



(12) 发明专利

(10) 授权公告号 CN 1833034 B

(45) 授权公告日 2014.04.16

(21) 申请号 200480022775.3

代理人 赵艳华

(22) 申请日 2004.06.18

(51) Int. Cl.

(30) 优先权数据

C12Q 1/68(2006.01)

PA200300933 2003.06.20 DK

PA200301066 2003.07.12 DK

PA200400242 2004.02.17 DK

PA200400353 2004.03.01 DK

60/549,346 2004.03.02 US

(56) 对比文件

CN 1066519 A, 1992.11.25, 摘要.

审查员 吴永庆

(85) PCT国际申请进入国家阶段日

2006.02.08

(86) PCT国际申请的申请数据

PCT/DK2004/000429 2004.06.18

(87) PCT国际申请的公布数据

W02004/113563 EN 2004.12.29

(73) 专利权人 埃克斯魁恩公司

地址 丹麦韦兹拜克

(72) 发明人 N·B·拉姆辛 P·莫里特曾

S·M·埃赫沃尔德

N·托尔斯特鲁普

(74) 专利代理机构 中国国际贸易促进委员会专

利商标事务所 11038

权利要求书6页 说明书40页

序列表14页 附图99页

(54) 发明名称

用于分析核酸混合物的探针、文库和试剂盒
及其构建方法

(57) 摘要

本发明涉及用于检测、分类或定量复杂的核酸混合物,例如转录物组中的成分的核酸探针、核酸探针文库和试剂盒、以及它们的使用方法。本发明还涉及鉴定可用于该探针文库的核酸探针的方法和鉴定可用于检测给定核酸的手段的方法。

1. 选择可用于寡核苷酸探针文库中的寡核苷酸序列的方法,包括
 - a) 提供具有预定核苷酸数目 N 的所有可能寡核苷酸的第一集合,其中所述寡核苷酸具有 8-9 个核苷酸的长度和至少 50°C 的熔解温度 T_m ,
 - b) 提供靶核酸序列的第二集合,所述第二集合是从 ENSMBL 得到的人转录物组序列,
 - c) 针对所述第一集合的每一个成员,分别从所述第二集合中鉴定和存储包括与所述第一集合的该成员互补的序列的成员的数目,
 - d) 选择在步骤 c 的鉴定中与步骤 c 中鉴定到的最大数目的所述第二集合的成员匹配的所述第一集合的成员,
 - e) 将步骤 d 中选择的成员添加至由选出的、可用于所述寡核苷酸探针文库中的寡核苷酸组成的第三集合中,
 - f) 从所述第一集合中扣除步骤 d 中选择的成员以提供修订的第一集合,
 - m) 重复步骤 d 至 f,直到所述第三集合的组成成员组合在一起可以与来自步骤 b 的靶核酸序列集合中的至少 30% 成员互补为止,
 - 1) 其中每个探针的靶存在于所述人转录物组内序列的至少 6% 中,
 - 2) 这些寡核苷酸不是自身互补的,
 - 3) 探针与其靶序列形成的双螺旋的 T_m 值为 60°C 或更高,并且所述寡核苷酸当组合在一起时覆盖人类转录物组中 >98% 的 mRNA,并且
 - 4) 所述文库包含说明书表 1 所列的序列得到的探针。
2. 根据权利要求 1 的方法,其中所述寡核苷酸的熔解温度 T_m 是至少 60°C。
3. 根据权利要求 1 或 2 的方法,其中所述寡核苷酸的第一集合仅仅包括不能够自身杂交的寡核苷酸。
4. 根据权利要求 1 或 2 的方法,其在步骤 f 之后并在步骤 m 之前包括如下步骤:
 - g) 从所述第二集合中扣除包括了与步骤 d 中选出的成员互补的序列的所有成员,以获得修订的第二集合,
 - h) 针对所述修订的第一集合中的每一个成员,分别地鉴定和存储包括了与所述该成员互补的序列的所述修订的第二集合的成员的数目,
 - i) 选择在步骤 h 的鉴定中与步骤 h 中鉴定到的最大数目的所述第二集合成员匹配的所述第一集合的成员,或者选择可以通过步骤 h 中鉴定到的数目与步骤 c 中鉴定的数目的乘积产生最大数目的所述第一集合的成员,
 - j) 将步骤 i 中选择的成员添加至所述第三集合,
 - k) 从所述修订的第一集合中扣除步骤 i 中选择的成员,和
 - l) 从所述修订的第二集合中扣除包括了与步骤 i 中选择的成员的序列或与之互补的序列的所有成员。
5. 根据权利要求 1 或 2 的方法,其中持续步骤 m 中的重复步骤,直到所述第三集合的组成成员组合在一起可以与来自步骤 b 的靶核酸序列集合中的至少 85% 成员互补为止。
6. 根据权利要求 1 或 2 的方法,其中,在选出所述第三集合的第一个成员后,在步骤 c 之后并在步骤 d 中的选择之前鉴定在所述第一集合中与超过所述最大数目的选定百分数的所述第二集合成员杂交的成员,由此仅有如此鉴定到的这些成员进行步骤 d 中的选择。
7. 根据权利要求 6 的方法,其中所述选定的百分数为 80%。

8. 根据权利要求 1 或 2 的方法,其按照图 2 所示方式实施。

9. 根据权利要求 1 或 2 的方法,其中,所述第一、第二和第三集合存储在计算机系统的存储器中。

10. 根据权利要求 9 的方法,其中,所述第一、第二和第三集合存储在数据库中。

11. 寡核苷酸探针文库,其中文库中的每一个探针均由识别序列标签和检测部分组成,其中文库中的每一个探针的识别序列标签由根据权利要求 1-10 任一项的方法选择,并且在每一个寡核苷酸探针中至少一个单体是 LNA 核苷酸,从而相对于相应的未修饰寡脱氧核苷酸而言增加了对互补靶序列的结合亲和力,由此这些文库探针具有足够的稳定性可以用于序列特异性地结合和检测任何给定靶群体中的靶核酸的实质性部分,并且,其中,不同识别序列的数目占具有给定长度的所有可能序列标签的不到 10%。

12. 根据权利要求 11 的寡核苷酸探针文库,其中文库中探针的识别序列标签片段已经以至少一种如下方式被修饰:

i) 用至少一个 LNA 核苷酸进行替代;

ii) 用至少一个化学部分进行替代以增加探针的稳定性。

13. 根据权利要求 11 或 12 的寡核苷酸探针文库,其中识别序列标签长 8 或 9 个核苷酸。

14. 根据权利要求 13 的寡核苷酸探针文库,其中识别序列标签含氧基 LNA 核苷酸。

15. 根据权利要求 11 或 12 的寡核苷酸探针文库,其中 90% 以上的寡核苷酸探针可以结合并检测核酸群体中的至少两个靶序列。

16. 根据权利要求 15 的文库,其中识别序列标签与核酸群体中的至少两个靶序列互补。

17. 具有 8 和 9 个核苷酸长度的寡核苷酸探针的文库,其包含权利要求 11-16 之任一项中定义的寡核苷酸探针的子集的混合物。

18. 根据权利要求 11 或 17 的寡核苷酸探针文库,其中核酸群体中的不同靶序列的数目是至少 100。

19. 根据权利要求 11 或 17 的寡核苷酸探针文库,其中每一个寡核苷酸探针中的至少一个核苷酸含核苷酸间键,所述核苷酸间键选自:烷基膦酸酯、亚磷酰胺、烷基磷酸三酯、硫代磷酸酯和二硫代磷酸酯键。

20. 根据权利要求 11 或 17 的寡核苷酸探针文库,其中检测部分是共价地或非共价地结合的小沟结合剂或嵌入剂,选自:不对称花菁染料、DAPI、SYBR Green I、SYBR Green II、SYBR Gold、PicoGreen、噻唑橙、Hoechst33342、溴化乙啶、1-O-(1-萘基甲基)丙三醇和 Hoechst33258。

21. 根据权利要求 11 或 17 的寡核苷酸探针文库,其中所述寡核苷酸探针含有非天然核苷酸,以增强探针与互补核酸序列的结合。

22. 权利要求 21 的寡核苷酸探针文库,其中所述非天然核苷酸是 2'-O-甲基、二胺嘌呤、2-硫代尿嘧啶、5-硝基吡啶、通用或简并碱基、嵌入核酸或小沟结合剂。

23. 根据权利要求 11 或 17 的寡核苷酸探针文库,其中所述不同识别序列占具有给定长度的所有可能寡核苷酸的 1% 以下。

24. 根据权利要求 11 或 17 的寡核苷酸探针文库,其中每一个探针都可以使用双标记物

利用分子信标试验原理进行检测。

25. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中每一个探针都可以使用双标记物利用 5' 核酸酶试验原理进行检测。

26. 根据权利要求 11 或 17 的文库, 其中每一个探针都含有单一一个检测部分, 该检测部分可以通过分子信标试验原理检测。

27. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中所述探针靶序列在靶核酸群体的 4% 以上不同靶核酸中出现至少一次。

28. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中从所述文库中已经除去了自身互补的探针序列。

29. 根据权利要求 28 的寡核苷酸探针文库, 其中所述自身互补的序列已经被选择除去。

30. 根据权利要求 28 的寡核苷酸探针文库, 其中所述自身互补的序列已经通过序列特异性修饰而被消除。

31. 根据权利要求 30 的寡核苷酸探针文库, 其中所述序列特异性修饰是非标准核苷酸、具有 SBC 核酸碱基的核苷酸、2' --O- 甲基、二胺嘌呤、2- 硫代尿嘧啶、通用或简并碱基或小沟结合剂。

32. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中每个探针的熔解温度 (T_m) 通过用非天然修饰进行替代以增强探针与互补核酸序列的结合, 而被调整以适用于基于 PCR 的试验。

33. 根据权利要求 32 的寡核苷酸探针文库, 其中所述非天然修饰是用 LNA 进行的替代, 其任选地用 SBC 核酸碱基、2' --O- 甲基、二胺嘌呤、2- 硫代尿嘧啶、5- 硝基吡啶、通用或简并碱基、嵌入核酸或小沟结合剂修饰。

34. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中每个探针的熔解温度 (T_m) 为至少 50°C。

35. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中每个探针在 5' 末端具有 DNA 核苷酸。

36. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中每个探针含有荧光团 - 淬灭剂对用于检测。

37. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中每个探针都可以通过分子信标原理进行检测。

38. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中每个探针与一旦结合双链 DNA 或 DNA-RNA 异源双链靶后将改变荧光的嵌入荧光团或小沟结合剂附着。

39. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中靶群体是人类转录物组。

40. 根据权利要求 11 或 17 的寡核苷酸探针文库, 其中每个寡核苷酸探针检测最大可能数目的不同靶核酸, 从而导致所述文库对给定靶核酸群体的最大覆盖。

41. 根据权利要求 11 或 17 的寡核苷酸探针文库, 所述文库包含表 1a 中定义的探针。

42. 根据权利要求 11 或 17 的文库, 其包含各具有说明书中表 1a 所列的识别元件的探针和 / 或其包含各具有与表 1 中所列的识别元件互补的识别元件的探针。

43. 鉴定用于检测靶核酸的特定检测手段的方法, 该方法包括:

A) 向计算机系统中输入唯一地标识所述靶核酸的核酸序列的数据,其中所述计算机系统包含容纳有至少一个根据权利要求 11-42 之任一项的核酸探针文库的组成信息的数据库,并且,其中该计算机系统还包含针对所述至少一个文库中的每一个探针的靶核酸序列的数据库和 / 或还包含用于获取和比较核酸序列数据的手段,

B) 在该计算机系统中鉴定来自所述至少一个文库的探针,其中该探针的序列存在于所述靶核酸序列中或存在于与该靶核酸序列互补的序列中,

C) 在该计算机系统中鉴定可以扩增该靶核酸序列的引物,和

D) 提供指出步骤 B 中鉴定的探针和步骤 C 中鉴定的引物序列的输出结果,作为对特定检测手段的鉴定。

44. 根据权利要求 43 的方法,其中步骤 A 还包括向所述计算机系统输入标识期望从中选出用于该特定检测手段中的成员的至少一个核酸文库的数据。

45. 根据权利要求 44 的方法,其中标识该至少一个文库的组成的数据是产品代码。

46. 根据权利要求 43-45 之任一项的方法,其中步骤 A 中的输入通过因特网界面实现。

47. 根据权利要求 43-45 之任一项的方法,其中选择步骤 C 中鉴定到的引物,以最小化在 PCR 反应中扩增基因组核酸的可能性。

48. 根据权利要求 47 的方法,其中至少一个引物经选择包括在基因组 DNA 中被内含子中断的核苷酸序列。

49. 根据权利要求 43-45 之任一项的方法,其中选择步骤 C 中选出的引物,以最小化在靶核酸序列上实施 PCR 获得的扩增子的长度。

50. 根据权利要求 43-45 的方法,其中选择步骤 C 中选出的引物,以优化 GC 含量以便实施 PCR。

51. 体外剖析多个靶序列的非诊断目的的方法,其包括使靶序列的样品与根据权利要求 11-42 之任一项的文库接触,和检测、表征或定量与所述靶序列结合的探针序列。

52. 根据权利要求 51 的方法,其检测所述多探针序列结合的所述多个序列中低于 10% 中存在的核酸序列。

53. 根据权利要求 52 的方法,其中所述靶 mRNA 序列或 cDNA 序列包含转录物组。

54. 根据权利要求 53 的方法,其中所述转录物组是人类转录物组。

55. 根据权利要求 51-54 之任一项的方法,其中所述探针文库与固相支持物共价结合。

56. 根据权利要求 55 的方法,其中所述固相支持物包括微量滴定板,该微量滴定板的每个孔包含一个不同的文库探针。

57. 根据权利要求 51-54 之任一项的方法,其中检测步骤通过扩增包含与文库探针互补的识别序列的靶核酸序列来进行。

58. 权利要求 57 的方法,其中使用位于与文库探针互补的识别序列侧翼的一对寡核苷酸引物进行靶核酸扩增。

59. 权利要求 51-54 之任一项的方法,其中一个或多个靶核酸序列的存在或表达水平与物种的表型相关。

60. 用于非诊断目的、使用根据权利要求 11-42 之任一项的文库分析核酸混合物的体外方法,其包括步骤:

(a) 使靶核酸与标记的寡核苷酸探针的文库接触,以便所述靶核酸与所述探针文库的

至少一个成员杂交,由此形成杂交的文库,其中所述寡核苷酸探针的每一个都具有已知的序列并与固相支持物上在已知位置结合;

(b) 使所述杂交的文库与能够切割双链寡核苷酸的核酸酶接触以从所述杂交的文库中释放出所述标记的寡核苷酸探针的一部分或其片段;和

(c) 鉴定所述杂交的文库上已经除去了标记的探针或其片段的位置,以确定所述未标记的靶核酸的序列。

61. 用于非诊断目的的定性或定量地确定样品中靶核酸的存在的体外方法,该方法包括:

i) 利用根据权利要求 43-50 之任一项的方法鉴定检测所述靶核酸的特定手段,其中所述特定检测手段包括寡核苷酸探针和一组引物,

ii) 获得在步骤 i) 中鉴定的引物和寡核苷酸探针,

iii) 在存在来自步骤 ii) 的引物和寡核苷酸探针的情况下对样品实施分子扩增操作,和

iv) 基于步骤 iii) 的结果,确定靶核酸的存在。

62. 根据权利要求 61 的方法,其中步骤 ii) 中获得的引物是通过合成获得的。

63. 根据权利要求 61 或 62 的方法,其中寡核苷酸探针是从根据权利要求 11-42 之任一项的文库中获得的。

64. 根据权利要求 61 或 62 的方法,其中步骤 iii) 中的所述操作是 PCR 或 NASBA 操作。

65. 根据权利要求 64 的方法,其中所述 PCR 操作是 qPCR。

66. 权利要求 11-42 之任一项的文库在制备用于剖析多个靶序列的制剂中的用途,其包括使靶序列的样品与所述文库接触,和检测、表征或定量与所述靶序列结合的探针序列。

67. 根据权利要求 66 的用途,其检测所述多探针序列结合的所述多个序列中低于 10% 中存在的核酸序列。

68. 根据权利要求 67 的用途,其中所述靶 mRNA 序列或 cDNA 序列包含转录物组。

69. 根据权利要求 68 的用途,其中所述转录物组是人类转录物组。

70. 根据权利要求 66-69 之任一项的用途,其中所述探针文库与固相支持物共价结合。

71. 根据权利要求 70 的用途,其中所述固相支持物包括微量滴定板,该微量滴定板的每个孔包含一个不同的文库探针。

72. 根据权利要求 66-69 之任一项的用途,其中检测步骤通过扩增包含与文库探针互补的识别序列的靶核酸序列来进行。

73. 权利要求 72 的用途,其中使用位于与文库探针互补的识别序列侧翼的一对寡核苷酸引物进行靶核酸扩增。

74. 权利要求 66-69 之任一项的用途,其中一个或多个靶核酸序列的存在或表达水平与物种的表型相关。

75. 权利要求 74 的用途,其中所述表型是疾病。

76. 权利要求 11-42 之任一项的文库在制备用于分析核酸混合物的制剂中的用途,其包括步骤:

(a) 使靶核酸与标记的寡核苷酸探针的文库接触,以便所述靶核酸与所述探针文库的至少一个成员杂交,由此形成杂交的文库,其中所述寡核苷酸探针的每一个都具有已知的

序列并与固相支持物上在已知位置结合；

(b) 使所述杂交的文库与能够切割双链寡核苷酸的核酸酶接触以从所述杂交的文库中释放出所述标记的寡核苷酸探针的一部分或其片段；和

(c) 鉴定所述杂交的文库上已经除去了标记的探针或其片段的位置，以确定所述未标记的靶核酸的序列。

用于分析核酸混合物的探针、文库和试剂盒及其构建方法

技术领域

[0001] 本发明涉及用于检测、分类或定量复杂的核酸混合物，例如转录物组 (transcriptome) 中的成分的核酸探针、核酸探针文库和试剂盒，以及它们的使用方法。

背景技术

[0002] 随着用于描绘成千上万基因的表达谱的微阵列，例如 GeneChip™ 阵列 (Affymetrix, Inc., Santa Clara, CA) 的出现，利用传统方法例如 Northern 分析和点印迹分析所需的财力和物力的一小部分，就可以鉴定出表达的基因和细胞表型之间的相关性。微阵列允许开发多个平行试验以鉴定和验证疾病的生物学标记和可以用于诊断和治疗的药物靶标。此外，基因表达谱也可以用于评估和预测暴露于某种因子（例如，药物、潜在的毒素或致癌物等）或情况（例如，温度、pH 等）的代谢和毒理学后果。

[0003] 微阵列实验常常产生仅其中一小部分对于实验者是有价值的冗余数据。此外，由于基于微阵列的分析的高度平行模式，对于各个捕获探针而言，条件可能不是最佳的。基于这些原因，微阵列实验十分经常地尾随使用单基因均相试验的验证研究或被该验证研究随后替代。这些单基因均相试验经常是基于定量 PCR 的方法，例如 5' 核酸酶试验或其它类型的双标记探针定量实验。然而，这些实验仍是耗时的单反应实验，受到高成本和耗时的探针设计程序的阻碍。而且，5' 核酸酶试验的探针相对较大（例如，15-30 个核苷酸）。因此，目前已知的均相试验系统中的这些限制性对微阵列发现结果的验证以及对聚焦的靶验证方法均构成了瓶颈。

[0004] 用于避免此瓶颈的一个方法是省去 5' 核酸酶试验中使用的昂贵双标记指示探针和分子信标，代之使用在结合双链而非单链 DNA 时发荧光的非序列特异性 DNA 嵌入染料，例如 SYBR Green。使用该染料，可以无任何例外地实时检测任何扩增的序列。然而，该技术受到几个问题的阻碍。例如，PCR 扩增过程中的非特异引物引发可以产生参与定量过程的与目的无关的非靶扩增子。而且，在该反应中普遍存在 PCR 引物之间的相互作用，从而形成“引物二聚体”。由于 PCR 反应中典型使用高浓度的引物，这可以导致显著数量的短双链非靶扩增子，该扩增子也结合嵌合染料。因此，通过实时 PCR 定量 mRNA 的优选方法是使用序列特异性检测探针。

[0005] 用于避免随机扩增和引物二聚体形成问题的一个方法是使用可以用于检测大量不同类型的核酸分子但又保持一定的序列特异性的通用检测探针，该方法已经由 Simeonov 等 (Nucleic Acid Research 30(17):91, 2002; 美国专利公布 20020197630) 描述，其涉及使用包含了具有给定的某个（或某些）长度的所有可能序列中 10% 以上的序列的探针文库。该文库可以包括各种非天然核酸碱基和其它修饰以稳定文库中探针 / 引物与靶序列的结合。即使如此，对于大多数序列，也需要至少 8 个碱基的最短长度来获得可以与大多数和应用（例如实时 PCR）相关的试验条件相容的稳定程度。由于具有所有可能的 8 聚体 (8-mer) 的通用文库将含有 65,536 个不同序列，因此先前由 Simeonov 等考虑的甚至最小的文库也含有所有可能性的 10% 以上，即，至少 6554 个序列，这在操作上是不实际的而且其构建成

本巨大。

[0006] 从实践的角度出发,几个因素限制了当前的均相试验应用的方便性和利用度。常规试验技术的使用者面临的问题包括:

[0007] • 由于购买针对每个转录物的探针的价格很高,当尝试在少数样本中检测许多不同基因时,将会价格高得惊人。

[0008] • 合成标记探针是耗时的,而且从订货到收到厂家发出的货物的时间常常超过 1 周。

[0009] • 用户设计的试剂盒可能不会在第一次时起作用,因此,按每试验计,有效的试剂盒是昂贵的。

[0010] • 难于快速地检测新靶标或反复地改进探针设计。

[0011] • 商业有效探针的确切探针序列可能不为消费者所知,从而导致在结果的评估和科学出版的适宜性上出现问题。

[0012] • 当试验条件或成分不清时,可能不能从替代来源定购试剂。

[0013] 本发明解决了这些实际问题并旨在确保快速和价廉地开发出用于定量基因转录物的准确而特异的试验。

[0014] 发明简述

[0015] 期望能够使用有限数量的寡核苷酸检测探针在均相试验系统中定量例如人类转录物组(transcriptome)中的大多数基因(例如,>98%)的表达。本发明通过提供构建如下通用多探针的方法解决了当前方法在均相试验方面面临的上述问题,其中所述通用多探针(genericmulti-probes)具有足够的序列特异性以致它们不可能检测随机扩增的序列片段或引物二聚体,但各自仍能够检测许多不同的靶序列。这些探针可以用于不同的试验中,并可以组合在小探针文库(50至500个探针)中,当与靶特异性引物组联合时其可以用于检测和/或定量由成千上万个不同核酸组成的复杂混合物中的各成分(例如在由>30,000个不同核酸组成的人类转录物组中检测各转录物)。

[0016] 每一个多探针(multi-probe)包含两个元件:1)由一个或多个标记物组成以检测探针和靶的结合的检测元件或检测部分;和2)确保与目的特异靶(一个或多个)结合的识别元件或识别序列标签。检测元件可以是均相试验中使用的任何检测原理。可以通过与靶结合后一个或多个标记物的可测量的性质改变直接地检测结合(例如,有或没有茎结构的分子信标类试验),或者可以通过结合后的随后反应(例如在5'核酸酶试验中通过DNA聚合酶的5'核酸酶活性进行的切割)来间接地检测结合。

[0017] 识别元件是新的本发明成分。它包含序列经选择以致能够检测给定的复杂样品混合物中一个大的靶核苷酸子集(subset)的短寡核苷酸部分。分别被设计用于检测许多不同靶分子的这些新探针被称作多探针。通过选择最频繁遇到的序列和利用短识别序列的重现来设计用于多个靶的探针的概念是新的,并与设计出来以便尽可能具有针对单个靶序列的特异性的常规探针不同。随后,周围引物和选择的探针序列相组合将确保该多探针的特异性。由试图用最少量的探针处理大量的靶而产生的此新的设计原理同样是本发明的一部分。这通过如下发现得以实现:非常短的8-9聚体LNA mix-mer探针与基于PCR的试验是相容的。在本发明的一个方面,在识别元件中掺入核酸碱基、核苷碱基或核苷酸的修饰物或类似物,以及可能地小沟结合剂及其它修饰,所有这些均旨在稳定探针和靶分子之间形

成的双螺旋,以便可以使用最短的可能探针序列与最宽的靶范围。在本发明的一个优选方面,所述修饰是掺入 LNA 残基以将识别元件的长度减少至 8 或 9 个核苷酸并同时保持所形成的双螺旋的足够稳定性以便可以在通常试验条件下进行检测。

[0018] 优选地,修饰多探针,以使该探针与具有相同序列但无修饰的探针相比在相同检测条件,例如 PCR 条件或严紧杂交条件下对靶序列的结合亲和力增加至少两倍。优选的修饰包括,但不限于,包含已经通过化学部分修饰或已经被类似物(例如,包括核糖或脱氧核糖类似物)替代或者已经通过使用非磷酸二酯键的核苷酸间键(例如非磷酸酯的核苷酸间键)进行过修饰的核酸碱基、核苷碱基或核苷酸,以增加结合亲和力。优选的修饰还可以包括附着双螺旋稳定剂,例如小沟结合剂(MGB)或嵌入核酸(INA)。此外,优选的修饰还可以包括添加不管靶链相对位置上的核酸碱基如何均能够稳定双螺旋形成的无区分性碱基,例如 5-硝基吡咯。最后,能够序列特异地与靶序列结合的、由非糖-磷酸主链组成的多探针,例如 PNA,也是所考虑的修饰。所有上述增加结合亲和力的不同修饰在下文中将称作“稳定化修饰”(stabilizing modification),由此产生的多探针在下文中也将称作“修饰的寡核苷酸”。更优选地,修饰的寡核苷酸比具有相同序列但无该稳定化修饰的探针具有高至少大约 3 倍、4 倍、5 倍或 20 倍的结合亲和力。

[0019] 最优选地,稳定化修饰是包括一个或多个 LNA 核苷酸类似物。根据本发明的 6 至 12 个核苷酸的探针可以包含 1 至 8 个稳定化核苷酸,例如 LNA 核苷酸。当包括至少两个 LNA 核苷酸时,这些核苷酸可以是连续的,或被一个或多个非 LNA 核苷酸隔开。在一方面,LNA 核苷酸是 α 和 / 或 xylo LNA 核苷酸。

[0020] 本发明还提供可以在 NASBA 试验条件下使用的寡聚体多探针文库。

[0021] NASBA 是一种适于扩增 RNA 的、特异性等温核酸扩增方法。通过用硫氰酸胍加上 Triton X-100 裂解以及最后从二氧化硅颗粒上洗脱纯化的核酸,进行核酸分离。通过 NASBA 进行的扩增涉及三种酶,即,AMV 逆转录酶、RNase H 和 T7 RNA 聚合酶的协同活性。通过在分离时加入的内部校准物实现定量检测,其中所述内部校准物与野生型 RNA 一起扩增并随后使用电化学发光进行鉴定。

[0022] 本发明还提供包含多探针的寡聚体多探针阵列,其中所述多探针包含至少一个以上定义的稳定化修饰。优选地,探针的长度小于大约 20 个核苷酸,更优选地小于 12 个核苷酸,最优选地大约 8 或 9 个核苷酸。此外,优选地,文库包含大约 3000 个以下的探针,更优选地,文库包含 500 个以下的探针,最优选地大约 100 个探针。含有标记的多探针的文库根据识别元件所连接的检测元件的类型,可以用于多种应用中。这些应用包括,但不限于,双或单标记试验,例如 5' 核酸酶试验,分子信标应用(参见例如 Tyagi 和 Kramer Nat. Biotechnol. 14:303-308, 1996)和其它基于 FRET 的试验。

[0023] 在本发明的一个方面,对上述多探针一起进行设计以便可以彼此补充成为具有给定长度的所有可能序列的一个如下预定子集,该子集经选择能够使用最少数量的多探针序列检测 / 表征 / 定量复杂混合物中最大数量的核酸。这些预先设计的所有可能序列的小子集构成多探针文库。本发明的多探针文库通过如下方式以极大降低的复杂度实现该功能性:有意地选择最常出现的、具有一种或多种给定长度的寡聚体、同时努力地使该选择多样化以在复杂核酸靶群体中达到最佳可能的覆盖率。在一个优选方面,文库的探针与核酸靶群体(例如,人类 mRNA 群体)中的大约 60% 以上杂交。更优选地,探针与靶分子群体的所

有靶核酸分子中的 70% 以上、80% 以上、90% 以上、95% 以上, 甚至 98% 以上杂交 (见, 例如图 1)。

[0024] 在本发明的一个最优选方面, 包含具有选定的探针长度的所有可能序列中的大约 0.1% 的探针文库 (即, 例如, 大约 100 个多探针) 能够检测、分类和 / 或定量任何特定物种, 尤其是哺乳动物, 更尤其是人的转录物组中 98% 以上的 mRNA 转录物 (即, > 35,000 个不同的 mRNA 序列)。事实上, 优选本发明多探针文库覆盖靶群体的所有靶核酸中的至少 85%。

[0025] 在现有均相试验中存在的上述问题通过使用本发明的多探针文库得以解决, 所述文库由短检测探针最小的组合组成, 该探针经选择能够识别或检测给定生物的给定细胞类型中所有表达基因的大多数。一方面, 文库包含可以检测具有超过大约 10,000 个基因、超过大约 15,000 个基因、超过大约 20,000 个基因、超过大约 25,000 个基因、超过大约 30,000 个基因或超过大约 35,000 个基因或者等同数量的不同 mRNA 转录物的转录物组中的每一个转录物的探针。一个优选方面, 文库包含可以检测哺乳动物转录物序列, 例如, 小鼠、大鼠、兔、猴或人序列的探针。

[0026] 通过提供可以用于快速开发定量实时终点 PCR 试验的成本有效的多探针集合, 本发明克服了在目前均相试验中存在的上述限制性。本发明多探针的检测元件可以是单标记的或双标记的 (例如, 通过在探针的每一个末端或在探针的内部位置包含标记物)。因此, 可以调整本发明探针使其适用于 5' 核酸酶试验、分子信标试验、FRET 试验和其它相似试验。一方面, 检测多探针包括能够彼此相互作用以产生信号或改变信号的两个标记物, 由此可以在探针与靶序列杂交后检测到信号或信号改变。一个特别的方面是两个标记物包含淬灭分子和报道分子。

[0027] 另一方面, 探针包含靶特异性识别片段, 该片段能够特异地与包含互补识别序列的多个不同核酸分子杂交。本发明的一个特定的检测方面称作“具有茎区的分子信标”, 即, 识别片段的两侧为可以退火形成发夹的第一和第二互补发夹形成序列。报道标记物连接于一个互补序列的末端, 而淬灭部分连接于另一互补序列的末端。在第一和第二互补序列杂交时 (即, 探针识别片段未与其靶杂交时) 形成的茎使得这两个标记物彼此紧靠, 从而导致报道分子产生的信号由于荧光共振能量转移 (FRET) 而被淬灭。当探针与靶序列杂交时降低了两个标记物之间的接近度, 接近度的改变引起两个标记物之间的相互作用发生改变。由此探针的杂交致使从报道分子产生信号 (例如, 荧光), 该信号可以被检测和 / 或定量。

[0028] 另一方面, 多探针包含位于短识别序列的相对末端的报道分子和淬灭分子, 这样两个部分彼此充分接近以致淬灭分子可以实质上地降低报道分子所产生的信号。这在探针游离于溶液中以及其与靶核酸结合时均是如此。在本发明的一个称作“5' 核酸酶试验”的特定检测方面, 多探针可以容易被 DNA 聚合酶的 5' 核酸酶活性切割。该反应可能可以导致淬灭分子和报道分子分离以及可检测信号产生。由此, 可以在基于扩增的试验中使用这样的探针以检测和 / 或定量靶核酸的扩增过程。

[0029] 第一方面, 本发明涉及上述多探针文库。在该寡核苷酸探针文库中, 每一个探针都包含具有大约 x 个核苷酸长度的识别片段和检测元件, 其中所述核苷酸中的一些核苷酸或所有核苷酸被相较于天然核酸碱基能够增加结合亲和力的非天然碱基替代, 和 / 或所述寡核苷酸探针中的一些或所有核苷酸单元被化学部分修饰以增加结合亲和力, 这样探针在适于检测的条件下将具有与靶序列结合的充分稳定性, 而且, 其中, 不同识别片段的数量占给

定长度的所有可能片段的 10% 以下, 而且其中 90% 以上的探针可以检测靶核酸群体中的一个以上互补靶, 由此该寡核苷酸探针文库可以检测靶核酸群体的所有靶序列中的实质上大部分靶序列。

[0030] 因此, 本发明涉及寡核苷酸探针文库, 其中文库中的每一个探针都由识别序列标签和检测部分组成, 其中每一个寡核苷酸探针中的至少一个单体是修饰的单体类似物, 从而与相应的未修饰寡脱氧核糖核苷酸相比, 增加了对互补靶序列的结合亲和力, 这样, 文库探针具有充分的稳定性可以用于序列特异性地结合和检测任何给定靶群体中的靶核酸的实质性部分, 而且, 其中不同识别序列的数量占给定长度的所有可能序列标签的 10% 以下。

[0031] 本发明还涉及寡核苷酸探针文库, 其中文库探针的识别序列标签片段已经以至少一种以下方式进行了修饰:

[0032] i) 被至少一个非天然核苷酸替代; 和

[0033] ii) 被至少一个化学部分替代以增加探针的稳定性。

[0034] 而且, 本发明还涉及寡核苷酸探针文库, 其中识别序列标签长 6-12 个核苷酸 (即, 6、7、8、9、10、11 或 12), 而且其中优选长度是 8 或 9 个核苷酸。

[0035] 此外, 本发明还涉及具有 LNA 核苷酸替代的识别序列标签。

[0036] 本发明还涉及本发明的文库, 其中, 90% 以上的寡核苷酸探针可以结合并检测核酸群体中的至少两个靶序列 — 优选地这是因为所结合的靶序列与探针的识别序列互补。

[0037] 而且, 优选地, 探针能够检测靶核酸群体中的一个以上靶标, 例如, 探针能够与靶核酸群体中所含的多个不同核酸分子杂交。

[0038] 本发明还提供用于设计包含至少一个稳定化核苷酸碱基的多探针的方法、系统和写在计算机可读介质上的计算机程序 (“计算机程序产品”)。所述方法包括查询靶序列数据库 (例如, 表达序列数据库) 和设计一个小的探针集合 (例如, 50 或 100 或 200 或 300 或 500 个), 其中所述探针: i) 具有在 PCR 条件下结合其相应靶序列的充分结合稳定性, ii) 具有有限的自身形成双螺旋结构的倾向, 和 iii) 能够结合和检测 / 定量给定序列数据库 (例如, 表达序列数据库) 的所有序列中的至少大约 60%、至少大约 70%、至少大约 80%、至少大约 90% 或至少大约 95%。

[0039] 计算机 (In silico) 设计探针, 以包含具有给定长度的所有可能核苷酸组合, 从而形成虚拟 (virtual) 候选探针数据库。这些虚拟探针相对于靶序列数据库进行检索, 以鉴定具有检测数据库中最多不同靶序列的最大能力的探针 (“最佳探针”)。从虚拟探针数据库取出如此鉴定的最佳探针。此外, 从靶核酸数据库中扣除通过此先前的最佳探针集合鉴定的靶核酸。然后剩余的探针相对于剩余的靶序列进行检索以鉴定第二组最佳探针。重复该方法直到鉴定到能够所需地覆盖靶序列数据库的一组探针。可以将该组探针存储在数据库中作为序列来源用于转录物组分析。可以合成具有相应于该数据库中序列的识别序列的多探针, 以产生多探针文库。

[0040] 在一个优选方面, 靶序列数据库包含相应于人类 mRNA 的核酸序列 (例如, mRNA 分子、cDNA 等等)。

[0041] 另一方面, 本发明方法还包括基于识别序列包含至少一个稳定化核苷酸 (例如, LNA 分子) 的前提的计算稳定性。在一个优选方面, 利用计算的稳定性, 在首次相对于靶序列数据库进行查询以起始最佳探针识别序列的鉴定之前, 从虚拟候选探针数据库中除去稳

定性不足的探针识别序列。

[0042] 另一方面,本发明方法还包括基于识别序列包含至少一个稳定化核苷酸(例如 LNA 分子)的前提,计算给定探针识别序列自身形成双螺旋结构的倾向。在一个优选方面,利用计算的倾向性,在首次相对于靶序列数据库进行查询以起始最佳探针识别序列的鉴定之前,从虚拟候选探针数据库中中除去可能形成探针双螺旋的探针识别序列。另一方面,本发明方法还包括通过相对于剩余靶序列的查询以及相对于起始靶序列集合的查询计算包含不断增长的最佳探针候选物的给定候选探针识别序列的通用性。在一个优选方面,只有同时频繁出现在剩余靶序列和最初靶序列中的探针识别序列才被添加至正在增长的最佳探针识别序列集合中。在最优选的方面,这通过如下方式实现:计算由这些查询得到的分数的乘积,并选择具有最高乘积并且在针对当前靶标的查询中仍属于具有最佳分数的 20% 的探针识别序列的探针识别序列。

[0043] 本发明还提供写入计算机可读介质的计算机程序,所述程序包含用于检索包含多个不同靶序列的数据库和用于鉴定能够确定所述数据库中的至少大约 60%、大约 70%、大约 80%、大约 90% 和大约 95% 序列的一组探针识别序列的指令。一方面,该程序提供实施如上所述方法的指令。另一方面,该程序提供实施如图 2 所示的算法的指令。本发明还提供系统,其中所述系统包括用于储存包含多个不同靶序列的序列信息的数据库的储存器,还包括用于执行如下程序指令的应用程序,所述程序指令用于从数据库中检索出能够与数据库中至少大约 60%、大约 70%、大约 80%、大约 90% 和大约 95% 序列杂交的一组探针识别序列。

[0044] 本发明另一方面涉及包含各自独立地具有大约 1 至 8 个核苷酸长度的识别片段和检测元件的寡核苷酸探针,其中寡核苷酸中的一些或所有核苷酸被相对于天然核苷酸碱基而言能够增加结合亲和力的非天然碱基或碱基类似物替代,和 / 或寡核苷酸探针的一些或所有核苷酸单元被化学部分修饰或被类似物置换以增加结合亲和力,和 / 或其中所述寡核苷酸被化学部分修饰或是寡核苷酸类似物以增加结合亲和力,这样,探针具有在适于检测的条件下结合靶序列的充分稳定性,而且其中探针能够检测靶核酸群体中一个以上的互补靶标。

[0045] 本发明的一个优选实施方式是表征或检测或定量靶核酸的试剂盒,所述试剂盒包含多探针文库样品。一方面,试剂盒包含针对其用途的计算机 (insilico) 操作方案。另一方面,试剂盒包含与获得廉价 DNA 引物的建议相关的信息。这些试剂盒中所包含的探针可以具有上述的任一或所有特征。在一个优选方面,多个探针都包含至少一个稳定化核苷酸,例如 LNA 核苷酸。另一方面,多个探针都包含与至少一个化学部分偶联或稳定结合的核苷酸以增加探针的结合稳定性。在再一优选方面,试剂盒包含大约 100 个不同探针。根据本发明的试剂盒允许用户快速且有效地开发用于成千上万个不同核酸靶标的试验。

[0046] 本发明还提供包含一个或多个 LNA 核苷酸、具有大约 8 或 9 个核苷酸的减少长度的多探针。通过选择通常出现的 8 和 9-mer 作为靶,可以使用相同探针检测许多不同基因。每一 8 或 9-mer 探针可以用于检测 7000 个以上的不同人类 mRNA 序列。而必需的特异性由针对靶基因的廉价 DNA 引物以及靶向所扩增的 DNA 的该 8 或 9-mer 探针序列的组合作用来保证(图 1)。

[0047] 在一个优选实施方案中,本发明涉及包含 LNA 替代的八聚体和九聚体的寡核苷酸

多探针文库,该文库具有不超过大约 1000 个的所选序列,优选不超过 500 个的所选序列,或更优选地不超作大约 200 个的所选序列,例如由大约 100 个不同的所选序列组成,由此该文库能够识别靶生物或靶器官中大约 90% 以上、更优选地大约 95% 以上,更优选地大约 98% 以上的 mRNA 序列。

[0048] 阳性对照样品:

[0049] 在设计用于多基因的实时 PCR 检测试验时反复出现的问题是:这些从头设计的成功率小于 100%。查明非功能性试验可能是棘手的,因为理想地每一个探针需要一个靶特异性模板以检验该检测探针的功能性。而且,如果未知在测试样品中是否存在靶标,则靶特异性模板可以用作阳性对照。当使用本发明所述探针文库试剂盒中的有限数量的检测探针(例如,90 个)进行操作时,也可以提供 PCR 可扩增模板形式的阳性对照靶,该阳性对照靶含有针对该有限数量的探针的所有可能靶标(例如,90 个)。该特征允许用户评价各探针的功能,而对于非重现的以探针为基础的试验,该特征是不适宜的,由此该特征构成了本发明的又一有益特征。针对图 13 列出的优选的提议探针识别序列,我们设计了用于所有探针的对照序列多联体,以包含用于此头 40 个探针中每一个探针的 PCR 可扩增靶。

[0050] 探针序列选择

[0051] 本发明的一个重要方面是在给定靶选择标准的情况下选择最佳的探针靶序列以便使用尽可能少的探针靶向众多靶标。这可以通过如下方式实现:有意地选择出现频率高于从随机分布可预期到的出现频率的靶序列。

[0052] 因此,本发明一方面涉及选择可在本发明多探针文库中使用的寡核苷酸序列的方法,该方法包括

[0053] a) 提供第一集合(first list)—具有预定核苷酸数量 N(典型地选自 6、7、8、9、10、11 和 12 的整数,优选地 8 或 9)的所有可能寡核苷酸,所述寡核苷酸具有至少 50°C(优选地至少 60°C)的熔解温度 T_m ;

[0054] b) 提供第二集合(second list)—靶核酸序列(例如本文讨论的一组靶核酸的群体);

[0055] c) 针对所述第一集合的每一成员,分别地从所述第二集合鉴定并存储包括了该成员的互补序列的第二集合成员的数目;

[0056] d) 选择在步骤 c 的鉴定中与步骤 c 中鉴定出的最大数量的第二集合成员匹配的所述第一集合成员;

[0057] e) 将步骤 d 中选择的成员添加至第三集合,其中该第三集合由选择的可用于前述权利要求之任一项的文库中的寡核苷酸组成;

[0058] f) 从所述第一集合中扣除步骤 d 中选择的成员以提供修订的第一集合;

[0059] m) 重复步骤 d 至 f,直到所述第三集合由下述成员组成,所述的组成成员合在一起能够与步骤 b 的该靶核酸序列集合中的至少 30% 成员互补(通常地,该百分数将更高,例如,至少 40%、至少 50%、至少 60%、至少 70%、至少 75%、至少 80%、至少 85%、至少 90%、至少 95%,或甚至更高,例如至少 97%、至少 98%,甚至高达至少 99%)。

[0060] 优选,该第一集合仅仅包括不能自我杂交的寡核苷酸,以便在随后的探针使用中具有较少的假阳性倾向。

[0061] 所述选择方法可以在步骤 f 之后但在步骤 m 之前包括数个步骤:

[0062] g) 从所述第二集合中扣除所有如下成员以获得修订的第二集合,其中所述成员包括与在步骤 d 中选出的成员互补的序列;

[0063] h) 针对所述修订的第一集合的每一成员,从所述修订的第二集合中鉴定和存储包括与所述每一成员互补的序列的成员的数量的;

[0064] i) 选择在步骤 h 的鉴定中与步骤 h 中鉴定到的最大数量的第二集合成员匹配的所述第一集合的成员;或者,选择通过步骤 h 中鉴定的数量和步骤 c 中鉴定的数量的乘积能够给出最大数的所述第一集合成员;

[0065] j) 将步骤 i 中选择的成员加入所述第三集合;

[0066] k) 从所述修订的第一集合中扣除步骤 i 中选出的成员;和

[0067] l) 从所述修订的第二集合中扣除所有如下成员,所述成员包括与步骤 i 中选出的成员互补的序列。

[0068] 步骤 c 后步骤 d 的选择可以方便地在鉴定所述第一集合中的如下成员之后进行,所述成员与所述第二集合中超过选定百分数(60%或更高,例如,优选地 80%)的最大的成员数的成员杂交,由此在步骤 d 中仅仅选择如此鉴定的成员。

[0069] 在该选择方法的实际实施中,所述第一、第二和第三集合存储在计算机系统的存储器,优选地数据库中。存储器(也称作“计算机可读介质”)可以是易失性的和非易失性的,即,常规用于计算机系统上的任何存储装置:随机存取存储器(RAM)、只读内存(ROM)、数据存储装置例如硬盘、CD-ROM、DVD-ROM 和任何其它已知的存储装置。

[0070] 本发明还提供在计算机可读介质(如上定义)中写入了实施本发明选择方法的指令的计算机程序产品。即,可以在快速计算机存储器中编译和负载该计算机程序,或者可以在永久性存储装置上(任选地以压缩格式)负载并从该存储装置执行该计算机程序。因此,本发明也包括如下系统,该系统包含靶序列数据库和用于执行该计算机程序的应用程序。图 17 中给出了用于该计算机程序的源代码。

[0071] 在随机分布的核酸群体中,具有给定长度的选定序列的出现率将遵循以下定义的统计学分布:

[0072] $N1$ = 该给定的核酸群体的全长(例如,在 2003 年 6 月 30 日第 1 次发布的 RefSeq 中为 76,002,917 碱基对)。

[0073] $N2$ = 包含该核酸群体的片段的数目(例如,在 2003 年 6 月 30 日第 1 次发布的 RefSeq 中为 38,556 个基因)。

[0074] $N3$ = 识别序列的长度(例如,9 个碱基对)

[0075] $N4$ = 出现率

[0076] $N4 = (N1 - ((N3 - 1) \times 2 \times N2)) / (4N3)$

[0077] 例如,

[0078]
$$\frac{76,002,917 - 8 \times 2 \times 38,556}{4^9} = 9\text{-mer}$$
 序列大约出现 287 次,

[0079] 或者

[0080]
$$\frac{76,002,917 - 7 \times 2 \times 38,556}{4^8} = 8\text{-mer}$$
 序列大约出现 1,151 次。

[0081] 因此,如在以上实例中所述,随机的 8-mer 和 9-mer 序列在具有所述 38,556 个

mRNA 序列的随机群体中分别地平均出现 1, 151 次和 287 次。

[0082] 在以上实例中,源于 38, 556 个基因的 76, 002, 917 个碱基对将相应于平均 1971bp 的转录物长度,各含有 1971-16 或者 1955 个 9-mer 靶序列。因此,作为统计学最小值,对于一个探针靶向一个基因而言,将需要 38, 556/1955/287 或 5671 个 9-mer 探针。

[0083] 然而,9-mer 序列的出现并非随机分布的。事实上,一小组序列有惊人的高出现率,高达从随机分布预期到的出现率的 30 倍以上。在根据优选标准选择的一个特定靶群体中,优选地,应当选择最常见的序列以增加所选的探针靶序列文库的覆盖率。如前所述,应当逐步实施选择,以便在起始的靶群体中以及在每一个选择步骤之后的剩余群体中都对选择的最常见的靶序列作出评价。

[0084] 在本发明的一个优选实施方案中,探针文库的靶标是整个表达的转录物组。

[0085] 由于逆转录酶反应的成功率缩短了与所用的 RT 引物的距离,而且,由于通常使用 poly-T 引物靶向 mRNA 的 poly-A 尾,故上述靶标可以进一步被限制到仅仅包括每个 mRNA 中的 1000 个最近侧的碱基。这可以导致对另一组最佳探针靶序列的选择以获得最佳覆盖率。

[0086] 同样地,上述靶标也可以被限制到仅仅包括基因内含子侧翼 50bp 的编码区序列,从而确保试验优选地仅监测 mRNA 而不监测基因组 DNA,或者,可以限制到仅仅包括不含有二-、三或四重复序列的区域,以避免探针或引物与不含已知等位基因变异的区域的反复结合,避免由于极高 GC 含量的靶序列或区域中的序列变异导致的引物或探针的错误退火,从而避免对 PCR 扩增的抑制作用。

[0087] 取决于每一个靶选择,最佳探针集合可以随着每一个靶选择中的靶序列的普遍性而发生变化。

[0088] 选择鉴定方式和鉴定单核酸

[0089] 本发明的另一部分涉及用于鉴定靶核酸的鉴定方法,该方法包括:

[0090] A) 向计算机系统输入唯一地确定所述靶核酸的核酸序列的数据,其中所述计算机系统包含容纳有至少一个本发明核酸探针文库的组成信息的数据库,而且,其中所述计算机系统还包含针对所述至少一个文库的每一探针的靶核酸序列数据库和/或还包含获取和比较核酸序列数据的手段;

[0091] B) 在该计算机系统中从所述至少一个文库鉴定探针,其中所述探针的序列存在于靶核酸序列中或存在于该靶核酸序列的互补序列中;

[0092] C) 在计算机系统中鉴定可以扩增所述靶核酸序列的引物;和

[0093] D) 提供指出步骤 B 中鉴定的探针和步骤 C 中鉴定的引物序列的输出结果,作为用于检测的特定鉴定方式。

[0094] 当期望快速而特异地鉴定特定核酸时,上述方法有几个优点。如果研究者已经获得了适宜的本发明多探针文库,该方法使得可以在数秒内获取有关应当应用于随后试验的文库探针的相关信息以及应当合成的引物的相关信息。由于引物对能够过夜合成,而探针的合成通常是十分费时而繁琐的,因此时间因素是重要的。

[0095] 为了有利于该方法的应用,可以标识探针文库(例如,通过可以基本上告知计算机系统该探针文库的组成的产品代码)。由此,步骤 A 包含向计算机系统输入标识所述至少一个核酸文库的数据,从该核酸文库将期望选出在特定检测方式中使用的成员。

[0096] 优选的输入界面是基于因特网的网络界面,因为该方法可以方便地存储在网络服

服务器上以允许已经获得本发明探针文库的用户进入。然而,该方法也可以用作可以安装在单个计算机或局域网上的安装型计算机应用的一部分。

[0097] 在该方法的优选实施方案中,选择步骤 C 中鉴定的引物以便最小化在 PCR 反应中扩增基因组核酸的可能性。这当然只有在样品可能含有基因组物质的情况下是恰当的。一种简单的使扩增基因组核酸的可能性最小化的方式是,在至少一个引物中包括在基因组 DNA 中被内含子打断的核苷酸序列。以此方式,引物将只引发已经拼接去除了内含子的转录物的扩增。

[0098] 本发明方法的另一优化是选择步骤 C 中的引物以最小化从靶核酸序列的 PCR 扩增获得的扩增子的长度,此外,还优选选择该引物以优化 GC 含量以便实施随后的 PCR。

[0099] 就此探针选择方法而言,可以以提供写在计算机可读介质上的实施本发明的指令的计算机程序产品形式,为终端用户提供选择检测手段的方法。因此,本发明还涉及包含本发明探针文库数据库和执行该计算机程序的应用程序的系统。

[0100] 该方法和计算机程序和系统允许定性或定量地确定样品中靶核酸的存在,其包括

[0101] i) 通过本发明的检测手段选择方法,鉴定用于检测靶核酸的特异手段,其中用于检测的该特定手段包含寡核苷酸探针和一组引物;

[0102] ii) 获得在步骤 i) 中鉴定的引物和寡核苷酸探针;

[0103] iii) 在来自步骤 ii) 的引物和寡核苷酸探针的存在下对样品实施分子扩增程序;和

[0104] iv) 基于步骤 iii) 的结果确定靶核酸的存在。

[0105] 方便地,步骤 ii) 中获得的引物是通过合成获得的,而且优选地该寡核苷酸探针获自本发明文库。

[0106] 分子扩增方法典型地是 PCR 或 NASBA 程序,但是任何用于核酸特异扩增(和可能地检测)的体外方法也均是有用的。优选的 PCR 方法是 qPCR(也称作实时逆转录 PCR 或动态 RT-PCR)。

[0107] 本发明的其它方面在以下进行讨论。

[0108] 附图简述

[0109] 图 1 在 (A) 图中说明常规长探针的使用,并在 (B) 图中说明来自根据本发明构建的文库的短多探针的特性和使用。该短多探针包含选择的识别片段,这样每个探针序列均可以用于检测和/或定量包含互补识别序列的几个不同靶序列。图 1A 显示根据现有技术的方法。图 1B 显示根据本发明一方面的方法。

[0110] 图 2 是显示用于设计根据本发明一方面的文库多探针序列的方法的流程图。该方法可以通过执行写在计算机可读介质上的计算机程序所提供的指令来实施。在一个方面,该程序指令通过包含序列数据库,例如表达序列数据库的系统来执行。

[0111] 图 3 说明在根据本发明一方面的 100 个探针的文库中靶向每一基因的探针的冗余性。Y 轴显示了人类转录物组中被文库中不同数量的探针靶向的基因数。明显地,在所有基因中大多数基因都被几个探针靶向。每基因的平均探针数为 17.4。

[0112] 图 4 显示选择的具有给定长度的超冗余寡核苷酸对人类转录物组的覆盖率。该图显示了大约 38,000 个人类 mRNA 序列中可以通过递增数量的不同长度的所选短多探针检测的百分数。该图说明不同长度的最佳选择的(即,超冗余的、非自我互补的、热稳定的)短

多探针对人类转录物组的理论覆盖率。该人类 (*Homo sapiens*) 转录物组序列从欧洲生物信息机构 (EMBL-EBI) 获得。最接近每个 mRNA 序列 3' 末端的 1000nt 区域 (3' 末端上游 50nt 至 1050nt) 被用于分析。在每个序列通过 PCR 扩增后, 扩增的双螺旋的两条链都被认为是探针文库中的多探针的有效靶标。排除甚至在具有 LNA 替代的情况下也不具有适当 T_m 的探针序列和自身互补的探针序列。

[0113] 图 5 显示寡核苷酸探针 EQ13992 的 MALDI-MS 谱, 说明 $[M-H]^- = 4121, 3Da$ 。

[0114] 图 6 显示在双标记探针试验中使用 9-mer 多探针检测靶序列的典型实时 PCR 曲线。结果来自使用 LNA 增强的双标记 9nt 长探针靶向同一基因内的不同 9-mer 序列的实时 PCR 反应。在 PCR 中分别分析三个不同的双标记探针, 产生 469、570 或 671 的 SSA4 扩增子 (每一个长 81 至 95nt)。图 a、b 和 c 分别显示双标记探针 469、570 和 671。每个探针仅仅检测其设计用于检测的扩增子。对于双标记探针 469、570 和 671, C_t 值分别是 23.7、23.2 和 23.4。加入 2×10^7 个拷贝的 SSA4 cDNA 作为模板。尽管在探针序列和其相应的引物对上均存在差异, 但结果之间高度相似, 这说明该试验是十分可靠的。

[0115] 图 7 显示具有 9-mer 和 10-mer 识别位点的分子信标的实时 PCR 曲线的实例。图 (A): 具有 10-mer 识别位点的分子信标探针检测 469SSA4 扩增子。仅仅在添加有 SSA4 cDNA 的样品中获得信号。得到 24.0 的 C_t 值。图 (B) 显示了使用具有 9-mer 识别位点的分子信标检测 570SSA4 扩增子的相似实验。仅仅在添加 SSA4 cDNA (2×10^7 个拷贝) 时获得信号。

[0116] 图 8 显示使用具有 9-mer 识别位点的 SYBR 探针靶向 570SSA4 扩增子的实时 PCR 曲线实例。仅仅在添加了 SSA4 cDNA (2×10^7 个拷贝) 的样品中获得信号, 而在不添加模板时检测不到信号。

[0117] 图 9 显示使用双标记探针试验原理得到的三个不同 9-mer 多探针的校准曲线。通过三个双标记探针检测不同拷贝数水平的 SSA4 cDNA。循环阈值 n_r 定义各 PCR 首次检测到信号时的循环数。三个线性回归线的斜率 (α) 和相关系数 (R^2) 为: $\alpha = -3.456$ & $R^2 = 0.9999$ (双标记的 469); $\alpha = -3.468$ & $R^2 = 0.9981$ (双标记的 570); $\alpha = -3.499$ & $R^2 = 0.9993$ (双标记的 671)。

[0118] 图 10 显示在暴露于热休克之前和之后在野生型酵母菌株以及突变菌株中利用 9-mer 双标记的多探针定量热休克蛋白, 其中在所述突变菌株中已经缺失了该相应的基因。显示了利用双标记的 570 探针对野生型 (wt) 酵母和 SSA4 敲除突变体中的 SSA4 转录物水平的实时检测。所述不同菌株在 30°C 培养直到收获 (-HS) 或者在收获前暴露于 40°C 30 分钟。在本实例中使用双标记的 570 探针。仅仅在 wt 型菌株中检测到该转录物, 在 +HS 培养物中该转录物极其大量。对于 +HS 和 -HS 培养物, C_t 值分别为 26.1 和 30.3。

[0119] 图 11 显示如何通过相同的 9-mer 探针检测一个以上基因同时不检测不含探针靶序列 (即, 与识别序列互补) 的核酸分子的一个实例。在 (a) 中, 双标记的 469 既检测 SSA4 (469 扩增子) 也检测 POL5 转录物, C_t 值分别为 29.7 和 30.1。从 APG9 和 HSP82 转录物未检测到信号。在 (b) 中, 双标记的 570 既检测 SSA4 (570 扩增子) 也检测 APG9 转录物, C_t 值分别为 31.3 和 29.2。从 POL5 和 HSP82 转录物未检测到信号。在 (c) 中, 双标记的 671 既检测 SSA4 (671 扩增子) 也检测 HSP82 转录物, C_t 值分别为 29.8 和 25.6。从 POL5 和 APG9 转录物未检测到信号。图注中显示在不同 PCR 中产生的扩增子。使用与图 10 所述试验相同量的 cDNA。仅仅使用来自未热休克的野生型酵母的 cDNA。

[0120] 图 12 显示在图 11 的实例所示的 PCR 反应中产生的部分扩增子的琼脂糖凝胶电泳,说明探针对包含识别序列的靶序列是特异的,但是其不与不含该靶序列的核酸分子杂交。泳道 1 含有 SSA4-469 扩增子 (81bp),泳道 2 含有 POL5 扩增子 (94bp),泳道 3 含有 APG9 扩增子 (97bp),泳道 4 含有 HSP82 扩增子 (88bp)。泳道 M 含有 50bp 梯作为大小指示剂。明显地,在所有 4 种情况中都形成了产物;然而,仅仅包含正确的多探针靶序列的扩增(即,SSA4-467 和 POL5)被双标记的探针 467 检测到。从泳道 1 和 2 中检测到的片段的大小差异,说明事实上产生并检测到了两种不同的扩增。

[0121] 图 13:优选的靶序列。

[0122] 图 14:其它优选的靶序列。

[0123] 图 15:长聚物(longmer)(阳性对照)。序列显示在 SEQ ID NOs. 32-46 中。

[0124] 图 16:选择探针和设计用于 qPCR 的引物的方案。

[0125] 图 17:在多探针数据集的计算中使用的程序的源代码。

[0126] 发明详述

[0127] 本发明涉及短寡核苷酸探针或多探针,这些探针或多探针被选择和设计用于检测、分类或表征和/或定量许多不同的靶核酸分子。这些多探针包含至少一个用于增加探针对识别序列的结合亲和力的非天然修饰(例如,LNA 核苷酸),其中所述识别序列是靶核酸分子的子序列。这些靶核酸分子在该识别序列之外是不同的。

[0128] 一方面,多探针包含至少一个用化学部分修饰以增加探针对识别序列——靶核酸序列的子序列——的结合亲和力的核苷酸。另一方面,探针包含至少一个非天然核苷酸和至少一个被化学部分修饰的核苷酸。再一方面,所述至少一个非天然核苷酸被化学部分修饰。本发明还提供包含所述探针的试剂盒、文库和其它组合物。

[0129] 本发明还提供 i) 针对给定的靶序列混合物选择和设计适宜的寡核苷酸探针的方法, ii) 具有这些能力的单个探针,和 iii) 经选择和设计以致能够用最少数量的探针序列检测、分类和/或定量最大数量的靶核苷酸的探针文库。根据本发明的每一探针因此均能够结合许多不同的靶,并且当与 PCR 试验中的一组特异引物组合时可以用于创建特异试验。

[0130] 本发明的优选核苷酸由大约 8 至 9 个核苷酸单元组成,其实质性部分包含稳定化核苷酸,例如 LNA 核苷酸。优选的文库含有大约 100 个这样的探针,这些探针经选择和设计可以表征一组特异核酸,例如,mRNA、cDNA 或基因组 DNA。该文库可以有多种用途,例如,基因表达分析、SNP 检测等(见例如图 1)。

[0131] 定义

[0132] 针对本发明公开中使用的特定术语,提供如下定义。

[0133] 除非上下文有清楚的另行说明,如本文中所用,单数形式“a”、“an”和“the”包括复数的情况。例如,术语“细胞”包括多个细胞,包括其混合物。术语“核酸分子”包括多个核酸分子。

[0134] 在本文中,术语“转录物组”(transcriptome)指由任何物种基因组中的所有被转录的元件构成的集合。

[0135] 除了 mRNA 外,该术语也指用于结构和调节目的的非编码 RNA。

[0136] 在本文中,术语“扩增子”指小的复制 DNA 片段。

[0137] 本文中,“样品”指从一个或多个生物体分离的组织或体液样品,包括但不限于,例如,皮肤、血浆、血清、脊髓液、淋巴液、滑液、尿、眼泪、血液细胞、器官、肿瘤以及体外细胞培养成分(包括但不限于通过细胞在细胞培养基中的生长而产生的条件培养基、重组细胞和细胞成分)的样品。

[0138] 本文中,“生物”指活的实体,包括但不限于,例如,人、小鼠、大鼠、果蝇(*Drosophila*)、秀丽隐杆线虫(*C. elegans*)、酵母、拟南芥(*Arabidopsis*)、斑马鱼、灵长类动物、家畜等。

[0139] 术语“SBC 核酸碱基”指“选择性结合的互补”核酸碱基,即,可以与其互补核酸碱基形成稳定的氢键但不能与其它 SBC 核酸碱基形成稳定氢键的修饰的核酸碱基。例如,SBC 核酸碱基 A' 可以与其互补的非修饰核酸碱基 T 形成稳定的氢键键合对。同样,SBC 核酸碱基 T' 可以与其互补的非修饰核酸碱基 A 形成稳定的氢键键合对。然而,SBC 核酸碱基 A' 和 T' 与碱基对 A'-T 和 A-T' 相比将形成不稳定的氢键键合对。同样地,SBC 核酸碱基 C 被称作 C', 可以与其互补的非修饰核酸碱基 G 形成稳定的氢键键合对,而 SBC 核酸碱基 G 被称作 G', 可以与其互补的非修饰核酸碱基 C 形成稳定的氢键键合对;而与碱基对 C'-G 和 C-G' 相比, C' 和 G' 将形成不稳定的氢键键合对。当形成 2 个或 2 个以上的氢键时获得稳定的氢键键合对,例如, A' 和 T、A 和 T'、C 和 G' 以及 C' 和 G 之间形成的对。当形成 1 个氢键或不形成氢键时获得不稳定的氢键键合对,例如, A' 和 T' 以及 C' 和 G' 之间的对。

[0140] 尤其有意义的 SBC 核酸碱基是 2,6-二氨基嘌呤(A', 也称作 D) 以及 2-硫代尿嘧啶(U', 也称作 ²⁵U) (2-硫代-4-氧代-嘧啶) 和 2-硫代胸腺嘧啶(T', 也称作 ²⁵T) (2-硫代-4-氧代-5-甲基-嘧啶)。图 4 说明 A-²⁵T 以及 D-T 对具有 2 个或 2 个以上的氢键,而 D-²⁵T 对形成单一一个(不稳定的)氢键。同样地,图 9 显示了 SBC 核酸碱基吡咯并-[2,3-d] 嘧啶-2(3H)-酮(C', 也称作 PyrroloPyr) 以及次黄嘌呤(G', 也称作 I) (6-氧代-嘌呤),其中 PyrroloPyr-G 以及 C-I 对各具有 2 个氢键,而 PyrroloPyr-I 对形成单一一个氢键。

[0141] “SBC LNA 寡聚体”指含有至少一个“LNA 单元”的“LNA 寡聚体”,其中该核酸碱基是“SBC 核酸碱基”。“具有 SBC 核酸碱基的 LNA 单元”是指“SBC LNA 单体”。一般来讲,SBC LNA 寡聚体包括除了 SBCLNA 单体外还含有其它修饰的或天然的核苷酸或核苷的寡聚体。“SBC 单体”是指具有 SBC 核酸碱基的非 LNA 单体。“等序列寡核苷酸”是指在 Watson-Crick 意义上与相应的修饰寡核苷酸具有相同序列的寡核苷酸,例如,序列 agTtcATg 等价于 agTscD²⁵Ug, 其中 s 等价于 SBCDNA 单体 2-硫代-t 或 2-硫代-u, D 等价于 SBC LNA 单体 LNA-D, ²⁵U 等价于 SBC LNA 单体 LNA²⁵U。

[0142] 如本文所用,术语“核酸”、“多核苷酸”和“寡核苷酸”指引物、探针、待检测的寡聚体片段、寡聚体对照和未标记的封闭寡聚体,并且应当属于多聚脱氧核糖核苷酸(含有 2-脱氧-D-核糖)、多聚核糖核苷酸(含有 D-核糖)、以及作为嘌呤或嘧啶碱基或修饰的嘌呤或嘧啶碱基的 N 糖苷的任何其它类型的多核苷酸。术语“核酸”、“多核苷酸”和“寡核苷酸”之间并不存在有意的长度区别,这些术语可以互换使用。这些术语仅仅指分子的一级结构。因此,这些术语包括双链和单链 DNA 以及双链和单链 RNA。寡核苷酸由相应于指定的核苷酸序列区域的大约至少 3 个核苷酸、优选至少大约 6 个核苷酸,更优选至少大约 8 至

30 个核苷酸的序列组成。“相应”指与该指定的序列相同或互补。

[0143] 物理上,寡核苷酸不一定来源于任何已有或天然的序列,而是可以以任何方式产生,包括化学合成、DNA 复制、逆转录或其组合。术语“寡核苷酸”或“核酸”意指半合成来源的或合成来源的基因组 DNA 或 RNA、cDNA 的多核苷酸,根据其来源或操作,该多核苷酸:(1) 与天然和其连接的所有或部分多核苷酸均不相连;和/或(2) 与非其天然连接的多核苷酸连接;和(3) 是天然中不存在的。

[0144] 由于单核苷酸之间的反应通过一个单核苷酸的戊糖环的 5' 磷酸与其相邻单核苷酸的 3' 氧按一定方向借助磷酸二酯键连接而形成寡核苷酸,故当寡核苷酸的 5' 磷酸未与单核苷酸戊糖环的 3' 氧连接时寡核苷酸的末端被称作“5' 末端”,而如果其 3' 氧未与随后的单核苷酸的戊糖环的 5' 磷酸连接时则该寡核苷酸的末端被称作“3' 末端”。在本文中,一段核酸序列,即使位于更大寡核苷酸的内部,也可以说具有 5' 和 3' 末端。

[0145] 当两个不同的非重叠的寡核苷酸与同一个线性互补核酸序列的不同区域退火时,一个寡核苷酸的 3' 末端将对着另一个寡核苷酸的 5' 末端;前者可以被称作“上游”寡核苷酸,而后者被称作“下游”寡核苷酸。

[0146] 术语“引物”可以指一个以上引物,其是指当被放置在催化与核酸链互补的引物延伸产物合成的条件下时,能够作为沿着互补链的合成的起始点起作用的寡核苷酸,而无论该寡核苷酸是天然的——以纯化的限制性消化产物的形式——还是合成产生的。所述条件包括在适宜的缓冲液(“缓冲液”包括作为辅因子或影响 pH、离子强度等的组分)中以及在适宜温度下存在 4 种不同的脱氧核糖核苷三磷酸以及聚合诱导剂例如 DNA 聚合酶或逆转录酶。引物优选是单链的以便获得最大扩增效率。

[0147] 如本文所用,术语“PCR 反应”、“PCR 扩增”、“PCR”和“实时 PCR”是可以互换使用的术语,其用于意指应用核酸扩增系统以扩增所检测的靶核酸。该系统的实例包括聚合酶链式反应(PCR)系统和连接酶链式反应(LCR)系统。本领域技术人员已知的和近来描述的其它方法有基于核酸序列的扩增(NASBA™, Cangene, Mississauga, Ontario)和 Q β 复制酶系统。通过所述扩增反应形成的产物可以进行实时检测或也可以不进行实时监测,或者仅在反应后进行终点测量。

[0148] 本文中,核酸序列的互补物是指当与该核酸序列对齐以致一个序列的 5' 末端与另一序列的 3' 末端配对时与该核酸序列存在反平行关系的寡核苷酸。在本发明的核酸中可以包括在天然核酸中不常见的碱基,这些碱基包括例如次黄嘌呤核苷和 7-脱氮杂鸟嘌呤(7-deazaguanine)。互补可以不是完全的;稳定的双螺旋可以含有错配碱基对或未配对的碱基。核酸技术领域的技术人员可以通过经验地考虑多个变量,包括例如寡核苷酸的长度、寡核苷酸中胞嘧啶和鸟嘌呤碱基的浓度百分数、离子强度以及错配碱基对的发生率,以确定双螺旋的稳定性。

[0149] 核酸双螺旋的稳定性可以通过熔解温度或“T_m”测定。在规定条件下特定核酸双螺旋的 T_m 是半数碱基对解离时的温度。

[0150] 本文中,术语“探针”指标记的寡核苷酸,其与靶核酸中的序列由于探针中的至少一段序列与该靶区中的序列互补而形成双螺旋结构。探针优选地不含有与用作聚合酶链式反应引物的序列互补的序列。一般地,探针的 3' 端被“封闭”以阻止探针掺入引物延伸产物。“封闭”可以通过使用非互补碱基或通过向最后一个核苷酸的 3' 羟基添加化学部分例

如生物素或甚至磷酸基团（取决于所选的部分，这可以通过还起到标记的作用而达到双重目的）来实现。

[0151] 术语“标记物”在本文中指可以用于提供可检测（优选地可定量的）信号并且可以与核酸或蛋白质连接的任何原子或分子。标记物可以提供能够通过荧光、放射性、比色、X 射线衍射或吸收、磁性、酶活性等检测的信号。

[0152] 本文中，“5′ → 3′ 核酸酶活性”或“从 5′ 至 3′ 的核酸酶活性”是指模板特异性核酸聚合酶的活性，包括常规与一些 DNA 聚合酶相关的 5′ → 3′ 外切核酸酶活性——通过该活性，核苷酸顺序地从寡核苷酸的 5′ 末端被除去（例如，大肠杆菌 DNA 聚合酶 I 具有该活性而 Klenow 片段则不具备该活性）——或者 5′ → 3′ 内切核酸酶活性——其中从 5′ 末端发生一个以上的核苷酸切除——或者两者。

[0153] 本文中，术语“热稳定核酸聚合酶”指与例如来自大肠杆菌的核苷酸聚合酶相比对加热具有相对较高的稳定性并且催化核苷聚合的酶。一般地，该酶在与靶序列退火的引物的 3′ 末端起始合成，并沿着模板朝着 5′ 方向前进，而且，如果其具有 5′ 至 3′ 的核酸酶活性时，将水解或置换介入其间的退火探针以释放标记和未标记的探针片段或完整探针，直到合成终止。分离自水生栖热菌 (*Thermus aquaticus*) 的代表性热稳定酶 (Taq) 描述在美国专利 4,889,818 中，在常规 PCR 中使用它的方法描述在 Saiki 等 (1988), *Science* 239 : 487 中。

[0154] 术语“核酸碱基”包括天然核酸碱基腺嘌呤 (A)、鸟嘌呤 (G)、胞嘧啶 (C)、胸腺嘧啶 (T) 和尿嘧啶 (U) 以及非天然核酸碱基例如黄嘌呤、二氨基嘌呤、8-氧代 -N⁶- 甲基腺嘌呤、7- 脱氮杂黄嘌呤、7- 脱氮杂鸟嘌呤、N⁴, N⁴- 桥亚乙基胞嘧啶、N⁶, N⁶- 桥亚乙基 -2, 6- 二氨基嘌呤、5- 甲基胞嘧啶、5-(C³-C⁶)- 炔基 - 胞嘧啶、5- 氟尿嘧啶、5- 溴尿嘧啶、假尿嘧啶、2- 羟基 -5- 甲基 -4- 三唑并吡啶、异胞嘧啶、异鸟嘌呤、次黄苷和 Benner 等的美国专利 5,432,272 以及 Susan M. Freier 和 Karl-Heinz Altmann, *Nucleic Acid Research*, 25 : 4429-4443, 1997 中描述的“非天然”核酸碱基。术语“核酸碱基”由此不仅包括已知的嘌呤和嘧啶杂环，还包括它们的杂环类似物和互变异构体。其它天然和非天然核酸碱基包括如下文献中描述的那些：美国专利 3,687,808 ; *Antisense Research and Application*, S. T. Crooke 和 B. Lebleu 编, CRC Press, 1993 中 Sanghvi 著 的第 15 章 ; *Englisch* 等, *Angewandte Chemie, International Edition*, 30 : 613-772, 1991 (见, 尤其是第 622 和 623 页) ; 以及 *Concise Encyclopedia of Polymer Science and Engineering*, J. I. Kroschwitz Ed., John Wiley & Sons, 858-859 页, 1990 ; Cook, *Anti-Cancer Drug Design* 6 : 585-607, 1991 ; 所有上述文献特此完整并入作为参考。

[0155] 术语“核苷碱基”或“核酸碱基类似物”也旨在包括可以象核苷碱基一样起作用的杂环化合物，包括某些“通用碱基”，这些碱基按最经典的意义不是核苷碱基但是其可以充当核苷碱基。尤其可以提及的“通用碱基”是 3- 硝基吡咯和 5- 硝基吡咯。其它优选化合物包括茚和吡啶并噁唑衍生物、茚基、茚基甲基丙三醇衍生物等。其它优选的通用甲基包括，吡咯、二唑或三唑衍生物，包括本领域已知的那些通用碱基。

[0156] “通用碱基”是指天然的或期望地非天然的化合物或部分，其可以与天然碱基（例如，腺嘌呤、鸟嘌呤、胞嘧啶、尿嘧啶和 / 或胸腺嘧啶）配对并且如本文所述具有差 15、12、10、8、6、4 或 2°C 或更少的 T_m。

[0157] “寡核苷酸”、“寡聚体”或“Oligo”指通过核苷间键连接的一条连续的单体（例如，杂环碱基的糖苷）链。在 Oligo 中两个连续单体之间的键由 2 至 4 个，期望地 3 个基团 / 原子组成，所述基团 / 原子选自： $-\text{CH}_2-$ 、 $-\text{O}-$ 、 $-\text{S}-$ 、 $-\text{NR}^{\text{H}}-$ 、 $>\text{C}=\text{O}$ 、 $>\text{C}=\text{NR}^{\text{H}}$ 、 $>\text{C}=\text{S}$ 、 $-\text{Si}(\text{R}'')_2-$ 、 $-\text{SO}-$ 、 $-\text{S}(\text{O})_2-$ 、 $-\text{P}(\text{O})_2-$ 、 $-\text{PO}(\text{BH}_3)-$ 、 $-\text{P}(\text{O},\text{S})-$ 、 $-\text{P}(\text{S})_2-$ 、 $-\text{PO}(\text{R}'')-$ 、 $-\text{PO}(\text{OCH}_3)-$ 和 $-\text{PO}(\text{NHR}^{\text{H}})-$ ，其中 R^{H} 选自氢和 C_{1-4} -烷基， R'' 选自 C_{1-6} -烷基和苯基。该键的说明性实例有 $-\text{CH}_2-\text{CH}_2-\text{CH}_2-$ 、 $-\text{CH}_2-\text{CO}-\text{CH}_2-$ 、 $-\text{CH}_2-\text{CHOH}-\text{CH}_2-$ 、 $-\text{O}-\text{CH}_2-\text{O}-$ 、 $-\text{O}-\text{CH}_2-\text{CH}_2-$ 、 $-\text{O}-\text{CH}_2-\text{CH}=($ （当用作与随后单体形成的键时包括 R^5 ））、 $-\text{CH}_2-\text{CH}_2-\text{O}-$ 、 $-\text{NRH}-\text{CH}_2-\text{CH}_2-$ 、 $-\text{CH}_2-\text{CH}_2-\text{NR}^{\text{H}}-$ 、 $-\text{CH}_2-\text{NR}^{\text{H}}-\text{CH}_2-$ 、 $-\text{O}-\text{CH}_2-\text{CH}_2-\text{NR}^{\text{H}}-$ 、 $-\text{NR}^{\text{H}}-\text{CO}-\text{O}-$ 、 $-\text{NR}^{\text{H}}-\text{CO}-\text{NR}^{\text{H}}-$ 、 $-\text{NR}^{\text{H}}-\text{CS}-\text{NR}^{\text{H}}-$ 、 $-\text{NR}^{\text{H}}-\text{C}(=\text{NR}^{\text{H}})-\text{NR}^{\text{H}}-$ 、 $-\text{NR}^{\text{H}}-\text{CO}-\text{CH}_2-\text{NR}^{\text{H}}-$ 、 $-\text{O}-\text{CO}-\text{O}-$ 、 $-\text{O}-\text{CO}-\text{CH}_2-\text{O}-$ 、 $-\text{O}-\text{CH}_2-\text{CO}-\text{O}-$ 、 $-\text{CH}_2-\text{CO}-\text{NR}^{\text{H}}-$ 、 $-\text{O}-\text{CO}-\text{NR}^{\text{H}}-$ 、 $-\text{NR}^{\text{H}}-\text{CO}-\text{CH}_2-$ 、 $-\text{O}-\text{CH}_2-\text{CO}-\text{NR}^{\text{H}}-$ 、 $-\text{O}-\text{CH}_2-\text{CH}_2-\text{NR}^{\text{H}}-$ 、 $-\text{CH}=\text{N}-\text{O}-$ 、 $-\text{CH}_2-\text{NR}^{\text{H}}-\text{O}-$ 、 $-\text{CH}_2-\text{O}-\text{N}=($ （当用作与随后单体形成的键时包括 R^5 ））、 $-\text{CH}_2-\text{O}-\text{NR}^{\text{H}}-$ 、 $-\text{CO}-\text{NR}^{\text{H}}-\text{CH}_2-$ 、 $-\text{CH}_2-\text{NR}^{\text{H}}-\text{O}-$ 、 $-\text{CH}_2-\text{NR}^{\text{H}}-\text{CO}-$ 、 $-\text{O}-\text{NR}^{\text{H}}-\text{CH}_2-$ 、 $-\text{O}-\text{NR}^{\text{H}}-$ 、 $-\text{O}-\text{CH}_2-\text{S}-$ 、 $-\text{S}-\text{CH}_2-\text{O}-$ 、 $-\text{CH}_2-\text{CH}_2-\text{S}-$ 、 $-\text{O}-\text{CH}_2-\text{CH}_2-\text{S}-$ 、 $-\text{S}-\text{CH}_2-\text{CH}=($ （当用作与随后单体形成的键时包括 R^5 ））、 $-\text{S}-\text{CH}_2-\text{CH}_2-$ 、 $-\text{S}-\text{CH}_2-\text{CH}_2-\text{O}-$ 、 $-\text{S}-\text{CH}_2-\text{CH}_2-\text{S}-$ 、 $-\text{CH}_2-\text{S}-\text{CH}_2-$ 、 $-\text{CH}_2-\text{SO}-\text{CH}_2-$ 、 $-\text{CH}_2-\text{SO}_2-\text{CH}_2-$ 、 $-\text{O}-\text{SO}-\text{O}-$ 、 $-\text{O}-\text{S}(\text{O})_2-\text{O}-$ 、 $-\text{O}-\text{S}(\text{O})_2-\text{CH}_2-$ 、 $-\text{O}-\text{S}(\text{O})_2-\text{NR}^{\text{H}}-$ 、 $-\text{NR}^{\text{H}}-\text{S}(\text{O})_2-\text{CH}_2-$ 、 $-\text{O}-\text{S}(\text{O})_2-\text{CH}_2-$ 、 $-\text{O}-\text{P}(\text{O})_2-\text{O}-$ 、 $-\text{O}-\text{P}(\text{O},\text{S})-\text{S}-$ 、 $-\text{O}-\text{P}(\text{S})_2-\text{O}-$ 、 $-\text{S}-\text{P}(\text{O})_2-\text{O}-$ 、 $-\text{S}-\text{P}(\text{O},\text{S})-\text{O}-$ 、 $-\text{S}-\text{P}(\text{S})_2-\text{O}-$ 、 $-\text{O}-\text{P}(\text{O})_2-\text{S}-$ 、 $-\text{O}-\text{P}(\text{O},\text{S})-\text{S}-$ 、 $-\text{O}-\text{P}(\text{S})_2-\text{S}-$ 、 $-\text{S}-\text{P}(\text{O})_2-\text{S}-$ 、 $-\text{S}-\text{P}(\text{O},\text{S})-\text{S}-$ 、 $-\text{S}-\text{P}(\text{S})_2-\text{S}-$ 、 $-\text{O}-\text{PO}(\text{R}'')-\text{O}-$ 、 $-\text{O}-\text{PO}(\text{OCH}_3)-\text{O}-$ 、 $-\text{O}-\text{PO}(\text{OCH}_2\text{CH}_3)-\text{O}-$ 、 $-\text{O}-\text{PO}(\text{OCH}_2\text{CH}_2\text{S}-\text{R})-\text{O}-$ 、 $-\text{O}-\text{PO}(\text{BH}_3)-\text{O}-$ 、 $-\text{O}-\text{PO}(\text{NHR}^{\text{H}})-\text{O}-$ 、 $-\text{O}-\text{P}(\text{O})_2-\text{NR}^{\text{H}}-$ 、 $-\text{NR}^{\text{H}}-\text{P}(\text{O})_2-\text{O}-$ 、 $-\text{O}-\text{P}(\text{O},\text{NR}^{\text{H}})-\text{O}-$ 、 $-\text{CH}_2-\text{P}(\text{O})_2-\text{O}-$ 、 $-\text{O}-\text{P}(\text{O})_2-\text{CH}_2-$ 和 $-\text{O}-\text{Si}(\text{R}'')_2-\text{O}-$ ；尤其期望的是其中的 $-\text{CH}_2-\text{CO}-\text{NR}^{\text{H}}-$ 、 $-\text{CH}_2-\text{NR}^{\text{H}}-\text{O}-$ 、 $-\text{S}-\text{CH}_2-\text{O}-$ 、 $-\text{O}-\text{P}(\text{O})_2-\text{O}-$ 、 $-\text{O}-\text{P}(\text{O},\text{S})-\text{O}-$ 、 $-\text{O}-\text{P}(\text{S})_2-\text{O}-$ 、 $-\text{NR}^{\text{H}}-\text{P}(\text{O})_2-\text{O}-$ 、 $-\text{O}-\text{P}(\text{O},\text{NR}^{\text{H}})-\text{O}-$ 、 $-\text{O}-\text{PO}(\text{R}'')-\text{O}-$ 、 $-\text{O}-\text{PO}(\text{CH}_3)-\text{O}-$ 和 $-\text{O}-\text{PO}(\text{NHR}^{\text{H}})-\text{O}-$ ，其中 R^{H} 选自氢和 C_{1-4} -烷基， R'' 选自 C_{1-6} -烷基和苯基。其它举例说明性实例可见于 Mesmaeker 等, *Current Opinion in Structural Biology* 1995, 5, 343-355 和 Susan M. Freier 和 Karl-Heinz Altmann, *Nucleic Acids Research*, 1997, 25 卷, 4429-4443 页。核苷间键的左侧作为取代基 P^* 与 5 元环的 3' 位置连接，而其右侧与前一单体的 5' 位置连接。

[0158] “LNA 单元”指单个 LNA 单体（例如，LNA 核苷或 LNA 核苷酸）或包括至少一个 LNA 单体的寡聚体（例如，寡核苷酸或核酸）。一般地，描述于 W099/14226 中的 LNA 单元是尤其期望掺入本发明寡核苷酸中的修饰核酸。此外，该核酸可以在 3' 和 / 或 5' 末端被本领域已知的任何修饰类型修饰。例如，任一端或两端可以用保护基加帽、连接柔性连接基团、连接反应性基团以有助于附着基质表面，等等。期望的 LNA 单元及其合成方法也公开在如下文献中：W000/47599、US6,043,060、US6,268,490、PCT/JP98/00945、W00107455、W00100641、W09839352、W00056746、W00056748、W00066604；Morita 等, *Bioorg. Med. Chem. Lett.* 12(1) : 73-76, 2002；Hakansson 等, *Bioorg. Med. Chem. Lett.* 11(7) : 935-938, 2001；Koshkin 等, *J. Org. Chem.* 66(25) : 8504-8512, 2001；Kvaerno 等, *J. Org. Chem.* 66(16) : 5498-5503, 2001；Hakansson 等, *J. Org. Chem.* 65(17) : 5161-5166, 2000；Kvaerno 等, *J. Org. Chem.* 65(17) : 5167-5176, 2000；Pfundheller 等, *Nucleosides Nucleotides* 18(9) : 2017-2030, 1999；和 Kumar 等, *Bioorg. Med. Chem. Lett.* 8(16) : 2219-2222, 1998。

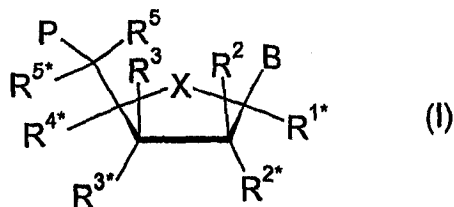
[0159] 优选的 LNA 单体，也称作“氧基-LNA”，是包括 PCT 公布 W003/020739 中所述双环化合物的 LNA 单体，其中下式 (I) 中所示 $\text{R}^{4'}$ 和 $\text{R}^{2'}$ 之间的桥在一起表示 $-\text{CH}_2-\text{O}-$ （甲基氧

基 LNA) 或 $-\text{CH}_2-\text{CH}_2-\text{O}-$ (乙基氧基 LNA, 也称作 ENA)。

[0160] 其它优选的 LNA 单体称作“硫基 -LNA”或“氨基 -LNA”, 包括 W099/14226 中所公开的双环结构, 其中下式 (I) 所示 $\text{R}^{4'}$ 和 $\text{R}^{2'}$ 之间的桥中的杂原子在一起表示 $-\text{CH}_2-\text{S}-$ 、 $-\text{CH}_2-\text{CH}_2-\text{S}-$ 、 $-\text{CH}_2-\text{NH}-$ 或 $-\text{CH}_2-\text{CH}_2-\text{NH}-$ 。

[0161] “LNA 修饰的寡核苷酸”是指包含至少一个下式 (I) 的 LNA 单体单元的寡核苷酸及其碱式盐和酸加成盐, 其具有以下描述的举例说明性修饰实例:

[0162]



[0163] 其中,

[0164] X 选自: $-\text{O}-$ 、 $-\text{S}-$ 、 $-\text{N}(\text{R}^N)-$ 、 $-\text{C}(\text{R}^6\text{R}^{6*})-$ 、 $-\text{O}-\text{C}(\text{R}^7\text{R}^{7*})-$ 、 $-\text{C}(\text{R}^6\text{R}^{6*})-\text{O}-$ 、 $-\text{S}-\text{C}(\text{R}^7\text{R}^{7*})-$ 、 $-\text{C}(\text{R}^6\text{R}^{6*})-\text{S}-$ 、 $-\text{N}(\text{R}^N)-\text{C}(\text{R}^7\text{R}^{7*})-$ 、 $-\text{C}(\text{R}^6\text{R}^{6*})-\text{N}(\text{R}^N)-$ 和 $-\text{C}(\text{R}^6\text{R}^{6*})-\text{C}(\text{R}^7\text{R}^{7*})$ 。

[0165] B 选自如上所述的修饰碱基, 例如, 任选取代的碳环芳基, 例如, 任选取代的茈或任选取代的茈基甲基丙三醇, 或任选取代的杂脂环或任选取代的杂芳族, 例如, 任选取代的吡啶并嘧啶、任选取代的吡咯、任选取代的二唑或任选取代的三唑部分; 氢、羟基、任选取代的 C_{1-4} -烷氧基、任选取代的 C_{1-4} -烷基、任选取代的 C_{1-4} -酰氧基、核酸碱基、DNA 嵌入剂、光化学活性基团、热化学活性基团、螯合基团、报道基团和配体。

[0166] P 表示用于和随后的单体形成核苷间键的基团位置或表示 5' 末端基团, 该核苷间键或 5' 末端基团任选地包括取代基 R^5 。取代基 R^2 、 R^{2*} 、 R^3 及 R^{3*} 中的一个基团 P^* , P^* 表示与前面的单体形成的核苷间键或 2' / 3' - 末端基团。 R^{1*} 、 R^{4*} 、 R^5 、 R^{5*} 、 R^6 、 R^{6*} 、 R^7 、 R^{7*} 、 R^N 取代基以及不表示 P^* 的 R^2 、 R^{2*} 、 R^3 和 R^{3*} 取代基各自可以表示选自如下的包含大约 1-8 个基团 / 原子的双基: $-\text{C}(\text{R}^a\text{R}^b)-$ 、 $-\text{C}(\text{R}^a)=\text{C}(\text{R}^a)-$ 、 $-\text{C}(\text{R}^a)=\text{N}-$ 、 $-\text{C}(\text{R}^a)-\text{O}-$ 、 $-\text{O}-$ 、 $-\text{Si}(\text{R}^a)_2-$ 、 $-\text{C}(\text{R}^a)-\text{S}-$ 、 $-\text{S}-$ 、 $-\text{SO}_2-$ 、 $-\text{C}(\text{R}^a)-\text{N}(\text{R}^b)-$ 、 $-\text{N}(\text{R}^a)-$ 和 $>\text{C}=\text{Q}$, 其中 Q 选自 $-\text{o}-$ 、 $-\text{S}-$ 和 $-\text{N}(\text{R}^a)-$, 且 R^a 和 R^b 各自独立地选自氢、任选取代的 C_{1-12} -烷基、任选取代的 C_{2-12} -链烯基、任选取代的 C_{2-12} -炔基、羟基、 C_{1-12} -烷氧基、 C_{2-12} -链烯基氧基、羧基、 C_{1-12} -烷氧羰基、 C_{1-12} -烷基羧基、甲酰基、芳基、芳基氧基羰基、芳基氧基、芳基羰基、杂芳基、杂芳基氧基羰基、杂芳基氧基、杂芳基羰基、氨基、单和双 (C_{1-6} -烷基) 氨基、氨基甲酰基、单和双 (C_{1-6} -烷基) 氨基羰基、氨基- C_{1-6} -烷基-氨基羰基、单和双 (C_{1-6} -烷基) 氨基- C_{1-6} -烷基-氨基羰基、 C_{1-6} -烷基-羰基氨基、脲基、 C_{1-6} -烷酰基氧基、sulphono、 C_{1-6} -烷基磺酰基氧基、硝基、叠氮基、硫烷基 (sulphanyl)、 C_{1-6} -烷基硫基、卤素、DNA 嵌入剂、光化学活性基团、热化学活性基团、螯合基团、报道基团和配体, 其中芳基和杂芳基可以被任选地取代, 并且其中两个季位取代基 R^a 和 R^b 一起可以是任选取代的亚甲基, 而且其中两个选自 R^a 、 R^b 非季位的或季位的取代基和存在的且不参与 P、 P^* 或所述双基的取代基 R^{1*} 、 R^2 、 R^{2*} 、 R^3 、 R^{3*} 、 R^{4*} 、 R^5 、 R^{5*} 、 R^6 、 R^{6*} 、 R^7 、 R^{7*} 之任一的可以一起形成选自前述相同类型的双基的连接的双基; 由此非季位取代基对与 (i) 和所述非季位取代基结合的原子及 (ii) 任何介于其间的原子一起形成单或双环实体。

[0167] 存在的但不参与 P、 P^* 或双基的取代基 R^{1*} 、 R^2 、 R^{2*} 、 R^3 、 R^{4*} 、 R^5 、 R^{5*} 、 R^6 、 R^{6*} 、 R^7 、 R^{7*} 各自

独立地选自：氢、任选取代的 C_{1-12} -烷基、任选取代的 C_{2-12} -链烯基、任选取代的 C_{2-12} -炔基、羟基、 C_{1-12} -烷氧基、 C_{2-12} -链烯基氧基、羧基、 C_{1-12} -烷氧羰基、 C_{1-12} -烷基羰基、甲酰基、芳基、芳基氧基羰基、芳基氧基、芳基羰基、杂芳基、杂芳基氧基羰基、杂芳基氧基、杂芳基羰基、氨基、单和双 (C_{1-6} -烷基) 氨基、氨基甲酰基、单和双 (C_{1-6} -烷基) 氨基羰基、氨基- C_{1-6} -烷基-氨基羰基、单和双 (C_{1-6} -烷基) 氨基- C_{1-6} -烷基-氨基羰基、 C_{1-6} -烷基-羰基氨基、脲基、 C_{1-6} -烷酰基氧基、sulphono、 C_{1-6} -烷基磺酰基氧基、硝基、叠氮基、硫烷基 (sulphanyl)、 C_{1-6} -烷基硫基、卤素、DNA 嵌入剂、光化学活性基团、热化学活性基团、螯合基团、报道基团和配体，其中芳基和杂芳基可以被任选地取代，并且其中两个孪位取代基可以一起表示氧代、硫代 (thioxo)、亚氨基或任选取代的亚甲基，或者可以一起形成由 1-5 个碳原子亚烷基链组成的螺双基，其中所述亚烷基链任选地被选自 -O-、-S- 和 $-(NR^N)-$ (其中 R^N 选自氢和 C_{1-4} -烷基) 的一个或多个杂原子 / 基团中断和 / 或终止，并且其中两个相邻 (非孪位) 取代基可以表示导致双键的另一键；而且，当存在但不参与双基时， R^{N*} 选自氢和 C_{1-4} -烷基。

[0168] 示例性 5'、3' 和 / 或 2' 末端基团包括 -H、-OH、卤代 (例如，氯代、氟代、碘代或溴代)、任选取代的芳基 (例如苯基或苄基)、烷基 (例如，甲基或乙基)、烷氧基 (例如甲氧基)、酰基 (例如，乙酰基或苯甲酰基)、芳酰基、芳烷基、羟基、羟基烷基、烷氧基、芳基氧基、芳烷氧基、硝基、氰基、羧基、烷氧羰基、芳氧羰基、芳烷氧基羰基、酰基氨基、芳酰基氨基、烷基磺酰基、芳基磺酰基、杂芳基磺酰基、烷基亚磺酰基、芳基亚磺酰基、杂芳基亚磺酰基、烷基硫基、芳基硫基、杂芳基硫基、芳烷基硫基、杂芳烷基硫基、脲基、氨基、氨基甲酰基、氨磺酰基、链烯基、炔基、保护基 (例如，甲硅烷基、4,4'-二甲氧基三苯甲基、单甲氧基三苯甲基、或三苯甲基 (trityl))、接头 (例如，含有胺、乙二醇、醌、蒽醌的接头)、可检测标记物 (例如，放射性标记或荧光标记) 和生物素。

[0169] 可以理解，此处提及核酸单元、核酸残基、LNA 单元或类似术语时包括单个的核苷单元和核苷酸单元以及寡核苷酸中的核苷单元和核苷酸单元。

[0170] “修饰的碱基”或其它类似术语指可以与天然碱基 (例如，腺嘌呤、鸟嘌呤、胞嘧啶、尿嘧啶和 / 或胸腺嘧啶) 配对和 / 或可以与非天然核酸碱基或核苷碱基配对的合成物 (例如，非天然的核酸碱基或核苷碱基)。期望地，修饰的碱基提供如所述的差 15、12、10、8、6、4 或 2°C 或更少的 T_m 。示例性修饰碱基描述在 EP1072679 和 W097/12896 中。

[0171] 术语“化学部分”指分子的一部分。“被化学部分修饰”由此指通过包括不寻常的化学结构对标准分子结构所进行的修饰。所述结构的附着可以是共价的或非共价的。

[0172] 术语在寡核苷酸探针中“包括化学部分”由此指附着分子结构。该化学部分包括但不限于共价地和 / 或非共价地结合的小沟结合剂 (MGB) 和 / 或嵌入核酸 (INA) -- 选自花菁染料、DAPI、SYBR Green I、SYBR Green II、SYBR Gold、PicoGreen、噻唑橙、Hoechst 33342、溴化乙啶、1-O-(1-苄基甲基)丙三醇和 Hoechst 33258。其它化学部分包括修饰的核酸碱基、核苷碱基或 LNA 修饰的寡核苷酸。

[0173] 术语“双标记的探针”指具有两个附着的标记物的寡核苷酸。一方面，一个标记物附着于探针分子的 5' 端，而另一标记物附着于分子的 3' 端。本发明的一个特定方面包含附着于一端的荧光分子和附着于另一端的能够通过荧光共振能量转移 (FRET) 淬灭该荧光团的分子。5' 核酸酶试验探针和一些分子信标是双标记的探针的实例。

[0174] 术语“5' 核酸酶试验探针”指可以被 DNA 聚合酶的 5' -3' 外切核酸酶活性水解

的双标记探针。在特定的 PCR 试验所用条件下,5' 核酸酶试验探针不一定被 DNA 聚合酶的 5' -3' 外切核酸酶活性水解。名称“5' 核酸酶试验”的使用与所观察到的水解程度无关,其并不表示实验者的任何预期。术语“5' 核酸酶试验探针”和“5' 核酸酶试验”仅仅指这样的试验,在该试验中并不特别关注避免所涉及的探针的水解。“5' 核酸酶试验探针”常常被称作“TaqMan 试验探针”,而“5' 核酸酶试验”则常常被称作“TaqMan 试验”。这些名称在本申请中可以互换使用。

[0175] 术语“寡核苷酸类似物”指能够识别特定的靶核苷酸序列的核酸结合分子。一种特定的寡核苷酸类似物是肽核酸 (PNA),在 PNA 中,寡核苷酸的糖磷酸主链被蛋白样主链替代。在 PNA 中,核酸碱基与不带电荷的聚酰胺主链结合,产生嵌合的假肽-核酸结构,该结构与核酸形式同形。

[0176] 术语“分子信标”指不可能受到 DNA 聚合酶的 5' -3' 外切核酸酶活性影响的单或双标记的探针。对探针、聚合酶或试验条件的特定修饰可以避免由于 DNA 聚合酶的 5' -3' 外切核酸酶活性而分离标记物或核苷酸组分。因此,检测的原则取决于分子信标与其靶序列结合后在标记物引起的信号上出现的可检测的差异。在本发明的一个方面,该寡核苷酸探针通过其 5' 和 3' 端的互补序列的介导而在选定的试验温度下形成分子内发夹结构。所述的寡核苷酸可以在一端结合荧光分子,在另一端结合当与发夹结构中的荧光团互相靠近时能淬灭该荧光团的分子。在本发明的另一方面,并不基于探针序列两端的互补结构形成发夹结构,而是可以由于一个或两个标记物与所形成的双螺旋结构之间的相互作用,或由于结合后探针空间构象的一般改变——或由于结合后标记物之间的降低的相互作用,而导致结合后所检测到的信号改变。分子信标的一个特定方面包含多个 LNA 残基以抑制 DNA 聚合酶的 5' -3' 外切核酸酶活性的水解作用。

[0177] 术语“多探针”在本文中指含有识别片段的探针,其中所述识别片段是与靶核酸分子中的识别序列充分互补以致可以在中等严谨条件下和 / 或适于 PCR、5' 核酸酶试验和 / 或分子信标分析 (或一般地任何基于 FRET 的方法) 的条件下与该序列结合的探针序列。所述条件是本领域技术人员熟知的。优选地,识别序列存在于所评价的多个序列,例如转录物组中。本发明的多探针可以含有非天然核苷酸 (“稳定化核苷酸”) 并且可以比含有相同序列但无稳定化修饰的探针具有更高的识别序列结合亲和力。优选地,多探针的至少一个核苷酸被化学部分修饰 (例如,在 PCR 反应的至少杂交阶段,共价地或以其它方式稳定地结合),以增加识别片段对识别序列的结合亲和力。

[0178] 本文中,对识别序列比含有相同序列但无稳定化核苷酸的探针具有增加的“结合亲和力”的多探针是指这样的探针,该探针的识别片段的结合常数 (K_d) 比双链分子的互补链的结合常数高。在另一优选实施方案中,该探针的识别片段的结合常数比双链分子中靶序列的识别序列的互补链的解离常数 (K_d) 高。

[0179] “多探针文库”或“多探针的文库”包含多个多探针,由此文库中探针的总数能够识别转录物组的大部分 (包括丰度最高的序列),这样转录物组中大约 60%、大约 70%、大约 80%、大约 85%,更优选地大约 90%,甚至更优选地 95% 的靶核酸可以被探针检测。

[0180] 当单体含有能够根据 Watson-Crick 碱基配对原则 (例如,G 与 C、A 与 T 或 A 与 U) 形成氢键或者能够形成其它氢键键合基序 (例如二氨基嘌呤与 T、次黄嘌呤与 C、假异胞嘧啶与 G 等) 的核酸碱基时,该单体被称作是“互补的”。

[0181] 术语“随后的单体”涉及沿着 5' 端方向的相邻单体,“前面的单体”涉及沿着 3' 端方向的相邻单体。

[0182] 本文中,术语“靶群体”是指多个不同的核酸序列,例如来自特定物种的基因组或其它核酸,包括基因组的转录物组,其中转录物组指任何物种的基因组的所有转录的元件的集合。通常,核酸群体中不同靶序列的数目为至少 100,但是清楚地,该数目常常高得多(超过 200、500、1000 和 10000—当靶群体是真核生物的转录物组时)。

[0183] 本文中,术语“靶核酸”指任何相关的具有单一特定序列的核酸,例如,生物核酸,例如来源于患者、动物(人或非人动物)、植物、细菌、真菌、古细菌(archae)、细胞、组织、生物体等。例如,当靶核酸来源于细菌、古细菌、植物、非人动物、细胞、真菌或非人生物时,本发明方法任选地还包括基于靶核酸的检测,选择该细菌、古细菌、植物、非人动物、细胞、真菌或非人生物。一个实施方案中,靶核酸来源于患者,例如人类患者。在该实施方案中,本发明任选地还包括基于靶核酸的检测,选择疗法、诊断疾病、或诊断疾病的遗传易感性。

[0184] 本文中,术语“靶序列”指任何靶核酸中的特定核酸序列。

[0185] 术语“严紧条件”在本文中是指在大约 $T_m - 5^\circ\text{C}$ (低于探针熔解温度(T_m)) 5°C) 至大约低于 $T_m 20^\circ\text{C} - 25^\circ\text{C}$ 的范围内出现的严紧性。本领域技术人员将理解,可以改变杂交的严紧性以鉴定或检测相同或相关的多核苷酸序列。杂交技术一般地可以参见 Nucleic Acid Hybridization, A practical Approach, Ed. Hames, B. D. 和 Higgins, S. J., IRL Press, 1985 ;Gall 和 Pardue, Proc. Natl. Acad. Sci., USA63 :378-383, 1969 ; 和 John 等, Nature 223 :582-587, 1969。

[0186] 多探针

[0187] 现参考图 1B,根据本发明的多探针优选为短的序列探针,其可以与多个不同靶核酸中存在的识别序列结合,由此该多探针特异地与靶核酸杂交但不与不含有该识别序列的核酸分子发生可检测水平的杂交。优选地,多探针的集合,或多探针文库,能够识别转录物组的大部分(包括丰度最高的序列),例如转录物组中的大约 60%、大约 70%、大约 80%、大约 85%、更优选地大约 90%、甚至更优选地 95%的靶核酸可以被探针检测。本发明的多探针包含“稳定化修饰”,例如非天然核苷酸(“稳定化核苷酸”),其比包含相同序列但无该稳定化序列的探针对识别序列具有更高的结合亲和力。优选地,多探针的至少一个核苷酸被化学部分修饰(例如,在 PCR 反应的至少杂交阶段,与探针共价地或以其它方式稳定地结合),以增加识别片段对识别序列的结合亲和力。

[0188] 一方面,具有 6 至 12 个核苷酸的多探针包含 1 至 6 个或甚至多达 12 个的稳定化核苷酸,例如 LNA 核苷酸。LNA 增强的探针文库含有识别短识别序列(例如,8 至 9 个核苷酸)的短探针。LNA 核酸碱基可以包含 α -LNA 分子(见例如,W000/66604)或 xylol-LNA 分子(见例如,W000/56748)。

[0189] 一方面,优选地,多探针与其识别序列结合时的 T_m 为大约 55°C 至 70°C 。

[0190] 另一方面,多探针包含一个或多个修饰的核酸碱基。修饰的碱基单元可以含有与核酸单元,例如呋喃糖基(furansyl)环的 1' 位通过接头,例如直链或支链亚烷基或亚链烯基基团连接环单元(例如,碳环单元如苊基)。亚烷基基团适宜地具有 1 个(即, $-\text{CH}_2-$)至大约 12 个碳原子,更典型地 1 至大约 8 个碳原子,甚至更典型地 1 至大约 6 个碳原子。亚链烯基基团适宜地具有 1、2 或 3 个碳-碳双键以及 2 至大约 12 个碳原子,更典型地 2 至大

约 8 个碳原子,甚至更典型地 2 至大约 6 个碳原子。

[0191] 本发明的多探针对于实施诸如实时 PCR 等试验是理想的,因为本发明的探针优选具有大约 25 个核苷酸、大约 15 个核苷酸、大约 10 个以下的核苷酸,例如 8 或 9 个核苷酸。优选地,多探针可以在 PCR 条件下与靶序列中的识别序列特异地杂交,并且优选地该识别序列存在于至少大约 50 个、至少大约 100 个、至少大约 200 个、至少大约 500 个不同的靶核酸分子中。本发明的多探针文库包含含有不同识别序列的多探针,这样任何两个多探针都可以与不同的靶核酸分子组杂交。在一方面,所述的靶核酸分子组包含一些相同的靶核酸分子,即,含有目的基因序列的靶核酸分子可以被一个以上的多探针结合。该靶核酸分子将含有至少两个不同的识别序列,这些序列可以重叠 1 个或多个多核苷酸,但少于包含 x 个核苷酸的识别序列的 x 个核苷酸。

[0192] 一方面,多探针文库包含多个不同的多探针,每一个不同的探针分别地定位在固体基质上的离散位置。在本文中,“定位”指被限制于或定址于所述位置上,这样,可以沿着在该位置检测到的杂交事件跟踪到具有已知序列身份的探针。定位的探针可以和基质稳定地结合或可以不和基质稳定地结合。例如,探针可以处于微滴定板孔中的溶液中,并由此被定位或定址于该孔。作为备选方案,或者此外,探针可以与基质稳定地结合,这样在一次或多次使用缓冲液洗涤该基质后探针仍保留在基质上的确定位置上。例如,探针可以通过化学方式与基质直接地或通过接头分子结合,其中所述接头分子可以是核酸序列、肽或对基质上的分子具有亲和力的任何类型的分子。

[0193] 或者,靶核酸分子可以定位在基质上(例如,以细胞、细胞裂解物或核酸的形式点在基质上)。

[0194] 一旦确定了适当的序列,优选地多 LNA 探针使用现有技术中(Tetrahedron 54 : 3607-30, 1998) 描述的商业上可获得的方法以及设备,通过化学方式合成。例如,可以使用固相亚磷酰胺方法制备短 LNA 探针(Caruthers 等, Cold Spring Harbor Symp. Quant. Biol. 47 : 411-418, 1982 ; Adams 等, J. Am. Chem. Soc. 105 : 661 (1983))。

[0195] 多探针文库中的多探针与一个或一个以上靶序列(优选地与多个靶序列)的杂交程度可以通过本领域已知的任何方法进行测定。如果没有可检测的杂交,杂交程度是 0。典型地,使用标记的信号核酸检测杂交。互补核酸或信号核酸可以通过典型地用于检测是否存在杂交的多核苷酸的几种方法中的任一种进行标记。最常见的检测方法是使用与标记的抗体结合的配体、荧光团或化学发光试剂。其它标记物包括可以充当特异结合对的成员用于标记的配体的抗体。标记的选择取决于所需的灵敏性与探针缀合的容易程度、稳定性要求和可用的设备。

[0196] 含有 LNA 的探针典型地在合成过程中进行标记。亚磷酰胺合成方法的灵活性还有利于容易地产生带有可用于该标准化学的所有商业接头、荧光团和标记分子的 LNA。LNA 也可以通过酶反应,例如激酶反应来标记。

[0197] 本发明的多探针可以含有单个标记物或多个标记物。在一个方面,所述多个标记物包含在多探针与靶序列发生杂交时可以彼此相互作用以产生信号或产生信号改变的成对标记物。

[0198] 另一方面,多探针包含荧光团部分和淬灭剂部分,这两个部分在多探针中所处的位置使得该探针的杂交状态可以通过来自该核酸的荧光信号的增加而区别于该探针的未

杂交状态。一方面,多探针除了含有识别元件外还含有第一和第二互补序列,这两个序列在探针未与靶分子中的识别序列杂交时将彼此特异地杂交,从而使得淬灭分子与所述报道分子足够接近以淬灭报道分子的荧光。靶分子的杂交使得淬灭分子与报道分子远离,导致与杂交量成比例的信号。

[0199] 另一方面,可以使用具有 5' 核酸酶活性的聚合酶检测核酸链的聚合。将荧光团和淬灭分子以充分接近的方式掺入探针中,以致在探针与其识别序列杂交时淬灭分子淬灭荧光团分子的信号。具有 5' 核酸酶活性的聚合酶对探针的切割将导致淬灭分子和荧光团分子分离以及随着核酸序列存在递增信号量。

[0200] 在本文中,术语“标记物”指报道基团,其本身可以被检测或者可以作为检测系列的一部分被检测。报道基团的功能性部分的实例是生物素、地高辛配基、荧光基团(能够吸收特定波长的电磁辐射,例如光或 x 射线,并随后以更短波长的辐射形式放出所吸收的能量的基团;举例说明实例有 DANSYL((5-二甲基氨基)-1-萘磺酰基)、DOXYL(N-氧基-4,4-二甲基噁唑烷)、PROXYL(N-氧基-2,2,5,5-四甲基吡咯烷)、TEMPO(N-氧基-2,2,6,6-四甲基哌啶)、二硝基苯基、吖啶、香豆素、Cy3 及 Cy5(Biological Detection Systems, Inc. 的商标)、赤藓红、香豆酸、7-羟基香豆素、得克萨斯红、罗丹明、四甲基罗丹明、Rox、7-硝基苯并-2-噁-1-二唑(NBD)、苝、荧光素、铕、钆、钇和其它稀土金属)、放射性同位素标记物、化学发光标记物(可以通过化学反应过程中释放的光而得到检测的标记物)、自旋标记物(与生物分子结合的自由基(例如,取代的有机氨基氧)或其它顺磁探针(例如, Cu^{2+} 、 Mg^{2+}),其可以通过使用电子自旋共振谱法检测)。尤其有意义的实例是生物素、荧光素、得克萨斯红、罗丹明、二硝基苯基、地高辛配基、钆、铕、Cy5、Cy3 等。

[0201] 靶核酸分子的适宜实例可以包含多种多样的真核和原核细胞,包括原生质体;或可以含有靶核酸的其它生物材料。因此,本发明的方法可以应用于组织培养动物细胞、动物细胞(例如,血液、血清、血浆、网织红细胞、淋巴细胞、尿、骨髓组织、脑脊液或从血液或淋巴制备的任何产物)或任何类型的组织活检物(例如,肌肉活检物、肝活检物、肾活检物、膀胱活检物、骨活检物、软骨活检物、皮肤活检物、胰腺活检物、肠道活检物、胸腺活检物、乳腺活检物、子宫活检物、睾丸活检物、眼活检物或脑活检物,例如在裂解缓冲液中匀浆)、归档的组织核酸、植物细胞或对渗透压休克敏感的其它细胞、以及细菌细胞、酵母、病毒、支原体、原生动物、立克次氏体、真菌和其它微生物细胞等等。

[0202] 可以分析被多个多探针识别的靶核酸以检测在靶核酸分子群体中占 10% 以下、大约 5% 以下、大约 1% 以下、大约 0.1% 以下、大约 0.01% 以下的序列(例如,特定的基因序列)。用于检测该序列的试验类型不是本发明的限制性特征,可以包括 PCR 或本领域已知的或开发用于检测靶核酸分子群体中所占不足 10% 的识别序列的一些其它适宜试验。

[0203] 一方面,检测较低丰度的识别序列的试验包括与至少一个引物杂交,其中所述引物能够特异地与该识别序列杂交,但是其不能实质性地与群体中大约 50 个以上、大约 25 个以上、大约 10 个以上、大约 5 个以上、大约 2 个以上(例如,该探针识别纯合基因序列的两个拷贝)的靶核酸分子,或 1 个以上的靶核酸(例如,样品中存在的单拷贝杂合基因序列的等位基因)杂交。在一个优选的方面,提供一对此类引物,该对引物位于多探针所鉴定的识别序列侧翼,即,位于识别序列的可扩增距离内,由此可以产生大约 40 至 5000 个碱基的扩增子,优选地产生 50 至 500 个或更优选地 60 至 100 个碱基的扩增子。可以标记这些引物

中的一个或多个。

[0204] 多种扩增反应是本领域技术人员熟知的,包括但不限于 PCR、RT-PCR、LCR、体外转录、滚环 PCR、OLA 等。也可以在多重 PCR 中使用多个引物以检测一组特定的靶分子。

[0205] 本发明还提供设计用于本发明方法和试剂盒中的多探针序列的方法。图 2 显示了略述该方法步骤的流程图。

[0206] 一方面,计算机 (in silico) 产生多个具有 n 个核苷酸的 n 聚体 (n -mer),以包含所有可能的 n 聚体。选择具有 $T_m \geq 60^\circ\text{C}$ 的 n 聚体的子集。另一方面,选择探针子集,在所述子集中的探针不能自身杂交,从而提供一系列候选 n 聚体或一个候选 n 聚体数据库。使用每一个 n 聚体的序列查询包含多个靶序列的数据库。优选地,该靶序列数据库含有表达的序列,例如人类 mRNA 序列。

[0207] 从用于查询数据库的该列候选 n 聚体,选择鉴别最大数量的靶序列的 n 聚体 (例如,包含与靶数据库中最大数量的靶序列的子序列互补的识别片段的 n 聚体),以产生 n 聚体 / 靶序列矩阵。将与最大数量的靶序列结合的 n 聚体的序列存储在最佳探针序列数据库中,并且从候选 n 聚体数据库中扣除这些 n 聚体。从靶序列数据库中除去被此第一组最佳探针鉴别的靶序列。然后针对剩余的候选探针重复该过程,直到鉴定到包含了能够覆盖大约 60% 以上、大约 80% 以上、大约 90% 以上,大约 95% 的靶序列的多个 n 聚体的一组多探针。可以使用每一步鉴定的最佳序列产生虚拟 (virtual) 多探针序列数据库。然后可以合成含有来自该多探针数据库的序列的多探针。

[0208] 另一方面,该方法还包括通过在剩余靶序列中进行查询和在最初的靶序列组中进行查询来评价将给定候选探针包括在不断增长的最佳候选探针组中的一般适用性。在一个优选方面,仅仅频繁地出现在剩余靶序列以及最初的靶序列中的探针识别序列才被加入到不断增长的该组最佳探针识别序列中。在一个最优选的方面,这通过如下方式实现:计算从这些查询获得的分数的乘积 (product),并选择具有最高乘积并且在相对于当前的靶进行的查询中仍属于具有 20% 的最佳得分的探针识别序列的探针识别序列。

[0209] 本发明还提供有利于上述方法实施的计算机程序产品 (见,例如图 2)。一方面,该计算机程序产品包含能够通过可与网络连接进行内存交流的计算机或用户设备来执行的程序指令。

[0210] 本发明还提供包含含有靶序列数据库的计算机存储器 and 用于执行该计算机程序产品提供的指令的应用系统的系统。

[0211] 包含多探针的试剂盒

[0212] 本发明的一个优选实施方案是多探针文库用于表征或检测或定量包含靶核酸的样品的试剂盒。一方面,该试剂盒含有针对其应用的计算机 (in silico) 操作方案。另一方面,该试剂盒含有与得到低价 DNA 引物的建议有关的信息。试剂盒中所含的探针可以具有上述的任何或所有特征。在一个优选方面,多个探针包含至少一个稳定化核酸碱基,例如 LNA 核酸碱基。

[0213] 另一方面,多个探针包含与至少一个化学部分偶联或稳定结合以增加该探针的结合稳定性的核苷酸。在再一优选方面,试剂盒包含多个不同的探针以覆盖具有不同靶序列的群体,例如转录物组的至少 60%。在一个优选方面,该转录物组是人类转录物组。

[0214] 另一方面,试剂盒包含至少一个用一个或多个标记物标记的探针。在再一方面,一

个或多个探针包含能够在基于 FRET 的试验中彼此相互作用的标记物,即,这些探针可以设计用于基于 5' 核酸酶或分子信标的试验。

[0215] 本发明的试剂盒允许用户快速且有效地开发用于许多不同核酸靶的试验。该试剂盒还可以包含一种或多种用于实施扩增反应例如 PCR 的试剂。

[0216] 实施例

[0217] 本发明现将参考如下实施例作进一步说明。应当理解,以下仅仅是举例的形式,并且可以对细节进行修改而仍落入本发明的范围内。

[0218] 在以下实施例中,探针的引用编号代表在以下合成实施例中所示的 LNA-寡核苷酸序列。

[0219] 实施例 1

[0220] 转录物组数据的来源

[0221] 从 ENSEMBL 获得人类转录物组 mRNA 序列。ENSEMBL 是 EMBL-EBI 与 Sanger Institute 的协作项目,用于开发软件系统以产生并维持对真核基因组的自动注释(见,例如,Butler, Nature 406(6794):333,2000)。ENSEMBL 主要由 Wellcome Trust 资助。应当注意的是,序列数据可以从包含表达序列的任何类型数据库获得,然而,ENSEMBL 是尤其有吸引力的,因为其给出最新的序列数据以及对后生动物基因组的最佳可能注释。2003 年 5 月 14 日自 ENSEMBL ftp 站点:

[0222] ftp://ftp.ensembl.org/pub/current_human/data 下载了“Homo_sapiens.cdna.fa”。该文件含有所有 ENSEMBL 转录物预报(即,37347 个不同序列)。从每一个序列中选取 3' 末端上游 50 个核苷酸至 3' 末端上游 1050 个核苷酸的区域。进一步相对于来自 NCBI 的参考序列(RefSeq)集中的人类 mRNA 序列,评价所选的探针序列组(见以下最佳方式)。RefSeq 标准被用作医学、功能性和多样性研究的基础;其为基因鉴定和表征、突变分析、表达研究、多态性的发现以及比较分析提供了稳定了参照。RefSeq 集旨在为主要研究的生物提供一组全面的、完整的非冗余序列,包括基因组 DNA、转录物(RNA)和蛋白质产物。对于来自 ENSEMBL 的 37347 个序列以及 RefSeq 集中的 19567 个序列,均发现了相似的覆盖率,即,说明数据库的类型不是本发明的限制性特征。

[0223] 实施例 2

[0224] 计算多探针数据集(dataset)(α 文库)

[0225] 设计在 UNIX 计算机上运行的特定软件以计算文库中的最佳探针集合。图 2 的流程图举例说明了该算法。

[0226] 分两步获得对转录物组的最佳覆盖。第一步,确定 n 聚物与基因的稀疏矩阵,由此可以容易地发现含有给定的 n 聚体的基因的数量。这通过使用 -p 选项以及 FASTA 格式输入的序列文件运行 getcover 程序来进行。

[0227] 第二步,基于第一步确定的矩阵,使用算法确定最佳覆盖率。为此目的,输入所述矩阵以运行程序,例如 getcover 程序。然而,本领域技术人员可以容易地设计出实现相似功能并用于执行相似步骤的程序。

[0228] 获得对转录物组的良好寡核苷酸覆盖

[0229] 1. 产生所有的 4^n 个 n 聚体,并计算预期的熔解温度。从该集合中除去具有 60°C 以下熔解温度以及具有高自身杂交能量的 n 聚体。这给出具有可接受的物理性质的一系列 n

聚体。

[0230] 2. 从 ENSEMBL 数据库中选取代表人类转录物组的一系列基因序列。

[0231] 3. 起始主循环 : 给定该 n 聚体和基因列, 通过在给定基因中鉴定所有的 n 聚体并将结果存储在矩阵中, 产生 n 聚体相对于基因的稀疏矩阵。

[0232] 4. 如果这是第一次迭代, 存储该矩阵的拷贝备用, 并命名为“总 n 聚体 / 基因矩阵”。

[0233] 5. 鉴定覆盖最多基因的 n 聚体并将其所覆盖的基因的数目存储为“max_gene”。

[0234] 6. 确定矩阵中剩余基因的覆盖率, 并将覆盖率达到至少 80% 的 max-gene 的基因存储在“具有良好覆盖率的 n 聚体集合”中。

[0235] 7. 最佳 n 聚体是其当前的覆盖率与总覆盖率的乘积为最大的 n 聚体。

[0236] 8. 从该 n 聚体集合中去除该最佳 n 聚体 (步骤 1)。

[0237] 9. 从该基因集合中去除此 n 聚体所覆盖的基因 (步骤 2)。

[0238] 10. 将该 n 聚体添加至最佳 n 聚体集合中, 从步骤 3 开始继续该程序直到不再能找到 n 聚体。

[0239] 用于计算多探针数据集的程序代码 (Niels Tolstrup 2003 的“getcover”1.0 版) 列在图 17 中。其由三个专利模块 : getcover.c, dyp.c, dyp.h 组成。

[0240] 该程序还并入了被 GNU Lesser General Public Licence 覆盖的 4 个模块 :

[0241] getopt.c, getopt.h, getoptl.c, getopt_init.c

[0242] /*Copyright (C) 1987, 88, 89, 90, 91, 92, 93, 94, 95, 96, 98, 99, 2000, 2001

[0243] Free Software Foundation, Inc.

[0244] 这些文件是 GNU C Library 的一部分。GNU C Library 是免费软件 ; 可以在 Free Software Foundation 公布的 GNU Lesser General Public License 的条件下对其进行重现分配和 / 或修改 */

[0245] 该软件用 aap 编译。用于制备该程序的 main.aap 文件同样列在图 17 中。

[0246] 为了运行已编译的程序, 使用以下命令 :

[0247] getcover-1 9 -p -f<h_sap_cdna_50_1050.fasta>h_sap_cdna_50_1050_19.stat

[0248] getcover -l 9 -i 1 -d 10 -t 60 -c -n -m -s <h_sap_cdna_50_1050_19.stat>

[0249] h_sap_cdna_50_1050_19.cover

[0250] 使用该计算机程序与用于执行上述算法的指令, 利用如下参数设置分析人类转录物组 :

[0251] L89 : 探针长度 = 8 或 9 个核苷酸

[0252] il : 包含分数 (inclusion fraction) = 100%

[0253] d10 : 靶双螺旋相对于自身双螺旋所需的 $\Delta T_m = 10^\circ\text{C}$

[0254] t60 : 靶双螺旋的最小 $T_m = 60^\circ\text{C}$

[0255] c : 还使用的互补靶序列

[0256] m80 : 使用乘积规则和 80% 规则从处理剩余靶的最一般探针中选择的最佳探针

[0257] n : LNA 核苷酸优选包括在识别片段的中心部分 ;

[0258] 结果鉴定到多探针靶序列数据库。

[0259] 该数据库中的靶序列是用于多探针文库的示例性最佳靶。这些最佳多探针列在下表 1 中,其包含 5' 荧光素荧光团和 3' Eclipse 淬灭剂(见下)。

[0260] 表 1:双标记的寡核苷酸探针

[0261]

cagcctcc	cagagcca	agctgtga	aggagggga
aggaggag	ctggaagc	cagagagc	tgtggaga
cccaggag	cagccaga	tgaggaga	ctggggaa
ctccagcc	cttctggg	acagtggg	ctcctgca
ctcctcca	ttctgcca	acagccat	tgagggtg
ctgtgtcc	aggagaga	tttctcca	aaggcagc
ctccagca	ttcctgca	cagtgtgt	ctgtggca
ctgtggg	tttgggga	aaagggga	agaagggc
cttctgg	caggcaga	tgtgggaa	tggtatga
acagcagc	ctgtgcca	actgggaa	ttctggca
cagctcca	ttccctgg	tcacagga	cagaaggc
ccccacc	aaccccat	ttctccc	atcccaga
tggtggtg	ctgcccag	aggtggaa	caggtgct
ttcctcca	ctgaggca	tgtggaca	ctgtctcc
ctgtctca	ctgttgt	tgagggcc	tgctgtga
tgagagaga	cagtgcca	atggtgaa	agctggat
aaggcaga	atggggaa	ctggaagg	tgagagagc
cagccagg	agggagag	caggcagc	cttggtgg
cagcagga	ctctgcca	tcaggagc	cacctgg
ctgtgtctg	ctgtgtgag	acacacac	cagccacc
agaggaga	ccctccca	catcttca	ctgtgacc
ctgtggct	aggaggca	cacctgca	agggggaa
cagtggct	cactgcca	ccagggcc	tgggacca

[0262]

ttctccca	ctgtgtgg	cagaggca	acagggaa
cctggagc	ttcccagt	ctgggact	ctgggcaa
cccagcag	tccagtgt	ctgcctgt	ctggagga
ttctcctg	ctctccc	tggaaggc	tccactgc
cttcctgc	cttcccca	ctgtgcct	ctgccacc
ccacctcc	ctctgcca	ctgtgctc	acagcctca
ttcctctg	cagcaggt	ctgtgagc	ctgtggtc
tggtgatg	ctccatcc	tcctcttc	cttcaggc
tgtggctg	tgctgtcc	ctcagcca	tctgggtc
cttctccc	tcctctcc	ctcttccc	cttggagc
ctgcctcc	ctctgcct	ctgggcac	ccaggctc
ctccttcc	ctggctgc	tgggcatc	tctctggt
tcctgctc	ccgccgcc	ctctggct	cttgggct
catcctcc	ctctctct	tgctgggc	ctgccatc
aggagctg	cagcctgg	ctgctctc	cactggga
tcctgctg	cagcagcc	ctggagtc	tgccctga
ctctcca	tgctggag	cttcagcc	ttggtggt
ccagccag	cttctccc	cttcagac	ttgggact
cagcccag	ttcttggc	tccaggtc	ctgctgga
ctccacca	tcctcagc	cagcatcc	caggagct
ctccagcc	aggagcag	cagaggct	ctcagcct
tggtctctg	ccaggagg	ctgccttc	ttctggct
caggcagc	cagcctcc	ctgggaga	ctgtctgc
ctgcctct	agctggag	cccagccc	ctgtccca
cttctgcc	ctgctgcc	cagctccc	tctgccca
ctgctccc	tggtctgtg	ccagccgc	ctggacac
tggtggaa	cctggaga	cctcagcc	ttgccatc
agctggga	ccagggcc	tcctcttct	cttcccct
ctgcttcc	ccaccacc	ctggctcc	cttgggca
cagcaggc	tctgtctgc	ccagggca	ttctggtc
tctggagc	cagccacc	ctccacct	ccgccgcc
catccagc	cagaggag	ctgccccca	cttcttctc
atggctgc	ctctcttc	tgggcagc	ttccctcc
ctcctgcc	caggagcc	ctggtctc	ttcctcaga
tggtggcc	tctggtcc	ctggggcc	tccaaggc
ctggggct	ctgtctcc	cagtggca	ttggggct

[0263] 这些超高丰度的 9-mer 和 8-mer 序列满足图 2 中的选择标准,即,

[0264] . 每一个探针的靶在人类转录物组的序列中占至少 6% (即,各 2200 个以上的靶序列,在最接近转录物 3' 末端的 1000nt 内靶向 800 个以上的序列)。

[0265] . 这些 9-mer 和 8-mer 序列不自身互补 (即,不可能形成探针双螺旋)。

[0266] 自身得分低于所估计的与靶形成的双螺旋的 T_m 至少 10。

[0267] . 与其靶序列形成的双螺旋具有 60℃或大约 60℃的 T_m 。

[0268] 这些 9-mer 和 8-mer 序列当组合在一起时覆盖人类转录物组中 > 98% 的 mRNA。

[0269] 下表 1a 中给出了表 1 中尤其优选的多探针形式：

[0270] 表 1a :100 个 LNA 替代的寡核苷酸

[0271]

cAgCCTCc	cAGAGCCa	aGCTGTGa	aGGAGGGa
aGGAGGAg	cTGGAAGc	cAGAGAGc	tGTGGAGa
ccCAGGAg	cAGCCAGa	tGAGGAGa	ctGGGGAAa
cTCCAgCc	cTTCTGGg	aCAGTGGa	cTCctGCa
cTCCTCCa	tTCTGCCa	aCAGCCAt	tGAGGtGg
cTgCTGcC	aGGAGAGa	tTTCTCCa	aAGGCAGc
cTCCAGCa	tTCCTGCa	cAGTGGTg	ctGTGGCa
cTGCTGgg	tTTGGGGa	aAAGGGGa	aGAAGGGc
cTTCCCTGg	cAGGCAGa	tGTGGGAa	tGGATGGA
aCAGCAGc	ctGTGCCa	aCTGGGAa	tTCTGGCa
caGCTCCa	tTCCCTGg	tCACAGGa	cAGAAGGc
cCCCACCCc	aACCCCAc	tTCCTCCc	aTCCCAGa
tGGTGGTg	ctGCCAg	aGGTGGAa	cAGGtGct
tTCCTCCa	cTGAGGCa	tGTGGACa	cTGTCTCc
cTGCTCCa	cTGctGGt	tGGAGgCc	tGCTGTGa
tGGAGAGa	cAGtGCCa	atGGTGAA	aGCTGGAt
aAGGCAGa	aTGGGGAAa	cTGGAAGg	tGGAGAGc
cAGCcAGg	aGGGAGAg	cAGGcAGc	cTTGGTGg
cAGCAGGa	cTctGCCa	tCAGGaGc	cACCTTGg
cTGTGCTg	cTGCTGAg	aCACACAC	cAgCCACc
aGAGGAGa	cCctCCCa	cATCTTCA	cTGTGACc
ctGTGGct	aGGAGGca	cACctGCa	aGGGGGAa
caGTGGct	cActGCCa	cCAGgGcc	tGgGACCa
tTCTCCCa	cTGTGTGg	cAGAGGCa	aCAGGGAAa

[0272] - 其中,小写字母表示 DNA 单体,大写字母表示 LNA 核苷酸。

[0273] > 95.0% 的 mRNA 序列在接近其 3' 末端的 1000nt 内 (距离 3' 末端 50 至 1050 位) 被靶向,而且 > 95% 的 mRNA 含有针对文库中一个以上探针的靶序列。在含有 37,347 个核酸序列的人类转录物组中鉴定到此 100 个多探针的超过 650,000 个的靶位点。在该转录物组中针对每一个转录物,多探针的平均数目是 17.4,中值是 14 个不同探针的靶位点。

[0274] 尽管未选择上述序列以针对特定的生物体进行优化,但是上述序列也是用于其它转录物组的优异探针选择。因此,尽管事实上上述探针设计用于仅在人类转录物组中鉴定 / 表征 / 定量转录物,但我们评价了上述文库对小鼠和大鼠基因组的覆盖率。例如,见表 2。

[0275]

表 2	转录物组		
	人	小鼠	大鼠
人类探针文库			
mRNA 序列数	37347	32911	28904
全长 mRNA 的覆盖率	96.7%	94.6%	93.5%
靠近 3'末端的 1000nt 的覆盖率	91.0%	—	—
两个探针覆盖至少	89.8%	80.2%	77.0%

[0276] nt ~核苷酸。

[0277] 实施例 3

[0278] 频繁出现的 9-mer 寡核苷酸对人类转录物组的预期覆盖率

[0279] 示例性试验数据（类似于图 6）说明，如果实时 PCR 试验所用的双标记探针用 LNA 进行增强，则可以将该探针的识别序列的长度根据该序列而减少到 8 或 9 个核苷酸。LNA 的独特双螺旋稳定化性质对于确保该短的双螺旋具有足够的稳定性（即， $T_m > 60^\circ\text{C}$ ）是必需的。该功能性实时 PCR 探针将几乎是纯的 LNA，在识别序列中具有 6 至 10 个 LNA 寡核苷酸。然而，该短识别序列使得可以使用相同的 LNA 探针检测和定量许多不同基因的丰度。通过根据图 2 所示算法适当地选择最佳（即，最常见的）8 或 9-mer 识别序列，可以覆盖含有大约 37347 个 mRNA 的人类转录物组（图 3）。

[0280] 图 3 显示了可以在各序列的近 3' 末端的长 1000nt 链（即，距离 3' 末端 50nt 至 1050nt 的序列）中被不同长度的优化探针检测到的人类转录物组中的 mRNA 序列的预期覆盖率，表示为人类转录物组中 mRNA 序列总数的百分数。这些探针被要求足够稳定（ $T_m > 60^\circ\text{C}$ ）并具有低的形成自身双螺旋的倾向，这就消除了许多 9-mer 以及甚至更多的 8-mer 探针序列。

[0281] 如果可以将具有给定长度的所有探针序列用作探针，则明显地可以通过这些最短的可能探针序列获得对转录物组的最佳覆盖。当仅有有限数量的探针（ < 55 个）被包括在文库中时，这确是如此（图 4）。然而，由于许多具有低 GC 含量的短探针不具有充分的热稳定性，故将其从文库中删除。可接受的 8-mer 探针的有限多样性在检测低 GC 含量的基因方面效力较低，因此，由 100 个不同的 9-mer 探针组成的文库比类似的 8-mer 文库对转录物组具有更好的覆盖。然而，最佳选择是由不同长度的序列组成的混合文库，例如以上列出的最佳文库模式提议。在图 4 中未显示该文库的覆盖率。

[0282] 所设计的含有 100 个最常见的 9-mer 和 8-mer 的探针文库，即，“人类 mRNA 探针文库”可以以方便的盒子或微量滴定板格式操作。

[0283] 可以改变此用于人类 mRNA 的 100 个探针的初始集合以产生相似文库试剂盒用于来自其它物种（小鼠、大鼠、果蝇（*Drosophila*）、秀丽隐杆线虫（*C. elegans*）、酵母、拟南芥、斑马鱼、灵长类动物、家畜等）的转录物组。构建这些新的探针文库几乎不需要费力，因为这些人类 mRNA 探针中大多数都可以重新用于这些新的文库试剂盒（表 2）。

[0284] 实施例 4

[0285] 文库中靶向每一基因的探针数目

[0286] 不仅所提议的文库中的有限数量的探针可以靶向人类转录物组的大部分,而且存在在大程度的冗余,因为大多数基因(几乎95%)可以被一个以上的探针检测。针对上述最佳文库模式中的100个探针,已经在人类转录物组(37347个基因)中鉴定到650,000个以上的靶位点。这导致每个探针平均6782个靶位点(即,转录物组的18%),范围是每个探针2527个序列至12066个序列。能够检测一个特定基因的平均探针数是17.4,中值为14。由此,在仅100个探针的该文库中,我们有至少14个探针针对所有人类mRNA序列中的50%以上序列。

[0287] 可以被文库中给定数目的探针所靶向的基因数显示在图4中。

[0288] 实施例5

[0289] 设计9-mer探针以证实可行性

[0290] 选择来自酵母(酿酒酵母(*Saccharomyces cerevisiae*))的SSA4基因用于表达试验,因为该基因转录水平可以通过热休克诱导,而且可以获得敲除表达的突变体。从人类转录物组中常见的9mer序列中选择3个不同的9-mer序列(表3)。这些序列存在于人类转录物组中1.8至6.4%的mRNA序列的3'末端附近。进一步的选择标准是中等水平的自身互补性和等于或大于60℃的T_m。所有三个序列都存在于SSA4ORF的末端1000个碱基内。通过合成在5'末端带有FITC荧光团而在3'末端带有Eclipse淬灭剂(EPOCH Biosciences)的这三个序列,构建了5'核酸酶试验探针。根据探针在ORF YER103W(SSA4)——其中位置1201被设定为位置1——中的位置,对探针进行命名。设计三对引物以产生各含有这三个探针序列之一的三个不重叠的扩增子。根据扩增子所包含的探针序列,命名扩增子。

[0291] 表3:设计的5'核酸酶试验探针和引物

[0292]

序列	探针名称	正向引物序列	反向引物序列	扩增子长度
aaGGAGAAG	双标记的 - 469	cgcgtttactttgaaaaattctg (SEQ ID NO: 1)	gcttccaatttctctggca tc (SEQ ID NO: 2)	81 bp
caAGGAAAg	双标记的 - 570	gccaagatgctataaatt-ggtag (SEQ ID NO: 3)	gggtttgcaacaccttct agttc (SEQ ID NO: 4)	95 bp
ctGGAGCaG	双标记的 - 671	tacggagctgcaggtggt (SEQ ID NO: 5)	gttgggccgttgctctggt (SEQ ID NO: 6)	86 bp

[0293] bp ~ 碱基对

[0294] 还设计了两个分子信标以检测SSA4 469-和SSA4 570序列,并分别命名为信标-469和信标-570。SSA4 469信标的序列是CAAGGAGAAGTTG(SEQ ID NO: 7, 10-mer识别位点),该序列应当使得该寡核苷酸形成分子内信标结构,具有通过5' -CAA和TTG-3'之间的LNA-LNA相互作用形成的茎。SSA4 570信标的序列是CAAGGAAAGttG(9-mer识别位点),其中该分子内信标结构可以在5' -CAA和t tG-3'之间形成。合成在5'末端带有荧光素荧光团而在3'末端带有Dabcyl淬灭剂的两个序列。

[0295] 还设计了一个SYBR Green标记的探针以检测SSA4 570序列,并将该探针命名为SYBR-探针-570。该探针的序列是CAAGGAAaG。合成在5'末端具有氨基-C6接头的该探针,在该5'末端将荧光团SYBR Green101(Molecular Probes)根据厂商说明书附着于其

上。在与靶序列杂交后,接头附着的荧光团应嵌入产生的 LNA-DNA 双螺旋区,造成来自 SYBR Green 101 的增加的荧光。

[0296] 表 4:序列

[0297]

EQ 号	名称	类型	序列	基因中的位置
13992	双标记的 - 469	5' 核酸酶检验探针	5'-Fluor-aaGGAGAAG-Eclipse-3'	469-477
13994	双标记的 - 570	5' 核酸酶检验探针	5'-Fluor-cAAGGAAAg-Eclipse-3'	570-578
13996	双标记的 - 671	5' 核酸酶检验探针	5'-Fluor-ctGGAGCaG-Eclipse-3'	671-679
13997	信标 - 469	分子信标	5'-Fluor-CAAGGAGAAGTTG-Dabcyl-3' (5'-Fluor-SEQ ID NO: 8-Dabcyl-3')	
14148	信标 - 570	分子信标	5'-Fluor-CAAGGAAAGttG-Dabcyl-3' (5'-Fluor-SEQ ID NO: 9-Dabcyl-3')	
14165	SYBR-探针 - 570	SYBR-探针	5'-SYBR101-NH2C6-cAAGGAAAg-3'	
14012	SSA4-469-F	引物	cgcgtttactttgaaaaattctg (SEQ ID NO: 10)	
14013	SSA4-469-R	引物	gdtccaatttctctggcatc (SEQ ID NO: 11)	
14014	SSA4-570-F	引物	gccaagatgctataaattggttag (SEQ ID NO: 12)	
14015	SSA4-570-R	引物	gggtttgcaacaccttctagttc (SEQ ID NO: 13)	
14016	SSA4-671-F	引物	tacggagctgcaggtggt (SEQ ID NO: 14)	
14017	SSA4-671-R	引物	gttgggccgttgtctggt (SEQ ID NO: 15)	
14115	POL5-469-F	引物	gcgagagaaaacaagcaagg (SEQ ID NO: 16)	
14116	POL5-469-R	引物	attcgtcttactggcatca (SEQ ID NO: 17)	
14117	APG9-570-F	引物	cagctaaaaatgatgacaataatgg (SEQ ID NO: 18)	
14118	APG9-570-R	引物	attacatcatgattaggaatgc (SEQ ID NO: 19)	
14119	HSP82-671-F	引物	gggtttgaacattgatgagga (SEQ ID NO: 20)	
14120	HSP82-671-R	引物	ggtgtcagctggaacctctt (SEQ ID NO: 21)	

[0298] 实施例 6

[0299] 合成、去保护和纯化双标记的寡核苷酸

[0300] 在自动 DNA 合成仪 (Expedite 8909 DNA 合成仪, PerSeptive Biosystems, 0.2 μmol 规模) 上使用亚磷酰胺方法 (Beaucage 和 Caurthers, Tetrahedron Lett. 22: 1859-1862, 1981) 以及 2- 氰基乙基保护的 LNA 和 DNA 亚磷酰胺 (Sinha 等, Tetrahedron Lett. 24: 5843-5846, 1983), 制备双标记的寡核苷酸 EQ13992 至 EQ14148 (表 4)。用 Eclipse

淬灭剂 (EQ13992-EQ13996) 或 Dabcyl (EQ13997-EQ14148) 以及 5' - 荧光素亚磷酰胺 (GLEN Research, Sterling, Virginia, USA) 衍生 CPG 固相支持物。相较于 DNA 亚磷酰胺, 针对 LNA 亚磷酰胺 (250s 偶联时间) 改变合成循环。在偶联步骤, 使用 1H- 四唑或 4, 5- 二氰基咪唑 (Proligo, Hamburg, 德国) 作为活化剂。

[0301] 使用 32% 氨水对该寡核苷酸进行去保护 (室温 1 小时, 然后 60°C 2 小时), 并通过 HPLC (Shimadzu-SpectraChrom 系列; Xterra™ RP18 柱, 10 ? m 7.8×150mm (Waters) 纯化。缓冲液 : A : 0.05M 乙酸三乙铵 pH7.4 ; B : 50% 乙腈水溶液。洗脱液 : 0-25min : 10-80% B ; 25-30min : 80% B)。寡核苷酸的组成和纯度通过 MALDI-MS (PerSeptive Biosystem, Voyager DE-PRO) 分析进行验证, 见表 5。图 5 是 EQ13992 的 MALDI-MS 谱, 显示 $[M-H]^- = 4121, 3\text{Da}$ 。这是本发明 9-mer 探针的典型 MALDI-MS 谱。

[0302] 表 5 :

[0303]

EQ#	序列	MW (计算值)	MW (实测值)
13992	5'-FItc-aaGGAGAAG-EQL-3'	4091,8 Da.	4091,6 Da.
13994	5'-FItc-cAAGGAAAg-EQL-3'	4051,9 Da.	4049,3 Da.
13996	5'-FItc-ctGGAGmCaG-EQL-3'	4020,8 Da.	4021,6 Da.
	5'-FItc-mCAAGGAGAAGTTG-dabcyl-3'		
13997	(5'-FItc-SEQ ID NO: 22-dabcyl-3')	5426,3 Da.	5421,2 Da.

[0304] 大写表示 LNA 单体 (A, G, mC, T), 其中 mC 是 LNA 甲基胞嘧啶。

[0305] 小写表示 DNA 单体 (a, g, c, t)。FItc = 荧光素; EQL = Eclipse 淬灭剂;

[0306] Dabcyl = Dabcyl 淬灭剂。MW = 分子量。

[0307] 实施例 7

[0308] 制备用 9-mer 探针进行检测的 SSA4 cDNA 标准

[0309] 在 PCR 试验中分析构建的 9mer 探针的功能, 其中检查能够检测不同 SSA4 PCR 扩增子的探针。体外转录表达载体 pTRIamp18 (Ambion) 中的 SSA4 基因下游区, 将得到的 cRNA 逆转录, 获得用作上述 PCR 反应的模板的 cDNA。SSA4 基因的下游区按如下方式克隆:

[0310] PCR 扩增

[0311] 使用酵母基因组 DNA 作为模板, 通过标准 PCR 扩增该部分酵母基因。从野生型酿酒酵母标准实验室菌株, 使用 Nucleon MIY DNA 提取试剂盒 (Amersham Biosciences), 根据供应商的说明书, 制备基因组 DNA。在 PCR 扩增的第一步, 使用含有限制性酶位点的正向引物和含有通用接头序列的反向引物。在该步中, 向扩增子的 3' 末端, 邻近终止密码子的地方, 添加 20bp。在扩增的第二步, 用含有 poly-T₂₀ 尾和限制性酶位点的嵌套引物更换反向引物。该 SSA4 扩增子含有 729bp 的 SSA4 ORF 加上一个 20bp 的通用接头序列以及一个 poly-A₂₀ 尾。

[0312] 所用 PCR 引物如下:

[0313] YER103W-For-SacI : acgtgagctcattgaaactgcaggtggtattatga (SEQ ID NO : 23)

[0314] YER103W-Rev-Uni : gatccccggaattgccatgctaataacacctttcaaccgttgg (SEQ ID NO : 24)

[0315] Uni-polyT-BamHI : acgtggatcctttttttttttttttttttgatccccggaattgccatg (SE

Q ID NO :25).

[0316] 质粒 DNA 构建体

[0317] 用限制性酶, EcoRI+BamHI, 切割 PCR 扩增子。使用 QuickLigation 试剂盒 (New England Biolabs), 根据供应商的说明书, 将该 DNA 片段与 pTRIamp18 载体 (Ambion) 连接, 并通过标准方法转化大肠杆菌 DH-5。

[0318] DNA 测序

[0319] 为了验证 PCR 扩增子的克隆, 使用 M13 正向引物和 M13 反向引物测序质粒 DNA, 并在 ABI 377 上分析。

[0320] 体外转录

[0321] 使用 Megascript T7 试剂盒 (Ambion), 根据供应商的说明书, 实施体外转录, 获得 SSA4 cRNA。

[0322] 逆转录

[0323] 用 $1\mu\text{g}$ cRNA 和 0.2U 逆转录酶 Superscript II RT (Invitrogen), 根据供应商的说明书——除了添加 20U Suprase-In (RNase 抑制剂—Ambion)——进行逆转录。在 QiaQuick PCR 纯化柱 (Qiagen) 上, 根据供应商的说明书, 使用所提供的 EB- 缓冲液进行洗脱, 纯化产生的 cDNA。测定洗脱的 cDNA 的 DNA 浓度, 并将该 cDNA 稀释成相应于 2×10^7 个拷贝 / μL 的 SSA4 cDNA 拷贝浓度。

[0324] 实施例 8

[0325] 用于双标记探针试验的方案

[0326] 根据如下方案 (表 6), 混合用于双标记探针 PCR 的试剂:

[0327] 表 6

[0328]

试剂	终浓度
H ₂ O	
GeneAmp 10 × PCR 缓冲液 II	1 ×
Mg ²⁺	5.5 mM
DNTP	0.2 mM
双标记的探针	0.1 或 0.3 μM
模板	1 μL
正向引物	0.2 μM
反向引物	0.2 μM
AmliTaq Gold	2.5 U
总共	50 μL

[0329] *) 5' 核酸酶试验探针的终浓度为 0.1 μM 而信标 / SYBR- 探针的终浓度为 0.3 μM 。

[0330] 在本实验中, 加入 2×10^7 个拷贝的 SSA4 cDNA 作为模板。在 DNAEngine Opticon® (MJ Research) 中, 使用如下 PCR 循环方案, 实施试验:

[0331] 表 7

[0332]

5'核酸酶试验

95℃ 7 分钟

&

40 个循环:

94℃ 20 秒

60℃ 1 分钟

荧光检测

信标& SYBR-探针试验

95℃ 7 分钟

&

40 个循环:

94℃ 30 秒

52℃ 1 分钟*

荧光检测

72℃ 30 秒

[0333] * 对于具有 9-mer 识别位点的信标 -570, 退火温度降至 44℃。

[0334] 表 6 中所示 PCR 反应的组成以及表 7 中所列 PCR 循环方案将称作标准 5' 核酸酶试验或标准信标试验条件。

[0335] 实施例 9

[0336] 9-mer 5' 核酸酶试验探针的特异性

[0337] 在如下试验中证实这些 5' 核酸酶试验探针的特异性, 在该试验中向 3 个不同的 PCR 反应分别加入各探针, 产生不同的 SSA4 PCR 扩增子。如图 6 所示, 每一个探针仅产生荧光信号以及其所设计检测的扩增子 (也见图 10、11 和 12)。重要的是, 不同探针具有十分相似的循环阈值 C_t (从 23.2 至 23.7), 说明这些试验和探针具有十分等同的效力。而且, 说明当将所述试验用于真实表达试验时, 这些试验应当检测到相似的表达水平。这是一个重要发现, 因为不同探针在性能上的变化性是不被期望的。

[0338] 实施例 10

[0339] 9-mer 和 10-mer 分子信标探针的特异性

[0340] 此外, 还针对分子信标设计概念, 证实了检测实时新产生的 PCR 扩增子的能力。设计用于 469 扩增子、具有 10-mer 识别序列的分子信标, 在 PCR 中存在用于产生该 469 扩增子的 SSA4 cDNA 模板和引物时, 产生了清晰的信号, 图 7A。所观察到的 C_t 值为 24.0, 十分类似于使用 5' 核酸酶试验探针获得的 C_t 值, 这再次说明不同探针的十分相似的灵敏度。当不加入 SSA4 模板时, 无信号产生。利用设计用于 570 扩增子、具有 9-mer 识别序列的分子信标, 得到相似结果, 见图 7B。

[0341] 实施例 11

[0342] 9-mer SYBR- 探针的特异性

[0343] 还针对 SYBR- 探针设计概念证实了检测新产生的 PCR 扩增子的能力。设计用于该 SSA4 cDNA 的 570 扩增子的此 9-mer SYBR- 探针, 在 PCR 中存在用于产生该 570 扩增子的 SSA4 cDNA 模板和引物时, 产生了清晰的信号, 见图 8。当不加入 SSA4 模板时, 无信号产生。

[0344] 实施例 12

[0345] 定量转录物拷贝数

[0346] 检测不同水平的基因转录物的能力是探针在真实表达试验中使用的基本需要。在

如下试验中通过三个 5' 核酸酶试验探针,证实了该需要得到满足,其中,在所述试验中将不同水平的来源于表达载体的 SSA4 cDNA 以及所述 5' 核酸酶试验探针之一一起加入不同 PCR 反应中(图 9)。根据标准 5' 核酸酶试验条件测定组成和循环条件。

[0347] 循环开始前 PCR 中的 cDNA 拷贝数通过循环阈值 C_t , 即, 首先检测到信号时的循环数, 来反映。在此, 信号仅定义为当荧光高于 3 至 10 个 PCR 循环中检测到的荧光的标准差 5 倍时的信号。结果显示在起始 cDNA 拷贝数的对数和 C_t 值之间存在良好的总体相关性(图 9)。这种相关性表现为取决于探针而具有 -3.456 至 -3.499 的斜率的直线以及 0.9981 至 0.9999 的相关系数。曲线的斜率反映 PCR 的效率, 其中, 假设在每一个 PCR 循环中扩增子加倍, 则 100% 效率相应于 -3.322 的斜率。目前 PCR 的斜率指示 94% 至 100% 的 PCR 效率。该相关系数和 PCR 效率与在使用 17 至 26 个核苷酸长度的 DNA 5' 核酸酶试验探针检测相同 SSA4 cDNA 水平的试验中获得值(结果未显示)同样高或更高。因此, 这些结果证实这 3 个 9-mer 5' 核酸酶试验探针满足用于真实表达探针的要求, 说明这些探针应当可以用于表达谱分析。

[0348] 实施例 13

[0349] 检测酵母中 SSA4 转录物水平

[0350] 在生长在不同培养条件(± 热休克)下的不同酵母菌株中, 检测 SSA4 转录物的表达水平。标准实验室酿酒酵母菌株在上述实验中用作野生型酵母。从 EUROSCARF(登记号 Y06101) 获得 SSA4 敲除突变体。该菌株在此称作 SSA4 突变体。两个酵母菌株均于 30℃ 生长于 YPD 培养基中, 直到 OD_{600} 等于 0.8A。将待进行热休克的酵母培养物转移至 40℃ 放置 30 分钟, 之后离心收获细胞并将沉淀冷冻在 -80℃。同时, 未热休克的细胞保留在 30℃ 生长 30 分钟, 之后按如上方式收获。

[0351] 使用 FastRNA 试剂盒(Bio 101) 以及 FastPrep 仪器根据供应商说明书从收获的酵母分离 RNA。

[0352] 使用 5 μg 锚定 oligo(dT) 引物以便在 1 μg 总 RNA 上引发反应, 并使用 0.2U 逆转录酶 Superscript II RT(Invitrogen), 根据供应商说明书——除了加入 20U Superase-In(RNase 抑制剂 -Ambion)——进行逆转录。2 小时孵育后, 70℃ 5 分钟使酶失活。用 10mM Tris 缓冲液 pH8.5 将 cDNA 反应物稀释 5 倍, 并通过在 MicroSpin™ S-400HR 柱(Amersham Pharmacia Biotech) 上纯化而去除寡核苷酸和酶。在实施表达试验前, 将 cDNA 稀释 20 倍。使用标准 5' 核酸酶试验条件——除了加入 2 μL 模板外——以及双标记 -570 探针, 进行该表达试验。模板是最初逆转录反应物的 100 倍稀释物。所用的 4 种不同 cDNA 模板来源于进行或未进行热休克的野生型或突变体。该试验产生预期的结果(图 10), 显示与未接受升高温度的野生型酵母 ($C_t = 30.3$) 相比, 热休克的野生型酵母的 SSA4 转录物水平 ($C_t = 26.1$) 增加。无论培养条件如何, 在突变酵母中均未检测到转录物。3.5 的 C_t 值差相应于热休克的野生型酵母相对于未热休克的野生型酵母而言诱导了 17 倍的表达水平, 该值接近文献(Causton 等, 2001) 中报道的大约 19 的值。这些值通过使用标准曲线获得, 其中所述标准曲线利用双标记的 -570 探针在具有已知量的 SSA4 转录物的定量实验中获得(见图 9)。这些实验说明, 这些 9-mer 探针能够检测到与公开的结果十分相符的表达水平。

[0353] 实施例 14

[0354] 使用单 9-mer 探针 (individual 9-mer probes) 进行多转录物检测

[0355] 为了验证这三个 5' 核酸酶试验探针检测其它基因的表达水平的能力, 选择了其中各存在所述探针序列之一的三个不同酵母基因。设计引物以扩增围绕探针序列的 60-100 碱基对区域。下表中显示了所选的三个酵母基因和相应的引物。

[0356] 表 8

[0357] 设计另一种表达试验

[0358]

序列/名称	匹配探针	正向引物 序列	反向引物 序列	扩增子 长度
YEL055C/POL5	双标记的 - 469	gcgagagaaaaca- agcaagg (SEQ ID NO: 26)	attcgtcttcactggcatca (SEQ ID NO: 27)	94 bp
YDL149W_AP9	双标记的 - 570	cagctaaaaatgat- gacaataatgg (SEQ ID NO: 28)	attacatcatgattaggga- atgc (SEQ ID NO: 29)	97 bp
YPL240C_HSP82	双标记的 - 671	gggtttgaacattg- atgagga (SEQ ID NO: 30)	ggtgtcagctggaacctctt (SEQ ID NO: 31)	88 bp

[0359] 来源于未热休克的野生型酵母的总 cDNA 用作该表达试验的模板, 该表达试验利用标准 5' 核酸酶试验条件——除了加入 2 μ L 模板外——进行。如图 11 所示, 根据表 8 所列的试验设计, 所有的三个探针均能够检测到基因的表达。使用表 8 所列之外的任何其它探针和引物组合均未检测到表达。对于 SSA4、POL5、HSP82 和 AP9, 在文献中可以获得表达数据 (Holstege, 等, 1998)。对于未热休克的酵母, 对于 SSA4 (0.8 个转录物拷贝 / 细胞)、POL5 (0.8 个转录物拷贝 / 细胞) 和 HSP82 (1.3 个转录物拷贝 / 细胞), 这些数据描述了相似的表达水平, 而 AP9 转录物水平稍低 (0.1 个转录物拷贝 / 细胞)。

[0360] 该数据与所获得的结果十分相符, 因为除了 HSP82 具有 25.6 的 Ct 值外, 所有其它基因均表示现出相似的 Ct 值。这提示, HSP82 转录物在这些实验所用的菌株中的丰度比文献显示的丰度要高。使用图 11a 所示针对双标记 -469 探针的 PCR, 进行琼脂糖凝胶电泳。琼脂糖凝胶 (图 12) 显示, 在未获得信号的反应中实际上产生了 PCR 产物, 因此缺乏来自这些反应的荧光信号并非由 PCR 失败所致。而且, 在表达试验中针对不同基因产生的不同长度的扩增子说明, 在表达试验中针对不同基因产生的信号实际上是特异于所涉及的基因的。

[0361] 实施例 15

[0362] 选择靶标

[0363] 使用来自 <http://www.ensembl.org/EnsMart> 的 EnsMart 软件 16.1 版, 从 Homo Sapiens NCBI 33 dbSNP115 Ensembl Genes 的所有外显子, 提取分别距各外显子的任一端的 50bp, 形成人类 Exon50 靶集合。使用 GetCover 程序 (参见图 17), 计算所有探针靶序列的出现率, 除去根据过度自我互补、过度 GC 含量等未通过选择标准的探针靶序列。在剩余序列中, 选择丰度最高的探针靶序列 (No. 1, 覆盖 3200 个靶), 之后选择出现率高于丰度

最高的探针靶序列的出现率的 0.8 倍 (3200×0.8) 或 2560 个靶的所有探针靶。从剩余的样品中计算各探针的新命中数, 而且计算每个探针靶的新命中数相较于已有选择的乘积以及该相同探针靶的总出现率, 并通过选择最高的乘积数来选择下一个丰度最高的探针靶序列。下表示例了探针靶的长度 (n)、序列 (nmer) 和在所有靶标中的出现率 (cover), 以及每个探针靶选择的新命中数 (Newhit)、Newhit 与 cover 的乘积 (newhit $\times$ cover) 和从所有累计的探针得到的靶群体中的累计命中数 (sum)。

[0364]

No	n	nmer	Newhit	Cover	newhit x cover	sum
1	8	ctcctct	3200	3200	10240000	3200
2	8	ctggagga	2587	3056	7905872	5787
3	8	aggagctg	2132	3074	6553768	7919
4	8	cagcctgg	2062	2812	5798344	9981
5	8	cagcagcc	1774	2809	4983166	11755
6	8	tgctggag	1473	2864	4218672	13228
7	8	agctggag	1293	2863	3701859	14521
8	8	ctgctgcc	1277	2608	3330416	15798
9	8	aggagcag	1179	2636	3107844	16977
10	8	ccaggagg	1044	2567	2679948	18021
11	8	tcctgctg	945	2538	2398410	18966
12	8	cttcctcc	894	2477	2214438	19860
13	8	ccgccgcc	1017	2003	2037051	20877
14	8	cctggagc	781	2439	1904859	21658
15	8	cagcctcc	794	2325	1846050	22452
16	8	tggtgtg	805	2122	1708210	23257
17	8	cctggaga	692	2306	1595752	23949
18	8	ccagccag	661	2205	1457505	24610
19	8	ccagggcc	578	2318	1339804	25188
20	8	cccagcag	544	2373	1290912	25732
21	8	ccaccacc	641	1916	1228156	26373
22	8	ctcctcca	459	3010	1381590	26832
23	8	ttctctg	534	1894	1011396	27366
24	8	cagcccag	471	2033	957543	27837
25	8	ctggctgc	419	2173	910487	28256
26	8	ctccacca	426	2097	893322	28682
27	8	cttctgc	437	1972	861764	29119
28	8	cttcagc	415	1883	781445	29534
29	8	ccacctcc	366	2018	738588	29900
30	8	ttctctg	435	1666	724710	30335
31	8	cccagccc	354	1948	689592	30689

[0365]

No	n	nmer	Newhit	Cover	newhit x cover	sum
32	8	tggtgatg	398	1675	666650	31087
33	8	tggctctg	358	1767	632586	31445
34	8	ctgccttc	396	1557	616572	31841
35	8	ctccagcc	294	2378	699132	32135
36	8	tgtggctg	304	1930	586720	32439
37	8	cagaggag	302	1845	557190	32741
38	8	cagctccc	275	1914	526350	33016
39	8	ctgcctcc	262	1977	517974	33278
40	8	tctgctgc	267	1912	510504	33545
41	8	ctgcttcc	280	1777	497560	33825
42	8	cttctccc	291	1663	483933	34116
43	8	cctcagcc	232	1863	432216	34348
44	8	ctccttcc	236	1762	415832	34584
45	8	cagcaggc	217	1868	405356	34801
46	8	ctgcctct	251	1575	395325	35052
47	8	ctccacct	215	1706	366790	35267
48	8	ctcctccc	205	1701	348705	35472
49	8	cttcccca	224	1537	344288	35696
50	8	cttcagcc	203	1650	334950	35899
51	8	ctctgcca	201	1628	327228	36100
52	8	ctgggaga	192	1606	308352	36292
53	8	cttctgcc	195	1533	298935	36487
54	8	cagcaggt	170	1711	290870	36657
55	8	tctggagc	206	1328	273568	36863
56	8	tcctgctc	159	1864	296376	37022
57	8	ctggggcc	159	1659	263781	37181
58	8	ctcctgcc	155	1733	268615	37336
59	8	ctgggcaa	185	1374	254190	37521
60	8	ctggggct	149	1819	271031	37670
61	8	tggtggcc	145	1731	250995	37815
62	8	ccagggca	147	1613	237111	37962
63	8	ctgctccc	146	1582	230972	38108
64	8	tgggcagc	135	1821	245835	38243
65	8	ctccatcc	161	1389	223629	38404
66	8	ctgcccga	143	1498	214214	38547
67	8	ttctgggc	155	1351	209405	38702

[0366]

No	n	nmer	Newhit	Cover	newhit x cover	sum
68	8	atggctgc	157	1285	201745	38859
69	8	tggtggaa	155	1263	195765	39014
70	8	tgctgtcc	135	1424	192240	39149
71	8	ccagccgc	159	1203	191277	39308
72	8	catccagc	122	1590	193980	39430
73	8	tcctctcc	118	1545	182310	39548
74	8	agctggga	121	1398	169158	39669
75	8	ctggtctc	128	1151	147328	39797
76	8	ttcccagt	142	1023	145266	39939
77	8	caggcagc	108	1819	196452	40047
78	8	tcctcagc	105	1654	173670	40152
79	8	ctggctcc	103	1607	165521	40255
80	9	tcctcttct	127	1006	127762	40382
81	8	tccagtgt	123	968	119064	40505

[0367] 实施例 16

[0368] 人类基因的 qPCR

[0369] 将该探针文库与如下实时 PCR 设计软件联合使用,其中所述软件能够:

[0370] . 通过独特的标识符或通过注册提交的核酸序列,识别输入的序列

[0371] . 鉴别可以靶向该核酸的所有探针

[0372] . 根据靶序列选择标准,例如最接近 3' 末端或最接近内含子-外显子边界,分拣探针

[0373] . 如果可以的话,根据 PCR 设计规则,设计位于靶向该核酸序列的探针侧翼的 PCR 引物

[0374] . 基于以上步骤,提出可用的实时 PCR 试验。

[0375] 通过 www.probelibrary.com 上的软件可以设计用于人类基因的有效和可靠的 qPCR 试验。

[0376] ProbeFinder 软件可以针对给定的人类基因快速而可靠地设计出最佳的 qPCR 探针和引物。

[0377] 该设计包括如下步骤:

[0378] 1) 确定内含子的位置

[0379] 通过选择跨 qPCR 的内含子,消除来自染色体 DNA 的噪音。通过 blast 检索人类基因组确定内含子。出现在 DNA 上但不出现在转录物上的区域被认为是内含子。

[0380] 2) 将探针文库与基因匹配

[0381] 这 90 个探针中的至少一个可以覆盖几乎所有人类转录物,识别序列标签的 LNA 修饰使得这种高覆盖率成为可能。

[0382] 3) 设计引物和选择最佳的 qPCR 试验

[0383] 用 'Primer3' (Whitehead Inst.For Biomedical Research, S.Rozen 和

H. J. Skaletsky), 设计引物。最后, 根据选定的规则排列探针的等级, 从而确保最佳可能的 qPCR。这些规则有利于跨扩增子的内含子以除去来自 DNA 污染 的假信号、并有利于用于可重复的相似试验的小的扩增子尺寸以及优化用于 PCR 的 GC 含量。

[0384] 实施例 17

[0385] 制备 ena⁻ 单体和寡聚体

[0386] 制备 ENA-T 单体并将其用于制备本发明的双标记探针。

[0387] 在如下序列中, X 代表 2' -O, 4' -C- 亚乙基 -5- 甲基尿嘧啶 (ENA-T)。该单体的合成在 W000/47599 中描述。用于掺入 5' -O- 二甲氧基三苯甲基 -2' -O, 4' -C- 亚乙基 -5- 甲基尿嘧啶 -3' -O- (2- 氰基乙基 -N, N, - 二异丙基) 亚磷酰胺的反应条件相应于用于制备实施例 6 中所述 LNA 寡聚体的反应条件。

[0388] 制备如下三个双标记的探针:

[0389]

EQ#	序列	MW (计算值)	MW (实测值)
16533	5' -Fitc-ctGmCXmCmCAg-EQL-3'	4002 Da.	4001 Da.
16534	5' -Fitc-cXGmCXmCmCA-EQL-3'	3715 Da.	3716 Da.
16535	5' -Fitc-tGGmCGAXXX-EQL-3'	4128 Da.	4130 Da.

[0390] X 表示 ENA-T 单体。小写字母表示 DNA 单体 (a, g, c, t)。Fitc = 荧光素 ;EQL = Eclipse 淬灭剂 ;Dabcyl = Dabcyl 淬灭剂。MW = 分子量。除了 'X' 之外的大写字母表示甲氧基 LNA 核苷酸。

[0391] 参考文献及注释

[0392] 1. Helen C. Causton, Bing Ren, Sang Seok Koh, Christopher T. Harbison, Elenita Kanin, Ezra G. Jennings, Tong Ihn Lee, Heather L. True, Eric S. Lander, 和 Richard A. Young (2001). 重建响应于环境改变的酵母基因组表达 . Mol. Biol. Cell 12 : 323-337 (2001)。

[0393] 2. Frank C. P. Holstege, Ezra G. Jennings, John J. Wyrick, Tong Ihn Lee, Christoph J. Hengartner, Michael R. Green, Todd R. Golub, Eric S. Lander, 和 Richard A. Young (1998), 解剖真核基因组的调节线路, Cell 1998 95 : 717-728。

[0394] 3. Simeonov, Anton 和 Theo T. Nikiforov, 使用短的固定荧光标记的核酸 (LNA) 探针以及荧光偏振检测进行单核苷酸多态性基因分型, Nucleic Acid Research, 2002, Vol. 30, No. 17 e 91。

[0395] 本领域技术人员可以改变、修饰和以其它方式实施本文的描述而不偏离此处描述和所要求保护的本发明的精神和范围, 这些改变、修饰和实施方式均包括在本发明的范围内。

[0396] 以上公开的参考文献、专利、专利申请和国际申请完整地并入此处作为参考。

序列表

<110>Exiqon A/S

<120> 用于分析核酸混合物的探针、文库和试剂盒及其构建方法

<130>15769PCT00

<160>46

<170>PatentIn version 3.2

<210>1

<211>23

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>1

cgcgtttact ttgaaaaatt ctg

23

<210>2

<211>20

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>2

gcttccaatt tcttgcatc

20

<210>3

<211>25

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>3

gcccaagatg ctataaattg gttag

25

<210>4

<211>23

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>4

gggttttgcaa caccttctag ttc

23

<210>5

<211>18

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>5

tacggagctg caggtggt

18

<210>6

<211>18

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>6

gttgggccgt tgtctggt

18

<210>7

<211>13

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>7

caaggagaag ttg

13

<210>8

<211>13

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>8

caaggagaag ttg

13

<210>9

<211>12

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>9

caaggaaagt tg

12

<210>10

<211>23

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>10

cgcgtttact ttgaaaaatt ctg

23

<210>11

<211>20

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>11

gcttccaatt tcctggcatc

20

<210>12

<211>25

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>12

gcccaagatg ctataaattg gttag

25

<210>13

<211>23

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>13

gggtttgcaa caccttctag ttc

23

<210>14

<211>18

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>14

tacggagctg caggtggt

18

<210>15

<211>18

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>15

gttgggccgt tgtctggt

18

<210>16

<211>20

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>16

gcgagagaaa acaagcaagg

20

<210>17

<211>20

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>17

attcgtcttc actggcatca 20

<210>18

<211>25

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>18

cagctaaaaa tgatgacaat aatgg 25

<210>19

<211>23

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>19

attacatcat gattaggga tgc 23

<210>20

<211>21

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>20

gggtttgaac attgatgagg a 21

<210>21

<211>20

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>21

ggtgtcagct ggaacctctt

20

<210>22

<211>13

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<220>

<221> 修饰的碱基

<222>(1)..(1)

<223>N 是 LNA 甲基胞嘧啶

<220>

<221>misc_ 特征

<222>(1)..(1)

<223>n 是 a, c, g, 或 t

<400>22

naaggagaag ttg

13

<210>23

<211>35

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>23

acgtgagctc attgaaactg caggtggtat tatga

35

<210>24

<211>44

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>24

gatccccggg aattgccatg ctaatcaacc tcttcaaccg ttgg 44

<210>25

<211>50

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>25

acgtggatcc tttttttttt tttttttttt gatccccggg aattgccatg 50

<210>26

<211>20

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>26

gcgagagaaa acaagcaagg 20

<210>27

<211>20

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>27

attcgtcttc actggcatca

20

<210>28

<211>25

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>28

cagctaaaaa tgatgacaat aatgg

25

<210>29

<211>23

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>29

attacatcat gattaggga tgc

23

<210>30

<211>21

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>30

gggtttgaac attgatgagg a

21

<210>31

<211>20

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>31

ggtgtcagct ggaacctctt

20

<210>32

<211>164

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>32

caccgttcgg catatccata ttccccacag ccaccaccag gaaggcagca gccaggagga 60

gcagcctcct cagagaagca gcctggagac ttctccagc tccagggccg ccgcctgctg 120

gagcagcagc accagaagag ggggaggtac ggttggttgt acga 164

<210>33

<211>108

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>33

tggcggacgc acaccgctta cccctgctgg aggaagctga ggaggagcag cctggagcag 60

cagcagccag ctccgccgcc aggaagccga ctcacgggcc acgcatta 108

<210>34

<211>115

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>34

gggtgcgacc	gtgagtcaat	ggtctccagg	aggctgtctt	ctggtgctgc	tcctctgctg	60
cctccagctt	ctctggccct	ggtggtggct	gtgggtaatg	cgtggcccgt	gagtc	115

<210>35

<211>106

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>35

attgactcac	ggtcgcacca	aactctgctg	ggctgcctgg	aagctccagg	agaacttcca	60
gccagctcct	ccaccagcag	gaagaataac	cgtggaacgc	ggtcac		106

<210>36

<211>124

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>36

atacccatcc	aaggcgtccc	taaaggaggc	agaggaaggg	agctgccttc	ccagcccttc	60
tcccagcaca	gcagagcaga	gccacctcca	gccacatcac	caaaatgacc	gcgttccacg	120
gtta						124

<210>37

<211>115

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>37

attgactcac ggtcgcacca aacctggaag gcagaggaac tgcctcctcc accatcacca 60
ctgctgggct gggaagcttc cagcacagca ggaaataacc gtggaacgcg gtcatt 115

<210>38

<211>121

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>38

atacccatcc aaggcgtecc taaacttctc ccagagccac ctccagccag ccacaccagc 60
agagcaggaa ggagctgcct ggagcagctc ccaggagaaa aatgaccgcg ttccacgggt 120
a 121

<210>39

<211>115

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>39

attgactcac ggtcgcacca aattcctctg ccttcctgct ctgctgggag aaggaggtgg 60
tgatgtggct ggaaggaggc agctccagga gaaaataacc gtggaacgcg gtcatt 115

<210>40

<211>114

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>40

atacccatcc aaggcgtecc taaacttcca ggcagctccc tccagccagc aggacttccc 60

agcccagctc ctccaccage acagcagage caaaatgacc gcgttccacg gtta 114

<210>41

<211>114

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>41

ttagggacgc cttaggatggg tatggctgag ggggctggct cctgcatcct cttctgcctc 60

tgctcccage tgagccatgc cctggcttcc accaattgcc gacccaccgg gata 114

<210>42

<211>122

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>42

attcgctacg gcccacaccc ttactccacc tcttgcccca ctggggctga agtccagtgt 60

ctggagctgc ttcccagtgg gcagccatcc agcaggccac catatcccgg tgggtcggca 120

at 122

<210>43

<211>124

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>43

taaggtgttg ggccgtagcg aatcgctctg ccactggggc ctggtctcca tctctcctc 60

cctgggcaac ctgctgtcct tggcagtggg gaagctgtgc caattgtcct ccgcccggac 120

tcat 124

<210>44

<211>122

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>44

ttagggacgc cttggatggg tatctctgcc actggctcca gatcctcttc tgccccactg	60
ccatgggcag ctggggcctc ctccctccac ctggcttccc caattgccga cccaccggga	120
ta	122

<210>45

<211>118

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>45

attcgctacg gcccaacacc ttacctcagc ccagctcca tccagccgcc aaggactggt	60
ctcctgccct gggcaactgg gaatggctgc ttccaccata tcccgggtggg tcggcaat	118

<210>46

<211>124

<212>DNA

<213> 人工序列

<220>

<223> 合成序列

<400>46

taaggtgttg ggccgtagcg aatctgcctc ttcagccgct ctgctcccag ctgagccatc	60
cagtgtgcag gagaggacag caggtggcac agcaggccac caattgtcct ccgcccggac	120
tcat	124

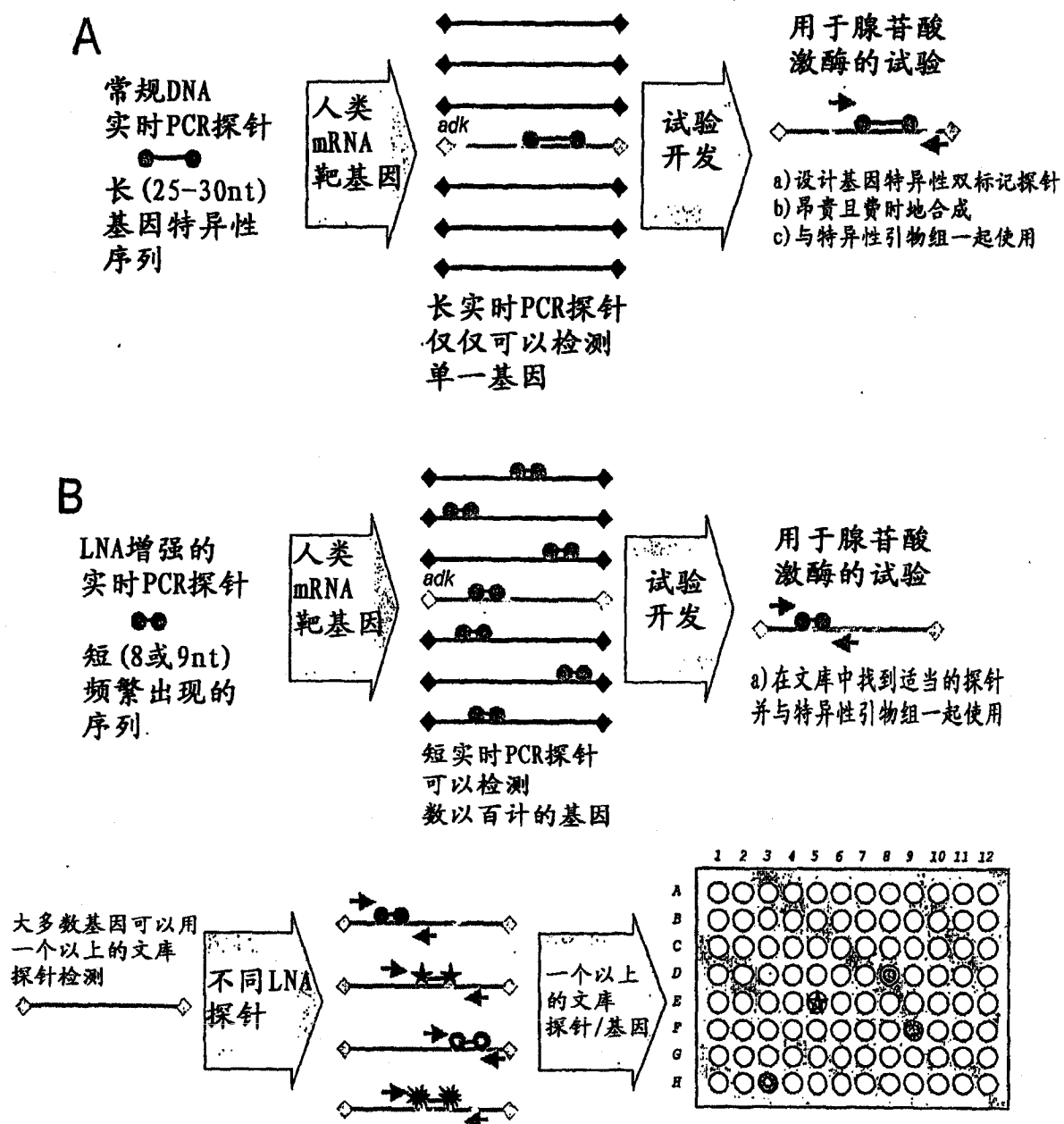


图1

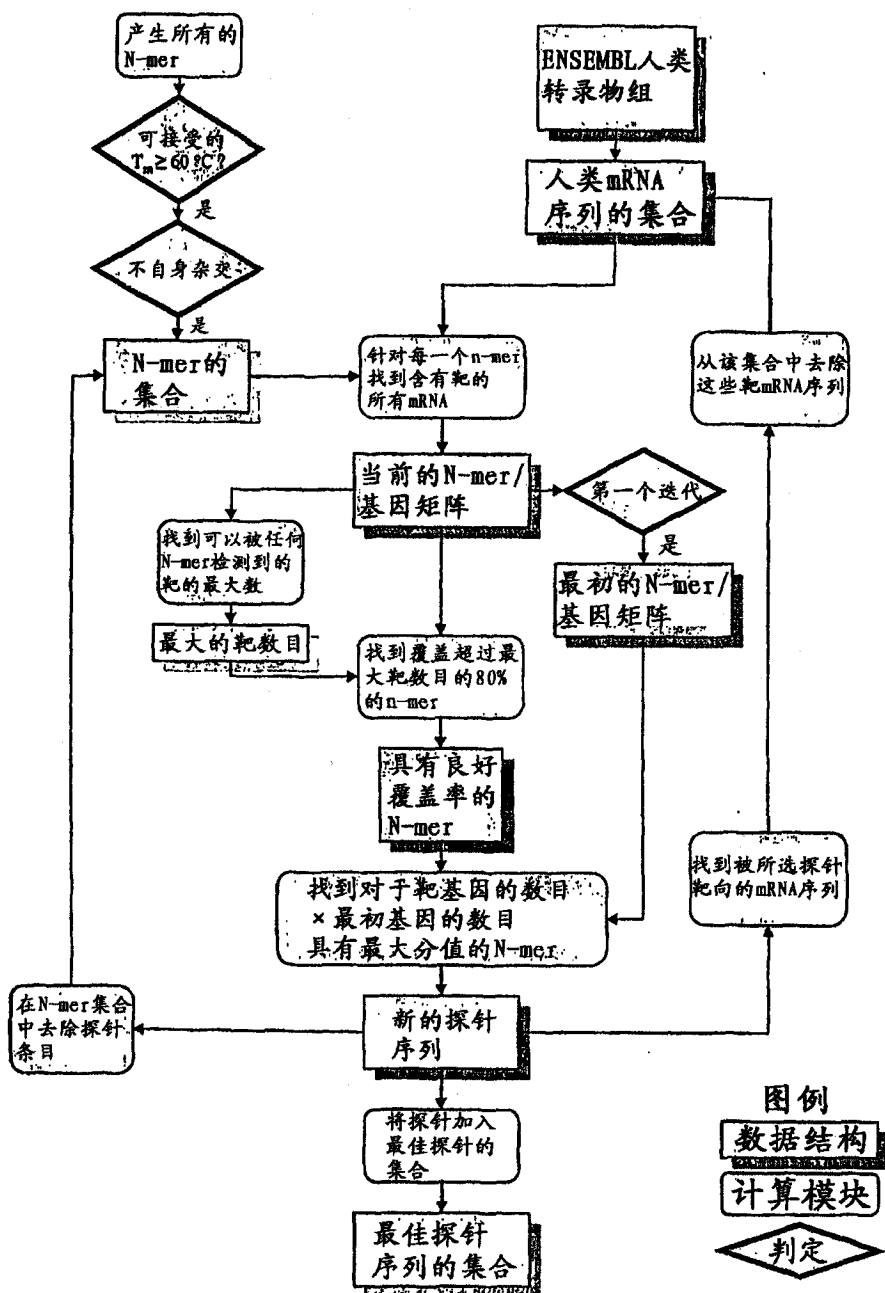


图 2

基因
数目

靶向各基因的探针的冗余性

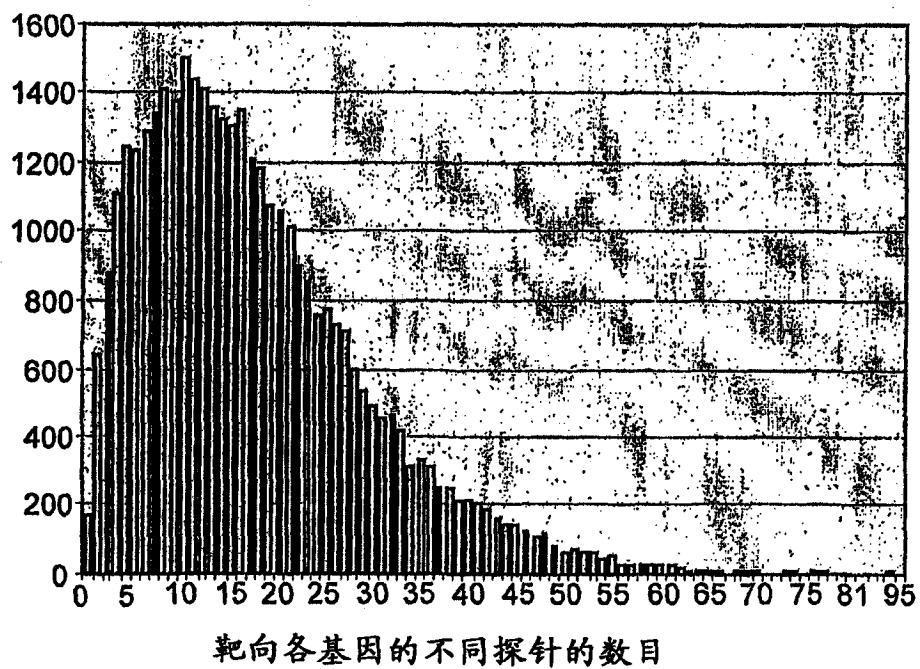


图 3

对人类mRNA转录物组的覆盖

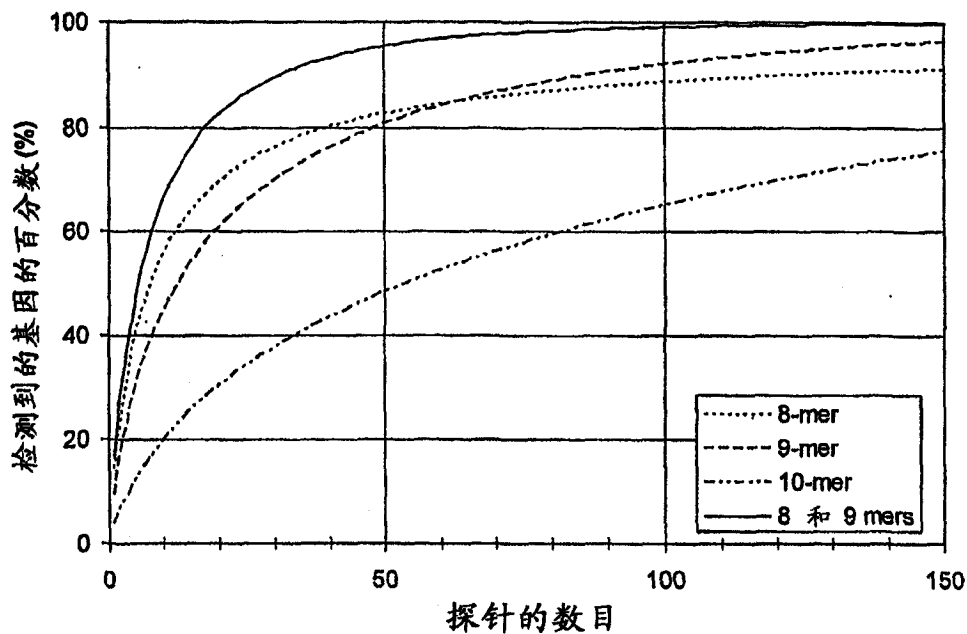


图 4

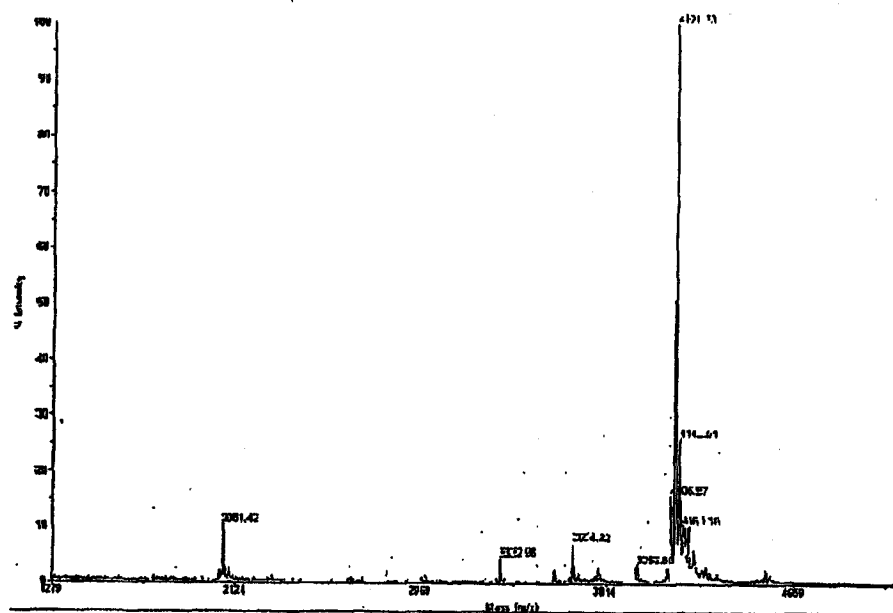


图5

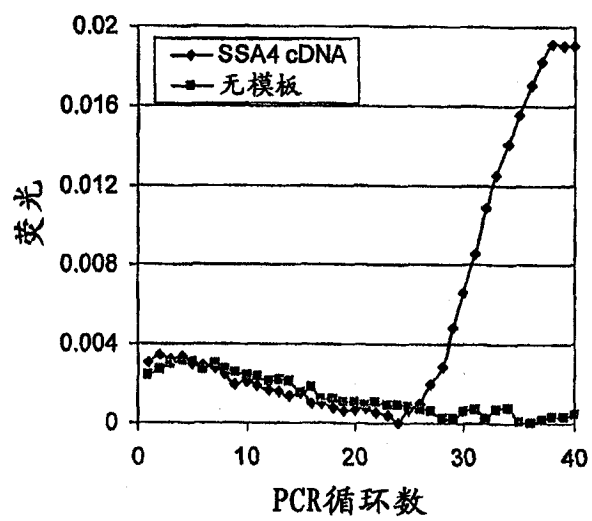


图8

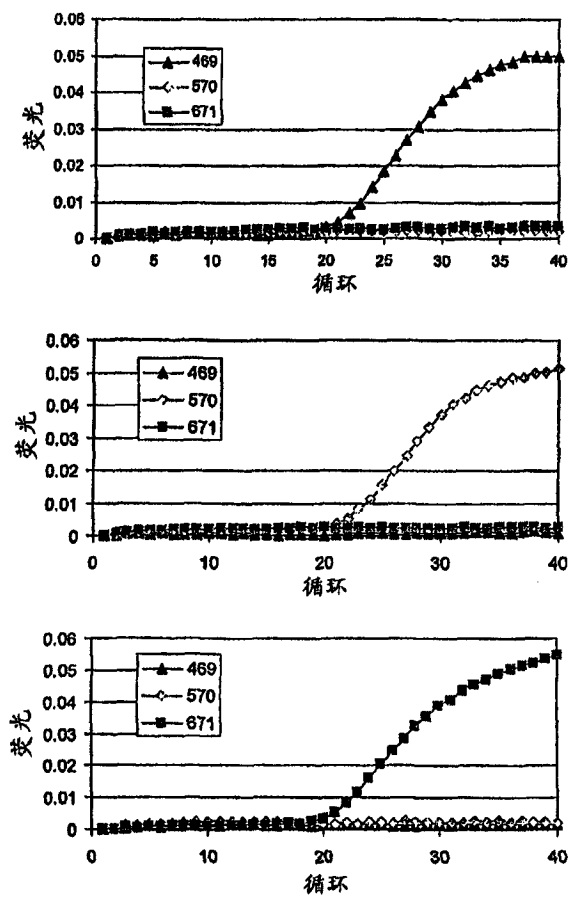


图6

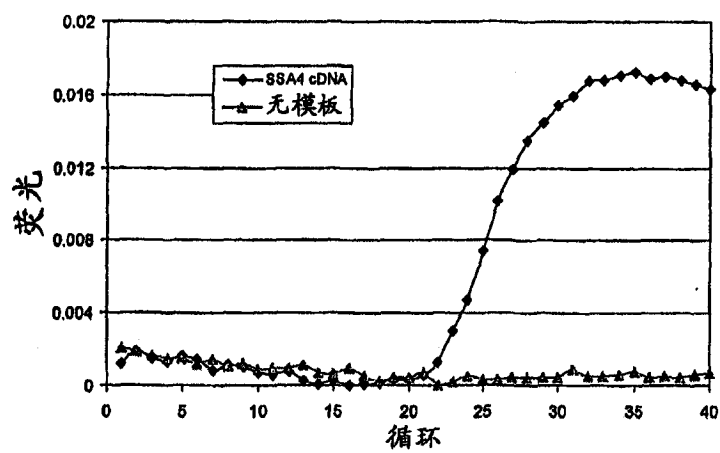


图 7A

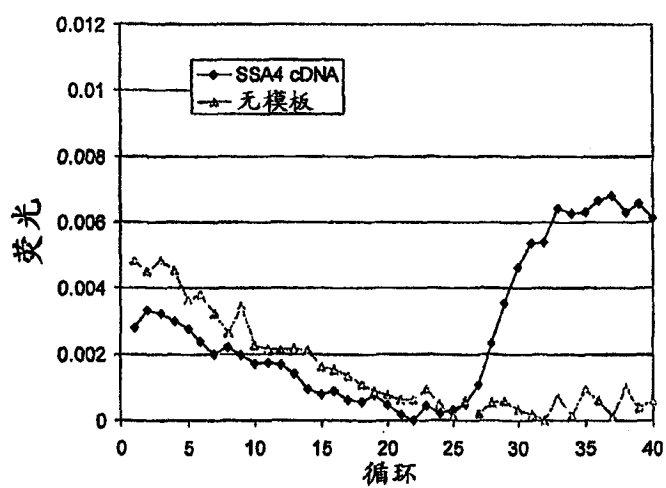


图 7B

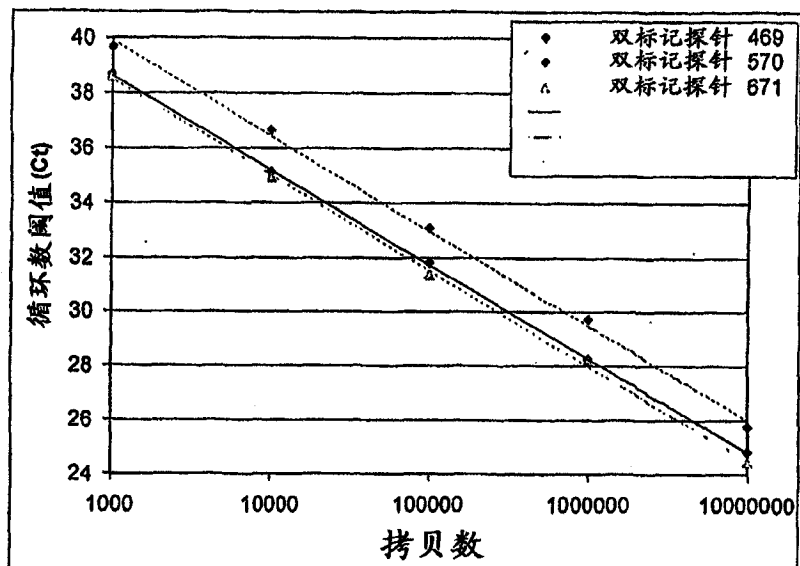


图9

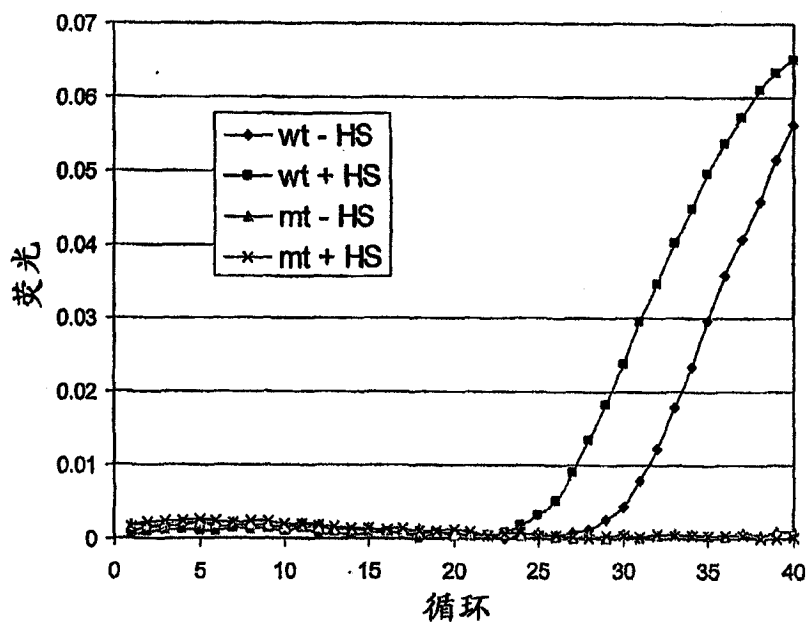


图10

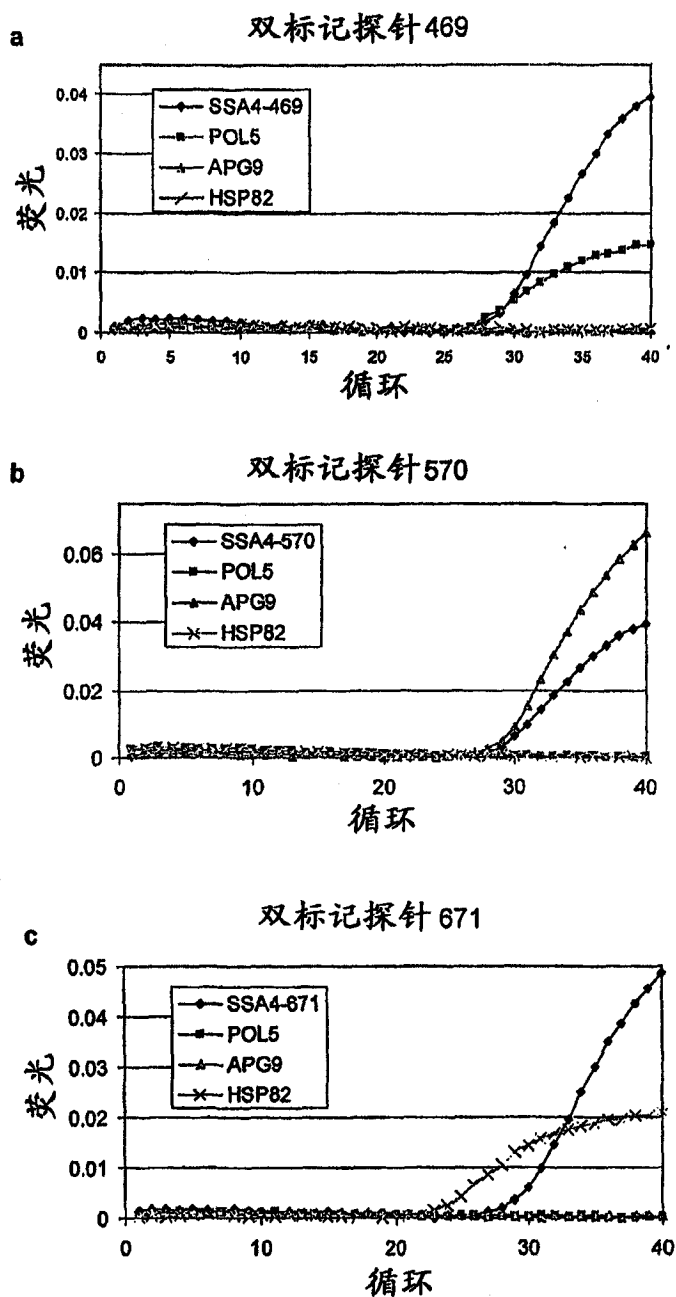


图 11

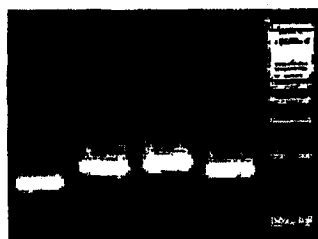


图 12

1	ctcctoct	Lib hum A 005	42	cttctccc	Lib hum B 127
2	ctggagga	Lib hum A 007	43	cctcagcc	Lib hum C 202
3	aggagctg	Lib hum B 101	44	ctccttcc	Lib hum C 246
4	cagcctgg	Lib hum A 008	45	cagcaggc	Lib hum C 203
5	cagcagcc	Lib hum A 009	46	ctgcctct	Lib hum C 204
6	tgtctggag	Lib hum A 010B	47	ctccacct	Lib hum C 205
7	agctggag	Lib hum A 019	48	ctcctccc	Lib hum C 206
8	ctgtgtcc	Lib hum A 020	49	cttcccca	Lib hum C 207
9	aggagcag	Lib hum A 011B	50	cttcagcc	Lib hum C 208
10	ccaggagg	Lib hum A 016	51	ctctgcca	Lib hum C 209
11	tcctgtctg	Lib hum B 102	52	ctgggaga	Lib hum C 247
12	cttctctcc	Lib hum A 015	53	cttctgtcc	Lib hum C 210
13	ccgccgcc	Lib hum A 004	54	cagcaggt	Lib hum C 211
14	cctggagc	Lib hum B 103	55	tctggagc	Lib hum C 214
15	cagcctcc	Lib hum A 017	56	tcctgtctc	Lib hum C 248
16	tggtgtgtg	Lib hum A 021	57	ctggggcc	Lib hum C 220
17	cctggaga	Lib hum A 022	58	ctcctgcc	Lib hum C 219
18	ccagccag	Lib hum B 105	59	ctgggcaa	Lib hum C 223
19	ccagggcc	Lib hum A 023	60	ctggggct	Lib hum C 226
20	cccagcag	Lib hum B 106	61	tggtggcc	Lib hum C 225
21	ccaccacc	Lib hum A 025	62	ccaggcca	Lib hum C 240
22	ctcctcca	Lib hum B 104	63	ctgtctcc	Lib hum C 227
23	ttctctctg	Lib hum B 109	64	tgggcagc	Lib hum C 239
24	cagcccag	Lib hum B 110	65	ctccatcc	Lib hum C 222
25	ctggctgc	Lib hum A 001	66	ctgccccca	Lib hum C 215
26	ctccacca	Lib hum B 112	67	ttctgtgc	Lib hum C 244
27	cttctctgc	Lib hum B 111	68	atggctgc	Lib hum C 217
28	cttccagc	Lib hum B 114	69	tggtggaa	Lib hum C 231
29	ccacctcc	Lib hum B 121	70	tgtgtgtcc	Lib hum C 228
30	ttctctctg	Lib hum B 122	71	ccagccgc	Lib hum C 224
31	cccagccc	Lib hum B 116	72	catccagc	Lib hum C 216
32	tggtgatg	Lib hum B 124	73	tcctctcc	Lib hum C 229
33	tggtctctg	Lib hum B 123	74	agctggga	Lib hum C 233
34	ctgccttc	Lib hum B 118	75	ctggtctc	Lib hum C 234
35	ctccagcc	Lib hum B 119	76	ttcccagt	Lib hum C 235
36	tgtggctg	Lib hum B 125	77	caggcagc	Lib hum C 250
37	cagaggag	Lib hum A 048B	78	tcctcagc	Lib hum C 245
38	cagctccc	Lib hum B 129	79	ctggctcc	Lib hum C 241
39	ctgcctcc	Lib hum B 128	80	tcctcttct	Lib hum C 236
40	ctctgtgc	Lib hum C 242	81	tcagtggt	Lib hum C 237
41	ctgcttcc	Lib hum C 201	82	acagcctca	GAPDH AQQ1
			83	cagccacc	Lib hum C 243

图 13

84	cttctggc	Lib_hum_D_301
85	tccactgc	Lib_hum_D_302
86	ctctgcgt	Lib_hum_D_303
87	ctggaggc	Lib_hum_D_304
88	ctggaagc	Lib_hum_D_305
89	ctgccacc	Lib_hum_D_306
90	tccaggtc	Lib_hum_D_307
91	cagcatcc	Lib_hum_D_308
92	cagaggct	Lib_hum_D_309
93	ctgaagcc	Lib_hum_D_310
94	tgcagggc	Lib_hum_D_311
95	atggcagc	Lib_hum_D_312
96	catectcc	Lib_hum_D_313
97	ctctgcct	Lib_hum_D_314
98	ccagtgcc	Lib_hum_D_315
99	cagtggca	Lib_hum_D_316
100	actgctgc	Lib_hum_D_317
101	cagcacc	Lib_hum_D_318
102	atgatggc	Lib_hum_D_319
103	ccaggagc	Lib_hum_D_320

图 14

容易地制备用于人类基因的qPCR

设计用于人类基因的有效可靠的qPCR试验是一项复杂的任务。

我们已经提供了一种新的网络工具ProbeFinder (见www.probelibrary.com) 用于快速并容易地选择ProbeLibrary™探针和设计用于人类基因的qPCR的引物。

该ProbeFinder网络服务器可以针对任何给定的人类基因快速且可靠地设计出最佳的qPCR探针和引物。对于具有特殊要求的基因，通过简单地使用网页，给出了备选的方案。

ProbeFinder分三步设计最佳qPCR

确定内含子位置

通过选择跨内含子的qPCR以消除来自染色体DNA*的噪音。利用blast检索人类基因组确定内含子。存在于DNA中但不存在于转录物中的区域被视为内含子。



*对于无染色体DNA的qPCR，可以消除对内含子的预测。

使ProbeLibrary™与基因匹配

几乎所有的人类转录物*都可以被这90个ProbeLibrary™探针中的至少一个覆盖。使用Exiqon的LNA™技术使得此高覆盖率成为可能。通过ProbeFinder™在数秒内可以鉴定到匹配的探针。

*来自RefSeq的人类转录物平均被17个ProbeLibrary™探针所覆盖，并每个探针平均在38556个转录物的超过7000个中发现。



设计引物和选择最佳的qPCR试验

用Primer3设计引物。最后，根据仔细选择的规则对试验进行分级，从而确保最佳的可能qPCR试验。所述规则有利于跨内含子的扩增子以除去来自DNA污染的信号、有利于用于可重复的相似试验的小的扩增子尺寸以及有利于优化用于PCR的GC含量。

图16

程序模块列表*Program main.aap*

CC = gcc -DLIB -O2

SOURCE =

```
5      getcover.c
      dyp.c
      getopt.c
      getopt1.c
      getopt_init.c
```

10

LIBS = -lm

TARGET = getcover\$EXESUF

图17

```
Program dyp.h
/*AUTHOR
    Copyright Niels Tolstrup 2003
    Exiqon
5  */

typedef struct sequences_t {
    int      nseq;
    int      maxnseq;
10    char    **seqs;
    } sequences_type;

typedef struct score_rec_t {
15    int      alph_len;
    int      gap_start;
    int      gap_cont;
    int      mismatch_cont;
    int      loop_score;
20    int      match_cont_factor;
    int      match_threshold;
    int      min_strong_ident_score;
    int      min_ident_score;
    int      min_sim_score;
25    char     *alph;
    int      *mat;
    } score_rec_type;

30 typedef struct param_t {
    char      *alph;
    int       *mathyp;
    int       depth;
    int       global;
35    int       min_score;
    int       max_res;
    sequences_type *seqs;
    sequences_type *seqs2;
    score_rec_type *score_rec;
40    } param_type;

typedef struct dynamic_str_t {
    int      len;
45    int      maxlen;
    char     *s;
    } dynamic_str_type;
```

图17 (续)

50

```
    param_type *init();  
55  char *empty_str( int len);  
    char *make_nmer( long number, int n_mer, char *alph);  
    dynamic_str_type *new_dynamic_str_type();  
60  dynamic_str_type *addchar( dynamic_str_type *seq, int  
    c);  
    long n_nmer( int n_mer, char *alph);  
65  long reverse_dna( long n, int oligolen);  
    long comp_dna( long n, int oligolen);  
70  long rev_comp_dna( long n, int oligolen);
```

Program getcover.c

```
/*  
75  #Id: $  
    SYNOPSIS  
        getcover -l 9 -p -f < h_sap.fasta > h_sap_19.stat  
        getcover -l 9 -i 1 -d 10 -t 60 -c -n -m -s <  
80  h_sap_19.stat > h_sap_19.cover  
  
    DESCRIPTION  
        Find a cover of n mers for human transcriptome  
85  Input: A fasta file  
        Options: getcover --help  
  
    INSTALATION  
        The program works udner solaris and linux.  
90  
  
    AUTHOR  
        Copyright Niels Tolstrup 2003  
        Exiqor  
95  
    */
```

图17 (续)

```

#include <stdio.h>      /* fprintf, rand..*/
#include <stdlib.h>     /* calloc..      */
100 #include <stdarg.h>  /* va_list..    */
#include <string.h>     /* strlen..     */
#include <limits.h>     /* USHRT_MAX..  */
#include <math.h>       /* pow          */
#include <ctype.h>      /* tolower      */
105 #include <time.h>    /* clock        */
#include "dyp.h"        /* make_nmer..  */
#include "getopt.h"     /* getopt_long   */

/***** GLOBAL VARIABLES *****/
110

#ifdef NODYP
int          verbose = 0;
char*        program_name;
115 #else
extern int verbose;
extern char* program_name;
#endif

120
int          t_start;
int          t_lap1;
int          t_lap2;
int          t_run;
125 int          t_debug;
int          t_last_debug;
int          t_debug_interval = 5;

/**** these should not be global ****/
130

#define TIME_INIT  0
#define LAP_TIME   1
#define TOTAL_TIME 2
135 #define DEBUG_TIME 4

/***** STRUCTURES *****/

140
typedef struct alph_t {
        char          *name;
        char          *alph;
        int           len;
145     } alph_type;

```

图17 (续)

```
typedef struct lna_t {
    long      lna_id;
150      char      self_score;
      char      target_score;
      char      tm;
    } lna_type;

155
typedef struct nmer_t {
    int      nmer_id;
    short     selected;
    char      ok; /*
160 ==0:undefined,<0:discard,>0:ok */
    lna_type  *lna;
    int      n_total_gene;
    int      n_gene;
    int      max_n_gene;
165      int      *gene;
    } nmer_type;

typedef struct stat_t {
170      int      n_nmer;
    int      oligolen;
    alph_type *alph;
    int      max_n_nmer;
    char      compact;
175      nmer_type **nmer;
    int      n_gene;
    int      max_n_gene;
    int      *gene;
    } stat_type;
180

typedef struct conf_t {
    int      fasta_in;
    int      stat_in;
185      int      oligolen;
    float     max_gene_hit_frac;
    int      max_gene_hit;
    int      target_minus_self;
    int      target_min_temp;
190      int      complement_flag;
    int      max_select;
    int      no_5end_spike;
    int      max_product;
    } conf_type;
```

图17 (续)

```
195
    #ifdef NODYP
    void die( char *fmt, ...){
        va_list ap;
200    fprintf( stderr, "Uhoh: ");
        va_start( ap, fmt);
        vfprintf( stderr, fmt, ap);
        va_end( ap);
        fprintf( stderr, "\n");
205    exit( 1);
    }
    #endif

210 void print_debug( char *fmt, ...){
    va_list ap;
    if( verbose >= 3) {
        va_start( ap, fmt);
        vfprintf( stderr, fmt, ap);
215    va_end( ap);
        fprintf( stderr, "\n");
        fflush( stderr);
    }
}

220

void print_debug_interval( char *fmt, ...){
    va_list ap;
    if( (verbose >= 3) && run_time( DEBUG_TIME)) {
225    fprintf( stderr, "%8d ", t_run);
        va_start( ap, fmt);
        vfprintf( stderr, fmt, ap);
        va_end( ap);
        fprintf( stderr, "\n");
230    fflush( stderr);
    }
}

235 void usage( char *fmt, ...){
    va_list ap;
    va_start( ap, fmt);
    vfprintf( stderr, fmt, ap);
    va_end( ap);
240    fprintf( stderr,
        "\nUsage: getcover -h,--help\n"
        "    -v,--verbose\n"
        "    -vv,--verbose --verbose\n"
```

图17 (续)

```

245      "    -l,--oligo_len len\n"
      "    -i,--max_gene_hit_frac fraction\n"
      "    -d,--target_minus_self delta\n"
      "    -t,--target_min_temp temp\n"
      "    -c,--complement\n"
      "    -m,--max_product\n"
250      "    -n,--no_5end_spike\n"
      "    -f,--fasta_in\n"
      "    -s,--stat_in\n"
      "    -p,--dump_stat\n"
      "\n"
255      "DESCRIPTION\n"
      "    ");
    fprintf( stderr, "\n");
    exit( 1);
}
260

int run_time( int type){
    if( type == DEBUG_TIME) {
        t_debug = clock()/CLOCKS_PER_SEC;
265        if( t_debug - t_last_debug >= t_debug_interval ) {
            t_run = t_debug - t_start;
            t_last_debug = t_debug;
        }
        else
270            t_run = 0;
    }
    else if( type == TIME_INIT) {
        t_start      = clock()/CLOCKS_PER_SEC;
        t_lap1       = t_start;
275        t_last_debug = t_start;
        t_run        = 0;
    }
    else if( type == LAP_TIME) {
        t_lap2 = clock()/CLOCKS_PER_SEC;
280        t_run  = t_lap2 - t_lap1;
        t_lap1 = t_lap2;
    }
    else if( type == TOTAL_TIME) {
        t_lap1 = clock()/CLOCKS_PER_SEC;
285        t_run  = t_lap1 - t_start;
    }
    return t_run;
}

290
#ifdef NODYP
void *salloc( int nobj, int size){

```

图17 (续)

```

        void *mem;
        /*mem =      calloc( nobj, size);*/
295    mem = malloc( nobj * size);
        /*printf( "calloc: allocating %d objects of size %d
        bytes\n", nobj, size);*/
        if( mem == NULL){
            die( "Could not allocate %d x %d bytes\n", nobj,
300    size);
        }
        return mem;
    }
    #endif
305

    void free_nmer_type( nmer_type *nmer) {
        if( nmer != NULL ) {
            free( nmer->gene);
310        free( nmer->lna);
        }
        free( nmer);
    }

315
    void free_stat( stat_type *stat) {
        int      i;

        for( i=0; i<stat->n_nmer; i++)
320        free_nmer_type( stat->nmer[i]);

        free_alph_type( stat->alph);
        free( stat->nmer);
        free( stat);
325    }

    /* set, union, intersection, complement would be nice
    to have */
330
        /* the lists must be sorted for this algorithm! */

    void set_union( int *list1, int n_list1, int *list2,
    int n_list2,
335        int **reslist, int *n_reslist){
        int i, j, k, n;
        int n1, n2;
        int *tmplist;

340    tmplist      = (int *) calloc( n_list1 + n_list2,
        sizeof( int));

```

图17 (续)

```
j=0;
k=0;
345 n=0;
while( j<n_list1 && k<n_list2 ) {
    n1 = list1[j];
    n2 = list2[k];
    if( n1 < n2 ) {
350     tmp1ist[ n] = n1;
        n++;
        j++;
    }
    else if( n2 < n1 ) {
355     tmp1ist[ n] = n2;
        n++;
        k++;
    }
    else {
360     tmp1ist[ n] = n1;
        n++;
        j++;
        k++;
    }
365 }

for( ; j<n_list1; j++) {
    n1 = list1[j];
    tmp1ist[ n] = n1;
370     n++;
}

for( ; k<n_list2; k++) {
    n2 = list2[k];
375     tmp1ist[ n] = n2;
    n++;
}

*n_reslist = n;
380 *reslist = (int *) calloc( *n_reslist, sizeof(
int));

for( k=0; k<n; k++) {
    (*reslist)[k] = tmp1ist[ k];
385 }
free( tmp1ist);
}

390 stat_type *compact_stat( stat_type *stat){
```

图17 (续)

```

    int i, j;

    for( i=0; i<stat->max_n_nmer && stat->nmer[i] != NULL
        && stat->nmer[i]->n_gene > 0; i++)
395      ;

    j=i;
    for( ; i<stat->max_n_nmer; i++) {
        if(stat->nmer[i] != NULL) {
400          if( stat->nmer[i]->n_gene > 0) {
              stat->nmer[j] = stat->nmer[i];
              stat->nmer[i] = NULL;

              j++;
405          }
          else {
              free_nmer_type( stat->nmer[i]);
              stat->nmer[i] = NULL;
          }
410      }
    }
    stat->n_nmer = j;
    stat->compact = 1;
}
415

stat_type *comp_stat( stat_type *stat){
    int      i, j, comp_i, comp_j;
    char      *seq, *compseq;
420    char      alph[10] = "acgt";
    int      mask = (n_nmer( stat->oligolen, alph) - 1);
    int      *tmpgene;
    int      ngene;
    int      comp_nmer_id;
425    nmer_type *the_nmer, *the_comp_nmer;

    if( stat->compact )
        die("Internal error, comp_stat does not work for com-
430 pact stat");

    for( i=0; i<stat->max_n_nmer; i++) {
        the_nmer = stat->nmer[i];
435        if( the_nmer != NULL ) {
            comp_nmer_id = rev_comp_dna( the_nmer->nmer_id,
stat->oligolen);
            the_comp_nmer = NULL;

```

图17 (续)

```

440     comp_j = comp_nmer_id;
        the_comp_nmer = stat->nmer[ comp_j];
        /*
        for( j=0; j<stat->n_nmer && the_comp_nmer ==
NULL; j++) {
445         if( stat->nmer[j] != NULL &&
            stat->nmer[j]->nmer_id == comp_nmer_id) {
            comp_j = j;
            the_comp_nmer = stat->nmer[ comp_j];
        }
450     }
        */

        if( the_comp_nmer != NULL && comp_nmer_id !=
the_nmer->nmer_id) {
455         set_union( the_nmer->gene, the_nmer->n_gene,
            the_comp_nmer->gene, the_comp_nmer-
>n_gene,
                &tmpgene, &ngene);
        if( the_nmer->nmer_id < the_comp_nmer->nmer_id
460     ) {
            the_nmer->n_gene = ngene;
            the_nmer->n_total_gene = ngene;
            free( the_nmer->gene);
            the_nmer->gene = tmpgene;
465         free_nmer_type( the_comp_nmer);
            stat->nmer[ comp_j] = NULL;
        }
        else if( the_nmer->nmer_id > the_comp_nmer-
>nmer_id ) {
470         the_comp_nmer->n_gene = ngene;
            the_comp_nmer->n_total_gene = ngene;
            free( the_comp_nmer->gene);
            the_comp_nmer->gene = tmpgene;
            free_nmer_type( the_nmer);
475         stat->nmer[i] = NULL;
        }
        else
            die("nmer_id %d found twice\n", the_nmer-
>nmer_id);
480     }
    }
    compact_stat( stat);
485 }

stat_type *init_stat_type( int *membyte, conf_type
*conf,

```

图17 (续)

```
                                alph_type *alph){
490     stat_type *stat;
        int i;

        stat = (stat_type *) calloc( 1, sizeof( stat_type));
495     *membyte      += sizeof( stat_type);
        stat->compact = 0;

        stat->oligolen = conf->oligolen;
        stat->alph      = alph;
500     stat->max_n_nmer = n_nmer( stat->oligolen, "acgt");
        stat->n_nmer    = 0;
        stat->nmer      =
            (nmer_type**) calloc( stat->max_n_nmer,
sizeof( nmer_type*));
505     membyte      += sizeof( nmer_type*) * stat-
>max_n_nmer;
        for( i=0;i<stat->max_n_nmer;i++) {
            stat->nmer[i] = NULL;
        }
510     stat->max_n_gene = 0;
        stat->n_gene    = 0;
        stat->gene      = NULL;

515     return stat;
    }

nmer_type *new_nmer_type(){
520     nmer_type *the_nmer;

        the_nmer = (nmer_type*) calloc( 1,
sizeof( nmer_type));
        the_nmer->nmer_id = -1;
525     the_nmer->selected = 0;
        the_nmer->ok      = 0;
        the_nmer->lna      = NULL;
        the_nmer->n_total_gene = 0;
        the_nmer->n_gene    = 0;
530     the_nmer->max_n_gene = 0;
        the_nmer->gene      = NULL;

        return the_nmer;
    }
535
```

图17 (续)

```

stat_type *read_stat( FILE *statfile, conf_type *conf,
alph_type *alph){
    char      s[256];
540    int      nmer_id    = -1;
        int      gene      = -1;
        stat_type *stat;
        nmer_type *the_nmer;
        int      i;
545    int      membyte     = 0;
        int      max_n_gene = 128;
        int      *tmpgene  = (int *)salloc( max_n_gene,
sizeof( int));
        int      n_gene     = 0;
550    int      maxgene     = 0;

        stat = init_stat_type( &membyte, conf, alph);

        while( fgets(s, 254, statfile)) {
555            if( s[0] == '#') {
                }
            else if( s[0] == '>') {
                if( n_gene > 0 ) {
                    the_nmer->max_n_gene    = n_gene;
560                    the_nmer->n_total_gene = n_gene;
                    the_nmer->gene = (int *)salloc( n_gene, sizeof(
int));
                    membyte    += sizeof( int) * n_gene;
                    for( i=0; i<n_gene; i++) {
565                        the_nmer->gene[ the_nmer->n_gene] =
tmpgene[i];
                        the_nmer->n_gene++;
                    }
                    n_gene = 0;
570                }

                print_debug_interval( "Read %d nmers.", stat-
>n_nmer);

575                sscanf( s, ">%d", &nmer_id);
                if( stat->n_nmer >= stat->max_n_nmer) {
                    die( "Unexpected high nmer number (stat-
>n_nmer=%d) stat->max_n_nmer=%d, is the oligo length
set correctly?\n",
580                    stat->n_nmer, stat->max_n_nmer);
                }
                if( nmer_id >= stat->max_n_nmer) {
                    die( "nmer_id (%d) over limit %d in %s, is the
oligo length set correctly?\n",
585                    nmer_id, stat->max_n_nmer, s);
                }
            }
        }
    }
}

```

图17 (续)

```

    }
    if( nmer_id < 0 ) {
        die( "Negative nmer_id (%d)\n", nmer_id);
    }
590   stat->nmer[ nmer_id] = new_nmer_type();
    membyte      += sizeof( nmer_type);
    the_nmer = stat->nmer[ nmer_id];
    the_nmer->nmer_id      = nmer_id;
    stat->n_nmer++;
595   }
    else {
        sscanf( s, "%d", &gene);
        if( n_gene >= max_n_gene) {
            max_n_gene *= 2;
600     if( ! (tmpgene = realloc( tmpgene, (max_n_gene
+ 1) * sizeof( int))) )
            die( "Failed to realloc tmpgene to %d
ints\n", max_n_gene + 1);
        }
605     if( gene > maxgene)
        maxgene = gene;
        tmpgene[ n_gene] = gene;
        n_gene++;
    }
610   }

    if( n_gene > 0 ) {
        the_nmer->max_n_gene = n_gene;
        the_nmer->gene = (int *)salloc(n_gene, sizeof(
615 int));
        membyte      += sizeof( int) * n_gene;
        for( i=0; i<n_gene; i++) {
            the_nmer->gene[ the_nmer->n_gene] =
tmpgene[i];
620         the_nmer->n_gene++;
        }
        n_gene = 0;
    }

625   stat->max_n_gene = maxgene + 1;
    stat->gene      = (int *)salloc( stat->max_n_gene,
sizeof( int));
    membyte      += sizeof( int) * stat->max_n_gene;
    for( i=0; i<stat->max_n_gene ; i++) {
630     stat->gene[i] = 0;
    }

    printf( "# Allocated %d Mbytes for stat\n", mem-
byte/1048576);

```

图17 (续)

```
635     return stat;
    }

    int numcmp( const void *x, const void *y) {
640     if ( *(int *)x == *(int *)y ) return ( 0);
        else return ( *(int *)x >
*(int *)y ) ? 1 : -1;
    }

645 #ifdef NODYP
char *empty_str( int len){
    char *str;
    str = calloc( len+1, sizeof( char));
    memset( str, ' ', len);
650     str[len] = 0;
    return str;
}

655 char *reverse( char *str){
    int i, len;
    char c;

    len = strlen( str);
660     for( i=0; i<len/2; i++){
        c = str[i];
        str[i] = str[(len-1) - i];
        str[(len-1) - i] = c;
    }
665     return str;
}

char *make_nmer( long number, int n_mer, char *alph) {
670     long i, j, n = number;
    int alphlen = strlen( alph);
    char *seq;

    seq = empty_str( n_mer);
675     if( alphlen == 0 )
        strcpy( seq, "-");
    else {
        for( j=0; j<n_mer; j++){
680             i = n % alphlen;
            n = n / alphlen;
            seq[j] = alph[i];
        }
    }
}
```

图17 (续)

```
    }
685    seq = reverse( seq);

    return seq;
}
#endif
690 char *num2nmer( long number, int oligolen, alph_type
*alph) {
    return make_nmer( number, oligolen, alph->alph);
}
695

void sort_gene( stat_type *stat) {
    int i, j;
    nmer_type *the_nmer;
700    for( i=0; i<stat->max_n_nmer; i++) {
        if( stat->nmer[i] != NULL ) {
            the_nmer = stat->nmer[i];
            qsort( the_nmer->gene, the_nmer->n_gene, sizeof(
705 int), numcmp);
        }
    }
}

710 void print_n_gene( stat_type *stat) {
    int i, j;
    nmer_type *the_nmer;

715    for( i=0; i<stat->n_nmer; i++) {
        printf( ">%d %d %d\n", stat->nmer[i]->nmer_id,
stat->nmer[i]->n_gene, stat->nmer[i]->selected);
    }
}
720

void remove_gene( stat_type *stat, int nmer_id, short
select_no) {
    int i, j, k, n;
725    nmer_type *the_nmer;
    int max_n_gene;
    int *r_genes;
    int n_r_genes;
    int rgene;
730    int gene;
    int *tmpgene;
```

图17 (续)

```

    tmpgene      = (int *) calloc( stat->max_n_gene,
735    sizeof( int));

    r_genes      = stat->nmer[nmer_id]->gene;
    n_r_genes    = stat->nmer[nmer_id]->n_gene;
    stat->nmer[nmer_id]->selected = select_no;
    stat->nmer[nmer_id]->ok      = -3;

740    for( i=0; i<n_r_genes; i++) {
        stat->gene[ r_genes[i]]++;
    }

745    stat->n_gene = 0;
    for( i=0; i<stat->max_n_gene; i++) {
        if( stat->gene[ i]) {
            stat->n_gene++;
        }
750    }

    for( i=0; i<stat->n_nmer; i++) {
        the_nmer = stat->nmer[i];
        if( the_nmer->ok >= 0) {
755            j=0;
            k=0;
            n=0;
            while( j<n_r_genes && k<the_nmer->n_gene ) {
                rgene = r_genes[j];
760                gene = the_nmer->gene[k];
                if( rgene > gene ) {
                    k++;
                    tmpgene[ n] = gene;
                    n++;
765                }
                else if( rgene < gene ) {
                    j++;
                }
                else {
770                    j++;
                    k++;
                }
            }

775            for( ; k<the_nmer->n_gene; k++) {
                gene = the_nmer->gene[k];
                tmpgene[ n] = gene;
                n++;
            }

780            the_nmer->n_gene = n;

```

图17 (续)

```

        for( k=0; k<n; k++) {
            the_nmer->gene[k] = tmpgene[ k];
        }
785     }
        free( tmpgene);
    }

790 char nmer_ok( nmer_type *the_nmer, conf_type *conf, int
oligolen) {

    if( the_nmer->n_total_gene  > conf->max_gene_hit) {
795         return -2;
    }

    int self, target;
    long nmr;
800    long j;
    char *seq, *rev_seq;
    param_type *param;
    lna_type *best_lna;
    int max_delta = -99999;
805    long max_lna_id = -1;
    int max_self_s = -9999;
    int max_target_s = -9999;
    int max_t_delta = -99999;
    long max_t_lna_id = -1;
810    int max_t_self_s = -9999;
    int max_t_target_s = -9999;
    int delta;
    char ok;
    long nmer;
815    int n_spike_pat = pow( 2, oligolen);

    best_lna = (lna_type *) calloc( 1, sizeof(
lna_type));
820    best_lna->lna_id = -1;
    best_lna->self_score = -1;
    best_lna->target_score = -1;
    best_lna->tm = -1;

825    param = init();
    param->min_score = 2;
    param->max_res = 1;

    rev_seq = empty_str( oligolen);
830

```

图17 (续)

```

    if( conf->no_5end_spike ) {
        n_spike_pat /= 2;
    }
835   for( j=0; j<n_spike_pat; j++) {
        nmer = dna2lna_spike( the_nmer->nmer_id, j);
        seq  = make_nmer( nmer, oligolen, "acgtACGT");
        calc_olig( nmer, seq, rev_seq, &self, &target, pa-
840   ram, 0);
        free( seq);
        delta = target - self;
        if( delta > max_delta){
            max_delta      = delta;
845   max_lna_id      = nmer;
            max_self_s     = self;
            max_target_s   = target;
        }
        if( delta > max_t_delta && target > conf-
850   >target_min_temp){
            max_t_delta    = delta;
            max_t_lna_id   = nmer;
            max_t_self_s   = self;
            max_t_target_s = target;
855   }
    }
    free_param_type( param);
    if( max_t_delta > 0) {
        best_lna->lna_id    = max_t_lna_id;
860   best_lna->self_score   = max_t_self_s;
        best_lna->target_score = max_t_target_s;
        if( max_t_delta >= conf->target_minus_self )
            ok              = 1;
        else
865   ok                    = -1;
    }
    else {
        best_lna->lna_id    = max_lna_id;
        best_lna->self_score = max_self_s;
870   best_lna->target_score = max_target_s;
        ok                  = -1;
    }

    the_nmer->lna = best_lna;
875   the_nmer->ok  = ok;

    return ok;
}

```

图17 (续)

880

```

int max_stat( stat_type *stat, conf_type *conf) {
    int i;
885    int max_n_gene      = 0;
    int max_n_gene_prod = 0;
    int max_nmer_id     = -1;

    if( conf->max_product) {
890        for( i=0; i<stat->n_nmer; i++) {
            if( stat->nmer[i]->ok          >= 0 &&
                stat->nmer[i]->n_gene > max_n_gene ) {
                if( stat->nmer[i]->ok == 0) {
                    nmer_ok( stat->nmer[i], conf, stat-
895 >oligolen);
                }
                if( stat->nmer[i]->ok > 0) {
                    max_n_gene  = stat->nmer[i]->n_gene;
                    max_nmer_id = i;
900                }
            }
        }
        max_n_gene = (int) (0.80 * max_n_gene);
        for( i=0; i<stat->n_nmer; i++) {
905            if( stat->nmer[i]->ok          >= 0 &&
                stat->nmer[i]->n_gene      > max_n_gene
&&
                stat->nmer[i]->n_gene * stat->nmer[i]-
>n_total_gene >
910 max_n_gene_prod ) {
                if( stat->nmer[i]->ok == 0) {
                    nmer_ok( stat->nmer[i], conf, stat->oligolen);
                }
                if( stat->nmer[i]->ok > 0) {
915                    max_n_gene_prod =
                        stat->nmer[i]->n_gene * stat->nmer[i]-
>n_total_gene;
                    max_nmer_id = i;
920                }
            }
        }
    }
    else {
925        for( i=0; i<stat->n_nmer; i++) {
            if( stat->nmer[i]->ok          >= 0 &&
                stat->nmer[i]->n_gene      > max_n_gene ) {
                if( stat->nmer[i]->ok == 0) {

```

图17 (续)

```

        nmer_ok( stat->nmer[i], conf, stat->oligolen);
930     }
        if( stat->nmer[i]->ok > 0) {
            max_n_gene = stat->nmer[i]->n_gene;
            max_nmer_id = i;
        }
935     }
    }
    /*
    if( max_nmer_id == -1) {
940     printf(" Found no nmers with \n");
        printf(" n_total_gene <= %d\n", conf-
>max_gene_hit);
        printf(" n_gene > %d\n", max_n_gene);
        printf(" selected == 0\n");
945     }
    /*
    return max_nmer_id;
}

950
int get_cover1( stat_type *stat, conf_type *conf){
    int i, j;
    int max_id;
    char *seq, *lnaseq;
955     int select_no = 0;
    char alph[10] = "acgt";

    printf("#dnaID nmer      cover    sum tm self lnaID
oligo\n");
960     max_id = max_stat( stat, conf);
    if(0)hi("stat->nmer[ max_id]->n_gene=%d\n", stat->nmer[
max_id]->n_gene);
    for( i=0; i < conf->max_select &&
        max_id >= 0 &&
965     stat->nmer[ max_id]->n_gene > 0; i++) {
        print_debug_interval( "Found %d n_mers.", i);
        if(0)print_n_gene( stat);
        select_no++;
        remove_gene( stat, max_id, select_no);
970     seq = make_nmer( stat->nmer[ max_id]->nmer_id,
conf->oligolen, alph);
        lnaseq = make_nmer( stat->nmer[ max_id]->lna-
>lna_id, conf->oligolen,
975     "acgtACGT");
        printf("%6d %s %4d %5d %2d %2d %9d %s\n",
            stat->nmer[ max_id]->nmer_id,

```

图17 (续)

```

        seq,
        stat->nmer[ max_id]->n_gene,
980    stat->n_gene,
        stat->nmer[ max_id]->lna->target_score,
        stat->nmer[ max_id]->lna->self_score,
        stat->nmer[ max_id]->lna->lna_id,
        lnaseq);
985    free( seq);
        free( lnaseq);
        if(0) {
            for(i=0;i<57;i++){ printf("%d", stat->gene[i]); }
            printf("\n");
990    for(i=0;i<stat->nmer[max_id]->n_gene;i++){
                printf("%d ", stat->nmer[max_id]->gene[i]);
            }
            printf("\n");
        }
995    max_id = max_stat( stat, conf);
    }

    if(0) {
1000    for(i=0;i<stat->n_nmer;i++){
        if( stat->nmer[i]->n_gene > 1 || stat->nmer[i]-
>selected != 0) {
            printf("+ %d %d %d %d :", i, stat->nmer[i]-
>nmer_id, stat->nmer[i]->n_gene, stat->nmer[i]-
1005 >selected);
            for(j=0;j<stat->nmer[i]->n_gene;j++){
                printf("%d ", stat->nmer[i]->gene[j]);
            }
            printf("\n");
1010    }
        }
    }

    return select_no;
1015 }

int nmer2num( char *seq, int len, alph_type *alph){
    int num = 0;
1020    int i;
    char nuc;
    int n;
    int mul = 1;

1025    for( i=0; i<len; i++){
        nuc = seq[len - i - 1];

```

图17 (续)

```

        n = strchr( alph->alph, nuc) - alph->alph;
        if( n < 0) {
            return num;
1030     }
        num += n * mul;
        mul *= alph->len;
    }

1035     return num;
}

void add_gene( nmer_type *the_nmer, int gene_num){
1040     int *genptr = NULL;

    if( the_nmer->n_gene > 0 ) {
        genptr = bsearch( &gene_num, the_nmer->gene,
                           the_nmer->n_gene, sizeof( the_nmer-
1045 >gene[0]),
                           numcmp);
    }

    if( genptr == NULL ) {
1050         the_nmer->n_gene++;
        if( the_nmer->n_gene > the_nmer->max_n_gene) {
            if( the_nmer->max_n_gene == 0) {
                the_nmer->max_n_gene = 2;
                the_nmer->gene = (int *)salloc( the_nmer-
1055 >max_n_gene + 1,
                                                sizeof(
the_nmer->gene[0]));
            }
            else {
1060                 the_nmer->max_n_gene *= 2;
                if( ! (the_nmer->gene = realloc( the_nmer-
>gene,
                                                (the_nmer->max_n_gene + 1)* sizeof(
the_nmer->gene[0]))) )
1065                 die( "Failed to realloc gene to %d ints\n",
the_nmer->max_n_gene);
            }
        }
        the_nmer->gene[ the_nmer->n_gene-1] = gene_num;
1070         /* Not necessary if gene_num is ascending
        qsort( the_nmer->gene, the_nmer->n_gene,
                sizeof( the_nmer->gene[0]), numcmp);
        */
    }
1075 }

```

图17 (续)

```

void add_seq2stat( char *seq, int seqlen,
                  stat_type *stat, alph_type *alph, int
1080 gene_num) {
    int i;
    char *s, *subseq;
    int num;
    int end = seqlen - stat->oligolen;
1085 char stmp[255];

    for( i=0; i<=end; i++) {
        subseq = seq + i;
        num = nmer2num( subseq, stat->oligolen, alph);
1090 strncpy( stmp, subseq, stat->oligolen);
        stmp[stat->oligolen] = 0;
        /*printf( "%s %d\n", stmp, num);*/

        if( num >= stat->max_n_nmer) {
1095 die( "%s (%d) higher than max %d\n", num, stmp,
        stat->max_n_nmer);
        }
        add_gene( stat->nmer[ num], gene_num);
        /* $stat->[ $num]{ $seq->{ name}}++; */
1100 }
    }

stat_type *read_fasta4stat( FILE *seqfile, conf_type
1105 *conf, alph_type *dna){
    int c;
    int i;
    char name[255];
    char comment[255];
1110 char s[255];
    int seq_flag = 0;
    dynamic_str_type *seq = NULL;
    stat_type *stat;
    int membyte = 0;
1115 int gene_num = -1;
    int n_in;

    stat = init_stat_type( &membyte, conf, dna);
    for( i=0; i<stat->max_n_nmer ;i++) {
1120 stat->nmer[i] = new_nmer_type();
        membyte += sizeof( nmer_type);
        stat->nmer[i]->nmer_id = i;
    }
    stat->n_nmer = stat->max_n_nmer;

```

图17 (续)

```

1125     while( (c = fgetc( seqfile)) != EOF){
        if( c == '>' ) {
            if( seq != NULL && seq->len > 0 ) {
                /*printf(">%s %s %d\n", name, comment,
1130 gene_num);*/
                add_seq2stat( seq->s, seq->len, stat, dna,
gene_num);
                free( seq);
            }
            seq = new_dynamic_str_type();
            fgets( s, 254, seqfile);
            n_in = sscanf( s, "%s %s", name, comment);
            if( n_in < 2)
                comment[0]=0;
1140         if( n_in < 1)
                name[0]=0;
                gene_num++;
                seq_flag = 1;
                print_debug_interval( "Read %d genes.",
1145 gene_num);
            }
            else if( seq_flag == 1 ) {
                c = tolower( c);
                if( strchr( dna->alph, c) != NULL ) {
1150                 seq = addchar( seq, c);
                }
            }
        }
        if( seq != NULL && seq->len > 0 )
1155         add_seq2stat( seq->s, seq->len, stat, dna,
gene_num);
        free( seq);

        for( i=0; i<stat->max_n_nmer ;i++) {
1160             membyte             += sizeof( int) * stat->nmer[i]-
>max_n_gene;
        }

        stat->max_n_gene = gene_num + 1;
1165         stat->gene             = (int *)salloc( stat->max_n_gene,
sizeof( int));
        membyte             += sizeof( int) * stat->max_n_gene;
        for( i=0; i<stat->max_n_gene ; i++) {
            stat->gene[i] = 0;
1170     }

    printf( "# Allocated %d Mbytes for stat\n", memby-
te/1048576);

```

图17 (续)

```

        return stat;
1175    }

    void print_stat( stat_type *stat) {
        int      i, j;
1180    nmer_type  *the_nmer;
        char      *s;
        int      str_flag = 0;

        if( verbose > 1 ) {
1185    str_flag = 1;
        }

        if( stat->compact ) {
1190    for( i=0; i<stat->n_nmer; i++) {
            print_debug_interval( "Wrote %d n_mers.", i);
            if( str_flag) {
                s = num2nmer( stat->nmer[i]->nmer_id, stat-
>oligolen, stat->alph);
1195    printf( ">%d %s\n", stat->nmer[i]->nmer_id,
s);
                free( s);
            }
            else {
1200    printf( ">%d\n", stat->nmer[i]->nmer_id);
            }
            the_nmer = stat->nmer[i];
            for( j=0; j<the_nmer->n_gene; j++) {
                printf( "%d\n", the_nmer->gene[j]);
1205    }
            }
        }
        else {
            for( i=0; i<stat->max_n_nmer; i++) {
1210    print_debug_interval( "Wrote %d n_mers.", i);
            if( stat->nmer[i] != NULL ) {
                if( str_flag) {
                    s = num2nmer( stat->nmer[i]->nmer_id, stat-
>oligolen, stat->alph);
1215    printf( ">%d %s\n", stat->nmer[i]->nmer_id,
s,i);
                    free( s);
                }
                else {
1220    printf( ">%d\n", stat->nmer[i]->nmer_id);
                }
                the_nmer = stat->nmer[i];

```

图17 (续)

```

        for( j=0; j<the_nmer->n_gene; j++) {
            printf( "%d\n", the_nmer->gene[j]);
1225     }
        }
    }
}
1230

    alph_type *new_alph( char *name, char *alph){
        alph_type *alpht;

1235     alpht = (alph_type*) calloc( 1, sizeof( alph_type));
        alpht->len = strlen( alph);
        alpht->name = ( char *) calloc( strlen( name) + 1,
        sizeof( char));
        alpht->alph = ( char *) calloc( strlen( alph) + 1,
1240     sizeof( char));
        strcpy( alpht->name, name);
        strcpy( alpht->alph, alph);

        return alpht;
1245 }

    free_alph_type( alph_type *alpht) {
        if( alpht != NULL) {
1250     free( alpht->name);
        free( alpht->alph);
        }
        free( alpht);
1255 }

    conf_type *new_conf_type( ){
        conf_type *conf;
        conf = (conf_type *) calloc( 1, sizeof( conf_type));
1260

        conf->oligolen = 9;
        conf->fasta_in = 0;
        conf->stat_in = 0;
        conf->max_gene_hit_frac = 0.1;
1265     conf->max_gene_hit = 999999999;
        conf->target_minus_self = 10;
        conf->target_min_temp = 60;
        conf->complement_flag = 0;
        conf->max_product = 0;
1270     conf->max_select = 10000;
        conf->no_5end_spike = 0;

```

图17 (续)

```

        return conf;
    }
1275

    int main (int argc, char* argv[]) {
        int i;
        stat_type *stat;
1280    int n;
        alph_type *dna = NULL;
        char *options;
        int next_option;
        conf_type *conf;
1285    int print_flag = 0;

        verbose = 1;

        conf = new_conf_type();
1290

        const char* const short_options =
            "vhl:fscrd:t:i:pnm";
        const struct option long_options[] = {
1295    { "verbose", 0, NULL, 'v' },
        { "help", 0, NULL, 'h' },
        { "oligo_len", 1, NULL, 'l' },
        { "fasta_in", 0, NULL, 'f' },
        { "stat_in", 0, NULL, 's' },
1300    { "complement", 0, NULL, 'c' },
        { "max_product", 0, NULL, 'm' },
        { "no_5end_spike", 0, NULL, 'n' },
        { "target_minus_self", 1, NULL, 'd' },
        { "target_min_temp", 1, NULL, 't' },
1305    { "max_gene_hit_frac", 1, NULL, 'i' },
        { "dump_stat", 0, NULL, 'p' },
        { NULL, 0, NULL, 0 } /* Re-
quired at end of array. */
    };
1310

        program_name = argv[0];
        run_time( TIME_INIT);

        do {
1315    next_option = getopt_long (argc, argv,
            short_options,
                long_options, NULL);
            switch (next_option)
            {
1320    case 'l': /* -l or --oligo_len*/

```

图17 (续)

```

        options = optarg;
        conf->oligolen = atoi( options);
        break;
1325     case 't':    /* -t or --target_min_temp*/
            options = optarg;
            conf->target_min_temp = atoi( options);
            break;
1330     case 'd':    /* -d or --target_minus_self*/
            options = optarg;
            conf->target_minus_self = atoi( options);
            break;
1335     case 'i':    /* -i or --max_gene_hit_frac*/
            options = optarg;
            conf->max_gene_hit_frac = atof( options);
            break;
            case 'f':    /* -f or --fasta_in*/
                conf->fasta_in = 1;
                break;
1340     case 's':    /* -s or --stat_in*/
                conf->stat_in = 1;
                break;
            case 'v':    /* -v or --verbose*/
                verbose++;
                break;
1345     case 'p':    /* -p or --dump_stat*/
                print_flag = 1;
                break;
            case 'c':    /* -c or --complement*/
                conf->complement_flag = 1;
1350                break;
            case 'm':    /* -m or --max_product*/
                conf->max_product = 1;
                break;
1355     case 'n':    /* -n or --no_5end_spike*/
                conf->no_5end_spike = 1;
                break;
            case 'h':    /* -h or --help*/
                usage("");
                break;
1360     }
    }
    while (next_option != -1);

1365     dna = new_alph( "dna", "acgt");

    if( verbose >= 1 ) {
        printf( "# Parameters:\n");

```

图17 (续)

```

1370     printf( "# oligo length      = %d\n", conf-
>oligolen);
        printf( "# max_gene_hit_frac = %f\n", conf-
>max_gene_hit_frac);
        printf( "# target_minus_self = %d\n", conf-
1375 >target_minus_self);
        printf( "# target_min_temp  = %d\n", conf-
>target_min_temp);
        printf( "# complement_flag   = %d\n", conf-
>complement_flag);
1380     printf( "# max_product        = %d\n", conf-
>max_product);
        printf( "# no_5end_spike_flag = %d\n", conf-
>no_5end_spike);
        printf( "# max_select        = %d\n", conf-
1385 >max_select);
    }

    if( conf->fasta_in) {
        printf("# Scanning fasta file\n");
1390     stat = read_fasta4stat( stdin, conf, dna);
        run_time( LAP_TIME);
        printf("# Read %d nmer statistics in %d s\n", stat-
>n_nmer, t_run);
    }
    else if( conf->stat_in) {
        printf("# Reading statistics\n");
        stat = read_stat( stdin, conf, dna);
        run_time( LAP_TIME);
        printf("# Read %d nmer statistics in %d s\n", stat-
1400 >n_nmer, t_run);
    }
    else
        usage( "Please use either fasta or stat as input
data.");
1405     printf( "# Found %d genes\n", stat->max_n_gene);
        conf->max_gene_hit = (conf->max_gene_hit_frac * stat-
>max_n_gene);
        printf( "# max_gene_hit      = %d\n", conf-
>max_gene_hit);
1410     printf("# Sorting genes\n");
        sort_gene( stat);
        run_time( LAP_TIME);
        printf("# Genes sorted in %d s\n", t_run);
1415     if( print_flag ) {
        compact_stat( stat);
        print_stat( stat);
    }

```

图17 (续)

```

    }
1420     else {

        if( conf->complement_flag){
            printf("# Join complement nmers\n");
            comp_stat( stat);
1425         run_time( LAP_TIME);
            printf("# Joined nmers in %d s\n", t_run);
        }
        else {
            compact_stat( stat);
1430         }

        printf("# Get cover\n");
        n = get_cover1( stat, conf);
        run_time( LAP_TIME);
1435         printf("# Got %d nmer cover in %d s\n", n, t_run);

        if(0)print_n_gene( stat);
        /* print_stat( stat); */
    }

1440     /*free_stat( stat); */
    free( conf);

    run_time( TOTAL_TIME);
1445     printf("# Run time %d s\n", t_run);
    exit(0);
}

1450
Program dyp.c
/*

    #Id: dyp.c,v 1.5 2003/04/23 09:33:50 nt Exp $
1455
DESCRIPTION
    Loading      sequences:
        -f fastafile
        -seq sequence
1460

    Set min_score to 0 to see all structures
        -min_score min_score

    Size of      sliding      window default is INT_MAX
1465     -depth depth

    Number of results to show

```

图17 (续)

```

        -max_res max_res

1470      -secondary_structure
        Calculates the secondary structure of an oligo

        -self_annealing
        Calculates the binding energy between an oligo
1475 and it self.

        -hybridization
        Calculates the binding energy between two differ-
1480 ent oligos, or an oligo and its target.

AUTHOR
1485      Copyright Niels Tolstrup 2002 and 2003
        Exiqon
        */

#include <stdio.h> /* fprintf, rand..*/
1490 #include <stdlib.h> /* calloc.. */
#include <stdarg.h> /* va_list.. */
#include <string.h> /* strlen.. */
#include <limits.h> /* USHRT_MAX.. */
#include <math.h> /* pow */
1495 #include <ctype.h> /* tolower */
#include "getopt.h" /* getopt_long */
#ifdef LIB
#include "dyp.h" /* param_t.. */

1500 #endif

/* export MALLOC_TRACE=dyp_memtrace
   mtrace dyp $MALLOC_TRACE
   #include <mcheck.h> mtrace
1505 */

/* memory debugging with dmalloc
   * Change Makefile:
   * LIBS = -ldmalloc
1510 * CFLAGS = -g -Wall -lm -
   L/home/sites/site2/users/lnatools/web/hybridization/src
   /dmalloc/dmalloc-4.8.2 -DDMALLOC -
   I/home/sites/site2/users/lnatools/web/hybridization/src
   /dmalloc/dmalloc-4.8.2
1515 *
   * run:

```

图17 (续)

```

    * ../dmalloc/dmalloc-4.8.2/dmalloc -b -l
/home/sites/site2/users/lnatools/web/hybridization/src/
dyp-0.0.1/dmalloc.log -i 10 low -p check-fence -p
1520 check-heap
    *
    * compile and run and check dmalloc.log
    *
    #include "dmalloc.h"
1525 */

1530
    /***** DEFINES *****/

#define THEVERSION "1.3 2002-11-06-13-52"
#define TRACEBACK_INT unsigned short
1535 #define TRACEBACK_INT_MAX USHRT_MAX

#define HI printf( "Hello\n");

#define SECONDARY_STRUCTURE 1
1540 #define SELF_ANEALING 2
#define HYBRIDIZATION 4
#define TM 8

#define ALLOLIG 1
1545 #define CLUSTER 2

/* Increase the stack by this amount in case of
overflow */
#define STACK_CHUNK 255
1550

/* Number of sequences to allocate space for before re-
alloc */
#define SEQS_CHUNK 255

1555 /* Size of sequence to allocate space for before re-
alloc */
#define SEQ_CHUNK 1024

/* Longest possible line in a matrix that can be read
*/
1560 #define MAX_LINE_LEN 5000

#define MIN(a,b) (a<b?a:b)
#define MAX(a,b) (a>b?a:b)
1565

```

图17 (续)

```

/***** GLOBAL VARIABLES *****/

1570  int      verbose      = 0;
      char*    program_name;

1575  /***** STRUCTURES *****/

      #ifndef LIB
      typedef struct sequences_t {
1580      int      nseq;
            int      maxnseq;
            char    **seqs;
            } sequences_type;

1585  typedef struct dynamic_str_t {
            int      len;
            int      maxlen;
            char     *s;
1590      } dynamic_str_type;
      #endif

1595  typedef struct pair_t {
            TRACEBACK_INT      i;
            TRACEBACK_INT      j;
            } pair_type;

1600  typedef struct stack_t {
            TRACEBACK_INT      sp;
            TRACEBACK_INT      size;
            pair_type          *stack;
1605      } stack_type;

      typedef struct tokenarray_t {
1610      int ntok;
            int maxntok;
            char    **tok;
            } tokenarray_type;
```

图17 (续)

```

1615     typedef      struct matrix_t  {
            char *seq1;
            char *seq2;
            int  l1;
            int  l2;
1620     int  max_size;
            int  *d;

            } matrix_type;

1625
        #ifndef LIB
        typedef      struct score_rec_t {
            int      alph_len;
            int      gap_start;
1630     int      gap_cont;
            int      mismatch_cont;
            int      loop_score;
            int      match_cont_factor;
            int      match_threshold;
1635     int      min_strong_ident_score;
            int      min_ident_score;
            int      min_sim_score;
            char      *alph;
            int      *mat;
1640     } score_rec_type;

        typedef      struct param_t {
            char      *alph;
1645     int      *mathyp;
            int      depth;
            int      global;
            int      min_score;
            int      max_res;
1650     sequences_type *seqs;
            sequences_type *seqs2;
            score_rec_type *score_rec;
        } param_type;
        #endif
1655

        /***** SYSTEM *****/

1660 void die( char *fmt, ...){
            va_list ap;
            fprintf( stderr, "Uhoh: ");
            va_start( ap,  fmt);

```

图17 (续)

```

    fprintf( stderr, fmt, ap);
1665    va_end( ap);
    fprintf( stderr, "\n");
    if(1)exit(    1);
}

1670
#ifdef LIB
void usage( char *fmt, ...){
    va_list ap;
    va_start( ap,    fmt);
1675    fprintf( stderr, fmt, ap);
    va_end( ap);
    fprintf( stderr,
        "\nUsage: dyp -h,--help\n"
        "    -v,--verbose\n"
1680    "    -r,--version\n"
        "    -u,--secondary_structure\n"
        "    -a,--self_anealing\n"
        "    -y,--hybridization\n"
        "    -t,--tm\n"
1685    "    -o,--output filename\n"
        "    -i,--input filename\n"
        "    -j,--input2 filename2\n"
        "    -s,--seq sequence\n"
        "    -z,--seq2 sequence2\n"
1690    "    -d,--depth depth\n"
        "    -g,--global\n"
        "    -n,--min_score score\n"
        "    -p,--max_res    n\n"
        "    -m,--matrix matfn\n"
1695    "    -l,--allolig length_of_oligo\n"
        "    -w,--n_mer_range nmer_start-
nmer_end[,spike_start-spike_end]\n"
        "    -c,--cluster score_cutoff\n"
        "    -k,--n_okseq include_first_n_seqs\n"
1700    "    -f,--spikefreq freq\n"
        "    -e,--sample n_samples\n"
        "\n"
        "DESCRIPTION\n"
        "    n_mer_range is used with allolig to specify
1705 a subset"
        "    if only nmer_start and nmer_end are given
        lna numbering is used"
        "    if the spike range is given, dna numbering
        is used");
1710    fprintf( stderr, "\n");
    exit(    1);
}

```

图17 (续)

```
#endif

1715 void hi( char *fmt, ...){
    va_list ap;
    fprintf( stdout, "Hi there " );
    va_start( ap,  fmt);
1720    vfprintf( stdout, fmt, ap);
    va_end( ap);
    fprintf( stdout, "\n");
    fflush( stdout);
}

1725 void *salloc( int nobj, int size){
    void *mem;
    /*mem =  calloc(      nobj, size);*/
1730    mem =      malloc(      nobj * size);
    /*printf( "calloc: allocating      %d objects of size
%d bytes\n",      nobj, size);*/
    if( mem == NULL){
        die( "Could not allocate %d x %d bytes\n", nobj,
1735 size);
    }
    return mem;
}

1740 FILE *sfopen( char *fn, const char *mode){
    FILE *file;
    file = fopen(  fn, mode);
    if( file == NULL )
1745    die( "Could not open %s for %s.", fn, mode);
    return file;
}

1750 int check_limits(){
    int int_bits;
    int long_bits;

    int_bits = log( UINT_MAX)/ log( 2) + 0.5;
1755    long_bits = log( ULONG_MAX)/ log( 2) + 0.5;
    if( int_bits != 32 ) {
        die( "Int must be 32 bits, but is %d\n", int_bits);
    }
    if( long_bits != 32 ) {
1760    die( "Long must be 32 bits, but is %d\n",
long_bits);
    }
```

图17 (续)

```
1765 void check_endian(){
    long n = 1;
    if( n >> 1) {
        /* Can this happen ??? */
1770     die( "Endian issue\n");
    }
}

1775 /***** STRING FUNCTIONS *****/

char *empty_str( int len){
    char *str;
1780     str = calloc( len+1, sizeof( char));
    memset( str, ' ', len);
    str[len] = 0;
    return str;
}

1785

char *reverse( char *str){
    int i, len;
    char c;
1790     len = strlen( str);
    for( i=0; i<len/2; i++){
        c = str[i];
        str[i] = str[(len-1) - i];
1795     str[(len-1) - i] = c;
    }
    return str;
}

1800

char *lower( char *s) {
    int i, l = strlen( s);
    for( i=0; i<l; i++)
        s[i] = tolower( s[i]);
1805     return s;
}

char tocomp( char c) {
1810     char fr_s[35] = "acgtumrwsykvhdbxnACGTUMRWSYKVHDBXN";
```

图17 (续)

```

char to_s[35] = "tgcaakywsrmbdhvxxTGCAAKYWSRMBDHVXX";
char *p;
char cc;

1815  p = strchr( fr_s, c);
      if( p ) {
          p = to_s + (p - fr_s);
          cc = *p;
      }
1820  else
      cc = 'x';
      return cc;
  }

1825  char *comp( char *s) {
      int i, l = strlen( s);
      for( i=0; i<l; i++)
          s[i] = tocomp( s[i]);
1830  return s;
  }

/***** DYNAMIC STRINGS *****/

1835

void free_dynamic_str_type( dynamic_str_type *seq) {
    free( seq->s);
    free( seq);
1840 }

dynamic_str_type *new_dynamic_str_type(){
    dynamic_str_type *seq;

1845  seq = calloc( 1, sizeof( dynamic_str_type));
    seq->len = 0;
    seq->maxlen = SEQ_CHUNK;
    seq->s = calloc( seq->maxlen + 1, sizeof(
1850 char));
    seq->s[0] = '\0';
    return seq;
}

1855 dynamic_str_type *addchar( dynamic_str_type *seq, int
c) {
    if( seq == NULL) {
        seq = new_dynamic_str_type();

```

图17 (续)

```

1860     )
        if( seq->len + 1 >= seq->maxlen ) {
            seq->maxlen += SEQ_CHUNK;
            if( ! (seq->s = realloc( seq->s, (seq->maxlen +
1865         1) * sizeof( char))) )
                die( "Failed to realloc seq to %d chars", seq-
>maxlen);
        }
        seq->s[ seq->len++] = c;
        return seq;
1870     }

    /***** I/O of strings *****/

1875 void add_seq2seqs( char *seq, sequences_type *seqs){
    int new_size, i;

    if( seqs->nseq < seqs->maxnseq ) {
1880     seqs->seqs[ seqs->nseq++] = seq;
    }
    else {
        new_size = (seqs->maxnseq + SEQS_CHUNK) * sizeof(
char *);
1885     if( (seqs->seqs = realloc( seqs->seqs, new_size)))
    {
        seqs->maxnseq = seqs->maxnseq + SEQS_CHUNK;
        for( i=seqs->nseq; i<seqs->maxnseq; i++)
            seqs->seqs[i] = NULL;
1890     seqs->seqs[ seqs->nseq++] = seq;
    }
    else {
        die( "Failed to realloc seq array to %d bytes",
new_size);
1895     }
    }
}

1900 sequences_type *new_sequences_type( int maxnseq){
    int i;
    sequences_type *seqs;
    seqs = calloc( 1, sizeof( se-
quences_type));
1905     seqs->nseq = 0;
    seqs->maxnseq = maxnseq;
    seqs->seqs = calloc( seqs->maxnseq, sizeof( char
*));

```

图17 (续)

```

    for( i=0; i<seqs->maxnseq; i++)
1910     seqs->seqs[i] = NULL;

    return seqs;
}

1915 void free_sequences_type( sequences_type *seqs) {
    free( seqs->seqs);
    free( seqs);
}

1920 void print_lines( FILE *outfile, char **str, int nstr){
    int linelen = 80;
    int i, j, p, len;

1925     if( nstr > 0 ) {
        len = strlen( str[0]);

        for( j=0; j<nstr; j++) {
            if( strlen( str[j]) != len ) {
1930                 die( "String len = %d differs from first string
                    len %d %s\n",
                        strlen( str[j]), len, str[j]);
            }
        }

1935         p = 0;

        while( p < len) {
            for( j=0; j<nstr; j++) {
                for( i = p; i < p + linelen && i < len; i++) {
1940                     putc( str[j][i], outfile);
                }
                putc( '\n', outfile);
            }
            p += linelen;
1945             putc( '\n', outfile);
        }
    }
}

1950 void print_fasta( FILE *outfile, sequences_type *seqs)
{
    int i;
    char *seq;

1955     for( i=0; i<seqs->nseq; i++) {
        fprintf( outfile, ">seq_%d\n", i);
    }
}

```

图17 (续)

```

        seq = seqs->seqs[i];
        print_lines( outfile, &seq, 1);
1960    }
    }

void read_fasta( FILE *seqfile,      sequences_type
1965  *seqs, char *alph){
    int          c;
    char          name[255];
    char          comment[255];
    char          s[255];
1970    int          seq_flag      = 0;
    dynamic_str_type *seq      = NULL;

    while( (c = fgetc( seqfile)) != EOF){
        if(      c == '>' ) {
1975            if( seq != NULL && seq->len > 0 ) {
                add_seq2seqs( seq->s, seqs);
                free( seq);
                seq = new_dynamic_str_type();
            }
            fgets( s, 254, seqfile);
1980            sscanf( s, "%s %s", name, comment);
            seq_flag = 1;
        }
        else if( seq_flag == 1 ) {
1985            if( strchr( alph, c) != NULL ) {
                seq = addchar( seq, c);
            }
        }
    }
1990    if( seq != NULL && seq->len > 0 )
        add_seq2seqs( seq->s, seqs);

    free(      seq);
1995 }

/***** STACK *****/

2000 stack_type *init_stack( TRACEBACK_INT size) {
    stack_type *stack;

    stack          = calloc( 1, sizeof( stack_type));
    stack->sp = 0;
2005    stack->size      = size;

```

图17 (续)

```
        stack->stack = calloc( size, sizeof(
            pair_type));

        return stack;
2010    }

    void free_stack( stack_type *stack){
        free(      stack->stack);
2015    free(      stack);
    }

    void print_pair( pair_type pair){
2020    printf( "%4d,  %4d\n",      pair.i,      pair.j);
    }

    void push( stack_type *stack, pair_type pair) {
2025        int      new_size;
        if(      stack->sp < stack->size ) {
            stack->stack[ stack->sp++] = pair;
            if(0){printf("pushed :"); print_pair( pair);}
        }
2030        else {
            new_size = (stack->size + STACK_CHUNK) * sizeof(
pair_type);
            if( (stack->stack = realloc( stack->stack,
new_size))) {
2035                stack->size = stack->size +
STACK_CHUNK;
                stack->stack[ stack->sp++] = pair;
            }
            else {
2040                die( "Failed to realloc stack to %d bytes",
new_size);
            }
        }
    }
2045

    pair_type setpair( int i, int j) {
        pair_type pair;
        pair.i = i;
2050    pair.j = j;
        return pair;
    }
```

图17 (续)

```
2055 pair_type pop( stack_type *stack) {
        if( stack->sp > 0 )
            return stack->stack[      --(stack->sp)];
        else
            return setpair( TRACEBACK_INT_MAX, TRACE-
2060 BACK_INT_MAX);
    }

    void test_stack(){
2065     stack_type *stack;
        int i;

        stack      = init_stack( STACK_CHUNK);

2070     for( i=0; i<30; i++)
        push( stack,  setpair( i, i+1));

        for( i=0; i<30; i++)
            print_pair( pop( stack));
2075     free_stack( stack);
    }

2080  /***** OLIGO FUNCTIONS *****/

    long n_nmer( int n_mer, char *alph){
        int alphlen = strlen( alph);
2085     long n_nmer  = 0;

        if( alphlen == 0 )
            n_nmer = 0;
        else
2090     n_nmer = pow( alphlen, n_mer);

        return n_nmer;
    }

2095     long big_nmer( int n){
        int i;
        long n_tmp;
        long mask = 1;
2100     long res = 0;

        for( i=0; i<16; i++){
            n_tmp = n & mask;
```

图17 (续)

```

    n_tmp = n_tmp << 3;
2105    res += n_tmp;
    mask = mask << 1;
}

return res;
2110 }

char *make_nmer( long number, int n_mer, char *alph) {
    long i, j, n = number;
2115    int alphlen = strlen( alph);
    char *seq;

    seq = empty_str( n_mer);

2120    if( alphlen == 0 )
        strcpy( seq, "--");
    else {
        for( j=0; j<n_mer; j++){
            i = n % alphlen;
2125            n = n / alphlen;
            seq[j] = alph[i];
        }
    }
    seq = reverse( seq);
2130    return seq;
}

2135 int rando( int max) {
    int r;
    do
        r = (rand() * max)/RAND_MAX;
    while( r==max);
2140    return r;
}

char *make_random_nmer( int n_mer, int spikefreq, char
2145 *alph) {
    long i, j;
    int alphlen = strlen( alph);
    int halflen = alphlen / 2;
    char *seq;
2150    char phase = 0;

    seq = empty_str( n_mer);
```

图17 (续)

```

2155     if( spikefreq)
        phase = rando( spikefreq);

        if( alphlen == 0 )
            strcpy( seq, "-");
2160     else {
        if( spikefreq)
            for( j=0; j<n_mer; j++){
                i = rando( halflen);
                if( ( (j+phase) % spikefreq) == 0)
2165                 i += halflen;
                seq[j] = alph[i];
            }
        else
            for( j=0; j<n_mer; j++){
2170                 i = rando( alphlen);
                seq[j] = alph[i];
            }
        }
        return seq;
2175     }

/***** MATRIX I/O *****/
2180

score_rec_type *init_score_rec_type( int gap_start, int
        gap_cont, int mismatch_cont,
        int loop_score, int
2185 match_threshold,
        int match_cont_factor,
        int min_strong_ident_score,
        int min_ident_score, int
        min_sim_score,
2190 char *alph, int *mat){
    score_rec_type *score_rec;

    score_rec = (score_rec_type *) calloc( 1,
sizeof( score_rec_type));
2195    score_rec->alph_len = strlen( alph);
    score_rec->gap_start = gap_start;
    score_rec->gap_cont = gap_cont;
    score_rec->mismatch_cont = mismatch_cont; /* a
negativ value turn this feature off */
2200    score_rec->loop_score = loop_score;
    score_rec->match_threshold = match_threshold;

```

图17 (续)

```

        score_rec->match_cont_factor    = match_cont_factor;
        score_rec->alph                  = alph;
        score_rec->mat                   = mat;
2205   score_rec->min_strong_ident_score =
min_strong_ident_score;
        score_rec->min_ident_score      = min_ident_score;
        score_rec->min_sim_score        = min_sim_score;

2210   return score_rec;
    }

void free_score_rec_type( score_rec_type *score_rec){
2215   free(      score_rec->alph);
        free(      score_rec->mat);
        free(      score_rec);
    }

2220
void free_param_type( param_type *param){
        free_score_rec_type( param->score_rec);
        free_sequences_type( param->seqs);
        free_sequences_type( param->seqs2);
2225   free(      param);
    }

void validate_score_rec_type( score_rec_type
2230   *score_rec, int min, int max){
    int i, j;
        if( strlen( score_rec->alph) < 1){ die( "No alpha-
bet\n"); }
        if( strlen( score_rec->alph) != score_rec->alph_len)
2235   {
            die( "alph length mismatch %d,%d\n", strlen(
score_rec->alph),
                                                    score_rec-
>alph_len);
2240   }
        for( i=0;i<score_rec->alph_len;i++){
            for( j=0;j<score_rec->alph_len;j++){
                if( score_rec->mat[i*score_rec->alph_len+j]<min )
                    die( "Matrix out of range min=%d", min);
2245   if( score_rec->mat[i*score_rec->alph_len+j]>max )
                    die( "Matrix out of range %d max=%d",
                        score_rec->mat[i*score_rec-
>alph_len+j], max);
            }
2250   }

```

图17 (续)

```

        if( score_rec->gap_start  > max ) die( "gap_start out
            of range max");
        if( score_rec->gap_start  < min ) die( "gap_start out
            of range max");
2255    if( score_rec->gap_cont    > max ) die( "gap_cont out
            of range    max");
        if( score_rec->gap_cont    < min ) die( "gap_cont out
            of range    max");
        if( score_rec->loop_score > max ) die( "loop_score
2260    out of range max");
        if( score_rec->loop_score < min ) die( "loop_score
            out of range max");
    }

2265    void print_mat(  char *seq1, char *seq2, int *mat){
        int i, j;
        int len1 =      strlen(      seq1);
        int len2 =      strlen(      seq2);
2270    int p_num =      1;

        if( p_num == 1) {
            printf( "      ");
            for( i=0;i<len1;i++){
2275                printf( " %2d  ", i);
            }
            printf( "\n  ");
        }
        printf( "  ");
2280    for( i=0;i<len1;i++){
            printf( "    %c ", seq1[i]);
        }
        printf( "\n\n");
        for( j=0;j<len2;j++){
2285            if(      p_num == 1) {
                printf( "%2d %c ", j, seq2[j]);
            }
            else {
                printf( "%c ", seq2[j]);
2290            }
            for( i=0;i<len1;i++){
                printf( "%5d ", mat[j*len1+i]);
            }
            printf( "\n");
2295        }
    }

    /***** TOKENIZE A LINE *****/

```

图17 (续)

2300

```
void addtoken( char* token, tokenarray_type *tokena){
    tokena->ntok++;
    if( tokena->ntok > tokena->maxntok ) {
2305     tokena->maxntok += 100;
        if( ! (tokena->tok =
                realloc( tokena->tok, tokena->maxntok *
                    sizeof( char))) )
            die( "Failed to realloc tokenarray to %d to-
2310 kens", tokena->maxntok);
        }
        tokena->tok[ tokena->ntok - 1] = token;
    }
}
```

2315

```
tokenarray_type *alloc_tokenarray_type(){
    tokenarray_type *tokena;

    tokena = calloc( 1, sizeof( tokenarray_type));
2320     tokena->maxntok = 30;
    tokena->ntok = 0;
    tokena->tok = calloc( tokena->maxntok, sizeof(
char*));
    return tokena;
2325 }
```

```
void free_tokenarray_type( tokenarray_type *tokena){
    free( tokena->tok);
2330     free( tokena);
}
```

```
void tokenize( char *s, tokenarray_type *tokena){
2335     char *token;

    tokena->ntok = 0;
    token = strtok( s, " \n");
    if( token )
2340         addtoken( token, tokena);
    while( (token = strtok( NULL, " \n")) )
        addtoken( token, tokena);
}
```

2345

```
int tokenarray_is_num_list( tokenarray_type *tokena){
    int i = 0;
    char *endp;
```

图17 (续)

```

    int res = 0;
2350    while( i<tokena->ntok && strtol( tokena->tok[i],
        &endp, 10) == i) {
        if(      *endp != 0 )
            res = 0;
2355        i++;
    }
    if( i      == tokena->ntok)
        res    = i;
    else
2360        res    = 0;
    return res;
}

2365 char *tokenarray2str( tokenarray_type *tokena){
    int i=0;
    char *alph;

    alph = calloc( tokena->ntok + 1, sizeof( char));
2370    for( i=0; i<tokena->ntok; i++) {
        if(      strlen(      tokena->tok[i])    != 1)
            die( "Sorry, the word %s was found were a char was
                expected.",
                                tokena-
2375    >tok[i]);
        alph[i] = *tokena->tok[i];
    }
    alph[i+1] = 0; /* EOL */

2380    return alph;
}

int tokenarray2dat( tokenarray_type *tokena, int
2385 with_num,
                                matrix_type *mt, char *c){
    int i = 0;
    int j = 0;
    char *endp;
2390    int err = 0;
    int *dat;

    if( with_num ) {
        if(      atoi( tokena->tok[i]) != mt->l2)
2395        err = 1;
        i++;
    }
}

```

图17 (续)

```

        if( strlen(tokena->tok[i]) != 1)
            err = 2;
2400    *c = *tokena->tok[i];
        i++;

        if( tokena->ntok - i != mt->l1 )
            err = 3;
2405    else {
        dat = calloc( mt->l1, sizeof( int));
        for( ; i<tokena->ntok; i++) {
            dat[j] = strtol( tokena->tok[i], &endp, 10);
            if( *endp != 0 )
2410                err = j + 10;
            j++;
        }
    }
    if( !err) {
2415        if( mt->l1 * mt->l2 > mt->max_size ) {
            mt->max_size += 100;
            if( ! (mt->d =
                realloc( mt->d, mt->max_size * sizeof(
2420 int))) )
                die( "Failed to realloc matrix to %d int's",
mt->max_size);
            }
            for( j=0; j<mt->l1; j++)
                mt->d[mt->l1 * mt->l2 + j] = dat[j];
2425        mt->l2++;
    }
    free( dat);
    return err;
}
2430

/***** I/O score_rec_type *****/

2435 matrix_type *read_mat( FILE *file){
    char *s, *res;
    int n = MAX_LINE_LEN;
    int col_name_found = 0;
    int numbers_found = 0;
2440 tokenarray_type *tokena;
    dynamic_str_type *seq = NULL;
    char c;
    matrix_type *mt;

2445    mt = calloc( 1, sizeof( matrix_type));
    mt->max_size = 100;

```

图17 (续)

```

mt->d          = calloc( mt->max_size, sizeof(
    int));
mt->l2         = 0;
2450 tokena      = alloc_tokenarray_type();

s             = calloc( n + 1, sizeof( char));
seq          = new_dynamic_str_type();
2455 do {
    res        = fgets( s, n + 1, file);
    if( res != NULL ) {
        tokenize( s, tokena);
2460 /*for(i=0;i<tokena->ntok;i++) printf( "%d %s\n",
i, tokena->tok[i]);*/
        if( !col_name_found ) {
            /* Is it a column number line? */
            if( !numbers_found && (mt->l1 = tokenar-
2465 ray_is_num_list( tokena))) {
                numbers_found = 1;
            }
            else {
                /* Is it a column name line? */
2470 mt->seq1 = tokenarray2str( tokena);
                if( numbers_found ) {
                    if( mt->l1 != strlen( mt->seq1))
                        die("Expected a string of length %d but
found %s", mt->l1,
2475 >seq1);
                    }
                    else
                        mt->l1 = strlen( mt->seq1);
2480 col_name_found = 1;
                }
            }
            else {
                /* skip empty lines */
2485 if( tokena->ntok > 0) {
                    /* Is it a data line? */
                    if( tokenarray2dat( tokena, numbers_found, mt,
&c) == 0) {
2490 seq = addchar( seq, c);
                    }
                    else {
                        res = NULL;
                    }
                }
2495 }

```

图17 (续)

```

    }
    } while( res);

    mt->seq2 = calloc( strlen( seq->s) + 1, sizeof(
2500 char));

    strcpy( mt->seq2, seq->s);
    free_dynamic_str_type( seq);

2505    free(      s);
    free_tokenarray_type( tokena);
    return mt;
}

2510 void print_score_rec_type( score_rec_type *score_rec){
    printf( "alph_len   %d\n", score_rec->alph_len);
    printf( "gap_start  %d\n", score_rec->gap_start);
    printf( "gap_cont   %d\n", score_rec->gap_cont);
2515    printf( "alph      %s\n", score_rec->alph);
    print_mat( score_rec->alph, score_rec->alph,
score_rec->mat);
}

2520 void read_score_rec_type( char *fn, score_rec_type
*score_rec){
    FILE      *file;
    matrix_type *mt;

2525    file = sfopen( fn, "r");
    fscanf( file,  "alph_len   %d\n", &(score_rec-
>alph_len));
    fscanf( file,  "gap_start  %d\n", &(score_rec-
2530 >gap_start));
    fscanf( file,  "gap_cont   %d\n", &(score_rec-
>gap_cont));
    fscanf( file,  "min_strong_ident_score   %d\n",
&(score_rec->min_strong_ident_score));
2535    fscanf( file,  "min_ident_score   %d\n", &(score_rec-
>min_ident_score));
    fscanf( file,  "min_sim_score   %d\n", &(score_rec-
>min_sim_score));
    fscanf( file,  "alph      %s\n", score_rec->alph);
2540    mt = read_mat( file);
    if( mt->l1 != mt->l2)
        die("Matrix not quadratic %d %d", mt->l1, mt->l2);
    if( strcmp( mt->seq1, mt->seq2) != 0)

```

图17 (续)

```

    die("Matrix sequences not identical %s %s",
2545     mt->seq1, mt->seq2);
    if( strcmp( mt->seq1, score_rec->alph) != 0)
        die("Matrix alphabet differs from alph %s %s", mt-
>seq1, score_rec->alph);
    if( score_rec->mat )
2550     free( score_rec->mat);
    score_rec->mat = mt->d;
    free( mt->seq1);
    free( mt->seq2);
    free( mt);
2555     fclose( file);
}

/***** MATRIX *****/
2560

int *seq2idx_seq( char *seq, score_rec_type *score_rec)
{
    char      c[2];
2565     int i, l;
    int *idx_seq;
    int id;

    l      = strlen( seq);
2570     idx_seq = calloc( l, sizeof(      int));

    for( i=0; i<l; i++) {
        c[0] = seq[i];
        c[1] = 0;
2575         id      = strcspn( score_rec->alph, c);
        if(      id >= score_rec->alph_len) {
            fprintf( stderr,
                "Sorry,      the character %c was not found in the
alphabet %s\n",
2580                 seq[i],
                    score_rec->alph);
            id      = score_rec->alph_len - 1;
        }
        idx_seq[i]      = id;
2585     }
    return idx_seq;
}

2590 int matidx( int i, int j, int l1, int l2){
    if( i<0 ) die( "matrix index      i (%d) less than
zero",      i);

```

图17 (续)

```

        if( i>=l1 )      die( "matrix index i (%d)      too
large (%d)", i, l1);
2595    if( j<0 ) die( "matrix index      j (%d) less than
zero",      j);
        if( j>=l2 )      die( "matrix index j (%d)      too
large (%d)", j, l2);
        return      j*l1 + i;
2600    }

int matsize( int l1, int l2){
    return      l1*l2;
2605    }

int m( int i, int j, matrix_type *mat) {
    return      mat->d[      matidx(      i, j, mat->l1, mat-
2610    >l2)];
    }

void print_annot( char *seq, char *annot){
2615    printf("%s\n", seq);
    printf("%s\n", annot);
    }

2620 char match_sym(      char a,      char b,      score_rec_type
*score_rec) {
    int i1, i2;
    char      *s;
    int score;
2625
    s = strchr( score_rec->alph, a);
    if(      s == NULL) return ' ';
    i1      = s - score_rec->alph;
    s = strchr( score_rec->alph, b);
2630    if(      s == NULL) return ' ';
    i2      = s - score_rec->alph;

    score = score_rec->mat[ i2*score_rec->alph_len+i1];

2635    if( score >= score_rec->min_strong_ident_score )
        return '|';
    else if( score >= score_rec->min_ident_score )
        return ':';
    else if( score >= score_rec->min_sim_score )
2640    return '.';

```

图17 (续)

```
        return ' ';
    }

2645  /***** ALIGN - self association
      *****/

2650  void align_global_traceback( FILE *outfile, param_type
      *param,
                                char *seq1, char *seq2, int
      *path){
    int  l1    = strlen( seq1);
2655  int  l2    = strlen( seq2);
    int  done  = 0;
    int  is    = 0;
    int  i, j;
    int  i1, i2;
2660  char *qseq, *mseq, *tseq;

    if( l1>0 && l2>0 ) {

        qseq = calloc( l1+l2+1, sizeof( char));
2665  mseq = calloc( l1+l2+1, sizeof( char));
        tseq = calloc( l1+l2+1, sizeof( char));

        i  = l1 - 1;
        j  = l2 - 1;
2670  i1 = l1 - 1;
        i2 = l2 - 1;

        do {
            if( verbose )
2675  printf(      "%d %d\n", i, j);

            if( path[j*l1+i] == 0) {
                if( i1 >= 0 ) {
                    qseq[i1] = seq1[i1];
2680  i1--;
                }
                else {
                    qseq[i1] = '-';
                    mseq[i1] = ' ';
2685  }
                if( i2 >= 0 ) {
                    tseq[i1] = seq2[i2];
                    i2--;
                }
                else {
2690  }
```

图17 (续)

```

        tseq[is] = '-';
        mseq[is] = ' ';
    }
    /*
2695     if( tseq[is] == qseq[is] ) { mseq[is] =
        ':'; }
        else { mseq[is] = ' '; }
    */
    mseq[is] = match_sym( tseq[is], qseq[is], pa-
2700 ram->score_rec);
    is++;
    i--;
    j--;
    if( i<0 && j<0 ) {
2705     done = 1;
    }
}
else if( path[j*11+i] == 2) {
2710     qseq[is] = '-';
    mseq[is] = ' ';
    tseq[is] = seq2[i2];
    i2--;
    is++;
    j--;
2715 }
else if( path[j*11+i] == 1) {
    qseq[is] = seq1[i1];
    i1--;
    mseq[is] = ' ';
2720     tseq[is] = '-';
    is++;
    i--;
}
if( i < 0 ) i = 0;
2725 if( j < 0 ) j = 0;
if( verbose )
    printf( "%d %d %c %c %c\n", i, j, qseq[is-1],
mseq[is-1], tseq[is-1]);
} while( !done);
2730 qseq[is] = 0;
mseq[is] = 0;
tseq[is] = 0;

fprintf( outfile, "%s\n", reverse( qseq));
2735 fprintf( outfile, "%s\n", reverse( mseq));
fprintf( outfile, "%s\n", reverse( tseq));

free( qseq);
free( mseq);

```

图17 (续)

```
2740     free( tseq);
        }
    }

2745 int align_local_traceback( FILE *outfile, param_type
    *param,
        int l1, int l2, int *idx_seq1, int *idx_seq2,
        char *mask_seq1, char *mask_seq2, char *seq1, char
    *seq2,
2750     int *mat, int *path){

        stack_type      *stack;
        int              i, j, k;
        pair_type pair;
2755     char              *annot;
        matrix_type      *mt;
        int              score, maxscore, firstscore=0, thes-
core=INT_MAX, max_i, max_j;
        int              n_hyp = 0;
2760     char              *outstr[3];
        char              *qseq, *mseq, *tseq;

        stack            = init_stack( STACK_CHUNK);

2765     mt              = calloc( 1, sizeof( matrix_type));
        mt->d            = mat;
        mt->l1            = l1;
        mt->l2            = l2;

2770     qseq = calloc( l1+l2+1, sizeof( char));
        mseq = calloc( l1+l2+1, sizeof( char));
        tseq = calloc( l1+l2+1, sizeof( char));

        while( thescore >= param->min_score && n_hyp < param-
2775     >max_res) {

            maxscore = INT_MIN;
            for( j=0; j<l2; j++) {
                for( i=0; i<l1; i++) {
2780                     score = mat[ matidx( i, j, l1, l2)];
                        if(      score >      maxscore) {
                            maxscore = score;
                            max_i    = i;
                            max_j    = j;
2785                     }
                }
            }
        }
```

图17 (续)

```

        if( ! firstscore && maxscore >= param->min_score)
firstscore = maxscore;
2790     push( stack, setpair( max_i, max_j));
        thescore = mat[ matidx( max_i, max_j, l1, l2)];
        mat[ matidx( max_i, max_j, l1, l2)] = 1;
        /* Trace back */
2795     k=0;
        while( stack->sp > 0 ) {
            pair = pop( stack);
            if(0){ printf( "pop          : " ); print_pair(
2800 pair);}
            i      = pair.i;
            j      = pair.j;
            if( path[ matidx( i, j, l1, l2)] == 1 ) {
                if( i>0 && mat[ matidx( i-1, j, l1, l2)] > 0 ) {
2805     push( stack, setpair( i-1, j));
                }
                qseq[k] = seq1[i];
                mseq[k] = ' ';
                tseq[k] = '-';
2810     k++;
            }
            else if( path[ matidx( i, j, l1, l2)] == 2 ) {
                if( j>0 && mat[ matidx( i, j-1, l1, l2)] > 0 ) {
                push( stack, setpair( i, j-1));
2815     }
                qseq[k] = '-';
                mseq[k] = ' ';
                tseq[k] = seq2[j];
                k++;
2820     }
            else if( path[ matidx( i, j, l1, l2)] <= 0 ) {
                if( i>0 && j>0 && mat[ matidx( i-1, j-1, l1,
12)] > 0 ) {
                push( stack, setpair( i-1, j-1));
2825     }

                qseq[k] = seq1[i];
                tseq[k] = seq2[j];
                mseq[k] = match_sym( tseq[k], qseq[k], param-
2830 >score_rec);
                k++;
                if(0){printf( " ( ) %d %d\n", j, i );
                }
            }
        }
2835     if( thescore >= param->min_score ) {
        n_hyp++;

```

图17 (续)

```

        if( n_hyp  <= param->max_res) {
            qseq[k]  = 0;
            mseq[k]  = 0;
2840      tseq[k]    = 0;
            outstr[0] = reverse( qseq);
            outstr[1] = reverse( mseq);
            outstr[2] = reverse( tseq);
            if( outfile != NULL ) {
2845      fprintf( outfile, "Score= %d\n", thescore);
                print_lines( outfile, outstr, 3);
            }
        }
    }
2850      annot      = empty_str( l1);
        free( annot);
    }

    free( mt);
2855      free( qseq);
        free( mseq);
        free( tseq);
        free_stack( stack);
        return firstscore;
2860  }

void out_align_fill( param_type *param, int l1, int l2,
int *idx_seq1,
2865      int *idx_seq2, int *mat, int *path, char *seq1,
char *seq2){
    int i, j, k,  max_path;
    int score, max_score;
    int h_size = 3;
2870      int gap_score;
        int match_factor;

    /*
        0 = match or mismatch, go diagonally
2875      1 = insertion in query, go horizontally
        2 = gap in query, go vertically
    */

    for( j=0; j<l2; j++) {
2880      for( i=0; i<l1; i++) {
            if(!printf( "%d %d\n", i, j));

            if( path[ matidx( i, j, l1, l2)] == -1 )
                gap_score = param->score_rec->gap_cont;
2885      else

```

图17 (续)

```

        gap_score = param->score_rec->gap_start;
        max_score = mat[ matidx( i, j, 11, 12)]
- gap_score;
        max_path = -1;
2890
        if( path[ matidx( i, j+1, 11, 12)] == -2
        )
            gap_score = param->score_rec->gap_cont;
        else
2895
            gap_score = param->score_rec->gap_start;
            score = mat[ matidx( i, j+1, 11, 12)]
- gap_score;
            if( score > max_score) { max_score = score;
max_path = -2;}
2900
            if( i-j <= h_size ) {
                score = param->score_rec->loop_score;
            }
            else {
2905
                if( path[ matidx( i-1, j+1, 11, 12)] == 0 )
                    match_factor = 1;
                else
                    match_factor = param->score_rec-
>match_cont_factor;
2910
                score = mat[ matidx( i-1, j+1, 11, 12)] +
                    match_factor *
                    param->score_rec->mat[ idx_seq1[i] *
                    param->score_rec->alph_len + idx_seq2[j]];
            }
2915
            if(0) {
                printf( "%d %d %d %d %d %d %d\n", i, j,
idx_seq1[i], idx_seq2[j],
                    param->score_rec->alph_len,
2920
                    param->score_rec->mat[ idx_seq1[i] * param-
>score_rec->alph_len +
                    idx_seq2[j]],
                    idx_seq1[i] * param->score_rec->alph_len +
idx_seq2[j]);
2925
            }

            if( score > max_score) { max_score = score;
max_path = 0;}
            for( k=j+1; k<i; k++) {
2930
                score = mat[ matidx( i, k+1, 11, 12)] + mat[
matidx( k, j, 11, 12)]
                    - param->score_rec->gap_start;
                if( score > max_score) { max_score = score;
max_path = k;}

```

图17 (续)

```

2935     if(0)printf( "%d %d  %d %d  %d\n", i, k, k, j,
score);
        }

        if( max_score > 0 ) {
2940     mat[ matidx( i, j, l1, l2)] = max_score;
        }
        else {
            mat[ matidx( i, j, l1, l2)] = 0;
        }
2945     path[ matidx( i, j, l1, l2)] = max_path;
        if(0) {
            printf("%c%c      %3d %3d      score: %3d  path:
%3d\n", seq1[i], seq2[j],
                i, j, max_score, max_path);
2950     }
        }
    }
}

2955 void align_fill( param_type *param, int l1, int l2, int
*idx_seq1,
    int *idx_seq2, int *mat, int *path, char *seq1,
char *seq2){
2960     int i, j;
    char c1[2], c2[2];
    int i1, i2;
    int sm, si, sd, max_s;
    int dir;
2965     int match_score;

    if( verbose > 3 )
        printf( "%d  %d\n", l1, l2);

2970     /*
        -1 = mismatch, go diagnoally
        0 = match, go diagnoally
        1 = insertion in query, go horizontally
        2 = gap in query, go vertically
2975     */

    for( j=0; j<l2; j++) {
        c2[0] = seq2[j];

2980     c2[1] = 0;
        i2 = strchr( param->score_rec->alph, c2);
        for( i=0; i<l1; i++) {
            c1[0] = seq1[i];

```

图17 (续)

```

2985      c1[1] = 0;
      i1     = strchrn( param->score_rec->alph, c1);

      if( i > 0  && j > 0) {
2990      sm     =    mat[(j-1)*11+(i-1)];
      }
      else {
        /*
2995      if( i > 0 ) {
          sm     = param->score_rec->gap_start +
              (i-1) * param->score_rec->gap_cont;
        }
        else if( j > 0 ) {
3000      sm     = param->score_rec->gap_start +
              (j-1) * param->score_rec->gap_cont;
        }
        */
        sm = 0;
      }
3005      match_score = param->score_rec->mat[ i2*param-
>score_rec->alph_len+i1];
      dir = 0;
      if( param->score_rec->mismatch_cont < 0 )
        sm     += match_score;
3010      else {
        if( match_score < 0 ) {
          dir     = -1;
          if( (i > 0) && (j > 0) && (path[(j-1)*11+(i-1)]
3015      != 0)) {
            sm     -= param->score_rec->mismatch_cont;
          }
          else {
            sm     += match_score;
          }
3020      }
          else {
            sm     += match_score;
          }
        }
3025      /* Add mismatch continuation score here */

      /*if( (i > 0) && (path[j*11+(i-1)] > 0)) { */
      if( i > 0 ) {
3030      si     =    mat[j*11+(i-1)];
      }
      else {
        si     =    0;
      }

```

图17 (续)

```

    }

3035    /*if( (j > 0) && (path[(j-1)*11+i] > 0)) { */
        if( j > 0 ) {
            sd = mat[(j-1)*11+i];
        }
        else {
3040    sd = 0;
        }

        if( i>0 && path[j*11+(i-1)] > 0) {
            si -= param->score_rec->gap_cont;
3045    }
        else {
            si -= param->score_rec->gap_start;
        }

3050    if( j>0 && path[(j-1)*11+i] > 0) {
            sd -= param->score_rec->gap_cont;
        }
        else {
            sd -= param->score_rec->gap_start;
3055    }

            max_s = sm;
            if( si > max_s) {
                max_s = si;
3060    dir = 1;
            }
            if( sd > max_s) {
                max_s = sd;
                dir = 2;
3065    }

            mat[j*11+i] = MAX( max_s, 0);
            path[j*11+i] = dir;
            /*printf( "%s %s %d %d %d %d\n", c1, c2, i1,
3070    i2, score, mat[j*11+i]);*/
        }
    }

    if( 0) {
3075    print_mat( seq1, seq2, mat);
        printf( "\n");
        print_mat( seq1, seq2, path);
        printf( "\n");
    }

3080    /* printf( "score= %d\n", mat[11*12-1]); */

```

图17 (续)

```
3085  int align( FILE *outfile, param_type *param, char
      *seq1, char *seq2){
      int l1, l2;
      int *mat, *path;

3090  char *mask_seq1, *mask_seq2;
      int *idx_seq1, *idx_seq2;
      int score = INT_MAX;

      l1 = strlen( seq1);
3095  l2 = strlen( seq2);
      mat = calloc( matsize( l1, l2), sizeof( int));
      path = calloc( matsize( l1, l2), sizeof(
int));
      mask_seq1 = calloc( l1 + 1, sizeof( char));
3100  mask_seq2 = calloc( l2 + 1, sizeof( char));
      strcpy( mask_seq1, seq1);
      strcpy( mask_seq2, seq2);

3105  if(0) {
      print_mat( seq1, seq2, mat);
      printf( "\n");
      if(1) print_mat( seq1, seq2, path);
      printf( "\n");
3110  }

      align_fill1( param, l1, l2, idx_seq1, idx_seq2, mat,
path,
      mask_seq1, mask_seq2);
3115  if(0) {
      print_mat( seq1, seq2, mat);
      printf( "\n");
      if(1) print_mat( seq1, seq2, path);
      printf( "\n");
3120  }

      score = align_local_traceback( outfile, param, l1,
l2, idx_seq1, idx_seq2,
      mask_seq1, mask_seq2, seq1, seq2, mat, path);
3125  free( mat);
      free( path);
      free( mask_seq1);
      free( mask_seq2);
      return score;
3130 }
```

图17 (续)

```

/***** Nussinov - secondary structure prediction
*****/
3135
int nussinov_local_traceback( FILE *outfile, param_type
    *param,      int depth,
                  int l, int* idx_seq, char *seq,
3140      int *mat, int *path){
    stack_type    *stack;
    int           i, j, k;
    pair_type pair;
    char          *annot;
3145    matrix_type   *mt;
    int           score, maxscore, max_i, max_j;
    int           maxpath;
    int           n_hyp = 0;
    char          *outstr[2];
3150
    stack          = init_stack( STACK_CHUNK);

    mt             = calloc( 1, sizeof( matrix_type));
    mt->d           = mat;
3155    mt->l1         = 1;
    mt->l2         = 1;

    annot          = empty_str( 1);

3160    maxscore = INT_MIN;
    for( j=0; j<l; j++) {
        for( i=j; i<depth + j + 1 && i<l; i++) {
            score = mat[ matidx( i, j, 1, 1)];
            if( score > maxscore) {
3165                maxscore = score;
                max_i      = i;
                max_j      = j;
            }
        }
    }
3170
    mat[ matidx( max_i, max_j, 1, 1)] = 1;

    push( stack, setpair( max_i, max_j));
3175
    /* Trace back */

    while( stack->sp > 0 ) {
        pair = pop( stack);

```

图17 (续)

```

3180     if(0){ printf( "pop          : " ); print_pair(
pair);}
        i      = pair.i;
        j      = pair.j;
        if( i > j ) {
3185     if( path[ matidx( i, j, 1, 1) ] == -1 ) {
            if( mat[ matidx( i-1, j, 1, 1) ] > 0 ) {
                push( stack, setpair( i-1, j));
            }
        }
        else if( path[ matidx( i, j, 1, 1) ] == -2 ) {
3190     if( mat[ matidx( i, j+1, 1, 1) ] > 0 ) {
            push( stack, setpair( i, j+1));
        }
        else if( path[ matidx( i, j, 1, 1) ] == 0 ) {
3195     if( mat[ matidx( i-1, j+1, 1, 1) ] > 0 ) {
            {
                push( stack, setpair( i-1, j+1));
            }
3200     if( param->score_rec->match_threshold <=
            param->score_rec->mat[ idx_seq[i] * param-
            >score_rec->alph_len + idx_seq[j]]) {
                annot[i] = ')';
                annot[j] = '(';
3205     }
            else {
                annot[i] = '.';
                annot[j] = '.';
            }
3210     if(0)printf( " ( ) %d %d\n", j, i );
        }
        else {
            k = path[ matidx( i, j, 1, 1)];
            push( stack, setpair( k, j));
3215     push( stack, setpair( i, k+1));
        }
    }
}

3220 if( maxscore >= param->min_score ) {
    n_hyp++;
    if( n_hyp <= param->max_res ) {
        if(0)printf( "%s\n", seq );
        if(0)printf( "%s\n", annot );
3225     outstr[0] = seq;
        outstr[1] = annot;
        if( outfile != NULL ) {
            fprintf( outfile, "Score= %d\n", maxscore);
        }
    }
}

```

图17 (续)

```

        print_lines( outfile, outstr, 2);
3230    }
        }
        free( annot);
        annot      = empty_str( 1);
3235
        free( mt);
        free( annot);
        free_stack( stack);
3240
        return maxscore;
    }

3245 char *nussinov_traceback( FILE *outfile, param_type
    *param,
                                int depth, int l, int* idx_seq,
                                char *seq, int *mat, int *path){
        stack_type      *stack;
3250    int      i, j, k, d;
        pair_type pair;
        char      *annot;
        matrix_type *mt;
        int      gap_score1, gap_score2;
3255    char      *outstr[2];

        stack      = init_stack( STACK_CHUNK);

        mt          = calloc( 1, sizeof( matrix_type));
3260    mt->d          = mat;
        mt->l1        = 1;
        mt->l2        = 1;

        annot      = empty_str( 1);
3265
        for( d=depth; d<1; d++) {

            push( stack, setpair( d, d-depth));

3270    while( stack->sp > 0 ) {
            pair = pop( stack);
            if(0){ printf( "pop      : " ); print_pair(
pair);}
            i      = pair.i;
3275    j      = pair.j;
            if( i > j ) {
                if( path[ matidx( i, j, 1, 1)] == -1 ) {

```

图17 (续)

```

        push(      stack, setpair(  i-1, j));
    }
3280     else if( path[ matidx( i, j, 1, 1)] == -2 ) {
        push(      stack, setpair(  i, j+1));
    }
        else if( path[ matidx( i, j, 1, 1)] == 0 ) {
            if( param->score_rec->match_threshold <=
3285             param->score_rec->mat[ idx_seq[i] * param-
>score_rec->alph_len + idx_seq[j]]) {
                annot[i] = ')';
                annot[j] = '(';
            }
            else {
3290                 annot[i] = '.';
                annot[j] = '.';
            }
            if(0)printf( " ( ) %d %d\n", j, i );
3295             push(      stack, setpair(  i-1, j+1));
        }
        else {
            k      = path[      matidx(      i, j, 1, 1)];
            push( stack, setpair( k ,      j));
3300             push( stack, setpair( i ,      k+1));
        }
    if(0){
        if( path[ matidx( i-1, j, 1, 1)] == -1 )
            gap_score1 = param->score_rec->gap_cont;
3305     else
            gap_score1 = param->score_rec->gap_start;
        if( path[ matidx( i, j+1, 1, 1)] == -2 )
            gap_score2 = param->score_rec->gap_cont;
        else
3310             gap_score2 = param->score_rec->gap_start;

        if(      m( i-1, j, mt) ==
            m( i,      j,      mt) - gap_score1) {
            if(0)printf( " mat: -1 %d %d\n", m( i-1, j,
3315 mt), m( i, j, mt));
            push(      stack, setpair(  i-1, j));
        }
        else if( m( i,      j+1, mt) ==
            m( i,      j,      mt) - gap_score2) {
3320             if(0)printf( " mat: -2 %d %d\n", m( i-1, j,
mt), m( i, j, mt));
            push(      stack, setpair(  i, j+1));
        }
        else if( (mat[ matidx( i-1, j+1, 1, 1)] +
3325             param->score_rec->mat[ idx_seq[i] * pa-
ram->score_rec->alph_len

```

图17 (续)

```

        + idx_seq[j]))
        ==
        mat[ matidx( i,      j,      1, 1)] ) {
3330      if(0)printf( " mat:  0 %d %d\n", m( i-1, j,
mt), m( i, j, mt));
        annot[i] = ')';
        annot[j] = '(';
        if(0)printf( " ( ) %d %d\n", j,  i );
3335      push(      stack, setpair(  i-1 , j+1));
    }
    else {
        for( k=j+1; k<i; k++) {
            if(0)printf( "  %d %d  %d %d\n", i, k, k,
3340      j);
            if(      mat[ matidx( i,      k+1, 1,      1)] +
mat[ matidx( k, j, 1, 1)]      ==
            mat[ matidx( i,      j,      1, 1)] ) {
                if(0)printf( " mat:  %d %d %d\n",  k, m( i-
3345      1, j, mt), m( i, j, mt));
                push( stack, setpair( k ,      j));
                push( stack, setpair( i ,      k+1));
                break;
            }
3350      }
        }
    }
}

3355      if( outfile != NULL ) {
        fprintf( outfile, "Score= %d start= %d\n",
        mat[ matidx( d, d-depth, 1, 1)], d - depth + 1);
        outstr[0] = seq;
3360      outstr[1] = annot;
        print_lines( outfile, outstr, 2);
    }
    free( annot);
    annot      = empty_str( 1);
3365
}

    free_stack( stack);
    return annot;
3370 )

void nussinov_fill( param_type *param, int depth, int
l, int *idx_seq,
3375      int *mat, int *path, char *seq){

```

图17 (续)

```

    int d;
    int i, j, k, max_path;
    int score, max_score;
    int h_size = 3;
3380 int gap_score;
    int match_factor;

    for( d=0; d<depth; d++) {
        for( i=d+1; i<l; i++) {
3385     if(0)printf( "%d %d %d\n", d, i, depth);
            j = i - 1 - d;

            if( path[ matidx( i-1, j, 1, 1)] == -1 )
3390 gap_score = param->score_rec->gap_cont;
            else
                gap_score = param->score_rec->gap_start;

            max_score = mat[ matidx( i-1, j, 1,
3395 1)] - gap_score;
            max_path = -1;

            if( path[ matidx( i, j+1, 1, 1)] == -2 )
3400 gap_score = param->score_rec->gap_cont;
            else
                gap_score = param->score_rec->gap_start;

            score = mat[ matidx( i, j+1, 1, 1)] -
3405 gap_score;
            if( score > max_score) {
                max_score = score;
                max_path = -2;
            }

3410
            if( i-j <= h_size ) {
                score = param->score_rec->loop_score;
            }
            else {
3415 if( path[ matidx( i-1, j+1, 1, 1)] == 0 )
                match_factor = 1;
            else
                match_factor = param->score_rec-
>match_cont_factor;
3420 score = mat[ matidx( i-1, j+1, 1, 1)] +
                match_factor *
                param->score_rec->mat[ idx_seq[i] * param-
>score_rec->alph_len +
                idx_seq[j]];

```

图17 (续)

```

3425     }
        if(0)printf( "%d %d %d %d      %d %d %d\n", i, j,
idx_seq[i], idx_seq[j],
        param->score_rec->alph_len,
        param->score_rec->mat[ idx_seq[i] * param-
3430 >score_rec->alph_len +
        idx_seq[j]], idx_seq[i] * param->score_rec-
>alph_len + idx_seq[j]);

        if( score > max_score) {
3435     max_score = score;
        max_path = 0;
        }

        for( k=j+1; k<i; k++) {
3440     if( path[ matidx( i, k+1, 1, 1)] == -2 ||
        path[ matidx( k, j, 1, 1)] == -1 )
            gap_score = param->score_rec->gap_cont;
        else
            gap_score = param->score_rec->gap_start;
3445     score =      mat[ matidx( i, k+1, 1, 1)] + mat[
matidx( k, j, 1, 1)]
        - gap_score;

3450     if( score > max_score) {
        max_score = score;
        max_path = k;
        }
        if(0)printf( "%d %d %d %d %d      (%d, %d)\n", i,
3455 k, k, j, score,
            mat[ matidx( i, k+1, 1, 1)],
            mat[ matidx( k, j, 1, 1)]);
        }

3460     if( max_score > 0 ) {
        mat[ matidx( i, j, 1, 1)] = max_score;
        }
        else {
3465     mat[ matidx( i, j, 1, 1)] = 0;
        }
        path[ matidx( i, j, 1, 1)] = max_path;
        if(0)printf("%c%c %3d %3d      score: %3d path:
%3d\n", seq[i], seq[j], i, j, max_score, max_path);
    }
3470 }
}

```

图17 (续)

```
int nussinov( FILE *outfile, param_type *param, char
3475 *seq){
    int l, i;
    int *mat, *path;
    int *idx_seq;
    int score = INT_MAX;
3480 int max_score = 0;
    int depth = param->depth;

    l = strlen( seq);
3485 if( depth > l-1 ) { depth = l-1;}
    mat = calloc( matsize( l, l), sizeof( int));
    path = calloc( matsize( l, l), sizeof( int));

3490 idx_seq = seq2idx_seq( seq, param->score_rec);
    nussinov_fill( param, depth, l, idx_seq, mat, path,
    seq);
    if( 0) {
        print_mat( seq, seq, mat);
3495 printf( "\n");
        print_mat( seq, seq, path);
        printf( "\n");
    }

3500 if( param->global) {
        nussinov_traceback( outfile, param, depth, l,
    idx_seq, seq, mat, path);
    }
    else {
3505 i = 0;
        while( score > param->min_score && i < param-
>max_res) {
            score = nussinov_local_traceback( outfile, pa-
ram, depth, l,
3510 idx_seq, seq, mat, path);
            i++;
            if( score > max_score) max_score = score;
        }
3515 }

    free( mat);
    free( path);
3520 free( idx_seq);
    return max_score;
}
```

图17 (续)

```

3525  /***** OLIGO *****/
void calc_olig( int id, char *seq, char *rev_seq,
               int *self, int *target, param_type *param, int
3530  p){
    int score_self, score_target, score_struct;

    strcpy( rev_seq, seq);
    reverse( rev_seq);
3535  score_self = align( NULL, param, seq, rev_seq);
    lower( rev_seq);
    comp( rev_seq);
    reverse( rev_seq);
    score_target = align( NULL, param, seq, rev_seq);
3540  if( p) {
        if(0) {
            score_struct = nussinov( NULL, param, seq);
            printf( "%7d %s %7d %7d %7d\n", id, seq,
3545  score_self, score_target,
                score_struct);
        }
        else {
            if( score_target > score_self + 5) {
                printf( "%d %s %d %d\n", id, seq, score_self,
3550  score_target);
            }
        }
    }
    *self = score_self;
3555  *target = score_target;
}

long spikenum( int n){
3560  int i;
    long n_tmp;
    long mask = 1;
    long res = 0;
    int nbit = 14; /* 16384 different spikings in 32
3565  bits */

    /*
        To get all spike variants of a given
    */
3570  for( i=0;i<nbit;i++){

```

图17 (续)

```
        n_tmp = n & mask;
        n_tmp = n_tmp << (i+i+2);
        res += n_tmp;
3575     mask = mask << 1;
    }

    return res;
}
3580

long lna2spikenum( long n){
    int i;
    long n_tmp;
3585     long mask = 4;
    long res = 0;
    int nbit = 14; /* 16384 different spikings in 32
bits */

3590 /*
    To get all spike variants of a given
    */

    for( i=0;i<nbit;i++){
3595         n_tmp = n & mask;
        n_tmp = n_tmp >> (i+i+2);
        res += n_tmp;
        mask = mask << 3;
    }

3600     return res;
}

3605 long reverse_dna( long n, int oligolen){
    int i;
    int nbit = oligolen * 2;
    long mask = 3;
    long n_tmp;
3610     long res = 0;

    for( i=0;i<nbit/2;i++){
        n_tmp = n & mask;
        n_tmp = n_tmp >> i*2;
3615         n_tmp = n_tmp << (nbit - (i+1)*2);
        res += n_tmp;
        mask = mask << 2;
    }
    return res;
3620 }
```

图17 (续)

```
long comp_dna( long n, int oligolen) {
    long    res;
3625    long    mask = (n_nmer( oligolen, "acgt") - 1);

    res = ~n & mask;
    return res;
3630 }

long rev_comp_dna( long n, int oligolen){
    long    res;
    res = reverse_dna( n, oligolen);
3635    res = comp_dna( res, oligolen);
}

long dna2lna( int n){
3640    int i;
    long n_tmp;
    long mask = 3;
    long res = 0;
    const int  nbit = 32;
3645
    /*
        To get all spike variants of a given
    */

3650    for( i=0;i<nbit/2;i++){
        n_tmp  = n & mask;
        n_tmp  = n_tmp << i;
        res    += n_tmp;
        mask   = mask << 2;
3655    }

    return res;
}

3660
long lna2dna( int n){
    int i;
    long n_tmp;
    long mask = 3;
3665    long res = 0;
    const int  nbit = 32;

    /*
        To get the dna only sequence
```

图17 (续)

```
3670  */
        for( i=0;i<nbit/2;i++){
            n_tmp = n & mask;
            n_tmp = n_tmp >> i;
3675     res += n_tmp;
            mask = mask << 3;
        }

        return res;
3680  }

long dna2lna_spike( int nmer, int spike_pat) {
    long res;
3685     res = dna2lna( nmer);
    res += spikenum( spike_pat);
    return res;
}

3690

void allolig( FILE *outfile, int oligolen, int
oligosample, int spikefreq,
    char *n_mer_range, param_type *param){
3695     char alph[255] = "acgtACGT";
    long n, nmer;
    char *seq, *rev_seq;
    long i, j;
    long n0 = 0, n1;
3700     int spike_0 = 0, spike_1, spike_n;
    int dna_flag = 0;
    int self_s, target_s;

    check_endian();
3705     check_limits();

    if( n_mer_range != NULL ) {
        if( sscanf( n_mer_range, "%d-%d:%d-%d", &n0, &n1,
&spike_0, &spike_1) == 4){
3710             dna_flag = 1;
            n = n1 + 1;
            spike_n = spike_1 - spike_0 + 1;
        }
        else if( sscanf( n_mer_range, "%d-%d", &n0, &n1)
3715 == 2){
            n = n1 + 1;
            spike_n = n_mer( oligolen, "aA");
        }
    }
```

图17 (续)

```
    else {
3720      die( "Failed to parse %s, expected %d-%d or
        %d-%d:%d-%d",
          n_mer_range);
    }
  }
3725  else {
    n      = n_nmer( oligolen, alph);
    spike_n = n_nmer( oligolen, "aA");
  }

3730  rev_seq = empty_str( oligolen);

    param->min_score = 2;
    param->max_res   = 1;

3735  if( oligosample ) {
    for( j=0; j<oligosample; j++) {
      seq = make_random_nmer( oligolen, spikefreq,
        alph);
      calc_olig( j, seq, rev_seq, &self_s, &target_s,
3740  param, 1);
      free( seq);
    }
  }
  else {
3745    if( dna_flag) {
      for( i=n0; i<n; i++) {
        for( j=spike_0; j<spike_n; j++) {
          nmer = dna2lna_spike( i, j);
          seq = make_nmer( nmer, oligolen, alph);
3750          calc_olig( nmer, seq, rev_seq, &self_s,
            &target_s, param, 1);
          free( seq);
        }
      }
3755    }
    else {
      for( j=n0; j<n; j++) {
        seq = make_nmer( j, oligolen, alph);
        calc_olig( j, seq, rev_seq, &self_s, &target_s,
3760  param, 1);
        free( seq);
      }
    }
  }
3765  free( rev_seq);
}
```

图17 (续)

```

/***** CLUSTER *****/
3770
int cluster( FILE *outfile, int score_cutoff, int
n_okseq, param_type *param){
    int i;
3775    int *seq_id2cluster, *cluster2seq_id;
    int seq_id = 0, next_seq_id = 0, cluster_id = 0;
    int score;
    int nseq;
    int seq_id_1;
3780    sequences_type *seqs;

    param->max_res = 1;
    param->min_score = 1;

3785    nseq          = param->seqs->nseq;
    seq_id2cluster = calloc( nseq, sizeof( int));
    cluster2seq_id = calloc( nseq, sizeof( int));

3790    for( i=0; i< nseq; i++)
        seq_id2cluster[i] = -1;

    do{
        seq_id          = next_seq_id;
3795        cluster2seq_id[ cluster_id] = seq_id;
        seq_id2cluster[ seq_id]      = cluster_id;
        seq_id_1 = seq_id + 1;

        if( seq_id_1 < n_okseq )
3800            next_seq_id = seq_id_1;
        else
            next_seq_id      = 0;

        printf( "%d\n", seq_id);
3805        for( i=seq_id_1; i<nseq; i++){
            if( seq_id2cluster[ i] < 0 ) {

                score = align( NULL, param,
3810                    param->seqs->seqs[ seq_id], param->seqs-
>seqs[i]);

                if( score >= score_cutoff) {
                    seq_id2cluster[ i] = cluster_id;
3815                }
                else if( next_seq_id == 0) {

```

图17 (续)

```

        next_seq_id = i;
    }

3820     }
        }

        cluster_id++;
    } while (next_seq_id > 0);

3825

    seqs = new_sequences_type( SEQS_CHUNK);
    for( i=0; i<cluster_id; i++) {
        add_seq2seqs( param->seqs->seqs[i], seqs);
3830     }
    print_fasta( outfile, seqs);
    free_sequences_type( seqs);

3835     free( cluster2seq_id);
    free( seq_id2cluster);
    return cluster_id;
}

3840     /***** ARGUMENT PARSING AND INITIALIZATION
    *****/

3845     param_type *init(){
        char          a[255];
        int  i, j;
        param_type *param;
        int  *mat;
3850     #define          ALPHLEN          9
        int  mathyp_init[ALPHLEN][ALPHLEN]= {
            { -3, -3, -3,  3, -5, -5, -5,  5, -
3855     99},
            { -3, -3,  5, -3, -5, -5,  9, -5, -
3855     99},
            { -3,  5, -3,  1, -5,  9, -5,  2, -
3855     99},
            {  3, -3,  1, -3,  5, -5,  2, -5, -
3860     99},
            { -5, -5, -5,  5, -8, -8, -8,  8, -
3860     99},
            { -5, -5,  9, -5, -8, -8, 14, -8, -
3860     99},
            { -5,  9, -5,  2, -8, 14, -8,  4, -
3865     99},

```

图17 (续)

```

        { 5, -5, 2, -5, 8, -8, 4, -8, -
99},
        {-99,-99,-99,-99,-99,-99,-99,-99, -
99}
3870     };
    /*
        int mathyp_init[ALPHLEN][ALPHLEN]= {
            { -2, -2, -2, 3, -2, -2, -2,
3875         { -2, -2, 5, -2, -2, -2, 9,
            -2, -99},
            { -2, 5, -2, 1, -2, 9, -2,
            2, -99},
            { 3, -2, 1, -2, 5, -2, 2,
3880     {-2, -99},
            { -2, -2, -2, 5, -2, -2, -2,
            8, -99},
            { -2, -2, 9, -2, -2, -2, 14,
            -2, -99},
3885     { -2, 9, -2, 2, -2, 14, -2,
            4, -99},
            { 5, -2, 2, -2, 8, -2, 4,
            -2, -99},
            {-99,-99,-99,-99,-99,-99,-99,-99, -
3890     99}
        };
    */

    param      = calloc( 1, sizeof( param_type));
3895
    /* alph */
    strcpy( a, "acgtACGT-");
    param->alph      = calloc( strlen( a) + 1,
sizeof( char));
3900     strcpy( param->alph, a);

    /* seqs */
    param->seqs      = new_sequences_type(
SEQS_CHUNK);
3905     param->seqs2      = new_sequences_type(
SEQS_CHUNK);

    /* hybridization matrix */

3910     mat = calloc( ALPHLEN * ALPHLEN, sizeof( int));
    for( i=0;i<ALPHLEN;i++){
        for( j=0;j<ALPHLEN;j++){
            mat[i*ALPHLEN+j] = mathyp_init[i][j];
        }
    }

```

图17 (续)

```

3915     }

        /* defaults */
        param->depth      = INT_MAX;
        param->global      = 0;
3920     param->min_score = 20;
        param->max_res     = INT_MAX;

        param->score_rec = init_score_rec_type(
3925         14, 7, 3,
         -40, 2, 1,
         8, 3, 1,
         param->alph, mat);

3930     validate_score_rec_type( param->score_rec, -200,
        200);

        return param;
    }
3935

#ifdef LIB
/***** MAIN *****/

3940 int main (int argc, char* argv[]) {
    int      i;
    char      *rev_seq;
    param_type *param;
3945     int      algo      = 0;

    char      *options;
    int      next_option;
    char*      outfn      = NULL;
3950     char*      infn      = NULL;
    FILE      *outfile, *seqfile;
    int      othermethod  = 0;
    int      oligolen     = 7;
    int      oligosample  = 0;
3955     int      spikefreq  = 0;
    int      score_cutoff = 100;
    int      n_okseq      = 1;
    char*      n_mer_range = NULL;

3960     const char* const short_options =
        "hvruiyto:i:j:s:z:d:gn:p:m:l:e:f:c:k:w:";
    const struct option long_options[] = {
        { "help", 0, NULL, 'h' },

```

图17 (续)

```

3965     { "verbose",          0,      NULL, 'v' },
        { "version",         0,      NULL, 'r' },
        { "secondary_structure", 0,      NULL, 'u' },
        { "self_anealing",    0,      NULL, 'a' },
        { "hybridization",    0,      NULL, 'y' },
        { "tm",               0,      NULL, 't' },
3970     { "output",           1,      NULL, 'o' },
        { "input",            1,      NULL, 'i' },
        { "input2",           1,      NULL, 'j' },
        { "seq",              1,      NULL, 's' },
        { "seq2",             1,      NULL, 'z' },
3975     { "depth",           1,      NULL, 'd' },
        { "global",          0,      NULL, 'g' },
        { "min_score",        1,      NULL, 'n' },
        { "max_res",          1,      NULL, 'p' },
        { "matrix",           1,      NULL, 'm' },
3980     { "allolig",          1,      NULL, 'l' },
        { "n_mer_range",      1,      NULL, 'w' },
        { "cluster",          1,      NULL, 'c' },
        { "n_okseq",          1,      NULL, 'k' },
        { "sample",           1,      NULL, 'e' },
3985     { "spikefreq",        1,      NULL, 'f' },
        { NULL,               0,      NULL, 0 } /* Required
at end of array. */
    };

3990     /* mtrace(); debug memory leaks under linux */

    param = init();

    program_name = argv[0];
3995
    do {
        next_option = getopt_long (argc, argv,
short_options,
        long_options, NULL);
4000    switch (next_option)
        {
            case 'h': /* -h or --help */
                usage ("");

4005            case 'i': /* -i or --input */
                infn = optarg;
                seqfile = sfopen( infn, "r");
                read_fasta( seqfile, param->seqs, param->alph);
                fclose( seqfile);
4010            break;

            case 'j': /* -j or --input2 */

```

图17 (续)

```

    infn = optarg;
    seqfile = sfopen(    infn, "r");
4015    read_fasta( seqfile, param->seqs2, param->alph);
    fclose( seqfile);
    break;

    case 'o':    /* -o or --output */
4020        outfn = optarg;
        break;

    case 'm':    /* -m or --matrix*/
        options =  optarg;
4025        read_score_rec_type( options, param->score_rec);
        break;

    case 's':    /* -s or --seq*/
        options =  optarg;
4030        add_seq2seqs( options, param->seqs);
        break;

    case 'z':    /* -z or --seq2*/
        options =  optarg;
4035        add_seq2seqs( options, param->seqs2);
        break;

    case 'd':    /* -d or --depth*/
        options =  optarg;
4040        param->depth = atoi( options);
        break;

    case 'g':    /* -d or --depth*/
        param->global = 1;
4045        break;

    case 'p':    /* -p or --max_res*/
        options =  optarg;
        param->max_res = atoi( options);
4050        break;

    case 'n':    /* -n or --min_score*/
        options =  optarg;
        param->min_score = atoi( options);
4055        break;

    case 'v':    /* -v or --verbose */
        verbose += 1;
        break;
4060

    case 'r':    /* -r or --version */
```

图17 (续)

```
        printf( "dyp version %s\n", THEVERSION);
        exit( 0);
        break;
4065     case 'u':      /* -s or --secondary_structure */
        algo = algo | SECONDARY_STRUCTURE;
        break;

4070     case 'a':      /* -s or --self_anealign */
        algo = algo | SELF_ANEALING;
        break;

        case 'y':      /* -s or --self_anealign */
4075     algo = algo | HYBRIDIZATION;
        break;

        case 't':      /* -t or --tm */
4080     algo = algo | TM;
        break;

        case 'l':      /* -l or --allolig */
        options = optarg;
        oligolen = atoi( options);
4085     othermethod = ALLOLIG;
        break;

        case 'w':      /* -w or --n_mer_range */
4090     n_mer_range = optarg;
        break;

        case 'c':      /* -c or --cluster */
        options = optarg;
        score_cutoff = atoi( options);
4095     othermethod = CLUSTER;
        break;

        case 'k':      /* -k or --n_okseq */
        options = optarg;
4100     n_okseq = atoi( options);
        break;

        case 'e':      /* -p or --sample */
        options = optarg;
4105     oligosample = atoi( options);
        break;

        case 'f':      /* -f or --spikefreq */
        options = optarg;
4110     spikefreq = atoi( options);
```

图17 (续)

```

        break;

        case '?':      /* The user specified an invalid op-
tion. */
4115      usage ("Sorry, one of the options was not
        recognised");

        case -1:      /* Done with options. */
        break;
4120      default:      /* Something else: unexpected. */
        abort ();
    }
}
4125 while (next_option != -1);

if( outfn == NULL )
    outfile = stdout;
4130 else
    outfile = sfopen( outfn, "w");

if( verbose)
    print_score_rec_type( param->score_rec);
4135

if( othermethod == ALLOLIG) {
    allolig( outfile, oligolen, oligosample, spikefreq,
n_mer_range, param);
4140 }
    else if( othermethod == CLUSTER) {
        cluster( outfile, score_cutoff, n_okseq, param);
    }
    else {
4145        if( !algo )
            usage( "please select a method: u, a or y");

        for( i=0; i<param->seqs->nseq; i++) {
            fprintf( outfile, ">sequence_%03d\n", i+1);
4150            if( algo & SECONDARY_STRUCTURE )
                nussinov( outfile, param, param->seqs->seqs[i]);
            if( algo & SELF_ANEALING ) {
                rev_seq = calloc( strlen( param->seqs-
>seqs[i]) + 1, sizeof( char));
4155                strcpy( rev_seq, param->seqs->seqs[i]);
                reverse( rev_seq);
                if( param->min_score < 2 ) {
                    param->min_score = 2;
                }
            }
        }
    }
}

```

图17 (续)

```
4160     align( outfile, param, param->seqs->seqs[i],
rev_seq);
        free( rev_seq);
        }
        if( algo & TM ) {
4165     rev_seq      = calloc( strlen( param->seqs-
>seqs[i]) + 1, sizeof( char));
        strcpy(      rev_seq, param->seqs->seqs[i]);
        lower( rev_seq);
        comp( rev_seq);
4170     if( param->min_score < 2 ) {
        param->min_score = 2;
        }
        align( outfile, param, param->seqs->seqs[i],
rev_seq);
4175     free( rev_seq);
        }
        if( algo & HYBRIDIZATION ) {
            if( param->seqs2->seqs[i] == NULL)
                die( "Sequence 2 is missing");
4180     rev_seq      = calloc( strlen( param->seqs2-
>seqs[i]) + 1, sizeof( char));
        strcpy(      rev_seq, param->seqs2->seqs[i]);
        reverse( rev_seq);
        align( outfile, param, param->seqs->seqs[i],
4185 rev_seq);
        free( rev_seq);
        }
        }
    }
4190     fclose( outfile);

        free_param_type( param);

        return 0;
4195 }
#endif
```

图17 (续)