-weight: bold;">Score</span>
            </td>
        </tr>
    </table>

```

```

        <td>
        <span style="font-weight: bold;">Compare</span>
        </td>
    </tr>

```

```
<%
```

```

        //Analyze the parameters and make assignments appropriately, if they
are blank redefault them
        int intensity, lmass, hmass, ppm, maxresults;
        boolean massSearch = false;
        if (request.getParameter("searchtype").compareTo("mass") == 0) {
            massSearch = true;
        }
        if (request.getParameter("intensity")!="") {
            intensity = Integer.parseInt(request.getParameter("intensity"));
        } else {
            intensity = 10;
        }

        if (request.getParameter("lMass") != "") {
            lmass = Integer.parseInt(request.getParameter("lMass"));
        } else {
            lmass = 4000;
        }

        if (request.getParameter("hMass") != "") {
            hmass = Integer.parseInt(request.getParameter("hMass"));
        } else {
            hmass = 14000;
        }

        if (request.getParameter("ppm") != "") {
            ppm = Integer.parseInt(request.getParameter("ppm"));
        } else {
            ppm=500;
        }

        if (request.getParameter("maxresults") != "" &
request.getParameter("maxresults") != null) {
            maxresults =
Integer.parseInt(request.getParameter("maxresults"));
        } else {
            maxresults = 10;
        }
        //Get the sql statement
        String sqlSel = "";
        if (massSearch) {
            sqlSel = dbBean.buildCompareStringForTempMassOnly(intensity,
lmass,hmass, ppm);
        } else {
            sqlSel = dbBean.buildCompareStringForTemp(intensity,
lmass,hmass, ppm);
        }
        //Get the array maldi ID list
        ArrayList mID =
dbBean.getAllMaldiDataID(request.getParameter("andor"),request.getParameter("identif
ier"),request.getParameter("lowRM"),
request.getParameter("highRM"),request.getParameter("genus"),request.getParameter("s
pecies"),request.getParameter("subspecies"),request.getParameter("other"),request.ge
tParameter("notes"),request.getParameter("history"), "");
        ArrayList rsIts = new ArrayList();
        dbBean.dbClose();
        dbBean.dbConnect();

```

```

//Loop through each ID and compare it
sqlSel = sqlSel.replace("$cmpr$", Integer.toString(id));

//If the mass range does not fall within 4 to 14 thousand then we
need to normalize the data before comparing it
//If it does fall within this mass range then the data is already
normalized and we can use that instead
if (lmass != 4000 | hmass != 14000) {
    for (int i = 0; i < mID.size(); i++) {

        String l = "";
        l += mID.get(i);
        int cmprToId;
        cmprToId = dbBean.dumpMALDIDataToTemp(Integer.parseInt(l),
intensity, lmass, hmass, "malditemp2");

        String tmp = sqlSel.replace("$cmprTo$",
Integer.toString(cmprToId) );

        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rslts.add(new box(l, myRS.getString("score")));
        myRS.close();
        dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
    }
} else {
    for (int i = 0; i < mID.size(); i++) {

        String l = "";
        l += mID.get(i);

        String tmp = sqlSel.replace("$cmprTo$", l);

        ResultSet myRS = dbBean.getRS(tmp);
        myRS.next();

        rslts.add(new box(l, myRS.getString("score")));
        myRS.close();
    }
}

mID = null;
//Sort the array, this is a simple bubble sort algorithm, it is fast
enough for our purposes
for (int i = 0 ; i< rslts.size(); i++) {
    for (int j=0; j < (rslts.size() - 1 - i) ; j++) {
        box x,y;
        x = (box) rslts.get(j + 1);
        y = (box) rslts.get(j);
        String a = x.getO2();
        String b = y.getO2();
        if (Integer.parseInt(x.getO2()) >
Integer.parseInt(y.getO2()) ) {
            rslts.set(j, x);
            rslts.set(j+1, y);
        }
    }
}
}%<%
for (int i = 0; i < maxresults & i < rslts.size() ; i++) {
    box x = (box) rslts.get(i);
    //Grab all the information pertaining to the result (id's dates,
sample, etc.)

```

```

        ResultSet rsldata = dbBean.getMaldiAndSampleInfo(x.getO1());
        rsldata.next();
        if (massSearch) {
            out.write("<tr><td><input type='checkbox' name='" +
x.getO1() + "'></td><td><a href='maldidatawithgraph.jsp?maldiid=" + x.getO1() + "'>"
+ x.getO1() + "</a></td><td><a href='maldidatawithgraph.jsp?maldiid=" + x.getO1() +
"'>" + rsldata.getString(2) + "</a></td><td><a href='strainDetail.jsp?sampleid=" +
rsldata.getString(5) + "'>" + rsldata.getString(1) + "</a></td><td><a
href='strainDetail.jsp?sampleid=" + rsldata.getString(5) + "'>" +
rsldata.getString(4) + "</a></td><td align=right>" + x.getO2() + "</td><td><a
href='comparefilegraphs.jsp?id1=" + id + "&id2=" + x.getO1() + "&rangeA=" + lmass +
"&rangeB=" + hmass + "&title=" + request.getParameter("title") + "'>Graphs</a>" +
"</td></tr>");
        } else {
            out.write("<tr><td><input type='checkbox' name='" +
x.getO1() + "'></td><td><a href='maldidatawithgraph.jsp?maldiid=" + x.getO1() + "'>"
+ x.getO1() + "</a></td><td><a href='maldidatawithgraph.jsp?maldiid=" + x.getO1() +
"'>" + rsldata.getString(2) + "</a></td><td><a href='strainDetail.jsp?sampleid=" +
rsldata.getString(5) + "'>" + rsldata.getString(1) + "</a></td><td><a
href='strainDetail.jsp?sampleid=" + rsldata.getString(5) + "'>" +
rsldata.getString(4) + "</a></td><td align=right><a
href='scorecomputationfile.jsp?id1=" + id + "&id2=" + x.getO1() + "&ppm=" +
request.getParameter("ppm") + "&intensity=" + request.getParameter("intensity") +
"&lMass=" + request.getParameter("lMass") + "&hMass=" +
request.getParameter("hMass") + "'>" + x.getO2() + "</a></td><td><a
href='comparefilegraphs.jsp?id1=" + id + "&id2=" + x.getO1() + "&rangeA=" + lmass +
"&rangeB=" + hmass + "&title=" + request.getParameter("title") + "'>Graphs</a>" +
"</td></tr>");
        }
        rsldata.close();
    }
    dbBean.dbClose();
%>
</table>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>MALDI-TOF Graph</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="maldifilegraphandoptions.jsp">
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &rangeA=<%=reque
st.getParameter("rangeA")%>&rangeB=<%=request.getParameter("rangeB")%>&topnum=<%=req
uest.getParameter("topnum")%>&title=<%=request.getParameter("title")%>">
            </td>
          </tr>
          <tr>
            <td>
              <div>
                Low Value:&nbsp;<input type="text" name="rangeA"
value="<%=request.getParameter("rangeA")%>">
              </div>
            </td>
            <td>
              <div>
                High Value:&nbsp;<input type="text" name="rangeB"
value="<%=request.getParameter("rangeB")%>">
              </div>
            </td>
            <td>
              <div>
                Number of Values Labeled: <input type="text" name="topnum"
value="<%=request.getParameter("topnum")%>">
              </div>
            </td>
          </tr>
          <tr>
            <td colspan="3"> <br>
              Maldi ID:&nbsp;<input type="text" name="maldiid" value=
<%=request.getParameter("maldiid")%> readonly>
            </td>
          </tr>
          <tr>
            <td>
              <div>
                Title:&nbsp;&nbsp;<input type="text" name="title"
value="<%=request.getParameter("title")%>">
              </div>
            </td>
            <td colspan="2" align = right>
              <input type=submit>
            </td>
          </tr>
        </table>
      </form>
    </body>
  </html>

```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

    <title>MALDI-TOF DATA and Stick Graph</title>

    <frameset cols="75%,25%">
    <frame
src="maldifilegraphandoptions.jsp?maldiid=<%=request.getParameter("maldiid")%>&range
A=4000&rangeB=14000&topnum=5&title=<%=request.getParameter("title")%>">
    <frame src="malditofdatatemp.jsp?maldiid=<%=request.getParameter("maldiid")%>">
    </frameset>

</html>
```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("maldiid");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,620,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,620);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,540);
biContext.drawLine(60,540,810,540);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,545);
biContext.drawLine(53, 540, 60, 540);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 545 - j*5);
        biContext.drawLine(53, 540-j*5, 60, 540-j*5);
    } else {
        biContext.drawLine(58 , 540 -j *5, 60, 540-j*5 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 570);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 570);

        biContext.drawLine(60+150*l, 540, 60+150*l, 555);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 540, 60+(75*l), 550);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 540, 60+lshift, 545);
        }
    }

    dbBean.dbConnect();

    //draw the lines
    biContext = gBean.gDraw(biContext, dbBean.getMaldiGraphData(id, intA, intB), intA,
    intB, topnum, 540, 5, 750);
    biContext.setColor(Color.BLACK);
    biContext = gBean.gDrawTitle(biContext, dbBean.getMaldiAndSampleInfo(id), 870);

    biContext.drawString("Mass to Charge (M/Z)", 365, 610);
    FontMetrics fm = biContext.getFontMetrics();
    biContext.rotate(-Math.PI / 2, 10, 325 - (fm.getHeight()/2));
    biContext.drawString("% Intensity", 10, 325);

    OutputStream output = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
    JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
    param.setQuality((float) 1, false);
    encoder.setJPEGEncodeParam(param);
    encoder.encode(bImg);
    output.close();
    dbBean.dbClose();
    %>

```



```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>MALDI-TOF Graph</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="maldigraphandoptions.jsp">
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &rangeA=<%=request
.getParameter("rangeA")%>&rangeB=<%=request.getParameter("rangeB")%>&topnum=<%=reque
st.getParameter("topnum")%>">
            </td>
          </tr>
          <tr>
            <td>
              Low Value:&nbsp;<input type="text" name="rangeA"
value="<%=request.getParameter("rangeA")%>">
            </td>
            <td>
              High Value:&nbsp;<input type="text" name="rangeB"
value="<%=request.getParameter("rangeB")%>">
            </td>
            <td>
              Number of Values Labeled: <input type="text" name="topnum"
value="<%=request.getParameter("topnum")%>">
            </td>
          </tr>
          <tr>
            <td colspan = "3"> <br>
              Maldi ID:&nbsp;<input type="text" name="maldiid" value=
<%=request.getParameter("maldiid")%> readonly>
            </td>
          </tr>
          <tr>
            <td colspan = 3 align = right>
              <input type=submit>
            </td>
          </tr>
          <tr>
            <td colspan=3 align=center>
              <a href="maldicomparesetup.jsp?maldiid=<%=request.getParameter("maldiid")%>"
target="_top">Run Database Comparison</a>
            </td>
          </tr>
        </table>
      </form>
    </body>
  </html>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("maldiid");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,400,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,600);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,340);
biContext.drawLine(60,340,810,340);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,345);
biContext.drawLine(53, 340, 60, 340);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 345 - j*3);
        biContext.drawLine(53, 340-j*3, 60, 340-j*3);
    } else {
        biContext.drawLine(58 , 340 -j *3, 60, 340-j*3 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 370);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 370);

        biContext.drawLine(60+150*l, 340, 60+150*l, 355);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 340, 60+(75*l), 350);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0 ) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 340, 60+lshift, 345);
        }
    }
}

dbBean.dbConnect();

//draw the lines
biContext = gBean.gDraw(biContext, dbBean.getMaldiGraphData(id, intA, intB), intA,
intB, topnum, 340, 3,750);
biContext.setColor(Color.BLACK);
biContext = gBean.gDrawTitle(biContext, dbBean.getMaldiAndSampleInfo(id), 870);

biContext.drawString("Mass to Charge (M/Z)", 375, 390);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 230 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,230);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("maldiid");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,415,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,620);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,340);
biContext.drawLine(60,340,810,340);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,345);
biContext.drawLine(53, 340, 60, 340);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 345 - j*3);
        biContext.drawLine(53, 340-j*3, 60, 340-j*3);
    } else {
        biContext.drawLine(58 , 340 -j *3, 60, 340-j*3 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 370);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 370);

        biContext.drawLine(60+150*l, 340, 60+150*l, 355);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 340, 60+(75*l), 350);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0) {
            double shift = l * 37.5;
            int Ishift = (int) shift;
            biContext.drawLine(60+Ishift, 340, 60+Ishift, 345);
        }
    }

    dbBean.dbConnect();

    //draw the lines
    biContext = gBean.gDraw(biContext, dbBean.getTempMaldiGraphData(id, intA, intB),
    intA, intB, topnum, 340, 3,750);
    biContext.setColor(Color.BLACK);

    String myTitle = request.getParameter("title");
    if (myTitle == null) { myTitle = "";}
    biContext.drawString(myTitle, 435 - ((myTitle.length()/2)*6) , 10);
    biContext.drawString("Mass to Charge (M/Z)", 365, 410);
    FontMetrics fm = biContext.getFontMetrics();
    biContext.rotate(-Math.PI / 2, 10, 230 - (fm.getHeight()/2));
    biContext.drawString("% Intensity", 10,230);

    OutputStream output = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
    encoder.encode(bImg);
    output.close();
    dbBean.dbClose();
    %>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("maldiid");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,620,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,620);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,540);
biContext.drawLine(60,540,810,540);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,545);
biContext.drawLine(53, 540, 60, 540);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 545 - j*5);
        biContext.drawLine(53, 540-j*5, 60, 540-j*5);
    } else {
        biContext.drawLine(58 , 540 -j *5, 60, 540-j*5 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 570);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 570);

        biContext.drawLine(60+150*l, 540, 60+150*l, 555);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 540, 60+(75*l), 550);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 540, 60+lshift, 545);
        }
    }

    dbBean.dbConnect();

    //draw the lines
    biContext = gBean.gDraw(biContext, dbBean.getTempMaldiGraphData(id, intA, intB),
    intA, intB, topnum, 540, 5,750);
    biContext.setColor(Color.BLACK);

    String myTitle = request.getParameter("title");
    if (myTitle == null) { myTitle = "";}
    biContext.drawString(myTitle, 435 - ((myTitle.length()/2)*6) , 10);
    biContext.drawString("Mass to Charge (M/Z)", 365, 610);
    FontMetrics fm = biContext.getFontMetrics();
    biContext.rotate(-Math.PI / 2, 10, 325 - (fm.getHeight()/2));
    biContext.drawString("% Intensity", 10,325);

    OutputStream output = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
    JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
    param.setQuality((float) 1, false);
    encoder.setJPEGEncodeParam(param);
    encoder.encode(bImg);
    output.close();
    dbBean.dbClose();
    %>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("maldiid");

Font font = new Font("ARIAL", Font.BOLD ,10);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(450,200,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,450,200);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,140);
biContext.drawLine(60,140,410,140);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,145);
biContext.drawLine(53, 140, 60, 140);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 145 - j);
        biContext.drawLine(53, 140-j, 60, 140-j);
    }/* else {
        biContext.drawLine(58 , 140 -j, 60, 140-j );
    }*/
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 170);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```



```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 70*(l) - centerint) , 170);
        biContext.drawLine(60+70*l, 140, 60+70*l, 155);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(35*l), 140, 60+(35*l), 150);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0 ) {
            double shift = l * 17.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 140, 60+lshift, 145);
        }
    }

    dbBean.dbConnect();

    //draw the lines
    biContext = gBean.gDraw(biContext, dbBean.getMaldiGraphData(id, intA, intB), intA,
        intB, topnum, 140, 1, 350);
    biContext.setColor(Color.BLACK);
    biContext = gBean.gDrawTitle(biContext, dbBean.getMaldiAndSampleInfo(id), 550);

    biContext.drawString("Mass to Charge (M/Z)", 180, 190);
    FontMetrics fm = biContext.getFontMetrics();
    biContext.rotate(-Math.PI / 2, 10, 230 - (fm.getHeight()/2));
    biContext.drawString("% Intensity", 10,70);

    OutputStream output = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
    JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
    param.setQuality((float) 1, false);
    encoder.setJPEGEncodeParam(param);
    encoder.encode(bImg);
    output.close();
    dbBean.dbClose();
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>MALDI-TOF Data</title>
</head>
<body>

<h1>MALDI-TOF Data</h1>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session" />

<b>Maldi ID:</b> <%=request.getParameter("maldiid")%><br>

<%
dbBean.dbConnect();
ResultSet myRS;
myRS = dbBean.getMaldiData(request.getParameter("maldiid"));
if (myRS.next()) {
    myRS.first();
%>
<table>
<th>Mass</th>
<th>Intensity</th>

<%
do {
%>

<tr>
<td>
<a
href="genesearchresults.jsp?locusstr=&strainstr=&genusstr=&speciesstr=&locationstr=&
location2str=&mw=<%=myRS.getString("mass")%>&ppm=500&productstr=&notesstr=&locus=on&
strain=on&genusspecies=on&location1=on&location2=on&start=on&end=on&GBK=on&fr=on&NT=
on&AA=on&MW=on&iip=on&gene=on&code=on&COG=on&ec1=on&ec2=on&product=on&notes=on"
target="_top"><%=myRS.getString("mass")%></a>
</td>
<td style="text-align: right;">
<%
double d = (double) myRS.getDouble("intensity");
out.write(Double.toString(d));
%>
</td>

<%} while (myRS.next());
} else {%>
    There is no MALDI-TOF data for this sample.
<%}

myRS.close();
dbBean.dbClose();
%>

</table>
</body>
<br>
<a href="deletemaldiconfirm.jsp?maldiid=<%=request.getParameter("maldiid")%>"

```

```
target="_top">DELETE Data</a>
</html>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>MALDI-TOF Data</title>
</head>
<body>

<h1>MALDI-TOF Data</h1>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session" />

<b>Your MALDI Data</b>

<%
dbBean.dbConnect();
ResultSet myRS;
myRS = dbBean.getTempMaldiData(request.getParameter("maldiid"));
if (myRS.next()) {
    myRS.first();
%>
<table>
<th>Mass</th>
<th>Intensity</th>

<%
do {
%>

<tr>
<td>
<a
href="genesearchresults.jsp?locusstr=&strainstr=&genusstr=&speciesstr=&locationstr=&
location2str=&mw=<%=myRS.getString("mass")%>&ppm=500&productstr=&notesstr=&locus=on&
strain=on&genusspecies=on&location1=on&location2=on&start=on&end=on&GBK=on&fr=on&NT=
on&AA=on&MW=on&iEP=on&gene=on&code=on&COG=on&ec1=on&ec2=on&product=on&notes=on"
target="_top"><%=myRS.getString("mass")%></a>
</td>
<td>
<%=myRS.getString("intensity")%>
</td>

<%} while (myRS.next());
} else {%>
    There is no MALDI-TOF data for this sample.
<%}

myRS.close();
dbBean.dbClose();
%>
</table>
</body>
</html>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%

String[] maldiids;
malidiids = request.getParameterValues("checkid");
if (malidiids != null ) {
    int x,y,l,m;
    //x = Integer.parseInt(request.getParameter("x"));
    //y = Integer.parseInt(request.getParameter("y"));
    x = 900;
    y = 690;

    if (malidiids[0].compareTo("on") == 0) {
        l = 1;
        m = 2;
    } else {
        l = 0;
        m = 1;
    }
    //Initial Setup
    BufferedImage bImg = new BufferedImage(x,y,BufferedImage.TYPE_INT_RGB);
    response.setContentType("image/jpeg");
    Graphics2D biContext = gBean.mirroredGraphs(bImg, dbBean,
Integer.parseInt(request.getParameter("rangeA")),
Integer.parseInt(request.getParameter("rangeB")), Integer.parseInt(malidiids[l]),
Integer.parseInt(malidiids[m]),10, x,y,100);

    //wrap up
    OutputStream output = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
    JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
    param.setQuality((float) 1, false);
    encoder.setJPEGEncodeParam(param);
    encoder.encode(bImg);
    output.close();
} else {
    out.write("Please select at least one strain to plot.");
}
%>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%

String[] maldiids;
malidiids = request.getParameterValues("checkid");
if (malidiids != null ) {
    int x,y,l,m;
    //x = Integer.parseInt(request.getParameter("x"));
    //y = Integer.parseInt(request.getParameter("y"));
    x = 900;
    y = 690;

    if (malidiids[0].compareTo("on") == 0) {
        l = 1;
        m =2;
    } else {
        l =0;
        m = 1;
    }
    //Initial Setup
    BufferedImage bImg = new BufferedImage(x,y,BufferedImage.TYPE_INT_RGB);
    response.setContentType("image/jpeg");
    Graphics2D biContext = gBean.mirroredMSMSGraphs(bImg, dbBean,
    Integer.parseInt(request.getParameter("rangeA")),
    Integer.parseInt(request.getParameter("rangeB")), Integer.parseInt(malidiids[l]),
    Integer.parseInt(malidiids[m]),10, x,y,100);

    dbBean.dbConnect();
    biContext = gBean.gDrawMSMSTitle(biContext,
    dbBean.getMSMSAndSampleInfo(malidiids[l]), 900, 70);

    biContext = gBean.gDrawMSMSTitle(biContext,
    dbBean.getMSMSAndSampleInfo(malidiids[m]), 900, 630);

    dbBean.dbClose();

    //Wrap up
    OutputStream output = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
    JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
    param.setQuality((float) 1, false);
    encoder.setJPEGEncodeParam(param);
    encoder.encode(bImg);
    output.close();
} else {
    out.write("Please select at least one strain to plot.");
}
%>

```

```

<%@page contentType="text"%><%@page pageEncoding="UTF-8"%><%@page
import="java.sql.*" %><jsp:useBean id="dbBean" class="psm.db.dbAccess" scope =
"session" />
In Silico<%out.write("\t");%>Ion
Type<%out.write("\t");%>Residue<%out.write("\t");%>MS/MS<%out.write("\t");%>Intensit
y
    <%out.write("\n");

    int lmass = 4000, hmass = 14000, intensity=2;
    String masserror = "2.5";

    if (request.getParameter("rangeA") != "") {
        lmass = Integer.parseInt(request.getParameter("rangeA"));
    }

    if (request.getParameter("rangeB") != "") {
        hmass = Integer.parseInt(request.getParameter("rangeB"));
    }

    if (request.getParameter("masserror") != "") {
        masserror = request.getParameter("masserror");
    }

    if (request.getParameter("intensity") != "") {
        intensity = Integer.parseInt(request.getParameter("intensity"));
    }
    ResultSet myRS;

    String getcmprdata = "SELECT r.insilico, r.msms, r.intensity, y.iontype,
y.residue FROM (SELECT b.mass as insilico, '' as msms, '' as intensity FROM
maldiTemp2 b WHERE b.intensity > $minIntensity$ and b.mass between $lowMass$ and
$highMass$ and b.maldi_tof_id = $cmprTo$ AND b.mass not in (SELECT insilico FROM
(select b.mass as insilico, a.mass as msms from maldiTemp a, maldiTemp2 b where
a.intensity > $minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between
(b.mass - $error$) and (b.mass + $error$) and a.mass between $lowMass$ and
$highMass$ and b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$
and b.maldi_tof_id = $cmprTo$) t) UNION select b.mass as insilico, a.mass as msms,
a.intensity as intensity from maldiTemp a, maldiTemp2 b where a.intensity >
$minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between (b.mass -
$error$) and (b.mass + $error$) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$ order by insilico) r, in_silico_data y WHERE r.insilico =
y.mass AND y.in_silico_id = $id2$";
    getcmprdata = getcmprdata.replace("$minIntensity$",
request.getParameter("intensity"));
    getcmprdata = getcmprdata.replace("$error$",
request.getParameter("masserror"));
    getcmprdata = getcmprdata.replace("$lowMass$",
request.getParameter("rangeA"));
    getcmprdata = getcmprdata.replace("$highMass$",
request.getParameter("rangeB"));
    getcmprdata = getcmprdata.replace("$id2$", request.getParameter("id2"));

    Thread.currentThread().sleep(700);
    dbBean.dbConnect();
    int maldiTempId =
dbBean.dumpMSMSDataToTemp(Integer.parseInt(request.getParameter("id1")), intensity,
lmass, hmass, "maldiTemp");
    int cmprToId =
dbBean.dumpINSILICODataToTemp(Integer.parseInt(request.getParameter("id2")),
intensity, lmass, hmass, "maldiTemp2", request.getParameter("residue"));

    getcmprdata = getcmprdata.replace("$cmpr$", Integer.toString(maldiTempId));
    getcmprdata = getcmprdata.replace("$cmprTo$", Integer.toString(cmprToId));

```

```
myRS = dbBean.getRS(getcmprdata);
if (myRS.next()) {
myRS.first();
do {
out.write(myRS.getString("insilico") + "\t" + myRS.getString("iontype") +
"\t" + myRS.getString("residue") + "\t" + myRS.getString("msms") + "\t" +
myRS.getString("intensity") + "\n");
} while (myRS.next());
} else {%>
There is no MS/MS data for this sample.
<%}
myRS.close();
dbBean.dbClose();
%>
```



```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

    <title>MS/MS In Silico Compare</title>

    <frameset cols="75%,25%">
        <frame
src="comparemsinsilicographs.jsp?id1=<%=request.getParameter("id1")%>&id2=<%=request
.getParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<%=request
.getParameter("rangeB")%>&topnum=1&intensity=<%=request.getParameter("intensity")%>&m
asserror=<%=request.getParameter("masserror")%>&residue=<%=request.getParameter("res
idue")%>">
        <frame
src="msinsilicodatacompare.jsp?id1=<%=request.getParameter("id1")%>&id2=<%=request.g
etParameter("id2")%>&rangeA=<%=request.getParameter("rangeA")%>&rangeB=<%=request.ge
tParameter("rangeB")%>&intensity=<%=request.getParameter("intensity")%>&masserror=<%
=request.getParameter("masserror")%>&residue=<%=request.getParameter("residue")%>">
    </frameset>

</html>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MS/MS Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form method=post action="checkmsmsparse.jsp" enctype="MULTIPART/FORM-DATA">
    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1>Upload MS/MS Data</h1></center>
            </td>
        </tr>
        <tr>
            <td>
                Find File:
            </td>
            <td>
                <input type="file" name="maldifile">
            </td>
        </tr>
        <tr>
            <td></td>
            <td colspan="1" align=left>
                <input type="submit" value="Send">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "MS/MS To <i>In Silico</i>
Comparison", "MaldiComparisonTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
    <form action="comparemsmssample.jsp">
    <table align="center">
    <tr>
        <td colspan=2>
        <center>
        <%
            dbBean.dbConnect();
            ResultSet myRS = dbBean.getRS("SELECT sample_id FROM ms_ms WHERE
ms_ms_id = " + request.getParameter("msmsid"));
            myRS.next();
            String smp1 = myRS.getString("sample_id");
            myRS.close();
            myRS = dbBean.getRS("SELECT concat(identifier.ident, sample.number)
as 'ident' FROM sample, identifier WHERE sample.identifier_id =
identifier.identifier_id AND sample.sample_id = " + smp1 );
            myRS.next();
            smp1 = myRS.getString("ident");
            myRS.close();
            myRS = dbBean.getRS("SELECT * FROM ms_ms WHERE ms_ms_id = " +
request.getParameter("msmsid"));
            myRS.next();
            String avgmass = myRS.getString("precursor_ion_mtoz");
            myRS.close();
            dbBean.dbClose();
        %>
        </center>
        </td>
    </tr>
    <tr>
        <td>
            Identifier:
        </td>
        <td>
            <input type="text" name="rm" value="<%=smp1%>" readonly="true">
        </td>
    </tr>
    <tr>
        <td>
            MS/MS ID:
        </td>
        <td>
            <input type="text" name="id"
value="<%=request.getParameter("msmsid")%>" readonly="true">
        </td>
    </tr>
    <tr>
        <td>
            Intensity Threshold (%):
        </td>
        <td>
            <input type="text" name="intensity" value="2">
        </td>
    </tr>
    <tr>

```

```

        <td>
        Lower Mass/Charge Bound:
        </td>
        <td>
        <input type="text" name="lMass" value="0">
        </td>
    </tr>
    <tr>
        <td>
        Upper Mass/Charge Bound:
        </td>
        <td>
        <input type="text" name="hMass" value="14000">
        </td>
    </tr>
    <tr>
        <td>
        Maximum Number of Results:
        </td>
        <td>
        <input type="text" name="maxresults" value="25">
        </td>
    </tr>
    <tr>
        <td>
        Ran After (2005-8-15):
        </td>
        <td>
        <input type="text" name="date">
        </td>
    </tr>
    <tr>
        <td>
        Search by Average Molecular Weight:
        </td>
        <td>
        <input type="text" name="mass" value="<%=avgmass%>">
        </td>
    </tr>
    <tr>
        <td>
        Protein AMW Tolerance (daltons):
        </td>
        <td>
        <input type="text" name="masserror" value="5">
        </td>
    </tr>
    <tr>
        <td>
        Fragment Ion Tolerance (Th):
        </td>
        <td>
        <input type="text" name="fierror" value="2.5">
        </td>
    </tr>
    <tr>
        <td>
        Residues (e.g. DEP):
        </td>
        <td>
        <input type="text" name="residue">
        </td>
    </tr>
    <tr>
        <td colspan="2" align="center">

```

Score ALPHA	<input name="sort" type="radio" value="scorea"/> Sort by USDA
Score BETA	<input name="sort" type="radio" value="scoreb"/> Sort by USDA
<i>P</i> Value	<input checked="checked" name="sort" type="radio" value="pval"/> Sort By
Score ALPHA	<input checked="checked" name="calcscorea" type="checkbox"/> Calculate USDA
Score BETA	<input checked="checked" name="calcscoreb" type="checkbox"/> Calculate USDA
<i>P</i> Value	<input checked="checked" name="calcpval" type="checkbox"/> Calculate
Compare Against Only Those That Meet This Search Criteria (Leave Blank for full search)	
	<input checked="checked" name="andor" type="radio" value="and"/> AND
	<input name="andor" type="radio" value="or"/> OR
Search by Identification (e.g. NCBI1203 or 1203):	
	<input name="identifier" type="text"/>
Search by Genus (e.g. Campylobacter):	
	<input name="genus" type="text"/>
Search by Species (e.g. Coli):	
	<input name="species" type="text"/>
Search by Subspecies (e.g. Fetus):	
	<input name="subspecies" type="text"/>
Search by Other Subtype:	

```

        </td>
        <td>
            <input type="text" name="other">
        </td>
    </tr>
    <tr>
        <td>
            Search Protein:
        </td>
        <td>
            <input type="text" name="protein">
        </td>
    </tr>
    <tr>
        <td>
            Search Inside Notes:
        </td>
        <td>
            <input type="text" name="notes">
        </td>
    </tr>
    <tr>
        <td colspan="2" align="center"><br>
        <input type="reset">&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit">
        </td>
    </tr>
</table>
</form>

```

```

<%dbBean.dbClose();
out.write(pageSource.getPageFooter());
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="frMSMS" class="psm.mformat.formatMSMSData" scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MS/MS Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <%
        dbBean.dbConnect();
        int rtn;
        rtn =
dbBean.insertMSMS(request.getParameter("ident"),request.getParameter("date"),
request.getParameter("precionmass"),request.getParameter("notes"),
request.getParameter("protein"),frMSMS.getMaldiMZ(), frMSMS.getMaldiIntensity());
        if (rtn == 1) {
            out.write("This identifier does not exist, please press back and choose an
identifier already in the database.");
        } else if (rtn !=0) {
            out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
        } else if (rtn ==0 ) {
            out.write("Insert successfull!");
        }
        dbBean.dbClose();
    %>
    <%out.write(pageSource.getPageFooter()); %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>MS/MS Data</title>
</head>
<body>

<h1>MS/MS Data</h1>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session" />

<b>MS/MS ID:</b> <%=request.getParameter("msmsid")%><br>
<%
dbBean.dbConnect();
ResultSet myRS;
myRS = dbBean.getRS("SELECT CONCAT_WS(' ', genus.name, species.name,
subspecies.name, subtype.name) FROM sample, genus, species, subspecies, subtype,
ms_ms, strain WHERE strain.strain_id = sample.strain_id AND strain.genus_id =
genus.genus_id AND strain.species_id = species.species_id AND strain.subspecies_id =
subspecies.subspecies_id AND strain.subtype_id = subtype.subtype_id AND
ms_ms.sample_id = sample.sample_id AND ms_ms.ms_ms_id = " +
request.getParameter("msmsid"));
myRS.next();
out.write(myRS.getString(1));
myRS.close();
myRS = dbBean.getMMSData(request.getParameter("msmsid"));
if (myRS.next()) {
myRS.first();
%>
<br>
<table>
<th>Mass</th>
<th>Intensity</th>

<%
do {
%>

<tr>
<td>
<%=myRS.getString("mass")%>
</td>
<td style="text-align: right;">
<%
double d = (double) myRS.getDouble("intensity");
out.write(Double.toString(d));
%>
</td>

<%} while (myRS.next());
} else {%>
There is no MS/MS data for this sample.
<%}

myRS.close();
dbBean.dbClose();

```



```
%>
    </table>
    <br>
    <a href="./deletemsmsconfirm.jsp?msmsid=<%=request.getParameter("msmsid")%>"
target="_top">DELETE Data</a><br>
    </body>
</html>
```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

    <title>MALDI-TOF DATA and Stick Graph</title>

    <frameset cols="75%,25%">
    <frame
src="msmsgraphandoptions.jsp?msmsid=<%=request.getParameter("msmsid")%>&rangeA=0&ran
geB=14000&topnum=5">
    <frame src="msmsdata.jsp?msmsid=<%=request.getParameter("msmsid")%>">
    </frameset>

</html>
```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("msmsid");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,620,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,620);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,540);
biContext.drawLine(60,540,810,540);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,545);
biContext.drawLine(53, 540, 60, 540);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 545 - j*5);
        biContext.drawLine(53, 540-j*5, 60, 540-j*5);
    } else {
        biContext.drawLine(58 , 540 -j *5, 60, 540-j*5 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 570);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 570);

        biContext.drawLine(60+150*l, 540, 60+150*l, 555);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 540, 60+(75*l), 550);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 540, 60+lshift, 545);
        }
    }
}

dbBean.dbConnect();

//draw the lines
biContext = gBean.gDraw(biContext, dbBean.getMSMSGGraphData(id, intA, intB), intA,
intB, topnum, 540, 5, 750);
biContext.setColor(Color.BLACK);
biContext = gBean.gDrawMSMSTitle(biContext, dbBean.getMSMSAndSampleInfo(id), 870);

biContext.drawString("Mass to Charge (M/Z)", 365, 610);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 325 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,325);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>MS/MS Graph</title>
  </head>
  <body>
    <center>If the graph does not appear to resemble the proper data, click refresh
or submit.<br>Your browser may have cached the previous image. <br><br></center>
    <form action="msmsgraphandoptions.jsp">
      <table border="0" cellpadding = "0" cellspacing = "0" align=center >
        <tr>
          <td colspan="3">
            &rangeA=<%=request.ge
tParameter("rangeA")%>&rangeB=<%=request.getParameter("rangeB")%>&topnum=<%=request.
getParameter("topnum")%>">
            </td>
          </tr>
          <tr>
            <td>
              Low Value:&nbsp;<input type="text" name="rangeA"
value="<%=request.getParameter("rangeA")%>">
            </td>
            <td>
              High Value:&nbsp;<input type="text" name="rangeB"
value="<%=request.getParameter("rangeB")%>">
            </td>
            <td>
              Number of Values Labeled: <input type="text" name="topnum"
value="<%=request.getParameter("topnum")%>">
            </td>
          </tr>
          <tr>
            <td colspan = "3"> <br>
              MS/MS:&nbsp;<input type="text" name="msmsid" value=
<%=request.getParameter("msmsid")%> readonly>
            </td>
          </tr>
          <tr>
            <td colspan = 3 align = right>
              <input type=submit>
            </td>
          </tr>
          <tr>
            <td colspan=3 align=center>
              <a href="msmscomparesetup.jsp?msmsid=<%=request.getParameter("msmsid")%>"
target="_top">Run Database <i>In Silico</i> Comparison</a><br>
              <a
href="msmstommscomparesetup.jsp?msmsid=<%=request.getParameter("msmsid")%>"
target="_top">Run Database MS/MS Comparison</a>
            </td>
          </tr>
        </table>
      </form>
    </body>
  </html>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("msmsid");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,400,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,600);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,340);
biContext.drawLine(60,340,810,340);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,345);
biContext.drawLine(53, 340, 60, 340);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 345 - j*3);
        biContext.drawLine(53, 340-j*3, 60, 340-j*3);
    } else {
        biContext.drawLine(58 , 340 -j *3, 60, 340-j*3 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 370);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 370);

        biContext.drawLine(60+150*l, 340, 60+150*l, 355);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 340, 60+(75*l), 350);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 340, 60+lshift, 345);
        }
    }
}

dbBean.dbConnect();

//draw the lines
biContext = gBean.gDraw(biContext, dbBean.getMMSGGraphData(id, intA, intB), intA,
intB, topnum, 340, 3,750);
biContext.setColor(Color.BLACK);
biContext = gBean.gDrawMSMSTitle(biContext, dbBean.getMSMSAndSampleInfo(id), 870);

biContext.drawString("Mass to Charge (M/Z)", 375, 390);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 230 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,230);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));
Thread.sleep(200);
String id = request.getParameter("id1");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,400,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,600);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,340);
biContext.drawLine(60,340,810,340);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,345);
biContext.drawLine(53, 340, 60, 340);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 345 - j*3);
        biContext.drawLine(53, 340-j*3, 60, 340-j*3);
    } else {
        biContext.drawLine(58 , 340 -j *3, 60, 340-j*3 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 370);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```



```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 370);

        biContext.drawLine(60+150*l, 340, 60+150*l, 355);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 340, 60+(75*l), 350);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0 ) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 340, 60+lshift, 345);
        }
    }
}

dbBean.dbConnect();

//draw the lines
biContext = gBean.gDrawPlusInSilicInfo(biContext,
dbBean.getInSilicoMSMSMatchedData(Integer.parseInt(request.getParameter("id1")),
Integer.parseInt(request.getParameter("id2")), request.getParameter("intensity"),
request.getParameter("masserror"), request.getParameter("rangeA"),
request.getParameter("rangeB"), request.getParameter("residue")), intA, intB,
topnum, 340, 3,750);
biContext.setColor(Color.BLACK);
biContext = gBean.gDrawMSMSTitle(biContext, dbBean.getMSMSAndSampleInfo(id), 870);

biContext.drawString("Mass to Charge (M/Z)", 375, 390);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 230 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,230);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));
Thread.sleep(200);
String id = request.getParameter("id1");

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,400,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,600);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,340);
biContext.drawLine(60,340,810,340);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,345);
biContext.drawLine(53, 340, 60, 340);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 345 - j*3);
        biContext.drawLine(53, 340-j*3, 60, 340-j*3);
    } else {
        biContext.drawLine(58 , 340 -j *3, 60, 340-j*3 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 370);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 150*(l) - centerint) , 370);

        biContext.drawLine(60+150*l, 340, 60+150*l, 355);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(75*l), 340, 60+(75*l), 350);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0 ) {
            double shift = l * 37.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 340, 60+lshift, 345);
        }
    }
}

dbBean.dbConnect();

//draw the lines
biContext = gBean.gDrawPlusInSilicInfoSingle(biContext,
dbBean.getInSilicoMSMSMatchedData(Integer.parseInt(request.getParameter("id1")),
Integer.parseInt(request.getParameter("id2")), request.getParameter("intensity"),
request.getParameter("masserror"), request.getParameter("rangeA"),
request.getParameter("rangeB"), request.getParameter("residue")), intA, intB,
topnum, 340, 3,750);
biContext.setColor(Color.BLACK);
biContext = gBean.gDrawMSMSTitle(biContext, dbBean.getMSMSAndSampleInfo(id), 870);

biContext.drawString("Mass to Charge (M/Z)", 375, 390);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 230 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,230);

OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));

String id = request.getParameter("msmsid");

Font font = new Font("ARIAL", Font.BOLD ,10);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(450,200,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,450,200);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,140);
biContext.drawLine(60,140,410,140);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,145);
biContext.drawLine(53, 140, 60, 140);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 145 - j);
        biContext.drawLine(53, 140-j, 60, 140-j);
    }/* else {
        biContext.drawLine(58 , 140 -j, 60, 140-j );
    }*/
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 170);

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

```

```

        if (num.length()%2 !=0) {
            centerint+= 3;
        }

        biContext.drawString(num, (60 + 70*(l) - centerint) , 170);
        biContext.drawLine(60+70*l, 140, 60+70*l, 155);
    }

    for (l=1; l <=10; l++) {
        biContext.drawLine(60+(35*l), 140, 60+(35*l), 150);
    }

    for (l=1; l <= 20; l++) {
        if (l % 2 != 0 ) {
            double shift = l * 17.5;
            int lshift = (int) shift;
            biContext.drawLine(60+lshift, 140, 60+lshift, 145);
        }
    }

    dbBean.dbConnect();

    //draw the lines
    biContext = gBean.gDraw(biContext, dbBean.getMMSGGraphData(id, intA, intB), intA,
        intB, topnum, 140, 1, 350);
    biContext.setColor(Color.BLACK);
    biContext = gBean.gDrawMSMSTitle(biContext, dbBean.getMSMSAndSampleInfo(id), 550);

    biContext.drawString("Mass to Charge (M/Z)", 180, 190);
    FontMetrics fm = biContext.getFontMetrics();
    biContext.rotate(-Math.PI / 2, 10, 230 - (fm.getHeight()/2));
    biContext.drawString("% Intensity", 10,70);

    OutputStream output = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
    JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
    param.setQuality((float) 1, false);
    encoder.setJPEGEncodeParam(param);
    encoder.encode(bImg);
    output.close();
    dbBean.dbClose();
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "MS/MS-MS/MS Comparison",
"MalDiComparisonTitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <form action="comparemsmstomsmsssample.jsp">
    <table align="center">
    <tr>
        <td colspan=2>
        <center>
        <%
            dbBean.dbConnect();
            ResultSet myRS = dbBean.getRS("SELECT sample_id FROM ms_ms WHERE
ms_ms_id = " + request.getParameter("msmsid"));
            myRS.next();
            String smpl = myRS.getString("sample_id");
            myRS.close();
            myRS = dbBean.getRS("SELECT concat(identifier.ident, sample.number)
as 'ident' FROM sample, identifier WHERE sample.identifier_id =
identifier.identifier_id AND sample.sample_id = " + smpl );
            myRS.next();
            smpl = myRS.getString("ident");
            myRS.close();
            myRS = dbBean.getRS("SELECT * FROM ms_ms WHERE ms_ms_id = " +
request.getParameter("msmsid"));
            myRS.next();
            String avgmass = myRS.getString("precursor_ion_mtoz");
            myRS.close();
            dbBean.dbClose();
        %>
        </center>
        </td>
    </tr>
    <tr>
        <td>
            Identifier:
        </td>
        <td>
            <input type="text" name="rm" value="<%=smpl%>" readonly="true">
        </td>
    </tr>
    <tr>
        <td>
            MS/MS ID:
        </td>
        <td>
            <input type="text" name="id"
value="<%=request.getParameter("msmsid")%>" readonly="true">
        </td>
    </tr>
    <tr>
        <td>
            Intensity Threshold (%):
        </td>
        <td>
            <input type="text" name="intensity" value="2">
        </td>
    </tr>
    <tr>

```

```

        <td>
        Lower Mass/Charge Bound:
        </td>
        <td>
        <input type="text" name="lMass" value="0">
        </td>
    </tr>
    <tr>
        <td>
        Upper Mass/Charge Bound:
        </td>
        <td>
        <input type="text" name="hMass" value="14000">
        </td>
    </tr>
    <tr>
        <td>
        Maximum Number of Results:
        </td>
        <td>
        <input type="text" name="maxresults" value="25">
        </td>
    </tr>
    <tr>
        <td>
        Ran After (2005-8-15):
        </td>
        <td>
        <input type="text" name="date">
        </td>
    </tr>
    <tr>
        <td>
        Average Molecular Weight:
        </td>
        <td>
        <input type="text" name="mass" value="<%=avgmass%>">
        </td>
    </tr>
    <tr>
        <td>
        Protein AMW Error (daltons):
        </td>
        <td>
        <input type="text" name="masserror" value="5">
        </td>
    </tr>
    <tr>
        <td>
        Fragment Ion Error (Th):
        </td>
        <td>
        <input type="text" name="fierror" value="2.5">
        </td>
    </tr>
    <tr>
        <td colspan="2" align = center>
        <input type="radio" name="sort" value="pval" checked>Sort By
        <input type="radio" name="sort" value="score">Sort by USDA Score
        </td>
    </tr>
    <tr>
        <td colspan="2">&nbsp;
        </td>
    </tr>

```

```

    </tr>
    <tr>
      <td colspan=2 align = center>
        Compare Against Only Those That Meet This Search Criteria<br>(Leave
Blank for full search)<br>
        <input type="radio" name="andor" value="and" checked>AND
        <input type="radio" name="andor" value="Or">OR
      </td>
    </tr>
    <tr>
      <td>
        Search by Identification (e.g. NCBI1203 or 1203):
      </td>
      <td>
        <input type="text" name="identifier">
      </td>
    </tr>
    <tr>
      <td>
        Search by Genus (e.g. Campylobacter):
      </td>
      <td>
        <input type="text" name="genus">
      </td>
    </tr>
    <tr>
      <td>
        Search by Species (e.g. Coli):
      </td>
      <td>
        <input type="text" name="species">
      </td>
    </tr>
    <tr>
      <td>
        Search by Subspecies (e.g. Fetus):
      </td>
      <td>
        <input type="text" name="subspecies">
      </td>
    </tr>
    <tr>
      <td>
        Search by Other Subtype:
      </td>
      <td>
        <input type="text" name="other">
      </td>
    </tr>
    <tr>
      <td>
        Search Protein:
      </td>
      <td>
        <input type="text" name="protein">
      </td>
    </tr>
    <tr>
      <td>
        Search Inside Notes:
      </td>
      <td>
        <input type="text" name="notes">
      </td>
    </tr>

```



```
<tr>
  <td colspan=2 align=center><br>
    <input type=reset>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit">
  </td>
</tr>
</table>
</form>
```

```
<%dbBean.dbClose();
out.write(pageSource.getPageFooter());
%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>

<%
//this gets the settings

String topnum, rangeA, rangeB;
if (request.getParameter("rangeA") != null && request.getParameter("rangeA")!="") {
rangeA = request.getParameter("rangeA");} else { rangeA = "4000";}
if (request.getParameter("rangeB") != null && request.getParameter("rangeB")!="") {
rangeB = request.getParameter("rangeB");} else { rangeB = "14000";}
if (request.getParameter("topnum") != null && request.getParameter("topnum")!="") {
topnum = request.getParameter("topnum");} else { topnum = "5";}

%>

<form action="multiplegraphdisplay.jsp?" name="myform" >
<table align=center cellpadding=0 cellspacing=0>
<tr>
<td colspan=3>

<%
//Display the graphs

if (request.getParameterValues("checkid") != null){
String[] maldiids;
malidiids = request.getParameterValues("checkid");
int l;
if (malidiids[0].compareTo("on") == 0) {
l=1;
} else {
l=0;
}
for (int i=l; i < malidiids.length; i++) {
if ((i-l) % 3 == 0) {
out.write("</tr>");
out.write("<tr>");
}

out.write("<td><img name='checkid' value='"+ malidiids[i] + "'
src='maldigraphtiny.jsp?malidiid=" + malidiids[i] + "&rangeA=" + rangeA + "&rangeB=" +
rangeB + "&topnum=" + topnum + "'></td>");
%>

<%
}
} else {
out.write("<center>Please Use the Checkboxes to select which graphs you wish to
display.</center>");
}
%>
</tr>
</table>
</form>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>

<%
//this gets the settings

String topnum, rangeA, rangeB;
if (request.getParameter("rangeA") != null && request.getParameter("rangeA")!="") {
rangeA = request.getParameter("rangeA");} else { rangeA = "4000";}
if (request.getParameter("rangeB") != null && request.getParameter("rangeB")!="") {
rangeB = request.getParameter("rangeB");} else { rangeB = "14000";}
if (request.getParameter("topnum") != null && request.getParameter("topnum")!="") {
topnum = request.getParameter("topnum");} else { topnum = "5";}

%>

<form action="multiplesmsgraphdisplay.jsp?" name="myform" >
<table align=center cellpadding=0 cellspacing=0>
<tr>
<td colspan=3>

<%
//Display the graphs

if (request.getParameterValues("checkid") != null){
String[] maldiids;
malidiids = request.getParameterValues("checkid");
int l;
if (malidiids[0].compareTo("on") == 0) {
l=1;
} else {
l=0;
}
for (int i=l; i < malidiids.length; i++) {
if ((i-l) % 3 == 0) {
out.write("</tr>");
out.write("<tr>");
}

out.write("<td><img name='checkid' value='" + malidiids[i] + '"
src='msmsggraphtiny.jsp?msmsid=" + malidiids[i] + "&rangeA=" + rangeA + "&rangeB=" +
rangeB + "&topnum=" + topnum + "'></td>");
%>

<%
}
} else {
out.write("<center>Please Use the Checkboxes to select which graphs you wish to
display.</center>");
}
%>
</tr>
</table>
</form>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Genus",
"InsertGenusTitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <center><h1>Insert New Genus</h1></center>
    <form action="genusconfirmation.jsp">
        <table align=center>
            <tr>
                <td>
                    Genus Name:
                </td>
                <td>
                    <input type="text" name="name">
                </td>
            </tr>
            <tr>
                <td colspan="2" align="right">
                    <input type="submit" value="Submit New Genus">
                </td>
            </tr>
        </table>
    </form>
    <%
        out.write(pageSource.getPageFooter());
    %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Identifier",
"InsertIdentifierTitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <center><h1>Insert Subtype</h1></center>
    <form action="identifierconfirmation.jsp">
        <table align=center>
            <tr>
                <td>
                    Identifier:
                </td>
                <td>
                    <input type="text" name="name">
                </td>
            </tr>
            <tr>
                <td>
                    Notes:
                </td>
                <td>
                    <input type="text" name="notes">
                </td>
            </tr>
            <tr>
                <td colspan="2" align="right">
                    <input type="submit" value="Submit Identifier">
                </td>
            </tr>
        </table>
    </form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Species",
"InsertSpeciesTitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <center><h1>Insert New Species</h1></center>
    <form action="speciesconfirmation.jsp">
        <table align=center>
            <tr>
                <td>
                    Species Name:
                </td>
                <td>
                    <input type="text" name="name">
                </td>
            </tr>
            <tr>
                <td colspan="2" align="right">
                    <input type="submit" value="Submit New Species">
                </td>
            </tr>
        </table>
    </form>

    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subspecies",
"InsertSubspeciesTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
<center><h1>Insert Subspecies</h1></center>
<form action="subspeciesconfirmation.jsp">
    <table align=center>
        <tr>
            <td>
                Subspecies Name:
            </td>
            <td>
                <input type="text" name="name">
            </td>
        </tr>
        <tr>
            <td colspan="2" align="right">
                <input type="submit" value="Submit Subspecies">
            </td>
        </tr>
    </table>
</form>

<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subtype",
"InsertSubtypeTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
<center><h1>Insert Subtype</h1></center>
<form action="subtypeconfirmation.jsp">
    <table align=center>
        <tr>
            <td>
                Subtype Name:
            </td>
            <td>
                <input type="text" name="name">
            </td>
        </tr>
        <tr>
            <td colspan="2" align="right">
                <input type="submit" value="Submit Subtype">
            </td>
        </tr>
    </table>
</form>

<%out.write(pageSource.getPageFooter());%>

```



```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@ page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gScatter" class="psm.graph.scatterPlot"/>
<%

String[] maldiids;
malDiids = request.getParameterValues("checkid");
if (malDiids != null ) {
    int x,y;
    x = Integer.parseInt(request.getParameter("x"));
    y = Integer.parseInt(request.getParameter("y"));
    //Initial Setup
    BufferedImage bImg = new BufferedImage(x,y,BufferedImage.TYPE_INT_RGB);
    response.setContentType("image/jpeg");
    Graphics2D biContext = gScatter.maldiScatter(dbBean, bImg, maldiids, x, y,
    Integer.parseInt(request.getParameter("rangeA")),
    Integer.parseInt(request.getParameter("rangeB")), 12, 120,
    Integer.parseInt(request.getParameter("topnum")),
    Integer.parseInt(request.getParameter("ppm")),
    Integer.parseInt(request.getParameter("intensity")));

    //wrap up
    OutputStream output = response.getOutputStream();
    JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
    JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
    param.setQuality((float) 1, false);
    encoder.setJPEGEncodeParam(param);
    encoder.encode(bImg);
    output.close();
} else {
    out.write("Please select at least one strain to plot.");
}
%>

```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gScatter" class="psm.graph.scatterPlot"/>
<%

String[] maldiids;
malidiids = request.getParameterValues("checkid");
if (malidiids != null ) {
int x,y;
x = Integer.parseInt(request.getParameter("x"));
y = Integer.parseInt(request.getParameter("y"));
//Initial Setup
BufferedImage bImg = new BufferedImage(x,y,BufferedImage.TYPE_INT_RGB);
response.setContentType("image/jpeg");
Graphics2D biContext = gScatter.MSMSScatter(dbBean, bImg, maldiids, x, y,
Integer.parseInt(request.getParameter("rangeA")),
Integer.parseInt(request.getParameter("rangeB")), 12, 120,
Integer.parseInt(request.getParameter("topnum")),
Double.parseDouble(request.getParameter("masserror")),
Integer.parseInt(request.getParameter("intensity")));

//wrap up
OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
} else {
    out.write("Please select at least one strain to plot.");
}
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Score", "ScoreTitle",
application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
    %>
        <table align="center" cellpadding = 2 cellspacing = 4>
        <tr>
            <td align=center colspan =5>
                <h1>Scoring</h1>
                The comparison program uses the following formula to compare
two sets of MALDI-TOF-DATA:<br>
                
$$\text{SUM}((A+B) - (2*|A-B|)) / \text{SUM}(A+B)$$

            </td>
        </tr>
        <tr>
            <td>
                <small><span style="font-weight: bold;">Average Mass</span></small>
            </td>
            <td>
                <small><span style="font-weight: bold;">Intensity A
(<%=request.getParameter("id1")%>)</span></small>
            </td>
            <td>
                <small><span style="font-weight: bold;">Intensity B
(<%=request.getParameter("id2")%>)</span></small>
            </td>
            <td>
                <small><span style="font-weight: bold;">Intensity [A +
B]</span></small>
            </td>
            <td>
                <small><span style="font-weight: bold;">Intensity [(A + B) - (2 * | A
- B |)]</span></small>
            </td>
        </tr>
        <%
            dbBean.dbConnect();
            //Dump maldi compare data to temp table for faster searches and to
normalize the data
            int maldiTempId =
dbBean.dumpMALDIDataToTemp(Integer.parseInt(request.getParameter("id1")),
Integer.parseInt(request.getParameter("intensity")),
Integer.parseInt(request.getParameter("lMass")),Integer.parseInt(request.getParamete
r("hMass")), "malidtemp");

            String selStr =
dbBean.buildCompareStringwithoutTotalScore(Integer.parseInt(request.getParameter("in
tensity")),Integer.parseInt(request.getParameter("lMass")),Integer.parseInt(request.
getParameter("hMass")),Integer.parseInt(request.getParameter("ppm")));
            //Set up the selStr inserting the ID's Appropriately
            int cmprToId = 0;
            if (request.getParameter("lMass").compareTo("4000") == 0 &
request.getParameter("hMass").compareTo("14000") == 0) {
                selStr = selStr.replace("$cmprTo$", request.getParameter("id2") );
            } else {
                //if not then we have to put back the temporary data then delete it
again
                //this dumps it to the temp table and also normalizes it

```

```

        cmprToId =
dbBean.dumpMALDIDataToTemp(Integer.parseInt(request.getParameter("id2")),
Integer.parseInt(request.getParameter("intensity")),
Integer.parseInt(request.getParameter("lMass")), Integer.parseInt(request.getParamete
r("hMass")), "malditemp2");
        selStr = selStr.replace("$cmprTo$", Integer.toString(cmprToId));
    }
    selStr = selStr.replace("$cmpr$", Integer.toString(maldiTempId));
    ResultSet myRS = dbBean.getRS(selStr);
    while (myRS.next()) {
        out.write("<tr><td align='center'><small>" + myRS.getString(1) +
"</small></td><td align='center'><small>" + myRS.getString(2) + "</small></td><td
align='center'><small>" + myRS.getString(3) + "</small></td><td
align='center'><small>" + myRS.getString(4) + "</small></td><td
align='center'><small>" + myRS.getString(5) + "</small></td></tr>");
    }
    myRS.close();

    //now delete the maldiTemp data to free up the memory
    dbBean.deleteMaldiTempData(maldiTempId, "malditemp");

    dbBean.dbClose();
    %>
</table>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Score", "ScoreTitle",
application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>

    <table align="center" cellpadding = 2 cellspacing = 4>
    <tr>
        <td align=center colspan =5>
            <h1>Scoring</h1>
            The comparison program uses the following formula to compare
two sets of MALDI-TOF-DATA:<br>
                
$$\text{SUM}((A+B) - (2*|A-B|)) / \text{SUM}(A+B)$$

            </td>
        </tr>
    <tr>
        <td>
            <small><span style="font-weight: bold;">Average Mass</span></small>
        </td>
        <td>
            <small><span style="font-weight: bold;">Intensity A
(<%=request.getParameter("id1")%>)</span></small>
        </td>
        <td>
            <small><span style="font-weight: bold;">Intensity B
(<%=request.getParameter("id2")%>)</span></small>
        </td>
        <td>
            <small><span style="font-weight: bold;">Intensity [A +
B]</span></small>
        </td>
        <td>
            <small><span style="font-weight: bold;">Intensity [(A + B) - (2 * |
A - B |)]</span></small>
        </td>
    </tr>
    <%
        dbBean.dbConnect();
        int cmprToId = 0;
        String selStr =
dbBean.buildCompareStringForTempwithoutTotalScore(Integer.parseInt(request.getParame
ter("intensity")),Integer.parseInt(request.getParameter("lMass")),Integer.parseInt(r
equest.getParameter("hMass")),Integer.parseInt(request.getParameter("ppm")));
        //If the mass range falls within the one given then we only need to
replace teh parameters
        if (request.getParameter("lMass").compareTo("4000") == 0 &
request.getParameter("hMass").compareTo("14000") == 0) {
            selStr = selStr.replace("$cmprTo$", request.getParameter("id2") );
        } else {
            //if not then we have to put back the temporary data then delete it
again
            //this dumps it to the temp table and also normalizes it
            cmprToId =
dbBean.dumpMALDIdataToTemp(Integer.parseInt(request.getParameter("id2")),
Integer.parseInt(request.getParameter("intensity")),
Integer.parseInt(request.getParameter("lMass")),Integer.parseInt(request.getParame
ter("hMass")), "malditemp2");
            selStr = selStr.replace("$cmprTo$", Integer.toString(cmprToId));

```

```

    }
    selStr = selStr.replace("$cmpr$", request.getParameter("id1"));
    ResultSet myRS = dbBean.getRS(selStr);
    while (myRS.next()) {
        out.write("<tr><td align='center'><small>" + myRS.getString(1) +
"</small></td><td align='center'><small>" + myRS.getString(2) + "</small></td><td
align='center'><small>" + myRS.getString(3) + "</small></td><td
align='center'><small>" + myRS.getString(4) + "</small></td><td
align='center'><small>" + myRS.getString(5) + "</small></td></tr>");
    }
    myRS.close();

    if (request.getParameter("lMass").compareTo("4000") == 0 &
request.getParameter("hMass").compareTo("14000") == 0) {
        dbBean.deleteMaldiTempData(cmprToId, "malditemp2");
    }

    dbBean.dbClose();
%>
</table>

<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Search Loci", "SearchLoci",
application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <form action="genesearchresults.jsp">
    <table align=center>
        <tr>
            <td colspan=2 style="text-align:center;">
                Please enter your search parameters
            <td>
        <tr>
        <tr>
            <td colspan=2>
                &nbsp;
            </td>
        </tr>
        <tr>
            <td>
                Locus:
            </td>
            <td>
                <input type="text" name="locustr">
            </td>
        </tr>
        <tr>
            <td>
                Strain:
            </td>
            <td>
                <input type="text" name="strainstr">
            </td>
        </tr>
        <tr>
            <td>
                Genus:
            </td>
            <td>
                <input type="text" name="genustr">
            </td>
        </tr>
        <tr>
            <td>
                Species:
            </td>
            <td>
                <input type="text" name="speciesstr">
            </td>
        </tr>
        <tr>
            <td>
                Location:
            </td>
            <td>
                <input type="text" name="locationstr">
            </td>
        </tr>
        <tr>
            <td>

```

```

        Location 2:
    </td>
    <td>
        <input type="text" name="location2str">
    </td>
</tr>
<tr>
    <td>
        Gene:
    </td>
    <td>
        <input type="text" name="genestr">
    </td>
</tr>
<tr>
    <td>
        Molecular weight:
    </td>
    <td>
        <input type="text" name="mw">
    </td>
</tr>
<tr>
    <td>
        Error Range (ppm):
    </td>
    <td>
        <input type="text" name="ppm" value=0>
    </td>
</tr>
<tr>
    <td>
        Product:
    </td>
    <td>
        <input type="text" name="productstr">
    </td>
</tr>
<tr>
    <td>
        Notes:
    </td>
    <td>
        <input type="text" name="notesstr">
    </td>
</tr>
<tr><td colspan=2>&nbsp;</td></td>
<tr>
    <td colspan=2>
        Select the columns you wish displayed.
    <td>
</tr>
<tr><td colspan=2>&nbsp;</td></td>
<tr>
    <td colspan=2>
        <input type="checkbox" name="locus" checked>Locus&nbsp;<input
type="checkbox" name="strain" checked>Strain&nbsp;<input type="checkbox"
name="genusspecies" checked>Genus-Species&nbsp;<input type="checkbox"
name="location1" checked>Location 1&nbsp;<input type="checkbox" name="location2"
checked>Location 2&nbsp;
    <td>
</tr>
<tr>
    <td colspan=2>
        <input type="checkbox" name="start" checked>start&nbsp;<input

```



```

type="checkbox" name="end" checked>end&nbsp;<input type="checkbox" name="GBK"
checked>GBK&nbsp;<input type="checkbox" name="fr" checked>FR&nbsp;<input
type="checkbox" name="NT" checked>NT&nbsp;<input type="checkbox" name="AA"
checked>AA&nbsp;<input type="checkbox" name="MW" checked>MW&nbsp;<input
type="checkbox" name="iep" checked>IEP&nbsp;
    </td>
  </tr>
  <tr>
    <td colspan=2>
      <input type="checkbox" name="gene" checked>Gene&nbsp;<input
type="checkbox" name="code" checked>code&nbsp;<input type="checkbox" name="COG"
checked>COG&nbsp;<input type="checkbox" name="ec1" checked>EC#1&nbsp;<input
type="checkbox" name="ec2" checked>EC#2&nbsp;<input type="checkbox" name="product"
checked>Product&nbsp;<input type="checkbox" name="notes" checked>notes&nbsp;
    </td>
  </tr>
  <tr>
    <td colspan=2 style="text-align: right;">
      <input type="submit">
    </td>
  <tr>
</table>
</form>
<%
out.write(pageSource.getPageFooter());
%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:setProperty name="dbBean" property="*/"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%int validlogin = dbBean.dbConnect(); %>

<%
if (validlogin ==0) {
    pageSource.splitTemplatePage(dbBean.getUsername(), "Search <i>In Silico</i>
Data", "StrainSearchTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
}
    if (validlogin != 0) {%>
        <center>The username/password combination you have entered does not
exist or is no longer valid. Please press back on your browser and try
again.</center>
        <%} else { dbBean.dbClose(); %>

        <table border="0" cellpadding=0 cellspacing=0>
            <tr>

                <form action="searchresultsinsilico.jsp">
                <br>
                <table align=center>
                    <tr>
                        <td colspan="2">
                            <center><b>Enter the parameters for your
search:</b></center><br>
                        </td>
                    </tr>
                    <tr>
                        <td colspan=2 align = center>
                            <input type="radio" name="andor" value="and" checked>AND
                            <input type="radio" name="andor" value="Or">OR
                        </td>
                    </tr>
                    <tr>
                        <td>
                            Search by Identification (e.g. NCBI1203 or 1203):
                        </td>
                        <td>
                            <input type=text name="identifier">
                        </td>
                    </tr>
                    <tr>
                        <td>
                            Search by Genus (e.g. Campylobacter):
                        </td>
                        <td>
                            <input type=text name="genus">
                        </td>
                    </tr>
                    <tr>
                        <td>
                            Search by Species (e.g. Coli):
                        </td>
                        <td>
                            <input type=text name="species">
                        </td>
                    </tr>
                    <tr>
                        <td>
                            Search by Subspecies (e.g. Fetus):

```

[illegible]

[illegible]

```

        myRS = dbBean.runANDSearch(request.getParameter("sampleid"),
request.getParameter("identifier"), request.getParameter("genus"),
request.getParameter("species"), request.getParameter("subspecies"),
request.getParameter("other"), request.getParameter("notes"),
request.getParameter("history"), request.getParameter("gtent"),
request.getParameter("sptent"), request.getParameter("subtent"),
request.getParameter("othertent"), maldionly, request.getParameter("mass"),
Integer.parseInt(request.getParameter("masserror")),
request.getParameter("minintensity"), request.getParameter("lowRM"),
request.getParameter("highRM"));
    }

    if (myRS.next() ) {
        myRS.first();
        %>

        <table border="0" cellpadding = "0" cellspacing = "0" width= 100%>
        <span style="color: rgb(255,255, 255);">
            <th bgcolor = #008700><span style="color: rgb(231,229, 38);">Sample
ID</th></span>
            <th bgcolor = #008700><span style="color: rgb(231,229, 38);">Sample
Number</th></span>
            <th bgcolor = #008700><span style="color: rgb(231,229,
38);">Date</th></span>
            <th bgcolor = #008700><span style="color: rgb(231,229, 38);">Sample
Genus-Species-Subspecies</th></span>
            <th bgcolor = #008700><span style="color: rgb(231,229,
38);">Notes</th></span>
            <th bgcolor = #008700><span style="color: rgb(231,229,
38);">History</th></span>
        </span>

        <% int i = 0;
            String mColor;
        %>
        <% do {
            i++;
            if ( i % 2 == 0) {
                mColor= "#ffd0ae";
            }else {
                mColor = "#ecac7d";
            }
        %>

            <tr>

                <td style="vertical-align: top; background-color:
<%=mColor%>" align="center"><a
href=".\\strainDetail.jsp?sampleid=<%=myRS.getString(1)%>" style="color:
rgb(0,0,0);font-weight: bold;"><%=myRS.getString("sample_id")%></a></td>
                <td style="vertical-align: top; background-color:
<%=mColor%>"><a href="strainDetail.jsp?sampleid=<%=myRS.getString("sample_id")%>"
style="color: rgb(0,0,0);font-weight: bold;"><span style="color:
rgb(0,0,0);font-weight: bold;"><%=myRS.getString("ident")%></span></a></td></span>
                <td style="text-align: center;vertical-align: top;
background-color: <%=mColor%>" width="100"><span style="color:
rgb(0,0,0);font-weight: bold;"><%=myRS.getString("date")%>&nbsp;&nbsp;&nbsp;</td></span>
                <td style="vertical-align: top; background-color:
<%=mColor%>"><a href="strainDetail.jsp?sampleid=<%=myRS.getString("sample_id")%>"
style="color: rgb(0,0,0);font-weight: bold; text-align: center;"><i><%=
myRS.getString("genus") != null) { if (myRS.getInt("tentative_genus") == 1 &&

```

```

        out.write("<span style='color:
rgb(225,43,43);font-weight: bold;'>");

        out.write(myRS.getString("genus"));

        out.write("</span>");

    } else if(myRS.getString("genus") !=null) {

        out.write(myRS.getString("genus"));

    }

    if (myRS.getInt("tentative_species") == 1 &&
myRS.getString("species") != null) {

        out.write("<span style='color:
rgb(225,43,43);font-weight: bold;'>");

        out.write(" ");

        out.write(myRS.getString("species"));

        out.write("</span>");

    } else if(myRS.getString("species") !=null) {

        out.write(" ");

        out.write(myRS.getString("species"));

    }

    if (myRS.getInt("tentative_subspecies") ==1 &&
myRS.getString("subspecies") != null) {

        out.write("<span style='color:
rgb(225,43,43);font-weight: bold;'>");

        out.write(" ");

        out.write(myRS.getString("subspecies"));

        out.write("</span>");

```

```

        } else if(myRS.getString("subspecies") !=null) {

            out.write(" ");

            out.write(myRS.getString("subspecies"));

        }

        if (myRS.getInt("tentative_subtype") ==1 &&
myRS.getString("subtype") != null) {

            out.write("<span style='color:
rgb(225,43,43);font-weight: bold;'>");

            out.write(" ");

            out.write(myRS.getString("subtype"));

            out.write("</span>");

        } else if(myRS.getString("subtype") !=null) {

            out.write(" ");

            out.write(myRS.getString("subtype"));

        }

        //This is to add seperation between the name and
notes

        out.write(" ");

        %></i></a></td>
        <td style="vertical-align: top; background-color:
<%=mColor%>"><a href="updatestrain.jsp?sampleid=<%=myRS.getString("sample_id")%>"
style="color: rgb(0,0,0);font-weight:bold;"><%=myRS.getString("notes")%></a></td>
        <td style="vertical-align: top; background-color:
<%=mColor%>"><span style="color: rgb(0,0,0);font-weight: bold;"><%=
        if (myRS.getString("history") != null ) {
            out.print(myRS.getString("history"));
        } else {
            out.print("No History");
        }
        %></td></span>

    </tr>
    <%
    } while (myRS.next());
    %>

```

```
<%
//dbBean.closemyRS();
//myRS.close();
dbBean.dbClose();%>

<%}else{%>

    <span style="color: white" ><center>No strain was found with the
entered search criteria. Press the back button on your browser to start a new
search.</center> </span>

    <%}} else {
    out.write("<span style='color: white' ><center>Your connection has
expired. Please Log In again.</center></span>");
    }%>

</table>
</body>
</html>
```



```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "<i>In Silico</i> Search",
"StrainDetailTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
    %>
    <%
        dbBean.dbConnect();
    %>

    <table border =0 cellspacing=8 cellpadding=0 style="text-align: left;width:
90%">
        <tr>
            <td colspan="2">
                <table cellspacing="0" cellpadding="0" style="width: 80%" align=center>
                    <%
                        ResultSet maldiRS;
                        String mstr = null;
                        if (request.getParameter("andor").compareTo("and") == 0) {
                            mstr =
dbBean.runANDInSilicoSearchStr(request.getParameter("identifier"),
request.getParameter("genus"),request.getParameter("species"),request.getParameter("
subspecies"),request.getParameter("other"), request.getParameter("protein"),
request.getParameter("notes"), request.getParameter("mass"),
request.getParameter("masserror"));
                        }
                        else {
                            mstr =
dbBean.runORInSilicoSearchStr(request.getParameter("identifier"),
request.getParameter("genus"),request.getParameter("species"),request.getParameter("
subspecies"),request.getParameter("subtype"), request.getParameter("protein"),
request.getParameter("notes"), request.getParameter("mass"),
request.getParameter("masserror"));
                        }
                        maldiRS = dbBean.getRS(mstr);
                        if (malDiRS != null) {
                            int j = 0;
                            while (malDiRS.next()) {
                                j++;
                                %>

                                <td style="vertical-align: top; text-align: left;">
                                    <table border="0" cellpadding="0" cellspacing = "0">
                                        <tr>
                                            <td>
                                                <b><i>In Silico</i> ID:</b>
                                            </td>
                                            <td>
                                                <a
href="./insilicodatawithgraph.jsp?insilicoid=<%=malDiRS.getInt(1)%>">
<%=malDiRS.getInt(1)%></a>
                                            </td>
                                        </tr>
                                        <tr>
                                            <td>
                                                <b>Identification:&nbsp;  </b>
                                            </td>

```

```

        <td>
            <%=maldiRS.getString(2)%>
        </td>
    </tr>
    <tr>
        <td>
            <b>Date:</b>
        </td>
        <td>
            <%=maldiRS.getDate(3)%>
        </td>
    </tr>
    <tr>
        <td>
            <b>Average Molecular weight:&nbsp;&nbsp;&nbsp;</b>
        </td>
        <td>
            <%=maldiRS.getString(4)%>
        </td>
    </tr>
    <tr>
        <td>
            <b>Protein:&nbsp;&nbsp;&nbsp;</b>
        </td>
        <td>
            <%=maldiRS.getString(5)%>
        </td>
    </tr>
    <tr>
        <td>
            <b>Strain:</b>
        </td>
        <td>
            <%=maldiRS.getString(7)%>&nbsp;&nbsp;& <%=maldiRS.getString(8)%>&nbsp;&nbsp;& <%=maldiRS.getString(9)
            %>&nbsp;&nbsp;& <%=maldiRS.getString(10)%>&nbsp;&nbsp;&
        </td>
    </tr>
    <tr>
        <td valign="top">
            <b>AA Seq:</b>
        </td>
        <td>
            <%=String aa = maldiRS.getString("aa_seq");
            out.write(aa.charAt(0));
            for(int v = 1; v < aa.length(); v++) {
                out.write(aa.charAt(v));
                if((v % 20) == 0) {
                    out.write("<br>");
                }
            }
            %>
        </td>
    </tr>
    <tr>
        <td>
            <b>Notes:</b>
        </td>
        <td>
            <a
href="updateinsiliconotes.jsp?insilicoid=<%=maldiRS.getString(1)%>">Update</a>
        </td>
    </tr>
    <tr>
        <td colspan=2>

```

```

                                <%=malDiRS.getString(6)%>
                            </td>
                        </tr>

                    </table>
                </td>
            <td>&nbsp;</td>

            <% if (j % 3 == 0) { %>
                <tr><td>&nbsp;</td></tr>
                <tr>
            <%}%>

            <%if (j % 6 == 0) { %>
            </tr>
            <%}%>

        <%
            }} else{
        %>
        No In Silico Data Found
        <%}%>
    </table>
</td>
</tr>
</table>
<%dbBean.dbClose();
out.write(pageSource.getPageFooter());%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Species",
"InsertSpeciesTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
<%
    dbBean.dbConnect();
    int rtn;
    rtn = dbBean.insertNewSpecies(request.getParameter("name"));
    if (rtn > 0) {
        out.write("Insert successfull!");
    } else if (rtn == -1) {
        out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
    }
    dbBean.dbClose();
%>
<br><center><a href="insertstrain.jsp">Insert New Strain</a></center>

<%out.write(pageSource.getPageFooter());%>

```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.connect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Strain Detail",
        "StrainDetailTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.close();
%>
<%
    dbBean.connect();

    ResultSet myRS;
    myRS = dbBean.searchBySampleId(request.getParameter("sampleid"));
    myRS.next();

    if (myRS != null) {
        String updateURL = "updatestrain.jsp?sampleid=" +
            request.getParameter("sampleid");

```

Strain Detail	
Update Strain	
Identification:	
Date:	
Strain:	
History	

```

        <td>
            <%if (myRS.getString(6) != null ) {
                out.print(myRS.getString(6));
            } else {
                out.print("No History");
            }
            %>
        </td>
    </tr>

    <tr>
        <td>
            <b>Tentative Genus:</b>
        </td>
        <td>
            <%if (myRS.getBoolean(7)) {
                out.write("<span style='color: RED;font-weight: bold;'>");
                out.write("True");
                out.write("</span>");
            }else{
                out.write("False");
            }%>
        </td>
    </tr>

    <tr>
        <td>
            <b>Tentative Species:</b>
        </td>
        <td>
            <%if (myRS.getBoolean(8)) {
                out.write("<span style='color: RED;font-weight: bold;'>");
                out.write("True");
                out.write("</span>");
            }else{
                out.write("False");
            }%>
        </td>
    </tr>

    <tr>
        <td>
            <b>Tentative Subspecies:</b>
        </td>
        <td>
            <%if (myRS.getBoolean(9)) {
                out.write("<span style='color: RED;font-weight: bold;'>");
                out.write("True");
                out.write("</span>");
            }else{
                out.write("False");
            }%>
        </td>
    </tr>
    <tr>

```

```

        <td>
        <b>Tentative Other Type:</b>
        </td>
        <td>
        <%if (myRS.getBoolean(10)) {
            out.write("<span style='color: RED;font-weight: bold;'>");
            out.write("True");
            out.write("</span>");
        }else{
            out.write("False");
        }%>
        </td>
    </tr>

    <tr>
        <td>
        <b>Notes: </b>
        </td>
    </tr>
    <tr>
        <td colspan="2">
        <%=myRS.getString(5)%>
        </td>
        <%=myRS.close();%>
    </tr>
    <tr align='center'>
        <%
            myRS = dbBean.getMaldi(request.getParameter("sampleid"));
            if (myRS.next()) {
                %><td><a
href="strainMALDIDATA.jsp?sampleid=<%=request.getParameter("sampleid")%>">MALDI
Data</a><br></td>
                <%
            }
            myRS.close();
            %>

            <%
            myRS = dbBean.getMSMS(request.getParameter("sampleid"));
            if (myRS.next()) {
                %>
                <td><a
href="strainMSMSData.jsp?sampleid=<%=request.getParameter("sampleid")%>">MS/MS
Data</a><br></td>
                <%
            }
            myRS.close();
            %>

        </tr>
    </table>
    <%dbBean.dbClose();%>
    <%}else{%>

        <center>No strain was found with Sample ID
        <%=request.getParameter("sampleid")%></center>
        <%}%>

    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "MALDI Data",
"StrainDetailTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
<%
    dbBean.dbConnect();
%>

<table border =0 cellspacing=8 cellpadding=0 style="text-align: left;width:
90%">
    <tr>
        <td>
            <b> <big>MALDI-TOF Data </big> </b><br>
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <table cellspacing="0" cellpadding="0" style="width: 90%" align=center>
                <tr><td>&nbsp;</td></tr>
            </table>
            <%
                ResultSet maldiRS;
                maldiRS = dbBean.getMaldi(request.getParameter("sampleid"));

                if (malDiRS != null) {
                    int j = 0;
                    while (malDiRS.next()) {
                        j++;
                    }
                }

                <td>
                    <table border="0" cellpadding="0" cellspacing = "0">
                        <tr>
                            <td>
                                <b>Maldi ID:</b>
                            </td>
                            <td>
                                <a
href="./maldidatawithgraph.jsp?malDid=<%=malDiRS.getInt(1)%>">
<%=malDiRS.getInt(1)%></a>
                            </td>
                        </tr>
                    </table>
                </td>
            </tr>
            <tr>
                <td>
                    <b>Date:</b>
                </td>
                <td>
                    <%=malDiRS.getDate(5)%>
                </td>
            </tr>
            <tr>
                <td>
                    <b>Target Spot Number:&nbsp;&nbsp;&nbsp;</b>
                </td>
                <td>
                    <%=malDiRS.getString(3)%>
                </td>
            </tr>
        </td>
    </tr>

```



```

        </tr>
        <tr>
            <td>
                <b>Notes:</b>
            </td>
            <td>
                <a
href="updateMALDInotes.jsp?MALDIid=<%=MALDIrs.getString(1)%>">Update</a>
            </td>
        </tr>
        <tr>
            <td colspan=2>
                <%=MALDIrs.getString(4)%>
            </td>
        </tr>
    </table>
</td>
<td>&nbsp;</td>

<% if (j % 3 == 0) { %>
    <tr><td>&nbsp;</td></tr>
    <tr>
        <%}%>

        <%if (j % 6 == 0) { %>
        </tr>
        <%}%>
    <%
        }} else{
    %>
    This sample does not currently have MALDI-TOF data
    <%}%>
</table>
</td>
</tr>
</table>
<%dbBean.dbClose();
out.write(pageSource.getPageFooter());%>

```

[illegible]

```

        <td style="vertical-align: top; text-align: left;">
            <b>Precursor Ion Mass:&nbsp;&nbsp;&nbsp;</b>
        </td>
        <td>
            <%=maldiRS.getString(4)%>
        </td>
    </tr>
    <tr>
        <td style="vertical-align: top; text-align: left;">
            <b>Protein:&nbsp;&nbsp;&nbsp;</b>
        </td>
        <td style="vertical-align: top; text-align: left;">
            <%=maldiRS.getString(5)%>
        </td>
    </tr>
    <tr>
        <td>
            <b>Notes:</b>
        </td>
        <td>
            <a
href="./updatesmsnotes.jsp?msmsid=<%=maldiRS.getString(1)%>">Update</a>
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <%=maldiRS.getString(6)%>
        </td>
    </tr>
</table>
</td>

<% if (j % 3 == 0) { %>
    <tr><td>&nbsp;</td></tr>
    <tr>
        <%}%>

    <%if (j % 6 == 0) { %>
    </tr>
    <%}%>

    <%
        }} else{
    %>
    This sample does not currently have MALDI-TOF data
    <%}%>
</table>
</td>
</tr>
</table>
<%dbBean.dbClose();
out.write(pageSource.getPageFooter());%>

```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subspecies",
"InsertSubspeciesTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
<%
    dbBean.dbConnect();
    int rtn;
    rtn = dbBean.insertNewSubspecies(request.getParameter("name"));
    if (rtn > 0) {
        out.write("Insert successfull!");
    } else if (rtn == -1) {
        out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
    }
    dbBean.dbClose();
%>
<br><center><a href="insertstrain.jsp">Insert New Strain</a></center>

<%out.write(pageSource.getPageFooter());%>
```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));
int b;
String[] maldiids = request.getParameterValues("checkid");

if (maldiids[0].compareTo("on") == 0 ) {
    b = 1;
} else {
    b = 0;
}

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,620,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,620);

//Turn on antialiasing
biContext.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,540);
biContext.drawLine(60,540,810,540);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,545);
biContext.drawLine(53, 540, 60, 540);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 545 - j*5);
        biContext.drawLine(53, 540-j*5, 60, 540-j*5);
    } else {
        biContext.drawLine(58 , 540 -j *5, 60, 540-j*5 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 570);

```

```

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

    if (num.length()%2 !=0) {
        centerint+= 3;
    }

    biContext.drawString(num, (60 + 150*(l) - centerint) , 570);

    biContext.drawLine(60+150*l, 540, 60+150*l, 555);
}

for (l=1; l <=10; l++) {
    biContext.drawLine(60+(75*l), 540, 60+(75*l), 550);
}

for (l=1; l <= 20; l++) {
    if (l % 2 != 0 ) {
        double shift = l * 37.5;
        int lshift = (int) shift;
        biContext.drawLine(60+lshift, 540, 60+lshift, 545);
    }
}

dbBean.dbConnect();

//draw the lines

//Draw the first id
biContext = gBean.gDraw(biContext, dbBean.getMaldiGraphSubtractionData(maldiids[b],
malidiids[b+1], intA, intB, Integer.parseInt(request.getParameter("ppm"))), intA,
intB, topnum/2 + (topnum % 2), 540, 5, 750, Color.RED);

//draw the secod id
biContext = gBean.gDraw(biContext,
dbBean.getMaldiGraphSubtractionData(maldiids[b+1], maldiids[b], intA, intB,
Integer.parseInt(request.getParameter("ppm"))), intA, intB, topnum /2, 540, 5, 750,
Color.CYAN);

biContext.setColor(Color.RED);
biContext = gBean.gDrawTitle(biContext, dbBean.getMaldiAndSampleInfo(maldiids[b]),
870, 10);

biContext.setColor(Color.CYAN);
biContext = gBean.gDrawTitle(biContext, dbBean.getMaldiAndSampleInfo(maldiids[b+1]),
870, 22);

biContext.setColor(Color.BLACK);

biContext.drawString("Mass to Charge (M/Z)", 365, 610);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 325 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,325);

```

```
OutputStream output = response.getOutputStream();
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>
```

```

<%@ page language="java" %>

<%@ page import="java.io.OutputStream" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="java.awt.*" %>
<%@ page import="com.sun.image.codec.jpeg.*" %>
<%@page import="java.sql.*" %>

<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="gBean" class="psm.graph.maldiGraph"/>
<%
int intA = Integer.parseInt(request.getParameter("rangeA"));
int intB = Integer.parseInt(request.getParameter("rangeB"));
int topnum = Integer.parseInt(request.getParameter("topnum"));
int b;
String[] maldiids = request.getParameterValues("checkid");

if (maldiids[0].compareTo("on") == 0 ) {
    b = 1;
} else {
    b = 0;
}

Font font = new Font("ARIAL", Font.BOLD , 12);

response.setContentType("image/jpeg");
BufferedImage bImg = new BufferedImage(850,620,BufferedImage.TYPE_INT_RGB);

Graphics2D biContext = bImg.createGraphics();
biContext.setBackground(Color.WHITE);
biContext.clearRect(0,0,850,620);

//Turn on antialiasing
biContext.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

biContext.setColor(Color.BLACK);
biContext.drawLine(60,40,60,540);
biContext.drawLine(60,540,810,540);
biContext.setFont(font);

//Draws the Y Numbers
int j =1;
biContext.drawString("0",35,545);
biContext.drawLine(53, 540, 60, 540);
for (j = 1; j<=100; j++) {
    if (j % 10 == 0) {
        String num;
        num = "";
        num = num + j;
        biContext.drawString(num, 30, 545 - j*5);
        biContext.drawLine(53, 540-j*5, 60, 540-j*5);
    } else {
        biContext.drawLine(58 , 540 -j *5, 60, 540-j*5 );
    }
}

int div = (intB - intA) / 5;
int l;

String num;
num = "";
num = num + intA;
biContext.drawString(num, 60 , 570);

```



```

//Draws the X numbers
for (l=1; l <=5; l++) {
    int i = 0;
    i = intA + div*l;
    num = "";
    num = num + i;
    int centerint = num.length() / 2 * 7;

    if (num.length()%2 !=0) {
        centerint+= 3;
    }

    biContext.drawString(num, (60 + 150*(l) - centerint) , 570);

    biContext.drawLine(60+150*l, 540, 60+150*l, 555);
}

for (l=1; l <=10; l++) {
    biContext.drawLine(60+(75*l), 540, 60+(75*l), 555);
}

for (l=1; l <= 20; l++) {
    if (l % 2 != 0 ) {
        double shift = l * 37.5;
        int Ishift = (int) shift;
        biContext.drawLine(60+Ishift, 540, 60+Ishift, 545);
    }
}

dbBean.dbConnect();

//draw the lines

//Draw the first id
biContext = gBean.gDraw(biContext, dbBean.getMSSMSGraphSubtractionData(maldiids[b],
malidiids[b+1], intA, intB, request.getParameter("masserror")), intA, intB, topnum/2
+ (topnum % 2), 540, 5, 750, Color.RED);

//draw the secod id
biContext = gBean.gDraw(biContext, dbBean.getMSSMSGraphSubtractionData(maldiids[b+1],
malidiids[b], intA, intB, request.getParameter("masserror")), intA, intB, topnum /2,
540, 5, 750, Color.CYAN);

biContext.setColor(Color.RED);
biContext = gBean.gDrawTitle(biContext, dbBean.getMSSMSAndSampleInfo(maldiids[b]),
870, 10);

biContext.setColor(Color.CYAN);
biContext = gBean.gDrawTitle(biContext, dbBean.getMSSMSAndSampleInfo(maldiids[b+1]),
870, 22);

biContext.setColor(Color.BLACK);

biContext.drawString("Mass to Charge (M/Z)", 365, 610);
FontMetrics fm = biContext.getFontMetrics();
biContext.rotate(-Math.PI / 2, 10, 325 - (fm.getHeight()/2));
biContext.drawString("% Intensity", 10,325);

OutputStream output = response.getOutputStream();

```

```
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bImg);
param.setQuality((float) 1, false);
encoder.setJPEGEncodeParam(param);
encoder.encode(bImg);
output.close();
dbBean.dbClose();
%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
<%
    dbBean.dbConnect();
    pageSource.splitTemplatePage(dbBean.getUsername(), "Insert Subtype",
"InsertSubtypeTitle", application.getRealPath("/"), dbBean);
    out.write(pageSource.getPageHeader());
    dbBean.dbClose();
%>
<%
    dbBean.dbConnect();
    int rtn;
    rtn = dbBean.insertNewSubtype(request.getParameter("name"));
    if (rtn > 0) {
        out.write("Insert successfull!");
    } else if (rtn == -1) {
        out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
    }
    dbBean.dbClose();
%>
<br><center><a href="insertstrain.jsp">Insert New Strain</a></center>

<%out.write(pageSource.getPageFooter());%>

```

```
<%@page import="java.math.*" %>
<%
    BigInteger l = BigInteger.valueOf(1);
    for (int i = 1; i <= 65; i++) {
        l = l.multiply(BigInteger.valueOf(i));
    }
    out.write(l.toString());
%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "MALDI Comparison",
"MalDIComparisonTitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form>
    <table align=center>
        <tr>
            <td>
                MALDI-TOF ID:
            </td>
            <td>
                <%=request.getParameter("maldiid")%>
            </td>
        </tr>
        <tr>
            <td>
                Threshold (%):&nbsp;
            </td>
            <td colspan="3">
                <input type="text" name="threshold" value="10">
            </td>
        </tr>
        <tr>
            <td>
                Mass Range:
            </td>
            <td>
                <input type="text" name="rangea" value="4000">
            </td>
            <td>
                To:
            </td>
            <td>
                <input type="text" name="rangeb" value="14000">
            </td>
        </tr>
        <tr>
            <td>
                Data Uncertainty (ppm):&nbsp;
            </td>
            <td colspan="3">
                <input type="text" name="uncertainty" value="50">
            </td>
        </tr>
        <tr>
            <td>
                Number of Matches to Return:&nbsp;
            </td>
            <td colspan="3">
                <input type="text" name="topmatches" value="10">
            </td>
        </tr>
        <tr>
            <td align=right colspan=4>
                <input type="submit" value="GO">
            </td>
        </tr>
    </table>

```

```
</form>  
<%out.write(pageSource.getPageFooter());%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Update <i>In Silico</i>
Data", "InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <table align = center>
        <tr>
            <td colspan=2>
                <%
                    dbBean.dbConnect();
                    dbBean.executeString("UPDATE in_silico SET notes='" +
request.getParameter("notes") + "' WHERE in_silico_id = " +
request.getParameter("insilicoid"));
                    out.write("Update Completed.");
                    dbBean.dbClose();
                %>
            </td>
        </tr>
    </table>

    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Update <i>In Silico</i>
Data", "InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());

        String[] insilico;
        insilico = dbBean.getInSilicoSampleInfo(request.getParameter("insilicoid"));
        dbBean.dbClose();
    %>

    <form action="updateinsilicoconfirm.jsp">
    <table align = center>
        <tr>
            <td>
                <i>In Silico</i> ID Number:
            </td>
            <td>
                <input type="text" readonly name="insilicoid"
value="<%=request.getParameter("insilicoid")%>">
            </td>
        </tr>
        <tr>
            <td>
                Notes
            </td>
            <td>
                <textarea name="notes"><%=insilico[2]%></textarea>
            </td>
        </tr>
        <tr>
            <td colspan="2" align=center>
                <input type="submit" value="Update">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```



```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Update MALDI Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <table align = center>
        <tr>
            <td colspan=2>
                <%
                    dbBean.dbConnect();
                    dbBean.executeString("UPDATE maldi_tof SET notes='" +
request.getParameter("notes") + "' WHERE maldi_tof_id = " +
request.getParameter("malidiid"));
                    out.write("Update Completed.");
                    dbBean.dbClose();
                %>
            </td>
        </tr>
    </table>

    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Update MALDI Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());

        String[] maldi;
        maldi = dbBean.getMaldiSampleInfo(request.getParameter("maldiid"));
        dbBean.dbClose();
    %>

    <form action="updatemaldiconfirm.jsp">
    <table align = center>

        <tr>
            <td>
                MALDI ID Number:
            </td>
            <td>
                <input type="text" readonly name="maldiid"
value="<%=request.getParameter("maldiid")%>">
            </td>
        </tr>
        <tr>
            <td>
                Notes
            </td>
            <td>
                <textarea name="notes"><%=maldi[2]%></textarea>
            </td>
        </tr>
        <tr>
            <td colspan="2" align=center>
                <input type="submit" value="Update">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Update MS/MS Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <table align = center>
        <tr>
            <td colspan=2>
                <%
                    dbBean.dbConnect();
                    dbBean.executeString("UPDATE ms_ms SET notes='" +
request.getParameter("notes") + "' WHERE ms_ms_id = " +
request.getParameter("msmsid"));
                    out.write("Update Completed.");
                    dbBean.dbClose();
                %>
            </td>
        </tr>
    </table>

    <%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Update MS/MS Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());

        String[] msms;
        msms = dbBean.getMSMSSampleInfo(request.getParameter("msmsid"));
        dbBean.dbClose();
    %>

    <form action="updatemsmsconfirm.jsp">
    <table align = center>

        <tr>
            <td>
                MS/MS ID Number:
            </td>
            <td>
                <input type="text" readonly name="msmsid"
value="<%=request.getParameter("msmsid")%>">
            </td>
        </tr>
        <tr>
            <td>
                Notes
            </td>
            <td>
                <textarea name="notes"><%=msms[2]%></textarea>
            </td>
        </tr>
        <tr>
            <td colspan="2" align=center>
                <input type="submit" value="Update">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Update Strain",
"UpdateStrainTitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <%
        dbBean.dbConnect();
    %>
    <%
        int sampleID, crntCycle;
        crntCycle=0;
        //This handles all the update code and the cycle code
        //Since we cycle using rm numbers, and sample_ids aren't necessarrily equivalent
or in the same order we need to
        //grab the sample ID from the RM number

        //First update the previous sample
        if (request.getParameter("submit") != null) {
            int gtent, sptent, subtent, othertent;

            if (request.getParameter("gtent") == null) {
                gtent = 0;
            } else {
                gtent = 1;
            }
            if (request.getParameter("sptent") == null) {
                sptent = 0;
            } else {
                sptent = 1;
            }
            if (request.getParameter("subtent") == null) {
                subtent = 0;
            } else {
                subtent = 1;
            }
            if (request.getParameter("othertent") == null) {
                othertent = 0;
            } else {
                othertent = 1;
            }
        }
        if
        (dbBean.updateStrain(Integer.parseInt(request.getParameter("sampleid")),Integer.pars
eInt(request.getParameter("ident")),Integer.parseInt(request.getParameter("number"))
,request.getParameter("date"),Integer.parseInt(request.getParameter("genus")),Integer
.parseInt(request.getParameter("species")),Integer.parseInt(request.getParameter("su
bspecies")),Integer.parseInt(request.getParameter("subtype")),request.getParameter("
notes"),request.getParameter("history"),gtent ,sptent,subtent,othertent) ==0) {
            out.write("Update Successfull!");
            //Now Handle the cycling if the update was successfull, otherwise dont
update
            if (request.getParameter("lowCycle") == "" ||
request.getParameter("lowCycle") == null ||
request.getParameter("lowCycle").compareTo("0") == 0 ||
request.getParameter("highCycle") == "" || request.getParameter("highCycle") == null
|| request.getParameter("lowCycle").compareTo(request.getParameter("highCycle")) ==
0) {
                sampleID = Integer.parseInt(request.getParameter("sampleid"));

```

```

    } else {
        crntCycle = Integer.parseInt(request.getParameter("lowCycle")) + 1;
        sampleID = dbBean.getSampleIDFromIdent(crntCycle,
Integer.parseInt(request.getParameter("ident")));
    }
    if (sampleID == 0) {
        //No sample existed or an error occurred
        crntCycle = crntCycle -1;
        sampleID = Integer.parseInt(request.getParameter("sampleid"));
    }
    } else{
        out.write("Update Failed, please press back and make sure that your data
is correct and that your login has not timed out.");
        sampleID = Integer.parseInt(request.getParameter("sampleid"));
    }
    } else {
        sampleID = Integer.parseInt(request.getParameter("sampleid"));
    }
%>
<%
//this handles all the 'get' code
ResultSet myRS;
myRS = dbBean.searchBySampleIdStandard(Integer.toString(sampleID));
myRS.next();
%>
<%
    String sIdent;
    if (myRS.getString(2)==null) {
        sIdent = "";
    } else {
        sIdent = myRS.getString(2);
    }
    String sNumber;
    if (myRS.getString(3) == null) {
        sNumber = "";
    } else {
        sNumber = myRS.getString(3);
    }
    String sDate;
    if(myRS.getString(4) == null) {
        sDate = "";
    } else {
        sDate = myRS.getString(4);
    }
    String sHistory;
    if (myRS.getString(10) == null) {
        sHistory = "";
    } else {
        sHistory = myRS.getString(10);
    }
    String sNotes;
    if (myRS.getString(9) == null) {
        sNotes = "";
    } else {
        sNotes = myRS.getString(9);
    }
    String sGenus;
    if (myRS.getString(5) == null) {
        sGenus = "";
    } else {
        sGenus = myRS.getString(5);
    }
    String sSpecies;
    if (myRS.getString(6) == null) {
        sSpecies = "";
    }

```

```

        } else {
            sSpecies = myRS.getString(6);
        }
        String sSubSpecies;
        if (myRS.getString(7) == null) {
            sSubSpecies = "";
        } else {
            sSubSpecies = myRS.getString(7);
        }
        String sOther;
        if (myRS.getString(8) == null) {
            sOther = "";
        } else {
            sOther = myRS.getString(8);
        }
        int tGenus = myRS.getInt(11);
        int tSpecies = myRS.getInt(12);
        int tSubSpecies = myRS.getInt(13);
        int tOther = myRS.getInt(14);

        myRS.close();
    %>
    <form action="updatestrain.jsp">
    <table border="0" cellpadding = "0" cellspacing = "2" align="center">
    <tr>
    <td colspan = "3">
        <center><a href="deletesampleconfirm.jsp?sampleid=<%=sampleID%>">Delete
Strain</a></center>
    </td>
    </tr>
    <tr>
    <td>
        Sample ID:
    </td>
    <td colspan="2">
        <input type="text" name = "sampleid" value="<%=sampleID%>" readonly>
    </td>
    </tr>
    <tr>
    <td>
        Identifier (RM):&nbsp;
    </td>
    <td>
        <select name="ident">
        <%
            myRS = dbBean.getIdentPrefix();
            if (myRS.next()) {
                myRS.first();
                do {
                    if (sIdent.compareTo(myRS.getString("ident")) == 0) {
                        out.write("<option value=" +
myRS.getInt("identifier_id") + " selected">" + myRS.getString("ident"));
                    } else {
                        out.write("<option value=" +
myRS.getInt("identifier_id") + ">" + myRS.getString("ident"));
                    }
                } while (myRS.next());
            }
            myRS.close();
        %>
        </select>
    </td>
    </tr>
    <tr>
    <td colspan="3">

```

```

        <td>
            Cycle Range:
        </td>
        <td>
            <input readonly type="text" name="lowCycle" size="5" value="<% if
(crntCycle != 0) { out.write(Integer.toString(crntCycle));} else
{out.write(sNumber);}%>"> To: <input type="text" name="highCycle" size="5"
value="<%if (request.getParameter("highCycle") != null)
{out.write(request.getParameter("highCycle"));}%>">
        </td>
    </tr>
    <tr>
        <td>
            Number (1222):&nbspnbspnbsp
        </td>
        <td>
            <input type="text" name="number" value="<%=sNumber%>">
        </td>
    </tr>
    <tr>
        <td>
            Date (2005-8-24):
        </td>
        <td>
            <input type="text" name="date" value="<%=sDate%>">
        </td>
    </tr>
    <tr>
        <td>
            Genus:
        </td>
        <td>
            <select name="genus" >
                <%
                    myRS = dbBean.getGenus();
                    if (myRS.next()) {
                        myRS.first();
                        do {

                            String cmpr = sGenus;

                            if (myRS.getString("name")!=null) {
                                if (cmpr.compareTo(myRS.getString("name")) == 0) {
                                    out.write("<option value='" +
myRS.getInt("genus_id") + "' selected>" + myRS.getString("name"));
                                } else {
                                    out.write("<option value='" +
myRS.getInt("genus_id") + "'>" + myRS.getString("name"));
                                }
                            } else {
                                if (cmpr.compareTo("")==0) {
                                    out.write("<option value='" +
myRS.getInt("genus_id") + "' selected>No Genus");
                                } else {
                                    out.write("<option value='" +
myRS.getInt("genus_id") + "'>No Genus");
                                }
                            }
                        } while (myRS.next());
                    }
                    myRS.close();
                %>
            </select>
        </td>
        <td>

```



```

        <%
            if (tGenus == 0) {
                out.write("<input type='checkbox' name='gtent'
value='1'>Tentative");
            } else {
                out.write("<input type='checkbox' name='gtent' value='1'
checked>Tentative");
            }
        %>
    </td>
</tr>
<tr>
    <td>
        Species:
    </td>
    <td>
        <select name="species">
            <%
                myRS = dbBean.getSpecies();
                if (myRS.next()) {
                    myRS.first();
                    do {
                        String cmpr = sSpecies;

                        if (myRS.getString("name")!=null) {
                            if (cmpr.compareTo(myRS.getString("name")) == 0) {
                                out.write("<option value='" +
myRS.getInt("species_id") + "' selected>" + myRS.getString("name"));
                            } else {
                                out.write("<option value='" +
myRS.getInt("species_id") + "'>" + myRS.getString("name"));
                            }
                        } else {
                            if (cmpr.compareTo("")==0) {
                                out.write("<option value='" +
myRS.getInt("species_id") + "' selected>No Species");
                            } else {
                                out.write("<option value='" +
myRS.getInt("species_id") + "'>No Species");
                            }
                        }
                    } while (myRS.next());
                }
                myRS.close();
            %>
        </select>
    </td>
    <td>
        <%
            if (tSpecies == 0) {
                out.write("<input type='checkbox' name='sptent'
value='1'>Tentative");
            } else {
                out.write("<input type='checkbox' name='sptent' value='1'
checked>Tentative");
            }
        %>
    </td>
</tr>
<tr>
    <td>
        Subspecies:
    </td>

```

```

<td>
    <select name="subspecies">
        <%
            myRS = dbBean.getSubSpecies();
            if (myRS.next()) {
                myRS.first();
                do {
                    String cmpr = sSubSpecies;

                    if (myRS.getString("name")!=null) {
                        if (cmpr.compareTo(myRS.getString("name")) == 0) {
                            out.write("<option value='" +
myRS.getInt("subspecies_id") + "' selected>" + myRS.getString("name"));
                        } else {
                            out.write("<option value='" +
myRS.getInt("subspecies_id") + "'>" + myRS.getString("name"));
                        }
                    } else {
                        if (cmpr.compareTo("")==0 ) {
                            out.write("<option value='" +
myRS.getInt("subspecies_id") + "' selected>No Subspecies");
                        } else {
                            out.write("<option value='" +
myRS.getInt("subspecies_id") + "'>No Subspecies");
                        }
                    }
                } while (myRS.next());
            }
            myRS.close();
        %>
    </select>
</td>
<td>
    <%
        if (tSubSpecies == 0) {
            out.write("<input type='checkbox' name='subtent'
value='1'>Tentative");
        } else {
            out.write("<input type='checkbox' name='subtent' value='1'
checked>Tentative");
        }
    %>
</td>
</tr>
<tr>
<td>
    Subtype:
</td>
<td>
        <select name="subtype">
            <%
                myRS = dbBean.getSubType();
                if (myRS.next()) {
                    myRS.first();
                    do {
                        String cmpr = sOther;

                        if (myRS.getString("name")!=null) {
                            if (cmpr.compareTo(myRS.getString("name")) == 0) {
                                out.write("<option value='" +
myRS.getInt("subtype_id") + "' selected>" + myRS.getString("name"));
                            } else {
                                out.write("<option value='" +
myRS.getInt("subtype_id") + "'>" + myRS.getString("name"));

```

```

        }
        } else {
            if (cmpr.compareTo("")==0) {
                out.write("<option value='" +
myRS.getInt("subtype_id") + "' selected>No Subtype");
            } else {
                out.write("<option value='" +
myRS.getInt("subtype_id") + "'>No Subtype");
            }
        }
    } while (myRS.next());
}
myRS.close();
%>
</select>
</td>
<td>
<%
    if (tother == 0) {
        out.write("<input type='checkbox' name='othertent'
value='1'>Tentative");
    } else {
        out.write("<input type='checkbox' name='othertent' value='1'
checked>Tentative");
    }
}
%>
</td>
</tr>
<tr>
    <td style="vertical-align: top">
        Notes:
    </td>
    <td colspan="2">
        <textarea name="notes" cols="65" rows="10"><%=sNotes%></textarea>
    </td>
</tr>
<tr>
    <td style="vertical-align: top">
        History:
    </td>
    <td colspan="2">
        <textarea name="history" cols="65" rows="5"><%=sHistory%></textarea>
    </td>
</tr>
<tr>
    <td colspan="3" align="right">
        <input type="submit" value="Submit Update" name="submit">
    </td>
</tr>
</table>
</form>
<%dbBean.dbClose();%>

<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="frMaldi" class="psm.mformat.formatMALDI" scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MALDI Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <%
        dbBean.dbConnect();
        int rtn;
        rtn =
dbBean.insertMALDI(request.getParameter("ident"),request.getParameter("date"),
request.getParameter("target"),request.getParameter("notes"),frMaldi.getMaldiMZ(),
frMaldi.getMaldiIntensity());
        if (rtn == 1) {
            out.write("This identifier does not exist, please press back and choose an
identifier already in the database.");
        } else if (rtn !=0) {
            out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
        } else if (rtn ==0 ) {
            out.write("Insert successfull!");
        }
        dbBean.dbClose();
    %>
    <%out.write(pageSource.getPageFooter()); %>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*" %>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <jsp:useBean id="frMaldi" class="psm.mformat.formatMALDITemp" scope="session"/>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope="session" />

    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Compare MS Data to
Database", "InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        frMaldi.formatMALDIFile(request);
        java.text.SimpleDateFormat df = new java.text.SimpleDateFormat("yyMMdd");
        Calendar cd = Calendar.getInstance();
    %>

        <form action="uploadfileredirect.jsp">
        <table align="center">
            <tr>
                <td colspan="2">
                    <center>Please check and make sure the data from the
file was pased correctly<br>and make any changes you wish before hitting
submit.<br><br></center>
                </td>
            </tr>
            <tr>
                <td>
                    Identification Number:&nbsp;
                </td>
                <td>
                    <input type="text" name="ident" readonly size=33
value="Temporary1">
                </td>
            </tr>
            <tr>
                <td>
                    Date:
                </td>
                <td>
                    <input readonly type="text" name="date" size=33
value="<%=df.format(cd.getTime())%>">
                </td>
            </tr>
            <tr>
                <td>
                    Filename:
                </td>
                <td>
                    <input type="text" name="notes" size=33
value="<%=frMaldi.getNotes()%>">
                </td>
            </tr>
            <tr>
                <td>
                    Mass to Charge/Intensity:
                </td>
                <td height=100px>
                    <textarea readonly name="mz" cols=25><%
                        ArrayList mz = frMaldi.getMaldiMZ();
                        ArrayList intensity = frMaldi.getMaldiIntensity();
                        for (int i= 0; i < mz.size(); i++) {
                            out.write(mz.get(i) + "\t\t" +intensity.get(i) +
"\n" );
                        }
                    %></textarea>
            </td>
        </tr>
        </table>
    </form>

```

```
        </tr>
        <tr>
        <td colspan="3" align="center">
        <br>
        <input type="submit">
        </td>
        </tr>
    </table>
</form>

<%
dbBean.dbClose();
out.write(pageSource.getPageFooter());
%>
```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MALDI Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>

    <form method=post action="uploadcompareparse.jsp" enctype="MULTIPART/FORM-DATA">
    <table align = center>
        <tr>
            <td colspan = 2>
                <center><h1>Upload MALDI-TOF MS Data</h1></center>
            </td>
        </tr>
        <tr>
            <td>
                Find File:
            </td>
            <td>
                <input type="file" name="maldifile">
            </td>
        </tr>
        <tr>
            <td></td>
            <td colspan="1" align=left>
                <input type="submit" value="Send">
            </td>
        </tr>
    </table>
</form>
<%out.write(pageSource.getPageFooter());%>

```

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <jsp:useBean id="dbBean" class="psm.db.dbAccess" scope = "session"/>
    <jsp:useBean id="frMaldi" class="psm.mformat.formatMALDITemp" scope="session"/>
    <jsp:useBean id="pageSource" class="psm.web.pageCode"/>
    <%
        dbBean.dbConnect();
        pageSource.splitTemplatePage(dbBean.getUsername(), "Insert MALDI Data",
"InsertMALDITitle", application.getRealPath("/"), dbBean);
        out.write(pageSource.getPageHeader());
        dbBean.dbClose();
    %>
    <%
        dbBean.dbConnect();
        int rtn;
        rtn =
dbBean.insertMALDI(request.getParameter("ident"),request.getParameter("date"),
request.getParameter("target"),request.getParameter("notes"),frMaldi.getMaldiMZ(),
frMaldi.getMaldiIntensity());
        if (rtn == 1) {
            out.write("This identifier does not exist, please press back and choose an
identifier already in the database.");
        } else if (rtn ==9) {
            out.write("An error occurred while processing this request, please try
again. If this continues contact support.");
        } else if (rtn !=9 ) {
            out.write("Insert successfull!");
        }
        dbBean.dbClose();%>
        <center><br>Please wait while the page is redirected to the comparison setup.
Alternatively you may also click <a href =
"./maldicomparesetup.jsp?maldiid=<%=rtn%>">here.</a></center>
        <script>window.location = "./maldicomparesetup.jsp?maldiid=<%=rtn%>"</script>
    %>out.write(pageSource.getPageFooter()); %>

```



```
/*
 * box.java
 *
 * Created on July 13, 2005, 12:58 PM
 *
 * This is used as a container class to keep a list of two related objects
 */

package psm.db.box;

/**
 *
 * @author Brandon
 */
public class box {
    private String o1;
    private String o2;
    /** Creates a new instance of box */
    public box(String mo1, String mo2) {
        setO1(mo1);
        setO2(mo2);
    }

    public String getO1() {
        return o1;
    }

    public void setO1(String o1) {
        this.o1 = o1;
    }

    public String getO2() {
        return o2;
    }

    public void setO2(String o2) {
        this.o2 = o2;
    }
}
```

```
/*
 * masscolorbox.java
 *
 * Created on August 25, 2005, 11:42 AM
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package psm.db.box;
import java.awt.*;
/**
 *
 * @author User
 */
public class masscolorbox {

    private int mass;
    private Color highlighted;

    /** Creates a new instance of masscolorbox */
    public masscolorbox() {

    }

    public int getMass() {
        return mass;
    }

    public void setMass(int mass) {
        this.mass = mass;
    }

    public Color getHighlighted() {
        return highlighted;
    }

    public void setHighlighted(Color highlighted) {
        this.highlighted = highlighted;
    }

}
```

```

/*
 * box.java
 *
 * Created on July 13, 2005, 12:58 PM
 *
 * This is a container that holds the values from an ms_ms to in silico compare and
 * computes the p value and stores the related information
 */

package psm.db.box;
import java.math.*;
/**
 *
 * @author Brandon
 */
public class scorecontainer {
    private String mid;
    private String score;
    private String scorebeta;
    private int lc_k;
    private int uc_k;
    private int n;
    private double nstar;
    //These are not input able
    private double s;
    private double pvalue;

    /** Creates a new instance of box */
    public scorecontainer(String fmid, String fscore, int flc_k, int fuc_k, int fn,
double fnstar) {
        setMid(fmid);
        setScore(fscore);
        setLc_k(flc_k);
        setUc_k(fuc_k);
        setN(fn);
        setNstar(fnstar);
        //Computation
        setS(1 - Math.exp(-n/nstar));
        setPvalue(Pk());
    }

    public scorecontainer(String fmid, String fscore, String fscorebeta, int flc_k,
int fuc_k, int fn, double fnstar) {
        setMid(fmid);
        setScore(fscore);
        setScorebeta(fscorebeta);
        setLc_k(flc_k);
        setUc_k(fuc_k);
        setN(fn);
        setNstar(fnstar);
        //Computation
        setS(1 - Math.exp(-n/nstar));
        setPvalue(Pk());
    }

    public scorecontainer(String fmid, String fscore, String fscorebeta) {
        setMid(fmid);
        setScore(fscore);
        setScorebeta(fscorebeta);

        setLc_k(0);
        setUc_k(0);
        setN(0);
        setNstar(0);
        setS(0);
    }

```

```

        setPvalue(0);
    }

    public scorecontainer(scorecontainer sc) {
        setMid(sc.getMid());
        setScore(sc.getScore());
        setLc_k(sc.getLc_k());
        setUc_k(sc.getUc_k());
        setN(sc.getN());
        setNstar(sc.getNstar());
        setS(sc.getS());
        setPvalue(sc.getPvalue());
    }

    private double Pk() {
        BigDecimal pval = BigDecimal.valueOf(0);

        for (int k=getLc_k();k<=getUc_k();k++) {
            BigDecimal p = factks(k).divide(fact(getUc_k() - k));

            p = p.multiply(BigDecimal.valueOf(Math.pow(getS(), k) *
Math.pow(1-getS(), getUc_k() - k)));
            pval = pval.add(p);
        }

        return pval.doubleValue();
    }

    private BigDecimal fact(int x) {
        BigDecimal l = BigDecimal.valueOf(1);
        for (int i = 2; i <= x; i++) {
            l = l.multiply(BigDecimal.valueOf(i));
        }
        return l;
    }

    private BigDecimal factks(int x) {
        //This function takes the first part of the p value,  $\kappa!/k!$ 
        //This is equivalent to the multiplication of all the numbers of k+1 to  $\kappa$ 
        //Makes for a smaller number, the larger one does not need to be stored.
        BigDecimal l = BigDecimal.valueOf(1);
        for (int i = x + 1; i <= getUc_k(); i++) {
            l = l.multiply(BigDecimal.valueOf(i));
        }
        return l;
    }

    public String getMid() {
        return mid;
    }

    public void setMid(String mid) {
        this.mid = mid;
    }

    public String getScore() {
        return score;
    }

    public void setScore(String score) {
        this.score = score;
    }

```

```
public String getScorebeta() {
    return scorebeta;
}

public void setScorebeta(String score) {
    this.scorebeta = score;
}

public int getLc_k() {
    return lc_k;
}

public void setLc_k(int lc_k) {
    this.lc_k = lc_k;
}

public int getUc_k() {
    return uc_k;
}

public void setUc_k(int uc_k) {
    this.uc_k = uc_k;
}

public int getN() {
    return n;
}

public void setN(int n) {
    this.n = n;
}

public double getNstar() {
    return nstar;
}

public void setNstar(double nstar) {
    this.nstar = nstar;
}

public double getS() {
    return s;
}

private void setS(double s) {
    this.s = s;
}

public double getPvalue() {
    return pvalue;
}

private void setPvalue(double pvalue) {
    this.pvalue = pvalue;
}

}
```

```

/*
 * dbAccess.java
 *
 * Created on June 22, 2005, 10:07 AM
 */

package psm.db;

import java.beans.*;
import java.io.*;
import java.sql.*;
import java.util.*;

/**
 * @author Brandon Garbus
 */
public class dbAccess {

    private String SQLQuery;
    private String password;
    private String username;
    private java.sql.Statement stmt;
    private Connection myConn;

    public String buildCompareString(int intensityBound, int lowerMassBound, int
higherMassBound, int error) {
        /*
         *THIS IS NOT USED ANYMORE use buildCompareStringForTemp
         *This is because all data is now stored in temporary tables for comparison
to reduce
         *search times!
        */

        //$cmpr$ = the comparison ID
        //$cmprTo$ = the comparison ID cmpr is currently being compared to
        //$lowMass$ = the lower mass bound
        //$highMass$ = the higher mass bound
        //$minIntensity$ = the minimum intensity range
        //$error$ = the error range in ppm (parts per million)

        /*
         *The Formula used for the comparison is  $\text{SUM}(A+B - [2*\text{ABS}(A-B)]) / \text{SUM}(A+B)$ 
where A and B are the intensities of two separate data sets to compare
         *This gives a range of -1.0 to 1.0 which is then multiplied by 100 to get a
percentage
         *The select statement allows for a multitude of search options, it deals
with correcting error and takes the average of the two masses (which is not used for
comparison)
         *and combines the two intensities into one row.
        */

        String selStr;
        //First create the select statement without any values in it
        //This is the select statement for data within the same table. For MALDI
data not in this table a separate
        //Temporary table will be built and joined the same way
        selStr = "SELECT ROUND(sum(score2)/sum(score) * 100,0) score FROM ((select
(a.mass + b.mass) / 2 mass, a.intensity as 'Ai', b.intensity as 'Bi', (SELECT Ai +
Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
maldi_tof_data a, maldi_tof_data b where a.mass between (b.mass - (b.mass *
$error$)) and (b.mass + (b.mass * $error$)) and a.intensity > $minIntensity$ and
a.mass between $lowMass$ and $highMass$ and b.mass between $lowMass$ and
$highMass$ and (a.maldi_tof_id < b.maldi_tof_id or a.maldi_tof_id < b.maldi_tof_id)
and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) union (select distinct
a.mass, a.intensity as 'Ai', 0 as 'Bi', (SELECT Ai + Bi) as 'score', (SELECT score -

```

```

(ABS(Ai - Bi) * 2.0)) as 'score2' from maldi_tof_data a where a.maldi_tof_id =
$cmpr$ and a.intensity > $minIntensity$ and a.mass between $lowMass$ and $highMass$
and a.mass not in";
    selStr += " (select distinct a.mass from maldi_tof_data b where a.mass
between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
(a.maldi_tof_id < b.maldi_tof_id or a.maldi_tof_id < b.maldi_tof_id) and
b.maldi_tof_id = $cmprTo$ and b.mass between $lowMass$ and $highMass$));";
    selStr += " union (select distinct b.mass, 0 as 'Ai', intensity as 'Bi',
(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
malDI_tof_data b where b.maldi_tof_id = $cmprTo$ and b.intensity > $minIntensity$
and b.mass between $lowMass$ and $highMass$";
    selStr += " and b.mass not in (select b.mass from maldi_tof_data a where
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.intensity > $minIntensity$ and (a.maldi_tof_id <
b.maldi_tof_id or a.maldi_tof_id < b.maldi_tof_id) and a.maldi_tof_id = $cmpr$ and
a.mass between $lowMass$ and $highMass$)) maldi";
    //now using the definitions above, insert the values
    selStr = selStr.replace("$lowMass$", Integer.toString(lowerMassBound));
    selStr = selStr.replace("$highMass$", Integer.toString(higherMassBound));
    selStr = selStr.replace("$minIntensity$", Integer.toString(intensityBound));
    //Convert ppm to a percentage and replace it everywhere
    double mErr = error/1000000.0;
    selStr = selStr.replace("$error$", Double.toString(mErr));

    return selStr;
}

public String buildCompareStringForTemp(int intensityBound, int lowerMassBound,
int higherMassBound, int error) {
    //$cmpr$ = the comparison ID
    //$cmprTo$ = the comparison ID cmpr is currently being compared to
    //$lowMass$ = the lower mass bound
    //$highMass$ = the higher mass bound
    //$minIntensity$ = the minimum intensity range
    //$error$ = the error range in ppm (parts per million)

    /*
    *The Formula used for the comparison is SUM(A+B - [2*ABS(A-B)]) / SUM (A+B)
    where A and B are the intensities of two separate data sets to compare
    *This gives a range of -1.0 to 1.0 which is then multiplied by 100 to get a
    percentage
    *The select statement allows for a multitude of search options, it deals
    with correcting error and takes the average of the two masses (which is not used for
    comparison)
    *and combines the two intensities into one row.
    */

    String selStr;
    //First create the select statement without any values in it
    //This is the select statement for data within the same table. For MALDI
    data not in this table a separate
    //Temporary table will be built and joined the same way
    //Check the mass range, if it is between 4 and 14 thousand use the
    comparison table of data that is already thresholded
    if (lowerMassBound == 4000 & higherMassBound == 14000) {
        selStr = "SELECT ROUND(sum(score2)/sum(score) * 100,0) score FROM
((select (a.mass + b.mass) / 2 mass, a.intensity as 'Ai', b.intensity as 'Bi',
(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
malDItemp a, malDIRat1 b where a.mass between (b.mass - (b.mass * $error$)) and
(b.mass + (b.mass * $error$)) and a.intensity > $minIntensity$ and a.mass between
$lowMass$ and $highMass$ and b.mass between $lowMass$ and $highMass$ and
a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) union (select distinct
a.mass, a.intensity as 'Ai', 0 as 'Bi', (SELECT Ai + Bi) as 'score', (SELECT score -
(ABS(Ai - Bi) * 2.0)) as 'score2' from malDItemp a where a.maldi_tof_id = $cmpr$ and
a.intensity > $minIntensity$ and a.mass between $lowMass$ and $highMass$ and a.mass

```

```

not in";
    selStr += " (select distinct a.mass from maldirat1 b where a.mass
between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
b.maldi_tof_id = $cmprTo$ and b.mass between $lowMass$ and $highMass$))";
    selStr += " union (select distinct b.mass, 0 as 'Ai', intensity as 'Bi',
(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
maldirat1 b where b.maldi_tof_id = $cmprTo$ and b.intensity > $minIntensity$ and
b.mass between $lowMass$ and $highMass$";
    selStr += " and b.mass not in (select b.mass from maldiTemp a where
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.intensity > $minIntensity$ and a.maldi_tof_id = $cmpr$ and a.mass between
$lowMass$ and $highMass$))) maldi";
    } else {
        selStr = "SELECT ROUND(sum(score2)/sum(score) * 100,0) score FROM
((select (a.mass + b.mass) / 2 mass, a.intensity as 'Ai', b.intensity as 'Bi',
(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
maldiTemp a, maldiTemp2 b where a.mass between (b.mass - (b.mass * $error$)) and
(b.mass + (b.mass * $error$)) and a.intensity > $minIntensity$ and a.mass between
$lowMass$ and $highMass$ and b.mass between $lowMass$ and $highMass$ and
a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) union (select distinct
a.mass, a.intensity as 'Ai', 0 as 'Bi', (SELECT Ai + Bi) as 'score', (SELECT score -
(ABS(Ai - Bi) * 2.0)) as 'score2' from maldiTemp a where a.maldi_tof_id = $cmpr$ and
a.intensity > $minIntensity$ and a.mass between $lowMass$ and $highMass$ and a.mass
not in";
        selStr += " (select distinct a.mass from maldiTemp2 b where a.mass
between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
b.maldi_tof_id = $cmprTo$ and b.mass between $lowMass$ and $highMass$))";
        selStr += " union (select distinct b.mass, 0 as 'Ai', intensity as 'Bi',
(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
maldiTemp2 b where b.maldi_tof_id = $cmprTo$ and b.intensity > $minIntensity$ and
b.mass between $lowMass$ and $highMass$";
        selStr += " and b.mass not in (select b.mass from maldiTemp a where
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.intensity > $minIntensity$ and a.maldi_tof_id = $cmpr$ and a.mass between
$lowMass$ and $highMass$))) maldi";
    }
    //now using the definitions above, insert the values
    selStr = selStr.replace("$lowMass$", Integer.toString(lowerMassBound));
    selStr = selStr.replace("$highMass$", Integer.toString(higherMassBound));
    selStr = selStr.replace("$minIntensity$", Integer.toString(intensityBound));
    //Convert ppm to a percentage and replace it everywhere
    double mErr = error/1000000.0;
    selStr = selStr.replace("$error$", Double.toString(mErr));
    //Since this is the tempTable the ID is always 0
    return selStr;
}

public String buildCompareStringForTempMassOnly(int intensityBound, int
lowerMassBound, int higherMassBound, int error) {
    //$cmpr$ = the comparison ID
    //$cmprTo$ = the comparison ID cmpr is currently being compared to
    //$lowMass$ = the lower mass bound
    //$highMass$ = the higher mass bound
    //$minIntensity$ = the minimum intensity range
    //$error$ = the error range in ppm (parts per million)

    /*
    *The Formula used for the comparison is SUM(A+B - [2*ABS(A-B)]) / SUM (A+B)
    where A and B are the intensities of two separate data sets to compare
    *This gives a range of -1.0 to 1.0 which is then multiplied by 100 to get a
    percentage
    *The select statement allows for a multitude of search options, it deals
    with correcting error and takes the average of the two masses (which is not used for
    comparison)

```



```

        *and combines the two intensities into one row.
        */

        String selStr;
        //First create the select statement without any values in it
        //This is the select statement for data within the same table. For MALDI
data not in this table a separate
        //Temporary table will be built and joined the same way
        //Check the mass range, if it is between 4 and 14 thousand use the
comparison table of data that is already thresholded
        if (lowerMassBound == 4000 & higherMassBound == 14000) {
            //This was the original, new one is used for calculating pvalue
            //selStr = "SELECT ROUND(((SELECT COUNT(*) FROM (select DISTINCT a.mass
mass from maldiTemp a, maldirat1 b where a.intensity > $minIntensity$ AND
b.intensity > $minIntensity$ AND a.mass between (b.mass - (b.mass * $error$)) and
(b.mass + (b.mass * $error$)) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) m)) * 2 /((select COUNT(*)from maldiTemp a where
a.intensity > $minIntensity$ AND a.mass between $lowMass$ and $highMass$ AND
a.maldi_tof_id = $cmpr$) +(Select COUNT(*)from maldirat1 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$)) * 100, 0) score";
            selStr = "SELECT lc_k, uc_k, n, Round(lc_k * 2 / (n + uc_k) * 100, 0) as
score FROM (SELECT (SELECT COUNT(*) FROM (select DISTINCT a.mass from maldiTemp a,
maldirat1 b where a.intensity > $minIntensity$ AND b.intensity > $minIntensity$ AND
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.mass between $lowMass$ and $highMass$ and b.mass between $lowMass$ and
$highMass$ and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) as m) as lc_k,
(select COUNT(*) from maldiTemp a where a.intensity > $minIntensity$ AND a.mass
between $lowMass$ and $highMass$ AND a.maldi_tof_id = $cmpr$) as n, (Select
COUNT(*) from maldirat1 b where b.intensity > $minIntensity$ AND b.mass between
$lowMass$ and $highMass$ AND b.maldi_tof_id = $cmprTo$) as uc_k) as tbl";
        } else {
            //selStr = "SELECT ROUND(((SELECT COUNT(*) FROM (select DISTINCT a.mass
mass from maldiTemp a, maldiTemp2 b where a.intensity > $minIntensity$ AND
b.intensity > $minIntensity$ AND a.mass between (b.mass - (b.mass * $error$)) and
(b.mass + (b.mass * $error$)) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) m)) * 2 /((select COUNT(*)from maldiTemp a where
a.intensity > $minIntensity$ AND a.mass between $lowMass$ and $highMass$ AND
a.maldi_tof_id = $cmpr$) +(Select COUNT(*)from maldiTemp2 b where b.intensity >
$minIntensity$ AND b.mass between $lowMass$ and $highMass$ AND b.maldi_tof_id =
$cmprTo$)) * 100, 0) score";
            selStr = "SELECT lc_k, uc_k, n, Round(lc_k * 2 / (n + uc_k) * 100, 0) as
score FROM (SELECT (SELECT COUNT(*) FROM (select DISTINCT a.mass from maldiTemp a,
maldiTemp2 b where a.intensity > $minIntensity$ AND b.intensity > $minIntensity$ AND
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.mass between $lowMass$ and $highMass$ and b.mass between $lowMass$ and
$highMass$ and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) as m) as lc_k,
(select COUNT(*) from maldiTemp a where a.intensity > $minIntensity$ AND a.mass
between $lowMass$ and $highMass$ AND a.maldi_tof_id = $cmpr$) as n, (Select
COUNT(*) from maldiTemp2 b where b.intensity > $minIntensity$ AND b.mass between
$lowMass$ and $highMass$ AND b.maldi_tof_id = $cmprTo$) as uc_k) as tbl";
        }
        //now using the definitions above, insert the values
        selStr = selStr.replace("$lowMass$", Integer.toString(lowerMassBound));
        selStr = selStr.replace("$highMass$", Integer.toString(higherMassBound));
        selStr = selStr.replace("$minIntensity$", Integer.toString(intensityBound));
        //Convert ppm to a percentage and replace it everywhere
        double mErr = error/1000000.0;
        selStr = selStr.replace("$error$", Double.toString(mErr));
        //Since this is the tempTable the ID is always 0

        return selStr;
    }

```

```

public String buildCompareStringwithoutTotalScore(int intensityBound, int
lowerMassBound, int higherMassBound, int error) {
    //$cmpr$ = the comparison ID
    //$cmprTo$ = the comparison ID cmpr is currently being compared to
    //$lowMass$ = the lower mass bound
    //$highMass$ = the higher mass bound
    //$minIntensity$ = the minimum intensity range
    //$error$ = the error range in ppm (parts per million)

    /*
    *The Formula used for the comparison is  $\text{SUM}(A+B - [2*\text{ABS}(A-B)]) / \text{SUM}(A+B)$ 
    where A and B are the intensities of two separate data sets to compare
    *This gives a range of -1.0 to 1.0 which is then multiplied by 100 to get a
    percentage
    *The select statement allows for a multitude of search options, it deals
    with correcting error and takes the average of the two masses (which is not used for
    comparison)
    *and combines the two intensities into one row.
    */

    String selStr;
    //First create the select statement without any values in it
    //This is the select statement for data within the same table. For MALDI
data not in this table a separate
    //Temporary table will be built and joined the same way
    //This is the statement to get the results of the comparison without the
score
    if (lowerMassBound == 4000 & higherMassBound == 14000) {
        selStr = "(select (a.mass + b.mass) / 2 mass, a.intensity as 'Ai',
b.intensity as 'Bi', (SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) *
2.0)) as 'score2' from maldiTemp a, maldirat1 b where a.mass between (b.mass -
(b.mass * $error$)) and (b.mass + (b.mass * $error$)) and a.intensity >
$minIntensity$ and a.mass between $lowMass$ and $highMass$ and b.mass between
$lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$)
union (select distinct a.mass, a.intensity as 'Ai', 0 as 'Bi', (SELECT Ai + Bi) as
'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from maldiTemp a where
a.maldi_tof_id = $cmpr$ and a.intensity > $minIntensity$ and a.mass between
$lowMass$ and $highMass$ and a.mass not in";
        selStr += " (select distinct a.mass from maldirat1 b where a.mass
between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
b.maldi_tof_id = $cmprTo$ and b.mass between $lowMass$ and $highMass$))";
        selStr += " union (select distinct b.mass, 0 as 'Ai', intensity as 'Bi',
(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
maldirat1 b where b.maldi_tof_id = $cmprTo$ and b.intensity > $minIntensity$ and
b.mass between $lowMass$ and $highMass$";
        selStr += " and b.mass not in (select b.mass from maldiTemp a where
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.intensity > $minIntensity$ and a.maldi_tof_id = $cmpr$ and a.mass between
$lowMass$ and $highMass$))";
    } else {
        selStr = "(select (a.mass + b.mass) / 2 mass, a.intensity as 'Ai',
b.intensity as 'Bi', (SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) *
2.0)) as 'score2' from maldiTemp a, malditemp2 b where a.mass between (b.mass -
(b.mass * $error$)) and (b.mass + (b.mass * $error$)) and a.intensity >
$minIntensity$ and a.mass between $lowMass$ and $highMass$ and b.mass between
$lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$)
union (select distinct a.mass, a.intensity as 'Ai', 0 as 'Bi', (SELECT Ai + Bi) as
'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from maldiTemp a where
a.maldi_tof_id = $cmpr$ and a.intensity > $minIntensity$ and a.mass between
$lowMass$ and $highMass$ and a.mass not in";
        selStr += " (select distinct a.mass from malditemp2 b where a.mass
between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
b.maldi_tof_id = $cmprTo$ and b.mass between $lowMass$ and $highMass$))";
        selStr += " union (select distinct b.mass, 0 as 'Ai', intensity as 'Bi',

```

```

(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
malditemp2 b where b.maldi_tof_id = $cmprTo$ and b.intensity > $minIntensity$ and
b.mass between $lowMass$ and $highMass$";
    selStr += " and b.mass not in (select b.mass from maldiTemp a where
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.intensity > $minIntensity$ and a.maldi_tof_id = $cmpr$ and a.mass between
$lowMass$ and $highMass$))";
}

```

```

//now using the definitions above, insert the values
selStr = selStr.replace("$lowMass$", Integer.toString(lowerMassBound));
selStr = selStr.replace("$highMass$", Integer.toString(higherMassBound));
selStr = selStr.replace("$minIntensity$", Integer.toString(intensityBound));
//Convert ppm to a percentage and replace it everywhere
double mErr = error/1000000.0;
selStr = selStr.replace("$error$", Double.toString(mErr));

```

```

return selStr;
}

```

```

public String buildCompareStringForTempwithoutTotalScore(int intensityBound, int
lowerMassBound, int higherMassBound, int error) {

```

```

/*
 *This is not the same as above because the data in the temp table is
deleted for in database comparisons
 *This is to help reduce clutter while increasing search times and reducing
the memory load
 */
// $cmpr$ = the comparison ID
// $cmprTo$ = the comparison ID cmpr is currently being compared to
// $lowMass$ = the lower mass bound
// $highMass$ = the higher mass bound
// $minIntensity$ = the minimum intensity range
// $error$ = the error range in ppm (parts per million)

/*
 *The Formula used for the comparison is SUM(A+B - [2*ABS(A-B)]) / SUM (A+B)
where A and B are the intensities of two separate data sets to compare
 *This gives a range of -1.0 to 1.0 which is then multiplied by 100 to get a
percentage
 *The select statement allows for a multitude of search options, it deals
with correcting error and takes the average of the two masses (which is not used for
comparison)
 *and combines the two intensities into one row.
 */

```

```

String selStr;
//First create the select statement without any values in it
//This is the select statement for data within the same table. For MALDI
data not in this table a separate
//Temporary table will be built and joined the same way
if (lowerMassBound == 4000 & higherMassBound == 14000) {
    selStr = "(select (a.mass + b.mass) / 2 mass, a.intensity as 'Ai',
b.intensity as 'Bi', (SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) *
2.0)) as 'score2' from maldiTemp a, maldirat1 b where a.mass between (b.mass -
(b.mass * $error$)) and (b.mass + (b.mass * $error$)) and a.intensity >
$minIntensity$ and a.mass between $lowMass$ and $highMass$ and b.mass between
$lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$)
union (select distinct a.mass, a.intensity as 'Ai', 0 as 'Bi', (SELECT Ai + Bi) as
'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from maldiTemp a where
a.maldi_tof_id = $cmpr$ and a.intensity > $minIntensity$ and a.mass between
$lowMass$ and $highMass$ and a.mass not in";
    selStr += " (select distinct a.mass from maldirat1 b where a.mass
between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
b.maldi_tof_id = $cmprTo$ and b.mass between $lowMass$ and $highMass$))";
}

```

```

        selStr += " union (select distinct b.mass, 0 as 'Ai', intensity as 'Bi',
(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
maldirat1 b where b.maldi_tof_id = $cmprTo$ and b.intensity > $minIntensity$ and
b.mass between $lowMass$ and $highMass$";
        selStr += " and b.mass not in (select b.mass from maldiTemp a where
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.intensity > $minIntensity$ and a.maldi_tof_id = $cmpr$ and a.mass between
$lowMass$ and $highMass$))";
    } else {
        selStr = "(select (a.mass + b.mass) / 2 mass, a.intensity as 'Ai',
b.intensity as 'Bi', (SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) *
2.0)) as 'score2' from maldiTemp a, maldiTemp2 b where a.mass between (b.mass -
(b.mass * $error$)) and (b.mass + (b.mass * $error$)) and a.intensity >
$minIntensity$ and a.mass between $lowMass$ and $highMass$ and b.mass between
$lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$)
union (select distinct a.mass, a.intensity as 'Ai', 0 as 'Bi', (SELECT Ai + Bi) as
'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from maldiTemp a where
a.maldi_tof_id = $cmpr$ and a.intensity > $minIntensity$ and a.mass between
$lowMass$ and $highMass$ and a.mass not in";
        selStr += " (select distinct a.mass from maldiTemp2 b where a.mass
between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
b.maldi_tof_id = $cmprTo$ and b.mass between $lowMass$ and $highMass$))";
        selStr += " union (select distinct b.mass, 0 as 'Ai', intensity as 'Bi',
(SELECT Ai + Bi) as 'score', (SELECT score - (ABS(Ai - Bi) * 2.0)) as 'score2' from
maldiTemp2 b where b.maldi_tof_id = $cmprTo$ and b.intensity > $minIntensity$ and
b.mass between $lowMass$ and $highMass$";
        selStr += " and b.mass not in (select b.mass from maldiTemp a where
a.mass between (b.mass - (b.mass * $error$)) and (b.mass + (b.mass * $error$)) and
a.intensity > $minIntensity$ and a.maldi_tof_id = $cmpr$ and a.mass between
$lowMass$ and $highMass$))";
    }
    selStr = selStr.replace("$lowMass$", Integer.toString(lowerMassBound));
    selStr = selStr.replace("$highMass$", Integer.toString(higherMassBound));
    selStr = selStr.replace("$minIntensity$", Integer.toString(intensityBound));
    //Convert ppm to a percentage and replace it everywhere
    double mErr = error/1000000.0;
    selStr = selStr.replace("$error$", Double.toString(mErr));
    //Since this is the tempTable the ID is always 0

    return selStr;
}

public boolean changePassowrd(String newpwd) {
    try {
        String mySQL = "SET PASSWORD = PASSWORD('";
        mySQL = mySQL.concat(newpwd);
        mySQL = mySQL.concat("')");
        stmt = myConn.createStatement();
        stmt.executeUpdate(mySQL);
        password = newpwd;
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

public void deleteMaldiTempData(int tmpID, String tmpTable) {
    try {
        stmt = myConn.createStatement();
        stmt.executeUpdate("DELETE FROM " + tmpTable + " WHERE maldi_tof_id = "

```

```

+ tmpID);
    } catch (SQLException e) {
    } catch (Exception e) {
    }
}

public dbAccess() {
    myConn = null;
    SQLQuery= "";
}

public int dbClose() {
    if (myConn != null) {
        try {myConn.close();} catch(Exception e) {e.printStackTrace();}
    }
    return 0;
}

public int dbConnect() {
    String url = "jdbc:mysql://localhost:3306/psm";
    try {
        Class.forName("com.mysql.jdbc.Driver");
        myConn = DriverManager.getConnection(url, username, password);
        if(myConn != null){
            if (username.compareTo("rmandrell") == 0 |
username.compareTo("psmlogin") == 0) {
                SQLQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number), sample.date, concat_ws(' ', genus.name,
species.name, subspecies.name, subtype.name), sample.notes, sample.history,
sample.tentative_genus, sample.tentative_species, sample.tentative_subspecies,
sample.tentative_subtype FROM identifier, sample, strain, genus, species,
subspecies, subtype WHERE strain.genus_id = genus.genus_id AND strain.species_id =
species.species_id AND subspecies.subspecies_id = strain.subspecies_id AND
subtype.subtype_id = strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";
            } else {
                SQLQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number), sample.date, concat_ws(' ', genus.name,
species.name, subspecies.name, subtype.name), CASE WHEN sample.notes LIKE
'%CONFIDENTIAL%' THEN 'CONFIDENTIAL See Rob Mandrell before using' ELSE sample.notes
END, sample.history, sample.tentative_genus, sample.tentative_species,
sample.tentative_subspecies, sample.tentative_subtype FROM identifier, sample,
strain, genus, species, subspecies, subtype WHERE strain.genus_id = genus.genus_id
AND strain.species_id = species.species_id AND subspecies.subspecies_id =
strain.subspecies_id AND subtype.subtype_id = strain.subtype_id AND sample.strain_id
= strain.strain_id AND identifier.identifier_id = sample.identifier_id";
            }
            return 0;
        } else {
            return 3;
        }
    } catch (SQLException e) {
        return 1;
    } catch (Exception e) {
        return 2;
    }
}

public int dbConnect(String db) {
    String url = "jdbc:mysql://localhost:3306/" + db;
    try {
        Class.forName("com.mysql.jdbc.Driver");

```

```

        myConn = DriverManager.getConnection(url, username, password);
        if(myConn != null){
            if (username.compareTo("rmandrell") == 0 |
username.compareTo("psmlogin") == 0) {
                SQLQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number), sample.date, concat_ws(' ', genus.name,
species.name, subspecies.name, subtype.name), sample.notes, sample.history,
sample.tentative_genus, sample.tentative_species, sample.tentative_subspecies,
sample.tentative_subtype FROM identifier, sample, strain, genus, species,
subspecies, subtype WHERE strain.genus_id = genus.genus_id AND strain.species_id =
species.species_id AND subspecies.subspecies_id = strain.subspecies_id AND
subtype.subtype_id = strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";
            } else {
                SQLQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number), sample.date, concat_ws(' ', genus.name,
species.name, subspecies.name, subtype.name), CASE WHEN sample.notes LIKE
'%CONFIDENTIAL%' THEN 'CONFIDENTIAL See Rob Mandrell before using' ELSE sample.notes
END, sample.history, sample.tentative_genus, sample.tentative_species,
sample.tentative_subspecies, sample.tentative_subtype FROM identifier, sample,
strain, genus, species, subspecies, subtype WHERE strain.genus_id = genus.genus_id
AND strain.species_id = species.species_id AND subspecies.subspecies_id =
strain.subspecies_id AND subtype.subtype_id = strain.subtype_id AND sample.strain_id
= strain.strain_id AND identifier.identifier_id = sample.identifier_id";
            }
            return 0;
        } else {
            return 3;
        }
    } catch (SQLException e) {
        return 1;
    } catch (Exception e) {
        return 2;
    }
}

public int dumpMALDIDataToTemp(int maldiid, int intensity, int lmass, int hmass,
String tmpTable) {
    int id;
    try {
        //FIRST cleanup the table, delete anything that is more than a day old
        stmt = myConn.createStatement();
        stmt.executeUpdate("DELETE FROM malditemp WHERE createTime <
DATE_ADD(CURRENT_TIMESTAMP, INTERVAL -30 MINUTE)");

        stmt = myConn.createStatement();
        stmt.executeUpdate("DELETE FROM malditemp2 WHERE createTime <
DATE_ADD(CURRENT_TIMESTAMP, INTERVAL -30 MINUTE)");
        //set then get the last ID used
        stmt = myConn.createStatement();
        stmt.executeUpdate("UPDATE lastidused set id = id + 1 WHERE description
= 'malDI'");
        java.sql.ResultSet myRS = getRS("SELECT id FROM lastidused WHERE
description = 'malDI'");
        myRS.next();
        id = myRS.getInt("id");
        myRS.close();

        stmt = myConn.createStatement();
        //WE only want to insert the data we are going to use, so only insert
        data with the proper ranges
        String maxStr = "SELECT intensity FROM maldi_tof_data where maldi_tof_id
= " + maldiid + " AND mass BETWEEN " + lmass + " AND " + hmass + " ORDER BY
intensity DESC Limit 1";
    }
}

```

```

        String templInStr = "";
        if (tmpTable.compareTo("malditemp") == 0) {
            templInStr = "INSERT INTO " + tmpTable + " (SELECT * FROM (SELECT "
+ id + " as 'maldi_tof_id', mass, (intensity * 100 / (" + maxStr + ")) as
'intensity', CURRENT_TIMESTAMP FROM maldi_tof_data where maldi_tof_id = " + maldiid
+ " AND mass between " + lmass + " AND " + hmass + ") m WHERE intensity > " +
intensity + ")";
        } else if (tmpTable.compareTo("malditemp2") == 0) {
            templInStr = "INSERT INTO " + tmpTable + " (SELECT * FROM (SELECT "
+ id + " as 'maldi_tof_id', mass, (intensity * 100 / (" + maxStr + ")) as
'intensity', CURRENT_TIMESTAMP FROM maldi_tof_data where maldi_tof_id = " + maldiid
+ " AND mass between " + lmass + " AND " + hmass + ") m WHERE intensity > " +
intensity + ")";
        }
        stmt.executeUpdate(templInStr);
        stmt.close();
    } catch (SQLException e) {
        return 0;
    } catch (Exception e) {
        return 0;
    }
    return id;
}

public int dumpINSILICODDataToTemp(int insilicoid, int intensity, int lmass, int
hmass, String tmpTable, String residue) {
    int id;
    try {

        //FIRST cleanup the table, delete anything that is more than a day old
        stmt = myConn.createStatement();
        stmt.executeUpdate("DELETE FROM malditemp WHERE createTime <
DATE_ADD(CURRENT_TIMESTAMP, INTERVAL -30 MINUTE)");
        //FIRST cleanup the table, delete anything that is more than a day old
        stmt = myConn.createStatement();
        stmt.executeUpdate("DELETE FROM malditemp2 WHERE createTime <
DATE_ADD(CURRENT_TIMESTAMP, INTERVAL -30 MINUTE)");
        //set then get the last ID used
        stmt = myConn.createStatement();
        stmt.executeUpdate("UPDATE lastidused set id = id + 1 WHERE description
= 'maldi'");
        java.sql.ResultSet myRS = getRS("SELECT id FROM lastidused WHERE
description = 'maldi'");
        myRS.next();
        id = myRS.getInt("id");
        myRS.close();

        stmt = myConn.createStatement();

        String templInStr = "";
        if (tmpTable.compareTo("malditemp") == 0) {
            templInStr = "INSERT INTO " + tmpTable + " (SELECT " + id + " as
'in_silico_id', mass, intensity, CURRENT_TIMESTAMP FROM in_silico_data where
in_silico_id = " + insilicoid + " AND mass between " + lmass + " AND " + hmass +
")";
        } else if (tmpTable.compareTo("malditemp2") == 0) {
            templInStr = "INSERT INTO " + tmpTable + " (SELECT " + id + " as
'in_silico_id', mass, intensity, CURRENT_TIMESTAMP FROM in_silico_data where
in_silico_id = " + insilicoid + " AND mass between " + lmass + " AND " + hmass;

            if (residue.length() > 0) {
                templInStr += " AND (" + residue + ")";
            }
        }
    }
}

```

```

        templInStr += "in_silico_data.residue LIKE '%" +
residue.charAt(v) + "%'";
        if (v != (residue.length() - 1)) {
            templInStr += " OR ";
        }
    }
    templInStr += ")";
}
templInStr += ")";

    }
    stmt.executeUpdate(templInStr);
    stmt.close();
} catch (SQLException e) {
    return 0;
} catch (Exception e) {
    return 0;
}
return id;
}

public int dumpMSMSDataToTemp(int msmsid, int intensity, int lmass, int hmass,
String tmpTable) {
    int id;
    try {
        //FIRST cleanup the table, delete anything that is more than a day old
        stmt = myConn.createStatement();
        stmt.executeUpdate("DELETE FROM malditemp WHERE createTime <
DATE_ADD(CURRENT_TIMESTAMP, INTERVAL -30 MINUTE)");
        //FIRST cleanup the table, delete anything that is more than a day old
        stmt = myConn.createStatement();
        stmt.executeUpdate("DELETE FROM malditemp2 WHERE createTime <
DATE_ADD(CURRENT_TIMESTAMP, INTERVAL -30 MINUTE)");
        //set then get the last ID used
        stmt = myConn.createStatement();
        stmt.executeUpdate("UPDATE lastidused set id = id + 1 WHERE description
= 'maldi'");
        java.sql.ResultSet myRS = getRS("SELECT id FROM lastidused WHERE
description = 'maldi'");
        myRS.next();
        id = myRS.getInt("id");
        myRS.close();

        stmt = myConn.createStatement();
        //WE only want to insert the data we are going to use, so only insert
data with the proper ranges
        String maxStr = "SELECT intensity FROM ms_ms_data where ms_ms_id = " +
msmsid + " AND mass BETWEEN " + lmass + " AND " + hmass + " ORDER BY intensity DESC
Limit 1";

        String templInStr = "";
        if (tmpTable.compareTo("malditemp") == 0) {
            templInStr = "INSERT INTO " + tmpTable + " (SELECT * FROM (SELECT "
+ id + " as 'ms_ms_id', mass, (intensity * 100 / (" + maxStr + ")) as 'intensity',
CURRENT_TIMESTAMP FROM ms_ms_data where ms_ms_id = " + msmsid + " AND mass between "
+ lmass + " AND " + hmass + ") m WHERE intensity > " + intensity + ")";
        } else if (tmpTable.compareTo("malditemp2") == 0) {
            templInStr = "INSERT INTO " + tmpTable + " (SELECT * FROM (SELECT "
+ id + " as 'ms_ms_id', mass, (intensity * 100 / (" + maxStr + ")) as 'intensity',
CURRENT_TIMESTAMP FROM ms_ms_data where ms_ms_id = " + msmsid + " AND mass between "
+ lmass + " AND " + hmass + ") m WHERE intensity > " + intensity + ")";
        }
        stmt.executeUpdate(templInStr);
        stmt.close();
    }
}

```



```

    } catch (SQLException e) {
        return 0;
    } catch (Exception e) {
        return 0;
    }
    return id;
}

public int executeString(String myString) {
    try {
        stmt = myConn.createStatement();
        stmt.executeUpdate(myString);
        stmt.close();
    } catch (SQLException e) {
        return 1;
    } catch (Exception e) {
        return 2;
    }
    return 0;
}

public int findAndReplace(int lRange, int hRange, String strReplace, String
strWith, int ident, String autoIncStr, int inStart, int genus, int species, int
subspecies, int other, String rdate) {
    //Find and replace searches through a list of identifier numbers and
replaces occurrences of the strReplace string with the strWith String
    //This function also supports comma delimited lists,
    try {
        int i=0;
        String selStr = "SELECT * FROM sample WHERE number BETWEEN " + lRange +
" AND " + hRange + " AND notes LIKE '%" + strReplace + "%' and identifier_id = " +
ident + " ORDER BY number ASC";
        java.sql.ResultSet myRS = getRS(selStr);

        String[] replLst = strWith.split(",");
        int[] smplids = new int[replLst.length];

        //if replLst onoly has one item then there is no list
        if (replLst.length == 1 || replLst == null) {
            while (myRS.next()) {
                String getRepl = myRS.getString("notes");
                smplids[0] = myRS.getInt("sample_id");
                String mRepl = strWith;
                if (autoIncStr != "" && autoIncStr != null) {
                    mRepl = mRepl.replace(autoIncStr, Integer.toString((i +
inStart)));
                }
                int mid;
                getRepl = getRepl.replace(strReplace, mRepl);
                mid = myRS.getInt("sample_id");
                //myRS.close();
                stmt = myConn.createStatement();
                if (rdate != "") {
                    stmt.executeUpdate("UPDATE sample set notes = '" + getRepl +
"', date = '" + rdate + "' WHERE sample_id = " + mid);
                } else {
                    stmt.executeUpdate("UPDATE sample set notes = '" + getRepl +
"' WHERE sample_id = " + mid);
                }
                i++;
                //myRS = getRS(selStr);
            }
        } else {
            //There is a list loop through it, if there are more items to
replace then there are items in the list thten use the last item in the list

```

```

        int z = 0;
        while (myRS.next()) {
            String getRepl = myRS.getString("notes");
            int mid;
            String mRepl = replLst[z];

            if (autoIncStr != "" && autoIncStr != null) {
                mRepl = mRepl.replace(autoIncStr, Integer.toString((i +
inStart)));
            }

            getRepl = getRepl.replace(strReplace, mRepl);
            mid = myRS.getInt("sample_id");
            smplids[z] = mid;
            //myRS.close();
            stmt = myConn.createStatement();
            if (rdate != "") {
                stmt.executeUpdate("UPDATE sample set notes = '" + getRepl +
'', date = '" + rdate + "' WHERE sample_id = " + mid);
            } else {
                stmt.executeUpdate("UPDATE sample set notes = '" + getRepl +
'' WHERE sample_id = " + mid);
            }
            i++;
            //Increase z until it hits the last item in the array.
            if (z < replLst.length - 1) {
                z++;
            }
        }
    }
    myRS.close();

```

//Check and see if the strain needs to be updated. These will all be zero if no update

```

        if ((genus + species + subspecies + other) > 0) {
            for(int z2=0; z2 < smplids.length; z2++) {
                String getStrfromSmpl = "SELECT s.genus_id , s.species_id
,s.subspecies_id, s.subtype_id FROM sample sm, strain s WHERE s.strain_id =
sm.strain_id AND sm.sample_id = " + smplids[z2];
                myRS = getRS(getStrfromSmpl);
                myRS.next();
                if (genus == 0) {
                    genus = myRS.getInt("genus_id");
                }
                if (species == 0) {
                    species = myRS.getInt("species_id");
                }
                if(subspecies == 0) {
                    subspecies = myRS.getInt("subspecies_id");
                }
                if(other == 0) {
                    other = myRS.getInt("subtype_id");
                }

                int strainID;
                //Next we need to get the strain id
                String slctStrain = "SELECT strain_id FROM strain WHERE
genus_id=" + genus + " AND species_id=" + species + " AND subspecies_id=" +
subspecies + " AND subtype_id=" + other;

                myRS = getRS(slctStrain);

                if (myRS.next()) {

```

```

        //the strain already exists, use this id
        strainID = myRS.getInt("strain_id");
    } else {
        //the strain does not exist, make a new one and use the new
        id
        myRS.close();
        //Insert the strain ID, now get it the same way as above
        String insrtStrain;
        insrtStrain = "INSERT INTO strain (genus_id, species_id,
subspecies_id, subtype_id) VALUES(" + genus + "," + species + "," + subspecies + "," +
+ other + ")";
        stmt.executeUpdate(insrtStrain);
        stmt.close();

        myRS = getRS(slctStrain);

        myRS.next();
        strainID = myRS.getInt("strain_id");
    }

    myRS.close();

    stmt = myConn.createStatement();
    stmt.executeUpdate("UPDATE sample SET strain_id = " + strainID +
" WHERE sample_id = " + smplids[z2]);
    }
}

    return i;
} catch (SQLException e) {
    return 0;
} catch (Exception e) {
    return 0;
}
}

    public int findAndReplaceAppend(int lRange, int hRange, String strWith, int
ident, String autoIncStr, int inStart, int append, int genus, int species, int
subspecies, int other, String rdate) {
    //Find and replace searches through a list of identifier numbers and
replaces occurrences of the strReplace string with the strWith String
    //This function also supports comma delimited lists,
    try {
        int i=0;
        int[] smplids = new int[hRange - lRange + 1];

        String selStr = "SELECT * FROM sample WHERE number BETWEEN " + lRange +
" AND " + hRange + " AND identifier_id = " + ident + " ORDER BY number ASC";
        java.sql.ResultSet myRS = getRS(selStr);

        String[] replLst = strWith.split(",");

        //if replLst onoly has one item then there is no list
        if (replLst.length == 1 || replLst == null) {
            while (myRS.next()) {
                String getRepl = myRS.getString("notes");
                smplids[i] = myRS.getInt("sample_id");
                String mRepl = strWith;

                if (autoIncStr != "" && autoIncStr != null) {
                    mRepl = mRepl.replace(autoIncStr, Integer.toString((i +
inStart)));
                }
            }
        }
    }
}

```

```

        int mid;
        mid = myRS.getInt("sample_id");

        if (append == 0) {
            getRepl += mRepl;
        } else {
            getRepl = mRepl + getRepl;
        }

        stmt = myConn.createStatement();
        if (rdate != "") {
            stmt.executeUpdate("UPDATE sample set notes = '" + getRepl +
            "'", date = "'" + rdate + "'" WHERE sample_id = " + mid);
        } else {
            stmt.executeUpdate("UPDATE sample set notes = '" + getRepl +
            "'" WHERE sample_id = " + mid);
        }
        i++;
    }
}
myRS.close();

//Check and see if the strain needs to be updated. These will all be
zero if no update

    if ((genus + species + subspecies + other) > 0) {
        for(int z=0; z < smplids.length; z++) {
            String getStrfromSmpl = "SELECT s.genus_id , s.species_id
            ,s.subspecies_id, s.subtype_id FROM sample sm, strain s WHERE s.strain_id =
            sm.strain_id AND sm.sample_id = " + smplids[z];
            myRS = getRS(getStrfromSmpl);
            myRS.next();
            if (genus == 0) {
                genus = myRS.getInt("genus_id");
            }
            if (species == 0) {
                species = myRS.getInt("species_id");
            }
            if(subspecies == 0) {
                subspecies = myRS.getInt("subspecies_id");
            }
            if(other == 0) {
                other = myRS.getInt("subtype_id");
            }

            int strainID;
            //Next we need to get the strain id
            String slctStrain = "SELECT strain_id FROM strain WHERE
            genus_id=" + genus + " AND species_id=" + species + " AND subspecies_id=" +
            subspecies + " AND subtype_id=" + other;

            myRS = getRS(slctStrain);

            if (myRS.next()) {
                //the strain already exists, use this id
                strainID = myRS.getInt("strain_id");
            } else {
                //the strain does not exist, make a new one and use the new
                id

                myRS.close();
                //Insert the strain ID, now get it the same way as above
                String insrtStrain;
                insrtStrain = "INSERT INTO strain (genus_id, species_id,
                subspecies_id, subtype_id) VALUES(" + genus + "," + species + "," + subspecies + "," +
                + other + ")";
            }
        }
    }

```

```

        stmt.executeUpdate(insrtStrain);
        stmt.close();

        myRS = getRS(slctStrain);

        myRS.next();
        strainID = myRS.getInt("strain_id");
    }

    myRS.close();

    stmt = myConn.createStatement();
    stmt.executeUpdate("UPDATE sample SET strain_id = " + strainID +
" WHERE sample_id = " + smplids[z]);
    }
}

return i;
} catch (SQLException e) {
    return 0;
} catch (Exception e) {
    return 0;
}
}

public ArrayList getAllMaldiDataID(String andor, String ident, String rmLo,
String rmHi, String Genus, String Species, String Subspecies, String Subtype, String
notes, String History, String mdate) {
    ArrayList maldiId = new ArrayList();
    //First connect to the database
    dbConnect();
    //Now get all the maldi_tof_ids that are in the maldi_tof_data table
    String selStr;
    if (mdate != "") {
        selStr = "SELECT DISTINCT a.maldi_tof_id FROM maldi_tof_data a,
maldi_tof b WHERE a.maldi_tof_id = b.maldi_tof_id AND b.date >= '" + mdate + "' AND
b.sample_id in (";
    } else {
        selStr = "SELECT DISTINCT a.maldi_tof_id FROM maldi_tof_data a,
maldi_tof b WHERE a.maldi_tof_id = b.maldi_tof_id AND b.sample_id in (";
    }
    if (andor.compareTo("and") == 0) {
        selStr += runANDSearchStr("", ident, Genus, Species, Subspecies,
Subtype, notes, History, 1, "", 500, "", rmLo, rmHi);
    } else {
        selStr += runORSearchStr("", ident, Genus, Species, Subspecies, Subtype,
notes, History, 1, rmLo, rmHi);
    }
    selStr += ")";
    ResultSet myRS = getRS(selStr);
    try {
        while (myRS.next()) {
            maldiId.add(new Integer(myRS.getInt("maldi_tof_id")));
        }
        //close the result set and connection
        myRS.close();
    } catch (SQLException e) {
        return null;
    } catch (Exception e) {
        return null;
    }
}

```

```

        dbClose();
        //Return the array list
        return maldiId;
    }

    public ArrayList getAllInSilicoDataID(String andor, String ident, String Genus,
String Species, String Subspecies, String Subtype, String protein, String notes,
String mass, String masserror) {
        ArrayList insilicoid = new ArrayList();
        //First connect to the database
        dbConnect();
        //Now get all the maldi_tof_ids that are in the maldi_tof_data table
        String selStr;

        if (andor.compareTo("and") == 0) {
            selStr = runANDInSilicoSearchStr(ident, Genus, Species, Subspecies,
Subtype, protein, notes, mass, masserror);
        } else {
            selStr = runORInSilicoSearchStr(ident, Genus, Species, Subspecies,
Subtype, protein, notes, mass, masserror);
        }
        ResultSet myRS = getRS(selStr);
        try {
            while (myRS.next()) {
                insilicoid.add(new Integer(myRS.getInt("in_silico_id")));
            }
            //close the result set and connection
            myRS.close();

        } catch (SQLException e) {
            return null;
        } catch (Exception e) {
            return null;
        }

        dbClose();
        //Return the array list
        return insilicoid;
    }

    public ArrayList getAllMSMSDataID(String andor, String ident, String Genus,
String Species, String Subspecies, String Subtype, String protein, String notes,
String mass, String masserror) {
        ArrayList msmsid = new ArrayList();
        //First connect to the database
        dbConnect();
        //Now get all the maldi_tof_ids that are in the maldi_tof_data table
        String selStr;

        if (andor.compareTo("and") == 0) {
            selStr = runANDMSMSSearchStr(ident, Genus, Species, Subspecies, Subtype,
protein, notes, mass, masserror);
        } else {
            selStr = runORMSMSSearchStr(ident, Genus, Species, Subspecies, Subtype,
protein, notes, mass, masserror);
        }
        ResultSet myRS = getRS(selStr);
        try {
            while (myRS.next()) {
                msmsid.add(new Integer(myRS.getInt("ms_ms_id")));
            }
            //close the result set and connection
            myRS.close();
        }
    }

```

```

    } catch (SQLException e) {
        return null;
    } catch (Exception e) {
        return null;
    }

    dbClose();
    //Return the array list
    return msmsid;
}

public ArrayList getMALDILISTwithSQLSel(String mdate, String selStmt) {
    ArrayList maldiId = new ArrayList();
    //First connect to the database
    dbConnect();
    //Now get all the maldi_tof_ids that are in the maldi_tof_data table
    String selStr;
    if (mdate != "") {
        selStr = "SELECT DISTINCT a.maldi_tof_id FROM maldi_tof_data a,
mal-di-tof b WHERE a.maldi_tof_id = b.maldi_tof_id AND b.date >= '" + mdate + "' AND
b.sample_id in ("";
    } else {
        selStr = "SELECT DISTINCT a.maldi_tof_id FROM maldi_tof_data a,
mal-di-tof b WHERE a.maldi_tof_id = b.maldi_tof_id AND b.sample_id in ("";
    }
    selStr += selStmt;
    selStr += ")";
    ResultSet myRS = getRS(selStr);
    try {
        while (myRS.next()) {
            maldiId.add(new Integer(myRS.getInt("mal-di-tof_id")));
        }
        //close the result set and connection
        myRS.close();
    } catch (SQLException e) {
        return null;
    } catch (Exception e) {
        return null;
    }

    dbClose();
    //Return the array list
    return maldiId;
}

public java.sql.ResultSet getGenus() {
    return getRS("SELECT * FROM genus ORDER BY name");
}

public java.sql.ResultSet getIdentPrefix() {
    return getRS("SELECT * FROM identifier");
}

public int getLastUsedStrainNumber() {
    try {
        java.sql.ResultSet myRS = getRS("SELECT MAX(number) FROM SAMPLE");
        myRS.next();
        return myRS.getInt(1);
    } catch (SQLException e) {
        return -1;
    } catch (Exception e) {
        return -1;
    }
}

```

```

    }
}

public java.sql.ResultSet getMaldi(String sampleid) {
    String mQuery;
    mQuery = "SELECT * FROM maldi_tof WHERE sample_id = ";
    mQuery = mQuery.concat(sampleid);
    return getRS(mQuery);
}

public java.sql.ResultSet getMaldiAndSampleInfo(String maldiid) {
    String mQuery;
    mQuery = "SELECT CONCAT(identifier.ident, sample.number), maldi_tof.date,
    maldi_tof.target_spot_number, concat_ws(' ', genus.name, species.name,
    subspecies.name, subtype.name), sample.sample_id, sample.notes FROM
    identifier, sample, maldi_tof, species, strain, genus, subspecies, subtype WHERE
    maldi_tof.sample_id = sample.sample_id AND strain.genus_id = genus.genus_id AND
    strain.species_id = species.species_id AND subspecies.subspecies_id =
    strain.subspecies_id AND subtype.subtype_id = strain.subtype_id AND
    identifier.identifier_id = sample.identifier_id AND sample.strain_id =
    strain.strain_id AND maldi_tof.maldi_tof_id = ";
    mQuery = mQuery.concat(maldiid);
    return getRS(mQuery);
}

public java.sql.ResultSet getInSilicoAndSampleInfo(String insilicoid) {
    String mQuery;
    mQuery = "SELECT in_silico.identification, in_silico.date, concat_ws(' ',
    genus.name, species.name, subspecies.name, subtype.name), in_silico.protein,
    in_silico.notes, in_silico.average_mw FROM in_silico, species, strain, genus,
    subspecies, subtype WHERE strain.genus_id = genus.genus_id AND strain.species_id =
    species.species_id AND subspecies.subspecies_id = strain.subspecies_id AND
    subtype.subtype_id = strain.subtype_id AND in_silico.strain_id = strain.strain_id
    AND in_silico.in_silico_id = ";
    mQuery = mQuery.concat(insilicoid);
    return getRS(mQuery);
}

public java.sql.ResultSet getMSMSAndSampleInfo(String msmsid) {
    String mQuery;
    mQuery = "SELECT CONCAT(identifier.ident, sample.number), ms_ms.date,
    ms_ms.protein, concat_ws(' ', genus.name, species.name, subspecies.name,
    subtype.name), sample.sample_id, sample.notes, ms_ms.precursor_ion_mtoz FROM
    identifier, sample, ms_ms, species, strain, genus, subspecies, subtype WHERE
    ms_ms.sample_id = sample.sample_id AND strain.genus_id = genus.genus_id AND
    strain.species_id = species.species_id AND subspecies.subspecies_id =
    strain.subspecies_id AND subtype.subtype_id = strain.subtype_id AND
    identifier.identifier_id = sample.identifier_id AND sample.strain_id =
    strain.strain_id AND ms_ms.ms_ms_id = ";
    mQuery = mQuery.concat(msmsid);
    return getRS(mQuery);
}

public java.sql.ResultSet getINSILICOAndSampleInfo(String msmsid) {
    String mQuery;
    mQuery = "SELECT in_silico.identification, in_silico.date,
    in_silico.protein, concat_ws(' ', genus.name, species.name, subspecies.name,
    subtype.name), in_silico.in_silico_id, in_silico.notes, in_silico.aa_seq FROM
    in_silico, species, strain, genus, subspecies, subtype WHERE strain.genus_id =
    genus.genus_id AND strain.species_id = species.species_id AND
    subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
    strain.subtype_id AND in_silico.strain_id = strain.strain_id AND
    in_silico.in_silico_id = ";
    mQuery = mQuery.concat(msmsid);
    return getRS(mQuery);
}

```



```

    }

    public String[] getMaldiSampleInfo(String maldiid) {
        String mQuery;
        mQuery = "SELECT CONCAT(identifier.ident, sample.number), maldi_tof.date,
        maldi_tof.target_spot_number, concat_ws(' ', genus.name, species.name,
        subspecies.name, subtype.name), sample.sample_id, maldi_tof.notes FROM
        identifier, sample, maldi_tof, species, strain, genus, subspecies, subtype WHERE
        maldi_tof.sample_id = sample.sample_id AND strain.genus_id = genus.genus_id AND
        strain.species_id = species.species_id AND subspecies.subspecies_id =
        strain.subspecies_id AND subtype.subtype_id = strain.subtype_id AND
        identifier.identifier_id = sample.identifier_id AND sample.strain_id =
        strain.strain_id AND maldi_tof.maldi_tof_id = ";
        mQuery = mQuery.concat(maldiid);
        java.sql.ResultSet myRS = getRS(mQuery);
        String[] rmNum = new String[3];
        try {
            myRS.next();
            rmNum[0] = myRS.getString(1);
            rmNum[1] = myRS.getString(4);
            rmNum[2] = myRS.getString(6);
        } catch (SQLException e) {
            rmNum[0] = "Not Found";
            rmNum[1] = "Not Found";
        }
        return rmNum;
    }

    public String[] getMSMSSampleInfo(String msmsid) {
        String mQuery;
        mQuery = "SELECT CONCAT(identifier.ident, sample.number), ms_ms.date,
        concat_ws(' ', genus.name, species.name, subspecies.name, subtype.name),
        sample.sample_id, ms_ms.notes FROM identifier, sample, ms_ms, species, strain, genus,
        subspecies, subtype WHERE ms_ms.sample_id = sample.sample_id AND strain.genus_id =
        genus.genus_id AND strain.species_id = species.species_id AND
        subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
        strain.subtype_id AND identifier.identifier_id = sample.identifier_id AND
        sample.strain_id = strain.strain_id AND ms_ms.ms_ms_id = ";
        mQuery = mQuery.concat(msmsid);
        java.sql.ResultSet myRS = getRS(mQuery);
        String[] rmNum = new String[3];
        try {
            myRS.next();
            rmNum[0] = myRS.getString(1);
            rmNum[1] = myRS.getString(3);
            rmNum[2] = myRS.getString(5);
        } catch (SQLException e) {
            rmNum[0] = "Not Found";
            rmNum[1] = "Not Found";
            rmNum[2] = "Not Found";
        }
        return rmNum;
    }

    public java.sql.ResultSet getInSilicoErrorData(int id1, int id2, String
    intensity, String masserror, String rangeA, String rangeB, String residue) {

        int maldiTempId = dumpMSMSDataToTemp(id1, Integer.parseInt(intensity),
        Integer.parseInt(rangeA), Integer.parseInt(rangeB), "malditemp");
        int cmpPrToId = dumpINSILICODataToTemp(id2, Integer.parseInt(intensity),
        Integer.parseInt(rangeA), Integer.parseInt(rangeB), "malditemp2", residue);

        String getcmprdata = "SELECT r.insilico, r.x, r.Error, y.residue FROM
        (SELECT c.insilico, c.msms as x, c.insilico - c.msms as Error FROM (SELECT b.mass as
        insilico, '' as msms, '' as intensity FROM maldiTemp2 b WHERE b.intensity >

```

```

$minIntensity$ and b.mass between $lowMass$ and $highMass$ and b.maldi_tof_id =
$cmprTo$ AND b.mass not in (SELECT insilico FROM (select b.mass as insilico, a.mass
as msms from maldiTemp a, maldiTemp2 b where a.intensity > $minIntensity$ AND
b.intensity > $minIntensity$ AND a.mass between (b.mass - $error$) and (b.mass +
$error$) and a.mass between $lowMass$ and $highMass$ and b.mass between $lowMass$
and $highMass$ and a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$) t) UNION
select b.mass as insilico, a.mass as msms, a.intensity as intensity from maldiTemp
a, maldiTemp2 b where a.intensity > $minIntensity$ AND b.intensity > $minIntensity$
AND a.mass between (b.mass - $error$) and (b.mass + $error$) and a.mass between
$lowMass$ and $highMass$ and b.mass between $lowMass$ and $highMass$ and
a.maldi_tof_id = $cmpr$ and b.maldi_tof_id = $cmprTo$ order by insilico) as c where
c.msms != '') r, in_silico_data y WHERE r.insilico = y.mass AND y.in_silico_id =
$id2$";

```

```

getcmprdata = getcmprdata.replace("$minIntensity$", intensity);
getcmprdata = getcmprdata.replace("$error$", masserror);
getcmprdata = getcmprdata.replace("$lowMass$", rangeA);
getcmprdata = getcmprdata.replace("$highMass$", rangeB);
getcmprdata = getcmprdata.replace("$cmpr$", Integer.toString(maldiTempId));
getcmprdata = getcmprdata.replace("$cmprTo$", Integer.toString(cmprToId));
getcmprdata = getcmprdata.replace("$id2$", Integer.toString(id2));

```

```

return getRS(getcmprdata);

```

```

}

```

```

public java.sql.ResultSet getInSilicoMSMSMatchedData(int id1, int id2, String
intensity, String masserror, String rangeA, String rangeB, String residue) {

```

```

    int maldiTempId = dumpMSMSDataToTemp(id1, Integer.parseInt(intensity),
Integer.parseInt(rangeA), Integer.parseInt(rangeB), "maldiTemp");
    int cmprToId = dumpINSILICODataToTemp(id2, Integer.parseInt(intensity),
Integer.parseInt(rangeA), Integer.parseInt(rangeB), "maldiTemp2", residue);

```

```

    String getcmprdata = "SELECT Round(r.insilico,1) as insilico,
ROUND(r.msms,1) as msms, Round(r.intensity,1) as intensity, y.iontype, y.residue
FROM (SELECT b.mass as insilico, '' as msms, '' as intensity FROM maldiTemp2 b WHERE
b.intensity > $minIntensity$ and b.mass between $lowMass$ and $highMass$ and
b.maldi_tof_id = $cmprTo$ AND b.mass not in (SELECT insilico FROM (select b.mass as
insilico, a.mass as msms from maldiTemp a, maldiTemp2 b where a.intensity >
$minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between (b.mass -
$error$) and (b.mass + $error$) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) t) UNION select b.mass as insilico, a.mass as msms,
a.intensity as intensity from maldiTemp a, maldiTemp2 b where a.intensity >
$minIntensity$ AND b.intensity > $minIntensity$ AND a.mass between (b.mass -
$error$) and (b.mass + $error$) and a.mass between $lowMass$ and $highMass$ and
b.mass between $lowMass$ and $highMass$ and a.maldi_tof_id = $cmpr$ and
b.maldi_tof_id = $cmprTo$) r, in_silico_data y WHERE r.insilico = y.mass AND
y.in_silico_id = $id2$ ORDER BY intensity DESC, msms DESC";

```

```

    getcmprdata = getcmprdata.replace("$minIntensity$", intensity);
    getcmprdata = getcmprdata.replace("$error$", masserror);
    getcmprdata = getcmprdata.replace("$lowMass$", rangeA);
    getcmprdata = getcmprdata.replace("$highMass$", rangeB);
    getcmprdata = getcmprdata.replace("$cmpr$", Integer.toString(maldiTempId));
    getcmprdata = getcmprdata.replace("$cmprTo$", Integer.toString(cmprToId));
    getcmprdata = getcmprdata.replace("$id2$", Integer.toString(id2));

```

```

    return getRS(getcmprdata);

```

```

}

```

```

public String[] getInSilicoSampleInfo(String insilicoid) {

```

```

    String mQuery;
    mQuery = "SELECT in_silico.identification, in_silico.date, concat_ws(' ',

```

```

genus.name, species.name, subspecies.name, subtype.name), in_silico.notes FROM
identifier, sample, in_silico, species, strain, genus, subspecies, subtype WHERE
strain.genus_id = genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND identifier.identifier_id = sample.identifier_id AND
in_silico.strain_id = strain.strain_id AND in_silico.in_silico_id = ";
    mQuery = mQuery.concat(insilicoid);
    java.sql.ResultSet myRS = getRS(mQuery);
    String[] rmNum = new String[3];
    try {
        myRS.next();
        rmNum[0] = myRS.getString(1);
        rmNum[1] = myRS.getString(3);
        rmNum[2] = myRS.getString(4);
    } catch (SQLException e) {
        rmNum[0] = "Not Found";
        rmNum[1] = "Not Found";
        rmNum[2] = "Not Found";
    }
    return rmNum;
}

public java.sql.ResultSet getMaldiData(String maldiid){
    String mdQuery = "SELECT mass, intensity FROM MALDI_TOF_DATA WHERE
maldi_tof_id = ";
    mdQuery = mdQuery.concat(maldiid);
    return getRS(mdQuery);
}

public java.sql.ResultSet getMSMS(String sampleid) {
    String mQuery;
    mQuery = "SELECT * FROM ms_ms WHERE sample_id = ";
    mQuery = mQuery.concat(sampleid);
    return getRS(mQuery);
}

public java.sql.ResultSet getMSMSData(String msmsid){
    String mdQuery = "SELECT mass, intensity FROM ms_ms_data WHERE ms_ms_id = ";
    mdQuery = mdQuery.concat(msmsid);
    return getRS(mdQuery);
}

public java.sql.ResultSet getINSILICOData(String insilicoid){
    String mdQuery = "SELECT mass, intensity FROM in_silico_data WHERE
in_silico_id = ";
    mdQuery = mdQuery.concat(insilicoid);
    return getRS(mdQuery);
}

public java.sql.ResultSet getTempMaldiData(String maldiid){
    String mdQuery = "SELECT mass, intensity FROM maldiTemp WHERE maldi_tof_id =
";
    mdQuery = mdQuery.concat(maldiid);
    return getRS(mdQuery);
}

public java.sql.ResultSet getMaldiGraphData(String maldiid, int rangeA, int
rangeB){
    String subStr = "SELECT MAX(intensity) FROM maldi_tof_data WHERE
maldi_tof_id = " + maldiid + " AND mass BETWEEN " + rangeA + " AND " + rangeB;
    String mdQuery = "SELECT mass, (intensity * 100 / (" + subStr + ")) as
'intensity' FROM MALDI_TOF_DATA WHERE maldi_tof_id = ";
    mdQuery = mdQuery + maldiid;
    mdQuery = mdQuery.concat(" AND mass BETWEEN '");
    mdQuery = mdQuery + rangeA;

```

```

        mdQuery = mdQuery.concat("' AND '");
        mdQuery = mdQuery + rangeB + "' ORDER BY intensity DESC;";
        return getRS(mdQuery);
    }

    public java.sql.ResultSet getMSMSGGraphData(String msmsid, int rangeA, int
rangeB){
        String subStr = "SELECT MAX(intensity) FROM ms_ms_data WHERE ms_ms_id = " +
msmsid + " AND mass BETWEEN " + rangeA + " AND " + rangeB;
        String mdQuery = "SELECT ROUND(mass, 1) mass, (intensity * 100 / (" + subStr
+ ")) as 'intensity' FROM ms_ms_DATA WHERE ms_ms_id = ";
        mdQuery = mdQuery + msmsid;
        mdQuery = mdQuery.concat(" AND mass BETWEEN '");
        mdQuery = mdQuery + rangeA;
        mdQuery = mdQuery.concat("' AND '");
        mdQuery = mdQuery + rangeB + "' ORDER BY intensity DESC;";
        return getRS(mdQuery);
    }

    public java.sql.ResultSet getINSILICOGraphData(String insilicoid, int rangeA,
int rangeB){
        String mdQuery = "SELECT ROUND(mass, 1) mass, intensity FROM in_silico_DATA
WHERE in_silico_id = ";
        mdQuery = mdQuery + insilicoid;
        mdQuery = mdQuery.concat(" AND mass BETWEEN '");
        mdQuery = mdQuery + rangeA;
        mdQuery = mdQuery.concat("' AND '");
        mdQuery = mdQuery + rangeB + "'";
        return getRS(mdQuery);
    }

    public java.sql.ResultSet getINSILICOGraphData(String insilicoid, int rangeA,
int rangeB, String residue){
        String mdQuery = "SELECT ROUND(mass, 1) mass, intensity FROM in_silico_DATA
WHERE in_silico_id = ";
        mdQuery = mdQuery + insilicoid;
        mdQuery = mdQuery.concat(" AND mass BETWEEN '");
        mdQuery = mdQuery + rangeA;
        mdQuery = mdQuery.concat("' AND '");
        mdQuery = mdQuery + rangeB + "'";

        if (residue.length() > 0) {
            mdQuery += " AND (";
            for(int i = 0; i < residue.length(); i++) {
                mdQuery += "residue LIKE '%" + residue.charAt(i) + "%'";
                if(i != (residue.length() - 1)) {
                    mdQuery += " OR ";
                }
            }
            mdQuery += ")";
        }

        return getRS(mdQuery);
    }

    public java.sql.ResultSet getMaldiGraphSubtractionData(String maldiid1, String
malidiid2, int rangeA, int rangeB, int error) {
        String subStr = "SELECT MAX(intensity) FROM maldi_tof_data WHERE
maldi_tof_id = " + maldiid1 + " AND mass BETWEEN " + rangeA + " AND " + rangeB;
        String sQuery = "SELECT mass, (intensity * 100 / (" + subStr + ")) as
'intensity' FROM maldi_tof_data where maldi_tof_id = $id1$ and mass not in (select
a.mass mass from maldi_tof_data a, maldi_tof_data b where a.mass between (b.mass -
(b.mass * ($error$))) and (b.mass + (b.mass * ($error$))) and a.mass between
$rangeA$ and $rangeB$ and b.mass between $rangeA$ and $rangeB$ and a.maldi_tof_id
= $id1$ and b.maldi_tof_id = $id2$) and mass between $rangeA$ and $rangeB$ ORDER by
intensity DESC";

```

```

        sQuery = sQuery.replace("$id1$", maldiid1);
        sQuery = sQuery.replace("$id2$", maldiid2);
        sQuery = sQuery.replace("$rangeA$", Integer.toString(rangeA));
        sQuery = sQuery.replace("$rangeB$", Integer.toString(rangeB));
        double perror = error/1000000.0;
        sQuery = sQuery.replace("$error$", Double.toString(perror));
        return getRS(sQuery);
    }

    public java.sql.ResultSet getMSMSGGraphSubtractionData(String msmsid1, String
msmsid2, int rangeA, int rangeB, String error) {
        String subStr = "SELECT MAX(intensity) FROM ms_ms_data WHERE ms_ms_id = " +
msmsid1 + " AND mass BETWEEN " + rangeA + " AND " + rangeB;
        String sQuery = "SELECT mass, (intensity * 100 / (" + subStr + ")) as
'intensity' FROM ms_ms_data where ms_ms_id = $id1$ and mass not in (select a.mass
mass from ms_ms_data a, ms_ms_data b where a.mass between (b.mass - $error$) and
(b.mass + $error$) and a.mass between $rangeA$ and $rangeB$ and b.mass between
$rangeA$ and $rangeB$ and a.ms_ms_id = $id1$ and b.ms_ms_id = $id2$) and mass
between $rangeA$ and $rangeB$ ORDER by intensity DESC";
        sQuery = sQuery.replace("$id1$", msmsid1);
        sQuery = sQuery.replace("$id2$", msmsid2);
        sQuery = sQuery.replace("$rangeA$", Integer.toString(rangeA));
        sQuery = sQuery.replace("$rangeB$", Integer.toString(rangeB));
        sQuery = sQuery.replace("$error$", error);
        return getRS(sQuery);
    }

    public java.sql.ResultSet getTempMaldiGraphData(String maldiid, int intensityA,
int intensityB){
        String subStr = "SELECT MAX(intensity) FROM maldiTemp WHERE maldi_tof_id = "
+ maldiid + " AND mass BETWEEN " + intensityA + " AND " + intensityB;
        String mdQuery = "SELECT mass, (intensity * 100 / (" + subStr + ")) as
'intensity' FROM maldiTemp WHERE maldi_tof_id = ";
        mdQuery = mdQuery + maldiid;
        mdQuery = mdQuery.concat(" AND mass BETWEEN '");
        mdQuery = mdQuery + intensityA;
        mdQuery = mdQuery.concat("' AND '");
        mdQuery = mdQuery + intensityB + "' ORDER BY intensity DESC;";
        return getRS(mdQuery);
    }

    public int getLocationID(String location) {
        String selStr = "SELECT location_id FROM location WHERE name = '" + location
+ "'";
        if (location.compareTo("") == 0) {
            selStr = "SELECT location_id FROM location WHERE name is null";
        }

        int id;
        try {
            ResultSet myRS = getRS(selStr);
            if (myRS != null && myRS.next()) {
                id = myRS.getInt("location_id");
                myRS.close();
                stmt.close();
                return id;
            } else {
                if (myRS != null) {myRS.close();}
                if (location.compareTo("") == 0) {
                    executeString("INSERT INTO location(name) VALUES(null)");
                } else {
                    executeString("INSERT INTO location(name) VALUES('" + location
+ "')");
                }
            }
        }
    }

```

```

        myRS = getRS(selStr);
        myRS.next();
        id = myRS.getInt("location_id");
        myRS.close();
        stmt.close();
        return id;
    }

    } catch (SQLException e) {
        return 0;
    }
}

public String getPassword() {
    return password;
}

public java.sql.ResultSet getRS(String myStmt) {
    //Handles all necessary code for creating a record set
    java.sql.ResultSet myRS;
    myRS = null;
    //Exit out if there is no connection
    if (myConn == null) {return null;}

    try {
        stmt = myConn.createStatement(ResultSet.FETCH_FORWARD,
ResultSet.CONCUR_UPDATABLE);
        myRS = stmt.executeQuery(myStmt);

    }catch (SQLException e) {
        e.printStackTrace();
    }catch (Exception e) {
        e.printStackTrace();
    }finally {
        if (myRS!=null) {
            return myRS;
        }else {
            return null;
        }
    }
}

public int getSampleIDFromIdent(int num, int ident) {
    try {
        java.sql.ResultSet myRS = getRS("SELECT sample_id FROM sample WHERE
number = " + num + " AND identifier_id = " + ident);
        myRS.next();
        return myRS.getInt("sample_id");
    } catch (SQLException e) {
        return 0;
    } catch (Exception e) {
        return 0;
    }
}

public java.sql.ResultSet getSpecies() {
    return getRS("SELECT * FROM species ORDER BY name");
}

public java.sql.ResultSet getSubSpecies() {
    return getRS("SELECT * FROM subspecies ORDER BY name");
}

public java.sql.ResultSet getSubType() {

```

```

        return getRS("SELECT * FROM subtype ORDER BY name");
    }

    public String getUsername() {
        return username;
    }

    public int insertMSMS (String identNum, String mdate, String precionmass,
String notes, String protein, ArrayList mz, ArrayList intensity) {

        //Fist check that the identification number exists
        String slctStr = "SELECT sample.sample_id, CONCAT(identifier.ident,
sample.number) FROM identifier, sample WHERE identifier.identifier_id =
sample.identifier_id AND CONCAT(identifier.ident, sample.number) = '" + identNum
+ "'";
        int sampleid, msmsid;
        ResultSet myRS = getRS(slctStr);
        try {
            if (myRS.next()) {
                sampleid = myRS.getInt("sample_id");
            } else {
                return 1;
            }
        }
        myRS.close();

        String insStr = "INSERT INTO ms_ms(date, sample_id, notes,
precursor_ion_mtoz, protein) VALUES('" + mdate + "','" + sampleid + "','" + notes +
"', '" + precionmass + "','" + protein + "')";
        stmt.executeUpdate(insStr, java.sql.Statement.RETURN_GENERATED_KEYS);
        myRS = stmt.getGeneratedKeys();
        myRS.next();
        msmsid = myRS.getInt(1);
        myRS.close();

        for (int i=0; i < mz.size(); i++) {
            String templInStr = "INSERT INTO ms_ms_data (ms_ms_id,
mass,intensity) VALUES(";
            templInStr += msmsid + "," + mz.get(i) + "," + intensity.get(i);
            templInStr += ")";
            stmt.executeUpdate(templInStr);
        }

    } catch (SQLException e) {
        return 9;
    }

    return 0;
}

public int insertLocus(String Locus, String Genus, String Species, String
Subspecies, String Subtype, String Strain, String Location, String Location2, String
Start, String End, String FR, String NT, String AA, String MW, String IEP, String
gene, String code, String Cog, String ec1, String ec2, String product, String notes)
{
    int g = insertNewGenus(Genus);
    int s = insertNewSpecies(Species);
    int sp = insertNewSubspecies(Subspecies);
    int sb = insertNewSubtype(Subtype);

    String slctStrain = "SELECT strain_id FROM strain WHERE genus_id=" + g + "
AND species_id=" + s + " AND subspecies_id=" + sp + " AND subtype_id=" + sb;
    int strainID;
    java.sql.ResultSet myRS;

    try {
        myRS = getRS(slctStrain);
    }

```

```

    if (myRS != null && myRS.next()) {
        //the strain already exists, use this id
        strainID = myRS.getInt("strain_id");
        myRS.close();
        stmt.close();
    } else {
        //the strain does not exist, make a new one and use the new id
        if (myRS != null) {myRS.close();}
        //Insert the strain ID, now get it the same way as above
        String insrtStrain;
        insrtStrain = "INSERT INTO strain (genus_id, species_id,
subspecies_id, subtype_id) VALUES(" + g + "," + s + "," + sp + "," + sb + ")";
        executeString(insrtStrain);

        myRS = getRS(slctStrain);
        myRS.next();
        strainID = myRS.getInt("strain_id");
    }
    myRS.close();
    stmt.close();
} catch (SQLException e) {
    return 1;
}

//Setup for entry into dbase
Cog = Cog.replace("COG", "");
gene = gene.replace("'", "\\");
product = product.replace("'", "\\");
product = product.replace("\\'", "\\");
product = product.replace("\\", "\\");
product = product.replace("\\n", "");
product = product.replace("\\r", "");

if (MW.compareTo("N/A") == 0 || MW == "") {MW= "null";}
if (IEP.compareTo("N/A") == 0 || IEP == "") {IEP= "null";}
if (AA.compareTo("N/A") == 0 || AA == "") {AA= "null";}

//Get the strain number
int l = 0;
String ident, num;
Strain = Strain.replace(" ", "");

//Check and See if the first character is a number or alphabet character
if (isDigit(Strain.substring(0,1))) {
    //The first character is a number, see if there are any Alphabet
    Characters in the string
    boolean alphaIn = false;
    for (int i = 0; i < Strain.length(); i++) {
        if (!isDigit(Strain.substring(i, i+1))) {
            alphaIn = true;
        }
    }

    if (alphaIn) {
        num = "null";
        ident = Strain;
    } else {
        num = Strain;
        ident = "null";
    }
} else {
    while (l < Strain.length() && (! isDigit(Strain.substring(l,l+1)) ||
Strain.substring(l,l+1).compareTo("0") == 0)) {
        l++;
    }
}

```



```

        ident = Strain.substring(0, 1);
        num = Strain.substring(1);
    }

    if (num == "" || num.compareTo("") == 0) {
        num = "null";
    }
    int smpID= 0;
    //now check and see if this sample is in the database, if not inser it
    try {
        String selStr = "SELECT sample_id FROM sample, identifier WHERE
sample.identifier_id = identifier.identifier_id and
CONCAT_WS(' ', identifier.name, sample.number) = '" + Strain + "'";
        myRS = getRS(selStr);
        if (myRS != null && myRS.next()) {
            //it exists
            smpID = myRS.getInt("sample_id");
            myRS.close();
            stmt.close();
        } else {
            //it does not exist
            if (myRS != null ) { myRS.close();stmt.close();}
            //Now we need to see if the identifier name exists
            String selId;
            if (ident.compareTo("null") == 0) {
                selId = "SELECT identifier_id FROM identifier WHERE name is " +
ident;
            } else {
                selId = "SELECT identifier_id FROM identifier WHERE name = '" +
ident + "'";
            }

            myRS = getRS(selId);
            int idID;
            if (myRS != null && myRS.next()) {
                //it exists
                idID = myRS.getInt("identifier_id");
                myRS.close();
            } else {
                //it doesnt exist
                if (myRS != null) {myRS.close();}
                String insidID;
                if (ident.compareTo("null") == 0) {
                    insidID = "INSERT INTO identifier(name) VALUES(" + ident +
");";
                } else {
                    insidID = "INSERT INTO identifier(name) VALUES('" + ident +
"'");
                }

                executeString(insidID);
                myRS = getRS(selId);
                myRS.next();
                idID = myRS.getInt("identifier_id");
                myRS.close();
                stmt.close();
            }
            String insSmp1 = "INSERT INTO SAMPLE(strain_id, identifier_id,
number) VALUES(" + strainID + "," + idID + "," + num + ")";
            executeString(insSmp1);
            myRS = getRS(selStr);
            myRS.next();
            smpID = myRS.getInt("sample_id");
            myRS.close();

```

```

        stmt.close();
    }

} catch (SQLException e) {
    return 1;
}

//the sample is now in the database, insert the locus
String insString;
if (notes == null || notes == "") {
    insString = "INSERT INTO locus (sample_id, name, location_id,
location2_id, start, end, FR, NT, AA,MW, IEP, gene, Code, COG, EC1, EC2, Product) ";
    insString += "VALUES(" + smplID + "," + Locus + "," +
getLocationID(Location) + "," + getLocationID(Location2) + "," + Start + "," + End +
"," + FR + "," + NT + "," + AA + "," + MW + "," + IEP + "," + gene +
","#CODE#,#COG#,#EC1#,#EC2#,'" + product + "'" + ")";
} else {
    insString = "INSERT INTO locus (sample_id, name, location_id,
location2_id, start, end, FR, NT, AA,MW, IEP, gene, Code, COG, EC1, EC2, Product,
notes) ";
    insString += "VALUES(" + smplID + "," + Locus + "," +
getLocationID(Location) + "," + getLocationID(Location2) + "," + Start + "," + End +
"," + FR + "," + NT + "," + AA + "," + MW + "," + IEP + "," + gene +
","#CODE#,#COG#,#EC1#,#EC2#,'" + product + "'," + notes + "'" + ")";
}

if (code.compareTo("-") == 0 || code == "") {
    insString = insString.replace("Code,", "");
    insString = insString.replace("#CODE#", "");
} else {
    insString = insString.replace("#CODE#", "" + code + "");
}

if (ec1.compareTo("-") == 0 || ec1 == "" || ec1.compareTo("") == 0) {
    insString = insString.replace("EC1,", "");
    insString = insString.replace("#EC1#", "");
} else {
    insString = insString.replace("#EC1#", "" + ec1 + "");
}

if (ec2.compareTo("-") == 0 || ec2 == "" || ec2.compareTo("") == 0) {
    insString = insString.replace("EC2,", "");
    insString = insString.replace("#EC2#", "");
} else {
    insString = insString.replace("#EC2#", "" + ec2 + "");
}

if (Cog.compareTo("-") == 0 || Cog == "" || Cog.compareTo("") == 0) {
    insString = insString.replace("COG,", "");
    insString = insString.replace("#COG#", "");
} else {
    insString = insString.replace("#COG#", "" + Cog + "");
}

executeString(insString);

return 0;
}

public int insertMALDI (String identNum, String mdate, String targetnum, String
notes, ArrayList mz, ArrayList intensity) {

    //Fist check that the identification number exists
    String slctStr = "SELECT sample.sample_id, CONCAT(identifier.ident,

```

```

sample.number) FROM identifier, sample WHERE identifier.identifier_id =
sample.identifier_id AND CONCAT(identifier.ident, sample.number) = ' + identNum
+ "'";
    int sampleid, maldiid;
    ResultSet myRS = getRS(slctStr);
    try {
        if (myRS.next()) {
            sampleid = myRS.getInt("sample_id");
        } else {
            return 1;
        }
        myRS.close();

        String insStr = "INSERT INTO MALDI_TOF(sample_id, target_spot_number,
notes,date) VALUES(" + sampleid + ", ' + targetnum + "', ' + notes + "', ' + mdate +
+ "'";

        stmt.executeUpdate(insStr, java.sql.Statement.RETURN_GENERATED_KEYS);
        myRS = stmt.getGeneratedKeys();
        myRS.next();
        maldiid = myRS.getInt(1);
        myRS.close();

        for (int i=0; i < mz.size(); i++) {
            String templInStr = "INSERT INTO maldi_tof_data (malDI_tof_id,
mass,intensity) VALUES(";
            templInStr += maldiid + "," + mz.get(i) + "," + intensity.get(i);
            templInStr += ")";
            stmt.executeUpdate(templInStr);
        }

        //Now make sure this data is normalized and in the maldirat1 table
        //First get the max intensity

        java.sql.ResultSet maxRS = getRS("SELECT max(intensity) as 'intensity'
FROM maldi_tof_data WHERE maldi_tof_id = " + maldiid + " AND mass between 4000 AND
14000");
        maxRS.next();
        int maxInt = maxRS.getInt("intensity");
        maxRS.close();
        //now update the intenisties
        stmt = myConn.createStatement();
        String updStr = "insert into maldirat1 (SELECT maldi_tof_id, mass,
(intensity * 100 / " + maxInt + ") FROM maldi_tof_data WHERE mass between 4000 and
14000 and maldi_tof_id = " + maldiid + ")";
        stmt.executeUpdate(updStr);

    } catch (SQLException e) {
        return 9;
    }

    //No MALDIID will be 9 when inserted b/c it already exists
    return maldiid;
}

public int insertINSILICO (String identNum, String Genus, String Species, String
subspecies, String subtype, String mw, String protein, String notes, ArrayList mz) {

    //Fist check that the identification number exists
    String strainname = Genus + " " + Species + " " + subspecies + " " +
subtype;
    strainname = strainname.trim();
    String slctStr = "SELECT strain_id FROM strain WHERE genus_id=" + Genus + "
AND species_id=" + Species + " AND subspecies_id=" + subspecies + " AND subtype_id="
+ subtype;
    int strainid, msmsid;

```

```

ResultSet myRS = getRS(slctStr);
try {
    if (myRS.next()) {
        strainid = myRS.getInt("strain_id");
    } else {
        //the strain does not exist, make a new one and use the new id
        myRS.close();
        //Insert the strain ID, now get it the same way as above
        String insrtStrain;
        insrtStrain = "INSERT INTO strain (genus_id, species_id,
subspecies_id, subtype_id) VALUES(" + Genus + "," + Species + "," + subspecies + ","
+ subtype + ")";
        stmt.executeUpdate(insrtStrain);
        stmt.close();

        myRS = getRS(slctStr);

        myRS.next();
        strainid = myRS.getInt("strain_id");
    }
    myRS.close();

    String insStr = "INSERT INTO in_silico(date, identification, strain_id,
average_mw, protein, notes) VALUES(CURDATE(),'" + identNum + "','" + strainid + "','" +
mw + "','" + protein + "','" + notes + "')";
    stmt.executeUpdate(insStr, java.sql.Statement.RETURN_GENERATED_KEYS);
    myRS = stmt.getGeneratedKeys();
    myRS.next();
    msmsid = myRS.getInt(1);
    myRS.close();

    for (int i=0; i < mz.size(); i++) {
        String templnStr = "INSERT INTO in_silico_data (in_silico_id,
mass,intensity) VALUES(";
        templnStr += msmsid + "," + mz.get(i) + ",100)";
        stmt.executeUpdate(templnStr);
    }

} catch (SQLException e) {
    return 9;
}

return 0;
}

public int insertTempMALDI (ArrayList mz, ArrayList intensity, int lmass, int
hmass) {
    int sampleid, maldiid;
    int id =0;
    try {
        //FIRST cleanup the table, delete anything that is more than a day old
        stmt = myConn.createStatement();
        stmt.executeUpdate("DELETE FROM malditemp WHERE createTime <
DATE_ADD(CURRENT_TIMESTAMP, INTERVAL -30 MINUTE)");
        //set then get the last ID used
        stmt = myConn.createStatement();
        stmt.executeUpdate("UPDATE lastidused set id = id + 1 WHERE description
= 'maldi'");
        java.sql.ResultSet myRS = getRS("SELECT id FROM lastidused WHERE
description = 'maldi'");
        myRS.next();
        id = myRS.getInt("id");
        myRS.close();
    }
}

```

```

        for (int i=0; i < mz.size(); i++) {
            stmt = myConn.createStatement();
            String templInStr = "INSERT INTO maldiTemp (malDI_tof_id,
mass,intensity, createdTime) VALUES(";
            templInStr += Integer.toString(id) + "," + mz.get(i) + "," +
intensity.get(i);
            templInStr += ", CURRENT_TIMESTAMP)";
            stmt.executeUpdate(templInStr);
            stmt.close();
        }
        //Now make sure this data is normalized
        //First get the max intensity
        java.sql.ResultSet maxRS = getRS("SELECT max(intensity) FROM maldiTemp
WHERE malDI_tof_id = " + id + " AND mass between " + lmass + " AND " + hmass);
        maxRS.next();
        int maxInt = maxRS.getInt(1);
        maxRS.close();
        //now update the intensities
        stmt = myConn.createStatement();
        String updStr = "UPDATE maldiTemp set intensity = (intensity * 100 / " +
maxInt + ")";
        stmt.executeUpdate(updStr);

    } catch (SQLException e) {
        return 0;
    } catch (Exception e) {
        return 0;
    }
}

return id;
}

public int insertNewIdentifier(String identifier, String notes) {
    try {
        //First check to see if the name is already in the db
        ResultSet myRS = getRS("SELECT * FROM identifier WHERE ident = '" +
identifier + "'");

        //If it is return the number 3
        if (myRS.next()) {
            myRS.close();
            return 3;
        }

        myRS.close();

        String instStmt;
        instStmt = "INSERT INTO identifier (ident,notes) VALUES('" + identifier
+ "',''" + notes + "'")";
        stmt.executeUpdate(instStmt);
        return 0;
    } catch (SQLException e) {
        return 1;
    } catch (Exception e) {
        return 2;
    }
}

public int insertNewGenus(String genusname) {
    try {
        //First check to see if the name is already in the db
        ResultSet myRS = getRS("SELECT * FROM genus WHERE name = '" + genusname
+ "'");

        //If it is return the id

```

```

        if (myRS.next()) {
            int id = myRS.getInt("genus_id");
            myRS.close();
            stmt.close();
            return id;
        }
        myRS.close();
        stmt.close();
        String instStmt;
        instStmt = "INSERT INTO genus (name) VALUES('" + genusname + "')";
        stmt = myConn.createStatement();
        stmt.executeUpdate(instStmt);
        ResultSet myrRS = stmt.getGeneratedKeys();
        myrRS.next();
        int id = myrRS.getInt(1);
        myrRS.close();
        stmt.close();
        return id;
    } catch (SQLException e) {
        return -1;
    } catch (Exception e) {
        return -1;
    }
}

public int insertNewSpecies(String speciesname) {
    try {
        //First check to see if the name is already in the db
        ResultSet myRS = getRS("SELECT * FROM species WHERE name = '" +
speciesname + "'");
        //If it is return the id
        if (myRS.next()) {
            int id = myRS.getInt("species_id");
            myRS.close();
            stmt.close();
            return id;
        }
        myRS.close();
        stmt.close();
        String instStmt;
        instStmt = "INSERT INTO species (name) VALUES('" + speciesname + "')";
        stmt = myConn.createStatement();
        stmt.executeUpdate(instStmt);
        ResultSet myrRS = stmt.getGeneratedKeys();
        myrRS.next();
        int id = myrRS.getInt(1);
        myrRS.close();
        stmt.close();
        return id;
    } catch (SQLException e) {
        return -1;
    } catch (Exception e) {
        return -1;
    }
}

public int insertNewSubspecies(String subspeciesname) {
    try {
        //First check to see if the name is already in the db
        ResultSet myRS = getRS("SELECT * FROM subspecies WHERE name = '" +
subspeciesname + "'");
        //If it is return the id
        if (myRS.next()) {
            int id = myRS.getInt("subspecies_id");

```

```

        myRS.close();
        stmt.close();
        return id;
    }
    myRS.close();
    stmt.close();
    String instStmt;
    instStmt = "INSERT INTO subspecies (name) VALUES('" + subspeciesname +
    "' )";
    stmt = myConn.createStatement();
    stmt.executeUpdate(instStmt);
    ResultSet myrRS = stmt.getGeneratedKeys();
    myrRS.next();
    int id = myrRS.getInt(1);
    myrRS.close();
    stmt.close();
    return id;
} catch (SQLException e) {
    return -1;
} catch (Exception e) {
    return -1;
}
}

public int insertNewSubtype(String othername) {
    try {
        //First check to see if the name is already in the db
        ResultSet myRS = getRS("SELECT * FROM subtype WHERE name = '" +
othername + "'");
        //If it is return the id
        if (myRS.next()) {
            int id = myRS.getInt("subtype_id");
            myRS.close();
            stmt.close();
            return id;
        }
        myRS.close();
        stmt.close();
        String instStmt;
        instStmt = "INSERT INTO subtype (name) VALUES('" + othername + "')";
        stmt = myConn.createStatement();
        stmt.executeUpdate(instStmt);
        ResultSet myrRS = stmt.getGeneratedKeys();
        myrRS.next();
        int id = myrRS.getInt(1);
        myrRS.close();
        stmt.close();
        return id;
    } catch (SQLException e) {
        return -1;
    } catch (Exception e) {
        return -1;
    }
}

public int insertNewStrain(int ident, int number, String mdate, int genus, int
species, int subspecies, int subtype, String notes, String history, int gtent, int
sptent, int subtent, int othertent) {
    int strainID;
    try {
        //First check and make sure that the ident + number does not already
exist.

        ResultSet myRS = getRS("SELECT * FROM sample WHERE identifier_id = " +

```

```

ident + " AND number = " + number);
    if (myRS.next()) {
        return 3;
    }
    notes = notes.replace("'", "''");
    notes = notes.replace("\n", "\\n");
    history = history.replace("'", "''");
    history = history.replace("\n", "\\n");
    //Next we need to get the strain id
    String slctStrain = "SELECT strain_id FROM strain WHERE genus_id=" +
genus + " AND species_id=" + species + " AND subspecies_id=" + subspecies + " AND
subtype_id=" + subtype;

    myRS = getRS(slctStrain);

    if (myRS.next()) {
        //the strain already exists, use this id
        strainID = myRS.getInt("strain_id");
    } else {
        //the strain does not exist, make a new one and use the new id
        myRS.close();
        //Insert the strain ID, now get it the same way as above
        String insrtStrain;
        insrtStrain = "INSERT INTO strain (genus_id, species_id,
subspecies_id, subtype_id) VALUES(" + genus + "," + species + "," + subspecies + "," +
+ subtype + ")";
        stmt.executeUpdate(insrtStrain);
        stmt.close();

        myRS = getRS(slctStrain);

        myRS.next();
        strainID = myRS.getInt("strain_id");
    }

    myRS.close();
    String insrtSample = "";
    insrtSample= "INSERT INTO sample (strain_id, identifier_id, number,
notes, history,date, tentative_genus, tentative_species, tentative_subspecies,
tentative_subtype) VALUES(" + strainID + "," + ident + "," + number + ",";

    if (notes == "") {
        insrtSample = insrtSample + "null,";
    } else {
        insrtSample = insrtSample + "'" + notes + "',";
    }
    if (history == "") {
        insrtSample = insrtSample + "null,";
    } else {
        insrtSample = insrtSample + "'" + history + "',";
    }
    if (mdate.trim() == "") {
        insrtSample = insrtSample + "null,";
    } else {
        insrtSample = insrtSample + "'" + mdate + "',";
    }
    insrtSample = insrtSample + gtent + "," + sptent + "," + subtent + "," +
othertent + ")";

    stmt = myConn.createStatement();

    stmt.executeUpdate(insrtSample);
    stmt.close();
    //Update teh last update date
    executeString("UPDATE lastupdate SET date=CURRENT_TIMESTAMP");

```



```

        return 0;
    } catch (SQLException e) {
        return 1;
    } catch (Exception e) {
        return 2;
    }
}

public java.sql.ResultSet runANDSearch(String sampleid, String identifier,
String genus, String species, String subspecies, String other, String notes, String
history, String gtent, String sptent, String subtent, String othertent, int
maldionly, String mass, int masserror, String minintensity, String lowRM, String
highRM) {
    String SQLSearchQuery;

    if (username.compareTo("rmandrell") == 0 | username.compareTo("psmlogin") ==
0) {
        SQLSearchQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number) 'ident', sample.date, genus.name 'genus',
species.name 'species', subspecies.name 'subspecies', subtype.name 'subtype',
sample.notes, sample.history, sample.tentative_genus, sample.tentative_species,
sample.tentative_subspecies, sample.tentative_subtype FROM identifier, sample,
strain, genus, species, subspecies, subtype WHERE strain.genus_id = genus.genus_id
AND strain.species_id = species.species_id AND subspecies.subspecies_id =
strain.subspecies_id AND subtype.subtype_id = strain.subtype_id AND sample.strain_id
= strain.strain_id AND identifier.identifier_id = sample.identifier_id";
    } else {
        SQLSearchQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number) 'ident', sample.date, genus.name 'genus',
species.name 'species', subspecies.name 'subspecies', subtype.name 'subtype', CASE
WHEN sample.notes LIKE '%CONFIDENTIAL%' THEN 'CONFIDENTIAL See Rob Mandrell before
using' ELSE sample.notes END 'notes', sample.history, sample.tentative_genus,
sample.tentative_species, sample.tentative_subspecies, sample.tentative_subtype FROM
identifier, sample, strain, genus, species, subspecies, subtype WHERE
strain.genus_id = genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";
    }

    if (sampleid == "" & identifier == "" & genus == "" & species == "" &
subspecies == "" & other == "" & notes == "" & history == "" &
gtent.compareTo("Any") == 0 & sptent.compareTo("Any") == 0 &
subtent.compareTo("Any") == 0 & othertent.compareTo("Any") == 0 & maldionly != 1 &
mass==" " & (lowRM == "" | highRM == "")) {
        return getRS(SQLSearchQuery + " ORDER BY identifier.ident,
sample.number");
    } else {
        String addToQuery = " AND (";

        boolean isFirst;
        isFirst = true;

        if (sampleid != "" ) {
            addToQuery = addToQuery.concat("sample.sample_id LIKE '%");
            addToQuery = addToQuery.concat(sampleid);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (identifier != ""){
            if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
            addToQuery = addToQuery.concat("CONCAT(identifier.ident,
sample.number) LIKE '%");
            addToQuery = addToQuery.concat(identifier);
            addToQuery = addToQuery.concat("%'");
        }
    }
}

```

```

        isFirst = false;
    }
    if (genus != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("genus.name LIKE '%");
        addToQuery = addToQuery.concat(genus);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (species != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("species.name LIKE '%");
        addToQuery = addToQuery.concat(species);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (subspecies != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("subspecies.name LIKE '%");
        addToQuery = addToQuery.concat(subspecies);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (other != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("subtype.name LIKE '%");
        addToQuery = addToQuery.concat(other);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (notes != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.notes LIKE '%");
        addToQuery = addToQuery.concat(notes);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (history != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.history LIKE '%");
        addToQuery = addToQuery.concat(history);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (gtent.compareTo("Any") != 0 ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.tentative_genus = '");
        if (gtent.compareTo("true") == 0) {
            addToQuery = addToQuery.concat("1");
        } else {
            addToQuery = addToQuery.concat("0");
        }
        addToQuery = addToQuery.concat("'");
        isFirst = false;
    }
    if (sptent.compareTo("Any") != 0 ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.tentative_species = '");
        if (sptent.compareTo("true") == 0) {
            addToQuery = addToQuery.concat("1");
        } else {
            addToQuery = addToQuery.concat("0");
        }
        addToQuery = addToQuery.concat("'");
        isFirst = false;
    }

```

```

}
if (subtent.compareTo("Any") != 0 ) {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("sample.tentative_subspecies = ");
    if (subtent.compareTo("true") == 0) {
        addToQuery = addToQuery.concat("1");
    } else {
        addToQuery = addToQuery.concat("0");
    }
    addToQuery = addToQuery.concat("'");
    isFirst = false;
}
if (othertent.compareTo("Any") != 0 ) {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("sample.tentative_subtype = ");
    if (othertent.compareTo("true") == 0) {
        addToQuery = addToQuery.concat("1");
    } else {
        addToQuery = addToQuery.concat("0");
    }
    addToQuery = addToQuery.concat("'");
    isFirst = false;
}
if (mass != "" ) {
    double percEr = masserror / 1000000.0;
    String[] allMass = mass.split(",");
    for(int j=0;j<allMass.length; j++) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        if (j > 0){ addToQuery += " AND ";}
        addToQuery = addToQuery.concat("maldi_tof_data.mass between ");
        addToQuery = addToQuery.concat(allMass[j]);
        addToQuery = addToQuery.concat(" - (");
        addToQuery = addToQuery.concat(allMass[j]);
        addToQuery = addToQuery.concat(")");
        addToQuery = addToQuery.concat(Double.toString(percEr));
        addToQuery = addToQuery.concat(" AND ");
        addToQuery = addToQuery.concat(allMass[j]);
        addToQuery = addToQuery.concat(" + (");
        addToQuery = addToQuery.concat(allMass[j]);
        addToQuery = addToQuery.concat(")");
        addToQuery = addToQuery.concat(Double.toString(percEr));
        addToQuery = addToQuery.concat(")");
    }

    addToQuery = addToQuery.concat(" AND");
    addToQuery = addToQuery.concat(" maldi_tof_data.intensity >= ");
    addToQuery = addToQuery.concat(minintensity);
    isFirst = false;
}
if (lowRM != "" & highRM != "" ) {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("sample.number >= ");
    addToQuery = addToQuery.concat(lowRM);
    addToQuery = addToQuery.concat(" AND sample.number <= ");
    addToQuery = addToQuery.concat(highRM);
    isFirst = false;
}

if (maldionly == 1) {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("maldi_tof.sample_id = ");
    sample.sample_id AND maldi_tof.maldi_tof_id = maldi_tof_data.maldi_tof_id";
    addToQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number) 'ident', sample.date, genus.name 'genus',

```

```

species.name 'species', subspecies.name 'subspecies', subtype.name 'subtype',
sample.notes, sample.history, sample.tentative_genus, sample.tentative_species,
sample.tentative_subspecies, sample.tentative_subtype FROM identifier, sample,
strain, genus, species, subspecies, subtype, maldi_tof, maldi_tof_data WHERE
strain.genus_id = genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id " + addToQuery;
    } else {
        if (mass != "") {
            addToQuery = addToQuery.concat(" AND maldi_tof.sample_id =
sample.sample_id AND maldi_tof.maldi_tof_id = maldi_tof_data.maldi_tof_id ");
            addToQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number) 'ident', sample.date, genus.name 'genus',
species.name 'species', subspecies.name 'subspecies', subtype.name 'subtype',
sample.notes, sample.history, sample.tentative_genus, sample.tentative_species,
sample.tentative_subspecies, sample.tentative_subtype FROM identifier, sample,
strain, genus, species, subspecies, subtype, maldi_tof, maldi_tof_data WHERE
strain.genus_id = genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id " + addToQuery;
        } else {
            addToQuery = SQLSearchQuery.concat(addToQuery);
        }
    }
    addToQuery = addToQuery.concat("") ORDER BY identifier.ident,
sample.number");
    return getRS(addToQuery);
}
}

public java.sql.ResultSet runORSearch(String sampleid, String identifier, String
genus, String species, String subspecies, String other, String notes, String
history, String gtent, String sptent, String subtent, String othertent, int
maldionly, String lowRM, String highRM) {
    String SQLSearchQuery;

    if (username.compareTo("rmandrell") == 0 | username.compareTo("psmlogin") ==
0) {
        SQLSearchQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number) 'ident', sample.date, genus.name 'genus',
species.name 'species', subspecies.name 'subspecies', subtype.name 'subtype',
sample.notes, sample.history, sample.tentative_genus, sample.tentative_species,
sample.tentative_subspecies, sample.tentative_subtype FROM identifier, sample,
strain, genus, species, subspecies, subtype WHERE strain.genus_id = genus.genus_id
AND strain.species_id = species.species_id AND subspecies.subspecies_id =
strain.subspecies_id AND subtype.subtype_id = strain.subtype_id AND sample.strain_id
= strain.strain_id AND identifier.identifier_id = sample.identifier_id";
    } else {
        SQLSearchQuery = "SELECT DISTINCT sample.sample_id,
CONCAT(identifier.ident, sample.number) 'ident', sample.date, genus.name 'genus',
species.name 'species', subspecies.name 'subspecies', subtype.name 'subtype', CASE
WHEN sample.notes LIKE '%CONFIDENTIAL%' THEN 'CONFIDENTIAL See Rob Mandrell before
using' ELSE sample.notes END 'notes', sample.history, sample.tentative_genus,
sample.tentative_species, sample.tentative_subspecies, sample.tentative_subtype FROM
identifier, sample, strain, genus, species, subspecies, subtype WHERE
strain.genus_id = genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";
    }

    if (sampleid == "" & identifier == "" & genus == "" & species == "" &
subspecies == "" & other == "" & notes == "" & history == "" &

```

```

gtent.compareTo("Any") == 0 & sptent.compareTo("Any") == 0 &
subtent.compareTo("Any") == 0 & othertent.compareTo("Any") == 0 & maldionly != 1 &
(lowRM == "" | highRM == "")) {
    return getRS(SQLSearchQuery + " ORDER BY identifier.ident,
sample.number");
} else {
    String addToQuery;

    addToQuery = SQLSearchQuery;

    addToQuery = addToQuery + " AND (";
    boolean isFirst = true;
    if (sampleid != "") {
        addToQuery = addToQuery.concat("sample.sample_id LIKE '%");
        addToQuery = addToQuery.concat(sampleid);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (identifier != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat("CONCAT(identifier.ident,
sample.number) LIKE '%");
        addToQuery = addToQuery.concat(identifier);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (genus != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat(" genus.name LIKE '%");
        addToQuery = addToQuery.concat(genus);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (species != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat(" species.name LIKE '%");
        addToQuery = addToQuery.concat(species);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (subspecies != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat(" subspecies.name LIKE '%");
        addToQuery = addToQuery.concat(subspecies);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (other != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat(" subtype.name LIKE '%");
        addToQuery = addToQuery.concat(other);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (notes != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat(" sample.notes LIKE '%");
        addToQuery = addToQuery.concat(notes);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (history != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat(" sample.history LIKE '%");
        addToQuery = addToQuery.concat(history);
    }
}

```

```

        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (gtent.compareTo("Any") != 0 ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat("sample.tentative_genus = '");
        if (gtent.compareTo("true") == 0) {
            addToQuery = addToQuery.concat("1");
        } else {
            addToQuery = addToQuery.concat("0");
        }
        addToQuery = addToQuery.concat("'");
        isFirst = false;
    }
    if (sptent.compareTo("Any") != 0 ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat("sample.tentative_species = '");
        if (sptent.compareTo("true") == 0) {
            addToQuery = addToQuery.concat("1");
        } else {
            addToQuery = addToQuery.concat("0");
        }
        addToQuery = addToQuery.concat("'");
        isFirst = false;
    }
    if (subtent.compareTo("Any") != 0 ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat("sample.tentative_subspecies = '");
        if (subtent.compareTo("true") == 0) {
            addToQuery = addToQuery.concat("1");
        } else {
            addToQuery = addToQuery.concat("0");
        }
        addToQuery = addToQuery.concat("'");
        isFirst = false;
    }
    if (othertent.compareTo("Any") != 0 ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat("sample.tentative_subtype = '");
        if (othertent.compareTo("true") == 0) {
            addToQuery = addToQuery.concat("1");
        } else {
            addToQuery = addToQuery.concat("0");
        }
        addToQuery = addToQuery.concat("'");
        isFirst = false;
    }
    if (lowRM != "" & highRM != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.number >= ");
        addToQuery = addToQuery.concat(lowRM);
        addToQuery = addToQuery.concat(" AND sample.number <= ");
        addToQuery = addToQuery.concat(highRM);
        isFirst = false;
    }
    addToQuery = addToQuery.concat(") ORDER BY identifier.ident,
sample.number");
    return getRS(addToQuery);
}
}
}

```

```

public String runANDSearchStr(String sampleid, String identifier, String genus,
String species, String subspecies, String other, String notes, String history, int
maldionly, String mass, int masserror, String minintensity, String lowRM, String
highRM) {

```

```

String SQLSearchQuery;

    if (username.compareTo("rmandrell") == 0 | username.compareTo("psmlogin") ==
0) {
        SQLSearchQuery = "SELECT DISTINCT sample.sample_id FROM identifier,
sample, strain, genus, species, subspecies, subtype WHERE strain.genus_id =
genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";
    } else {
        SQLSearchQuery = "SELECT DISTINCT sample.sample_id FROM identifier,
sample, strain, genus, species, subspecies, subtype WHERE strain.genus_id =
genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";
    }

    if (sampleid == "" & identifier == "" & genus == "" & species == "" &
subspecies == "" & other == "" & notes == "" & history == "" & maldionly != 1 &
mass==" " & (lowRM == "" | highRM == "")) {
        return SQLSearchQuery;
    } else {
        String addToQuery = " AND (";

        boolean isFirst;
        isFirst = true;

        if (sampleid != "" ) {
            addToQuery = addToQuery.concat("sample.sample_id LIKE '%");
            addToQuery = addToQuery.concat(sampleid);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (identifier != ""){
            if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
            addToQuery = addToQuery.concat("CONCAT(identifier.iden,
sample.number) LIKE '%");
            addToQuery = addToQuery.concat(identifier);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (genus != "" ) {
            if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
            addToQuery = addToQuery.concat("genus.name LIKE '%");
            addToQuery = addToQuery.concat(genus);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (species != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
            addToQuery = addToQuery.concat("species.name LIKE '%");
            addToQuery = addToQuery.concat(species);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (subspecies != "" ) {
            if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
            addToQuery = addToQuery.concat("subspecies.name LIKE '%");
            addToQuery = addToQuery.concat(subspecies);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (other != "") {

```

```

        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("subtype.name LIKE '%");
        addToQuery = addToQuery.concat(other);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (notes != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.notes LIKE '%");
        addToQuery = addToQuery.concat(notes);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (history != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.history LIKE '%");
        addToQuery = addToQuery.concat(history);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (mass != "" ) {
        String[] allMass = mass.split(",");
        for(int j = 0; j < allMass.length; j++) {
            double percEr = masserror / 1000000.0;
            if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
            addToQuery = addToQuery.concat("maldi_tof_data.mass between ");
            addToQuery = addToQuery.concat(allMass[j]);
            addToQuery = addToQuery.concat(" - (");
            addToQuery = addToQuery.concat(allMass[j]);
            addToQuery = addToQuery.concat("*");
            addToQuery = addToQuery.concat(Double.toString(percEr));
            addToQuery = addToQuery.concat(")");
            addToQuery = addToQuery.concat(" AND ");
            addToQuery = addToQuery.concat(allMass[j]);
            addToQuery = addToQuery.concat(" + (");
            addToQuery = addToQuery.concat(allMass[j]);
            addToQuery = addToQuery.concat("*");
            addToQuery = addToQuery.concat(Double.toString(percEr));
            addToQuery = addToQuery.concat(")");
            addToQuery = addToQuery.concat(" AND maldi_tof_data.intensity >=
");
            addToQuery = addToQuery.concat(minintensity);
        }
        isFirst = false;
    }
    if (lowRM != "" & highRM != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.number >= ");
        addToQuery = addToQuery.concat(lowRM);
        addToQuery = addToQuery.concat(" AND sample.number <= ");
        addToQuery = addToQuery.concat(highRM);
        isFirst = false;
    }
    if (maldionly == 1) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("maldi_tof.sample_id =
sample.sample_id AND maldi_tof.maldi_tof_id = maldi_tof_data.maldi_tof_id");
        addToQuery = "SELECT DISTINCT sample.sample_id FROM identifier,
sample, strain, genus, species, subspecies, subtype, maldi_tof, maldi_tof_data WHERE
strain.genus_id = genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id " + addToQuery;
    } else {

```



```

        if (mass != "") {
            addToQuery = addToQuery.concat(" AND maldi_tof.sample_id =
sample.sample_id AND maldi_tof.maldi_tof_id = maldi_tof_data.maldi_tof_id");
            addToQuery = "SELECT DISTINCT sample.sample_id FROM identifier,
sample, strain, genus, species, subspecies, subtype, maldi_tof, maldi_tof_data WHERE
strain.genus_id = genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id " + addToQuery;
        } else {
            addToQuery = SQLSearchQuery.concat(addToQuery);
        }
    }
    addToQuery = addToQuery.concat(")");
    return addToQuery;
}
}

```

```

public String runANDInSilicoSearchStr(String identifier, String genus, String
species, String subspecies, String other, String protein, String notes, String mass,
String masserror) {
    String SQLSearchQuery;

```

```

    SQLSearchQuery = "SELECT DISTINCT i.in_silico_id, i.identification, i.date,
i.average_mw, i.protein, i.notes, g.name Genus, sp.name Species, sub.name
Subspecies, st.name Subtype, i.aa_seq FROM in_silico i, genus g, species sp,
subspecies sub, subtype st, strain sn WHERE sn.strain_id = i.strain_id AND
sn.genus_id = g.genus_id AND sn.species_id = sp.species_id AND sub.subspecies_id =
sn.subspecies_id AND st.subtype_id = sn.subtype_id";

```

```

    if (identifier == "" & genus == "" & species == "" & subspecies == "" &
other == "" & notes == "" & mass == "" & protein == "") {
        return SQLSearchQuery;
    } else {
        String addToQuery = " AND (";

```

```

        boolean isFirst;
        isFirst = true;

```

```

        if (protein != "") {
            addToQuery = addToQuery.concat("i.protein LIKE '%");
            addToQuery = addToQuery.concat(protein);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }

```

```

        if (identifier != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" AND "); }
            addToQuery = addToQuery.concat("i.identification LIKE '%");
            addToQuery = addToQuery.concat(identifier);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }

```

```

        if (genus != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" AND "); }
            addToQuery = addToQuery.concat("g.name LIKE '%");
            addToQuery = addToQuery.concat(genus);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }

```

```

        if (species != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" AND "); }
            addToQuery = addToQuery.concat("sp.name LIKE '%");
            addToQuery = addToQuery.concat(species);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }

```

```

    }
    if (subspecies != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sub.name LIKE '%");
        addToQuery = addToQuery.concat(subspecies);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (other != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("st.name LIKE '%");
        addToQuery = addToQuery.concat(other);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (notes != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("i.notes LIKE '%");
        addToQuery = addToQuery.concat(notes);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (mass != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("i.average_mw > ");
        addToQuery = addToQuery.concat(mass + "-" + masserror + " AND
i.average_mw < " + mass + "+" + masserror);
        isFirst = false;
    }
    addToQuery = addToQuery.concat(")");
    return SQLSearchQuery.concat(addToQuery);
}
}

```

```

public String runANDMSMSSearchStr(String identifier, String genus, String
species, String subspecies, String other, String protein, String notes, String mass,
String masserror) {
    String SQLSearchQuery;

```

```

    SQLSearchQuery = "SELECT DISTINCT i.ms_ms_id, i.sample_id, i.date,
i.precursor_ion_mtoz, i.protein, i.notes, g.name Genus, sp.name Species, sub.name
Subspecies, st.name Subtype FROM ms_ms i, genus g, species sp, subspecies sub,
subtype st, strain sn, sample sm, identifier ident WHERE sm.sample_id = i.sample_id
AND sm.strain_id = sn.strain_id AND sm.identifier_id = ident.identifier_id AND
sn.genus_id = g.genus_id AND sn.species_id = sp.species_id AND sub.subspecies_id =
sn.subspecies_id AND st.subtype_id = sn.subtype_id";

```

```

    if (identifier == "" & genus == "" & species == "" & subspecies == "" &
other == "" & notes == "" & mass==" " & protein == "") {
        return SQLSearchQuery;
    } else {
        String addToQuery = " AND (";

        boolean isFirst;
        isFirst = true;

```

```

        if (protein != "" ) {
            addToQuery = addToQuery.concat("i.protein LIKE '%");
            addToQuery = addToQuery.concat(protein);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (identifier != ""){
            if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
            addToQuery = addToQuery.concat("CONCAT_WS(' ', ident.ident,

```

```

sm.number) LIKE '%");
    addToQuery = addToQuery.concat(identifier);
    addToQuery = addToQuery.concat("%'");
    isFirst = false;
}
if (genus != "" ) {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("g.name LIKE '%");
    addToQuery = addToQuery.concat(genus);
    addToQuery = addToQuery.concat("%'");
    isFirst = false;
}
if (species != "") {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("sp.name LIKE '%");
    addToQuery = addToQuery.concat(species);
    addToQuery = addToQuery.concat("%'");
    isFirst = false;
}
if (subspecies != "" ) {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("sub.name LIKE '%");
    addToQuery = addToQuery.concat(subspecies);
    addToQuery = addToQuery.concat("%'");
    isFirst = false;
}
if (other != "") {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("st.name LIKE '%");
    addToQuery = addToQuery.concat(other);
    addToQuery = addToQuery.concat("%'");
    isFirst = false;
}
if (notes != "") {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("i.notes LIKE '%");
    addToQuery = addToQuery.concat(notes);
    addToQuery = addToQuery.concat("%'");
    isFirst = false;
}
if (mass != "" ) {
    if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
    addToQuery = addToQuery.concat("i.precursor_ion_mtoz > ");
    addToQuery = addToQuery.concat(mass + "-" + masserror + " AND
i.precursor_ion_mtoz < " + mass + "+" + masserror);
    isFirst = false;
}
addToQuery = addToQuery.concat(")");
return SQLSearchQuery.concat(addToQuery);
}
}

```

```

public String runORSearchStr(String sampleid, String identifier, String genus,
String species, String subspecies, String other, String notes, String history, int
maldionly, String lowRM, String highRM) {
    String SQLSearchQuery;

```

```

    if (username.compareTo("rmandrell") == 0 | username.compareTo("psmlogin") ==
0) {
        SQLSearchQuery = "SELECT DISTINCT sample.sample_id FROM identifier,
sample, strain, genus, species, subspecies, subtype WHERE strain.genus_id =
genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";

```

```

    } else {
        SQLSearchQuery = "SELECT DISTINCT sample.sample_id FROM identifier,
sample, strain, genus, species, subspecies, subtype WHERE strain.genus_id =
genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";
    }

    if (sampleid == "" & identifier == "" & genus == "" & species == "" &
subspecies == "" & other == "" & notes == "" & history == "" & maldionly != 1 &
(lowRM == "" | highRM == "")) {
        return SQLSearchQuery;
    } else {
        String addToQuery;

        addToQuery = SQLSearchQuery;

        addToQuery = addToQuery + " AND (";
        boolean isFirst = true;
        if (sampleid != "") {
            addToQuery = addToQuery.concat("sample.sample_id LIKE '%");
            addToQuery = addToQuery.concat(sampleid);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (identifier != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
            addToQuery = addToQuery.concat("CONCAT(identifier.ident,
sample.number) LIKE '%");
            addToQuery = addToQuery.concat(identifier);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (genus != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
            addToQuery = addToQuery.concat(" genus.name LIKE '%");
            addToQuery = addToQuery.concat(genus);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (species != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
            addToQuery = addToQuery.concat(" species.name LIKE '%");
            addToQuery = addToQuery.concat(species);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (subspecies != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
            addToQuery = addToQuery.concat(" subspecies.name LIKE '%");
            addToQuery = addToQuery.concat(subspecies);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (other != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
            addToQuery = addToQuery.concat(" subtype.name LIKE '%");
            addToQuery = addToQuery.concat(other);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (notes != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
            addToQuery = addToQuery.concat(" sample.notes LIKE '%");

```

```

        addToQuery = addToQuery.concat(notes);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (history != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
        addToQuery = addToQuery.concat(" sample.history LIKE '%");
        addToQuery = addToQuery.concat(history);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (lowRM != "" & highRM != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" AND ");}
        addToQuery = addToQuery.concat("sample.number >= ");
        addToQuery = addToQuery.concat(lowRM);
        addToQuery = addToQuery.concat(" AND sample.number <= ");
        addToQuery = addToQuery.concat(highRM);
        isFirst = false;
    }
    addToQuery = addToQuery.concat(")");
    return addToQuery;
}
}

public String runORInSilicoSearchStr(String identifier, String genus, String
species, String subspecies, String other, String protein, String notes, String mass,
String masserror) {
    String SQLSearchQuery;

    SQLSearchQuery = "SELECT DISTINCT i.in_silico_id, i.identification, i.date,
i.average_mw, i.protein, i.notes, g.name Genus, sp.name Species, sub.name
Subspecies, st.name Subtype, i.aa_seq FROM in_silico i, genus g, species sp,
subspecies sub, subtype st, strain sn WHERE sn.strain_id = i.strain_id AND
sn.genus_id = g.genus_id AND sn.species_id = sp.species_id AND sub.subspecies_id =
sn.subspecies_id AND st.subtype_id = sn.subtype_id";

    if (identifier == "" & genus == "" & species == "" & subspecies == "" &
other == "" & notes == "" & mass==" " & protein == "") {
        return SQLSearchQuery;
    } else {
        String addToQuery = " OR (";

        boolean isFirst;
        isFirst = true;

        if (protein != "" ) {
            addToQuery = addToQuery.concat("i.protein LIKE '%");
            addToQuery = addToQuery.concat(protein);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (identifier != ""){
            if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
            addToQuery = addToQuery.concat("i.identification LIKE '%");
            addToQuery = addToQuery.concat(identifier);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (genus != "" ) {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
            addToQuery = addToQuery.concat("g.name LIKE '%");
            addToQuery = addToQuery.concat(genus);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
    }
}

```

```

        if (species != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
            addToQuery = addToQuery.concat("sp.name LIKE '%");
            addToQuery = addToQuery.concat(species);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (subspecies != "" ) {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
            addToQuery = addToQuery.concat("sub.name LIKE '%");
            addToQuery = addToQuery.concat(subspecies);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (other != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
            addToQuery = addToQuery.concat("st.name LIKE '%");
            addToQuery = addToQuery.concat(other);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (notes != "") {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
            addToQuery = addToQuery.concat("i.notes LIKE '%");
            addToQuery = addToQuery.concat(notes);
            addToQuery = addToQuery.concat("%'");
            isFirst = false;
        }
        if (mass != "" ) {
            if (!isFirst) { addToQuery = addToQuery.concat(" OR ");}
            addToQuery = addToQuery.concat("i.average_mw > ");
            addToQuery = addToQuery.concat(mass + "-" + masserror + " AND
i.average_mw < " + mass + "+" + masserror);
            isFirst = false;
        }
        addToQuery = addToQuery.concat(")");
        return SQLSearchQuery.concat(addToQuery);
    }
}

```

```

public String runORMSMSSearchStr(String identifier, String genus, String
species, String subspecies, String other, String protein, String notes, String mass,
String masserror) {
    String SQLSearchQuery;

```

```

    SQLSearchQuery = "SELECT DISTINCT i.ms_ms_id, i.sample_id, i.date,
i.precursor_ion_mtoz, i.protein, i.notes, g.name Genus, sp.name Species, sub.name
Subspecies, st.name Subtype FROM ms_ms i, genus g, species sp, subspecies sub,
subtype st, strain sn, sample sm, identifier ident WHERE sm.sample_id = i.sample_id
AND sm.strain_id = sn.strain_id AND sm.identifier_id = ident.identifier_id AND
sn.genus_id = g.genus_id AND sn.species_id = sp.species_id AND sub.subspecies_id =
sn.subspecies_id AND st.subtype_id = sn.subtype_id";

```

```

        if (identifier == "" & genus == "" & species == "" & subspecies == "" &
other == "" & notes == "" & mass==" " & protein == "") {
            return SQLSearchQuery;
        } else {
            String addToQuery = " OR (";

            boolean isFirst;
            isFirst = true;

            if (protein != "" ) {
                addToQuery = addToQuery.concat("i.protein LIKE '%");
                addToQuery = addToQuery.concat(protein);
            }

```

```

        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (identifier != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
        addToQuery = addToQuery.concat("CONCAT_WS(' ', ident.ident,
sm.number) LIKE '%");
        addToQuery = addToQuery.concat(identifier);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (genus != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
        addToQuery = addToQuery.concat("g.name LIKE '%");
        addToQuery = addToQuery.concat(genus);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (species != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
        addToQuery = addToQuery.concat("sp.name LIKE '%");
        addToQuery = addToQuery.concat(species);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (subspecies != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
        addToQuery = addToQuery.concat("sub.name LIKE '%");
        addToQuery = addToQuery.concat(subspecies);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (other != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
        addToQuery = addToQuery.concat("st.name LIKE '%");
        addToQuery = addToQuery.concat(other);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (notes != "") {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
        addToQuery = addToQuery.concat("i.notes LIKE '%");
        addToQuery = addToQuery.concat(notes);
        addToQuery = addToQuery.concat("%'");
        isFirst = false;
    }
    if (mass != "" ) {
        if (!isFirst) { addToQuery = addToQuery.concat(" OR "); }
        addToQuery = addToQuery.concat("i.precursor_ion_mtoz > ");
        addToQuery = addToQuery.concat(mass + "-" + masserror + " AND
i.precursor_ion_mtoz < " + mass + "+" + masserror);
        isFirst = false;
    }
    addToQuery = addToQuery.concat(")");
    return SQLSearchQuery.concat(addToQuery);
}

}

public java.sql.ResultSet searchBySampleId(String sampleid) {
    String addToQuery = " AND sample.sample_id = '";
    addToQuery = addToQuery.concat(sampleid);
    addToQuery = addToQuery.concat("'");
    return getRS(SQLQuery.concat(addToQuery));
}

```

```

    public java.sql.ResultSet searchBySampleIdStandard(String sampleid) {
        String mQuery;
        if (username.compareTo("rmandrell") == 0 | username.compareTo("psmlogin") ==
0) {
            mQuery = "SELECT sample.sample_id, identifier.ident, sample.number,
sample.date, genus.name, species.name, subspecies.name, subtype.name, sample.notes,
sample.history, sample.tentative_genus, sample.tentative_species,
sample.tentative_subspecies, sample.tentative_subtype FROM identifier, sample,
strain, genus, species, subspecies, subtype WHERE strain.genus_id = genus.genus_id
AND strain.species_id = species.species_id AND subspecies.subspecies_id =
strain.subspecies_id AND subtype.subtype_id = strain.subtype_id AND sample.strain_id
= strain.strain_id AND identifier.identifier_id = sample.identifier_id";
        } else {
            mQuery = "SELECT sample.sample_id, identifier.ident, sample.number,
sample.date, genus.name, species.name, subspecies.name, subtype.name, CASE WHEN
sample.notes LIKE '%CONFIDENTIAL%' THEN 'CONFIDENTIAL See Rob Mandrell before using'
ELSE sample.notes END, sample.history, sample.tentative_genus,
sample.tentative_species, sample.tentative_subspecies, sample.tentative_subtype FROM
identifier, sample, strain, genus, species, subspecies, subtype WHERE
strain.genus_id = genus.genus_id AND strain.species_id = species.species_id AND
subspecies.subspecies_id = strain.subspecies_id AND subtype.subtype_id =
strain.subtype_id AND sample.strain_id = strain.strain_id AND
identifier.identifier_id = sample.identifier_id";
        }
        mQuery = mQuery + " AND sample_id = " + sampleid;
        return getRS(mQuery);
    }

    public boolean isDigit(String schar) {
        if (schar.compareTo("0") == 0) {
            return true;
        } else if (schar.compareTo("1") == 0) {
            return true;
        } else if (schar.compareTo("2") == 0) {
            return true;
        } else if (schar.compareTo("3") == 0) {
            return true;
        } else if (schar.compareTo("4") == 0) {
            return true;
        } else if (schar.compareTo("5") == 0) {
            return true;
        } else if (schar.compareTo("6") == 0) {
            return true;
        } else if (schar.compareTo("7") == 0) {
            return true;
        } else if (schar.compareTo("8") == 0) {
            return true;
        } else if (schar.compareTo("9") == 0) {
            return true;
        } else {
            return false;
        }
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public int updateStrain(int sampleid, int ident, int number, String mdate, int
genus, int species, int subspecies, int subtype, String notes, String history, int

```



```

gentent, int sptent, int subtent, int othertent) {
    //Similar to insert except the insert is an update statement instead of an
    insert statement, and we need the sample_id
    String slctStrain = "SELECT strain_id FROM strain WHERE genus_id=" + genus +
    " AND species_id=" + species + " AND subspecies_id=" + subspecies + " AND
    subtype_id=" + subtype;
    int strainID;
    notes = notes.replace("'", "\\'");
    notes = notes.replace("\\'", "'");
    history = history.replace("'", "\\'");
    history = history.replace("\\'", "'");
    try {
        ResultSet myRS = getRS(slctStrain);
        if (myRS.next()) {
            //the strain already exists, use this id
            strainID = myRS.getInt("strain_id");
        } else {
            //the strain does not exist, make a new one and use the new id
            myRS.close();
            //Insert the strain ID, now get it the same way as above
            String insrtStrain;
            insrtStrain = "INSERT INTO strain (genus_id, species_id,
subspecies_id, subtype_id) VALUES(" + genus + "," + species + "," + subspecies + "," +
+ subtype + ")";
            stmt.executeUpdate(insrtStrain);
            stmt.close();
            myRS = getRS(slctStrain);

            myRS.next();
            strainID = myRS.getInt("strain_id");
        }

        myRS.close();
        String insrtSample = "";
        insrtSample = "UPDATE sample set " + "strain_id=" + strainID +
        ",identifier_id=" + ident + ",number=" + number;

        if (notes != "") {
            insrtSample = insrtSample + ", notes='" + notes + "'";
        }
        if (history != "") {
            insrtSample = insrtSample + ", history='" + history + "'";
        }
        if (mdate.trim() != "") {
            insrtSample = insrtSample + ", date = '" + mdate + "'";
        }
        insrtSample = insrtSample + ",tentative_genus=" + gtent +
        ",tentative_species=" + sptent + ",tentative_subspecies=" + subtent +
        ",tentative_subtype=" + othertent + " WHERE sample_id = " + sampleid;

        stmt = myConn.createStatement();

        stmt.executeUpdate(insrtSample);
        return 0;
    } catch (SQLException e) {
        return 1;
    } catch (Exception e) {
        return 2;
    }
}
}

```

```

/*
 * nucelotideinsert.java
 *
 * Created on August 8, 2005, 3:57 PM
 */

package psm.gene;

import java.beans.*;
import java.io.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletInputStream;
import java.io.Serializable;
import java.sql.*;
import psm.db.dbAccess;

/**
 * @author User
 */
public class nucelotideinsert {
    private String fileName;

    public nucelotideinsert() {

    }

    public int insertNucleotide(HttpServletRequest request, psm.db.dbAccess myDB)
    throws IOException {
        /*
         *String ext is short for extension, this is used to determine where the
info should be placed
         *AA or NT, becuae both have the same format
         */
        ServletInputStream inStr = request.getInputStream();
        byte[] line = new byte[128];

        int i = inStr.readLine(line, 0, 128);

        //The length of the boundary should be greater than three, anything less
means an error has occurred, so exit.
        if (i < 3) {
            return 1;
        }

        //Boundary length is the first line minus the first two CR's that terminate
it
        int boundaryLength = i - 2;
        //Retrieves the boundary minus the two CR's
        String boundary = new String(line, 0, boundaryLength);
        fileName = null;
        int locusID = -1;
        String sequence = "";
        boolean isFile = false;
        while (i != -1) {
            //Store the current line as a string
            //iterate to next line
            i = inStr.readLine(line, 0, 128);
            String newLine = new String(line, 0, i);
            if (!isFile) {
                //All we care about is the file so loop through all the fields until
you hit this.
                if (newLine.indexOf("filename=\"") != -1) {
                    fileName = new String(line, newLine.indexOf("filename=\"") + 10

```

```

, newLine.indexOf("\'", newLine.indexOf("filename=\'\')));
    fileName = fileName.replace("\'", "");
    fileName = fileName.trim();
    isFile= true;
}
} else {
    //if no filename was entered exit
    if (fileName==null) {
        return 2;
    }
    //Do not read the very last line
    if (!newLine.startsWith(boundary) & newLine.indexOf("Content") == -1
& newLine.length() > 4) {
        //This is where reading of the lines of the file occurs.
        while (i != -1 && !newLine.startsWith(boundary)) {
            i = inStr.readLine(line, 0, 128);

            //There are two types of lines in this file, one with locus
with a preceeding >
            // and a line that contains the code
            if (newLine.indexOf(">") != -1) {
                //We are dealing with a new locus
                //The locusName is the entire line minus teh > sign and
minus the last carriage return
                if (locusID != -1){
                    //this means there is data we need to insert
                    String insStr = "";
                    if (fileName.substring(fileName.indexOf(".") +
1).compareToIgnoreCase("AA") == 0) {
                        insStr = "INSERT INTO
genelist.nucleotidesequence VALUES(" + locusID + "," + sequence.trim() + "')";
                    } else if (fileName.substring(fileName.indexOf(".")
+ 1).compareToIgnoreCase("NT") == 0) {
                        insStr = "INSERT INTO genelist.nucleotide
VALUES(" + locusID + "," + sequence + "')";
                    }

                    if (myDB.executeString(insStr) != 0) {
                        return 6;
                    } else {
                        //insrt successfull reset the variable
                        sequence = "";
                    }
                }
                String locusName =
newLine.substring(newLine.indexOf(">") + 1, newLine.length() - 2);

                //Now we want to discover the ID of this locus
                java.sql.ResultSet myRS;
                String selStr = "SELECT locus_id FROM genelist.locus
WHERE name = '" + locusName + "'";
                myRS = myDB.getRS(selStr);
                try {
                    if (myRS.next()) {
                        locusID = myRS.getInt("locus_id");
                    } else {
                        //Do not continue without an ID
                        return 5;
                    }
                }
                myRS.close();
            } catch (SQLException e) {
                return -1;
            }
        }
    }
}

```

```

        } catch (Exception e) {
            return -2;
        }

        } else if(newLine.trim() != "" &&
newLine.trim().compareTo("") != 0) {
        //Now we are dealing with a line that has nucleotide
info in it
            sequence = sequence.concat(newLine);
        } else {
            i = inStr.readLine(line, 0, 128);
            if (i == -1) {
                String insStr = "";
                if (fileName.substring(fileName.indexOf(".") +
1).compareToIgnoreCase("AA") == 0) {
                    insStr = "INSERT INTO
genelist.nucleotidesequence VALUES(" + locusID + ", '" + sequence.trim() + "'";
                } else if (fileName.substring(fileName.indexOf(".")
+ 1).compareToIgnoreCase("NT") == 0) {
                    insStr = "INSERT INTO genelist.nucleotide
VALUES(" + locusID + ", '" + sequence + "'";
                }

                if (myDB.executeString(insStr) != 0) {
                    return 6;
                } else {
                    //insrt successfull reset the variable
                    sequence = "";
                }
            }
        }
    }
    if (i != -1) {
        newLine = new String(line, 0, i);
    }
}

    }
    newLine = null;
}

    return 0;
}

public String getFileName() {
    return fileName;
}

public void setFileName(String fileName) {
    this.fileName = fileName;
}

}

```

```

/*
 * maldiGraph.java
 *
 * Created on June 27, 2005, 12:03 PM
 */

package psm.graph;

import java.beans.*;
import java.io.Serializable;
import java.awt.image.BufferedImage;
import com.sun.image.codec.jpeg.*;
import java.awt.*;
import java.sql.*;
import psm.db.dbAccess;
/**
 * @author Brandon
 */
public class maldiGraph extends Object implements Serializable {

    public maldiGraph() {

        public Graphics2D gDraw(Graphics2D biCtx, java.sql.ResultSet myRS, int intA, int
        intB, int topWrite, int bottom, int step, int xlength) {
            float iteration = (float)(intB - intA) / xlength; //gives what value
            coordinate 1 has for the iteration

            int topTen=0;
            try {
                if (myRS.next()) {
                    myRS.first();

                    do {
                        topTen++;

                        float dx = myRS.getInt("mass");
                        dx = (dx - intA) / iteration;
                        int y = Math.round(myRS.getInt("intensity"));
                        y = y * step;
                        int x = Math.round(dx);
                        biCtx.setColor(Color.RED);
                        biCtx.drawLine((60 + x),(bottom - y), (60 + x),bottom);
                        if (topTen <= topWrite) {
                            String mass = myRS.getString("mass");
                            int offset = mass.length() / 2 * 7;
                            if (offset % 2 !=0) {offset+=3;}
                            biCtx.setColor(Color.BLACK);
                            biCtx.drawString(mass, 60 + x - offset, (bottom -y) - 10);
                        }
                    } while (myRS.next());
                    myRS.close();
                }
            } catch (SQLException e) {
            } catch (Exception e) {
            } finally {
            }
            return biCtx;
        }
    }
}

```

```

    public Graphics2D gDrawPlusInSilicInfo(Graphics2D biCtx, java.sql.ResultSet
myRS, int intA, int intB, int topwrite, int bottom, int step, int xlength) {
    float iteration = (float)(intB - intA) / xlength; //gives what value
coordinate 1 has for the iteration

    int topTen=0;
    try {
        Thread.sleep(50);
        if (myRS.next()) {
            myRS.first();
            do {
                topTen++;
                while(myRS.getInt("msms") == 0.0) {
                    myRS.next();
                }
                float dx = myRS.getFloat("msms");
                dx = (dx - intA) / iteration;
                int y = Math.round(myRS.getInt("intensity"));
                y = y * step;
                int x = Math.round(dx);
                biCtx.setColor(Color.RED);
                biCtx.drawLine((60 + x), (bottom - y), (60 + x), bottom);

                if (topTen <= topwrite) {

                    String mass = myRS.getString("msms") + ", " +
myRS.getString("insilico");
                    int ofset = mass.length() / 2 * 7;
                    if (ofset % 2 != 0) {ofset+=3;}
                    biCtx.setColor(Color.BLACK);
                    biCtx.drawString(mass, 60 + x - ofset, (bottom - y) - 18);

                    String type = myRS.getString("iontype") + ", " +
myRS.getString("residue");
                    biCtx.drawString(type, 60 + x - ofset, (bottom - y) - 6);

                }

            } while (myRS.next());
            myRS.close();
        }
    } catch (SQLException e) {
    } catch (Exception e) {
    } finally {
    }
    return biCtx;
}

    public Graphics2D gDrawPlusInSilicInfoSingle(Graphics2D biCtx,
java.sql.ResultSet myRS, int intA, int intB, int topwrite, int bottom, int step, int
xlength) {
    float iteration = (float)(intB - intA) / xlength; //gives what value
coordinate 1 has for the iteration

```

```

int topTen=0;
try {
    Thread.sleep(50);
    if (myRS.next()) {
        myRS.first();
        do {
            topTen++;
            if(myRS.getInt("msms") == 0.0) {
                myRS.last();
            } else {
                boolean different = false;
                while(!different) {
                    double msms = myRS.getDouble("msms");
                    double insilico = myRS.getDouble("insilico");
                    myRS.next();
                    if(msms == myRS.getDouble("msms") | insilico ==
myRS.getDouble("insilico")) {
                        while(msms == myRS.getDouble("msms") | insilico ==
myRS.getDouble("insilico")) {
                            myRS.next();
                        }
                    } else {
                        myRS.previous();
                        different = true;
                    }
                }
                float dx = myRS.getFloat("msms");
                dx = (dx - intA) / iteration;
                int y = Math.round(myRS.getInt("intensity"));
                y = y * step;
                int x = Math.round(dx);
                biCtx.setColor(Color.RED);
                biCtx.drawLine((60 + x),(bottom - y), (60 + x),bottom);
                if (topTen <= topWrite) {
                    String mass = myRS.getString("msms") + ", " +
myRS.getString("insilico");
                    int ofset = mass.length() / 2 * 7;
                    if (ofset % 2 !=0) {ofset+=3;}
                    biCtx.setColor(Color.BLACK);
                    biCtx.drawString(mass, 60 + x - ofset, (bottom -y) -
18);
                    String type = myRS.getString("iontype") + ", " +
myRS.getString("residue");
                    biCtx.drawString(type, 60 + x - ofset, (bottom - y) -
6);
                }
            }
        } while (myRS.next());
        myRS.close();
    }
} catch (SQLException e) {
    return null;
} catch (Exception e) {
    return null;
}

```

```

        } finally {
        }
        return biCtx;
    }

    public Graphics2D gDraw(Graphics2D biCtx, java.sql.ResultSet myRS, int intA, int
    intB, int topWrite, int bottom, int step, int xlength, Color lColor) {
        float iteration = (float)(intB - intA) / xlength; //gives what value
        coordinate 1 has for the iteration

        int topTen=0;
        try {
            if (myRS.next()) {
                myRS.first();

                do {
                    topTen++;

                    float dx = myRS.getInt("mass");
                    dx = (dx - intA) / iteration;
                    int y = Math.round(myRS.getInt("intensity"));
                    y = y * step;
                    int x = Math.round(dx);
                    biCtx.setColor(lColor);
                    biCtx.drawLine((60 + x), (bottom - y), (60 + x), bottom);
                    if (topTen <= topWrite) {
                        String mass = myRS.getString("mass");
                        int offset = mass.length() / 2 * 7;
                        if (offset % 2 != 0) {offset+=3;}
                        biCtx.setColor(Color.BLACK);
                        biCtx.drawString(mass, 60 + x - offset, (bottom - y) - 10);
                    }
                } while (myRS.next());
                myRS.close();
            }
        } catch (SQLException e) {
        } catch (Exception e) {
        } finally {
        }
        return biCtx;
    }

    public Graphics2D gDrawTitle(Graphics2D biCtx, java.sql.ResultSet myRS, int
    xlength) {
        try {
            if (myRS.next()) {
                myRS.first();
                String myTitle = myRS.getString(1) + "    Date: " +
                myRS.getString(2) + "    Target Spot Number: " + myRS.getString(3);
                biCtx.drawString(myTitle, (xlength/2) - ((myTitle.length()/2)*6) ,
                10) ;
            }
            myRS.close();
        } catch (SQLException e) {
        } catch (Exception e) {
        }
    }

```



```

        return biCtx;
    }

    public Graphics2D gDrawTitle(Graphics2D biCtx, java.sql.ResultSet myRS, int
xlength, int yPlac) {
        try {
            if (myRS.next()) {
                myRS.first();
                String myTitle = myRS.getString(1) + "    Date: " +
myRS.getString(2) + "    Target Spot Number: " + myRS.getString(3);
                biCtx.drawString(myTitle, (xlength/2) - ((myTitle.length()/2)*6) ,
yPlac) ;
            }
            myRS.close();
        } catch (SQLException e) {
        } catch (Exception e) {
        }
    }

    return biCtx;
}

    public Graphics2D gDrawMSMSTitle(Graphics2D biCtx, java.sql.ResultSet myRS, int
xlength) {
        try {
            if (myRS.next()) {
                myRS.first();
                String myTitle = myRS.getString(1) + "    Date: " +
myRS.getString(2) + "    Protein: " + myRS.getString(3);
                biCtx.drawString(myTitle, (xlength/2) - ((myTitle.length()/2)*6) ,
10) ;
            }
            myRS.close();
        } catch (SQLException e) {
        } catch (Exception e) {
        }
    }

    return biCtx;
}

    public Graphics2D gDrawMSMSTitleAA(Graphics2D biCtx, java.sql.ResultSet myRS,
int xlength) {
        try {
            if (myRS.next()) {
                myRS.first();
                String myTitle = myRS.getString(1) + "    Date: " +
myRS.getString(2) + "    Protein: " + myRS.getString(3);
                biCtx.drawString(myTitle, (xlength/2) - ((myTitle.length()/2)*6) ,
10) ;

                Font font = new Font("ARIAL", Font.BOLD , 10);
                biCtx.setFont(font);
                String aa = myRS.getString(7);
                String aa1 = aa.substring(0, (aa.length()-1)/2);
                String aa2 = aa.substring(((aa.length()-1)/2), aa.length());

                biCtx.drawString(aa1, xlength/2 - (aa1.length() * 6 / 2), 22);
                biCtx.drawString(aa2, xlength/2 - (aa2.length() * 6 / 2), 34);
            }
        }
    }

```

```

        font = new Font("ARIAL", Font.BOLD , 12);
        biCtx.setFont(font);

    }
    myRS.close();
} catch (SQLException e) {
} catch (Exception e) {
}

    return biCtx;
}

public Graphics2D gDrawMSMSTitle(Graphics2D biCtx, java.sql.ResultSet myRS, int
xlength, int yPlac) {
    try {
        if (myRS.next()) {
            myRS.first();
            String myTitle = myRS.getString(1) + "    Date: " +
myRS.getString(2) + "    Protein: " + myRS.getString(3);
            biCtx.drawString(myTitle, (xlength/2) - ((myTitle.length()/2)*6) ,
yPlac) ;
        }
        myRS.close();
    } catch (SQLException e) {
    } catch (Exception e) {
    }

    return biCtx;
}

public Graphics2D mirroredGraphs(BufferedImage bImg, dbAccess myDB, int intA,
int intB, int maldiid1, int maldiid2, int topnum, int xSize, int ySize, int spacing)
{
    Graphics2D mGraph = bImg.createGraphics();

    //Set Antialiasing on
    mGraph.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

    Font font = new Font("ARIAL", Font.BOLD, 12);
    mGraph.setFont(font);
    //Get the length and width of the border
    int x, y;
    x = xSize - spacing;
    y = ySize - spacing;

    //Set the background color and clear the space
    mGraph.setBackground(Color.WHITE);
    mGraph.clearRect(0, 0, xSize, ySize);

    //Get the first MALDI-TOF Data
    myDB.dbConnect();

    double singleItera = (double) (intB - intA) / x;
    double intItera = (y/200.0);
    try {
        java.sql.ResultSet myRS =
myDB.getMaldiGraphData(Integer.toString(maldiid1), intA, intB);
        int z = 0;

```

```

        while (myRS.next()) {
            int x1 = (spacing/2) + (int) ((myRS.getInt("mass") - intA) /
singleItera + .5);
            int y2 = (int) (intItera * (myRS.getInt("intensity")) + .5);
            mGraph.setColor(Color.RED);
            mGraph.drawLine(x1, ySize / 2, x1, ySize / 2 - y2);
            z++;
        }
        myRS.close();
    } catch (SQLException e) {
        //This shouldnt happen if it does there is nothing to do
    }

    //Get the second malditofdata
    try {
        java.sql.ResultSet myRS2 =
myDB.getMaldiGraphData(Integer.toString(maldiid2), intA, intB);
        int z = 0;

        while (myRS2.next()) {
            int x1 = (spacing/2) + (int) ((myRS2.getInt("mass") - intA) /
singleItera + .5);
            int y2 = (int) (intItera * (myRS2.getInt("intensity")) + .5);
            mGraph.setColor(Color.RED);
            mGraph.drawLine(x1, ySize / 2, x1, ySize / 2 + y2);
            z++;
        }
        myRS2.close();
    } catch (SQLException e) {
        //This shouldnt happen if it does there is nothing to do
    }

    myDB.dbClose();

    //Draw the labels and tick marks onn y axis
    //get the iterations half the total size of the rect area divided by 100
    double itera = (y / 200.0);
    mGraph.setColor(Color.BLACK);
    for (int i = 0; i <= 100; i++) {
        if (i % 10 == 0) {
            //Every ten steps draw a label and a larger tick mark

            int x1, x2;
            x1 = (spacing/2);
            x2 = (spacing/2) - 5;

            int y1, y2;
            y1 = (ySize / 2) + (int) ((itera*i) + .5);
            y2 = (ySize / 2) - (int)((itera*i) + .5);
            //Draw left
            mGraph.drawLine(x1, y1, x2, y1);
            mGraph.drawString(Integer.toString(i), x2 - 25, y1 + 5);

            mGraph.drawLine(x1, y2, x1 - 5, y2);
            mGraph.drawString(Integer.toString(i), x2 - 25, y2 + 5);

            //Draw Right
            mGraph.drawLine(xSize - x1, y1, xSize - x1 + 5, y1);
            mGraph.drawString(Integer.toString(i), xSize - spacing/2 + 10, y1 +
5);

            mGraph.drawLine(xSize - x1, y2, xSize - spacing/2 + 5, y2);
            mGraph.drawString(Integer.toString(i), xSize - spacing/2 + 10, y2 +

```

```

5);

    } else {
        int x1,x2;
        x1 = (spacing/2);
        x2 = (spacing/2) - 2;

        int y1, y2;
        y1 = (ysize /2) + (int) ((itera*i) + .5);
        y2 = (ysize /2) - (int) ((itera*i) + .5);

        //draw left
        mGraph.drawLine(x1, y1, x2, y1);
        mGraph.drawLine(x1, y2, x2, y2);

        //Draw Right
        mGraph.drawLine(xSize - x1, y1, xSize - x1 + 2, y1);
        mGraph.drawLine(xSize - x1, y2, xSize - spacing/2 + 2, y2);

    }
}

//Draw Labels for x axis
double iterate = x / 25.0;
double intIter = (intB - intA) / 5;
for (int i =0; i <=25; i ++) {
    if (i % 5 == 0) {

        int x1;
        x1 = (spacing/2 + (int) (i*iterate + .5));

        //Draw labels and large tick marks
        //Draw bottom
        mGraph.drawLine(x1, ySize - (spacing/2), x1, ySize - (spacing/2) +
10 );

        int lIntensity = intA + ((int) ((i / 5)*intIter + .5));
        String lbl = Integer.toString(lIntensity);
        mGraph.drawString(lbl, x1 - lbl.length() * 3, ySize - spacing/2 +
25) ;

        //Draw top
        mGraph.drawLine(x1, (spacing/2), x1, (spacing/2) - 10 );
        mGraph.drawString(lbl, x1 - lbl.length() * 3, spacing/2 - 16) ;

    } else {

        int x1;
        x1 = (spacing/2 + (int) (i*iterate + .5));

        //draw bottom
        mGraph.drawLine(x1, ySize - (spacing/2), x1, ySize - (spacing/2) + 5
);

        //draw top
        mGraph.drawLine(x1, (spacing/2), x1, (spacing/2) - 5 );

    }
}

//Draw the border
mGraph.setColor(Color.BLACK);
mGraph.drawRect(spacing/2, spacing/2, x,y);

```

```

mGraph.drawLine((spacing/2), (ysize/2), xSize - (spacing/2), (ysize/2));

//Draw title
myDB.dbConnect();
String[] m1 = myDB.getMaldiSampleInfo(Integer.toString(maldiid1));
String[] m2 = myDB.getMaldiSampleInfo(Integer.toString(maldiid2));
mGraph.drawString("Combined Graph of " + m1[0] + " And " + m2[0], xSize / 2
-70, 10);
myDB.dbClose();

//Draw Axis labels
Font titleFont = new Font("ARIAL", Font.BOLD, 14);
mGraph.setFont(titleFont);

mGraph.drawString("Mass to Charge (M/Z)", xSize/2 - 70, ySize - 5);
FontMetrics fm = mGraph.getFontMetrics();
mGraph.rotate(-Math.PI / 2, 10, (ysize + fm.getHeight())/2);
mGraph.drawString("% Intensity", 10, (ysize + fm.getHeight())/2);

return mGraph;
}

public Graphics2D mirroredMSMSGraphs(BufferedImage bImg, dbAccess myDB, int
intA, int intB, int maldiid1, int maldiid2, int topnum, int xSize, int ySize, int
spacing) {
    Graphics2D mGraph = bImg.createGraphics();

    //Set Antialiasing on
    mGraph.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

    Font font = new Font("ARIAL", Font.BOLD, 12);
    mGraph.setFont(font);
    //Get the length and width of the border
    int x, y;
    x = xSize - spacing;
    y = ySize - spacing;

    //Set the background color and clear the space
    mGraph.setBackground(Color.WHITE);
    mGraph.clearRect(0, 0, xSize, ySize);

    //Get the first MALDI-TOF Data
    myDB.dbConnect();

    double singleItera = (double) (intB - intA) / x;
    double intItera = (y/200.0);
    try {
        java.sql.ResultSet myRS =
myDB.getMSMSGraphData(Integer.toString(maldiid1), intA, intB);
        int z = 0;

        while (myRS.next()) {
            int x1 = (spacing/2) + (int) ((myRS.getInt("mass") - intA) /
singleItera + .5);
            int y2 = (int) (intItera * (myRS.getInt("intensity")) + .5);
            mGraph.setColor(Color.RED);
            mGraph.drawLine(x1, ySize / 2, x1, ySize / 2 - y2);
            z++;
        }
        myRS.close();
    } catch (SQLException e) {

```

```

    } //This shouldnt happen if it does there is nothing to do

    //Get the second malditofdata
    try {
        java.sql.ResultSet myRS2 =
myDB.getMMSGraphData(Integer.toString(maldiid2), intA, intB);
        int z = 0;

        while (myRS2.next()) {
            int x1 = (spacing/2) + (int) ((myRS2.getInt("mass") - intA) /
singleItera + .5);
            int y2 = (int) (intItera * (myRS2.getInt("intensity")) + .5);
            mGraph.setColor(Color.RED);
            mGraph.drawLine(x1, ySize / 2, x1, ySize / 2 + y2);
            z++;
        }
        myRS2.close();
    } catch (SQLException e) {
        //This shouldnt happen if it does there is nothing to do
    }

    myDB.dbClose();

    //Draw the labels and tick marks onn y axis
    //get the iterations half the total size of the rect area divided by 100
    double itera = (y / 200.0);
    mGraph.setColor(Color.BLACK);
    for (int i = 0; i <= 100; i++) {
        if (i % 10 == 0) {
            //Every ten steps draw a label and a larger tick mark

            int x1, x2;
            x1 = (spacing/2);
            x2 = (spacing/2) - 5;

            int y1, y2;
            y1 = (ySize / 2) + (int) ((itera*i) + .5);
            y2 = (ySize / 2) - (int) ((itera*i) + .5);
            //Draw left
            mGraph.drawLine(x1, y1, x2, y1);
            mGraph.drawString(Integer.toString(i), x2 - 25, y1 + 5);

            mGraph.drawLine(x1, y2, x1 - 5, y2);
            mGraph.drawString(Integer.toString(i), x2 - 25, y2 + 5);

            //Draw Right
            mGraph.drawLine(xSize - x1, y1, xSize - x1 + 5, y1);
            mGraph.drawString(Integer.toString(i), xSize - spacing/2 + 10, y1 +
5);

            mGraph.drawLine(xSize - x1, y2, xSize - spacing/2 + 5, y2);
            mGraph.drawString(Integer.toString(i), xSize - spacing/2 + 10, y2 +
5);

        } else {
            int x1, x2;
            x1 = (spacing/2);
            x2 = (spacing/2) - 2;

            int y1, y2;
            y1 = (ySize / 2) + (int) ((itera*i) + .5);
            y2 = (ySize / 2) - (int) ((itera*i) + .5);

```

```

        //draw left
        mGraph.drawLine(x1, y1, x2, y1);
        mGraph.drawLine(x1, y2, x2, y2);

        //Draw Right
        mGraph.drawLine(xSize - x1, y1, xSize - x1 + 2, y1);
        mGraph.drawLine(xSize - x1, y2, xSize - spacing/2 + 2, y2);

    }
}

//Draw Labels for x axis
double iterate = x / 25.0;
double intIter = (intB - intA) / 5;
for (int i = 0; i <= 25; i++) {
    if (i % 5 == 0) {
        int x1;
        x1 = (spacing/2 + (int) (i*iterate + .5));

        //Draw labels and large tick marks
        //Draw bottom
        mGraph.drawLine(x1, ySize - (spacing/2), x1, ySize - (spacing/2) +
10 );

        int lIntensity = intA + ((int) ((i / 5)*intIter + .5));
        String lbl = Integer.toString(lIntensity);
        mGraph.drawString(lbl, x1 - lbl.length() * 3, ySize - spacing/2 +
25) ;

        //Draw top
        mGraph.drawLine(x1, (spacing/2), x1, (spacing/2) - 10 );
        mGraph.drawString(lbl, x1 - lbl.length() * 3, spacing/2 - 16) ;

    } else {
        int x1;
        x1 = (spacing/2 + (int) (i*iterate + .5));

        //draw bottom
        mGraph.drawLine(x1, ySize - (spacing/2), x1, ySize - (spacing/2) + 5
);

        //draw top
        mGraph.drawLine(x1, (spacing/2), x1, (spacing/2) - 5 );
    }
}

//Draw the border
mGraph.setColor(Color.BLACK);
mGraph.drawRect(spacing/2, spacing/2, x,y);
mGraph.drawLine((spacing/2), (ySize/2), xSize - (spacing/2), (ySize/2));

//Draw Axis labels
Font titleFont = new Font("ARIAL", Font.BOLD, 14);
mGraph.setFont(titleFont);

mGraph.drawString("Mass to Charge (M/Z)", xSize/2 - 70, ySize - 5);
FontMetrics fm = mGraph.getFontMetrics();
mGraph.rotate(-Math.PI / 2, 10, (ySize + fm.getHeight())/2);
mGraph.drawString("% Intensity", 10, (ySize + fm.getHeight())/2);

```

```
        titleFont = new Font("ARIAL", Font.PLAIN, 10);  
        mGraph.setFont(titleFont);  
        mGraph.rotate(+Math.PI / 2, 10, (ySize + fm.getHeight())/2);  
        return mGraph;  
    }  
}
```



```

/*
 * scatterPlot.java
 *
 * Created on August 24, 2005, 10:35 AM
 */

package psm.graph;

import java.io.OutputStream;
import java.beans.*;
import java.io.Serializable;
import java.awt.*;
import com.sun.image.codec.jpeg.*;
import java.awt.image.BufferedImage;
import java.sql.*;
import psm.db.dbAccess;
import psm.db.box.masscolorbox;
/**
 * @author Brandon Garbus
 */
public class scatterPlot {

    public scatterPlot() {

        public Graphics2D maldiScatter(dbAccess myDB, BufferedImage bImg, String[]
        maldiids, int xSize, int ySize, int lRange, int hRange, int fontSize, int spacing,
        int topnum, int err, int intensity) {

            //Create the set of colors for the highlighted regions
            int alpha = 60;
            Color[] highlighter = new Color[17];
            highlighter[0] = new Color(215,25,25, alpha);
            highlighter[1] = new Color(215,25,130, alpha);
            highlighter[2] = new Color(215,25,190, alpha);
            highlighter[3] = new Color(155,25,215, alpha);
            highlighter[4] = new Color(25,120,215, alpha);
            highlighter[5] = new Color(25,175,215, alpha);
            highlighter[6] = new Color(25,215,180, alpha);
            highlighter[7] = new Color(25,215,137, alpha);
            highlighter[8] = new Color(25,215,85, alpha);
            highlighter[9] = new Color(195,215,25, alpha);
            highlighter[10] = new Color(215,160,25, alpha);
            highlighter[11] = new Color(195,35,130, alpha);
            highlighter[12] = new Color(215,80,25, alpha);
            highlighter[13] = new Color(195,215,25, alpha);
            highlighter[14] = new Color(35,180,195, alpha);
            highlighter[15] = new Color(215,130,25, alpha);
            highlighter[16] = new Color(40,170,50, alpha);

            //This array is used to store highlighted mass regions
            masscolorbox[] mBox = new masscolorbox[topnum];

            //xSize and ySize are the size of the image, x and y are the size of the
plot

            Graphics2D plot = bImg.createGraphics();
            plot.setBackground(Color.WHITE);
            plot.clearRect(0,0,xSize,ySize);

            //Set rendering hints
            plot.setRenderingHint(RenderingHints.KEY_ANTIALIASING,

```

```

RenderingHints.VALUE_ANTIALIAS_ON);

//Set the size of the graph to make room for labels and iterations
int x,y;
x = xSize - spacing;
y = ySize - spacing;

//Set the font
Font font = new Font("ARIAL", Font.BOLD , fontSize);
plot.setFont(font);

plot.setColor(Color.BLACK);
//Draw the plot centered
plot.draw3DRect( (spacing/2), (spacing/2), x, y, true);

//This helps keep track and make sure that the select all checkbox isnt in
the list when we go get stuff
int l;
if (maldiids[0].compareTo("on") == 0) {
    l=1;
} else {
    l =0;
}

//Draw y iterations
double yIter = (double) y / (maldiids.length -l + 1);

for (int i = 1; i <= maldiids.length -l; i++) {
    int y1 = ySize - (spacing/2);
    y1 -= Math rint(yIter * i);
    //Draw the iteration marks on the left for the RM numbers
    plot.setColor(Color.BLACK);
    plot.drawLine((spacing/2), y1 , (spacing/2) - 5, y1);

    //Draw the guidelines inside the plot
    plot.setColor(Color.LIGHT_GRAY);
    plot.drawLine((spacing/2), y1 , xSize - (spacing/2), y1);
}

//Draw the x iterations and labels
double mIter = (double) (hRange - lRange) / 5;
double xIter = (double) (x / 10);

for (int i = 0; i <=10; i ++) {
    int x1 = (spacing/2);
    x1 += (int) xIter * i;

    if (i % 2 ==0) {
        //Draw the tick marks
        plot.setColor(Color.BLACK);
        plot.drawLine( x1, ySize - (spacing/2), x1, ySize - (spacing/2) +
10);

        //Draw the label
        int lblInt = (int) ((lRange + mIter * i/2));
        String xLabel = Integer.toString(lblInt);
        int x2 = x1;
        x2 -= (int) Math.rint(xLabel.length() * 3.5);
        plot.drawString(xLabel, x2, ySize - (spacing/4));
    } else {
        //draw smaller tick marks
        plot.drawLine( x1, ySize - (spacing/2), x1, ySize - (spacing/2) +
5);
    }
}
}

```

```

//Cycle through the maldi samples and add them to the plot
//Connect to the database
//counter
int z = 0;
//this counts how many topnumbers were actualyl highlighted
int totalcount = 0;

myDB.dbConnect();

//write the title
Font fontTitle = new Font("ARIAL", Font.BOLD , fontSize + 4);
plot.setFont(fontTitle);

//l (lower case "l")being the first maldi sample
java.sql.ResultSet myTitleRS = myDB.getMaldiAndSampleInfo(maldiids[l]);
String title;
try {
    myTitleRS.next();
    title = "Comparison Scatter Plot for Strain: " + myTitleRS.getString(1)
+ " Sample: " + myTitleRS.getString(2);
} catch (SQLException e) {
    title= "Comparison Scatter Plot";
}
plot.drawString(title, (xSize/2) - title.length() * 4, 15);
//set the font back to the original
plot.setFont(font);

//Draw Labels
plot.drawString("Mass to Charge (M/Z)", (xSize/2) - 60, ySize - 15);

//This keeps the RM number of the cpared to sample
String rmNumberCmp = "";
String rmName = "";
for (int i =1; i < maldiids.length; i++) {
    try {
        //get the y coordinate
        int y1 = ySize - (spacing/2);
        y1 -= Math rint(yIter * (i-1 + 1));
        String[] rmNum = myDB.getMaldiSampleInfo(maldiids[i]);
        plot.setColor(Color.BLACK);
        Font fontLabel = new Font("ARIAL", Font.BOLD , fontSize - 4);
        plot.setFont(fontLabel);
        plot.drawString(rmNum[0], 0, y1 + 3);
        ResultSet myRS = myDB.getMaldiGraphData(maldiids[i], lRange,
hRange);
        plot.setFont(font);
        if (i==1) {
            rmNumberCmp = rmNum[0];
            rmName = rmNum[1];
            plot.setColor(highlighter[15]);
            plot.fillRect(0, y1 - 3, xSize - (spacing/2), (y1 + 3) - (y1 -
3));
        } else if (rmNumberCmp.compareTo(rmNum[0]) == 0) {
            plot.setColor(highlighter[15]);
            plot.fillRect(0, y1 - 3, xSize - (spacing/2), (y1 + 3) - (y1 -
3));
        } else if (rmName.compareTo(rmNum[1])==0) {
            plot.setColor(highlighter[16]);
            plot.fillRect(0, y1 - 3, xSize - (spacing/2), (y1 + 3) - (y1 -
3));
        }
        while (myRS.next()) {
            //normally checking the intensty would be done i nthe select
statement however since the statement first normalizes the data
//its easier to do it here.

```

```

intensity) {
    if (myRS.getInt("mass") != 0 && myRS.getInt("intensity") >=
        //This gets the increment for 1 mass/charge unit
        double massIter = (double) (hRange - lRange) / x;
        int x1;
        double x0;
        //Get the x coordiante
        x0 = (myRS.getInt("mass") - lRange) / massIter;
        x1 = (int) x0;
        x1 += (spacing/2);
        //Draw the point on the plot
        int radius = 2;
        plot.setColor(new Color(15,15,165));
        plot.drawOval(x1, y1 - (radius/2), radius, radius);
        plot.fillOval(x1, y1 - (radius/2), radius, radius);
        //Draw highlighted error regions
        if (z < topnum) {
            //select z mod 15 for automatic rollover
            plot.setColor(highlighter[z % 15]);
            mBox[z] = new masscolorbox();
            mBox[z].setMass(myRS.getInt("mass"));
            mBox[z].setHighlighted(highlighter[z%15]);
            double x1Hd, x2Hd;
            double error = (err / 1000000.0) * myRS.getInt("mass");
            x1Hd = (double) ( myRS.getInt("mass") - error - lRange)
            / massIter;
            x2Hd = (double) ( myRS.getInt("mass") + error - lRange)
            / massIter;
            //Round
            x1Hd += 0.5;
            x2Hd += 0.5;
            int x1H, x2H;
            x1H = (int) x1Hd;
            x2H = (int) x2Hd;
            x1H += (spacing/2);
            x2H += (spacing/2);
            plot.fillRect(x1H, spacing / 2, x2H - x1H, ySize -
spacing);
            //plot.drawRect(x1H, spacing / 2, x2H - x1H, ySize -
spacing);
            totalcount++;
        }
        z++;
    }
    //set z = to the topnumber in case there are less points inthe first
sample than the topnumber
    z = topnum;
    myRS.close();
} catch (SQLException e) {
}

}

//Build the legend for the highlighted colors
//first sort the array from least mass to greatest mass
//Bubble sort is simple and fast enough for this
for (int i=0; i < totalcount; i++) {
    for (int j =0; j < totalcount -1; j++) {
        if (mBox[j + 1].getMass() < mBox[j].getMass()) {
            masscolorbox tmp = mBox[j];
            mBox[j] = mBox[j+1];

```

```

        mBox[j+1] = tmp;
    }
}

for (int i=0; i < totalcount; i++){
    plot.setColor(mBox[i].getHighlighted());
    plot.fillRect(xSize - 50, (spacing / 2) + i * 15, 10, 5);
    plot.setColor(Color.BLACK);
    plot.drawString(Integer.toString(mBox[i].getMass()), xSize - 37,
(spacing/2) + 7 + i*15);
}

//close the database
myDB.dbClose();
return plot;
}

public Graphics2D MSMSScatter(dbAccess myDB, BufferedImage bImg, String[]
maldiids, int xSize, int ySize, int lRange, int hRange, int fontSize, int spacing,
int topnum, double err, int intensity) {

    //Create the set of colors for the highlighted regions
    int alpha = 60;
    Color[] highlighter = new Color[17];
    highlighter[0] = new Color(215,25,25, alpha);
    highlighter[1] = new Color(215,25,130, alpha);
    highlighter[2] = new Color(215,25,190, alpha);
    highlighter[3] = new Color(155,25,215, alpha);
    highlighter[4] = new Color(25,120,215, alpha);
    highlighter[5] = new Color(25,175,215, alpha);
    highlighter[6] = new Color(25,215,180, alpha);
    highlighter[7] = new Color(25,215,137, alpha);
    highlighter[8] = new Color(25,215,85, alpha);
    highlighter[9] = new Color(195,215,25, alpha);
    highlighter[10] = new Color(215,160,25, alpha);
    highlighter[11] = new Color(195,35,130, alpha);
    highlighter[12] = new Color(215,80,25, alpha);
    highlighter[13] = new Color(195,215,25, alpha);
    highlighter[14] = new Color(35,180,195, alpha);
    highlighter[15] = new Color(215,130,25, alpha);
    highlighter[16] = new Color(40,170,50, alpha);

    //This array is used to store highlighted mass regions
    masscolorbox[] mBox = new masscolorbox[topnum];

    //xSize and ySize are the size of the image, x and y are the size of the
plot

    Graphics2D plot = bImg.createGraphics();
    plot.setBackground(Color.WHITE);
    plot.clearRect(0,0,xSize,ySize);

    //Set rendering hints
    plot.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

    //Set the size of the graph to make room for labels and iterations
    int x,y;
    x = xSize - spacing;
    y = ySize - spacing;

    //Set the font

```

```

Font font = new Font("ARIAL", Font.BOLD , fontSize);
plot.setFont(font);

plot.setColor(Color.BLACK);
//Draw the plot centered
plot.draw3DRect( (spacing/2), (spacing/2), x, y, true);

//This helps keep track and make sure that the select all checkbox isnt in
the list when we go get stuff
int l;
if (maldiids[0].compareTo("on") == 0) {
    l=1;
} else {
    l =0;
}

//Draw y iterations
double yIter = (double) y / (maldiids.length -l + 1);

for (int i = 1; i <= maldiids.length -l; i++) {
    int y1 = ySize - (spacing/2);
    y1 -= Math rint(yIter * i);
    //Draw the iteration marks on the left for the RM numbers
    plot.setColor(Color.BLACK);
    plot.drawLine((spacing/2), y1 , (spacing/2) - 5, y1);

    //Draw the guidelines inside the plot
    plot.setColor(Color.LIGHT_GRAY);
    plot.drawLine((spacing/2), y1 , xSize - (spacing/2), y1);
}

//Draw the x iterations and labels
double mIter = (double) (hRange - lRange) / 5;
double xIter = (double) (x / 10);

for (int i = 0; i <=10; i ++) {
    int x1 = (spacing/2);
    x1 += (int) xIter * i;

    if (i % 2 ==0) {
        //Draw the tick marks
        plot.setColor(Color.BLACK);
        plot.drawLine( x1, ySize - (spacing/2), x1, ySize - (spacing/2) +
10);

        //Draw the label
        int lblInt = (int) ((lRange + mIter * i/2));
        String xLabel = Integer.toString(lblInt);
        int x2 = x1;
        x2 -= (int) Math.rint(xLabel.length() * 3.5);
        plot.drawString(xLabel, x2, ySize - (spacing/4));
    } else {
        //draw smaller tick marks
        plot.drawLine( x1, ySize - (spacing/2), x1, ySize - (spacing/2) +
5);
    }
}

//Cycle through the maldi samples and add them to the plot
//Connect to the database
//counter
int z = 0;
//this counts how many topnumbers were actualyl highlighted
int totalcount = 0;

myDB.dbConnect();

```

```

//write the title
Font fontTitle = new Font("ARIAL", Font.BOLD , fontSize + 4);
plot.setFont(fontTitle);

//l (lower case "L")being the first maldi sample
java.sql.ResultSet myTitleRS = myDB.getMSSMSAndSampleInfo(maldiids[1]);
String title;
try {
    myTitleRS.next();
    title = "Comparison Scatter Plot for Strain: " + myTitleRS.getString(1)
+ " Sample: " + myTitleRS.getString(2);
} catch (SQLException e) {
    title= "Comparison Scatter Plot";
}

plot.drawString(title, (xSize/2) - title.length() * 4, 15);
//set the font back to the original
plot.setFont(font);

//Draw Labels
plot.drawString("Mass to Charge (M/Z)", (xSize/2) - 60, ySize - 15);

//This keeps the RM number of the cpared to sample
String rmNumberCmp = "";
String rmName = "";
for (int i =1; i < maldiids.length; i++ ) {
    try {
        //get the y coordinate
        int y1 = ySize - (spacing/2);
        y1 -= Math rint(yIter * (i-1 + 1));
        String[] rmNum = myDB.getMSSMSSampleInfo(maldiids[i]);
        plot.setColor(Color.BLACK);
        Font fontLabel = new Font("ARIAL", Font.BOLD , fontSize - 4);
        plot.setFont(fontLabel);
        plot.drawString(rmNum[0], 0, y1 + 3);
        ResultSet myRS = myDB.getMSSMSGraphData(maldiids[i], lRange, hRange);
        plot.setFont(font);
        if (i==1) {
            rmNumberCmp = rmNum[0];
            rmName = rmNum[1];
            plot.setColor(highlighter[15]);
            plot.fillRect(0, y1 - 3, xSize - (spacing/2), (y1 + 3) - (y1 -
3));
        } else if (rmNumberCmp.compareTo(rmNum[0]) == 0) {
            plot.setColor(highlighter[15]);
            plot.fillRect(0, y1 - 3, xSize - (spacing/2), (y1 + 3) - (y1 -
3));
        } else if (rmName.compareTo(rmNum[1])==0) {
            plot.setColor(highlighter[16]);
            plot.fillRect(0, y1 - 3, xSize - (spacing/2), (y1 + 3) - (y1 -
3));
        }
        while (myRS.next()) {
            //normally checking the intensty would be done i nthe select
            statement however since the statement first normalizes the data
            //its easier to do it here.
            if (myRS.getInt("mass") != 0 && myRS.getInt("intensity") >=
intensity) {
                //This gets the increment for 1 mass/charge unit
                double massIter = (double) (hRange - lRange) / x;
                int x1;
                double x0;
                //Get the x coordiante

```

```

x0 = (myRS.getInt("mass") - lRange) / massIter;
x1 = (int) x0;
x1 += (spacing/2);
//Draw the point on the plot
int radius = 2;
plot.setColor(new Color(15,15,165));
plot.drawOval(x1, y1 - (radius/2), radius, radius);
plot.fillOval(x1, y1 - (radius/2), radius, radius);
//Draw highlighted error regions
if (z < topnum) {
    //select z mod 15 for automatic rollover
    plot.setColor(highlighter[z % 15]);
    mBox[z] = new masscolorbox();
    mBox[z].setMass(myRS.getInt("mass"));
    mBox[z].setHighlighted(highlighter[z%15]);
    double x1Hd, x2Hd;
    double error = err;
    x1Hd = (double) ( myRS.getInt("mass") - error - lRange)
/ massIter;
    x2Hd = (double) ( myRS.getInt("mass") + error - lRange)
/ massIter;

    //Round
    x1Hd += 0.5;
    x2Hd += 0.5;
    int x1H, x2H;
    x1H = (int) x1Hd;
    x2H = (int) x2Hd;
    x1H += (spacing/2);
    x2H += (spacing/2);
    plot.fillRect(x1H, spacing / 2, x2H - x1H, ySize -
spacing);
    //plot.drawRect(x1H, spacing / 2, x2H - x1H, ySize -
spacing);
    totalcount++;
}
z++;
}

}
//set z = to the topnumber in case there are less points inthe first
sample than the topnumber
z = topnum;
myRS.close();
} catch (SQLException e) {
}

}

//Build the legend for the highlighted colors
//first sort the array from least mass to greatest mass
//Bubble sort is simple and fast enough for this
for (int i=0; i < totalcount; i++) {
    for (int j =0; j < totalcount -1; j++) {
        if (mBox[j + 1].getMass() < mBox[j].getMass()) {
            masscolorbox tmp = mBox[j];
            mBox[j] = mBox[j+1];
            mBox[j+1] = tmp;
        }
    }
}

for (int i=0; i < totalcount; i++){
    plot.setColor(mBox[i].getHighlighted());

```



```
        plot.fillRect(xSize - 50, (spacing / 2) + i * 15, 10, 5);
        plot.setColor(Color.BLACK);
        plot.drawString(Integer.toString(mBox[i].getMass()), xSize - 37,
(spacing/2) + 7 + i*15);
    }

    //close the database
    myDB.dbClose();
    return plot;
}

}
```

```

/*
 * formatMALDI.java
 *
 * Created on June 30, 2005, 2:45 PM
 */

package psm.mformat;

import java.beans.*;
import java.io.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletInputStream;
import java.util.*;
import java.text.SimpleDateFormat;
import java.text.DecimalFormat;

/**
 * @author Brandon
 */
public class formatinsilico {
    private ArrayList maldiMZ;
    private String number;
    private String notes;
    private String fileName;
    private double MW;
    private String protein;
    private String Genus;
    private String Species;
    private String subspecies;
    private String subtype;

    public formatinsilico() {
        setMaldiMZ(new ArrayList());
    }

    public int formatMALDIFile(HttpServletRequest request) throws IOException {
        setMaldiMZ(new ArrayList());
        setNumber("");
        setNotes("");
        setGenus("");
        setSpecies("");
        setSubspecies("");
        setSubtype("");
        ServletInputStream inStr = request.getInputStream();
        byte[] line = new byte[1024];

        int i = inStr.readLine(line, 0, 1024);

        //The length of the boundary should be greater than three, anything less
means an error has occurred, so exit.
        if (i < 3) {
            return 1;
        }

        //Boundary length is the first line minus the first two CR's that terminate
it
        int boundaryLength = i - 2;
        //Retrieves the boundary minus the two CR's
        String boundary = new String(line, 0, boundaryLength);
        setFileName(null);
        String tempStr = "";

        boolean isFile = false;
        while (i != -1) {

```

```

//Store the current line as a string
String newLine = new String(line, 0, i);
if (!isFile) {
    //All we care about is the file so loop through all the fields until
you hit this.
    if (newLine.indexOf("filename=\"") != -1) {
        setFileName(new String(line, newLine.indexOf("filename=\"") +
10, newLine.indexOf("\"", newLine.indexOf("filename=\""))));
        setFileName(getFileName().replace("\"", ""));
        setFileName(getFileName().trim());
        isFile= true;
    }
} else {
    //if no filename was entered exit
    if (getFileName()==null) {
        return 2;
    }
    //Do not read the very last line
    if (!newLine.startsWith(boundary) & newLine.indexOf("Content") ==
-1) {
        //This is where reading of the lines of the file occurs.
        int lCount = 0; //keeps track of the line count
        while (i != -1 && !newLine.startsWith(boundary)) {
            lCount ++; //Increment the line count indicating the current
position relative to the file contents
            i = inStr.readLine(line, 0, 1024);

            if (lCount > 1) {
                //if lCount is greater than 11 then we are now on the
data lines and can start processing that data
                //Columns are seperated by a series of spaces, and the
peak numbers have a width of a minimum of 3 characters, and both Mass and intensity
columns have
                //a decimal point, this will be used to determine the
position of the numbers and to grab them
                if (newLine.length() < 5) { return 0;} //the last line
could possibly be a carriage return, we dont care about this, so just exit.

                //This format is going to assume a tab delimited file,
with the first line containing:
                //RM# MW Protein Notes
                //M/Z Intensity
                if (lCount == 2) {
                    //This is the first line
                    String [] fstline = null;
                    newLine = newLine.trim();
                    fstline = newLine.split("\t");

                    //The first string in fstline should be the rm
number
                    setNumber(fstline[0]);

                    //The second should be the strain
                    String [] strain = fstline[1].split(" ");
                    switch (strain.length) {
                        case 4:
                            setSubtype(strain[3]);
                        case 3:
                            setSubspecies(strain[2]);
                        case 2:
                            setSpecies(strain[1]);
                        case 1:
                            setGenus(strain[0]);
                    }
                }
            }
        }
    }
}

```

```

    }
    //The third should be the MW weight
    setMW(Double.parseDouble(fstline[2]));

    if (fstline.length > 3) {
        //This is the protein
        setProtein(fstline[3]);
    }
    if (fstline.length > 4) {
        //the rest are notes
        setNotes(fstline[4]);
    }
} else {
    try {
        //Not the first line, dump into array
        String [] dataLine = null;
        newLine = newLine.trim();
        dataLine = newLine.split("\t");

        //This should contain one set of strings, the first
        //and the second being the intensity.

        double mz = Double.parseDouble(dataLine[0]);
        getMaldiMZ().add(new Double(mz));
    }
    catch (Exception e) {
        return 9;
    }
}

}
}
/*
if ((i==boundaryLength+2 || i==boundaryLength+4 && new
String(line, 0, i).startsWith(boundary))) {
    pw.print(newLine.substring(0, newLine.length()-2));
} else {
    pw.print(newLine);
}
*/
newLine = new String(line, 0, i);
}

}
}
//iterate to next line
i = inStr.readLine(line, 0, 1024);
}

return 0;
}

public String getNumber() {
    return number;
}

public void setNumber(String number) {
    this.number = number;
}

public String getNotes() {
    return notes;
}

```

```
public void setNotes(String notes) {
    this.notes = notes;
}

public ArrayList getMaldiMZ() {
    return maldiMZ;
}

public void setMaldiMZ(ArrayList maldiMZ) {
    this.maldiMZ = maldiMZ;
}

public String getFileName() {
    return fileName;
}

public double getMW() {
    return MW;
}

public void setMW(double MW) {
    this.MW = MW;
}

public String getProtein() {
    return protein;
}

public void setProtein(String protein) {
    this.protein = protein;
}

public void setFileName(String fileName) {
    this.fileName = fileName;
}

public String getGenus() {
    return Genus;
}

public void setGenus(String Genus) {
    this.Genus = Genus;
}

public String getSpecies() {
    return Species;
}

public void setSpecies(String Species) {
    this.Species = Species;
}

public String getSubspecies() {
    return subspecies;
}

public void setSubspecies(String subspecies) {
    this.subspecies = subspecies;
}

public String getSubtype() {
    return subtype;
}

public void setSubtype(String subtype) {
```

```
        this.subtype = subtype;  
    }  
}
```

```

/*
 * formatMALDI.java
 *
 * Created on June 30, 2005, 2:45 PM
 */

package psm.mformat;

import java.beans.*;
import java.io.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletInputStream;
import java.util.*;
import java.text.SimpleDateFormat;
import java.text.DecimalFormat;

/**
 * @author Brandon
 */
public class formatMALDI {
    private ArrayList maldiMZ;
    private ArrayList maldiIntensity;
    private String targetNum;
    private Date mdate;
    private String number;
    private String notes;
    private String fileName;

    public formatMALDI() {
        maldiMZ = new ArrayList();
        maldiIntensity = new ArrayList();
    }

    public int formatMALDIFile(HttpServletRequest request) throws IOException {
        maldiMZ = new ArrayList();
        maldiIntensity = new ArrayList();
        mdate= null;
        number = "";
        notes = "";
        targetNum = "";

        ServletInputStream inStr = request.getInputStream();
        byte[] line = new byte[1024];

        int i = inStr.readLine(line, 0, 1024);

        //The length of the boundary should be greater than three, anything less
means an error has occurred, so exit.
        if (i < 3) {
            return 1;
        }

        //Boundary length is the first line minus the first two CR's that terminate
it
        int boundaryLength = i - 2;
        //Retrieves the boundary minus the two CR's
        String boundary = new String(line, 0, boundaryLength);
        fileName = null;
        String tempStr = "";

        boolean isFile = false;
        while (i != -1) {
            //Store the current line as a string
            String newLine = new String(line, 0, i);

```

```

        if (!isFile) {
            //All we care about is the file so loop through all the fields until
you hit this.
            if (newLine.indexOf("filename=\"") != -1) {

                fileName = new String(line, newLine.indexOf("filename=\"") + 10
, newLine.indexOf("\"", newLine.indexOf("filename=\"") + 10));
                fileName = fileName.replace("\"", "");
                fileName = fileName.trim();
                if (fileName.contains("\\")) {
                    fileName = fileName.substring(fileName.lastIndexOf("\\") +
1, fileName.length());
                }
                fileName = fileName.substring(0, fileName.length() - 4);

                String[] info = fileName.split("_");

                number = info[0];
                targetNum = info[2];
                if (info.length > 3) {
                    notes = info[3];
                }

                SimpleDateFormat df = new SimpleDateFormat("yyMMdd");
                try {
                    mdate = df.parse( info[1]);
                } catch (Exception e){
                    mdate = null;
                }
                isFile= true;
            }
        } else {
            //if no filename was entered exit
            if (fileName==null) {
                return 2;
            }
            //Do not read the very last line

            if (!newLine.startsWith(boundary) & newLine.indexOf("Content") ==
-1) {
                //This is where reading of the lines of the file occurs.
                int lCount = 0; //keeps track of the line count
                while (i != -1 && !newLine.startsWith(boundary)) {
                    lCount ++; //Increment the line count indicating the current
position relative to the file contents
                    i = inStr.readLine(line, 0, 1024);

                    //On the filename three is the sample name date and target
number, we want this information
                    if (lCount > 1) {
                        //if lCount is greater than 11 then we are now on the
data lines and can start processing that data
                        //Columns are seperated by a series of spaces, and the
peak numbers have a width of a minimum of 3 characters, and both Mass and intensity
columns have
                        //a decimal point, this will be used to determine the
position of the numbers and to grab them
                        if (newLine.length() < 5) { return 0;} //the last line
could possibly be a carriage return, we dont care about this, so just exit.

                        String[] pline = newLine.split("\t");

                        //now assign this into the mass array, in a new positon.
                        double dx = Double.parseDouble(pline[0]);

```



```

        //This is for rounding, add .5 to the decimal, and when
converted to an int the decimal drops off
        dx += .5;
        int di = (int)dx;
        maldiMZ.add(new Integer(di));

        //Change to a double and round to the first decimal
place
        dx = Double.parseDouble(pline[1]);
        dx += 0.05;
        DecimalFormat dF = new DecimalFormat("00.0");
        maldiIntensity.add(new
Double(Double.parseDouble(dF.format((dx)))));
    }
    /*
String(line, 0, i).startsWith(boundary))) {
        pw.print(newLine.substring(0, newLine.length()-2));
    } else {
        pw.print(newLine);
    }
    **/
    newLine = new String(line, 0, i);
}

    }
    }
    //iterate to next line
    i = inStr.readLine(line, 0, 1024);
}

    return 0;
}

public String getTargetNum() {
    return targetNum;
}

public void setTargetNum(String targetNum) {
    this.targetNum = targetNum;
}

public Date getMdate() {
    return mdate;
}

public void setMdate(Date mdate) {
    this.mdate = mdate;
}

public String getNumber() {
    return number;
}

public void setNumber(String number) {
    this.number = number;
}

public String getNotes() {
    return notes;
}

public void setNotes(String notes) {
    this.notes = notes;
}

```

```
public ArrayList getMaldiMZ() {  
    return maldiMZ;  
}  
  
public void setMaldiMZ(ArrayList maldiMZ) {  
    this.maldiMZ = maldiMZ;  
}  
  
public ArrayList getMaldiIntensity() {  
    return maldiIntensity;  
}  
  
public void setMaldiIntensity(ArrayList maldiIntensity) {  
    this.maldiIntensity = maldiIntensity;  
}  
  
public String getFileName() {  
    return fileName;  
}  
}
```

```

/*
 * formatMALDI.java
 *
 * Created on June 30, 2005, 2:45 PM
 */

package psm.mformat;

import java.beans.*;
import java.io.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletInputStream;
import java.util.*;
import java.text.SimpleDateFormat;
import java.text.DecimalFormat;

/**
 * @author Brandon
 */
public class formatMALDITemp {
    private ArrayList maldiMZ;
    private ArrayList maldiIntensity;
    private String notes;
    private String fileName;

    public formatMALDITemp() {
        maldiMZ = new ArrayList();
        maldiIntensity = new ArrayList();
    }

    public int formatMALDIFile(HttpServletRequest request) throws IOException {
        maldiMZ = new ArrayList();
        maldiIntensity = new ArrayList();
        notes = "";

        ServletInputStream inStr = request.getInputStream();
        byte[] line = new byte[1024];

        int i = inStr.readLine(line, 0, 1024);

        //The length of the boundary should be greater than three, anything less
        means an error has occurred, so exit.
        if (i < 3) {
            return 1;
        }

        //Boundary length is the first line minus the first two CR's that terminate
        it
        int boundaryLength = i - 2;
        //Retrieves the boundary minus the two CR's
        String boundary = new String(line, 0, boundaryLength);
        fileName = null;
        String tempStr = "";

        boolean isFile = false;
        while (i != -1) {
            //Store the current line as a string
            String newLine = new String(line, 0, i);
            if (!isFile) {
                //All we care about is the file so loop through all the fields until
                you hit this.
                if (newLine.indexOf("filename=\"") != -1) {
                    fileName = new String(line, newLine.indexOf("filename=\"") + 10
, newLine.indexOf("\\"", newLine.indexOf("filename=\""))));

```

```

        fileName = fileName.replace("\"", "");
        fileName = fileName.trim();
        fileName = fileName.substring(0, fileName.length() - 4);

        notes = fileName;

        isFile= true;
    }
} else {
    //if no filename was entered exit
    if (fileName==null) {
        return 2;
    }
    //Do not read the very last line

    if (!newLine.startsWith(boundary) & newLine.indexOf("Content") ==
-1) {
        //This is where reading of the lines of the file occurs.
        int lCount = 0; //keeps track of the line count
        while (i != -1 && !newLine.startsWith(boundary)) {
            lCount++; //Increment the line count indicating the current
position relative to the file contents
            i = inStr.readLine(line, 0, 1024);

            //On the filename three is the sample name date and target
number, we want this information
            if (lCount > 1) {
                //if lCount is greater than 11 then we are now on the
data lines and can start processing that data
                //Columns are seperated by a series of spaces, and the
peak numbers have a width of a minimum of 3 characters, and both Mass and intensity
columns have
                //a decimal point, this will be used to determine the
position of the numbers and to grab them
                if (newLine.length() < 5) { return 0;} //the last line
could possibly be a carriage return, we dont care about this, so just exit.

                String[] pline = newLine.split("\t");

                //now assign this into the mass array, in a new positon.
                double dx = Double.parseDouble(pline[0]);
                //This is for rounding, add .5 to the decimal, and when
converted to an int the decimal drops off
                dx += .5;
                int di = (int)dx;
                maldiMZ.add(new Integer(di));

                //Change to a double and round to the first decimal
place
                dx = Double.parseDouble(pline[1]);
                dx += 0.05;
                DecimalFormat dF = new DecimalFormat("00.0");
                maldiIntensity.add(new
Double(Double.parseDouble(dF.format((dx)))));
            }
            /*
            if ((i==boundaryLength+2 || i==boundaryLength+4 && new
String(line, 0, i).startsWith(boundary))) {
                pw.print(newLine.substring(0, newLine.length()-2));
            } else {
                pw.print(newLine);
            }
            */
            newLine = new String(line, 0, i);
        }
    }
}

```

```
        }
    }
    //iterate to next line
    i = inStr.readLine(line, 0, 1024);
}

return 0;
}

public String getNotes() {
    return notes;
}

public void setNotes(String notes) {
    this.notes = notes;
}

public ArrayList getMaldiMZ() {
    return maldiMZ;
}

public void setMaldiMZ(ArrayList maldiMZ) {
    this.maldiMZ = maldiMZ;
}

public ArrayList getMaldiIntensity() {
    return maldiIntensity;
}

public void setMaldiIntensity(ArrayList maldiIntensity) {
    this.maldiIntensity = maldiIntensity;
}

public String getFileName() {
    return fileName;
}
}
```

```

/*
 * formatMALDI.java
 *
 * Created on June 30, 2005, 2:45 PM
 */

package psm.mformat;

import java.beans.*;
import java.io.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletInputStream;
import java.util.*;
import java.text.SimpleDateFormat;
import java.text.DecimalFormat;

/**
 * @author Brandon
 */
public class formatMSMSData {
    private ArrayList maldiMZ;
    private ArrayList maldiIntensity;
    private String targetNum;
    private Date mdate;
    private String number;
    private String notes;
    private String fileName;
    private double MW;
    private String protein;

    public formatMSMSData() {
        maldiMZ = new ArrayList();
        maldiIntensity = new ArrayList();
    }

    public int formatMALDIFile(HttpServletRequest request) throws IOException {
        maldiMZ = new ArrayList();
        maldiIntensity = new ArrayList();
        mdate= null;
        number = "";
        notes = "";
        targetNum = "";

        ServletInputStream inStr = request.getInputStream();
        byte[] line = new byte[1024];

        int i = inStr.readLine(line, 0, 1024);

        //The length of the boundary should be greater than three, anything less
means an error has occurred, so exit.
        if (i < 3) {
            return 1;
        }

        //Boundary length is the first line minus the first two CR's that terminate
it
        int boundaryLength = i - 2;
        //Retrieves the boundary minus the two CR's
        String boundary = new String(line, 0, boundaryLength);
        fileName = null;
        String tempStr = "";

        boolean isFile = false;
        while (i != -1) {

```

```

//Store the current line as a string
String newLine = new String(line, 0, i);
if (!isFile) {
    //All we care about is the file so loop through all the fields until
you hit this.
    if (newLine.indexOf("filename=\"") != -1) {
        fileName = new String(line, newLine.indexOf("filename=\"") + 10
, newLine.indexOf("\"", newLine.indexOf("filename=\"")));
        fileName = fileName.replace("\"", "");
        fileName = fileName.trim();
        isFile= true;
    }
} else {
    //if no filename was entered exit
    if (fileName==null) {
        return 2;
    }
    //Do not read the very last line
    if (!newLine.startsWith(boundary) & newLine.indexOf("Content") ==
-1) {
        //This is where reading of the lines of the file occurs.
        int lCount = 0; //keeps track of the line count
        while (i != -1 && !newLine.startsWith(boundary)) {
            lCount ++; //Increment the line count indicating the current
position relative to the file contents
            i = inStr.readLine(line, 0, 1024);

            if (lCount > 1) {
                //if lCount is greater than 11 then we are now on the
data lines and can start processing that data
                //Columns are seperated by a series of spaces, and the
peak numbers have a width of a minimum of 3 characters, and both Mass and intensity
columns have
                //a decimal point, this will be used to determine the
position of the numbers and to grab them
                if (newLine.length() < 5) { return 0;} //the last line
could possibly be a carriage return, we dont care about this, so just exit.

                //This format is going to assume a tab delimited file,
with the first line containing:
                //RM# MW Protein Notes
                //M/Z Intensity
                if (lCount == 2) {
                    //This is the first line
                    String [] fstline = null;
                    newLine = newLine.trim();
                    fstline = newLine.split("\t");

                    //The first string in fstline should be the rm
number
                    setNumber(fstline[0]);

                    //Second line is the date
                    if (fstline.length > 2) {
                        //This is the date
                        SimpleDateFormat df = new
SimpleDateFormat("MM/dd/yy");

                        try {
                            mdate = df.parse( fstline[1]);
                        } catch (Exception e){
                            mdate = null;
                        }
                    }
                }
            }
        }
    }
}

```

```

//The third should be the MW weight
setMW(Double.parseDouble(fstline[2]));

//The third is teh protein name and the fourth are
any notes
//These two are optional and may not exist
if (fstline.length > 3) {
    //This is the protein
    setProtein(fstline[3]);
}
if (fstline.length > 4) {
    //the rest are notes
    setNotes(fstline[4]);
}
} else {
    //Not the first line, dump into array
    String [] dataLine = null;
    newLine = newLine.trim();
    dataLine = newLine.split("\t");

    //This should containt one set of strings, the first
being the mass to charge ratio
//and the second being the intensity.

    double mz = Double.parseDouble(dataLine[0]);
    maldiMZ.add(new Double(mz));

    double intens = Double.parseDouble(dataLine[1]);
    maldiIntensity.add(new Double(intens));

}
}
/*
if ((i==boundaryLength+2 || i==boundaryLength+4 && new
String(line, 0, i).startsWith(boundary))) {
    pw.print(newLine.substring(0, newLine.length()-2));
} else {
    pw.print(newLine);
}
*/
newLine = new String(line, 0, i);
}

}
}
//iterate to next line
i = inStr.readLine(line, 0, 1024);
}

return 0;
}

public String getTargetNum() {
    return targetNum;
}

public void setTargetNum(String targetNum) {
    this.targetNum = targetNum;
}

public Date getMdate() {
    return mdate;
}
}

```



```
public void setDate(Date mdate) {
    this.mdate = mdate;
}

public String getNumber() {
    return number;
}

public void setNumber(String number) {
    this.number = number;
}

public String getNotes() {
    return notes;
}

public void setNotes(String notes) {
    this.notes = notes;
}

public ArrayList getMaldiMZ() {
    return maldiMZ;
}

public void setMaldiMZ(ArrayList maldiMZ) {
    this.maldiMZ = maldiMZ;
}

public ArrayList getMaldiIntensity() {
    return maldiIntensity;
}

public void setMaldiIntensity(ArrayList maldiIntensity) {
    this.maldiIntensity = maldiIntensity;
}

public String getFileName() {
    return fileName;
}

public double getMW() {
    return MW;
}

public void setMW(double MW) {
    this.MW = MW;
}

public String getProtein() {
    return protein;
}

public void setProtein(String protein) {
    this.protein = protein;
}
}
```

```

/*
 * importGenelist.java
 *
 * Created on November 17, 2005, 1:01 PM
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package psm.mformat;

import java.sql.*;
import java.beans.*;
import java.io.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletInputStream;
import java.util.*;
import psm.db.dbAccess;
/**
 *
 * @author User
 */
public class importGenelist {

    /** Creates a new instance of importGenelist */
    public importGenelist() {

        public int uploadGenelist(HttpServletRequest request, dbAccess db) throws
IOException{
        ServletInputStream inStr = request.getInputStream();

        byte[] line = new byte[1024];

        int i = inStr.readLine(line, 0, 1024);

        //The length of the boundary should be greater than three, anything less
means an error has occurred, so exit.
        if (i < 3) {
            return 1;
        }

        //Boundary length is the first line minus the first two CR's that terminate
it
        int boundaryLength = i - 2;
        //Retrieves the boundary minus the two CR's
        String boundary = new String(line, 0, boundaryLength);
        String fileName = null;
        boolean isFile = false;
        while (i != -1) {
            //Store the current line as a string
            String newLine = new String(line, 0, i);
            if (!isFile) { //determine when we reach the file
                if (newLine.indexOf("filename=\"") != -1) {
                    fileName = new String(line, newLine.indexOf("filename=\"") + 10
, newLine.indexOf("\\"", newLine.indexOf("filename=\"")));
                    fileName = fileName.replace("\\"", "");
                    fileName = fileName.trim();
                    isFile= true;
                } else if (newLine.startsWith(boundary) && isFile) {
                    return 0;
                }
            }
        }
    } else {

```

```

//if no filename was entered exit
if (fileName==null) {
    return 2;
}
//Do not read the very last line
if (!newLine.startsWith(boundary) & newLine.indexOf("Content") ==
-1) {

    while (i != -1 && !newLine.startsWith(boundary)) {

        i = inStr.readLine(line, 0, 1024);
        newLine = new String(line, 0, i);
        if (newLine.compareTo("\r\n") == 0) {
            //end of file exit
            return 0;
        }
        if (newLine.trim() != "") {
            String[] row = newLine.split("\t");

            /*
            *0 Locus
            *1 Genus
            *2 Species
            *3 Subspecies
            *4 Subtype
            *5 Strain
            *6 Location
            *7 Location2
            *8 Start
            *9 End
            *10 GBK Coordinates
            *11 F/R
            *12 NT
            *13 AA
            *14 MW
            *15 IEP
            *16 Gene
            *17 Code
            *18 COG
            *19 EC#1
            *20 EC#2
            *21 Product
            *22 Notes
            */
            String notes;
            if (row.length == 23) {
                notes = row[22];
            } else {
                notes = "";
            }

            int p = db.insertLocus(row[0],
row[1],row[2],row[3],row[4],row[5],row[6],row[7],row[8],row[9],row[11],row[12],row[1
3],row[14],row[15],row[16],row[17],row[18],row[19],row[20],row[21],notes);
            if (p != 0) {return 1;}
            row = null;

        } //Terminate loop if statement
    } //terminate second loop
    i = inStr.readLine(line, 0, 1024);
} //terminate if statenemt

} //terminate else statemtn
//iterate to next line

```

```
        i = inStr.readLine(line, 0, 1024);  
    }//terminate loop  
    return 0;  
}  
}
```

```

/*
 * importStrain.java
 *
 * Created on October 21, 2005, 11:20 AM
 */

package psm.mformat;

import java.sql.*;
import java.beans.*;
import java.io.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletInputStream;
import java.util.*;
import psm.db.dbAccess;
/**
 *
 * @author User
 */
public class importStrain {

    /** Creates a new instance of importStrain */
    public importStrain() {

        public int parseandImportStrain(HttpServletRequest request, dbAccess db) throws
        IOException {
            boolean updDate = false, updGenus = false, updSpecies = false,
            updSubspecies = false, updSubtype = false, updNotes = false, updHist = false,
            append = false;
            ServletInputStream inStr = request.getInputStream();
            int len = 1024;

            byte[] line = new byte[len];

            int i = inStr.readLine(line, 0, len);
            //The length of the boundary should be greater than three, anything less
            means an error has occurred, so exit.
            if (i < 3) {
                return 1;
            }

            //Boundary length is the first line minus the first two CR's that terminate
            it
            int boundaryLength = i - 2;
            //Retrieves the boundary minus the two CR's
            String boundary = new String(line, 0, boundaryLength);
            String fileName = null;
            boolean isFile = false;
            while (i != -1) {
                //Store the current line as a string
                String newLine = new String(line, 0, i);
                if (!isFile) { //determine when we reach the file
                    if (newLine.indexOf("filename=\"") != -1) {
                        fileName = new String(line, newLine.indexOf("filename=\"") + 10
, newLine.indexOf("\\"", newLine.indexOf("filename=\""))));
                        fileName = fileName.replace("\\"", "");
                        fileName = fileName.trim();
                        isFile = true;
                    } else if (newLine.startsWith(boundary) && isFile) {
                        return 0;
                    } else if ( newLine.indexOf("name=\"") != -1) {

```

```

        //get options
        //this is a field
        // get the field name
        int pos = newLine.indexOf("name=\"");
        String fieldName = newLine.substring(pos+6, newLine.length()-3);
        //All the field names are options, and are only there if checked
        if (fieldName.compareTo("date") == 0) {
            updDate = true;
        } else if(fieldName.compareTo("Genus") == 0) {
            updGenus = true;
        } else if(fieldName.compareTo("Species") == 0) {
            updSpecies = true;
        } else if(fieldName.compareTo("Subspecies") == 0) {
            updSubspecies = true;
        } else if(fieldName.compareTo("Subtype") == 0) {
            updSubtype = true;
        } else if(fieldName.compareTo("notes") == 0) {
            updNotes = true;
        } else if(fieldName.compareTo("History") == 0) {
            updHist = true;
        } else if(fieldName.compareTo("append") == 0) {
            append = true;
        }
    }
} else {
    //if no filename was entered exit
    if (fileName==null) {
        return 2;
    }
    //Do not read the very last line
    if (!newLine.startsWith(boundary) & newLine.indexOf("Content") ==
-1) {

        while (i != -1) {
            i = inStr.readLine(line, 0, len);
            newLine = new String(line, 0, i);
            if (newLine.compareTo("\r\n") == 0) {
                //end of file exit
                return 0;
            }
            if (newLine.trim() != "" &&
newLine.trim().compareTo(boundary + "--") != 0) {
                String[] row = newLine.split("\t");

                //The String columns should be as follows
                //1) Identifier + Number
                //2) Date
                //3) Genus
                //4) Species
                //5) Subspecies
                //6) Subtype
                //7) Notes
                //8) History
                int sampleid = 0;
                int gtent = 0, stent = 0, sptent = 0, otent = 0;
                //Check for name tentativity and parse out any ? mark

                if (row[2].indexOf("?") != -1 ) {
                    row[2].replace("?", "");
                    gtent = 1;
                }
                if (row[3].indexOf("?") != -1 ) {
                    row[3].replace("?", "");

```

```

        stent = 1;
    }
    if (row[4].indexOf("?") != -1 ) {
        row[4].replace("?", "");
        sptent = 1;
    }
    if (row[5].indexOf("?") != -1 ) {
        row[5].replace("?", "");
        otent = 1;
    }
    //First get the sample_id
    String selSample = "SELECT sample_id FROM
sample,identifier WHERE identifier.identifier_id = sample.identifier_id AND
CONCAT(identifier.ident, sample.number) = '" + row[0] + "'";
    java.sql.ResultSet myRS;

    try {
        myRS = db.getRS(selSample);
        if (myRS.next()) {
            sampleid = myRS.getInt("sample_id");
            myRS.close();
        } else {
            myRS.close();
            //This sample does not currently exist in the
database
            //Get a list of all the identifiers in the DB,
compare it to the sample and try to separate the prefix from the numbers
            myRS = db.getIdentPrefix();
            int prID = -1; //keeps track of the prefix id
            String prfx = "";
            //Cycle through the prefixes and compare it to
the ident string
            while (myRS.next()) {
                if
                (row[0].startsWith(myRS.getString("ident"))) {
                    //If the string starts with one of the
prefixes check and see if it identified with an earlier one as well
                    //If the string did, then choose the
longer of the two, assuming that is the correct one.
                    if (prID == -1) {
                        prID = myRS.getInt("identifier_id");
                        prfx = myRS.getString("ident");
                    } else {
                        if (prfx.length() <
myRS.getString("ident").length()) {
                            prID =
myRS.getInt("identifier_id");
                            prfx = myRS.getString("ident");
                        }
                    }
                }
            }
            //end while
            //if prID still equals -1 then the identifier
doesn't exist, so exit
            if (prID == -1) {
                return 1;
            } else {
                //now insert the strain
                int idNumber =
Integer.parseInt(row[0].substring(prfx.length()));
                int erd;
                if (row.length == 8) {
                    erd = db.insertNewStrain(prID, idNumber,
row[1], db.insertNewGenus(row[2].trim()), db.insertNewSpecies(row[3].trim()),
db.insertNewSubspecies(row[4].trim()), db.insertNewSubtype(row[5].trim()),

```

```

row[6].trim(), row[7].trim(), gtent, stent, sptent, otent);
    } else {
        erd = db.insertNewStrain(prID, idNumber,
row[1], db.insertNewGenus(row[2].trim()), db.insertNewSpecies(row[3].trim()),
db.insertNewSubspecies(row[4].trim()), db.insertNewSubtype(row[5].trim()),
row[6].trim(), "", gtent, stent, sptent, otent);
    }
    //if a db error occurred exit
    if (erd != 0) {
        return 1;
    } else {
        //skip the rest of the loop
        continue;
    }
}

}

} catch (SQLException e) {
    return 1;
}

String slctStrain = "SELECT strain_id FROM strain WHERE
genus_id=" + db.insertNewGenus(row[2].trim()) + " AND species_id=" +
db.insertNewSpecies(row[3].trim()) + " AND subspecies_id=" +
db.insertNewSubspecies(row[4].trim()) + " AND subtype_id=" +
db.insertNewSubtype(row[5].trim());
int strainID;

try {
    myRS = db.getRS(slctStrain);
    if (myRS.next()) {
        //the strain already exists, use this id
        strainID = myRS.getInt("strain_id");
    } else {
        //the strain does not exist, make a new one and
use the new id

        myRS.close();
        //Insert the strain ID, now get it the same way
as above

        String insrtStrain;
        insrtStrain = "INSERT INTO strain (genus_id,
species_id, subspecies_id, subtype_id) VALUES(" + db.insertNewGenus(row[2].trim())
+ "," + db.insertNewSpecies(row[3].trim()) + "," +
db.insertNewSubspecies(row[4].trim()) + "," + db.insertNewSubtype(row[5].trim())
+ ")";

        db.executeString(insrtStrain);
        myRS = db.getRS(slctStrain);
        myRS.next();
        strainID = myRS.getInt("strain_id");
    }
    myRS.close();
} catch (SQLException e) {
    return 1;
}

//Check waht to update then update it
String updStmt = "UPDATE sample SET ";
//Date
if (updDate) {
    updStmt += "date = '" + row[1] + "',";
}
//Genus
if (updGenus) {

```



```

        updStmt += "tentative_genus = " + gtent + ",";
    }

    //Species
    if (updSpecies) {
        updStmt += "tentative_species = " + stent + ",";
    }

    //Subspecies
    if (updSubspecies) {
        updStmt += "tentative_subspecies = " + sptent + ",";
    }

    //Subtype
    if (updSubtype) {
        updStmt += "tentative_subtype = " + otent + ",";
    }

    if (updGenus || updSpecies || updSubtype ||
updSubspecies) {
        updStmt += "strain_id = " + strainID + ",";
    }

    //Notes
    if (updNotes) {
        if (append) {
            updStmt += " notes=CONCAT_WS(' ', sample.notes,
' " + row[6].trim() + "'),";
        } else {
            updStmt += " notes='" + row[6].trim() + "',";
        }
    }

    //History
    if (updHist && row.length == 8) {
        if (append) {
            updStmt += " history=CONCAT_WS(' ',
sample.history,'" + row[7].trim() + "'),";
        } else {
            updStmt += " history='" + row[7].trim() + "',";
        }
    }

    //Remove the last comma from the statement
    updStmt = updStmt.substring(0, updStmt.length() - 1);

    updStmt += " WHERE sample_id = " +
Integer.toString(sampleid);

    db.executeStatement(updStmt);
    } //Terminate loop if statement
    } //terminate second loop
    i = inStr.readLine(line, 0, len);
    } //terminate if statenemt

    } //terminate else statemtn
    //iterate to next line
    i = inStr.readLine(line, 0, len);
    } //terminate loop

    return 0;
} //terminante function

public int parseandImportNoteData(HttpServletRequest request, dbAccess db)
throws IOException {

```

```

        boolean updDate = false, updGenus = false, updSpecies = false,
updSubspecies = false, updSubtype = false, updNotes = false, updHist = false,
append = false;
        ServletInputStream inStr = request.getInputStream();
        int len = 1024;

        byte[] line = new byte[len];

        int i = inStr.readLine(line, 0, len);
//The length of the boundary should be greater than three, anything less
means an error has occurred, so exit.
        if (i < 3) {
            return 1;
        }

//Boundary length is the first line minus the first two CR's that terminate
it
        int boundaryLength = i - 2;
//Retrieves the boundary minus the two CR's
        String boundary = new String(line, 0, boundaryLength);
        String fileName = null;
        boolean isFile = false;
        while (i != -1) {
            //Store the current line as a string
            String newLine = new String(line, 0, i);
            if (!isFile) { //determine when we reach the file
                if (newLine.indexOf("filename=\"") != -1) {
                    fileName = new String(line, newLine.indexOf("filename=\"") + 10
, newLine.indexOf("\"", newLine.indexOf("filename=\"")));
                    fileName = fileName.replace("\"", "");
                    fileName = fileName.trim();
                    isFile = true;
                } else if (newLine.startsWith(boundary) && isFile) {
                    return 0;
                }
            } else {
                //if no filename was entered exit
                if (fileName == null) {
                    return 2;
                }
                //Do not read the very last line
                if (!newLine.startsWith(boundary) & newLine.indexOf("Content") ==
-1) {

                    while (i != -1) {

                        i = inStr.readLine(line, 0, len);
                        newLine = new String(line, 0, i);
                        if (newLine.compareTo("\r\n") == 0) {
                            //end of file exit
                            return 0;
                        }
                        if (newLine.trim() != "" &&
newLine.trim().compareTo(boundary + "--") != 0) {
                            String updStmt = "UPDATE sample s,identifier i set
s.notes = CONCAT_WS('%notes%', s.notes, ' ') WHERE CONCAT(i.ident, s.number) =
'%RMNUM%' AND s.identifier_id = i.identifier_id";

                            String[] row = newLine.split("\t");
                            updStmt = updStmt.replaceAll("%notes%", row[1].trim());
                            updStmt = updStmt.replaceAll("%RMNUM%", row[0].trim());

                            db.executeString(updStmt);

```

```
        } //Terminate loop if statement
    } //terminate second loop
    i = inStr.readLine(line, 0, len);
} //terminate if statenemt

    } //terminate else statemtn
    //iterate to next line
    i = inStr.readLine(line, 0, len);
} //terminate loop

    return 0;
} //termiante function

} //terminate class
```

```

/*
 * pageCode.java
 *
 * Created on August 11, 2005, 1:10 PM
 */

package psm.web;

import java.beans.*;
import java.io.Serializable;
import java.io.*;
import psm.db.dbAccess;
import java.sql.*;
/**
 * @author User
 */
public class pageCode {

    private String pageHeader;
    private String pageFooter;

    public pageCode() {

    }

    public int splitTemplatePage(String username, String title, String titleSource,
String Appdirectory, dbAccess myDB) {
        File templateFile = new File( Appdirectory + "/template/template.html");
        try {
            FileReader myFilereader = new FileReader(templateFile);
            BufferedReader inStr = new BufferedReader(myFilereader);
            String tmpStr = "";
            boolean footer = false;
            while (tmpStr != null & tmpStr != "null") {
                if (tmpStr != "") {
                    if (tmpStr.indexOf("$MIDDLE$") != -1) {
                        footer = true;
                    } else if (footer==true) {
                        pageFooter += tmpStr;
                    } else {
                        pageHeader += tmpStr;
                    }
                }
                tmpStr = inStr.readLine();
            }

            String splitPage[];

            //Setup the page by replacing the appropriate markers with what is
required
            pageHeader = pageHeader.replace("$TITLE$", title);
            pageHeader = pageHeader.replace("$USERNAME$", username);
            pageHeader = pageHeader.replace("null", "");
            pageFooter = pageFooter.replace("null", "");

            //Get the strains last update
            ResultSet myRS = myDB.getRS("SELECT date FROM lastupdate");
            myRS.next();
            pageHeader = pageHeader.replace("$LASTDATE$", myRS.getString("date"));
            myRS.close();

            if (myDB.getUsername().compareTo("rmandrell") != 0 &&
myDB.getUsername().compareTo("psmlogin") != 0) {
                //find admin only items and remove it from the page

```

```

        int start = pageHeader.indexOf("<!--adminonly{-->");
        int end = pageHeader.indexOf("<!--}adminonly-->");

        while (start != -1 && end != -1) {
            end += 17;

            pageHeader = pageHeader.replace(pageHeader.substring(start,end),
""");
            start = pageHeader.indexOf("<!--adminonly{-->");
            end = pageHeader.indexOf("<!--}adminonly-->");

        }

        start = pageFooter.indexOf("<!--adminonly{-->");
        end = pageFooter.indexOf("<!--}adminonly-->");

        while (start != -1 && end != -1) {
            end +=17;
            pageFooter = pageFooter.replace(pageFooter.substring(start,end),
""");
            start = pageFooter.indexOf("<!--adminonly{-->");
            end = pageFooter.indexOf("<!--}adminonly-->");

        }
    }

    myFileReader.close();
    return 0;
} catch (Exception e) {
    return 1;
}

}

public String getPageHeader() {
    return pageHeader;
}

public void setPageHeader(String pageHeader) {
    this.pageHeader = pageHeader;
}

public String getPageFooter() {
    return pageFooter;
}

public void setPageFooter(String pageFooter) {
    this.pageFooter = pageFooter;
}

}

```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- You may freely edit this file. See commented blocks below for -->
<!-- some examples of how to customize the build. -->
<!-- (If you delete it and reopen the project it will be recreated.) -->
<project name="PSM_web" default="default" basedir=".">
  <description>Builds, tests, and runs the project PSM_web.</description>
  <import file="nbproject/build-impl.xml"/>
  <!--
```

There exist several targets which are by default empty and which can be used for execution of your tasks. These targets are usually executed before and after some main targets. They are:

-pre-init:	called before initialization of project properties
-post-init:	called after initialization of project properties
-pre-compile:	called before javac compilation
-post-compile:	called after javac compilation
-pre-compile-single:	called before javac compilation of single file
-post-compile-single:	called after javac compilation of single file
-pre-compile-test:	called before javac compilation of JUnit tests
-post-compile-test:	called after javac compilation of JUnit tests
-pre-compile-test-single:	called before javac compilation of single JUnit
test	
-post-compile-test-single:	called after javac compilation of single JUnit
test	
-pre-dist:	called before jar building
-post-dist:	called after jar building
-post-clean:	called after cleaning build products
-pre-run-deploy:	called before deploying
-post-run-deploy:	called after deploying

Example of plugging an obfuscator after the compilation could look like

```
<target name="post-compile">
  <obfuscate>
    <fileset dir="${build.classes.dir}"/>
  </obfuscate>
</target>
```

For list of available properties check the imported nbproject/build-impl.xml file.

Other way how to customize the build is by overriding existing main targets. The target of interest are:

init-macrodef-javac:	defines macro for javac compilation
init-macrodef-junit:	defines macro for junit execution
init-macrodef-debug:	defines macro for class debugging
do-dist:	jar archive building
run:	execution of project
javadoc-build:	javadoc generation

Example of overriding the target for project execution could look like

```
<target name="run" depends="<PROJNAME>-impl.jar">
  <exec dir="bin" executable="launcher.exe">
    <arg file="${dist.jar}"/>
  </exec>
</target>
```

Notice that overridden target depends on jar target and not only on compile target as regular run target does. Again, for list of available properties which you can use check the target you are overriding in nbproject/build-impl.xml file.

-->
</project>

Installation and Setup of PSMDB

Author: Brandon Garbus
Last Update: 9/18/2005

Description

This documentation provides a series of steps for installing and securing the ?psmdb? server on a Dell PowerEdge 2800. While these steps may be applied to any PC the installation steps for windows 2000 are specific for any RAID configuration. The client machine in this document refers to the PC that has direct access to the psmdb server. This machine is able to directly connect to MySQL to make backups and insert MALDI-TOF MS data. Note that these instructions assume that the SCSI RAID device is already installed and configured correctly. In the event of a failure of the hardware please consult the hardware documentation or call Dell technical support.

Materials

Windows 2000 CD-ROM	Manufacturer SCSI/RAID Drivers
Restore Disc	Most Recent psmdb backup
Norton Internet Security	Most Recent MySQL DB backup
USB Floppy Drive	Dell PowerEdge Service and Diagnostic Utilities
Procedure	

1. Installation of windows 2000 on the Dell PowerEdge 2800
 - a. First attach the USB Floppy drive to one of the USB ports on the machine.
 - b. Power on the server and insert the Windows 2000 CD-ROM.
 - c. Insert the Manufacturer's SCSI/RAID Drivers into the floppy drive.
 - d. When prompted press F8 to install the third party RAID drivers and follow the onscreen instructions to install the drivers.
 - e. If an error occurs, or the amount of disk space present looks incorrect it may be necessary to reconfigure the RAID array. Consult manual documentation for this.
 - f. Continue following onscreen instructions until windows 2000 is installed.
2. Installation of hardware drivers.
 - a. Insert the DellPowerEdge Service and Diagnostic Utilities CD-ROM
 - b. If AutoRun is enabled Internet Explorer should open and display a web page.
 - c. Select PowerEdge 2800 under Select Server:
 - d. Select Microsoft windows 2000 under Select Drivers/Utilities Set and press continue.
 - e. Install the following sets of drivers by extracting the respective zip files (These are self extracting zip files) by clicking on the hyperlinks.
 - i. Intel: Chipset Software Installation Utility (Driver)
 - ii. Microsoft: SCSI Port fix for BSOD 0xD1 for w2K (Patch/Upgrade)
 - iii. Intel: PRO Gigabit Adapters (Driver)
 - iv. ATI:Radeon 7000M (Driver)
3. Installing Service Pack 3 and Service Pack 4
 - a. Insert the custom Restore Disc.
 - b. Navigate to the directory Restore Disc\Server.
 - c. Double click on w2ksp3.exe and follow the on screen instructions for installation. A reboot may be necessary.
 - d. When finished do the same for w2kSP4_EN.exe. Again, a reboot may be required.
4. Installing the Microsoft Installer
 - a. With the Restore Disc still in the CD-ROM and in the .\Server directory install InstMsiw.exe by double clicking on the icon and following the onscreen instructions. Another reboot may be required.
5. Installing Norton Internet Security.
 - a. Insert the Norton Internet Security CD-ROM and follow the onscreen instructions for installation. A reboot may be necessary.
6. Setting up the network connection
 - a. Plug in the RJ-45 Ethernet cable into the server.
 - b. Navigate to Network and Dial-up Connections. Start > Settings > Network and Dial-up Connections. There should be two connections already listed. Local Area Connection and Local Area Connection 2.

- c. Right click on Local Area Connection and select Properties.
- d. Select the Sharing tab. Make sure that Enable Internet Connection Sharing for this connection is disabled, NOT checked.
- e. Select the General tab. Select Internet Protocol (TCP/IP) and click the Properties button.
- f. Select the option Use the following IP address. Use the below information to fill out the fields.
 - i. IP Address: 147.49.4.115
 - ii. Subnet mask: 255.255.0.0
 - iii. Default gateway: 147.49.254.252
 - iv. Preferred DNS server: 147.49.50.52
 - v. Alternate DNS server: 147.49.50.49
- g. Click the Advanced button.
- h. Select the WINS tab.
- i. Disable NetBIOS by selecting the Disable NetBios over TCP/IP option.
- j. Select the Options tab.
- k. Select TCP/IP filtering.
- l. Click the Properties button
- m. For UDP Ports and IP Protocols Select the Permit Only option.
- n. Press OK. Until you exit out of Local Area Connection.
- 7. Password protect the screen saver
 - a. Right Click on an empty area of the desktop.
 - b. Select Properties.
 - c. Select the Screen Saver tab.
 - d. For a screen saver select Logon Screen Saver, or Blank.
 - e. Make sure the Password protected checkbox is selected.
 - f. Press OK.
- 8. Disable Unnecessary services
 - a. Click Start > Settings > Control Panel.
 - b. Double click Administrative Tools.
 - c. Double click Component Services.
 - d. In the left tree-view select Services (Local).
 - e. Disable the following services by right clicking on a service, selecting Properties, and select Disabled for the Startup type.
 - i. Messenger
 - ii. Print Spooler
 - iii. Remote Registry Service
 - iv. Routing and Remote Access
 - v. Server
 - f. Reboot the machine.
- 9. Installing windows Updates
 - a. Open Internet Explorer.
 - b. If prompted by Norton Internet Security, make sure to permit IE to access the internet. If not prompted, and IE will not display a web page, manually add IE to the permitted applications list in Norton Internet Security Firewall.
 - c. In IE select Tools > Windows Update.
 - d. Follow the onscreen instructions to install all critical security patches and hardware updates. Do NOT install the following
 - i. Windows Media Player Upgrade
 - ii. .Net
 - iii. Direct X
 - e. It will be necessary to reboot and repeat this process many times until all updates have installed successfully.
- 10. Configuring IPSec
 - a. For further information on IPSec documentation can be obtained at <http://www.microsoft.com/technet/itsolutions/network/security/ipsecld.msp>
 - b. Click Start > Settings > Control Panel.
 - c. Double click Administrative Tools.
 - d. Double click Local Security Policy.
 - e. In the left tree-view select IP Security Policies on Local Machine.
 - f. Right click in the right pane and select Manage IP filter lists and filter actions.
 - g. Click Add.
 - h. For the name type ?psmdb?
 - i. Click Add?

- j. A wizard will appear, click next.
- k. For Source address select Any IP Address. Press next
- l. For Destination address select A specific IP Address.
- m. Enter 147.49.4.115 for the IP Address. Press next.
- n. For Select a protocol type: choose TCP. Press next.
- o. Select the To this port: option and enter 80 (From any port to this port: 80). Press Next.
- p. Press Finish.
- q. Press OK.
- r. Follow the same procedure described above for the following filter
 - i. Name: MySQL Rule
 - ii. Protocol: TCP
 - iii. Source Port: ANY
 - iv. Destination Port: 3306
 - v. Source: A specific IP Address
 - 1. IP Address: 147.49.3.158 (This is the client IP address)
 - vi. Destination: A specific IP Address
 - 1. IP Address: 147.49.4.115
- s. Follow that same procedure again for this rule
 - i. Name: All Inbound Traffic
 - ii. Protocol: Any
 - iii. Source: Any
 - iv. Source Port: Any
 - v. Destination: A specific IP Address
 - 1. IP Address: 147.49.4.115
 - vi. Destination Port: Any
- t. Select the Manage Filter Actions tab.
- u. Click Add?
- v. Press next. For the name enter Block. Press next.
- w. Select the Block option. Press next.
- x. Press finish.
- y. Press Close.
- z. Back in the Local Security Settings right click in the right pane and select Create IP Security Policy.
- aa. Press Next.
- bb. For name enter PSMDBPolicy. Press next.
- cc. Make sure the Activate the default response rule box is unchecked. Press next.
- dd. Make sure the Edit properties checkbox IS checked.
- ee. Select Add?
- ff. Press next.
- gg. Press next. (This rule does not specify a tunnel)
- hh. Press next. (All network connections)
- ii. Press next. (Windows 2000 default (Kerberos v5 protocol))
- jj. Select All Inbound Traffic. Make sure the option circle is selected.
- kk. Press next.
- ll. Select Block. Press next. Press finish.
- mm. Repeat the same steps for MySQL Rule and the psmdb Rule except choose Permit. Instead of Block.
- nn. Press OK.
- oo. Right click on the PSMDBPolicy in the right pane and choose Assign.
- pp. Close Local Security Settings.
- 11. User Account Settings
 - a. In Control Panel double click on Users and Passwords.
 - b. Make sure the Users must enter a user name and password to use this computer. checkbox is checked.
 - c. Select the Guest account under Users for this computer:
 - d. Create a long hard to guess password for this account. By click Set Password?
 - e. Select the Advanced tab.
 - f. Check Require users to press Ctrl-Alt-Delete before logging on.
 - g. Click the Advanced button.
 - h. In the left pane select the Users folder.
 - i. Make sure the Guest account is disabled by right clicking on Guest, and selecting Properties. Make sure these checkboxes are checked.
 - i. User cannot change password
 - ii. Password never expires

- iii. Account is disabled
- j. Press OK.
- k. Close Local Users and Groups.
- l. Press OK.
- 12. Installation of Java
 - a. With the Restore Disc in the CD Reader double click on jdk-1_5_0_05-windows-i586-p.exe.
 - b. Follow the instructions to install Java.
 - c. Right click on My Computer. Click Properties.
 - d. Click the Advanced tab.
 - e. Click Environment Variables? button.
 - f. Click New? under User variables for Administrator.
 - g. Enter the following information
 - i. Variable Name: JAVA_HOME
 - ii. Variable Value: c:\Program Files\Java\jdk1.5.0_04
 - 1. This should be where you installed java
 - h. Press OK. Repeat the process for:
 - i. Variable Name: PATH
 - ii. Variable Value: c:\Program Files\Java\jdk1.5.0_04\bin
 - 1. This should be where the \bin directory is in the Java directory
- 13. Installation of MySQL
 - a. With the Restore Disc in the CD Reader double click on Setup.exe
 - b. Follow the onscreen installation instructions, check the box at the end that permits you to configure the server.
 - c. Press next.
 - d. Select Detailed Configuration. Press next.
 - e. Select Server Machine. Press next.
 - f. Select Multifunctional Database. Press next.
 - g. Press next. Select Manual Setting. For Concurrent connections choose 50. Press next.
 - h. Press next.
 - i. Press next.
 - j. Press next.
 - k. Enter the root password. Press next.
 - l. Press execute.
- 14. Installation of MySQL utilities
 - a. Install mysql-administrator-1.1.2-win.msi from the Restore CD. Simply follow the onscreen instructions.
 - b. Do the same for mysql-query-browser-1.1.14-win.msi.
 - c. Copy mysql-connector-java-3.1.10-bin.jar from the CD to C:\mysqljdbc (This directory will need to be created).
- 15. Restoring the databases
 - a. Open MySQL Administrator. Click the ellipses button.
 - b. Click Add new Connection. And fill out the form.
 - i. Username: root
 - ii. Password: (The root password)
 - iii. Hostname: localhost
 - c. Click Apply. Click Close.
 - d. Enter the password again and press OK.
 - e. In the left pane select Restore. Then click on the Open Backup File button.
 - f. Inert the most recent backup CD and find the most recent database backup. It will be psmdbbak DATE CREATED.sql
 - g. Next click on start restore.
 - h. Close MySQL Administrator.
 - i. Naviage to the MySQL Installation Directory (C:\Program Files\MySQL\MySQL Server 4.1) and go to the Data directory.
 - j. Copy the mysql folder from the backup cd and replace the folder and all of its contents on the machine.
 - k. Make sure the folder is not read only. Right click on the folder. Select Properties. If the checkbox for Read-only is checked, uncheck it. Do the same procedure for every file in the /Data/mysql directory.
- 16. Installation of Apache
 - a. Run jakarta-tomcat-5.5.9.exe from the CD-ROM
 - b. Press next until you reach the Choose Components part of the install.
 - c. Click the + sign next to the word Tomcat and make sure the Service checkbox is

selected. Press next.

d. Press next.

e. Press next.

f. For HTTP/1.1 Connector Port erase 8080 and enter 80.

g. Enter a password for administrator login.

h. Press next.

i. Press Install.

17. Installation of PSM_Web

a. From the backup CD copy the PSM_Web directory into the webapps directory of the Apache Tomcat installation (Apache Tomcat Directory/webapps)

b. Navigate to the Directory, Tomcat Installation/conf/Catalina/localhost

c. Copy PSM_Web.xml from the Restore CD to this location

d. Follow the same steps above, except with USDAClub.

WHAT IS CLAIMED IS:

1. A method for identification of an unknown protein comprising:
- (i) comparing mass-to-charge (m/z) fragment ion peaks from an mass spectrum of the unknown protein to that of m/z fragment ion peaks from mass spectra of known protein sequences; and
 - (ii) matching the m/z fragment ion peaks from the mass spectrum of the unknown protein to the m/z fragment ion peaks from the mass spectra of known protein sequences to provide mass spectrometry peak matches;
 - (iii) scoring the mass spectrometry peak matches using a USDA peak matching algorithm thereby providing a USDA peak matching score; and
 - (iv) ranking the USDA peak matching score from highest to lowest;
- wherein
- the highest USDA peak matching score constitutes an identification of the unknown protein.
2. The method of claim 1, wherein the USDA peak matching algorithm is USDA peak matching algorithm α , which has the formula:

$$USDA\alpha Score = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}.$$

3. The method of claim 1, wherein the USDA peak matching algorithm is USDA peak matching algorithm β , which has the formula:

$$USDA\beta Score = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}.$$

4. The method of claim 1, wherein the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than 2.

5. The method of claim 1, wherein the mass spectra are members selected from the group consisting of MS spectra and MS/MS spectra.
6. The method of claim 5, wherein the mass spectrum of the unknown protein is an MS spectrum and wherein the mass spectra of known protein sequences are MS spectra of known proteins.
7. The method of claim 5, wherein the mass spectrum of the unknown protein is an MS/MS spectrum and wherein the mass spectra of known protein sequences are members of the group consisting of *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences and MS/MS spectra of known proteins.
8. The method of claim 7, wherein the mass spectra of known protein sequences are *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences.
9. The method of claim 8, further comprising:
 - (v) confirming that the highest USDA peak matching score constitutes an identification of the unknown protein,
wherein the confirming comprises:
plotting the difference in m/z ($\Delta m/z$) between an MS/MS fragment ion of the unknown protein and its matched fragment ion from the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences as a function of MS/MS fragment ion m/z ,
thereby providing a plot of $\Delta m/z$ vs MS/MS fragment ion m/z ,
wherein
a correct protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals a characteristic systematic error related to the inherent calibration error of the mass analyzer, and
wherein
an incorrect protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals statistically random error.

10. The method of claim 8, further comprising:

- (v) confirming that the highest USDA peak matching score constitutes an identification of the unknown protein,

wherein the confirming comprises:

performing steps (i) – (iv) , wherein the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences are limited to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue or proline (P) residue, and

- (vi) observing that the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when steps (i) – (iv) are performed using *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences that are not limited spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue or proline (P) residue,

thereby confirming that the highest USDA peak matching score constitutes an identification of the unknown protein.

11. The method of claim 10, wherein the confirming comprises:

performing steps (i) – (iv) , wherein the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences are limited to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue; and

- (vi) observing that the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when steps (i) – (iv) are performed using *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences that are not limited spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue,

thereby confirming that the highest USDA peak matching score constitutes an identification of the unknown protein.

12. The method of claim 1, wherein the unknown protein is a protein toxin.
13. The method of claim 1, wherein identification of the unknown protein is used to identify the source organism from which the unknown protein derived.
14. The method of claim 13, wherein the source organism is a member selected from the group consisting of: viruses, microorganisms, insects, plants, and animals.
15. The method of claim 13, wherein the source organism is a microorganism selected from the group consisting of: bacteria and fungi.
16. The method of claim 15, wherein the source organism is a bacterium that is a member selected from the group consisting of *Campylobacter*, *Escherichia coli*, *Listeria*, and *Salmonella*.
17. A computer readable storage medium storing instructions that, when executed by a computer, causes the computer to perform a method for identification of an unknown protein comprising:
 - (i) comparing mass-to-charge (m/z) fragment ion peaks from an mass spectrum of the unknown protein to that of m/z fragment ion peaks from mass spectra of known protein sequences; and
 - (ii) matching the m/z fragment ion peaks from the mass spectrum of the unknown protein to the m/z fragment ion peaks from the mass spectra of known protein sequences to provide mass spectrometry peak matches;
 - (iii) scoring the mass spectrometry peak matches using a USDA peak matching algorithm thereby providing a USDA peak matching score; and
 - (iv) ranking the USDA peak matching score from highest to lowest;wherein
the highest USDA peak matching score constitutes an identification of the unknown protein.

18. The method of claim 17, wherein the USDA peak matching algorithm is USDA peak matching algorithm α , which has the formula:

$$USDA\alpha Score = 100 \times \frac{2 \times (\text{Number of peak matches})}{(\text{Number of peaks in the mass spectrum of the unknown protein} + \text{Number of peaks in the mass spectrum of known protein sequences})}.$$

19. The method of claim 17, wherein the USDA peak matching algorithm is USDA peak matching algorithm β , which has the formula:

$$USDA\beta Score = 100 \times \frac{\text{Number of peak matches}}{(\text{Number of MS/MS fragment ion peaks})}.$$

20. The method of claim 17, wherein the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than 2.

21. The method of claim 17, wherein the mass spectra are members selected from the group consisting of MS spectra and MS/MS spectra.

22. The method of claim 21, wherein the mass spectrum of the unknown protein is an MS spectrum and wherein the mass spectra of known protein sequences are MS spectra of known proteins.

23. The method of claim 21, wherein the mass spectrum of the unknown protein is an MS/MS spectrum and wherein the mass spectra of known protein sequences are members of the group consisting of *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences and MS/MS spectra of known proteins.

24. The method of claim 23, wherein the mass spectra of known protein sequences are *in silico* MS/MS spectra generated from a genomically-derived database of protein

sequences.

25. The method of claim 24, further comprising:

(v) confirming that the highest USDA peak matching score constitutes an identification of the unknown protein,

wherein the confirming comprises:

plotting the difference in m/z ($\Delta m/z$) between an MS/MS fragment ion of the unknown protein and its matched fragment ion from the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences as a function of MS/MS fragment ion m/z ,

thereby providing a plot of $\Delta m/z$ vs MS/MS fragment ion m/z ,

wherein

a correct protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals a characteristic systematic error related to the inherent calibration error of the mass analyzer, and

wherein

an incorrect protein identification is indicated when the plot of $\Delta m/z$ vs MS/MS fragment ion m/z reveals statistically random error.

26. The method of claim 24, further comprising:

(v) confirming that the highest USDA peak matching score constitutes an identification of the unknown protein,

wherein the confirming comprises:

performing steps (i) – (iv), wherein the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences are limited to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue or proline (P) residue, and

(vi) observing that the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when steps (i) – (iv) are performed using *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences that are not

limited spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, glutamate (E) residue or proline (P) residue,

thereby that the highest USDA peak matching score constitutes an identification of the unknown protein.

27. The method of claim 26, wherein the confirming comprises:

performing steps (i) – (iv) , wherein the *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences are limited to spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue; and

(vi) observing that the difference between the highest USDA peak matching score and the second highest USDA peak matching score is equal to or greater than the difference between the highest USDA peak matching score and the second highest USDA peak matching score obtained when steps (i) – (iv) are performed using *in silico* MS/MS spectra generated from a genomically-derived database of protein sequences that are not limited spectra generated by displaying only fragment ions generated by fragmentation at an aspartate (D) residue, thereby confirming that the highest USDA peak matching score constitutes an identification of the unknown protein.

28. The method of claim 17, wherein the unknown protein is a protein toxin.

29. The method of claim 17, wherein identification of the unknown protein is used to identify the source organism from which the unknown protein derived.

30. The method of claim 29, wherein the source organism is a member selected from the group consisting of: viruses, microorganisms, insects, plants, and animals.

31. The method of claim 30, wherein the source organism is a microorganism selected from the group consisting of: bacteria and fungi.

32. The method of claim 31, wherein the source organism is a bacterium that is a member selected from the group consisting of *Campylobacter*, *Escherichia coli*, *Listeria*, and *Salmonella*.

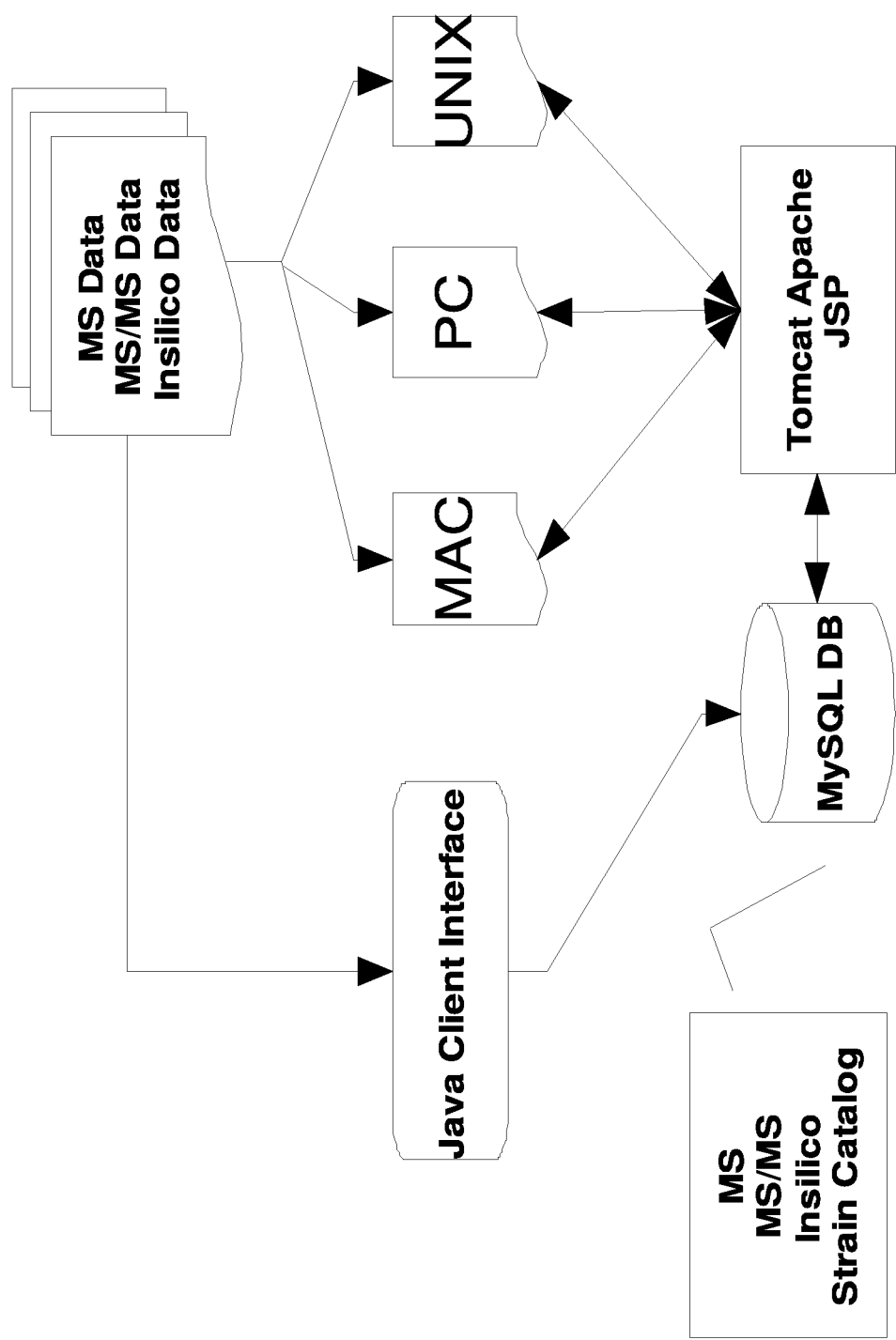
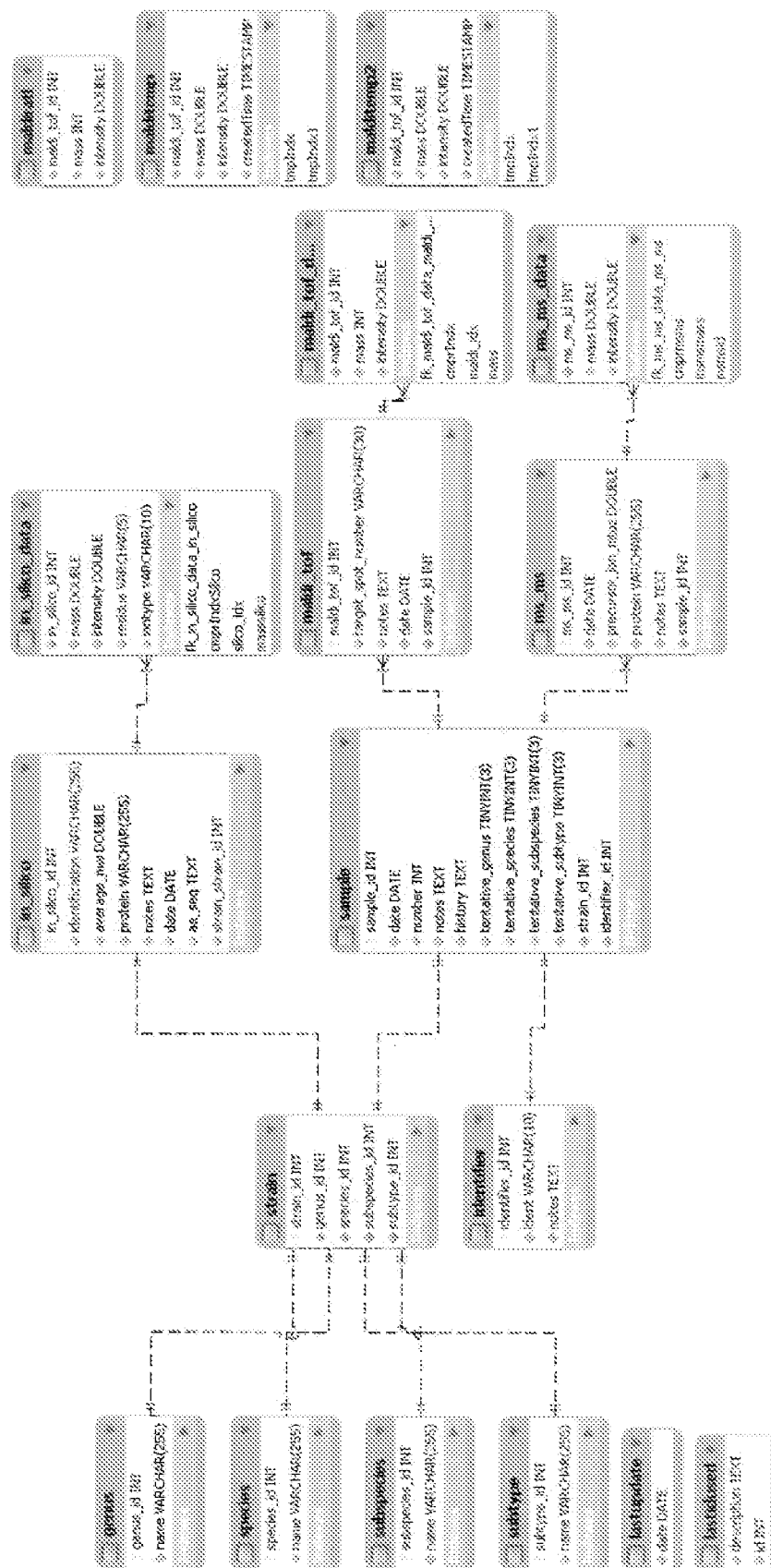


FIG. 1



264

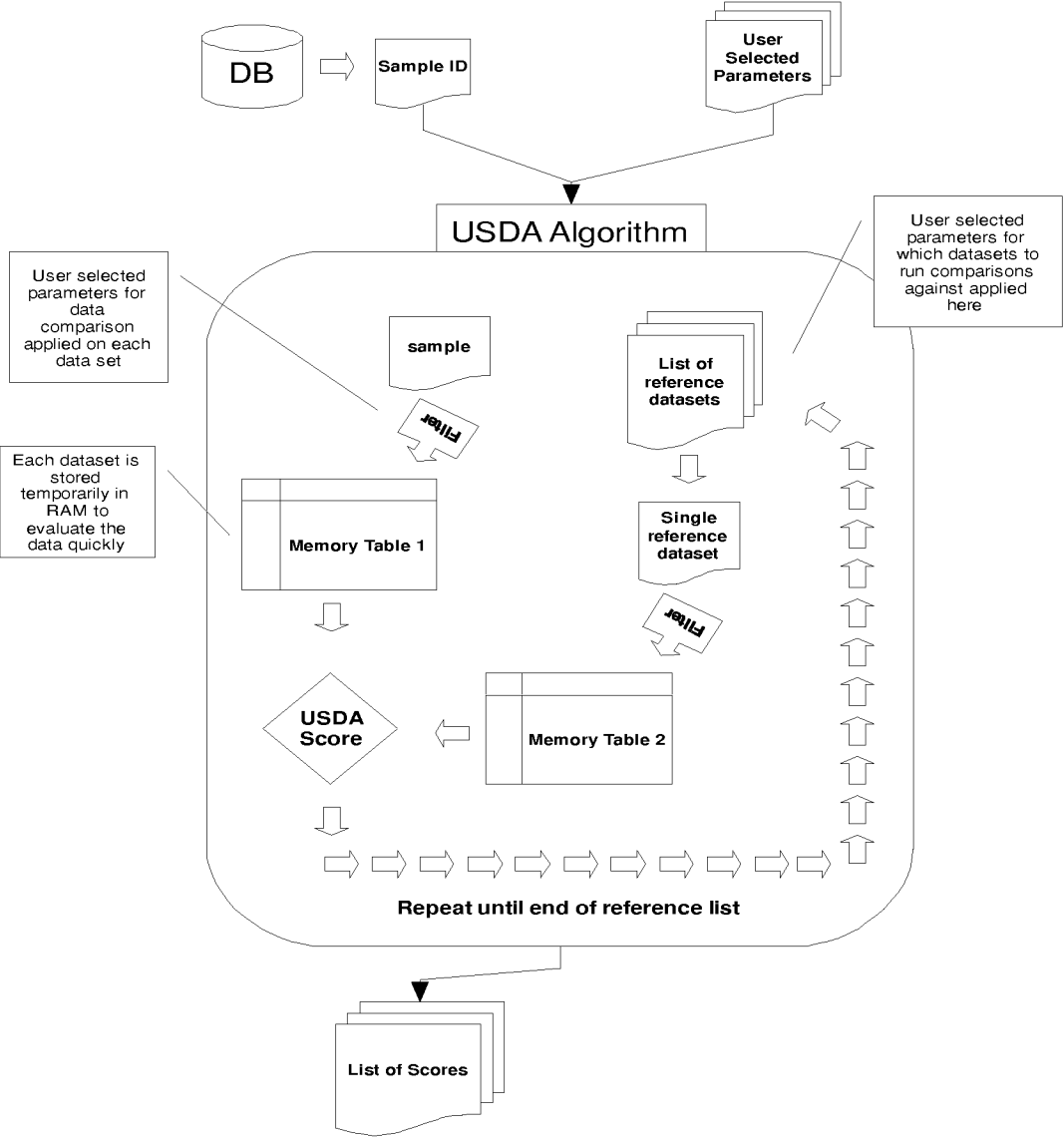


FIG. 3

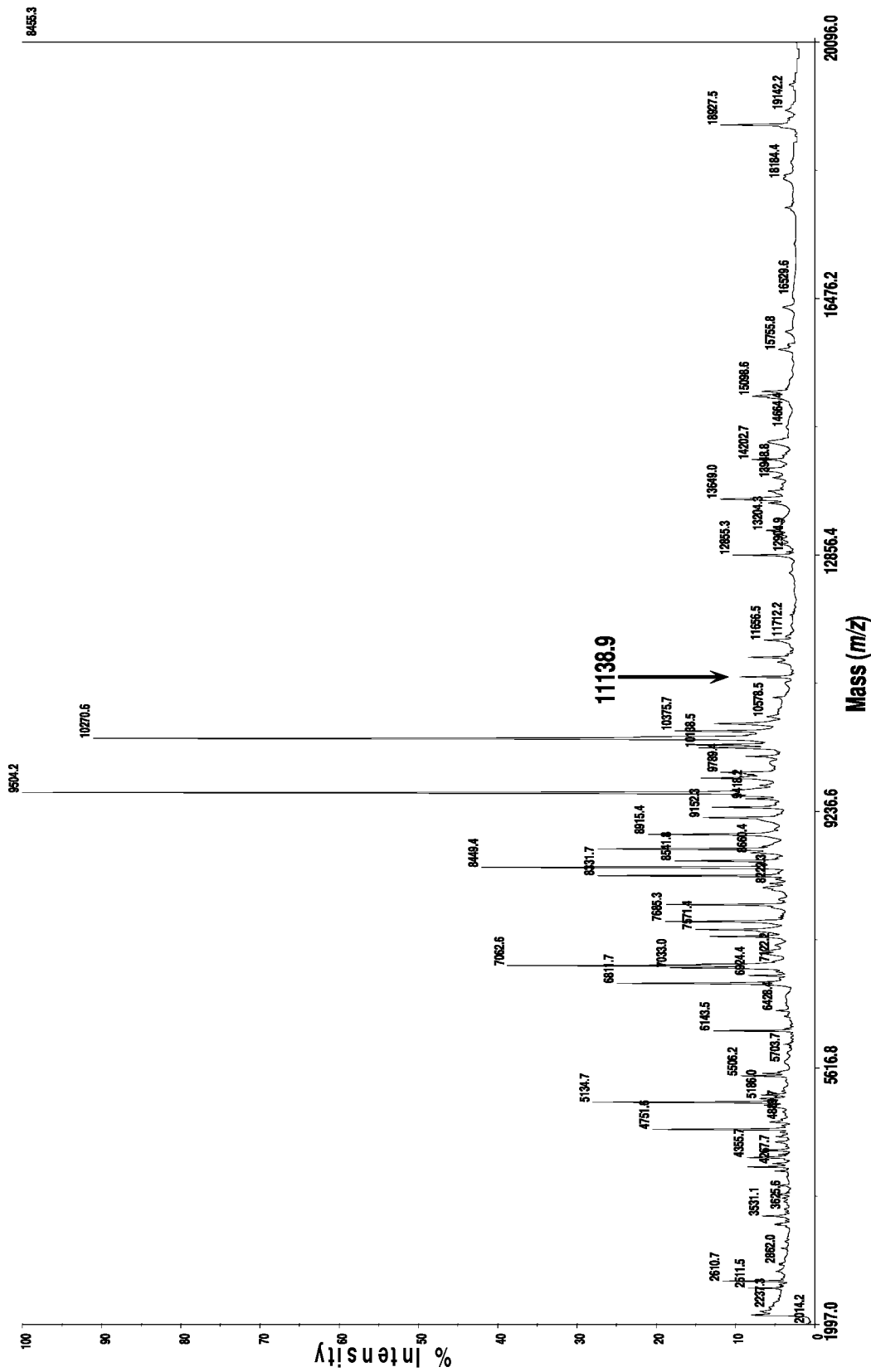


FIG. 4

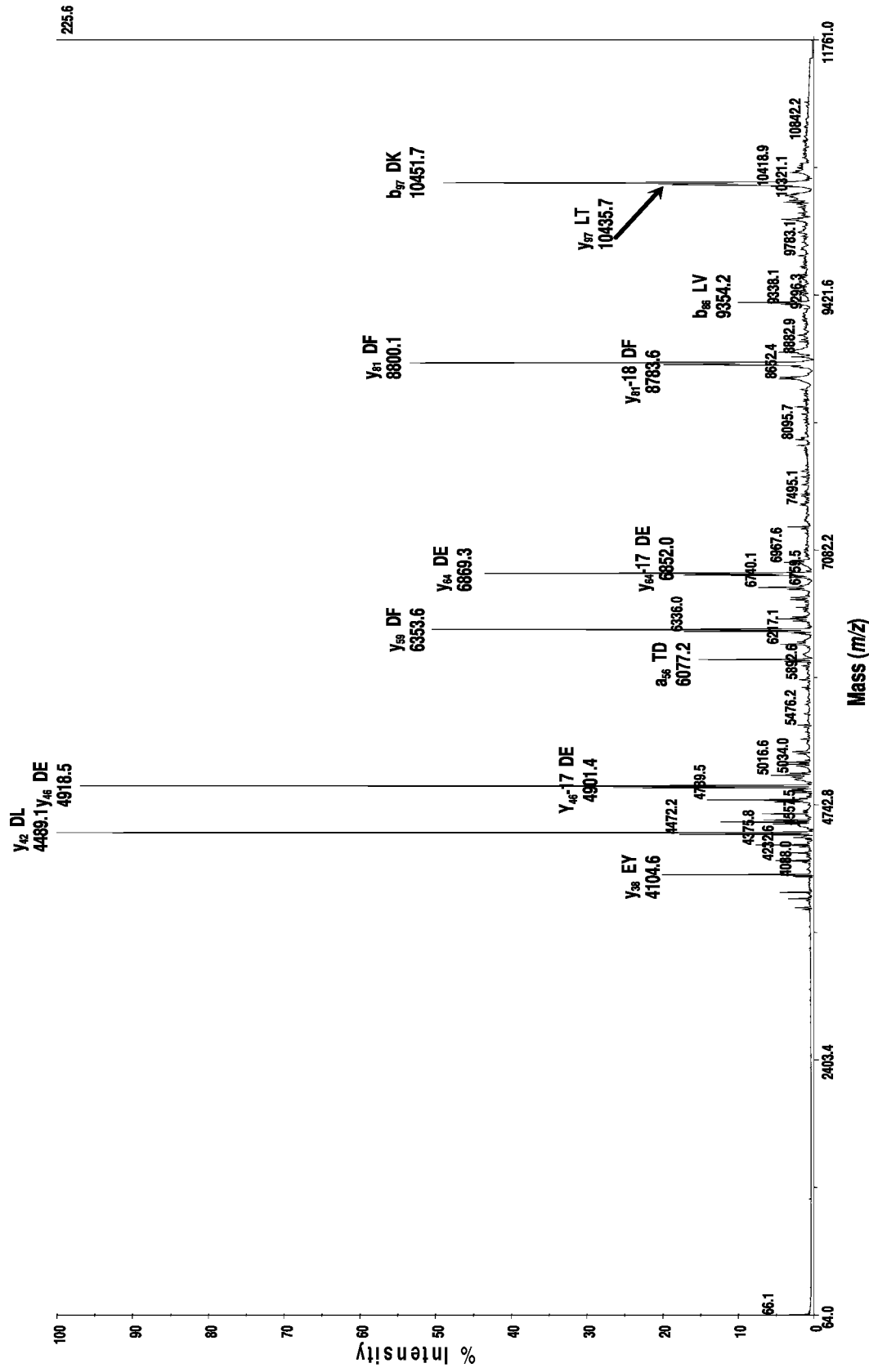


FIG. 5

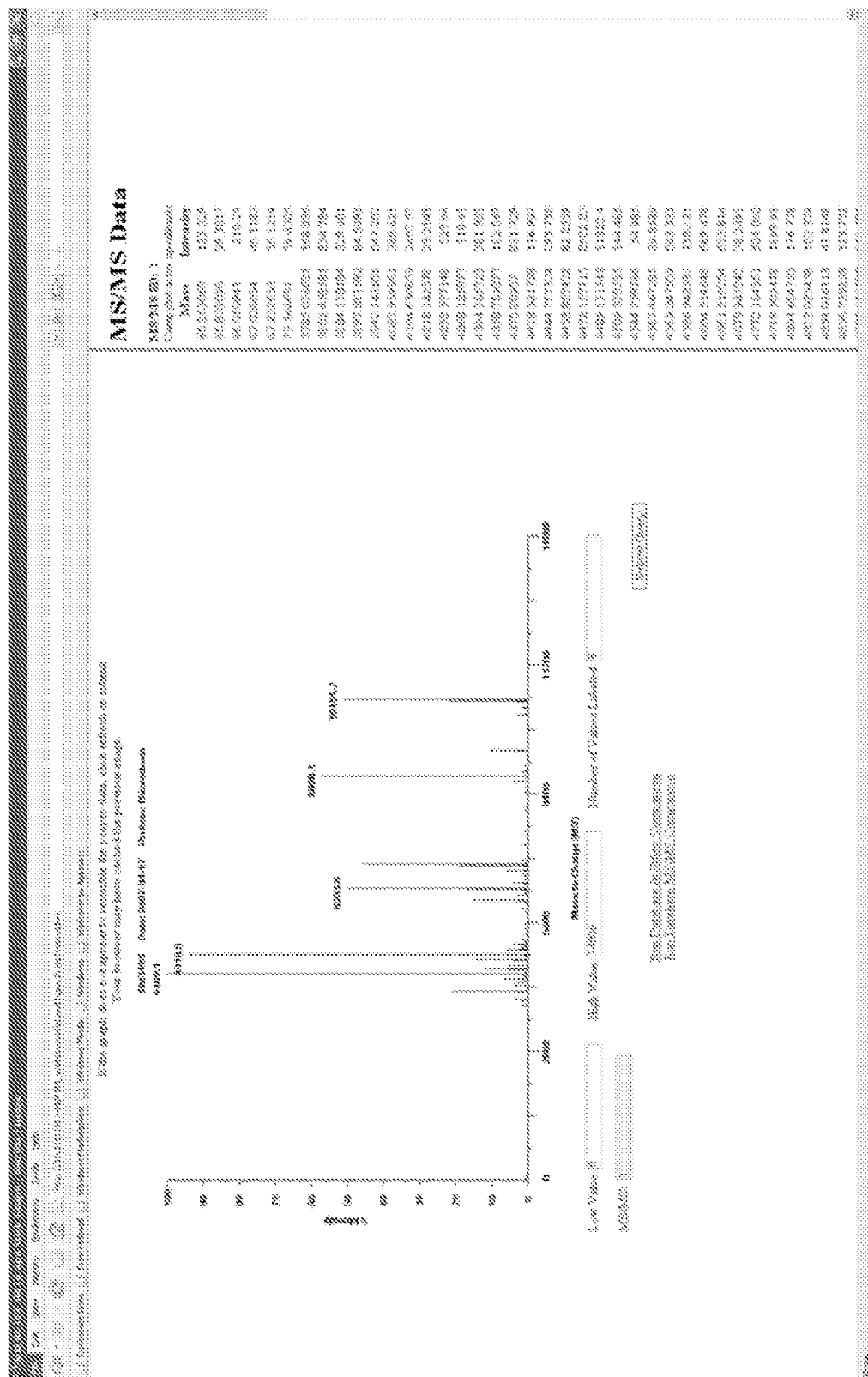


FIG. 6

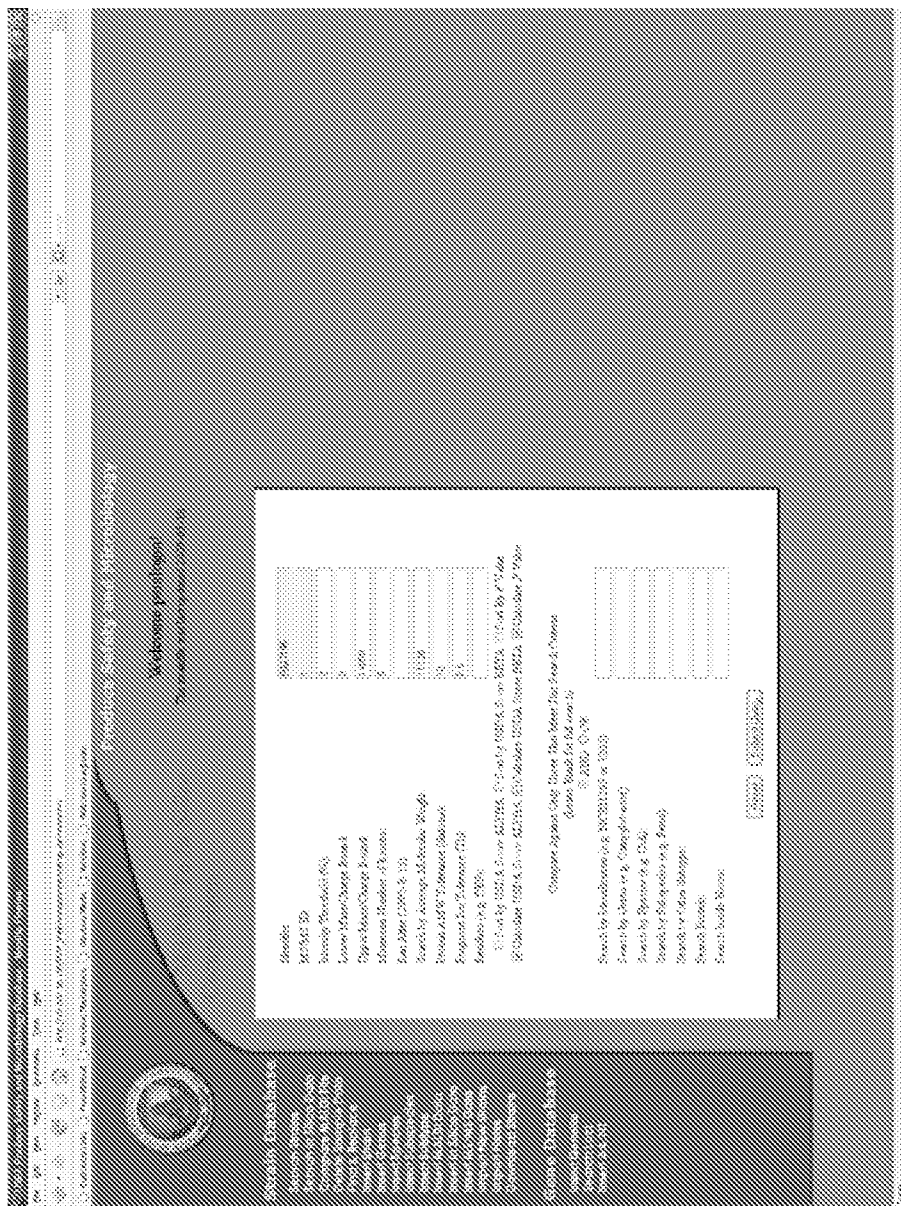


FIG. 7

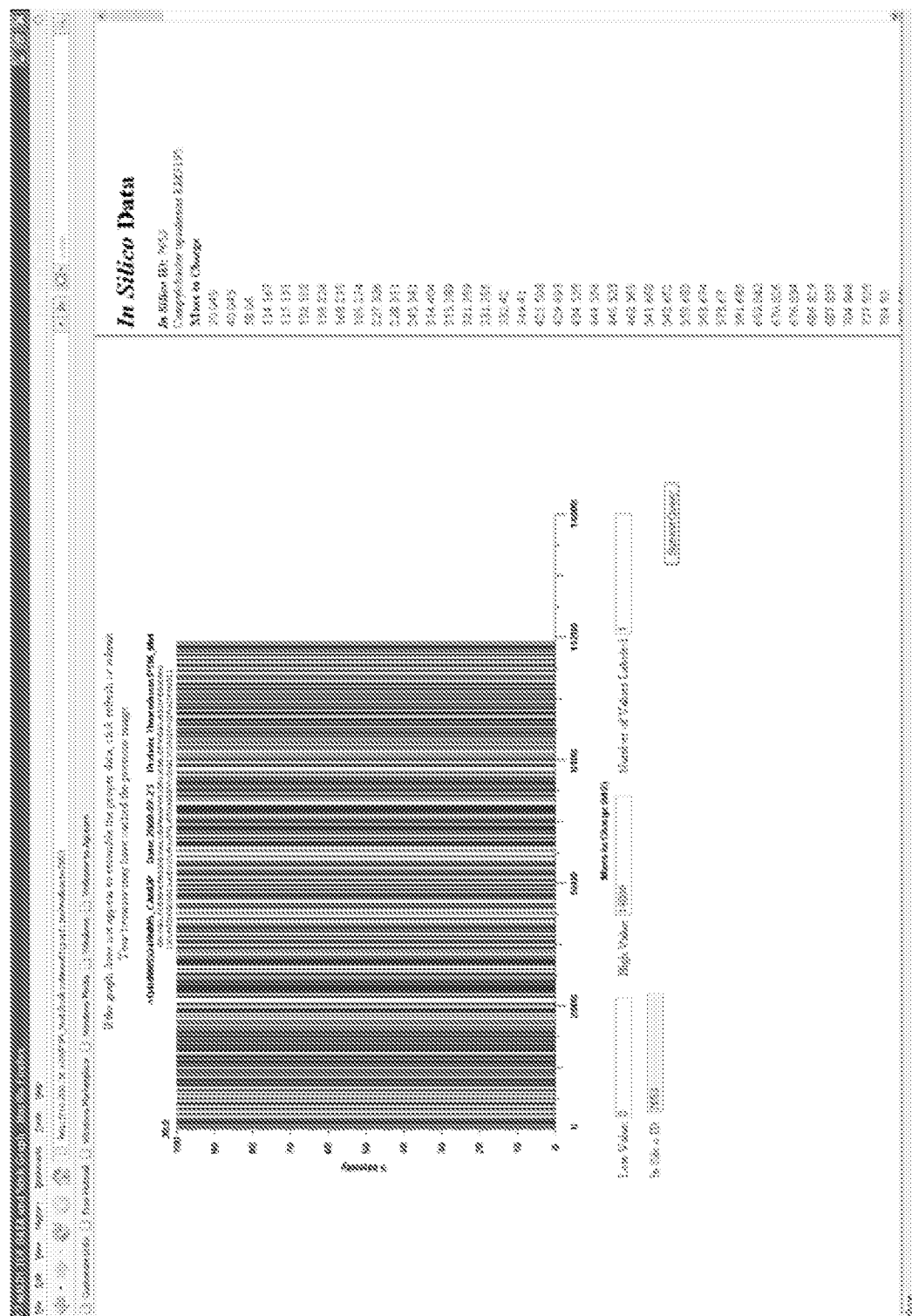


FIG. 8

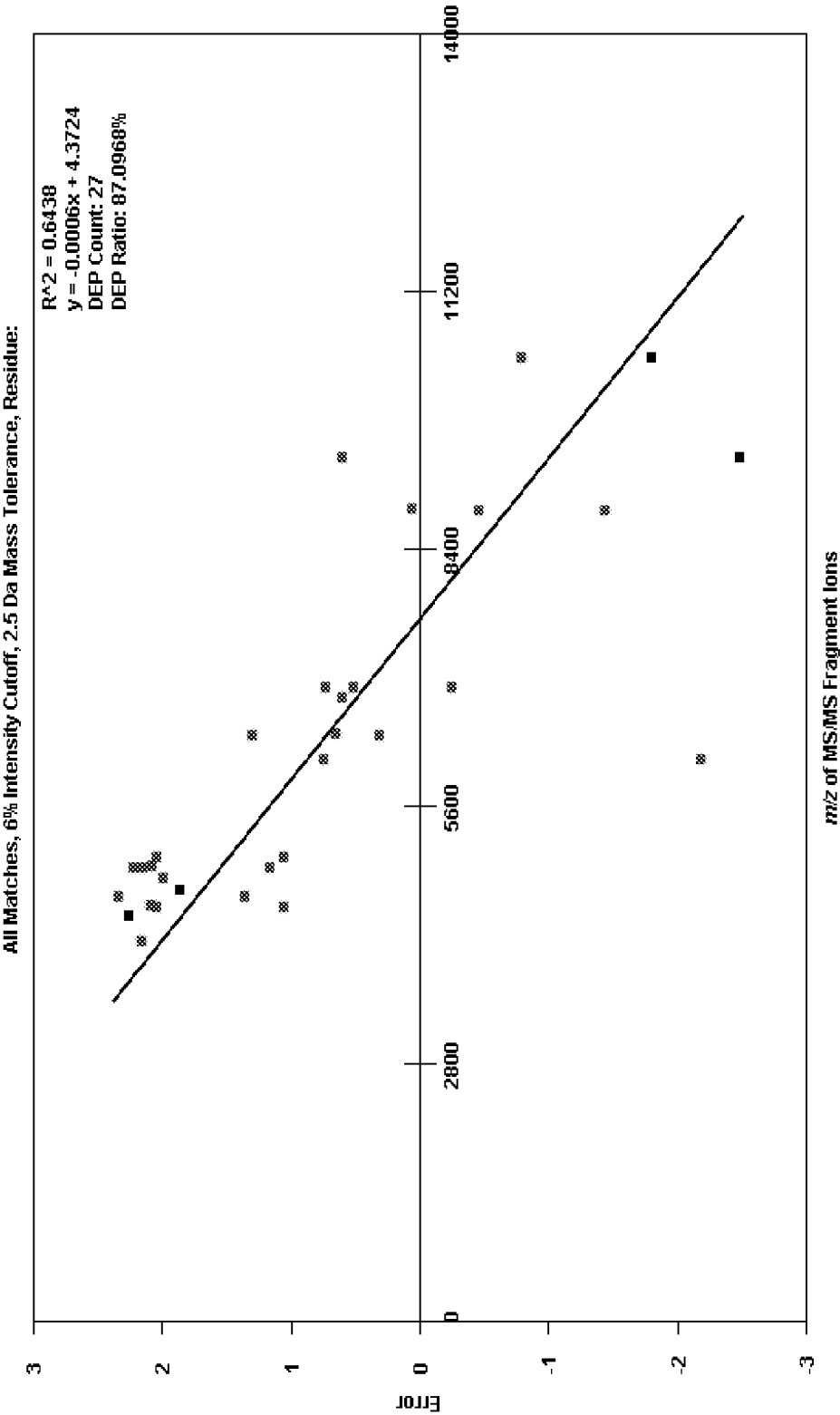


FIG. 9

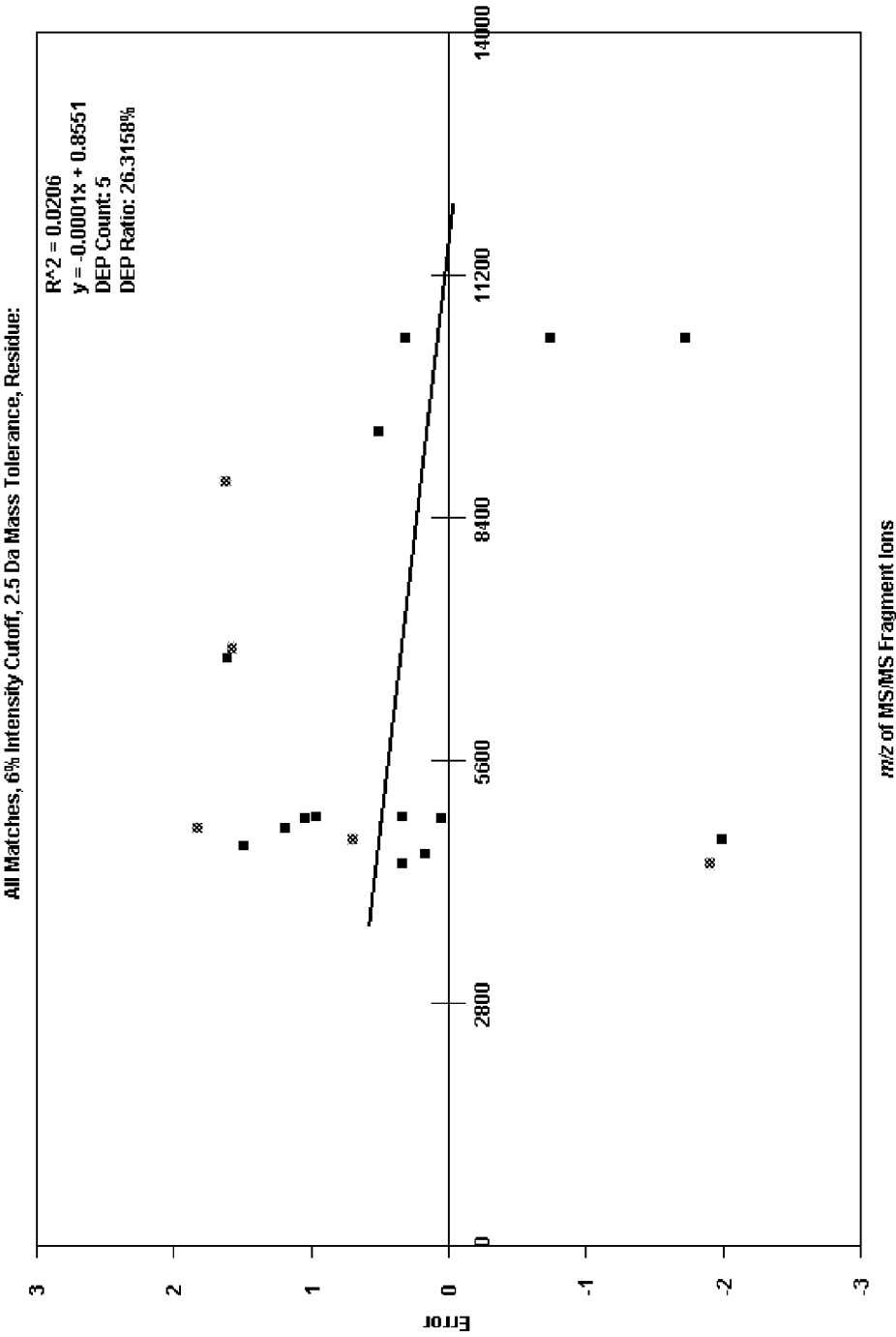


FIG. 10

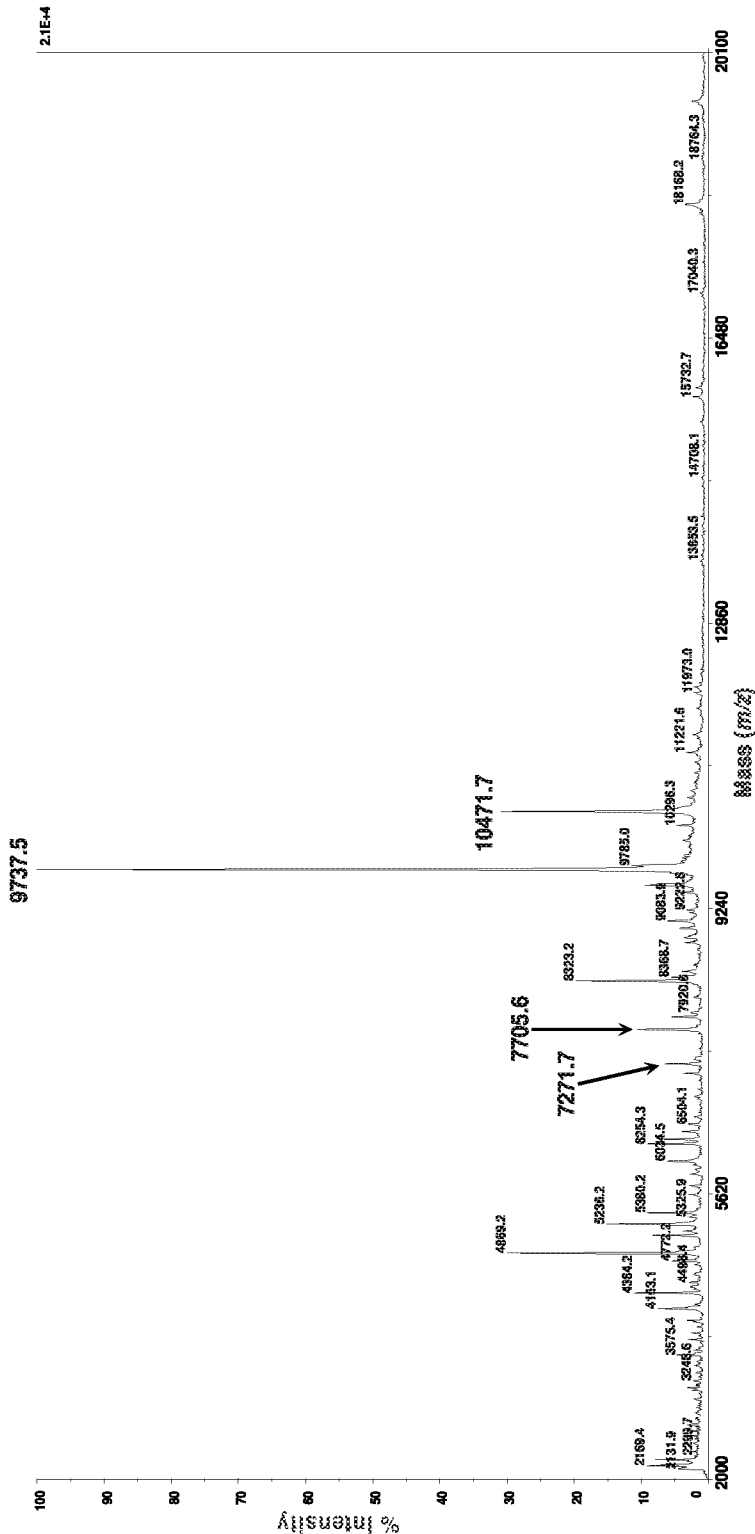
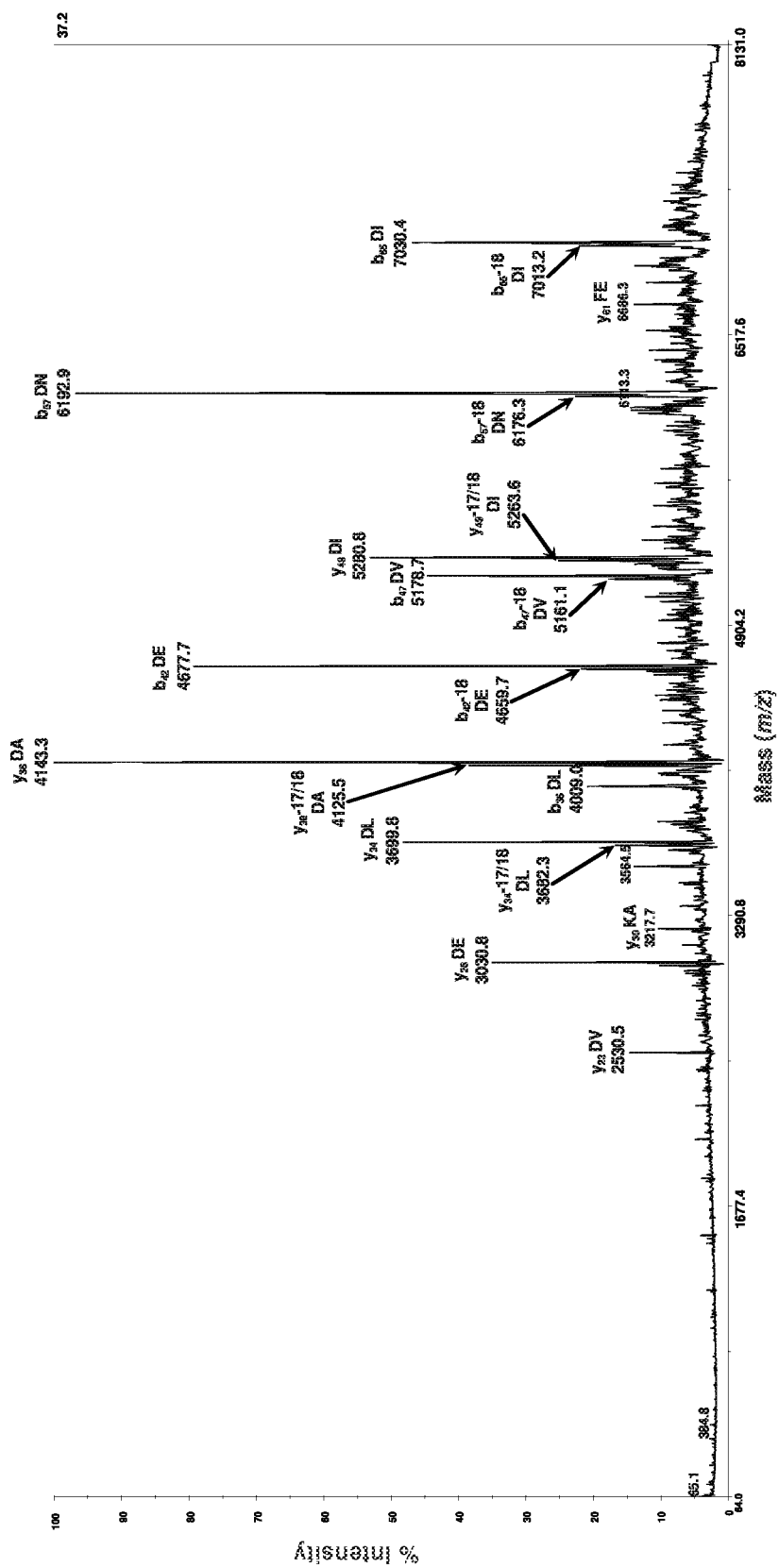


FIG. 11



Putative uncharacterized protein yahO of *E. coli* O157:H7 strain EDL-933

MKIISKMLVGALAFAVTNVYA**AELMTKAEFEKVESQYE**
KIGDISTSNEMSTADAKEDLIKKADEKGADVLVLTSGQT
DNKIHGTA**D**IYKKK

MW = 9930.41 Da

Mature protein MW = 7707.62 Da

UPF0379 protein yahO of *E. coli* strain K-12

MKIISKMLVGALALAVTNVYA**AELMTKAEFEKVESQYE**
KIGDISTSNEMSTADAKEDLIKKADEKGADVLVLTSGQT
DNKIHGTA**N**IYKKK

MW = 9895.41 Da

Mature protein MW = 7706.64 Da

FIG. 13