**L1谱系的进化分析**

**实验目标：**本环节的关键在于根据L1各个亚家族的序列信息，构建共识序列。后根据各个亚家族间共识序列的变异，从碱基的数目，类型以及突变所在的位置等来分析亚家族间的差异。

1. **亚家族的分类**
2. 明确L1的亚家族的数目

1.cat "/home/xxzhang/workplace/project/CRISPRa/basic/allTE/index/hg38\_bedtools\_TE.chr1-Y.processed.final.gtf" |grep "L1:" |awk '{print $4}' |sort |uniq -c >"/home/xxzhang/Resource/Reference/GTF/TE\_gtf/L1.subfamily.count.txt"

在整个L1家族中，共有123个亚家族，其中L1PA谱系为灵长类特异性的谱系。

1. 提取亚家族的序列

Full-length elements belonging to the L1 families that amplified since before the origin of primates (families L1MA5–L1MA1, L1PA17–L1PA1, and L1PB4–L1PB1) were collected by searching table RepeatMasker (assembly of April 2003) at http:// genome.ucsc.edu for all L1 elements **longer than 6 Kb**1.

我们根据上述L1的注释信息，提取L1PA谱系各个亚家族中，长度≥6kb的成员的序列。

1. cat "/home/xxzhang/workplace/project/CRISPRa/basic/allTE/index/hg38\_bedtools\_TE.chr1-Y.processed.final.fasta" |grep "L1:L1PA7:>6k:" -A 1 |grep -v ^-- >/home/xxzhang/workplace/project/CRISPRa/basic/bed/6kb\_fasta/fasta/align/align\_L1PA7\_6kb.fasta

将我们要提取的家族信息保存在文件中，批量运行。

1.cat L1PA.txt |parallel -j 4 '

2.input\_file="/home/xxzhang/workplace/project/CRISPRa/basic/allTE/index/hg38\_bedtools\_TE.chr1-Y.processed.final.fasta";

3.output\_file="/home/xxzhang/workplace/project/CRISPRa/basic/bed/6kb\_fasta/fasta/{}\_6kb.fasta";

4. grep "L1:{}:>6k:" -A 1 "$input\_file" | grep -v "^--" >$output\_file

5.'

上述代码完成提取各个家族的6kb的序列。

1. **构建共识序列**
2. 多序列比对

使用mafft对上述序列进行多序列比对。

1. #PBS -N mafft

2. #PBS -q fat

3. #PBS -l nodes=1:ppn=4

4. #PBS -l mem=100gb

5. #PBS -M 2456392738@qq.com

6. #PBS -m abe

7. cd /home/xxzhang/workplace/project/CRISPRa/basic/bed/6kb\_fasta/fasta/

8. source activate genomics

9. cat L1PA.txt |parallel -j 8 'mafft --auto --thread 8 {}\_6kb.fasta >./align/align\_{}\_6kb.fasta'

1. 提取共识序列

1. import argparse

2. from Bio import AlignIO

3. from Bio.SeqRecord import SeqRecord

4. from Bio.Seq import Seq

5. from collections import Counter

6.

7. # IUPAC 混合碱基符号（支持 GAP）

8. IUPAC\_CODES = {

9. frozenset(['A']): 'A',

10. frozenset(['C']): 'C',

11. frozenset(['G']): 'G',

12. frozenset(['T']): 'T',

13. frozenset(['-']): '-', # GAP

14. frozenset(['A', 'G']): 'R',

15. frozenset(['C', 'T']): 'Y',

16. frozenset(['G', 'C']): 'S',

17. frozenset(['A', 'T']): 'W',

18. frozenset(['G', 'T']): 'K',

19. frozenset(['A', 'C']): 'M',

20. frozenset(['C', 'G', 'T']): 'B',

21. frozenset(['A', 'G', 'T']): 'D',

22. frozenset(['A', 'C', 'T']): 'H',

23. frozenset(['A', 'C', 'G']): 'V',

24. frozenset(['A', 'C', 'G', 'T']): 'N',

25. }

26.

27. # # 从比重生成 IUPAC 符号

28. # def get\_iupac\_from\_frequencies(freqs):

29. # # 按比重降序排列碱基

30. # sorted\_bases = sorted(freqs.items(), key=lambda x: x[1], reverse=True)

31. # max\_freq = sorted\_bases[0][1]

32. # selected\_bases = {base for base, freq in sorted\_bases if freq == max\_freq}

33.

34. # # 如果 gap 存在但不是唯一最高频率，排除 gap

35. # if '-' in selected\_bases and len(selected\_bases) > 1:

36. # selected\_bases.remove('-')

37.

38. # # 打印调试信息：查看当前位点的选中碱基和最大比重

39. # print(f"DEBUG: Selected bases: {selected\_bases}, Frequencies: {freqs}")

40.

41. # # 返回对应的 IUPAC 符号

42. # iupac = IUPAC\_CODES.get(frozenset(selected\_bases), 'N') # 默认返回 N 表示不确定

43. # print(f"DEBUG: Returning IUPAC: {iupac}")

44. # return iupac

45.

46. # # 计算共识序列和每个位点比重

47. # def get\_frequencies\_and\_consensus(alignment):

48. # frequencies = []

49. # consensus\_sequence = []

50. # for i in range(alignment.get\_alignment\_length()):

51. # column = alignment[:, i]

52. # counts = Counter(column)

53. # total = sum(counts.values())

54. # freqs = {base: round(count / total, 3) for base, count in counts.items()}

55. # frequencies.append(freqs)

56.

57. # consensus\_base = get\_iupac\_from\_frequencies(freqs)

58. # consensus\_sequence.append(consensus\_base)

59.

60. # # 调试输出每个位点的共识碱基

61. # print(f"DEBUG: Position {i+1}, Frequencies: {freqs}, Consensus Base: {consensus\_base}")

62.

63. # consensus\_str = "".join(consensus\_sequence)

64. # print(f"DEBUG: Final consensus sequence: {consensus\_str}")

65. # return frequencies, consensus\_str

66.

67. # # 从比重生成 IUPAC 符号

68. # def get\_iupac\_from\_frequencies(freqs):

69. # sorted\_bases = sorted(freqs.items(), key=lambda x: x[1], reverse=True)

70. # max\_freq = sorted\_bases[0][1]

71. # selected\_bases = {base for base, freq in sorted\_bases if freq == max\_freq}

72.

73. # # 如果 GAP 存在但不是唯一最高频率，排除 GAP

74. # if '-' in selected\_bases and len(selected\_bases) > 1:

75. # selected\_bases.remove('-')

76.

77. # # 如果只有一个碱基具有最高频率，直接返回

78. # if len(selected\_bases) == 1:

79. # return next(iter(selected\_bases))

80.

81. # # 返回对应的 IUPAC 符号

82. # iupac = IUPAC\_CODES.get(frozenset(selected\_bases), 'N')

83. # return iupac

84. # 从比重生成 IUPAC 符号

85. def get\_iupac\_from\_frequencies(freqs):

86. sorted\_bases = sorted(freqs.items(), key=lambda x: x[1], reverse=True)

87. max\_freq = sorted\_bases[0][1]

88. selected\_bases = {base.upper() for base, freq in sorted\_bases if freq == max\_freq}

89.

90. # 如果 GAP 存在但不是唯一最高频率，排除 GAP

91. if '-' in selected\_bases and len(selected\_bases) > 1:

92. selected\_bases.remove('-')

93.

94. # 如果只有一个碱基具有最高频率，直接返回

95. if len(selected\_bases) == 1:

96. return next(iter(selected\_bases))

97.

98. # 返回对应的 IUPAC 符号

99. iupac = IUPAC\_CODES.get(frozenset(selected\_bases), 'N')

100. return iupac.upper()

101.

102.

103. # 核心函数：计算比重和共识序列

104. def get\_frequencies\_and\_consensus(alignment):

105. frequencies = []

106. consensus\_sequence = []

107. for i in range(alignment.get\_alignment\_length()):

108. column = alignment[:, i]

109. counts = Counter(column)

110. total = sum(counts.values())

111. freqs = {base: round(count / total, 3) for base, count in counts.items()}

112. frequencies.append(freqs)

113.

114. consensus\_base = get\_iupac\_from\_frequencies(freqs)

115. consensus\_sequence.append(consensus\_base)

116. return frequencies, "".join(consensus\_sequence)

117.

118.

119. # 将共识序列写入 FASTA 文件

120. def write\_consensus\_to\_fasta(consensus\_sequence, output\_file):

121. with open(output\_file, 'w') as f:

122. f.write(f">Consensus\_Sequence\n{consensus\_sequence}\n")

123. print(f"DEBUG: Writing consensus sequence to FASTA file: {consensus\_sequence}")

124.

125. # 清理 FASTA 文件中的换行符，整理成连续序列

126. def clean\_fasta\_sequences(input\_file, cleaned\_file):

127. records = []

128. with open(input\_file, 'r') as infile:

129. lines = infile.readlines()

130. current\_id = None

131. current\_seq = []

132. for line in lines:

133. line = line.strip()

134. if line.startswith(">"):

135. # 使用正则去除标题行中的非标准字符（如双引号）

136. line = line.replace('"', '').replace('>', '')

137. if current\_id:

138. records.append(SeqRecord(Seq("".join(current\_seq)), id=current\_id))

139. current\_id = line # 更新基因ID

140. current\_seq = []

141. else:

142. current\_seq.append(line)

143. if current\_id: # 添加最后一个序列

144. records.append(SeqRecord(Seq("".join(current\_seq)), id=current\_id))

145.

146. # 写入清理后的文件

147. with open(cleaned\_file, 'w') as outfile:

148. for record in records:

149. outfile.write(f">{record.id}\n{record.seq}\n")

150.

151. return cleaned\_file

152.

153. # 将频率写入文本文件

154. def write\_frequencies\_to\_txt(frequencies, output\_file):

155. with open(output\_file, 'w') as f:

156. for i, freq in enumerate(frequencies):

157. f.write(f"Position {i+1}: {freq}\n")

158. print(f"DEBUG: Writing frequencies for position {i+1}: {freq}")

159.

160. # 主函数

161. def main():

162. # 设置命令行参数解析

163. parser = argparse.ArgumentParser(description="计算多序列比对的共识序列和每个位点的比重，并输出到文件")

164. parser.add\_argument(

165. "input\_file",

166. type=str,

167. help="输入的多序列比对文件（FASTA 格式）"

168. )

169. parser.add\_argument(

170. "output\_prefix",

171. type=str,

172. help="输出文件的前缀"

173. )

174. args = parser.parse\_args()

175.

176. # 清理输入文件

177. cleaned\_file = f"{args.output\_prefix}\_cleaned.fasta"

178. cleaned\_file = clean\_fasta\_sequences(args.input\_file, cleaned\_file)

179.

180. # 读取比对文件

181. try:

182. alignment = AlignIO.read(cleaned\_file, "fasta")

183. except FileNotFoundError:

184. print(f"错误: 文件 '{args.input\_file}' 不存在。")

185. return

186. except Exception as e:

187. print(f"错误: 无法读取文件 '{args.input\_file}'。请确保文件格式正确。")

188. print(f"详细错误: {e}")

189. return

190.

191. # 计算比重和共识序列

192. frequencies, consensus\_sequence = get\_frequencies\_and\_consensus(alignment)

193.

194. # 输出文件名

195. fasta\_file = f"{args.output\_prefix}\_consensus.fasta"

196. txt\_file = f"{args.output\_prefix}\_frequencies.txt"

197.

198. # 写入文件

199. write\_consensus\_to\_fasta(consensus\_sequence, fasta\_file)

200. write\_frequencies\_to\_txt(frequencies, txt\_file)

201.

202. print(f"共识序列已保存到 {fasta\_file}")

203. print(f"位点比重已保存到 {txt\_file}")

204.

205. # 调用主函数

206. if \_\_name\_\_ == "\_\_main\_\_":

207. main()

208.

运行上述代码：

1. python consensusFretch.py align\_L1PA17\_6kb.fasta L1PA17

得到所有的共识序列。

1. 合并共识序列

手动粘贴到同一个文档，修改序列名。

1. 多序列比对共识序列。

使用clustalw比对上述得到的共识序列，下载比对得到的clustalw.aln文件。

将.aln文件相同行名的行合并。

1. from collections import defaultdict

2.

3. def merge\_lines(filename):

4. # 使用defaultdict来存储每个键对应的所有行

5. merged\_lines = defaultdict(str)

6.

7. with open(filename, 'r') as file:

8. for line in file:

9. # 跳过空行

10. if line.strip() == '':

11. continue

12. # 分割行名和内容

13. parts = line.split(maxsplit=1)

14. if len(parts) < 2:

15. continue # 跳过不符合格式的行

16. name, content = parts

17. merged\_lines[name] += content.strip() # 拼接内容

18.

19. # 将结果写入新文件或打印

20. with open('merged\_output.txt', 'w') as outfile:

21. for name, content in merged\_lines.items():

22. outfile.write(f"{name:<15} {content}\n")

23.

24. print("合并完成，结果已保存到 L1PA1-17\_align\_consensus.txt")

25.

26. # 使用示例

27. merge\_lines('clustalw.aln')

28.

L1PA1-17\_align\_consensus.txt即为最后的比对结果。

通过查看L1PA谱系的共识序列，发现L1PA1-6跟其它家族在序列上差距挺大的。

1. **分析变异信息**
2. 中值联结网络

（这个暂时找不到方法了。之前似乎参考的是一篇博客，那篇博客来自科学网，但是近日科学网关闭了）

但是从目前的各个家族的序列信息来看，到后面家族间的序列就差距很大了，并且序列之间的变异会往复。

本来想通过中值联结网络生成碱基变异的关系图，但是现在似乎这个方法没法用了。

我们现在的当务之急仍然是设计crRNA。

We found that L1s expressed in the adult human brain primarily belonged to primate-specific families, including both hominoid-specific (L1PA2 to L1PA4) and human-specific elements (L1HS)2.

1. Khan, H., Smit, A., and Boissinot, S. (2006). Molecular evolution and tempo of amplification of human LINE-1 retrotransposons since the origin of primates. Genome Res *16*, 78-87. 10.1101/gr.4001406.

2. Garza, R., Atacho, D.A.M., Adami, A., Gerdes, P., Vinod, M., Hsieh, P., Karlsson, O., Horvath, V., Johansson, P.A., Pandiloski, N., et al. (2023). LINE-1 retrotransposons drive human neuronal transcriptome complexity and functional diversification. Sci Adv *9*, eadh9543. 10.1126/sciadv.adh9543.