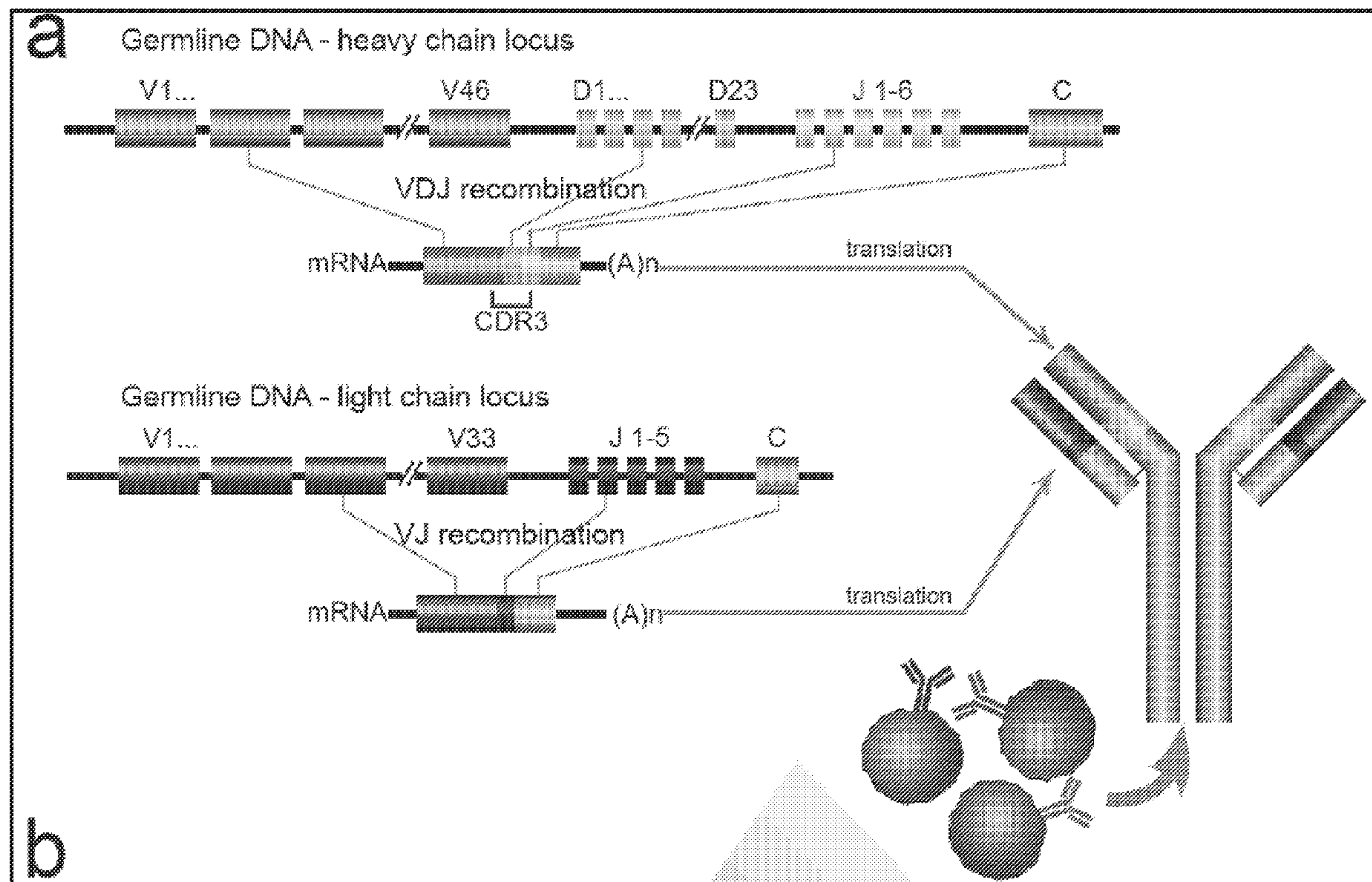




(86) **Date de dépôt PCT/PCT Filing Date:** 2011/10/11
(87) **Date publication PCT/PCT Publication Date:** 2012/04/12
(45) **Date de délivrance/Issue Date:** 2017/11/14
(85) **Entrée phase nationale/National Entry:** 2013/04/05
(86) **N° demande PCT/PCT Application No.:** US 2011/055801
(87) **N° publication PCT/PCT Publication No.:** 2012/048340
(30) **Priorité/Priority:** 2010/10/08 (US61/391,337)

(51) **Cl.Int./Int.Cl. C12Q 1/68** (2006.01),
C12Q 1/00 (2006.01)
(72) **Inventeurs/Inventors:**
CHURCH, GEORGE M., US;
VIGNEAULT, FRANCOIS, US;
LASERSON, URI, US;
BACHELET, IDO, US
(73) **Propriétaire/Owner:**
PRESIDENT AND FELLOWS OF HARVARD COLLEGE,
US
(74) **Agent:** KIRBY EADES GALE BAKER

(54) **Titre : SEQUENCAGE IMMUNITAIRE A HAUT DEBIT**
(54) **Title: HIGH-THROUGHPUT IMMUNE SEQUENCING**



(57) **Abrégé/Abstract:**
Methods and compositions for determining and/or monitoring the immune state of an individual are provided.



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

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
12 April 2012 (12.04.2012)

(10) International Publication Number
WO 2012/048340 A3

(51) International Patent Classification:
C12Q 1/68 (2006.01)

(21) International Application Number:
PCT/US2011/055801

(22) International Filing Date:
11 October 2011 (11.10.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/391,337 8 October 2010 (08.10.2010) US

(71) Applicant (for all designated States except US): **PRESIDENT AND FELLOWS OF HARVARD COLLEGE** [US/US]; 17 Quincy Street, Cambridge, MA 02138 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **CHURCH, George M.** [US/US]; 218 Kent Street, Brookline, MA 02446 (US). **VIGNEAULT, Francois** [CA/US]; 100 Station Landing, Unit 610, Medford, MA 02155 (US). **LASERSON, Uri** [US/US]; 134 Marlborough Street, #4, Boston, MA 02116 (US). **BACHELET, Ido** [IL/US]; 53 Harvard Avenue, Brookline, MA 02446 (US).

(74) Agent: **IWANICKI, John P.**; Banner & Witcoff, Ltd., 28 State Street, Suite 1800, Boston, MA 02109 (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, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, 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, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, 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, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

- with international search report (Art. 21(3))
- before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))

(88) Date of publication of the international search report:
21 June 2012

(54) Title: HIGH-THROUGHPUT IMMUNE SEQUENCING

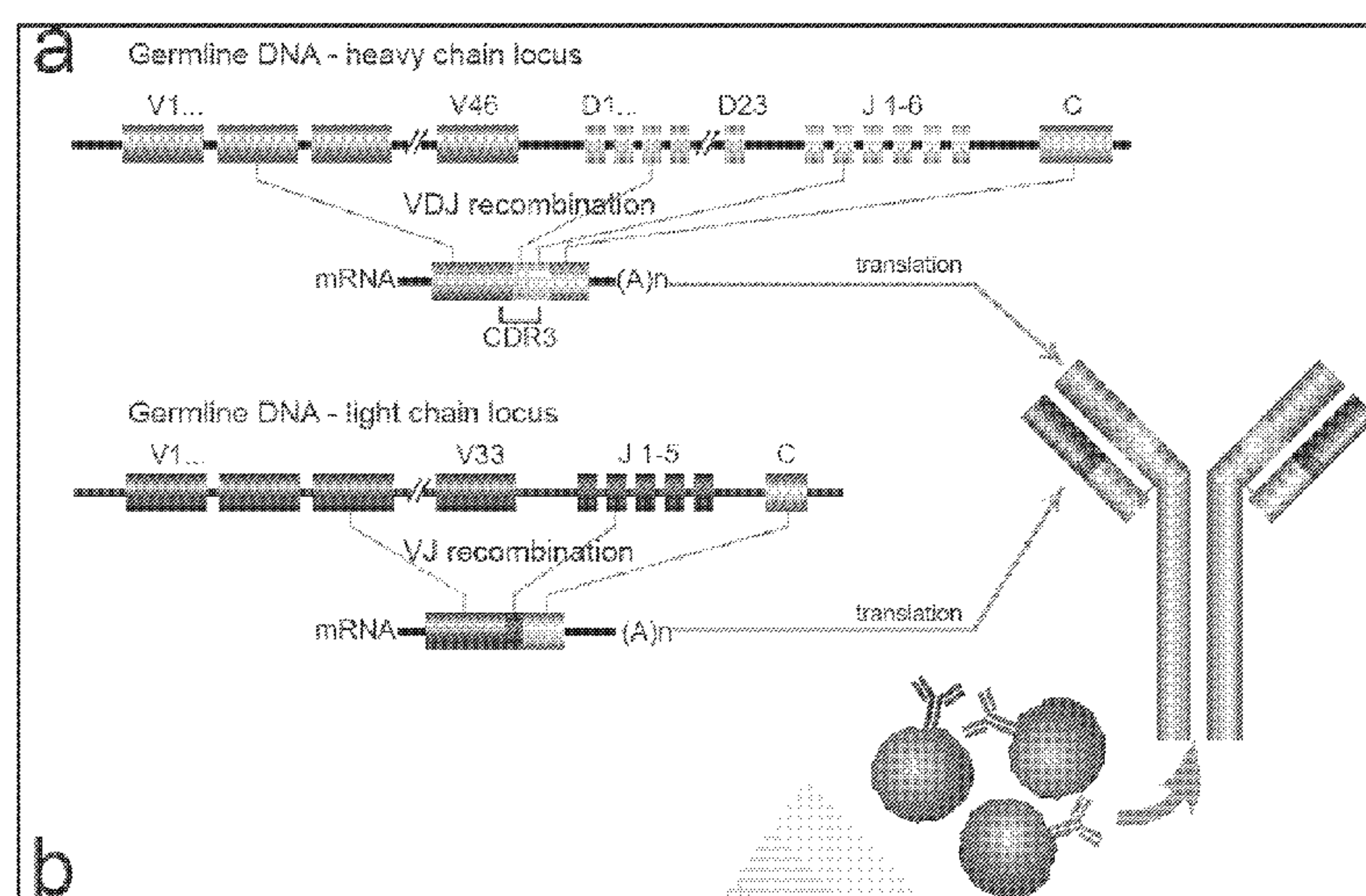


Figure 1

(57) Abstract: Methods and compositions for determining and/or monitoring the immune state of an individual are provided.

HIGH-THROUGHPUT IMMUNE SEQUENCING

[001]

[002]

FIELD

[003] The present invention relates to methods and compositions for diagnostics and therapeutics.

BACKGROUND

[004] The function of the adaptive immune system is largely mediated by lymphocytes (B and T cells) that express a diverse repertoire of immune receptors against virtually any foreign substance (Abbas et al., Cellular and Molecular Immunology, Edn. 6th. (Saunders Elsevier, Philadelphia; 2007); Murphy et al., Janeway's Immunobiology, Edn. 7th. (Garland Science, New York; 2008); Paul, Fundamental Immunology, Edn. 6th. (Wolters Kluwer/Lippincott Williams & Wilkins, Philadelphia; 2008)). In order to generate the repertoire of antibodies necessary for antigen recognition, each lymphocyte independently constructs a unique receptor through the process of VDJ recombination. *Id.* Each cell randomly selects a single V, D, and J gene segment through genetic recombination, introducing additional non-germline-encoded nucleotides at the junctions (Figure 1). This process creates antibody diversity, the majority of which is encoded in the heavy

chain complementarity determining region 3 (CDR3) (Rock et al. (1994,) *J. Exp. Med.* 179:323 ; Xu and Davis (2000) *Immunity* 13:37). The complexity and dynamics of a single human immune repertoire have yet to be deeply probed, and the ability to track the repertoire dynamically has not been demonstrated in any organism.

SUMMARY

- [004a]** Certain exemplary embodiments provide a method for identifying a therapeutic antibody, comprising: (a) linking polynucleotide sequences encoding a V_H and a V_L isolated from a single cell of a plurality of cells from a single subject to form paired sequences; (b) sequencing polynucleotides amplified from the paired sequences; (c) determining the frequency of two or more paired V_H and V_L sequences; (d) determining a variance from a reference library of the paired sequences from (c); and (e) identifying a therapeutic antibody based on (c) and (d).
- [004b]** Further exemplary embodiments provide a method for identifying a therapeutic antibody, comprising: (a) linking polynucleotide sequences encoding a V_H and a V_L isolated from a single cell to form paired sequences; (b) sequencing polynucleotides amplified from the paired sequences; (c) determining the frequency of two or more paired V_H and V_L sequences; (d) determining a variance from a reference library of the paired sequences from (c); and (e) identifying a therapeutic antibody based on (c) and (d).
- [005]** Characterizing the diversity and dynamics of the immune repertoire or "VDJ-ome" has significant implications in understanding the immune system, particularly in the context of personalized diagnostics and therapeutic discovery. Accordingly, the present invention is directed in part to methods and compositions that enable personalized clinical diagnostics and therapeutics utilizing the unique nucleic acid sequence information contained in an individual immune system. Toward this goal, high-throughput, high-resolution methods and compositions for profiling the antibody repertoire of a single individual are provided. For example, embodiments of the present

invention include the use of high-throughput sequencing technology to dynamically track the nature and extent of antibodies produced by an individual as a result of an immune reaction, to clone full antibodies of paired light and heavy chains from millions of single cells in parallel, and to select for high-affinity antibodies against multiple antigens in a single reaction.

[006] In certain exemplary embodiments, methods for determining an immune state of an individual are provided. The methods include the steps of obtaining nucleic acid sequences encoding lymphocyte receptors from the biological sample, and performing high-throughput sequencing of the amplified nucleic acid sequences to determine a plurality of sequences representing the immune state of the biological sample. In certain aspects, the biological sample is selected from the group consisting of blood, saliva, synovial fluid, cultured cells, a tissue section and a biopsy. In certain aspects, the biological sample is obtained from an individual. In other aspects, an immune state of the individual is determined. In still other aspects, the methods further include the step of amplifying the nucleic acid sequences encoding lymphocyte receptors prior to the step of

high-throughput sequencing, optionally using primers specific to heavy chain or light chain nucleic acid sequences. In yet other aspects, the methods include the step of obtaining lymphocytes (e.g., T cells, B cells or a combination thereof) from the biological sample prior to the step of purifying. According to one aspect, an individual's T cell receptor usage is correlated with the individual's MHC type to diagnose certain diseases or conditions. In certain aspects, the nucleic acid sequences encode one or more of V regions (heavy chain or light chain), D regions (heavy chain or light chain), J regions (heavy chain or light chain) and combinations thereof. In other aspects, the nucleic acid sequences are DNA (e.g., genomic DNA) or RNA (e.g., mRNA). In still other aspects, the methods include the step of comparing the immune state of the individual to a control sample to diagnose a disease or a disorder such as, e.g., an infection, an autoimmune disorder, a cellular proliferative disorder and any combination thereof. In other aspects, the plurality of sequences includes at least one common nucleic acid sequence.

[007] In certain exemplary embodiments, methods for monitoring the effect of an agent on an immune state of a biological sample are provided. The methods include the steps of isolating lymphocytes from the first biological sample, purifying a first set of nucleic acid sequences encoding lymphocyte receptors, performing high-throughput sequencing of the purified first set of nucleic acid sequences to provide a first reference library, contacting a second biological sample with an agent, isolating lymphocytes from the second biological sample, purifying a second set of nucleic acid sequences encoding lymphocyte receptors, performing high-throughput sequencing of the purified second set of nucleic acid sequences to provide a second reference library, and comparing the sequences of the first reference library and the second reference library to monitor the affect of an agent on an immune state of the biological sample. In certain aspects, the agent is selected from the group consisting of an antigen, a vaccine, a drug, a small molecule, a biologic and any combination thereof. In other aspects, the biological sample is selected from the group consisting of blood, saliva, synovial fluid, cultured cells, a tissue section and a biopsy. In other aspects, the biological sample is obtained from an individual. In still other aspects,

an immune state of the individual is determined. In yet other aspects, the individual is afflicted with a disease or disorder selected from the group consisting of an infection, an autoimmune disorder, a cellular proliferative disorder and any combination thereof. In certain aspects, efficacy of the agent for treating a disease or a disorder is determined. In other aspects, a prognosis for treating a disease or a disorder is determined. In other aspects, the sequencing steps are performed simultaneously or at separate times. In certain aspects, multiple samples are obtained over a time course and, optionally, the steps of isolating, purifying and performing high-throughput sequencing are performed for each of the multiple samples obtained over the time course.

[008] In certain exemplary embodiments, methods for monitoring an immune response of a biological sample are provided. The methods include the steps of obtaining a first biological sample at a first point in time, isolating lymphocytes from the first biological sample, purifying a first set of nucleic acid sequences encoding lymphocyte receptors, performing high-throughput sequencing of the purified first set of nucleic acid sequences to provide a first reference library, obtaining a second biological sample at a second point in time, isolating lymphocytes from the second biological sample, purifying a second set of nucleic acid sequences encoding lymphocyte receptors, performing high-throughput sequencing of the purified second set of nucleic acid sequences to provide a second reference library, and comparing the sequences of the first reference library and the second reference library to track an immune response in a biological sample. In certain aspects, the immune response is against an agent selected from the group consisting of an antigen, a vaccine, a drug, a small molecule, a biologic and any combination thereof. In other aspects, wherein the sequencing steps are performed simultaneously or at separate times. In yet other aspects, multiple samples are obtained over a time course and, optionally, the steps of isolating, purifying and performing high-throughput sequencing are performed for each of the multiple samples obtained over the time course. In still other aspects, a disease or disorder diagnosis is determined. In other aspects, the biological sample is selected from the group consisting of blood, saliva, synovial fluid,

cultured cells, a tissue section and a biopsy. In still other aspects, the biological sample is obtained from an individual. In other aspects, the immune state of the individual is determined.

[009] Further features and advantages of certain embodiments of the present invention will become more fully apparent in the following description of the embodiments and drawings thereof, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[010] The patent or application file contains at least one drawing executed in color. Copies of this patent or patent application publication with color drawing(s) will be provided by the Office upon request and payment of the necessary fee. The foregoing and other features and advantages of the present invention will be more fully understood from the following detailed description of illustrative embodiments taken in conjunction with the accompanying drawings in which:

[011] *Figure 1* depicts an overview of VDJ recombination and experimental design. An antibody contains a heavy chain and a light chain, where a unique V, D, and J segment are ligated together from a set of germline-encoded genes.

[012] *Figures 2A-2E* depict technical reproducibility and immune diversity of certain methods described herein. (A) The number of unique clones (top) and corresponding reads (bottom) shared between two sequencing replicate runs (SR1 and SR2) and an independent technical replicate run (TR1). A small number of shared unique clones accounted for the majority of reads. (B) Day 1 scatter plot of SR1 versus TR1 of clone frequencies demonstrated strong correlation and reproducibility of the sampling approach described herein. (C) For multiple sample sizes, bootstrap distributions were sampled by computing the Pearson correlation between 50 pairs of independently drawn samples. This showed that 10^5 reads were sufficient to achieve high correlations among independent samples. (D) Clone frequency abundance distributions from diversity

replicate 1 (DR1) and all time points from SR1+SR2+TR1. Clone frequencies spanned multiple orders of magnitude and were power-law distributed with approximately inverse square scaling. (E) Unique sequences were observed and estimates of the expressed human heavy chain diversity in 1 mL blood samples were made using the Chao1 and ACE non-parametric estimators. Numbers were computed using clone counts (post-clustering, error-corrected) as well as unique CDR3 junction sequences (unprocessed, non-clustered) (top). Estimates of the total expressed human heavy chain diversity in blood using multiple non-parametric estimators and the Arstila method (Arstila et al. (1999) *Science* 286:958) (bottom).

[013] **Figures 3A-3C** depict static repertoire analysis. (A) Distribution of all functional V and J segments observed in replicate runs across all time points shows highly non-uniform usage of gene segments. Circle sizes are proportional to read counts. (B) The CDR3 junction length distribution of all reads in SR1+SR2+TR1 (light) and all Ig heavy chain junctions in IMGT/LIGM (dark) showed large variability in CDR3 loop size. The sawtooth pattern reflected in-frame junctions. (C) Representative examples of CDR3 pseudo-spectratypes for various V-J combinations at multiple time points. The 16 amino acid peak was annotated in all spectratypes. Annotated numbers showed expansions of clones identified in Figure 4B, which confirmed the utility of the spectratype method for identifying candidates of interest.

[014] **Figures 4A-4D** depict dynamic repertoire analysis. (A) The time series of all clones in SR1+SR2+TR1 revealed significant fluctuations of the repertoire through time. A handful of clones showed significantly higher expression, while the extent of the fluctuations is shown on a logarithmic scale (inset). The appearance of many gradually rising clones from day -14 to day 0 was due to the lack of sampling resolution among these time points. (B) Self-organizing map (SOM) clustering of clone time series, where each cluster was located at a constant map position, and colors represent clone expression levels. Groups of clusters with similar dynamic behavior were selected and the corresponding clones are plotted as time series. Time series were colored based on the

most common isotype observed at the time of peak expression. Annotations referred to specific clones of interest corresponding to specific spectratype peaks in Figure 3C. (C) Dynamics of the global isotype distribution indicated potential class switching. Thick lines represented the mean of 6 replicate runs (shown as thin lines) with one standard deviation. (D) Specific isotype distributions for selected annotated clones; stacked bar charts represented the isotype distribution at a given time point, while the black line represents the clone frequency time series. Numbers referred to maximum clone frequency, with the corresponding number of reads. Isotype color scheme is consistent among panels B, C, and D.

- [015] *Figure 5* schematically depicts high throughput immune sequencing and analysis of immune and antibody repertoire.
- [016] *Figures 6A-6C* graphically depict FACS sorting analysis of selected sequences of Figure 4.
- [017] *Figures 7A-7C* graphically depict quartz crystal microbalance (QCM) analysis.
- [018] *Figure 8* depicts a graph showing ROC curves for the VDJ alignment algorithm described herein using IMGT/V-QUEST results as a reference. As expected, V region alignments were the most reliable, followed by J, followed by D.
- [019] *Figure 9* depicts scatter plots showing correlation of expression levels among replicates SR1, SR2, TR1, and OR3. Points in red are zero-valued. Barcode 1 (-14 d) was significantly lost in the preparation of replicate TR1. Note that OR3 consistently showed worse correlation. As it was produced with a freshly mixed primer set, without intending to be bound by scientific theory, this indicated that the protocol described herein was highly sensitive to the primer concentrations.

- [020] *Figure 10* graphically depicts coefficients of variation (CV) computed for various replicate comparisons. Higher clone frequencies showed smaller CV, consistent with the higher reproducibility of high-expression clones. CV values of 0.5 were attained at frequencies of 10^{-4} to 10^{-3} , indicating limits to the ability to accurately quantitate clone frequency before the onset of Poisson noise.
- [021] *Figure 11* graphically depicts clone frequency rank accumulation for replicate DR1 and the various time points of SR1+SR2+TR1.
- [022] *Figure 12A-12D* depict rarefaction curves using sampling without replacement. The final values (computed on the full data sets) are summarized herein. (A) Chao 1, linear and logarithmic axis. (B) ACE, linear and logarithmic axis. (C) Number of unique sequences discovered, linear and logarithmic axis. (D) Estimator value versus number of reads. Lack of positive correlation indicates that estimator values are independent of the sample sizes, and thus are sampled sufficiently.
- [023] *Figures 13A-13B* depict CDR3 pseudo-spectratypes for various V-J combinations at multiple time points for all the clones selected from Figure 4B.
- [024] *Figure 14* depicts replicate plots of time series of randomly selected clones. Each panel contains 10 randomly selected clones, plotted using data from SR1, SR2, and TR1, separately. Unique clones are plotted in the same color. Clones qualitatively tracked one another from separate replicates, showing high reproducibility of time series between replicates.
- [025] *Figures 15A-15B* depict k-means clustering of clone time series. (A) Clustering of all clones into 100 clusters using the Euclidean distance metric. (B) Similar clustering of all clones observed in at least two time points. Curves represent cluster means; their thickness is proportional to cluster size.

- [026] *Figures 16A-16C* depict isotype distributions of the most highly expressed clones at days 1 (A), 7 (B), and 21 (C). Stacked bar charts represent the isotype distribution at a given time point, while the black line represents the clone frequency time series. Numbers refer to maximum clone frequency, with the corresponding number of reads. IgA; green, IgG; red, IgM; blue.
- [027] *Figures 17A-17B* depict Collier de Perles (Ruiz and Lefranc (2002) *Immunogenetics* 53:857) representation of the anti-digoxin IMGT/LIGM entry (AR161172; top) and of the corresponding sequencing read found in a subject (FXQ8H8O01CY52L; bottom).
- [028] *Figure 18* graphically depicts the raw high quality read length distribution of a single run of 454 GS FLX sequencing. The most significant peak occurred at the expected size of 420 bp.
- [029] *Figures 19A-19B* depict (A) distribution of cophenetic distances for single and complete linkage hierarchical clustering on sequences from a V-J combination with a small number of reads; and (B) number of clusters obtained as a function of distance at which the linkage tree was cut. Both plots show rapid descents to 4.5 edits, followed by shoulders. This indicated 4.5 edits as the proper distance to clip the linkage tree to define clones.
- [030] *Figures 20A-20B* depict (A) distribution of log(fold change) for each transition in time series that were positive at all 8 time points; and (B) correlation between the standard deviation of the log(fold change) and the duration of the transition.
- [031] *Figure 21* depicts a first example alignment for hepatitis. Query: FXQ8H8O01DXEUI rank=0514859 x=1493.0 y=2520.0 length=408; Target: I55621 | anti-hepatitis B virus (HBV) surface antigen (HBsAg) (human); Model: affine:local:dna2dna; Raw score: 1740; Query range: 8 -> 398; Target range: 27 -> 423; aln_summary: FXQ8H8O01DXEUI 408 8 398 + I55621 423 27 423 + 1740 390 372 95.38.

- [032] **Figure 22** depicts a second example alignment for hepatitis. Query: FX4D8HT02FMIIW rank=0345098 x=2189.0 y=2486.0 length=408; Target: I55621 | anti-hepatitis B virus (HBV) surface antigen (HBsAg) (human); Model: affine:local:dna2dna; Raw score: 1743; Query range: 5 -> 404; Target range: 24 -> 423; aln_summary: FX4D8HT02FMIIW 408 5 404 + I55621 423 24 423 + 1743 399 371 92.
- [033] **Figure 23** depicts a junction-only alignment. Query: FXQ8H8O01CY52L [translate(1)]; Target: AR161172 | anti-digoxin; Model: protein2dna:local; Raw score: 58; Query range: 0 -> 9; Target range: 237 -> 264; aln_summary: FXQ8H8O01CY52L 9 0 9 . AR161172 282 237 264 + 58 9 9 100.00.
- [034] **Figure 24** depicts a full-read alignment. Query: FXQ8H8O01CY52L; Target: AR161172 | anti-digoxin; Model: coding2coding; Raw score: 512; Query range: 73 -> 355; Target range: 0 -> 282; aln_summary: FXQ8H8O01CY52L 390 73 355 + AR161172 282 0 282 + 512 94 93 98.94.
- [035] **Figure 25** depicts an overview of an experimental design according to certain aspects of the invention. Lymphocyte total RNA was extracted from blood samples collected at multiple time points following the vaccination of a human individual. A multiplex set of gene-specific primers were used to amplify the immunoglobulin heavy chain from B cell RNA, and processed into 454-compatible sequencing libraries. The sequences were aligned and clustered for repertoire analysis.
- [036] **Figure 26** shows a table depicting read sequencing distribution and analysis.
- [037] **Figure 27** schematically depicts exemplary chain coupling methods.
- [038] **Figure 28** schematically depicts exemplary cell insulation methods.
- [039] **Figure 29** schematically depicts exemplary methods for coupling heavy and light chains.

- [040] *Figure 30* schematically depicts exemplary methods for multiplex affinity selections.
- [041] *Figure 31* depicts an immune cell presenting an antibody containing a heavy chain and a light chain.
- [042] *Figure 32* schematically depicts a method of optimizing cross-linking PCR.
- [043] *Figure 33* schematically depicts a method of optimizing cell insulation.
- [044] *Figure 34* schematically depicts a method of characterizing cross-contamination.
- [045] *Figure 35* schematically depicts a method of characterizing cross-contamination using a TaqMan assay.
- [046] *Figure 36* schematically depicts generation of an antibody-antigen lookup table for influenza virus.
- [047] *Figure 37* schematically depicts antibodies coupled to antigens.
- [048] *Figure 38* schematically depicts a comparison of libraries against libraries for combinatorial selection of replicating antigen-antibody pairs (Bowley et al. (2009) *Proc. Natl. Acad. Sci. USA* 106(5):1380).
- [049] *Figure 39* schematically depicts exemplary bead emulsion capture methods.
- [050] *Figure 40* schematically depicts small oligonucleotide transfection of single cells.
- [051] *Figure 41* schematically depicts SOE-PCR.
- [052] *Figure 42* schematically depicts SOE-PCR USER ligation.

- [053] *Figure 43* schematically depicts SOE-PCR restriction ligation.
- [054] *Figure 44* schematically depicts 3x SOE-PCR.
- [055] *Figure 45* graphically depicts heavy and light chain distributions.
- [056] *Figure 46* depicts histograms of read lengths of light chains.
- [057] *Figure 47* depicts the distribution of junctions.
- [058] *Figure 48* depicts a clone time series.
- [059] *Figure 49* depicts a time series for both heavy and light chains using unique junctions.
- [060] *Figure 50* depicts a re-plot time series of highly expressed heavy and light chain junctions.
- [061] *Figure 51* depicts a gel showing Y adapter ligation.
- [062] *Figure 52* depicts a diagnostic gel showing PCR efficiency.
- [063] *Figure 53* depicts a gel showing a test pippin preparation with control sample number 8.
- [064] *Figure 54* depicts samples run on a pippin preparation.
- [065] *Figure 55* depicts imaging data showing clonality of beads. All three assays demonstrated clonality, so emulsion of a human genomic library worked well, even when using a hydrocycler (which is a major time saving and technically facile solution). Top row: 20x white light, 200% zoom. Middle row: 20x, Cy5, Cy3, FITC color align. Bottom row: 20x, 200% zoom, Cy5, Cy3, FITC color align.

- [066] *Figure 56* depicts imaging data of a single cell post lysis and PCR with VDJ capturing beads.
- [067] *Figure 57* depicts imaging data of PCR with VDJ capturing beads.
- [068] *Figure 58* depicts a gel showing individual primer testing. Lanes from left to right: 1, 100 bp ladder; 2, full; 3, full-2; 4, J1; 5, J2; 6, J3; 7, V30; 8, V29; 9, V28; 10, V27.
- [069] *Figure 59* depicts gels showing individual primer testing.
- [070] *Figure 60* depicts a gel showing combined Illumina primer overlap testing.
- [071] *Figure 61* depicts a gel showing an assembly.
- [072] *Figure 62* depicts a gel showing cDNA and coriel DNA.
- [073] *Figure 63* depicts an SDS PAGE gel. 200 volts, ran for 45 minutes.
- [074] *Figure 64* depicts an SDS PAGE gel. 200 volts, ran for 30 minutes.
- [075] *Figure 65* depicts an SDS PAGE gel. Ran for 35 minutes, blue staining.
- [076] *Figure 66* depicts neutralization assays.
- [077] *Figure 67* graphically depicts neutralization results.
- [078] *Figure 68* depicts a diagnostic gel used to confirm PCR efficiency.
- [079] *Figure 69* depicts a diagnostic gel used to confirm efficiency of blunting and ampure beads.

- [080] *Figure 70* depicts a diagnostic gel used to confirm AMPure efficiency.
- [081] *Figure 71* depicts a diagnostic gel used to confirm combined reaction efficiency.
- [082] *Figure 72* depicts a diagnostic gel used to confirm PCR efficiency using primers that amplify the final product.
- [083] *Figure 73* depicts gels pre-pippin extraction (left panel) and post-pippin extraction (right panel).
- [084] *Figure 74* depicts a diagnostic gel used to confirm PCR efficiency.
- [085] *Figure 75* depicts a diagnostic gel used to confirm ligation efficiency.
- [086] *Figure 76* depicts pippin test gels.

DETAILED DESCRIPTION

- [087] In certain exemplary embodiments, a set of technologies and analytical tools to efficiently survey the information recorded in the immune system, therefore enabling personalized clinical diagnostics and/or therapeutics, are provided. Certain aspects of the invention utilize next-generation sequencing in order to screen large amounts of antibody coding sequences from millions of single cells in a single reaction, in contrast to one cell per well assays typically performed in the art. Certain aspects of the invention are directed to sequencing and identifying portions of antibody coding sequences. Other aspects of the invention are directed to methods for expressing complex antibody libraries e.g., for use with functional screens and/or evolution analyses against a library of antigens or proteins. In certain aspects, the methods and compositions described herein can be used for biomarker identification for development of diagnostic or screening assays against infectious diseases, cellular proliferative disorders such as cancer, allergies and/or autoimmune disorders. In other aspects, the methods and compositions described herein

can be used for the discovery and development of therapeutics, using, e.g., monoclonal, multi-pooled monoclonal, polyclonal, fragment, full and/or partial antibody sequence(s).

- [088] Certain additional exemplary embodiments utilize the characterization of the diversity and dynamics of the immune repertoire according to the methods described herein to identify vaccine targets and neutralizing antibodies. Such characterization on an individual level is useful in methods of diagnosis and/or prognosis of autoimmunity and/or allergic diseases. The characterization methods described herein are further useful to create databases of antigen/antibody affinities which are useful for predicting antibody specificity from sequence and for diagnosis of diseases or conditions characterized by certain antigen/antibody affinities. The characterization methods described herein are still further useful in methods of determining transplant rejection and histoincompatibility based on analysis of the components of the immune repertoire of an individual at a single point in time or at various points in time. Antibodies characterized according to the methods described herein are useful in methods to extract and identify certain immune effectors and antigens from a sample or subject.
- [089] In certain exemplary embodiments, methods for determining the genetic state of a plurality of individual immune cells (e.g., lymphocytes) at a given point in time (e.g., a “snapshot” of a collection of individual immune cells) are provided. In certain aspects, the identity of heavy and/or light chains (i.e., an immune receptor nucleic acid sequence) for each of a plurality of receptors will be ascertained at one or more given points in time. A collection of immune cell identities is also referred to herein as an “immune repertoire” or a “VDJ-ome.”
- [090] Embodiments of the characterization methods described herein enable the identification, creation and/or use of rapid response antibodies in diagnostic, therapeutic or research methods. Such rapid response antibodies are expressed quickly, typically within hours, as a result of an immune challenge. According to aspects of the present invention, the rapid response antibodies can be characterized in their breadth of affinity against targets or

antigens and their ability to stimulate other required host immune functions, Rapid response antibodies identified by the methods described herein can be used as an initial therapeutic agent to rapidly assist the immune system in responding to a disease or other condition while the immune system develops additional antibodies.

[091] Embodiments of the present invention further include the use of human IGHV3-23-derived sequences or human IGHV1-69-derived sequences identified using high-throughput sequencing in functionally-used antibodies. For purposes of the present invention, derived sequences include the sequence itself, as well as any fragments or mutants thereof.

[092] Major factors in the failure of monoclonal antibody development have been the lack of transition from animal model to human as well as limited capacity of screening sufficient amount of antibody candidates in order to allow efficient discovery of therapeutics. The methods and compositions described herein solve these issues by allowing screening of the immune repertoire or immune response directly from one or more humans, therefore producing antibody candidates that are fully human (i.e., not humanized antibodies), thus avoiding the risks of immunogenicity. In certain aspects, the methods and compositions described herein are not limited in their throughput, allowing screening of an unlimited amount of antibody coding sequences simultaneously, which would not be possible using classical immunological methods.

[093] Since the methods and compositions described herein enable a large amount of sequences to be screened per sample, efficient investigation of the immune reaction in an individual can be performed. For example, the presence of one or more natural immune reaction(s) or specific antibody sequences made by the patient can be used as a diagnostic screen on a subject, e.g., a patient having a current or previous infection. The methods and compositions described herein allow for the instantaneous determination of antibody sequences in an individual at any point in time (e.g., because the RNA and/or DNA coding for the antibody protein is sequenced), instead of having to conduct complex and

non-multiplexable protein sequence determination. An overview of certain methods described herein is shown in Figure 5. An application of the methods according to certain aspects of the invention is depicted in Figure 25.

[094] According to certain aspects, methods of characterizing components of an immune response in an individual using high-throughput sequencing methods described herein enable selection of specific antibody sequences, including IgA isotype antibodies, useful in diagnosing a disease or condition, or therapeutically treating a disease or condition. The methods described herein enable methods of comparing immune reaction components in an individual at different points in time (time-series information) which is further useful in assigning function to identified antibodies. Such methods include the identification of antibodies useful in diagnostic or therapeutic methods, and include the identification of human-derived single domain antibodies (nanobodies) which are useful as affinity reagents, for example, as diagnostic or therapeutic biomarkers. The methods describe herein also allow the identification of human-derived sequences using high-throughput sequencing which can then be formatted into different affinity molecules, such as diabodies, nanobodies, doubles scFv and other components defined below as being an antibody.

[095] According to alternate aspects, once the methods described herein are used to identify the components in an individual resulting from an immune response (referred to herein as an immune repertoire), the components can be cloned and expressed to create antibody libraries using either cell-free expression systems or cloning into expression vectors for in-cell expression. The antibodies can then be tested or screened for their ability to bind to antigens, their affinity and/or neutralization ability. In addition, once the methods described herein are used to identify the components in an individual resulting from an immune response, the components can be cloned followed by gene synthesis of selected antibodies based on functional information and then expressed to create antibody libraries using either cell-free expression systems or cloning into expression vectors for in-cell expression. The antibodies can then be tested or screened for their ability to bind to

antigens, their affinity and/or neutralization ability. According to certain aspects, the antibodies resulting from the immune response that are identified by the methods described herein can be synthesized using parallel synthesis assembly from oligonucleotides or gene fragment derived standard oligonucleotide synthesis chemistry, from on chip printing synthesis, or any other method known to those skilled in the art.

- [096] According to an additional aspect, the antibodies identified by the high-throughput methods described herein are cloned directly into surface-display technologies including yeast, phage, bacterial, mammalian, ribosomal, mRNA display and the like and then screened for functional specificity or affinity or neutralization ability. An additional aspect of this embodiment includes the antibodies being cloned directly into surface-display technologies and then evolved by directed evolution approaches known to those skilled in the art and optionally, characterizing the directed evolution approaches at various points in time, at any step or at any cycle using high-throughput sequencing to identify the state of enrichment, level of diversity, etc. Such directed evolution methods can be performed with a single antigen, multiple antigens, or a large library of antigens in a single reaction.
- [097] As used herein, the terms “subject,” “individual” and “host” are intended to include living organisms such as mammals. Examples of subjects and hosts include, but are not limited to, horses, cows, camels, sheep, pigs, goats, dogs, cats, rabbits, guinea pigs, rats, mice (e.g., humanized mice), gerbils, non-human primates (e.g., macaques), humans and the like, non-mammals, including, e.g., non-mammalian vertebrates, such as birds (e.g., chickens or ducks) fish (e.g., sharks) or frogs (e.g., *Xenopus*), and non-mammalian invertebrates, as well as transgenic species thereof.
- [098] In certain aspects, a subject refers to a single organism (e.g., human). In certain aspects, or a group of individuals composing a small cohort having either a common immune factor to study and/or disease, and/or a cohort of individuals without the disease (e.g., negative/normal control) are provided. A subject from whom samples are obtained can

either be inflicted with a disease and/or disorder (e.g., one or more allergies, infections, cancers or autoimmune disorders or the like) or be injected with an agent (e.g., one or more of a vaccine, an antigen of interest, a drug, a small molecule, a biologic or the like) to stimulate an immune challenge, and be compared against a negative control subject which is not affected by the disease and/or not contacted with the agent.

[099] In certain aspects, one or more biological samples are isolated from one or more subjects. As used herein, a “biological sample” may be a single cell or many cells. A biological sample may comprise a single cell type or a combination of two or more cell types. A biological sample further includes a collection of cells that perform a similar function such as those found, for example, in a tissue. As used herein, a tissue includes, but is not limited to, epithelial tissue (e.g., skin, the lining of glands, bowel, skin and organs such as the liver, lung, kidney), endothelium (e.g., the lining of blood and lymphatic vessels), mesothelium (e.g., the lining of pleural, peritoneal and pericardial spaces), mesenchyme (e.g., cells filling the spaces between the organs, including fat, muscle, bone, cartilage and tendon cells), blood cells (e.g., red and white blood cells), neurons, germ cells (e.g., spermatozoa, oocytes), amniotic fluid cells, placenta, stem cells and the like. A tissue sample includes microscopic samples as well as macroscopic samples. In certain aspects, a biological sample is peripheral blood. In other aspects, a biological sample is a fluid such as saliva, synovial fluid, or the like. In still other aspects, a biological sample is from one or more cell cultures, tissue sections and/or biopsies.

[0100] In certain exemplary embodiments, an immune cell (e.g., a lymphocyte) fraction is isolated from a biological sample using any technique known to one of ordinary skill in the art. In certain aspects, a lymphocyte fraction is isolated using ficoll centrifugation. In other aspects, a lymphocyte fraction is immobilized on a substrate or a support (e.g., a substrate that binds one or more of B cells and/or T cells). The support can be simple square grids, checkerboard (e.g., offset) grids, hexagonal arrays and the like. Suitable supports include, but are not limited to, slides, beads, chips, particles, strands, gels, sheets, tubing, spheres, containers, capillaries, pads, slices, films, culture dishes, plates

(e.g., 96-well, 48-well, 24-well, 12-well, eight-well, six-well, four-well, single-well and the like), cell surfaces (e.g., *S. aureus* cells) and the like. In various embodiments, a solid support may be biological, non-biological, organic, inorganic, or any combination thereof.

[0101] In certain exemplary embodiments, beads and bead-based arrays are provided. As used herein, the term “bead” refers to a discrete particle that may be spherical (e.g., microspheres) or have an irregular shape. Beads may be as small as approximately 0.1 μm in diameter or as large approximately several millimeters in diameter. Beads may comprise a variety of materials including, but not limited to, paramagnetic materials, ceramic, plastic, glass, polystyrene, methylstyrene, acrylic polymers, titanium, latex, sepharose, cellulose, nylon and the like.

[0102] In accordance with certain examples, a support (e.g., a bead) may have functional groups attached to its surface which can be used to bind one or more reagents described herein to the bead. One or more reagents can be attached to a support (e.g., a bead) by hybridization, covalent attachment, magnetic attachment, affinity attachment and the like. For example, a support (e.g., a bead) can be coated with a secondary antibody for use with a primary antibody. In another example, a support (e.g., a bead) may be coated with glycidyl ether (epoxy) reactive groups and/or p-toluenesulphonyl (tosyl) reactive groups for use with a primary antibody. Beads coated with a variety of substrates are commercially available (Dynabeads, Invitrogen). Supports (e.g., beads) may also be functionalized using, for example, solid-phase chemistries known in the art (see, e.g., U.S. Pat. No. 5,919,523).

[0103] As used herein, the term “attach” refers to both covalent interactions and noncovalent interactions. A covalent interaction is a chemical linkage between two atoms or radicals formed by the sharing of a pair of electrons (i.e., a single bond), two pairs of electrons (i.e., a double bond) or three pairs of electrons (i.e., a triple bond). Covalent interactions are also known in the art as electron pair interactions or electron pair bonds. Noncovalent interactions include, but are not limited to, van der Waals interactions, hydrogen bonds,

weak chemical bonds (i.e., via short-range noncovalent forces), hydrophobic interactions, ionic bonds and the like. A review of noncovalent interactions can be found in Alberts et al., in *Molecular Biology of the Cell*, 3d edition, Garland Publishing, 1994.

[0104] In certain exemplary embodiments, nucleic acid sequences (e.g., DNA and/or RNA) are extracted from immune cells (e.g., lymphocytes, e.g., B cells and/or T cells). In certain aspects, nucleic acid sequences (e.g., DNA and/or RNA) are extracted from one or more enriched fractions immune cells (e.g., lymphocytes, e.g., B cells and/or T cells). In other aspects, nucleic acid sequences (e.g., DNA and/or RNA) are extracted from samples that have not been enriched for immune cells. In certain aspects, specific subsets of cell populations are preliminarily enriched, e.g., using FACS or paramagnetic beads, therefore allowing enrichment of a memory B cell versus plasma cell fraction, which can be useful to increase the quality of the data for downstream use, e.g., for use in prognosing, diagnosing and the like. In certain aspects, at least 25, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1,000, 2,000, 3,000, 4,000, 5,000, 6,000, 7,000, 8,000, 9,000, 10,000, 15,000, 20,000, 25,000, 30,000, 35,000, 40,000, 45,000, 50,000, 55,000, 60,000, 65,000, 70,000, 75,000, 80,000, 85,000, 90,000, 95,000, 100,000, 150,000, 200,000, 250,000, 300,000, 450,000, 500,000, 550,000, 600,000, 650,000, 700,000, 750,000, 800,000, 850,000, 900,000, 950,000, 1,000,000, 1,500,000, 2,000,000, 2,500,000, 3,000,000, 3,500,000, 4,000,000, 4,500,000, 5,000,000, 5,500,000, 6,000,000, 6,500,000, 7,000,000, 7,500,000, 8,000,000, 8,500,000, 9,000,000, 9,500,000, 10,000,000, 20,000,000, 30,000,000, 40,000,000, 50,000,000, 60,000,000, 70,000,000, 80,000,000, 90,000,000, 100,000,000, 200,000,000, 300,000,000, 400,000,000, 500,000,000, 600,000,000, 700,000,000, 800,000,000, 900,000,000, 1,000,000,000, 2,000,000,000, 3,000,000,000, 4,000,000,000, 5,000,000,000, 6,000,000,000, 7,000,000,000, 8,000,000,000, 9,000,000,000, 10,000,000,000, 15,000,000,000, 20,000,000,000, 30,000,000,000, 40,000,000,000, 50,000,000,000, 60,000,000,000, 70,000,000,000, 80,000,000,000, 90,000,000,000, 100,000,000,000, 200,000,000,000, 300,000,000,000, 400,000,000,000, 500,000,000,000, 600,000,000,000, 700,000,000,000, 800,000,000,000, 900,000,000,000,

1,000,000,000,000, 2,000,000,000,000, 3,000,000,000,000, 4,000,000,000,000, 5,000,000,000,000, 6,000,000,000,000, 7,000,000,000,000, 8,000,000,000,000, 9,000,000,000,000, 10,000,000,000,000 or more different nucleic acid sequences (e.g., DNA (e.g., genomic) and/or RNA (e.g., mRNA)) or any values in between and/or range(s) defined by the above numbers or ranges in between or overlapping are extracted from immune cells.

[0105] In certain exemplary embodiments, non-specific primers, degenerate primers, or specific primers (e.g., primers that hybridize to J and/or C-regions for the heavy and/or light chains of B or T cells) are used to amplify nucleic acid sequences (e.g., DNA and/or RNA). In certain aspects, cDNA (if using mRNA) or DNA (if using genomic DNA) is amplified by PCR using primer sets that hybridize to the antibody coding gene of the heavy and light chains of B cells and/or T cells. In certain aspects, the one set of primers hybridizes to V regions and another set of primers hybridizes to the C regions. In other aspects, certain primer sets hybridize to other locations on the V region or upstream in the leaders region (which is favorable for RNA sample, allowing amplification of properly spliced RNA converted cDNA and not of DNA), while other primer sets hybridize downstream of the C-region, and/or to the J region.

[0106] In certain aspects, a segment spanning the CDR3 region is amplified using primers that hybridize to either side of the CDR3 region. The further apart the primers are on the VDJ segments, the more antibody sequence will be recovered and sequenced, thus adding additional immune state information (e.g., information regarding CDR1, CDR2 and CDR3 regions and/or hypermutation(s) across the antibody coding segments). Because there are many V, D, J and C regions, degenerate primers can be designed that hybridize to many segments, or a single primer sequence can be used for each of these segments in order to amplify representatively the repertoire of the subject (e.g., via multiplex PCR). In other aspects, amplification of a very specific unique set of VDJ combination is performed using one set of primers. In certain aspects, pull down-assays can be used to enrich immune cell nucleic acid sequences (e.g., RNA and/or DNA). In certain aspects, a

pull-down assay using either RNA or DNA probe complementary to some part of the VDJ segments of interests is performed. In certain aspects, labels are used to enrich immune cell nucleic acid sequences (e.g., RNA and/or DNA) (e.g., fluorophores for FACS sorting, biotin with streptavidin, covalently coated beads, and the like). Targeted enrichment strategies can also be used such as the use of molecular inversion probes, or by the use of array hybridization.

[0107] As used herein, the term “hybridization” refers to the process in which two single-stranded polynucleotides bind non-covalently to form a stable double-stranded polynucleotide. The term “hybridization” may also refer to triple-stranded hybridization. The resulting (usually) double-stranded polynucleotide is a “hybrid” or “duplex.” “Hybridization conditions” will typically include salt concentrations of less than about 1 M, more usually less than about 500 mM and even more usually less than about 200 mM. Hybridization temperatures can be as low as 5 °C, but are typically greater than 22 °C, more typically greater than about 30 °C, and often in excess of about 37 °C. Hybridizations are usually performed under stringent conditions, i.e., conditions under which a probe will hybridize to its target subsequence. Stringent conditions are sequence-dependent and are different in different circumstances. Longer fragments may require higher hybridization temperatures for specific hybridization. As other factors may affect the stringency of hybridization, including base composition and length of the complementary strands, presence of organic solvents and extent of base mismatching, the combination of parameters is more important than the absolute measure of any one alone. Generally, stringent conditions are selected to be about 5 °C lower than the T_m for the specific sequence at a defined ionic strength and pH. Exemplary stringent conditions include salt concentration of at least 0.01 M to no more than 1 M Na ion concentration (or other salts) at a pH 7.0 to 8.3 and a temperature of at least 25 °C. For example, conditions of 5XSSPE (750 mM NaCl, 50 mM Na phosphate, 5 mM EDTA, pH 7.4) and a temperature of 25-30 °C are suitable for allele-specific probe hybridizations. For stringent conditions, see for example, Sambrook, Fritsche and Maniatis, *Molecular*

Cloning A Laboratory Manual, 2nd Ed. Cold Spring Harbor Press (1989) and Anderson *Nucleic Acid Hybridization*, 1st Ed., BIOS Scientific Publishers Limited (1999). “Hybridizing specifically to” or “specifically hybridizing to” or like expressions refer to the binding, duplexing, or hybridizing of a molecule substantially to or only to a particular nucleotide sequence or sequences under stringent conditions when that sequence is present in a complex mixture (e.g., total cellular) DNA or RNA.

[0108] In certain exemplary embodiments, methods for amplifying nucleic acid sequences are provided. Exemplary methods for amplifying nucleic acids include the polymerase chain reaction (PCR) (see, e.g., Mullis et al. (1986) *Cold Spring Harb. Symp. Quant. Biol.* 51 Pt 1:263 and Cleary et al. (2004) *Nature Methods* 1:241; and U.S. Patent Nos. 4,683,195 and 4,683,202), anchor PCR, RACE PCR, ligation chain reaction (LCR) (see, e.g., Landegran et al. (1988) *Science* 241:1077-1080; and Nakazawa et al. (1994) *Proc. Natl. Acad. Sci. U.S.A.* 91:360-364), self sustained sequence replication (Guatelli et al. (1990) *Proc. Natl. Acad. Sci. U.S.A.* 87:1874), transcriptional amplification system (Kwoh et al. (1989) *Proc. Natl. Acad. Sci. U.S.A.* 86:1173), Q-Beta Replicase (Lizardi et al. (1988) *BioTechnology* 6:1197), recursive PCR (Jaffe et al. (2000) *J. Biol. Chem.* 275:2619; and Williams et al. (2002) *J. Biol. Chem.* 277:7790), the amplification methods described in U.S. Patent Nos. 6,391,544, 6,365,375, 6,294,323, 6,261,797, 6,124,090 and 5,612,199, isothermal amplification (e.g., rolling circle amplification (RCA), hyperbranched rolling circle amplification (HRCA), strand displacement amplification (SDA), helicase-dependent amplification (HDA), PWGA) or any other nucleic acid amplification method using techniques well known to those of skill in the art.

[0109] “Polymerase chain reaction,” or “PCR,” refers to a reaction for the *in vitro* amplification of specific DNA sequences by the simultaneous primer extension of complementary strands of DNA. In other words, PCR is a reaction for making multiple copies or replicates of a target nucleic acid flanked by primer binding sites, such reaction comprising one or more repetitions of the following steps: (i) denaturing the target nucleic acid, (ii) annealing primers to the primer binding sites, and (iii) extending the

primers by a nucleic acid polymerase in the presence of nucleoside triphosphates. Usually, the reaction is cycled through different temperatures optimized for each step in a thermal cycler instrument. Particular temperatures, durations at each step, and rates of change between steps depend on many factors well-known to those of ordinary skill in the art, e.g., exemplified by the references: McPherson et al., editors, *PCR: A Practical Approach* and *PCR2: A Practical Approach* (IRL Press, Oxford, 1991 and 1995, respectively). For example, in a conventional PCR using Taq DNA polymerase, a double stranded target nucleic acid may be denatured at a temperature greater than 90 °C, primers annealed at a temperature in the range 50-75 °C, and primers extended at a temperature in the range 72-78 °C.

[0110] The term “PCR” encompasses derivative forms of the reaction, including but not limited to, RT-PCR, real-time PCR, nested PCR, quantitative PCR, multiplexed PCR, assembly PCR and the like. Reaction volumes range from a few hundred nanoliters, e.g., 200 nL, to a few hundred microliters, e.g., 200 microliters. “Reverse transcription PCR,” or “RT-PCR,” means a PCR that is preceded by a reverse transcription reaction that converts a target RNA to a complementary single stranded DNA, which is then amplified, e.g., Tecott et al., U.S. Patent No. 5,168,038. “Real-time PCR” means a PCR for which the amount of reaction product, i.e., amplicon, is monitored as the reaction proceeds. There are many forms of real-time PCR that differ mainly in the detection chemistries used for monitoring the reaction product, e.g., Gelfand et al., U.S. Patent No. 5,210,015 (“Taqman”); Wittwer et al., U.S. Patent Nos. 6,174,670 and 6,569,627 (intercalating dyes); Tyagi et al., U.S. Patent No. 5,925,517 (molecular beacons). Detection chemistries for real-time PCR are reviewed in Mackay et al., *Nucleic Acids Research*, 30:1292-1305 (2002). “Nested PCR” means a two-stage PCR wherein the amplicon of a first PCR becomes the sample for a second PCR using a new set of primers, at least one of which binds to an interior location of the first amplicon. As used herein, “initial primers” in reference to a nested amplification reaction mean the primers used to generate a first amplicon, and “secondary primers” mean the one or more primers used to generate a

second, or nested, amplicon. "Multiplexed PCR" means a PCR wherein multiple target sequences (or a single target sequence and one or more reference sequences) are simultaneously carried out in the same reaction mixture, e.g. Bernard et al. (1999) *Anal. Biochem.*, 273:221-228 (two-color real-time PCR). Usually, distinct sets of primers are employed for each sequence being amplified. "Quantitative PCR" means a PCR designed to measure the abundance of one or more specific target sequences in a sample or specimen. Techniques for quantitative PCR are well-known to those of ordinary skill in the art, as exemplified in the following references: Freeman et al., *Biotechniques*, 26:112-126 (1999); Becker-Andre et al., *Nucleic Acids Research*, 17:9437-9447 (1989); Zimmerman et al., *Biotechniques*, 21:268-279 (1996); Diviacco et al., *Gene*, 122:3013-3020 (1992); Becker-Andre et al., *Nucleic Acids Research*, 17:9437-9446 (1989); and the like.

[0111] In certain exemplary embodiments, methods of determining the sequence identities of nucleic acid sequences are provided. Determination of the sequence of a nucleic acid sequence of interest (e.g., immune cell nucleic acid sequences) can be performed using variety of sequencing methods known in the art including, but not limited to, sequencing by hybridization (SBH), sequencing by ligation (SBL), quantitative incremental fluorescent nucleotide addition sequencing (QIFNAS), stepwise ligation and cleavage, fluorescence resonance energy transfer (FRET), molecular beacons, TaqMan reporter probe digestion, pyrosequencing, fluorescent *in situ* sequencing (FISSEQ), FISSEQ beads (U.S. Pat. No. 7,425,431), wobble sequencing (PCT/US05/27695), multiplex sequencing (U.S. Serial No. 12/027,039, filed February 6, 2008; Porreca et al (2007) *Nat. Methods* 4:931), polymerized colony (POLONY) sequencing (U.S. Patent Nos. 6,432,360, 6,485,944 and 6,511,803, and PCT/US05/06425); nanogrid rolling circle sequencing (ROLONY) (U.S. Serial No. 12/120,541, filed May 14, 2008), allele-specific oligo ligation assays (e.g., oligo ligation assay (OLA), single template molecule OLA using a ligated linear probe and a rolling circle amplification (RCA) readout, ligated padlock probes, and/or single template molecule OLA using a ligated circular padlock probe and a

rolling circle amplification (RCA) readout) and the like. High-throughput sequencing methods, e.g., on cyclic array sequencing using platforms such as Roche 454, Illumina Solexa, ABI-SOLiD, ION Torrents, Complete Genomics, Pacific Bioscience, Helicos, Polonator platforms (Worldwide Web Site: Polonator.org), and the like, can also be utilized. High-throughput sequencing methods are described in U.S. Serial No. 61/162,913, filed March 24, 2009. A variety of light-based sequencing technologies are known in the art (Landegren et al. (1998) *Genome Res.* 8:769-76; Kwok (2000) *Pharmacogenomics* 1:95-100; and Shi (2001) *Clin. Chem.* 47:164-172).

[0112] In certain exemplary embodiments, enriched VDJ segments are transformed so that they can be compatible with next-generation high-throughput sequencing technologies. In certain aspects, compatible sequencing adapters are ligated or added using PCR and primers with overhang on the VDJ segments. In other aspects, the adapter is bar-coded so that many samples can be pooled together and sequenced on a single instrument run. According to one aspect, samples belonging to the same research study are processed as much as possible in the same manner at the same time to achieve high degree of correlation and maximize quality of the data obtained. Ideally, read length should be sufficient to enable the identification of each V, D, and J, segment originated from each unique molecule. Use of bar-coding can also be used to achieve remapping of short segments to their original starting molecule (PMID: 20081835).

[0113] In certain exemplary embodiments, sequencing data are aligned against known or expected V, D and J segments using personal database, NCBI databases, IMGT databases or the like to aid in the identification of V, D and J segments. In this manner, V and J segments can be identified, allowing the extraction of the central junctional region of the V with D and D with J segments, composing the CDR3 region. All the sequencing reads (e.g., millions), can be binned according to their V and J usage, and clustered for their unique CDR3 using various clustering algorithms described herein. This will allow grouping of highly similar VDJ segments into VDJ clones, which likely encode for the same or a highly similar antibody. Each read (or unique VDJ segments sequenced) and/or

each VDJ clone (or highly similar VDJ segments from a same cluster group) can then be analyzed for biological relevancy (e.g., for a disease, disorder and/or immune response).

[0114] Clone frequency can be plotted to examine reproducibility among experiments and/or to identify unique or relevant VDJ clones of different expression across different samples or cohorts. In the case of time course study, clone frequency can be plotted in a time course plot to identify the dynamic of expression of the antibody expressed in the subject (Figure 4A). This allows for zoning on potential antibody sequences that are part of the immune response against the immune challenge that represents potential therapeutic or relevant biomarkers against the immune challenge (Figure 4B). In the case of static analysis research for a disease (for example various samples from a cohort having a disease against a cohort not having the disease), comparative clustering of which sequence has been made and at which frequency in one cohort against another can reveal therapeutics or biomarker candidates. Further study of the divergence of an antibody sequence (such as somatic hyper-mutation accumulation) of VDJ reads or clones can also be used to reveal potential candidates of interest as well as other biological information that can be derived from the data such as isotype distribution of the antibodies (such as IgA, IgD, IgE, IgG and IgM), which can be used to focus on a specific type of antibody or pseudo-spectratyping, for example.

[0115] In certain exemplary embodiments, methods of determining and/or analyzing an immune state of an individual include the step obtaining immune cells from a biological sample are provided. As used herein, the term “immune cell” is intended to include, but is not limited to, a cell that is of hematopoietic origin and that plays a role in the immune response. Immune cells include, but are not limited to, lymphocytes, such as B cells and T cells; natural killer cells; and myeloid cells, such as monocytes, macrophages, eosinophils, mast cells, basophils, and granulocytes.

[0116] The term “antigen presenting cell” includes professional antigen presenting cells (e.g., B lymphocytes, monocytes, dendritic cells, and Langerhans cells) as well as other antigen

presenting cells (e.g., keratinocytes, endothelial cells, astrocytes, fibroblasts, and oligodendrocytes).

[0117] As used herein, the term “immune response” includes, but is not limited to, T cell-mediated and/or B cell-mediated immune responses that are influenced by modulation of T cell costimulation. Exemplary immune responses include B cell responses (e.g., antibody production) T cell responses (e.g., cytokine production, and cellular cytotoxicity) and activation of cytokine responsive cells, e.g., macrophages. As used herein, the term “down-modulation” with reference to the immune response includes a diminution in any one or more immune responses, while the term “up-modulation” with reference to the immune response includes an increase in any one or more immune responses. It will be understood that up-modulation of one type of immune response may lead to a corresponding down-modulation in another type of immune response. For example, up-modulation of the production of certain cytokines (e.g., IL-10) can lead to down-modulation of cellular immune responses.

[0118] As used herein, the term “antibody” refers to immunoglobulin molecules and immunologically active portions of immunoglobulin molecules, i.e., molecules that contain an antigen binding site which specifically binds (immunoreacts with) an antigen. The term “antibody” is understood in its full general term to include components in the immune chain including an antibody, immunoglobulin, B cell receptor, antibody heavy chain, antibody light chain, T cell receptor (TCR), TCR alpha chain, TCR beta chain, TCR gamma chain, TCR delta chain or any variations or modification of antibody chains and further including scFv, Fab, Fab2, Fab3, Bis-scFv, minibody, triabody, diabody, tetrabody, nanobody and any and all various isotype variants of antibodies including IgG, IgA, IgD, IgM and IgE. Examples of immunologically active portions of immunoglobulin molecules include F(ab) and F(ab')₂ fragments which can be generated by treating the antibody with an enzyme such as pepsin. In certain exemplary embodiments, polyclonal and monoclonal antibodies are provided that bind one or more immune cell antigens. The terms “monoclonal antibody” and “monoclonal antibody

composition,” as used herein, refer to a population of antibody molecules that contains only one species of an antigen binding site capable of immunoreacting with a particular epitope of an immune cell antigen. A monoclonal antibody composition thus typically displays a single binding affinity for a particular immune cell antigen with which it immunoreacts.

[0119] In certain exemplary embodiments, screening assays for identifying immune system modulators, i.e., candidate or test compounds or agents (e.g., antibodies, peptides, cyclic peptides, peptidomimetics, small molecules, small organic molecules, or other drugs) are provided. The test compounds described herein can be obtained using any of the numerous approaches in combinatorial library methods known in the art, including: biological libraries; spatially addressable parallel solid phase or solution phase libraries; synthetic library methods requiring deconvolution; the “one-bead one-compound” library method; and synthetic library methods using affinity chromatography selection. The biological library approach is limited to peptide libraries, while the other four approaches are applicable to peptide, non-peptide oligomer or small molecule libraries of compounds (Lam, K. S. (1997) *Anticancer Drug Des.* 12:145).

[0120] In certain exemplary embodiments, one or more agents or pharmaceutically acceptable salts thereof described herein are provided in a pharmaceutically acceptable carrier. As used herein, the language “pharmaceutically acceptable carrier” is intended to include any and all solvents, dispersion media, coatings, antibacterial and antifungal agents, isotonic and absorption delaying agents, and the like, compatible with pharmaceutical administration. The use of such media and agents for pharmaceutically active substances is well known in the art. Except insofar as any conventional media or agent is incompatible with the active compound, use thereof in the compositions is contemplated. Supplementary active compounds can also be incorporated into the compositions. Pharmaceutically acceptable carriers and their formulations are known to those skilled in the art and described, for example, in Remington’s Pharmaceutical Sciences, (19th edition), ed. A. Gennaro, 1995, Mack Publishing Company, Easton, PA.

[0121] In certain exemplary embodiments, pharmaceutical formulations of one or more agents described herein or pharmaceutically acceptable salts thereof, are administered by intravenous injection, intraperitoneal injection, oral administration or by other parenteral routes (e.g. intradermal, subcutaneous, oral (e.g., inhalation), transdermal (topical), transmucosal, and rectal administration), or by intrathecal and intraventricular injections into the CNS, in an admixture with a pharmaceutically acceptable carrier adapted for the route of administration.

[0122] Solutions or suspensions used for parenteral, intradermal, subcutaneous or central nervous system application can include the following components: a sterile diluent such as water for injection, saline solution, fixed oils, polyethylene glycols, glycerin, propylene glycol or other synthetic solvents; antibacterial agents such as benzyl alcohol or methyl parabens; antioxidants such as ascorbic acid or sodium bisulfite; chelating agents such as ethylenediaminetetraacetic acid; buffers such as acetates, citrates or phosphates and agents for the adjustment of tonicity such as sodium chloride or dextrose. pH can be adjusted with acids or bases, such as hydrochloric acid or sodium hydroxide. The parenteral preparation can be enclosed in ampoules, disposable syringes or multiple dose vials made of glass or plastic.

[0123] Methods well known in the art for making formulations are found, for example, in Remington's Pharmaceutical Sciences (19th edition), ed. A. Gennaro, 1995, Mack Publishing Company, Easton, PA. Compositions intended for oral use may be prepared in solid or liquid forms according to any method known to the art for the manufacture of pharmaceutical compositions. The compositions may optionally contain sweetening, flavoring, coloring, perfuming, and/or preserving agents in order to provide a more palatable preparation. Solid dosage forms for oral administration include capsules, tablets, pills, powders, and granules. In such solid forms, the active compound is admixed with at least one inert pharmaceutically acceptable carrier or excipient. These may include, for example, inert diluents, such as calcium carbonate, sodium carbonate, lactose, sucrose, starch, calcium phosphate, sodium phosphate, or kaolin. Binding agents,

buffering agents, and/or lubricating agents (e.g., magnesium stearate) may also be used. Tablets and pills can additionally be prepared with enteric coatings.

[0124] Pharmaceutical compositions suitable for injectable use include sterile aqueous solutions (where water soluble) or dispersions and sterile powders for the extemporaneous preparation of sterile injectable solutions or dispersion. For intravenous administration, suitable carriers include physiological saline, bacteriostatic water, CREMOPHOR ELTM (BASF, Parsippany, NJ) or phosphate buffered saline (PBS). In all cases, the composition must be sterile and should be fluid to the extent that easy syringability exists. It must be stable under the conditions of manufacture and storage and must be preserved against the contaminating action of microorganisms such as bacteria and fungi. The carrier can be a solvent or dispersion medium containing, for example, water, ethanol, polyol (for example, glycerol, propylene glycol, and liquid polyethylene glycol, and the like), and suitable mixtures thereof. The proper fluidity can be maintained, for example, by the use of a coating such as lecithin, by the maintenance of the required particle size in the case of dispersion and by the use of surfactants. Prevention of the action of microorganisms can be achieved by various antibacterial and antifungal agents, for example, parabens, chlorobutanol, phenol, ascorbic acid, thimerosal, and the like. In certain exemplary embodiments, isotonic agents, for example, sugars, polyalcohols such as mannitol, sorbitol, and/or sodium chloride, will be included in the composition. Prolonged absorption of the injectable compositions can be brought about by including in the composition an agent which delays absorption, for example, aluminum monostearate and gelatin.

[0125] Sterile, injectable solutions can be prepared by incorporating agents described herein or pharmaceutically acceptable salts thereof in the required amount in an appropriate solvent with one or a combination of ingredients enumerated above, as required, followed by filtered sterilization. Generally, dispersions are prepared by incorporating the active compound into a sterile vehicle which contains a basic dispersion medium and the required other ingredients from those enumerated above. In the case of sterile powders

for the preparation of sterile injectable solutions, exemplary methods of preparation are vacuum drying and freeze-drying which yields a powder of the active ingredient plus any additional desired ingredient from a previously sterile-filtered solution thereof.

[0126] Oral compositions generally include an inert diluent or an edible carrier. They can be enclosed in gelatin capsules or compressed into tablets. For the purpose of oral therapeutic administration, the active compound can be incorporated with excipients and used in the form of tablets, troches, or capsules. Oral compositions can also be prepared using a fluid carrier for use as a mouthwash, wherein the compound in the fluid carrier is applied orally and swished and expectorated or swallowed. Pharmaceutically compatible binding agents, and/or adjuvant materials can be included as part of the composition. The tablets, pills, capsules, troches and the like can contain any of the following ingredients, or compounds of a similar nature: A binder such as microcrystalline cellulose, gum tragacanth or gelatin; an excipient such as starch or lactose, a disintegrating agent such as alginic acid, Primogel, or corn starch; a lubricant such as magnesium stearate or Sterotes; a glidant: such as colloidal silicon dioxide; a sweetening agent such as sucrose or saccharin; or a flavoring agent such as peppermint, methyl salicylate, or orange flavoring.

[0127] In one embodiment, one or more agents described herein or pharmaceutically acceptable salts thereof are prepared with carriers that will protect the compound against rapid elimination from the body, such as a controlled release formulation, including implants and microencapsulated delivery systems. Biodegradable, biocompatible polymers can be used, such as ethylene vinyl acetate, polyanhydrides, polyglycolic acid, collagen, polyorthoesters, and polylactic acid. Methods for preparation of such formulations will be apparent to those skilled in the art. The materials can also be obtained commercially from Alza Corporation and Nova Pharmaceuticals, Inc. Liposomal suspensions (including liposomes targeted to infected cells with monoclonal antibodies to viral antigens) can also be used as pharmaceutically acceptable carriers. These may be prepared according to methods known to those skilled in the art, for example, as described in U.S. Patent No. 4,522,811.

[0128] Nasal compositions generally include nasal sprays and inhalants. Nasal sprays and inhalants can contain one or more active components and excipients such as preservatives, viscosity modifiers, emulsifiers, buffering agents and the like. Nasal sprays may be applied to the nasal cavity for local and/or systemic use. Nasal sprays may be dispensed by a non-pressurized dispenser suitable for delivery of a metered dose of the active component. Nasal inhalants are intended for delivery to the lungs by oral inhalation for local and/or systemic use. Nasal inhalants may be dispensed by a closed container system for delivery of a metered dose of one or more active components.

[0129] In one embodiment, nasal inhalants are used with an aerosol. This is accomplished by preparing an aqueous aerosol, liposomal preparation or solid particles containing the compound. A non-aqueous (e.g., fluorocarbon propellant) suspension could be used. Sonic nebulizers may be used to minimize exposing the agent to shear, which can result in degradation of the compound.

[0130] Ordinarily, an aqueous aerosol is made by formulating an aqueous solution or suspension of the agent together with conventional pharmaceutically acceptable carriers and stabilizers. The carriers and stabilizers vary with the requirements of the particular compound, but typically include nonionic surfactants (Tweens, Pluronic, or polyethylene glycol), innocuous proteins like serum albumin, sorbitan esters, oleic acid, lecithin, amino acids such as glycine, buffers, salts, sugars or sugar alcohols. Aerosols generally are prepared from isotonic solutions.

[0131] Systemic administration can also be by transmucosal or transdermal means. For transmucosal or transdermal administration, penetrants appropriate to the barrier to be permeated are used in the formulation. Such penetrants are generally known in the art, and include, for example, for transmucosal administration, detergents, bile salts, and fusidic acid derivatives. Transmucosal administration can be accomplished through the use of nasal sprays or suppositories. For transdermal administration, the active

compounds are formulated into ointments, salves, gels, or creams as generally known in the art.

[0132] One or more agents described herein or pharmaceutically acceptable salts thereof can also be prepared in the form of suppositories (e.g., with conventional suppository bases such as cocoa butter and other glycerides) or retention enemas for rectal delivery.

[0133] In one embodiment, one or more agents described herein or pharmaceutically acceptable salts thereof are prepared with carriers that will protect them against rapid elimination from the body, such as a controlled release formulation, including implants and microencapsulated delivery systems. Biodegradable, biocompatible polymers can be used, such as ethylene vinyl acetate, polyanhydrides, polyglycolic acid, collagen, polyorthoesters, and polylactic acid. Methods for preparation of such formulations will be apparent to those skilled in the art. The materials can also be obtained commercially from Alza Corporation and Nova Pharmaceuticals, Inc. Liposomal suspensions (including liposomes targeted to infected cells with monoclonal antibodies to viral antigens) can also be used as pharmaceutically acceptable carriers. These can be prepared according to methods known to those skilled in the art, for example, as described in U.S. Patent No. 4,522,811.

[0134] It is especially advantageous to formulate oral, parenteral or CNS direct delivery compositions in dosage unit form for ease of administration and uniformity of dosage. Dosage unit form as used herein refers to physically discrete units suited as unitary dosages for the subject to be treated; each unit containing a predetermined quantity of active compound calculated to produce the desired therapeutic effect in association with the required pharmaceutical carrier. The specification for the dosage unit forms of the invention are dictated by and directly dependent on the unique characteristics of the active compound and the particular therapeutic effect to be achieved, and the limitations inherent in the art of compounding such an active compound for the treatment of individuals.

[0135] Toxicity and therapeutic efficacy of one or more agents described herein or pharmaceutically acceptable salts thereof can be determined by standard pharmaceutical procedures in cell cultures, experimental animals or in an individual, e.g., in a human. The dose ratio between toxic and therapeutic effects is the therapeutic index and it can be expressed as the ratio LD50/ED50. Compounds which exhibit large therapeutic indices are preferred. While compounds that exhibit toxic side effects may be used, care should be taken to design a delivery system that targets such compounds to the site of affected tissue in order to minimize potential damage to uninfected cells and, thereby, reduce side effects.

[0136] Data obtained from monitoring the immune state of an individual can be used in formulating a range of dosage for use in the individual (e.g., personalized medicine). The dosage typically will lie within a range of circulating concentrations that include the ED50 with little or no toxicity. The dosage may vary within this range depending upon the dosage form employed and the route of administration utilized. For any compound used in the method of the invention, the therapeutically effective dose can be estimated initially from cell culture assays. A dose may be formulated in animal models to achieve a circulating plasma concentration range that includes the IC50 (i.e., the concentration of the test compound which achieves a half-maximal inhibition of symptoms) as determined in cell culture. Such information can be used to more accurately determine useful doses in humans. Levels in plasma may be measured, for example, by high performance liquid chromatography.

[0137] Monitoring the influence of a pharmaceutical composition on the immune response of an individual can be applied not only in basic drug screening, but also in clinical trials. For example, the effectiveness of a pharmaceutical composition determined by a screening assay as described herein to alter the immune response of a subject can be monitored in clinical trials.

[0138] In certain exemplary embodiments, a method for monitoring the effectiveness of treatment of a subject with an agent described herein including the steps of (i) obtaining a pre-administration sample from a subject prior to administration of the agent; (ii) detecting the level of expression of one or more immune system nucleic acid sequences in the preadministration sample; (iii) obtaining one or more post-administration samples from the subject; (iv) detecting the level of one or more immune system nucleic acid sequences in the post-administration samples; (v) comparing the level of expression of one or more immune system nucleic acid sequences in the pre-administration sample with the level of expression of one or more immune system nucleic acid sequences in the post administration sample or samples; and (vi) altering the administration of the agent to the subject accordingly, is provided. According to such an embodiment, altered expression levels of one or more immune system nucleic acid sequences may be used as an indicator of the effectiveness of an agent, even in the absence of an observable phenotypic response.

[0139] In certain exemplary embodiments, a method for diagnosing, prognosing, monitoring, treating, ameliorating and/or preventing in a subject a disease, disorder, symptom and/or condition is provided. Subjects at risk for a disease and/or disorder described herein can be identified by, for example, by any or a combination of diagnostic or prognostic assays as described herein. Administration of a prophylactic agent can occur prior to the manifestation of symptoms characteristic of a disease or disorder, such that the disease or disorder is prevented or, alternatively, delayed in its progression. Depending on the type disease or disorder, one or more agents or pharmaceutically acceptable salts thereof can be used for treating the subject. The appropriate agent can be determined based on screening assays described herein.

[0140] In certain exemplary embodiments, methods of prognosing, diagnosing and/or monitoring one or more disorders or diseases associated with an infectious agent are provided. Infectious agents include, but are not limited to, viruses, bacteria, fungi, parasites, infectious proteins and the like.

[0141] Viruses include, but are not limited to, DNA or RNA animal viruses. As used herein, RNA viruses include, but are not limited to, virus families such as *Picornaviridae* (e.g., polioviruses), *Reoviridae* (e.g., rotaviruses), *Togaviridae* (e.g., encephalitis viruses, yellow fever virus, rubella virus), *Orthomyxoviridae* (e.g., influenza viruses), *Paramyxoviridae* (e.g., respiratory syncytial virus, measles virus, mumps virus, parainfluenza virus), *Rhabdoviridae* (e.g., rabies virus), *Coronaviridae*, *Bunyaviridae*, *Flaviviridae*, *Filoviridae*, *Arenaviridae*, *Bunyaviridae* and *Retroviridae* (e.g., human T cell lymphotropic viruses (HTLV), human immunodeficiency viruses (HIV)). As used herein, DNA viruses include, but are not limited to, virus families such as *Papovaviridae* (e.g., papilloma viruses), *Adenoviridae* (e.g., adenovirus), *Herpesviridae* (e.g., herpes simplex viruses), and *Poxviridae* (e.g., variola viruses).

[0142] Bacteria include, but are not limited to, gram positive bacteria, gram negative bacteria, acid-fast bacteria and the like.

[0143] As used herein, gram positive bacteria include, but are not limited to, *Actinomedurae*, *Actinomyces israelii*, *Bacillus anthracis*, *Bacillus cereus*, *Clostridium botulinum*, *Clostridium difficile*, *Clostridium perfringens*, *Clostridium tetani*, *Corynebacterium*, *Enterococcus faecalis*, *Listeria monocytogenes*, *Nocardia*, *Propionibacterium acnes*, *Staphylococcus aureus*, *Staphylococcus epiderm*, *Streptococcus mutans*, *Streptococcus pneumoniae* and the like.

[0144] As used herein, gram negative bacteria include, but are not limited to, *Afipia felis*, *Bacteriodes*, *Bartonella bacilliformis*, *Bortadella pertussis*, *Borrelia burgdorferi*, *Borrelia recurrentis*, *Brucella*, *Calymmatobacterium granulomatis*, *Campylobacter*, *Escherichia coli*, *Francisella tularensis*, *Gardnerella vaginalis*, *Haemophilus aegyptius*, *Haemophilus ducreyi*, *Haemophilus influenzae*, *Heliobacter pylori*, *Legionella pneumophila*, *Leptospira interrogans*, *Neisseria meningitidis*, *Porphyromonas gingivalis*, *Providencia stuartii*, *Pseudomonas aeruginosa*, *Salmonella enteridis*, *Salmonella typhi*, *Serratia marcescens*, *Shigella boydii*, *Streptobacillus moniliformis*, *Streptococcus*

pyogenes, *Treponema pallidum*, *Vibrio cholerae*, *Yersinia enterocolitica*, *Yersinia pestis* and the like.

[0145] As used herein, acid-fast bacteria include, but are not limited to, *Mycobacterium avium*, *Mycobacterium leprae*, *Mycobacterium tuberculosis* and the like.

[0146] As used herein, other bacteria not falling into the other three categories include, but are not limited to, *Bartonella henselae*, *Chlamydia psittaci*, *Chlamydia trachomatis*, *Coxiella burnetii*, *Mycoplasma pneumoniae*, *Rickettsia akari*, *Rickettsia prowazekii*, *Rickettsia rickettsii*, *Rickettsia tsutsugamushi*, *Rickettsia typhi*, *Ureaplasma urealyticum*, *Diplococcus pneumoniae*, *Ehrlichia chaffeensis*, *Enterococcus faecium*, *Meningococci* and the like.

[0147] As used herein, fungi include, but are not limited to, *Aspergilli*, *Candidae*, *Candida albicans*, *Coccidioides immitis*, *Cryptococci*, and combinations thereof.

[0148] As used herein, parasitic microbes include, but are not limited to, *Balantidium coli*, *Cryptosporidium parvum*, *Cyclospora cayatanensis*, *Encephalitozoa*, *Entamoeba histolytica*, *Enterocytozoon bieneusi*, *Giardia lamblia*, *Leishmaniae*, *Plasmodii*, *Toxoplasma gondii*, *Trypanosomae*, trapezoidal amoeba and the like.

[0149] As used herein, parasites include worms (e.g., helminthes), particularly parasitic worms including, but not limited to, Nematoda (roundworms, e.g., whipworms, hookworms, pinworms, ascarids, filarids and the like), Cestoda (e.g., tapeworms)

[0150] As used herein, infectious proteins include prions. Disorders caused by prions include, but are not limited to, human disorders such as Creutzfeldt-Jakob disease (CJD) (including, e.g., iatrogenic Creutzfeldt-Jakob disease (iCJD), variant Creutzfeldt-Jakob disease (vCJD), familial Creutzfeldt-Jakob disease (fCJD), and sporadic Creutzfeldt-Jakob disease (sCJD)), Gerstmann-Sträussler-Scheinker syndrome (GSS), fatal familial

insomnia (fFI), sporadic fatal insomnia (sFI), kuru, and the like, as well as disorders in animals such as scrapie (sheep and goats), bovine spongiform encephalopathy (BSE) (cattle), transmissible mink encephalopathy (TME) (mink), chronic wasting disease (CWD) (elk, mule deer), feline spongiform encephalopathy (cats), exotic ungulate encephalopathy (EUE) (nyala, oryx, greater kudu), spongiform encephalopathy of the ostrich and the like.

[0151] In certain exemplary embodiments, methods of prognosing, diagnosing and/or monitoring one or more cellular proliferative disorders are provided. Cellular proliferative disorders are intended to include disorders associated with rapid proliferation. As used herein, the term “cellular proliferative disorder” includes disorders characterized by undesirable or inappropriate proliferation of one or more subset(s) of cells in a multicellular organism. The term “cancer” refers to various types of malignant neoplasms, most of which can invade surrounding tissues, and may metastasize to different sites (see, for example, PDR Medical Dictionary 1st edition (1995)). The terms “neoplasm: and “tumor” refer to an abnormal tissue that grows by cellular proliferation more rapidly than normal. *Id.* Such abnormal tissue shows partial or complete lack of structural organization and functional coordination with the normal tissue which may be either benign (i.e., benign tumor) or malignant (i.e., malignant tumor).

[0152] The language “treatment of cellular proliferative disorders” is intended to include the prevention of the induction, onset, establishment or growth of neoplasms in a subject or a reduction in the growth of pre-existing neoplasms in a subject. The language also can describe inhibition of the invasion of neoplastic cells into neighboring tissues or the metastasis of a neoplasm from one site to another. Examples of the types of neoplasms intended to be encompassed by the present invention include but are not limited to those neoplasms associated with cancers of the breast, skin, bone, prostate, ovaries, uterus, cervix, liver, lung, brain, larynx, gallbladder, pancreas, rectum, parathyroid, thyroid,

adrenal gland, immune system, neural tissue, head and neck, colon, stomach, bronchi, and/or kidneys.

- [0153] Cellular proliferative disorders can further include disorders associated with hyperproliferation of vascular smooth muscle cells such as proliferative cardiovascular disorders, e.g., atherosclerosis and restenosis. Cellular proliferation disorders can also include disorders such as proliferative skin disorders, e.g., X-linked ichthyosis, psoriasis, atopic dermatitis, allergic contact dermatitis, epidermolytic hyperkeratosis, and seborrheic dermatitis. Cellular proliferative disorders can further include disorders such as autosomal dominant polycystic kidney disease (ADPKD), mastocytosis, and cellular proliferation disorders caused by infectious agents such as viruses.
- [0154] In certain exemplary embodiments, methods of prognosing, diagnosing and/or monitoring one or more autoimmune disorders are provided. As used herein, the term “autoimmune disorder” is a disease or disorder caused by a subject producing an inappropriate immune response against its own tissues. As used herein, an autoimmune disorder includes, but is not limited to, disorders such as Addison’s disease, alopecia areata, ankylosing spondylitis, antiphospholipid syndrome, autoimmune hemolytic anemia, autoimmune hepatitis, autoimmune inner ear disease (AIED), autoimmune lymphoproliferative syndrome (ALPS), autoimmune thrombocytopenic purpura (ATP), Balo disease, Bechet disease, bullous pemphigoid, cardiomyopathy, celiac sprue-dermatitis herpetiformis, chronic fatigue immune dysfunction syndrome (CFIDS), chronic inflammatory demyelinating polyneuropathy, cicatricial pemphigoid, cold agglutinin disease, CREST syndrome, Crohn’s disease, Degos disease, dermatomyositis, dermatomyositis-juvenile, discoid lupus, essential mixed cryoglobulinemia, fibromyalgia-fibromyositis, Graves disease, Guillain-Barré, Hashimoto thyroiditis, idiopathic pulmonary fibrosis, idiopathic thrombocytopenia purpura (ITP), IgA nephropathy, insulin-dependent diabetes, juvenile arthritis, lichen planus, lupus, Ménière disease, mixed connective tissue disease, multiple sclerosis, myasthenia gravis, pemphigus vulgaris, pernicious anemia, polyarteritis nodosa, polychondritis, polyglandular syndromes, polymyalgia rheumatica, polymyositis

and dermatomyositis, primary agammaglobulinemia, primary biliary cirrhosis, psoriasis, Raynaud phenomenon, Reiter syndrome, rheumatic fever, rheumatoid arthritis, sarcoidosis, scleroderma, Sjögren syndrome, stiff-person syndrome, Takayasu arthritis, temporal arteritis/giant cell arteritis, ulcerative colitis, vasculitis, vitiligo, Wegener granulomatosis and the like (See the American Autoimmune Related Diseases Association, Inc. website: aarda.org).

[0155] In certain exemplary embodiments, the nature and characteristics of symptoms, conditions, diseases and/or disorders are reduced by the methods of the present invention compared to the nature and characteristics of symptoms, conditions, diseases and/or disorders observed in a patient or a sample (e.g., a test sample or a sample taken from a subject prior to, during or after treatment). In certain aspects, the nature and characteristics of symptoms, conditions and diseases and/or disorder phenotypes are reduced by about 5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 55%, 60%, 65%, 70%, 75%, 80%, 85%, 90%, 91%, 92%, 93%, 94%, 95%, 96%, 97%, 98%, 99%, 99.9% or more, or any range(s) in between.

[0156] In certain exemplary embodiments, a method for modulating, ameliorating, preventing and/or treating diseases, symptoms and/or disorders as described herein includes the step of administering a therapeutically effective amount of an agent to a subject. As defined herein, a therapeutically effective amount of agent (i.e., an effective dosage) ranges from about 0.0001 to 30 mg/kg body weight, from about 0.001 to 25 mg/kg body weight, from about 0.01 to 20 mg/kg body weight, from about 0.1 to 15 mg/kg body weight, or from about 1 to 10 mg/kg, 2 to 9 mg/kg, 3 to 8 mg/kg, 4 to 7 mg/kg, or 5 to 6 mg/kg body weight. The skilled artisan will appreciate that certain factors may influence the dosage required to effectively treat a subject, including but not limited to the severity of the disease or disorder, previous treatments, the general health and/or age of the subject, and other diseases present. Moreover, treatment of a subject with a therapeutically effective amount of one or agents or pharmaceutically acceptable salts thereof can include a single treatment or, in certain exemplary embodiments, can include a series of treatments. It

will also be appreciated that the effective dosage of agent used for treatment may increase or decrease over the course of a particular treatment. Changes in dosage may result from the results of diagnostic assays as described herein. The pharmaceutical compositions can be included in a container, pack, or dispenser together with instructions for administration.

[0157] Embodiments of the present invention are still further directed to methods of pairing heavy and light chains by matching their relative clone frequencies or by matching their similar time series. Further embodiments include pairing heavy and light chains at the single cell level simultaneously in many cells in parallel in a single reaction medium. Such methods include high-throughput sequencing of components resulting from an immune reaction using methods for insulating cells including in-cell biochemistry on formaldehyde cross-linked/permeabilized cells, emulsion encapsulated cells, agarose-bead encapsulated cells, liposome transfection of single cells, small oligonucleotide-transfection of single cells including locked nucleic acids or fluorescent probes, and the like. Such methods reduce cross contamination and incorrect pairings. Methods of pairing heavy and light chains described herein utilize enzymatic methods including SOE-PCR which may result in incorporation of scFv linker sequences, direct ligation which may result in incorporation of scFv linker sequence and may also include bimolecular or trimolecular ligations, USER cloning, bead capture such as by either amplification onto beads or hybridization onto beads or capture by pre-annealed oligonucleotides with immune chain-specific complementary sequences.

[0158] Embodiments of the present invention are even still further directed to methods of obtaining full immune chain molecules by hybridization using CDR3 information. Such an embodiment includes the use of short-read sequencing technologies to select CDR3 sequences and using these CDR3 sequences to hybridize to a full immune chain molecule using complementary oligonucleotides.

[0159] Embodiments of the present invention still further include methods for reducing cross-contamination or incorrect pairing of heavy and light chains by bead-attached exonucleases and methods for monitoring cross-contamination or incorrect pairing of heavy and light chains by real-time PCR. The same or similar methods for pairing of heavy and light chains described herein can also be extended to the screening of antibody libraries against antigen libraries. According to one aspect, large and complex libraries of affinity reagents such as antibodies, nanobodies and the like are screened against large and complex libraries of antigen targets in a single reaction, followed by coupling DNA coding for the antibody sequence to the DNA coding for the antigen sequence.

[0160] Embodiments of the invention include the use of computer software to automate design and/or analysis of nucleic acid sequences. Such software may be used in conjunction with individuals performing polynucleotide synthesis by hand or in a semi-automated fashion or combined with an automated synthesis system. In at least some embodiments, the gene/oligonucleotide design/analysis software is implemented in a program written in the JAVA programming language. The program may be compiled into an executable that may then be run from a command prompt in the WINDOWS XP operating system. Exemplary software code is set forth as Appendix D. Unless specifically set forth in the claims, the invention is not limited to implementations using the same algorithms, organizational structure or other specific features of the exemplary software code. The invention is similarly not limited to implementation using a specific programming language, operating system environment or hardware platform.

[0161] It is to be understood that the embodiments of the present invention which have been described are merely illustrative of some of the applications of the principles of the present invention. Numerous modifications may be made by those skilled in the art based upon the teachings presented herein without departing from the scope of the invention.

[0162] The following examples are set forth as being representative of the present invention. These examples are not to be construed as limiting the scope of the invention as these and other equivalent embodiments will be apparent in view of the present disclosure, tables, figures, appendices and accompanying claims.

EXAMPLE I

Tracking Human Immune Response Dynamics Through VDJ Sequencing

[0163] Peripheral blood samples were drawn from a subject at -14 days prior to vaccination, approximately one hour after vaccination (day 0), and 1, 3, 7, 14, 21, and 28 days post-vaccination. For each sample, total RNA was extracted from peripheral blood leukocytes, from which cDNA and bar-coded 454 sequencing libraries were generated using primers specific for the entire V_H region (Figure 1B). B cell mRNA was analyzed to avoid cross-priming the germline DNA, to avoid non-functional receptor rearrangements, to benefit from possible clonal expansion of antigen-specific cells, and to take advantage of higher mRNA copy-numbers in antibody-expressing cells. The Roche 454 sequencing platform was used, as it provided read-lengths long enough to cover the entire V_H region.

[0164] Through the course of 7 runs of 454 GS FLX sequencing, approximately 5.5 million size-filtered reads were obtained that were subsequently aligned to the reference IMGT database (Worldwide Web Site: imgt.cines.fr/) (Lefranc et al. (2009) *Nucleic Acids Res.* 37:D1006). To do so, the V, then J, and finally D regions were identified, as their decreasing length allowed for the most reliable identification. The novel algorithm described herein selected the few best matches by comparing word frequencies and scores them using dynamic programming alignment (Durbin, R., Eddy, S.R., Krogh, A. & Mitchison, G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. (Cambridge Univ. Press, Cambridge; 1998)). To define clones, the algorithm then partitioned the reads according to their V-J usage and clusters their CDR3 junctions using agglomerative hierarchical clustering with edit distance as the metric (Gan, G., Ma, C. & Wu, J. *Data Clustering : Theory, Algorithms, and Applications*. (SIAM American Statistical Association, Philadelphia, PA, Alexandria, VA; 2007)).

While V and J region matches were generally unambiguous, D region identifications were not focused on as the algorithms often produced ambiguous alignments due to the short length of the D regions along with the highly mutated nature of the CDR3 (Figure 8). A total of 4,027,253 reads successfully passed this process (including bar-code identification), from which the immunoglobulin isotype was identified by searching for the corresponding primer sequences at the constant region end of the read (Figure 26).

[0165] Since the potential human repertoire diversity is estimated to be quite significant (up to 10^{14} by some estimates) (Davis et al. (1988) *Nature* 334:395), the technical feasibility and reproducibility of the methods described herein established as follows. A library of all time points was sequenced twice (sequencing replicates SR1 and SR2), and a completely independent third library prepared from the same RNA samples was sequenced once (technical replicate TR1). Among these three sequencing runs, 477,118 unique clones were identified where only 3% were shared between the three runs and 14% were observed in at least two runs (Figure 2A). However, those shared clones accounted for 59% and 71% of all reads respectively, indicating that the highly expressed clones are actually sampled significantly between replicate runs. This was further validated by a strong correlation between technical replicate samples, confirming technical reproducibility of the approach described herein (Figures 2B and 9). Furthermore, resampling of the data showed that 10^5 reads were sufficient to obtain high correlations between replicates (Figure 2C). By computing coefficients of variation (CV), it was determined that reliable quantification of clone frequencies as low as 10^{-4} could be obtained, though this number was dependent on a given sample's sequencing depth (Figure 10).

[0166] Methods developed to assess ecological population structures exist. (Magurran, A.E. *Measuring Biological Diversity*. (Blackwell Pub., Malden, Ma.; 2004)). Rank-abundance and clone-size distribution plots showed that the clonal structure follows a power law distribution with an exponent of 1.97 across >2 decades (Figures 2D and 11). Without intending to be bound by scientific theory, this could reflect the evolutionary selection

pressures experienced by the cells during the processes of differentiation and affinity maturation (Caldarelli et al. (2002) *Phys. Rev. Lett.* 89:258702; Bianconi and Barabasi (2001) *Europhysics Letters* 54:436). To estimate the total V_H repertoire diversity in a typical 1 mL blood sample, the Chao1 and ACE non-parametric species richness estimators were used (Magurran, *Supra*). A full sequencing run was also committed to a single time point (run DR1; day -14) to obtain a more reliable estimate. Inflection points and convexity were observed in almost all re-sampled estimator curves, indicating that the samples were sufficient to make reliable estimates (Figures 12A-12D). It was determined that a 1 mL blood sample contained approximately 100,000-500,000 unique heavy chain clones, depending on the blood sample and estimator used (Figure 2E). Finally, as only considering immunoglobulin heavy chains were considered, and without intending to be bound by scientific theory, it is expected the actual diversity will be at least an order of magnitude greater when paired with light chains. These data demonstrate that adequate sampling of the human repertoire diversity can be achieved in a single 1 mL blood draw, indicating the potential utility of such an approach as a personalized clinical diagnostic tool to identify current or previous pathogen exposure (Lerner et al. (1991) *Proc. Natl. Acad. Sci. USA* 88:9705). A large database of specificity-annotated antibody sequences will be generated.

[0167] Next the characteristics of the static immune repertoire were assessed. Overall, V and J usage was highly non-uniform (Figure 3A). The most frequently observed V segments were IGHV3-23 (11.4% of all reads), IGHV4-59 (8.5%), IGHV3-30-3 (7.3%) while the most frequent J segments were IGHJ4 (35.1%) and IGHJ6 (18.9%), consistent with previous studies (Boyd, *Supra*; Brezinschek et al. (1995) *J. Immunol.* 155:190; Glanville et al. (2009) *Proc. Natl. Acad. Sci. USA* (106:20216). The distribution of CDR3 junction lengths was then analyzed (Figure 3B). The two-sided, 95th percentile of the observed CDR3 lengths was 33 nucleotides (nt) to 75 nt, with median length of 54 nt (with longest observed junction at 146 nt), which shows similarity to the T cell receptor CDR3 size distribution (Freeman et al., *Supra*) and IMGT/LIGM data (Lefranc, *Supra*). In an

attempt to compare with traditional low-resolution methods for analyzing immune repertoires, pseudo-spectratype plots were also generated in which receptor-length diversity is used as a surrogate for overall diversity (Figure 3C and Figures 13A-13B) (Pannetier et al. (1993) *Proc. Natl. Acad. Sci. USA* 90:4319). Finally, the immunoglobulin isotype distribution was analyzed, and it was observed that IgM antibodies were the most abundant (37% of all reads), followed by IgA (36%), IgG (23%), IgD (4%), and IgE (0.01%). Though expressed mRNA levels in circulating B cells were measured, these numbers were still consistent with the subject's age and expected plasma protein titers (Lefranc, M.-P. & Lefranc, G.E. *The Immunoglobulin Factsbook*. (Academic Press, San Diego; 2001)). Altogether, these results show the first in-depth profile of the expressed immunoglobulin heavy chain repertoire in an individual human.

[0168] While the above analyses were focused on establishing reproducibility and profiling the static VDJ-ome, it was of great interest to survey the dynamic fluctuations of the immune repertoire. In the hope of capturing at least one immunological event, responses were induced to multiple immune challenges by vaccinating the subject against three pathogens for which his medical history suggests he should have immune memory: influenza (3 strains), hepatitis A, and hepatitis B. The clone frequencies were tracked through time to produce almost 500,000 time series over the eight time points, revealing the high complexity of the repertoire dynamics (Figure 4A). By randomly selecting clones across technical replicate runs, strong correlations were qualitatively observed at varying levels of expression, which further confirmed the reproducibility of the approach described herein (Figure 14). Initial clustering of the time series using a k-means algorithm showed that the most populated clusters were of ephemeral clones of various expression levels observed in each individual time point. However, after removing such single time point clones, the resulting clusters exhibited a vast array of dynamic behaviors (Figures 15A-15B).

[0169] Traditionally, attempts to characterize responses to immune challenges are performed by sorting for cells with certain surface markers, allowing the generation of antigen-specific antibodies by analyzing the heavy and light chains of single cells (Paul, *Supra*; Wrammert et al. (2008) *Nature* 453:667). In contrast to this approach, an attempt to identify antigen-specific lymphocytes without prior knowledge of specific antigens or cell-activation states and without using functional assays was performed. In an attempt to identify clones specific to the immune challenges, a self-organizing map (SOM) clustering method was used (Kohonen, T. *Self-Organizing Maps*, Edn. 2nd. (Springer, Berlin ; New York; 1997); Eichler et al. (2003) *Bioinformatics* 19:2321). This revealed groups of clones at days 7 and 21 that exhibited the typically expected activation patterns of antibody-secreting cells (ASC) and memory B cells (Figure 4B) (Abbas et al., *Supra*; Murphy et al., *Supra*; Wrammert et al., *Supra*). These clones also corresponded to the major peaks observed in the computed pseudo-spectratypes (identified by the numbers associated with each peak in Figure 3C). Surprisingly, the most dynamic and highly expressed clones (by orders of magnitude) were observed at day 0 (approximately 1 hr after vaccination) and day 1, with a significant drop at day 3 (Figure 4A). While the possibility that the subject was inadvertently experiencing an immune response unrelated to the vaccines cannot be excluded, it is believed that a vaccination-induced memory response was observed. While memory responses are known for rapidly generating high antibody titers, they are expected to peak 3-5 days post-exposure (Abbas et al., *Supra*; Murphy et al., *Supra*; Paul, *Supra*). While the highest exposure-induced serum titers may occur at days 3-5, the highest mRNA expression levels of the corresponding cells were observed at day 1, suggesting that functional studies relying on cDNA repertoire cloning of samples collected on days 3-5 would likely miss candidates of interest. However, it is important to note that \ mRNAs of circulating B cells were sampled, which precede the accumulation of actual antibody proteins. In contrast with the responses of days 7 and 21, the rapid onset and greater magnitude of the day 1 memory response indicates that these clones may have been overlooked in functional studies of therapeutic monoclonal

antibody discovery (Wrammert et al., *Supra*; Lanzavecchia et al. (2006) *Immunol. Rev.* **211**:303; Jin et al. (2009) *Nat. Med.* **15**:1088).

[0170] Next, global fluctuations in the isotype distribution over the course of the experiment were observed. Interestingly, a small rise of IgG was observed at day 7, while at day 21, a concurrent rise in IgA and drop in IgM was observed, indicating potential class switching in vaccine-induced clones (Figure 4C) (Cerutti (2008) *Nat. Rev. Immunol.* **8**:421). To obtain a more detailed view of isotype dynamics, the isotype distributions of the SOM-selected clones were analyzed (Figure 4D and Figure 16). The day 7 clones were primarily composed of the IgM and IgG isotype, while the day 21 clones were mostly composed of IgM and IgA. Most interestingly, the day 1 clones were dominated by IgA, reflecting an important role for this isotype in antiviral defense (Cerutti, *Supra*; Qiao et al. (2006) *Nat. Immunol.* **7**:302). Such an IgA memory response is characteristic of re-exposure to influenza antigens (Cox et al. (2006) *Vaccine* **24**:6577; Wright et al. (1983) *Infect. Immun.* **40**:1092). While IgA antibodies are largely responsible for defense in mucosal secretions (Cerutti, *Supra*; Stavnezer (1996) *Adv. Immunol.* **61**:79), it is reported that influenza vaccination (both nasal and intramuscular) elicits a rapid, strong, and short-lived systemic response of influenza-specific IgA ASC in peripheral blood (Cox et al., *Supra*). Nevertheless, the magnitude of the response observed in the hours following vaccination indicates a novel effector role for plasma IgA in antiviral memory, and highlights IgA as a potential non-traditional immunotherapy in viral diseases.

[0171] Figure 6 depicts FACS sorting analysis of the some of the selected sequences of Figure 4, where the hemagglutinin antigen (Brisbane 10 or Brisbane 59) was coated on beads and the antibody candidates were fluorescently labeled such that correct interaction between the antigen and antibody would result in a shift of the peaks during FACS analysis. Further quartz crystal microbalance (QCM) analysis in Figure 7 further demonstrates the binding affinity (with kD calculation) of the newly discovered antibody, demonstrating the potential of the methods and compositions described herein for efficiently identifying antibody candidates.

[0172] In accordance with aspect of the invention being a personalized diagnostic tool, the reads were aligned to the subset of antibody sequences in the IMGT/LIGM database for which there are known antigen specificities (Lefranc et al., *Supra*). The CDRs were attempted to be perfectly matched. As the specificity-annotated IMGT/LIGM data set was quite small (fewer than 9,000 sequences), it was not expected that close-to-perfect matches would be found. Surprisingly, however, a single clone of sequences successfully matched an anti-digoxin antibody with a single amino-acid mutation at the periphery of CDR1 (Figure 17). As the subject experienced two cardiac episodes in 1989 and 1994, it could not be ruled out that he was administered digoxin and developed sensitivity to it. As for the SOM-selected candidates of days 1, 7, and 21, multiple matches were found with various levels of sequence identity, some of which matched influenza-specific or hepatitis-specific antibodies.

[0173] The high-throughput sequencing analysis of the human B cell antibody heavy-chain repertoire in response to a vaccine in a time course experiment is described herein. It has been demonstrated herein that, without prior knowledge of antigens or cell activation states, one milliliter blood samples were sufficient to reproducibly track the immune system's dynamics, despite the large diversity and rapid fluctuations of its repertoire. Because of this impressive diversity and the influence of exposure history, the response made during human vaccination is an important component of personalized medicine and required characterization at the individual level. Thus, in the context of personalized medicine, the approach described herein has been successfully demonstrated for clinical diagnostics and therapeutic discovery.

References

[0174] Freeman et al. (2009) *Genome Res.* 19:1817

[0175] Weinstein et al. (2009) *Science* 324:807

- [0176] Boyd et al. (2009) *Sci. Transl. Med.* 1:12ra23
- [0177] Allison (2008) *Nat. Biotechnol.* 26:509
- [0178] Wrammert et al. (2008) *Nature* 453:667
- [0179] Draenert et al. (2006) *J. Exp. Med.* 203:529
- [0180] Pereyra et al. (2008) *J. Infect. Dis.* 197:563
- [0181] Weinstein et al. (2009) *Science* 324:807
- [0182] Freeman et al. (2009) *Genome Res.* 19:1817
- [0183] Glanville et al. (2009) *Proc. Natl. Acad. Sci. USA* 106:20216
- [0184] Kenneth et al. (2008) Janeway's Immunobiology. Garland Science, New York, 7th ed.
- [0185] Lefranc and Lefranc (2001) The Immunoglobulin Factsbook. Academic Press, San Diego
- [0186] Moore and Clayton (2003) *Nature* 426 :725
- [0187] Lerner et al. (1001) *Proc. Natl. Acad. Sci. USA* 88:9705
- [0188] Brezinschek et al. (1998) *J. Immunol.* 160:4762
- [0189] Brezinschek et al. (1995) *J. Immunol.* 155:190
- [0190] Pannetier et al. (1995) *Immunol. Today* 16:176
- [0191] Pannetier et al. (1993) *Proc. Natl. Acad. Sci. USA* (1993) 90:4319

[0192] Marasco and Sui (2007) *Nat. Biotechnol.* 25:1421

EXAMPLE II

Repertoire Cloning of Paired Antibody Heavy and Light Chains

[0193] To rigorously confirm that certain clones are involved in immune responses or to discover new antibodies against antigens of interest, it is desirable to successfully capture both the heavy and light chains of individual antibodies. Because of the lack of methods for capturing paired VH and VL chains in high-throughput, protocols known in the art involve sorting single cells into individual wells and performing PCR for the heavy and light chains serially (e.g., Meijer et al. (2006) *J. Mol. Biol.* 358:764). However, even with automated liquid handling robots, typical throughputs are practically limited to 10^6 (at great expense). Another popular solution is to capture heavy and light chain repertoires separately, and associate them randomly with each other in expression vectors (Wright et al. (2009) *Proc. Natl. Acad. Sci. USA* 106:19078). However, heavy and light chain pairing is likely far from randomly distributed. Accordingly, these methods do not provide accurate portrayals of the underlying repertoires.

[0194] This example describes a general method for capturing paired heavy and light chains in millions of single cells in a single-reaction format. Multiple methods are provided that allow the simultaneous manipulation of millions of cells in parallel, while keeping them isolated from each other to maintain the natural chain pairing.

Methods for single-cell coupling of chains

[0195] Multiple methods are provided for coupling the heavy and light chains from single cells. The overall methods are split into two parts: the chain-linking biochemistry and the cell insulation method. Solutions for each part are primarily chosen independently, and multiple combinations are explored.

[0196] Certain of the biochemical methods are PCR based. A choice between amplifying the target chains from the genomic DNA or performing RT-PCR is made based on the nature of the expressed mRNAs. The former requires no reverse transcription step but has the risk of amplifying non-functional receptors, while the latter can benefit from higher copy numbers and should only capture functional, expressed receptors. The physical cross-linking optionally occurs through multiple mechanisms. The first is standard splicing-by-overlap-extension PCR (SOE-PCR) (or fusion PCR or crossover PCR), whereby two of the PCR primers have complementary sequences so that the two amplicons function as primers and they fuse to each other (Heckman and Pease (2007) *Nat. Protocol.* 2:924). One advantage to this method is that the overlap sequence can be designed so that the fused construct is immediately in a usable scFv format. The next mechanism is similar to the SOE-PCR in that tags are incorporated into the PCR primers. In this case, the tags contain loxP sites, so that fusion will occur upon Cre-mediated recombination (Albert et al. (1995) *Plant J.* 7:649; Chapal et al. (1997) *Biotechniques* 23:518). Finally, in the case of emulsion methods, another biochemical option is to amplify both the heavy and light chains onto beads (Diehl et al. (2006) *Nat. Methods* 3:551; Shendure et al. (2005) *Science* 309:1728). One advantage is that the beads can be processed immediately for sequencing on bead-based next-generation sequencing systems. However, this also can be a disadvantage, as it can limit the range of options after chain coupling. An alternative bead-based method is to amplify both chains onto beads, and then couple the chains on the beads. This increases the specificity of the entire process, albeit at increased complexity of the protocol. Each of these methods are summarized in Figure 27.

[0197] The cell insulation methods fall into two main categories: in-cell methods and emulsion methods. For in-cell methods (Embleton et al. (1992) *Nucleic Acids Res.* 20:3831), the cells are fixed in formalin and permeabilized to allow the diffusion of biochemical reagents into the cell. The cell membrane functions as the barrier that prevents cross-contamination of heavy and light chains between cells. The advantages of this general approach are the relative simplicity of fixing the cells and also the ability to serially apply

reagent sets to all cells in parallel. However, without intending to be bound by scientific theory, the permeabilization may potentially increase the chance that Ig chains will leak out of cells and lead to cross-contamination.

[0198] For the emulsion-based methods, single cells are placed into individual compartments of a water-in-oil emulsion (Clausell-Tormos et al. (2008) *Chem. Biol.* 15:427; Leamon et al. (2006) *Nat. Methods* 3:541). An advantage of such an approach is that the oil-based separation of compartments can potentially provide nearly absolute insulation from chain cross-contamination. But while the oil-separated compartments can stop any exchange of material between compartments, a common problem of thermal cycling emulsions is that compartments fuse together, leading to non-clonality. Furthermore, it is considerably more difficult to manipulate emulsions. Emulsions are generally formed using physical methods (e.g., vortexing) that depend on Poisson statistics to achieve clonality (Nakano et al. (2005) *J. Biosci. Bioeng.* 99:293; Williams et al. (2006) *Nat. Methods* 3:545). However, this tends to lead to a small fraction of non-clonal compartments, and also leads to a large number of unoccupied compartments. To combat these problems, emulsions are generated using microfluidic technology (Clausell-Tormos et al., *supra*). A disadvantage to using emulsion methods is that once an emulsion is formed, it is difficult to exchange additional material with the compartments in a controlled fashion. However, a technology for fusing emulsion droplets in a controlled fashion (Tewhey et al. (2009) *Nat. Biotechnol.* 27:1025; Meyers and Gelfand (1991) *Biochemistry* 30:7661) is used to address this disadvantage. Finally, emulsion PCR is optionally performed using conditions that are far from standardized protocols. The cell insulation methods are summarized in Figure 28.

[0199] With these general considerations, at least six strategies have been identified to achieve robust coupling of heavy and light chains. These strategies are depicted in Figure 29 and include, but are not limited to: 1) emulsion PCR from gDNA; 2) in-cell RT and SOE-PCR or Cre-Lox coupling; 3) Tth-mediated emulsion RT-PCR; 4) acrylamide-

encapsulated in-cell RT-PCR; 5) emulsion PCR onto beads; and 6) emulsion RT-PCR with droplet fusion.

Emulsion PCR from gDNA

[0200] This is the simplest emulsion-based approach. The cells are placed in the emulsion along with reagents for a traditional PCR reaction. SOE-PCR is then performed using the gDNA as a template.

In-cell RT and SOE-PCR or Cre-Lox coupling

[0201] This involves fixing cells in formalin and permeabilizing them using one of several methods (e.g., proteinase K) (Chapal et al., *supra*; Embleton et al., *supra*). Because all cells are in solution, a traditional RT-PCR reaction is performed by applying the relevant enzymes serially.

Tth-mediated emulsion RT-PCR

[0202] It is desirable to capture the Ig chains from the mRNA sequence, as this avoids any non-functional receptor rearrangements and also benefits from the potentially higher copy-numbers of expressed cells. However, emulsion PCR only allows the addition of biochemical reagents one time. Therefore, Tth polymerase, which is capable of performing both RT and PCR (Myers and Gelfand, *supra*), is used. The performance of the polymerase is characterized and it is used in the context of emulsions.

Acrylamide-encapsulated in-cell RT-PCR

[0203] This technique is similar to the in-cell RT-PCR, but involves the additional step of encapsulating individual cells in polyacrylamide gels (Yokoyama et al. (1990) *Jinrui*

Idengaku Zasshi 35:131), which adds an additional layer of protection from cross-contamination.

Emulsion PCR onto beads

[0204] As an alternative to SOE-PCR for cross-linking the heavy and light chains, the two chains are captured by conventional PCR onto beads (Diehl et al., *supra*). Each emulsion compartment aims to have a single cell and a single bead. The beads are coated with two different primers: one for the heavy chain and one for the light chain. After breaking the emulsion, the beads are optionally manipulated in a variety of ways. One method, which can increase the specificity of the technique, is to cross-link the two chains on the beads using a modified Cre-Lox system.

Emulsion RT-PCR with droplet fusion

[0205] As an alternative to using Tth polymerase, emulsion droplets are fused in a controlled manner. In this way, “bags” of enzymes can be serially fused with the emulsion compartments to perform separate biochemical steps, such as RT followed by PCR. This approach allows the use of robust and well characterized enzymes to separately perform RT and PCR.

Design of primers and vectors for multiple chain-coupling

[0206] Primers are designed for the various RT and PCR protocols described herein. Primers include, but are not limited to: RT primers; multiplex PCR primers for the V and J segments for cDNA (the V primers are specific for cDNA as they will span the L1-L2 exon boundary for the leader sequence); multiplex PCR primers for the V and J segments for gDNA (these are used during emulsion PCR); PCR primers that are immobilized on

beads; PCR primers that contain complementary tags for overlap PCR (it is important that the overlap is suitable as a linker sequence for an scFv chain and will place the heavy and light chain in-frame); PCR primers that contain LoxP sites for cross linking heavy and light chains (Chapal et al., *supra*); and primers that incorporate common sequences into primers for easily adapting sequences to next-generation sequencing and for easily cloning sequences into various expression vectors (i.e., for various protein display technologies). Furthermore, vectors are designed that are suitable for a variety of display technologies, using standard vectors through the use of Gateway cloning protocols.

Optimization of PCR conditions for cross-linking PCR

[0207] PCR reaction conditions are optimized in several stages. First gDNA or mRNA prepared from clonal populations of cells is used to ensure that the heavy and light chains are cross-linked in an idealized environment. After robust PCR conditions are achieved, Ig chains (from whole cells) are cross-linked in solutions using either SOE-PCR or Cre-Lox recombination. Microfluidic technology is used to place single cells into emulsion compartments 100 μm in diameter (approximately 0.5 nL) (Clausell-Tormos et al., *supra*). In certain embodiments, the compartment size is reduced to ensure that the emulsion does not break during thermal cycling. A cell concentration equivalent to 20 μm emulsion compartments, which would correspond to about 1.5×10^6 cells in a typical 50 μL reaction, can be used.

Optimization of cell insulation protocol

[0208] Heavy and light chains are cross-linked in single cells. Using a single monoclonal cell line, multiple methods for insulating single cells and performing PCR are performed. After cell fixation using formalin, permeabilization of the cell is performed (detergents such as Nonidet P40 (Embleton et al., *supra*) may be too aggressive for such use).

Proteinase K digestions can be used to permeabilize cells by chewing away membrane proteins and pores (Bagasra (2007) *Nat. Protocol.* 2:2782). Overlapping tags are used to cross-link the two amplicons during PCR. An alternative method involves incorporating LoxP sites into the two internal primers and to cross-linking the two chains via Cre recombination (Chapal et al., *supra*).

Characterization of cross-contamination in chain cross-linking

[0209] Following successful cross-linking of heavy and light chains in single cells, the level of cross-contamination is measured. The Embleton et al. study found that when mixing two known hybridoma cell lines at 1:1 and 9:1 ratios, they observed no mispaired amplicons (Embleton et al., *supra*). However, these ratios are far too lenient compared with real effective ratios from complex mixtures. Accordingly, the level of mispairing using two cell lines in ratios up to 10^6 :1 are characterized. Furthermore, the method used in Embleton et al. relied on capturing clones from the “rare” cell line using traditional screening methods. These methods are not practical when attempted on more stringent cell ratios, as the rare cell type may never be observed. To address this, PCR-based methods are provided that are far more sensitive than the method used in Embleton et al.

PCR-based cross-contamination assay

[0210] A TaqMan assay is used in order to quantify the level of cross-contamination. Without intending to be bound by scientific theory, after performing the single-cell cross-linking PCR, four species are expected in the combined mixture: A-a, A-b, B-a, B-b (where A and B are the two clones and A versus a is heavy versus light chain). As the scFv linker sequence connecting the heavy and light chains is the same for all species, one TaqMan probe with fluorophore X is used to hybridize to this common portion. Separately, four constructs of the same length that are amplified by only one of the four primer combinations, which all contain a common unique sequence that hybridizes to a second TaqMan probe with fluorophore Y are used. The synthetic constructs are set to known

concentrations. A real time PCR reaction is used in which the experimental mixture of four constructs is split into four separate tubes. Each tube is then amplified with one of the four primer combinations along with the corresponding synthetic template at a known concentration. Each tube is also amplified in the presence of both TaqMan probes. Fluorescence is measured at both wavelengths at each cycle. Ultimately, the fluorescence measurements of Y are used as a standard to compare the relative concentrations of the various species in the starting mixture.

Quantification of cross-contamination

- [0211] Following PCR-based mispairing, the levels of cross-contamination at different ratios of the two cell lines are measured. For in-cell PCR methods, one source of cross-contamination is mRNA/cDNA that has leaked into the supernatant solution. In order to combat this problem, different permeabilization parameters, e.g., such as modifying proteinase K digestion, times are assayed.
- [0212] For emulsion-based methods, one source of cross-contamination is the existence of non-clonal compartments. This can occur during the formation of the emulsion when multiple cells are placed into a single compartment, as well as during thermal cycling of the emulsion, where different emulsion compartments can fuse together. In order to avoid non-clonality at the emulsion formation step, microfluidic platforms are provided to determine emulsion formation methods that are easier to control. A variety of different emulsion oils and/or surfactants and/or reduced size emulsion compartments are provided to avoid droplet fusion during thermal cycling.
- [0213] The level of cross-contamination is obtained using mixtures of two known cell lines (A and B) at A:B ratios ranging from 1:1 through $10^6:1$. As the concentration of A increases relative to B, the likelihood that a B chain will mispair with an A chain will increase, leading to a higher relative amount of mispaired B chains. Given a complex mixture of

lymphocytes obtained from blood, the lowest frequency level for which heavy-light chain pairing is accurately ascertained is determined.

Creation of scFv libraries from complex samples

[0214] Complex blood-derived samples of lymphocytes are used to generate new 454 sequencing libraries. Certain of the vaccination samples collected are sequenced to confirm reproducibility.

Analysis of properties of chain pairing including total diversity estimates

[0215] The newly generated full scFv data is run through an informatics pipeline to determine VDJ usage and split the data into unique clones. This data provides the most accurate estimates to date of expressed antibody diversities. Furthermore, many studies assume that heavy and light chain pairing is independent and random. However, without intending to be bound by scientific theory, it is believed that generating actual paired heavy and light chains will show that the distribution is far from random, and provides novel methods to characterize it.

Characterizing multiple solutions to influenza H1N1 infection

[0216] A new protocol to collect blood samples from volunteers who will be immunized to the H1N1 influenza strain ("swine flu") is provided. The protocol selects volunteers who have not been exposed to swine flu and also volunteers that contracted the disease and fought it successfully. The immune responses of people who have been exposed to the virus can thus be compared with the immune of to those who have not been exposed to it (though they likely contracted some other strain of influenza at sometime in their history).

[0217] Furthermore, each sample obtained from a subject that successfully fought H1N1 infection contains a unique immunological solution to combating the virus.

Commercially available protein preparations of glycosylated H1N1 hemagglutinin and neuraminidase are obtained, and the expressed scFv libraries from each individual is cloned using an *in vitro* display system (e.g., phage display). Anti-flu antibodies are functionally enriched using standard panning techniques, and the resulting enriched scFv pools are sequenced using a next-generation sequencing technology. This allows the direct observation of how different individuals fight H1N1, and also confirms the results obtained by sequencing the repertoire without panning against antigen.

[0218] In addition, samples are saved in order to express the putative responding antibodies and characterize their interaction against larger antigen libraries.

EXAMPLE III

High-Throughput Characterization and Selection of Antibody Repertoires Against Pathogen Libraries

[0219] A similar approach to that described above is used to characterize antigen-antibody interactions in high-throughput. Current technologies require selecting for new antibodies against a single antigen at a time (Carter (2006) *Nat. Rev. Immunol.* 6:343). A typical experiment involves purifying and immobilizing some antigen of interest, and exposing it to some type of protein display technology encoding a library of candidate antibody sequences (typically in scFv format). After multiple rounds of panning, washing and amplifying, a small number of clones is sequenced and carried through for further analysis. Alternatively, an animal is immunized with the antigen of interest to generate a polyclonal response. Antigen-specific lymphocytes are then harvested (to obtain the polyclonal response) and screened to obtain high-affinity monoclonal antibodies. *Id.* Methods are provided that allow the selection of new antibodies against multiple antigens in parallel.

[0220] Two chains of nucleic acids are coupled together. Antibodies can be encoded in a protein display format (e.g., phage (Clackson et al. (1991) *Nature* 352:624; McCafferty et al. (1990) *Nature* 348:552), yeast (Boder and Wittrup (1997) *Nat. Biotechnol.* 15:553; Boder et al. (2000) *Proc. Natl. Acad. Sci. USA* 97:10701; Chao et al. (2006) *Nat. Protoc.* 1:755), or ribosome display (Zahnd et al. (2007) *Nat. Methods* 4:269)) and antigens are packaged with their coding information (e.g., peptide libraries in display format, or whole virus particles). The two libraries are allowed to interact and interacting complexes are selected (Bowley et al. (2009) *Proc. Natl. Acad. Sci. USA* 106:1380). The coding chains of the interacting antigen and antibody are then physically coupled using methods similar to those described herein. This will allow the characterization of antigen-specific antibodies for multiple antigens in parallel. Ultimately, this type of approach will be scaled up to assay very large antigen libraries against very large antigen libraries (e.g., entire human proteome, all known viral proteins and the like).

[0221] One unique aspect of the methods described herein is that protocols for capturing full antibodies will have already formatted the captured repertoires in an scFv format, allowing for easy expression using one of the protein display technologies. Compared with many previous studies that generate random antibody libraries, each antibody in the libraries described herein are generated by a functioning, natural immune system. In this way, millions of years of evolution can be used to provide efficient antibody libraries that will allow one of skill in the art the ability to quickly discover new functional antibodies.

Methods for multiplex affinity selections

[0222] One approach for capturing antigen-antibody (Ag-Ab) interactions involves placing single Ag-Ab complexes into individual emulsion compartments. In order to properly display antibodies and antigens, several choices are suitable including, but not limited to: phage, yeast, or ribosome display, whole viral particles (in the case of viral antigens). Ribosome display offers the largest potential libraries and is fully *in vitro*. Phage display is the oldest method and provides large library sizes. Yeast display offers the smallest library

sizes, but is particularly appropriate for antigen libraries as it can carry larger payloads and has glycosylation machinery (Hoogenboom (2005) *Nat. Biotechnol.* 23:1105). Use of whole virus particles provides access to the most realistic antigens, but has a potentially large genome and additional safety concerns for lab work. It is important to note that because naturally expressed repertoires are cloned, the maximum library size of all of these technologies is expected to be sufficient to capture the diversity of sequences in any practical blood samples or antigen sets. Phage-Ab against yeast-Ag is a promising general approach for library-against-library selections (Bowley et al., *supra*). Yeast is well suited to larger protein fragments, the two systems can replicate independently. Additionally, ribosome-Ab against ribosome-Ag is also utilized as this system is entirely *in vitro*. This will enable coupling methods that do not depend on emulsions. Finally, these methods are also used against whole viral particles.

[0223] For the phage-based systems, an emulsion-based method is used to capture interacting antibodies and antigens. After several or more rounds of affinity selection, phage-Ab-yeast-Ag duplexes are double-purified using magnetic sorting or FACS to eliminate non-interacting particles. *Id.* These duplexes are then placed into individual emulsion compartments where a cross-linking PCR reaction will physically associate the Ag and Ab coding sequence. These cross-linked species are then prepared for next-generation sequencing and interactions are detected by analyzing the sequencing data. When the Ag used is whole virus particles, the genome sequence can be relatively large. However, as a relatively small number of unique viruses is assayed, the method is sufficient to find a unique barcode to identify the specific strain of virus.

[0224] For the fully *in vitro* ribosome display system, in addition to attempting the emulsion based protocol, non-emulsion methods are used. A proximity ligation assay (Soderberg et al. (2006) *Nat. Methods* 3:995) is used to capture unique tags on both Ag and Ab libraries. These tags will supply enough information to obtain the full corresponding sequences.

Assaying a positive control for Ab-Ag interaction

[0225] A single Ag-Ab pair that is known to interact with high affinity and specificity is cross-linked. An scFv fragment is created in a phage display format from a sequence known to bind to influenza neuraminidase, along with the corresponding antigen fragment in a yeast display format. Preparations of the two species are allowed to interact in solution, and methods described herein are used to cross-link the two coding chains.

[0226] In a first method, only the flu antibody and flu antigen are allowed to interact in solution, and PCR is used to cross-link the two coding strands. The primers are assayed for cross-linking via overlap PCR and/or Cre/lox recombination. Likewise, the same positive control Ab and Ag pair are placed into a commercial ribosome display system the pair is allowed to interact. Reverse transcription and cross-linking PCR are performed using the methods described herein to assay for successful fusion of the Ab and Ag coding strands.

Quantification of double-purification enrichment for interacting Ab-Ag pairs

[0227] In addition to the positive control Ab-Ag pair, a second, orthogonal Ab-Ag pair, along with a phage-Ab that is not specific for any antigen used and a yeast-Ag that is not specific for any of the antibodies are chosen. After allowing the mixture to interact, cognate Ab-Ag complexes are purified by flow cytometry of yeast cells using fluorescent anti-phage antibody, as described in (Bowley et al., *supra*). From the sorted cells, cross-linking and PCR are performed and the amount of contaminating DNA from the phage and yeast that are expected to not show binding affinity is measured. In order to reduce the amount of background, double-purification is performed followed by amplification of the phage and yeast libraries separately. *Id.* For the ribosome display system, two different affinity tags are encoded into the Ag library and the Ab library (e.g., His and flag tags). His-tag purification is then performed followed by flag-tag purification to obtain interacting particles.

Quantification of insulation of Ab-Ag complexes in emulsions

[0228] After performing selection for interacting particles, the Ab-Ag complexes are placed into emulsions using methods described herein. In parallel, more extreme ratios of the two orthogonal positive control pairs are progressively assayed to determine the amount of cross contamination during the coding strand fusion. A summary is set forth at Figure 30.

Initial controls for ribosome display proximity ligation

[0229] One advantage of the ribosome display methods described herein is that the entire system is *in vitro* and the coding sequences are exposed to solution. Using only one of the positive control Ab-Ag pairs, rolling circle-based proximity ligation system (Soderberg et al., *supra*) is assayed. The two libraries contain unique sequence(s) that function as barcode(s). Additional probes are added to the reaction which form closed circles when the two strands of the Ab and Ag mRNAs are in close proximity to each other.

[0230] The same assay is then performed with two positive controls following double purification for interacting complexes. This will elucidate whether the proximity ligation assay is specific only for Ab-Ag pairs that are actually interacting, or whether the circles can be closed promiscuously.

Building Ab-Ag lookup table using viral display libraries

[0231] After successfully linking the coding chains of interacting Ab-Ag pairs of a small number of controls, the method is applied to complex libraries. Phage display and ribosome display libraries of naturally expressed scFvs that were cloned as described herein are made. For the vaccination samples, a small library of viral proteins that correspond to the same strains to which the subjects were exposed is generated. These libraries are then used to characterize which antibodies bind to which proteins. Results obtained are

correlated with the antibody clones that were identified to react to the administered vaccines in the patients.

[0232] In addition to generating viral protein libraries artificially, a similar method is used with full viral particles. Viruses are a convenient system for this strategy, as they function as their own protein display particles. Furthermore, since only a small number of viral strains is needed, it is enough to identify which strain has been captured by cross-linking with one or more unique genomic barcode sequences.

EXAMPLE IV

Materials and Methods

Biological samples and immunization

[0233] Peripheral blood samples were drawn into 9 mL K3-EDTA tubes from a single individual 14 days prior and 0, 1, 3, 7, 14, 21, and 28 days following vaccination. Vaccines administered were as follows: Hepatitis A+B (Twinrix) and seasonal influenza vaccine, including strains A/Brisbane/59/2007 (H1N1-like), A/Brisbane/10/2007 (H3N2-like), and B/Florida/4/2006.

RNA extraction

[0234] Blood samples were immediately processed through a filtering unit to isolate the leukocyte fraction (LeukoLOCKTM, Ambion, Austin, TX), then stored at -80°C until all samples were obtained. Total RNA was later extracted according to the manufacturer's protocol. RNA integrity and concentration was assessed using a NanoDrop spectrophotometer (Thermo Scientific, Wilmington, DE) and a 2100 Bionanalyzer (Agilent, Foster City, CA).

cDNA synthesis and PCR

[0235] For each sample, 2 x 1 µg of total RNA was reverse transcribed in 2 independent 20 µL reactions using SuperScriptTM III reverse transcriptase (Invitrogen, Carlsbad, CA) at 55°C for 60 min in the presence of 5 pmole of C-region gene-specific primer mix (IGHC-RT) (Table 1), followed by enzyme inactivation and RNase H digestion (Epicentre Biotechnologies, Madison, WI) according to Invitrogen protocols. Each 20 µL cDNA reaction was then split in 4 x 50 µL PCR reactions each, for a total of 8 independent PCR reactions per originating sample. PCR was performed in the presence of 200 nM dNTPs (Enzymatics, Beverly, MA), 25 pmole of V-region gene-specific primer mix (IGHV-PCR), 25 pmole of C-region gene-specific primer mix (IGHC-PCR), and 1 U of PhusionTM Hot Start High-Fidelity DNA Polymerase (New England Biolabs, Ipswich, MA) and thermal cycled as follows: 98°C for 1 min, 16 cycles of 98°C for 10 sec, 62°C for 20 sec, 72°C for 20 sec, followed by 72°C for 5 min and then cooled down to 4°C. Following PCR, all reactions originating from the same sample were pooled into one 400 µL mixture to which 160 U of Exonuclease I (Epicentre Biotechnologies, Madison, WI) was added and incubated at 37°C for 45 min. Samples were then purified using the QIAquick PCR Purification Kit (Qiagen, Valencia, CA) and eluted in 36 µL of EB buffer according to the manufacturer's protocol.

Reverse transcription oligonucleotides

C-region gene specific primers mix (IGHC-RT)

VDJ-20080707-IGHG	AGGGYGCCAGGGGGAAGA (SEQ ID NO:133)
VDJ-20080707-IGHM	GGAGACGAGGGGGAAAAGG (SEQ ID NO:134)
VDJ-20080707-IGHA	CAGCGGGAAGACCTTGGG (SEQ ID NO:135)
VDJ-20080707-IGHD	CACATCCGGAGCCTTGGT (SEQ ID NO:136)
VDJ-20080707-IGHE	TCAAGGGGAAGACGGATGG (SEQ ID NO:137)

PCR oligonucleotides

C-region gene specific primers (IGHC-PCR)

VDJ-20080924-IGHG-1	CCGATGGGCCCTTGGTGG (SEQ ID NO:138)
VDJ-20080924-IGHG-2	CGGATGGGCCCTTGGTGG (SEQ ID NO:139)
VDJ-20080924-IGHM	GGGTTGGGGCGGATGCAC (SEQ ID NO:140)
VDJ-20080924-IGHA	CCTTGGGGCTGGTCGGGG (SEQ ID NO:141)
VDJ-20080924-IGHD	CATCCGGAGCCTTGGTGG (SEQ ID NO:142)
VDJ-20080924-IGHE	CGGATGGGCTCTGTGTGG (SEQ ID NO:143)

V-region gene specific primers (IGHV-PCR)

VDJ-20080924-IGHV1-1	GAGCAGCGACAGGTGCCC (SEQ ID NO:144)
VDJ-20080924-IGHV1-2	CAGCAGCCACAGGTGCCC (SEQ ID NO:145)

VDJ-20080924-IGHV1-3	GCAGCAGCTACAGGTGTCC (SEQ ID NO:146)
VDJ-20080924-IGHV1-4	CTGTAGCACCAGGTGCCC (SEQ ID NO:147)
VDJ-20080924-IGHV1-5	GCTGTAGCTCCAGGTGCTC (SEQ ID NO:148)
VDJ-20080924-IGHV1-6	CAGCACCAACAGGTGCCC (SEQ ID NO:149)
VDJ-20080924-IGHV1-7	CAGCAGCCACAGNTGCCT (SEQ ID NO:150)
VDJ-20080924-IGHV1-8	CAGCAGCTACAAGTGCCC (SEQ ID NO:151)
VDJ-20080924-IGHV1-9	CAGCAGCCACAGGAGCCC (SEQ ID NO:152)
VDJ-20080924-IGHV1-10	CAGCAGCCACAGGTGTCC (SEQ ID NO:153)
VDJ-20080924-IGHV1-11	CAGCAGCTACAGGCACCC (SEQ ID NO:154)
VDJ-20080924-IGHV1-12	CCTGTTTTTGGTGCCC (SEQ ID NO:155)
VDJ-20080924-IGHV1-13	TGGCAGCACCAGGCGCCC (SEQ ID NO:156)
VDJ-20080924-IGHV1-14	TCATAGCTGCAGGTGCCC (SEQ ID NO:157)
VDJ-20080924-IGHV2-1	CTGTCCCGTCCTGGGTCT (SEQ ID NO:158)
VDJ-20080924-IGHV2-2	ACCATCCCTTCATGGGTCT (SEQ ID NO:159)
VDJ-20080924-IGHV2-3	CCACCCCTTCCTGGGTCT (SEQ ID NO:160)
VDJ-20080924-IGHV3-1	TTCTGTGCTATATTAAAGCTGTCC (SEQ ID NO:161)
VDJ-20080924-IGHV3-2	TTGTTGCTATTTTAAAAGGTGTCC (SEQ ID NO:162)
VDJ-20080924-IGHV3-3	CGTTGCTCTTTTAAAGAGGTGTCC (SEQ ID NO:163)
VDJ-20080924-IGHV3-4	TTGTTGCTATTTTAAAGGTGTCC (SEQ ID NO:164)
VDJ-20080924-IGHV3-5	TTGTTGCTATATTAGAAGGTGTCC (SEQ ID NO:165)
VDJ-20080924-IGHV3-6	GCTATTTTAAAAGGTGTCC (SEQ ID NO:166)
VDJ-20080924-IGHV3-7	TTGTTGCTATTTTAGAAGGTGTCC (SEQ ID NO:167)
VDJ-20080924-IGHV3-8	TTGTGGCTATTTTAAAAGGTGTCC (SEQ ID NO:168)
VDJ-20080924-IGHV3-9	TTGTTGTTATTTTACAAGGTGTCC (SEQ ID NO:169)
VDJ-20080924-IGHV3-10	TTCTGCTATTTTAAAAGGTGTCC (SEQ ID NO:170)
VDJ-20080924-IGHV3-11	TTGCTGCTATTTTAAAAGGTGTCC (SEQ ID NO:171)
VDJ-20080924-IGHV3-12	TTTTGGCTATTTTAAAAGGTGTCC (SEQ ID NO:172)
VDJ-20080924-IGHV3-13	TTGTGGCTAAAATAAAAGGTGTCC (SEQ ID NO:173)
VDJ-20080924-IGHV3-14	TTGTTGCTATAATAAAAGGTGTCC (SEQ ID NO:174)
VDJ-20080924-IGHV3-15	TTGCTGGTATTTTAAAAGGTGTCC (SEQ ID NO:175)
VDJ-20080924-IGHV3-16	TTGTTGGTATTTTAAAAGGTGTCC (SEQ ID NO:176)
VDJ-20080924-IGHV4-1	CAGCTCCCAGATGGGTCC (SEQ ID NO:177)
VDJ-20080924-IGHV4-2	CGGCTCCCAGATGGGTCC (SEQ ID NO:178)
VDJ-20080924-IGHV4-3	GCTCCCAGATGTGGGTCC (SEQ ID NO:179)
VDJ-20080924-IGHV5-1	GGCTGTTCTCCAAGGAGTCT (SEQ ID NO:180)
VDJ-20080924-IGHV5-2	CCTCCACAGTGAGTGAGTCT (SEQ ID NO:181)
VDJ-20080924-IGHV5-3	CCTAGCTATTCTCCAAGGAGTCT (SEQ ID NO:182)
VDJ-20080924-IGHV6-1	GCCTCCCATGGGGTGTCC (SEQ ID NO:183)
VDJ-20080924-IGHV6-2	GGGCCTCCATGGGTGTCC (SEQ ID NO:184)
VDJ-20080924-IGHV7-1	CAGCAGCAACAGGTGCCC (SEQ ID NO:185)
VDJ-20080924-IGHV7-2	GCAGCAGCAACAGGTACCT (SEQ ID NO:186)

454 library construction oligonucleotides

Adapter A oligonucleotides

454 adapter A1_MID1	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGACGAGT*G*C*G* T (SEQ ID NO:187)
454 adapter A2_MID2	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGACGCTC*G*A*C*A (SEQ ID NO:188)

454 adapter A3_MID3	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGAGACGC*A*C*T*C (SEQ ID NO:189)
454 adapter A4_MID4	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGAGCACT*G*T*A*G (SEQ ID NO:190)
454 adapter A5_MID5	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGATCAGA*C*A*C*G (SEQ ID NO:191)
454 adapter A6_MID6	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGATATCG*C*G*A*G (SEQ ID NO:192)
454 adapter A8_MID8	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGCTCGCG*T*G*T*C (SEQ ID NO:193)
454 adapter A9_MID9	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGTAGTAT*C*A*G*C (SEQ ID NO:194)
454 adapter A10_MID10	C*C*A*T*CTCATCCCTGCGTGTCTCCGACTCAGTCTCTA*T*G*C*G (SEQ ID NO:195)
454 adapter Aprime1_MID1	A*C*G*C*ACTCGTCTGAGTCG*G*A*G*A (SEQ ID NO:196)
454 adapter Aprime2_MID2	T*G*T*C*GAGCGTCTGAGTCG*G*A*G*A (SEQ ID NO:197)
454 adapter Aprime3_MID3	G*A*G*T*GCGTCTCTGAGTCG*G*A*G*A (SEQ ID NO:198)
454 adapter Aprime4_MID4	C*T*A*C*AGTGCTCTGAGTCG*G*A*G*A (SEQ ID NO:199)
454 adapter Aprime5_MID5	C*G*T*G*TCTGATCTGAGTCG*G*A*G*A (SEQ ID NO:200)
454 adapter Aprime6_MID6	C*T*C*G*CGATATCTGAGTCG*G*A*G*A (SEQ ID NO:201)
454 adapter Aprime8_MID8	G*A*C*A*CGCGAGCTGAGTCG*G*A*G*A (SEQ ID NO:202)
454 adapter Aprime9_MID9	G*C*T*G*ATACTACTGAGTCG*G*A*G*A (SEQ ID NO:203)
454 adapter Aprime10_MID10	C*G*C*A*TAGAGACTGAGTCG*G*A*G*A (SEQ ID NO:204)
Adapter B oligonucleotides	
454 adapter B	/5BioTEG/C*C*T*A*TCCCCTGTGTGCCTTGGCAGTC*T*C*A*G (SEQ ID NO:205)
454 adapter B prime	C*T*G*A*GACT*G*C*C*A (SEQ ID NO:206)

[0236] Table 1.

454 GS FLX library construction and sequencing

[0237] Immunoglobulin V_H PCR samples were processed following the 454 GS FLX Titanium General Library Preparation protocol. Briefly, samples were blunt ended and 5' phosphorylated using End-ItTM DNA End-Repair Kit (Epicentre Biotechnologies,

Madison, WI) in 50 μ L reactions according to manufacturer's instructions, followed by QIAquick PCR Purification (Qiagen, Valencia, CA) in the presence of 5 μ L of 3M NaOAc, pH 5.5. Roche 454 compatible DNA adapters containing a unique molecular identifier bar-code (MID) for each sample were ligated in a 100 μ L reaction volume in the presence of 30 pmoles of both adapter A (including MID 1 to 8) and adapter B (identical for all samples) and 1200 U of Rapid T4 DNA Ligase (Enzymatics, Beverly, MA) at 22°C for 30 min. In order to remove unligated adapters, samples were purified using a 1:1 ratio of AMPure XP beads (Agencourt Bioscience Corporation, Beverly MA) and eluted in 50 μ L of 10 mM Tris-HCL. The ends of the ligated fragments of the library-adapters were filled-in in 100 μ L reactions using 32 U of Bst DNA Polymerase (NEB, Ipswich, MA) in the presence of 625 μ M dNTPs and 1X ThermoPol Reaction Buffer. Reactions were purified using the MinElute PCR Purification Kit (Qiagen, Valencia, CA), and eluted in 15 μ L of EB buffer. Samples were separated on 6% Novex® TBE Gels (Invitrogen, Carlsbad, CA) at 200 volts for 26 min, from which the proper band (approximately 450 bp) was extracted at 50°C for 1 hr and resuspended in 20 μ L of 10 mM Tris-HCL as detailed previously (Vigneault et al. (2008) *Nat. Methods* 5:777). All steps related to the single-stranded template DNA (sstDNA) library isolation were found to be unnecessary for subsequent emulsion PCR and 454 sequencing. Library concentrations were determined using a 2100 Bionanalyzer (Agilent, Foster City, CA), then diluted to equivalent concentrations. All 8 blood samples, each representing a unique time point and characterized by a unique MID bar-code, were then pooled at an equimolar concentration. Emulsion PCR, bead enrichment and 454 GS FLX sequencing were performed at the 454 Life Sciences facility according to the manufacturer's standard protocol. This entire process (including cDNA synthesis and PCR) was repeated independently multiple times, or specifically on one sample only in order to generate complete technical replicates for the various analyses of the study.

Informatics pipeline

[0238] Raw sequencing data was analyzed with the 454 amplicon pipeline to produce quality-trimmed reads. The reads were size filtered for the appropriate amplicon size (350-500 bp) and converted to coding strands by comparing k-mer compositions of the query reads against the IMGT/GENE-DB (Giudicelli et al. 2005 *Nucleic Acids Res.* 33:D256) database and its reverse complement. The reads were then aligned to the IMGT/GENE-DB reference database to determine the V, D, and J usage, and extract the CDR3 junction region. Briefly, k-mer compositions of query sequences were compared with reference segments to find the top matches, and the best match was chosen using traditional dynamic programming alignment. Reads were then partitioned by V-J usage, and their CDR3 junctions were clustered using single-linkage or complete-linkage agglomerative hierarchical clustering with the Levenshtein edit distance as metric. Clones were defined by cutting the linkage tree at 4.5 edits. By partitioning reads into the separate time points (by identifying their MID bar-code), a time series was defined for each clone. The time series were clustered using k-means clustering with k=100 and the Euclidean distance metric. To eliminate trivial clones that appeared at single time points, clones showing positive counts in at least two time points were filtered. Time series were also clustered using self-organizing maps (SOM) as implemented in the GEDI software package. The analysis software described herein is available for download as a python module (Worldwide Web Site: arep.med.harvard.edu/vdj).

Primer design

[0239] All oligonucleotides were ordered from Integrated DNA Technologies (IDT, Coralville, IA). For the design of the upstream variable-region oligonucleotides (IGHV-PCR), the L-PART1 and L-PART2 sequences were extracted from all IMGT/GENE-DB³ reference segments annotated as “functional” or “ORF.” These two segments are spliced together *in vivo* to form the leader sequence. The primer sequence was positioned to cross the exon-exon boundary to ensure amplification from cDNA rather than gDNA. The primers were arranged to have 6 nucleotides 3' of the exon boundary, and ranged from 18-24 nucleotides in length to target their predicted melting temperatures to 60°C. All duplicate

sequences were eliminated. For the design of the downstream constant-region oligonucleotides (IGHC-RT and IGHG-PCR), the first 100 nucleotides of the CH1 exon were extracted from the IMGT/GENE-DB³. Oligonucleotides were then selected as close as possible to the 5' end of the C-region (close to the variable region of the receptor), starting with 3 nt from the end. These were hand-picked both for IGHG-RT and IGHG-PCR to take advantage of sequence conservation between different variants, and to ensure that isotypes would be distinguishable. All oligonucleotide sequences are shown in Table 1.

Control library

[0240] In order to assess PCR bias and amplification efficiency, total RNA from multiple individuals was extracted, pooled together, and reverse transcribed as described in the Methods Summary. Independent PCR reactions were conducted for each C-region gene-specific primer (IGHG-PCR) for 25 cycles, from which the proper band of interest was gel extracted as described previously (Vigneault et al., *Supra*). To monitor the amplification efficiency of the constant region primers, the individual PCR amplicons were quantified using the 2100 Bionanalyzer (Agilent, Foster City, CA), followed by real-time PCR performed in triplicate using the KAPA SYBR® FAST qPCR Kit (Kapa Biosystems, Woburn, MA) on a Bio-Rad CFX96 real time PCR instrument (Bio-Rad, Hercules, CA). Both linear regression PCR (LinRegPCR) (Ramakers et al. (2003) *Neurosci. Lett.* 339:62) and standard dilution curve analysis were used to determine and compare amplification efficiencies (Table 2).

Individual primer / reaction	LinRegPCR mean efficiency on uniform template concentration (triplicate of 1/125)	LinRegPCR mean efficiency on STD curves (1/5 dilution series)	STD curve mean efficiency (1/10 dilution series)	Average efficiency
VDJ-20080924-IGHG-1	1.774	1.650	1.803	1.742

VDJ-20080924-IGHG-2	1.784	1.740	1.824	1.783
VDJ-20080924-IGHM	1.729	1.757	1.766	1.751
VDJ-20080924-IGHA	1.777	1.731	1.782	1.763
VDJ-20080924-IGHD	1.779	1.726	1.755	1.753
VDJ-20080924-IGHE	1.607	1.541	1.831	1.660

[0241] Table 2.

Positive strand identification of 454 sequencing reads

[0242] Following size-filtering of reads for the expected V_H size (Figure 18), coding strands were identified, as it was important for the subsequent alignment pipeline. To do so k-mers were pre-computed using the Positive Strand comb (Table 3) for the entire reference segment database, and separately for its reverse complement. The same k-mers for the query read were then computed, and the coding strand determined by computing the number of shared k-mers of the positive versus negative version of the database.

Comb	Pattern
Positive Strand	111111111111
Alignment A	111011001011010111
Alignment B	1111000100010011010111
Alignment C	111111111111
Alignment D	110100001100010101111
Alignment E	1110111010001111
Mini-Alignment A	111011
Mini-Alignment B	110111

[0243] Table 3.

VDJ alignment overview

[0244] VDJ alignment allows identification of which V, D, and J segments are used in a given sequencing read. For each segment, a heuristic method was employed by computing k-mer counts using multiple seed combs (Table 3). The candidate segments were ranked by

the amount of overlap in their k-mer distributions. A full dynamic programming alignment was then performed on the highest scoring candidates to determine the best match.

VDJ alignment process

[0245] To maximize the number of distinguishing nucleotides, the alignment was performed in order of decreasing segment length (V then J then D). In order to minimize the amount of off-target sequence (especially for the k-mer counts), the previously aligned V or J region was pruned off before attempting alignment of the next segment. In order to align the V and J segments, 11-mer or 12-mer seeds were computed using the combs listed in Table 3. The k-mer counts were also pre-computed on all reference segments. The top 5 reference V segments and the top 2 reference J segments were chosen for dynamic programming alignment and scoring. Since it was known that the V and J segments must reside at the ends of the reads, a method that is similar to the Needleman-Wunsch algorithm (Needleman and Wunsch (1970) *J. Mol. Biol.* 48:443) was used. In contrast to the canonical algorithm, zero initial conditions were used to allow the start of the alignment to occur anywhere without penalty. The alignment was then reconstructed and scored by starting at the maximum value of the score matrix along the last row or last column, and backtracing. Finally, the identified V or J segments were removed before proceeding to the J or D alignment, respectively.

D region alignment

[0246] The D alignments were performed similarly. However, the two Mini-Alignment 5-mer seeds listed in Table 3 were used to pick the top 10 reference D segments to carry through to dynamic programming alignment. In this case, the canonical Smith-Waterman local alignment algorithm (Smith and Waterman (1981) *J. Mol. Biol.* 147:195) was performed, as no prior information as to where the D segment should reside was available.

CDR3 extraction

[0247] The V and J segments were pruned according to the IMGT annotations for the CDR3 junction region: from the second conserved cysteine residue in the V region through the conserved tryptophan or phenylalanine in the J region. This segment was stored as the junction region of the corresponding variable region.

VDJ alignment performance

[0248] Since there was currently no large database of human heavy chain immunoglobulin sequences to validate the performance of the algorithm, the aligner was calibrated using the V-QUEST algorithm provided as part of the IMGT database. 66,497 reads from three different sequencing runs were aligned using the V-QUEST alignment software. For each read, the dynamic programming alignment scores for the V, D, and J regions were computed separately for every possible reference segment, in addition to the correct segment produced with the V-QUEST aligner. ROC curves showed that V region alignment was the most reliable, followed by J, followed by D, as expected (Figure 8). The D region showed the most ambiguous alignment, which was expected due to its short length and high mutated nature. Therefore, the D region alignments were ignored in downstream analyses, and the V and J alignments were focused upon, along with the corresponding CDR3 junction region. The ROC curves were used to choose scoring thresholds to keep the number of falsely matched alignments to approximately 10% or lower.

Sequence clustering

[0249] Sequence clustering was performed in order to group the sequences (reads) into unique clones. This process was primarily used to associate sequences that originated from the

same cell/clone, while allowing minor variations attributable to sequencing errors. Single- or complete-linkage agglomerative hierarchical clustering was used with Levenshtein edit distance as the metric. To make the clustering process more tractable, the reads were partitioned based on V-J identity. Within each partition, sequence clustering was performed using only the CDR3 junction sequence. Furthermore, all identical junctions were collapsed prior to clustering to improve performance. Because cluster distances only depended on the unique cluster members, this did not affect the final clustering results (though this is not true for average-linkage clustering methods). As described herein, cluster sizes varied over several orders of magnitude. Below are multiple alignments (using CLUSTALW) of several smaller clusters as illustrations. Most of the differences between sequences were indels near homopolymers, which are the most common mutation type for 454 sequencing.

[0250] Example Cluster 1:

```

1  TGTGCGAGAGAGGGCTACGGTGACTACCGTTACTACTACGGTATGGACGTCTGG 54
   (SEQ ID NO:1)
2  TGTGCGAGAGAGGGCTACGGTGACTACCGTTACTACTACGGTATGGACGTCTGG 54
   (SEQ ID NO:2)
5  TGTGCGAGAGAGGGCTACGGTGACTACCGTTACTACTACGGTATGGACGTCTGG 54
   (SEQ ID NO:3)
6  TGTGCGAGAGAGGGCTACGGTGACTACCGTTACTACTACGGTATGGACGTCTGG 54
   (SEQ ID NO:4)
3  TGTGCGAGAGAGGACTACGGTGACTAC-GT--CTACTACGGTATGGACGTCTGG 51
   (SEQ ID NO:5)
4  TGTGCGAGAGGGTACTACGGTGACTACGG---CCACTACGGTATGGACGTCTGG 51
   (SEQ ID NO:6)
   ***** * ***** * * *****

```

[0251] Example Cluster 2:

```

1  TGTGCGGCAGTTCCCCCCCCT-NAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 57
   (SEQ ID NO:7)
3  TGTGCGGCAGTTCCCCCCCCT-NAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 58
   (SEQ ID NO:8)
9  TGTGCGGCAGTTCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 59
   (SEQ ID NO:9)
22 TGTGCGGCAGTTCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 58
   (SEQ ID NO:10)

```


5 TGTGCGGCAGTTCCCCCCCCCTCAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 58
 (SEQ ID NO:11)
 24 TGTGCGGCAGTTCCCCCCCCCTC---AGGAACGACATTT---GGGTGCTTTT-GAGATCTGG 53
 (SEQ ID NO:12)
 6 TGTGCGGCAGTTCCCCCCCCCTC--AGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 56
 (SEQ ID NO:13)
 10 TGTGCGGCAGTTCCCCCCCCCTC--AGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 56
 (SEQ ID NO:14)
 11 TGTGCGGCAGTTCCCCCCCCCTC--AGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 56
 (SEQ ID NO:15)
 13 TGTGCGGCAGTTCCCCCCCCCTC--AGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 56
 (SEQ ID NO:16)
 19 TGTGCGGCAGTTCCCCCCCCCTC--AGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 56
 (SEQ ID NO:17)
 21 TGTGCGGCAGTTCCCCCCCCCTC--AGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 56
 (SEQ ID NO:18)
 4 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 57
 (SEQ ID NO:19)
 17 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 57
 (SEQ ID NO:20)
 8 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 57
 (SEQ ID NO:21)
 18 TGTGCGGCAGTTCCCCC--CT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 55
 (SEQ ID NO:22)
 23 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 57
 (SEQ ID NO:23)
 12 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTTGGGGTGCTTTTGGAGATCTGG 59
 (SEQ ID NO:24)
 14 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTTGGAGATCTGG 58
 (SEQ ID NO:25)
 16 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 57
 (SEQ ID NO:26)
 0 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTT-GAGATCTGG 57
 (SEQ ID NO:27)
 7 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTTGGAGATCTGG 58
 (SEQ ID NO:28)
 2 TGTGCGGCAGTTCCCCCCCCCT-CAGGGAACGACATTTT-GGGGTGCTTTTGGAGATCTGG 58
 (SEQ ID NO:29)
 20 TGTGCGGCAGTTCCCCCCTCA---GGGAACGACATTT---GGGGTGCTTTT---GAGATCTGG 53
 (SEQ ID NO:7)
 15 TGTGCGGCAGTTCCCCCCCCCTCCAGGGAACGACATTT---GGGGTGCTTTT-GAGATCTGG 57
 (SEQ ID NO:30)

[0252] Example Cluster 3:

4 TGTGCGACGG-TGGGA-GTTCCCC-ACCGGTTTT-GATATCTGG 40
 (SEQ ID NO:31)
 79 TGTGCGACGG-TGGGACGTTCCCCTACCGGTTTT-GATATCTGG 42
 (SEQ ID NO:32)

1 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:33)
16 TGTGCGACGG-TGGGA-GTTCCC--AC-GGTTTT-GATATCTGG 38
(SEQ ID NO:34)
3 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:35)
6 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:36)
8 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:37)
11 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:38)
20 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:39)
27 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:40)
32 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:41)
38 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:42)
43 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:43)
49 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:44)
55 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:45)
59 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:46)
64 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:47)
69 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:48)
74 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:49)
12 TGTGCGACGG-TGGGA-GTTCTC--ACCGGTTTTTGATATCTGG 40
(SEQ ID NO:50)
67 TGTGCGACGG-TGGGA-GTTC-C--ACCGGTTTT-GATATCTGG 38
(SEQ ID NO:51)
70 TGTGCGACGG-TGGGA-ATTC-C--ACCGGTTTT-GATATCTGG 38
(SEQ ID NO:52)
10 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTTTGATATCTGG 40
(SEQ ID NO:53)
24 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTTTGATATCTGG 40
(SEQ ID NO:54)
34 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTTTGATATCTGG 40

(SEQ ID NO:55)
44 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTTTGATATCTGG 40
(SEQ ID NO:56)
47 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTTTGATATCTGG 40
(SEQ ID NO:57)
85 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTTTGATATCTGG 40
(SEQ ID NO:58)
87 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTTTGATATCTGG 40
(SEQ ID NO:59)
13 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:60)
15 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:61)
22 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:62)
28 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:63)
33 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:64)
39 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:65)
45 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:66)
50 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:67)
56 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:68)
61 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:59)
65 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:70)
71 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:71)
78 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:72)
83 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:73)
89 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:74)
92 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:75)
95 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:76)
98 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:77)

101 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:78)
23 TGTGCGACGG-TGG-A-GTTCCC--ACCGGTTTT-GATATCTGG 38
(SEQ ID NO:79)
51 TGTGCGACGG-TGG-A-GTTCCC--ACCGGTTTT-GATATCTGG 38
(SEQ ID NO:80)
80 TGTGCGACGG-TGG-A-GTTCCC--ACCGGTTTT-GATATCTGG 38
(SEQ ID NO:81)
17 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:82)
25 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:83)
30 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:84)
35 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:85)
40 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:86)
46 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:87)
52 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:88)
57 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:89)
62 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:90)
66 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:91)
72 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:92)
81 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:93)
84 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:94)
90 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:95)
93 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:96)
96 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:97)
99 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:98)
100 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:99)
97 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39

(SEQ ID NO:100)
94 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:101)
91 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:102)
86 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:103)
82 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:104)
77 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:105)
75 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:106)
73 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:107)
68 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:108)
63 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:109)
58 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:110)
53 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:111)
48 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:112)
41 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:113)
37 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:114)
31 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:115)
26 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:116)
19 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:117)
9 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:118)
7 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:119)
5 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:120)
2 TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTT-GATATCTGG 39
(SEQ ID NO:121)
14 TGTGCGACGG-TGGGA-GTTCCC-TACCGGTTTT-GATATCTGG 40
(SEQ ID NO:122)


```

21  TGTGCGACGG-TGGGA-GTTCCC-TACCGGTTTT-GATATCTGG 40
   (SEQ ID NO:123)
42  TGTGCGACGG-TGGGA-GTTCCC-TACCGGTTTT-GATATCTGG 40
   (SEQ ID NO:124)
102 TGTGCGACGG-TGGGA-GTTCCC-TACCGGTTTT-GATATCTGG 40
   (SEQ ID NO:125)
18  TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTT--GATATCTGG 38
   (SEQ ID NO:126)
54  TGTGCGACGG-TGGGA-GTTCCCCTACCGGTTTT-GATATCTGG 41
   (SEQ ID NO:127)
60  TGTGCGACGG-TGGGA-GTTCCCCTACCGGTTTT-GATATCTGG 41
   (SEQ ID NO:128)
76  TGTGCGACGGGTGGGACGTTCCCCTACCGGTTTT-GATATCTGG 43
   (SEQ ID NO:129)
88  TGTGCGACGG-TGGGA-CTTCCC--ACCGCTTTT-GATTCTGG 39
   (SEQ ID NO:130)
29  TGTGCGACGG-TGGGA-GTTCCC--ACCGGTTTGTATTATCGTG 40
   (SEQ ID NO:131)
36  TGTGCGACGG-TGGGACGTTCCCCTACCGGTTTGTAT-ATCGTG 42
   (SEQ ID NO:132)
*****  ***  *  ***  *  **  *  ***  **  *

```

[0253] The distribution of cophenetic distances observed in the linkage tree was then examined in order to determine the optimal distance to clip the tree. It was expected that a large number of linkage events occurring at low cophenetic distance that represent sequencing errors would be observed. Indeed, a rapid drop in the distribution was observed until a distance of 4-5 edits, after which the distribution showed a shoulder and a slower descent (Figure 19A). This was consistent with the observed relationship between distance cutoff and number of clusters obtained (Figure 19B), as there was also a rapid drop at short distance followed by a shoulder and marked slowing.

Time series clustering: k-means

[0254] While the sequence clustering described above was performed on combined data for all eight time points, each unique clone identifier was further partitioned into the 8 time points by identifying the 454 MID bar-codes. Each clone thus defined a time series where the number of reads was a proxy for expression level. As an initial analysis, k-means clustering was performed on all time series using combined data from runs SR1,

SR2, and TR1. The k-means algorithm was performed on clone frequencies (read counts in a given time point normalized by the number of reads) using the Euclidean distance metric with 100 clusters (chosen arbitrarily). The largest clusters contained clones that appeared for a single time point and disappeared. This was observed at a variety of expression levels and at all time points (Figure 15A). When these clusters were filtered out and k-means clustering performed again, a large variety of dynamic behavior was observed. However, some of the largest clusters were still clones that rapidly rose and fell in frequency (Figure 15B).

Time series clustering: self-organizing maps (SOM)

[0255] Due to the arbitrary selection in clone number for k-means clustering and the large diversity of dynamic behavior, the time series was clustered using a SOM method as implemented in the GEDI software package (Eichler et al. (2003) *Bioinformatics* 19:2321). Each tile within a mosaic represented a mini-cluster of clones that have highly similar expression patterns across all the analysed time points. By defining a topology on the tiles, similar clusters are grouped together, allowing simple determination of the proper number of cluster groups. The same clones are forced to the same mosaic position for all SOM maps, allowing direct comparison of the clones based on the overall mosaic pattern. The color of tiles indicated the centroid value of gene expression level for each mini-cluster. SOM clustering was performed on a rectangular, 50x49 grid using random initialization. An Euclidean distance was defined on the time series, and two training iteration phases were performed (20 and 80). Following clustering, clusters of interest were manually identified for further analysis.

Reproducibility, sampling requirements, and limits of quantitation

[0256] Multiple analyses were performed to determine whether the VDJ-ome was sampled deep enough and to what level the dynamics could be reliably quantified (i.e., the limit of quantitation (Currie (1968) *Analytical Chemistry* 40:586)). The replicate time series of

randomly chosen clones was plotted (from runs SR1, SR2, and TR1) to get a qualitative view of the replication (Figure 15). Scatter plots of replicate runs showed high levels of correlation (Figure 9, Pearson correlations shown). To determine the sampling characteristics of the samples, pairs of subsamples of the data were drawn of different sizes and correlation coefficients were computed (Figure 2C). The correlation coefficients approached unity at a sampling level of 10^5 . To determine limits of quantitation, scatter plots were generated for each time point of a given replicate against another. The data were binned on one axis and the mean and standard deviation were computed on the other axis. Using these values, the coefficient of variation (CV) was computed as a function of expression level (Figure 10). It was expected that higher expression levels would have lower relative variances and so should have resulted in smaller CVs. The noise in the middle of the curves was an artifact of the weighted sampling scheme used, and noise was also observed at high expression levels due to poor sampling. A lower limit CV of approximately 0.5 occurred at clone frequencies of $<10^{-4}$ - 10^{-3} , depending on the samples. Sampling of lower frequency clones rapidly approached the Poisson noise regime (CV of 1). Different numbers of reads in different time points/replicates caused the onset of Poisson noise at different frequencies.

Diversity estimates

[0257] Multiple methods were utilized to estimate antibody diversity. To estimate typical diversities of small 1 mL blood samples, the Chao1 and ACE abundance-based estimators were used (Magurran, A.E. Measuring Biological Diversity. (Blackwell Pub., Malden, MA; 2004)). Briefly, Chao1 and ACE were derived as non-parametric abundance-based estimators for diversity. Following Magurran, if S_{obs} was the number of species observed in the sample, F_1 was the number of species observed once (singletons), and F_2 was the number of species observed twice (doubletons), then the Chao1 estimator was defined as

$$S_{Chao1} = S_{obs} + \frac{F_1^2}{2F_2}.$$

[0258] If S_{rare} was the number of rare species (species with less than or equal to 10 observations), S_{abund} was the number of abundant species (species with greater than 10 observations), N_{rare} was the number of individuals in rare species, F_i was the number of species with i observations, $C_{ACE} = 1 - F_1/N_{rare}$, and

$$\gamma_{ACE}^2 = \max \left\{ \frac{S_{rare}}{C_{ACE}} \frac{\sum_{i=1}^{10} i(i-1)F_i}{N_{rare}(N_{rare}-1)} - 1, 0 \right\}$$

then

$$S_{ACE} = S_{abund} + \frac{S_{rare}}{C_{ACE}} + \frac{F_1}{C_{ACE}} \gamma_{ACE}^2.$$

[0259] However, the samples used were non-homogeneous in that they were drawn at varying time points after an immune challenge, which may affect the antibody distribution, as compared to homogeneous samples. It was expected that such effects would be small, and so the estimators were still applied. More significantly, the coverage level just passed an inflection point (indicating the start of convergence towards an asymptote). In such cases, the actual diversities tended to be higher than the estimate values, and the estimators functioned as lower bounds on diversity. The re-sampled versions of these estimators are shown in Figures 12A-12D. Inflection points were observed for almost all curves, indicating approach towards an asymptote.

[0260] Similarly, the analogous incidence-based estimators Chao2, ICE, Jack1, Jack2 (Magurran, *Supra*) were used to estimate total blood diversity, using the multiple blood samples analogously to multiple quadrants in ecology. The same statistical considerations applied to these sets of estimators. The results are shown in Figure 2E. The diversity estimation method of Arstila et al. (Arstila et al. (1999) *Science* 286:958) was also simulated using every possible V-J combination and every possible pseudo-spectratype peak. This method did not take statistical considerations into account a very large range of possible

diversity estimates was expected. Indeed, a very broad distribution of values was obtained, though the mean value was still near the 10^6 range (Figure 2E).

Distribution of fluctuations

[0261] To characterize the dynamics of the antibody clones further, we analyzed the distribution of their typical fluctuations. For time series that were positive at all time points, the log-fold change from one time point was computed over the preceding time point. The distributions are shown in Figures 20A-20B. They appeared to be log-normally distributed and centered at zero. The -14 d time point distribution was off-center due to a sampling artifact. It was also expected that the variance of the distribution would be correlated with the duration of the time interval (i.e., longer durations lead to larger fluctuations). Indeed, Figures 20A-20B show a strong correlation, with a Pearson correlation coefficient of 0.91.

Pseudo-spectratype generation

[0262] For each time point, the count data was partitioned by V-J usage and then by CDR3 junction length. For each possible in-frame length (multiple of 3) the number of in-frame reads were computed versus out-of-frame reads that were one base too long or too short, and fitted this data to a Gaussian distribution. For each V-J combination, all such Gaussians were superposed and the curve renormalized to the total number of reads. Full pseudo-spectratypes are shown in Figure 13. The use of Gaussians were for visualization purposes only.

Alignment to specificity-annotated IMGT/LIGM database

[0263] The IMGT/LIGM database (Giudicelli et al. (2006) *Nucleic Acids Res.* 34:D781) contains about 9000 sequences with annotated specificities, 14 of which are specific for influenza hemagglutinin, 6 for hepatitis A, and 85 for hepatitis B. Though this represents a very

small sample, the sequencing reads to this set of specificity-annotated sequences were aligned to search for highly identical sequences. To perform the alignments, the exonerate software package (Slater and Birney (2005) *BMC Bioinformatics* 6:31), which allows alignment using various alignment models, was used. Alignments were performed using either the full read or only the CDR3 junction, and using translated versions and/or nucleotide versions of the sequences. Because many background matches were expected (in the case of full read alignment), and due to the sensitivity of binding affinity to the exact amino acid sequence, the alignments were ranked by sequencing identity, and only matches that were almost completely identical were analyzed by hand. Examples of matches to influenza-specific or hepatitis-specific (Figures 21 and 22) antibodies were performed using full nucleotide sequence alignment. Poor alignment was observed at the CDR3 junction region, which is the most common form of misalignment. The best matching alignment was found by aligning translated junction regions only. An anti-digoxin antibody was successfully matched. The junction-only alignment (Figure 23) and the full read alignment (Figure 24) are shown.

Software tools and data availability

[0264] The analysis software package used for the experiments described herein is available for download at the Worldwide Web Site: arep.med.harvard.edu/vdj, or the latest version can be pulled as a git repository from GitHub at the Worldwide Web Site: github.com/laserson/vdj. The software was implemented primarily in python with some C code used to increase performance. The NumPy (the Worldwide Web Site: numpy.scipy.org/) and SciPy (the Worldwide Web Site: scipy.org/) packages were also heavily relied upon for numerical calculations, and the matplotlib (the Worldwide Web Site: matplotlib.sourceforge.net/) package was relied upon for data visualization. The sequencing data is available for download in FASTA format at the Worldwide Web Site: arep.med.harvard.edu/vdj along with processed versions of it in XML format. The sequencing data will be submitted to the NCBI Short Read Archive, and to the

international ImMunoGeneTics (IMGT) database for incorporation into Laboratoire d'ImmunoGénétique Moléculaire (LIGM).

EXAMPLE V

Heavy and Light Chain Capture From A Unique Cell

[0265] Methods are provided to capture both the heavy and light chains of an antibody originating from a single, enabling the simultaneous capture of millions of cells at once (instead of prior arts of sorting one cell per well in a plate).

Small oligo-transfection of single cells

[0266] Two oligos, each presenting a complementary sequence to the heavy and light chain respectively, are partially annealed to each other through a universal sequence. This partial annealing bond is maintained in subsequent step and the use of locked nucleic acid (LNA) can be useful for this purpose. The oligos can also harbor phosphorothioates to protect the oligo from nucleases. Additionally, a distal or internal biotin can be included in one or both of the oligos, which will be use in recovery or the molecules downstream. This oligo linker is then transfected into lymphocytes samples using common transfection technologies such as TransIT®-Oligo Transfection Reagent (Mirus Bio LLC) for example. The transfection component can also include RNase inhibitor as a precaution. The cells are then incubated at higher temperature so that annealing of the cell RNA that one which to capture is accomplished (can also be done with DNA). Ideally, the temperature is raised enough to favor melting of the targeted RNA secondary structure and to achieve specific annealing to the oligo linker by matching melting temperature as close as possible (generally near 65°C), while the partial annealing of the two oligos is maintain due to the inherent sequence design (particularly with the use of LNA primers, where the annealing junction can be design to reach temperature above 90°C). Following incubation annealing of the targeted RNAs, the cells can be chemically lysed and the

oligo linker-target RNA molecule recovered through its biotin using a streptavidin coated magnetic bead (such as Dynabead M280 from Invitrogen, or other equivalent). One could also use any other solid substrate coated with streptavidin. The oligo linker 3' end can then be used directly for reverse transcription (using superscript III, Invitrogen, or other equivalent reverse transcriptase) of both the heavy and light chains, followed by second strand cDNA synthesis. The final molecule is amplified off the magnetic beads using distal primer with results in the recovery of a full ScFv composed of the heavy and light chains originating from a single cell, and this accomplished in millions of cells at once in the same reaction sample.

Bead Capture

[0267] In certain exemplary embodiments, a magnetic bead is coupled with a mixture of oligos complementary to the heavy and light chains but also harboring a universal sequence upstream (a non-magnetic bead will work, however magnetic beads facilitate downstream washes and recoveries). In certain aspects, covalent coupling chemistry or streptavidin-biotin linkages (e.g., if a biotin is added to the oligos during manufacturing) are used to produce oligo-beads. The oligo-bead is then either transfected into single cells using common transfection technologies, or liposome based transfection approaches (as long as the bead is small enough to be transfected), or the bead (or a few beads), is encapsulated inside an emulsion in the presence of a single lymphocyte. The cell is then lysed using heat denaturation or using freeze thaw cycles. In certain aspects, RNase inhibitors are added during the emulsification step. The emulsions are then either PCR amplified (assuming PCR and primer were also introduced during the emulsification), or the incubated at an optimal oligo hybridization temperature and/or salt concentration such that both heavy and light chains will anneal to their complementary sequence on the bead. The emulsions are then broken and the beads recovered. Unbound cell components and non-specific DNA or RNA are washed away and the resulting bead-RNA is subjected to reverse transcription (for RNA) or primer extension (for DNA), so that the bead now harbor the full heavy and light chains sequence of a same single cell. These beads can

then be subjected to any of the described techniques to link both chains together prior sequencing, such as overlapping PCR (SOE-PCR), or using CRE-LOW recombination of both chains either in emulsion, or in a super diluted reaction. Cre-lox linking of single cell heavy and light chains has been demonstrated previously in formaldehyde cross-linked lymphocytes but with very poor efficiency (Chapal et al. (1992) *Biotechniques*, PMID: 9298226), whereas methods provided herein have coupled cross-linking to the process of high-throughput sequencing, thus enabling one of ordinary skill in the art the ability to survey large immune repertoire.

SOE-PCR variation

[0268] SOE-PCR has never been coupled to the methods of high-throughput sequencing and analysis of large immune repertoire, but has instead always been limited to the study of a limited number of cells (such as conducted by Symphogen in their Symplex technology (Worldwide Web: symphogen.com/web/guest/symplex) described by Meijer (2006) *J. Mol. Biol.*, PMID: 16563430, for example). Accordingly, multiple methods of SOE-PCR are provided that improve efficiency. These methods are used in single cell SOE-PCR by either capturing single cell in emulsion or by cross-linking cells in formaldehyde as described further herein. This allows for generation of single cell ScFv of millions of single cells in a single reaction volume, so that it can be coupled with high-throughput sequencing and analysis of immune repertoires.

EXAMPLE VI

Expression and Functional Assay of Antibody and Nanobody Selected Candidates From High-Throughput Sequencing Analysis of a Human Immune Repertoire

In vitro expression of single domain antibody

- [0269] 1. A random colony from each clone was lifted by toothpick and inoculated into 5 mL of LB/Amp, and bacteria were incubated overnight at 37 °C in shaking at 225 rpm.
- [0270] 2. Bacteria were centrifuged at 4,700 g for 10 min, sup was discarded and pellets were used for miniprep (using a Qiagen miniprep kit).
- [0271] 3. Transcription reactions were set up: Nuclease free water 8 µL; 5X Transcription buffer 4 µL; NTP mix 4 µL; DNA template (~0.5 µg/mL) 2 µL; T7 RNA polymerase 2 µL; TOTAL 20 µL.
- [0272] 4. Reaction was incubated for 120 min at 32 °C. The solutions became cloudy after this time, indicating efficient transcription (calcium precipitates).
- [0273] 5. Translation reactions were set up directly: Lysate for protein expression 12.5 µL; Accessory proteins 2.5 µL; Nuclease free water 3.75 µL; Salt Solution A 1 µL; Amino acids –Met 0.5 µL; Amino acids –Leu 0.5 µL; RNase inhibitor 1 µL; Energy mix 1.25 µL; RNA template from transcription reaction 2 µL; TOTAL 25 µL.
- [0274] 6. Translation reactions were incubated at 32 °C for 2 hours, then 2 µL of the transcription reaction was added.
- [0275] 7. The reaction was incubated for further 2 hours at 32 °C, then spiked again with 1 µL of the transcription reaction and incubated for further 2 hours.

Proteins were purified as follows

- [0276] 1. The reaction was diluted with 28 µL of 2X His binding buffer (40 mM phosphate, 1 M NaCl, 40 mM imidazole, pH 7.4).

- [0277] 2. Samples were cleaned on Vivaspin 0.2 μ m columns to get rid of all debris and large complexes.
- [0278] 3. Ni-NTA sepharose resin (Amersham) was gently swirled for 5 minutes until all slurry was homogenous, then a total amount of (50 μ L x (# samples + 1)) was transferred to a 1.5 mL tube.
- [0279] 4. Resin was washed 3 times with 1X His binding buffer (20 mM phosphate, 0.5 M NaCl, 20 mM imidazole, pH 7.4) to remove ethanol and equilibrate resin.
- [0280] 5. 50 μ L of the resin were added to each sample.
- [0281] 6. The samples were flipped overnight at 4°C.
- [0282] 7. Samples were washed 3 times with 150 μ L of His binding buffer.
- [0283] 8. Proteins were eluted from resin in 100 μ L of His elution buffer (20 mM phosphate, 0.5 M NaCl, 0.5 M imidazole, pH 7.4) by flipping at RT for 1 hour.
- [0284] 9. Resin was removed and samples were desalted using Vivaspin 5K as follows: a. Vivaspin 5K columns were equilibrated with 0.4 mL of PBS. b. 100 μ L of sample were added to the column. c. Columns were placed in centrifuge with filter facing outwards, then centrifuged (15,000 g, 15 min). d. Flow-thru was discarded, and sample was resuspended in 0.4 mL of PBS and centrifuged again. e. Step d was repeated two additional times. f. Retentates (~50 μ L average) were transferred to fresh tubes, sodium azide solution was added to a final concentration of 0.02% and the samples were stored at 4 °C. Protein concentration and/or purity was evaluated using Nanodrop.

Gene synthesis and cloning protocol

[0285] PCR amplify IDT gene synthesis using M13 forward and reverse primer

[0286] Assemble a reaction as follows (1 tube / 13 tubes): ~50 ng/ μ l Gene synthesis (1 μ l / -); dH₂O (34.6 μ l / 449.8 μ l); 5x HF buffer (10 μ l / 130 μ l); 15 μ M primer 1* (1.7 μ l / 22.1 μ l); 15 μ M primer 2* (1.7 μ l / 22.1 μ l); 25 mM dNTP (0.5 μ l / 6.5 μ l); Phusion Hotstart (0.5 μ l / 6.5 μ l); (49 μ l/tube). Thermal cycle as follows: 9 to 12 cycles

- 1- 98°C for 30 sec
- 2- 98°C for 10 seconds
- 3- 50°C for 20 seconds
- 4- 72°C for 20 seconds go to step 2, 25x
- 5- 72°C for 5 min
- 6- 4°C pause

[0287] Qiagen purify, resuspend in 30 μ l.

Double Digest with EcoRI-HF and NotI-HF

[0288] Assemble reaction as follows

	1tubes	13tubes
• H ₂ O	10.5 (to 50 μ l)	136.5
• DNA	30 μ l	-
• 10xNEB buffer 4	5 μ l	65
• 100x BSA	0.5 μ l	6.5
• NotI-HF	2 μ l	26
• EcorI-HF	2 μ l	26

Incubate at 37°C for 15 min (20 μ l/tubes). Heat inactivate at 65°C for 20 min

[0289] Qiagen purify, elute in 42 (will load 20 one gel); Nanodrop a few ~50ng/ μ l. Extract using size select gels : Qiagen purify, elute in 30 μ l; Nanodrop : ~9ng/ μ l.

Ligation into thermo expression vector

[0290] Used 3 :1 ratio inserts to vector Assembled reaction as follows (1tube / 14tubes) : H₂O (4μl / 56 μl); 2x Ligase buffer (10 μl / 140 μl); Vector 50ng/μl (2μl (0.04 pmole) / 28 μl); ~350 bp Inserts (9ng/μl) (3μl (0.12 pmole = 30ng) / - μl); Rapid ligase (1 μl / 14 μl); (17μl/tubes). Incubate at 22°C for 30 min. Performed the cloning with X10 cells. Plated 200μl using beads, this morning plate were full, nice results. Recloned in NEB Shuffle T7 competent e coli (at C3026H). Used Ido mini prep, diluted 3-fold, used 1 μl (~80 ng) in 25μl of *E. coli*. Grew midi batch of selected clones.

[0291] Picked one colony into 50 ml LB +1x Ampicilin, grew overnight at 30°C

clone ID	name of clone	synthesis ID	binding	large batch
IDT plate (yello sticker row A) sampler un form left to right.				
1	GMC_1688_Day1_2	1-IDT		
2	GMC_1337_Day1_3	2-IDT		
-	GMC_299_Day1_4	3-IDT	not done yet	
3	GMC_162_Day1_5	4-IDT	10, 59	X
4	GMC_961_Day7_2	5-IDT		
5	GMC_1458_Day7_3	6-IDT		
6	GMC_1494_Day7_4	7-IDT	10, 59	X
7	GMC_769_Day7_5	8-IDT	59	X
8	GMC_967_Day21_2	9-IDT	59	X
9	GMC_750_Day21_3	10-IDT		
10	GMC_2629_Day21_4	11-IDT	10, 59	X
11	GMC_668_Day21_5	12-IDT		
DNA2.0 plate (red sticker any well form 1 to 5) sampler un from top to down.				
12	IDO_1_NEGCONT_BCELLTUMOR_optHEK1	IDO1		
13	GMC_1687_Day1_1	IDO2		
14	GMC_932_Day7_1	IDO3	59	X
15	GMC_393_Day7_6	IDO4	10, 59	X
16	GMC_376_Day21_1	IDO5		
17	IDO_6_POSCONT_GFPMINIMIZER_optHEK1	IDO6		X

Reclone

[0292] Recloned in T7 shuffle *E. coli* : 3, 6, 7, 8, 10, 12. Uses 2 µl of 13 dilution in 20 µl of T7 shuttle *E. coli* (NEB C3026H). Plated 200 µl on super large plate (too much, plenty of clone, try 100 ul next time). Grew one colony each in 5 ml of LB at 30 °C overnight. Used this a starter culture 500 µl for 50 ml LB + 1000x ampi (50 mg/ml use at 1X). Grew for 3-4 hr, checked OD. Nanodrop. Cuvette : 0.3. Pedestal : 0.08 (nanodrop says that should be 10-fold lower then regular OD600). Kept some of 10 cell lysate for direct SPR testing.

[0293] Q8: How should I express my protein of interest in Shuffle? A8: For initial conditions we recommend using rich media at 30 °C. Otherwise, overnight at 16 °C is possible. At 30 °C or 16°C, inoculate 1% overnight culture and grow cells at 30 °C for 3 hours until OD600 ~ 0.8 and then induce expression of protein for at least 5 hours at 30 °C or overnight at 16 °C. If using 37°C, inoculate 1% overnight culture and grow cells for 2 hours at 37°C until OD600 ~0.8 and then induce expression of protein for at least 6 hours at 37°C.

Protocol

[0294] Transformed expression plasmid into the T7 Express strain. Plated on antibiotic selection plates and incubate overnight at 37°C. Resuspended a single colony in 10 ml liquid culture with antibiotic. Incubated at 37°C until OD600 reached 0.4 - 0.6. Induced with 40 µl of a 100 mM stock of IPTG (final concentration of 0.4 mM) and induced for 2 hours at 37 °C. Checked for expression either by Coomassie stained protein gel, Western Blot or activity assay. Checked expression in both the total cell extract (soluble + insoluble) and the soluble fraction alone. For large scale, inoculated 1 L of liquid medium (with antibiotic) with a freshly grown colony or 10 ml of freshly grown culture. Incubated at 37 °C until OD600 reached 0.4 - 0.6. Added IPTG to 0.4 mM. Induced 2 hours at 37°C or 15°C overnight.

IPTG induction

[0295] Cultures were grown for 4 hr total. IPTG was at 800 mM thus, added 25 μ l to each 50 ml culture flask. Incubated for 6 hr at 30 °C (so stop it not before 8h30). Pelleted bacterial cells by centrifugation at 5,000 $\times$ g for 10 minutes. (ultracentrifuge). Removed supernatant and stored at -80 °C

Procedure for Extracting Protein from Bacteria (B-PER protocol from Thermo)

[0296] Optional: Add 2 μ l of lysozyme and 2 μ l of DNase I per 1 ml of B-PER Reagent. Add EDTA-free protease inhibitors. Add 4 ml of B-PER Reagent per gram of cell pellet. 0.5g was obtained thus, add 2 ml. Pipetted the suspension up and down until it is homogeneous. Incubated 10-15 minutes at room temperature. Centrifuged lysate at 15,000 $\times$ g for 5 minutes to separate soluble proteins from the insoluble proteins. Note: If a large percentage of over-expressed protein remains in the pellet, the protein of interest might be expressed in inclusion bodies. Either use the Inclusion Body Solubilization Reagent (Product No. 78115) or alter the expression conditions to minimize inclusion body formation.

Purification of His-tag using Ni-NTA resin

[0297] For native conditions prepared the following buffers: Equilibration Buffer: 20 mM sodium phosphate, 300 mM sodium chloride (PBS) with 10 mM imidazole; pH 7.4; Wash Buffer: PBS with 25 mM imidazole; pH 7.4; Elution Buffer: PBS with 250 mM imidazole; pH 7.4.

Procedure for Purification of His-Tagged Proteins by Batch Method

[0298] The HisPur Ni-NTA Resin allows for purification strategy customization. Purification conditions can be scaled as needed. The procedure may be performed at room temperature or at 4°C. Added an appropriate amount of Ni-NTA resin to a tube. Centrifuged tube for 2 minutes at 700 $\times$ g and carefully remove and discard the

supernatant. Add two resin-bed volumes of Equilibration Buffer and mix until the resin is fully suspended. Centrifuged tube for 2 minutes at $700 \times g$ and carefully remove and discard buffer. Prepared sample by mixing protein extract with Equilibration Buffer so that the total volume equals two resin-bed volumes (instead added all component of equilibration buffer to sample so that sample is at 1x, then added 1 bed volume to resin, then added 650 μ l of resin slurry to sample).

[0299] Added the prepared protein extract to the tube and mix on an end-over-end rotator for 60 minutes. Centrifuged the tube for 2 minutes at $700 \times g$. If desired, save supernatant for downstream analysis. Washed the resin with two resin-bed volumes of Wash Buffer. Centrifuged the tube for 2 minutes at $700 \times g$. If desired, save supernatant for downstream analysis. Repeated wash step and monitored supernatant by measuring its absorbance at 280 nm until baseline was reached. Eluted bound His-tagged proteins using one resin-bed volume of Elution Buffer. Centrifuged tube for 2 minutes at $700 \times g$. Carefully removed and saved the supernatant. Repeated this step twice, saving each supernatant fraction in a separate tube. Monitored protein elution by measuring the absorbance of the fractions at 280 nm or by Coomassie Plus (Bradford) Assay Reagent (Product No. 23238) or Pierce® 660 nm Protein Assay (Product No. 22660). The eluted protein could be directly analyzed by SDS-PAGE. Note: To remove imidazole for downstream applications, used gel filtration (e.g., Thermo Scientific Zeba Spin Desalting Columns) or dialysis (e.g., Thermo Scientific Slide-A-Lyzer Dialysis Cassettes). Samples containing 6 M guanidine•HCl must be dialyzed against a buffer containing 8 M urea before SDS-PAGE analysis. The Thermo Scientific Pierce SDS-PAGE Sample Prep Kit (Product No. 89888) may also be used to remove guanidine.

[0300] Nanodrop phase A280

Sample ID	Protein Conc. Mg/ml
bind lysate	34.649
Wash 1	5.319

Wash 2	1.19
Wash 3	0.343
Elute 1	0.875
Elute 2	0.308

[0301] Vivaspeen clean up and concentrate. Tubes were centrifuged (15,000 g, 5 min) and supernatant was desalted by running on vivaspin 5K columns (15,000 g, 15 min), after each wash flow-thru was discarded and retentate was reconstituted to 0.5 mL with PBS, repeated 4 times. Average retentate volume after each wash was ~100 mL. Proteins were examined in Nanodrop. A280

[0302] Nanodrop phase A280

Sample ID	Protein Conc. Mg/ml
3	9.304
6	9.115
7	12.924
8	7.937
10	9.823
12	5.792

[0303] SDS-PAGE

	Sample ID	$\mu\text{g}/\mu\text{l}$	dilute 1/10 or 1/5	for 5 μg	to 6.5 h ₂ o
1	Protein ladder NEB				
2	bind lysate	34.65	3.46	1.44	5.06
3	Wash 1	5.32	-	0.94	5.56
4	Wash 2	1.19	-	4.20	2.30
5	Wash 3	0.34	-	14.58	-8.08
6	3	9.30	1.86	2.69	3.81
7	6	9.12	1.82	2.74	3.76
8	7	12.92	2.58	1.93	4.57
9	8	7.94	1.59	3.15	3.35
10	10	9.82	1.96	2.55	3.95
11	12	5.79	1.16	4.32	2.18
12	Inclusion				
13	Pro inhib				

[0304] MOPS, 200 volt 45 minutes (Figures 63, 64)SDS take 2

[0305] Ran first batch of protein on, also loaded sample protein clone 3 post miltenyi his-tag second purification. MOPS, 200 volts, 30 minutes (should have done 45 minutes) (Figure 95). Protein was in 0.5% BSA and 10% glycerol. Big stuff was BSA. 1, NEB ladder; 2, clone 6 Ni-NTA; 3, clone 2 post myltenyi; 4-11, Mi-NTA (+BSA)

Re-clean and concentrate DNA of clone 393 (the potential neutralizer)

[0306] Pulled 3 wells, AMPure purified, resuspend in 15 μl H₂O. Nanodrop:500ng/ μl

In vitro translate (thermo human in vitro expression kit)transcription

[0307] Will make 600 μ l of final protein kit (20x 25 μ kit), thus need 40 μ l of transcription reaction (s0 2 reaction worth).

	<u>1 tube</u>	<u>8 tubes</u>
• Nuclease-free Water	8	16
• 5X Transcription Buffer	4	8
• NTP Mix	4	8
• Cloned DNA (500 ng/ μ l)	2	4
• T7 RNA Polymerase	2	4
<u>Total</u>	<u>20</u>	<u>40</u>

[0308] Gently mixed tubes and incubated for 60-75 minutes at 32 °C.

Translation

[0309] Added the reagents in the order listed into a 1.5 or 0.5 ml RNase/DNase-free tube. For best results, incubated the Lysate with the Accessory Proteins for 5 minutes before adding subsequent components.

	<u>1 tube</u>	<u>20 tubes (in one 1.5 ml)</u>
• Lysate for Protein Expression	12.5	250
• Accessory Proteins	2.5	50
• Nuclease-free Water	3.75	75
• Salt Solution A	1	20
• Amino Acid minus Met	0.5	10
• Amino Acid minus Leu	0.5	10
• RNase Inhibitor (optional)	1	20
• Energy Mix	1.25	25
• Transcription Mix	2	40
<u>Total</u>	<u>25</u>	<u>500</u>

[0310] Split in 5 tubes of 100 μ l. Incubated at 30 °C on PCR for 4 hr, then 4 °C overnight

Purification test

[0311] Split in 5 tubes of 95 μ l: 1, lysate; 2, mylteniy (use 50 μ l of beads in 500 μ l of lysis buffer); 3, Invitrogen; 4, Cobalt Thermo; 5, Ni (Table 17).

#	description		ug/ml or ng/ μ l	A280	260/280	ug/ μ l	for 10 ug	dilution factor		volume for 10ug	water to 6.5	
1			in vitro lysate	46297	46.297	1.6	46.30	0.22	10	4.63	2.16	4.3
2	mylteniy	post bind	186235	186.235	0.56	186.24	0.05	50	3.72	2.68	3.82	
3		AB/Ag elution	882	0.882	1.11	0.88	11.34	1	0.88	11.34	0.00	
4		invitrogen elution	27	0.027	2.65	0.03	370.37	1	0.03	370.37	0.00	
5		mylteniy SDS elution	3390	3.39	0.71	3.39	2.95	1	3.39	2.95	3.55	
6	Invitrogen	post bind	34384	34.384	1.65	34.38	0.29	10	3.44	2.91	3.59	
7		invitrogen elution	837	0.837	1.79	0.84	11.95	1	0.84	11.95	0.00	
8	Cobalt thermo	post bind	22703	22.703	1.73	22.70	0.44	10	2.27	4.40	2.10	
9		elution 1	43	0.043	2.2	0.04	232.56	1	0.04	232.56	0.00	
10		elution 2	0	0	82.49	0.00	#DIV/0!	1	0.00	#DIV/0!	0.00	
11	Ni-Nta nugen	post bind	14513	14.513	1.66	14.51	0.69	5	2.90	3.45	3.05	
12		elution 1	833	0.833	1.4	0.83	12.00	1	0.83	12.00	0.00	
13		elution 2	417	0.417	1.64	0.42	23.98	1	0.42	23.98	0.00	
14	ladder											

Table 17.

[0312] SDS page. MES buffer, run 35 minutes, simply blue staining (Figure 65). The cloning may have put the His tag out of frame, but the protein is in frame.

Functional assay protocolsFACS assay preparation

[0313] 1. Half of the protein samples were labeled with FITC as follows: a. $\sim$ 20 μ L of the protein sample was transferred to a 0.2 mL tube, and the volume was reconstituted to 100 μ L with PBS (final concentration of protein was estimated to be 0.01 mg/mL); b. $\sim$ 1 mg of FITC (Pierce) was dissolved in 1.5 mL of DMF; c. 1 μ L of the FITC/DMF solution was

added to the sample, and the tubes were incubated in the dark at RT for 30 min; d. Proteins were cleaned from excess FITC by Vivaspin 5K columns and washed 2 times with 0.4 mL PBS; e. Sodium azide was added to a final concentration of 0.02% and the samples (~20 μ L) were stored at 4 °C until use.

[0314] 2. HA proteins (Bris59, Bris10 and Flor04, purchased from Sino Biologicals, China) were biotinylated using EZ-Link-NHS-Biotin (Pierce) by incubation on ice for 1 hour and then cleaning on Vivaspin 10K columns, with 2 washes with 0.4 mL PBS.

[0315] 3. The biotinylated HA proteins were used to coat Dynal streptavidin-coated magnetic beads by incubation in PBS (10 μ L beads, 5 μ L biotinylated protein, in 100 μ L PBS total volume), with flipping, at RT for 1 hour.

[0316] 4. Beads were washed 3 times with 100 μ L of bead assay buffer (0.1% BSA, 10% glycerol, 0.02% sodium azide in PBS, pH 7.4) using a magnet.

[0317] 5. Finished beads were stored in 100 μ L assay buffer at 4 °C until use.

FACS assay

[0318] 1. 1 μ L of bead suspension (vortexed shortly before use) was mixed with 5 μ L of antibody solution and 19 μ L of FX buffer (0.1% BSA, 0.02% sodium azide in HBSS, pH 7.4) 30 min on ice (final antibody concentration ~ 10 μ g/mL).

[0319] 2. Samples were reconstituted with 25 μ L cold FX buffer and analyzed by Accuri C6 flow cytometer.

QCM measurements of antibody/antigen binding

[0320] Crystals were coated with antibody a day prior to experiment as follows:

- [0321] 1. QCM gold crystals (purchased from Qsense) were coated with HS-PEG-NH₂ (from Nanocs) by incubating with 100 μ L of a 5 mg/mL solution of HS-PEG-NH₂ in ultrapure water in a humid chamber, 1 hour at RT.
- [0322] 2. Crystals were washed gently with 0.5 mL of PBS.
- [0323] 3. 30 μ L of antibody solution ($\sim$ 1.5 μ g total protein) was mixed with 70 μ L PBS containing 200 pmol of EDC and 200 pmol of sulfo-NHS (both from Pierce, sulfo-NHS is no weigh, mixed with 22 μ L of ultrapure water prior to mixing), incubated at RT for 1 minute, then applied onto the crystal surface and incubated at RT for 1 hour in a humid chamber.
- [0324] 4. 1 μ L of a 10 mg/mL solution of BSA was added and the liquid phase was mixed gently.
- [0325] 5. Humid chamber was moved to 4 $^{\circ}$ C for overnight storage without washing off antibody.

QCM measurements were performed on a Qsense E4 instrument as follows

- [0326] 1. PBS was flown through the cells at 0.1 mL/min until crystals were equilibrated ($\sim$ 30 minutes).
- [0327] 2. 2 mL of HA protein solutions (1 and 10 μ g/mL) were flown through the cells switching back to PBS after sample is finished.
- [0328] 3. Crystals were washed with 0.1 M NaOH in between different concentrations.

Purification of E. coli expressed nanobodies

- [0329] 1. Cell pellets (supernatant thoroughly removed, pellets frozen at -20 °C) were resuspended in 1 mL lysis buffer (10 mM Tris-HCl pH 7.5, 1 mM EDTA, 0.05% Triton X-100, 5 µg/mL DNase I, 1X protease inhibitor cocktail from Pierce)
- [0330] 2. Suspensions were transferred to conical 15 mL tubes and sonicated by a standard probe sonicator, while kept on ice. Sonication parameters: [10 sec on + 5 sec off] x 4 pulses, 100 % output.
- [0331] 3. Sonicates were centrifuged (4,800 g, 10 min, 4 °C) and supernatants were transferred to 1.5 mL tubes.
- [0332] 4. 4 mL (8 samples x 0.5 mL/sample) of Ni-NTA sepharose resin (Amersham) were washed in a 15 mL tube to remove storage buffer (contains 20% ethanol), by 3 repeated centrifugations (500 g, 1 min) and washes with 10 mL His-binding buffer (20 mM phosphate, 0.5 M NaCl, 20 mM imidazole, pH 7.4), after the last wash, resin pellet was reconstituted with 4 mL of 2X His binding buffer.
- [0333] 5. 0.5 mL of resin were added to each sample and the tubes were rotated at 4 °C overnight.
- [0334] 6. Resin was washed two times with 1.5 mL of His binding buffer (30 min each wash, centrifugation 1000 g, 1 min).
- [0335] 7. After the final wash, resin bed was left wet (~0.6 mL) and stored overnight at 4 °C.
- [0336] 8. 0.5 mL of His elution buffer (20 mM phosphate, 0.5 M NaCl, 0.5 M imidazole, pH 7.4) were added and the tubes were rotated overnight at 4 °C.

[0337] 9. Tubes were centrifuged (15,000 g, 5 min) and supernatant was desalted by running on vivaspin 5K columns (15,000 g, 15 min, after each wash flow-thru was discarded and retentate was reconstituted to 0.5 mL with PBS, repeated 4 times. Average retentate volume after each wash was ~100 μ L).

[0338] 10. Proteins were examined in Nanodrop.

[0339] 11. 50 μ L samples were taken for labeling.

Labeling of nanobodies

[0340] 1. 50 μ L (~35 μ g according to nanodrop) of each sample was mixed with 450 μ L of 50 mM borate buffer pH 8.5.

[0341] 2. Samples were cleaned on vivaspin 5K columns (15,000 g, 15 min, retentate ~20 μ L). Calculated protein concentration (average in samples) after this step was ~1.5 mg/mL.

[0342] 3. DyLight-649 (1 mg) was dissolved in 100 μ L DMF, and 1 μ L was added to each sample.

[0343] 4. Tubes were incubated in the dark for 1 h at RT.

[0344] 5. Labeled proteins were cleaned from excess dye by vivaspin 5K columns, with 3 washes with 470 μ L PBS and one wash into 470 μ L of 10 mM Tris buffer pH 7.5 to quench residual dye.

[0345] 6. Calculated concentration in retentate (~20-25 μ L each): 1.3-1.5 mg/mL.

Neutralization assayFlu strains - growth in A549 cells and immunotitration

- [0346] 1. The cells used were A549 human lung epithelial cells (Carcinoma). Cells were grown in 48-well plates in 0.4 mL medium (ATCC medium # F-12K + 10% FBS + Pen/Strep). The cells were transferred to the plate following plating of A549 from a T75 culture flask [confluency > aspirate medium > 1.3 mL of TrypLE or trypsin-EDTA > 3-5 min at 37 °C > completed to 6 mL with medium > take 50 µL per well in a 48-well plate containing 0.35-0.4 mL medium].
- [0347] 2. At 80% confluency, infected cells with virus, by adding 10 µL of virus at the desired dilution, mix well using pipette.
- [0348] 3. Incubated at 37 for 24-48 h.
- [0349] 4. Aspirated medium, apply 100 µL TrypLE, incubate 5 min at 37 °C.
- [0350] 5. Mixed well with pipette, take 200 µL of cells to a fresh u-shaped 96 well plate.
- [0351] 6. Fixed with formaldehyde, 2.5 % final conc. On ice for at least 10 min.
- [0352] 7. Washed (centrifuged 3000 rpm, 3 min, 4 °C), resuspended in 100 µL of freezing-cold 100% methanol. Mixed well.
- [0353] 8. Incubated on ice for 10 min.
- [0354] 9. Added 100 µL cold FX buffer, spun again as in 7. Resuspend in 100 µL cold FX buffer.

- [0355] 10. Added 1 μ L of FITC-conjugated anti-Influenza A nucleoprotein (from Millipore) per well, mixed well and incubated on ice for 60 min.
- [0356] 11. Spun as in 7, resuspended in 200 μ L cold FX and spun again.
- [0357] 12. Resuspended in 200 μ L cold FX, analyzed by FACS.
- [0358] 13. Determined titer as follows: $\text{Titer (FIU}/\mu\text{L}) = \text{D50} \times \text{C} / (2 \times \text{V})$. D50: The concentration of virus at which 50% of the cells are infected (FITC+) C, Starting number of cells in the well (refer to Worldwide Web Site: invitrogen.com/etc/medialib/en/filelibrary/pdf.Par.4786.File.dat/Useful_Numbers_Y1447_2_Useful_Nmbrs.pdf for useful details); V, Volume of virus (in μ L) used to infect the well

Neutralization assay

- [0359] 1. A549 cells at confluency 75% were incubated at 0.4 mL medium in 48 well plate. Nanobodies at various concentrations (50, 5, 0.5, 0.05 μ g/mL, diluted in A549 medium) were added.
- [0360] 2. Immediately afterwards the cells were infected with Bris10 strain at MOI of ~33% (0.33) at a volume of 10 μ L.
- [0361] 3. Cells were incubated for 24 h.
- [0362] 4. Medium was aspirated (and bleached). Cells were washed once with 0.5 mL HBSS.
- [0363] 5. 100 μ L TrypLE was added and the plate was incubated at 37 deg. for 5 min.

[0364] 6. Cells were aspirated into a 96 well plate and immediately fixed with formaldehyde as in step 6 in previous section.

[0365] 7. Protocol continued from step 6 in previous section.

[0366] Neutralization results, Figures 66 and 67.

EXAMPLE VII

Heavy and Light Chain Correlation

Initial light chain analysis

[0367] Light chain data was obtained from libraries made from a few time points of an experiment. Histograms of the read lengths are set forth at Figure 46. There was a peak at 380 (expected), and a smaller aggregation around 325. 10 reads from each peak were randomly aligned. 7/10 from the 380 peak were light chains (all kappa). None of the 325 peaks were light chains or immune chains. They were random proteins that must have amplified. The cutoffs for size selection was 365-405.

VDJ alignment/classification and clustering of light chain data

[0368] Light chain data was taken through an initial pipeline. The contents of the README file in are set forth in the Table 4 below. The work was performed on orchestra. Symlinks were placed in the stable-data directory to the results.

Running the full pipeline on orchestra: These scripts were run when the vdj repository is at tag v1.3 The intermediate data is in the corresponding directory on orchestra: /home/ul2/vdj-ome/analysis/20100810_light_chain_pipeline 1. fasta2vdjxml.py 2. size_select.py First I convert the fasta file to vdjxml, and, Size select the reads based on the readlen hist, 365-405, (from /home/ul2/vdj-ome/stable-


```

data/raw_light_chains): python ~/code/vdj/bin/fasta2vdjxml.py
light_chains.20100802.fasta | python ~/code/vdj/bin/size_select.py --min 365 --max 405 >
light_chains.20100802.size365-405.vdjxml      There are 496605 chains of the selected
size in the file. 3. vdjxml2parts.py          Split vdjxml into small chunks and place in
working directory      python ~/code/vdj/bin/vdjxml2parts.py --packetsize 10000 --
basename ~/vdj-ome/analysis/light_chain_pipeline/data/light_chains.20100802.size365-
405.vdjxml light_chains.20100802.size365-405.vdjxml  Change directory to all the
parts: cd ~/vdj-ome/analysis/light_chain_pipeline/data 4. barcode_id.py 5.
coding_strand.py      Identify barcodes for each read, and      Determine whether we have
the correct strand or not      for FILE in light_chains.20100802.size365-405.vdjxml.*;
do      NAME=${FILE%.size*} ".prealign.vdjxml." "${FILE#*.vdjxml.*}      bsub -
qshared_2h -o pre-alignment.log "python ~/code/vdj/bin/barcode_id.py --barcodes ~/vdj-
ome/stable-data/barcodes/454MID.barcodes.fasta $FILE | python
~/code/vdj/bin/coding_strand.py --locus IGK --locus IGL > $NAME"      done      #
python ~/code/vdj/bin/barcode_id.py --barcodes ~/vdj-ome/stable-
data/barcodes/454MID.barcodes.fasta      # python ~/code/vdj/bin/coding_strand.py --locus
IGK --locus IGL      Some STATS:      # Num of chains      cat *prealign* | grep
"<ImmuneChain>" | wc -l      496605      # Num with barcodes      cat *prealign* | grep
"<barcode>" | wc -l      486258      # Num that were reverse-complemented      cat
*prealign* | grep "revcomp" | wc -l      240928      # Barcode breakdown      for NUM
in 1 2 3 4 5 6 8 9; do      cat *prealign* | grep "<barcode>0$NUM" | wc -l      done
01 71      0.015%      02 0      03 174550      35.9%      04 7      0.001%
05 142036      29.2%      06 0      08 169594      34.9%      09 0 6. align_vdj.py
for FILE in light_chains.20100802.prealign.vdjxml.*; do
NAME=${FILE%.prealign*} ".vdjxml." "${FILE#*.vdjxml.*}      bsub -qshared_12h -o
alignment.log python ~/code/vdj/bin/align_vdj.py --locus IGK --locus IGL $FILE $NAME
done 7. cat_vdjxml.py      python ~/code/vdj/bin/cat_vdjxml.py
light_chains.20100802.vdjxml.* > light_chains.20100802.aligned.vdjxml      Move
processed data down one directory: mv light_chains.20100802.aligned.vdjxml .. cd ..
8. filter_VJ.py      python ~/code/vdj/bin/filter_VJ.py
light_chains.20100802.aligned.vdjxml light_chains.20100802.VJ_filtered.vdjxml
grep "<ImmuneChain>" light_chains.20100802.VJ_filtered.vdjxml | wc -l      356813
for NUM in 3 5 8; do      grep "<barcode>0$NUM"
light_chains.20100802.VJ_filtered.vdjxml | wc -l      done      03 131837      05
99618      08 117703 9. partition_VJ.py      mkdir partitions      python
~/code/vdj/bin/partition_VJ.py --basename partitions/light_chains.20100802
light_chains.20100802.VJ_filtered.vdjxml      cd partitions      # How many chains in
each partition?      for FILE in light_chains.20100802.*.vdjxml; do      grep
"<ImmuneChain>" $FILE | wc -l      done | sort -n      partial results:      ...
10021      11239      11285      12592      13451      13999      17935      10.
cluster_cdr3.py      for INFILE in light_chains.20100802.*.vdjxml; do
VJID=${INFILE#light_chains.20100802.}      VJID=${VJID%.vdjxml}
OUTFILE=${INFILE%.vdjxml}.clustered.vdjxml      bsub -qshared_unlimited -o

```



```

clustering.log python ~/code/vdj/bin/cluster_cdr3.py --cutoff 4.5 --tag $VJID --linkage
single $INFILE $OUTFILE done # How long did it take? grep "CPU"
clustering.log | sort -n -k4 partial results: CPU time : 25.38 sec. CPU
time : 29.70 sec. CPU time : 34.28 sec. CPU time : 42.63 sec.
CPU time : 44.29 sec. CPU time : 46.46 sec. CPU time : 48.57 sec.
CPU time : 57.70 sec. CPU time : 59.88 sec. 11. cat_vdxml.py
python ~/code/vdj/bin/cat_vdxml.py light_chains.20100802.*.clustered.vdxml >
../light_chains.20100802.clustered.vdxml cd .. # How many chains? grep
"<ImmuneChain>" light_chains.20100802.clustered.vdxml | wc -l 356813 #
How many unique clones in total? grep "<clone>"
light_chains.20100802.clustered.vdxml | sort | uniq | wc -l 5009 # How many
unique junctions in total? grep "<junction>" light_chains.20100802.clustered.vdxml |
sort | uniq | wc -l 53880

```

Table 4.

Time series and clone distribution of light chain data

[0369] Time series and clone frequency histograms were generated from the light chain data. There were only about 5000 clones. Without intending to be bound by scientific theory, it is possible that the cutoff value to define clones needs to be recalibrated for light chains. Without intending to be bound by scientific theory, lesser diversity may be expected since there is no D region. See Figures 47 and 48.

Light chain time series using unique junctions instead of clones

[0370] Because of a concern that the cluster definitions for heavy chains may not translate to light chains, time series figures were generated for both heavy and light chains using only unique junctions. A plot of the two is set forth at Figure 49 (NOTE: heavy chains on top, light chains on bottom).

[0371] The top junction sequences are set for in the Table 5 below.

heavy chain junctions: 1d

TGTGCGCAAACCGATAGCAGCAACTTAGACTACTACTACCACGGAATGGGCGTCTGG
 TGTGCGAAAGGGGAGATTGTGGTGGTGCTAGTTGCCCCCATTTAGACTACTACTACTACGGTATGGACGTCTGG
 TGC GCGAGAGGGACGGGGGACCCTACGGTGACTATTATGGTGGTGCTTTTGATGTCTGG
 TGTGCGCAAACCGATAGCACCAACTTAGACTTCTACTTCTACGGTTTGGACGACTGG
 TGTGCGAGATCGGGGGATAGTTGGAGTCCTCCACAATTTGACTTCTGG
 TGTGCGCAAACCGATAGCAGCCACATAGATTTCTACTACTATGGTATGGACGACTGG
 TGTGCGGTTCAAGATTGTAGTACTACCACCTGCTATCCTGCGAGTTCCTACTACTACTATAACATGGGCGTCTGG
 TGTGCGGTTCAAGATTGTAGTACTACCACCTGTTATCCTGCGAGTTCCTACTACTACTACAACATGGGCGTCTGG
 TGTGCGCAAACCGATAGCAGCCACATAGACTTCTACTACTACGGTATGGACGACTGG
 TGTGCGAGATCGGGGGATAGTTGGAGCCCTCCACAATTTGACTTCTGG 7d
 TGTGCGCAAACCGATAGCAGCAACTTAGACTACTACTACCACGGAATGGGCGTCTGG
 TGTGCGAGAGATGATCCATATTACGGCAGTATTGGTTATCGTATTGACTCCTGG
 TGTGCGAGAATCCCCGCTAAGATCGAGTGGGACGCCTACTACTACTACGGTATGGACGTCTGG
 TGTGCGAGAGACCTTACCTGGAGATACTTTGACTCCTGG
 TGTGCGAGAGTGTCTCTAGTAGTGGTTATTACTACTACTTTGATTACTGG
 TGTGCAAGATCCCTCATTCTATATAGTGACTACATTGCCTACTGG
 TGTGCGAGGCAGTCAGGTAACCGAGGATTCGGTGACTCTTACTCCTACTATTACTTCATGGACGTCTGG
 TGC GCGAGAACTTTGTATTCTCTGGTAAAGTATAGTACTGGCTGGTACTACTTTGACTACTGG
 TGTGCGAGAACAAATGCTTTTCATATCTGG
 TGTGTAAGAGTTAAGGGTGGCATAGCAGCAGCTGGTACCACTGCGGGGTACTTCGATCTCTGG 21d
 TGTGCGAGAGTACGGGGATATTGTAATGGTGGTAGCTGCTACTTTGACTACTGG
 TGTGCGAGAGAGAGGGCATTAGTGGGAGGTAGTACGACTCTCGGATACTGG
 TGTGCGAGAGGCAGGGCTTCAACCTTTAAAGTCTACTATCACTACATGGACGTCTGG
 TGTGCGAGACATATGCGGGGTGGGAGCCCTAGTCAAACCTGCTTTTGATGTCTGG
 TGTGCGAGACATATGCGGGCTGGGAGTCCTAGTCAAACCTGCTTTTGATGTCTGG
 TGTGCGAGACATATGCGGGGTGGGAGTCCTAGTCAAACCTGCTTTTGATGTCTGG
 TGTGTGAAAGCGGTTTCGGGGTCGAACTACATCTTTGACTACTGG
 TGTGCGAGAGGCCGAGTGGGAGCTAAAGAGCCGACCGTTTACTACTTTGACCACTGG
 TGTGCGAAAATTTTTGAGGCGAATTTGGAAAACCTACTGGTATGGTTTGGACGTCTGG
 TGTGCGAGACATCAGTATAATGTTGGTAACTCCTGGGCTTTTGATATCTGG light chain junctions: 1d
 TGTCAACAGGCTAACAGTTTCCCGCTCACTTTC TGTCAATCACCAGACAGCAGTGGTACTAATGAAGTCTTC
 TGTCAATCACCAGACAGCAGTGGTACTTATGAAGTCTTC
 TGCGGAACATGGGATAACCAGCCTGGGTACTAATTGGGTGTTC
 TGCGGAACATGGGATAACCAGCCTGAGAATTAATTGGGTGTTC TGTCAACAGTTTACTAATTTCCCGCTCACTTTC
 TGTCAACAGTATGGTAGTTCATGGCGCACTTTT TGTCAAGCGTGGGACAGCAGCACTGTGGTATTC
 TGCGGAACATGGGATAACCAGCCTGAGGATTAATTGGGTGTTC
 TGCTCAGCATGGGACAGCAGCCTCAGTGCTTGGGTGTTC 7d TGTCAAGCGTGGGACAGCAGCACTGTGGTATTC
 TGCTGCTCATATGCAGGCAACTCTTATGTCTTC TGTCAATCACCAGACAGCAGTGGTACTAATGAAGTCTTC
 TGTCAAGCAGTATGGTAGCTCACCTTGGACGTTC TGTCAAGCAATATGGTAGCTCACCGACGTTC
 TGTAACCTCCCGGGACAGCAGTGGTAACCATGTGGTATTC TGTCAAGCAGCGTAGCACCTGGCCTGCGACTTTC
 TGTCAAGGTGTGGGATAGTAGTAGTGATCATGTGGTATTC


```

TGTC AATCAGCAGACAGCAGTGGTACTTATGTGGTATTC
TGCGGAGCGTGGGATAGCAGCCTGAGTGCTGTGGTCTTC 21d TGTCAGGCGTGGGACAGCAGCACTGTGGTATTC
TGTC AACAACATGGTAACTCACCGGTCACTTTC TGCTTCCTCTACTTTGATGGTCCTATAGTTTTC
TGTCAGCATTATCATACTCCACCGTACACTTTT TGCCAACAGTATAATACCTGGTGGACATTC
TGTTTACTCTATTATAATGGTGTGAGGGTGTTT TGTAAC TCCCGGGACAGCAGTGGTAACCATGTGGTATTC
TGCTGCTCATATGCAGGTAGTAGCACTTTTGTCTTC TGTC AATCAGCAGACAGCAGTGGTACTTATGTGGTATTC
TGCTCAGCATGGGACAGCAGCCTCAGTGCTTGGGTGTTT

```

Table 5.

[0372] Sequences corresponding to highest-expressed junctions from light chain and heavy chain time series are set forth in the Table 6 below. The script was pull_sequences.py and the output was manually put in sequences_highly_expressed_junctions.fasta.

```

>heavy_1d|FWMETYS01DQGF4|2
TTGTGGCTATTTTAAAAGGTGTCCAGTGTGACGTGCAGCTGGTGGAGTCTGGGGGAGGCTTGGTACAGCCTGGGAAGT
CCCTGAGACTCTCCTGTGCAGCCTCTGGATTACCTTTAGCAACTATGCCATGTCCTGGGTCCGCCAGGCTCCAGGGA
AGGGCCTGGAGTGGGTCTCAGGTATTAGTGGTAGTGGTGCTGCCACATACTACGCAGACTCCGTGAAGGGCCGGTTCA
CCATCTCCAGAGTCAATTCCAGGAACACGCTCCTTCTGCAAATGAACAGCCTGAGAGCCGAGGACACGGCCCTATATT
ACTGTGCGCAAACCGATAGCAGCAACTTAGACTACTACTACCACGGAATGGGCGTCTGGGGCCAAGGGACCACGGTCA
CCGTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_1d|FWMETYS01A182M|2
TTGTGGCTATTTTAAAAGGTGTCCAGTGTGAGGTGCAGCTCTTGGAGTCTGGGGGAGGCTTGGCACAGCCTGGGGGGT
CCCTGAGACTCTCCTGTGGAGCCTCTGGATTACCTTTAGTAACTATGGCATGACCTGGGTCCGCCAGGCTCCAGGGA
AGGGGCTGGAGTGGGTCTCAGGTATTAGTGGTAGTGGTGCTAGTACATTCTACGCAGACTCCGTGAAGGGCCGGTTCA
CCATCTCCAGAGACAATTCCAAGAACACGCTGTATCTGCAAATGGACAGCCTGAGAGCCGAGGACACGGCCGTCTATT
ACTGTGCGAAAGGGGGAGATTGTGGTGGTGCTAGTTGCCCCATTTAGACTACTACTACTACGGTATGGACGTCTGGG
GCCAAGGGACCACGGTCACCGTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_1d|FWMETYS01ESRCQ|2
CGGCTCCCAGATGGGTCTGTCCCAGCTGCAGTTGCAGGAGTCGGGGCCAGGAGTGGTGAAGCCTTCGGAGACCCTGT
CCCTCACCTGCACTGTCTCTGGTGGCTCCATCAGCACTAGCAGTTATTACTGGGGCTGGATCCGCCAGCCCCCAGGGA
AGGGGCTGGAGTGGATTGGGACTATCTATTCTAGTGGGGGCTCCTACTACAACCCGTCCCTCAAGAGTCAAGTCGCCA
TATCCGTTGACATGTCCAAGAATCAATTCTCCCTGAAGGTGAACTCTATAATCGCCGCAGACACGGCTGTGTATTACT
GCGCGAGAGGGACGGGGGGACCCTACGGTGACTATTATGGTGGTGCTTTTGTATGTCTGGGGCCAAGGGAAAAGGGTCG
CCGTCTCTTCAGCATCCCCGACCAGCCCCAAGG >heavy_1d|FWMETYS01ALGIE|2
TTGTGGCTATTTTAAAAGGTGTCCAGTGTGAGGTGCGGCTGTTGGAGTCTGGGGGAGGCTTAGTGCAGCCTGGGGGGT
CCCTGAGACTCTCCTGTGCAGCCTCTGGATTACCTTTAGCAACTATGCCATGACCTGGGTCCGCCAGGCTCCAGGGC
AGGGGCTGGAGTGGGTCTCAGGAATTAGTGGTAGTGGTGCAAGCACATACTACGAAGACTCCGTGAAGGGCCGGTTCA
CCATCTCCAGACAGAATTCCAAGAACACGCTGTATCTGCAAATGAATAGCCTGAGAGCCGAGGACACGGCCGTATACT
ACTGTGCGCAAACCGATAGCACCAACTTAGACTTCTACTTCTACGGTTTGGACGACTGGGGCCAAGGGACCACGGTCA
CCGTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_1d|FWMETYS01EMQ99|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTGCAGGAGTCGGGGCCAGGACTGGTGAAGCCTTCGGAGACCCTGT
CCCTCACCTGCAAAGTTTCTATTGGCTCCGTGAGGAATTATTACTGGAGCTGGATCCGGCAGTCCCCCGGGAAGGGAC
TGGAGTGGATTGCATATATCTTTCCCAATGGGAGGACCAGCCGCAATCCCTCCCTCCAGAGTCGAGTCACCATATCAA

```


TTGACACACCCAAAAATCAGTTCTCCATGTTGCTGAGCTCTGCGACCGCCGCAGACACGGCCGTCTATTACTGTGCGA
GATCGGGGGATAGTTGGAGTCCTCCACAATTTGACTTCTGGGGCCAGGGAATCCTAGTCACCGTCTCCTCAGCATCCC
CGACCAGCCCCAAGG >heavy_1d|FWMETYS01A36YN|2
TTGTGGCTATTTTAAAAGGTGTCCAGTGTGAGGTGCAGCTGTTGGAGTCTGGGGGAGGCTTGGTGCAGCCTGGGGGGT
CCCTGAGACTCTCCTGTGCAGCCTCTGGATTACGTTTCAGCAACTATGCCATGAGCTGGGTCCGCCAGGCTCCAGGGA
AGGGGCTGGAGTGGGTCTCAGGAATGAGTGGTAGTGGTGCTAGCACATACTACGAAGACTCCGTGAAGGGCCGGTTCA
CCATCTCCAGAGACAATTCCAAGAACACGCTGTATCTGCAAATGAATAGCCTGAGAGCCGAGGACACGGCCGTATATT
ACTGTGCGCAAACCGATAGCAGCCACATAGATTTCTACTACTATGGTATGGACGACTGGGGCCAAGGGACCACGGTCA
CCGTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_1d|FWMETYS01A36AI|0
GCAGCAGCTACAGGTGTCCAGTCCCAGGTGCAGTTGGTGCAGTCTGGGGCTGAGGTGAAGAGGCCTGGGTCTCGGTG
AAGGTCTCCTGCAAGGCTTCTGGAGGCACCTTCAGCAACTATGCAATCAGCTGGGTGCGACAGGCCCCCTGGACAAGGG
CTTGAGTGGATGGGAGGGATCATCCCTACTTTTGGGACACCGACGTACGCACAGAAGTTCAGGCCAGAGTCACGATT
ACCGCGGACGAATCTACGAGCACAGCCTACATGGAGCTGAGCAGCCTGAGATCTGAGGACACGGCCGTCTATTGGTGT
GCGGTTCAAGATTGTAGTACTACCACCTGCTATCCTGCGAGTTCCTACTACTATAACATGGGCGTCTGGGGCAAA
GGGACCACGGTCACCGTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_1d|FWMETYS01BAHLT|2
CAGCAGCTACAGGTGTCCAGTCCCAGGTGCAGCTGGTGCAGTCTGGGGCTGAGGTGAAGAGGCCTGGGTCTCGGTGA
AGGTCTCCTGCCAGGCTTCTGGAGGCACCTTCAGCAACTATGCTATCAGCTGGGTGCGACAGGCCCCCTGGACAAGGGC
TTGAGTGGATGGGAGGGATCATCCCTATGTTTGGTACACCAAAGTACGCACAGCAGTTCTTGGACAGAGTCACGATAA
CCGCGGACGAATCCACGAGTACAGCCTACATGGAGCTGAGCAGCCTGAGATCTGCGGACATGGCCGTTTATTGGTGTG
CGGTTCAAGATTGTAGTACTACCACCTGTTATCCTGCGAGTTCCTACTACTACTACAACATGGGCGTCTGGGGCAAAG
GGACCACGGTCACCGTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_1d|FWMETYS01COGNO|0
GTGGCTATTTTAAAAGGTGTCCAGTGTGAGGTGCAGCTGTTGGAGTCTGGGGGAGGCTTGGTGCAGCCTGGGGGGTCC
CTGAGACTCTCCTGTGTAGCCTCTGGATTACCTTTAGCAACTATGCCATGAGCTGGGTCCGCCAGGCTCCAGGGAAG
GGGCTGGAGTGGGTCTCAGGAATTAGTGGTAGTGGTGCTAGCACATACTACGAAGACTCCGTGAAGGGCCGGTTCACC
ATCTCCAGAGACAATTCCAAGAACACGCTGTATCTGGACATGAATAGCCTGAGAGCCGAGGACACGGCCGTATATTAC
TGTGCGCAAACCGATAGCAGCCACATAGACTTCTACTACTACGGTATGGACGACTGGGGCCAAGGGACCACGGTCACC
GTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_1d|FWMETYS01B1GQ7|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTGCAGGAGTCGGGCCCAGGACTGGTGAAGCCTTCGGAGACCCTGT
CCCTCACTTGCACTGTTTCTAGTGGCTCCATCAGGAATTACTACTGGAGCTGGATCCGGCAGACCCCAGGGAAGGGAC
TGGAGTGGATTGGATATATCTTTTCCAATGGGAGGATCAAGTACAATTCCTCCCTCCAAGGTCGACTCACCATGTCAC
TAAACACGCCCCGAGAATCAGTTCTCCCTGTGGCTGAGCTCTGTGACCGCCGCAGACACGGCCGTCTATTACTGTGCGA
GATCGGGGGATAGTTGGAGCCCTCCACAATTTGACTTCTGGGGCCAGGGAATCCTGGTCACCGTCTCCTCAGCATCCC
CGACCAGCCCCAAGG >heavy_21d|FWMETYS01ENDE7|2
TTGTGGCTATTTTAAAAGGTGTCCAGTGTGAGGTGCAGTTGTTGGAGTCTGGGGGAGGCTTGGTACAGCCTGGGGGGT
CCCTGAGACTCTCCTGTGCAGCCTCTGGATTACCTTTAACAACCTATGCCACAAGCTGGGTCCGCCAGGCTCCAGGGA
AGGGGCTGGAGTGGGTCTCAACTATTAGTGGTGTGGTGATACCACATACTACGCAAATTCGTGAAGGGCCGGTTCA
CCATCTCCAGAGACACTTCCAAGAACACCCTGTATCTGCAAATGAACAGCCTGAGAGCCGAGGACACGGCCGAATATT
ACTGTGCGAGAGTACGGGGATATTGTAATGGTGGTAGCTGCTACTTTGACTACTGGGGCCAGGGAACCCCGGTACCG
TCTCCTCAGGGAGTGCATCCGCCCC >heavy_21d|FWMETYS01BI6DS|2
CAGCAGCCACAGGAGCCCACTCCCAGGTGCAACTGGTGCAGTCTGGGGCTGAGGTGAAGAGGCCTGGGGCCTCAGTGA
AGGTCTCCTGCAAGGCCTCTGGATACACCTTCACCGGCTACTATGTACACTGGGTGCGACAGGCCCCCTGGACAAGGTC
TTGAGTGGATGGGATGGATCAACCCTAACAGTGGTGTCAAACTACGCACAGAAGTTTCAGGACAGGGTCACCATGA
CCAGGGACACGTCCATCAGCACAGCCTACATGGAGCTGACCAGGCTGAGATCTGACGACACGGCCCTATATTACTGTG
CGAGAGAGAGGGCATTAGTGGGAGGTAGTACGACTCTCGGATACTGGGGCCAGGGAACCCCTGGTCACCGTCTCCTCAG
GGAGTGCATCCGCCCCAACCC >heavy_21d|FWMETYS01ASJFG|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGTTACAGCAGTGGGGCGCAGGACTGTTGAAGCCTTCGGAGACCCTGT

CCCTCACCTGCGTTGTCTATGGTGGGTCTTTCAGTCCTTATTACTGGAGCTGGATCCGCCAGACCCCAGGGAAGGGGC
TGGAGTGGATTGGGGAAATCAATCATAGTGGAAGCACCAACTACAACCCGTCCCTCAAGAGTCGAGTCTCCATATCAC
TAGACACGTCCAAGAATGAGTTCTCCCTGAGGCTGAACTCTCTGACCGCCGCGGACACGGCTGTGTATTACTGTGCGA
GAGGCAGGGCTTCAACCTTTAAAGTCTACTATCACTACATGGACGTCTGGGGCAATGGGACCACGGTCACCGTCTCCT
CAGCATCCCCGACCAGCCCCAAGG >heavy_21d|FWMETYS01EEVTL|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTGCAGGAGTCGGGCCCAGGACTGGTGAAGCCTGCGGAGACCCTGT
CCCTCACCTGCAATGTCTCTGGTGGCTCCATGAGTAGTTACTACTGGAGCTGGATCCGGCAGCCCCCAGGGAAGGGAC
TGGAGTGGATCGGGTACATCCACTACAGGGGGACCACCAAATACAATCCCTCCCTCAAGAGTCGCGTCACCATATCAA
TAGACCTGTCCAAGAACCAGTTCTCCCTGAACTGAGCTCTATGACCGCCGCGAGATACGGCCAGATATTACTGTGCGA
GACATATGCGGGGTGGGAGCCCTAGTCAAACCTGCTTTTGATGTCTGGGGCCAAGGGACAATGGTCACCGTCTCTTCAG
CATCCCCGACCAGCCCCAAGG >heavy_21d|FWMETYS01BGHT7|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTGCAGGAGTCGGGCCCAGGACTGGTGAAGCCTTCGGAGACCCTGT
CCCTCACCTGCAATGTCTCTGGTGGCTCCATCAGTAGTTACTACTGGAGCTGGATCCGGCAGCCCCCAGGGAAGGCAC
TGGAGTGGATCGGGTATATCCACTACAGGGGGGACTACCAAATACAATCCCTCCCTCAAGAGTCGCGTCACCATATCAG
TAGACATGTCCAAGAACCAGTTCTCCCTGAAGCTGAGCTCTATGACCGCCGCGAGATACGGCCATCTATTACTGTGCGA
GACATATGCGGGGTGGGAGTCCTAGTCAAACCTGCTTTTGATGTCTGGGGCCAAGGGACAATGGTCACCGTCTCTTCAG
CATCCCCGACCAGCCCCAAGG >heavy_21d|FWMETYS01DUYF3|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTGCAGGAGTCGGGCCCAGGACTGGTGAAGCCTTCGGGGACCCTGT
CCCTCACCTGCAATGTCTCTGGTGGCCCCATCAGTAGTTACTACTGGAGCTGGATCCGGCAGCCCCCAGGGAAGGGAC
TGGAGTGGATCGGCTATATCTATTACAGGGGGGACTACCAAATACAATCCCTCCCTCAAGAGTCGCGTCACCATATCAG
TAGACATGTCCAAGAACCAGTTCTCCCTGAACCTGAGCTCTATGACCGCCGCGAGATACGGCCATGTACTACTGTGCGA
GACATATGCGGGGTGGGAGTCCTAGTCAAACCTGCTTTTGATGTCTGGGGCCAAGGGACAATGGTCACCGTCTCTTCAG
CATCCCCGACCAGCCCCAAGG >heavy_21d|FWMETYS01DI6U3|2
TTGTGGCTATTTTAAAAGGTGTCCAGTGTGAGGTGCAGCTGTTGGAGTCTGGGGGAGGCTTGATACAGCCTGGGGGGT
CCCTGAGACTCTCCTGTGCAGCCTCTGGATTACCTTTAGCAAGTATGCCATGACCTGGGTCCGCCAGGCTCCAGGGA
AGGGCCTGGAGTGGGTCTCGACTATTAGTGGTAGTGCTACTGCCACATACTACGCAGACTCCGTGAAGGGCCGCTTCA
CCATCTCCAGAGACAATTCCAAGAACACGCTGTATCTGCAAATGAACAGCCTGAGAGCCGCGGACACGGCCGTTTATT
ACTGTGTGAAAGCGGTTTCGGGGTCGAACTACATCTTTGACTACTGGGGCCAGGGAACCCAGGTACCGTCTCCTCAG
CATCCCCGACCAGCCCCAAGG >heavy_21d|FWMETYS01D3L49|2
TTGTGGCTATTTTAAAAGGTGTCCAATGTGAGGTGCAACTGTTAGAATATGGGGGAGGCTTGGTACAGCCGGGGGGGT
CCCTGAGACTCTCCTGTGAAGCCTCTGGAATCCCCTTTAAACAACTATGCCATGAGCTGGGTCCGCCAGGCTCCAGGGA
AGGGGCTGGAGTGGGTCTCAAGTATCAGTGGTAGTGGTAGTGGCACATATTACGGAGACTCCGTGAAGGGCCGTTTCA
CCATCTCCAGAGACAATTCCAAGAACACACTGTATCTGCAAATGAATAGCCTGAGAGCCGAGGACACGGCCGTATATT
ATTGTGCGAGAGGCCGAGTGGGAGCTAAAGAGCCGACCGTTTACTACTTTGACCACTGGGGCCAGGGAACCCCTGGTCA
CCGTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_21d|FWMETYS01EOI8H|0
GTGGCTATTTTAAAAGGTGTCCAGTGTGAGGTGCAGCTGTTGGAGTCTGGGGGAGGCTTGGTACAGCCTGGGGGGTCC
TTGAGACTCTCCTGTGCAGCCTCTGGATTACCTTCAGCAGCTATGCCATGAACTGGGTCCGCCTGCCTCCAGGGATG
GGGATGGAGAGCATCTCATCCATTAGTCGTAGTGGTGATAGAACATACTACGCAGACTCCGTGAAGGGCCGTTTACC
ATCTCCAGAGACAATTCCAAGAACACGATGTATCTGGAAATGAACAGCCTGAGAGCCGAAGACACGGCCGTATATTAC
TGTGCGAAAATTTTGGAGGCGAATTTGGAAAACCTACTGGTATGGTTTGGACGTCTGGGGCCAAGGGACCACGGTCACC
GTCTCCTCAGGGAGTGCATCCGCCCCAACCC >heavy_21d|FWMETYS01BIG98|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTGCAGGAGTCGGGCCCAGGACTGGTGAAGCCTTCGGAGACCCTGT
CCCTCACCTGCGCTGTCTCTGGTGCCTCCATTAGAAGTTACTATTGGAGCTGGATCCGGCTGCCCCCAGGGAAGGGAC
TGGAGTGGATTGGGCATGTGTATCACAGTGGGAGCACCAGTTACAATCCCTCCCTCAAGAGTCGAGTCACCATATCAG
TGGACACGTCCAAGATGCAGATCTCCCTGAGGCTGAACTCTGCGACTGCTGCGGACACGGCCGTGTATTACTGTGCGA
GACATCAGTATAATGTTGGTAACTCCTGGGCTTTTGATATCTGGGGCCAAGGGACAGTGGTCACCGTCTCTTCAGCAT

CCCCGACCAGCCCCAAGG >heavy_7d|FWMETYS01DQGF4|2
TTGTGGCTATTTTAAAAGGTGTCCAGTGTGACGTGCAGCTGGTGGAGTCTGGGGGAGGCTTGGTACAGCCTGGGAAGT
CCCTGAGACTCTCCTGTGCAGCCTCTGGATTACCTTTAGCAACTATGCCATGTCCTGGGTCCGCCAGGCTCCAGGGA
AGGGCCTGGAGTGGGTCTCAGGTATTAGTGGTAGTGGTGCTGCCACATACTACGCAGACTCCGTGAAGGGCCGGTTCA
CCATCTCCAGAGTCAATTCCAGGAACACGCTCCTTCTGCAAATGAACAGCCTGAGAGCCGAGGACACGGCCCTATATT
ACTGTGCGCAAACCGATAGCAGCAACTTAGACTACTACTACCACGGAATGGGCGTCTGGGGCCAAGGGACCACGGTCA
CCGTCTCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_7d|FWMETYS01EO8GA|2
CAGCAGCAACAGGTGCCCACTCCCAGGTTCAACTGGTGCATCTGGAGCTGAGGTGAAGAGGCCTGGGGCCTCAGTAA
AGGTCTCCTGCACGGCTTCTGGTTACACGTTTGACACTTATGGAGTCAGCTGGTTGCGACAGGCCCCCTGGACAAGGGC
TTGAGTGGATGGGCTGGATCAGCGGTGACAGTAGTCATACCAGATATGCAATGAGACTCCAGGGCAGAGTCACCATGA
CCACAGACTCATCCACGAGCACAGCCTACATGGAAGTGGAGAGCCTGAGATCTGACGACACGGCCGTCTATTACTGTG
CGAGAGATGATCCATATTACGGCAGTATTGGTTATCGTATTGACTCCTGGGGCCAGGGAACCCTGGTCACCGTCTCCT
CAGCATCCCCGACCAGCCCCAAGG >heavy_7d|FWMETYS02GPFII|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAACTGCAGGAGTCGGGGCCAGGACTGGTGAAGCCTTCGGAGACCCTGT
CCCTCACCTGCACTGTCTCTGGTGGCTCCATCAGTAGTTACTACTTGAGCTGGATCCGGCAGCCCCCAGGGAAGGGGC
TGGAGTGGATTGGGTATATCTATTACAGTGGGAGCACCAACTACAACCCCTCCCTCAAGAGTCGAGTCACCATATCAG
TAGACACGTCCAAGAACCAGTTCTCCCTGAGGCTGAGCTCTGTGACCGCTGCGGACACGGCCGTGTATTACTGTGCGA
GAATCCCCGCTAAGATCGAGTGGGACGCCTACTACTACTACGGTATGGACGTCTGGGGCCAAGGGACCACGGTCACCG
TCTCCTCAGGGAGTGCATCCGCCCC >heavy_7d|FWMETYS01D3CAE|2
TTGTTGCTATTTTAAAGGTGTCCAGTGTGAGGTGCAGTTGGTGGAGTCTGGGGGAGGCTTGGTACAGCCTGGGGGGT
CCCTGAGACTCTCCTGTGCAGCCTCTGGATTCACTTTAGTGGCTATAGCATGGACTGGGTCCGCCAGGCTCCGGGGA
AGGGGCTGGAGTGGGTTCATTTCATAAGTGATAATACTGGCAGTCACATATACTACGCAGACTCTGTGAAGGGCCGAT
TCACCATCTCTAGAGACAATGCCGAGAACTCACTGTATCTACAAATGAACAGCCTGAGAGACGAGGACACGGCTGTGT
ATTACTGTGCGAGAGACCTTACCTGGAGATACTTTGACTCCTGGGGCCATGGAGTCCTGGTCACCGTCTCCTCACGGA
GTGCATCCGCCCC >heavy_7d|FWMETYS01ASETU|2
TTGTTGCTATTTTAAAAGGTGTCCAGTGTGAGGTGCAGCTGGTGGAGTCTGGAGGAGGCCTGATCCAGCCTGGGGGGT
CCCTGAGACTCTCCTGTGCAGCCTCTGGGTTACCGTCAGTAGCAGCTACATGACCTGGGTCCGCCAGGCTCCAGGGA
AGGGGCTGGAGTGGGTCTCAGTTATTTATAGTGGTGGTACCACATACTACGCAGACTCCGTGAAGGGCCGATTACCA
TCTCCAGAGACAATTCCAAGAACACGCTGTATCTTCAAATGAACAGCCTGAGAGCCGAGGACACGGCCGTGTATTACT
GTGCGAGAGTGTTCTCTAGTAGTGGTTATTACTACTACTTTGATTACTGGGGCCAGGGAACCCTGGTCACCGTCTCCT
CAGGGAGTGCATCCGCCCC >heavy_7d|FWMETYS01EL10A|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTGCAGGAGTCGGGGCCAGGACTGGTGAAGCCTTCGGAGACCCTGT
CCCTCACCTGCACTGTCTCTGGTGGCTCCATTAGTAGTTACTCCTGGAGCTGGATCCGGCAGTCCCCAGGGAAGGGAC
TGGAGTGGATTGGGTCTACCTATTACAGTGGGAGCACCAACTACACCAACTACAACCCCTCCCTCAAGAGTCGAGTCA
CCACATCAGTAGACACGTCCAAGAACCAGTTGTCCCTGGGGCTGAACTCTGTGACCGCAGCGGACACGGCCATTTATT
ACTGTGCAAGATCCCTCATTCTATATAGTACTACATTGCCTACTGGGGCCAGGGAACCCTGGTCACCGTCTCCTCAG
CATCCCCGACCAGCCCCAAGG >heavy_7d|FWMETYS01CU1FY|2
CAGCAGCAACAGGTGCCCGCTCCCAGGTTCAACTGATGCAGTCTGGAGCTGAAGTGAGGAAGCCTGGGGCCTCAGTGA
CGGTCTCCTGCAAGACTTCTGGTTACACCTTTACCTACTATGGTATCAGTTGGGTGCGACAGGCCCCCTGGACAAGGCC
TTGAGTGGATGGGATGGTTCAGCGCTTACAATGGTAAGACAAAATATGCACAGAATCTCCAGGACAGAGTCACCATGA
CAATTGACACATCCACGAGGACAGCCTACATGGAGCTGAGGAGCCTGAGATCTGACGACACGGCCGTATATTACTGTG
CGAGGCAGTCAGGTAACCGAGGATTCGGTGACTCTTACTCCTACTATTACTTCATGGACGTCTGGGGCAAAGGGACCA
CGGTCACCGTCGCCTCAGCATCCCCGACCAGCCCCAAGG >heavy_7d|FWMETYS01DFT1H|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTACAGCAGTGGGGCGCAGGACTGTTGAAGCCTTCGGAGACCCTGT
CCCTCACCTGCGCTGTTTATGGTGAGTCCTTCCGTGGTTACTCCTGGAGCTGGATCCGGCAGCCCCCAGGGAAGGGGC
TGGAGTGGATTGGTGAAATCAATCTTACTGGAAGCACCAACTACAACCCGTCCCTCAAGAGTCGGATCACCGTATCAA

TTGACCCGTCCAAGACTCAGTTCTCCCTGAAGCTGACCTCTGTGACCGCCGCGGACACGGCTGTATATTACTGCGCGA
GAACTTTGTATTCTCTGGTAAAGTATAGTACTGGCTGGTACTACTTTGACTACTGGGGCCAGGGAATCCTGGTCACCG
TCTCCTCAGCCTCCACCAAGGGCCCATCCG >heavy_7d|FWMETYS01DE08R|2
CGGCTCCCAGATGGGTCTGTCCCAGGTGCAGCTGCAGGAGTCGGGCCCAGGACTGGTGAAGCCTTCGGAGACTCTGT
CCCTCACCTGCACTGTCTCTGGTGACTCCATTAGTAGTGACTATTGGAGCTGGATCCGGCAGTCCCCAGGGAAGGGAC
TGGAGTGGATTGGGTTTATTTCAGTATACTGGGAGATCCCACTCCAACCCCTCCCTCCAGAGTCGAGTCACCATATCAC
TAGACACGTCCAAGAACAACCTTCTCCCTGAGGCTGACCTCTGTGACCGCTGCGGACACGGCCGTGTATTATTGTGCGA
GAACAAATGCTTTTTCATATCTGGGGCCAAGGGACAATGGTCACCGTCTCTTCAGGGAGTGCATCCGCCCC
>heavy_7d|FWMETYS01BLJ3E|0
GTTGCTATATTAGAAGGTGTCCAGTGTGAGGTGCAGCTGGTGAATCTGGGGGAGGCTTAGTACAACCTGGGGGTTCG
CTGAGACTCTCCTGTGCAGCCTCTGGATTCACTTTCAGTGATTACGACATGCACTGGGTCCGCCAAGCTACAGGAAAA
GGTCTGGAGTGGGTCTCAGCTATAGGAACTCTTGATGACACATACTATCCAGGCTCCGTGGAGGGCCGATTACCGTCT
TCCAGAGACAATGCCAGGGATTCTTGTATCTTCAAATGAAGAGCCTCAGAGTCGCGGACACGGCTGTATATTACTGT
GTAAGAGTTAAGGGTGGCATAGCAGCAGCTGGTACCACTGCGGGGTACTTCGATCTCTGGGGCCGTGGCACCCTGGTC
ACTGTCTCCTCAGGGAGTGCATCCGCCCCAACCC >light_1d|012171_0240_0412|2
TCTGGTTCCCAGGTTCCAGATGCGACATCCAGATGACCCAGTCTCCGTCTTCCGTGTCTGCATCTGTGGGGGACAGAG
TCACCATCACTTGCCGGGCGAGTCAGAGTCTTAGCGGCTTTTTAGCCTGGTATCAGCAGAAACCAGGGAAAGCCCCTA
AGTTCCTGATCGATACTACCTCCATTTTGCAAAGTGGGGTCCCATCTAGATTTCAGTGGCAGTGGATCTGGGACATTTT
TCACTCTCACCATCAGCAGCCTCCAGCCTGAAGATTTTGCAACTTACTATTGTCAACAGGCTAACAGTTTCCCGCTCA
CTTTCGGCGGAGGGACCAAGGTGGAGAGGAAACGAACCTGTGGCTGCACCATCTG
>light_1d|016824_0202_0719|2
CTCACTGCACAGGCTCTGAGGCCTCCTATGAGCTGACACAGCCACCCTCGGTGTCTAGTGTCCCCAGGACAGACGGCCA
GGATCACCTGCTCTGGAGATGCATTGCCAAAGCACTTTGCTTATTGGTACCAGCAGAAGCCAGGCCAGGCCCTGTAC
TGGTGATATATAAAGACACTGAGAGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAGCTCAGGGACAACAGTCA
CGTTGACCATCAGTGGAGTCCAGGCAGAAGACGAGGCTGACTATTACTGTCAATCACCAGACAGCAGTGGTACTAATG
AAGTCTTCGGAACCTGGGACCAAGGTCACCGTCCTAAGTCAGCCCAAGGCCAACCCACGGT
>light_1d|025990_0147_0876|2
CTCACTGCACAGGCTCTGAGGCCTCCTATGAGCTGACACAGCCACCCTCGGTGTCTAGTGTCCCCAGGACAGACGGCCA
GGATCACCTGCTCTGGAGATGCATTGCCAAAGCACTATGCTTATTGGTACCAGCAGAAGCCAGGCCAGGCCCTGTGT
TGGTGATATATAAAGACAGTGAAGAGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAGCTCAGGGACAACAGTCA
CGTTGACCATCAGTGGAGTCCAGGCAGAAGACGAGGCTGACTATTACTGTCAATCACCAGACAGCAGTGGTACTTATG
AAGTCTTCGGAACCTGGGACCAAGGTCACCGTCCTAGGTGAGCCCAAGGCCAACCCCACTGT
>light_1d|001721_0226_0129|2
CTCACTGCACAGGGTCTGAGGCCCTCCTATGAGCTGACACAGCCACCCTCGGTGTCTAGTGTCTGCGGCCCCAGGACAGAAGGTCA
CCATCTCCTGCTCTGGAAGCAGCTCCAACATTGGCAAAAATTATGTATCCTGGTACCAGCATCTCCCAGGAACAGCCC
CCAACTCCTCATCTATGAAAATGATGAGCGACCCTCAGGGATTCTGACCGATTCTCTGGCTCCAAGTATGGCACGT
CAGCCACCCTGGGCATCACCGGACTCCAGACTGGGGACGAGGCCACTTATTTCTGCGGAACATGGGATACCAGCCTGG
GTACTAATTGGGTGTTTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTGAGCCCAAGGCTGCCCCCTCGGT
>light_1d|035774_0389_0427|2
CTCACTGCACAGGGTCTGAGGCCCTCCTATGAGCTGACACAGCCACCCTCGGTGTCTAGTGTCTGCGGCCCCAGGACAGAAGGTCA
CCATCTCCTGCTCTGGAAGCAGCTCCAACATTGGGAGTAATCGTGTATCCTGGTACCAGCAGCTCCCAGGAACAGCCC
CCAACTCCTCATCTATGAAAATAATGAGCGACCCTCAGGGATTCTGACCGATTCTCTGCCTCCAAGTCTGGCACGT
CAGCCACCCTGGTCATCACCGGACTCCAGACTGGGGACGAGGCCACTTATTTCTGCGGAACATGGGATACCAGCCTGA
GAATTAATTGGGTGTTTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTGAGCCCAAGGCTGCCCCCTCGGT
>light_1d|007823_0201_0454|2
TCTGGCTCCCAGGTGCCAGATGTGACATCCTGTTGACCCAGTCTCCATCCTTCTGTCTGCAGCTGTAGGAAACAGAA

TCACCATTACTTGCCGGGCCAGTCAGGGCATTAGTAGTTATTTAGCCTGGTTTCAGGAAAAACCAGGGAAAGCCCCTA
AACTCCTGATTTATGGTGCATCCATTTTGCAAAGTGGGGTCCCATCAAGGTTTCAGCGGCAGTGGATCTGGGACAGAGT
TCACTCTCACAAATCAGGAGCCTGCAGCCTGAAGATTTTGCAACTTATTACTGTCAACAGTTTACTAATTTCCCGCTCA
CTTTCGGCGGAGGGACCAAGGTGGAGATCAAACGAACTGTGGCTGCACCATCTG

>light_1d|008086_0088_0336|1

ACTCTGGCTCACAGATACCACCGGAGAAATTGTGTTGACGCAGTCTCCAGGCACCCTGTCTTTGTCTCCAGGGGAAAG
AGCCACCCTCTCCTGCAGGGCCAGTCAGAGAGTTAGCAGCAACTACTTAGCCTGGTACCAGCAGAAATCAGGCCAGGC
TCCCAGGCTCCTCATCTATAGTGCATCCCGCAGGGCCACTGGCATCCCAGACAGGTTTCAGTGGCAGTGGGTCTGGGAC
AGACTTCACTCTCACCATCAGCAGACTGGAGCCTGAAGATTTTGCCGTGTATCACTGTCAACAGTATGGTAGTTCATG
GCGCACTTTTGGCCAGGGGACCAAGGTGGAGATCAGACGAACTGTGGCTGCACCATCTG

>light_1d|004730_0220_0168|2

CTTACTGCACAGGATCCGTGGCCTCCTATGAGCTGACTCAGCCACCCTCAGTGTCCGTGTCCCCAGGACAGACAGCCA
GCATCACCTGCTCTGGAGATAAATTGGGGGATAAATATGCTTGCTGGTATCAGCAGAAGCCAGGCCAGTCCCCCTGTGC
TGGTCATCTATCAAGATAGCAAGCGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAACCTCTGGGAACACAGCCA
CTCTGACCATCAGCGGGACCCAGGCTATGGATGAGGCTGACTATTACTGTGAGGCGTGGGACAGCAGCACTGTGGTAT
TCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT

>light_1d|018024_0239_0556|2

CTCACTGCACAGGGTCTTGGGCCAGTCTGTGTTGACGCAGCCGCCCTCAGTGTCTGCGGCCTCAGGACAGGAGGTCA
CCATCTCCTGCTCTGGAAGCAGCTCCAACATTGGGAGTTATCGTGTATCCTGGTATCAGCACCTCCCAGGAACAGCCC
CCAACTCCTCATCTATGAAAATGATCAGCGACCCTCAGGGATTCTTGACCGATTCTCTGGCTCCAAGTCTGGCACGT
CAGCCACCCTGGTCATCACCGGACTCCTGACTGCGGACGAGGCCGATTATTACTGCGGAACATGGGATAACAGCCTGA
GGATTAATTGGGTGTTTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT

>light_1d|003754_0225_0116|1

CACTCACTCTGCAGTGTGAGTGGTCCAGGCAGGACTGACTCAGCCACCCTCGGTGTCCAAGGACTTGAGACAGACCGC
CACACTCACCTGCACTGGGAACAGCAACAATGTTGGCAACCTAGGAGCAGCTTGGCTGCAGCAGCACCAGGGGCCACCC
TCCCACACTCCTATCCTACAGGGATAACAACCGGCCCTCAGGGATCTCAGAGAGATTCTCTGCATCCAGGTCAGGAAA
TACAGCCTCCCTGTCCATTACTGGACTCCAGCCTGACGACGAGGCTGACTATTACTGCTCAGCATGGGACAGCAGCCT
CAGTGCTTGGGTGTTTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT

>light_21d|004730_0220_0168|2

CTTACTGCACAGGATCCGTGGCCTCCTATGAGCTGACTCAGCCACCCTCAGTGTCCGTGTCCCCAGGACAGACAGCCA
GCATCACCTGCTCTGGAGATAAATTGGGGGATAAATATGCTTGCTGGTATCAGCAGAAGCCAGGCCAGTCCCCCTGTGC
TGGTCATCTATCAAGATAGCAAGCGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAACCTCTGGGAACACAGCCA
CTCTGACCATCAGCGGGACCCAGGCTATGGATGAGGCTGACTATTACTGTGAGGCGTGGGACAGCAGCACTGTGGTAT
TCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT

>light_21d|010947_0053_0325|1

ACTCTGGCTCACAGATACCACCGGAGAGATTGTGTTGACGCAGTCTCCAAGCACCCCTGTCTTTCTCTCCAGGAGAAAG
AGCCACCCTCTCCTGCAGGGCCAGTCAGATTATTACCAGCAGCTACTTAGCCTGGTACCAGCAGAAACCTGGCCAGGC
TCCCAGGCTCCTCATCTATGGTGGATCCAGCAGGGCCACTGGCATCCCAAACAGGTTTACTGGCAGTGGGGCTGGGAC
AGACTTCACGCTCACCATCAGCAGACTGGAGCCTGAAGATTTTGCAAGTGTATTACTGTCAACAACATGGTAACTCACC
GGTCACTTTTCGGCGGAGGGACCAAGGTGGAGATCAAAGGAACTGTGGCTGCACCATCTG

>light_21d|016345_0101_0575|2

CTTGCTGCCCAGGGTCCAATTCCCAGGCTGTGGTGACTCAGGAGCCCTCACTGACTGTGTCCCCAGGAGGGACAGTCA
CTCTCACCTGTGGCTCCAGCACTGGAGCTGTCACCAGTGGTCATTATCCCTACTGGTTCCAGCAGAAGCCTGGCCAAG
CCCCAGGACACTGATTTCTGATAACAAGCAACAAATGTTCTTGACCCCTGGCCGGTTCTCAGGCTCCCTCCTTGGGG
GCAAAGCTGCCCTGACCCTTTCGGGTGCGCAGCCTGAGGATGAGGCTGACTATTATTGCTTCCTCTACTTTGATGGTC
CTATAGTTTTTCGGCGGAGGGACCAAGCTGACCGTCCTAAGTCAGCCCAAGGCTGCCCCCTCGGT

>light_21d|016683_0138_0575|0

CTCTGGATCTCTGGTGCCTACGGGGACATCGTGATGACCCAGTCTCCAGATTCCCTGGCTGTGTCTCTGGGCGAGAGG
GCCACCATCAACTGCAAGTCCAGCCAGAGTGTTTTGTGGGGCCCCAACAAATAAGAACTACTTAGCTTGGTACCAGCAG
AAACCAGGACAGTCTCCTAAGTTGCTCATTTACTGGGGATCTACCCGGAAATCCGGGGTCCCTGACCGATTCACTGGC
AGCGGGTCTGGGACAGATTTCACTCTCACCATCAGCAGCCTGCAGGCTGAAGATGTGGCAGTTTATTACTGTGAGCAT
TATCATACTCCACCGTACACTTTTGGCCAGGGGACCAAGCTGGAGATCAAACGAACTGTGGCTGCACCATCTG

>light_21d|014243_0165_0699|2

TCTGGGTCCCAGGTGCCAAATGTGTCTGTCAGATGACCCAGTCTCCTTCCACCCTGTCTGCATCTGTAGGAGACAGAG
TCACCATCACTTGCCGGGCCAGTGAGACTGTTGGAACGTGGTTGGCCTGGTATCGGCAGAAACCAGGGAAAGCCCCCTA
ACCTCCTGATCTATGAGGCCTCTATTTTAGAAAGTGGGGTCCCATCGAGGTTGAGCGGCAGTGGATCTGGGACAGAGT
TCACTCTCACCATCAGCAGCCTGCAGCCTGATGATTTTGCAACTTATTACTGCCAACAGTATAATACTGGTGGACAT
TCGGCCAAGGGACCAAGGTGGAAATCAAGCGAACTGTGGCTGCACCATCTG

>light_21d|007238_0089_0294|2

CTTGCTGCCCAGGTCCAATTCCCAGGCTGTGGTGACTCAGGAGCCCTCACTGACTGTGTCCCCAGGAGGGACAGTCA
CTCTCACCTGTGGCTCCAGCACTGGAGCTGTCACCAGTGGTCATTATCCCTACTGGTTTCAGCAGAAGGCTGGCCAAG
CCCCCAGGACACTGATGTATGATATAAGCATCAAACGTCTCTGGACCCCTGCCCGGTTCTCAGGCGCCCTCCTTGGGG
GCAAAGCTGCCCTGACCCTTTCGGGTGCGCAGCCTGAGGATGAGGCTGAATATTATTGTTTACTCTATTATAATGGTG
TCAGGGTGTTCGGCGGAGGGACCAAACTGACAGTCTAGGTCAGCCCAAGGCTGCCCCCTCGGT

>light_21d|015152_0165_0715|1

CCTCACTCTTTGCATAGGTTCTGTGGTTTCTTCTGAGCTGACTCAGGACCCTGCTGTGTCTGTGGCCTTGGGACAGAC
AGTCAGGATCACATGCCAAGGAGACAGCCTCAGAAGCTATTATGCAAGCTGGTACCAGCAGAAGCCAGGACAGGCCCC
TGTACTTGTCTATCTATGGTAAAAACAACCGGCCCTCAGGGATCCCAGACCGATTCTCTGGCTCCAGCTCAGGAAACAC
AGCTTCCTTGACCATCACTGGGGCTCAGGCGGAAGATGAGGCTGACTATTACTGTAACCTCCCGGGACAGCAGTGGTAA
CCATGTGGTATTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT

>light_21d|003565_0121_0193|2

CTCGGGACACAGGGTCTTGGGCCAGTCTGCCCTGACTCAGCCTGCCTCCGTGTCTGGGTCTCCTGGACAGTCGATCA
CCATCTCCTGCACTGGAACCAGCAGTGATGTTGGGAGATATAACCTTGTCTCCTGGTACCAACAACACCCAGGCAAAG
CCCCCAAATTCTTGATTTATGAGGTCAGTAAGGGGCCCTCAGGGGTTTCTAATCGCTTCTCTGGCTCCAAGTCTGGCA
ACACGGCCTCCCTGACAATCTCTGGGCTCCAGGCTGAGGACGAGGCAGATTATTACTGCTGCTCATATGCAGGTAGTA
GCACTTTTGTCTTCGGAACCTGGGACCACGGTCACCGTCCTAGGTCAGCCCAAGGCCAACCCACGGT

>light_21d|010775_0076_0480|2

CTCACTGCACAGGCTCTGAGGCCTCCTATGAGCTGACACAGCCACCCTCGGTGTGAGTGTCCCCAGGACAGACGGCCA
GGATCACCTGCTCTGGAGATGCATTGCCAAAGCAATATGCTTATTGGTACCAGCAGAAGCCAGGCCAGGCCCTGTGC
TGGTGATATATAAAGACAGTGAGAGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAGCTCAGGGACAACAGTCA
CGTTGACCATCAGTGGAGTCCAGGCAGAAGACGAGGCTGACTATTACTGTCAATCAGCAGACAGCAGTGGTACTTATG
TGGTATTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT

>light_21d|003754_0225_0116|1

CACTCACTCTGCAGTGTGAGTGGTCCAGGCAGGACTGACTCAGCCACCCTCGGTGTCCAAGGACTTGAGACAGACCGC
CACACTCACCTGCACTGGGAACAGCAACAATGTTGGCAACCTAGGAGCAGCTTGGCTGCAGCAGCACCAGGGCCACCC
TCCCACACTCCTATCCTACAGGGATAACAACCGGCCCTCAGGGATCTCAGAGAGATTCTCTGCATCCAGGTCAGGAAA
TACAGCCTCCCTGTCCATTACTGGACTCCAGCCTGACGACGAGGCTGACTATTACTGCTCAGCATGGGACAGCAGCCT
CAGTGCTTGGGTGTTTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT

>light_7d|004730_0220_0168|2

CTTACTGCACAGGATCCGTGGCCTCCTATGAGCTGACTCAGCCACCCTCAGTGTCCGTGTCCCCAGGACAGACAGCCA
GCATCACCTGCTCTGGAGATAAATTGGGGGATAAATATGCTTGCTGGTATCAGCAGAAGCCAGGCCAGTCCCCTGTGC
TGGTCATCTATCAAGATAGCAAGCGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAACCTCTGGGAACACAGCCA

CTCTGACCATCAGCGGGACCCAGGCTATGGATGAGGCTGACTATTACTGTCAGGCGTGGGACAGCAGCACTGTGGTAT
TCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT
>light_7d|016513_0160_0702|2
CTCAGGGCACAGGGTCTTGGGCTCAGTCTGCCCTGACTCAGCCTCGCTCAGTGTCCGGGTCTCCTGGACAGTCAGTCA
CCATCTCCTGCACCTGGAACCAGCAGTGATGTTGGTAGTTATAACTATGTCTCCTGGTACCAACAGCAGCCAGGCAAAG
CCCCACACTCATGATTTATGATGTCACTAAGCGGCCCTCAGGGGTCCCTGATCGCTTCTCTGGTTCCAAGTCTGGCA
ACACGGCCTCCCTGACCATCTCTGGGCTCCAGGCTGAGGATGAGGCTGATTATTACTGCTGCTCATATGCAGGCAACT
CTTATGTCTTCGGAAGTGGGACCAAGGTCACCGTCCTAGGTCAGCCCAAGGCCAACCCACGGT
>light_7d|016824_0202_0719|2
CTCACTGCACAGGCTCTGAGGCCTCCTATGAGCTGACACAGCCACCCTCGGTGTCAGTGTCCCCAGGACAGACGGCCA
GGATCACCTGCTCTGGAGATGCATTGCCAAAGCACTTTGCTTATTGGTACCAGCAGAAGCCAGGCCAGGCCCTGTAC
TGGTGATATATAAAGACACTGAGAGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAGCTCAGGGACAACAGTCA
CGTTGACCATCAGTGGAGTCCAGGCAGAAGACGAGGCTGACTATTACTGTCAATCACCAGACAGCAGTGGTACTAATG
AAGTCTTCGGAAGTGGGACCAAGGTCACCGTCCTAAGTCAGCCCAAGGCCAACCCACGGT
>light_7d|076796_0530_0700|1
ACTCTGGCTCACAGATAACCACCGGAGAAATTGTGTTGACGCAGTCTCCAGGCACCCTGTCTTTGTCTCCAGGGGAAAG
AGCCACCCTCTCCTGCAGGGCCAGTCAGAGTGTTAGCAGCAGCTACTTAGCCTGGTACCAGCAGAAACCTGGCCAGGC
TCCCAGGCTCCTCATCTATGGTGCATCCAGCAGGGCCACTGGCATCCCAGACAGGTTTCAGTGGCAGTGGGTCTGGGAC
AGACTTCACTCTCACCATCAGCAGACTGGAGCCTGAAGATTTTGCAGTGTATTACTGTCAGCAGTATGGTAGCTCACC
TTGGACGTTTCGGCCAAGGGACCAAGGTGGAAATCAAACGAACTGTGGCTGCACCATCTG
>light_7d|006092_0191_0398|1
ACTCTGGCTCACAGATAACCACCGGAGAAATTGTGTTGACGCAGTCTCCAGGCACCCTGTCTTTGTCTCCAGGGGAAAG
AGCCTTCCTCTCCTGCAGGGCCAGTCAGACTGTTCCCAGCAGCTACTTAGCCTGGTACCAGCAGAGACCTGGCCAAGT
TCCCAGGCTCCTCATCTATGATGCATCCAGCAGGGCCACTGGCATCTCAGACAGGTTTTCAGTGGCAGTGGGTCTGGGAC
AGACTTCACTCTCACCATCAACACACTGGAGCCTGAAGATTCTGCTGTGTATTACTGTCAGCAATATGGTAGCTCACC
GACGTTTCGGCCAAGGGACCAAGGTGGAAATCAAACGAACTGTGGCTGCACCATCTG
>light_7d|015152_0165_0715|1
CCTCACTCTTTGCATAGGTTCTGTGGTTTCTTCTGAGCTGACTCAGGACCCTGCTGTGTCTGTGGCCTTGGGACAGAC
AGTCAGGATCACATGCCAAGGAGACAGCCTCAGAAGCTATTATGCAAGCTGGTACCAGCAGAAGCCAGGACAGGCCCC
TGTAATTGTCATCTATGGTAAAAACAACCGGCCCTCAGGGATCCCAGACCGATTCTCTGGCTCCAGCTCAGGAAACAC
AGCTTCCTTGACCATCACTGGGGCTCAGGCGGAAGATGAGGCTGACTATTACTGTAACTCCCGGGACAGCAGTGGTAA
CCATGTGGTATTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT
>light_7d|037667_0422_0484|1
ACTCTGGCTCACAGATAACCACCGGAGAAATTGTGTTGACACAGTCTCCAGGCACCCTGTCTTTGTCTCCAGGGGAAAG
AGCCACCCTCTCCTGCAGGGCCAGTCAGAGTGTTAGCACCTACTTAGCCTGGTACCAACAGAAACCTGGCCAGCCTCC
CAGACTCCTCATCTATGATGCATCCAACAGGGCCACTGGCATCCCAGCCAGGTTTCAGTGGCAGTGGGTCTGGGACAGA
CTTCACTCTCACCATCAGCAGCCTAGAGCCTGAAGATTCTGCAGTTTATTACTGTCAGCAGCGTAGCACCTGGCCTGC
GACTTTTCGGCCCTGGGACCAAGTGGATATCAAACGAACTGTGGCTGCACCATCTG
>light_7d|059523_0412_0968|2
CTCACTGCACAGGCTCTGTGACCTCCTATGTGCTGACTCAGCCACCCTCGGTGTCAGTGGCCCCAGGAAAGACGGCCA
GGATTACCTGTGGGGGAAACAACATTGGAAGTAAAAGTGTGCACTGGTACCAGCAGAAGCCAGGCCAGGCCCTGTGC
TGGTCGTCTATGATGATAGCGACCGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAACCTCTGGGAACACGGCCA
CCCTGACCATCAGCAGGGTCGAAGCCGGGGATGAGGCCGACTATTACTGTCAGGTGTGGGATAGTAGTAGTGATCATG
TGGTATTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT
>light_7d|010775_0076_0480|2
CTCACTGCACAGGCTCTGAGGCCTCCTATGAGCTGACACAGCCACCCTCGGTGTCAGTGTCCCCAGGACAGACGGCCA


```

GGATCACCTGCTCTGGAGATGCATTGCCAAAGCAATATGCTTATTGGTACCAGCAGAAGCCAGGCCAGGCCCTGTGC
TGGTGATATATAAAGACAGTGAGAGGCCCTCAGGGATCCCTGAGCGATTCTCTGGCTCCAGCTCAGGGACAACAGTCA
CGTTGACCATCAGTGGAGTCCAGGCAGAAGACGAGGCTGACTATTACTGTCAATCAGCAGACAGCAGTGGTACTTATG
TGGTATTCGGCGGAGGGACCAAGCTGACCGTCCTAGGTCAGCCCAAGGCTGCCCCCTCGGT
>light_7d|030140_0479_0232|2
CTCACTGCACAGGGTCCTGGGCCAGTCTGTATTGACGCAGCCGCCCTCAGTGTCTGCAGCCCCAGGACAGAAGGTCA
CCATCTCCTGCTCTGGAAGCACCTCCAACATAGTTTCATAATTTTGTATCGTGGTTCCAGCATCTCCCAGGAACAGCCC
CCAAACTTCTCATATATGACAATAAGAGGCGGCCCTCAGGGATTCCTGACCGATTCTCTGGCTCCAAGTCTGGCGCGT
CAGCCACCCTGGACATCACTGGACTCCAGACTGGGGACGAGGCCGATTATTACTGCGGAGCGTGGGATAGCAGCCTGA
GTGCTGTGGTCTTCGGCGGAGGGACCAAGCTGACCGTCCTGGGTCAGCCCAAGGCTGCCCCCTCGGT

```

Table 6.

[0373] The most highly expressed clones were manually matched to the original clones. Two of them don't match up perfectly. The names of the sequences include the reading frame (last field; 0, 1, or 2)

[0374] To match the original synthesized heavy chains identified using highly-expressed clones with the new heavy chains identified via highly-expressed junctions, both sets of sequences and were analyzed with multiple sequence alignment (CLUSTAL) to determine how they paired. For the most part, they had identical sequence pairs, which was a consistent check that the two methods produced the same heavy chains. A re-plot of highly expressed time series for heavy and light chain junctions is set forth at Figure 50.

EXAMPLE VIII

Heavy Chain Immune Sequencing

[0375] Samples were concentrated using RNeasy kit from Qiagen, elute in 2x 30µl and nanodrop (See Table 7 below)

VDJ-ID	MID	vaccination sample	date	RNA concentration stock(ng/µl) prior DNase digestion	Volume for 8µg DNase Digestion	Water to 150µl

1	GMC 1	-8 day	Dec 7 th 2009	91.71	87.23	62.77
2	GMC 2	-2 day	Dec 13 th 2009	75.6	105.82	44.18
3	GMC 3	-1hr	Dec 15 th 2009	89.51	89.38	60.62
4	GMC 4	+1hr	Dec 15 th 2009	96.87	82.58	67.42
5	GMC 5	+1day	Dec 16 th 2009	117.3	68.20	81.80
6	GMC 6	+3day	Dec 18 th 2009	90.22	88.67	61.33
7	GMC 7	+7day	Dec 22 nd 2009	87.45	91.48	58.52
8	GMC 8	+14day	Dec 29 th 2009	131.12	61.01	88.99
9	GMC 9	+21day	Jan 5 th 2010	127.17	62.91	87.09
10	GMC 10	+28day	Jan 12 th 2010	139.24	57.45	92.55
11	IDO 1	-8 day	Dec 7 th 2009	98.84	80.94	69.06
12	IDO 2	-2 day	Dec 13 th 2009	84.45	94.73	55.27
13	IDO 3	-1hr	Dec 15 th 2009	72.56	110.25	39.75
14	IDO 4	+1hr	Dec 15 th 2009	78.14	102.38	47.62
15	IDO 5	+1day	Dec 16 th 2009	51	156.86	-6.86
16	IDO 6	+3day	Dec 18 th 2009	45.22	176.91	-26.91
17	IDO 7	+7day	Dec 22 nd 2009	101.42	78.88	71.12
18	IDO 8	+14day	Dec 29 th 2009	109.09	73.33	76.67
19	IDO 9	+21day	Jan 5 th 2010	127.49	62.75	87.25
20	IDO 10	+28day	Jan 12 th 2010	89.01	89.88	60.12
21	FV 1	-8 day	Dec 7 th 2009	96.87	82.58	67.42
22	FV 2	-2 day	Dec 13 th 2009	127.13	62.93	87.07
23	FV 3	-1hr	Dec 15 th 2009	115.25	69.41	80.59
24	FV 4	+1hr	Dec 15 th 2009	136.05	58.80	91.20
25	FV 5	+1day	Dec 16 th 2009	142.15	56.28	93.72
26	FV 6	+3day	Dec 18 th 2009	113.49	70.49	79.51
27	FV 7	+7day	Dec 22 nd 2009	91.81	87.14	62.86
28	FV 8	+14day	Dec 29 th 2009	84.92	94.21	55.79
29	FV 9	+21day	Jan 5 th 2010	131.06	61.04	88.96
30	FV 10	+28day	Jan 12 th 2010	90.98	87.93	62.07

Table 7.

Vaccination and Blood draw

[0376] Vaccine info: Seasonal flu 2010. Samples were extracted from 9 mL of blood from a single individual at various time points and processed using the leukolock kit (alternate protocol). Blood flow through was centrifuged, and plasma was saved and stored at -80°C. RBC fraction was saved and cross-linked in formaldehyde and store at 4°C.

RNA extraction

[0377] Extracted RNA using the leukolock alternative protocol (trizol) with the following modifications: Conducted the total RNA extraction protocol which included small RNAs. Only vacuumed the first binding step, for each wash, centrifuged at maximum of 10,000 x G. Eluted with 250 µl of dH₂O at 85°C, spun at max speed. Transferred RNA in solution without the little white pellet (probably some of the filter). Nanodropped the RNA, and stored at -80°C.

Average	ng/µl	260/280	260/230	A260	A280
	100.95	2.09	1.99	2.52	1.20

[0378] Protected RNA stock (bring to 10 mM Tris, 0.1 mM EDTA). To the 250µl of RNA, added the following:

	1 tube	50 tubes
• 1M Tris-HCl	2.5µl	125µl
• 50 mM EDTA	0.5µl	25µl
	(3µl/tubes)	

Digest 8 µg with DNase and concentrate

[0379] Conducted DNase digestion using Ambion TURBO DNA-free (AM1907) with the following modification. Added 0.1 volume (15 µl) 10X TURBO DNase Buffer and 1 µL TURBO DNase to the RNA, and mixed gently. Incubated at 37°C for 20–30 min. Added resuspended DNase Inactivation Reagent (typically 0.1 volume) (15µl) and mixed well. Incubated 5 min at room temperature, mixing occasionally. Centrifuged at 10,000 x g for 5 min and transferred the RNA to a fresh tube. Conducted acid phenol and CHCl₃ extraction, followed by ethanol (EtOH) precipitation with NaOAc and glycoblue. Eluted in 50µl of 10 mM Tris 0.1 mM EDTA, QC and stored at -80. This was repeated with 5 µg for sample 5 and 22 because RNA was too low (resuspend in 25µl).

[0380] QC DNase treated RNA. Nanodrop Quantitation (see Table 8 below)

ID	Sample ID	ng/ul	260/280	260/230	for 500 ng	water to 7µl
1	GMC 1	136.32	2.09	1.95	3.67	3.33
2	GMC 2	132.63	2.12	1.98	3.77	3.23
3	GMC 3	95.13	2.07	1.85	5.26	1.74
4	GMC 4	96.39	2.08	1.89	5.19	1.81
5	GMC 5	91.17	2	1.61	5.48	1.52
6	GMC 6	131.56	2.03	1.89	3.80	3.20
7	GMC 7	138.75	2.1	1.99	3.60	3.40
8	GMC 8	120.75	2.09	1.92	4.14	2.86
9	GMC 9	82.29	2.05	1.92	6.08	0.92
10	GMC 10	87.76	2.08	1.85	5.70	1.30
11	IDO 1	111.26	2.12	1.9	4.49	2.51
12	IDO 2	119.51	2.11	1.9	4.18	2.82
13	IDO 3	203.4	2.12	2.06	2.46	4.54
14	IDO 4	190.66	2.11	2.04	2.62	4.38
15	IDO 5	308.04	2.12	2.09	1.62	5.38
16	IDO 6	311.67	2.13	2.12	1.60	5.40
17	IDO 7	137.57	2.11	1.97	3.63	3.37
18	IDO 8	155.92	2.1	1.99	3.21	3.79
19	IDO 9	121.76	2.12	1.99	4.11	2.89
20	IDO 10	159.76	2.11	1.95	3.13	3.87
21	FV 1	112.87	2.09	1.91	4.43	2.57
22	FV 2	83.53	2.03	1.67	5.99	1.01
23	FV 3	95.56	2.08	1.87	5.23	1.77
24	FV 4	84.47	2.1	1.85	5.92	1.08
25	FV 5	95.18	2.12	1.9	5.25	1.75
26	FV 6	100.07	2.11	1.9	5.00	2.00
27	FV 7	113.7	2.1	1.89	4.40	2.60
28	FV 8	182.87	2.12	2.05	2.73	4.27
29	FV 9	86.99	2.09	1.83	5.75	1.25
30	FV 10	128.91	2.11	1.91	3.88	3.12
average	133.88	2.09	1.92			

Table 8.

[0381] Primer mixing (See Table 9 below).

Primer type	plate date	plate stock	Well location	µl for each primer	µl H ₂ O	Final mix
5 x J IGH RT-PCR primer	20080707					2 µM
6 x J IGH PCR primer	20080924	200 µM	Plate 1 A1-D7	20 µl each (100 µl	880 µl	4 µM (1000

				total)		μl)
43x V IGH PCR primer	20080924	20 0μM	Plate 3 A1-A6	20 μl each (860 μl total)	140 μl	4 μM (1000 μl)

Table 9.

Reverse transcription

[0382] Assembled an RT reaction as follows:

	<u>1 tube</u>	<u>35 tubes</u>
• H ₂ O (DEPC)	0 μl(to 12 μl)	-
• 2 μM Gene-specific primer (IGHC mix old)	5 μl (5 pmole)	-
• Total RNA	7 μl (750 ng)	-

Heated at 95°C for 1 min. followed by 65 °C for 5 min and ice for 1 min.

[0383] Spun down and added the following:

	<u>1 tube</u>	<u>35 tubes</u>
• 5x First strand buffer	4μl	140
• 10 mM dNTP mix	1μl	35
• 0.1 M DTT	1μl	35
• RnaseInh (Enzy)	1μl	35
• Superscript III	1μl	35

Incubated at 55°C for 60 min (8μl/tubes)

[0384] Inactivated enzyme by heating at 70°C for 15 min. Removed RNA/DNA hybrid by adding 1μl of *E. coli* RNaseH. Incubated at 37°C for 20 min, then ice.

[0385] Made a global master mix, then split across all samples (3x 10 times point of 200μl), then split each sample in 4 x. Used half for PCR and kept other half as backup. Assembled a PCR test reaction as follows:

	1t	2x35t
• dH ₂ O	21.6 µl (to 50 µl)	756
• cDNA	5 µl	-
• 5x HF buffer	10 µl	350
• dNTP(25 mM)	0.4 µl	14
• primer up (IGHV new-short) (4 µM)	6.25 µl (25 pmole)	218.75
• primer low (IGHC new-short)(4 µM)	6.25 µl (25 pmole)	218.75
• phusion	0.5 µl	17.5

Added 90 PCR mix to 10 µl cDNA, split PCR reactions in 2 tube of 50, for each sample.

[0386] Performed thermal cycling as follows:

- 1- 98°C for 1 min
- 2- 98°C for 10 seconds
- 3- 62°C for 20 seconds
- 4- 72°C for 20 seconds go to step 2, 20x
- 5- 72°C for 5 min
- 6- 4°C pause

[0387] Re-pooled each 2 reactions into 1. Added 2µl of Exonuclease I (20U/µl) to each tube and incubate at 37°C for 20 min. AMPure XP purified ratio 1.8:1. Resuspend in 40 µl.

[0388] Conducted SPRI purification as follows:

- Add the require amount of AMPure XP beads (1.8:1 ratio) to the DNA sample in buffer EB.
- Vortex to mix
- Incubate for 5 minutes at room temperature.
- Magnet (MPC) for 5 minutes. Leave the tube of beads in the MPC during all wash steps
- Remove the supernatant (keep in case of failure)and wash the beads twice with 500 µl of 70% ethanol, incubating for 30 sec each time.
- Remove all the supernatant, quick spin , remove last drop and allow the AMPure beads to air dry completely (2 min).
- Remove the tube from the MPC, add 40 µl of 10 mM Tris-HCl, 0.1 mM EDTA, pH 8.0 (or Qiagen's Buffer EB), and vortex for 30 sec to resuspend the beads, let sit for 3 min Magnet for 2 min and transfer supernatant to a new tube.

[0389] A diagnostic gel was run to check PCR efficiency. Ran 2 µl on a 2% Egel-X for 12 min.

Finalize library

[0390] The following step was written to finalize all remaining library. The plate was ordered as set forth in the Table 10 below.

Library type (all heavy chains)	Row placement and BC info 1=13, etc.
Triple VDJ-HC GMC	A1-A10
Triple VDJ-HC IDO	B1-B10
Triple VDJ-HC FV	C1-C10
RA	D1-D8
HIV 1/2	E1-E8 (5 progressor+5 negative) BC; 1-2-3-4-5/6-7-8-9-10
HIV 2/2	F1-F7 (5 elite) BC; 1-2-3-4-5

Table 10.

[0391] Performed blunting reaction using Enzymatics End repair kit as follows:

	1t	<u>60</u>
• H ₂ O	0 µl(to 25 µl)	0
• Post PCR Purified DNA	19 µl	-
• 10× End repair bufferbuffer	2.5 ul	150
• 1mM dNTP mix	2.5 µl (0.1 mM)	150
• enzyme mix (HC).	1 µl	60

[0392] Incubated at 25°C temperature for 30 min. (19/t). Heat inactivated at 75°C for 20 min.

[0393] A-tailed by adding the following directly to the mixture:

	<u>1t</u>	<u>60t</u>
• H ₂ O	20 µl(to 50µl)	1200
• Blunt DNA sample	25 µl	-
• 10X Klenow buffer	2.5 µl	150
• 10 mM dATP	0.5 µl	30
• Klenow exo HC(3' to 5' exo minus)	2 µl	120
Total reaction was now 50 ul (25/t)		

[0394] Incubated at 37°C for 30 min. Heat inactivated at 75 °C for 20 min. Prepared 454 Y-adapters. 10µl of 100µM of each primer A and B + 30µl of (10 mM Tris 0.1 mM EDTA at 50 mM NaCl)) (each adapter are at 20 µM final). Incubated 95 for 3 min, ramp to 15 forever at 0.1°C/sec.

[0395] For Y adapter ligation, the following was add directly to the reaction:

	1t	2x30t
• H ₂ O	0 (add µl to 100 µl)	-
• A-tailed DNA	50 ul	-
• 2x Quick ligase buffer	46 µl	1380
• 20µM adapter	2 µl	-
• T4 DNA ligase (quick)	2 µl	60

Incubate PCR at 22°C for 15 min. (48/t). Total reaction was then 100 µl.

[0396] Ampure was performed, and the substrate was eluted in 40 µl (using the liquidator). Library QC for pooling was tested on an HIV sample. Only the first 5 samples were tested (progressor) (Figure 51).

EXAMPLE IX

HIV VDJ

[0397] Samples are set forth in the Table 11 below.

Patient	Cohort	Date	HLA Type	Viral Load	Sex	DOB	Ethnicity
CR0744R	Progressor	4/12/2010	A-0201 A-0301 B-3701 B-5201 C-0602 C-1202	4-12-10 - 31100	M	1970	African American
CR0795Q	Progressor	4/19/2010	A-0101 A-0201 B-2705 B-5101 C-0102 C-1502	4-19-10 - 61700	M	1964	White
CR0216R	Progressor	4/29/2010	A-0301 A-6801 B-1401 B-3801 C-0802 C-1203	4-29-10 - 86600	F	1963	White
CR0778Z	Progressor	5/3/2010	A-3303 A-6801 B-0702 B- 1503 C-0702 C1800	5-3-10- 45700	M	1964	African American
CR0757R	Progressor	5/6/2010	A-0201 A-2902 B-1501 B-4404	5-6-10-	M	1964	White

			C-0102 C-1601	26300			
CTR0147	Elite Controller	4/15/2010	A-0103 A-3201 B-3508 B-7301 C-0401 C-1505	4-18-10 - <48	M	1953	White
CTR24	Elite Controller	4/15/2010	A-0201 A-3001 B-1302 B-5701 C-0602 C-0602	4-15-10 - <48	F	1954	Biracial
523507	Elite Controller	5/14/2010	A-3002 A-2402 B-1801 B-5301 C-0401 C-1203	5-14-10 - 56	M	1962	White
CTR174	Elite Controller	6/28/2010	A-0201 A-2301 B-5201 B-5801 C-0302 C-1601	2-18-10 - <48	M	1961	African American
FHCR0013E	Elite Controller	7/1/2010	A-0301 A-1101 B-0702 B-3501 C-0401 C-0702	2-17-10 - 133	M	1947	White
924426	HIV Negative	4/7/2010	A-0201 A-0205 B-5701 B-5701 C-0602 C-0602	0			
714739	HIV Negative	4/7/2010	A-0101 A-0201 B-1801 B-5101 C-0602 C-1402	0			
564006	HIV Negative	4/7/2010	A-0201 A-0201 B-1801 B-3901 C-0501 C-1203	0			
084928	HIV Negative	4/6/2010	A-0201 A-0301 B-1401 B-1801 C-0501 C-0802	0			
039701	HIV Negative	4/6/2010	A-0201 A-3303 B-1501 B-5801 C-0302 C-0801	0			

Table 11.

Sample aquisition

[0398] A fresh sample was obtained. 2 tubes of EDTA were prepared for each progressor, elite controller and HIV negative sample. Samples were processed immediately after receipt (ficolled. count cells). B Cell isolation was performed as follows:

1. Centrifuged cell suspension at 300 x G for 10 min., aspirated supernatant.
2. Resuspended cell pellet in 80 µL of cold buffer per 10⁷ cells.
3. Buffer was a 1:20 dilution of MACS BSA stock solution with autoMACS rinsing solution. Degassed before use.
4. Added 20 µL of CD19 microbeads per 10⁷ cells.
5. Incubated for 15 minutes in the refrigerator (2-8 °C).

6. Washed cells by adding 1-2 mL of cold buffer per 10^7 cells and centrifuge at 300 x G for 10 min. aspirate supernatant.
7. Resuspended up to 10^8 cells in 500 μ L cold buffer
8. Placed MS column in a magnetic field and rinse the column with 500 μ L of degassed buffer.
9. Added cell suspension.
10. Collected unlabeled cells that passed through and washed three times with 500 μ L of buffer.
11. Removed column and placed in a new collection tube.
12. Added 1 mL of buffer and plunged out the magnetically labeled cells using the plunger.
13. Counted cells.
14. In a small centrifuge, centrifuged the CD19+ cells for 10 minutes at 9,000 x G.
15. Aspirated the supernatant and added 300 μ L of lysis buffer. Snap froze and stored at -80 °C.

RNA extraction

[0399] Used mirvana as per manufacturer protocol, eluted in 150 (did double phenol, phase was smaller on second run). QC nanodrop, too low, so etoh ppt with glycobblue, elute in 20ul, nanodrop. Dilute number 4 1 :2. See Table 12 below.

ID	ng/ul RNA	260/280	260/230	Total RNA	Volume for 150ng	Water to 7
1	65.9	1.98	1.7	1252.1	2.276	4.724
2	32.4	1.92	1.33	615.6	4.630	2.370
3	26.1	2	1.48	495.9	5.747	1.253
4	216.1/120.6	2.03/1.97	2.05/1.58	4106/4583	1.244	5.756
5	103.4	2.02	1.93	1964.6	1.451	5.549
6	54.3	1.98	1.66	1031.7	2.762	4.238
7	89.9	2.02	1.89	1708.1	1.669	5.331
8	51.4	1.66	0.88	976.6	2.918	4.082
9	35.8	2.05	1.66	680.2	4.190	2.810
10	79.8	1.85	1.25	1516.2	1.880	5.120
11	104.4	1.96	1.62	1983.6	1.437	5.563
12	82.2	1.71	0.91	1561.8	1.825	5.175
13	154.8	1.97	1.72	2941.2	0.969	6.031
14	56.2	1.87	1.33	1067.8	2.669	4.331
15	84.4	1.8	1.06	1603.6	1.777	5.223

Table 12. Used extra mix to make 5x FV1 (180ng/ul... 1.67 ul RNA + 5.33 ul H₂O) for light chains and 5 for heavy chains.

[0400] Sample 8 was concentrated by ethanol precipitation, resuspended in 20 µl, nanodrop = 134.0. These were B cells RNA only, so there was no need for the same amount of starting RNA for VDJ research. Theorically, lymphocytes represent 30% of PBMC, for which B cells represent 10%, accordingly 15 ng should be sufficient. 150 ng was used.

[0401] A reverse transcription (RT) reaction was assembled as follows:

2x 1 tube

- H₂O(DEPC) 0 µl(to 12 µl)
- 2 µM Gene-specific primer (heavy or light) 5 µl (5 pmole)
- Total RNA 7 µl (150 ng)

Heated at 95°C for 1 min. followed by 65 °C for 5 min., then ice for 1 min.

Spun down and the following was added:

	<u>1 tube</u>	<u>30 tubes</u>
• 5x First strand buffer	4µl	120
• 10 mM dNTP mix	1µl	30

- 0.1M DTT 1µl 30
 - RnaseInhibitor-Enzymatics 1µl 30
 - Superscript III 1µl 30
- Incubated at 55 °C for 60 min (8µl/tubes)

[0402] Inactivated enzyme by heating at 70°C for 15 min. Removed RNA/DNA hybrid by adding 1µl of *E. coli* RNaseH (Enzy). Incubated at 37°C for 20 min., then ice. See Table 13 below.

Primer type	plate date	plate stock	Well location	µl for each primer	µl H2O	Final mix
5 x J IGH RT-PCR primer	Tube order 5933901			20 ul*5 tubes	+400 ul	4 µM* Dilute 1é2 before using
6 x J IGH PCR primer	20080924	200 µM	Plate 3 A1-A6	20 µl each (100 µl total)	880 µl	4 µM (1000µl)
43x V IGH PCR primer	20080924	200 µM	Plate 1 A1-D7	20 µl each (860 µl total)	140 µl	4 µM (1000µl)

Table 13.

[0403] Assembled a PCR test reaction as follows:

	1x	x 2per sample	x 15samples (45t)	(12t light chain)
dH ₂ O	21µl (to 50µl)	945	(1084.46)	252 (259.2)
cDNA	5µl	-	-	-
5x HF buffer	10µl	450		120
dNTP(10 mM) (25 mM)	1µl (0.4)	45 (18)		12 (4.8)
primer up (IGHV new-short) (4µM)	6.25µl (25pmole)	281.25 (225)		75
primer low (IGHC new-short)(4µM)	6.25µl (25pmole)	281.25 (224)		75
phusion	0.5µl	22.5		6

[0404] Added 90 PCR mix to 10 µl cDNA, split PCR reaction in 2 tubes of 50, for each sample.

Was low on primer, so used bolded values.

[0405] Thermal cycled as follows:

- 1- 98°C for 1 min
- 2- 98°C for 10 seconds
- 3- 62°C for 20 seconds
- 4- 72°C for 20 seconds go to step 2, 23x
- 5- 72°C for 5 min
- 6- 4°C pause

[0406] Re-pooled each 2 reaction into 1. Added 2 µl of Exonuclease I (20U/µl) to each tube and incubate at 37°C for 20 min. AMPure XP purify ratio 1.8:1. Resuspend in 40 µl.

[0407] Conducted SPRI purification: Added the require amount of AMPure XP beads (1.8:1 ratio) to the DNA sample in buffer EB. Vortexed to mix. Incubated for 5 minutes at room temperature. Magnet (MPC) for 5 minutes. Left the tube of beads in the MPC during all wash steps. Removed the supernatant (kept in case of failure) and washed the beads twice with 500 µl of 70% ethanol, incubating for 30 sec each time. Removed all the supernatant, quick spun, removed last drop and allow the AMPure beads to air dry completely (2 min). Removed the tube from the MPC, added 40 µl of 10 mM Tris-HCl, 0.1 mM EDTA, pH 8.0 (or Qiagen's Buffer EB), and vortexed for 30 sec to resuspend the beads, let sit for 3 min., Magnet for 2 min. and transferred supernatant to a new tube.

[0408] A diagnostic gel was run to check PCR efficiency (Figure 52). Ran 2 µl on a 2% Egel-X for 12 min.

Finalize library

[0409] The following method was developed to finalize the remaining library.

[0410] Plate was ordered as follows in the Table 14 below.

Library type (all heavy chains)	Row placement and BC info 1=13, etc.
Tripple VDJ-HC GMC	A1-A10
Tripple VDJ-HC IDO	B1-B10
Tripple VDJ-HC FV	C1-C10
RA	D1-D8
HIV 1/2	E1-E8 (5 progressor+5 negative) BC; 1-2-3-4-5/6-7-8-9-10
HIV 2/2	F1-F7 (5 elite) BC; 1-2-3-4-5

Table 14.

[0411] Performed blunting reaction using Enzymatics End repair kit as follows:

	1t	<u>60</u>
• H ₂ O	0 µl(to 25 µl)	0
• Post PCR Purified DNA	19 µl	-
• 10× End repair buffer	2.5 µl	150
• 1 mM dNTP mix	2.5 µl (0.1 mM)	150
• enzyme mix (HC)	1 µl	60
Incubated at 25°C temperature for 30 min		(19/t)

[0412] Heat inactivated at 75°C for 20 min.

[0413] A-tailed by adding the following directly to the mixture :

	<u>1t</u>	<u>60t</u>
• H ₂ O	20µl(to 50µl)	1200
• Blunt DNA sample	25µl	-
• 10X Klenow buffer	2.5µl	150
• 10 mM dATP	0.5µl	30
• Klenow exo HC(3' to 5' exo minus) 2µl		120
Total rx is now 50ul (25/t)		

Incubated at 37°C for 30 min. Heat inactivated at 75°C for 20 min.

[0414] Prepared 454 Y-adapters : 10µl of 100µM of each primer A and B + 30µl of (10 mM Tris 0.1 mM EDTA at 50 mM NaCl)) (each adapter are at 20 µM final). Incubated 95 °C for 3 min, ramp to 15 °C forever at 0.1 °C/sec.

[0415] Y adapter ligation, added directly to the reaction as follow;

	1t	2x30t
• H ₂ O	0 (add µl to 100 µl)	-
• A-tailed DNA	50 µl	-
• 2x Quick ligase buffer	46 µl	1380
• 20µM adapter	2 µl	-
• T4 DNA ligase (quick)	2 µl	60

Incubated on PCR at 22°C for 15 min (48/t). Total reaction was now 100 µl.

[0416] Ampure, elute in 40ul (used the liquidator)

Library QC foor pooling

[0417] Test on HIV sample, test only first 5 samples (progressor). Nanodrop progressor sample.

Dilute to 5ng/ul. Nanodrop :

1	36.8
2	31.7
3	24.8
4	33.5
5	31.3

Diluted to 1ng/µl using 10 mM tris, 0.05% Tween 20. Diluted to 0.01ng/µl (in 200 µl) in 10 mM tris, 0.05% Tween 20,

[0418] Preapred SYBR fast mix enough for triplicate run. Used 18 µl of mix, and added 2 µl of template to it.

	1t	20
• H ₂ O	7.2ul (to 20ul)	144
• 2x SYBR fast	10ul	200
• 10 µm Primer up	0.4ul	8

- 10 µm Primer low 0.4ul 8
- 0.01ng/ul DNA template 2ul -

[0419] Cycled:

95 1 min
 95 5sec
 60 20sec
 72 15 sec tepest 40 time

[0420] Prepared SYBR fast mix enough for triplicate run. Did 2 run, one set from the 0.01ng sample, and one from the initial 5 ng/ul sample. Used 15ul of mix, and add 5ul of template to it. For 0.01 ng sample, used as follows:

0.01 ng volume to sue	Tris
1.76	3.24
2.50	2.50
3.52	1.48
2.17	2.83
5.00	0.00

For the ~5 ng sample, the following dilution was performed down to a real 1 ng/µl and diluted 1/100 down to a real 0.01 ng/µl. Used 5 µl.

Sample	dilution fold
1	13.2
2	16.4
3	15.3
4	30.1
5	10.9

- H₂O 1t 20
- 2x SYBR fast 4.2ul (to 20ul) 84
- 10 µm Primer up 10ul 200
- 10 µm Primer low 0.4ul 8
- 0.01 or 5ng/ul DNA template 0.4ul 8
- 5ul -

Plate reader

[0421] Diluted sample as follows. Volume of H₂O added to volume 25ul of sample to achieve same concentration of 250 nM.

GMC	26.95	11.56	8.99	9.23	15.31	10.02	10.67	0.22	14.69	14.23
IDO	11.40	12.94	8.58	9.95	11.15	11.05	26.07	8.04	11.02	10.27
FV	15.61	21.09	17.18	16.65	6.26	12.26	20.76	14.12	13.01	15.07
RA	7.58	6.95	8.50	7.51	8.94	5.92	9.59	6.97		
HIV	45.28	23.65	21.59	29.23	23.86	27.46	29.26	29.74	34.51	28.37
HIVelite	37.04	29.70	27.82	34.51	37.93					

[0422] GMC1, mixed 25 µl of each sample for a total of 30*10=300 µl. IDO1, mixed 30 µl. FV1, mixed 30 µl. RA, mixed first 4 at 35 µl and last 4 at 35 µl in seprate batched, precipitated, and loaded on a gel to decide how to mix them together. HIV, pool equal ratio of each (40 µl), but did not mix elites samples with the combo neg+prog, these will be run with a gasket (so more read total for elite).

[0423] Qiagen prep +5 µl NaOAc (2 min wash), eluted in 30 µl, nanodrop.

#	Sample ID	Nucleic Acid Conc.	260/280	260/230	total ng	µM
1	GMC	16	2.01	1.98	480	1.39
2	IDO	25.7	1.93	1.87	771	2.22
3	FV	31.4	1.87	1.99	942	2.72
4	RA1	8.4	1.95	2.02	252	0.73
5	RA2	6.4	2.36	1.87	192	0.55
6	HIV1	75.7	1.85	2.17	2271	6.55
7	HIV2	37.9	1.87	2.18	1137	3.28
8	HIV control	22.4	1.95	1.99	672	1.94

[0424] Tested pipin prep with control sample number 8; 475-525, 475-550, 500-575 (Figure 53).
Ran all sample on pippin prep. Split HIV1 in 2 runs, Extract 475-550. 1, GMC; 2, IDO; 3, FV; 4, HIV1/2 (Figure 54).

Qubit library HS dsDNA

[0425] Pooled both RA samples together and re Mapure because RA2 has nothing.

no	ID / Qubit HS dsDNA	/ dilute to 5 ng/ μ l, take X μ l / add Y H ₂ O / volume total
1	GMC 8.44	3 2.064 5.064
2	IDO 7.99	3 1.794 4.794
3	FV 6.59	3 0.954 3.954
4	RA1 6.59	3 0.954 3.954
5	RA2	pool with RA1
6	HIV1 49.8	3 26.88 29.88
7	HIV2 25.2	3 12.12 15.12

[0426] Diluted 1/100 to 0.05ng/ul. Load 1 ul on e GEL of each sample.

Run real time

[0427] Prepared SYBR fast mix enough for triplicate run. Use 18ul of mix, and add 2ul of template to it.

	1t	20
• H ₂ O	7.2 μ l (to 20 μ l)	144
• 2x SYBR fast	10 μ l	200
• 10 μ m Primer up	0.4 μ l	8
• 10 μ m Primer low	0.4 μ l	8
• 0.05ng/ul DNA template	2 μ l	-

[0428] Cycle

95	1 min
95	5 sec

60 20 sec
 72 15 sec tepest 40 time

[0429] Diluted all to 5ng/μl using 13.5 μl of sample. Added H₂O to 13.5 μl of sample.

1 9.45
 2 8.073
 3 4.293
 4 4.293
 5 120.96
 6 54.54

Shiped 15 μl to 454.

EXAMPLE X

Emulsion PCR of Single Cell and Bead

Oligonucleotide 5'-3' (ordered form IDT HPLC purified when applicable)	
PR1-F4-2BIO (1 mM)	/52-Bio/CCA CTA CGC CTC CGC TTT CCT CTC TAT GG
PR1-R_Short (2 mM)	CTG CCC CGG GTT CCT CAT T

Tube	beads	process
1	Dynabeads 1μm myone C1 10mg/ml	all wash with magnet
2	Dynabeads 1μm myone C1 10mg/ml	all wash by centrifugation (1 min max speed)
3	IBS 1μm non-magnetic 10 mg/ml	all wash by centrifugation (1 min max speed)

Bead-loading with emulsion anchor primer

[0430] Magneted or centrifuged beads during all steps according to assay setup table. Vortexed beads to resuspend and transferred 100µl to 1.5 ml Ambion silicon tube. Washed twice with 200µl of 2x bind and wash buffer (2xB&W). Resuspended beads in 100µl of 2xB&W, add premixed anchor primer; 1 mM PR1-F4-2Bio (2 µl); H₂O (98 µl). Incubated on rotator for 20 min. Washed twice with 200 µl of 1xB&W. Washed once with 200µl of TE. Resuspend beads in 100µl of TE and stored on rotator in cold room until used.

[0431] Emulsion PCR

Prepared aqueous mix as follows:

1 tube

• dH ₂ O	316.4 µl (to 800 µl)
• 10x PCR buffer (enzymatics)	96 µl
• 50 mM MgCl ₂	242 µl
• 25 mM dNTP mix	135 µl
• 2 mM PR1R-S	6 µl
• 30% (w/v) BSA (Sigma)	1.6 µl
• Library template*	3 µl

[0432] Vortexed thoroughly to mix. *for Library template, a 1:20 dilution of a 1 nM 280 bp human genomic library was used.

Prepared oil mix:

1 tube

4 tubes (prep all in one 509 ml falcon)

• Tegosoft DEC	4.4 ml	17.6 ml
• mineral oil 1.2ml	4.8 ml	
• ABIL WE09	425 µl	1.7 ml

[0433] Vortexed thoroughly, allowed to degas. Aliquoted 5.5 ml into 50 ml Teflon-coated aluminum falcon tubes.

[0434] Create emulsion. To the 800 µl PCR mix, 100 µl Enzymatics Taq (5U/µl) was added, quickly vortexed and spun. 60 µl PR1F4 beads were immediately added, vortex quickly and spun. The total 960 µl was immediately transferred a to tube of oil, and vortexed for 2.25 min at 2200 rpm. The tube was placed into hydrocycler rack and the PCR program was started for overnight cycling.

[0435] Cycle as follows:

- 1- 94°C for 5 min
- 2- 94°C for 15 sec
- 3- 58°C for 30 sec (1 degree above to compensate hydrocycler setting)
- 4- 70°C for 75 sec
- 5- Got to 2 for 119x
- 6- 72°C for 2 min
- 7- Front cold tank (~10°C forever)

Breaking emulsion

[0436] 22 ml isopropanol was added to each aluminum falcon tube, and the tubes were vortexed for 50 sec at 3000 rpm. The contents were transferred to a new 50 ml polypropylene falcon tube and centrifuged (touch 4000 rpm and stop). The supernatants were poured off (pellet stuck well to bottom). 30 ml isopropanol was added and the tubes were vortexed for 40 sec at 3000 rpm, centrifuged (touch 4000 rpm and stop), and the supernatant was poured off (pellet stuck well to bottom)

[0437] 30 ml NXS buffer (10 mM tris-HCl pH 8.0, 100 mM NaCl, 1% (v/v) Triton X-100, 1% (v/v) SDS) was added and the tubes were vortexed for 1.5 min at 3000 rpm (vortex more if pellet was still stuck at bottom), centrifuged (maintain 4000 rpm for 15 sec and stop), and the supernatant was pipetted (carefully since the pellet did not stick to the bottom very well this time). 100 µl of TE was added, and the pellet was resuspend by pipetting.

The solution was transferred to a 1.5 siliconized Eppendorf tube, washed 1x with 200 μ l NXS, washed 2x with 200 μ l of TE, incubated for 5 min in 100 μ l of 100 mM NaOH at RT, washed 1x with 120 μ l of 100 mM NaOH, washed 2x with 150 μ l TE, resuspended in 30 μ l of TE and transferred to a new siliconized tube for storage.

Sequencing by ligation diagnostic of beads

[0438] Sequencing by ligation of the minus one position was conducted using methods known in the art.

Capture bead on acrylamide slide for SBL

[0439] An acrylamide bound-silane slide was prepared as follows:

- Beads in TE 7 μ l
- 40% Acryl 19:1 1.25 μ l
- 5% TEMED 0.5 μ l
- 0.5% APS 0.75 μ l

[0440] This was put on a slide by half wicking under coverslip and sliding a coverslip on top and the slide was flipped upside down. The slide polymerized (~30 min, max 2 hr) and a dry circle formed (prepare wash 1 during this time). Removed coverslip and put slide in conical tube with TE (shake for 10 min)

[0441] Prepared wash 1:

- 1M Tri-HCl pH 7.5 10 ml
- 2M KCl 25 ml
- 0.5M EDTA 4 ml
- 10% Triton X 1 ml

[0442] Prepared primer mix:

- 6X SSPE with 0.01% Triton X-100 100 μ l

- 1 mM anchor primer 1 μ l

[0443] Removed slide from conical tube, dried white Teflon surface with Kimwipe, added 100 μ l of anchor primer mix, incubated at 56°C for 1 min., placed in conical tube with Wash 1, immediately changed to fresh Wash 1, incubated at RT for 5 min in Wash 1 with gentle agitation.

[0444] Prepared ligation mix as follow;

- H₂O 41 μ l
- 10x T4 DNA ligase buffer 5 μ l
- Fluorescent nonamer mix 4 μ l
- T4 DNA ligase (2KU/ μ l) 1 μ l

[0445] Removed slide from conical tube, dried white Teflon surface with Kimwipe, added 50 μ l of ligation mix, incubated at 35°C for 5 min., placed in conical tube with Wash 1, immediately changed to fresh Wash 1, incubated at RT for 5 min in Wash 1 with gentle agitation, changed Wash 1 for TE, and incubated with gentle agitation for 10 min to 2 hr. Imaged under microscope and analyzed imaging data (See Figures 55-57).

EXAMPLE XI

Illumina VDJ

Objective

[0446] To develop a sample Illumina paired end sequencing protocol for immunoglobulin DNA.

Oligonucleotides

[0447] Primers were ordered having hotplate, standard desalting, first batch plain DNA compatible second section, with the proper Illumina adapter primary extension (not clustering) (Table 15). Plates were stocked at 200μM

Ref #	Name	Pos		TM
50494259	VDJ_IGHV_gDNA_01	A1	ACA GAA GTT CCA GGG CAG	54.62 °C
50494263	VDJ_IGHV_gDNA_02	A2	AGA CTC CAG GAA GGG CAG	56.48 °C
50494264	VDJ_IGHV_gDNA_03	A3	AGA CTC CAT GAA GGG CCA	55.89 °C
50494265	VDJ_IGHV_gDNA_04	A4	AGA CTC TGT GAA AGG CCG	54.53 °C
50494266	VDJ_IGHV_gDNA_05	A5	AGC ACA TCT CTG AAG ACC AG	54.19 °C
50494267	VDJ_IGHV_gDNA_06	A6	AGC ACA TCT CTG AAG AGC AG	54.46 °C
50494268	VDJ_IGHV_gDNA_07	A7	AGC ACG TCT CTG AAG AAC AG	54.53 °C
50494269	VDJ_IGHV_gDNA_08	A8	ATT ATG CAG TAT CTG TGA AAA GTC G	53.36 °C
50494270	VDJ_IGHV_gDNA_09	A9	CAG AAG CTC CAG GGC AG	55.68 °C
50494260	VDJ_IGHV_gDNA_10	A10	CAG ACT CTG TGA AGG GCA G	55.21 °C
50494261	VDJ_IGHV_gDNA_11	A11	CAG AGA AGT TCC AGG GCA G	55.28 °C
50494262	VDJ_IGHV_gDNA_12	A12	CAG GGC TTC ACA GGA CG	55.85 °C
50494271	VDJ_IGHV_gDNA_13	B1	CCC CTC CCT CAA GAG TCG	57.07 °C
50494275	VDJ_IGHV_gDNA_14	B2	CCC GTC CCT CAA GAG TCT	56.07 °C
50494276	VDJ_IGHV_gDNA_15	B3	CCG TCC CTC AAG AGT CG	54.86 °C
50494277	VDJ_IGHV_gDNA_16	B4	CCG TCC TTC CAA GGC CA	57.77 °C
50494278	VDJ_IGHV_gDNA_17	B5	CGC ACA GAA ATT CCA GGA CAG	56.24 °C
50494279	VDJ_IGHV_gDNA_18	B6	CGC ACA GAA GTT CCA GGA AAG	56.18 °C
50494280	VDJ_IGHV_gDNA_19	B7	CGC GTC TGT GAA AGG CAG	56.79 °C
50494281	VDJ_IGHV_gDNA_20	B8	GAC TCC GTG AAG GGC CG	58.62 °C
50494282	VDJ_IGHV_gDNA_21	B9	GAC TCA GTG AAG GGC CG	55.55 °C
50494272	VDJ_IGHV_gDNA_22	B10	GAC TCC GTG AAG GGC AG	55.55 °C
50494273	VDJ_IGHV_gDNA_23	B11	GAC TCT GTG AAG GGC CG	55.55 °C
50494274	VDJ_IGHV_gDNA_24	B12	GCA AAC TCT GTG AAG GGC AG	56.54 °C
50494283	VDJ_IGHV_gDNA_25	C1	GCA CAG AAG TTT CAG GGC AG	56.54 °C
50494284	VDJ_IGHV_gDNA_26	C2	GCA CCC GTG AAA GGC AG	57.09 °C
50494285	VDJ_IGHV_gDNA_27	C3	GCC CAT CTC TGA AGA GCA G	55.37 °C

50494286	VDJ_IGHV_gDNA_28	C4	GCG TCG GTG AAA GGC AG	56.97 °C
50494287	VDJ_IGHV_gDNA_29	C5	CTC CGT GAA GCG CCG	56.96 °C
50494288	VDJ_IGHV_gDNA_30	C6	TGC GTC TGT GAA AGG CAG	55.64 °C
50494292	VDJ_IGHJ_gDNA_01	D1	CTT ACC TGA GGA GAC GGT GAC C	58.73 °C
50494293	VDJ_IGHJ_gDNA_02	D2	CTC ACC TGA GGA GAC AGT GAC C	58.72 °C
50494294	VDJ_IGHJ_gDNA_03	D3	CTT ACC TGA AGA GAC GGT GAC C	56.83 °C

50494295	VDJ_IGHJ_gDNA_01I lluoverPE	D7	CTC GGC ATT CCT GCT GAA CCG CTC TTC CGA TCT CTT ACC TGA GGA GAC GGT GAC C	71.88 °C
50494296	VDJ_IGHJ_gDNA_02I lluoverPE	D8	CTC GGC ATT CCT GCT GAA CCG CTC TTC CGA TCT CTC ACC TGA GGA GAC AGT GAC C	71.90 °C
50494297	VDJ_IGHJ_gDNA_03I lluoverPE	D9	CTC GGC ATT CCT GCT GAA CCG CTC TTC CGA TCT CTT ACC TGA AGA GAC GGT GAC C	71.28 °C
50494298	VDJ_IGHV_gDNA_01 lluover	E1	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT ACA GAA GTT CCA GGG CAG	69.95 °C
50494302	VDJ_IGHV_gDNA_02 lluover	E2	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGA CTC CAG GAA GGG CAG	70.46 °C
50494303	VDJ_IGHV_gDNA_03 lluover	E3	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGA CTC CAT GAA GGG CCA	70.54 °C
50494304	VDJ_IGHV_gDNA_04 lluover	E4	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGA CTC TGT GAA AGG CCG	69.81 °C
50494305	VDJ_IGHV_gDNA_05 lluover	E5	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGC ACA TCT CTG AAG ACC AG	69.24 °C
50494306	VDJ_IGHV_gDNA_06 lluover	E6	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGC ACA TCT CTG AAG AGC AG	69.30 °C
50494307	VDJ_IGHV_gDNA_07 lluover	E7	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGC ACG TCT CTG AAG AAC AG	69.25 °C
50494308	VDJ_IGHV_gDNA_08 lluover	E8	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT ATT ATG CAG TAT CTG TGA AAA GTC G	67.88 °C
50494309	VDJ_IGHV_gDNA_09 lluover	E9	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CAG AAG CTC CAG GGC AG	71.06 °C
50494299	VDJ_IGHV_gDNA_10	E10	ACA CTC TTT CCC TAC ACG ACG	70.35 °C

	Illuover		CTC TTC CGA TCT CAG ACT CTG TGA AGG GCA G	
50494300	VDJ_IGHV_gDNA_11 Illuover	E11	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CAG AGA AGT TCC AGG GCA G	70.42 °C
50494301	VDJ_IGHV_gDNA_12 Illuover	E12	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CAG GGC TTC ACA GGA CG	71.03 °C
50494310	VDJ_IGHV_gDNA_13 Illuover	F1	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CCC CTC CCT CAA GAG TCG	71.14 °C
50494314	VDJ_IGHV_gDNA_14 Illuover	F2	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CCC GTC CCT CAA GAG TCT	71.05 °C
50494315	VDJ_IGHV_gDNA_15 Illuover	F3	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CCG TCC CTC AAG AGT CG	70.66 °C
50494316	VDJ_IGHV_gDNA_16 Illuover	F4	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CCG TCC TTC CAA GGC CA	71.86 °C
50494317	VDJ_IGHV_gDNA_17 Illuover	F5	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CGC ACA GAA ATT CCA GGA CAG	70.19 °C
50494318	VDJ_IGHV_gDNA_18 Illuover	F6	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CGC ACA GAA GTT CCA GGA AAG	70.16 °C
50494319	VDJ_IGHV_gDNA_19 Illuover	F7	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CGC GTC TGT GAA AGG CAG	70.93 °C
50494320	VDJ_IGHV_gDNA_20 Illuover	F8	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GAC TCC GTG AAG GGC CG	71.82 °C
50494321	VDJ_IGHV_gDNA_21 Illuover	F9	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GAC TCA GTG AAG GGC CG	71.03 °C
50494311	VDJ_IGHV_gDNA_22 Illuover	F10	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GAC TCC GTG AAG GGC AG	71.03 °C
50494312	VDJ_IGHV_gDNA_23 Illuover	F11	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GAC TCT GTG AAG GGC CG	71.03 °C
50494313	VDJ_IGHV_gDNA_24 Illuover	F12	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GCA AAC TCT GTG AAG GGC AG	70.65 °C
50494322	VDJ_IGHV_gDNA_25 Illuover	G1	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GCA CAG AAG TTT CAG GGC AG	70.65 °C
50494323	VDJ_IGHV_gDNA_26	G2	ACA CTC TTT CCC TAC ACG ACG	71.50 °C

	Illuover		CTC TTC CGA TCT GCA CCC GTG AAA GGC AG	
50494324	VDJ_IGHV_gDNA_27 Illuover	G3	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GCC CAT CTC TGA AGA GCA G	70.51 °C
50494325	VDJ_IGHV_gDNA_28 Illuover	G4	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GCG TCG GTG AAA GGC AG	71.35 °C
50494326	VDJ_IGHV_gDNA_29 Illuover	G5	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CTC CGT GAA GCG CCG	71.68 °C
50494327	VDJ_IGHV_gDNA_30 Illuover	G6	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT TGC GTC TGT GAA AGG CAG	70.40 °C

Table 15.

Extract genomic DNA

[0448] Extracted blood. PBMC extraction (ficoll), froze some in DMOS for eventual live pull down. Frozen cell stocks were used (3 vial, one for day 0 – day 7 and day 21 total cell 5.5 millions 70% viable). Extracted DNA (Agencourt kit; used 10x lysis mix + 50µl RNase cocktail; used bead binding at 50% volume, i.e., 300 µl of bead for 600 µl of solution). Nanodrop: 62.2 ng/µl (260/180 1.90, 280/230 2.0) total ~18 µg.

Primer setup

[0449] Made a dilute plate at 5µM each; 2µl of each +78µl of H₂O, but also to pooled tubes at 5µM once pooled. V; 3µl of each (=90µl total) + 30µl H₂O = 120µl at 5µM. J; 2µl of each (=6µl total) + 74µl H₂O = 80µl at 5µM

Individual primer testing

[0450] Tested every primer one by one (non-Illumina overlap only): each V against a pool of all 3 J, (30 PCR reaction); each J against a pool of all V (3 PCR reaction).

Assembled a reaction as follows:

	<u>1 tube</u>	<u>5t</u>	<u>35t</u>
dH ₂ O	9.8µl (to 20µl)	49	343
62.2 ng/µl DNA	1.60 µl 100 ng	8	56
5x HF buffer	4 µl 1X	20	140
10 mM dNTP	0.4 µl 0.2 µM	2	14
<u>5 µM primer v</u>	2 µl 0.5 µM	10	-
<u>5 µM primer J</u>	2 µl 0.5 µM	-	70
phusion hot start	0.2 µl 0.02 U/µl	1	7
	(18 µl /tubes)		

[0451] Thermal cycled as follows:

- 1- 98°C for 1 min
- 2- 98°C for 10 seconds
- 3- 55°C for 30 seconds
- 4- 72°C for 30 seconds go to step 2, 30x
- 5- 72°C for 5 min
- 6- 4°C pause

[0452] Added 5µl of 5X novex buffer and load 6µl on agarose gels (couldn't see anything). Ran 15µl on 6% TBE, stain SYBR gold 12 min (Figure 58)

Primer setup part 2

[0453] Plate (VDJ_gDNA_April02_10) was diluted at 400µM. Diluted plates were made at 5µM each: 2µl for each V and J +158µl of H₂O =160µl, but also to pooled tubes at 5µM once pooled. V; 2µl of each (=90µl total) + 70µl H₂O = 160µl at 5µM (well F1). J; 2µl of each (=6µl total) + 154µl H₂O = 160µl at 5µM (well F2).

Individual primer testing

[0454] Tested every primer one by one (non-Illumina overlap only): each V against a pool of all 3 J, (45 PCR reaction); each J against a pool of all V (3 PCR reaction); and all V and J in one reaction (1 PCR reaction) (Figure 59).

[0455] Assembled a reaction as follows:

	1 tube	5t (J and all)	50t
• dH ₂ O	9.8 µl (to 20 µl)	49	490
• 62.2 ng/µl DNA	1.60 µl 100 ng	8	80
• 5x HF buffer	4 µl 1X	20	140
• 10 mM dNTP	0.4 µl 0.2 µM	2	20
• 5 µM primer V	2 µl 0.5 µM	10	-
• 5 µM primer J	2 µl 0.5 µM	-(10 all)	100
• phusion hot start	0.2 µl 0.02 U/µl (18 µl /tubes)	1	10

[0456] Thermal cycling was as follows:

- 1- 98°C for 1 min
- 2- 98°C for 10 seconds
- 3- 58°C for 20 seconds
- 4- 72°C for 30 seconds go to step 2, 30x
- 5- 72°C for 5 min
- 6- 4°C pause

[0457] With Illumina overhang (Table 16).

Reference	Pos	Sequence Name	Sequence	T _m (50mM NaCl) C
51403096	A1	Il_VDJ_IGHV_gDNA_trunc135_01	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CCT GGA GTG GAT GGG GAT CA	71.5
51403097	A2	Il_VDJ_IGHV_gDNA_trunc135_10	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GGC TGG AGT GGG TCT CAG TT	71.7
51403098	A3	Il_VDJ_IGHV_gDNA_trunc135_11	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGG GCT TGA GTG GAT GGG AC	70.9
51403099	A4	Il_VDJ_IGHV_gDNA_trunc135_12	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGA GCT TGG GTG GAT GGG AC	70.9
51403100	A5	Il_VDJ_IGHV_gDNA_trunc135_02	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AAG GGG CTG GAG TGG GTT T	71.4
51403101	A6	Il_VDJ_IGHV_gDNA_trunc135_03	ACA CTC TTT CCC TAC ACG	70.7

			ACG CTC TTC CGA TCT CCT GGA GTG GAT TGG GTA CAT CT	
51403102	A7	II_VDJ_IGHV_gDNA_trunc135_04	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GGA AGG GGC TGG AGT GGA TT	71.8
51403103	A8	II_VDJ_IGHV_gDNA_trunc135_05	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGG GTC TGG AGT GGG TCT CA	71.1
51403104	A9	II_VDJ_IGHV_gDNA_trunc135_06	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CTG GAG TGG CTT GCA CAC A	71.5
51403105	A10	II_VDJ_IGHV_gDNA_trunc135_07	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CTG GAG TGG ATG GGG AGG ATT	71.2
51403106	A11	II_VDJ_IGHV_gDNA_trunc135_08	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AAA GGC CCT GGA GTG GCT T	71.4
51403107	A12	II_VDJ_IGHV_gDNA_trunc135_09	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CCT GGA GTG GCT TGC ACT CA	71.8
51403108	B1	II_VDJ_IGHV_gDNA_trunc135_13	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGG GCT TGA GTG GAT GGG AG	70.8
51403109	B2	II_VDJ_IGHV_gDNA_trunc135_22	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CAG GGA AGG GAC TGG AAT ATG TTT C	69.8
51403110	B3	II_VDJ_IGHV_gDNA_trunc135_23	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GGA AAG GGC TGG AGT GGG TT	72.0
51403111	B4	II_VDJ_IGHV_gDNA_trunc135_24	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GAG TGG GTC TCT CTT ATT AGT TGG GA	69.8
51403112	B5	II_VDJ_IGHV_gDNA_trunc135_14	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GCT GGA GTG GGT CTC ATC CA	71.6
51403113	B6	II_VDJ_IGHV_gDNA_trunc135_15	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GGG CTG GAG TGG GTA GGT TT	71.8
51403114	B7	II_VDJ_IGHV_gDNA_trunc135_16	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GTT GGC CGT ACT AGA AAC AAA GCT	70.2
51403115	B8	II_VDJ_IGHV_gDNA_trunc135_17	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CTG GAG TGG GTA TCG GGT GT	71.4
51403116	B9	II_VDJ_IGHV_gDNA_trunc135_18	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GAG TGG ATG GGA TTG GTG TGC	71.0
51403117	B10	II_VDJ_IGHV_gDNA_trunc135_19	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CTT GAG TGG ATG GGA GGT TTT GAT C	69.9
51403118	B11	II_VDJ_IGHV_gDNA_trunc135_20	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGG CTT GAG TGG ATG GGA TGG	70.7

51403119	B12	II_VDJ_IGHV_gDNA_trunc135_21	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CCT GGA GTG GCT TGC TCA CA	71.8
51403120	C1	II_VDJ_IGHV_gDNA_trunc135_25	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGT GGG TGG CAG TTA TAT GGT ATG A	69.5
51403121	C2	II_VDJ_IGHV_gDNA_trunc135_34	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT TGA GTG GAT AGG ATG GAT CGT CG	69.9
51403122	C3	II_VDJ_IGHV_gDNA_trunc135_35	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GGG TTG GCC GTA TTA AAA GCA AAA C	70.0
51403123	C4	II_VDJ_IGHV_gDNA_trunc135_36	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT ACT GGA GTG GAT TGG GTA CAT CTA TT	69.3
51403124	C5	II_VDJ_IGHV_gDNA_trunc135_26	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGG GCT TGA GTG GAT GGG AA	70.9
51403125	C6	II_VDJ_IGHV_gDNA_trunc135_27	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CTG GAG TGG ATT GGG TAC ATC TAT CA	69.9
51403126	C7	II_VDJ_IGHV_gDNA_trunc135_28	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CGC TTG AGT GGA TGG GAT GG	71.1
51403127	C8	II_VDJ_IGHV_gDNA_trunc135_29	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GGG TGG CCA ACA TAA AGC AAG A	71.0
51403128	C9	II_VDJ_IGHV_gDNA_trunc135_30	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GGG ACT GGA GTG GAT TGG GT	71.7
51403129	C10	II_VDJ_IGHV_gDNA_trunc135_31	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GCT AGA GTG GGT GGC AGT TAT ATC A	70.2
51403130	C11	II_VDJ_IGHV_gDNA_trunc135_32	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT CTT GAG TGG ATG GGA TGG ATG AAC	70.1
51403131	C12	II_VDJ_IGHV_gDNA_trunc135_33	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGT GGG TCT CAG CTA TTA GTG GTA G	69.2
51403132	D1	II_VDJ_IGHV_gDNA_trunc135_37	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GCT GGA GTG GGT CTC TGG TA	71.4
51403133	D2	II_VDJ_IGHV_gDNA_trunc135_38	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GCT GGT GTG GGT CTC ACG TA	71.5
51403134	D3	II_VDJ_IGHV_gDNA_trunc135_39	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GAG TGG GTC TCA GGT ATT AGT TGG A	70.1
51403135	D4	II_VDJ_IGHV_gDNA_trunc135_40	ACA CTC TTT CCC TAC ACG	71.1

			ACG CTC TTC CGA TCT CAG GAA AAG GTC TGG AGT GGG T	
51403136	D5	II_VDJ_IGHV_gDNA_trunc135_41	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT TGG AGT GGG TGG CAG TTA TAT CA	70.2
51403137	D6	II_VDJ_IGHV_gDNA_trunc135_42	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT GGG ACT GGA GTG GGT TTC ATA CA	70.9
51403138	D7	II_VDJ_IGHV_gDNA_trunc135_43	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGG GCT TGA GTG GAT GGG ATG	70.7
51403139	D8	II_VDJ_IGHV_gDNA_trunc135_44	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT AGG GAA AGG GCT AGA GTT GGT AG	69.8
51403140	D9	II_VDJ_IGHV_gDNA_trunc135_45	ACA CTC TTT CCC TAC ACG ACG CTC TTC CGA TCT TCC CCA TCG AGA GGC CTT GA	71.6
51403141	E1	II_VDJ_IGHJ_gDNA_01	CTC GGC ATT CCT GCT GAA CCG CTC TTC CGA TCT CTT ACC TGA GGA GAC GGT GAC C	71.9
51403142	E2	II_VDJ_IGHJ_gDNA_02	CTC GGC ATT CCT GCT GAA CCG CTC TTC CGA TCT CTC ACC TGA GGA GAC AGT GAC C	71.9
51403143	E3	II_VDJ_IGHJ_gDNA_03	CTC GGC ATT CCT GCT GAA CCG CTC TTC CGA TCT CTT ACC TGA AGA GAC GGT GAC C	71.3

Table 16.

Primer setup part 2

[0458] Plate (VDJ_gDNA_April23_10_illumina) was diluted at 400µM. Plates were supplied at 10 nM, resuspend all oligos in 25 µl each. They were not tested one by one, instead overall band intensity was tested, and also to pooled tubes at 5 µM once pooled. V; 2 µl of each (=90 µl total) + 70 µl H₂O = 160 µl at 5 µM (well F1). J; 2 µl of each (=6 µl total) + 154 µl H₂O = 160 µl at 5 µM (well F2)

[0459] combined Illumina primer overlap testing

[0460] Assembled a reaction as follows:

	<u>1 tube</u>	<u>4 tubes</u>
• dH ₂ O	14.42 µl (to 25 µl)	57.68
• 180 ng/µl DNA	0.83 µl 150 ng	3.33
• 5x HF buffer	5 µl 1X	20
• 10 mM dNTP	0.5 µl 0.2 µM	2
• <u>5 µM</u> primer V	2 µl 0.5 µM	8
• <u>5 µM</u> primer J	2 µl 0.5 µM	8
• phusion hot start	0.25 µl 0.02 U/µl (25 µl /tubes)	1

[0461] Thermal cycle was performed as follows:

- 1- 98°C for 1 min
- 2- 98°C for 10 seconds
- 3- 58°C for 20 seconds
- 4- 72°C for 20 seconds go to step 2, 30x
- 5- 72°C for 5 min
- 6- 4°C pause

[0462] Qiagen purified, eluted in 30µl. Nanodrop

1	70 ng/µl	no overhang
2	128 ng/µl	Illumina overhang
3	90 ng/µl	Illumina overhang, Veraseq

[0463] To 6 µl, added 6 µl of 2X blue juice and loaded 6 µl. Ran 2x 6 µl on 6% TBE (20 min), stained half with ethidium bromide (EtBR) and and SYBR gold 12 min. Inversed loading (so number 3 is Phusion with primer no-overhang) (Figure 60).

[0464] Assembled a reaction as follows:

	<u>1 tube</u>	<u>5t</u>
• dH ₂ O	10.85 µl (to 20 µl)	54.25
• 180 ng/µl DNA	0.55 µl 100 ng	-
• 5x HF buffer	4 µl 1X	20
• 10 mM dNTP	0.4 µl 0.2 µM	2
• <u>5 µM</u> primer V	2 µl 0.5 µM	-
• <u>5 µM</u> primer J	2 µl 0.5 µM	-
• phusion hot start	0.2 µl 0.02 U/µl (15.45µl /tubes)	1

[0465] Thermal cycling performed as follows (Figure 61):

- 1- 98°C for 1 min
- 2- 98°C for 10 seconds
- 3- 55°C for 30 seconds
- 4- 72°C for 30 seconds go to step 2, 30x
- 5- 72°C for 5 min
- 6- 4°C pause

[0466] Repeated with cDNA of 62 ng/μl and coriel DNA (dilute 1:5). Tried from FV1 cDNA (RT-PCR done as in heavy chain 454 sequencing).

[0467] Assembled a reaction as follow:

	<u>1 tube</u>
• dH2O	6.4 μl (to 20 μl)
• cDNA	5 μl 100 ng
• 5x HF buffer	4 μl 1X
• 10 mM dNTP	0.4 μl 0.2 μM
• <u>5 μM</u> primer V	2 μl 0.5 μM
• <u>5 μM</u> primer J	2 μl 0.5 μM
• phusion hot start	0.2 μl 0.02 U/μl (18 μl /tubes)

[0468] Thermal cycling performed as follows:

- 1- 98°C for 1 min
- 2- 98°C for 10 seconds
- 3- 58°C for 20 seconds
- 4- 72°C for 20 seconds go to step 2, 25x
- 5- 72°C for 5 min
- 6- 4°C pause

Loaded 5μl on 6% TBE gel, ran 25 min, stained with 10 μl EtBr 15 min (Figure 62).

EXAMPLE XII**Rheumatoid Arthritis Immune Sequencing**

[0469] Sample information is set forth in Table 17.

CCP+ RA															
Specimen ID	Sp type	Sp Des	Diagnosis	Gender	Age	RF	RF date	RF highest	RF highest date	CCP	CCP date	CCP highest	CCP highest date	CRP	CRP date
DL-SF-0306	S F	knee	RA	F	6 3	48	6/2/2011	59	8/4/2008	> 1 0 0	6/2/2011	>10 0	6/2/2011	11.6	6/2/2011
DL-SF-0331	S F	knee	RA	F	5 9	17 3	9/14/2009	17 3	9/14/2009	> 1 0 0	9/14/2009	>10 0	9/14/2009		
DL-SF-0263	S F	knee	Psoriatic	M	2 6	48	4/3/2009	48	3/12/2009	> 1 0 0	3/12/2009	>10 0	3/12/2009		
DL-SF-0375	S F	knee	RA + Sec OA	M	6 0	15 4	10/3/2007	23 3	5/2/2004	> 1 0 0	10/3/2007	>10 0	10/3/2007	82.6	10/3/2007
Non-RA															
Specimen ID	Sp type	Sp Des	Diagnosis	Gender	Age	RF	RF date	RF highest	RF highest date	CCP	CCP date	CCP highest	CCP highest date	CRP	CRP date
DL-SF-0291	S F	knee	OA	M	0										
DL-SF-0237	S F	knee	OA	M	0										
DL-SF-0313	S F	knee	Gout	F	7 3										
DL-SF-0264	S F	knee	OA	F	7 7										

Table 17.

RNA extraction

[0470] Used mirvana as per manufacturer protocol. QC nanodrop. Diluted all to 250 ng/μl (Table 18).

ID	ng/ul RNA	260/280	260/230	1/3 dilution	Volume for 500ul	Water to 7
1	768.6	2.06	1.4	260.6	1.92	5.08
2	644.7	2.09	2.05	213.1	2.35	4.65
3	349.3	2.07	2.05	112.0	4.46	2.54
4	181.7	2.04	1.7	-	2.75	4.25
5	434.6	2.1	1.11	141.6	3.53	3.47
6	1071.3	2.09	1.59	360.4	1.39	5.61
7	597	2.04	1.82	199.2	2.51	4.49
8	18.7 /134.0	1.88	1.13	-	3.73	3.27

Table 18.

[0471] Concentrated sample 8 by EtOH precipitation, reuspended in 20 ml, nanodrop=134.0.

Reverse transcription. Proceeded as follows for RT-PCR:

2x 1 tube

- H₂O (DEPC) 0 µl (to 12 µl)
- 2 µM Gene-specific primer (IGHC K and L) 5 (5 pmole)
- Total RNA 7 µl (500 ng)

Heated at 95 °C for 1 min. followed by 65 °C for 5 min., than ice for 1 min. Spun down and added the following:

	1 tube	10 tubes
• 5x First strand buffer	4µl	40
• 10 mM dNTP mix	1µl	10
• 0.1 M DTT	1µl	10
• RNase Inhibitor-Enzy	1µl	10
• Superscript III	1µl	10

[0472] Incubated at 55°C for 60 min. (8µl/tubes). Inactivated enzyme by heating at 70°C for 15 min. Removed RNA/DNA hybrid by adding 1 µl of *E. coli* RNaseH. Incubated at 37°C for 20 min., then ice. Assembled PCR reaction as follows (used half for PCR, kept other half as backup):

	1x	x2per sample x 12samples (16t)
• dH ₂ O	21µl (to 50 µl)	420
• cDNA	5 µl	-

• 5x HF buffer	10 µl	240
• dNTP(10 mM)	1 µl	24
• primer up (IGHV new-short) (4 µM)	6.25 µl (25 pmole)	150
• primer low (IGHC new-short)(4 µM)	6.25 µl (25 pmole)	150
• phusion	0.5 µl	12

[0473] Added 45 µl to 5 µl twice for each cDNA sample. Thermal cycled as follows:

- 1- 98 °C for 1 min.
- 2- 98 °C for 10 seconds
- 3- 62 °C for 20 seconds
- 4- 72 °C for 20 seconds went to step 2, 23x
- 5- 72 °C for 5 min.
- 6- 4 °C pause

[0474] Pooled both tubes together (total 100 µl). Added 2 µl of Exonuclease I (20U/µl) to each 100 µl tube and incubated at 37°C for 20 min. AMPure XP purified, ratio 1.8:1. Resuspended in 40 µl, transferred to PCR strip. Conducted SPRI purification as follows:

- Added the require amount of AMPure XP beads (1.8:1 ratio) to the DNA sample in buffer EB.
- Vortexed to mix.
- Incubated for 5 minutes at room temperature.
- Magnet (MPC) for 5 minutes. Left the tube of beads in the MPC during all wash steps.
- Removed the supernatant (kept in case of failure) and washed the beads twice with 500 µl of 70% ethanol, incubating for 30 sec. each time.
- Removed all the supernatant, quick spun, removed last drop and allowed the AMPure beads to air dry completely (2 min.).
- Removed the tube from the MPC, added 40 µl of 10 mM Tris-HCl, 0.1 mM EDTA, pH 8.0 (or Qiagen's Buffer EB), and vortexed for 30 sec. to resuspend the beads, let sit for 3 min. Magnet for 2 min. and transfer supernatant to a new tube.

[0475] A diagnostic gel was used to check PCR efficiency. 2 μ l was run on a 2% Egel-X for 12 min. (Figure 68). Blunting reactions were performed using Enzymatics End repair kit as follows:

	1t	12t
• H ₂ O	0 μ l(to 50 μ l)	0
• Purified DNA	38 μ l	-
• 10 $\times$ End repair buffer	5 μ l	60
• 1 mM dNTP mix	5 μ l	60
• enzyme mix.	5 μ l	60

[0476] Incubated at 25 °C temperature for 30 min. Heat inactivated at 75 °C for 20 min. AMPure XP purified, ratio 1.8:1. Resuspended in 37 μ l (kept non-bind volume as backup). Saved 2 μ l of each sample for gel diagnostic. A diagnostic gel was run to check efficiency of blunting and ampure beads. 2 μ l of sample was run on a 2% Egel-X for 12 min. (Figure 69). AMPure purified to remove all dNTP, eluted in 44 μ l. E Gel diagnostic 2 μ l. Tested the following steps on one sample only: (sample 10 FV1, the one with the extra band at 275 bp).

A-tailing – FV1 sample only

[0477] Klenow exo- “A” and “T” tailing. The following reaction mix was prepared:

• H ₂ O	42 μ l (to 50 μ l)
• DNA sample	41 μ l
• 10X Klenow buffer	5 μ l
• 10 mM dATP	1 μ l
• Klenow exo (3' to 5' exo minus)	3 μ l

[0478] Incubated for 30 minutes at 37 °C. AMPure purified, eluted in 26 μ l. Ran on diagnostic gel (Figure 70). Ln1 post A-tail, ln2, second round of AMPure of supernatant non-bind post end it. Everything looked fine. Without intending to be bound by scientific theory,

the previous gel bands of FV1 (samples 9 and 10), were probably faint from bad gel loading.

Y Adaptor ligation

[0479] Self annealed 454 oligo into adapter A and B together. Followed 454 protocol (10 µl of 100 µM of each primer + 30 µl of (10 mM Tris, 0.1 mM EDTA at 50 mM NaCl)) (each adapter was at 20µM final), 95 °C for 3 min., ramp to 15 °C forever at 0.1 °C/sec.

[0480] Ligation of rapid 454 Y adapter as follows (kept 2 µl of DNA on the side for gel comparison):

	1 tube
• H ₂ O	0 µl (to 50 µl)
• 2x Quick ligase buffer	25 µl
• VDJ DNA	23 µl
• 20 µM adapter	1 µl
• T4 DNA ligase (quick)	2 µl

[0481] Incubated on PCR at 22°C for 15 min. AMPure XP purified, ratio 1.8:1. Resuspended in 25 µl. Ran sample on a diagnostic gel.

[0482] Tested all combined reactions, like in the 454 rapid protocol. Used sample 9 (FV1 for this), pretended endit was not done yet (cause need buffer in the mixture). Performed blunting reaction using Enzymatics End repair kit as follows (simultaneous Endit and A-tailing):

	1 tube
• H ₂ O	0 µl(to 50 µl)
• Purified DNA	40 µl
• 10× End repair buffer	5 µl
• 1 mM dNTP mix	5 µl
• enzyme mix.	2 µl (T4 pol + PNK)
• TAQ polymerase	2 µl

[0483] Incubated at 25 °C temperature for 20 min., then 72 °C for 20 min., then 4°C hold. Y adapter ligation, add directly to the reaction as follows:

- Endit A-tailed DNA 50 µl
- 2x Quick ligase buffer 50 µl
- 20 µM adapter 1 µl
- T4 DNA ligase (quick) 2 µl

[0484] Incubated on PCR at 22 °C for 15 min. (total volume was at 107 µl) so 1.8 ampure = 193 µl. AMPure XP purified, ratio 1.8:1. Resuspended in 25 µl. Performed diagnostic gel on sample (Figure 71), 2% egel, ran 16 min. Lane 1, marker; lane 2, pre-ligation control; lane 3, Klenow A tail; lane 4, Taq A tail.

[0485] The extra bands in lane 2 were the RNase H non-specific bands that got ligated. Now the question was to know if ligation will show one Y adapter vs 2 Y adapters. Primer A 41 nt, primer B 43 nt = total 84. Without intending to be bound by scientific theory, this would mean that the first band would be proper ligation and the other one above would be something else because the final product is 100 bp or more, unless the FAM is causing this migration pattern. But ideally a phopho primer set, follow in PCR by a cycle of TAQ, would probably be the way to go... but phusion would compete for A tail of taq. Test ligation of Y adapter on forever ladder test a few sizes. (no a-tailing concern here). Then test Taq A tailing sequential vs. mixed with the blunt ending.

PCR test

[0486] Used primers (Table 19) that amplified the final product, so only double adapter ligation should get amplified exponentially. Started with a 1/10 dillution, did PCR of 15 cycles and see what happen on gel.

Reference	Purification	Sequence Name	Sequence	Tm
53164334	Standard Desalting	454 rapid top PCR	CCA TCT CAT CCC TGC GTG TCT CC	61.4

53164335	Standard Desalting	454 rapid low PCR	CCT ATC CCC TGT GTG CCT TGA GAG	60.9

Table 19.

[0487] Assembles a PCR test reaction as follows:

	<u>1x</u>	<u>5x</u>
• dH ₂ O	15.75 µl (to 25 µl)	78.75
• 1/10 dilution of DNA	1 µl	-
• 5x HF buffer	5 µl	25
• dNTP(10 mM)	0.5 µl	2.5
• primer up 10 µM	1.25	6.25
• primer low 10 µM	1.25	6.25
• phusion	0.25 µl	1.25

[0488] Thermal cycled as follows:

- 1- 98 °C for 1 min.
- 2- 98 °C for 10 seconds
- 3- 65 °C for 20 seconds
- 4- 72 °C for 20 seconds went to step 2, 15x
- 5- 72 °C for 5 min.
- 6- 4° C pause

[0489] Loaded 2 µl and 20 µl samples on 2% ex gel and ran for 16 min. (Figure 72). Sample 1, pre-ligation control; sample 2, Klenow A tail; sample 3, Taq A tail; sample 4, no template PCR control. Without intending to be bound by scientific theory, it appeared that the upper band was the correct one, the absence of FAM there would make it migrate slightly lower. An extraction would confirm once and for all, but the resulting enrichment looked very nice. (Re-ordered the primer with a 5' FAM on top primer and both HPLC purified.)

[0490] Performed a pippin prep extraction test. Band extraction pattern was as follows:

1	475	450-500
2	500	475-525 nicely done
3	538	500-575

4 500 460-540 100 bp ladder (7 µl pippin) extract 500 bp

[0491] Loaded 20 µl of resulting elution chamber (60-80 µl) on 2% exgel (Figure 73. Left panel pre-pippin; right panel, post-pippin. Lane 1, control FV-µl VDJ, HC purified. Validated PCR extraction. 1. negative control FV-µl (used 1 µl). 2. 475-525 (hypo :one Y adapter only). 3. 500-550 (hypo : 2 Y adapter).

[0492] Assembled a PCR test reaction as follow

	1x	4
• dH ₂ O	11.7 5µl (to 25 µl)	47
• pipin elution dilution of DNA	5 µl	-
• 5x HF buffer	5 µl	20
• dNTP(10 mM)	0.5 µl	2
• primer up 10 µM	1.25	5
• primer low 10 µM	1.25	5
• phusion	0.25 µl	1

[0493] Thermal cycled as follows (20 µl / tube):

- 1- 98 °C for 1 min.
- 2- 98 °C for 10 seconds
- 3- 65 °C for 20 seconds
- 4- 72 °C for 20 seconds went to step 2, 15x
- 5- 72 °C for 5 min.
- 6- 4 °C pause

[0494] Loaded 2 µl and 20 µl samples on an ex gel, ran 16 min. (Figure 74).

Optimize A-tailing

[0495] Tried adding Klenow or Taq right after blunt ending (no cleanup), with an excess of dATP and fresh buffer. Then combined with or without clenup prior ligaiton. Used post-ligation sample for initial testing. Two good-sized bands were obtained to evaluate efficiency: 1, klenow- lig; 2, klenow- clean+lig; 3, Taq lig; 4, Taq clean+lig; 5, bst largefrag lig; 6, bst largefrag clean+lig.

[0496] Used one of FV1 heavy chain, post PCR, post cleanup. Split reaction in 6 for testing. Performed blunting reaction using Enzymatics End repair kit as follows (use only half of PCR product maybe?):

	1t	<u>8</u>
• H ₂ O	0 µl(to 25 µl)	104
• Purified DNA	19µl (6 DNA + 13 water)	-
• 10× End repair buffer	2.5 µl	20
• 1 mM dNTP mix	2.5 µl (0.1 mM)	20
• enzyme mix (HC)	1 µl	8

[0497] Incubated at 25 °C temperature for 30 min. (19/t). Heat inactivated at 75 °C for 20 min. A-tailed by adding the following directly to the mixture (also did with TAQ and BST with proper buffers) :

	<u>1t</u>	<u>2t</u>
• H ₂ O	20 µl(add 25 to 50 µl)	40
• Blunt DNA sample	25 µl	50
• 10X Klenow buffer	2.5 µl	5
• 10 mM dATP	0.5 µl	1
• Klenow exo HC (3' to 5' exo minus)	2 µl	4

[0498] Total reaction was now 50 µl. Klenow, incubated at 37 °C for 30 min., heat inactivated at 75 °C for 20 min. Taq, incubated at 72 °C for 30 min. BST, incubated at 65 °C for 30 min., heat inactivated at 80 °C for 10 min. Split each reaction in 2 tubes of 50. Apure XP purified half of each reaction, eluted in 50ul. The other half was run directly in the ligaiton.

[0499] Y adapter ligation. The following were add directly to the reaction:

	1t	7t
• H ₂ O	0 (add µl to 100 µl)	-
• A-tailed DNA	50 µl	-
• 2x Quick ligase buffer	46 µl	322
• 20 µM adapter	2 µl	14
• T4 DNA ligase (quick)	2 µl	14

[0500] Incubated on PCR at 22°C for 15 min. Total reaction was then 100 µl. Ampure, 1.8 ratio. Eluted in 25 µl (Figure 75).

Pippin prep test 2 (loading quantity)

[0501] Pooled all samples. Nanodrop : 25ng/µl ratio were way high (FAM?). Load on pippin prep at different quantity. Kept some to run as negative control. Extract range 475 to 550. Loaded on pippin nanodrop post-pippin. Sample 1, 2.5 µl; sample 2, 5; sample 3, 10; sample 4, 20; sample 5, ref. Ampure XP, eluted in 20 µl. Nanodrop all 0. Loaded 5 µl on 2% e-gel ex (Figure 76). Did not look like the correct size was extracted. Without intending to be bound by scientific theory, this may be due to the fact that 475-550 was used instead of 475-525.

Finalize library

[0502] The following protocol was designed to finalize remaining library. Plate was setup as follows (Table 20):

Library type (all heavy chains)	Row placement and BC info 1=13, etc.
Tripple VDJ-HC GMC	A1-A10
Tripple VDJ-HC IDO	B1-B10
Tripple VDJ-HC FV	C1-C10
RA	D1-D8
HIV 1/2	E1-E8 (5 progressor+5 negative) BC; 1-2-3-4-5/6-7-8-9-10
HIV 2/2	F1-F7 (5 elite) BC; 1-2-3-4-5

Table 20.

[0503] Perform blunting reaction using Enzymatics End repair kit as follows (use only half of PCR product maybe):

• H ₂ O	0 µl (to 25 µl)	0
• Post PCR Purified DNA	19µl	-
• 10× End repair buffer	2.5 µl	150
• 1 mM dNTP mix	2.5 µl (0.1 mM)	150
• enzyme mix (HC)	1 µl	60

[0504] Incubated at 25 °C temperature for 30 min. (19/t). Heat inactivated at 75 °C for 20 min.

A-tailed by adding the following directly to the mixture:

	<u>1t</u>	<u>60t</u>
• H ₂ O	20 µl(to 50 µl)	1200
• Blunt DNA sample	25 µl	-
• 10X Klenow buffer	2.5 µl	150
• 10 mM dATP	0.5 µl	30
• Klenow exo HC (3' to 5' exo minus)	2 µl	120

[0505] Total reaction was then 50 µl (25/t). Incubated at 37 °C for 30 min. Heat inactivated at 75 °C for 20 min.

[0506] Prepared 454 Y-adapters. 10 µl of 100 µM of each primer A and B + 30 µl of (10 mM Tris, 0.1 mM EDTA at 50 mM NaCl)) (each adapter was at 20 µM final). Incubated at 95 °C for 3 min., ramp to 15 °C forever at 0.1 °C/sec.

[0507] Y adapter ligation. The following was added directly to the reaction:

	<u>1t</u>	<u>2x30t</u>
• H ₂ O	0 (add µl to 100 µl)	-
• A-tailed DNA	50 µl	-
• 2x Quick ligase buffer	46 µl	1380
• 20 µM adapter	2 µl	-
• T4 DNA ligase (quick)	2 µl	60

[0508] Incubated on PCR at 22 °C for 15 min. (48/t). Total reaction was then 100 µl. Ampured, eluted in 40 µl (used the liquidator). Library QC for pooling. Tested on HIV sample, tested only first 5 samples (progressor).

[0509] Appendix A sets forth ScFv primers. Appendix B sets forth methods of SOE-PCR and ScFV generation from single cells. Appendix C sets forth VDJome analysis methods.

ScFv Primers

From Notebook

Contents

- (for nested PCR strategy)
 - 1.1 Info
 - 1.2 Plates
 - 1.3 Mixing
 - _attB (Gateway cloning)
 - 2.1 Info
 - 2.2 Sequences
 - 2.3 Ordering Info
 - 2.4 Mixing Info
 - _scFv_USER and _USER
 - 3.1 Info
 - 3.2 Sequences
 - 3.3 Ordering Info
 - 3.4 Mixing Info
 - _SOE_PEDS_11_405_LLA4
 - 4.1 Info
 - 4.2 Sequences
 - 4.3 Plate information
 - 4.4 Mixing info
 - 5.1 Info
 - 5.2 Sequences
 - 5.3 Plate information
 - 5.4 Mixing info
 - _SOE_PEDS_11_405_LLA4
 - 6.1 Info
 - 6.2 Sequences
 - 6.3 Plate info
 - 6.4 Mixing info
 - _SOE_PNAS_85_5879_G4S3

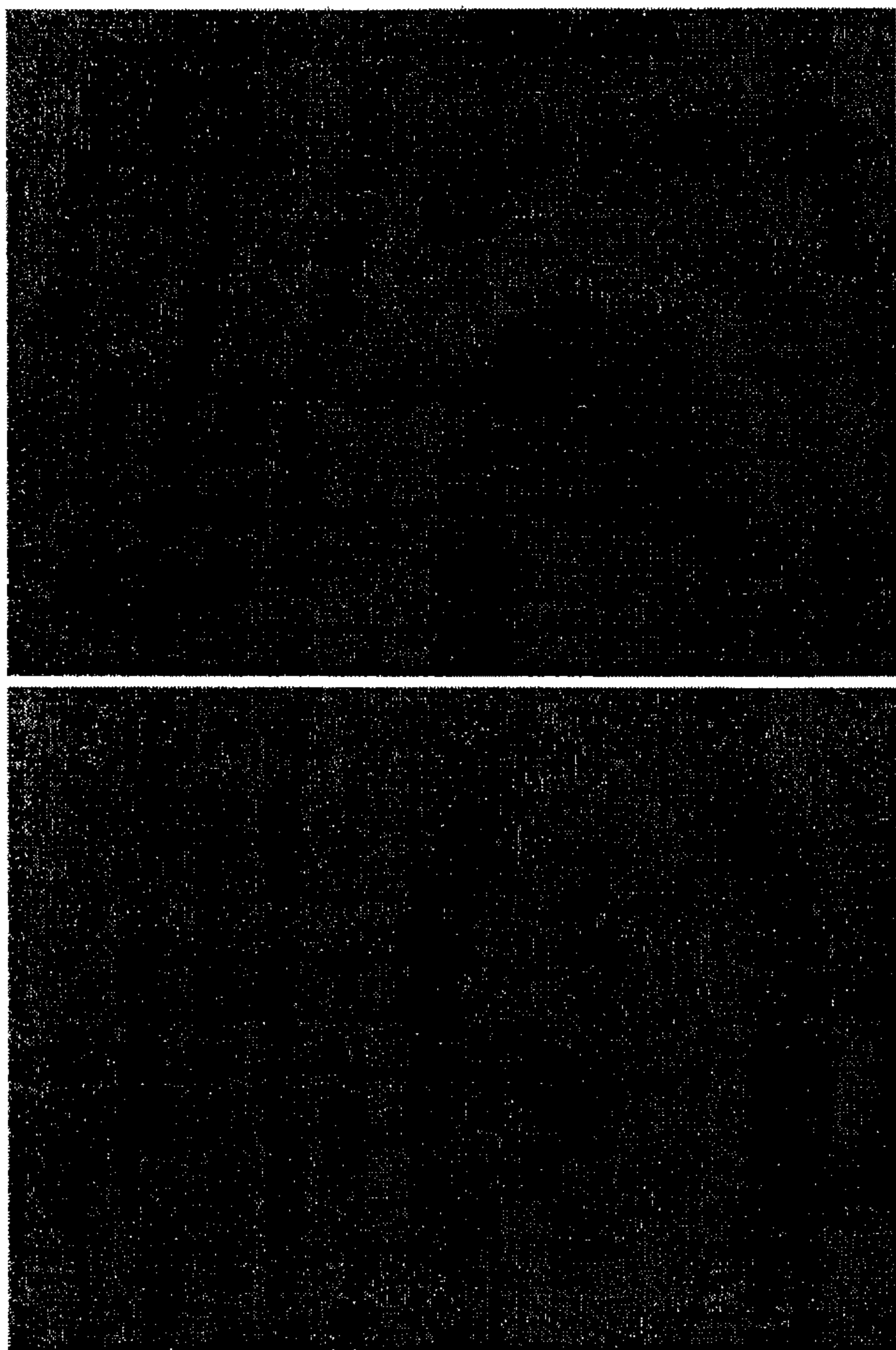
(for nested PCR strategy)

Info

- These are the same primers used for the original VDJ-ome project
- See VDJ-ome_Primers for more detail on their design

Plates

- These primers are ordered on two plates, arranged as:



- The excel files for IDT are:
 - Media: _left.xls
 - Media: _right_ _RT.xls
 - NOTE: This plate has only 16 primers, so IDT won't let me order it. I ordered it in separate tubes instead.

Mixing

- The plate are ordered at 200 μ M
- The individual tubes are ordered LabReady at 100 μ M
- For plates: To make 2.5 μ M each primer, add 1 μ L stock into 80 μ L total volume of each primer
- For tubes: To make 2.5 μ M each primer, add 2 μ L stock into 80 μ L total volume of each primer
- Note, the IGHV, IGK/LV, IGHC, IHC/LC are four different primer mixes.

bly

attB (Gateway cloning)

Info

- The primer design is on
- These are the "outer" primers of with attB tags added on for Gateway cloning.

Sequences

■ The _attB primer sequences are:

```

>20100505_IGHV_left_attB1_1
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGTGGAGTCCG
>20100505_IGHV_left_attB1_2
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_attB1_3
ggggacaagtttgtacaaaaaagcaggcttcCAGGTACACCTTGAGGGAGTCTGGTCC
>20100505_IGHV_left_attB1_4
ggggacaagtttgtacaaaaaagcaggcttcCAGGTTCAGCTGTTGCAGCCTGG
>20100505_IGHV_left_attB1_5
ggggacaagtttgtacaaaaaagcaggcttcCAGGTGCAGCTACAGCAGTGGGG
>20100505_IGHV_left_attB1_6
ggggacaagtttgtacaaaaaagcaggcttcCAGGTGCAGCTGGTGAATCTGG
>20100505_IGHV_left_attB1_7
ggggacaagtttgtacaaaaaagcaggcttcCAGGTACACCTGAAGGAGTCTGGTCC
>20100505_IGHV_left_attB1_8
ggggacaagtttgtacaaaaaagcaggcttcCAGGTCCAGCTGGTACAGTCTGGG
>20100505_IGHV_left_attB1_9
ggggacaagtttgtacaaaaaagcaggcttcCAGGACCAGTTGGTGCAGTCTGGG
>20100505_IGHV_left_attB1_10
ggggacaagtttgtacaaaaaagcaggcttcCAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_attB1_11
ggggacaagtttgtacaaaaaagcaggcttcCAGATGCAGCTGGTGCAGTCTGG
>20100505_IGHV_left_attB1_12
ggggacaagtttgtacaaaaaagcaggcttcCAAATGCAGCTGGTGCAGTCTGGG
>20100505_IGHV_left_attB1_13
ggggacaagtttgtacaaaaaagcaggcttcGAAGTGCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_attB1_14
ggggacaagtttgtacaaaaaagcaggcttcCAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_attB1_15
ggggacaagtttgtacaaaaaagcaggcttcGAGGATCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_attB1_16
ggggacaagtttgtacaaaaaagcaggcttcGAGGTCCAGCTGGTACAGTCTGGG
>20100505_IGHV_left_attB1_17
ggggacaagtttgtacaaaaaagcaggcttcCAGCTGCAGCTGCAGGAGTCC
>20100505_IGHV_left_attB1_18
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_attB1_19
ggggacaagtttgtacaaaaaagcaggcttcCAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_attB1_20
ggggacaagtttgtacaaaaaagcaggcttcCAGGTACAGCTGCAGCAGTCAGGT
>20100505_IGHV_left_attB1_21
ggggacaagtttgtacaaaaaagcaggcttcGAGATGCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_attB1_22
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCATCTGGTGGAGTCTGGG
>20100505_IGHV_left_attB1_23
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_attB1_24
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_attB1_25
ggggacaagtttgtacaaaaaagcaggcttcCAGGTCCAACCTGGTGTAGTCTGGAGC
>20100505_IGHV_left_attB1_26
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_attB1_27
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGTGGAGTCTCG
>20100505_IGHV_left_attB1_28
ggggacaagtttgtacaaaaaagcaggcttcGAGGTTCAGCTGGTGCAGTCTGGG
>20100505_IGHV_left_attB1_29
ggggacaagtttgtacaaaaaagcaggcttcCAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_attB1_30
ggggacaagtttgtacaaaaaagcaggcttcGAAGTGCAGCTGGTGCAGTCTGG
>20100505_IGHV_left_attB1_31
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGGAGCTGATAGAGTCCATAGA
>20100505_IGHV_left_attB1_32
ggggacaagtttgtacaaaaaagcaggcttcACAGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_attB1_33
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGAGGAGTCTGG
>20100505_IGHV_left_attB1_34
ggggacaagtttgtacaaaaaagcaggcttcGAGGTACAGCTGGTGGAGTCTGAAGA
>20100505_IGHV_left_attB1_35
ggggacaagtttgtacaaaaaagcaggcttcCAGGTGCAGCTGCAGGAGTCG
>20100505_IGHV_left_attB1_36
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGTTGGAGTCTGGG
>20100505_IGHV_left_attB1_37
ggggacaagtttgtacaaaaaagcaggcttcCAGGTGCAGCTGGGGCAGTC
>20100505_IGHV_left_attB1_38
ggggacaagtttgtacaaaaaagcaggcttcCAGCTGCAGCTGCAGGAGTCG
>20100505_IGHV_left_attB1_39
ggggacaagtttgtacaaaaaagcaggcttcCAGGTTCAGCTGGTGCAGTCTGGA
>20100505_IGHV_left_attB1_40

```


ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGGTAGAGTCTGGG
>20100505_IGHV_left_attB1_41
ggggacaagtttgtacaaaaaagcaggcttcCAGGTCCAGCTTGTGCAGTCTGGG
>20100505_IGHV_left_attB1_42
ggggacaagtttgtacaaaaaagcaggcttcGAGGTGCAGCTGTTGCAGTCTGC
>20100505_IGHV_left_attB1_43
ggggacaagtttgtacaaaaaagcaggcttcGAGGTACAACCTGGTGGAGTCTGGGGG
>20100505_IGHV_left_attB1_44
ggggacaagtttgtacaaaaaagcaggcttcCAGATCACCTTGAAGGAGTCTGGTCC
>20100505_IGHV_left_attB1_45
ggggacaagtttgtacaaaaaagcaggcttcCAGGTACAGCTGATGCAGTCTGGG

>20100505_IGKJ_right_attB2_1
ggggaccactttgtacaagaaagctgggtcTTTGATCTCCACCTTGGTCCCTCCGC
>20100505_IGKJ_right_attB2_2
ggggaccactttgtacaagaaagctgggtcTTTGATCTCCAGCTTGGTCCCTGG
>20100505_IGKJ_right_attB2_3
ggggaccactttgtacaagaaagctgggtcTTTGATATCCACTTGGTCCCAGGGC
>20100505_IGKJ_right_attB2_4
ggggaccactttgtacaagaaagctgggtcTTTGATTCCACCTTGGTCCCTTGGC
>20100505_IGKJ_right_attB2_5
ggggaccactttgtacaagaaagctgggtcTTTAATCTCCAGTCGTGTCCCTTGGC
>20100505_IGLJ_right_attB2_1
ggggaccactttgtacaagaaagctgggtcGAGGACGGTCACCTTGGTGCCA
>20100505_IGLJ_right_attB2_2
ggggaccactttgtacaagaaagctgggtcTAGGACGGTCAGCTTGGTCCCTCC
>20100505_IGLJ_right_attB2_3
ggggaccactttgtacaagaaagctgggtcGAGGACGGTCAGCTGGGTGCC
>20100505_IGLJ_right_attB2_4
ggggaccactttgtacaagaaagctgggtcTAAATGATCAGCTGGGTTCCTCCAC
>20100505_IGLJ_right_attB2_5
ggggaccactttgtacaagaaagctgggtcTAGGACGGTGACCTTGGTCCCAGT
>20100505_IGLJ_right_attB2_6
ggggaccactttgtacaagaaagctgggtcTAGGACGGTCAGCTCGGTCCCC

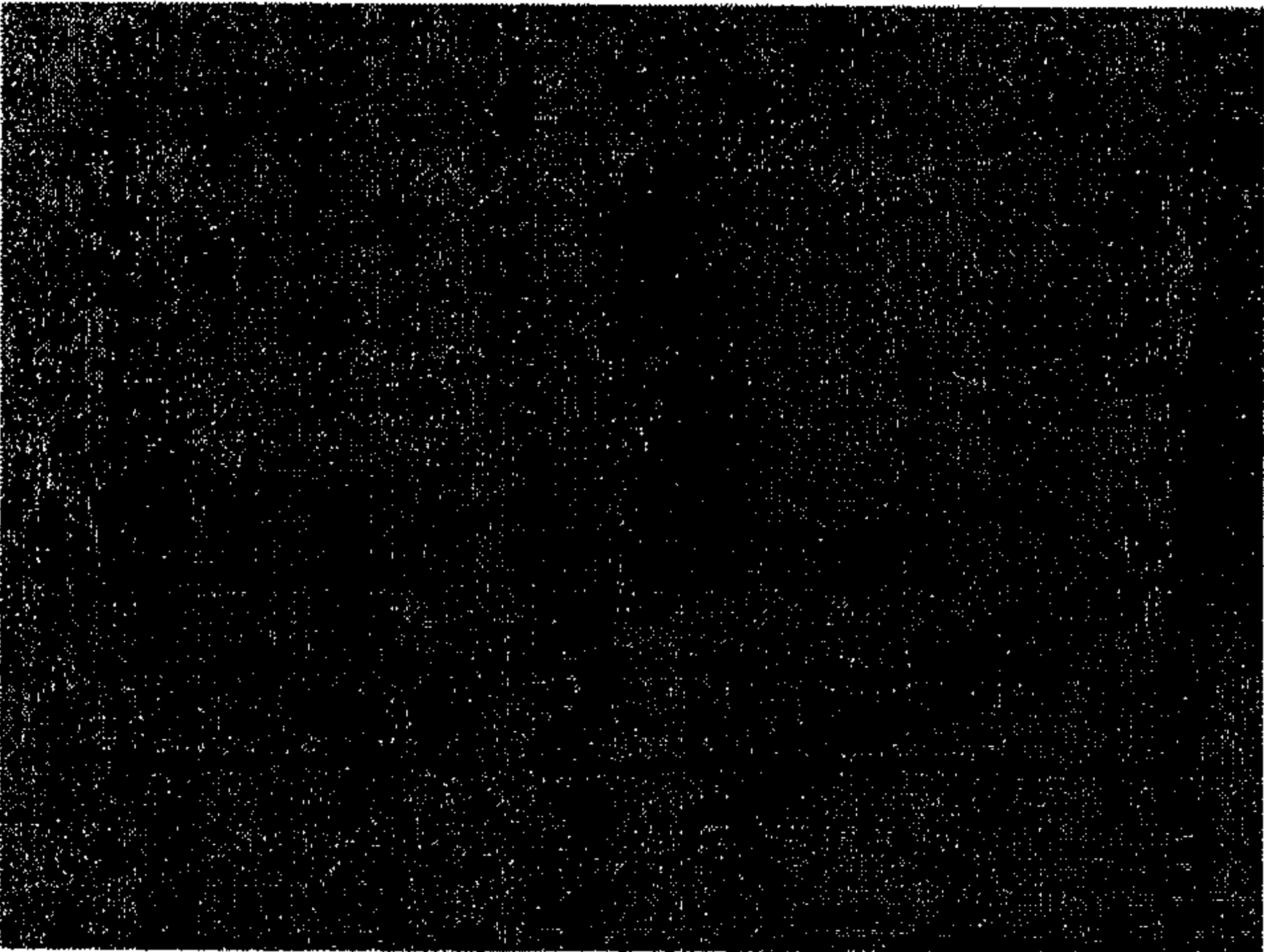
- The linker primers only (note truncation of one of them to match Tm):

>attB1_left (Tm: 61.7)
GGGGACAAGTTTGTACAAAAAAGCAGGCTTC

>attB2_right (Tm: 62.8)
GGGGACCACTTTGTACAAGAAAGCTGGG

Ordering Info

- The plate is arranged as follows:



- The excel file for ordering is here:

Media attB.xls

Mixing Info

- All primers are ordered at 200 μ M. We make mixes to 2.5 μ M for each primer.
- Per 80 μ L mix, add μ L stock of each primer (diluted in Tris-HCl)

_scFv_USER and _USER

Info

- Primer design
- These primers incorporate uracils for later USER excision.

Sequences

▪ _scFv_USER

```

>20100826_scFv_USER_VH1/2for
gttaggUGAGGAGACRGTGACCAGGGTG
>20100826_scFv_USER_VH4/5for
gttaggUGAGGAGACGGTGACCAGGGTT
>20100826_scFv_USER_VH3for
gttaggUGAAGAGACGGTGACCATTGT
>20100826_scFv_USER_VH6for
gttaggUGAGGAGACGGTGACCGTGGTCC

>20100826_scFv_USER_VL1back
gggatcuCAGTCTGTSTGACGCAGCCGCC
>20100826_scFv_USER_VL3back
gggatcuTCCTATGWGCTGACWCAGCCAC
>20100826_scFv_USER_VL38back
gggatcuTCCTATGAGCTGAYRCAGCYACC
>20100826_scFv_USER_VL4back
gggatcuCAGCCTGTGCTGACTCARYC
>20100826_scFv_USER_VL7/8back
gggatcuCAGDCTGTGGTGACYCAGGAGCC
>20100826_scFv_USER_VL9back
gggatcuCAGCCWGXGCTGACTCAGCCMCC
>20100826_scFv_USER_VL11back
gggatcuTCCTCTGAGCTGASTCAGGASCC
>20100826_scFv_USER_VL13back
gggatcuCAGTCTGYCTGAYTCAGCCT
>20100826_scFv_USER_VL15back
gggatcuAATTTTATGCTGACTCAGCCCC

>20100826_scFv_USER_VK1back
gggatcuGACATCCRGDTGACCCAGTCTCC
>20100826_scFv_USER_VK2backts
gggatcuGAAATTGTRWTGACRCAGTCTCC
>20100826_scFv_USER_VK9back
gggatcuGATATTGTGMTGACBCAGWCTCC
>20100826_scFv_USER_VK12back
gggatcuGAAACGACACTCACGCAGTCTC

>20100826_scFv_linker_upper_PEDS_11_405_LLA4_term_T
AGCGCGAGCCACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCT

>20100826_scFv_linker_lower_PEDS_11_405_LLA4_term_T
GGAGCAGAACTAGTATTCGGAGCAGAACCCTGTGGCTCGCGCTGTAGGt

```

▪ _USER

```

>20100827_USER_VH1/2for
agtctagUGAGGAGACRGTGACCAGGGTG
>20100827_USER_VH4/5for
agtctagUGAGGAGACGGTGACCAGGGTT
>20100827_USER_VH3for
agtctagUGAAGAGACGGTGACCATTGT
>20100827_USER_VH6for
agtctagUGAGGAGACGGTGACCGTGGTCC

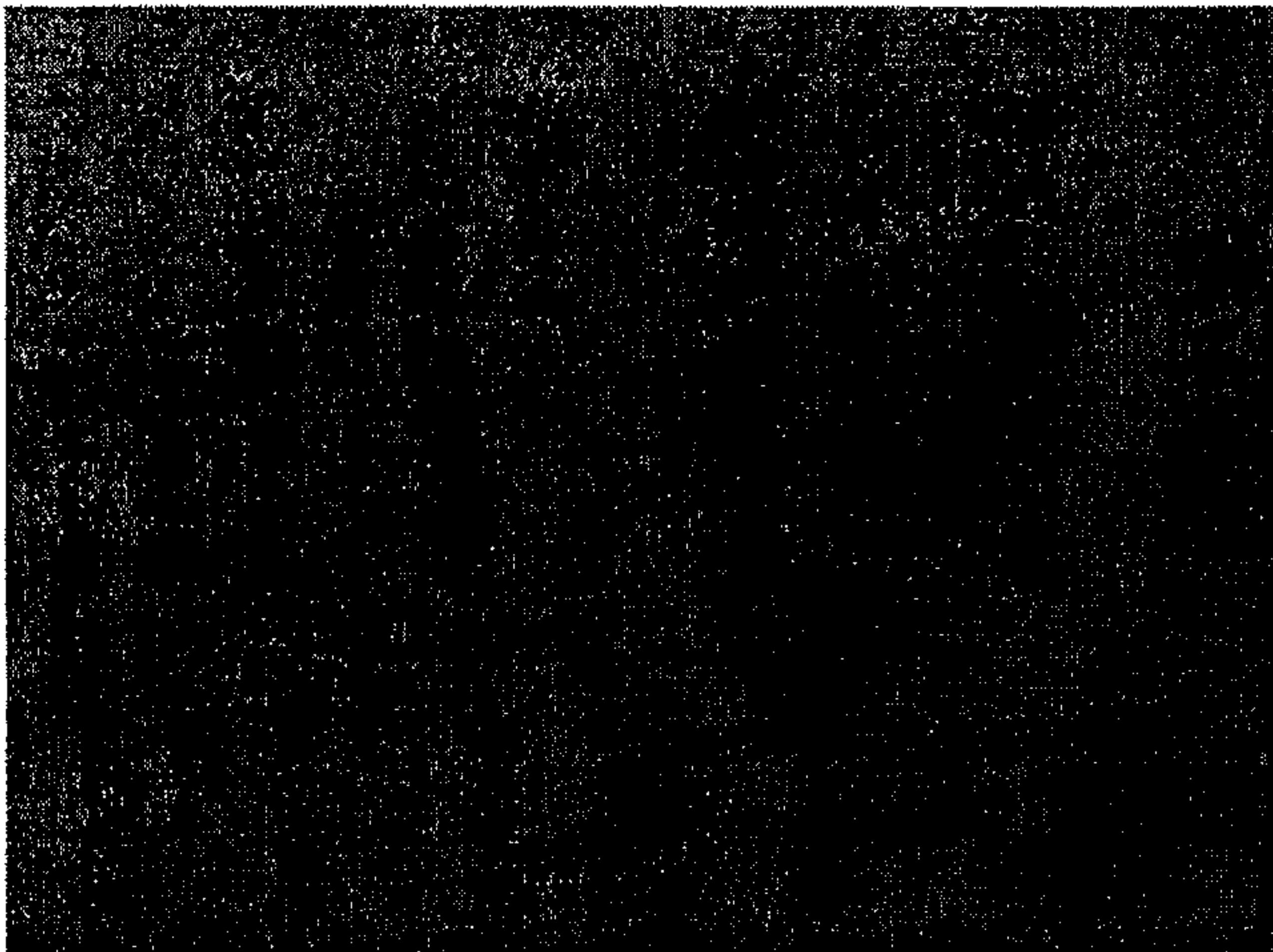
```



```
>20100827_USER_VL1back
actagacuCAGTCTGTSBTGACGCAGCCGCC
>20100827_USER_VL3back
actagacuTCCTATGWGCTGACWCAGCCAC
>20100827_USER_VL38back
actagacuTCCTATGAGCTGAYRCAGCYACC
>20100827_USER_VL4back
actagacuCAGCCTGTGCTGACTCARYC
>20100827_USER_VL7/8back
actagacuCAGDCTGTGGTGACYCAGGAGCC
>20100827_USER_VL9back
actagacuCAGCCWKGCTGACTCAGCCMCC
>20100827_USER_VL11back
actagacuTCCTCTGAGCTGASTCAGGASCC
>20100827_USER_VL13back
actagacuCAGTCTGYICTGAYTCAGCCT
>20100827_USER_VL15back
actagacuAATTTTATGCTGACTCAGCCCC
>20100827_USER_VK1back
actagacuGACATCCRGDTGACCCAGTCTCC
>20100827_USER_VK2backts
actagacuGAAATTGTRWTGACRCAGTCTCC
>20100827_USER_VK9back
actagacuGATATTGTGMTGACBCAGWCTCC
>20100827_USER_VK12back
actagacuGAAACGACACTCACGCAGTCTC
```

Ordering Info

- The scFv linker portions are ordered in individual tubes.
- The rest of the primers are ordered in a plate as arranged below:



- NOTE: to order deoxyuridine, the IDT code is: /ideoxyU/

```
20100826_scFv_USER_VH1/2for gttagg/ideoxyU/GAGGAGACRGTGACCAGGGTG
20100826_scFv_USER_VH4/5for gttagg/ideoxyU/GAGGAGACGGTGACCAGGGTT
20100826_scFv_USER_VH3for gttagg/ideoxyU/GAAGAGACGGTGACCATTGT
20100826_scFv_USER_VH6for gttagg/ideoxyU/GAGGAGACGGTGACCCTGGTCC
20100826_scFv_USER_VL1back gggatc/ideoxyU/CAGTCTGTSBTGACGCAGCCGCC
20100826_scFv_USER_VL3back gggatc/ideoxyU/TCCTATGWGCTGACWCAGCCAC
20100826_scFv_USER_VL38back gggatc/ideoxyU/TCCTATGAGCTGAYRCAGCYACC
20100826_scFv_USER_VL4back gggatc/ideoxyU/CAGCCTGTGCTGACTCARYC
20100826_scFv_USER_VL7/8back gggatc/ideoxyU/CAGDCTGTGGTGACYCAGGAGCC
20100826_scFv_USER_VL9back gggatc/ideoxyU/CAGCCWKGCTGACTCAGCCMCC
20100826_scFv_USER_VL11back gggatc/ideoxyU/TCCTCTGAGCTGASTCAGGASCC
20100826_scFv_USER_VL13back gggatc/ideoxyU/CAGTCTGYICTGAYTCAGCCT
20100826_scFv_USER_VL15back gggatc/ideoxyU/AATTTTATGCTGACTCAGCCCC
20100826_scFv_USER_VK1back gggatc/ideoxyU/GACATCCRGDTGACCCAGTCTCC
20100826_scFv_USER_VK2backts gggatc/ideoxyU/GAAATTGTRWTGACRCAGTCTCC
20100826_scFv_USER_VK9back gggatc/ideoxyU/GATATTGTGMTGACBCAGWCTCC
20100826_scFv_USER_VK12back gggatc/ideoxyU/GAAACGACACTCACGCAGTCTC
```



```

20100827_USER_VH1/2for agtctag/ideoxyU/GAGGAGACRGTGACCAGGGTG
20100827_USER_VH4/5for agtctag/ideoxyU/GAGGAGACGGTGACCAGGGTT
20100827_USER_VH3for agtctag/ideoxyU/GAAGAGACGGTGACCATTGT
20100827_USER_VH6for agtctag/ideoxyU/GAGGAGACGGTGACCGTGGTCC
20100827_USER_VL1back actagac/ideoxyU/CAGTCTGTSBTGACGCAGCCGCC
20100827_USER_VL3back actagac/ideoxyU/TCCTATGWGCTGACWCAGCCAC
20100827_USER_VL38back actagac/ideoxyU/TCCTATGAGCTGAYRCAGCYACC
20100827_USER_VL4back actagac/ideoxyU/CAGCCTGTGCTGACTCARYC
20100827_USER_VL7/8back actagac/ideoxyU/CAGDCTGTGGTGACYCAGGAGCC
20100827_USER_VL9back actagac/ideoxyU/CAGCCNGKGCTGACTCAGCCMCC
20100827_USER_VL11back actagac/ideoxyU/TCCTCTGAGCTGASTCAGGASCC
20100827_USER_VL13back actagac/ideoxyU/CAGTCTGYICTGAYTCAGCCT
20100827_USER_VL15back actagac/ideoxyU/AATTTTATGCTGACTCAGCCCC
20100827_USER_VK1back actagac/ideoxyU/GACATCCRGDTGACCCAGTCTCC
20100827_USER_VK2backts actagac/ideoxyU/GAAATTGTRWTGACRCAGTCTCC
20100827_USER_VK9back actagac/ideoxyU/GATATTGTGMTGACBCAGWCTCC
20100827_USER_VK12back actagac/ideoxyU/GAAACGACACTCACGCAGTCTC

```

- The plate is here:

Media: plate.xls

Mixing Info

- The scFv linker sequences are ordered LabReady as 100 μ M in pH 8 TE
- The primers in the plate are at 200 μ M so must be mixed and diluted.
- Make 2.5 μ M mix in each primer. Since they are 200 μ M, we must divide by 80.
- Per 80 μ L total, put in 1 μ L of each primer.
- Put K and L primers together
- Make tubes with Tris-HCl first:

Tube	Initial Tris-HCl per 80 μ L
A	76
B	67
E	76
F	67

_SOE_PEDS_11_405_LLA4

Info

- Primers based on design from
- Linker is:

```

>PEDS_11_405_LLA4
CCTAACAGCGCGAGCCACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCC

```

- The overlap is:

```

VL ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGTTGACGCAGTCT
VH CTGGTGCCAGTGGCAGAGGAGTGGATTGTCGCGCTCGGTGTCACCAAGACGAGGCTTATGATC

```

Primer target	Number of primers
IGHV (VH/heavy left)	45
IGHJ (VH/heavy right)	8
IGK/LV (VL/light left)	68
IGK/LJ (VL/light right)	11

Sequences

```
>20100505_IGHV_left_1
GAGGTGCAGCTGGTGGAGTCCG
>20100505_IGHV_left_2
GAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_3
CAGGTCACCTTGAGGGAGTCTGGTCC
>20100505_IGHV_left_4
CAGGTTTCAGCTGTTGCAGCCTGG
>20100505_IGHV_left_5
CAGGTGCAGCTACAGCAGTGGGG
>20100505_IGHV_left_6
CAGGTGCAGCTGGTGCAATCTGG
>20100505_IGHV_left_7
CAGGTCACCTTGAAGGAGTCTGGTCC
>20100505_IGHV_left_8
CAGGTCCAGCTGGTACAGTCTGGG
>20100505_IGHV_left_9
CAGGACCAGTTGGTGCAGTCTGGG
>20100505_IGHV_left_10
CAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_11
CAGATGCAGCTGGTGCAGTCTGG
>20100505_IGHV_left_12
CAAATGCAGCTGGTGCAGTCTGGG
>20100505_IGHV_left_13
GAAGTGCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_14
CAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_15
GAGGATCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_16
GAGGTCCAGCTGGTACAGTCTGGG
>20100505_IGHV_left_17
CAGCTGCAGCTGCAGGAGTCC
>20100505_IGHV_left_18
GAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_19
CAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_20
CAGGTACAGCTGCAGCAGTCAGGT
>20100505_IGHV_left_21
GAGATGCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_22
GAGGTGCATCTGGTGGAGTCTGGG
>20100505_IGHV_left_23
GAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_24
GAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_25
CAGGTCCAACCTGGTGTAGTCTGGAGC
>20100505_IGHV_left_26
GAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_27
GAGGTGCAGCTGGTGGAGTCTCG
>20100505_IGHV_left_28
GAGGTTTCAGCTGGTGCAGTCTGGG
>20100505_IGHV_left_29
CAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_30
GAAGTGCAGCTGGTGCAGTCTGG
>20100505_IGHV_left_31
GAGGTGGAGCTGATAGAGTCCATAGA
>20100505_IGHV_left_32
ACAGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_33
GAGGTGCAGCTGGAGGAGTCTGG
>20100505_IGHV_left_34
GAGGTACAGCTGGTGGAGTCTGAAGA
>20100505_IGHV_left_35
CAGGTGCAGCTGCAGGAGTCG
>20100505_IGHV_left_36
GAGGTGCAGCTGTTGGAGTCTGGG
>20100505_IGHV_left_37
CAGGTGCAGCTGGGGCAGTC
>20100505_IGHV_left_38
CAGCTGCAGCTGCAGGAGTCG
>20100505_IGHV_left_39
CAGGTTTCAGCTGGTGCAGTCTGGA
```


>20100505_IGHV_left_40
GAGGTGCAGCTGGTAGAGTCTGGG
>20100505_IGHV_left_41
CAGGTCCAGCTTGTGCAGTCTGGG
>20100505_IGHV_left_42
GAGGTGCAGCTGTGCAGTCTGC
>20100505_IGHV_left_43
GAGGTACAACCTGGTGGAGTCTGGGGG
>20100505_IGHV_left_44
CAGATCACCTTGAAGGAGTCTGGTCC
>20100505_IGHV_left_45
CAGGTACAGCTGATGCAGTCTGGGG
>20100505_IGHJ_right_1_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCACCTGTGGCTCGCGCTGTTAGGTGAGGAGACGGTGACCGTGGTC
>20100505_IGHJ_right_2_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCACCTGTGGCTCGCGCTGTTAGGTGAGGAGACAGTGACCAGGGTGCC
>20100505_IGHJ_right_3_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCACCTGTGGCTCGCGCTGTTAGGTGATGTGGCTGCGGTCTCAGGG
>20100505_IGHJ_right_4_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCACCTGTGGCTCGCGCTGTTAGGTGAGGAGACGGTGACCAGGGTTCC
>20100505_IGHJ_right_5_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCACCTGTGGCTCGCGCTGTTAGGGACCCCTCAGAAGCCAGACCACC
>20100505_IGHJ_right_6_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCACCTGTGGCTCGCGCTGTTAGGTGAGGAGACGGTGACCAGGGTGC
>20100505_IGHJ_right_7_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCACCTGTGGCTCGCGCTGTTAGGGACGTTCCAGGGAGACGGTGT
>20100505_IGHJ_right_8_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCACCTGTGGCTCGCGCTGTTAGGTGAAGAGACGGTGACCATTGTCCCTT
>20100505_IGKV_left_1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGTTGACGAGTCTCCAGC
>20100505_IGKV_left_2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGTTGACGAGTCTCCAGG
>20100505_IGKV_left_3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGCCATCCAGTTGACCCAGTCTCCA
>20100505_IGKV_left_4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGATGACTCAGCCTCCATC
>20100505_IGKV_left_5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAACGACACTCAGCAGTCTCCAGC
>20100505_IGKV_left_6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGATGACCCAGTCTCCATC
>20100505_IGKV_left_7_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGGTGACCCAGTCTCCATC
>20100505_IGKV_left_8_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATTGTGATGACCCAGACTCCACT
>20100505_IGKV_left_9_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCGTGATGACCCAGTCTCCAGA
>20100505_IGKV_left_10_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGCCATCCGGATGACCCAGTCTCC
>20100505_IGKV_left_11_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATGTTGTGATGACACAGTCTCCAGC
>20100505_IGKV_left_12_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACCCAGCATCTGCT
>20100505_IGKV_left_13_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATAGTGATGACGAGTCTCCAGC
>20100505_IGKV_left_14_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACCCAGACTCCACC
>20100505_IGKV_left_15_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGATGATCCAGTCTCCATC
>20100505_IGKV_left_16_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGTTGACACAGTCTCCAGC
>20100505_IGKV_left_17_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACTCAGTCTCCACT
>20100505_IGKV_left_18_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGCCATCCGGATGACCCAGTCTCC
>20100505_IGKV_left_19_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAAATCCAGATGACCCAGTCTCCATC
>20100505_IGKV_left_20_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTAATGACACAGTCTCCAGC
>20100505_IGKV_left_21_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTAATGACACAGTCTCCACC
>20100505_IGKV_left_22_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATGTTGTGATGACTCAGTCTCCACT
>20100505_IGKV_left_23_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATTGTGCTGACCCAGTCTCCAGC
>20100505_IGKV_left_24_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGATGACCCAGTCTCCTTC
>20100505_IGKV_left_25_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGCTGACTCAGTCTCCAGA
>20100505_IGKV_left_26_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAGATTGTGATGACCCAGACTCCACT
>20100505_IGKV_left_27_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGCCATCCAGATGACCCAGTCTCCATC
>20100505_IGKV_left_28_SOE_PEDS_11_405_LLA4

ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGTTGACCCAGTCTCCATC
>20100505_IGKV_left_29_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACCCAGACTCCACT
>20100505_IGKV_left_30_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGTCATCTGGATGACCCAGTCTCCATC
>20100505_IGKV_left_31_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGATGACCCAGCCTCCATC
>20100505_IGKV_left_32_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGTTGACACAGTCTCCAGG
>20100505_IGKJ_right_1
TTTGATCTCCACCTTGGTCCCTCCGC
>20100505_IGKJ_right_2
TTTGATCTCCAGCTTGGTCCCTGG
>20100505_IGKJ_right_3
TTTGATATCCACTTTGGTCCAGGGC
>20100505_IGKJ_right_4
TTTGATTTCCACCTTGGTCCCTTGGC
>20100505_IGKJ_right_5
TTTAATCTCCAGTCGTGTCCCTTGGC
>20100505_IGLV_left_1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGTCTGCCCTGATTCAGCCTCCC
>20100505_IGLV_left_2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCTGCCTGTGCTGACTCAGCCCC
>20100505_IGLV_left_3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGCCTGTGCTGACTCAGCCAC
>20100505_IGLV_left_4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCTTCTGAGCTGACTCAGGACCCCTGC
>20100505_IGLV_left_5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAATCTGCCCTGACTCAGCCTCCT
>20100505_IGLV_left_6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCTATGAGCTGACACAGCCACCC
>20100505_IGLV_left_7_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGGCCCGTGTGACTCAGC
>20100505_IGLV_left_8_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGCCTGTGCTGACTCAATCATCCTC
>20100505_IGLV_left_9_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTGAGCTGAGTCAGGAGCCT
>20100505_IGLV_left_10_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGTCTGCCCTGACTCAGCCTCC
>20100505_IGLV_left_11_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGTCTGTGTTGACGCAGCCGC
>20100505_IGLV_left_12_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGGCAGGGCTGACTCAGCCA
>20100505_IGLV_left_13_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGTCTGTGTTGACGCAGCCGC
>20100505_IGLV_left_14_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGTCTGTGCTGACTCAGCCACCC
>20100505_IGLV_left_15_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACTCAGCCACTCTC
>20100505_IGLV_left_16_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGCCTGTGCTGACTCAGCCAAC
>20100505_IGLV_left_17_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACACAGCCACCC
>20100505_IGLV_left_18_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACACAGCCATCCTC
>20100505_IGLV_left_19_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGCCTGTGCTGACTCAGCCAAC
>20100505_IGLV_left_20_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACACAGCTACCCTC
>20100505_IGLV_left_21_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGCCTGTGCTGACTCAGCCATCT
>20100505_IGLV_left_22_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTCTGGGCCAACTCAGGTGC
>20100505_IGLV_left_23_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACTCAGCCACCCT
>20100505_IGLV_left_24_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGACTGTGGTGACCCAGGAGCC
>20100505_IGLV_left_25_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGTCTGCCCTGACTCAGCCTGC
>20100505_IGLV_left_26_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAATTTTATGCTGACTCAGCCCCACTC
>20100505_IGLV_left_27_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGATGCAGCCACCC
>20100505_IGLV_left_28_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGGCTGTGGTGACTCAGGAGCC
>20100505_IGLV_left_29_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGCTTGTGCTGACTCAATCGCCC
>20100505_IGLV_left_30_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGACTGTGGTGACTCAGGAGCCC
>20100505_IGLV_left_31_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACTCAGCCACACTC
>20100505_IGLV_left_32_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAGGCTGTGCTGACTCAGCCG


```
>20100505_IGLV_left_33_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGTGCTGACTCAGCCACCC
>20100505_IGLV_left_34_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGCCCTGACTCAGCCTCG
>20100505_IGLV_left_35_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTTCTGACTCAGCCTCGCT
>20100505_IGLV_left_36_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTGCTGACGCAGCCG
>20100505_IGLJ_right_1
GAGGACGGTCACCTTGGTGCCA
>20100505_IGLJ_right_2
TAGGACGGTCAGCTTGGTCCCTCC
>20100505_IGLJ_right_3
GAGGACGGTCAGCTGGGTGCC
>20100505_IGLJ_right_4
TAAAATGATCAGCTGGGTTCCCTCCAC
>20100505_IGLJ_right_5
TAGGACGGTGACCTTGGTCCCAGT
>20100505_IGLJ_right_6
TAGGACGGTCAGCTCGGTCCCC
```

Plate information

- Ordered from IDT:
 - Media: i_SOE_PEDS_11_405_LLA4.xls



Mixing info

- Make 2.5 μ M mix in each primer. Since they are 200 μ M, we must divide by 80.
- Per 80 μ L total, put in 1 μ L of each primer.
- Make tubes with Tris-HCl first:

Tube	Initial buffer per 80 μ L
A	18
B	17
C	72
D	4
E	4
F	4
G	69
H	

- Then using a multichannel, pipette 1 µL per 80 of each column into the tubes.
- Then combine tubes A+B, C, D+E+F, G

20100505

Info

- See design
- The T_m was designed for 62°C

Primer target	Number of primers
IGHV (VH/heavy left)	45
IGHJ (VH/heavy right)	8
IGK/LV (VL/light left)	68
IGK/LJ (VL/light right)	11

Sequences

```
>20100505_IGHV_left_1
GAGGTGCAGCTGGTGGAGTCCG
>20100505_IGHV_left_2
GAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_3
CAGGTACACCTTGAGGGAGTCTGGTCC
>20100505_IGHV_left_4
CAGGTTCAGCTGTTGCAGCCTGG
>20100505_IGHV_left_5
CAGGTGCAGCTACAGCAGTGGGG
>20100505_IGHV_left_6
CAGGTGCAGCTGGTGCAATCTGG
>20100505_IGHV_left_7
CAGGTACACCTTGAAGGAGTCTGGTCC
>20100505_IGHV_left_8
CAGGTCCAGCTGGTACAGTCTGGG
>20100505_IGHV_left_9
CAGGACCAGTTGGTGCAGTCTGGG
>20100505_IGHV_left_10
CAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_11
CAGATGCAGCTGGTGCAGTCTGG
>20100505_IGHV_left_12
CAAATGCAGCTGGTGCAGTCTGGG
>20100505_IGHV_left_13
GAAGTGCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_14
CAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_15
GAGGATCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_16
GAGGTCCAGCTGGTACAGTCTGGG
>20100505_IGHV_left_17
CAGCTGCAGCTGCAGGAGTCC
>20100505_IGHV_left_18
GAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_19
CAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_20
CAGGTACAGCTGCAGCAGTCAGGT
>20100505_IGHV_left_21
GAGATGCAGCTGGTGGAGTCTGGG
>20100505_IGHV_left_22
GAGGTGCATCTGGTGGAGTCTGGG
>20100505_IGHV_left_23
GAGGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_24
GAGGTGCAGCTGGTGGAGTCTGG
```


>20100505_IGHV_left_25
CAGGTCCAACCTGGTGTAGTCTGGAGC
>20100505_IGHV_left_26
GAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_27
GAGGTGCAGCTGGTGGAGTCTCG
>20100505_IGHV_left_28
GAGGTTCAGCTGGTGCAGTCTGGG
>20100505_IGHV_left_29
CAGGTGCAGCTGGTGCAGTCTG
>20100505_IGHV_left_30
GAAGTGCAGCTGGTGCAGTCTGG
>20100505_IGHV_left_31
GAGGTGGAGCTGATAGAGTCCATAGA
>20100505_IGHV_left_32
ACAGTGCAGCTGGTGGAGTCTGG
>20100505_IGHV_left_33
GAGGTGCAGCTGGAGGAGTCTGG
>20100505_IGHV_left_34
GAGGTACAGCTGGTGGAGTCTGAAGA
>20100505_IGHV_left_35
CAGGTGCAGCTGCAGGAGTCG
>20100505_IGHV_left_36
GAGGTGCAGCTGTTGGAGTCTGGG
>20100505_IGHV_left_37
CAGGTGCAGCTGGGGCAGTC
>20100505_IGHV_left_38
CAGCTGCAGCTGCAGGAGTCG
>20100505_IGHV_left_39
CAGGTTCAGCTGGTGCAGTCTGGA
>20100505_IGHV_left_40
GAGGTGCAGCTGGTAGAGTCTGGG
>20100505_IGHV_left_41
CAGGTCCAGCTTGTGCAGTCTGGG
>20100505_IGHV_left_42
GAGGTGCAGCTCTTGCAGTCTGC
>20100505_IGHV_left_43
GAGGTACAACCTGGTGGAGTCTGGGGG
>20100505_IGHV_left_44
CAGATCACCTTGAAGGAGTCTGGTCC
>20100505_IGHV_left_45
CAGGTACAGCTGATGCAGTCTGGGG

>20100505_IGHJ_right_1
TGAGGAGACGGTGACCGTGGTC
>20100505_IGHJ_right_2
TGAGGAGACAGTGACCAGGGTGCC
>20100505_IGHJ_right_3
TGATGTGGCTGCCGTCTCAGGG
>20100505_IGHJ_right_4
TGAGGAGACGGTGACCAGGGTTCC
>20100505_IGHJ_right_5
GACCCCTCAGAAGCCAGACCACC
>20100505_IGHJ_right_6
TGAGGAGACGGTGACCAGGGTGC
>20100505_IGHJ_right_7
GACGTTCCCAGGGAGACGGTGT
>20100505_IGHJ_right_8
TGAAGAGACGGTGACCATTGTCCCTT

>20100505_IGKV_left_1
GAAATTGTGTTGACGCAGTCTCCAGC
>20100505_IGKV_left_2
GAAATTGTGTTGACGCAGTCTCCAGG
>20100505_IGKV_left_3
GCCATCCAGTTGACCCAGTCTCCA
>20100505_IGKV_left_4
GACATCCAGATGACTCAGCCTCCATC
>20100505_IGKV_left_5
GAAACGACACTCACGCAGTCTCCAGC
>20100505_IGKV_left_6
GACATCCAGATGACCCAGTCTCCATC
>20100505_IGKV_left_7
GACATCCAGGTGACCCAGTCTCCATC
>20100505_IGKV_left_8
GACATTGTGATGACCCAGACTCCACT
>20100505_IGKV_left_9
GACATCGTGATGACCCAGTCTCCAGA
>20100505_IGKV_left_10
GCCATCCGGATGACCCAGTCTCC
>20100505_IGKV_left_11
GATGTTGTGATGACACAGTCTCCAGC
>20100505_IGKV_left_12

GATATTGTGATGACCCAGCATCTGCT
>20100505_IGKV_left_13
GAAATAGTGATGACGCAGTCTCCAGC
>20100505_IGKV_left_14
GATATTGTGATGACCCAGACTCCACC
>20100505_IGKV_left_15
GACATCCAGATGATCCAGTCTCCATC
>20100505_IGKV_left_16
GAAATTGTGTTGACACAGTCTCCAGC
>20100505_IGKV_left_17
GATATTGTGATGACTCAGTCTCCACT
>20100505_IGKV_left_18
GCCATCCGGATGACCCAGTCTCC
>20100505_IGKV_left_19
AACATCCAGATGACCCAGTCTCCATC
>20100505_IGKV_left_20
GAAATTGTAATGACACAGTCTCCAGC
>20100505_IGKV_left_21
GAAATTGTAATGACACAGTCTCCACC
>20100505_IGKV_left_22
GATGTTGTGATGACTCAGTCTCCACT
>20100505_IGKV_left_23
GACATTGTGCTGACCCAGTCTCCAGC
>20100505_IGKV_left_24
GACATCCAGATGACCCAGTCTCCTTC
>20100505_IGKV_left_25
GAAATTGTGCTGACTCAGTCTCCAGA
>20100505_IGKV_left_26
GAGATTGTGATGACCCAGACTCCACT
>20100505_IGKV_left_27
GCCATCCAGATGACCCAGTCTCCATC
>20100505_IGKV_left_28
GACATCCAGTTGACCCAGTCTCCATC
>20100505_IGKV_left_29
GATATTGTGATGACCCAGACTCCACT
>20100505_IGKV_left_30
GTCATCTGGATGACCCAGTCTCCATC
>20100505_IGKV_left_31
GACATCCAGATGACCCAGCCTCCATC
>20100505_IGKV_left_32
GAAATTGTGTTGACACAGTCTCCAGG
.
>20100505_IGLV_left_1
CAGTCTGCCCTGATTGAGCCTCCC
>20100505_IGLV_left_2
CTGCCCTGTGCTGACTCAGCCCC
>20100505_IGLV_left_3
CAGCCTGTGCTGACTCAGCCAC
>20100505_IGLV_left_4
TCTTCTGAGCTGACTCAGGACCCTGC
>20100505_IGLV_left_5
CAATCTGCCCTGACTCAGCCTCCT
>20100505_IGLV_left_6
TCCTATGAGCTGACACAGCCACCC
>20100505_IGLV_left_7
CGGCCCGTGCTGACTCAGC
>20100505_IGLV_left_8
CAGCCTGTGCTGACTCAATCATCCTC
>20100505_IGLV_left_9
TCCTCTGAGCTGAGTCAGGAGCCT
>20100505_IGLV_left_10
CAGTCTGCCCTGACTCAGCCTCC
>20100505_IGLV_left_11
CAGTCTGTGTTGACGCAGCCGC
>20100505_IGLV_left_12
CAGGCAGGGCTGACTCAGCCA
>20100505_IGLV_left_13
CAGTCTGTGTTGACGCAGCCGC
>20100505_IGLV_left_14
CAGTCTGTGCTGACTCAGCCACCC
>20100505_IGLV_left_15
TCCTATGAGCTGACTCAGCCACTCTC
>20100505_IGLV_left_16
CAGCCTGTGCTGACTCAGCCAAC
>20100505_IGLV_left_17
TCCTATGAGCTGACACAGCCACCC
>20100505_IGLV_left_18
TCCTATGAGCTGACACAGCCATCCTC
>20100505_IGLV_left_19
CAGCCTGTGCTGACTCAGCCAAC
>20100505_IGLV_left_20
TCCTATGAGCTGACACAGCTACCCTC
>20100505_IGLV_left_21

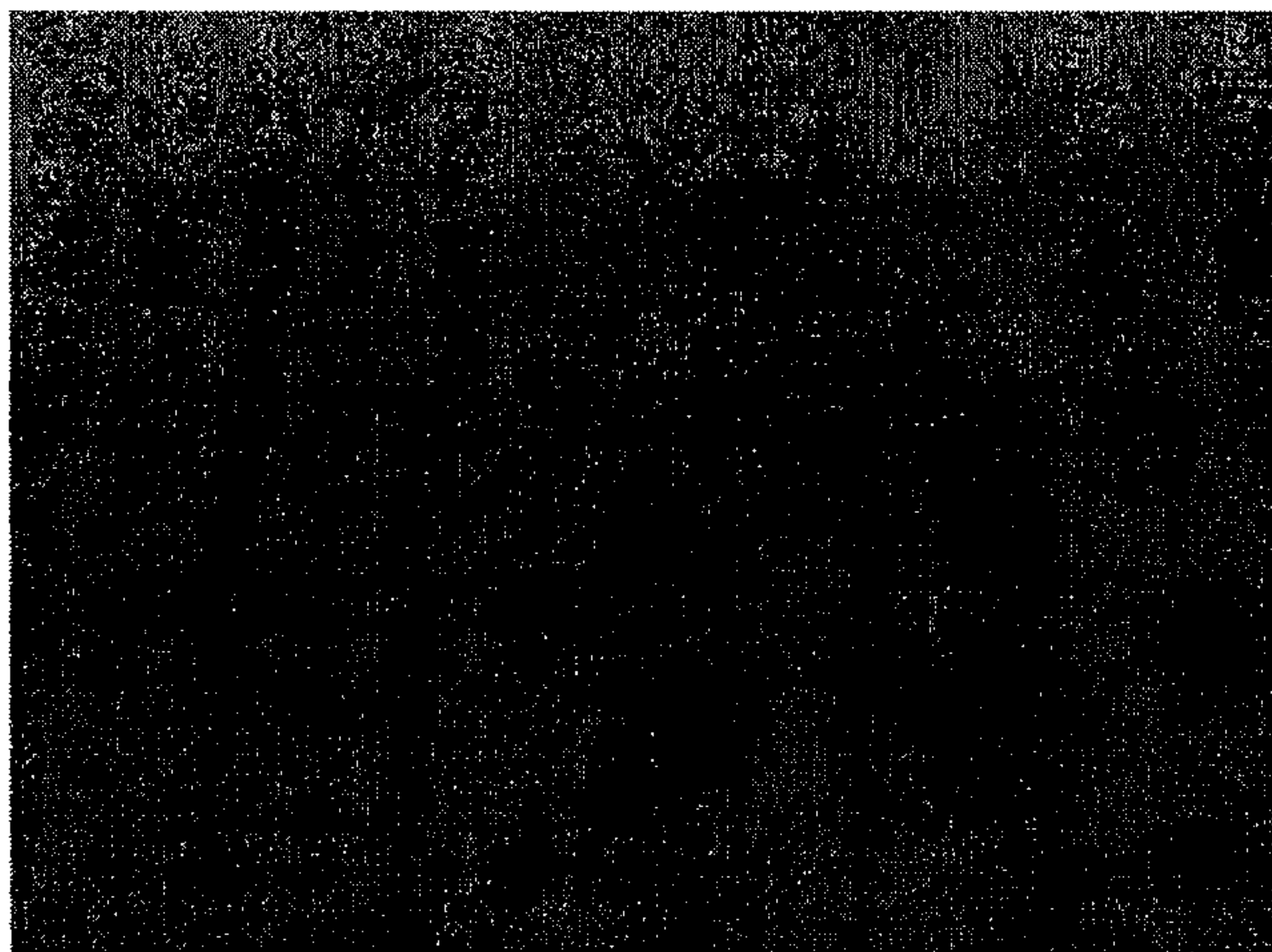

```
CAGCCTGTGCTGACTCAGCCATCT
>20100505_IGLV_left_22
TCCTCTGGGCCAACTCAGGTGC
>20100505_IGLV_left_23
TCCTATGAGCTGACTCAGCCACCCT
>20100505_IGLV_left_24
CAGACTGTGGTGACCCAGGAGCC
>20100505_IGLV_left_25
CAGTCTGCCCTGACTCAGCCTGC
>20100505_IGLV_left_26
AATTTTATGCTGACTCAGCCCCACTC
>20100505_IGLV_left_27
TCCTATGAGCTGATGCAGCCACCC
>20100505_IGLV_left_28
CAGGCTGTGGTGACTCAGGAGCC
>20100505_IGLV_left_29
CAGCTTGTGCTGACTCAATCGCCC
>20100505_IGLV_left_30
CAGACTGTGGTGACTCAGGAGCCC
>20100505_IGLV_left_31
TCCTATGAGCTGACTCAGCCACACTC
>20100505_IGLV_left_32
CAGGCTGTGCTGACTCAGCCG
>20100505_IGLV_left_33
TCCTATGTGCTGACTCAGCCACCC
>20100505_IGLV_left_34
CAGTCTGCCCTGACTCAGCCTCG
>20100505_IGLV_left_35
CAGTCTGTTCTGACTCAGCCTCGCT
>20100505_IGLV_left_36
CAGTCTGTGCTGACGCAGCCG

>20100505_IGKJ_right_1
TTTGATCTCCACCTTGGTCCCTCCGC
>20100505_IGKJ_right_2
TTTGATCTCCAGCTTGGTCCCTGG
>20100505_IGKJ_right_3
TTTGATATCCACTTTGGTCCCAGGGC
>20100505_IGKJ_right_4
TTTGATTTCACCTTGGTCCCTTGGC
>20100505_IGKJ_right_5
TTTAATCTCCAGTCGTGTCCCTTGGC

>20100505_IGLJ_right_1
GAGGACGGTCACCTTGGTGCCA
>20100505_IGLJ_right_2
TAGGACGGTCAGCTTGGTCCCTCC
>20100505_IGLJ_right_3
GAGGACGGTCAGCTGGGTGCC
>20100505_IGLJ_right_4
TAAAATGATCAGCTGGGTTCCTCCAC
>20100505_IGLJ_right_5
TAGGACGGTGACCTTGGTCCCAGT
>20100505_IGLJ_right_6
TAGGACGGTCAGCTCGGTCCCC
```

Plate information

- Ordered form IDT:



Mixing info

- Make 2.5 μM mix in each primer. Since they are 200 μM , we must divide by 80.
- Per 80 μL total, put in 1 μL of each primer.
- Make tubes with Tris-HCl first:

Tube	Initial buffer per 80 μL
A	18
B	17
C	72
D	4
E	4
F	4
G	69
H	

- Then using a multichannel, pipette 1 μL per 80 of each column into the tubes.
- Then combine tubes A+B, C, D+E+F, G

_SOE_PEDS_11_405_LLA4

Info

- Uses primers with a new linker sequence.
- Linker is;

```
>PEDS_11_405_LLA4  
CCTAACAGCGCGAGCCACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCC
```

- Example overlap is:

VL ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTGTTGACGCAGCC
VH CCTGGTGCCAGTGGCAGAGGAGTGGATTGTCGCGCTCGGTGTCACCAAGACGAGGCTTATGATC

Primer target	Number of primers
IGHV (VH/heavy left)	22
IGHJ (VH/heavy right)	5
IGK/LV (VL/light left)	76
IGK/LJ (VL/light right)	10

Sequences

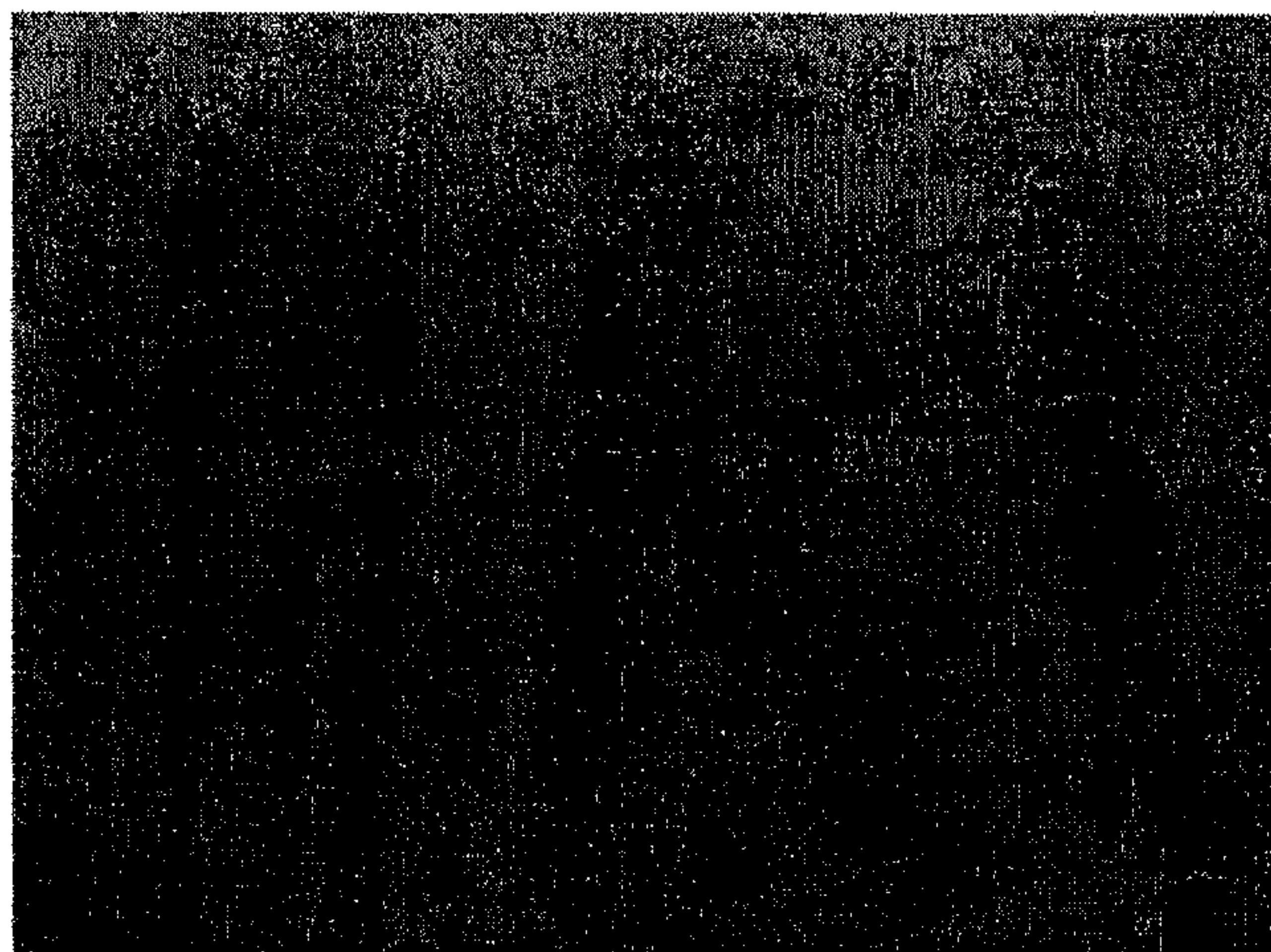
```
>20100329_VH4back|1
CAGGTGCAGCTGCAGGAGTCCG
>20100329_VH4back|2
CAGGTGCAGCTGCAGGAGTCCG
>20100329_VH5back|1
CAGGTACAGCTGCAGCAGTCA
>20100329_VH6back|1
CAGGTGCAGCTACAGCAGTGGG
>20100329_VH10back|1
GAGGTGCAGCTGGTGGAGACC
>20100329_VH10back|2
GAGGTGCAGCTGGTGGAGACT
>20100329_VH10back|3
GAGGTGCAGCTGGTGGAGTCC
>20100329_VH10back|4
GAGGTGCAGCTGGTGGAGTCT
>20100329_VH10back|5
GAGGTGCAGCTGTTGGAGACC
>20100329_VH10back|6
GAGGTGCAGCTGTTGGAGACT
>20100329_VH10back|7
GAGGTGCAGCTGTTGGAGTCC
>20100329_VH10back|8
GAGGTGCAGCTGTTGGAGTCT
>20100329_VH12back|1
CAGGTCCAGCTGGTACAGTCTGG
>20100329_VH12back|2
CAGGTCCAGCTGGTGCAGTCTGG
>20100329_VH12back|3
CAGGTCCAGCTTGTACAGTCTGG
>20100329_VH12back|4
CAGGTCCAGCTTGTGCAGTCTGG
>20100329_VH14back|1
CAGATCACCTTGAAGGAGTCTG
>20100329_VH14back|2
CAGGTACCTTGAAGGAGTCTG
>20100329_VH22back|1
CAGGTGCAGCTGGTGCAATCTGG
>20100329_VH22back|2
CAGGTGCAGCTGGTGCAGTCTGG
>20100329_VH22back|3
CAGGTGCAGCTGGTGGAACTCTGG
>20100329_VH22back|4
CAGGTGCAGCTGGTGGAGTCTGG
>20100329_VH1/2for|1_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCCTGTGGCTCGCGCTGTTAGGTGAGGAGACAGTGACCAGGGTG
>20100329_VH1/2for|2_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCCTGTGGCTCGCGCTGTTAGGTGAGGAGACGGTGACCAGGGTG
>20100329_VH4/5for|1_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCCTGTGGCTCGCGCTGTTAGGTGAGGAGACGGTGACCAGGGTG
>20100329_VH3for|1_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCCTGTGGCTCGCGCTGTTAGGTGAAGAGACGGTGACCATTTGT
>20100329_VH6for|1_SOE_PEDS_11_405_LLA4
CTAGTATTCGGAGCAGAACCCTGTGGCTCGCGCTGTTAGGTGAGGAGACGGTGACCGTGGTCC
>20100329_VL1back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTCTTGACGCAGCCGCC
>20100329_VL1back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTCTTGACGCAGCCGCC
>20100329_VL1back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTCTTGACGCAGCCGCC
>20100329_VL1back|4_SOE_PEDS_11_405_LLA4
```


ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTGCTGACGCAGCCGCC
>20100329_VL1back|5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTGCTGACGCAGCCGCC
>20100329_VL1back|6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTGCTGACGCAGCCGCC
>20100329_VL3back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACACAGCCAC
>20100329_VL3back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACTCAGCCAC
>20100329_VL3back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGTGCTGACACAGCCAC
>20100329_VL3back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGTGCTGACTCAGCCAC
>20100329_VL38back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACACAGCCACC
>20100329_VL38back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACACAGCTACC
>20100329_VL38back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACGCAGCCACC
>20100329_VL38back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGACGCAGCTACC
>20100329_VL38back|5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGATACAGCCACC
>20100329_VL38back|6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGATACAGCTACC
>20100329_VL38back|7_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGATGCAGCCACC
>20100329_VL38back|8_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTATGAGCTGATGCAGCTACC
>20100329_VL4back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCTGTGCTGACTCAACC
>20100329_VL4back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCTGTGCTGACTCAATC
>20100329_VL4back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCTGTGCTGACTCAGCC
>20100329_VL4back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCTGTGCTGACTCAGTC
>20100329_VL7/8back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGACTGTGGTGACCCAGGAGCC
>20100329_VL7/8back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGACTGTGGTGACTCAGGAGCC
>20100329_VL7/8back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGGCTGTGGTGACCCAGGAGCC
>20100329_VL7/8back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGGCTGTGGTGACTCAGGAGCC
>20100329_VL7/8back|5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTGGTGACCCAGGAGCC
>20100329_VL7/8back|6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCTGTGGTGACTCAGGAGCC
>20100329_VL9back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCAGGGCTGACTCAGCCACC
>20100329_VL9back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCAGGGCTGACTCAGCCCCC
>20100329_VL9back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCAGTGTGACTCAGCCACC
>20100329_VL9back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCAGTGTGACTCAGCCCCC
>20100329_VL9back|5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCTGGGCTGACTCAGCCACC
>20100329_VL9back|6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCTGGGCTGACTCAGCCCCC
>20100329_VL9back|7_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCTGTGCTGACTCAGCCACC
>20100329_VL9back|8_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGCCTGTGCTGACTCAGCCCCC
>20100329_VL11back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTCTGAGCTGACTCAGGAGCC
>20100329_VL11back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTCTGAGCTGACTCAGGAGCC
>20100329_VL11back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTCTGAGCTGAGTCAGGAGCC
>20100329_VL11back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCTCCTCTGAGCTGAGTCAGGAGCC
>20100329_VL13back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGCCCTGACTCAGCCT
>20100329_VL13back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGCCCTGATTGAGCCT
>20100329_VL13back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGCTCTGACTCAGCCT
>20100329_VL13back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGCTCTGATTGAGCCT
>20100329_VL13back|5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTCTGACTCAGCCT

>20100329_VL13back|6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTCTGATTGAGCCT
>20100329_VL13back|7_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTCTGACTCAGCCT
>20100329_VL13back|8_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCCAGTCTGTCTGATTGAGCCT
>20100329_VL15back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCAATTTATGCTGACTCAGCCCC
>20100329_VK1back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGATGACCCAGTCTCC
>20100329_VK1back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGGTGACCCAGTCTCC
>20100329_VK1back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCAGTTGACCCAGTCTCC
>20100329_VK1back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCGGATGACCCAGTCTCC
>20100329_VK1back|5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCGGGTGACCCAGTCTCC
>20100329_VK1back|6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGACATCCGGTTGACCCAGTCTCC
>20100329_VK2backts|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTAATGACACAGTCTCC
>20100329_VK2backts|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTAATGACGCAGTCTCC
>20100329_VK2backts|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTATTGACACAGTCTCC
>20100329_VK2backts|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTATTGACGCAGTCTCC
>20100329_VK2backts|5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGATGACACAGTCTCC
>20100329_VK2backts|6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGATGACGCAGTCTCC
>20100329_VK2backts|7_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGTTGACACAGTCTCC
>20100329_VK2backts|8_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAATTGTGTTGACGCAGTCTCC
>20100329_VK9back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACCCAGACTCC
>20100329_VK9back|2_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACCCAGTCTCC
>20100329_VK9back|3_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACGCAGACTCC
>20100329_VK9back|4_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACGCAGTCTCC
>20100329_VK9back|5_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACTCAGACTCC
>20100329_VK9back|6_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGATGACTCAGTCTCC
>20100329_VK9back|7_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGCTGACCCAGACTCC
>20100329_VK9back|8_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGCTGACCCAGTCTCC
>20100329_VK9back|9_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGCTGACGCAGACTCC
>20100329_VK9back|10_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGCTGACGCAGTCTCC
>20100329_VK9back|11_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGCTGACTCAGACTCC
>20100329_VK9back|12_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGATATTGTGCTGACTCAGTCTCC
>20100329_VK12back|1_SOE_PEDS_11_405_LLA4
ACAGTGGTTCTGCTCCGAATACTAGTTCTGCTCCGGGATCCGAAACGACACTCACGCAGTCTC
>20100329_VL1/2for|1
TAGGACGGTCACCTTGGTCC
>20100329_VL1/2for|2
TAGGACGGTCAGCTTGGTCC
>20100329_VL1/2for|3
TAGGACGGTGACCTTGGTCC
>20100329_VL1/2for|4
TAGGACGGTGAGCTTGGTCC
>20100329_VL7for|1
GAGGACGGTCAGCTGGGTGC
>20100329_VK1for|1
TTTGATTCCACCTTGGTCC
>20100329_VK2/4for|1
TTTGATCTCCACCTTGGTCC
>20100329_VK2/4for|2
TTTGATCTCCAGCTTGGTCC
>20100329_VK3for|1
TTTGATATCCACTTGGTCC
>20100329_VK5for|1
TTTAATCTCCAGTCGTGTCC

Plate info

- The primers were ordered from IDT in IDTE pH 8 at 200 μ M concentration.
 - Media _SOE_PEDS_11_405_LLA4.xls

**Mixing info**

- Make 2.5 μ M mix in each primer. Since they are 200 μ M, we must divide by 80.
- Per 80 μ L total, put in 1 μ L of each primer.
- Make tubes with Tris-HCl first:

Tube	Initial buffer per 80 μ L
A	58
B	75
C	1
D	1
E	1
F	1
G	35
H	35

- Then using a multichannel, pipette 1 μ L per 80 of each column into the tubes.
- Then combine tubes A,B, C+D+E+F, G+H

_SOE_PNAS_85_5879_G4S3

-
-
-
- scFv linker sequence:

```
>(G_4 S)_3 scFv linker from Huston et al, PNAS 85: 5879
GGTGGCGGTGGCTCGGGCGGTGGTGGGTCGGGTGGCGGCGGATCT
```

- The counts are:

Primer target	Number of primers
IGHV (VH/heavy left)	7
IGHJ (VH/heavy right)	4
IGK/LV (VL/light left)	13
IGK/LJ (VL/light right)	6

- The primers are below. "back" are the 5' primers (to the left) and "for" are the 3' primers (to the right).

```
>20100329_VH4back
CAGGTGCAGCTGCAGGAGTCSG
>20100329_VH5back
CAGGTACAGCTGCAGCAGTCA
>20100329_VH6back
CAGGTGCAGCTACAGCAGTGGG
>20100329_VH10back
GAGGTGCAGCTGKTGGAGWCY
>20100329_VH12back
CAGGTCCAGCTKGTRCAGTCTGG
>20100329_VH14back
CAGRTCACCTTGAAGGAGTCTG
>20100329_VH22back
CAGGTGCAGCTGGTGSARTCTGG

>20100329_VH1/2for_SOE
CCCGACCCACCACCGCCGAGCCACCGCCACCTGAGGAGACRGTGACCAGGGTG
>20100329_VH4/5for_SOE
CCCGACCCACCACCGCCGAGCCACCGCCACCTGAGGAGACGGTGACCAGGGTT
>20100329_VH3for_SOE
CCCGACCCACCACCGCCGAGCCACCGCCACCTGAAGAGACGGTGACCATTGT
>20100329_VH6for_SOE
CCCGACCCACCACCGCCGAGCCACCGCCACCTGAGGAGACGGTGACCGTGGTCC

>20100329_VL1back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTCAGTCTGTSBTGACGCAGCCGCC
>20100329_VL3back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTTCCTATGWGCTGACWCAGCCAC
>20100329_VL38back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTTCCTATGAGCTGAYRCAGCYACC
>20100329_VL4back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTCAGCCTGTGCTGACTCARYC
>20100329_VL7/8back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTCAGDCTGTGGTGACYCAGGAGCC
>20100329_VL9back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTCAGCCWGXGCTGACTCAGCCMCC
>20100329_VL11back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTTCCTCTGAGCTGASTCAGGASCC
>20100329_VL13back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTCAGTCTGYCTGAYTCAGCCT
>20100329_VL15back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTAATTTTATGCTGACTCAGCCCC

>20100329_VL1/2for
TAGGACGGTSASCTGGTCC
>20100329_VL7for
GAGGACGGTCAGCTGGGTGC

>20100329_VK1back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTGACATCCRGDTGACCCAGTCTCC
>20100329_VK2backts_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTGAAATTGTRWTGACRCAGTCTCC
>20100329_VK9back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTGATATTGTGMTGACBCAGWCTCC
>20100329_VK12back_SOE
CGGGCGGTGGTGGGTGCGGTGGCGGCGGATCTGAAACGACACTCACGCAGTCTC

>20100329_VK1for
TTTGATTTCCACCTTGGTCC
>20100329_VK2/4for
TTTGATCTCCASCTTGGTCC
>20100329_VK3for
TTTGATATCCACTTGGTCC
>20100329_VK5for
TTTAATCTCCAGTCGTGTCC
```


- The IDT plate is arranged as follows:

WellPosition	Name	Sequence
A1	20100329_VH4back	CAGGTGCAGCTGCAGGAGTCSG
A2	20100329_VH5back	CAGGTACAGCTGCAGCAGTCA
A3	20100329_VH6back	CAGGTGCAGCTACAGCAGTGGG
A4	20100329_VH10back	GAGGTGCAGCTGKTGGAGWCY
A5	20100329_VH12back	CAGGTCCAGCTKGTRCAGTCTGG
A6	20100329_VH14back	CAGRTCACCTTGAAGGAGTCTG
A7	20100329_VH22back	CAGGTGCAGCTGGTGSARTCTGG
B1	20100329_VH1/2for_SOE	CCCGACCCACCACCGCCCCGAGCCACCGCCACCTGAGGAGACRGTGACCAGGGTG
B2	20100329_VH4/5for_SOE	CCCGACCCACCACCGCCCCGAGCCACCGCCACCTGAGGAGACGGTGACCAGGGTT
B3	20100329_VH3for_SOE	CCCGACCCACCACCGCCCCGAGCCACCGCCACCTGAAGAGACGGTGACCATTGT
B4	20100329_VH6for_SOE	CCCGACCCACCACCGCCCCGAGCCACCGCCACCTGAGGAGACGGTGACCGTGGTCC
C1	20100329_VL1back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTCAGTCTGTSBTGACGCAGCCGCC
C2	20100329_VL3back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTTCCTATGWGCTGACWCAGCCAC
C3	20100329_VL38back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTTCCTATGAGCTGAYRCAGCYACC
C4	20100329_VL4back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTCAGCCTGTGCTGACTCARYC
C5	20100329_VL7/8back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTCAGDCTGTGGTGACYCAGGAGCC
C6	20100329_VL9back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTCAGCCWKGCTGACTCAGCCMCC
C7	20100329_VL11back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTTCCTCTGAGCTGASTCAGGASCC
C8	20100329_VL13back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTCAGTCTGYCTGAYTCAGCCT
C9	20100329_VL15back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTAATTTTATGCTGACTCAGCCCC
D1	20100329_VL1/2for	TAGGACGGTSASCTTGGTCC
D2	20100329_VL7for	GAGGACGGTCAGCTGGGTGC
E1	20100329_VK1back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTGACATCCRGDTGACCCAGTCTCC
E2	20100329_VK2backts_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTGAAATTGTRWTGACRCAGTCTCC
E3	20100329_VK9back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTGATATTGTGMTGACBCAGWCTCC
E4	20100329_VK12back_SOE	CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTGAAACGACACTCACGCAGTCTC
F1	20100329_VK1for	TTTGATTTCCACCTTGGTCC
F2	20100329_VK2/4for	TTTGATCTCCASCTTGGTCC
F3	20100329_VK3for	TTTGATATCCACTTGGTCC
F4	20100329_VK5for	TTTAATCTCCAGTCGTGTCC

- I also ordered "linker primers" that are only complementary to the linker.

```

5'                                     CGGGCGGTGGTGGGTCTGGGTGGCGGCGGATCTCAGTCTGTSBTGACGCAGCCGCC
3' TTGGGACCACTGGCAGAGGAGTCCACCGCCACCGAGCCCGCCACCACCCAGCCC

>VHfor_SOE_linker
CCCGACCCACCACCGCCCCG
>VLback_SOE_linker
CGGGCGGTGGTGGGTCTGGG

```



```

>20100329_VH1/2for
TGAGGAGACRGTGACCAGGGTG
>20100329_VH4/5for
TGAGGAGACGGTGACCAGGGTT
>20100329_VH3for
TGAAGAGACGGTGACCATTGT
>20100329_VH6for
TGAGGAGACGGTGACCGTGGTCC

>20100329_VL1back
CAGTCTGTSBTGACGCAGCCGCC
>20100329_VL3back
TCCTATGWGCTGACWCAGCCAC
>20100329_VL38back
TCCTATGAGCTGAYRCAGCYACC
>20100329_VL4back
CAGCCTGTGCTGACTCARYC
>20100329_VL7/8back
CAGDCTGTGGTGACYCAGGAGCC
>20100329_VL9back
CAGCCWKGCTGACTCAGCCMCC
>20100329_VL11back
TCCTCTGAGCTGASTCAGGASCC
>20100329_VL13back
CAGTCTGYCTGAYTCAGCCT
>20100329_VL15back
AATTTTATGCTGACTCAGCCCC

>20100329_VK1back
GACATCCRGDTGACCCAGTCTCC
>20100329_VK2backts
GAAATTGTRWTGACRCAGTCTCC
>20100329_VK9back
GATATTGTGMTGACBCAGWCTCC
>20100329_VK12back
GAAACGACACTCACGCAGTCTC

```

- These are arranged in the plate as follows:

WellPosition	Name	Sequence
A1	20100329_VH1/2for	TGAGGAGACRGTGACCAGGGTG
A2	20100329_VH4/5for	TGAGGAGACGGTGACCAGGGTT
A3	20100329_VH3for	TGAAGAGACGGTGACCATTGT
A4	20100329_VH6for	TGAGGAGACGGTGACCGTGGTCC
C1	20100329_VL1back	CAGTCTGTSBTGACGCAGCCGCC
C2	20100329_VL3back	TCCTATGWGCTGACWCAGCCAC
C3	20100329_VL38back	TCCTATGAGCTGAYRCAGCYACC
C4	20100329_VL4back	CAGCCTGTGCTGACTCARYC
C5	20100329_VL7/8back	CAGDCTGTGGTGACYCAGGAGCC
C6	20100329_VL9back	CAGCCWKGCTGACTCAGCCMCC
C7	20100329_VL11back	TCCTCTGAGCTGASTCAGGASCC
C8	20100329_VL13back	CAGTCTGYCTGAYTCAGCCT
C9	20100329_VL15back	AATTTTATGCTGACTCAGCCCC
E1	20100329_VK1back	GACATCCRGDTGACCCAGTCTCC
E2	20100329_VK2backts	GAAATTGTRWTGACRCAGTCTCC
E3	20100329_VK9back	GATATTGTGMTGACBCAGWCTCC
E4	20100329_VK12back	GAAACGACACTCACGCAGTCTC
H6	garbage 1	ACGTATGCATGCTAG
H7	garbage 2	ACGTATGCAGTCTAG

H8 H9 H10 H11 H12

H8	garbage 3	ACCCGTGCATGCTAG
H9	garbage 4	GGTTATGCATGCTAG
H10	garbage 5	ACGTATGCAGACTAG
H11	garbage 6	ACGTATGCATTTTAG
H12	garbage 7	ACGTATGGGTGCTAG

Appendix B

SOE_PCR and ScFV generation in single cells

SOE VDJ (heavy and light chain pairing)

Use extra mix to make 5x FV1 (180ng/ul... 1.67ulRNA+5.33ul H2O) for light chains and 5 for heavy chains

1. Reverse transcription; Proceed as follow for RT-PCR

Did 5 tubes for heavy chain and 5 tubes for light chain (mix shown here was because this was done at the same time as HIV VDJ)

1.1	Assemble a RT reaction as follow;	2x 1tube
	• H2O(DEPC)	0μl(to 12μl)
	• 2μM Gene-specific primer(heavy or light)	5ul (5pmole)
	• Total RNA	7 μl (150ng)

Heat at 95°C for 1min followed by
65°C for 5 min than ice for 1 min

Spin down and Add the following:	1tube	30tubes
• 5x First strand buffer	4μl	120
• 10mM dNTP mix	1μl	30
• 0.1M DTT	1μl	30
• RnaseInhibitor-Enzymatics	1μl	30
• Superscript III	1μl	30
Incubate at 55°C for 60 min	(8μl/tubes)	

1.2 Inactivate enzyme by heating at 70°C for 15 min

1.3 Remove RNA/DNA hybrid by adding 1μl of E.coli RNaseH (Enzy)

Incubate at 37°C for 20 min, then ice.

2. Assemble PCR reaction as follow :

Use half for PCR keep other half as backup... *extra amount made because done in parallel with HIV_VDJ project*

2.1	Assemble a PCR test reaction as follow	1x	x2per sample	x 15samples (45t)	(12t light chain)
	• dH ₂ O	21μl (to 50μl)		945(1084.46)	
	252(259.2)				
	• cDNA	5μl	-	-	
	• 5x HF buffer	10μl	450	120	
	• dNTP(10 mM) (25mM)	1μl (0.4)		45 (18)	
	12(4.8)				
	• primer up (IGHV new-short) (4μM)	6.25μl (25pmole)		281.25(225)	
	75				
	• primer low (IGHC new-short)(4μM)	6.25μl (25pmole)			
	281.25(224) 75				

Appendix B

- phusion 0.5µl 22.5
6

Add 90 per mix to 10ul cDNA, split PCR rx in 2 tube of 50, for each sample
Was low on primer, so did number in red

2.2 Thermal cycle as follows:

- 1- 98°C for 1 min
- 2- 98°C for 10 seconds
- 3- 62°C for 20 seconds
- 4- 72°C for 20 seconds go to step 2, 23x
- 5- 72°C for 5 min
- 6- 4°C pause

2.3 Repool each 2 reaction into 1

2.4 Add 2µl of ExonucleaseI (20U/µl) to each tube and incubate at 37°C for 20 min

2.5 AMPure XP purify ratio 1.8:1. Ressuspend in 40ul (lower if possible)

Conduct SPRI purification as follow:

- Add the require amount of AMPure XP beads (1.8:1 ratio) to the DNA sample in buffer EB.
- Vortex to mix
- Incubate for 5 minutes at room temperature.
- Magnet (MPC) for 5 minutes. Leave the tube of beads in the MPC during all wash steps
- Remove the supernatant (keep in case of failure)and wash the beads twice with 500 µl of 70% Ethanol,incubating for 30 sec each time.
- Remove all the supernatant, quick spin , remove last drop and allow the AMPure beads to air dry completely (2 min).
- Remove the tube from the MPC, add 40µl of 10 mM Tris-HCl, 0.1 mM EDTA, pH 8.0 (or Qiagen's Buffer EB), and vortex for 30 sec to resuspend the beads, let sit for 3 min Magnet for 2 min and transfer supernatant to a new tube.

3. Diagnostic gel (check PCR efficiency)

Run 2ul on a 2% Egel-X for 12 min

VH =410bp

VL= 360bp

Appendix B

Pool all 5 tubes. And re-concentrate with ampure down to 60ul so that it can be loaded across 2 lane on the pipipin prep.

3.1.1 Phusion PCR of SOE LLA4 primers from cDNA

Plan

Reactions

- Make a master mix of the following components.

		4.1x master mix
dH ₂ O	11	45.1
5x HF buffer (det. free)	4	16.4
10 mM dNTP	0.4	1.64
cDNA	4	16.4
Phusion Hot Start	0.2	0.82
Total	19.6	

- Aliquot 19.6 µL into 4 PCR tubes. Add 0.2 µL of each appropriate primer, as follows (8 different primer mixes total):
 - VH primers
 - LLA4 SOE VH primers
 - VL primers
 - LLA4 SOE VL primers

Cycling

- 98°C, 30 s
- 98°C, 10 s
- 65°C, 30 s
- 72°C, 30 s
- Goto 2, 29x
- 72°C, 5 min
- 4°C, forever

Results

- The gel is below. 2% E-gel EX run for 10 min.

Appendix B

Lane	Sample	Amount
M	NEB 50 bp ladder	1 μ L in 20
1		
2	VH PCR	10 μ L in 20
3	VH LLA4 SOE PCR	10 μ L in 20
4		
5	VL PCR	10 μ L in 20
6	VL LLA4 SOE PCR	10 μ L in 20
7		
8	Billy's sample	
9		
10		

3.1.2 Gradient PCR of LLA4 primers with Phusion from cDNA Reactions

- Mix the following. There are two master mixes.

Appendix B

		VH master (4.1x)	VL master (4.1x)
dH ₂ O	11	45.1	45.1
5x HF buff (det. free)	4	16.4	16.4
10 mM dNTP	0.4	1.64	1.64
Primer 1	0.2	0.82	0.82
Primer 2	0.2	0.82	0.82
cDNA	4	16.4	16.4
Phusion Hot Start	0.2	0.82	0.82
Total	20		

- Aliquot 20 µL of each mix into 4 tubes of a PCR strip, using tubes 1, 3, 5, 7.
- The strips are to be placed in positions 3-10 of the gradient block
- The gradient temperatures are:
 1. 64.9°C
 2. 66.9°C
 3. 70.0°C
 4. 72.5°C

Cycling

1. 98°C, 30 s
2. 98°C, 10 s
3. **Grad 64°C-74°C** (machine set), 30 s
4. 72°C, 30 s
5. Goto 2, **24x**
6. 4°C, forever

Results

- The gel is below: 2% E-gel EX run for 10 min

Lane	Sample	Amount
M	NEB 50 bp ladder	1 µL in 20
1		
2	VH (gradient) 64.9°C	10 µL in 20
3	VH (gradient) 66.9°C	10 µL in 20
4	VH (gradient) 70.0°C	10 µL in 20
5	VH (gradient) 72.5°C	10 µL in 20
6		

Appendix B

7	VL (gradient) 64.9°C	10 µL in 20
8	VL (gradient) 66.9°C	10 µL in 20
9	VL (gradient) 70.0°C	10 µL in 20
10	VL (gradient) 72.5°C	10 µL in 20

3.1.3 Kapa 2G Robust PCR with LLA4 from cDNA

Plan

- I will try the Kapa 2G Robust enzyme to see if it performs better using the same primer set than Phusion
- I also believe that Kapa has no exo activity, and I think that the primer dimer band may require exo in order to form robustly, so perhaps this will help avoid it.

Reactions

- Mix the following in separate PCR tubes

	VH neg control	VH	VL neg control	VL
dH ₂ O	19	15	19	15
Kapa 2G buffer A	5	5	5	5
dNTP 10 mM	0.5	0.5	0.5	0.5
Primer 1	0.2	0.2	0.2	0.2

Appendix B

Primer 2	0.2	0.2	0.2	0.2
cDNA	0	4	0	4
Kapa 2G Robust	0.1	0.1	0.1	0.1
Total	25	25	25	25

Cycling

1. 95°C, 1 min
2. 95°C, 20 s
3. 65°C, 20 s
4. 72°C, 30 s
5. Goto 2, 24x
6. 4°C, forever

Results

- Gel is below (2% E-gel EX)

Lane	Sample	Amount
M	NEB 50 bp ladder	1 µL in 20
1		
2	VH neg control	10 µL in 20
3	VH	10 µL in 20
4		
5	VL neg control	10 µL in 20
6	VL	10 µL in 20

Appendix B

3.1.4 Heavy/light chain clone matching

- Manually matched some possible heavy and light chains for our synthesis. The info is on the spreadsheet.

3.1.5 Phusion PCR using different numbers of cycles

Plan

- See if reducing the number of cycles more will help the PCRs

Reactions

		VH master (5.2x)	VL master (5.2x)
dH ₂ O	11	57.2	57.2
HF buffer (det. free buff)	4	20.8	20.8
10 mM dNTP	0.4	2.08	2.08
Primer 1	0.2	1.04	1.04
Primer 2	0.2	1.04	1.04
cDNA	4	20.8	20.8
Phusion Hot Start	0.2	1.04	1.04

- NOTE:** I ran out of one of the cDNA samples so I had to use two different cDNA samples:

Appendix B

- VH
- VL
- Aliquot 20 μ L into 5 PCR tubes for each of VH and VL
- The PCR tubes are cut apart and loaded onto a thermocycler

Cycling

- Testing different amounts of cycling: 9, 13, 17, 21, 25 cycles.
- After 9 cycles, we pull out one of the tubes for VH and VL, and then pull out a tube every 4 cycles thereafter till the last tube does 25 cycles.

The program is:

1. 98°C, 30 s
2. 98°C, 10 s
3. 72°C, 30 s
4. Goto 2, 8x
5. 20°C, forever (pull tube)
6. 98°C, 10 s
7. 72°C, 30 s
8. Goto 6, 3x
9. Goto 5, 3x
10. 4°C, forever

Results

- The gel is below (2% E-gel EX 10 min)

Lane	Sample	Amount
M	NEB 50 bp ladder	1 μ L in 20
1	VH 9 cycles	10 μ L in 20
2	VH 13 cycles	10 μ L in 20
3	VH 17 cycles	10 μ L in 20
4	VH 21 cycles	10 μ L in 20
5	VH 25 cycles	10 μ L in 20
6	VL 9 cycles	10 μ L in 20
7	VL 13 cycles	10 μ L in 20
8	VL 17 cycles	10 μ L in 20
9	VL 21 cycles	10 μ L in 20

Appendix B

10	VL 25 cycles	10 μ L in 20

3.1.6 cDNA synthesis using SuperScript III

- Used the RNA labeled PBMC RNA that Francois gave me. I finished the whole tube (which is why I used somewhat less RNA than in the previous cDNA synthesis)

Reactions

- Mix the following:

		(correct) 16.5x master mix	actual mix
total RNA (PBMC) (70 ng/ μ L)	5	82.5	66
random hexamers	1	16.5	16.5
10 mM dNTP	1	16.5	16.5
DEPC H ₂ O	3	49.5	72

- NOTE:** The actual mix is overdiluted because instead of adding 16.5 μ L water like I should've, I actually added 22.5 for some strange reason.
- I aliquoted 10 μ L of this mix into 16 PCR tubes
- Incubate @ 65°C for 5 min
- Incubate @ 4°C for >1 min
- Meanwhile, make the following mix:

Appendix B

		16.5x master
10x RT buffer	2	33
25 mM MgCl ₂	4	66
0.1 M DTT	2	33
RNaseOUT (40 U/μL)	1	16.5
SuperScript III (200 U/μL)	1	16.5

- Aliquot 10 μL into each tube
- Incubate @ 25°C, 10 min
- Incubate @ 50°C, 50 min
- Incubate @ 85°C, 5 min
- Chill on ice
- I combined all 16 reactions into a single Eppy
- I added 16 μL of RNase H and incubates @ 37°C for 20 min

3.1.7 Primer concentration gradient for Phusion PCR from cDNA

Plan

- One of the final optimizations that I want to perform before moving on to the cross-linking is checking for the optimal primer concentrations
- I will do a primer concentration gradient to check for secondary products and efficiency

Reactions

- Make the following master mix. (I can do VH and VL together because primers will be added separately.)

		16.5x master
dH ₂ O	0.2	3.3
HF buffer (det. free)	4	66
10 mM dNTP	0.4	6.6
cDNA	4	66
Phusion hot start	0.2	3.3

- Aliquot 8.8 μL into each of 16 PCR tubes (8 VH, 8 VL)
- The primers will be added as follows:

Tube	Primer conc. (nM; each primer)	Primer mix volume	dH ₂ O volume
1	12.5	0.1	5.5

Appendix B

2	25	0.2	5.4
3	40	0.32	5.28
4	70	0.56	5.04
5	125	1	4.6
6	250	2	3.6
7	500	4	1.6
8	700	5.6	0

- Note that the amounts added are for each primer mix, so that each individual tube requires 4 pipetting steps (right primer + water, left primer + water).

Cycling

1. 98°C, 30 s
2. 98°C, 10 s
3. 72°C, 30 s
4. Goto 2, 24x
5. 4°C, forever

Results

- Gels are below (2% E-gel EX):

Lane	Sample	Amount
M	NEB 50 bp ladder	1 µL in 20
1		
2	VH 12.5 nM	10 µL in 20
3	VH 25 nM	10 µL in 20
4	VH 40 nM	10 µL in 20
5	VH 70 nM	10 µL in 20
6	VH 125 nM	10 µL in 20
7	VH 250 nM	10 µL in 20
8	VH 500 nM	10 µL in 20
9	VH 700 nM	10 µL in 20
10		

Appendix B

Analysis

- The VH PCR shows a clear maximum of produced product. It is a bit difficult to see any secondary products (probably because of the slightly reduced number of cycles), but it seems that the optimal amount is probably 40-70 nM of each primer (0.32-0.56 μ L primer mix at 2.5 μ M each primer).

3.1.8 PCRs for gel purification of VH and VL chains

Plan

- Perform PCR using SOE primers and USER primers to gel extract the material

Reactions

- Prepare the following reactions:

LLA4 (SOE) primers	VH	VL
dH ₂ O	11	11
HF buffer (det. free)	4	4
dNTP 10 mM	0.4	0.4
Primer 1	0.2	0.2
Primer 2	0.2	0.2
cDNA	4	4

Appendix B

Phusion	0.2	0.2

USER primers	VH	VL
dH ₂ O	14.4	14.4
Buffer A (5x)	5	5
dNTP 10 mM ea	0.5	0.5
Primer 1	0.5	0.5
Primer 2	0.5	0.5
Ligation reaction	4	4
Kapa 2G Robust	0.1	0.1

Cycling

- LLA4 SOE
 1. 98°C, 30 s
 2. 98°C, 10 s
 3. 67°C, 30 s
 4. 72°C, 30 s
 5. Goto 2, 24x
 6. 4°C, forever
- USER
 1. 95°C, 1 min
 2. 95°C, 10 s
 3. 55°C, 30 s
 4. 72°C, 15 s
 5. Goto 2, 4x
 6. 95°C, 10 s
 7. 60°C, 30 s
 8. 72°C, 15 s
 9. Goto 6, 19x
 10. 4°C, forever

Extraction

- The ~450 bp bands are cut out and purified using the Qiagen MinElute kit.
- The DNA is eluted using 10 µL of EB buffer, and afterwards I add 9 µL of Tris-HCl to each tube.

Appendix B

Results

- 2% E-gel EX 10 min

Lane	Sample	Amount
M	NEB 50 bp ladder	1.25 μ L in 25
1		
2	SOE VH	20 μ L in 25
3		
4	SOE VL	20 μ L in 25
5		
6	USER VH	20 μ L in 25
7		
8	USER VL	20 μ L in 25
9		
10		

Appendix B

3.1.9 Assembling linked VH/VL from size-selected immune chains via SOE-PCR or USER/T4 ligase

3.1.9.1 SOE-PCR assembly

Reactions

- Mix the following.
- NOTE: DO NOT ADD THE PRIMERS. They will be added after the first 5 cycles of PCR.
- NOTE: The VH amplicons here are the LLA4 SOE primed amplicons

dH ₂ O	13
HF buffer (det. free)	4
dNTP 10 mM	0.4
Primer 1	(0.2)
Primer 2	(0.2)
VH amplicon	1
VL amplicon	1
Phusion	0.2

Cycling

1. 98°C, 30 s
2. 98°C, 10 s
3. 67°C, 30 s
4. 72°C, 30 s
5. Goto 2, 4x
6. 16°C, forever (ADD primer here)
7. 98°C, 10 s
8. 67°C, 30 s
9. 72°C, 30 s
10. Goto 7, 24x
11. 4°C, forever

3.1.9.2 USER/T4 ligase

Protocol

- NOTE: The amplicons here are the 0827 primed amplicons for USER digestion (no scFv)

Appendix B

USER digestion

- Combine the following:
 - 3 μ L VH amplicon
 - 3 μ L VL amplicon
 - 4 μ L dH₂O
 - 0.5 μ L UDG (Enzymatics)
 - 0.5 μ L Endo VIII (Enzymatics)
- Incubate at 37°C, 30 min

Ligation reaction

- To the digestion reaction, add the following:
 - 7 μ L dH₂O
 - 1 μ L 10x T4 ligase buffer
 - 1 μ L T4 ligase (NEB)
- Incubate at 16°C, 30 min
- Incubate at 65°C, 10 min

PCR amplification of linked product

- Make the following PCR mix:

dH ₂ O	13.4
Buffer A (5x)	5
dNTP 10 mM ea	0.5
Primer 1	0.5
Primer 2	0.5
Ligation reaction	5
Kapa 2G Robust	0.1

- Cycle as follows:
 1. 95°C, 1 min
 2. 95°C, 10 s
 3. 55°C, 30 s
 4. 72°C, 15 s
 5. Goto 2, 4x
 6. 95°C, 10 s
 7. 60°C, 30 s

Appendix B

8. 72°C, 15 s
9. Goto 6, 19x
10. 4°C, forever

3.1.9.3 Results

- 2% E-gel EX 10 min

Lane	Sample	Amount
M	NEB 50 bp ladder	1 µL in 20
1		
2	SOE-PCR	10 µL in 20
3		
4	USER/T4 ligation	10 µL in 20

3.1.10 Multistep PCR of VH and VL

- Attempt to get scFvs using serial PCR reactions with different primer sets

3.1.10.1 Control PCR with original primers

- Mix the following (note, separate primers for Kappa and Lambda chains)

	VH	VL

Appendix B

dH ₂ O	11.15	10.9
HF buff. (det. free; 5x)	4	4
dNTP (10 mM ea)	0.4	0.4
Primer 1 (4 μM ea)	0.125	0.125 x2 (K and L chains)
Primer 2 (4 μM ea)	0.125	0.125 x2 (K and L chains)
cDNA	4	4
Phusion	0.2	0.2

- Cycle:
 1. 98°C, 1 min
 2. 98°C, 10 s
 3. 62°C, 20 s
 4. 72°C, 20 s
 5. Goto 2, 24x
 6. 4°C, forever

3.1.10.2 Multistep PCR

PCR A

- First PCR using the original PCR outer primers
- Mix the following. Heavy and Light chains are amplified together.

	VH+VK+VL
dH ₂ O	10.65
HF buff. (det. free; 5x)	4
dNTP (10 mM ea)	0.4
Primer (4 μM ea)	0.125 x6 (all loci)
cDNA	4
Phusion	0.2

- Cycle:
 1. 98°C, 1 min
 2. 98°C, 10 s
 3. 62°C, 20 s
 4. 72°C, 20 s
 5. Goto 2, 9x
 6. 4°C, forever
- Add 1 μL Exo I and incubate for:

Appendix B

- 37°C, 30 min
- 80°C, 20 min

PCR B & C

- Inner nested PCR using the original primers. Trying with both the non-SOE and SOE versions just to test the PCR reactions.
- Mix the following

	non-SOE PCR B	SOE (LLA4) PCR C
dH ₂ O	13.6	13.6
HF buff. (det. free; 5x)	4	4
dNTP (10 mM ea)	0.4	0.4
Primer (2.5 µM ea)	0.2 x4 (VH+VL)	0.2 x4 (VH+VL)
PCR A	1	1
Phusion	0.2	0.2

- Cycle:
 1. 98°C, 30 s
 2. 98°C, 10 s
 3. 67°C, 30 s
 4. 72°C, 30 s
 5. Goto 2, 9x
 6. 4°C, forever
- Add 1 µL Exo I and incubate for:
 - 37°C, 30 min
 - 80°C, 20 min

PCR D

- We are now adding the SOE LLA4 tags to PCR B
- Mix the following

	0505 SOE (LLA4) PCR D
dH ₂ O	13.6
HF buff. (det. free; 5x)	4
dNTP (10 mM ea)	0.4
Primer (2.5 µM ea)	0.2 x4 (VH+VL)
PCR B	1
Phusion	0.2

Appendix B

- Cycle:
 1. 98°C, 30 s
 2. 98°C, 10 s
 3. 67°C, 30 s
 4. 72°C, 30 s
 5. Goto 2, 9x
 6. 4°C, forever
- Add 1 µL Exo I and incubate for:
 - 37°C, 30 min
 - 80°C, 20 min
- NOTE: The rest was performed another day
- Overlap extension cycling
 - This is some cycling with no primer to get SOE scFv fragments.
 - To PCR reactions C and D, add the following:

dNTP (10 mM ea)	0.2
Phusion	0.4

- Cycle:
 1. 98°C, 30 s
 2. 98°C, 10 s
 3. 62°C, 30 s
 4. 72°C, 30 s
 5. Goto 2, 9x
 6. 4°C, forever

Final outer-PCR

- This is the final PCR to amplify up the scFv. We do 25 cycles here.
- To the above cycled reactions, now add the following. Note, we are only adding the "outer primers":

VH left primer 2.5 µM	0.2
VL right primer 2.5 µM	0.2

- Cycle:
 1. 98°C, 30 s
 2. 98°C, 10 s
 3. 67°C, 30 s
 4. 72°C, 30 s
 5. Goto 2, 24x

Appendix B

6. 4°C, forever

- The results are in the gel below (2% EX E-gel, 10 min)

Lane	Sample	Amount
M	NEB 100 bp ladder	1 µL in 20
1		
2	VH control PCR	10 µL in 20
3	VL control PCR	10 µL in 20
4		
5	PCR A	10 µL in 20
6	PCR B	10 µL in 20
7	PCR C (final)	10 µL in 20
8	PCR D (final)	10 µL in 20

Appendix B

- 4.
5. **Full cDNA->scFv pipeline with reduced cycles**
6. **PCR from cDNA with primers**

PCR A

- Mix the following. The primers are the primers:

	VH	VL	VH+VL
dH ₂ O	11.15	10.9	10.65
HF buff. (det. free; 5x)	4	4	4
dNTP (10 mM ea)	0.4	0.4	0.4
Primer (4 μ M ea)	0.125 x2	0.125 x4 (K+L)	0.125 x6 (all loci)
cDNA	4	4	4
Phusion	0.2	0.2	0.2

- Cycle:
 1. 98°C, 30 s
 2. 98°C, 10 s
 3. 60°C, 30 s
 4. 72°C, 30 s
 5. Goto 2, 14x
 6. 10°C, forever
- Add 1 μ L Exo I and incubate for:
 - 37°C, 30 min
 - 80°C, 20 min

7. **PCR from PCR A using SOE attB primers**

PCR B

- Mix the following. Not the primers are the SOE-attB primers

	VH	VL	VH+VL
dH ₂ O	14.5	14.5	14.1
HF buff. (det. free; 5x)	4	4	4
dNTP (10 mM ea)	0.4	0.4	0.4
Primer (2.5 μ M ea)	0.2 x2	0.2 x2	0.2 x4
PCR A	0.5	0.5	0.5
Phusion	0.2	0.2	0.2

- Cycle:

Appendix B

1. 98°C, 30 s
 2. 98°C, 10 s
 3. 65°C, 30 s
 4. 72°C, 30 s
 5. Goto 2, 14x
 6. 10°C, forever
- Add 1 µL Exo I and incubate for:
 - 37°C, 30 min
 - 80°C, 20 min
- NOTE: The rest of the experiment to be finished tomorrow.
8. **PCR from PCR B using attB universal primers**

PCR C

- NOTE: The individual VH and VL reactions must now be combined to get scFvs. This combined reaction will be labeled "VH/VL sep."
- Mix the following:

	VH/VL sep.	VH+VL
dH ₂ O	12.4	12.9
HF buff. (det. free; 5x)	4	4
dNTP (10 mM ea)	0.4	0.4
PCR B	0.5 x2 (VH and VL)	0.5
Phusion	0.2	0.2
Total	18	18

- Add 1 µL each 10 µM universal attB primers at pause point
- Cycle:
 1. 98°C, 30 s
 2. 98°C, 10 s
 3. 67°C, 30 s
 4. 72°C, 30 s
 5. Goto 2, 9x
 6. 10°C, forever

Add primer here

 7. 98°C, 30 s
 8. 98°C, 10 s
 9. 63°C, 30 s

Appendix B

10. 72°C, 30 s
11. Goto 8, 21x
12. 10°C, forever

- Run on 2% E-gel EX for 10 min

Lane	Sample	Amount
M	NEB 100 bp ladder	1 µL in 20
1		
2	VH/VL sep.	10 µL in 20
3		
4	VH+VL	10 µL in 20

Appendix C

VDJome/ Immune-seq / light chain only

Conduct light chain library prep on sample of day 1, 7 and 21 only

1. Sample ID

Sample were extracted from 9 ML of blood from a single individual at various time point and processed using the Leukolock extraction kit from Ambion:

Then samples were concentrated using RNeasy kit form Qiagen, elute in 2x 30µl and nanodro

2. Light chain , the real deal

Used original sample

ID		for 500ng	water to 7
3	90.57ng/ul	5.52	1.48
5	84.74ng/ul	5.9	1.1
7	71.89ng/ul	6.96	0
FV1	180ng/ul	2.78	4.22

Or second batch

	for 500ng	water to 7
	2.0	5
	2.16	4.84
	2.7	4.3

ID	MID	vaccination sample	concentration (ng/µl)	260/280	Vol for 500ng	H2O for 9.5µl
1	1	-n	84.28	2.13	8.93	0.57
2	2	0	63.15	2.12	7.92	1.58
3	3	24hr	90.57	2.13	5.52	3.98
4	4	3days	94.84	2.15	5.27	4.23
5	5	7days	84.74	2.14	5.90	3.60
6	6	14days	72.97	2.13	6.85	2.65
7	8	21days	71.89	2.14	6.96	2.54
8	9	28days	79.11	2.17	6.32	3.18
9	10	FV test sample	133.09	2.16	3.75	5.75

Then samples were concentrated using RNeasy kit formQiagen, elute in 2x 30µl and nanodrop

VDJ-ID	MID	vaccination sample	concentration (ng/µl)	Bioanalyzer- RNA nano	RIN	Vol for 1000ng	H2O for 9.5µl
1	1	-n	128.32	128	7.20	7.79	1.71
2	2	0	167.18	163	6.10	5.98	3.52
3	3	24hr	251.42	287	7.20	3.98	5.52
4	4	3days	263.68	200	6.60	3.79	5.71
5	5	7days	231.84	235	8.90	4.31	5.19
6	6	14days	193.86	165	7.70	5.16	4.34
7	8	21days	185.23	186	7.60	5.40	4.1
8	9	28days	193.29	156	6.30	5.17	4.33

Tube	ID	notes
1	old-3	

Appendix C

- 2 old-5
- 3 old-7 barely nothing 1ul maybe
- 4 FV1
- 5 rneasy-3
- 6 rneasy-5
- 7 rneasy-7

3. Reverse transcription; Proceed as follow for RT-PCR

- 3.1 Assemble a RT reaction as follow; 2x 1tube
- H₂O(DEPC) 0μl(to 12μl)
 - 2μM Gene-specific primer(IGHC K and L) 5 (5pmole)
 - Total RNA 7 μl (500ng)
- Heat at 95°C for 1min followed by
65°C for 5 min than ice for 1 min
- Spin down and Add the following: 1tube 9tubes
- 5x First strand buffer 4μl 36
 - 10mM dNTP mix 1μl 9
 - 0.1M DTT 1μl 9
 - RnaselInhibitor-Enzy 1μl 9
 - Superscript III 1μl 9
- Incubate at 55°C for 60 min (8μl/tubes)
- 3.2 Inactivate enzyme by heating at 70°C for 15 min
- 3.3 Remove RNA/DNA hybrid by adding 1μl of E.coli RNaseH (Enzy)
Incubate at 37°C for 20 min, then ice.

4. Assemble PCR reaction as follow :

Use half for PCR keep other half as backup

- 4.1 Assemble a PCR test reaction as follow
- | | <u>1x</u> | <u>x2per sample x 7samples (16t)</u> |
|------------------------------------|------------------|--------------------------------------|
| • dH ₂ O | 21μl (to 50μl) | 336 |
| • cDNA | 5μl | - |
| • 5x HF buffer | 10μl | 160 |
| • dNTP(10 mM) | 1μl | 16 |
| • primer up (IGHV new-short) (4μM) | 6.25μl (25pmole) | 100 (50of each K and L) |
| • primer low (IGHC new-short)(4μM) | 6.25μl (25pmole) | 100 |
| • phusion | 0.5μl | 8 |

Add 90 pcr mix to 10ul cDNA, then split in 2 tube of 50 for each sample

- 4.2 Thermal cycle as follows:
- 1- 98°C for 1 min
 - 2- 98°C for 10 seconds
 - 3- 62°C for 20 seconds
 - 4- 72°C for 20 seconds go to step 2, 23x
 - 5- 72°C for 5 min
 - 6- 4°C pause

Appendix C

Freeze sample

5. Diagnostic gel (check PCR efficiency)

Run 2ul on a 2% Egel-X for 12 min (did 11 min)

5.1 Add 2µl of ExonucleaseI (20U/µl) to each tube and incubate at 37°C for 45 min (did 20 min)

5.2 AMPure XP purify ratio 1.8:1. Ressuspend in 40ul (keep non-bind volume as backup)
Save 2ul of each sample for gel diagnostic

Conduct SPRI purification as follow:

- Add the require amount of AMPure XP beads (1.8:1 ratio) to the DNA sample in buffer EB.
- Vortex to mix
- Incubate for 5 minutes at room temperature.
- Magnet (MPC) for 5 minutes. Leave the tube of beads in the MPC during all wash steps
- Remove the supernatant (keep in case of failure)and wash the beads twice with 500 µl of 70% Ethanol,incubating for 30 sec each time.
- Remove all the supernatant, quick spin , remove last drop and allow the AMPure beads to air dry completely (2 min).
- Remove the tube from the MPC, add 40µl of 10 mM Tris-HCl, 0.1 mM EDTA, pH 8.0 (or Qiagen's Buffer EB), and vortex for 30 sec to resuspend the beads, let sit for 3 min Magnet for 2 min and transfer supernatant to a new tube.

5.3 Diagnostic gel (just to check efficiency of
AMPure beads)

Run 2ul on a 2% Egel-X for 12 min

First 4 lanes

Looks good

6. Blunt ends

6.1 Perform blunting reaction using Enzymatics End repair kit as follows:

	1t
• H2O	0µl(to 50µl)
• Purified DNA	38µl
• 10× End repair bufferbuffer	5ul
• 1mM dNTP mix	5µl
• enzyme mix.	5µl

Incubate at 25°C temperature for 30 min

Heat inactivate at 75°C for 20 min.

6.2 AMPure XP purify ratio 1.8:1. Ressuspend in 37ul (keep non-bind volume as backup)

Appendix C

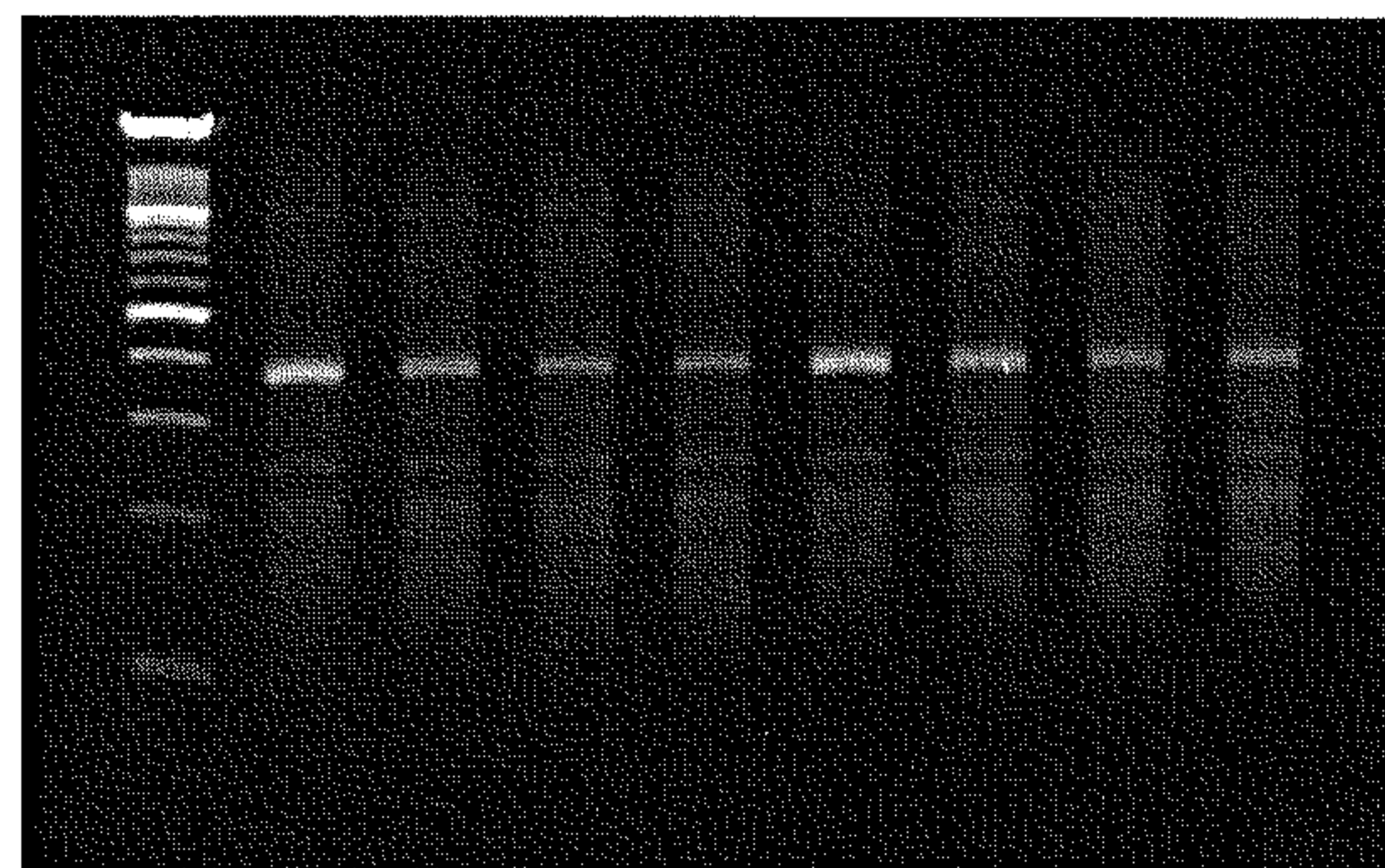
Save 2ul of each sample for gel diagnostic

- 6.3 Diagnostic gel (check efficiency of blunting and ampure beads)

Run 2ul on a 2% Egel-X for 12 min

Last 4 lanes

Looks good



7. Adaptor ligation (25x excess)

- 7.1 Self Anneal 454 oligo into adapter A and B together

Follow 454 protocol (10μl of 100μM of each primer + 30μl of (10mM Tris 0.1mM EDTA at 50mM NaCL)) (each adapter are at 20μM final), 95 for 3 min, ramp to 15 forever at 0.1°C/sec

Prepare adapter A MID 3-5-8 and adapter B only.

- 7.2 Ligation of adapter 454-A and 454-B (454 use a 15:1 ratio of adapter DNA)

	1t
• H2O	10μl (to 100μl)
• 2x Quick ligase buffer	50μl
• VDJ DNA	35μl (~1.25 pmole at 400bp)
• 20μM 454-A	1.5μl (~30pmole)
• 20μM 454-B	1.5μl (~30pmole)
• T4 DNA ligase (quick)	2μl

Incubate on PCR at 22°C for 30 min

- 7.3 AMPure XP purify ratio 1.8:1. Ressuspend in 37ul

- 7.4 Diagnostic gel (test)

Run 2ul on a 2% Egel-X for 12 min

100bp ladder

1- FV1 pre ligation

2-5 post ligation, post purification

8. Fill-In Reaction

*If using BST be aware of the presence of a A-tail at the 3' ends

- 8.1 In a microcentrifuge tube, add the following reagents, in the order indicated, and mix:

	1t	5t
• H2O	5.5 (to 50μl)	27.5
• DNA	35	-
• 10x Thermopol bufferI	2.5μl	12.5

Appendix C

• 10x Phi29 buffer	2.5	12.5
• 100x BSA (10mg/ml)	0.5	2.5
• 10mM dNTP Mix	1 µl (0.2mM)	5
• Bst Polymerase (8U/µl)	2 µl	10
• Phi29	1ul	5

Incubate at 30°C for 10 min, then 60°C for 15 min (thermocycler)
Heat inactivate at 80°C for 15 min

8.2 AMPure XP purify ratio 1.8:1. Ressuspend in 30ul

9. Gel extraction of heavy chain-ligated library

9.1 Test loading capacity with FV1 sample only, load 10ul 5 ul and 2.5 ul (space between each lane)
Run on a 2% e-gel ex, run 15 min

9.2 Load other sample across 3 lanes (10ul/lanes)

9.3 Use qiagen buffer gel extraction kit (melt band at room temp on thermomixer)
Elute in 17 ul (use mini-elute column)

10. QC and pooling

10.1 Nanodrop: number are weird (230 is real high)
10.2 Add 34ul of TE, AMPure XP purify ratio 1.8:1. Ressuspend in 20ul (remagnet manually in PCR tube)

10.3	Nanodrop ID	Qubit dsDNA HS	area density
	ng/ul		
FV1	6.4	9.84	
24hr	7.6	16.6	18.0=15.6
7days	7.9	13.8	15.9=13.8
21days	9.1	16.0	18.7=16.23

10.4 Diagnostic gel (2% e gel ex 11 min, 2ul each sample)
(sage lader, 100bp lader, 25bp ladder, VDJ samples)

Dilute all sample to same concentration of lowestng/ul;

Sample volumeSwaterQbit water gel

Appendix C

24hr	12ul	2.43	1.56
7days	12ul	0	0
21days	12ul	1.91	2.11

10.5 Mix 12 ul of each sample together. Store at -20

vdj_package.txt

Attachment D

```

import types
import xml.etree.cElementTree as ElementTree

import numpy as np
import

Bio.SeqIO

import seqtools

# import refseq
# import sequtils
# import alignment
# import

clustering
# import LSF
# import params
# import analysis

# =====
# = DATA

STRUCTURES =
# =====

class ImmuneChain(object):
    """Data structure to
    represent an immune chain."""
    def __init__(self,**kw):
        """Initialize
        ImmuneChain
        seq is 5'->3'
        """
        def kw_init(attrib):

            if kw.has_key(attrib):
                self.__setattr__(attrib,kw[attrib])

        kw_init('seq')
        kw_init('descr')
        kw_init('locus')
        kw_init('v')

        kw_init('d')
        kw_init('j')
        kw_init('c')
        kw_init('junction')

```



```

                                vdj_package.txt
        if kw.has_key('tags'):
            tags = kw['tags']
            if
isinstance(tags,types.StringTypes): tags = [tags]
            self.tags = set(tags)

else:
    self.tags = set([])

    def get_cdr3(self):
        return
len(self.junction)
    def set_cdr3(self,value):
        pass
    cdr3 =
property(fget=get_cdr3,fset=set_cdr3)
    def add_tags(self,tagset):
        if
isinstance(tagset,types.StringTypes): tagset = [tagset]
        self.tags.update(tagset)

    def add_tag(self,tag):
        self.add_tags(tag)

    def
remove_tags(self,tagset):
        if isinstance(tagset,types.StringTypes): tagset =
[tagset]
        for tag in tagset: self.tags.remove(tag)

    def
remove_tag(self,tag):
        self.remove_tags(tag)

    def has_tag(self,tag):

if tag in self.tags:
    return True
    else:
        return False

def __len__(self):
    return len(self.seq)

    def __str__(self):
        return
self.__repr__()

    def __repr__(self):
        return self.get_XML()

```


vdj_package.txt

```

def
get_XML(self):
    format_xml = lambda attrib,value:
"\t<%(attrib)s>%(value)s</%(attrib)s>\n" % {'attrib':attrib,'value':value}
    xmlstring
= '<ImmuneChain>\n'
    for (attrib,value) in self.__dict__.iteritems():
        if
attrib == 'tags':
            for tag in self.tags:
                xmlstring +=
format_xml('tag',tag)
            else:
                xmlstring +=
format_xml(attrib,value)
        xmlstring += '</ImmuneChain>\n'
    return
xmlstring

# =====
# = INPUT/OUTPUT =
# =====

class
ParserVDJXML(object):
    """Parser for VDJXML"""
    def __init__(self):
        self.chain

= None

    def start_handler(self,elem):
        if elem.tag == 'ImmuneChain':

        self.chain = ImmuneChain()

    def end_handler(self,elem):
        if elem.tag ==

'tag':
            self.chain.add_tags(elem.text)
            elif elem.tag == 'v_end_idx' or
elem.tag == 'j_start_idx':
                self.chain.__setattr__(elem.tag,int(elem.text))

        else:
            self.chain.__setattr__(elem.tag,elem.text)

    def

```


vdj_package.txt

```

parse(self,inputfile):
    for event, elem in
ElementTree.iterparse(inputfile,events=('start','end')):
    if event == 'start':

        self.start_handler(elem)
    elif event == 'end':
        if

elem.tag == 'ImmuneChain':
    yield self.chain
    else:

        self.end_handler(elem)

class PredicateParserVDJXML(ParserVDJXML):

    """VDJXML Parser that takes a predicate function"""
    def __init__(self,predicate):

        ParserVDJXML.__init__(self)
        self.predicate = predicate

    def
parse(self,inputfile):
    for event, elem in
ElementTree.iterparse(inputfile,events=('start','end')):
    if event == 'start':

        self.start_handler(elem)
    elif event == 'end':
        if

elem.tag == 'ImmuneChain':
    if self.predicate(self.chain) == True:

        yield self.chain
    else:

self.end_handler(elem)

def parse_VDJXML(inputfile):
    vdxmlparser = ParserVDJXML()

    return vdxmlparser.parse(inputfile)

def filter_parse_VDJXML(inputfile,predicate):

```



```

                                vdj_package.txt

vdjxmlparser = PredicateParserVDJXML(predicate)
    return vdjxmlparser.parse(inputfile)

#

=====
# = Counting =
# =====

def counts_VJ(inputfile):
    if

instance(inputfile,types.StringTypes):
        ip = open(inputfile,'r')
    elif

instance(inputfile,file):
        ip = inputfile

    counts = np.zeros(
(len(refseq.IGHV_list),len(refseq.IGHJ_list)) )
    for chain in parseVDJXML(ip):

counts[refseq.IGHV_idx[chain.v],refseq.IGHJ_idx[chain.j]] += 1
    if

instance(inputfile,types.StringTypes):
        ip.close()

    return counts

def

counts_VDJ(rep):
    cn = np.zeros(
(len(refseq.IGHV_list),len(refseq.IGHD_list),len(refseq.IGHJ_list)) )
    for chain in

rep.chains:

cn[refseq.IGHV_idx[chain.v],refseq.IGHD_idx[chain.d],refseq.IGHJ_idx[chain.j]] += 1

return cn

def reshape_counts_VDJ_2D(counts):
    return

counts.reshape(len(refseq.IGHV_list),len(refseq.IGHD_list)*len(refseq.IGHJ_list))

def

count_dict_clone_idx(clone_idx,reference_clones=None):

```



```

        vdj_package.txt
    """Takes a dictionary of
cluster names mapped to a sequence of indices into an ImmuneChain list.

Returns an
np array of the same length as reference_clones with the counts of each
cluster in
reference_clones.

The need for reference_clones is due to the fact that splitting a
given repertoire
may result in some parts not observing any of a given clone, so there
needs to be a common way
to compare two clone sets.

If reference_clones is left
out, then the set of clones present in clone_idx is used.
    """
    if
reference_clones == None:
        reference_clones = clone_idx.keys()
        counts =
np.zeros(len(reference_clones))
        for (i,name) in enumerate(reference_clones):

counts[i] = len(clone_idx.get(name,[]))
        return counts

def
count_dict_clone_counts(clone_counts,reference_clones=None):
    if reference_clones ==
None:
        reference_clones = clone_counts.keys()
        counts =
np.zeros(len(reference_clones))
        for (i,name) in enumerate(reference_clones):

counts[i] = clone_counts.get(name,0)
        return counts

#

=====
# = Retrieving tags and filtering =
#
=====

```



```

                                vdj_package.txt
def get_tag_with_prefix(chain, prefix):
    for tag in
chain.tags:
    if tag.startswith(prefix):
        return tag
    raise ValueError,
"Tag that starts with " + prefix + " not found."

def get_clone(chain):
    return
get_tag_with_prefix(chain, 'clone')

def get_barcode(chain):
    try:
        return
    except ValueError:
        return ''

def
filter_tags_and(tags, inhandle, outhandle):
    if isinstance(tags, types.StringTypes): tags =
[tags]
    tags = set(tags)
    for chain in parse_VDJXML(inhandle):
        if tags <=
chain.all_tags:    # test that everything in tags is in all_tags
    print
>>outhandle, chain

def filter_tags_or(tags, inhandle, outhandle):
    if
isinstance(tags, types.StringTypes): tags = [tags]
    tags = set(tags)
    empty_set =
set()
    for chain in parse_VDJXML(inhandle):
        if tags & chain.all_tags !=
empty_set:    # test that tags and all_tags share something
        print >>outhandle,
chain

def is_full_VJ(chain):
    if (chain.v in refseq.IGHV_seqs.keys()) and (chain.j in
refseq.IGHJ_seqs.keys()):

```



```

                                vdj_package.txt
        return True
    else:
        return False

def
get_clone_idxes(inhandle):
    clusters = {}
    i = 0
    for chain in
parse_VDJXML(inhandle):
    try: clusters[get_clone(chain)] += [i]
    except

KeyError: clusters[get_clone(chain)] = [i]
        i += 1
    return clusters

def
get_clone_counts(inhandle):
    clusters = {}
    for chain in parse_VDJXML(inhandle):

    try: clusters[get_clone(chain)] += 1
    except KeyError: clusters[get_clone(chain)] =

1
    return clusters

# =====
# = Pipeline functions =
#
=====

def vjxml2fasta(inhandle,outhandle):
    for chain in

parse_VDJXML(inhandle):
    print >>outhandle, '>'+chain.descr
    print

>>outhandle, chain.seq

# for generating identifiers from VJ combos
def
vj_id(v_seg,j_seg):
    return seqtools.cleanup_id(v_seg)+'_'+seqtools.cleanup_id(j_seg)

def split_vjxml_into_VJ_parts(inhandle,outname):
    parts = []
    vj_ids = []

```



```

                                vdj_package.txt
outhandles = {}

    # open output files for all vj combos
    i = 0
    for v_seg in
refseq.IGHV_seqs.keys():
    for j_seg in refseq.IGHJ_seqs.keys():

curr_outname = outname + '.' + str(i)
    curr_vj_id = vj_id(v_seg,j_seg)

    parts.append(curr_outname)
    vj_ids.append(curr_vj_id)

outhandles[curr_vj_id] = open(curr_outname,'w')
    i += 1

    for chain in
parse_VDJXML(inhandle):
    curr_vj_id = vj_id(chain.v,chain.j)
    print

>>outhandles[curr_vj_id], chain
    for handle in outhandles.itervalues():

handle.close()

    return (parts,vj_ids)

def parse_VDJXML_parts(parts):
    for part
in parts:
    for chain in parse_VDJXML(part):
        yield chain

def
wait_for_subprocesses(process_list,interval=30):
    finished = False
    while not
finished:
        finished = True
        time.sleep(interval)
        for p in
process_list:
            if p.poll() == None:
                finished = False

    break
import warnings

```



```

import copy
import numpy as np
import seqtools
import vdj
import
refseq
import sequtils
import alignmentcore
warnings.simplefilter('always')
class
vdj_aligner(object):
    def __init__(self,**kw):

self.numCrudeVCandidates = 5
    self.numCrudeDCandidates = 10

self.numCrudeJCandidates = 2
    # Define seed patterns

patternA='111011001011010111'
    patternB='1111000100010011010111'

patternC='111111111111'
    patternD='110100001100010101111'

patternE='1110111010001111'
    self.seedpatterns =

[patternA,patternB,patternC,patternD,patternE]
    self.miniseedpatterns =

['111011','110111']
    self.patternPos = '111111111111'

    self.minVscore

= 100    # derived from calibration data 20090710
    self.minDscore = 4

self.minJscore = 13

    # set reference sequences (locus) and generate hashes
from ref data
    self.locus = kw['locus']

    self.refv_seqs =

```



```

                                vdj_package.txt

refseq.__getattr__(self.locus+'V_seqs')
    self.Vseqlistkeys =

vdj_aligner.seqdict2kmers( self.refV_seqs, self.seedpatterns )

self.refJ_seqs = refseq.__getattr__(self.locus+'J_seqs')
    self.Jseqlistkeys =

vdj_aligner.seqdict2kmers( self.refJ_seqs, self.seedpatterns )

    try:    #
this locus may not have D segments
    self.refD_seqs =

refseq.__getattr__(self.locus+'D_seqs')
    self.Dseqlistkeysmini =

vdj_aligner.seqdict2kmers( self.refD_seqs, self.miniseedpatterns )
    except

AttributeError:
    pass

    self.refV_offset =

refseq.__getattr__(self.locus+'V_offset')
    self.refJ_offset =

refseq.__getattr__(self.locus+'J_offset')

    # Generate reference data

for positive sequence ID
    posVseqlistkeys = vdj_aligner.seqdict2kmers(
self.refV_seqs, [self.patternPos] )
    posJseqlistkeys = vdj_aligner.seqdict2kmers(
self.refJ_seqs, [self.patternPos] )
    negVseqlistkeys = vdj_aligner.seqdict2kmers(
vdj_aligner.seqdict2revcompseqdict(self.refV_seqs), [self.patternPos] )

negJseqlistkeys = vdj_aligner.seqdict2kmers(
vdj_aligner.seqdict2revcompseqdict(self.refJ_seqs), [self.patternPos] )

    #

collect possible keys
    posset = set([])
    for key in posVseqlistkeys.keys():

        posset.update(posVseqlistkeys[key][self.patternPos])
    for key in

posJseqlistkeys.keys():

```



```

                                vdj_package.txt
posset.update(posVseqlistkeys[key][self.patternPos])

negset = set([])
for key in negVseqlistkeys.keys():

negset.update(negVseqlistkeys[key][self.patternPos])
    for key in
negJseqlistkeys.keys():
    negset.update(negJseqlistkeys[key][self.patternPos])

    # get keys unique to positive or negative versions of reference set

possetnew = posset - negset
negsetnew = negset - posset

    self.posset
= possetnew
    self.negset = negsetnew

def
valign_chain(self,chain,verbose=False):
    query = chain.seq

    # compute
hashes from query seq
    querykeys = vdj_aligner.seq2kmers(query,self.seedpatterns)

    # for each reference V segment and each pattern, how many shared k-mers are
there?
    vscores_hash = vdj_aligner.hashscore(self.vseqlistkeys,querykeys)

    # get numCrudeVCandidates highest scores in vscores and store their names in
descending
order
    goodVseglist = sorted(self.refV_seqs.keys(),key=lambda k:
vscores_hash[k],reverse=True)[0:self.numCrudeVCandidates]
    goodVsegdict =
dict([(seg,self.refV_seqs[seg]) for seg in goodVseglist])

    #
Needleman-Wunsch of V segment
    (bestVseg,bestVscore,bestVscoremat,bestVtracemat) =

```



```

                                vdj_package.txt
vdj_aligner.bestalignNW(goodVsegdict,query,self.minVscore)
    # if successful
alignment
    if bestVseg is not None:
        chain.v = bestVseg

    # find CDR3 boundary
    # construct alignment

valnref,valnrefcoords,valnquery,valnquerycoords = vdj_aligner.construct_alignment(
self.refV_seqs[bestVseg], query, bestVscoremat, bestVtracemat )
    #
find CDR3 boundary
    chain.v_end_idx = vdj_aligner.pruneVregion( valnref,
valnrefcoords, valnquery, valnquerycoords, self.refV_offset[bestVseg] )

return bestVscore

def jalign_chain(self,chain,verbose=False):
    # try
    pruning off v region for j alignment
    try:
        query =
chain.seq[chain.v_end_idx:]
    except AttributeError:
        query = chain.seq

    # compute hashes from query seq
    querykeys =
vdj_aligner.seq2kmers(query,self.seedpatterns)
    # for each reference j
    segment and each pattern, how many shared k-mers are there?
    jscores_hash =
vdj_aligner.hashscore(self.jseqlistkeys,querykeys)
    # get
numCrudeJCandidates highest scores in jscores and store their names in descending
order

goodJseglist = sorted(self.refj_seqs.keys(),key=lambda k:

```



```

                                vdj_package.txt
jscores_hash[k],reverse=True)[0:self.numCrudeJCandidates]
    goodJsegdict =
dict([(seg,self.refJ_seqs[seg]) for seg in goodJseglist])
    #
Needleman-Wunsch of J segment
    (bestJseg,bestJscore,bestJscoremat,bestJtracemat) =
vdj_aligner.bestalignNW(goodJsegdict,query,self.minJscore)
    # if successful
alignment
    if bestJseg is not None:
        chain.j = bestJseg

    # find CDR3 boundary
    # construct alignment

Jalnref,Jalnrefcoords,Jalnquery,Jalnquerycoords = vdj_aligner.construct_alignment(
self.refJ_seqs[bestJseg], query, bestJscoremat, bestJtracemat )
    #
find CDR3 boundary
    j_start_offset = vdj_aligner.pruneJregion( Jalnref,
Jalnrefcoords, Jalnquery, Jalnquerycoords, self.refJ_offset[bestJseg] )
    try:

        chain.j_start_idx = chain.v_end_idx + j_start_offset
    except
AttributeError:
        chain.j_start_idx = j_start_offset

    return

bestJscore

def Dalign_chain(self,chain,verbose=False):
    # prune off V

and J regions for D alignment
    # we should not be attempting D alignment unless we
have
    # a well-defined CDR3
    query = chain.junction

    # compute
hashes from query seq
    querykeys =

```



```

                                vdj_package.txt
vdj_aligner.seq2kmers(query,self.miniseedpatterns)
    # for each reference D
segment and each pattern, how many shared k-mers are there?
    Dscores_hash =
vdj_aligner.hashscore(self.Dseqlistkeysmini,querykeys)
    # get
numCrudeJCandidates highest scores in Jscores and store their names in descending
order

    goodDseglist = sorted(self.refD_seqs.keys(),key=lambda k:
Dscores_hash[k],reverse=True)[0:self.numCrudeJCandidates]
    goodDsegdict =
dict([(seg,self.refD_seqs[seg]) for seg in goodDseglist])
    #
Needleman-wunsch of J segment
    (bestDseg,bestDscore,bestDscoremat,bestDtracemat) =
vdj_aligner.bestalignSW(goodDsegdict,query,self.minDscore)
    # if successful
alignment
    if bestDseg is not None:
        chain.d = bestDseg

return bestDscore

def align_chain(self,chain,verbose=False):

if not chain.has_tag('positive') and not chain.has_tag('coding'):

warnings.warn('chain %s may not be the correct strand' % chain.descr)

scores = {}
    scores['v'] = self.Valign_chain(chain,verbose)

scores['j'] = self.Jalign_chain(chain,verbose)
    # only process junction if v
and J successful
    if hasattr(chain,'v') and hasattr(chain,'j'):

```


vdj_package.txt

```

chain.junction = chain.seq[chain.v_end_idx:chain.j_start_idx]
    #
only align D if I am in a locus that has D chains
    if self.locus in
['IGH', 'TRB', 'TRD']:
    scores['d'] = self.Dalign_chain(chain, verbose)

    return scores

    def align_seq(self, seq, verbose=False):
        chain =
vdj.ImmuneChain(descr='sequence', seq=seq)
        self.align_chain(chain, verbose)

return chain

    def coding_chain(self, chain, verbose=False):
        strand =
self.seq2coding(chain.seq)
        if strand == -1:
            chain.seq =
seqtools.reverse_complement(chain.seq)
            chain.add_tag('revcomp')

chain.add_tag('coding')

    def seq2coding(self, seq):
        seqkeys =
vdj_aligner.seq2kmers(seq, [self.patternPos])
        seqwords = seqkeys[self.patternPos]

strandid = 1
        if len(self.negset & seqwords) > len(self.posset & seqwords):

strandid = -1
        return strandid

    @staticmethod
    def
seq2kmers(seq, patterns):
        """Given sequence and patterns, for each pattern, compute
all corresponding k-mers from sequence.

        The result is

```



```

                                vdj_package.txt
seqannot[pattern][key]=[pos1,pos2,...,posN] in seq
                                seqkeys[pattern] =
set([kmers])
    """
    seqkeys = {}
    patlens = []
    for
pattern in patterns:
    patlens.append(len(pattern))
    seqkeys[pattern] =
set()

    maxpatlen = max(patlens)
    for i in xrange(len(seq)):

        word = seq[i:i+maxpatlen]
        for pattern in patterns:

            patlen = len(pattern)
            if len(word) >= patlen:
                key = ''

                for j in xrange(patlen):
                    if pattern[j] == '1':

                        key += word[j]
                        seqkeys[pattern].add(key)

    return seqkeys

@staticmethod
def seqdict2kmers(seqdict,patterns):

    seqlistkeys = {}
    for seq in seqdict.iteritems():

seqlistkeys[seq[0]] = vdj_aligner.seq2kmers(seq[1],patterns)
    return seqlistkeys

@staticmethod
def hashscore(refkeys,querykeys):
    """Compute number of
common keys for each reference sequence.
    querykeys is dict of sets, where dict
keys are patterns
    reference keys is dict of ref seqs, where each elt is a

```


vdj_package.txt

dict of patterns with sets as values. the patterns must be
the same
"""

```

    scores = {}
    for seg in refkeys.iterkeys():
        score = 0
        for
pattern in querykeys.iterkeys():
            score += len( refkeys[seg][pattern] &
querykeys[pattern] )
            scores[seg] = score
        return scores

@staticmethod
def bestalignNW(candidatedict, query, minscore):
    bestseg = None

    bestscore = minscore
    bestscoremat = None
    besttracemat = None

    seq2 = query
    for (seg, seq1) in candidatedict.iteritems():
        # C

implementation:
    # carve out memory
    # note that we are using zero

initial conditions, so matrices are initialized too
    # notation is like Durbin

p.29
    scores = np.zeros( [len(seq1)+1, len(seq2)+1] )
    Ix = np.zeros(
[ len(seq1)+1, len(seq2)+1] )
    Iy = np.zeros( [len(seq1)+1, len(seq2)+1] )

    trace = np.zeros( [len(seq1)+1, len(seq2)+1], dtype=np.int)

alignmentcore.alignNW( scores, Ix, Iy, trace, seq1, seq2 )

    currscore =
vdj_aligner.scoreVJalign(scores)
    if currscore > bestscore:
```



```

                                vdj_package.txt
bestscore = currscore
                bestseg = seg
                bestscoremat = scores

                besttracemat = trace

        return

(bestseg,bestscore,bestscoremat,besttracemat)

    @staticmethod
    def
bestalignsw(candidatedict,query,minsore):
    bestseg = None
    bestscore =
minsore
    bestscoremat = None
    besttracemat = None

    seq2 =
query
    for (seg,seq1) in candidatedict.iteritems():
        # C implementation:

        # carve out memory
        # note that we are using zero initial conditions,
so matrices are initialized too
        # notation is like Durbin p.29

scores = np.zeros( [len(seq1)+1, len(seq2)+1] )
        trace = np.zeros( [len(seq1)+1,
len(seq2)+1], dtype=np.int)
        alignmentcore.alignsw( scores, trace, seq1, seq2 )

        currscore = vdj_aligner.scoreDalign(scores)
        if currscore
> bestscore:
                bestscore = currscore
                bestseg = seg

        bestscoremat = scores
                besttracemat = trace

        return

(bestseg,bestscore,bestscoremat,besttracemat)

    @staticmethod
    def

```


vdj_package.txt

```
pruneVregion( alnref, alnrefcoords, alnquery, alnquerycoords, offset ):
    """Prune V
```

```
region out of query sequence based on alignment.
```

```
    Given ref and query
```

```
alignments of V region, refID, and the original
query sequence, return a sequence
```

```
with the V region cut out, leaving
the 2nd-CYS. Also needs query alignment coords.
```

```
    """
```

```
    # FR3end = self.refV_offset[refID] - alnrefcoords[0]          #
```

```
first candidate position
```

```
    FR3end = offset - alnrefcoords[0]          # first
```

```
candidate position
```

```
    refgaps = alnref[:FR3end].count('-')      # count gaps up to
```

```
putative CYS pos
```

```
    seengaps = 0
```

```
    while refgaps > 0:      # iteratively find all
```

```
gaps up to the CYS
```

```
        seengaps += refgaps
```

```
        FR3end    += refgaps      #
```

```
adjust if for gaps in ref alignment
```

```
    refgaps = alnref[:FR3end].count('-') -
```

```
seengaps      # any add'l gaps?
```

```
    querygaps = alnquery[:FR3end].count('-')
```

```
    # v_end_idx = idx of start of aln of query + distance into aln - # of gaps
```

```
v_end_idx = alnquerycoords[0] + FR3end - querygaps
```

```
    return v_end_idx
```

```
@staticmethod
```

```
def pruneJregion( alnref, alnrefcoords, alnquery, alnquerycoords,
```

```
offset ):
```

```
    """Prune J region out of query sequence based on alignment.
```

```
    Given ref and query alignments of J region, refID, and the original
query
```



```

                                vdj_package.txt
sequence, return a sequence with the J region cut out, leaving
    the J-TRP. Also
needs query alignment coords.

    """
    # FR4start =
self.refJ_offset[refID] - alnrefcoords[0] # first candidate position of J-TRP start

FR4start = offset - alnrefcoords[0] # first candidate position of J-TRP start

refgaps = alnref[:FR4start].count('-') # count gaps up to putative TRP pos
seengaps
= 0
    while refgaps > 0: # iteratively find all gaps up to the TRP

seengaps += refgaps
    FR4start += refgaps # adjust for gaps in ref alignment

    refgaps = alnref[:FR4start].count('-') - seengaps # any add'l gaps?

    querygaps = alnquery[:FR4start].count('-')
    # j_start_offset = idx of
start of aln of query + distance into aln - # of gaps
    # note: j_start_offset is from
the pruned query seq
    j_start_offset = alnquerycoords[0] + FR4start - querygaps

    return j_start_offset

    @staticmethod
    def
construct_alignment(seq1,seq2,scoremat,tracemat):
    """Construct alignment of ref
segment to query from score and trace matrices."""
    nrows,ncols = scoremat.shape

    # do some error checking
    if len(seq1)+1 != nrows or len(seq2)+1 !=
ncols:
        raise Exception, "nrows and ncols must be equal to len(seq1)+1 and
len(seq2)+1"

```



```

                                vdj_package.txt
# translate integer traces to coords
deltas = {

0 : (1,1),
    1 : (1,0),
    2 : (0,1),
    3 : (0,0)

}

# compute col where alignment should start
if nrows <= ncols:

    col = np.argmax( scoremat[nrows-1,:] )
    row = nrows-1
else:

    col = ncols-1
    row = np.argmax( scoremat[:,ncols-1] )

# if
row is coord in matrix, row-1 is coord in seq (b/c of init conditions)
aln1
= seq1[row-1]
  aln2 = seq2[col-1]

  aln1end = row
  aln2end =

col

    while (row-1 > 0) and (col-1 > 0):
        # compute direction of

moves
        rowchange,colchange = deltas[ tracemat[row,col] ]

# emit appropriate symbols
    if rowchange == 1:
        row -= 1

    aln1 = seq1[row-1] + aln1
    elif rowchange == 0:
        aln1 = '-'

+ aln1
        else:
            raise Exception, "Trace matrix contained jump of
greater than one row/col."

            if colchange == 1:
                col

```


vdj_package.txt

```

-= 1
    aln2 = seq2[col-1] + aln2
    elif colchange == 0:

    aln2 = '-' + aln2
    else:
        raise Exception, "Trace matrix
contained jump of greater than one row/col."
        aln1start = row-1

aln2start = col-1
    # the coords refer to coords in the sequence (pythonic)

    return aln1, (aln1start,aln1end), aln2, (aln2start,aln2end)

@staticmethod
def scoreVJalign(scorematrix):
    """Computes score of V alignment
given Needleman-Wunsch score matrix
    ASSUMES num rows < num cols, i.e.,
refseq V seg is on vertical axis
    """
    nrows,ncols =
scorematrix.shape
    if nrows <= ncols:
        return np.max(
scorematrix[nrows-1,:] )
    else:
        return np.max( scorematrix[:,ncols-1] )

@staticmethod
def scoreDalign(scorematrix):
    """Computes score of D
alignment given Smith-Waterman score matrix
    """
    return np.max(
scorematrix )

@staticmethod

```



```

                                vdj_package.txt
def seqdict2revcompseqdict(seqdict):

revcompdict = {}
    for item in seqdict.iteritems():
        revcompdict[item[0]] =
sequtils.reverse_complement(item[1])
    return revcompdict

class
vdj_aligner_combined(object):
    """vdj aligner for 'light' chains
        this class will
perform alignment for both loci, e.g., IGK and IGL
    and pick the one with the better V
score
    """
    def __init__(self,**kw):
        self.loci = kw['loci']

self.aligners = [vdj_aligner(locus=locus) for locus in self.loci]

self.patternPos = '111111111111'
    self.posset = set()
    self.negset = set()

    for aligner in self.aligners:
        self.posset.update(aligner.posset)

self.negset.update(aligner.negset)
    def align_chain(self,chain,verbose=False):

        alignments = []
        for aligner in self.aligners:
            curr_chain =
copy.deepcopy(chain)
            curr_score = aligner.align_chain(curr_chain)

alignments.append((curr_chain,curr_score))
        alignments = sorted(filter(lambda a:
hasattr(a[0], 'v'),alignments),key=lambda a:a[1]['v'],reverse=True)
            if
len(alignments) > 0:
                bestchain = alignments[0][0]
                if

```



```

                                vdj_package.txt
hasattr(bestchain, 'v'):
    chain.v = bestchain.v

chain.v_end_idx = bestchain.v_end_idx
    if hasattr(bestchain, 'j'):

chain.j = bestchain.j
    chain.j_start_idx = bestchain.j_start_idx

if hasattr(bestchain, 'junction'):
    chain.junction = bestchain.junction

    if hasattr(bestchain, 'd'):
        chain.d = bestchain.d
    return

alignments[0][1]    # NOTE: I only return the scores upon successful aln
def
align_seq(self, seq, verbose=False):
    chain =
    vdj.ImmuneChain(descr='sequence', seq=seq)
        self.align_chain(chain, verbose)

return chain

    def coding_chain(self, chain):
        strand =

self.seq2coding(chain.seq)
    if strand == -1:
        chain.seq =

seqtools.reverse_complement(chain.seq)
    chain.add_tag('revcomp')

chain.add_tag('coding')

    def seq2coding(self, seq):
        seqkeys =

vdj_aligner.seq2kmers(seq, [self.patternPos])
    seqwords = seqkeys[self.patternPos]

    strandid = 1
        if len(self.negset & seqwords) > len(self.posset & seqwords):

        strandid = -1
        return strandid

def igh_aligner():
    return

```



```

                                vdj_package.txt
vdj_aligner(locus='IGH')
def igk_aligner():
    return vdj_aligner(locus='IGK')
def
igl_aligner():
    return vdj_aligner(locus='IGL')
def igkl_aligner():
    return
vdj_aligner_combined(loci=['IGK','IGL'])
def trb_aligner():
    return
vdj_aligner(locus='TRB')
def tra_aligner():
    return vdj_aligner(locus='TRA')
def
trd_aligner():
    return vdj_aligner(locus='TRD')
def trg_aligner():
    return
vdj_aligner(locus='TRG')
import numpy as np
from numpy import ma
import scipy as sp
import
scipy.stats
import scipy.special
import matplotlib as mpl
import matplotlib.pyplot as plt
import vdj
def barcode_clone_counts(inhandle):
    """Return count dict from vdjxml file
with counts[barcode][clone]"""
    counts = dict()
    for chain in
vdj.parse_VDJXML(inhandle):
    try:      # chain may not have barcode
counts_barcode = counts.setdefault(chain.barcode,dict())
    except AttributeError:
        continue
    counts_barcode[chain.clone] = counts_barcode.get(chain.clone,0) + 1

```



```

                                vdj_package.txt
return counts

def barcode_junction_counts(inhandle):
    """Return count dict from
    vidxml file with counts[barcode][junction]"""
    counts = dict()
    for chain in
    vdj.parse_VDXML(inhandle):
        try:      # chain may not have barcode

counts_barcode = counts.setdefault(chain.barcode,dict())
    except AttributeError:

        continue
        counts_barcode[chain.junction] =
counts_barcode.get(chain.junction,0) + 1
    return counts

def
barcode_clone_counts2matrix(counts,barcodes=None,clones=None):
    """Generates matrix from
    count dict"""
    if barcodes == None:
        barcodes = counts.keys()
    if clones ==
None:
        clones = list( reduce( lambda x,y: x|y, [set(c.keys()) for c in
counts.itervalues()] ) )
        matrix = np.zeros((len(clones),len(barcodes)))
        for
(col,barcode) in enumerate(barcodes):
            for (row,clone) in enumerate(clones):

                matrix[row,col] = counts.get(barcode,dict()).get(clone,0)
            return

(clones,barcodes,matrix)
barcode_junction_counts2matrix = barcode_clone_counts2matrix

#
=====
# = OLD OLD OLD OLD OLD
OLD OLD OLD OLD OLD OLD OLD OLD OLD OLD =
#
=====

```


vdj_package.txt

```

# =====
# =

Time series =
# =====

def clone_timeseries(inhandle, barcodes,
reference_clones=None):
    # generate count data
    for chain in
vdj.parse_VDJXML(inhandle):
    counts

def
clone_timeseries(inhandle,time_tags,reference_clones=None):
    # get count data

time_tags_set = set(time_tags)
clone_counts = {}
for tag in time_tags:

clone_counts[tag]={}
    for chain in vdj.parse_VDJXML(inhandle):
        try:

curr_time_tag = (chain.tags&time_tags_set).pop()
        except KeyError:

continue

        try:

clone_counts[curr_time_tag][vdj.get_clone(chain)] += 1
        except KeyError:

clone_counts[curr_time_tag][vdj.get_clone(chain)] = 1
        # set up reference clones

if reference_clones == None:
    reference_clones = set()
    for counts in

clone_counts.itervalues():
    reference_clones.update(counts.keys())

reference_clones = list(reference_clones)

    # build timeseries matrix
    num_clones

```



```

                                vdj_package.txt

= len(reference_clones)
    num_times = len(time_tags)
    countdata =

np.zeros((num_clones,num_times))
    for (i,tag) in enumerate(time_tags):

countdata[:,i] = vdj.count_dict_clone_counts(clone_counts[tag],reference_clones)

return countdata,reference_clones

def
timeseries2proportions(countdata,freq=True,log=True,pseudocount=1e-1):
    num_time_series,
num_times = countdata.shape
    num_transitions = num_times - 1
    if pseudocount != 0:

        proportions = np.zeros((num_time_series,num_transitions))
        countdata_pseudo =
countdata + np.float_(pseudocount)
        if freq == True:
            countdata_pseudo =
countdata_pseudo / countdata_pseudo.sum(axis=0)
            for i in range(num_transitions):

                proportions[:,i] = countdata_pseudo[:,i+1] / countdata_pseudo[:,i]
            else: #
only look at time series that are non-zero the whole way through
            idxs =
np.sum(countdata>0,axis=1)==num_times
            proportions =
np.zeros((np.sum(idxs),num_transitions))
            if freq == True:

countdata_modified = np.float_(countdata) / countdata.sum(axis=0)
            else:

countdata_modified = np.float_(countdata)
            for i in range(num_transitions):

                proportions[:,i] = countdata_modified[idxs,i+1] / countdata_modified[idxs,i]
            if
log==True:
            return np.log10(proportions)

```



```

                                vdj_package.txt
else:
    return proportions

def
timeseries2autocorrelation(timeseries):
    ac = [1.] +
    [sp.stats.pearsonr(timeseries[:-i],timeseries[i:])[0] for i in
    range(1,len(timeseries)-2)]

    return ac

# =====
# = Spectratypes =
# =====

def
cdr3s2spectratype(cdr3s):
    min_raw_cdr3 = np.min(cdr3s)
    max_raw_cdr3 = np.max(cdr3s)

    min_cdr3 = np.int(np.ceil( min_raw_cdr3 / 3.) * 3) # will be a nonzero mult of 3

max_cdr3 = np.int(np.floor(max_raw_cdr3 / 3.) * 3) # will be a mult of 3

    # bin the
    CDR3s lengths. The first elt is rep zero len (and should be zero)
    # and the last bin
    always represents one greater than the biggest mult of 3
    binnedcdr3s =
    np.histogram(cdr3s,bins=np.arange(0,max_cdr3+2))[0] # the +2 is due to the peculi.
    of
    np.hist.

    gaussians = []
    for cdr3len in np.arange(min_cdr3,max_raw_cdr3,3):

        totalcdr3s = np.sum(binnedcdr3s[cdr3len-1:cdr3len+2])
        goodcdr3s =
        binnedcdr3s[cdr3len]
        if totalcdr3s == 0:
            continue
        mu = cdr3len

        x = cdr3len-0.5
        tail = (1 - (np.float(goodcdr3s)/totalcdr3s)) / 2.

    sigma = (x-mu) / (np.sqrt(2.)*sp.special.erfinv(2*tail-1))
    rv =

```



```

                                vdj_package.txt

sp.stats.norm(loc=mu,scale=sigma)
    gaussians.append( (totalcdr3s,rv) )

    t =
np.linspace(0,max_cdr3+1,1000)
    y = np.zeros(len(t))
    for (s,rv) in gaussians:

y += s*rv.pdf(t)
    return (t,y)

def spectratype_curves(inhandle):
    # NOTE: requires
chains with V and J alns

    # init data structure
    cdr3s = {}
    for v_seg in
vdj.refseq.IGHV_seqs.keys():
        for j_seg in vdj.refseq.IGHJ_seqs.keys():

cdr3s[vdj.vj_id(v_seg,j_seg)] = []

    # load data
    for chain in
vdj.parse_VDJXML(inhandle):
        if chain.v == '' or chain.j == '' or chain.junction ==
        '':
            continue
        cdr3s[vdj.vj_id(chain.v,chain.j)].append(chain.cdr3)

spectras = {}
    for v_seg in vdj.refseq.IGHV_seqs.keys():
        for j_seg in
vdj.refseq.IGHJ_seqs.keys():
            if len(cdr3s[vdj.vj_id(v_seg,j_seg)]) == 0:

                # empty VJ combo:
                spectras[vdj.vj_id(v_seg,j_seg)] =
(np.array([0,150]),np.array([0,0]))
            else:

spectras[vdj.vj_id(v_seg,j_seg)] = cdr3s2spectratype(cdr3s[vdj.vj_id(v_seg,j_seg)])

return spectras

```


vdj_package.txt

```

# =====
# = Diversity estimation =
#
=====

def estimator_chao1(counts):
    """Bias corrected. See
    Estimates doc (Colwell)"""
    Sobs = len(counts)
    F1 =
    np.float_(np.sum(np.int_(counts)==1))
    F2 = np.float_(np.sum(np.int_(counts)==2))

    chao1 = Sobs + F1*(F1-1)/(2*(F2+1))
    return chao1

def
estimator_chao1_variance(counts):
    F1 = np.float_(np.sum(np.int_(counts)==1))
    F2 =
    np.float_(np.sum(np.int_(counts)==2))
    if F1 > 0 and F2 > 0:
        chao1_var =
        (F1*(F1-1)/(2*(F2+1))) + (F1*(2*F1-1)*(2*F1-1)/(4*(F2+1)*(F2+1))) +
        (F1*F1*F2*(F1-1)*(F1-1)/(4*(F2+1)*(F2+1)*(F2+1)*(F2+1)))
    elif F1 > 0 and F2 == 0:

        Schao1 = estimator_chao1(counts)
        chao1_var = (F1*(F1-1)/2) +
        (F1*(2*F1-1)*(2*F1-1)/4) - (F1*F1*F1*F1/(4*Schao1))
    elif F1 == 0:
        N =
    np.float_(np.sum(counts))
    Sobs = np.float_(len(counts))
    chao1_var =
    Sobs*np.exp(-1*N*Sobs) * (1-np.exp(-1*N*Sobs))
    return chao1_var

def
estimator_ace(counts, rare_cutoff=10):
    Sobs = np.float_(len(counts))
    Srare =
    np.float_(np.sum(np.int_(counts)<=rare_cutoff))
    Sabund = Sobs - Srare

```



```

                                vdj_package.txt
F1 =
np.float_(np.sum(np.int_(counts)==1))
    F = lambda i:
np.float_(np.sum(np.int_(counts)==i))
    Nrare = np.float_(np.sum([i*F(i) for i in
range(1,rare_cutoff+1)]))
    #Nrare =
np.float_(np.sum(counts[np.int_(counts)<=rare_cutoff]))
    if Nrare == F1: # in accordance
with Estimates
    return estimator_chao1(counts)
    Cace = 1 - (F1/Nrare)

gamma_squared = Srare*np.sum([i*(i-1)*F(i) for i in
range(1,rare_cutoff+1)])/(Cace*Nrare*(Nrare-1))
    if gamma_squared < 0:

gamma_squared = 0
    Sace = Sabund + (Srare/Cace) + (F1/Cace)*gamma_squared
    return

Sace

def accumulation_curve(sample,sampling_levels):
    pass

#
=====
# = Statistical utilities =
# =====

def
counts2sample(counts):
    """Computes a consistent sample from a vector of counts.

Takes a vector of counts and returns a vector of indices x
    such that len(x) = sum(c)

and each elt of x is the index of
    a corresponding elt in c
    """
    x =

np.ones(np.sum(counts),dtype=np.int_)

    start_idx = 0
    end_idx = 0
    for i in

```



```

                                vdj_package.txt
xrange(len(counts)):
    start_idx = end_idx
    end_idx = end_idx + counts[i]

    x[start_idx:end_idx] = x[start_idx:end_idx] * i
    return x

def
sample2counts(sample):
    """Return count vector from list of samples.

    The
    ordering etc is ignored; only the uniqueness
    of the objects is considered.
    """

    num_categories = len(set(sample))
    count_dict = {}
    for elt in sample:
        try:
            count_dict[elt] += 1
        except KeyError: count_dict[elt] = 1
    return
count_dict.values()

# def sample2counts(sample, categories=0):
#     """Return count
vector from list of samples.
#
#     Take vector of samples and return a vector of
counts. The elts
#     refer to indices in something that would ultimately map to the
#
#     originating category (like from a multinomial). Therefore, if there
#     are,
say, 8 categories, then valid values in sample should be 0-7.
#     If categories is not
given, then i compute it from the highest value
#     present in sample (+1).
#     """
#
#     counts = np.bincount(sample)
#     if (categories > 0) and (categories > len(counts)):
#
#         counts = np.append( counts, np.zeros(categories-len(counts)) )
#     return
counts

```


vdj_package.txt

```

def scoreatpercentile(values,rank):
    return
sp.stats.scoreatpercentile(values,rank)

def percentileofscore(values,score):

values.sort()
    return values.searchsorted(score) / np.float_(len(values))

def
bootstrap(x, nboot, theta, *args):
    '''return n bootstrap replications of theta from
x'''
    N = len(x)
    th_star = np.zeros(nboot)
    for i in xrange(nboot):

th_star[i] = theta( x[ np.random.randint(0,N,N) ], *args )    # bootstrap repl from
x

    return th_star

def subsample(x, num_samples, sample_size, theta, *args):
    """return
num_samples evaluations of the statistic theta
on subsamples of size sample_size"""

N = len(x)
    th_star = np.zeros(num_samples)
    for i in xrange(num_samples):

        th_star[i] = theta( x[ np.random.randint(0,N,sample_size) ], *args )    # subsample
from
from x
        return th_star

def randint_without_replacement(low,high=None,size=None):

    if high == None:
        high = low
        low = 0
    if size == None:

```



```

                                vdj_package.txt
    size = 1

    urn = range(low,high)
    N = len(urn)
    flip = False
    if size > N/2:
        flip =

True
        size = N - size
    sample = []
    for i in xrange(size):
        draw =

np.random.randint(0,N-i)
        sample.append(urn.pop(draw))
    if not flip:

return np.asarray(sample)
    else:
        return np.asarray(urn)

def
subsample_without_replacement(x, num_samples, sample_size, theta, *args):
    """return

num_samples evaluations of the statistic theta
    on subsamples of size sample_size"""

N = len(x)
    th_star = np.zeros(num_samples)

    for i in xrange(num_samples):

        th_star[i] = theta( x[ randint_without_replacement(0,N,sample_size) ], *args )    #

subsample from from x
    return th_star

# =====
# = Visualization =
#
=====

class ConstWidthRectangle(mpl.patches.Patch):
    def __init__(self,
x, y1, y2, w, **kwargs):
        self.x = x
        self.y1 = y1
        self.y2 = y2

```


vdj_package.txt

```

self.w = w
mpl.patches.Patch.__init__(self,**kwargs)

def
get_path(self):
    return mpl.path.Path.unit_rectangle()

def
get_transform(self):
    box = np.array([[self.x,self.y1],
[self.x,self.y2]])
    box = self.axes.transData.transform(box)

    w =
self.w * self.axes.bbox.width / 2.0

    box[0,0] -= w
    box[1,0] += w

    return mpl.transforms.BboxTransformTo(mpl.transforms.Bbox(box))

class
ConstwidthLine(mpl.lines.Line2D):
    def __init__(self,x,y,w,**kwargs):
        self.x
= x
        self.y = y
        self.w = w

mpl.lines.Line2D.__init__(self,[0,1],[0,0],**kwargs) # init to unit line
def
get_transform(self):
    # define transform that takes unit horiz line seg
    # and
places it in correct position using display
    # coords

    box =
np.array([[self.x,self.y],
          [self.x,self.y+1]])
    box =
self.axes.transData.transform(box)

    w = self.w * self.axes.bbox.width / 2.0

```


vdj_package.txt

```

        box[0,0] -= w
        box[1,0] += w

        #xdisp,ydisp =
self.axes.transData.transform_point([self.x,self.y])
        #xdisp -= w
        #xleft =
xdisp - w
        #xright = xdisp + w

        return

mpl.transforms.BboxTransformTo(mpl.transforms.Bbox(box))
        #return

mpl.transforms.Affine2D().scale(w,1).translate(xdisp,ydisp)
    def
draw(self,renderer):
    # the ONLY purpose of redefining this function is to force the
Line2D
    # object to execute recache(). Otherwise, certain changes in the scale

    # do not invalidate the Line2D object, and the transform will not be
    # recomputed
    (and so the Axes coords computed earlier will be obsolete)
    self.recache()

return mpl.lines.Line2D.draw(self,renderer)

class
ConstHeightRectangle(mpl.patches.Patch):
    def __init__(self, x1, x2, y, h,
**kwargs):
        self.x1 = x1
        self.x2 = x2
        self.y = y
        self.h =
h
        mpl.patches.Patch.__init__(self,**kwargs)
    def get_path(self):

return mpl.path.Path.unit_rectangle()
    def get_transform(self):
        box =

```



```

                                vdj_package.txt
np.array([[self.x1,self.y],
          [self.x2,self.y]])
    box =
self.axes.transData.transform(box)
    h = self.h * self.axes.bbox.height /
2.0

    box[0,1] -= h
    box[1,1] += h

    return

mpl.transforms.BboxTransformTo(mpl.transforms.Bbox(box))
class
ConstHeightLine(mpl.lines.Line2D):
    def __init__(self,x,y,h,**kwargs):

self.x = x
    self.y = y
    self.h = h

mpl.lines.Line2D.__init__(self,[0,0],[0,1],**kwargs) # init to unit line
    #
self.x = x
    # self.y = y
    # self.w = w
    #
mpl.lines.Line2D.__init__(self,[0,1],[0,0],**kwargs) # init to unit line
    def
get_transform(self):
    # define transform that takes unit horiz line seg
    # and
places it in correct position using display
    # coords

    box =
np.array([[self.x,self.y],
          [self.x+1,self.y]])
    box =
self.axes.transData.transform(box)
    h = self.h * self.axes.bbox.height /
2.0

    box[0,1] -= h
    box[1,1] += h

```


vdj_package.txt

```

        #xdisp,ydisp =
self.axes.transData.transform_point([self.x,self.y])
        #xdisp -= w
        #xleft =

xdisp - w
        #xright = xdisp + w

        return

mpl.transforms.BboxTransformTo(mpl.transforms.Bbox(box))
        #return

mpl.transforms.Affine2D().scale(w,1).translate(xdisp,ydisp)
    def
draw(self,renderer):
        # the ONLY purpose of redefining this function is to force the
Line2D
        # object to execute recache(). Otherwise, certain changes in the scale

        # do not invalidate the Line2D object, and the transform will not be
        # recomputed

        (and so the Axes coords computed earlier will be obsolete)
        self.recache()

return mpl.lines.Line2D.draw(self,renderer)

def boxplot(ax, x, positions=None,
widths=None, vert=1):
    # adapted from matplotlib

    # convert x to a list of
vectors
        if hasattr(x, 'shape'):
            if len(x.shape) == 1:
                if
hasattr(x[0], 'shape'):
                    x = list(x)
                else:
                    x =

[x,]
            elif len(x.shape) == 2:
                nr, nc = x.shape
                if nr == 1:

                    x = [x]
                elif nc == 1:
                    x = [x.ravel()]

```


vdj_package.txt

```

else:
    x = [x[:,i] for i in xrange(nc)]
    else:
        raise
ValueError, "input x can have no more than 2 dimensions"
    if not hasattr(x[0],
'__len__'):
        x = [x]
        col = len(x)

        # get some plot info
        if positions
is None:
            positions = range(1, col + 1)
            if widths is None:
                widths =
min(0.3/len(positions),0.05)
            if isinstance(widths, float) or isinstance(widths, int):

                widths = np.ones((col,), float) * widths
            # loop through columns, adding each
to plot
            for i,pos in enumerate(positions):
                d = np.ravel(x[i])
                row =
len(d)
                if row==0:
                    # no data, skip this position
                    continue

                # get distrib info
                q1, med, q3 = mpl.mlab.prctile(d,[25,50,75])
                dmax =
np.max(d)
                dmin = np.min(d)

                line_color = '#074687'

face_color = '#96B7EC'
                if vert == 1:
                    medline =
ConstwidthLine(pos,med,widths[i],color=line_color,zorder=3)
                    box =
ConstwidthRectangle(pos,q1,q3,widths[i],facecolor=face_color,edgecolor=line_color,zorder=2)

                    vertline =
mpl.lines.Line2D([pos,pos],[dmin,dmax],color=line_color,zorder=1)

```


vdj_package.txt

```

else:
    medline = ConstHeightLine(med,pos,widths[i],color=line_color,zorder=3)

    box =
ConstHeightRectangle(q1,q3,pos,widths[i],facecolor=face_color,edgecolor=line_color,z
order=2)
    vertline =
mpl.lines.Line2D([dmin,dmax],[pos,pos],color=line_color,zorder=1)

    ax.add_line(vertline)
    ax.add_patch(box)

ax.add_line(medline)

#=====
#=====
#=====

def
rep2spectratype(rep):
    """Compute spectratype curves from Repertoire object."""

    cdr3s = np.array([c.cdr3 for c in rep if c.junction != ''])
    min_raw_cdr3 =
np.min(cdr3s)
    max_raw_cdr3 = np.max(cdr3s)
    min_cdr3 = np.int(np.ceil( min_raw_cdr3 /
3.) * 3) # will be a nonzero mult of 3
    max_cdr3 = np.int(np.floor(max_raw_cdr3 / 3.) *
3) # will be a mult of 3
    # bin the CDR3s lengths. The first elt is rep zero len
    (and should be zero)
    # and the last bin always represents one greater than the biggest
    mult of 3
    binnedcdr3s = np.histogram(cdr3s,bins=np.arange(0,max_cdr3+2))[0] # the +2
    is due to the peculi. of np.hist.

    gaussians = []
    for cdr3len in
np.arange(min_cdr3,max_raw_cdr3,3):

```



```

                                vdj_package.txt
    totalcdr3s =
np.sum(binnedcdr3s[cdr3len-1:cdr3len+2])
    goodcdr3s = binnedcdr3s[cdr3len]

if totalcdr3s == 0:
    continue
    mu = cdr3len
    x = cdr3len-0.5

    tail = (1 - (np.float(goodcdr3s)/totalcdr3s)) / 2.
    sigma = (x-mu) /
(np.sqrt(2.)*sp.special.erfinv(2*tail-1))
    rv = sp.stats.norm(loc=mu,scale=sigma)

    gaussians.append( (totalcdr3s,rv) )
    t = np.linspace(0,max_cdr3+1,1000)
    y =
np.zeros(len(t))
    for (s,rv) in gaussians:
        y += s*rv.pdf(t)
    return
(t,y)

def
scatter_repertoires_ontology(reps,info='VJCDR3',gooddata=False,measurement='proportions'):

    """Create a grid of scatter plots showing correlations between all pairs of
    repertoires.

    reps -- list of Repertoire objects
    """
    numreps = len(reps)

    datalist = []
    for rep in reps:
        datalist.append(
vdj.counts_ontology_1D(rep,info,gooddata) )
        if measurement == 'proportions':

for i in xrange(len(datalist)):
    datalist[i] = np.float_(datalist[i]) /

```


vdj_package.txt

```

np.sum(datalist[i])
    min_nonzero = np.min([np.min(data[data>0]) for data in
datalist])
    max_nonzero = np.max([np.max(data[data>0]) for data in datalist])
    axislo
= 10**np.floor( np.frexp(min_nonzero)[1] * np.log10(2) )
    axishi = 10**np.ceil(
np.frexp(max_nonzero)[1] * np.log10(2) )
    fig = plt.figure()
    hist_axs = []

    for row in xrange(numreps):
        col = row
        plotnum = numreps*row + col + 1

        ax = fig.add_subplot(numreps,numreps,plotnum)

        ax.hist(datalist[row],bins=100,log=True,facecolor='k')
        hist_axs.append(ax)

    scatter_axs = []
    for row in xrange(numreps-1):
        for col in
xrange(row+1,numreps):
            plotnum = numreps*row + col + 1
            ax =
fig.add_subplot(numreps,numreps,plotnum)

ax.scatter(datalist[row],datalist[col],c='k',marker='o',s=2,edgecolors=None)

ax.set_xscale('log')
    ax.set_yscale('log')

ax.axis([axislo,axishi,axislo,axishi])
    scatter_axs.append(ax)

    return
fig
def scatter_repertoires_clusters(reps,refclusters,measurement='proportions'):

"""Create a grid of scatter plots showing correlations between all pairs of
repertoires.

```



```

                                vdj_package.txt
reps -- list of Repertoire objects
"""
numreps = len(reps)

datalist = []
for rep in reps:
    clusters = vdj.getClusters(rep)

datalist.append( vdj.countsClusters(clusters,refclusters) )

    if measurement ==
'proportions':
    for i in xrange(len(datalist)):
        datalist[i] =
np.float_(datalist[i]) / np.sum(datalist[i])

        min_nonzero =
np.min([np.min(data[data>0]) for data in datalist])
        max_nonzero =
np.max([np.max(data[data>0]) for data in datalist])
        axislo = 10**np.floor(
np.frexp(min_nonzero)[1] * np.log10(2) )
        axishi = 10**np.ceil( np.frexp(max_nonzero)[1]
* np.log10(2) )

        fig = plt.figure()

        hist_axs = []
        for row in
xrange(numreps):
            col = row
            plotnum = numreps*row + col + 1
            ax =

fig.add_subplot(numreps,numreps,plotnum)

ax.hist(datalist[row],bins=100,log=True,facecolor='k')
        hist_axs.append(ax)

scatter_axs = []
        for row in xrange(numreps-1):
            for col in
xrange(row+1,numreps):
                plotnum = numreps*row + col + 1
                ax =

fig.add_subplot(numreps,numreps,plotnum)

```



```

                                vdj_package.txt
ax.scatter(datalist[row],datalist[col],c='k',marker='o',s=0.5,edgecolors=None)

ax.set_xscale('log')
    ax.set_yscale('log')

ax.axis([axislo,axishi,axislo,axishi])
    scatter_axs.append(ax)

    return

fig
def reps2timeseries(reps,refclusters):
    """Return time series matrix from list or
    Repertoire objs in chron order.
    reps is list of Repertoire objects
    refclusters
    is the master list of reference clusters
    """
    numreps = len(reps)
    numclusters =
len(refclusters)

    countdata = np.zeros((numclusters,numreps))
    for (i,rep) in
enumerate(reps):
        clusters = vdj.getClusters(rep)
        countdata[:,i] =
vdj.countsClusters(clusters,refclusters)

    return countdata

def
timeseries_repertoires(times,reps,refclusters,idxsbool=None,allpositive=False):

    """Create a time-series of the different clusters in refclusters.
    If allpositive is
    True, then it will limit itself to drawing timeseries
    only for those clusters that are
    non-zero at all timepoints.
    """
    ax = plt.gca()

    numreps = len(reps)

numclusters = len(refclusters)

```



```

                                vdj_package.txt
countdata = np.zeros((numclusters,numreps))
for
(i,rep) in enumerate(reps):
    clusters = vdj.getClusters(rep)
    countdata[:,i] =
vdj.countsClusters(clusters,refclusters)

    sums = countdata.sum(0)
    proportions =
np.float_(countdata) / sums
    if idxsbool == None:
        if allpositive == True:

            idxsbool = np.sum(proportions,axis=1) > 0
        else:
            idxsbool =

np.array([True]*proportions.shape[0])

ax.plot(times,countdata[idxsbool,:].transpose(),'k-',linewidth=0.2)

plt.draw_if_interactive()
return ax

def rep2spectratype(rep):
    """Compute
spectratype curves from Repertoire object."""
    cdr3s = np.array([c.cdr3 for c in rep
if c.junction != ''])
    min_raw_cdr3 = np.min(cdr3s)
    max_raw_cdr3 = np.max(cdr3s)

min_cdr3 = np.int(np.ceil( min_raw_cdr3 / 3.) * 3) # will be a nonzero mult of 3

max_cdr3 = np.int(np.floor(max_raw_cdr3 / 3.) * 3) # will be a mult of 3
    # bin the
CDR3s lengths. The first elt is rep zero len (and should be zero)
    # and the last bin
always represents one greater than the biggest mult of 3
    binnedcdr3s =

np.histogram(cdr3s,bins=np.arange(0,max_cdr3+2))[0] # the +2 is due to the peculi.
of
np.hist.

```


vdj_package.txt

```

gaussians = []
for cdr3len in np.arange(min_cdr3,max_raw_cdr3,3):

    totalcdr3s = np.sum(binnedcdr3s[cdr3len-1:cdr3len+2])
    goodcdr3s =

binnedcdr3s[cdr3len]
    if totalcdr3s == 0:
        continue
    mu = cdr3len

    x = cdr3len-0.5
    tail = (1 - (np.float(goodcdr3s)/totalcdr3s)) / 2.

sigma = (x-mu) / (np.sqrt(2.)*sp.special.erfinv(2*tail-1))
rv =

sp.stats.norm(loc=mu,scale=sigma)
    gaussians.append( (totalcdr3s,rv) )

t =

np.linspace(0,max_cdr3+1,1000)
y = np.zeros(len(t))
for (s,rv) in gaussians:

y += s*rv.pdf(t)
    return (t,y)

def
circlemapVJ(ax,counts,rowlabels=None,collabels=None,scale='linear'):
    numV =

counts.shape[0]
    numJ = counts.shape[1]
    X,Y = np.meshgrid(range(numJ),range(numV))

    # mask zero positions
    X,Y = ma.array(X), ma.array(Y)
    C = ma.array(counts)

    zeromask = (counts == 0)
    X.mask = zeromask
    Y.mask = zeromask
    C.mask =

zeromask

    # ravel nonzero elts (deletes zero-positions)
    x = ma.compressed(X)

y = ma.compressed(Y)
    c = ma.compressed(C)

```



```

                                vdj_package.txt
# log normalize counts if requested

if scale == 'log':
    c = ma.log10(c)

    if scale == 'linear' or scale ==
'log':
        # normalize counts to desired size-range
        max_counts = ma.max(c)

        min_counts = ma.min(c)
        counts_range = max_counts - min_counts

max_size = 100
    min_size = 5
    size_range = max_size - min_size

sizes = (np.float(size_range) / counts_range) * (c - min_counts) + min_size
    if
scale == 'custom':
        trans_counts = 1000
        linear_positions = c >= trans_counts

        log_positions = c < trans_counts

        min_size = 3
        trans_size =
40 # 30
        max_size = 200 # 150
        log_size_range = trans_size - min_size

linear_size_range = max_size - trans_size

        linear_max_counts =
ma.max(c[linear_positions])
        linear_min_counts = ma.min(c[linear_positions])

linear_counts_range = linear_max_counts - linear_min_counts
    log_max_counts =
ma.max(c[log_positions])
    log_min_counts = ma.min(c[log_positions])

log_counts_range = np.log10(log_max_counts) - np.log10(log_min_counts)

sizes = np.zeros(len(c))

```



```

        vdj_package.txt
        sizes[linear_positions] = (np.float(linear_size_range) /
linear_counts_range) * (c[linear_positions] - linear_min_counts) + trans_size

sizes[log_positions] = (np.float(log_size_range) / log_counts_range) *
(ma.log10(c[log_positions]) - ma.log10(log_min_counts)) + min_size
    collection =
mpl.collections.CircleCollection(
        sizes,

        offsets = zip(x,y),

transOffset = ax.transData, # i may need to explicitly set the xlim and ylim info
for this
to work correctly
        facecolors = '#1873C1',

        linewidths = 0.25,

clip_on = False)
    ax.add_collection(collection)
    ax.set_aspect('equal')

ax.autoscale_view()

ax.xaxis.set_major_locator(mpl.ticker.FixedLocator(range(counts.shape[1])))
ax.yaxis.set_major_locator(mpl.ticker.FixedLocator(range(counts.shape[0])))
    if
rowlabels != None:

ax.xaxis.set_major_formatter(mpl.ticker.FixedFormatter(collabels))
    if collabels !=
None:
        ax.yaxis.set_major_formatter(mpl.ticker.FixedFormatter(rowlabels))

for ticklabel in ax.xaxis.get_ticklabels():

ticklabel.set_horizontalalignment('left')
    ticklabel.set_rotation(-45)

```


vdj_package.txt

```

ticklabel.set_size(8)
    for ticklabel in ax.yaxis.get_ticklabels():

ticklabel.set_size(8)
    if scale == 'linear' or scale == 'log':
        return
(min_counts,max_counts),(min_size,max_size)
    else:
        return
(linear_min_counts,trans_counts,log_max_counts),(min_size,trans_size,max_size)
# define
colormap for -1 to 1 (green-black-red) like gene expression
redgreendict = {'red': [(0.0,
0.0,    0.0),
                        (0.5,    0.0,    0.0),
                        (1.0,
1.0,    0.0)],
                'green': [(0.0,    0.0,    1.0),
                          (0.5,    0.0,    0.0),
                          (1.0,    0.0,    0.0)],
                'blue': [(0.0,    0.0,    0.0),
                        (0.5,    0.0,    0.0),
                        (1.0,    0.0,    0.0)]}

redgreen =
mpl.colors.LinearSegmentedColormap('redgreen',redgreendict,256)
redgreen.set_bad(color='w')
import numpy as np
import scipy as sp
import scipy.cluster

import vdj
import clusteringcore

def pdist(X,metric):
    m = len(X)
    dm = np.zeros((m
* (m - 1) / 2,), dtype=np.double)
    k = 0
    for i in xrange(0, m - 1):
        for j in

```



```

                                vdj_package.txt
xrange(i+1, m):
    dm[k] = metric(X[i], X[j])
    k += 1
    return dm

def cluster_seqs(seqs,cutoff=4.5,linkage='single'):
    # check trivial cases
    if
len(seqs) == 0:
    return (np.array([]),{})
    # raise Exception, "chains has
nothing it"

    # collapse identical seqs into each other
    unique_seqs =

list(set(seqs))
    seq_idx = dict( [(j,i) for (i,j) in enumerate(unique_seqs)] )

# check trivial case
    if len(unique_seqs) == 1:
        T = np.array([1]*len(seqs))

    return (T,seq_idx)

    # compute the distance matrix
    Y = pdist( unique_seqs,
clusteringcore.levenshtein )

    # compute the linkage
    Z =

sp.cluster.hierarchy.linkage(Y,method=linkage)

    # determine the clusters at level
cutoff
    T = sp.cluster.hierarchy.fcluster(Z,cutoff,criterion='distance')

    return

(T,seq_idx)
import types

import vdj

# def parse_VDJXML(inputfile):
#     """Generator to

return ImmuneChain objects from a vdxml file.
#
#     Utilizes python XML libraries.
#
#
#
#     """
#     # NOTE: this can probably be made more elegant if implemented

```



```

                                vdj_package.txt
#      # as a
class
#
#      # global state variables.
#      chain = None
#      data_buffer = None
#
#      def start_handler(name,attributes):
#          if name == 'ImmuneChain':
#
#              chain = ImmuneChain()
#              else:
#                  data_buffer = ''
#
#      def
end_handler(name):
#          if name == 'ImmuneChain':
#              yield chain
#
elif name == 'tag':
#          chain.add_tags(data_buffer)
#          else:
#
chain.__setattr__(name,data_buffer)
#
#      def data_handler(data):
#
data_buffer += data
#
#      xmlparser = xml.parsers.expat.ParserCreate()
#
xmlparser.StartElementHandler = start_handler
#      xmlparser.EndElementHandler =
end_handler
#      xmlparser.CharacterDataHandler = data_handler
#
#      if not
hasattr(inputfile,'read'):
#          inputfile = open(inputfile,'r')
#
#
xmlparser.ParseFile(inputfile)
#
=====
# = START DEPRECATED
=====
#

```


vdj_package.txt

```

=====
def
parse_VDJXML_old(inputfile):
    """Load a data from a VDJXML file as a Repertoire or list
of ImmuneChains

    NOTE: this fn does NOT utilize the XML libraries; it implements a
manual parser
    that takes input line by line.

    THIS ASSUMES THAT EVERY XML
ELEMENT TAKES ONE AND ONLY ONE LINE
    """
    if
instance(inputfile,types.StringTypes):
    ip = open(inputfile,'r')
    elif
instance(inputfile,file):
    ip = inputfile

    numChains = 0

possible_elements = [
    'descr',
    'seq',
    'v',

    'd',
    'j',
    'ighc',
    'cdr3',

    'junction',
    'func',
    'tag'
]

for line in ip:
    line = line.strip()
    endelementpos = line.find('>') + 1

    xmlelement = line[0:endelementpos]
    element = xmlelement[1:-1]

```



```

                                vdj_package.txt
if xmlelement == '<ImmuneChain>':
    chain = vdj.ImmuneChain()
    elif
xmlelement == '</ImmuneChain>':
    numChains += 1
    yield chain

elif element in possible_elements:
    if element == 'tag':
        tagdata

= line[endelementpos:-1*(endelementpos+1)].split('|')
    if tagdata[0] ==

'experiment':
        chain.__setattr__('experiment','|'.join(tagdata[1:]))

        elif tagdata[0] == 'clone':

chain.__setattr__('clone','|'.join(tagdata[1:]))
    elif tagdata[0] ==

'barcode':
        chain.__setattr__('barcode','|'.join(tagdata[1:]))

        elif tagdata[0] == 'v_end_idx':

chain.__setattr__('v_end_idx','|'.join(tagdata[1:]))
    elif tagdata[0] ==

'j_start_idx':
        chain.__setattr__('j_start_idx','|'.join(tagdata[1:]))

        else:

chain.add_tags(line[endelementpos:-1*(endelementpos+1)])
    elif element ==

'ighc':
        eltdata = line[endelementpos:-1*(endelementpos+1)]

if eltdata != '':
    chain.c = eltdata
    else:

pass
    elif element == 'func':
        pass
    else:

    eltdata = line[endelementpos:-1*(endelementpos+1)]
    if eltdata !=

```


vdj_package.txt

```

':
        chain.__setattr__(element, eltdata)
    else:

        pass

    if isinstance(inputfile, types.StringTypes):
        ip.close()

#
=====
# = END DEPRECATED
=====
#
=====

# def
filter_parse_VDXML(inputfile, predicate):
#     """Load a data from a VDXML file as a
#     Repertoire or list of ImmuneChains
#     #
#     predicate is a function that takes a chain
#     and return True or False. Things
#     that return false are skipped.
#     #
#     NOTE:
#
#     this fn does NOT utilize the XML libraries; it implements a manual parser
#     that takes
#
#     input line by line.
#     #
#     THIS ASSUMES THAT EVERY XML ELEMENT TAKES ONE AND ONLY ONE
#
#     LINE
#     #
#     """
#     if isinstance(inputfile, types.StringTypes):
#         ip
#
#     = open(inputfile, 'r')
#     elif isinstance(inputfile, file):
#         ip = inputfile
#
#     numChains = 0
#
#     possible_elements = [
#         'descr',
#
#         'seq',
#         'v',

```



```

                                vdj_package.txt
#           'd',
#           'j',
#
#       'ighc',
#           'cdr3',
#           'junction',
#
# 'func',
#           'tag'
#           ]
#       for line in ip:
#
line = line.strip()
#       endelementpos = line.find('>') + 1
#       xmlelement =
line[0:elementpos]
#       element = xmlelement[1:-1]
#
#       if
xmlelement == '<ImmuneChain>':
#           chain = ImmuneChain()
#           elif xmlelement
== '</ImmuneChain>':
#           numChains += 1
#           if predicate(chain) ==
True:
#               yield chain
#           else:
#               pass
#
elif element in possible_elements:
#       if element == 'cdr3':
#
chain.cdr3 = eval(line[elementpos:-1*(elementpos+1)])
#       elif element ==
'tag':
#           chain.add_tags(line[elementpos:-1*(elementpos+1)])
#
#       else:
#
chain.__setattr__(element, line[elementpos:-1*(elementpos+1)])
#
#       if
isinstance(inputfile, types.StringTypes):
#           ip.close()
"""params.py
Define

```


vdj_package.txt

directory and file names that must be manually modified
to point to certain resources.

HACK. Figure out better way to refer to this directory
vdj_dir =

'/Users/laserson/research/church/code/lib/vdj'

packaged data dir

data_dir = 'data'

IGHV_fasta = 'IGHV.fasta'

IGHD_fasta = 'IGHD.fasta'

IGHJ_fasta = 'IGHJ.fasta'

IGKV_fasta =

'IGKV.fasta'

IGKJ_fasta = 'IGKJ.fasta'

IGLV_fasta = 'IGLV.fasta'

IGLJ_fasta = 'IGLJ.fasta'

TRBV_fasta = 'TRBV.fasta'

TRBD_fasta = 'TRBD.fasta'

TRBJ_fasta = 'TRBJ.fasta'

TRAV_fasta =

'TRAV.fasta'

TRAJ_fasta = 'TRAJ.fasta'

TRDV_fasta = 'TRDV.fasta'

TRDD_fasta = 'TRDD.fasta'

TRDJ_fasta = 'TRDJ.fasta'

TRGV_fasta = 'TRGV.fasta'

TRGJ_fasta = 'TRGJ.fasta'

The

following directory and files will not be packaged with vdj

but will be computed the first

time refseq is imported. After

that, it will not be recomputed unless it is forced

#

persistent data directory

pickle_dir = 'pickle'

Relevant LIGM records in pickle format

#

If the file exists, refseq will not try to recompute it

unless it's forced

IGHV_pickle =

'IGHV.pickle'

IGHD_pickle = 'IGHD.pickle'

IGHJ_pickle = 'IGHJ.pickle'

IGKV_pickle =

'IGKV.pickle'

IGKJ_pickle = 'IGKJ.pickle'

IGLV_pickle = 'IGLV.pickle'

vdj_package.txt

```

IGLJ_pickle =
'IGLJ.pickle'
TRBV_pickle = 'TRBV.pickle'
TRBD_pickle = 'TRBD.pickle'
TRBJ_pickle =
'TRBJ.pickle'
TRAV_pickle = 'TRAV.pickle'
TRAJ_pickle = 'TRAJ.pickle'
TRDV_pickle =
'TRDV.pickle'
TRDD_pickle = 'TRDD.pickle'
TRDJ_pickle = 'TRDJ.pickle'
TRGV_pickle =
'TRGV.pickle'
TRGJ_pickle = 'TRGJ.pickle'

# # full IMGT flatfile database dir
# imgt_dir =

'/Users/laserson/research/church/vdj-ome/ref-data/IMG'
# ligm_filename = 'imgt.dat'

#

refdatadir = '/Users/laserson/research/church/vdj-ome/ref-data/IMG'
# imgtdat =

'imgt.dat'
# imgtfasta = 'imgt.fasta'
# imgtvseq = 'vdj_ref.fasta'
# imgtrefseqfasta =

'imgtrefseq.fasta'
# imgtspecfasta = 'imgtspec.fasta'
# imgtspecvdjxml =

'imgtspec.vdjxml'
# imgtrefseqFR3endcoords = 'imgtrefseqFR3endcoords.dat'
#

imgtrefseqJTRPstartcoords = 'imgtrefseqJTRPstartcoords.dat'
# functions for pipeline

operations

import vdj
import vdj.clustering
import seqtools

def

iterator2parts(iterator,basename,packetsize,prefix='',suffix=''):
    """Split data from

iterator into multiple files"""
    parts = []
    num_processed = 0
    file_num = 1

```


vdj_package.txt

```

curr_outname = basename+'.'+str(file_num)
    for obj in iterator:
        if
num_processed == 0:
    op = open(curr_outname, 'w')
    print >>op, prefix

    parts.append(curr_outname)

    print >>op, chain
    num_processed += 1

    if num_processed == packetsize:
        print >>op, suffix

op.close()
    num_processed = 0
    file_num += 1
    curr_outname
= basename+'.'+str(file_num)
    if not op.closed:
        print >>op, suffix

op.close()
    return parts

def load_barcode(barcode_file):
    bcip =

open(barcode_file, 'r')
    barcodes = {}
    for (descr, seq) in
seqtools.FastaIterator(bcip):
        barcodes[seq.upper()] = descr
    bcip.close()

# check that barcodes meet necessary criteria
    barcode_len = len(barcodes.keys()[0])

for bc in barcodes.keys():
    if len(bc) != barcode_len:
        raise Exception,

"ERROR: All barcode lengths must be equal."

    return barcodes

```



```

                                vdj_package.txt
def
    id_barcode(chain, barcodes):
        # barcodes assumed to be single length
        barcode_len =
            len(barcodes.keys()[0])
        try:
            curr_barcode =
                barcodes[chain.seq[:barcode_len].upper()]
        except KeyError:
            # barcode not found; chain
            unchanged
            return # chain remains unchanged
            chain.seq =
                chain.seq[barcode_len:] # prune off barcode from seq
            chain.barcode = curr_barcode
def
    load_isotypes(isotype_file):
        ighcip = open(isotype_file, 'r')
        isotypes = {}
        for
            (descr, seq) in seqtools.FastaIterator(ighcip):
                isotypes[seq.upper()] = descr
    ighcip.close()
    return isotypes
def id_isotype(chain, isotypes):
    if not
        chain.has_tag('positive') and not chain.has_tag('coding'):
            warnings.warn('chain %s
                may not be the correct strand' % chain.descr)
            for iso in isotypes.iteritems():
                if iso[0] in chain.seq[-50:]: # arbitrary cutoff from 3' end
                    chain.c =
                        iso[1]
def cat_vdjxml(files, outhandle):
    print >>outhandle, "<root>"
    for f in
        files:
            inhandle = open(f, 'r')
            for chain in vdj.parse_VDJXML(inhandle):
                print >>outhandle, chain
            print >>outhandle, "</root>"

```



```

                                vdj_package.txt
def
partition_VJ(inhandle,basename):
    # ignores allele numbers
    def
vj_id_no_allele(chain):
    return seqtools.cleanup_id(chain.v.split('*')[0]) + '_' +
seqtools.cleanup_id(chain.j.split('*')[0])
    def outname(basename,vj_id):

return "%s.%s.vdjxml" % (basename,vj_id)
    outhandles = {}
    for chain in
vdj.parse_VDJXML(inhandle):
    curr_vj_id = vj_id_no_allele(chain)
    try:

    print >>outhandles[curr_vj_id], chain
        except KeyError:

outhandles[curr_vj_id] = open( outname(basename,curr_vj_id), 'w' )
    print
>>outhandles[curr_vj_id], "<root>"
    print >>outhandles[curr_vj_id], chain

    for outhandle in outhandles.itervalues():
        print >>outhandle, "</root>"

return [outname(basename,vj_id) for vj_id in outhandle.iterkeys()]
# =====
# =

matplotlib =
# =====

# some tools to automate MPL manipulations that I makeimport
os
import sys
import types
import cPickle as pickle

import params
import refsequtils

#

=====
# = First-time initializations =

```



```

                                vdj_package.txt
#
=====
# does the pickle directory exist?
if not os.path.exists(
os.path.join(params.vdj_dir,params.pickle_dir) ):
    os.mkdir(
os.path.join(params.vdj_dir,params.pickle_dir) )
# test for each gene type pickle file
if
not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.IGHV_pickle)):
refsequtils.process_IMGT_references(refsequtils.VReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.IGHV_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.IGHV
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.IGHD_pickle)):
refsequtils.process_IMGT_references(
refsequtils.ReferenceEntry,os.path.join(params.vdj_dir,params.data_dir,params.IGHD_f
asta),os
.path.join(params.vdj_dir,params.pickle_dir,params.IGHD_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.IGHJ_pickle)):
refsequtils.process_IMGT_references(refsequtils.JReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.IGHJ_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.IGHJ
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.IGKV_pickle)):
refsequtils.process_IMGT_references(refsequtils.VReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.IGKV_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.IGKV
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.IGKJ_pickle)):
refsequtils.process_IMGT_references(refsequtils.JReferenceEntry,os.path.join(params.
vdj_dir,

```


vdj_package.txt

```

params.data_dir,params.IGKJ_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.IGKJ
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.IGLV_pickle)):
refsequtils.process_IMGT_references(refsequtils.VReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.IGLV_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.IGLV
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.IGLJ_pickle)):
refsequtils.process_IMGT_references(refsequtils.JReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.IGLJ_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.IGLJ
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRBV_pickle)):
refsequtils.process_IMGT_references(refsequtils.VReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.TRBV_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.TRBV
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRBD_pickle)):
refsequtils.process_IMGT_references(
refsequtils.ReferenceEntry,os.path.join(params.vdj_dir,params.data_dir,params.TRBD_f
asta),os
.path.join(params.vdj_dir,params.pickle_dir,params.TRBD_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRBJ_pickle)):
refsequtils.process_IMGT_references(refsequtils.JReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.TRBJ_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.TRBJ
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRAV_pickle)):

```



```

                                vdj_package.txt
refsequtils.process_IMGT_references(refsequtils.VReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.TRAV_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.TRAV
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRAJ_pickle)):
refsequtils.process_IMGT_references(refsequtils.JReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.TRAJ_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.TRAJ
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRDV_pickle)):
refsequtils.process_IMGT_references(refsequtils.VReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.TRDV_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.TRDV
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRDD_pickle)):
refsequtils.process_IMGT_references(
refsequtils.ReferenceEntry,os.path.join(params.vdj_dir,params.data_dir,params.TRDD_f
asta),os
.path.join(params.vdj_dir,params.pickle_dir,params.TRDD_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRDJ_pickle)):
refsequtils.process_IMGT_references(refsequtils.JReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.TRDJ_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.TRDJ
_pickle),verbose=True)
if not
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRGV_pickle)):
refsequtils.process_IMGT_references(refsequtils.VReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.TRGV_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.TRGV
_pickle),verbose=True)
if not

```



```

                                vdj_package.txt
os.path.exists(os.path.join(params.vdj_dir,params.pickle_dir,params.TRGJ_pickle)):
refseutils.process_IMGT_references(refseutils.JReferenceEntry,os.path.join(params.
vdj_dir,
params.data_dir,params.TRGJ_fasta),os.path.join(params.vdj_dir,params.pickle_dir,par
ams.TRGJ
_pickle),verbose=True)
# at this point, there should be pickle files with fully processed
reference genes
# (including the LIGM-dependent parts)

# =====
# = Load

reference data =
# =====

IGHV =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.IGHV_pickle)))
IGHD =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.IGHD_pickle)))
IGHJ =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.IGHJ_pickle)))
IGKV =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.IGKV_pickle)))
IGKJ =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.IGKJ_pickle)))
IGLV =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.IGLV_pickle)))
IGLJ =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.IGLJ_pickle)))
# TRBV

=
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRBV_pickle)))
#

TRBD =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRBD_pickle)))
# TRBJ

=
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRBJ_pickle)))
#

TRAV =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRAV_pickle)))
# TRAJ

```



```

                                vdj_package.txt
=
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRAJ_pickle)))
#
TRDV =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRDV_pickle)))
# TRDD
=
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRDD_pickle)))
#
TRDJ =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRDJ_pickle)))
# TRGV
=
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRGV_pickle)))
#
TRGJ =
pickle.load(open(os.path.join(params.vdj_dir,params.pickle_dir,params.TRGJ_pickle)))

#
=====
# = Define data structures for
compatibility with aligner =
# =====
#
IGHn_list -- list of the refseq identifiers (e.g., IGHJ1, IGHV6-1, etc.)
# IGHn_seqs -- dict
where the keys are refseq IDs and the values are the actual sequences
#
refseq.
# IGHn_idx -- dict where the keys are refseq IDs and the values are the indices into
the
# IGHn_list list.
# IGHV_offset
def legacy_data(ref_data):
    """Generate data structure used by aln tools etc."""
    locus_list = [elt.allele for elt in
ref_data]
    locus_seqs = dict([(elt.allele,elt.seq) for elt in ref_data])
    locus_idx =
dict([(g,i) for i,g in enumerate(locus_list)])
    try:

```



```

                                vdj_package.txt
    locus_offset =

dict([(elt.allele,elt.cdr3_boundary-elt.accession_coords[0]) for elt in ref_data])

return (locus_list,locus_seqs,locus_idx,locus_offset)
    except AttributeError:

return (locus_list,locus_seqs,locus_idx)

(IGHV_list,IGHV_seqs,IGHV_idx,IGHV_offset) =
legacy_data(IGHV)
(IGHD_list,IGHD_seqs,IGHD_idx)          = legacy_data(IGHD)
(IGHJ_list,IGHJ_seqs,IGHJ_idx,IGHJ_offset) = legacy_data(IGHJ)
(IGKV_list,IGKV_seqs,IGKV_idx,IGKV_offset) = legacy_data(IGKV)
(IGKJ_list,IGKJ_seqs,IGKJ_idx,IGKJ_offset) = legacy_data(IGKJ)
(IGLV_list,IGLV_seqs,IGLV_idx,IGLV_offset) = legacy_data(IGLV)
(IGLJ_list,IGLJ_seqs,IGLJ_idx,IGLJ_offset) = legacy_data(IGLJ)
#

(TRBV_list,TRBV_seqs,TRBV_idx,TRBV_offset) = legacy_data(TRBV)
#

(TRBD_list,TRBD_seqs,TRBD_idx)              = legacy_data(TRBD)
#

(TRBJ_list,TRBJ_seqs,TRBJ_idx,TRBJ_offset) = legacy_data(TRBJ)
#

(TRAV_list,TRAV_seqs,TRAV_idx,TRAV_offset) = legacy_data(TRAV)
#

(TRAJ_list,TRAJ_seqs,TRAJ_idx,TRAJ_offset) = legacy_data(TRAJ)
#

(TRDV_list,TRDV_seqs,TRDV_idx,TRDV_offset) = legacy_data(TRDV)
#

(TRDD_list,TRDD_seqs,TRDD_idx)              = legacy_data(TRDD)
#

(TRDJ_list,TRDJ_seqs,TRDJ_idx,TRDJ_offset) = legacy_data(TRDJ)
#

(TRGV_list,TRGV_seqs,TRGV_idx,TRGV_offset) = legacy_data(TRGV)
#

(TRGJ_list,TRGJ_seqs,TRGJ_idx,TRGJ_offset) = legacy_data(TRGJ)


# # refseq.py
#
#
"""Contains data and functions for dealing with reference IMGT data.
#
# refdatadir must be

```



```

                                vdj_package.txt
set to the directory where the IMGT flat file release
# is present. This must include

imgt.dat, imgt.fasta, and imgtrefseq.fasta.
#
# IGHn_list -- list of the refseq identifiers
(e.g., IGHJ1, IGHV6-1, etc.)
# IGHn_acc -- dict where keys are refseq IDs and values are the
IMGT accession numbers
# IGHn_coords -- dict where keys are refseq IDs and values are pairs
of coords that are
#                  extracted from the refseq database. These numbers are
unmodified seq
#                  coords of the reference element within the LIGM entry
#
IGHn_seqs -- dict where the keys are refseq IDs and the values are the actual
sequences
#
#                  from refseq.
# IGHn_idx -- dict where the keys are refseq IDs and the values
are the indices into the
#                  IGHn_list list.
# IGHV_FR3_IMGT_end_coord -- dict
where the keys are refseq IDs and the values are the coord of
#
the end of FR3-IMGT (incl the 2nd-CYS) unmodified
# IGHJ_J_TRP_start_coord -- dict where the
keys are refseq IDs and the values are the
#                  coord of the
start of the J-TRP site unmodified
#
# ""
#
# import os
# import cPickle
#
# from Bio

import SeqIO
#
# import vdj
# import params
# import sequtils
#
# #

=====
# # = UTILITY FNS for parsing reference

databases =
# # =====
#

```



```

                                vdj_package.txt
# def
get_refseq_elements(locus,alleles,func,species,getcoords,refdatadir,imgtrefseqfasta)
:
#
"""Extract all identifiers that meet certain criteria from IMGT/GENE-DB (refseq).
#
#
    locus -- identifier such as 'IGH', 'TRBV', etc
#     alleles -- can be either '01' or
'all'
#     func -- list of func IDs, e.g., ['F','ORF','P']. If empty, includes
everything
#     species -- species of animal (e.g., 'Homo+sapiens')
#     getcoords --
determines whether to process and return coords (boolean)
#
#     refdatadir --
directory where IMGT resides
#     imgtrefseqfasta -- filename of imgtrefseq fasta file
#

#     """
#
#     refIDs = []
#     refacc = {}
#     refseqs = {}
#     if

getcoords:
#         refcoords = {}
#         allrefseqs = sequtils.load_fasta(
os.path.join(refdatadir,imgtrefseqfasta) )
#
#         if func == []:
#             func =
['F','ORF','P']
#
#         for seq in allrefseqs:
#             # get info on curr seq
#
currdescr = seq.description
#         currlocus = currdescr.split(',')[0].split('*')[0]
#
#         currallele = currdescr.split(',')[0].split('*')[1]
#         currspecies =
currdescr.split(',')[1]
#         currfunc = currdescr.split(',')[2].split()[-1].strip('
()')
#         currID = currdescr.split(',')[0].lstrip('>')
```



```

                                vdj_package.txt
#         curracc =
currdescr.split(',')[3]
#         if getcoords:
#             try:
#
currcoords = ( eval(currdescr.split(',')[4].split('.')[0]),
eval(currdescr.split(',')[4].split('.')[2]) )
#         except:
#             #print
"Coordinates for", currlocus, curracc, "are not interpretable"
#             pass
#

#         # perform tests to exclude current seq
#         if locus not in currlocus:
#
#             continue
#         if alleles != 'all':
#             if currallele !=
alleles:
#                 continue
#             if currspecies != species:
#
#                 continue
#         if currfunc not in func:
#             continue
#
#         # check for "partial"
#         if len(currdescr.split(',')) > 5 and
'partial' in ' '.join(currdescr.split(',')[5:]):
#             continue
#
#
refIDs.append( currID )
#         refacc[currID] = curracc
#         refseqs[currID] =
seq.seq.toString().upper()
#         if getcoords:
#             refcoords[currID] =
currcoords
#
#         refIDs.append( ' ' )
#         refIDs.sort()
#
#         if not

```



```

getcoords:
#         return refIDs,refacc,refseqs
#         return

refIDs,refacc,refseqs,refcoords
#
#
#
# # =====
# # =

Definition of reference data =
# # =====
#
# LOCI =

['IGH','IGK','IGL','TRA','TRB','TRD','TRG']
#
# IGHV_list,IGHV_acc,IGHV_seqs,IGHV_coords =

get_refseq_elements(locus='IGHV',alleles='01',func=['F','ORF'],species='Homo+sapiens',getcoo

rds=True,refdatadir=params.refdatadir,imgtrefseqfasta=params.imgtrefseqfasta)
#

IGHD_list,IGHD_acc,IGHD_seqs =

get_refseq_elements(locus='IGHD',alleles='01',func=['F','ORF'],species='Homo+sapiens',getcoo

rds=False,refdatadir=params.refdatadir,imgtrefseqfasta=params.imgtrefseqfasta)
#

IGHJ_list,IGHJ_acc,IGHJ_seqs,IGHJ_coords =

get_refseq_elements(locus='IGHJ',alleles='01',func=['F','ORF'],species='Homo+sapiens',getcoo

rds=True,refdatadir=params.refdatadir,imgtrefseqfasta=params.imgtrefseqfasta)
#
# IGHV_idx

= dict([(g,i) for i,g in enumerate(IGHV_list)])
# IGHD_idx = dict([(g,i) for i,g in
enumerate(IGHD_list)])
# IGHJ_idx = dict([(g,i) for i,g in enumerate(IGHJ_list)])
#
#

IGHC_list = [
#
#         'IGHA1',
#         'IGHA2',
#
#
#         'IGHD',
#         'IGHE',
#         'IGHG1',
#         'IGHG2',
#

```


vdj_package.txt

```

'IGHG3',
#           'IGHG4',
#           'IGHM'
#       ]
# IGHC_idx =

dict([(g,i) for i,g in enumerate(IGHC_list)])
#
# ALL_IDS = list(set(IGHV_list + IGHD_list
+ IGHJ_list + IGHC_list))
# ALL_IDS.sort()
#
#
# # =====
# # =

Specificity data =
# # =====
#
# def

get_LIGM_with_specificities(refdatadir,imgtdat,imgtfasta,outputfasta,outputvdjxml):
#
'''
#     Take the IMGT LIGM flat and fasta files, and return a fasta with only those
seqs
#
#     that have a specificity associated with them in a fasta file. The header
contains
#
#     the accession and the specificity.
#
#
vdj.refseq.get_LIGM_with_specificities("imgt.dat","imgt.fasta","imgt.specificities.f
asta","i
mgt.specificities.vdjxml")
#
#     specificities = {}           # list of 2-tuples:
#
#     (ACCESSION,specificity)
#     LIGMflat = open(os.path.join(refdatadir,imgtdat),'r')
#
LIGMfasta = open(os.path.join(refdatadir,imgtfasta),'r')
#     opSpecificityfasta =
open(os.path.join(refdatadir,outputfasta),'w')
#
#     numRecords = 0
#
numRecordswithSpec = 0
#
#     ID = ''
#     DE = ''
#     for line in LIGMflat:

```


vdj_package.txt

```

#
    splitline = line.split()
#
#     if splitline[0] == 'ID':
#
inRecord = True
#         ID = splitline[1]
#         numRecords += 1
#     elif
splitline[0] == 'DE':
#         if inRecord:
#             DE += '
'.join(splitline[1:]) + ' '
#         elif splitline[0] == 'XX':
#             if DE == '':
#
# if i haven't stored the description yet
#         continue
#     else:
# finished record
#         if 'specificity' in DE:
#             specidx =
DE.rfind('specificity')
#             spec =
DE[specidx+len('specificity'):].strip().rstrip('.')
#             if
spec.startswith('anti'):
#                 specificities[ID] = spec
#
#         numRecordswithSpec += 1
#             ID = ''
#             DE = ''
#
#         inRecord = False
#
#     print "Number of LIGM records read: " +
str(numRecords)
#     print "Number of LIGM records that have specificities: " +
str(numRecordswithSpec)
#
#     numFasta = 0
#
#     for seq in
SeqIO.parse(LIGMfasta, 'fasta'):
#         spec = specificities.get(seq.id, '')
#         if
spec == '':
#             continue
#         else:

```



```

                                vdj_package.txt
#         print >>opSpecificityfasta,
">" + seq.id + " | " + spec
#         print >>opSpecificityfasta,

seq.seq.toString().upper()
#         numFasta += 1
#
#     print "Number of Fasta
records with specificities found and printed: " + str(numFasta)
#
#

LIGMflat.close()
#     LIGMfasta.close()
#     opSpecificityfasta.close()
#
#

VDJXML and alignment
#     rep =

vdj.initial_import([os.path.join(refdatadir,outputfasta)],os.path.join(refdatadir,ou
tputvdjx

ml),metatags=['specificity_reference : '+vdj.timestamp()])
#     rep =

vdj.positive_strand(rep)
#     rep = vdj.align_rep(rep)
#
#     #repfiltered =

rep.get_chains_fullVJ()
#     repfiltered = rep
#     for chain in repfiltered:
#

spec = specificities.get(chain.descr,'')
#         if spec == '':
#             print

"Reference chain has empty specificity: " + chain.descr
#         continue
#

else:
#         chain.add_tags( 'specificity|'+spec )
#
#

vdj.writeVDJ(repfiltered,os.path.join(refdatadir,outputvdjxml))
#
#     return
#
#
#

if not os.path.exists(os.path.join(params.refdatadir,params.imgtspecfasta)) or not
os.path.exists(os.path.join(params.refdatadir,params.imgtspecvdjxml)):
#

```


vdj_package.txt

```

get_LIGM_with_specificities(params.refdatadir,params.imgtdat,params.imgtfasta,params
.imgtspe
cfasta,params.imgtspecvdjxml)
#
#
# if
os.path.exists(os.path.join(params.refdatadir,params.imgtspecfasta)) and
os.path.exists(os.path.join(params.refdatadir,params.imgtspecvdjxml)):
#     ipspecfasta =
open(os.path.join(params.refdatadir,params.imgtspecfasta),'r')
#
#     SPEC_list =
set()
#     for line in ipspecfasta:
#         if line[0] == '>':
#             currspec =
'|'.join(line.split('|')[1:]).strip()
#             SPEC_list.add(currspec)
#
ipspecfasta.close()
#     SPEC_list.add('')
#     SPEC_list = list(SPEC_list)
#
SPEC_list.sort()
import sys
import string
from cStringIO import StringIO
import cPickle as
pickle
import urllib2

import warnings # this is because IMGT has tons of errors

import
ClientForm
from Bio import SeqIO

import seqtools
import params

identity =
string.maketrans('','')

# =====
# = Data structures =
#
=====

class ReferenceEntry(object):
    """Data structure to hold a reference

```


vdj_package.txt

sequence from IMGT/GENE-DB or
IMGT/V-QUEST.

Some of the attributes are
computed, and some are taken from IMGT.

accession_coords uses converted python
numbering
gapped_seq includes IMGT gaps that they provide
frame uses python coords:

0 means already in frame.

1 means skip one nt

2 means skip two nts
partial will take values of "partial in 5'", "partial
in 3'", or "partial in 5' and 3'"
depending where the del is.

def

```
__init__(self,**kw):
    def kw_init(attrib):
        if kw.has_key(attrib):

            self.__setattr__(attrib,kw[attrib])
            kw_init('accession')
```

```
kw_init('seq')
    kw_init('gapped_seq')
    kw_init('description')
```

```
kw_init('locus')
    kw_init('gene')
    kw_init('allele')
```

```
kw_init('species')
    kw_init('functional')
    kw_init('imgt_label')
```

```
kw_init('accession_coords')
    kw_init('length')
    kw_init('frame')
```

```
kw_init('partial')
```

```
def init_from_imgt(self,fasta_header,seq):
    """Initialize
```

```
object from IMGT fasta header and seq"""
```



```

                                vdj_package.txt
    data =

    fasta_header.lstrip('>').rstrip().split('|')
        self.accession = data[0]

    self.gapped_seq = seq
        # self.seq = seq.translate(None, '.').upper() # remove periods
    (gaps) # python >2.5
        self.seq = seq.translate(identity, '.').upper()

    self.description = data[1]
        self.locus = self.description[0:4]
        self.gene =

    self.description.split('*')[0]
        self.allele = self.description
        self.species =

    data[2]
        self.functional = data[3]
        self.imgt_label = data[4]

    raw_coords_start = int(data[5].split('.')[0])
        raw_coords_end =

    int(data[5].split('.')[-1])
        coords = (raw_coords_start - 1, raw_coords_end) # note
    the conversion to python coord system
        self.accession_coords = coords

    self.length = len(self.seq)
        if self.length != int(data[6].split()[0]):

    raise ValueError, "Lengths are inconsistent: %s" % fasta_header
        self.frame =

    int(data[7]) - 1 # note change to python numbering (0-based)
        self.partial =

    data[13]

    def pull_LIGM_record(self):
        """Get SeqRecord object for LIGM
    record from IMGT server"""

        # NOTE: this can potentially be significantly
    simplified by accessing the URL
        # interface to LIGM, through:
        #
    http://imgt.cines.fr/cgi-bin/IMGTelect.jv?query=5+numacc
        # where numacc is the
    accession number

```


vdj_package.txt

```

    request =
urllib2.Request('http://imgt.cines.fr/cgi-bin/IMGTlect.jv?livret=0')
    # LIGM page

    response = urllib2.urlopen(request)
    forms = ClientForm.ParseResponse(response,

form_parser_class=ClientForm.XHTMLCompatibleFormParser,

    backwards_compat=False)
    form = forms[1]
    form['101p01c02'] =

self.accession
    request2 = form.click()
    # data format page
    response2

= urllib2.urlopen(request2)
    forms2 = ClientForm.ParseResponse(response2,

                                form_parser_class=ClientForm.XHTMLCompatibleFormParser,

                                backwards_compat=False)

    form2 = forms2[0]

assert( form2.controls[8].attrs['value'] == '2 IMGT flat-file' )

form2.controls[8].id = 'flatfile'
    request3 = form2.click(id='flatfile')
    #

LIGM record results
    response3 = urllib2.urlopen(request3)

    # ghetto

parse of the results.  the text of the LIGM record is in <pre>...</pre> tags

rawdata1 = response3.read()
    rawdata2 =

rawdata1.split('<pre>')[1].split('</pre>')[0].lstrip()
    rawdata3 =

StringIO(rawdata2)
    self.record = SeqIO.read(rawdata3,'imgt')

class

VReferenceEntry(ReferenceEntry):

```



```

                                vdj_package.txt
def __init__(self,**kw):

ReferenceEntry.__init__(self,**kw)

    def set_CDR3_boundary(self):    # FR3 end

        """Get coord of end of FR3 from IMGT LIGM database."""
        # some records can have
        multiple references in them
        target_allele = self.allele
        feature_iter =
self.record.features.__iter__()
        v_gene =
seqtools.advance_to_features(feature_iter,['V-REGION','V-GENE'])
        while
v_gene.qualifiers.get('allele',[''])[0] != target_allele: # advance to the target
gene

        v_gene = seqtools.advance_to_features(feature_iter,['V-REGION','V-GENE'])

conserved_cys = seqtools.advance_to_feature(feature_iter,'2nd-CYS')

    # note:
biopython features already use pythonic indexing
    self.cdr3_boundary =
conserved_cys.location.start.position
class JReferenceEntry(ReferenceEntry):
    def
__init__(self,**kw):
        ReferenceEntry.__init__(self,**kw)

    def
set_CDR3_boundary(self):    # FR4 start
        """Get coord of start of FR4 from IMGT LIGM
database."""
        # some records can have multiple references in them

        target_allele = self.allele
        feature_iter = self.record.features.__iter__()

        j_gene = seqtools.advance_to_features(feature_iter,['J-REGION','J-GENE'])
        while
j_gene.qualifiers.get('allele',[''])[0] != target_allele: # advance to the target
gene

```



```

                                vdj_package.txt
j_gene = seqtools.advance_to_features(feature_iter,['J-REGION','J-GENE'])
#
note, there can be a conserved TRP or PHE
    conserved_trp =
seqtools.advance_to_features(feature_iter,['J-TRP','J-PHE'])
    # note:
biopython features already use pythonic indexing
    self.cdr3_boundary =
conserved_trp.location.end.position

# =====
# = Parsing IMGT =
#
=====

# import pdb
def
process_IMGT_references(ref_entry_cls,fasta_infilename,pickle_outfilename,verbose=False):

    """Load references from the IMGT/V-QUEST fasta file refs
        e.g., IGHV.fasta, present
in the data directory
    ref_entry_cls is the class object for the reference type,

e.g., VReferenceEntry, JReferenceEntry, etc.

    # pdb.set_trace()

references = []
    ip = open(fasta_infilename,'r')
    for record in
SeqIO.parse(ip,'fasta'):
        curr_reference = ref_entry_cls()
        curr_header =
record.description
        curr_seq = record.seq.tostring()

        # Potential
problems with FASTA headers
        try:

curr_reference.init_from_imgt(curr_header,curr_seq)

```



```

                                vdj_package.txt
except ValueError:

warnings.warn("Invalid header: %s" % curr_header)
    continue

    # I
don't want to deal with partial seqs right now
    if 'partial' in
curr_reference.partial:
    continue

curr_reference.pull_LIGM_record()
    # Potential problems with finding annotated CDR3
boundary
    try:
        curr_reference.set_CDR3_boundary()
    except
AttributeError:
    # ReferenceEntry has no set_CDR3_boundary. Used for D segments

    pass
except ValueError, e:
    warnings.warn("Failed to find CDR3
boundary in %s. skipping..." % curr_reference.allele)
    continue

references.append(curr_reference)
    if verbose: print "Finished processing %s" %
curr_reference.allele
    sys.stdout.flush()
    ip.close()

    op =
open(pickle_outfilename, 'w')
    pickle.dump(references, op, protocol=2)
    op.close()

    return references
# setup.py
from numpy.distutils.core import setup, Extension
from
numpy.distutils.misc_util import get_numpy_include_dirs
alignmentcoreext = Extension(

    "alignmentcore",

```



```

                                vdj_package.txt
                                ["alignmentcore.c"],

                                include_dirs = get_numpy_include_dirs()
                                )

clusteringcoreext = Extension(
                                "clusteringcore",

                                ["clusteringcore.c"],
                                include_dirs = get_numpy_include_dirs()

                                )

setup( name = "vdj",
        version = "0.1",
        ext_modules =

        [alignmentcoreext,clusteringcoreext]
        )

```

Lots of scripts for performing vdj operations.

fasta2vdjxml.py	Convert fasta file to vdjxml. Takes first white-space
delim	
field for descr	
size_select.py	Size select reads
vdjxml2parts.py	
split vdjxml file into parts	
barcode_id.py	Annotate barcode onto sequences
coding_strand.py	Convert chains to coding sequence
isotype_id.py	
Annote isotypes	
align_vdj.py	Perform vdj classification
cat_vdjxml.py	
cat operation on vdjxml files (handles root elements)	
filter_VJ.py	
Select only chains with V and J alignments	
cluster_cdr3.py	Perform hierarchical
clustering of ImmuneChains using their junctions	
partition_VJ.py	Partitions
vdjxml into files by VJ combo	
update_vdjxml.py	Update vdjxml from older version
to newer version	
Older generation:	
cluster_split_VJ.py	
cluster_split_VJ_LSF.py	
filter_tags_and.py	
filter_tags_not.py	
filter_tags_or.py	
split_on_tags.py	
tag_chains.py	

vdj_package.txt

```

vdj_full_pipeline_LSF.py
vdjxml2clone_counts.py
vdjxml2fasta.py
#!/usr/bin/env python

import sys
import optparse

import vdj
import vdj.alignment

parser =

optparse.OptionParser()
parser.add_option('-L', '--locus', action='append', dest='loci')
(options, args) = parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')

    outhandle = open(args[1], 'w')
elif len(args) == 1:
    inhandle = open(args[0], 'r')

outhandle = sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outhandle =

sys.stdout
else:
    raise Exception, "Wrong number of arguments."

aligner =

vdj.alignment.vdj_aligner_combined(loci=options.loci)
print >>outhandle, "<root>"
for chain

in vdj.parse_VDXML(inhandle):
    aligner.align_chain(chain)
    print >>outhandle, chain
print >>outhandle, "</root>"
#!/usr/bin/env python

import sys
import optparse

import

seqtools

import vdj
import vdj.pipeline

parser = optparse.OptionParser()
parser.add_option('-b', '--barcodes', dest='barcodes_fasta')
(options, args) =

parser.parse_args()

```



```

                                vdj_package.txt
if len(args) == 2:
    inhandle = open(args[0], 'r')
    outhandle =

open(args[1], 'w')
elif len(args) == 1:
    inhandle = open(args[0], 'r')
    outhandle =

sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outhandle = sys.stdout

# NOTE:
all barcodes must be the same length
barcodes =

vdj.pipeline.load_barcodes(options.barcodes_fasta)

# iterate through chains
print

>>outhandle, "<root>"
for chain in vdj.parse_VDXML(inhandle):

vdj.pipeline.id_barcode(chain, barcodes)
    print >>outhandle, chain
print >>outhandle,

"</root>"
#!/usr/bin/env python

import sys
import optparse
import glob

import

vdj.pipeline

parser = optparse.OptionParser()
(options, args) = parser.parse_args()

files

= []
for arg in args:
    files.extend(glob.glob(arg))

vdj.pipeline.cat_vdxml(files, sys.stdout)
#!/usr/bin/env python

import sys
import

optparse

import vdj
import vdj.clustering

```


vdj_package.txt

```

parser = optparse.OptionParser()
parser.add_option('-c', '--cutoff', type='float', default=4.5)
parser.add_option('-t', '--tag', default='')
parser.add_option('-l', '--linkage', type='choice', choices=['single', 'complete'], default='single')
(options, args) = parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')
    outhandle = open(args[1], 'w')
elif len(args) == 1:
    inhandle = open(args[0], 'r')
    outhandle = sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outhandle = sys.stdout

# NOTE: this script requires there to be a well-defined junction
#
#       sequence. It raises an exception if not. Therefore, seqs
#       must be
#
# pre-filtered for having legit junctions
# NOTE: this script must hold all chains in memory
#
# in order to
#       perform the clustering and then assign cluster names
#
# load data
chains = []
junctions = []
for chain in vdj.parse_VDJXML(inhandle):
    # check for presence
    # of V, J, and non-trivial junction
    if not hasattr(chain, 'v') or not hasattr(chain, 'j') or
    not hasattr(chain, 'junction'):
        raise ValueError, "Chain %s has no junction of V-J
aln." % chain.descr
        chains.append(chain)
        junctions.append(chain.junction)

# perform
# the sequence clustering
(T, seq_idx) =
vdj.clustering.cluster_seqs(junctions, options.cutoff, options.linkage)
# tag chains with

```


vdj_package.txt

```

unique cluster IDs
if options.tag == '':
    tag = ''
else:
    tag = options.tag+'|'

print

>>outhandle, "<root>"
for (i,chain) in enumerate(chains):
    cloneID = '%s%s' %

(tag,T[seq_idx[chain.junction]])
    chain.clone = cloneID
    print >>outhandle, chain
print >>outhandle, "</root>"
#!/usr/bin/env python

import sys
import optparse
import

subprocess
import time

import vdj

parser = optparse.OptionParser()
parser.add_option('-c', '--cutoff', default=4.5, type='float')
(options, args) =

parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')
    outname =

args[0]
    outhandle = open(args[1], 'w')
elif len(args) == 1:
    inhandle =

open(args[0], 'r')
    outname = args[0]
    outhandle = sys.stdout
elif len(args) == 0:

inhandle = sys.stdin
    outname = 'VJ_parts.vdxml'
    outhandle = sys.stdout

print

>>sys.stderr, "NOTE: chains must be filtered for valid VJ aln and junctions BEFORE
clustering."

(VJ_parts,VJ_IDS) = vdj.split_vdxml_into_VJ_parts(inhandle,outname)

VJ_parts_clustered = []
processes = []
for (vj_file,vj_id) in zip(VJ_parts,VJ_IDS):

```


vdj_package.txt

```

vj_file_clustered = vj_file + '.clustered'
VJ_parts_clustered.append(vj_file_clustered)

params = {'cutoff':options.cutoff,
          'tag':vj_id,

'infile':vj_file,
          'outfile':vj_file_clustered}
cluster_cmd = r'python
cluster_cdr3 --cutoff %(cutoff)f --tag %(tag)s %(infile)s %(outfile)s' % params
p =
subprocess.Popen(cluster_cmd,shell=True)
processes.append(p)
# perform serially:

# vdj.cluster_chains(options.cutoff,vj_id,vj_file,vj_file_clustered)
vdj.wait_for_subprocesses(processes)
for chain in
vdj.parse_VDXML_parts(VJ_parts_clustered):
    print >>outhandle, chain#! /usr/bin/env

python

import sys
import optparse
import subprocess
import os
import tempfile

import
vdj

parser = optparse.OptionParser()
parser.add_option('-c','--cutoff',default=4.5,type='float')
parser.add_option('-l','--linkage',type='choice',choices=['single','complete'],defau
lt='sing
le')
parser.add_option('-q','--queue')
parser.add_option('-o','--LSFoutput')
(options, args)

= parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0],'r')
    outname =

args[0]
    tempdirname = args[1]
    outhandle = open(args[1],'w')
elif len(args) == 1:

```


vdj_package.txt

```

inhandle = open(args[0], 'r')
    outname = args[0]
    tempdirname = outname
    outhandle =

sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outname = 'VJ_parts.vdjxml'

tempdirname = outname
    outhandle = sys.stdout

print >>sys.stderr, "NOTE: chains must be
filtered for valid VJ aln and junctions BEFORE clustering."
# temporary directory to dump
intermediate files
tempdir =

tempfile.mkdtemp(prefix=tempdirname+'.intermediate.', dir='.')
(VJ_parts, VJ_IDs) =
vdj.split_vdjxml_into_VJ_parts(inhandle, os.path.join(tempdir, outname))
VJ_parts_clustered =
[]
jobs = []
for (vj_file, vj_id) in zip(VJ_parts, VJ_IDs):
    vj_file_clustered =
vj_file+'.clustered'    # NOTE: vj_file already has directory prefix

VJ_parts_clustered.append(vj_file_clustered)
    params = {'cutoff': options.cutoff,

        'linkage': options.linkage,
            'tag': vj_id,

'infile': vj_file,
        'outfile': vj_file_clustered}
    cluster_cmd =
r'cluster_cdr3.py --cutoff %(cutoff)f --tag %(tag)s --linkage %(linkage)s %(infile)s

%(outfile)s' % params
    if os.stat(vj_file).st_size < 8e6:
        jobID =

vdj.LSF.submit_to_LSF(options.queue, options.LSFoutput, cluster_cmd)
    else:
        jobID =

```


vdj_package.txt

```
vdj.LSF.submit_to_LSF(options.queue,options.LSFoutput,cluster_cmd,mem_usage=4096)

jobs.append(jobID)
vdj.LSF.wait_for_LSF_jobs(jobs)
for chain in
vdj.parse_VDJXML_parts(VJ_parts_clustered):
    print >>outhandle, chain
#! /usr/bin/env

python

import sys
import optparse

import vdj
import vdj.alignment

parser =

optparse.OptionParser()
parser.add_option('-L','--locus',action='append',dest='loci')
(options, args) = parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0],'r')

    outhandle = open(args[1],'w')
elif len(args) == 1:
    inhandle = open(args[0],'r')

outhandle = sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outhandle =

sys.stdout

aligner = vdj.alignment.vdj_aligner_combined(loci=options.loci)
print

>>outhandle, "<root>"
for chain in vdj.parse_VDJXML(inhandle):

aligner.coding_chain(chain)
    print >>outhandle, chain
print >>outhandle, "</root>"
#!

/usr/bin/env python

import sys
import optparse

import seqtools
```



```

                                vdj_package.txt
import vdj

parser =

optparse.OptionParser()
(options, args) = parser.parse_args()

if len(args) == 2:

    inhandle = open(args[0], 'r')
    outhandle = open(args[1], 'w')
elif len(args) == 1:

    inhandle = open(args[0], 'r')
    outhandle = sys.stdout
elif len(args) == 0:
    inhandle =

sys.stdin
    outhandle = sys.stdout

vdj.pipeline.fasta2vdjxml(inhandle, outhandle):
#!

/usr/bin/env python

import sys
import optparse

import vdj

parser =

optparse.OptionParser()
(options, args) = parser.parse_args()

if len(args) == 2:

    inhandle = open(args[0], 'r')
    outhandle = open(args[1], 'w')
elif len(args) == 1:

    inhandle = open(args[0], 'r')
    outhandle = sys.stdout
elif len(args) == 0:
    inhandle =

sys.stdin
    outhandle = sys.stdout

print >>outhandle, "<root>"
for chain in

    vdj.parse_VDXML(inhandle):
        if hasattr(chain, 'v') and hasattr(chain, 'j'):
            print

>>outhandle, chain
print >>outhandle, "</root>"

```


vdj_package.txt

```

#!/usr/bin/env python

import sys
import

optparse

import vdj

parser = optparse.OptionParser()
parser.add_option('-t', '--tag', action='append', dest='tags')
(options, args) =

parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')
    outhandle =

open(args[1], 'w')
elif len(args) == 1:
    inhandle = open(args[0], 'r')
    outhandle =

sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outhandle = sys.stdout

query_tags = set(options.tags)
for chain in vdj.parse_VDJXML(inhandle):
    if query_tags <=

chain.all_tags:
    print >>outhandle, chain
#!/usr/bin/env python

import sys
import

optparse

import vdj

parser = optparse.OptionParser()
parser.add_option('-t', '--tag', action='append', dest='tags')
(options, args) =

parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')
    outhandle =

open(args[1], 'w')
elif len(args) == 1:
    inhandle = open(args[0], 'r')
    outhandle =

sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outhandle = sys.stdout

```


vdj_package.txt

```

empty_set = set()
query_tags = set(options.tags)
for chain in vdj.parse_VDJXML(inhandle):

    if query_tags & chain.all_tags == empty_set:
        print >>outhandle, chain
    #!

/usr/bin/env python

import sys
import optparse

import vdj

parser =

optparse.OptionParser()
parser.add_option('-t', '--tag', action='append', dest='tags')
(options, args) = parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')

    outhandle = open(args[1], 'w')
elif len(args) == 1:
    inhandle = open(args[0], 'r')

outhandle = sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outhandle =

sys.stdout

empty_set = set()
query_tags = set(options.tags)
for chain in

vdj.parse_VDJXML(inhandle):
    if query_tags & chain.all_tags != empty_set:
        print

>>outhandle, chain
#! /usr/bin/env python

import sys
import warnings
import optparse

import seqtools
import vdj
import vdj.pipeline

parser = optparse.OptionParser()
parser.add_option('-i', '--IGHC', dest='ighc_fasta')
(options, args) = parser.parse_args()

if

```


vdj_package.txt

```

len(args) == 2:
    inhandle = open(args[0], 'r')
    outhandle = open(args[1], 'w')
elif
len(args) == 1:
    inhandle = open(args[0], 'r')
    outhandle = sys.stdout
elif len(args)
== 0:
    inhandle = sys.stdin
    outhandle = sys.stdout

# load isotypes
isotypes =
vdj.pipeline.load_isotypes(options.ighc_fasta)
for chain in parse_VDJXML(inhandle):

vdj.pipeline.id_isotype(chain, isotypes)
    print >> outhandle, chain
#! /usr/bin/env

python

import sys
import optparse

import seqtools

import vdj

parser =
optparse.OptionParser()
parser.add_option('-b', '--basename')
(options, args) =

parser.parse_args()

if len(args) == 1:
    inhandle = open(args[0], 'r')
elif len(args) ==
0:
    inhandle = sys.stdin

# NOTE: this script ignores the allele numbers

vdj.pipeline.partition_VJ(inhandle, options.basename)
# base name for all data files
basename

# input files (full paths)
input_fasta      # the initial fasta data
barcode_fasta

# barcode identifiers
isotype_fasta    # isotype identifiers

```


vdj_package.txt

```

# working directories
analysis_dir    # full path; base directory for everything

# analysis parameters
min_size
max_size
packet_size
loci

#!/usr/bin/env python

import sys
import optparse

import
vdj

parser = optparse.OptionParser()
parser.add_option('-m', '--min', type='int', default=0)
parser.add_option('-M', '--max', type='int', default=float('inf'))
(options, args) =

parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')
    outhandle =

open(args[1], 'w')
elif len(args) == 1:
    inhandle = open(args[0], 'r')
    outhandle =

sys.stdout
elif len(args) == 0:
    inhandle = sys.stdin
    outhandle = sys.stdout

print

>>outhandle, "<root>"

for chain in vdj.parse_VDJXML(inhandle):
    if len(chain) >=

options.min and len(chain) <= options.max:
    print >>outhandle, chain

print

>>outhandle, "</root>"
#!/usr/bin/env python

import sys
import optparse
import os

import

```



```

                                vdj_package.txt
vdj

parser = optparse.OptionParser()
parser.add_option('-t', '--tag', action='append', dest='tags')
(options, args) =

parser.parse_args()

if len(args) == 1:
    inname = args[0]
    inhandle =

open(args[0], 'r')
else:
    raise Exception, "Need a single input file."

(basename, ext) =

os.path.splitext(inname)
basename = os.path.basename(basename)

outhandles = {}
for tag in

options.tags:
    outname = basename+'.'+vdj.sequutils.cleanup_id(tag)+ext

outhandles[tag] = open(outname, 'w')

query_tags = set(options.tags)
for chain in

vdj.parse_VDXML(inhandle):
    try:
        tag = (query_tags & chain.all_tags).pop()

        print >>outhandles[tag], chain
        except KeyError:
            continue

for handle in

outhandles.itervalues():
    handle.close()
#!/usr/bin/env python

import sys
import

optparse

import vdj

parser = optparse.OptionParser()
parser.add_option('-t', '--tag', action='append', dest='tags')
(options, args) =

parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')

```



```

                                vdj_package.txt
    outhandle =
    open(args[1], 'w')
    elif len(args) == 1:
        inhandle = open(args[0], 'r')
        outhandle =

    sys.stdout
    elif len(args) == 0:
        inhandle = sys.stdin
        outhandle = sys.stdout

    for
    chain in vdj.parse_VDXML(inhandle):
        chain.add_tags(options.tags)
        print >>outhandle,

    chain
    #! /usr/bin/env python

    import sys

    import vdj
    import vdj.legacy

    if len(sys.argv) ==
    3:
        inhandle = open(sys.argv[1], 'r')
        outhandle = open(sys.argv[2], 'w')
    elif
    len(sys.argv) == 2:
        inhandle = open(sys.argv[1], 'r')
        outhandle = sys.stdout
    elif
    len(sys.argv) == 1:
        inhandle = sys.stdin
        outhandle = sys.stdout

    for chain in
    vdj.legacy.parse_VDXML_old(inhandle):
        print >>outhandle, chain
    #! /usr/bin/env
    python

    import os

    import lsf
    import seqtools
    import vdj
    import vdj.pipeline

    join =
    os.path.join

    # PARAMETER DEFINITION

```



```

                                vdj_package.txt
# process jobfile for input parameters
# defines the
following variables:
    # basename          # unique base identifier for data
    #
input_fasta      # the initial fasta data
    # barcode_fasta    # barcode identifiers
    #
isotype_fasta    # isotype identifiers
    # analysis_dir     # full path; base directory for
everything
    # min_size         # min size selection
    # max_size         # max size
selection
    # packet_size      # packet size for alignment jobs
    # loci             # the
loci to use for VDJ aln

# working directories
parts_dir = 'parts'
log_dir = 'logs'
partition_dir = 'partitions'

# output files
raw_vdjxml = basename+'.raw.vdjxml'
aligned_file = basename+'.aligned.vdjxml'
vj_filtered_file =
basename+'.VJ_filtered.vdjxml'
size_selected_file = basename + '.size%i-%i' %
(min_size,max_size) + '.vdjxml'
locus_options = ' '.join([' --locus %s' % locus for locus
in loci.split()])

# PIPELINE STARTS HERE

# 0. CONVERSION TO VDXML
# 1. SIZE

SELECTION
inhandle = open(input_fasta,'r')
outhandle =
open(join(analysis_dir,size_selected_file),'w')
# iterate through fasta entries
for

```



```

                                vdj_package.txt
(descr,seq) in seqtools.FastaIterator(inhandle,lambda d: d.split()[0]):
    # convert to

ImmuneChain
    chain = vdj.ImmuneChain(descr=descr,seq=seq)

    # size select
    if

len(chain) < options.min or len(chain) > options.max:
    continue

    print

>>outhandle, chain
inhandle.close()
outhandle.close()

# 2. SPLIT INTO PARTS
inhandle =

open(join(analysis_dir,size_selected_file),'r')
parts = vdj.pipeline.iterator2parts(

vdj.parse_VDJXML(inhandle),

join(analysis_dir,parts_dir,size_selected_file),

packetsize,

                                prefix='<root>',

                                suffix='</root>')

cmd = 'barcode_id --barcodes %s ' % barcode_file           # 3.
BARCODE IDENTIFICATION
cmd += ' | coding_strand' + locus_options                   # 4. CODING
STRAND
if 'IGH' in loci:                                           # 5. ISOTYPE ID (heavy chain
only)
    cmd += ' | isotype_id --IGHC %s' % isotype_file
cmd += ' | align_vdj' +

locus_options                                               # 6. VDJ CLASSIFICATION

# submit cmd to LSF for each part
jobIDs = []
logfiles = []
outnames = []
for part in parts:
    partID = part.split('.')[0]

    partoutname = basename+'.prealign.vdjxml.'+partID
    outnames.append(partoutname)

```


#####

vdj_package.txt

```
#####
#####
```

This is analysis

of 454 data generated from the second vaccination experiment
from the second individual. I

believe this data comes from Ido.

The raw data is located at

vdj-ome/raw-seq-data/454/raw_20100927_flu2/
with the relevant files

RD_analysis/*GAC*.fna

I am concatenating these files into a single FASTA file located here

vdj-ome/stable-data/raw_flu2_20100927/
using this command:
cat *GAC*.fna >

~/research/church/vdj-ome/stable-data/raw_flu2_20100927/heavy_chains.flu2.20100927.f
asta

from the directory

vdj-ome/raw-seq-data/454/raw_20100927_flu2/RD_analysis/
The file is

called:

heavy_chains.flu2.20100927.fasta

The initial read length histograms are

generated in this script:

runstats.py

The output is:

Number of reads: 912388

Shortest read: 40 bp

Longest read: 600 bp

along with two figures which show a

very tight peak at ~455 bp.

The figure suggests size cutoffs of 395 to 520 bp.

1.

fasta2vdjxml.py

2. size_select.py

First I convert the fasta file to vdjxml, and,

size select the reads based on the readlen hist, 395-520,

vdj_package.txt

```
(from
/home/u12/vdj-ome/stable-data/raw_flu2_20100927):
python ~/code/vdj/bin/fasta2vdjxml.py
heavy_chains.flu2.20100927.fasta | python ~/code/vdj/bin/size_select.py --min 395
--max 520
> heavy_chains.flu2.20100927.size395-520.vdjxml
    There are 902805 chains of the selected
size in the file.
3. vdjxml2parts.py
    Split vdjxml into small chunks and place in
working directory
    mkdir ~/vdj-ome/analysis/flu2_seq_pipeline/data
    python
~/code/vdj/bin/vdjxml2parts.py --packetsize 10000 --basename
~/vdj-ome/analysis/flu2_seq_pipeline/data/heavy_chains.flu2.20100927.size395-520.vdj
xml
heavy_chains.flu2.20100927.size395-520.vdjxml
    Change directory to all the parts:
    cd
~/vdj-ome/analysis/flu2_seq_pipeline/data
4. barcode_id.py
5. coding_strand.py

Identify barcodes for each read, and
    Determine whether we have the correct strand or
not
    Ensure the barcode fasta file is correctly referenced.
    for FILE in
heavy_chains.flu2.20100927.size395-520.vdjxml.*; do

NAME=${FILE%.size*}".prealign.vdjxml"."${FILE#*.vdjxml}.*"
    # echo $NAME
    bsub

-qshared_2h -o pre-alignment.log "python ~/code/vdj/bin/barcode_id.py --barcodes
~/vdj-ome/stable-data/barcodes/IDT.454.rapid.MIDs.fasta $FILE | python
~/code/vdj/bin/coding_strand.py --locus IGH > $NAME"
```



```

done
                                vdj_package.txt

# python
~/code/vdj/bin/barcode_id.py --barcodes
~/vdj-ome/stable-data/barcodes/IDT.454.rapid.MIDs.fasta
# python
~/code/vdj/bin/coding_strand.py --locus IGH

Some STATS:
# Num of chains
cat


```
prealign | grep "<ImmuneChain>" | wc -l
902805

Num with barcodes
cat


```
*prealign* | grep "<barcode>" | wc -l
882978

# Num that were
reverse-complemented
cat *prealign* | grep "revcomp" | wc -l
434034

#
Barcode breakdown
for NUM in 13 14 15 16 17 18 19 20 21 22; do
    cat *prealign* |
    grep "<barcode>RL0$NUM" | wc -l
done
    RL013    18255    2.1%
    RL014    96568
10.9%
    RL015    88732    10.0%
    RL016    80088    9.1%
    RL017    79637
9.0%
    RL018    97962    11.1%
    RL019    223744    25.3%
    RL020    60479
6.8%
    RL021    73613    8.3%
    RL022    63900    7.2%

6. align_vdj.py
for
FILE in heavy_chains.flu2.20100927.prealign.vdjxml.*; do

```


```


```



```

                                vdj_package.txt
NAME=${FILE%.prealign*}".vdjxml."${FILE#*.vdjxml.*}
# echo $NAME
bsub

-qshared_12h -o alignment.log python ~/code/vdj/bin/align_vdj.py --locus IGH $FILE
$NAME

done

7. cat_vdjxml.py
    # concat data and dump into parent dir
    python
~/code/vdj/bin/cat_vdjxml.py heavy_chains.flu2.20100927.vdjxml.* >
../heavy_chains.flu2.20100927.aligned.vdjxml
    cd ..

8. filter_VJ.py
    python
~/code/vdj/bin/filter_VJ.py heavy_chains.flu2.20100927.aligned.vdjxml
heavy_chains.flu2.20100927.VJ_filtered.vdjxml
    grep "<ImmuneChain>"
heavy_chains.flu2.20100927.VJ_filtered.vdjxml | wc -l
749069

    for NUM
in 13 14 15 16 17 18 19 20 21 22; do
    grep "<barcode>RL0$NUM"
heavy_chains.flu2.20100927.VJ_filtered.vdjxml | wc -l
done
    RL013    12809
1.7%
    RL014    84053    11.4%
    RL015    77583    10.6%
    RL016    70310
9.6%
    RL017    69649    9.5%
    RL018    85832    11.7%
    RL019    172655
23.5%
    RL020    47802    6.5%
    RL021    62478    8.5%
    RL022    50969
6.9%

9. partition_VJ.py

```


vdj_package.txt

```

mkdir partitions
python ~/code/vdj/bin/partition_VJ.py
--basename partitions/heavy_chains.flu2.20100927
heavy_chains.flu2.20100927.VJ_filtered.vdjxml
cd partitions
# How many chains in
each partition?
for FILE in heavy_chains.flu2.20100927.*.vdjxml; do
    grep
"<ImmuneChain>" $FILE | wc -l
done | sort -n

    partial results:
    ...

14270
    14383
    16067
    16215
    16817
    22472
    31988

51680

10. cluster_cdr3.py
    for INFILE in heavy_chains.flu2.20100927.*.vdjxml;
do
    VJID=${INFILE#heavy_chains.flu2.20100927.}
    VJID=${VJID%.vdjxml}

    OUTFILE=${INFILE%.vdjxml}.clustered.vdjxml
    # echo $VJID $OUTFILE
    bsub

-qshared_unlimited -o clustering.log python ~/code/vdj/bin/cluster_cdr3.py --cutoff
4.5
--tag $VJID --linkage single $INFILE $OUTFILE
done

```

```
#####
```

```
#####
```

```
#####
```


vdj_package.txt

#####

#####

How

long did it take?

```
grep "CPU" clustering.log | sort -n -k4
partial results:
```

...

```
CPU time : 2476.57 sec.
CPU time : 2537.17 sec.
CPU
```

```
time : 2897.51 sec.
CPU time : 2958.58 sec.
CPU time : 3086.20
```

```
sec.
CPU time : 4735.64 sec.
CPU time : 5885.87 sec.
CPU
```

```
time : 6538.90 sec.
CPU time : 7931.02 sec.
CPU time : 16996.05
```

```
sec.
CPU time : 34048.12 sec.
```

11. cat_vdjxml.py

python

~/code/vdj/bin/cat_vdjxml.py heavy_chains.GMC.20100907.*.clustered.vdjxml >

../heavy_chains.GMC.20100907.clustered.vdjxml

cd ..

```
# How many chains?
grep
```

```
"<ImmuneChain>" heavy_chains.GMC.20100907.clustered.vdjxml | wc -l
793698
```

How

many unique clones in total?

```
grep "<clone>" heavy_chains.GMC.20100907.clustered.vdjxml |
```

```
sort | uniq | wc -l
401970
```

```
# How many unique junctions in total?
grep
```


465771

#####

```
import subprocess
import sys
import optparse
import tempfile
import os

import vdj

parser =

optparse.OptionParser()
parser.add_option('-q', '--queue')
parser.add_option('-o', '--LSFoutput')
parser.add_option('-p', '--packetsize', type='int')
```



```

                                vdj_package.txt
parser.add_option('-b', '--barcodes', dest='barcodes_fasta')
parser.add_option('-i', '--IGHC', dest='ighc_fasta')
parser.add_option('-m', '--min', type='int')
parser.add_option('-M', '--max', type='int')
(options, args) = parser.parse_args()

if len(args) == 2:
    inhandle = open(args[0], 'r')

    outhandle = open(args[1], 'w')
else:
    raise Exception, "Must have an input file and
output file."

# temporary directory to dump intermediate files
tempdir =

tempfile.mkdtemp(prefix=args[1]+'.intermediate.', dir='.')
os.chdir(tempdir)

cmd1 =
'fasta2vdjxml.py'
cmd2 = 'size_select.py --min %d --max %d' % (options.min, options.max)
cmd3

= 'vdjxml2parts.py --packetsize %d --basename %s' % (options.packetsize, args[1])
cmd = ' |

'.join([cmd1, cmd2, cmd3])
p =

subprocess.Popen(cmd, shell=True, stdin=inhandle, stdout=subprocess.PIPE)
parts = [f.strip()

for f in p.stdout.readlines()]
outparts = [part+'.out' for part in parts]

cmd4 = r'"cat

%s'
cmd5 = 'barcode_id.py --barcodes %s' % (options.barcodes_fasta)
cmd6 =

'positive_strand.py'
cmd7 = 'isotype_id.py --IGHC %s' % (options.ighc_fasta)
cmd8 =

r'align_vdj.py > %s"'
cmd = ' | '.join([cmd4, cmd5, cmd6, cmd7, cmd8])
jobs = []
for

part, outpart in zip(parts, outparts):
    jobID =

vdj.LSF.submit_to_LSF(options.queue, options.LSFoutput, cmd % (part, outpart))

jobs.append(jobID)
vdj.LSF.wait_for_LSF_jobs(jobs)

```


vdj_package.txt

```

file_list = ' '.join(outparts)
subprocess.Popen('cat ' + file_list, shell=True, stdout=outhandle)
#! /usr/bin/env python

```

```

import sys
import optparse

```

```

import numpy as np

```

```

import vdj

```

```

parser =

```

```

optparse.OptionParser()
(options, args) = parser.parse_args()

```

```

if len(args) == 2:

```

```

    inhandle = open(args[0], 'r')
    outhandle = open(args[1], 'w')
elif len(args) == 1:

```

```

    inhandle = open(args[0], 'r')
    outhandle = sys.stdout
elif len(args) == 0:
    inhandle =

```

```

    sys.stdin
    outhandle = sys.stdout

```

```

counts_dict = vdj.get_clone_counts(inhandle)
counts =

```

```

vdj.count_dict_clone_counts(counts_dict)

```

```

for count in counts:
    print >>outhandle,

```

```

np.int_(count)
#! /usr/bin/env python

```

```

import sys
import optparse

```

```

import vdj

```

```

parser =

```

```

optparse.OptionParser()
(options, args) = parser.parse_args()

```

```

if len(args) == 2:

```

```

    inhandle = open(args[0], 'r')
    outhandle = open(args[1], 'w')
elif len(args) == 1:

```

```

    inhandle = open(args[0], 'r')

```



```

                                vdj_package.txt
        outhandle = sys.stdout
    elif len(args) == 0:
        inhandle =

    sys.stdin
        outhandle = sys.stdout

    vdj.vdjxml2fasta(inhandle,outhandle)
    #! /usr/bin/env

    python

    import sys
    import optparse

    import vdj
    import vdj.pipeline

    parser =

    optparse.OptionParser()
    parser.add_option('-b','--basename')
    parser.add_option('-p','--packetsize',type='int')
    (options, args) = parser.parse_args()

    if

    len(args) == 1:
        inhandle = open(args[0],'r')
    elif len(args) == 0:
        inhandle =

    sys.stdin
    else:
        raise Exception, "Too many arguments."

    parts =

    vdj.pipeline.iterator2parts( vdj.parse_VDJXML(inhandle),

        options.basename,

                                options.packetsize,

                                prefix='<root>',

                                suffix='</root>')

    for part in parts:
        print part
    #include <Python.h>
    #include

    <numpy/arrayobject.h>
    #include <stdlib.h>

    // DEBUG
    // #include <stdio.h>

    int arrayargmax(

```



```

                                vdj_package.txt
double *data, int n ) {
    int maxidx, curidx ;
    for ( maxidx = 0, curidx = 1 ; curidx <
n ; curidx++ ) {
        if ( data[curidx] > data[maxidx] ) {
            maxidx = curidx ;
        }
    }
    return maxidx ;
}

void alignNW( PyArrayObject *M, PyArrayObject *IX,
PyArrayObject *Iy, PyArrayObject *BT,
               const char *seq1, const int len1, const
char *seq2, const int len2) {
    // define parameters
    double match      = 0.5 ;

    double mismatch   = -0.75 ;
    double gapopen     = -2.0 ;
    double gapextend   = -1.5 ;

    int nrows = len1 + 1 ;
    int ncols = len2 + 1 ;
    int i, j ;

    double
*Mij, *Mim1jm1, *Mim1j, *Mijm1 ;
    double *Ixiij, *Ixim1jm1, *Ixim1j ;
    double *Iyij,
*Iyim1jm1, *Iyijm1 ;
    int *BTij ;

    double s ;
    double ext[3] ;
    double

IXGO, IXGE, IyGO, IyGE ;
    int best ;

    for ( i = 1 ; i < nrows ; i++ ) {

for ( j = 1 ; j < ncols ; j++ ) {
    Mij =
(double*)PyArray_GETPTR2(M, i,j) ;
    Ixiij = (double*)PyArray_GETPTR2(IX,i,j) ;

    Iyij = (double*)PyArray_GETPTR2(Iy,i,j) ;

```



```

                                vdj_package.txt
        Mim1jm1 =
(double*)PyArray_GETPTR2(M, i-1,j-1) ;
        Ixim1jm1 =
(double*)PyArray_GETPTR2(Ix,i-1,j-1) ;
        Iyim1jm1 =
(double*)PyArray_GETPTR2(Iy,i-1,j-1) ;
        Mim1j = (double*)PyArray_GETPTR2(M,i-1,j)
;
        Mijm1 = (double*)PyArray_GETPTR2(M,i,j-1) ;
        Ixim1j =
(double*)PyArray_GETPTR2(Ix,i-1,j) ;
        Iyijm1 = (double*)PyArray_GETPTR2(Iy,i,j-1)
;
        BTij = (int*)PyArray_GETPTR2(BT,i,j) ;
        s = (seq1[i-1]
== seq2[j-1]) ? match : mismatch ;
        ext[0] = *Mim1jm1 + s ;
        ext[1] =
        *Ixim1jm1 + s ;
        ext[2] = *Iyim1jm1 + s ;
        IXGO = *Mim1j + gapopen
;
        IXGE = *Ixim1j + gapextend ;
        IyGO = *Mijm1 + gapopen ;

        IyGE = *Iyijm1 + gapextend ;
        best = arrayargmax(ext,3) ;

        *Mij = ext[best] ;
        *IXij = (IXGO >= IXGE) ? IXGO : IXGE ;

        *Iyij = (IyGO >= IyGE) ? IyGO : IyGE ;
        *BTij = best ; // 0 =
(i-1,j-1) ; 1 = (i-1,j) ; 2 = (i,j-1)
        // DEBUG

//printf("char1: %c\tchar2: %c\tmatch? %d\tts: %f\tM: %f\tIx: %f\tBT:
%i\n",seq1[i-1],seq2[j-1],seq1[i-1]==seq2[j-1],s,*Mij,*IXij,*BTij) ;
    }
}

return ;

```



```

                                vdj_package.txt
}

void alignSW( PyArrayObject *F, PyArrayObject *BT,
              const char
*seq1, const int len1, const char *seq2, const int len2) {
    // define parameters

    double match      = 0.5 ;
    double mismatch   = -0.75 ;
    double gapextend  = -1.5 ;

    int nrows = len1 + 1 ;
    int ncols = len2 + 1 ;
    int i, j ;

    double *Fij,
    *Fim1jm1, *Fim1j, *Fijm1 ;
    int *BTij ;

    double s ;
    double ext[4] ;
    int
best ;

    for ( i = 1 ; i < nrows ; i++ ) {
        for ( j = 1 ; j < ncols ; j++ ) {

            Fij      = (double*)PyArray_GETPTR2(F,i,j) ;
            Fim1jm1 =
            (double*)PyArray_GETPTR2(F,i-1,j-1) ;
            Fim1j    =
            (double*)PyArray_GETPTR2(F,i-1,j) ;
            Fijm1    = (double*)PyArray_GETPTR2(F,i,j-1)
            ;
            BTij     = (int*)PyArray_GETPTR2(BT,i,j) ;
            s =
            (seq1[i-1] == seq2[j-1]) ? match : mismatch ;
            ext[0] = *Fim1jm1 + s ;

            ext[1] = *Fim1j + gapextend ;
            ext[2] = *Fijm1 + gapextend ;
            ext[3]
            = 0 ;

            best = arrayargmax(ext,4) ;
            *Fij =
            ext[best] ;

```



```

                                vdj_package.txt
        *BTij = best ; // 0 = (i-1,j-1) ; 1 = (i-1,j) ; 2 = (i,j-1) ; 3 =
END (0)

        // DEBUG
        //printf("char1: %c\tchar2: %c\tmatch?
%d\tts: %f\tF: %f\tBT: %i\n",seq1[i-1],seq2[j-1],seq1[i-1]==seq2[j-1],s,*Fij,*BTij) ;

    }
    }
    return ;
}

static PyObject *alignmentcore_alignNW( PyObject *self, PyObject
*args ) {
    char *seq1, *seq2 ;
    int len1, len2 ;
    PyArrayObject *M, *Ix, *Iy, *BT
;

    // get sequence args
    if ( !PyArg_ParseTuple(args,"0000s#s#",

        &M,

                                &Ix,
                                &Iy,

        &BT,

                                &seq1, &len1,

                                &seq2, &len2) ) {
        return NULL ;
    }

    // call function
    alignNW( M, Ix,
Iy, BT, seq1, len1, seq2, len2 ) ;
    return Py_BuildValue( "d", 0.0 ) ;
}

// wrapper
functions

static PyObject *alignmentcore_alignSW( PyObject *self, PyObject *args ) {

char *seq1, *seq2 ;
    int len1, len2 ;
    PyArrayObject *F, *BT ;

    // get
sequence args

```



```

                                vdj_package.txt
if ( !PyArg_ParseTuple(args, "OOs#s#",
                                &F,

                                &BT,
                                &seq1, &len1,

                                &seq2, &len2) ) {
    return NULL ;
}

// call function
alignSW( F,
BT, seq1, len1, seq2, len2 ) ;
    return Py_BuildValue( "d", 0.0 ) ;
}

static
PyMethodDef alignmentcoremethods[] = {
    {"alignNW", alignmentcore_alignNW,
METH_VARARGS},
    {"alignSW", alignmentcore_alignSW, METH_VARARGS}
} ;

void
initalignmentcore() {
    Py_InitModule( "alignmentcore", alignmentcoremethods ) ;

import_array();
}
// clusteringcore.c
// Defines clusteringcore python extension module

#include <Python.h>
#include <numpy/arrayobject.h>

// DEBUG
// #include <stdio.h>

int
intarraymin( int *data, int n ) {
    int i, minval ;
    for ( minval = data[0], i = 1 ; i
< n ; i++ ) {
        if ( data[i] < minval ) {
            minval = data[i] ;
        }
    }

    return minval ;
}

static PyObject *clusteringcore_levenshtein( PyObject *self,

```


vdj_package.txt

```

PyObject *args ){
    char *seq1, *seq2 ;
    int len1, len2 ;
    npy_intp dim[2] ;
    int

i, j ;
    int cost, best ;
    int ext[3] ;
    PyArrayObject *scores = NULL;

    //
get sequence args
    if ( !PyArg_ParseTuple(args,"s#s#",
                            &seq1,

&len1,
                            &seq2, &len2) ) {
        return NULL ;
    }
    //
check for trivial case
    if ( len1 == 0 || len2 == 0 ) {
        return Py_BuildValue(
"i", (len1 < len2 ? len2 : len1) ) ;
    }

    // allocate and initialize score matrix
F
    dim[0] = len1+1 ;
    dim[1] = len2+1 ;
    scores = (PyArrayObject *)PyArray_ZEROS(
2, dim, NPY_INT, 0 ) ;
    if (scores == NULL) return NULL ;

    for ( i = 0, j = 0 ; i
<= len1 ; i++ ) {
        *((int*)PyArray_GETPTR2(scores,i,j)) = i ;
    }
    for ( i = 0,
j = 0 ; j <= len2 ; j++ ) {
        *((int*)PyArray_GETPTR2(scores,i,j)) = j ;
    }

    // compute DP score matrix
    for ( i = 1 ; i <= len1 ; i++ ) {
        for ( j = 1 ; j <=
len2 ; j++ ) {
            cost = (seq1[i-1] == seq2[j-1]) ? 0 : 1 ;

```


vdj_package.txt

```

    ext[0] = *((int*)PyArray_GETPTR2(scores,i-1,j-1)) + cost ;
        ext[1] =
*((int*)PyArray_GETPTR2(scores,i-1,j))    + 1 ;
        ext[2] =
*((int*)PyArray_GETPTR2(scores,i,j-1))    + 1 ;

*((int*)PyArray_GETPTR2(scores,i,j)) = intarraymin(ext,3) ;
    }
    best =
*((int*)PyArray_GETPTR2(scores,len1,len2)) ;
    Py_DECREF(scores) ;
    return
Py_BuildValue( "i", best ) ;
}

static PyMethodDef clusteringcoremethods[] = {

{"levenshtein", clusteringcore_levenshtein, METH_VARARGS}
} ;

PyMODINIT_FUNC
initclusteringcore() {
    Py_InitModule( "clusteringcore", clusteringcoremethods ) ;

import_array() ;
}

```


CLAIMS

1. A method for identifying a therapeutic antibody, comprising:
 - (a) linking polynucleotide sequences encoding a V_H and a V_L isolated from a single cell to form paired sequences;
 - (b) sequencing polynucleotides amplified from the paired sequences;
 - (c) determining the frequency of two or more paired V_H and V_L sequences;
 - (d) determining a variance from a reference library of the paired sequences from (c); and
 - (e) identifying a therapeutic antibody based on (c) and (d).
2. The method of claim 1, further comprising screening the therapeutic antibody for functional specificity.
3. The method of claim 1, further comprising screening the therapeutic antibody for affinity.
4. The method of claim 1, further comprising screening the therapeutic antibody for neutralization ability.
5. The method of claim 1, further comprising cloning the therapeutic antibody directly into surface-display technology.
6. The method of claim 1, wherein the therapeutic antibody is a neutralizing antibody.
7. The method of claim 1, wherein the therapeutic antibody is identified without prior knowledge of an antigen to which it binds.

8. The method of claim 1, wherein the therapeutic antibody is a rapid response antibody.
9. The method of claim 1, further comprising determining the isotype of the therapeutic antibody.
10. The method of claim 1, further comprising isolating and purifying a therapeutic antibody identified in (e).
11. The method of claim 1, wherein the amplified polynucleotides are amplified by PCR with non-specific primers, degenerate primers, or specific primers.
12. The method of claim 1, wherein the amplified polynucleotides are amplified by PCR with specific primer sets that hybridize to the heavy and light chains of: B-cells, T-cells, or B-cells and T-cells.
13. The method of claim 1, wherein the amplified polynucleotides are amplified by PCR with two sets of primers, wherein the first set of primers hybridizes to V regions and the second set of primers hybridizes to C regions.
14. The method of claim 1, further comprising generating a database of paired V_H and V_L sequences.
15. The method of claim 14, wherein the identifying comprises aligning sequence data from the database against known or expected V, D and J segments in: a personal database, a NCBI database, or an IMGT database.
16. The method of claim 1, wherein the cell is from a plurality of B-cells.

17. The method of claim 1, wherein the therapeutic antibody is fully human.
18. The method of claim 1, wherein the amplified polynucleotides are DNA.
19. The method of claim 1, wherein the amplified polynucleotides comprise sequence adaptors compatible with next-generation high-throughput sequencing.
20. The method of claim 1, wherein the sequencing is next-generation high-throughput sequencing.
21. The method of claim 1, wherein the linked polynucleotide sequences encoding a V_H and a V_L are bar-coded, wherein bar-coding can be used to achieve remapping of the amplified polynucleotides to their original starting molecules.
22. The method of claim 1, wherein in (c), a coefficient of variation of about 0.5 occurred at clone frequencies of $<10^{-4}$ - 10^{-3} .
23. The method of claim 1, wherein determining a variance from a reference library of the paired sequences from (c) comprises aligning the paired sequences from (c) to the reference library sequences.
24. The method of claim 1, wherein in (c), a coefficient of variation approaches 1 at a sampling level of 10^5 .
25. The method of claim 1, further comprising clustering the paired sequences from (c) into unique clones.
26. The method of claim 4, further comprising identifying an antigen that interacts with the therapeutic antibody as a biomarker for a disease.

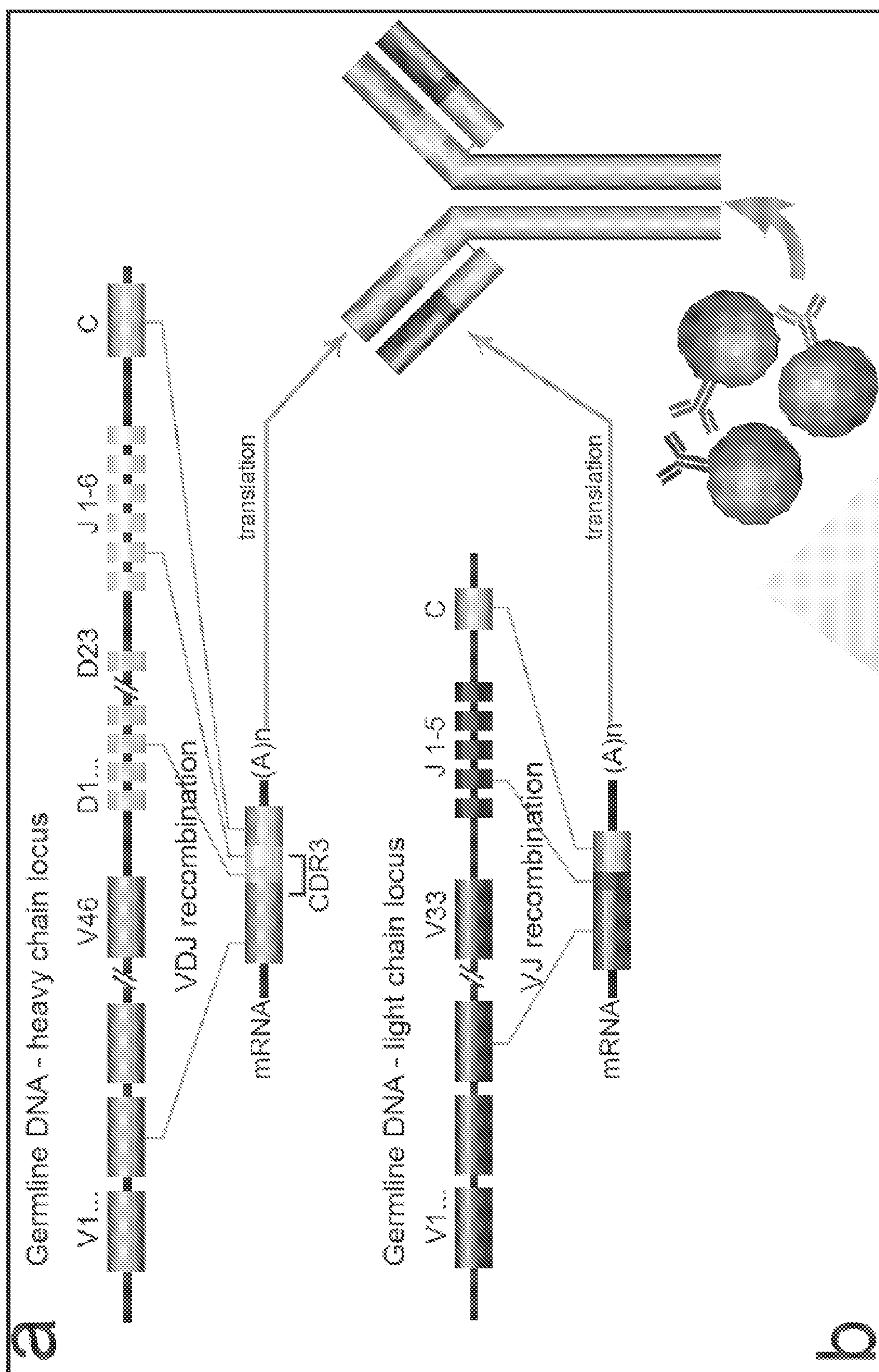
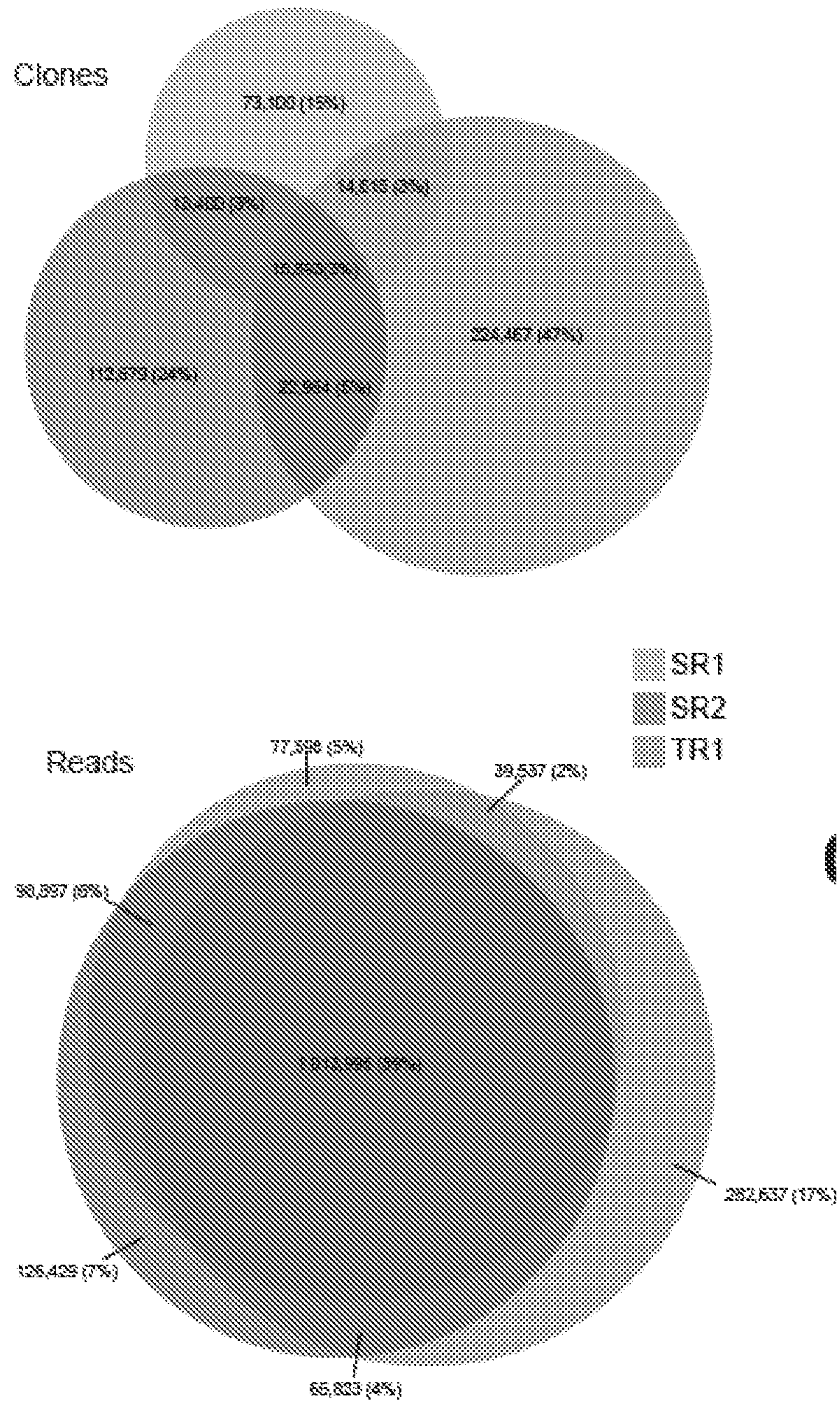
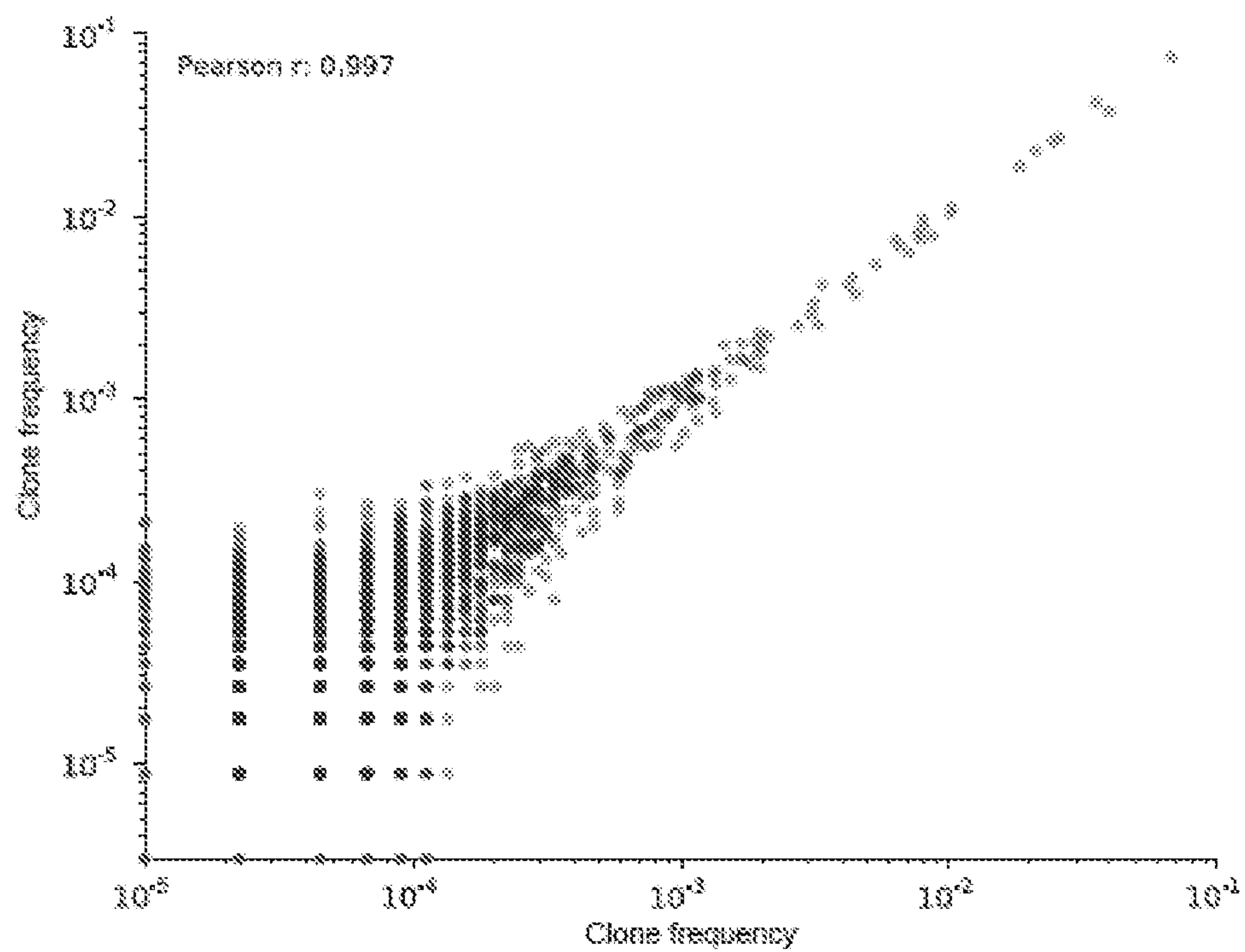
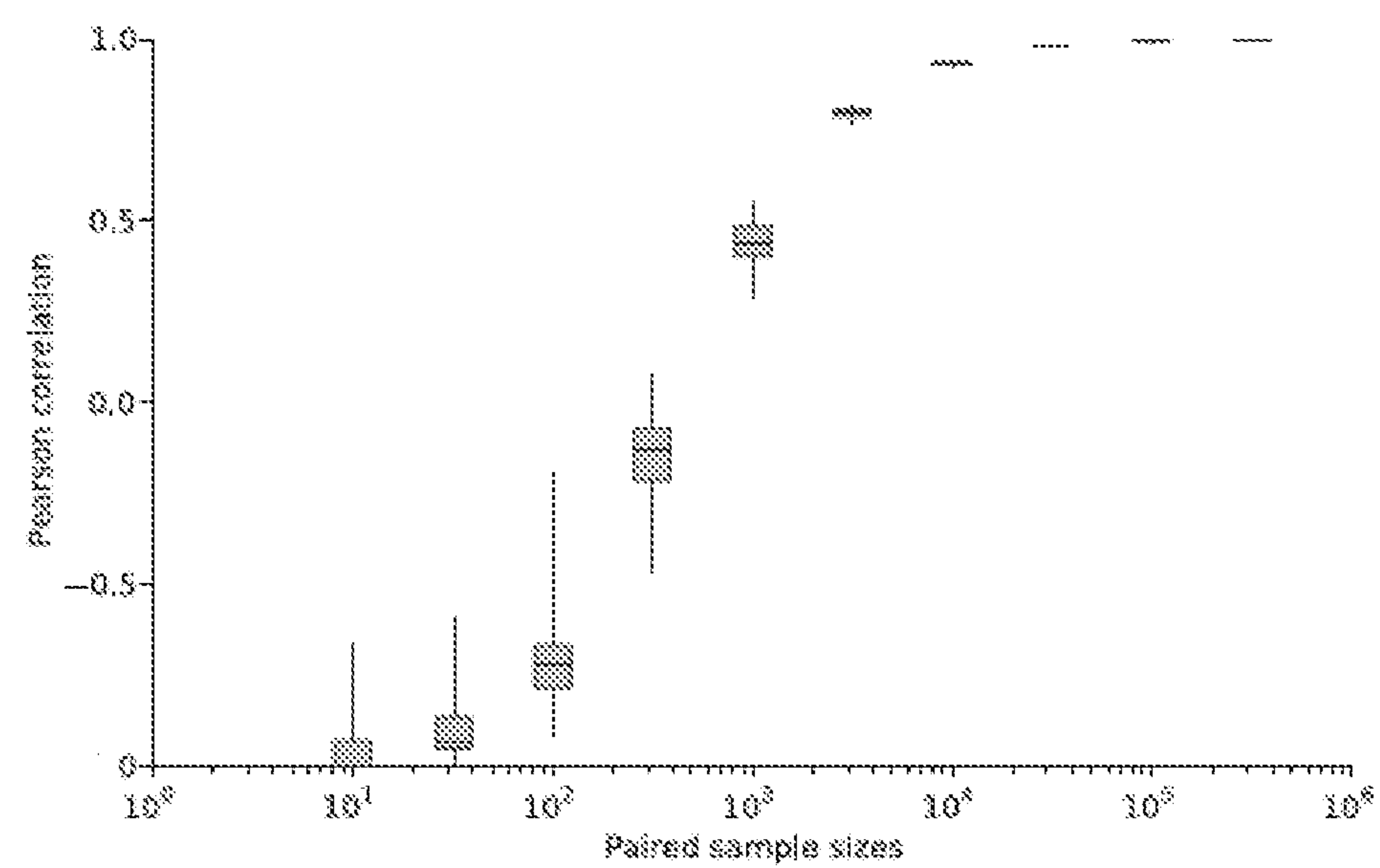


Figure 1

A**Figure 2**

B**C****Figure 2 (Cont.)**

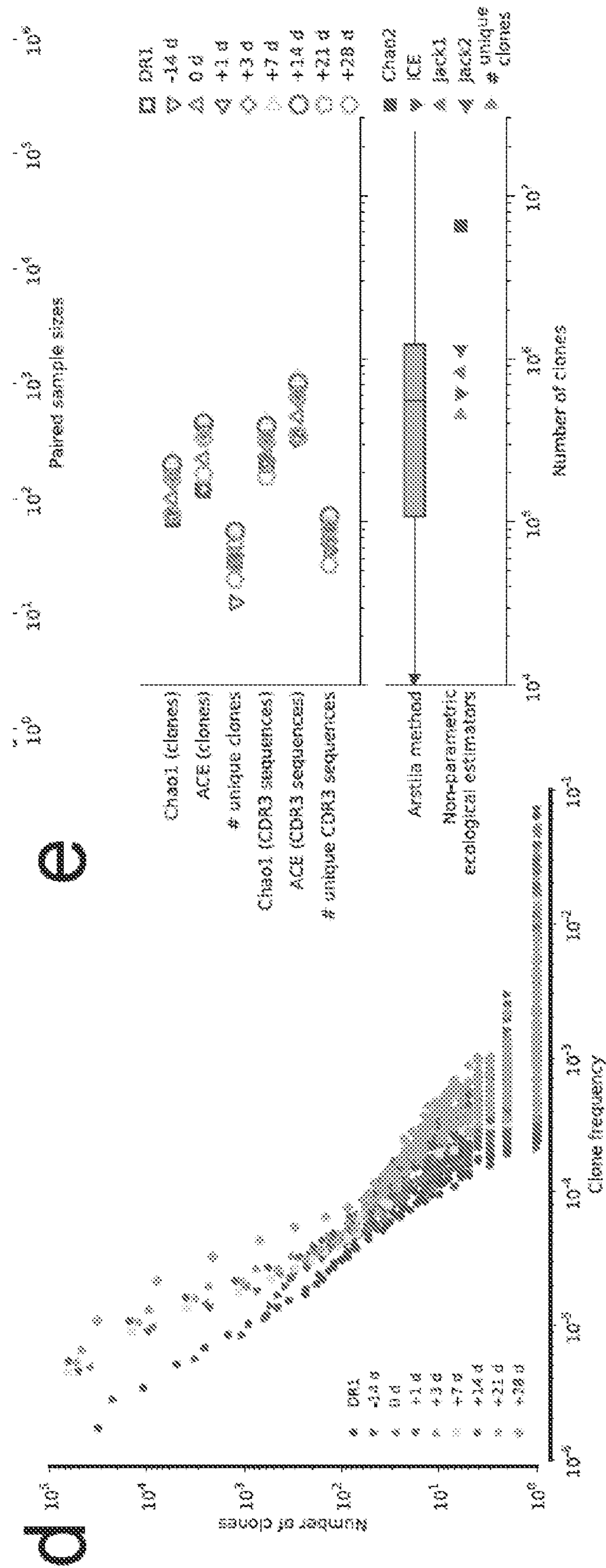


Figure 2 (Cont.)

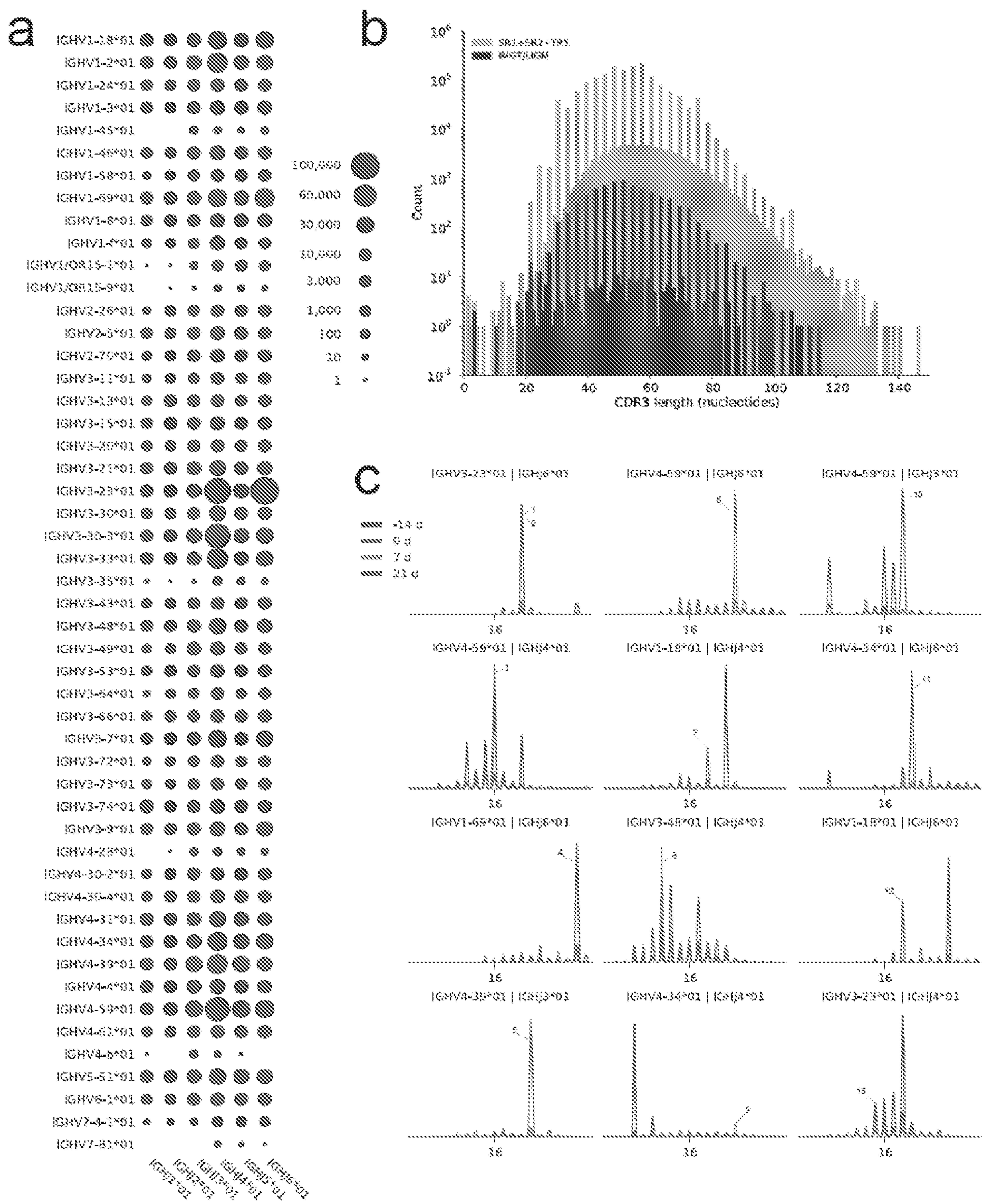


Figure 3

a

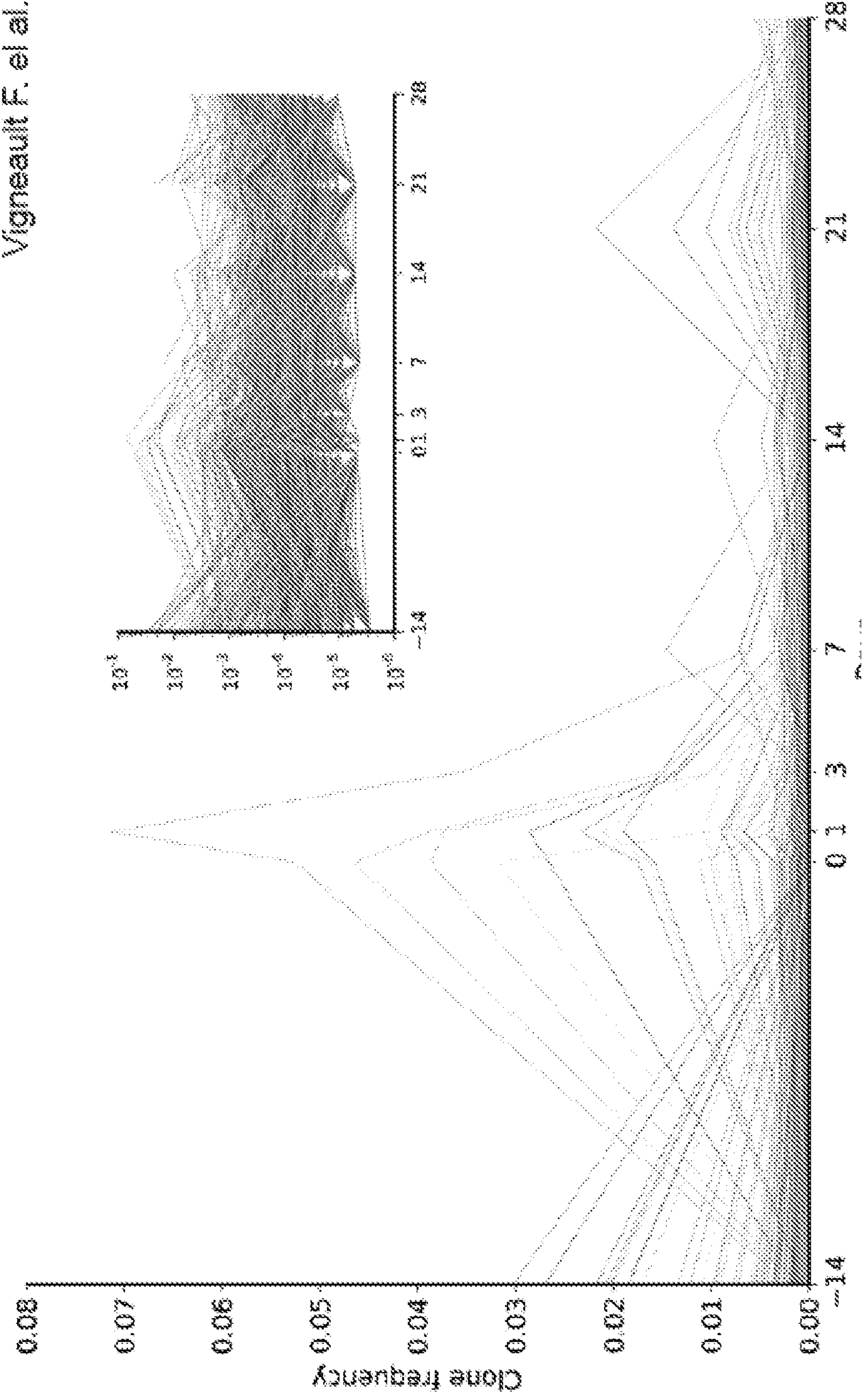


Figure 4

Vigneault F. et al. Figure 4

B

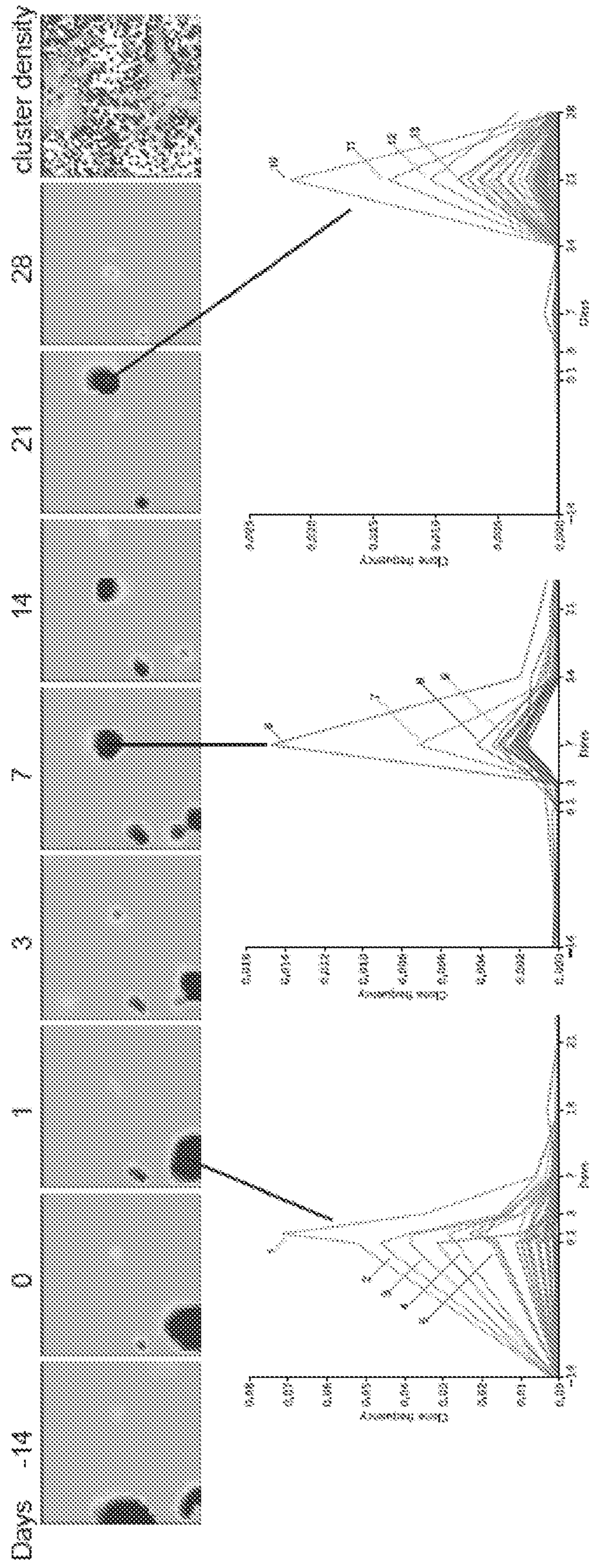


Figure 4 (Cont.)

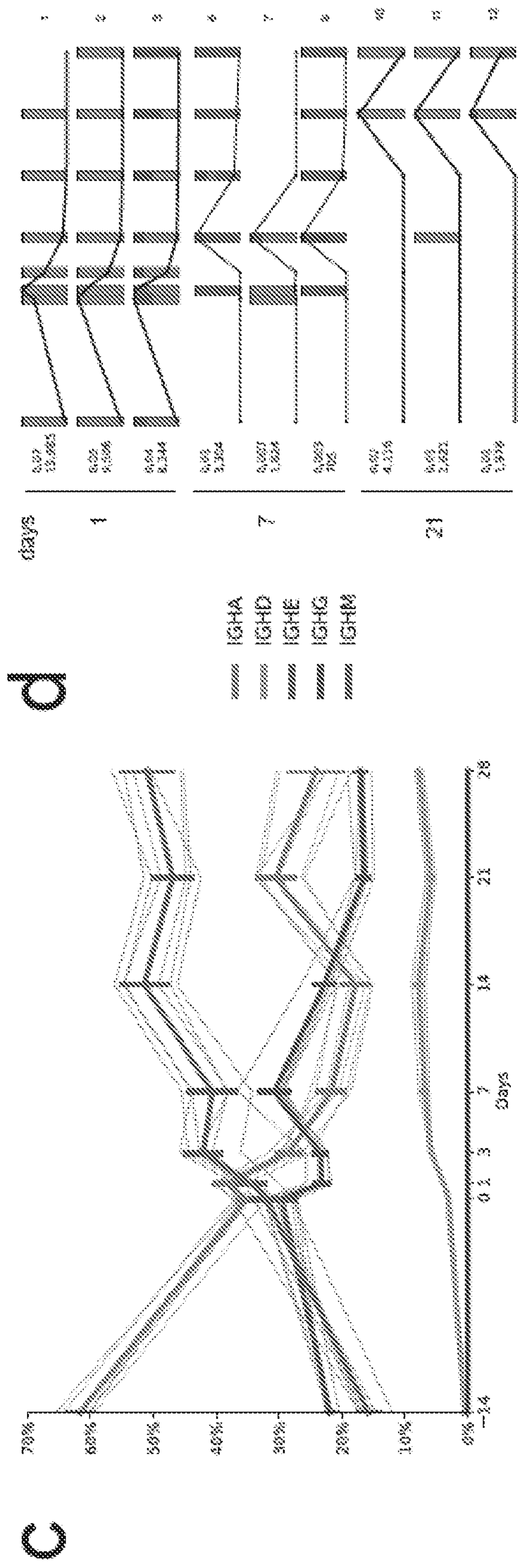
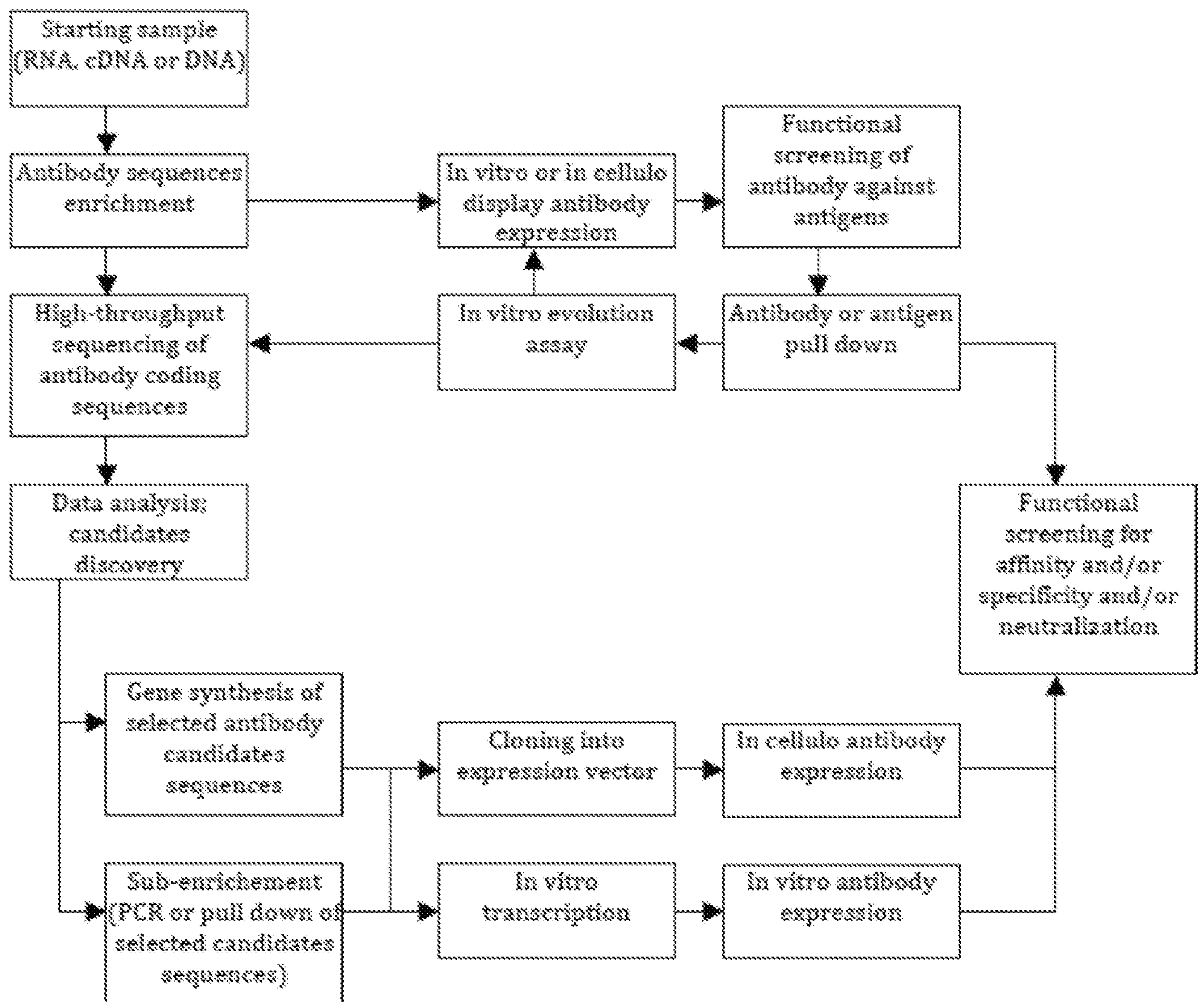


Figure 4 (Cont.)

**Figure 5**

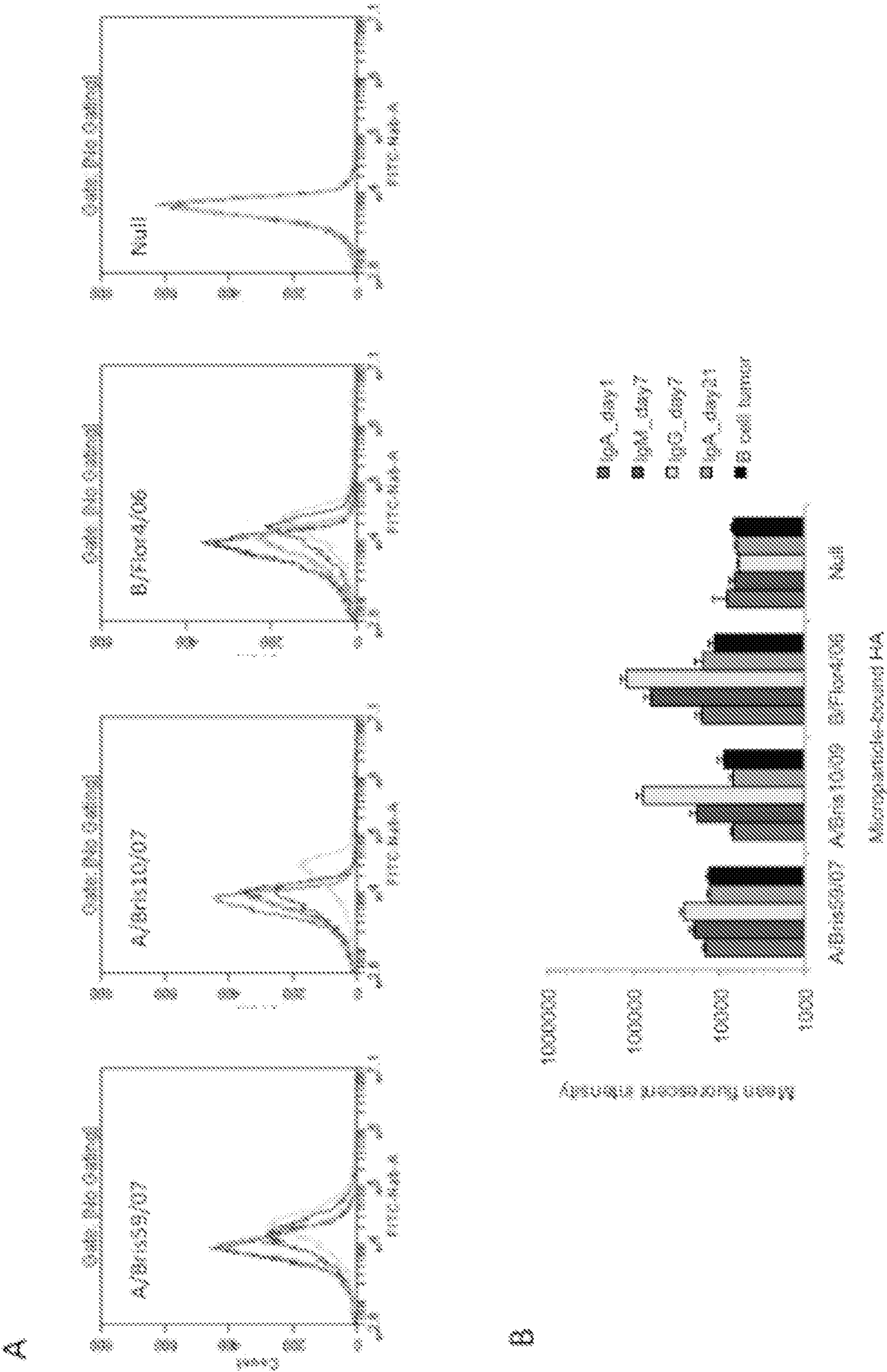


Figure 6

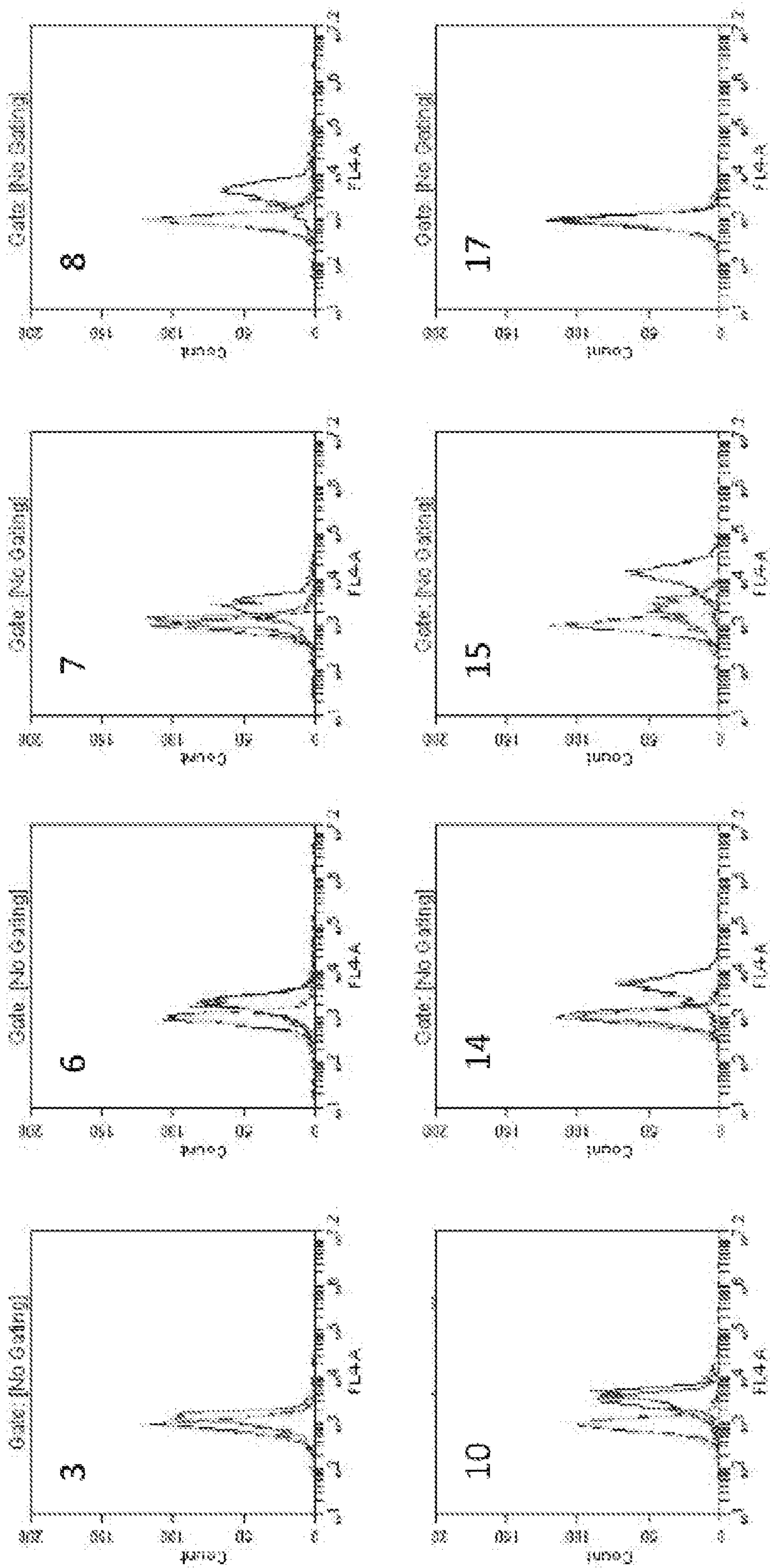


Figure 6 (Cont.)

A

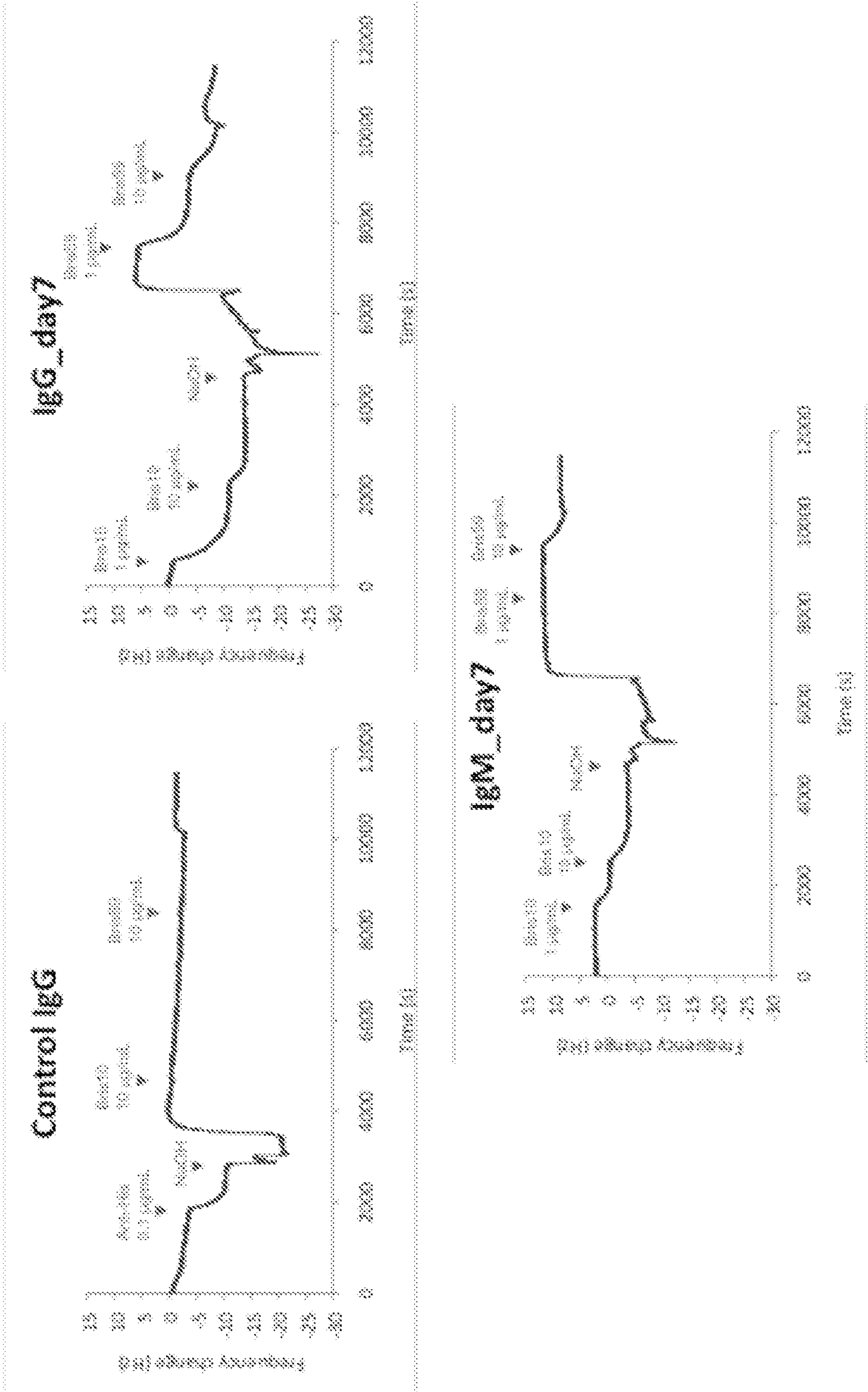


Figure 7

B

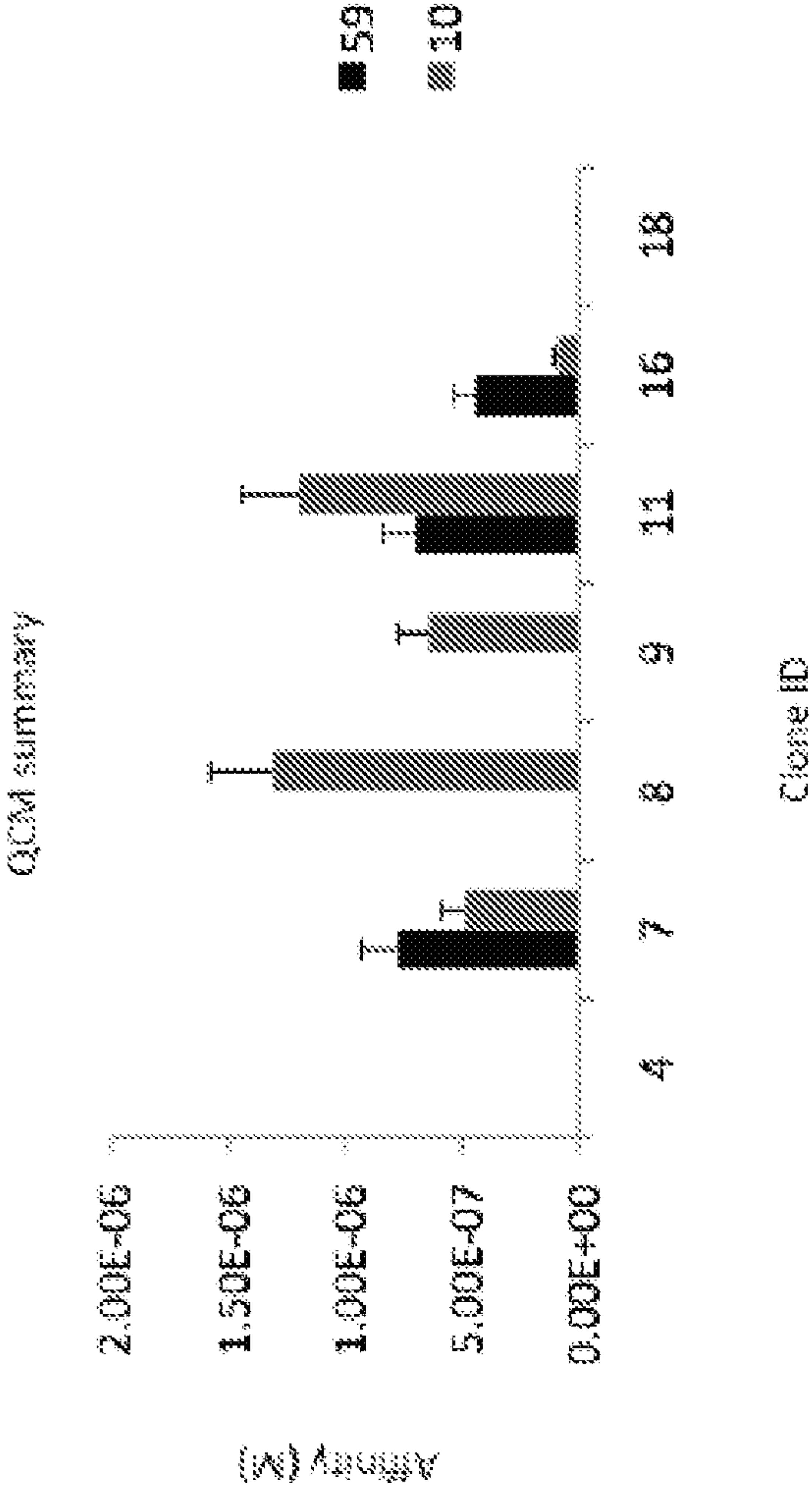


Figure 7 (Cont.)

C

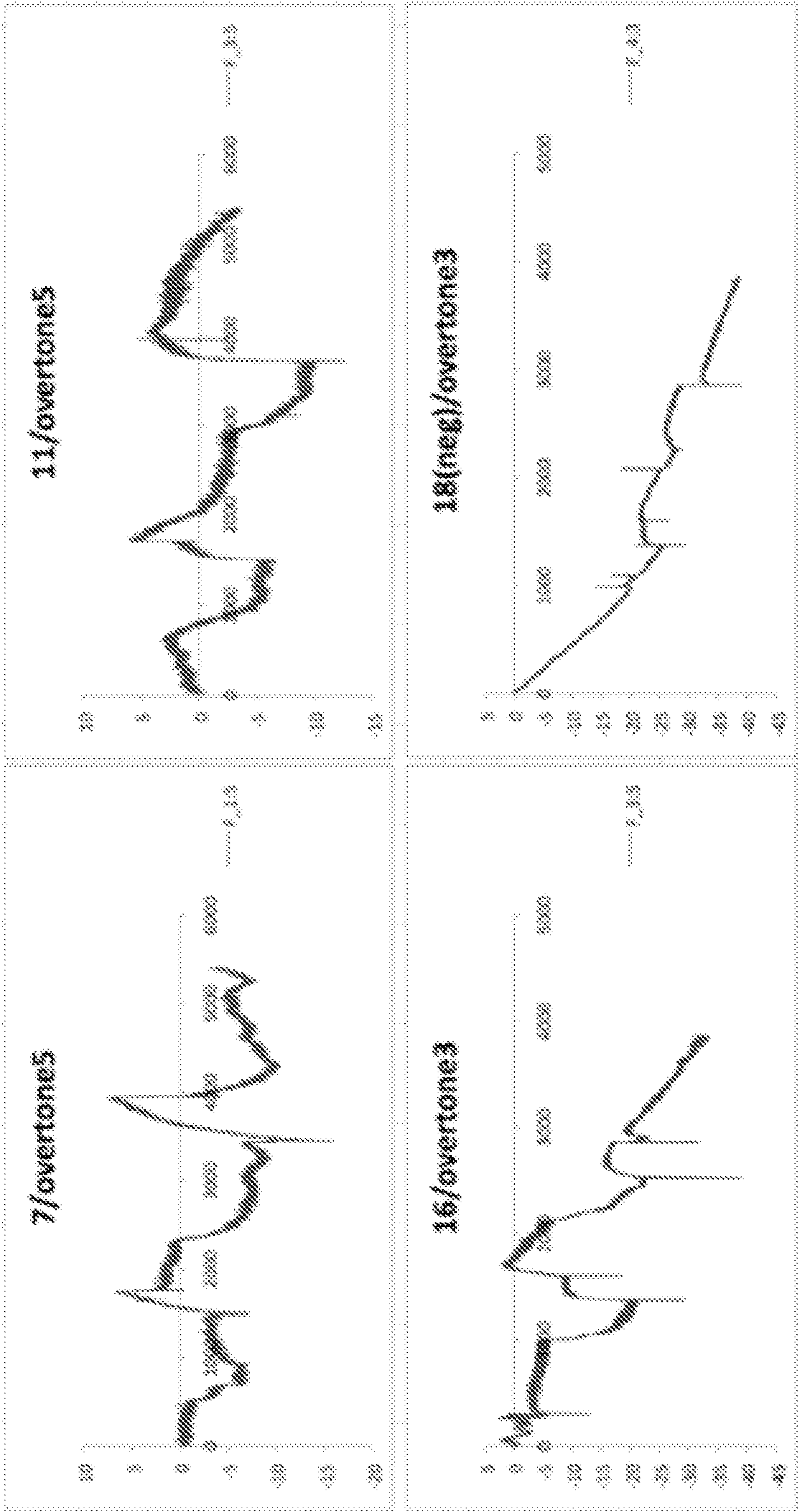
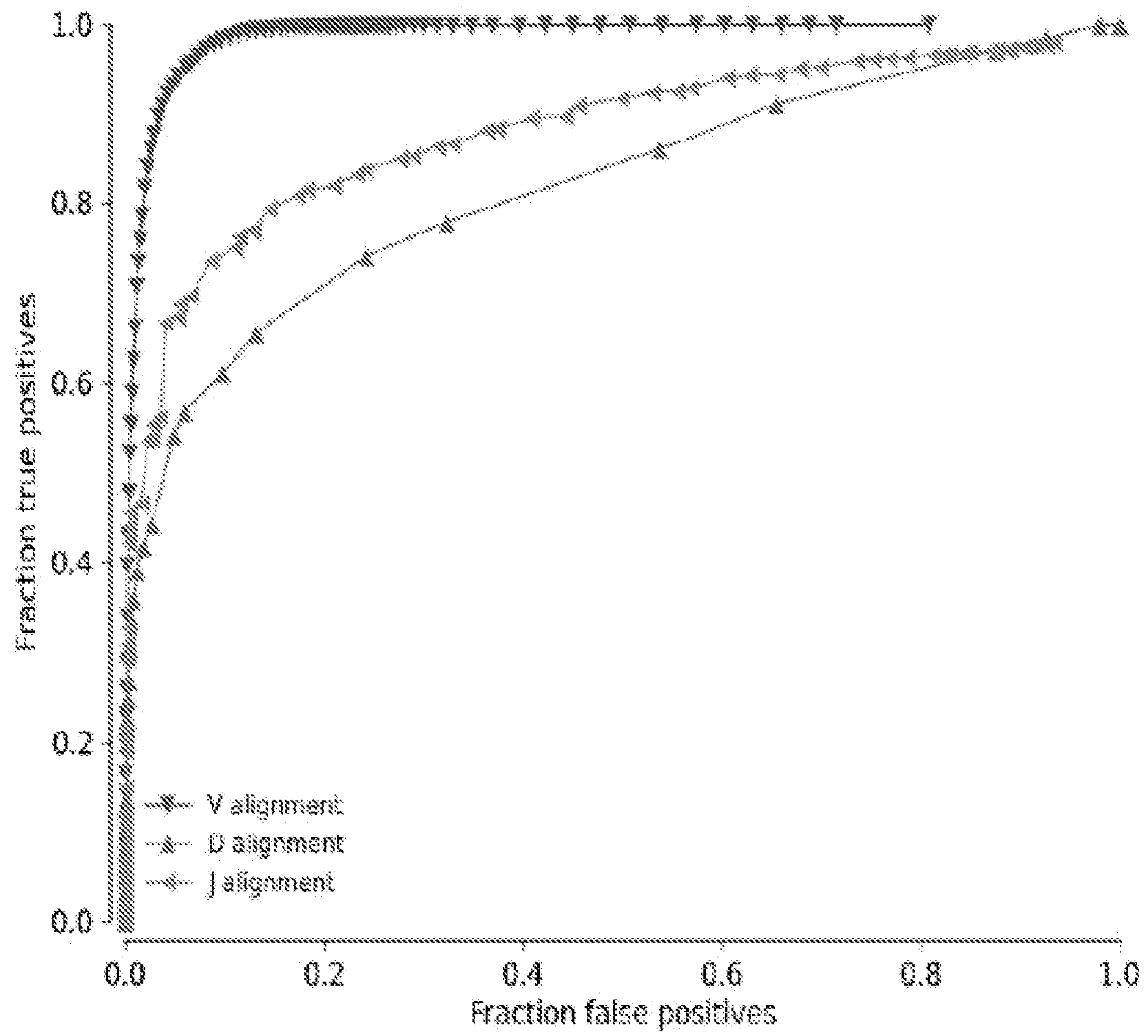


Figure 7 (Cont.)

**Figure 8**

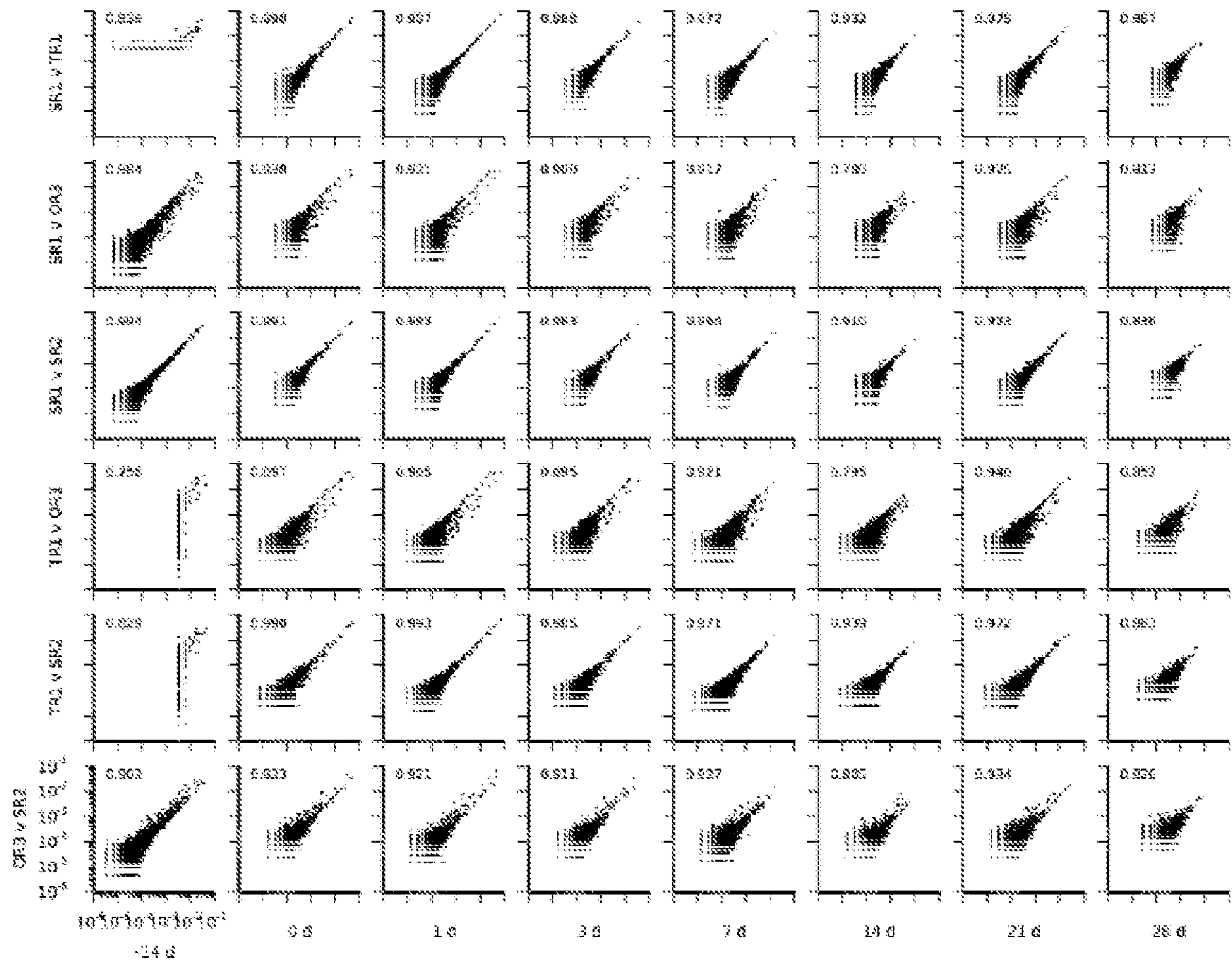
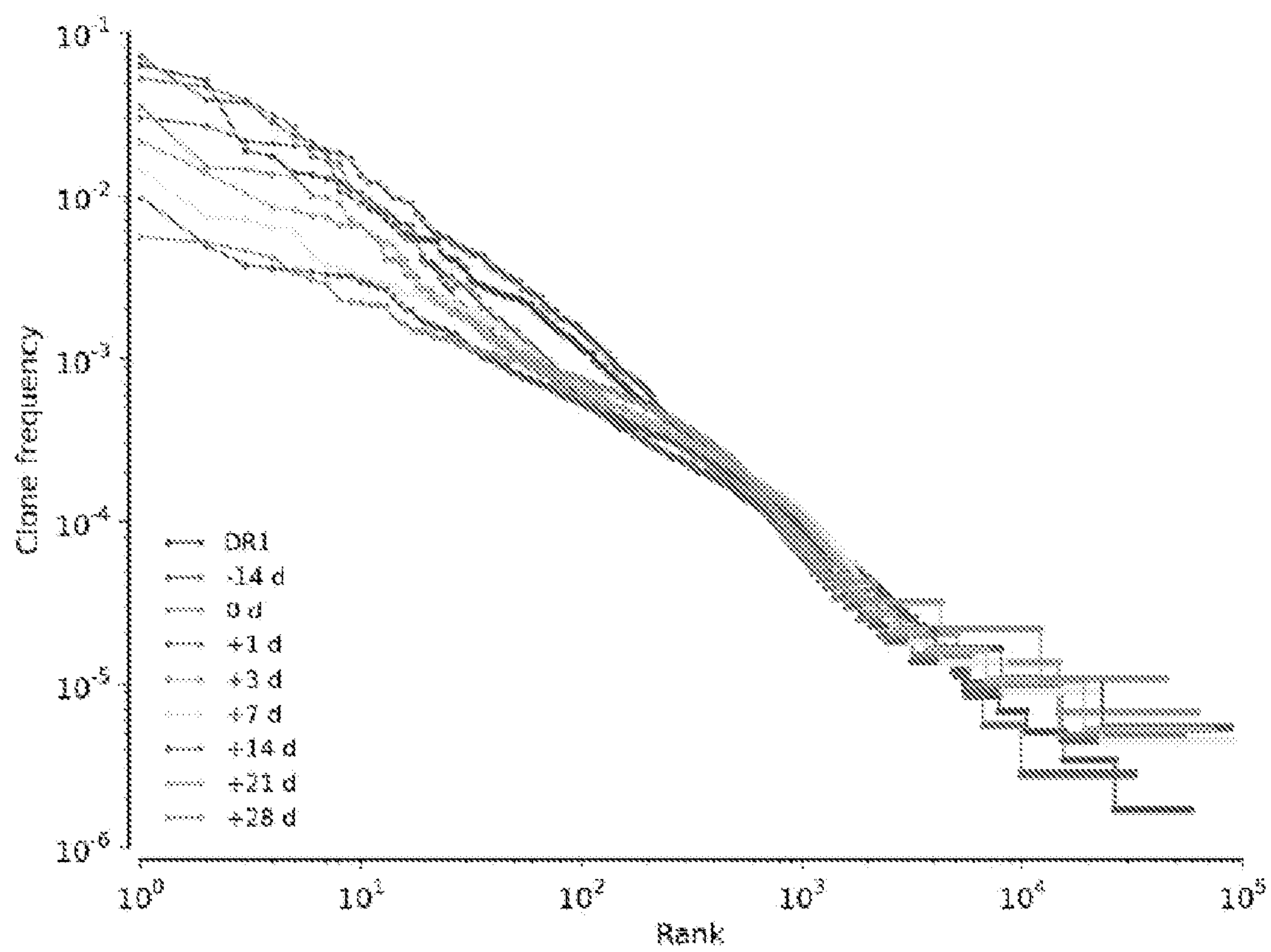
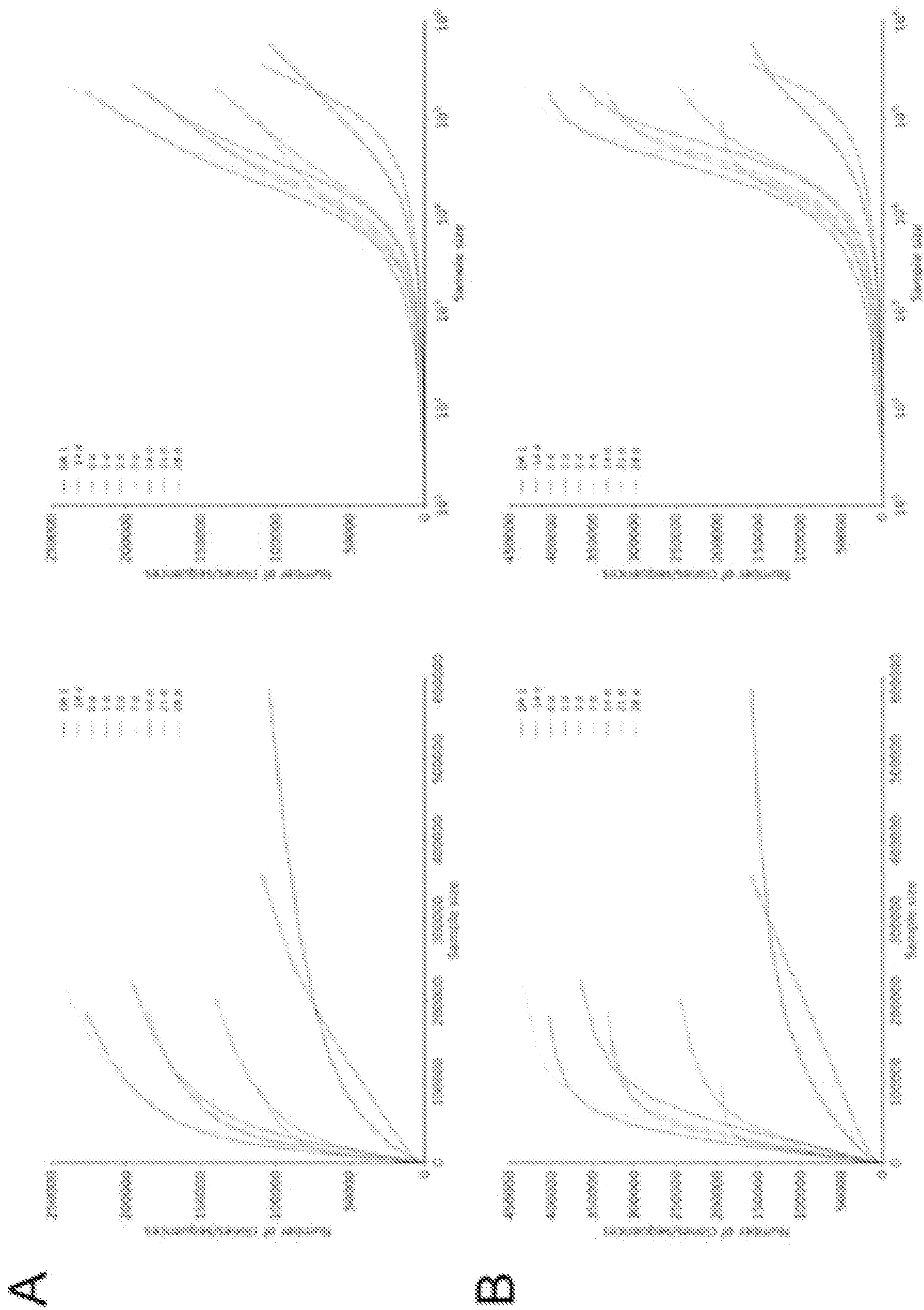


Figure 9

Figure 10

**Figure 11**

**Figure 12**

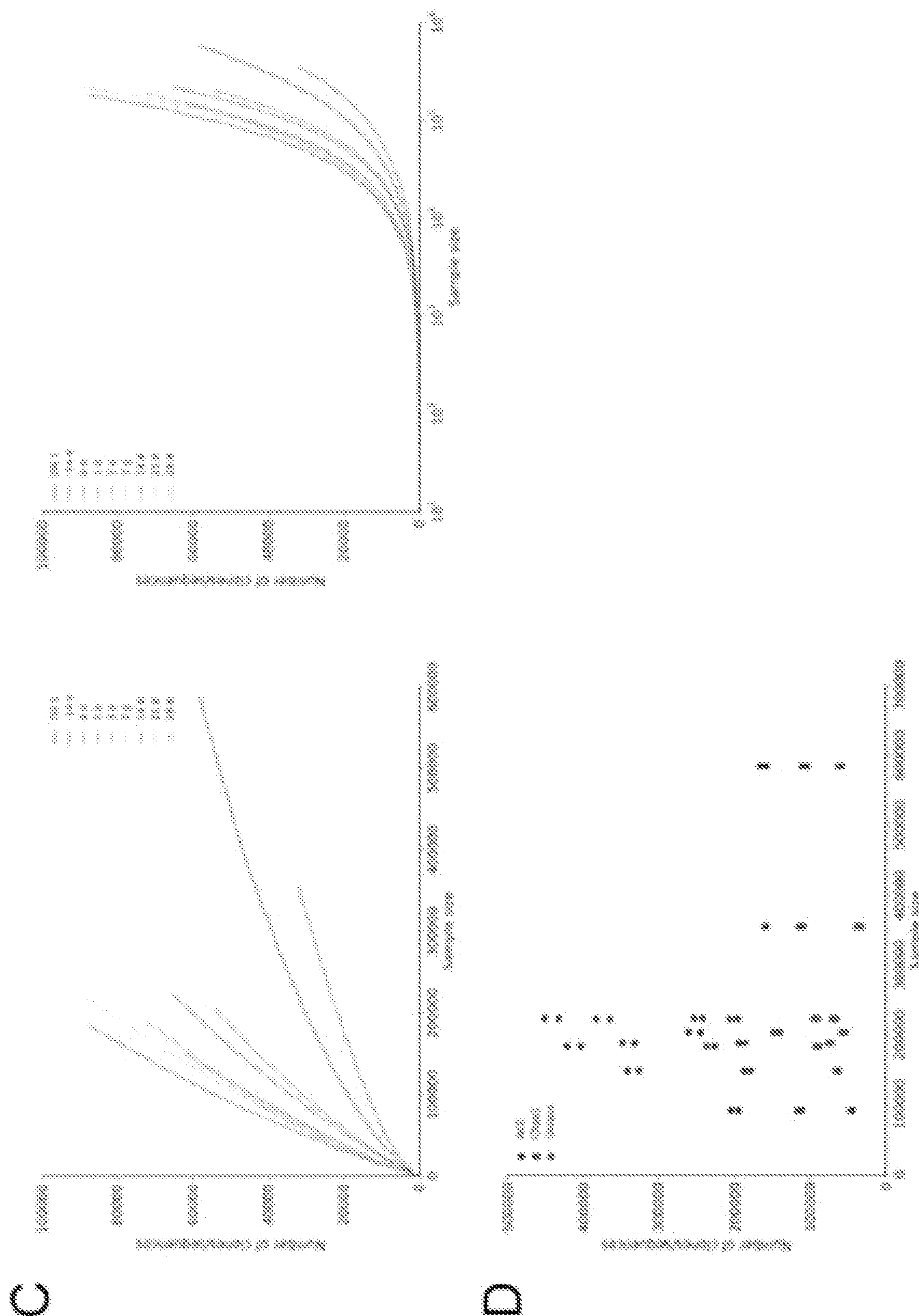
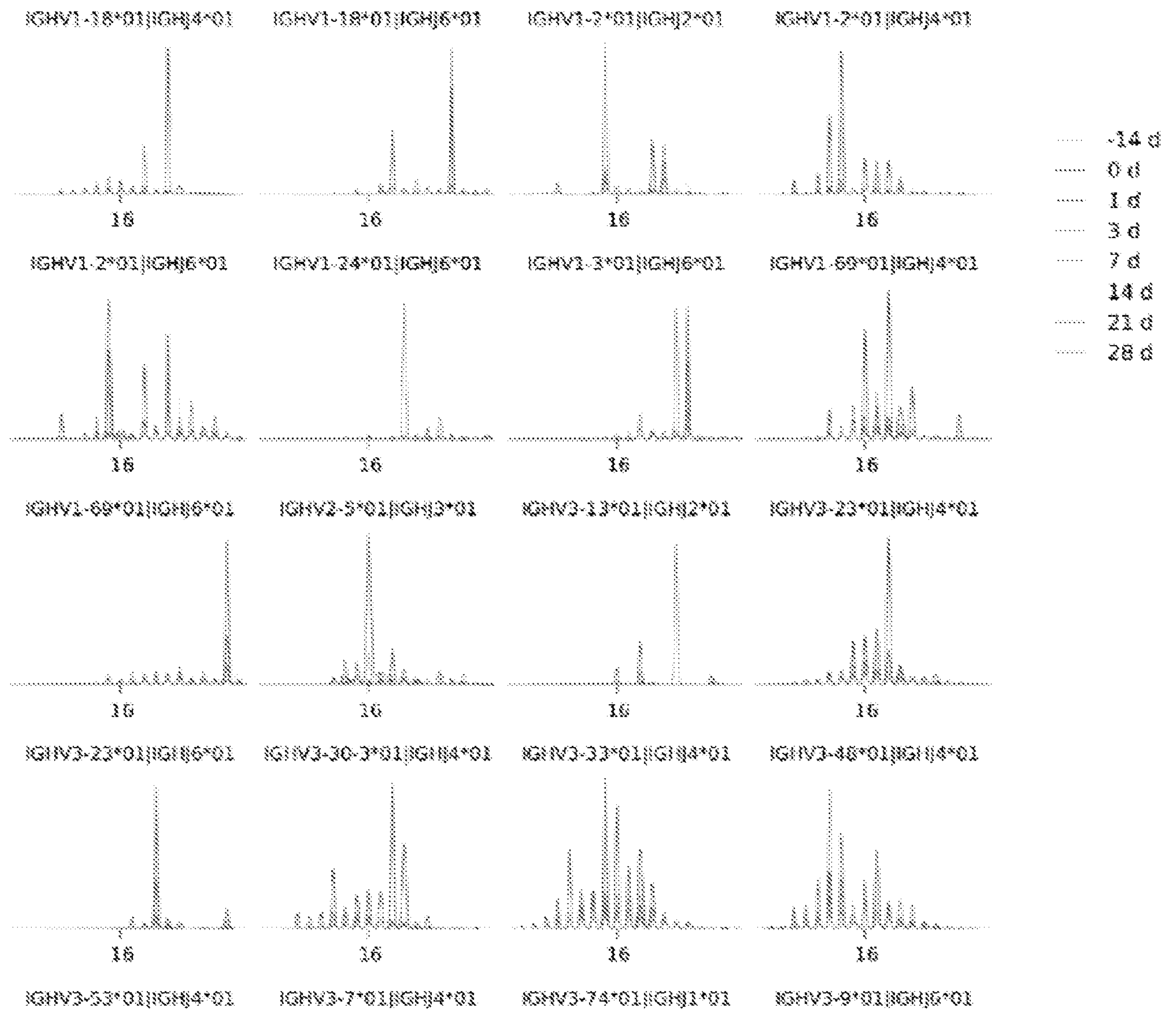
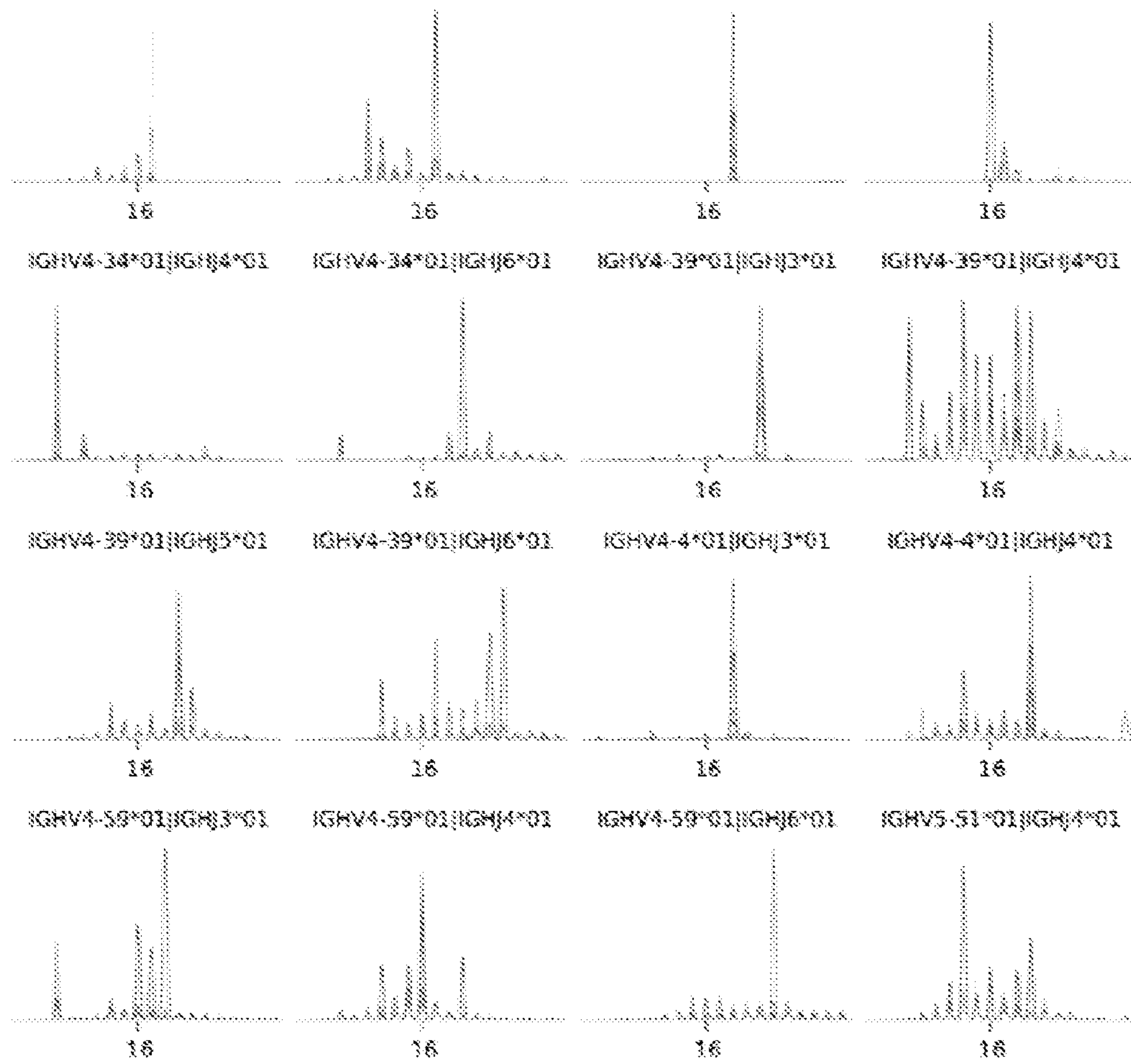
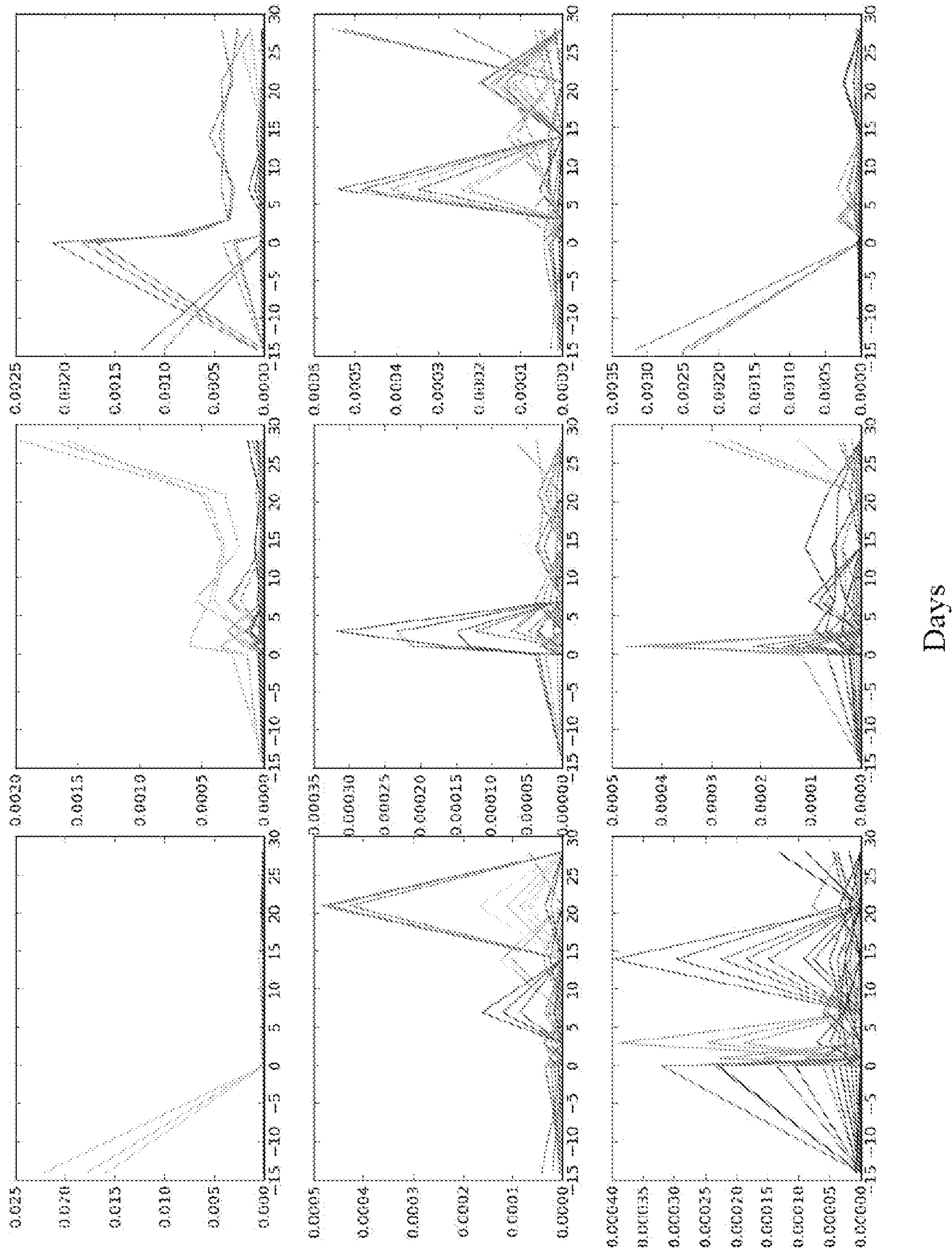


Figure 12 (Cont.)

A**Figure 13**

B**Figure 13 (Cont.)**

**Figure 14**

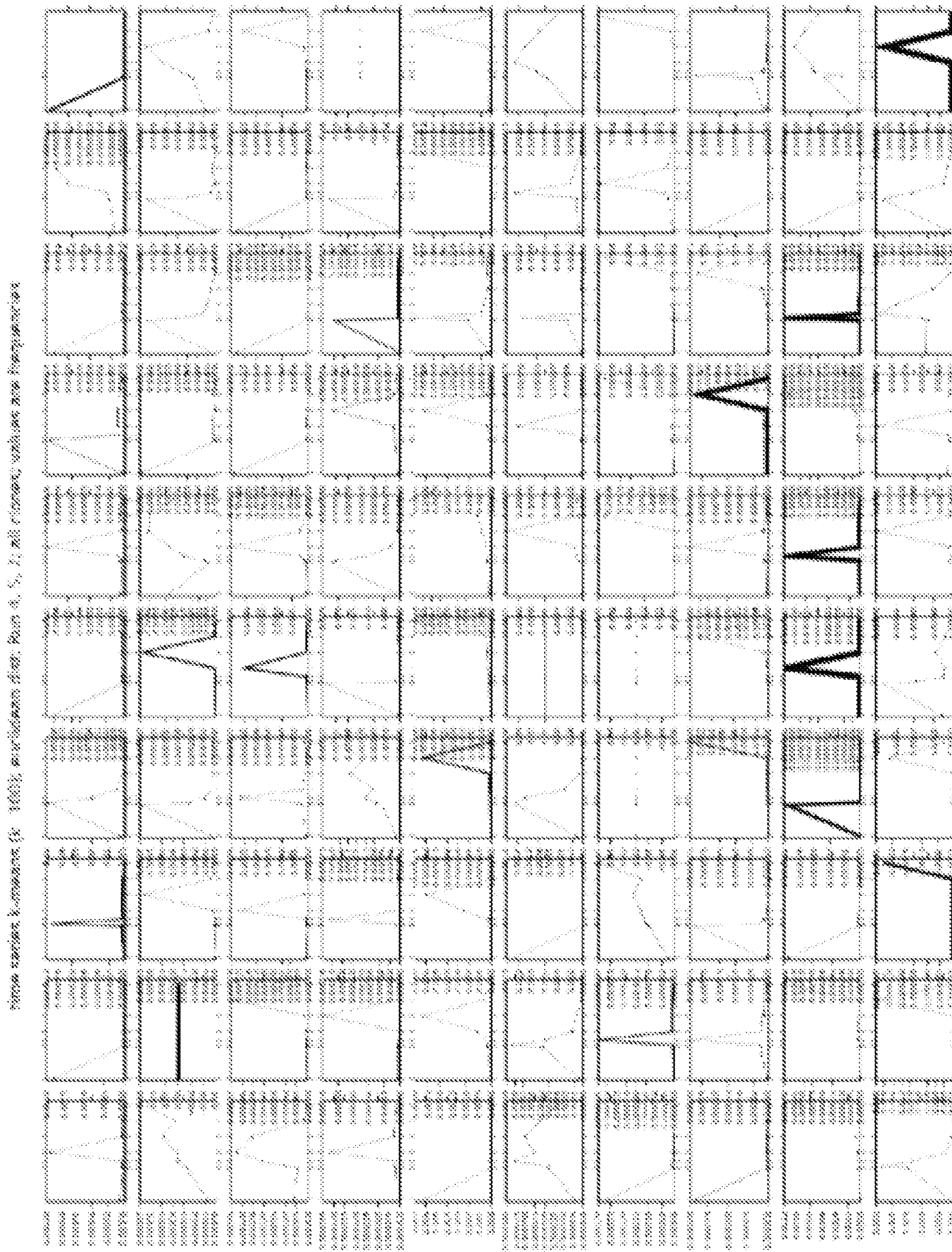


Figure 15

Figure 15 (Cont.)

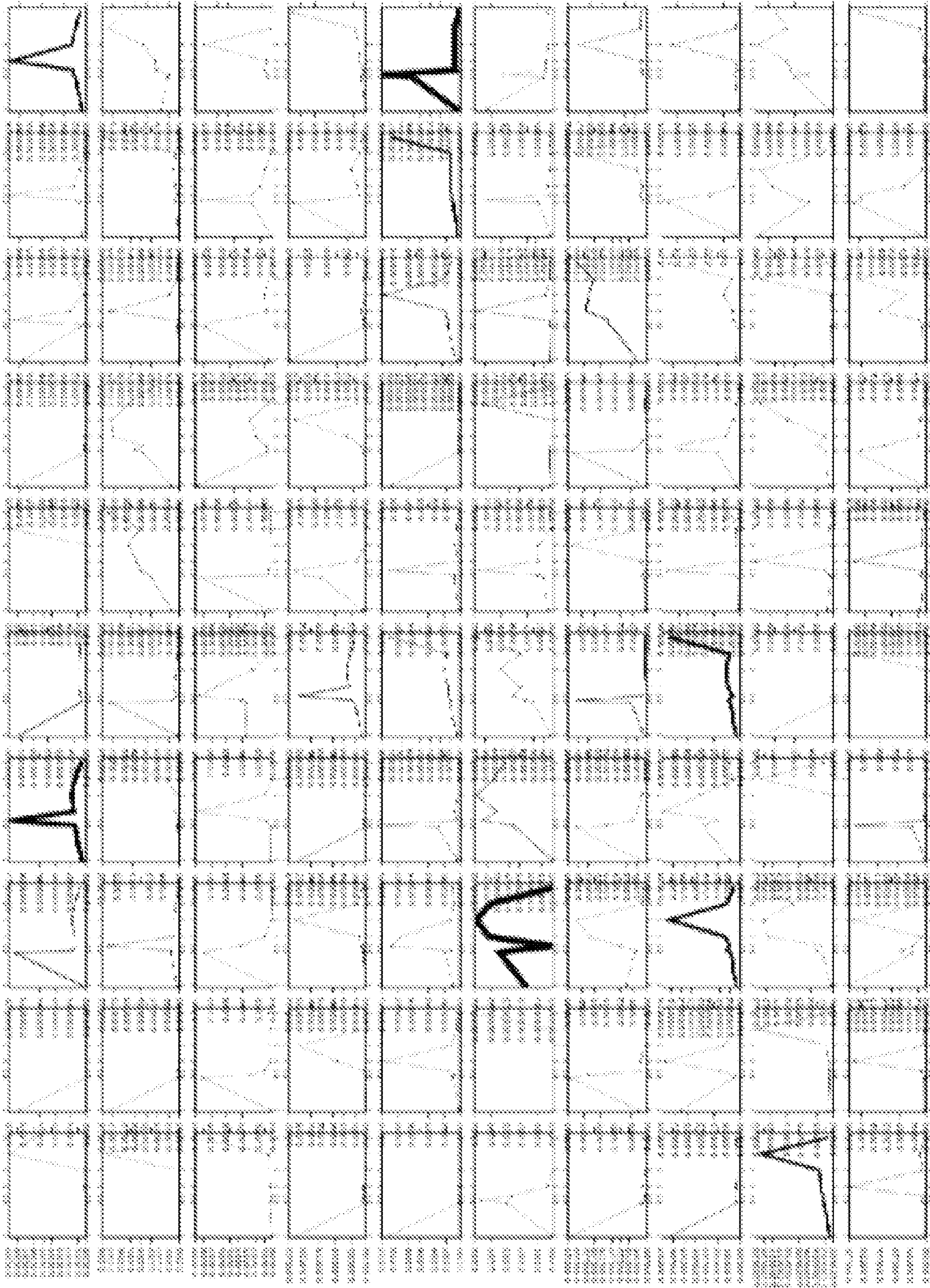


Figure 15 (Cont.)

A

Day 1

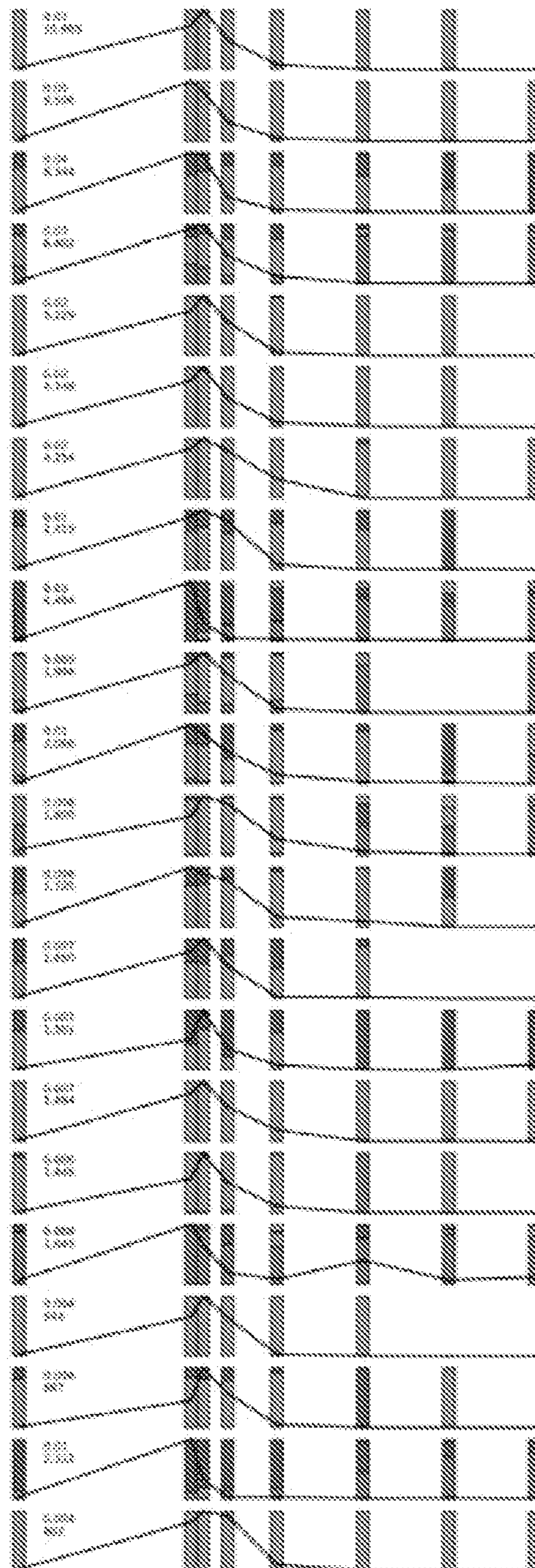
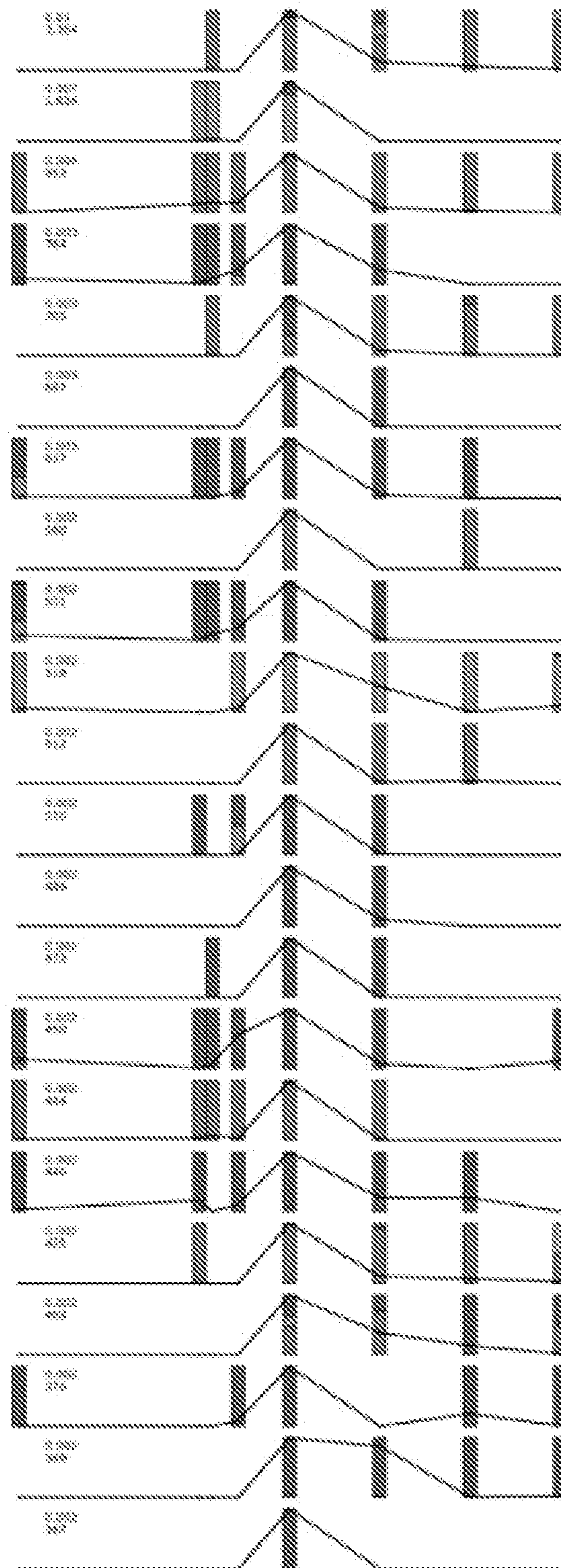


Figure 16

B

Day 7

**Figure 16 (Cont.)**

C

Day 21

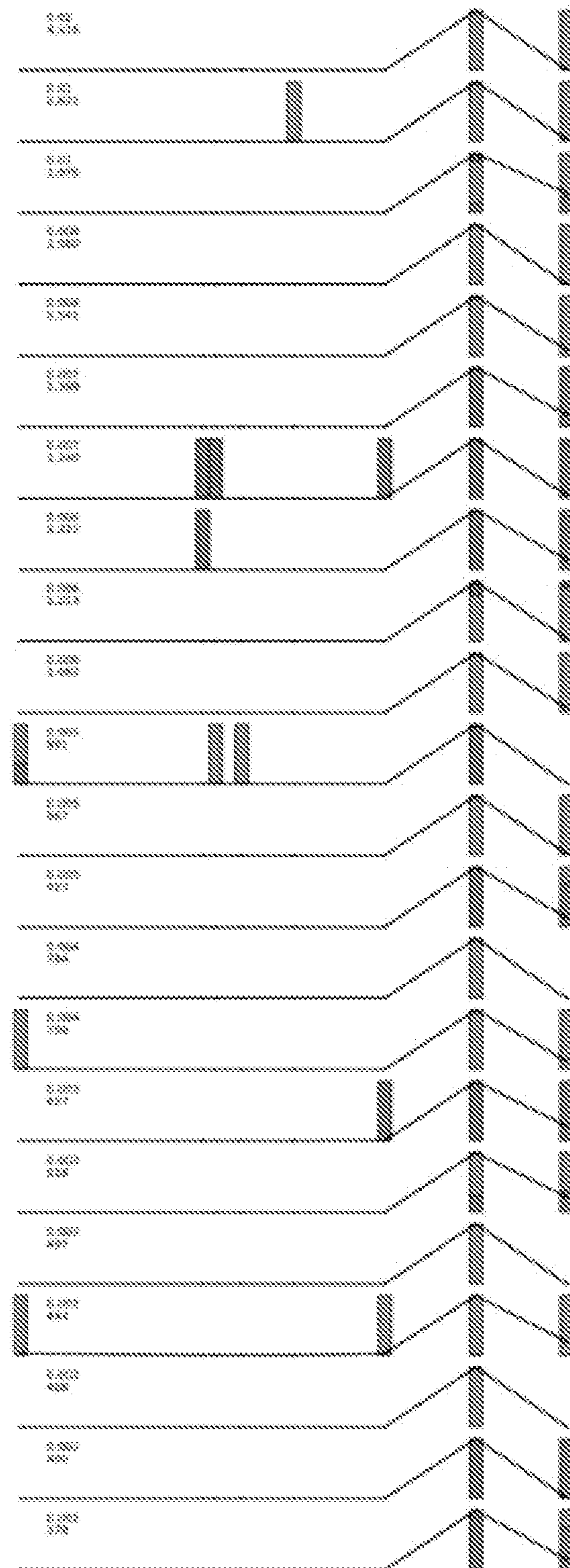
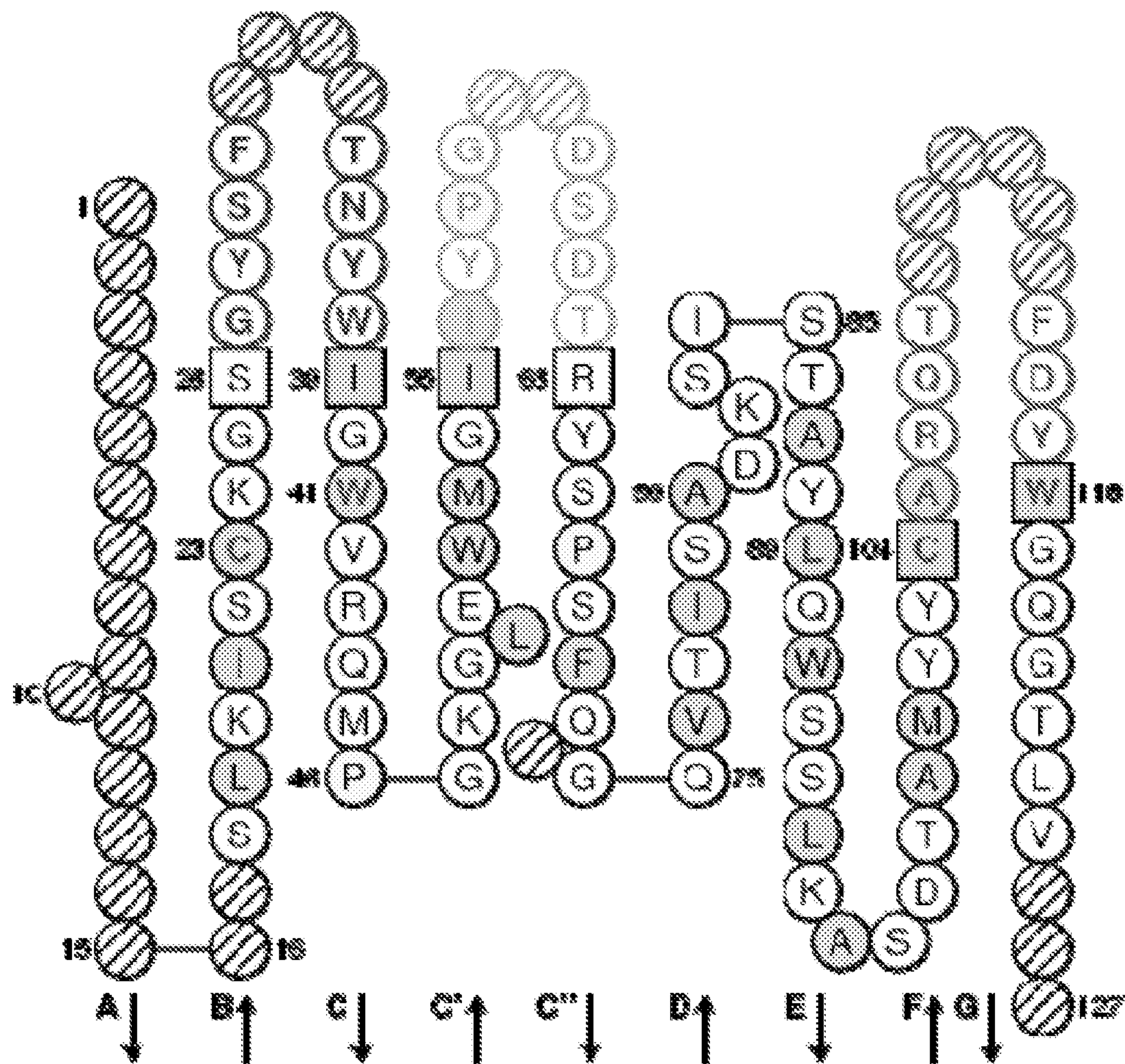


Figure 16 (Cont.)

A



B

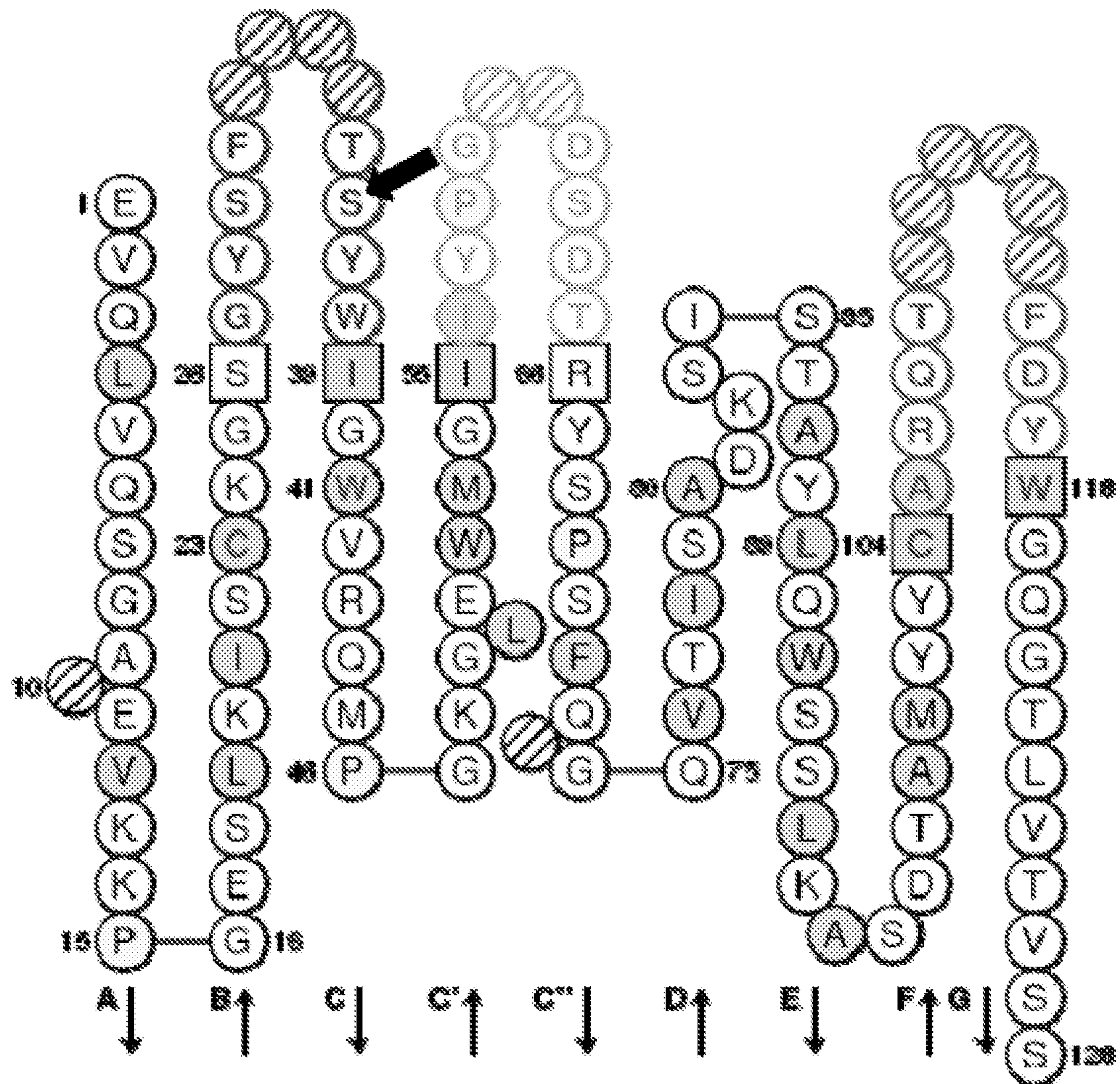


Figure 17 (Cont.)

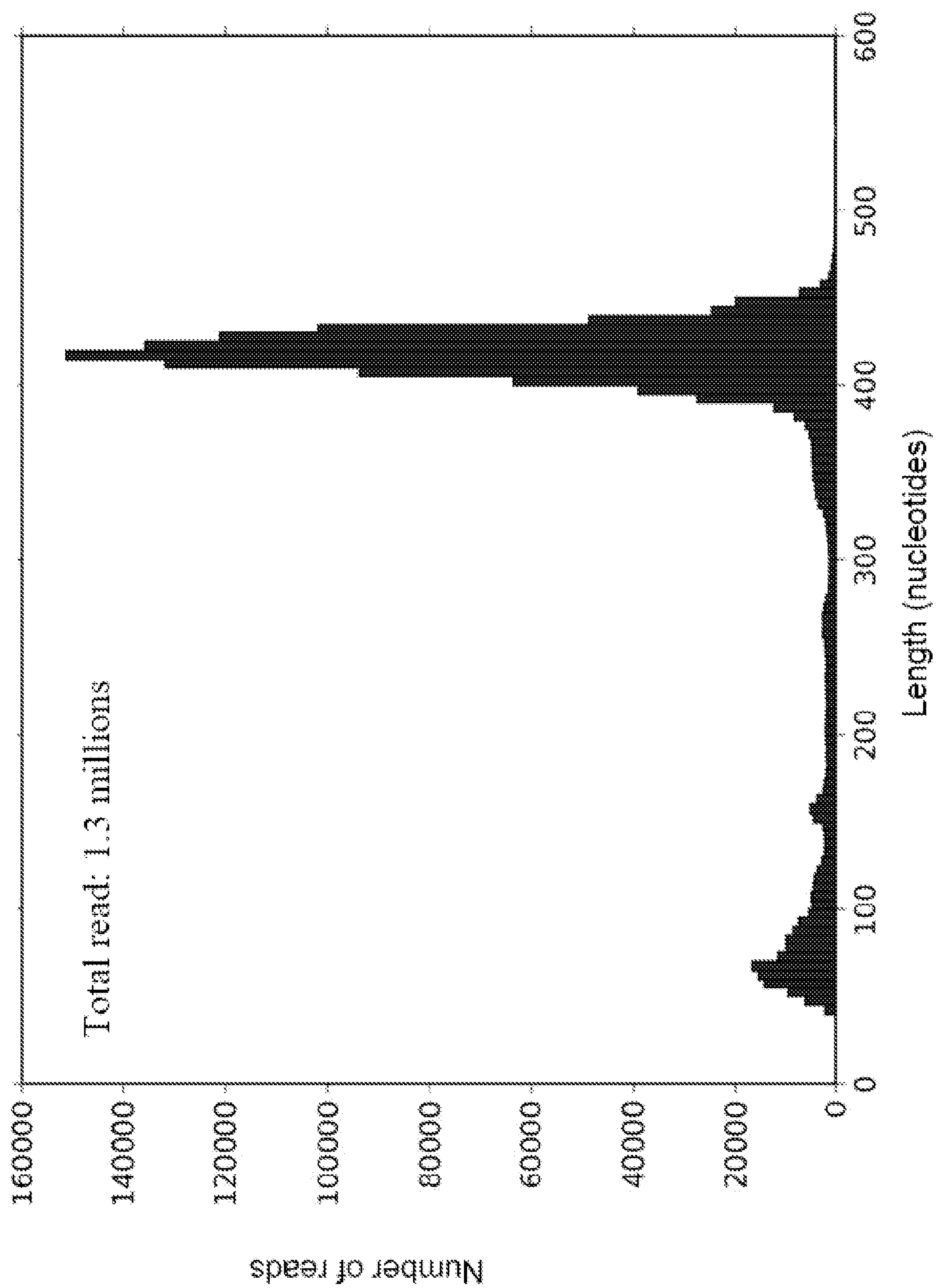
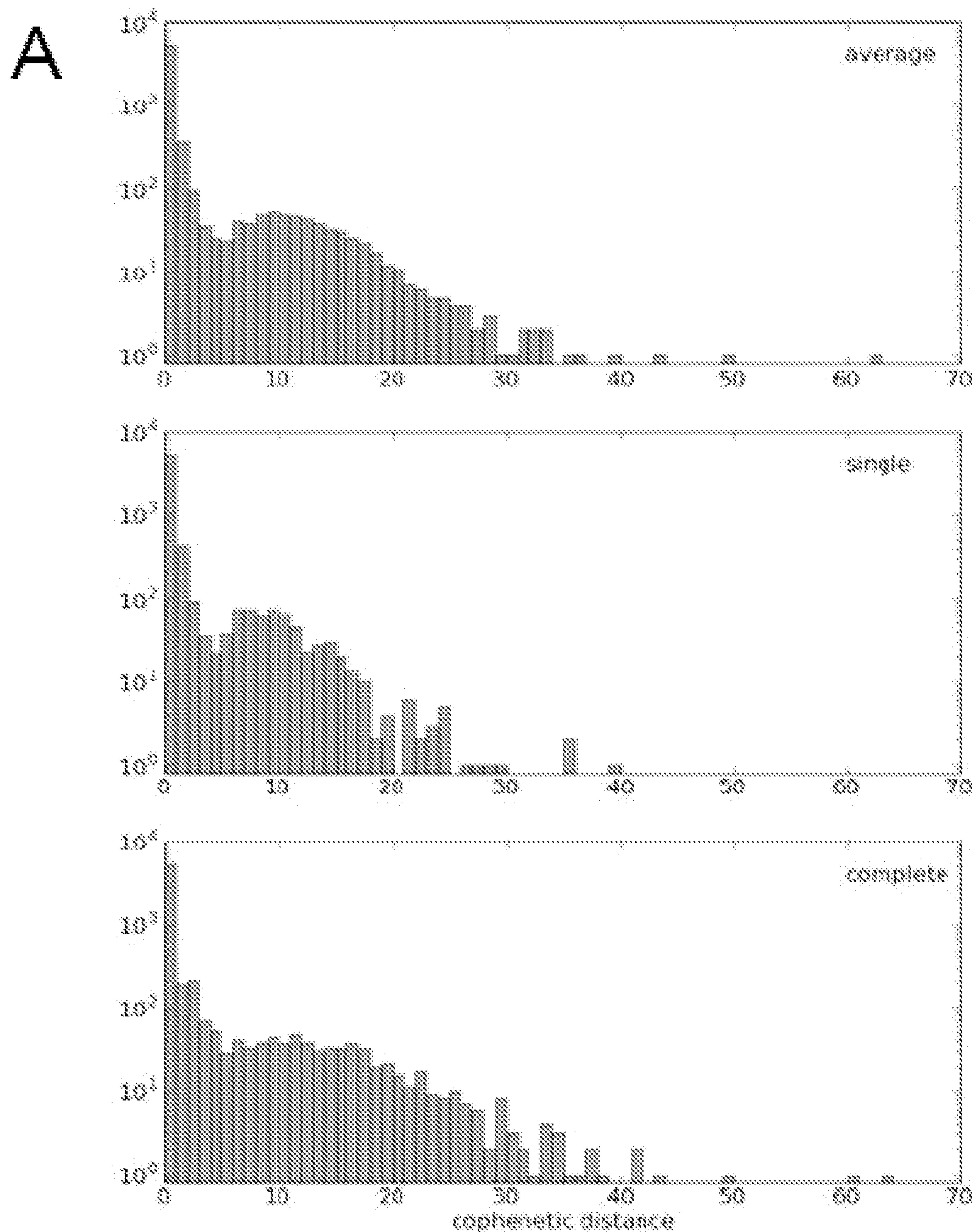
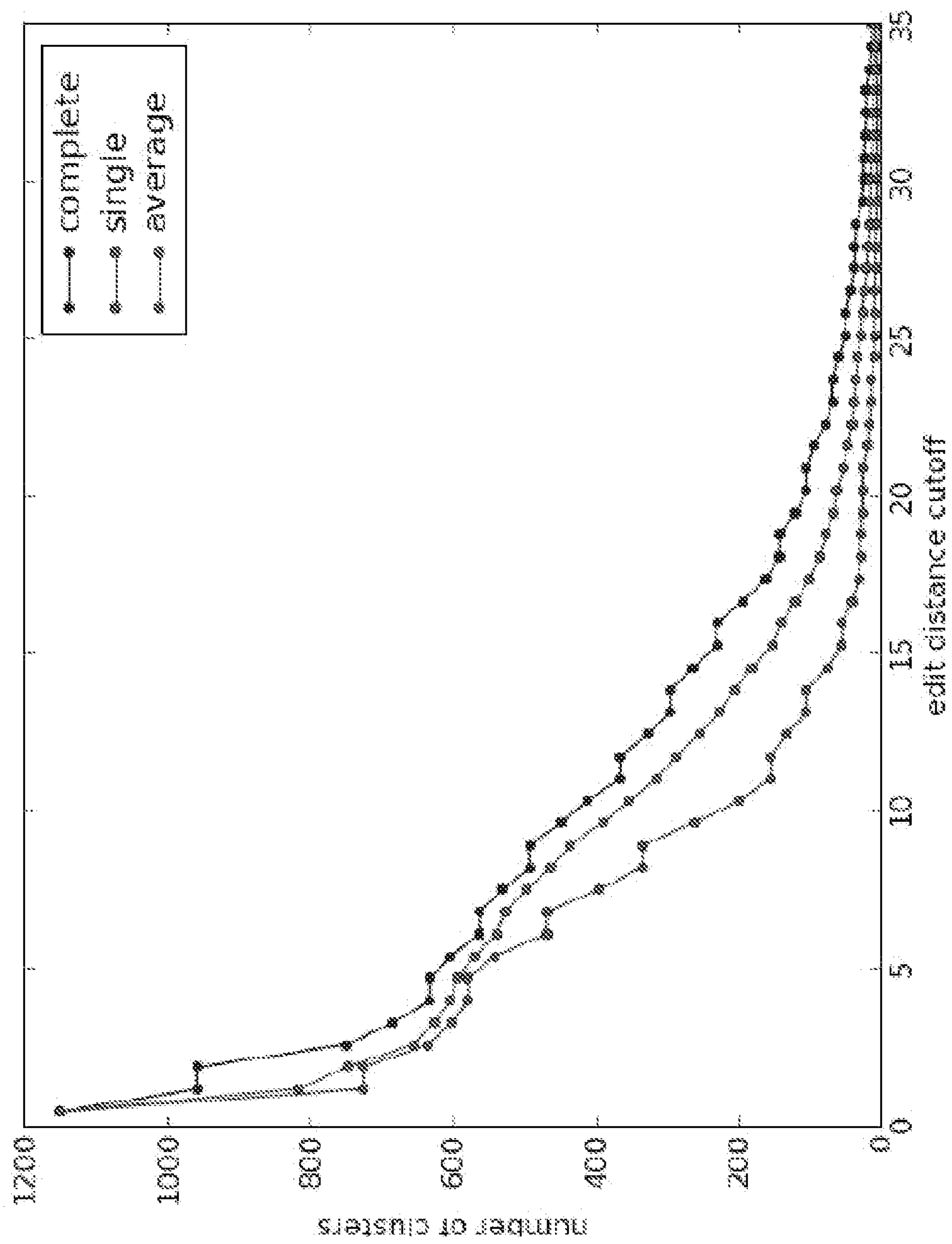
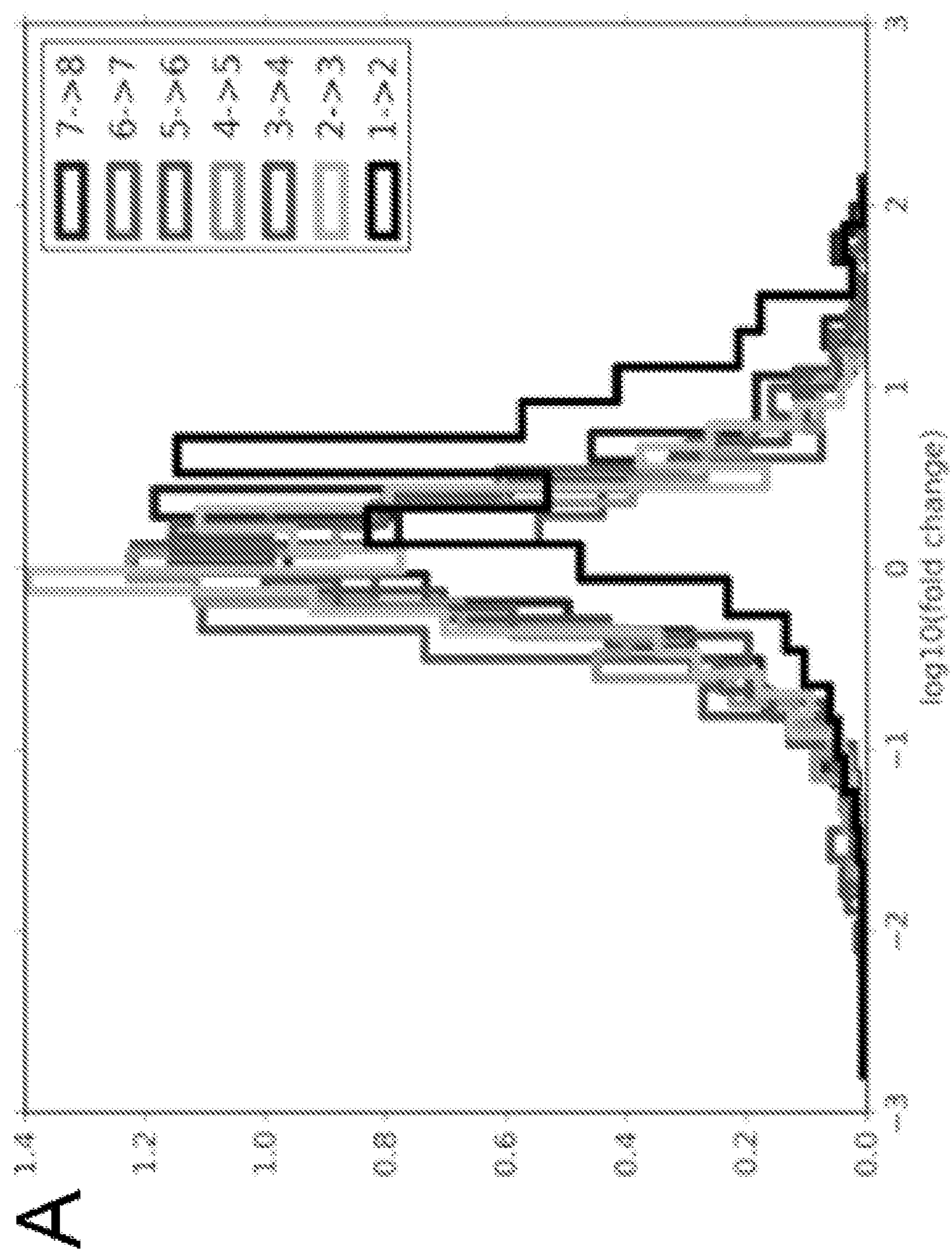


Figure 18

**Figure 19**

B**Figure 19 (Cont.)**

**Figure 20**

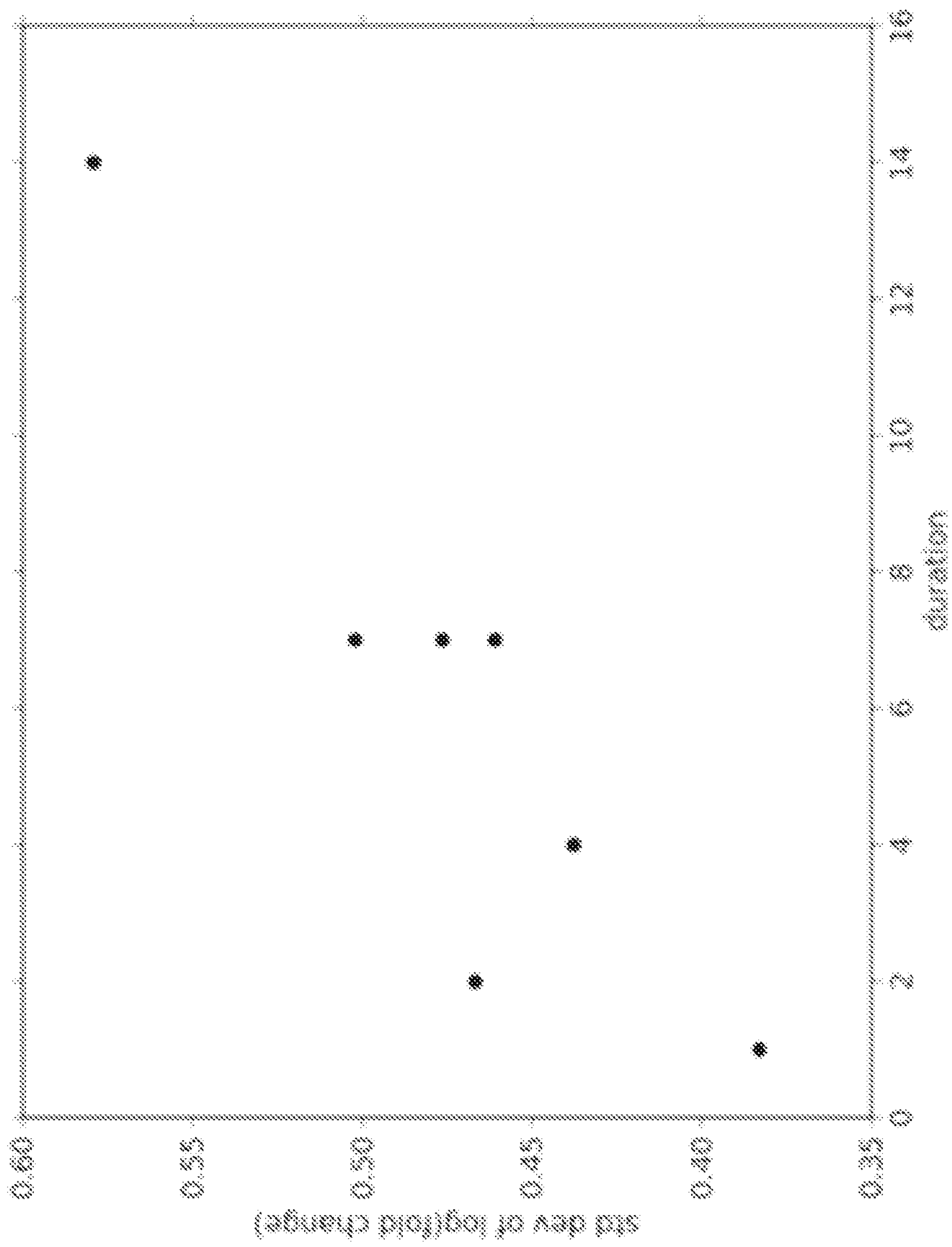


Figure 20 (Cont.)

Figure 21

Figure 22

1 : CysAlaArgGlnThrPheAspTyrTrp : 9 (SEQ ID NO:137)
| | | | | | | | | | | | | | | | | |
CysAlaArgGlnThrPheAspTyrTrp
238 : TGTGCGAGACAAACTTTGACTACTGG : 264 (SEQ ID NO:138)

Figure 23

74 : TCTCTGAAGATCTCCTGTAAAGGTTCTGGATACAGCTTTACCAGCTACTGGATCGGCTGGGTGC : 137
SerLeuLysIleSerCysLysGlySerGlyTyrSerPheThrSerTyrTrpIleGlyTrpVala
|||||
SerLeuLysIleSerCysLysGlySerGlyTyrSerPheThrAsnTyrTrpIleGlyTrpVala
1 : TCTCTGAAGATCTCCTGTAAAGGTTCTGGATACAGCTTTACCAACTACTGGATCGGCTGGGTGC : 64

138 : GCCAGATGCCCGGGAAGCCCTGGAGTGGGATCATCTATCCTGGTGACTCTGATACCCAG : 200
rgGlnMetProGlyLysGlyLeuGluTrpMetGlyIleIleTyrProGlyAspSerAspThrAr
|||||
rgGlnMetProGlyLysGlyLeuGluTrpMetGlyIleIleTyrProGlyAspSerAspThrAr
65 : GCCAGATGCCCGGGAAGCCCTGGAGTGGGATCATCTATCCTGGTGACTCTGATACCCAG : 127

201 : ATACAGCCCCGTCCCTTCCAAGGCCAGGTCACCATCTCAGCCGACAAAGTCCATCAGCACCCGCCTAC : 263
gTyrSerProSerPheGlnGlyGlnValThrIleSerAlaAspLysSerIleSerThrAlaTyr
|||||
gTyrSerProSerPheGlnGlyGlnValThrIleSerAlaAspLysSerIleSerThrAlaTyr
128 : ATACAGCCCCGTCCCTTCCAAGGCCAGGTCACCATCTCAGCCGACAAAGTCCATCAGCACCCGCCTAC : 190

264 : CTGCAGTGGAGCAGCCTGAAGGCCCTCGGACACCGCCATGTATTACTGTGCGAGACAGACTTTTG : 329
LeuGlnTrpSerSerLeuLysAlaSerAspThrAlaMetTyrTyrCysAlaArgGlnThrPheA
|||||
LeuGlnTrpSerSerLeuLysAlaSerAspThrAlaMetTyrTyrCysAlaArgGlnThrPheA
191 : CTGCAGTGGAGCAGCCTGAAGGCCCTCGGACACCGCCATGTATTACTGTGCGAGACAAACTTTTG : 256

330 : ACTACTGGGGCCAGGGAACCCCTGGTC : 355 (SEQ ID NO:139)
spTyrTrpGlyGlnGlyThrLeuVal
|||||
spTyrTrpGlyGlnGlyThrLeuVal
257 : ACTACTGGGGCCAGGGAACCCCTGGTC : 282 (SEQ ID NO:140)

Figure 24

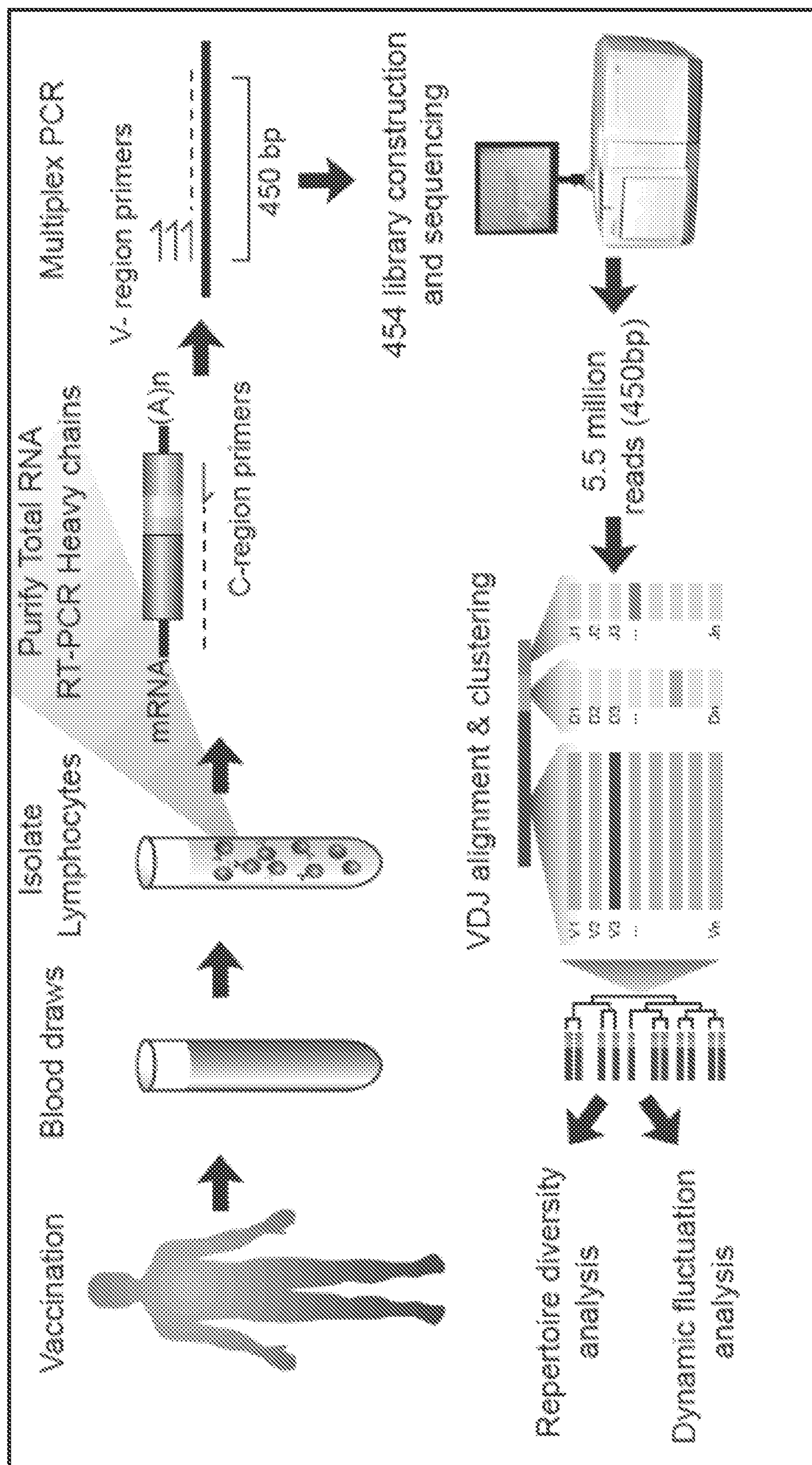


Figure 25

	DR1	%	OR1	%	OR2	%	SR1	%	TR1	%	OR3	%	SR2	%	Total
Total reads	1,183,769	100	455,187	100	1,147,232	100	622,356	100	1,473,502	100	1,268,232	100	1,004,209	100	7,154,487
Size-filtered reads	793,976	67	396,339	87	994,138	87	491,747	79	1,087,398	74	1,021,570	81	709,138	71	5,494,306
Num with barcode	783,514	66	388,467	85	978,201	85	470,122	76	1,038,683	70	947,307	75	676,660	67	5,282,954
Num with BC	549,751	46	264,802	58	675,828	59	271,529	44	517,539	35	618,670	49	434,549	43	3,332,668
and isotype															
Num with	411,370	35	214,338	4	546,118	48	218,622	35	423,888	29	486,541	38	350,936	35	2,651,813
BC, isotype, and VJ aln															
Num with BC, VJ aln	578,114	49	310,792	68	785,941	69	354,136	57	743,527	50	727,521	57	527,222	53	4,027,253
barcode 01	n/a		76,904	19	193,546	19	203,132	41	492	0	424,293	42	288,285	41	1,186,652
barcode 02	n/a		62,404	16	153,280	15	37,082	8	177,188	16	73,609	7	53,070	7	556,633
barcode 03	n/a		47,901	12	117,265	12	55,773	11	154,035	14	91,313	9	80,222	11	546,509
barcode 04	n/a		46,742	12	115,577	12	35,350	7	108,827	10	73,035	7	51,734	7	431,265
barcode 05	n/a		31,452	8	79,290	8	48,500	10	192,850	18	89,292	9	72,972	10	514,356
barcode 06	n/a		41,098	10	109,170	11	33,395	7	164,853	15	74,007	7	47,885	7	470,408
barcode 08	n/a		36,920	9	89,610	9	37,904	8	163,523	15	80,067	8	54,807	8	462,831
barcode 09	n/a		45,046	11	120,463	12	18,986	4	76,915	7	41,691	4	27,685	4	330,786
no barcode	n/a		7,872	2	15,937	2	21,625	4	48,715	4	74,263	7	32,478	5	200,890
IGHA	239,447	31	99,500	26	252,049	26	113,414	24	129,354	12	293,256	31	197,176	29	1,324,196
IGHD	1,950	0	12,498	3	31,868	3	8,634	2	28,929	3	20,042	2	13,434	2	117,355
IGHE	20	0	18	0	55	0	23	0	71	0	58	0	36	0	281
IGHG	127,371	16	59,533	15	160,522	16	62,561	13	124,327	12	151,614	16	94,596	1	780,524
IGHM	180,963	23	93,253	24	231,334	24	86,897	18	234,858	23	153,700	16	129,307	19	1,110,312
no isotype	233,763	30	123,665	32%	302,373	31	198,593	42	521,144	50	328,637	35	242,111	36	1,950,286

Figure 26

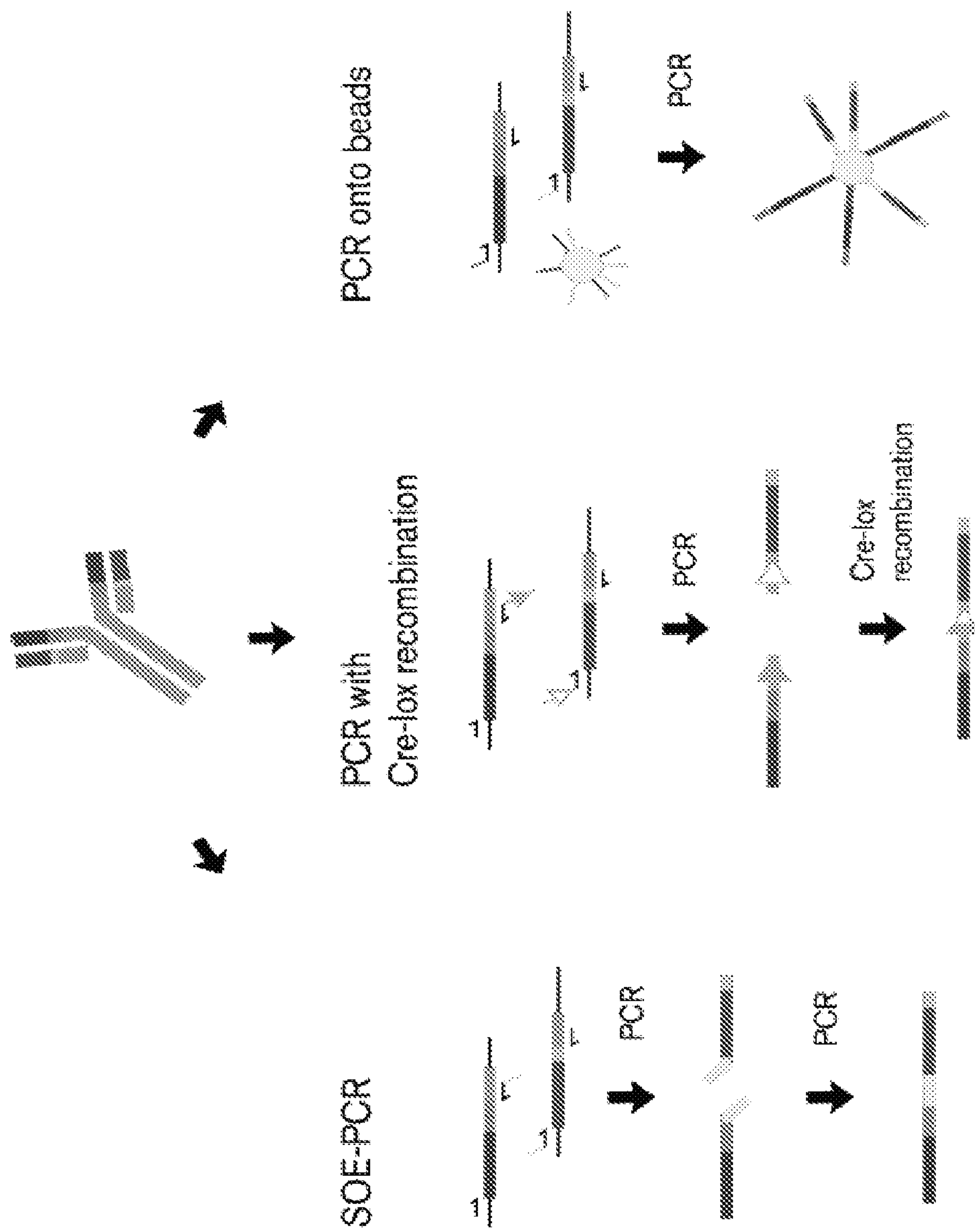
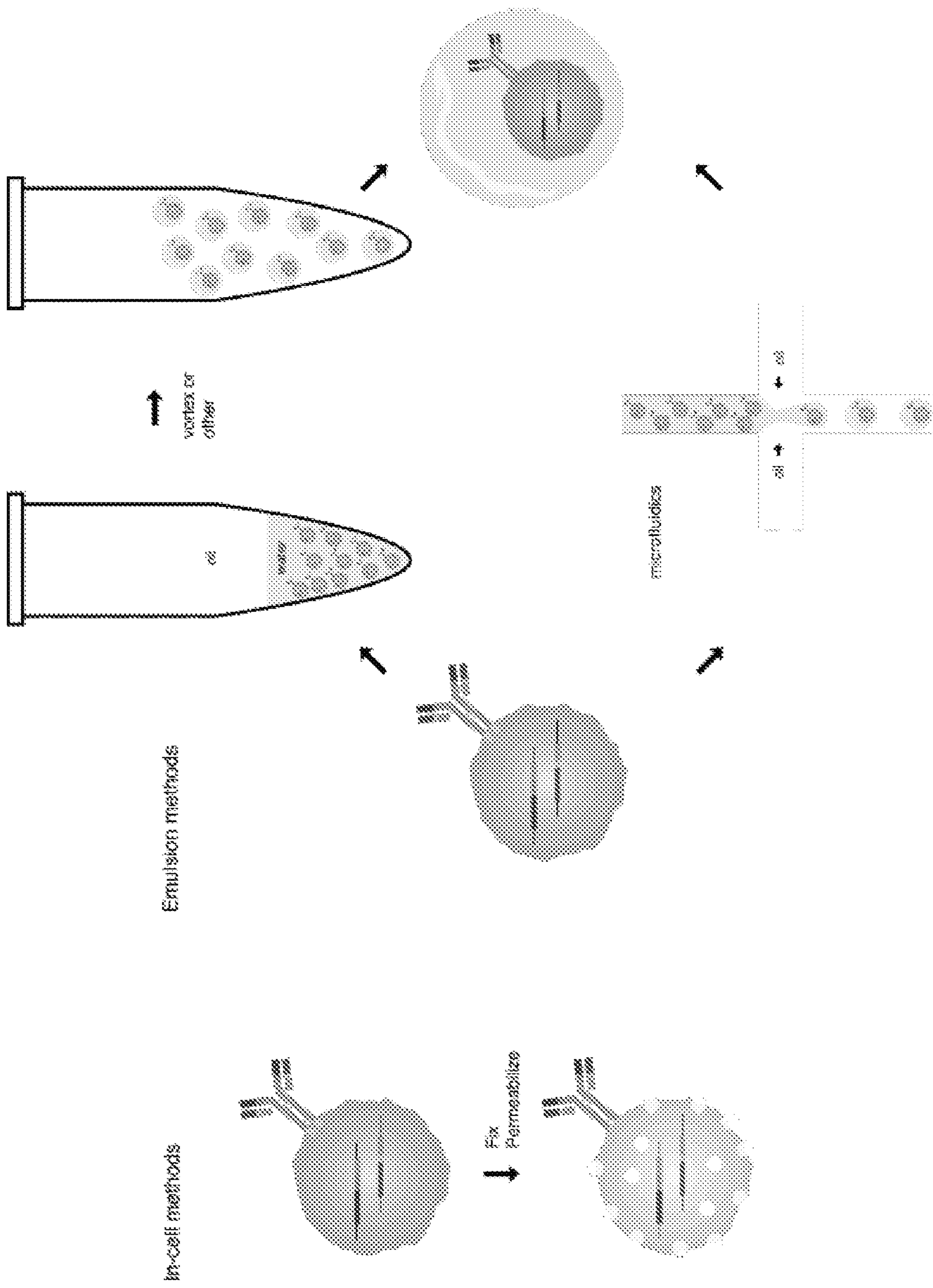
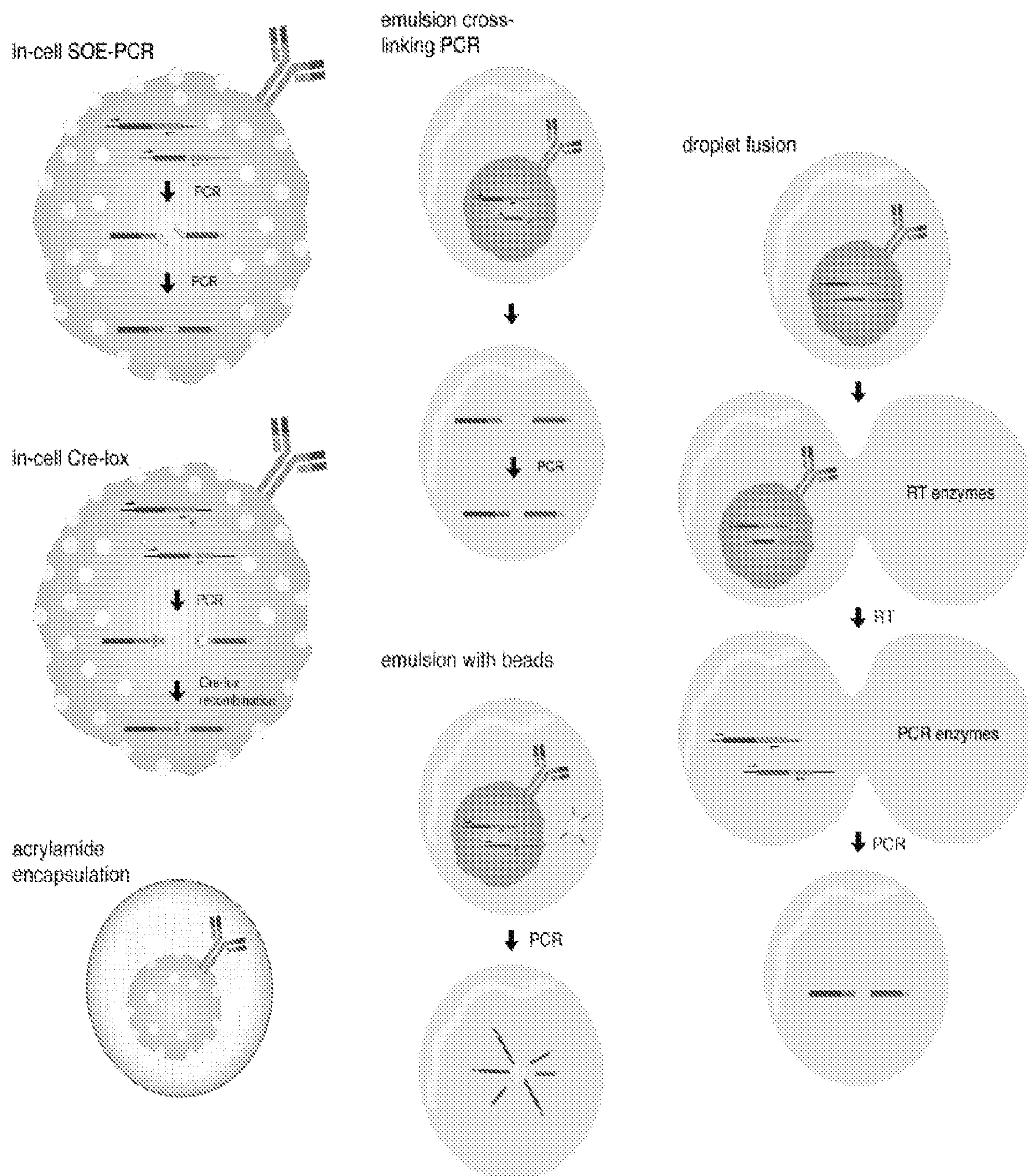
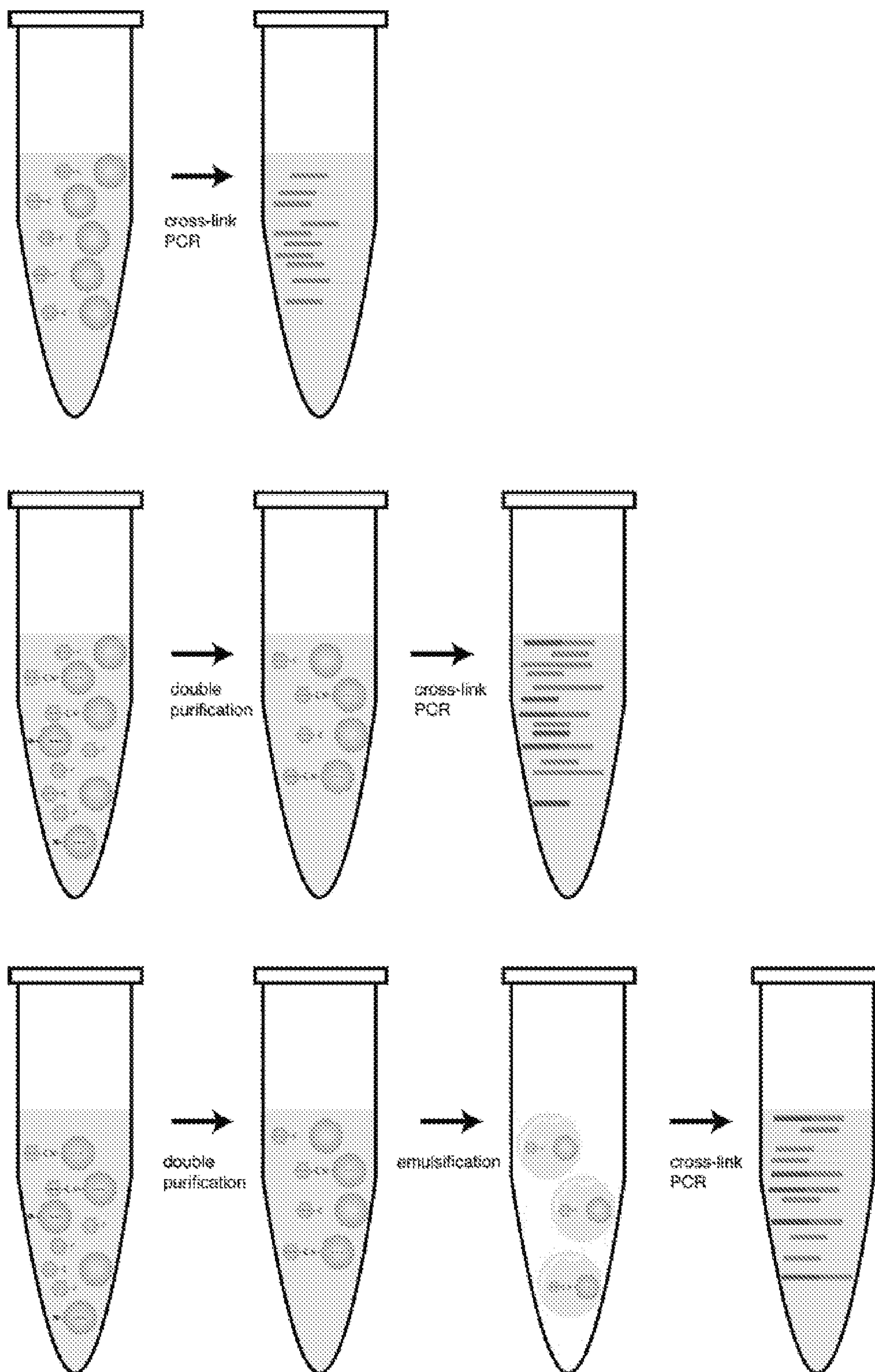


Figure 27

**Figure 28**

**Figure 29**

**Figure 30**

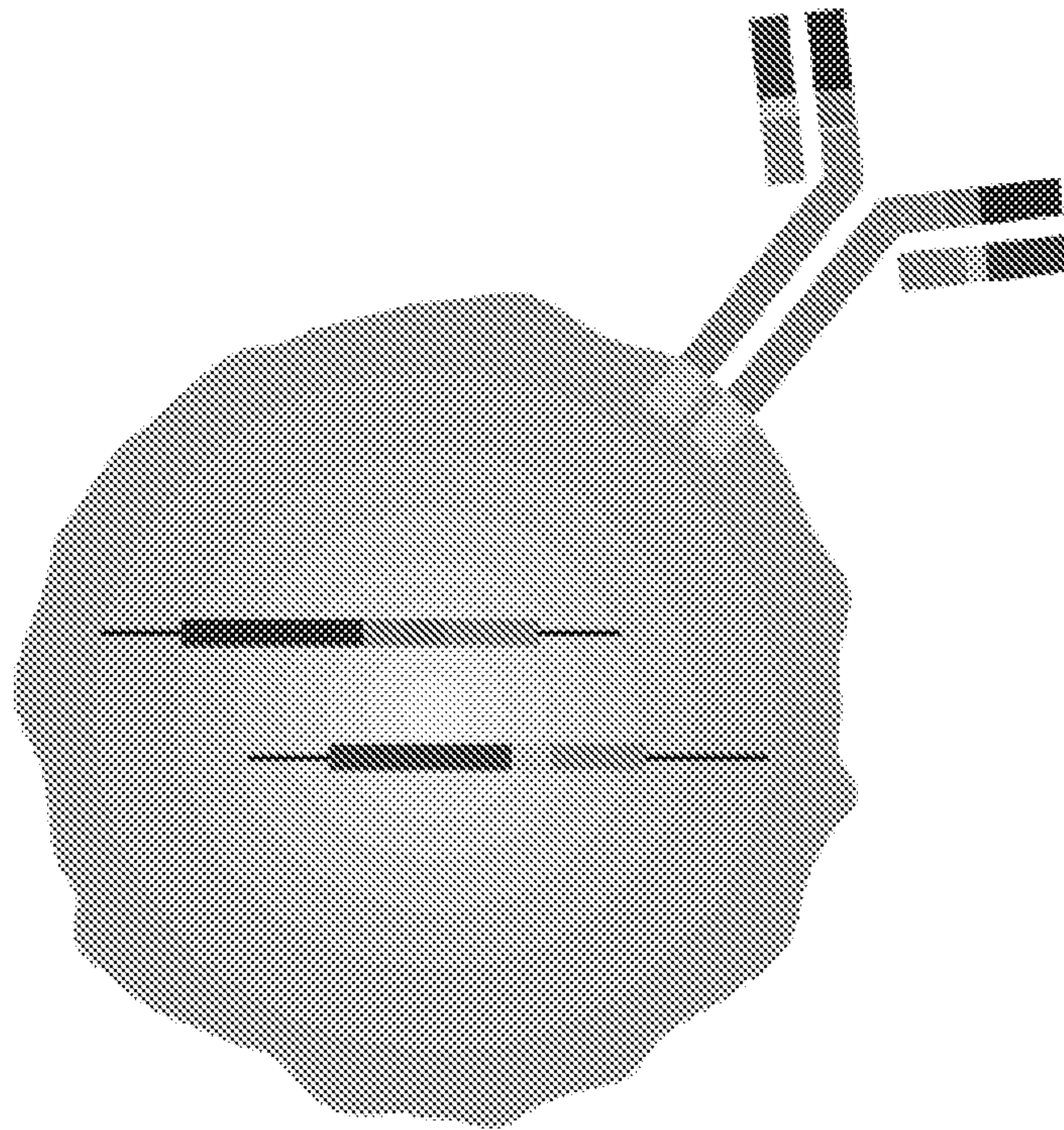


Figure 31

47/91

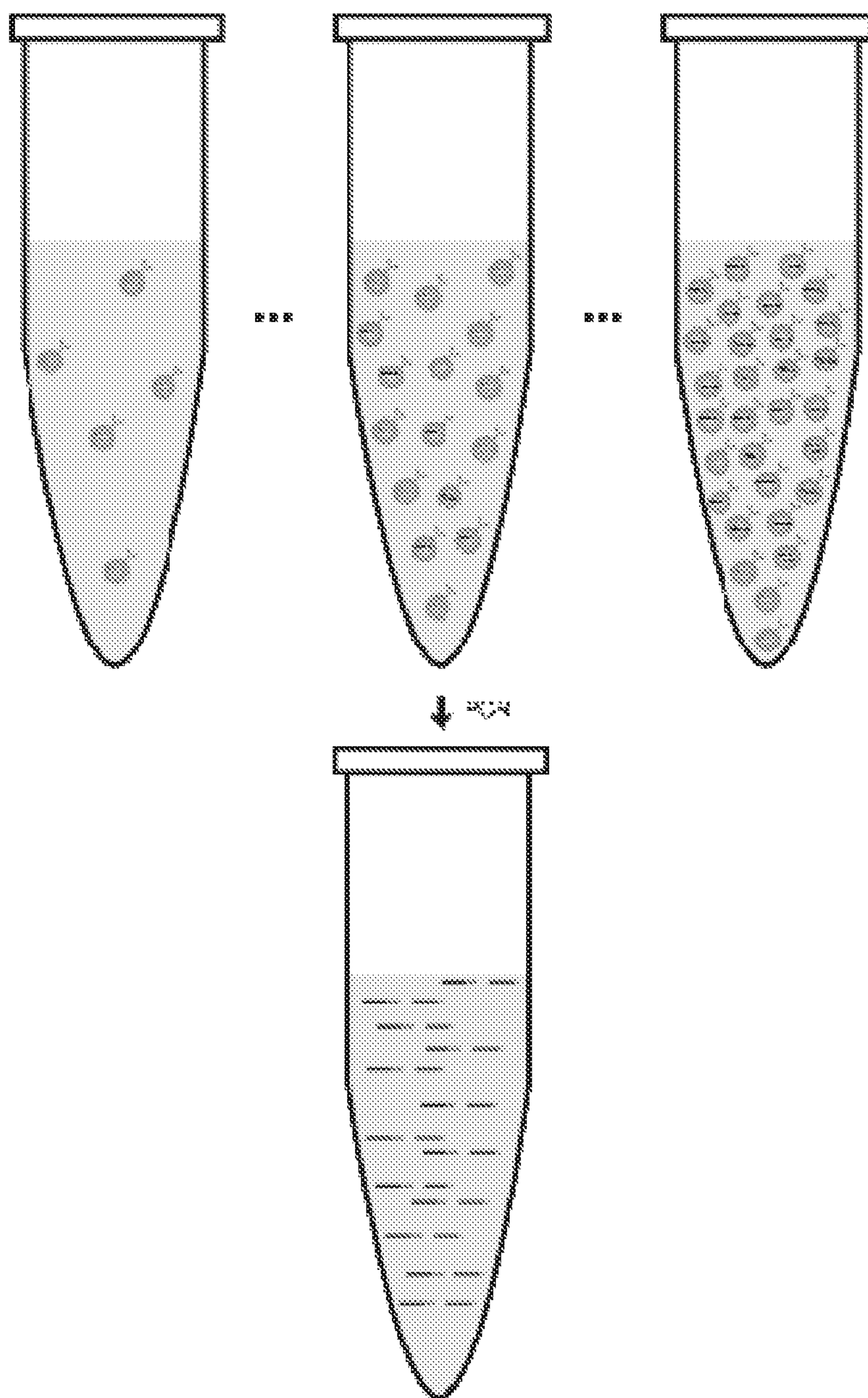
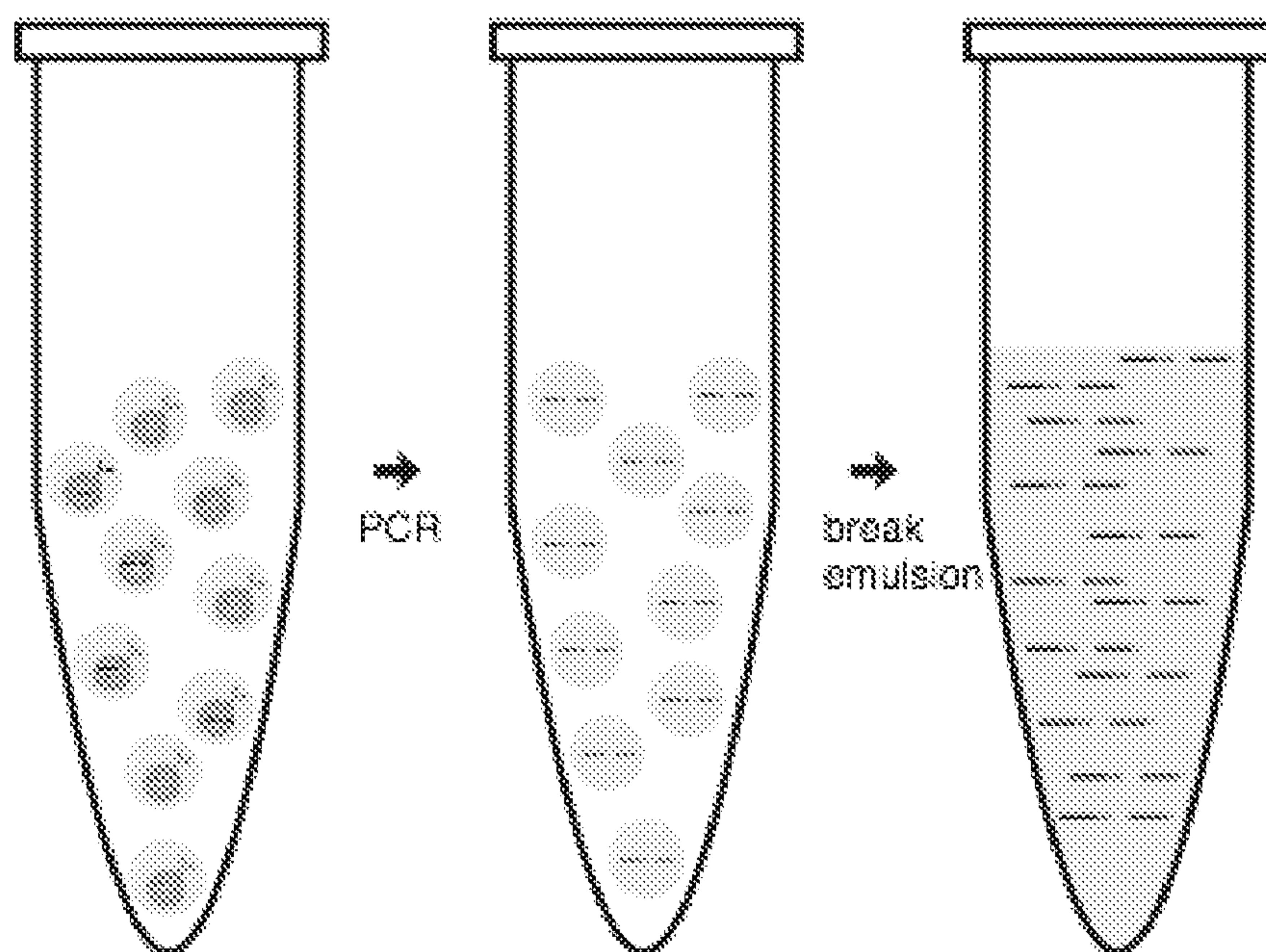
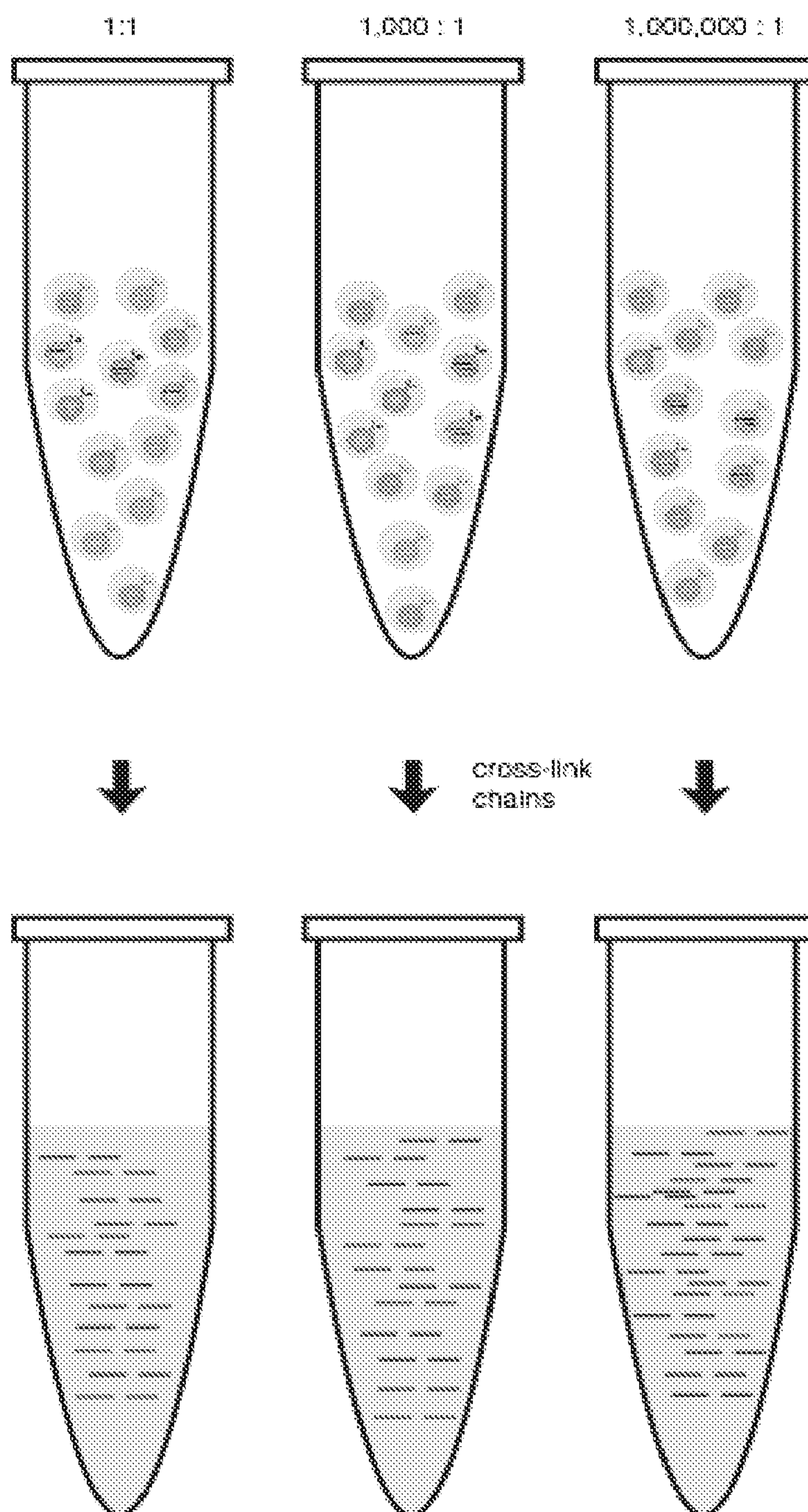


Figure 32

**Figure 33**

**Figure 34**

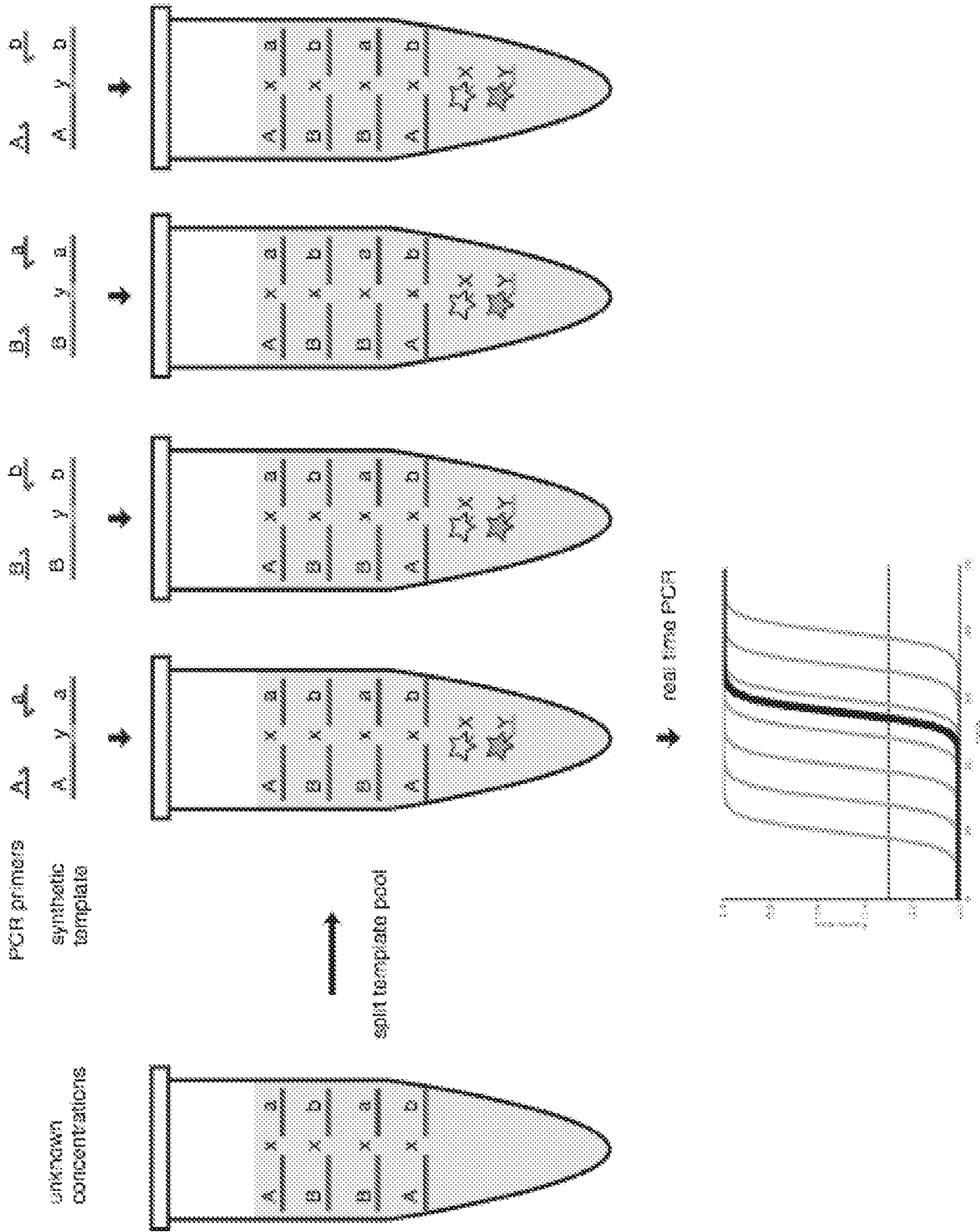


Figure 35

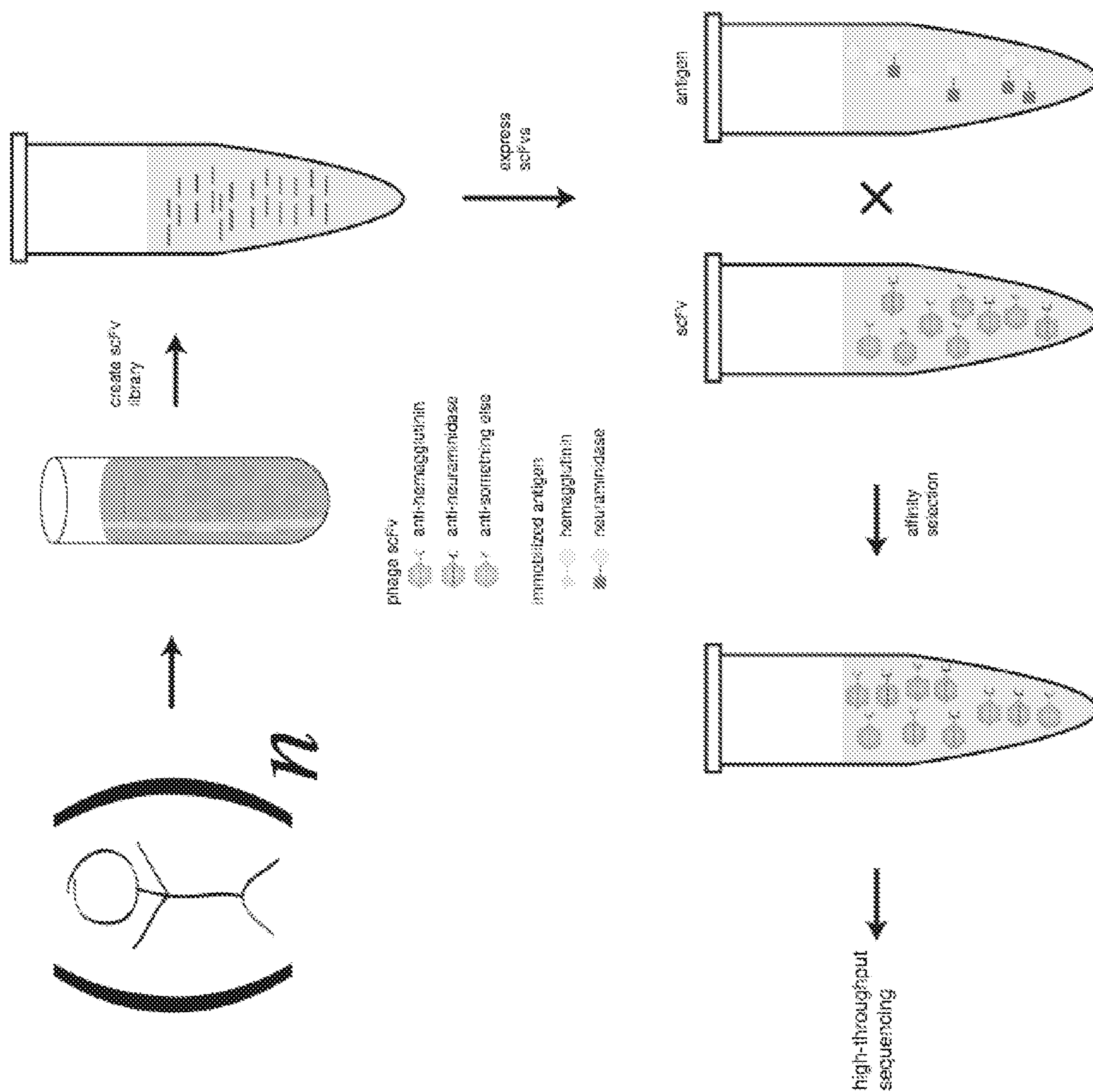
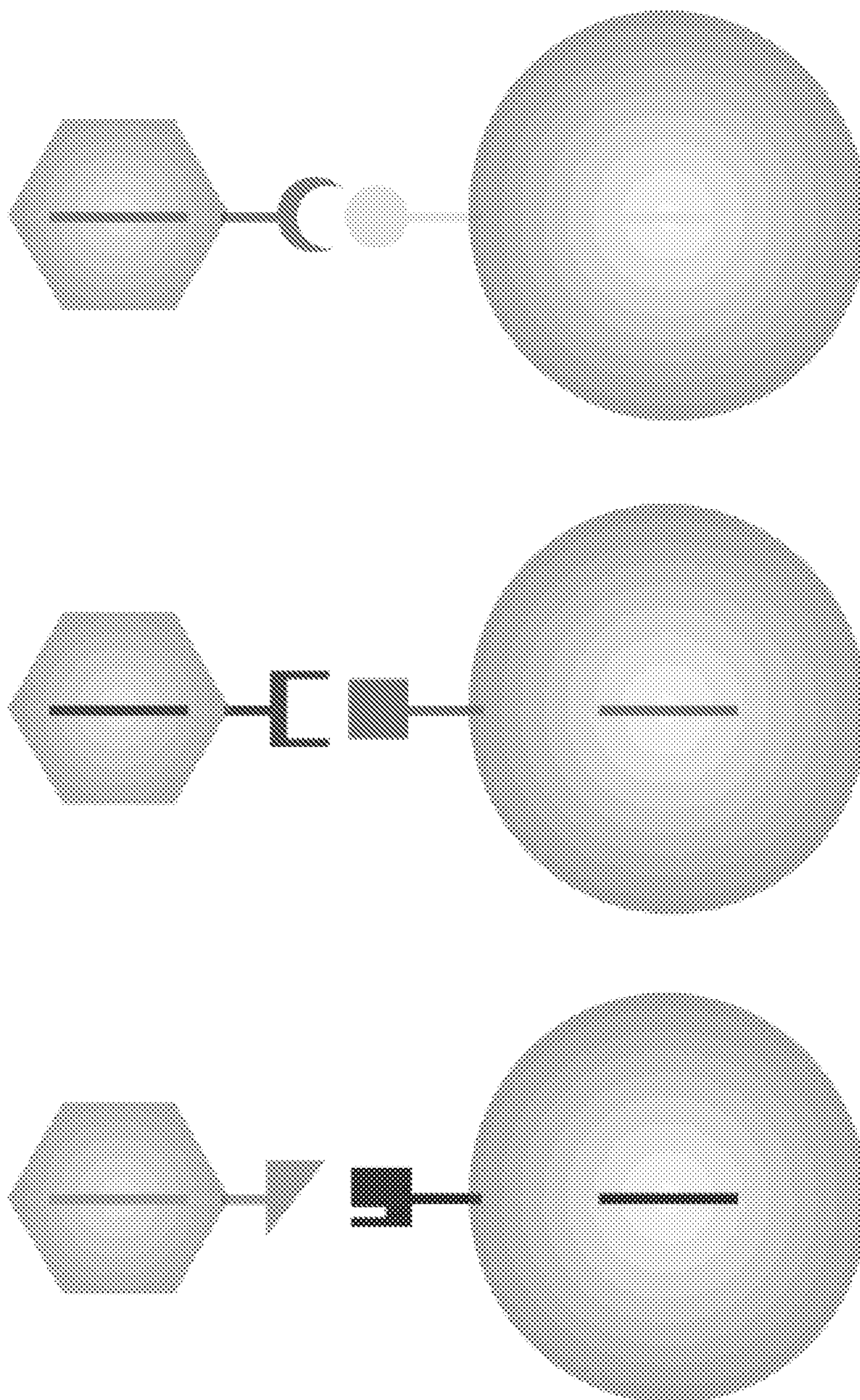


Figure 36

**Figure 37**

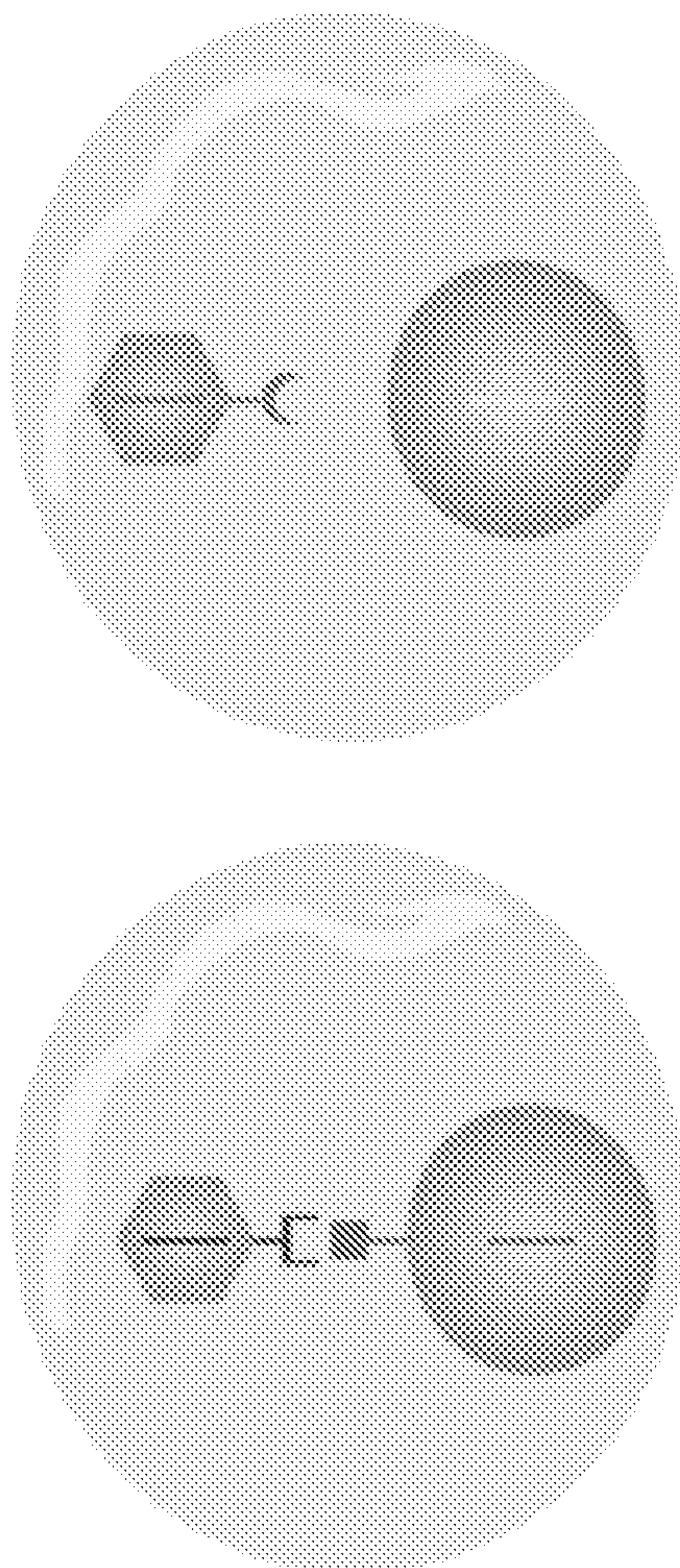


Figure 38

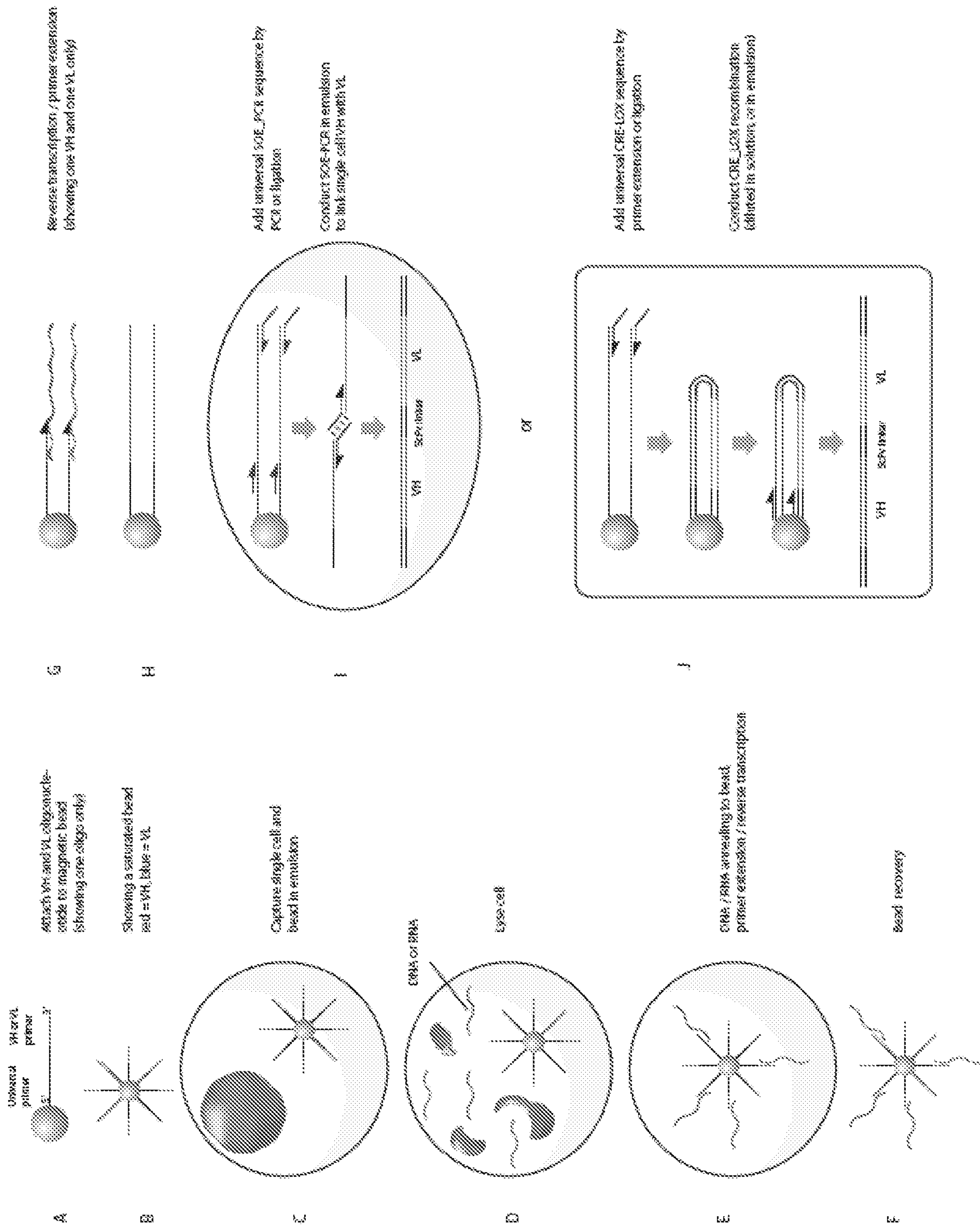


Figure 39

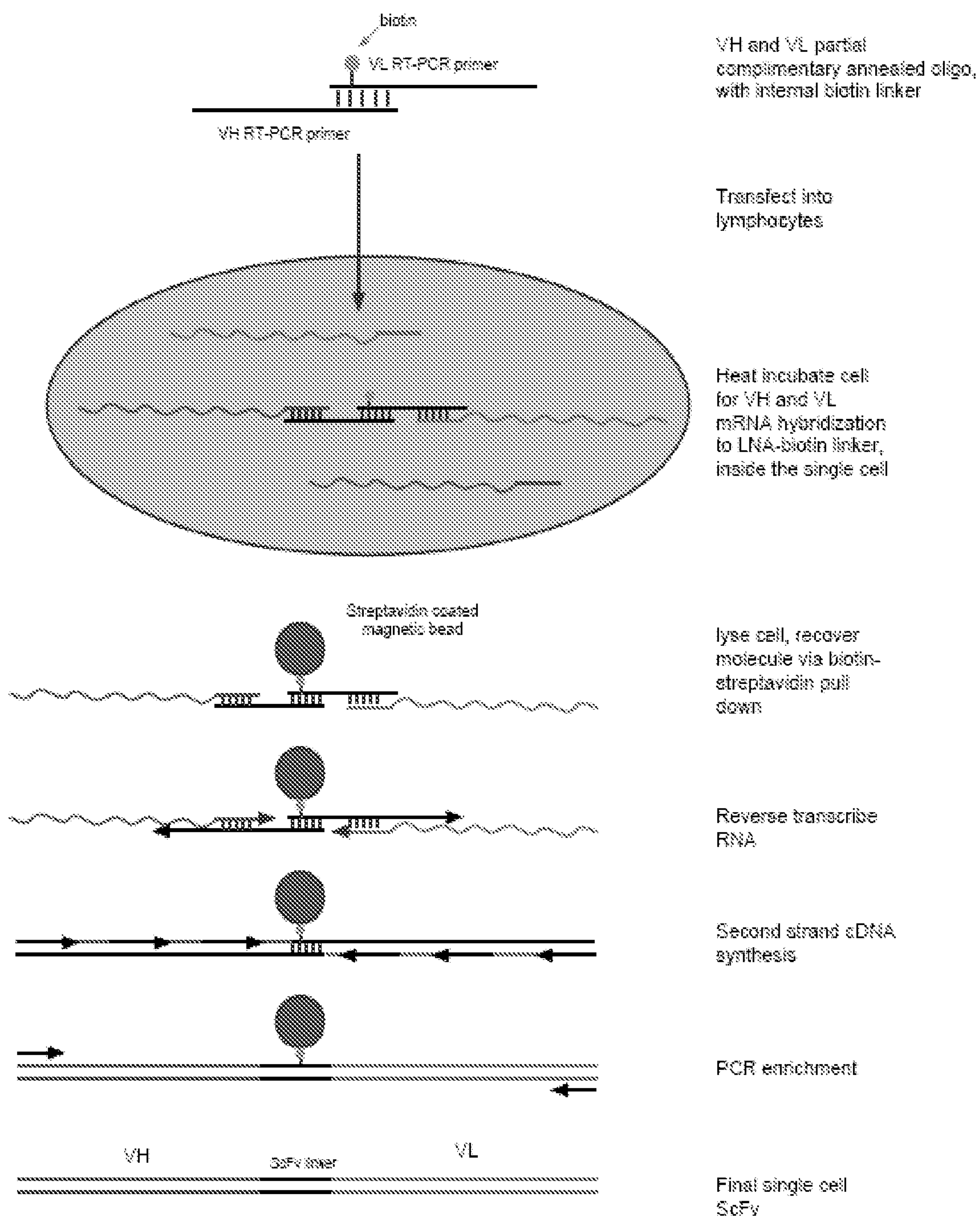


Figure 40

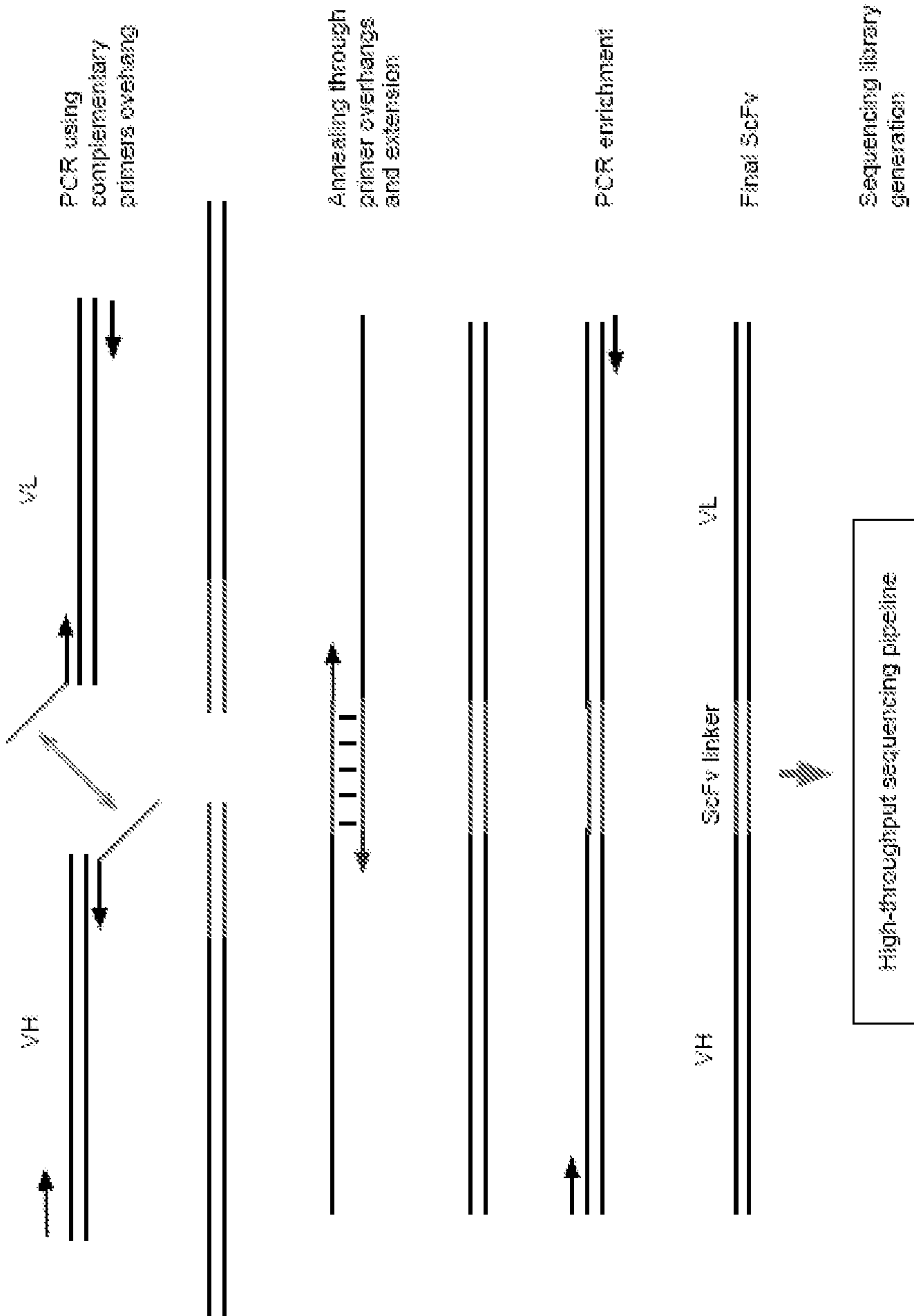


Figure 41

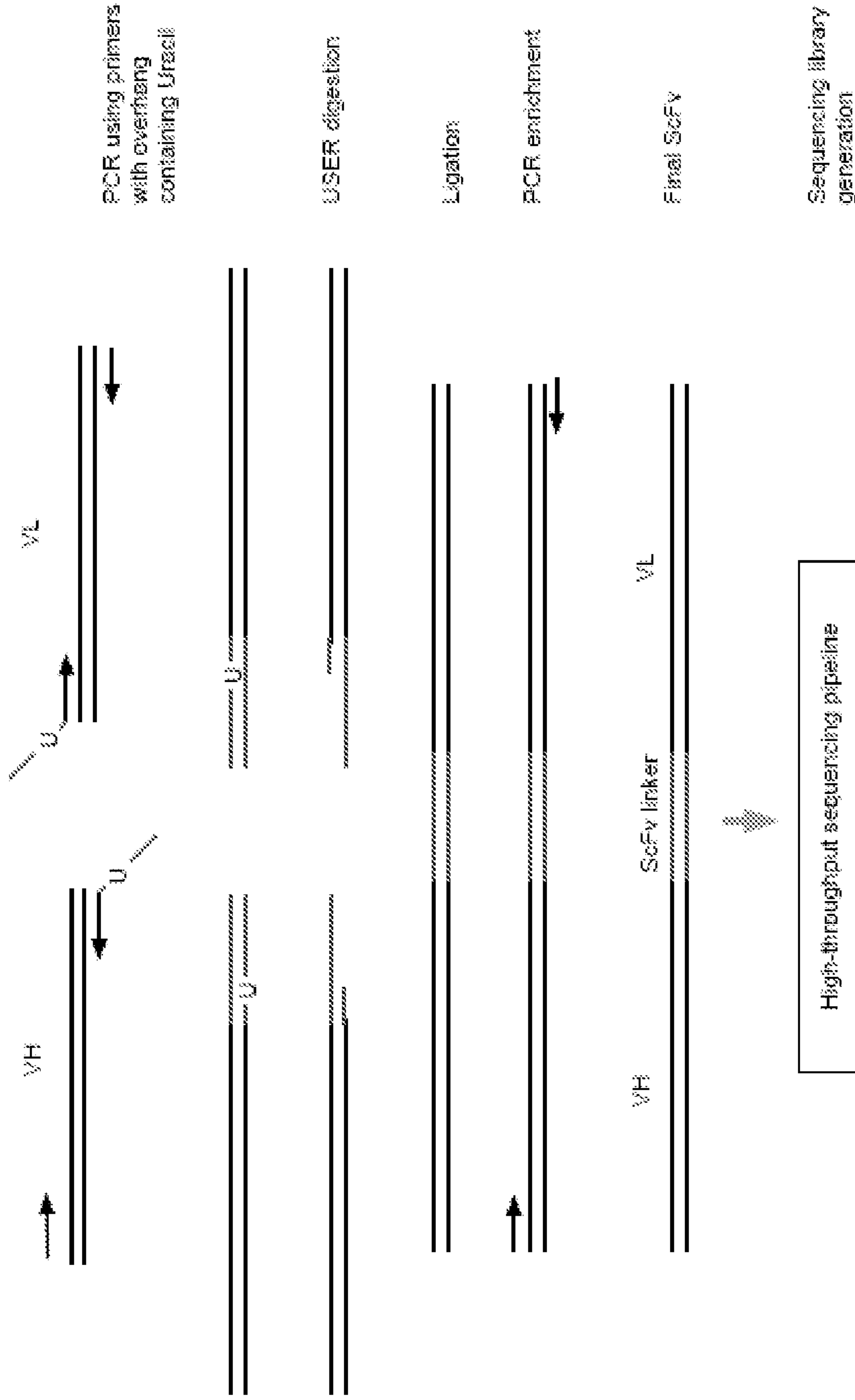


Figure 42

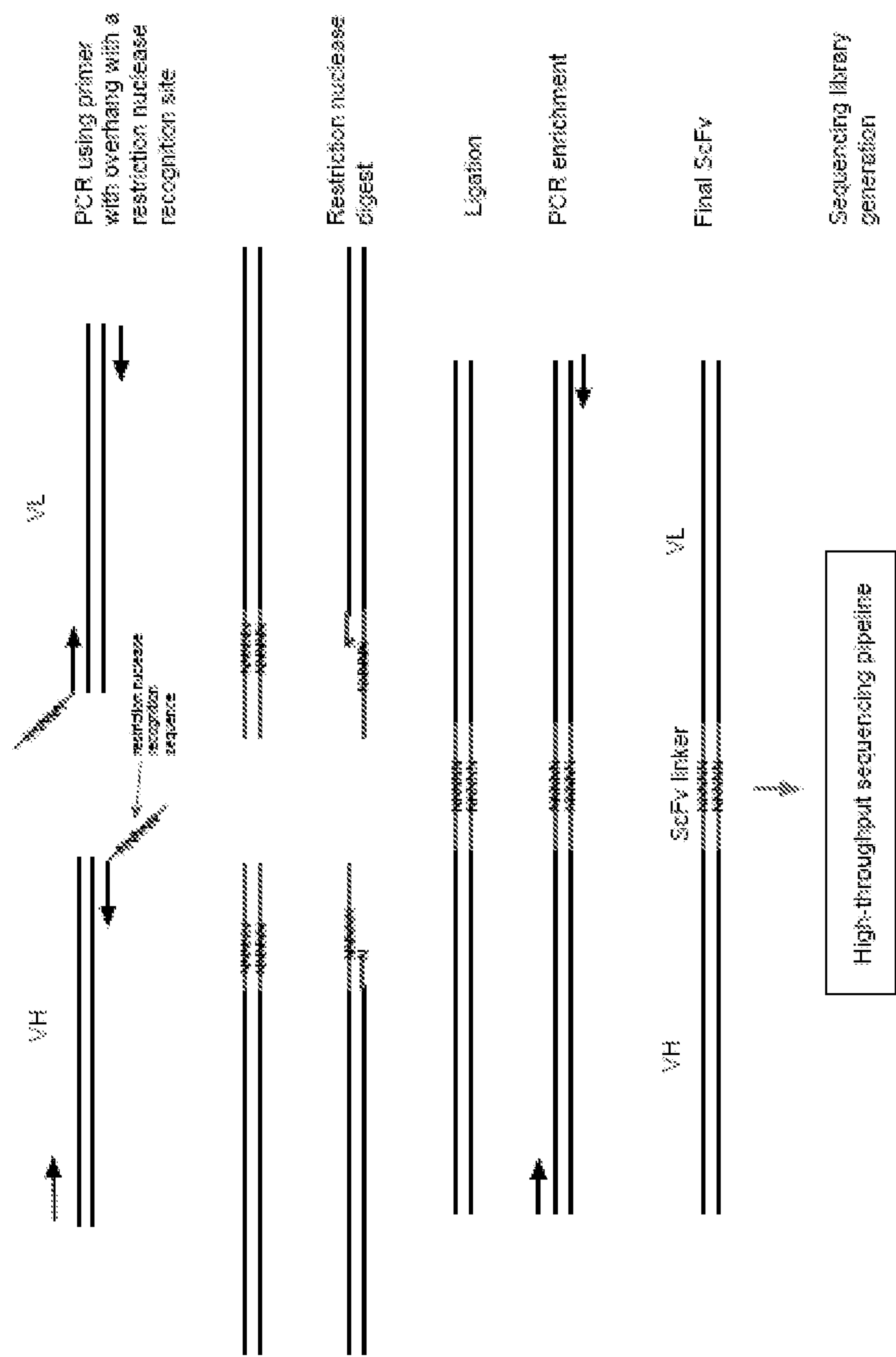


Figure 43

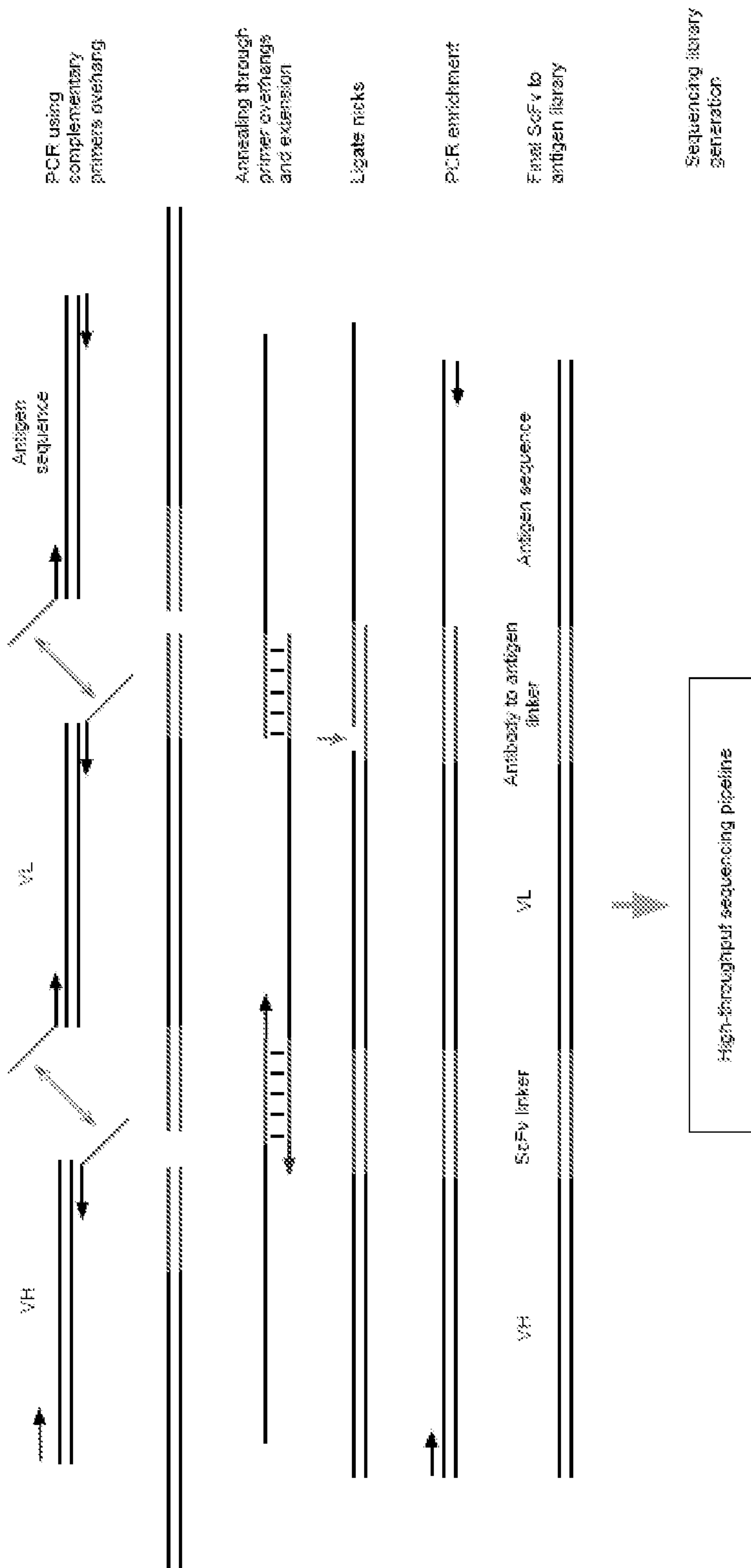


Figure 44

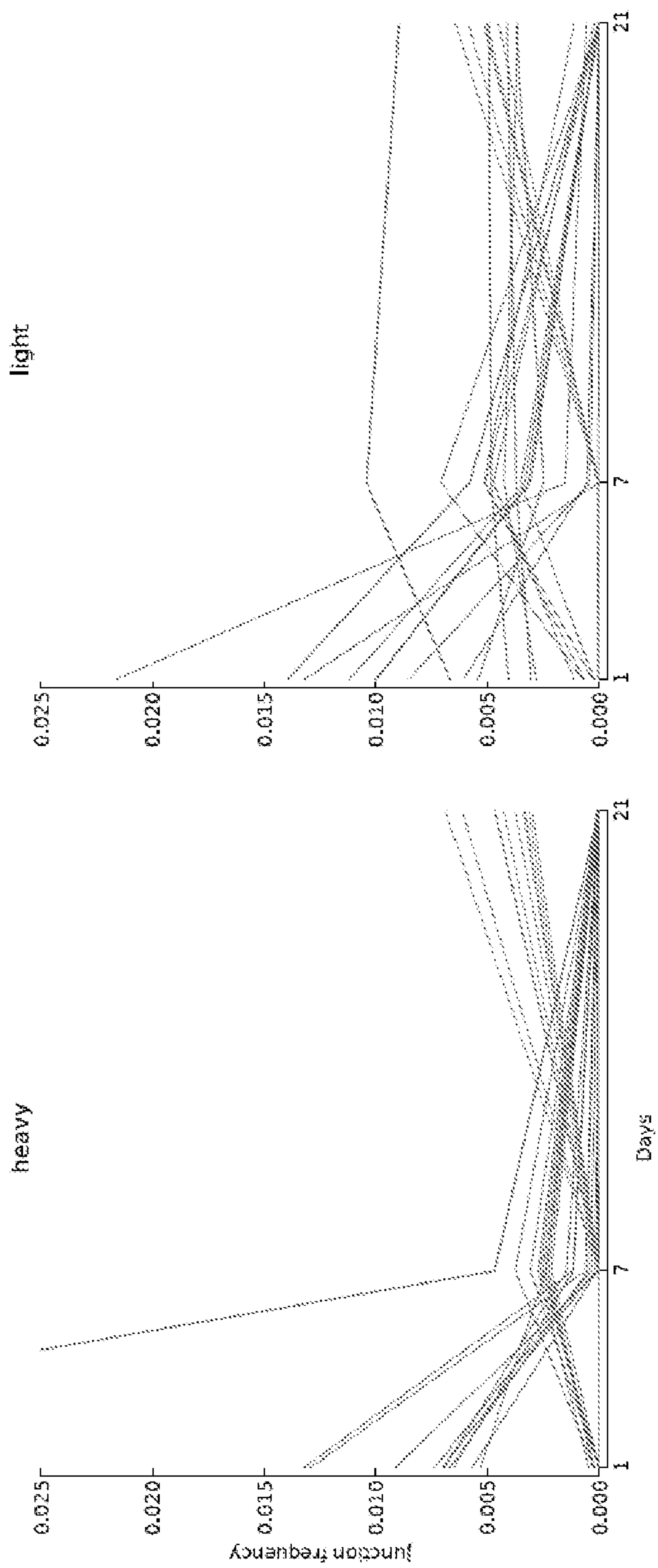
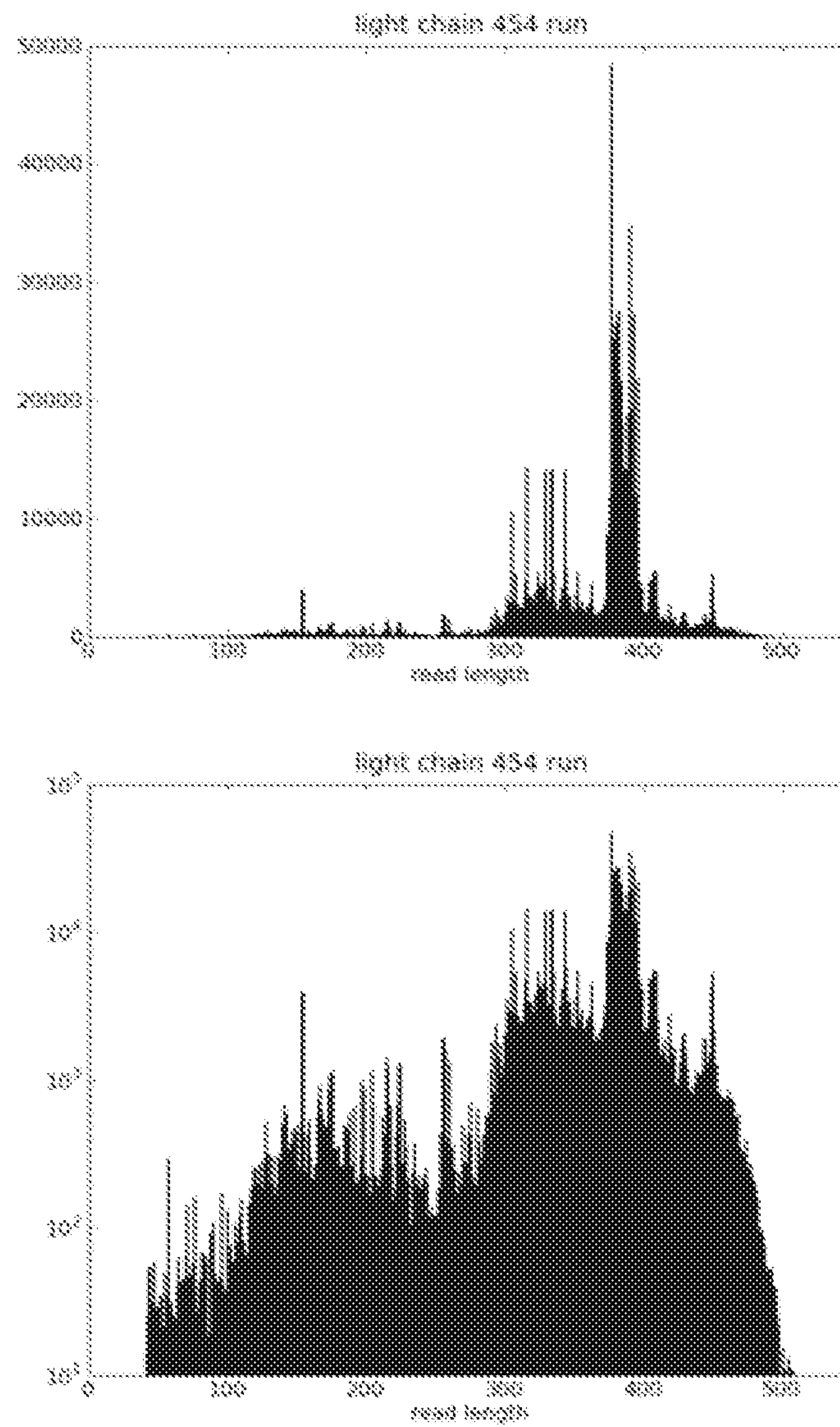
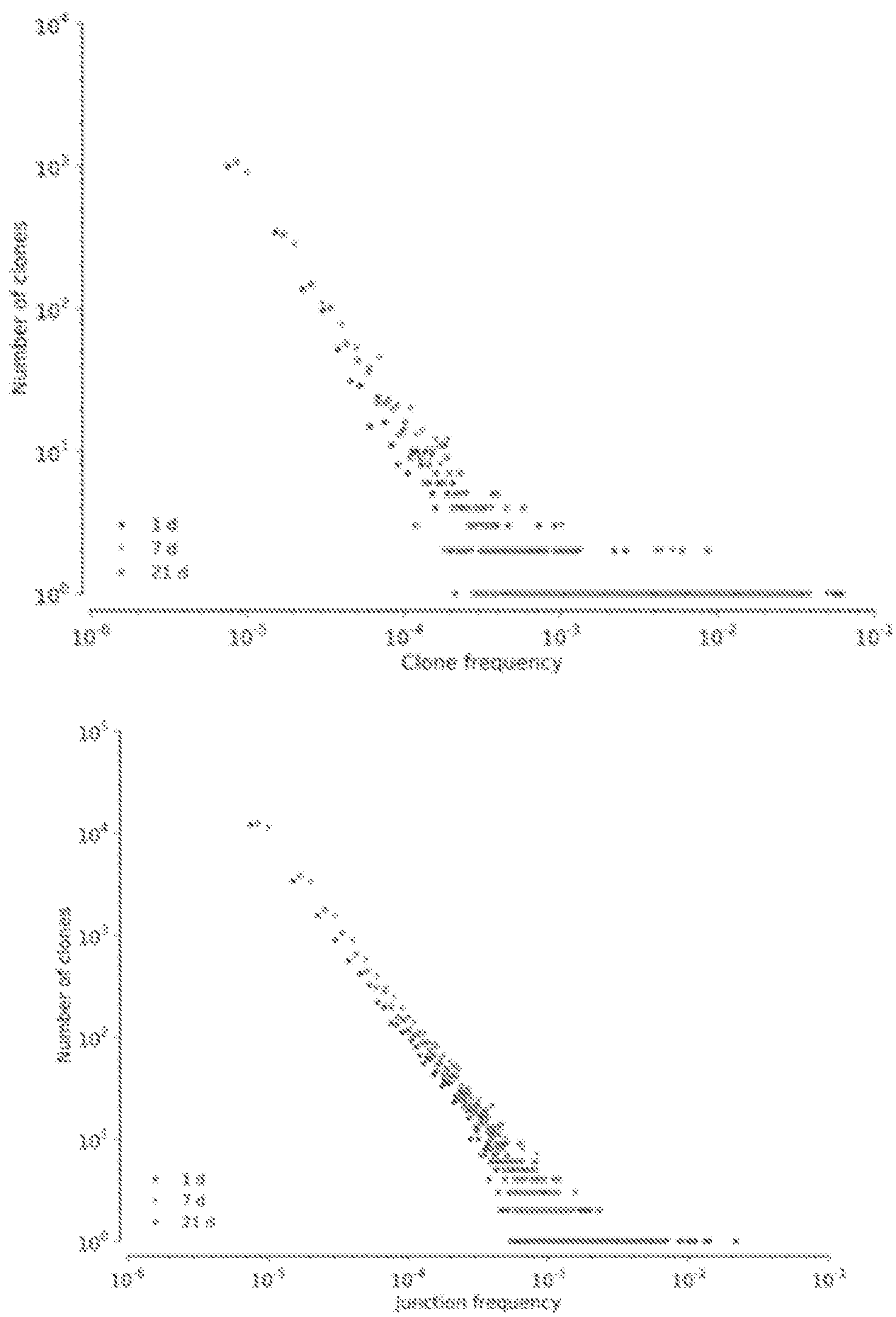
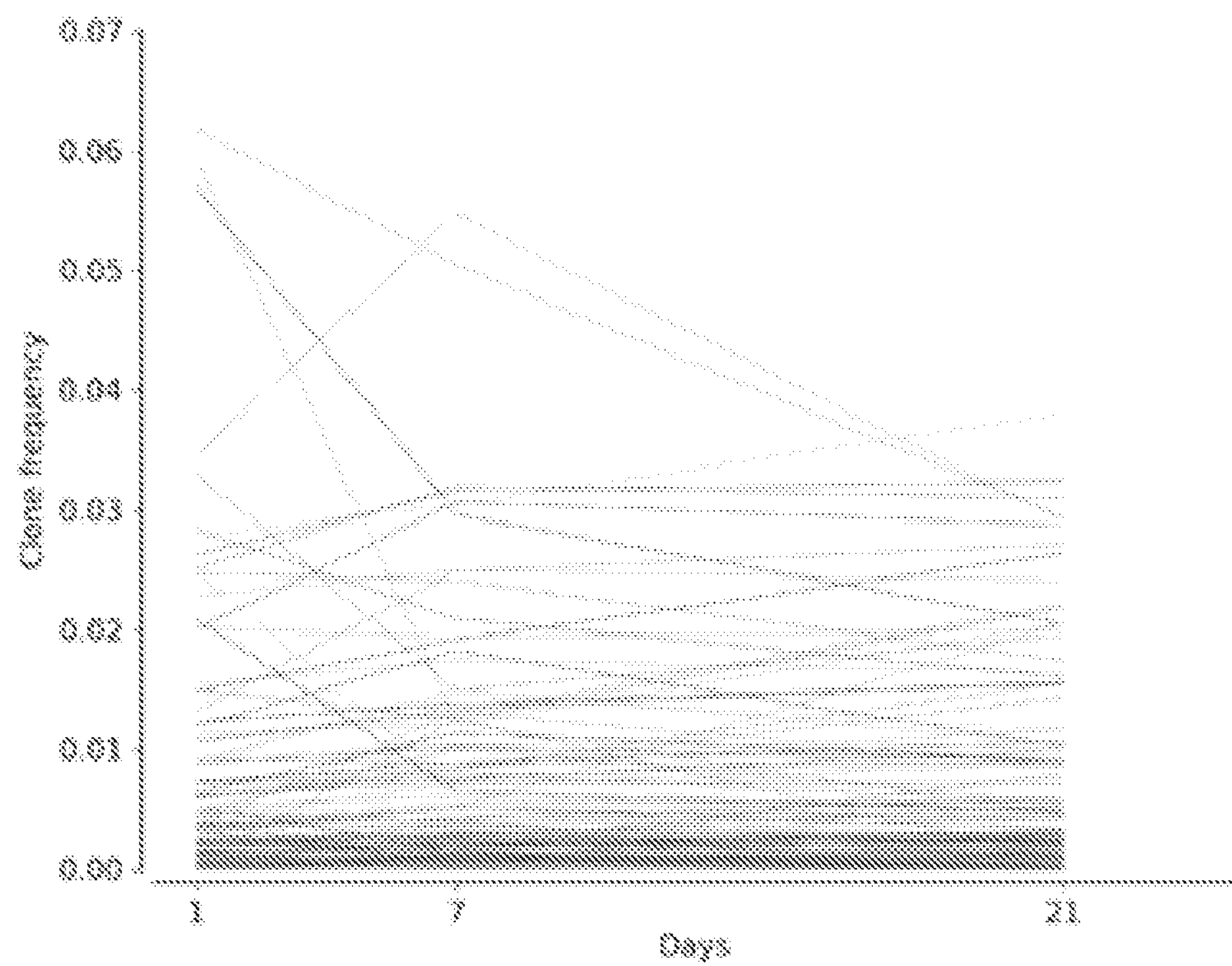
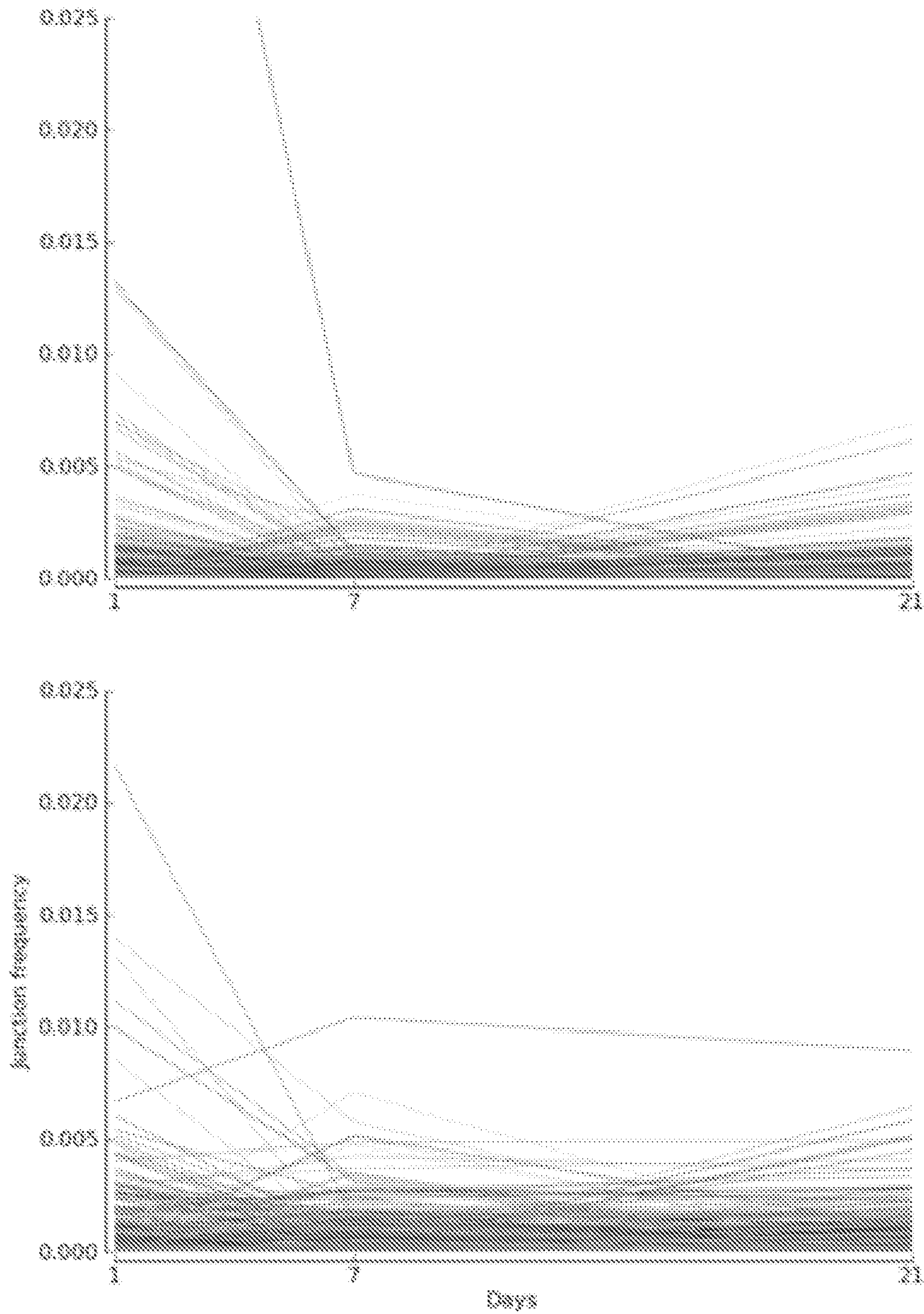


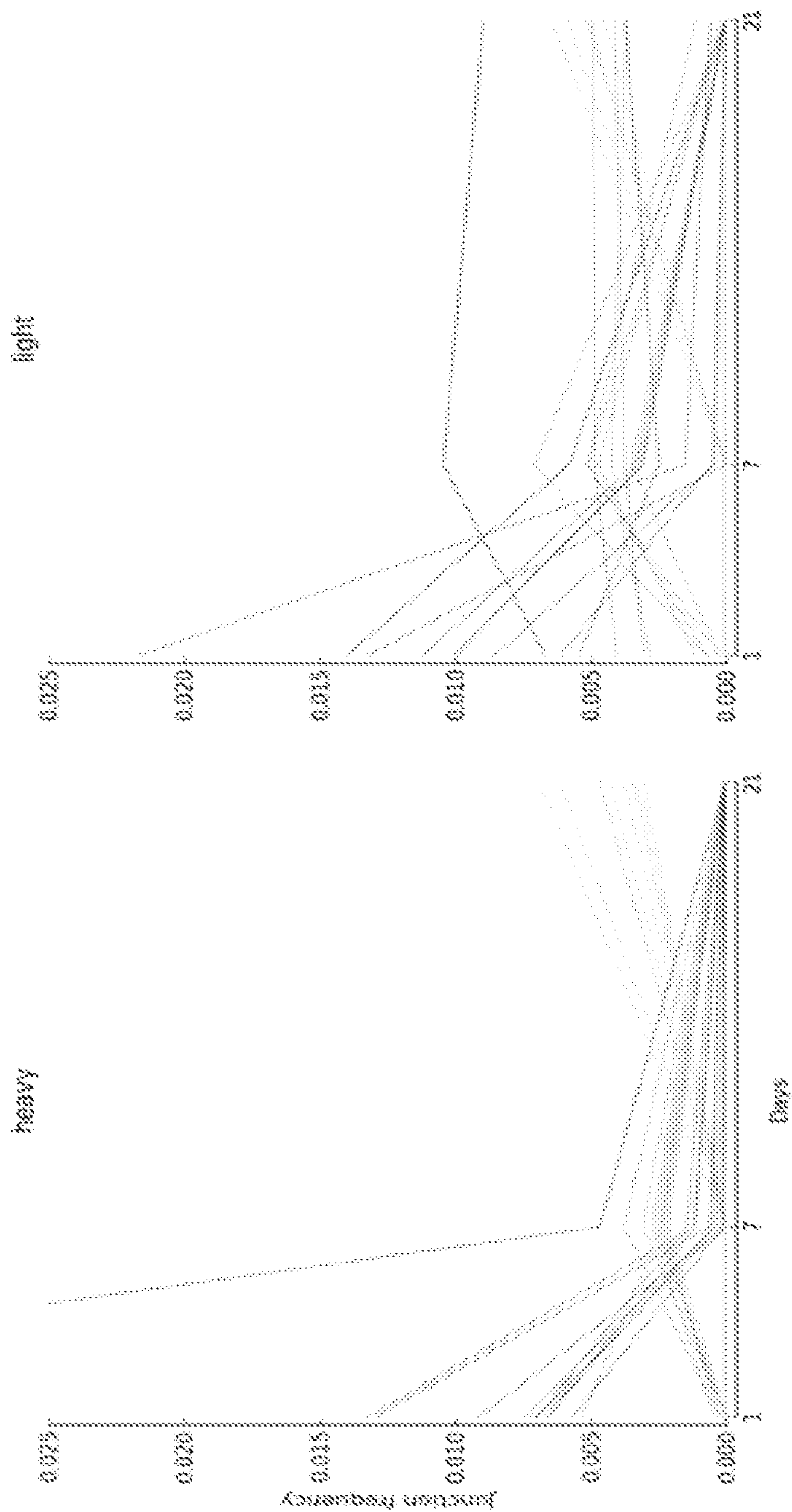
Figure 45

**Figure 46**

**Figure 47**

**Figure 48**

**Figure 49**

**Figure 50**

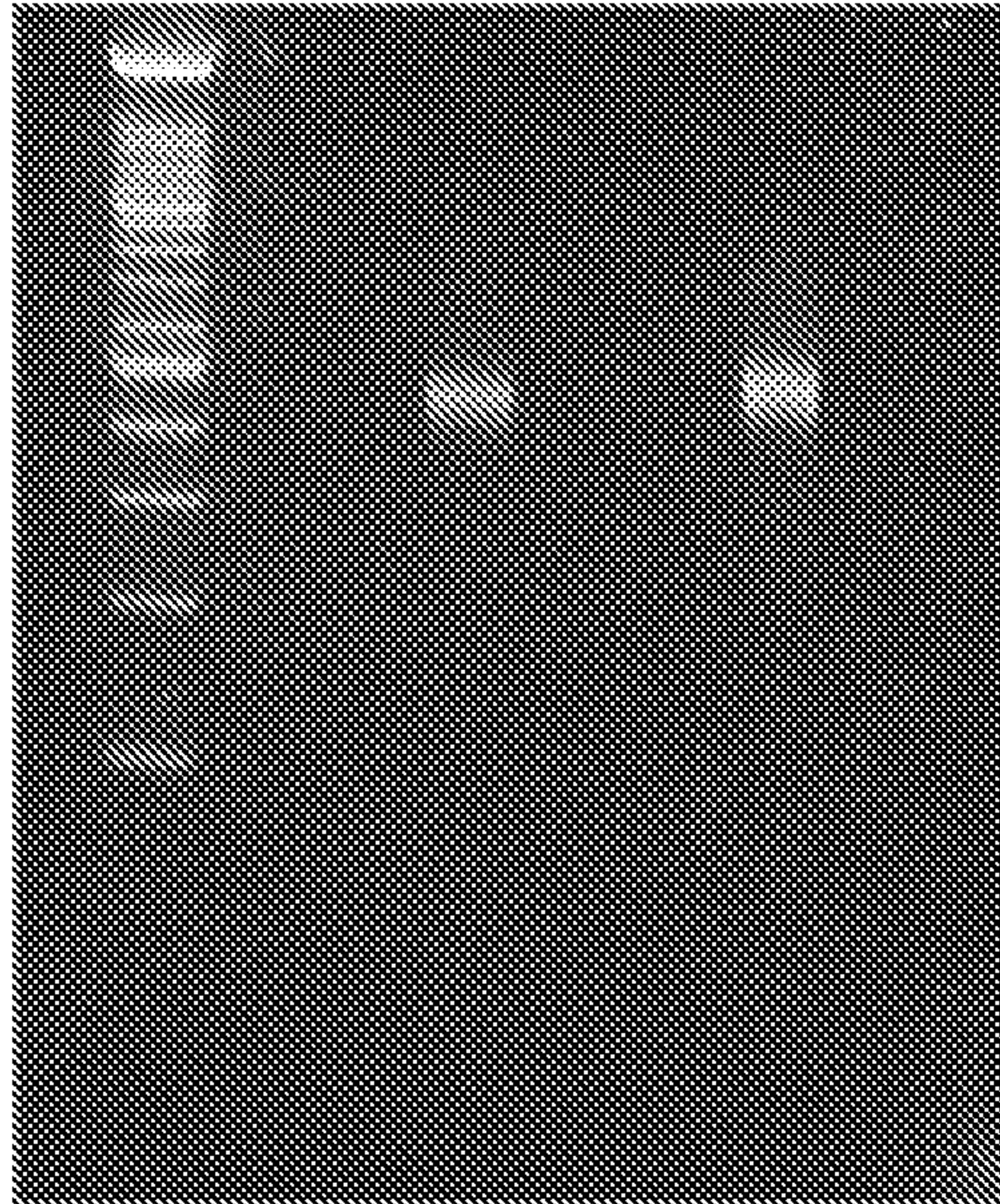


Figure 51

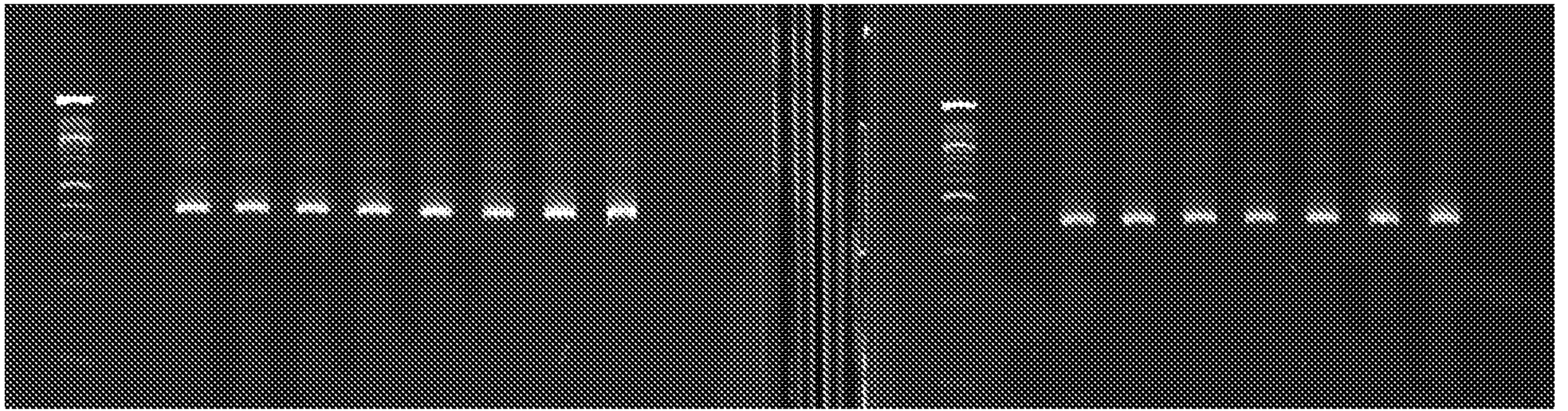


Figure 52

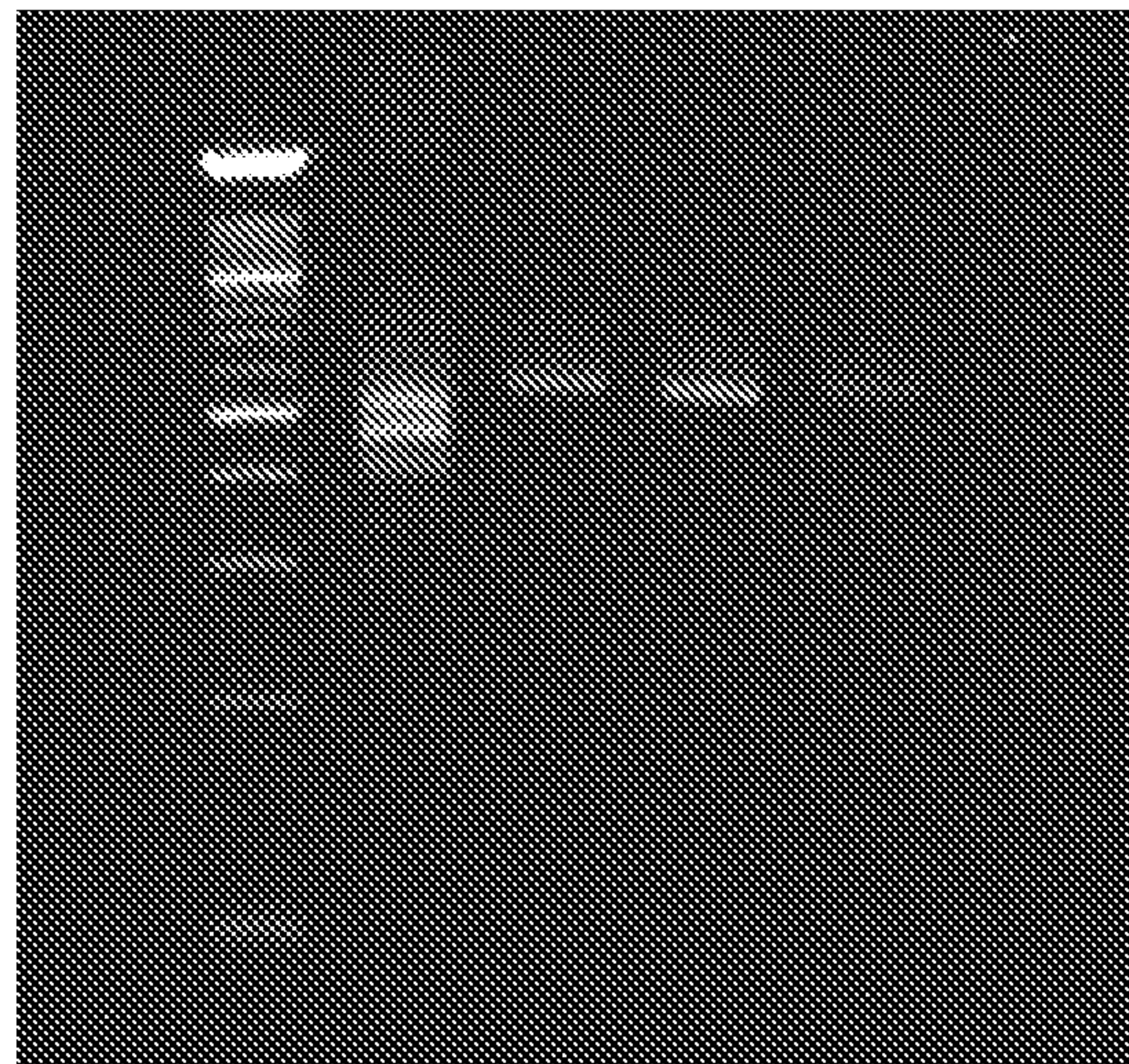


Figure 53

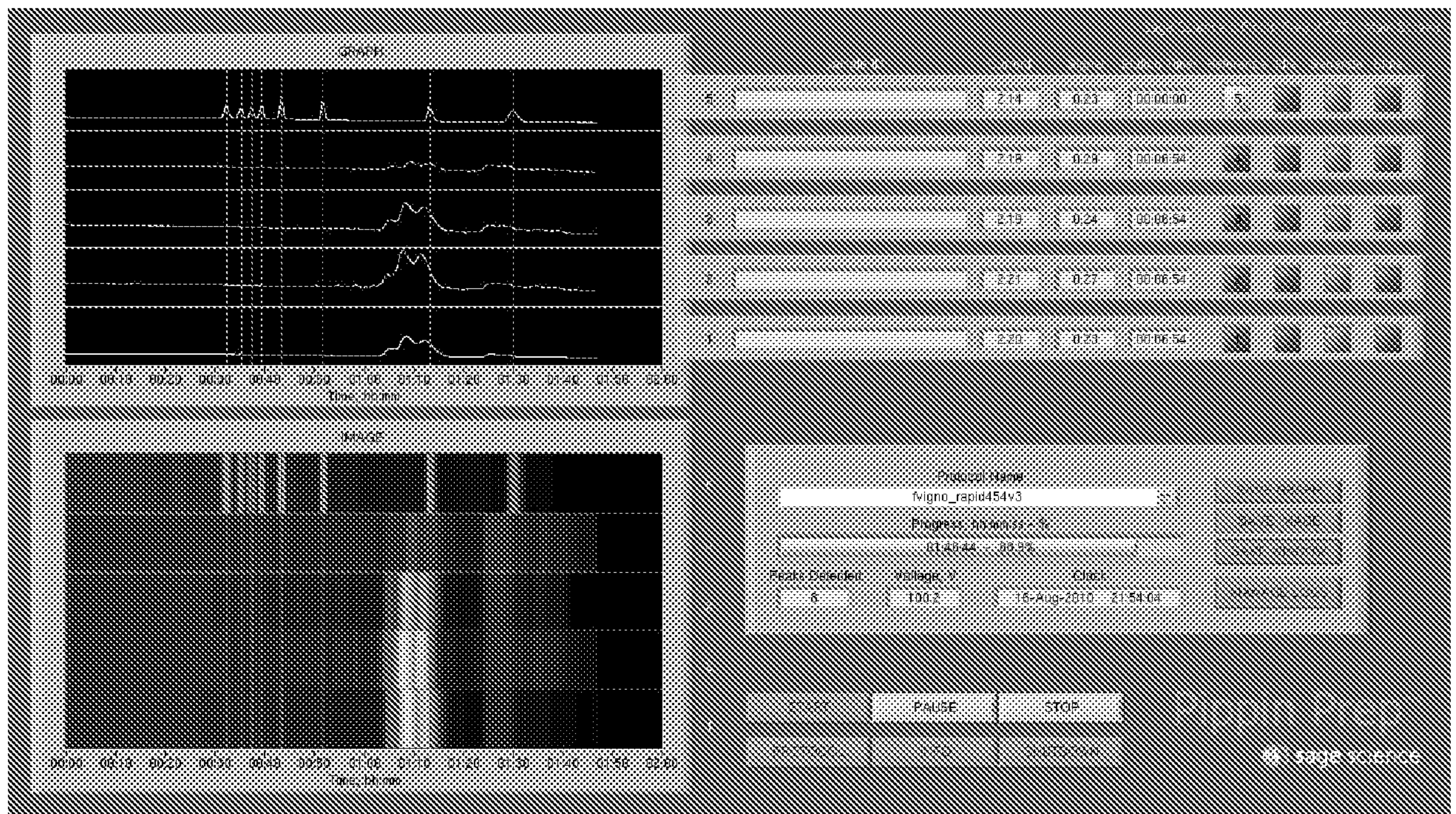
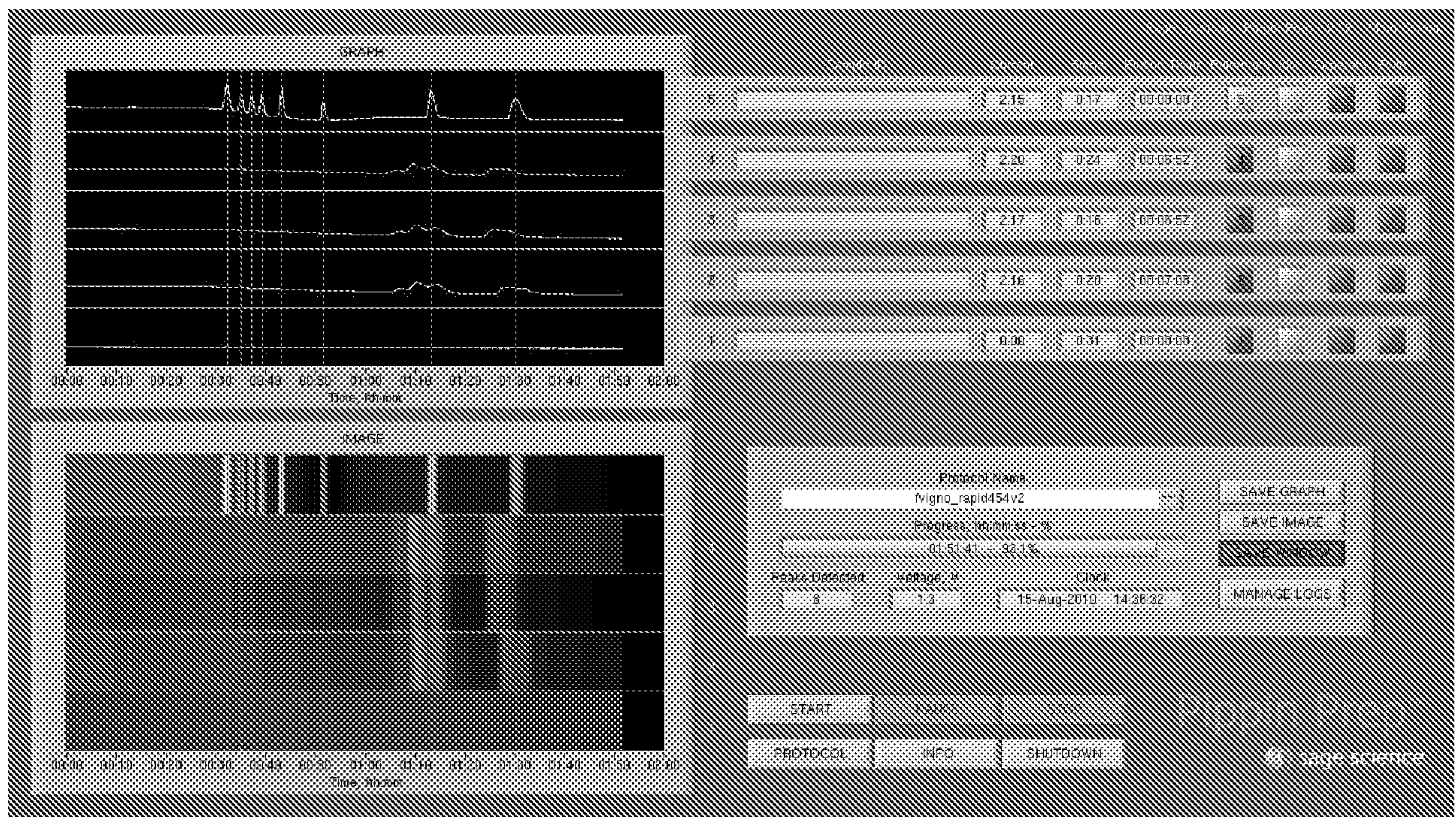


Figure 54

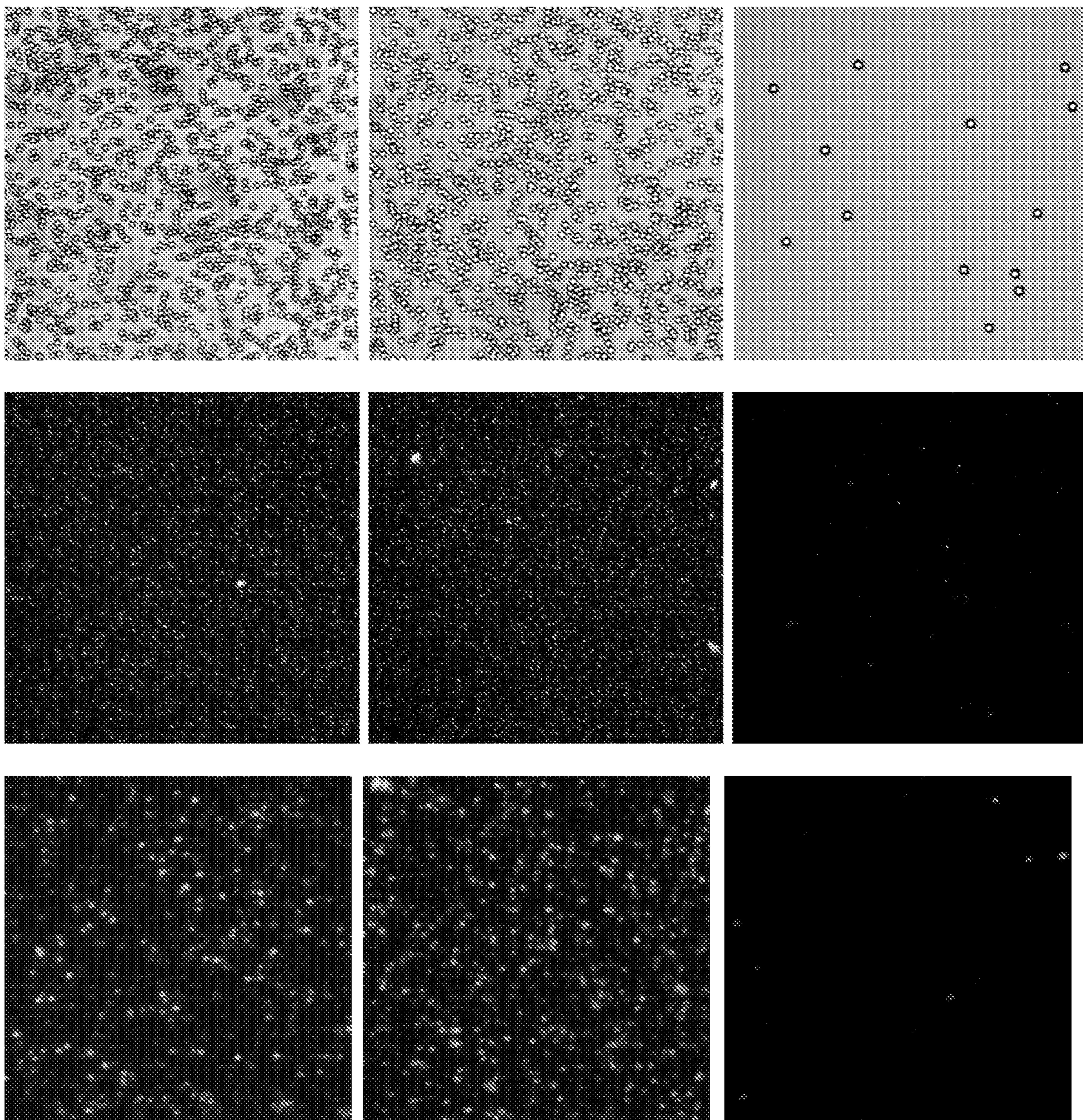


Figure 55

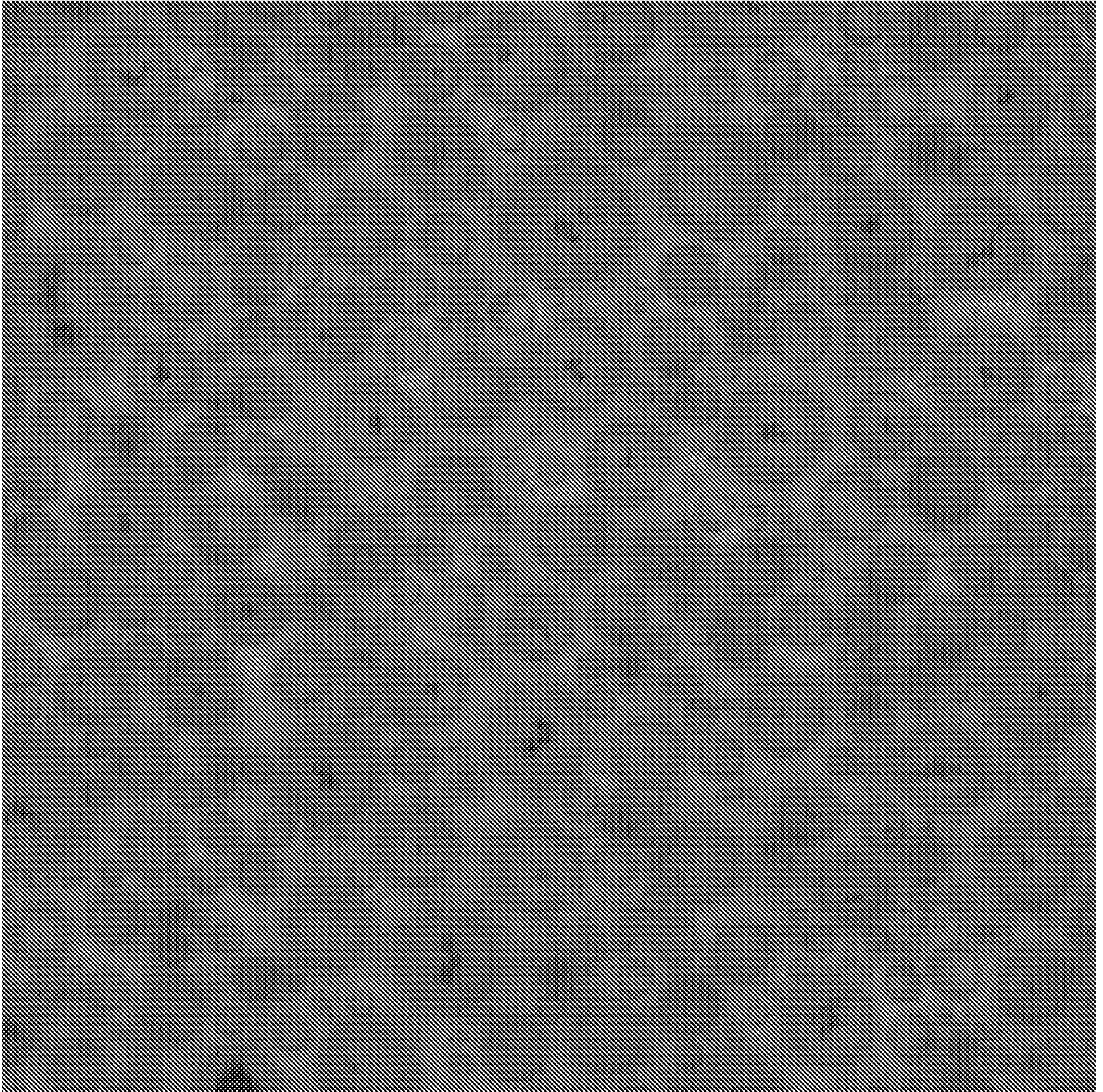


Figure 56

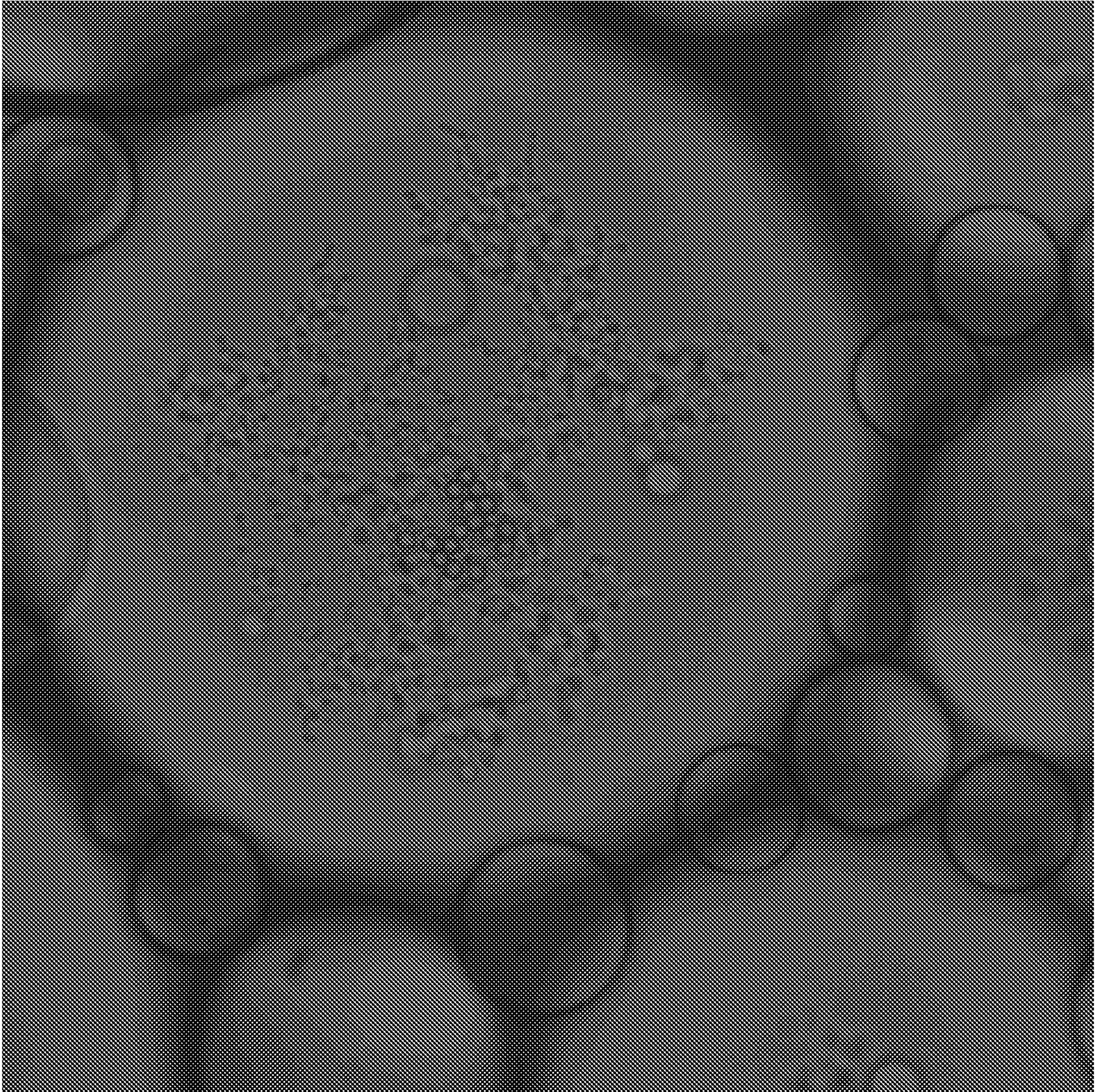


Figure 57

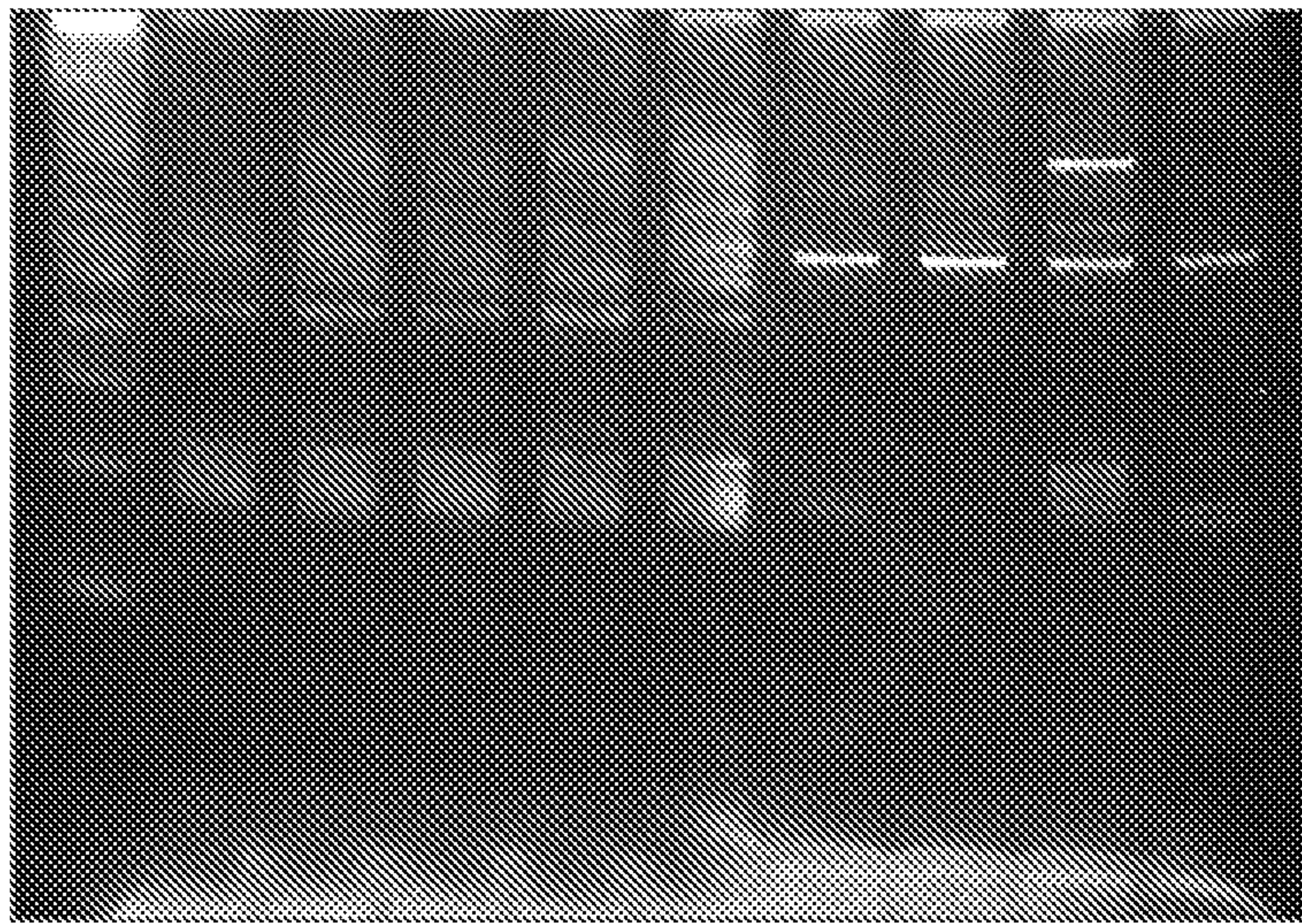
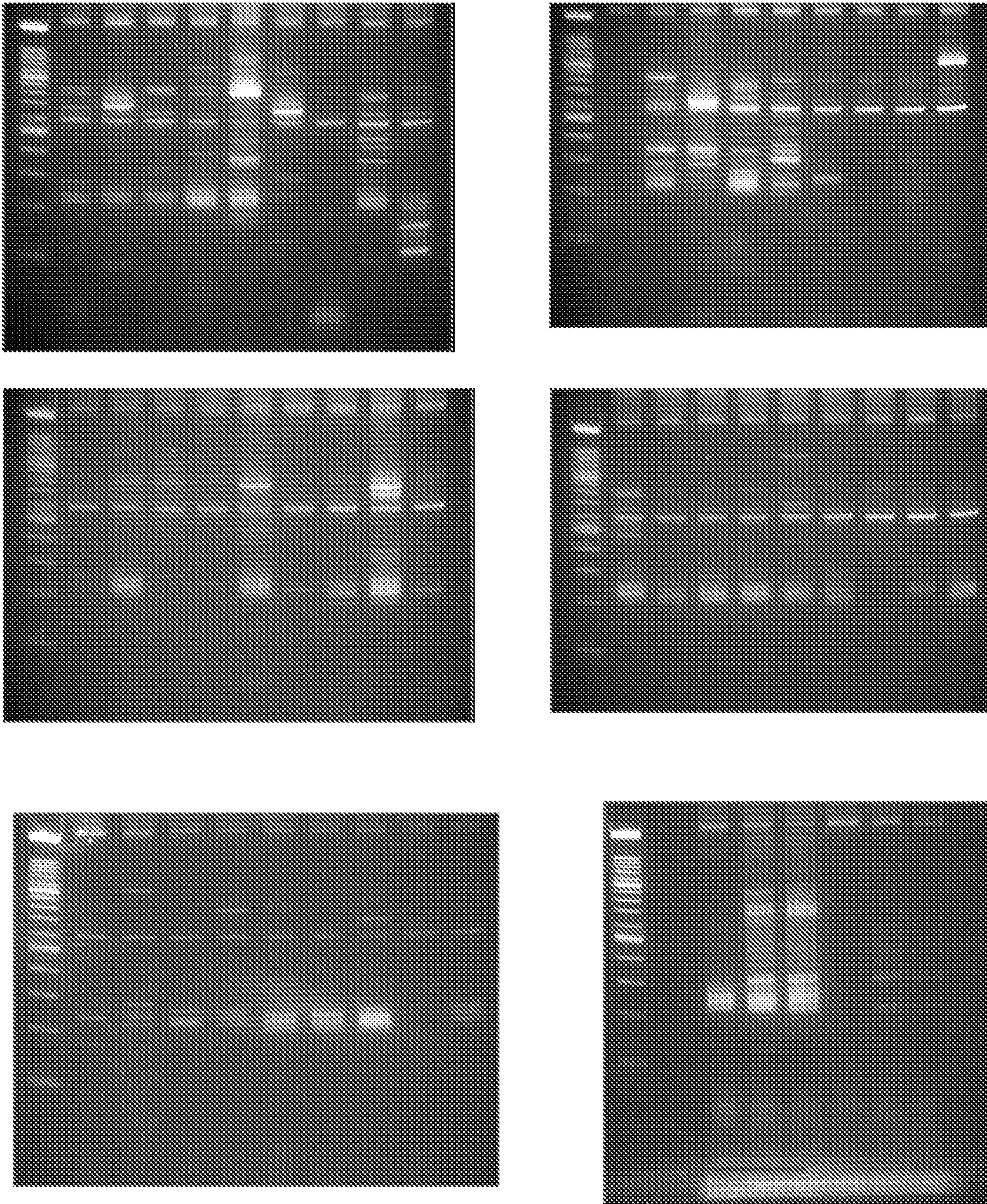


Figure 58

**Figure 59**

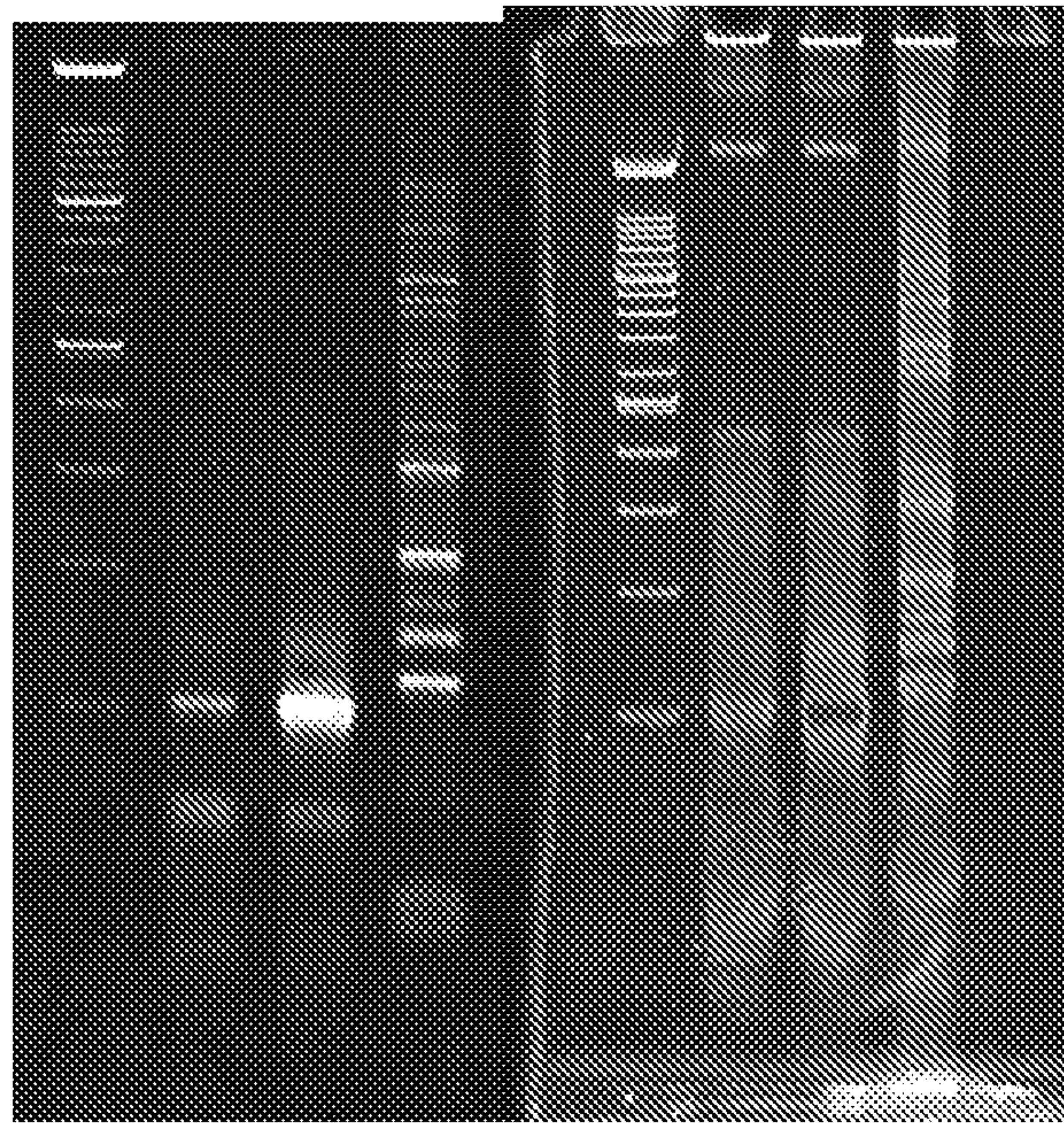


Figure 60

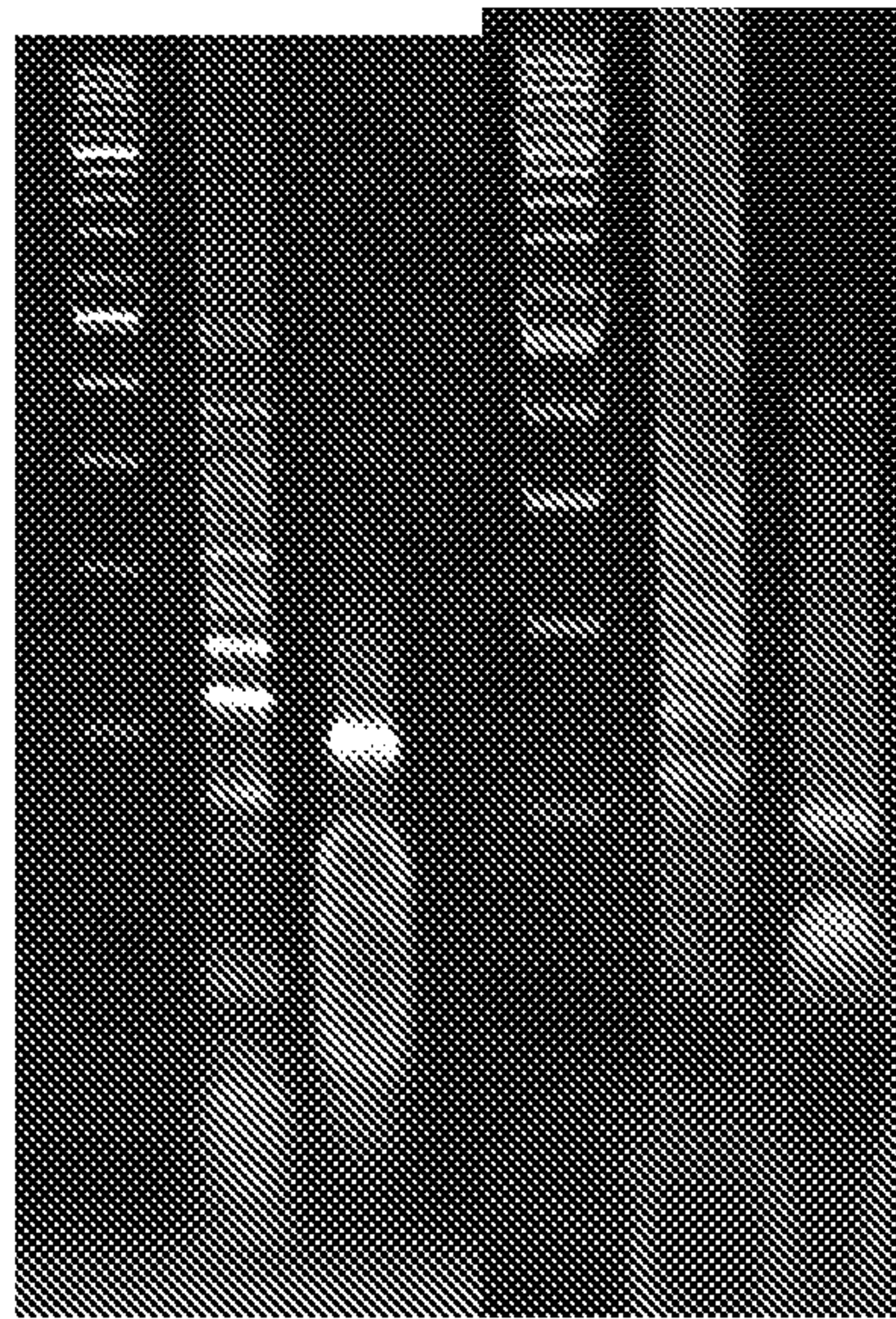


Figure 62

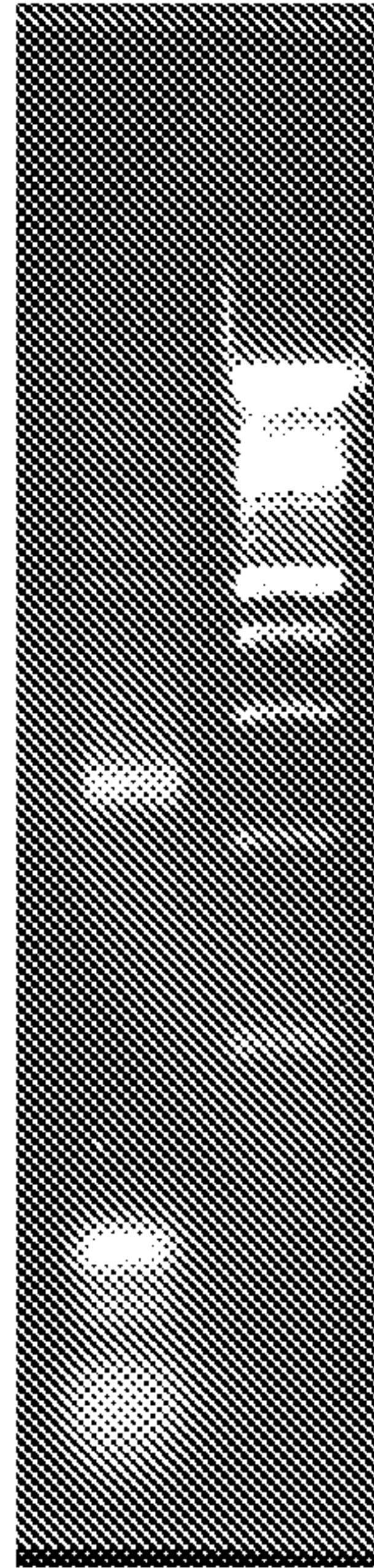


Figure 62

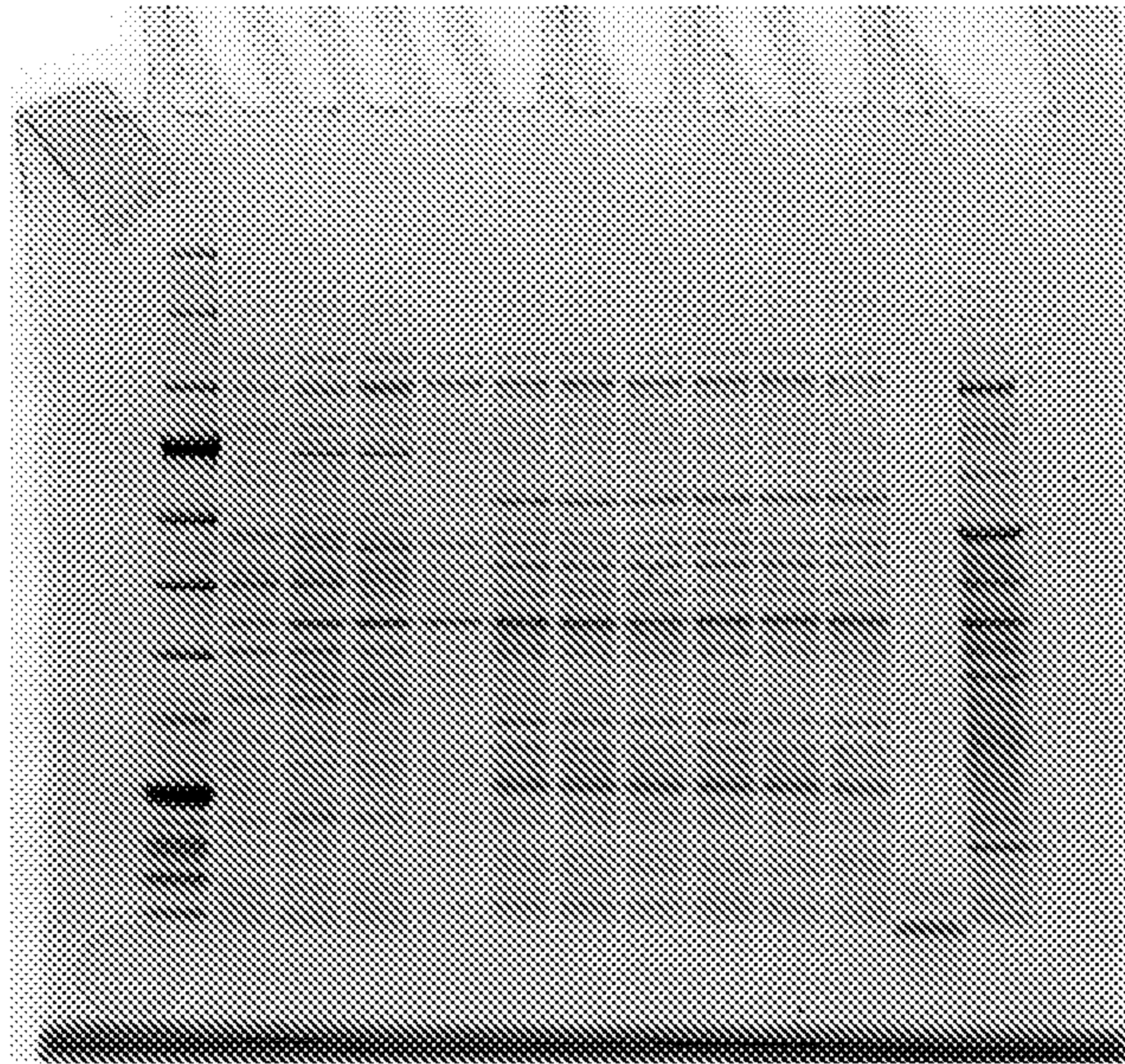
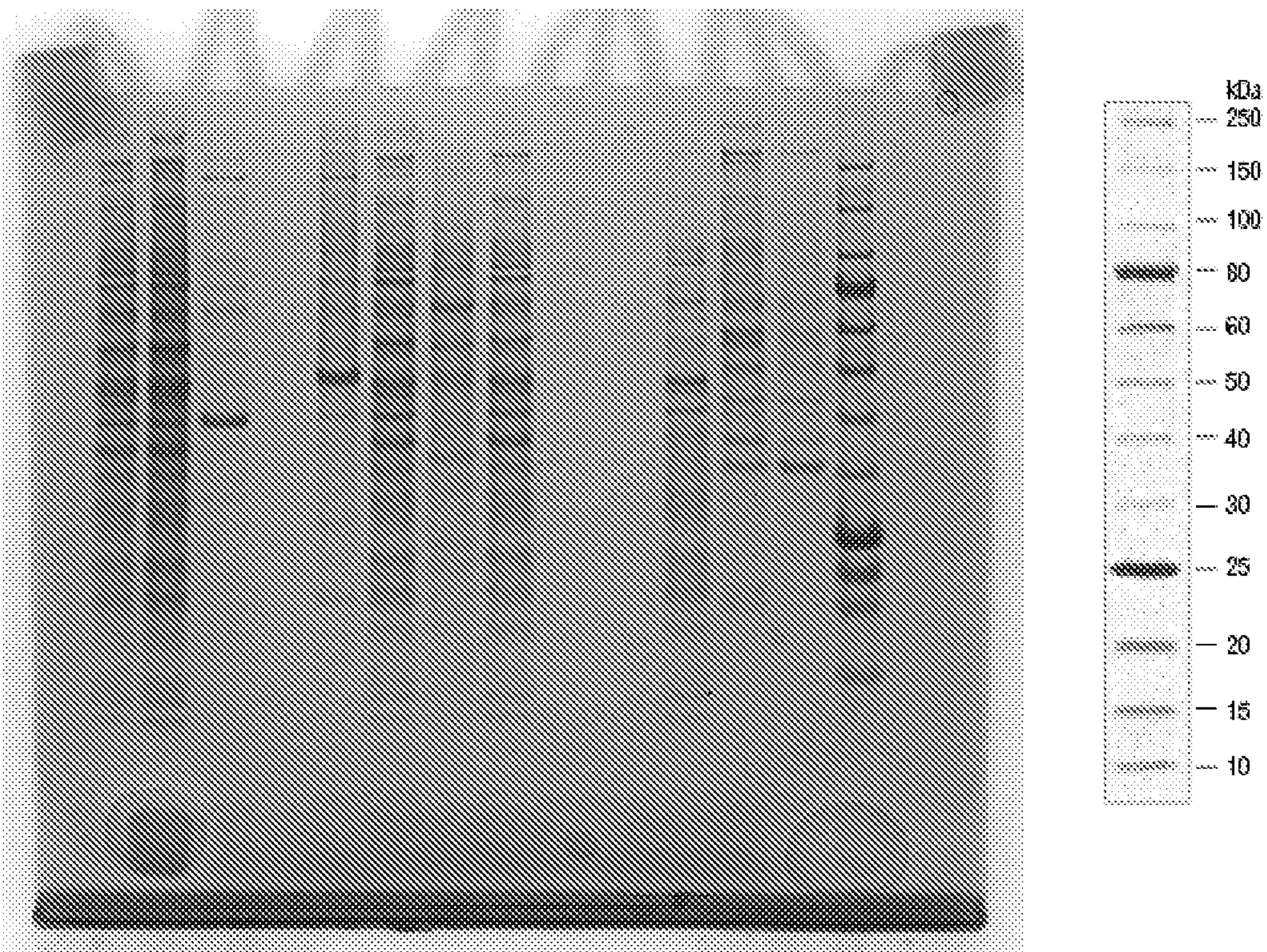
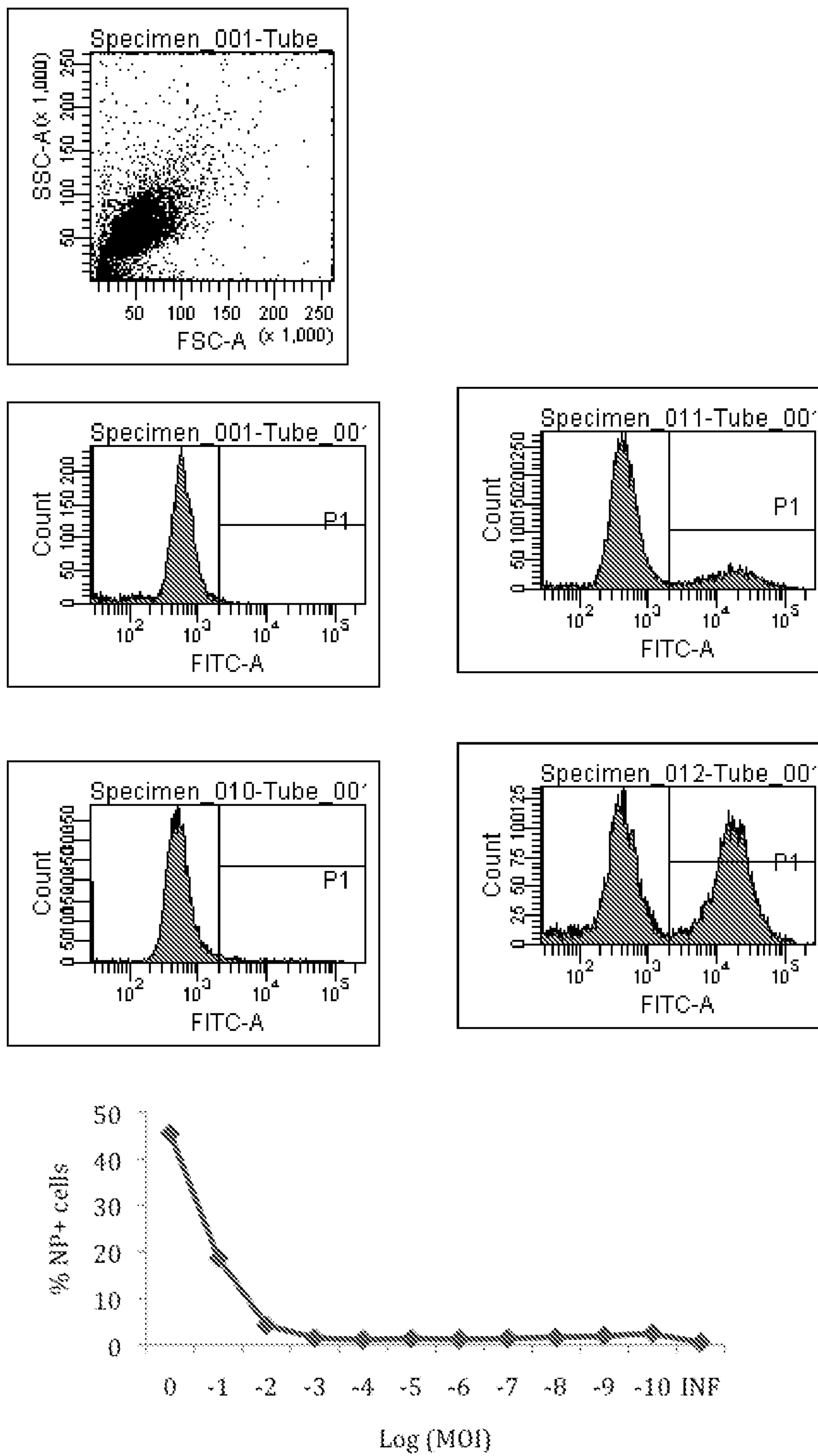
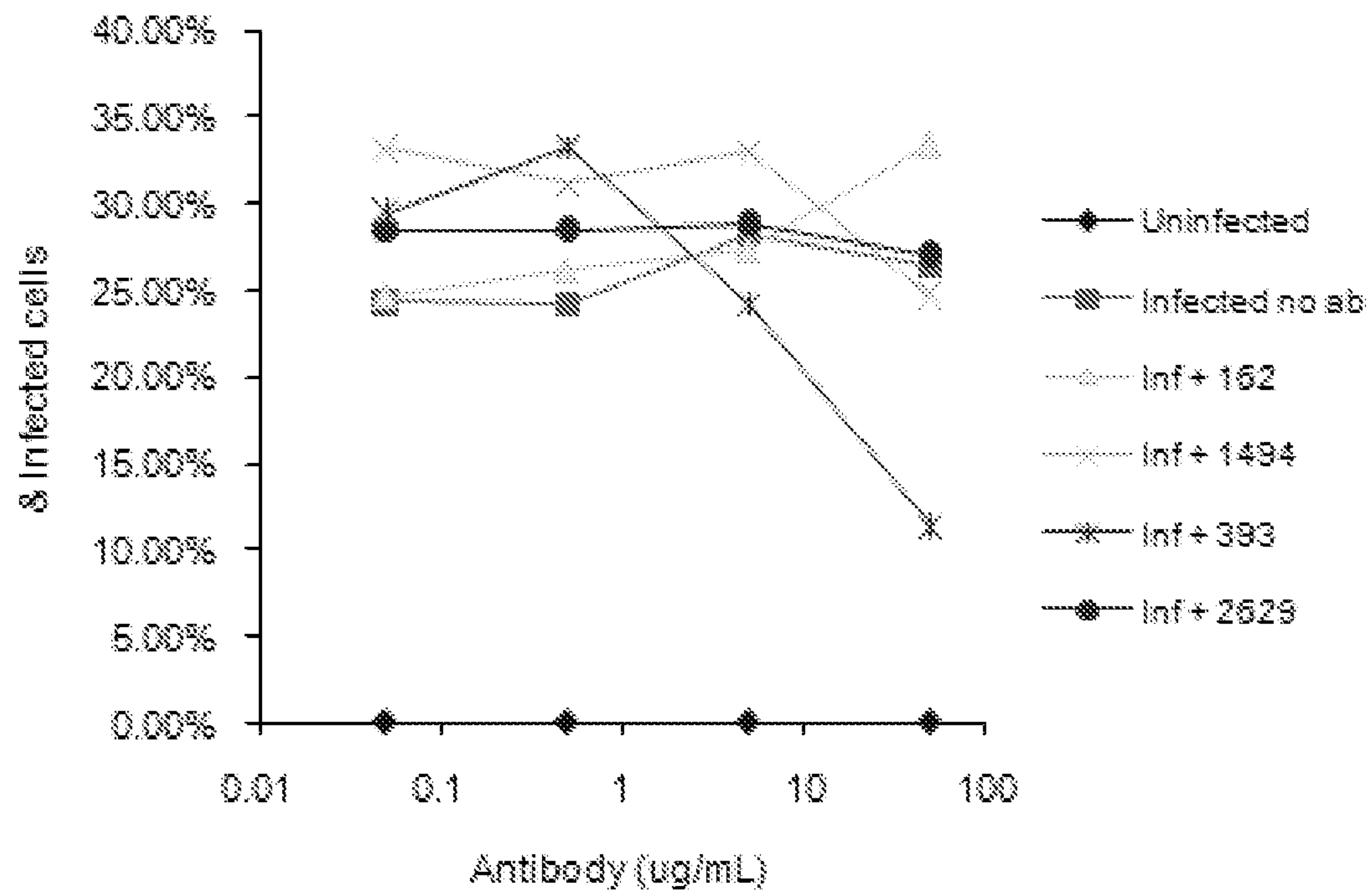
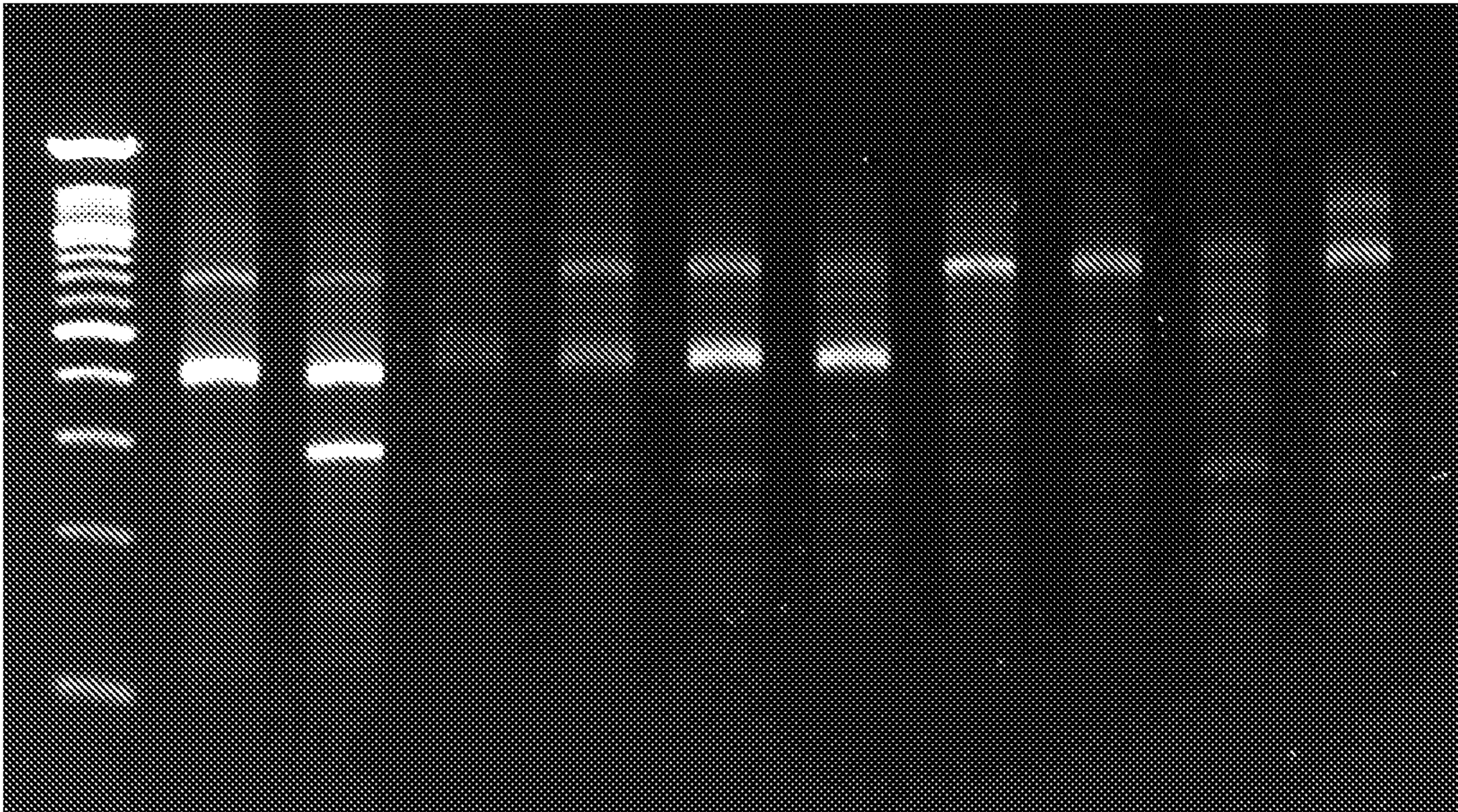


Figure 63

**Figure 65**

**Figure 66**

**Figure 67**

**Figure 68**

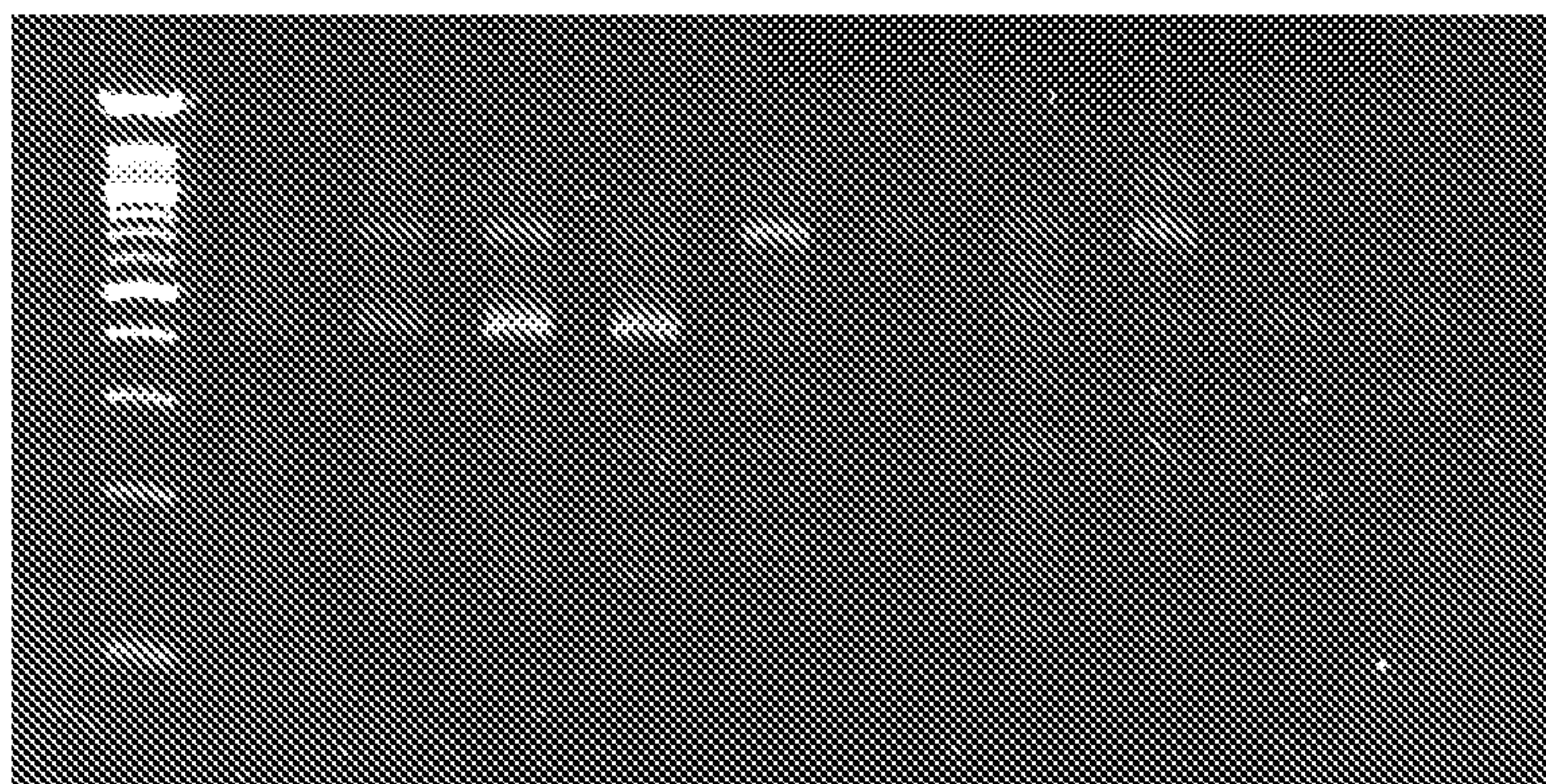


Figure 69

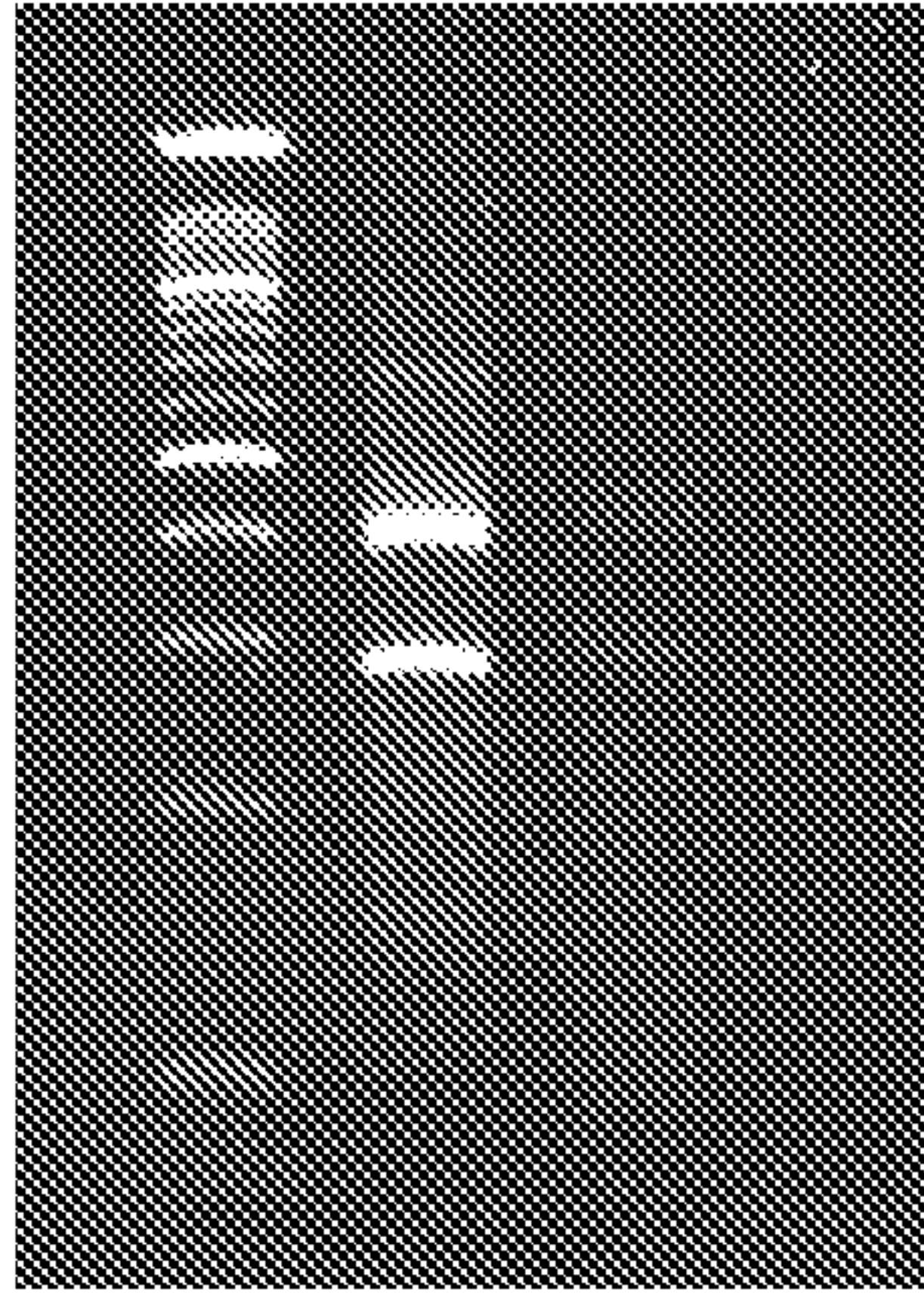


Figure 70

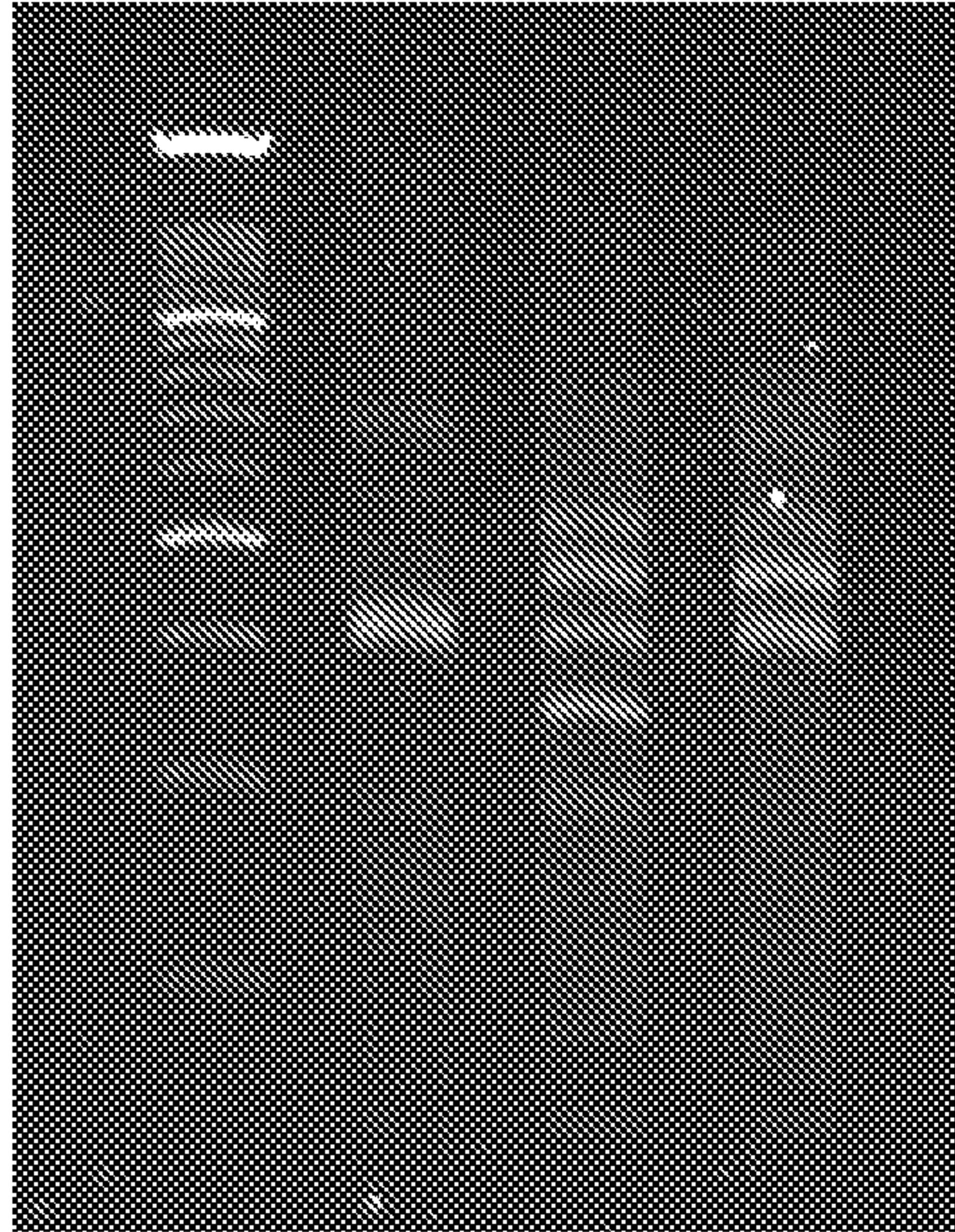


Figure 71

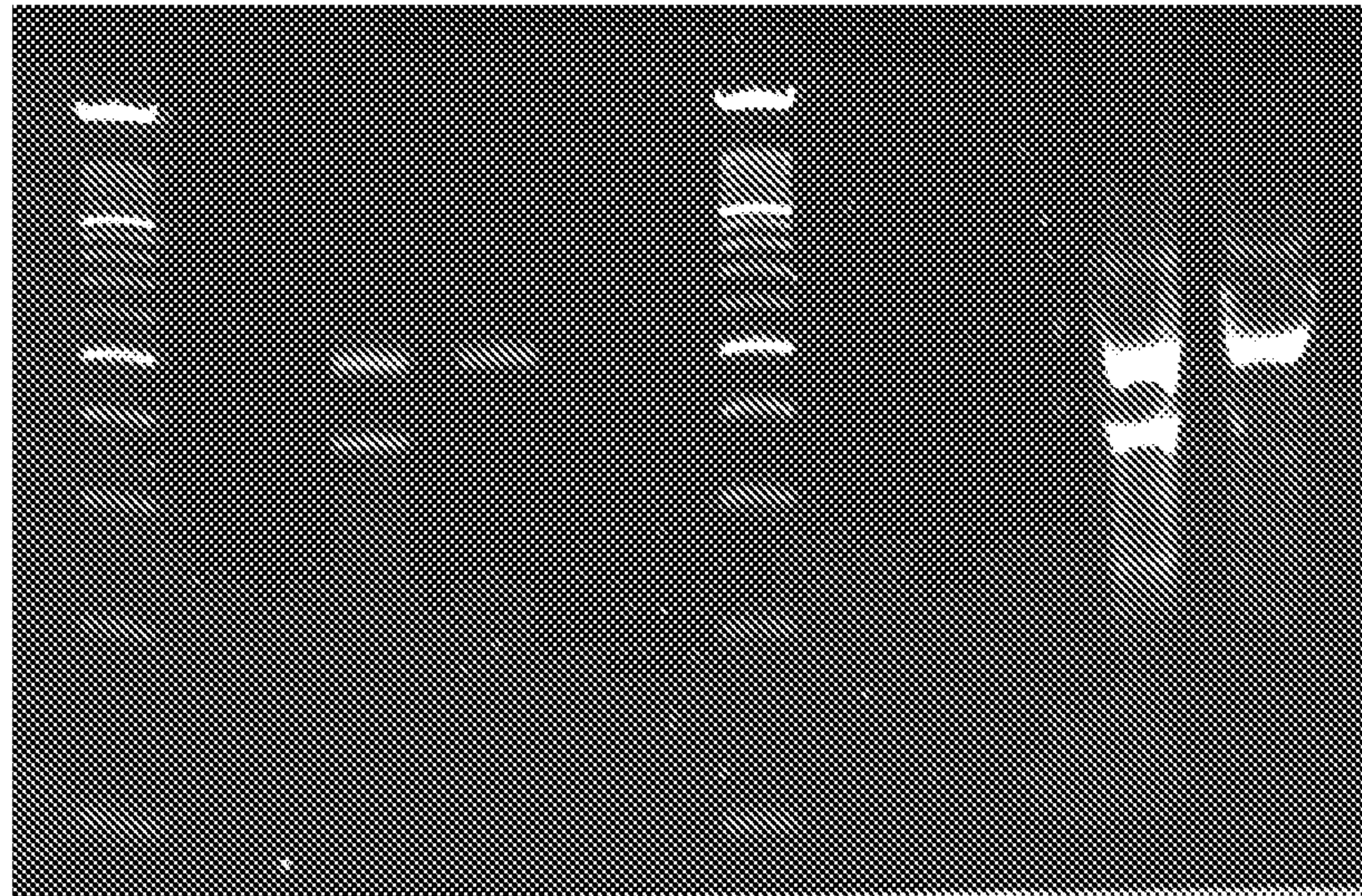
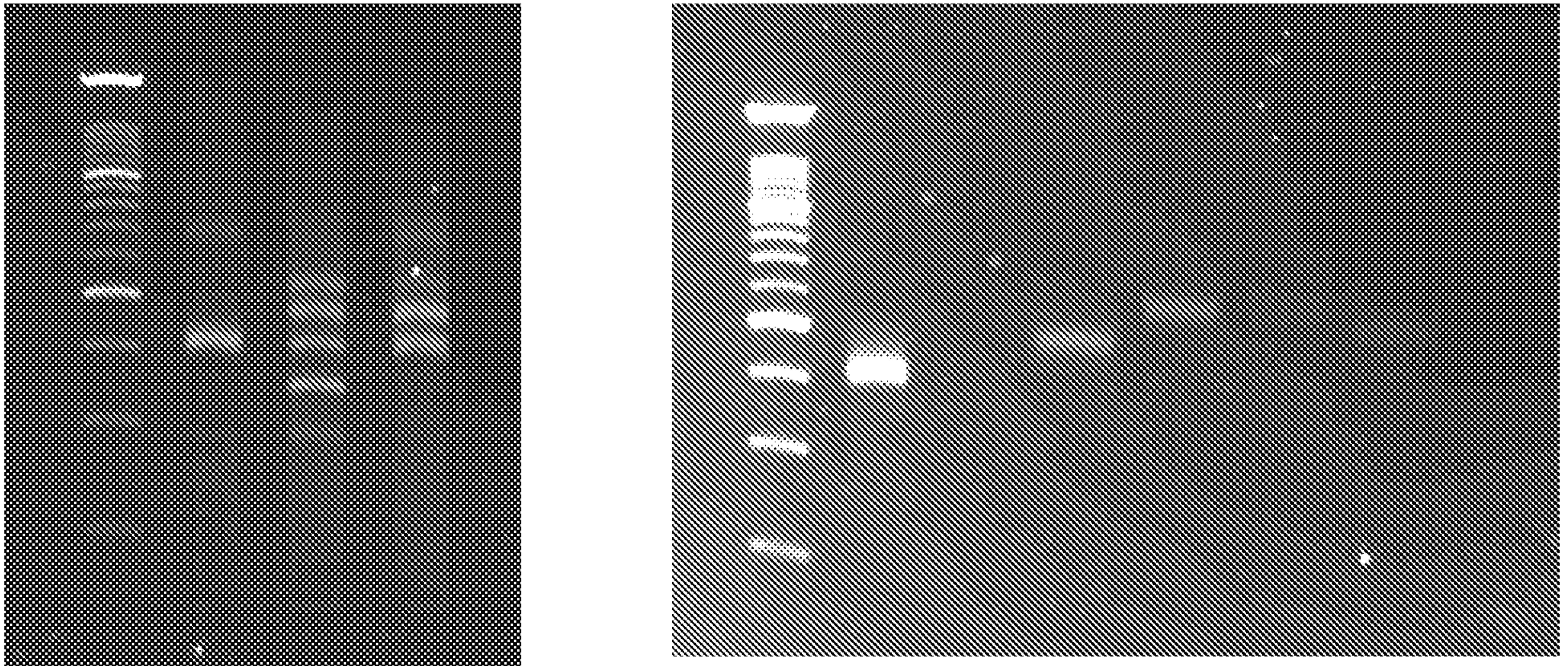


Figure 72

**Figure 73**

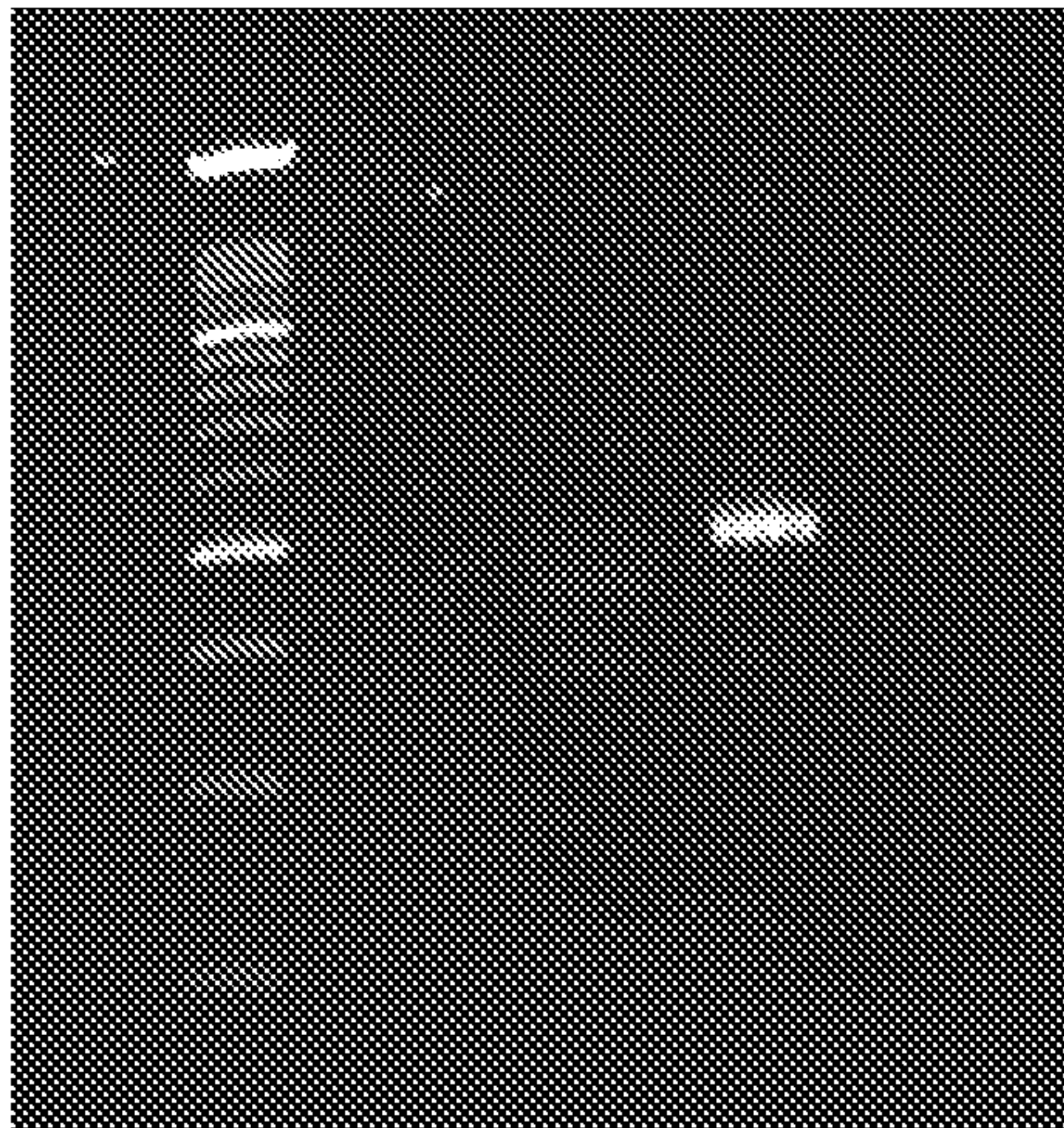


Figure 74

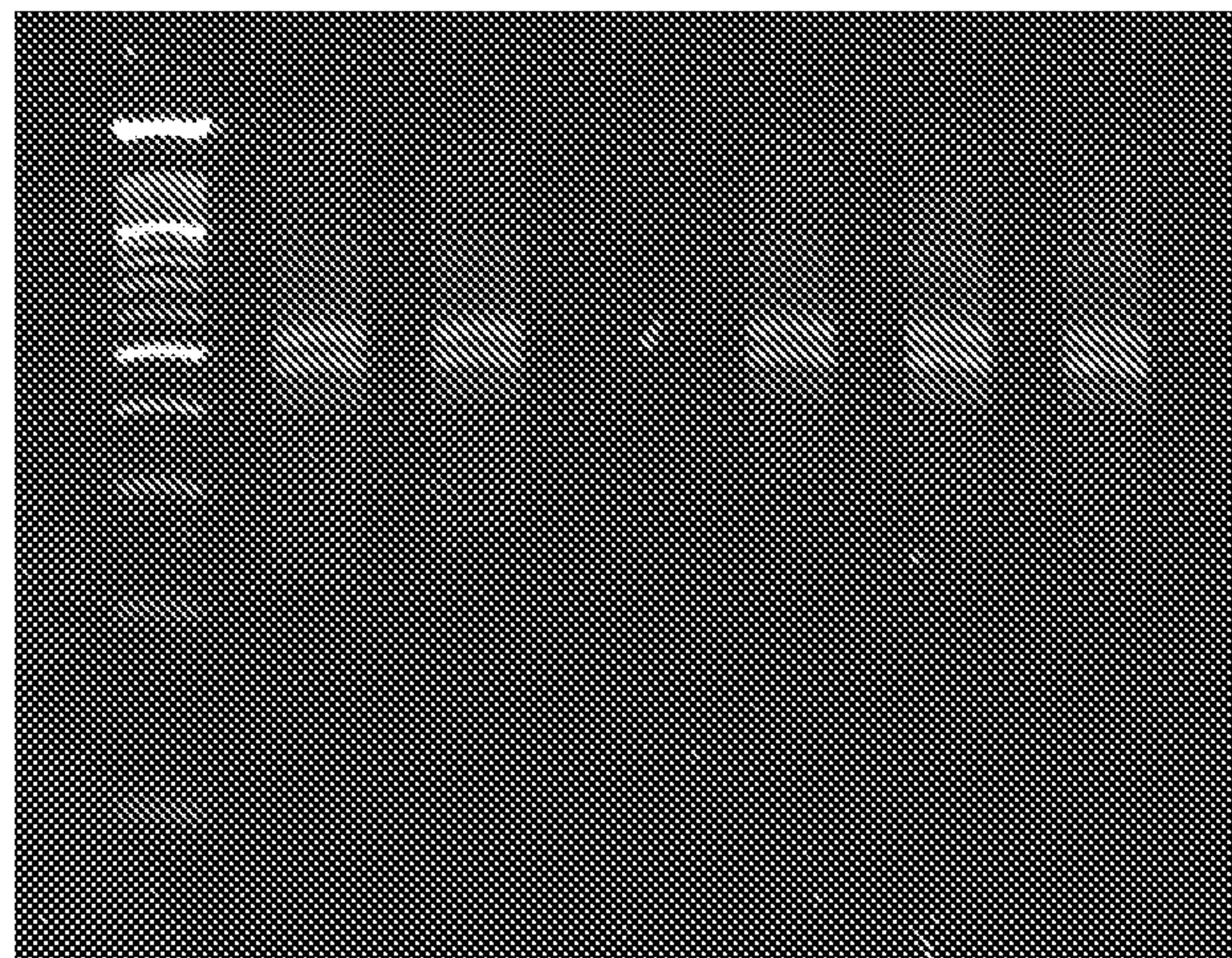
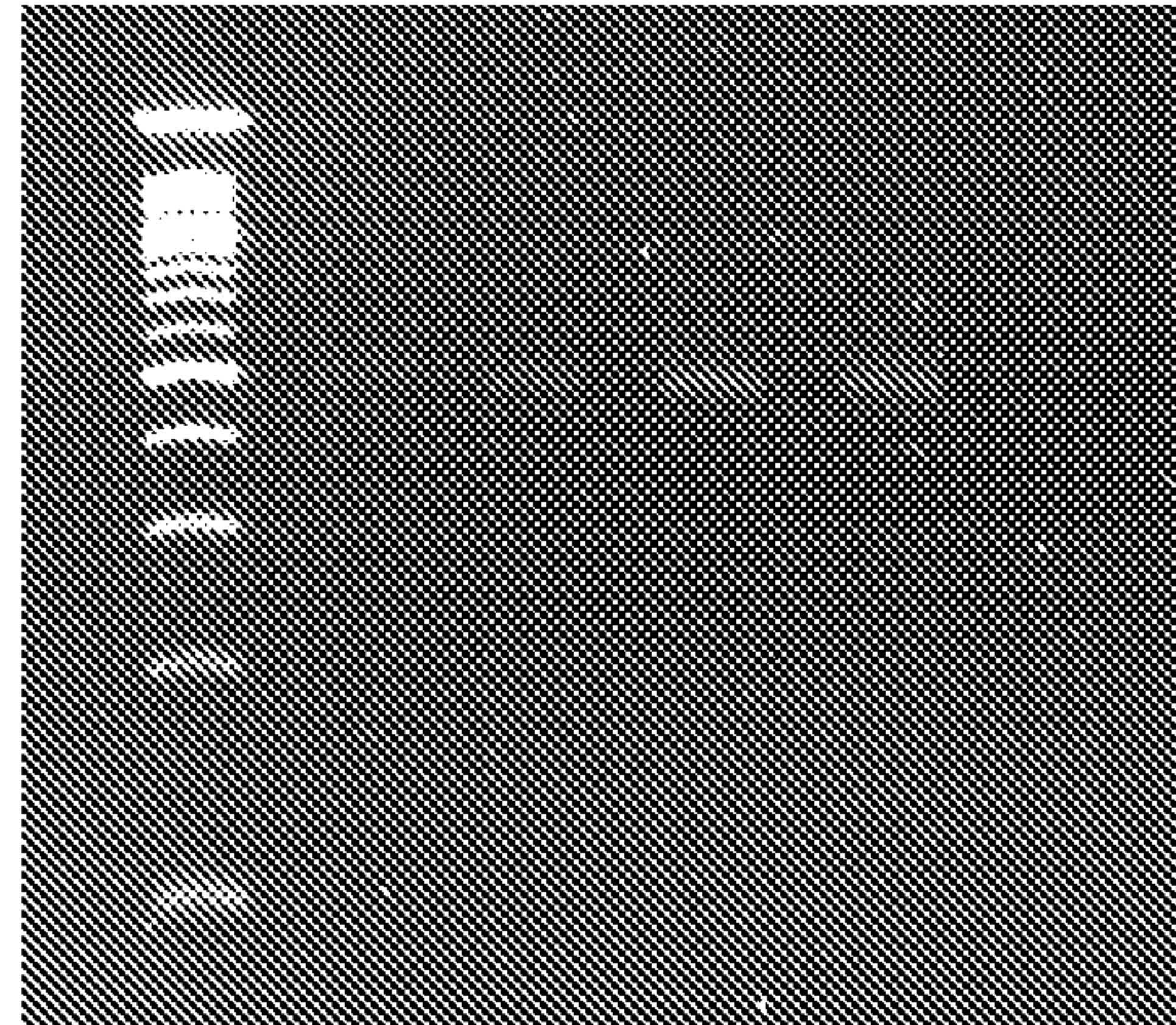
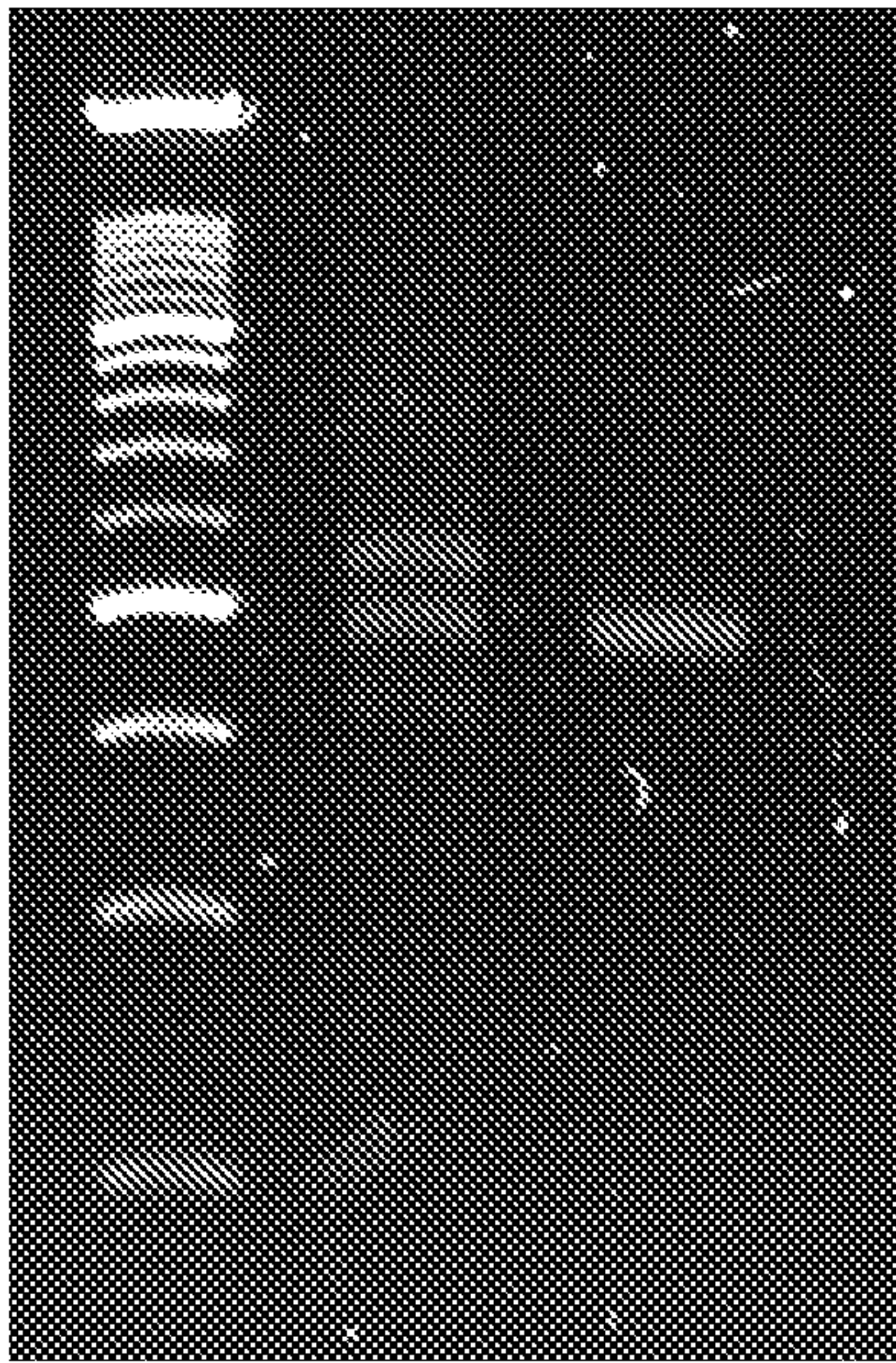
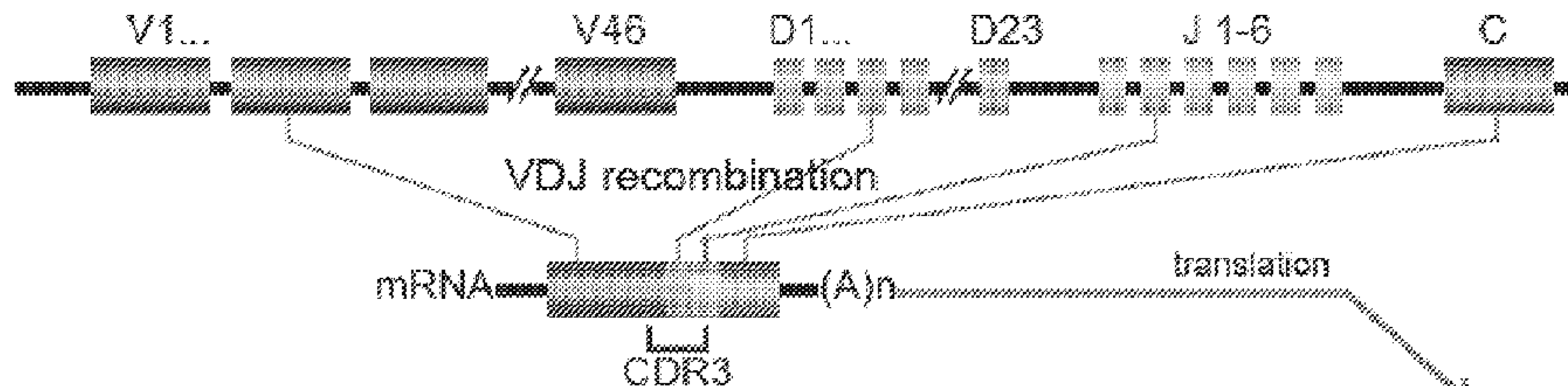


Figure 75

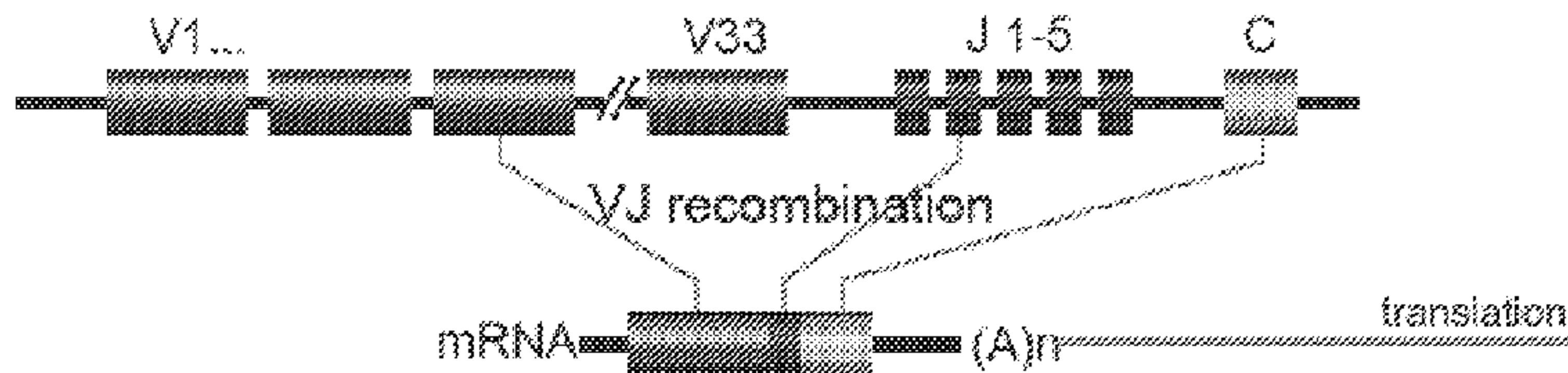
**Figure 76**

a

Germline DNA - heavy chain locus



Germline DNA - light chain locus

**b**