实验 2.2 序列比对分析

一、实验背景

要分析 DNA 序列当中 A、T、C、G 碱基字符与我们独特的个体特征有什么关系，首先需要解答：序列从哪里来？即，该序列位于人类基因组上的什么位置。序列比对是指，将两个或多个序列排列在一起，以确定相似字符或子字符串之间的对应关系，由此判断序列之间的相似性和一致性。该方法在脱氧核糖核酸（DNA）、核糖核酸（RNA）和蛋白质序列分析当中有广泛应用。在本实验中，我们通过序列比对，确定测序片段在参考序列上的位置与来源。

二、教学目标

本节实验课程，我们将依次介绍人类参考基因组、序列比对分析方法和分析工具，及比对结果的图形化展示与解读。在操作过程中，我们需要把测序数据与参考序列进行比对，获得这些测序 reads 在参考基因组上的位置信息。

三、实验原理

第一部分 序列比对问题模型：

来自不同生物体的序列有不同程度的差异，要把它们排列对齐就需要在序列比对不齐的位置插入“空格”或“间隙”，以使其具有相同的长度。

举个例子，假设我们有以下两个 DNA 序列 GACGGATTAG 和 GATCGGAATAG。这两个序列之间一个可能的排列如下：

```
GA-CGGATTAG
||  |||| |||
GATCGGAATAG
```

在这种对齐方式下，第一条序列上有一个碱基 T 在第二条序列上变成了碱基 A；第二条序列中多了一个 T，通过在第一条序列中添加一个“空格”，使两条序列重新对齐。

如何评价序列比对结果？换句话讲，如何量化两条被比较序列的相似程度？这里引入一个概念，序列一致性，其数值代表进行序列比对时，所选序列当中，相同位点的碱基完全一致的数量占总碱基数的比例。序列一致性，最高 100%，表示序列完全一致。在上面的例子当中，两条序列的一致性为 9/11*100%=81.82%

序列比对的常见类型：

根据比对范围不同，有两种类型的对齐方式：

1. 全局比对：遍历全长，将目标序列与参考序列进行比对。全局比对是一个全局性的优化过程，跨越了两个查询序列的全长。

2. 局部比对：将目标序列的一个子字符串与参考序列的一个子字符串进行比对，即找到两个序列之间高相似度的局部区域。

根据参与比对序列的数量，也有两种不同的比对方式：

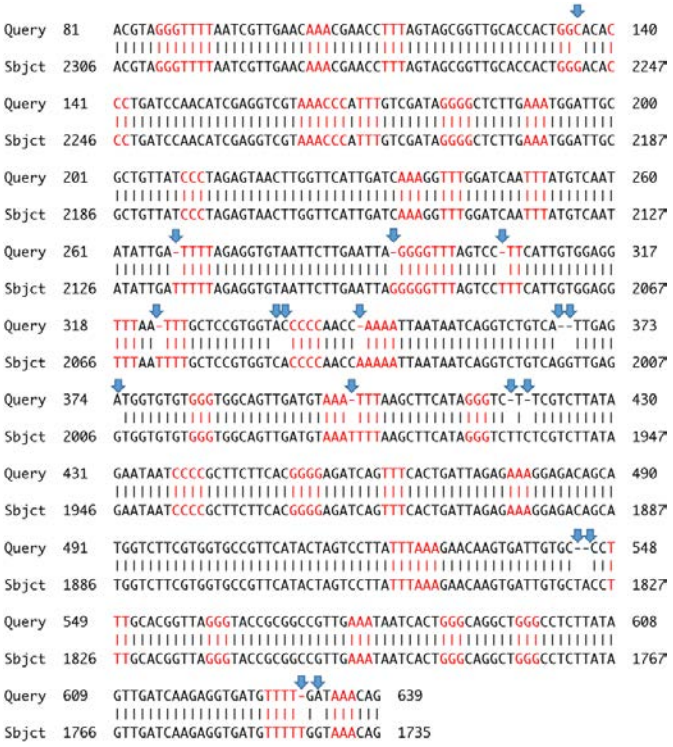
1. 成对比对：涉及两个序列，即参考序列和与之比对的目标序列。

2. 多重比对：涉及到两个以上的序列的比对。

第二部分 序列比对核心问题与算法

寻找最长公共子序列（LCS，Longest Common Subsequence）

当参与比对的序列足够长的时候，比对情况会变复杂，一条序列可能同时比对上多个不同的区域，如何找到最佳匹配情况？举个例子：如抑癌基因 p53，全长超过 25kb，比对情况复杂，如何进行比对评价？



我们可以通过寻找最长公共子串或最长公共子序列的方法来解决上述问题。两条序列的最长公共子序列，就是指这两条序列共有的子序列，且长度最长。

以下述两条序列为例：

GACGGATTAG

GATCGGAATAG

这两条序列的 LCS 是 CGGA。当然，我们很容易找到其他长度较短的子序列，

如 CGG，TAG，但其中最长的是 CGGA。一般情况下，参与比对的序列足够长时，可能会出现多个 LCS，但此处 LCS 唯一，长度为 4。

LCS 非常有助于计算两个序列的相似程度：LCS 越长，序列相似度越高。

序列比对算法：

如何设计算法，快速、精准地完成长序列、多序列等复杂比对计算？

我们将被比对序列之间可能的位点关系进行归纳，可以分为以下 3 类：

1）匹配（|）

2）替换/错配（*）

3）空位（插入/缺失 -）

现在，我们引入另外一个概念，相似性。

相似性的意义与一致性类似，其数值也是用来衡量序列的相似程度。在对 DNA 序列进行比对时，我们可以采用以下三种不同的核酸序列替换矩阵（计分矩阵）对序列的相似性进行打分：

1） 等价替换矩阵：相同核苷酸之间的匹配得分为 1，不同核苷酸之间的替换得分为 0。这种打分方式的优点是简单，但由于没有考虑到实际的统计规律或生物演化规律，实际应用较少；

	A	T	C	G
A	1	0	0	0
T	0	1	0	0
C	0	0	1	0
G	0	0	0	1

2） 转换-颠换矩阵：DNA 序列由 4 种基本的脱氧核糖核苷酸单分子组成，核苷酸分子的核心结构差异是 4 种含氮碱基，分别是腺嘌呤（Adenine，简称 A），鸟嘌呤（Guanine，简称 G），胸腺嘧啶（Thymine，简称 T），胞嘧啶（Cytosine，简称 C）。在数据分析过程中，我们往往用碱基序列指代 DNA 序列。在生物演化过程中，4 种碱基发生相互替换的概率并不均等，嘌呤容易被嘌呤替换（A□G），嘧啶容易被嘧啶替换（C□T），这样的替换被称为转换；嘌呤与嘧啶之间发生替换，则被称为颠换。转换比颠换更容易发生，为体现这样的演化规律，在对序列比对进行打分时，通常对转换赋分为-1，对颠换赋分为-5.

	A	T	C	G
A	1	-5	-5	-1
T	-5	1	-1	-5
C	-5	-1	1	-5
G	-1	-5	-5	1

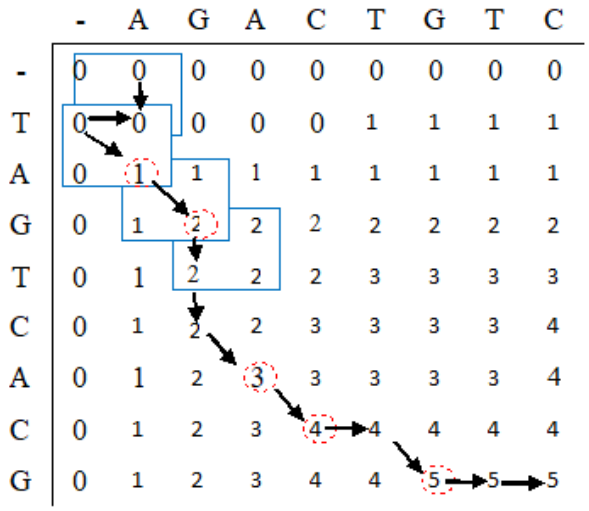
3） BLAST 矩阵：在实际使用过程中，根据统计经验，将碱基完全相同的情况赋分为 5，比对不上的情况赋分为-4，计分矩阵计算所得结果较好，因此将此方法广泛应用于 DNA 序列比对。

	A	T	C	G
A	5	-4	-4	-4
T	-4	5	-4	-4
C	-4	-4	5	-4
G	-4	-4	-4	5

动态规划算法

我们假设有两个长度分别为 m 和 n 的序列 S_1 和 S_2，其中 $S_1=a_1\ a_2\ ...\ a_m$，$S_2=b_1\ b_2\ ...\ b_n$。

我们将构建一个矩阵 A，其中 $A_{i,j}$ 表示 $a_1\ a_2\ ...\ a_i$ 和 $b_1\ b_2\ ...\ b_j$ 的最长共同子序列的长度。



	-	A	G	A	C	T	G	T	C
-	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	1	1	1	1
A	0	1	1	1	1	1	1	1	1
G	0	1	2	2	2	2	2	2	2
T	0	1	2	2	2	3	3	3	3
C	0	1	2	2	3	3	3	3	4
A	0	1	2	3	3	3	3	3	4
C	0	1	2	3	4	4	4	4	4
G	0	1	2	3	4	4	5	5	5

序列 S_1 是竖着写的，S_2 是横着写的。逐行、逐列地将代表行的每个字母与

代表列的每个字母进行比较。

1. 如果 $a_i = b_j$，那么我们就找到了一个匹配（match）。我们为当前的匹配赋值 1 分；

2. 如果 $a_i \neq b_j$，那么我们就有一个不匹配（mismatch）。在这种情况下，我们需要考虑两种可能性：由 $a_1 \dots a_i$ 和 $b_1 \dots b_{j-1}$ 得到的 LCS，以及由 $a_1 \dots a_{i-1}$ 和 $b_1 \dots b_j$ 的 LCS，

可由以下公式表示：

$$A_{i,j} = \begin{cases} A_{i-1,j-1} + 1 & \text{if } a_i = b_j \\ max(A_{i-1,j}, A_{i,j-1}) & \text{if } a_i \neq b_j \end{cases}$$

假如还是不理解动态规划算法，看这里：

writes down "1+1+1+1+1+1+1+1 =" on a sheet of paper

"What's that equal to?"

counting "Eight!"

writes down another "1+" on the left

"What about that?"

quickly "Nine!"

"How'd you know it was nine so fast?"

"You just added one more"

"So you didn't need to recount because you remembered there were eight! Dynamic Programming is just a fancy way to say 'remembering stuff to save time later'"

第三部分 序列比对工具

常见的序列比对软件类型：

1. 数据库搜索：

   BLAST, FASTA, PSI-BLAST……

2. 两序列比对：

   LASTZ, MUMmer……

3. 多序列比对：

   ClustalW, MUSCLE, T-Coffee

4. 基因组比对：

GMAP, BLAT, ACT

5．短序列比对：

BWA, SOAP, Bowtie, ELAND, GSNAP, Stampy

BWA 的下载、安装及配置方式：

本实验采用 BWA 进行比对分析。以下是 Bwa 软件主页：

http://bio-bwa.sourceforge.net/

linux 环境下，进行下列四步操作就可以使用了。

1．下载软件安装包：

wget http://jaist.dl.sourceforge.net/project/bio-bwa/bwa-0.7.12.tar.bz2

2．解压缩：tar jxf bwa-0.7.12.tar.bz2

3．进入解压缩目录：cd bwa-0.7.12

4．编译安装：make

使用时写明全路径/home/bin/bwa-0.7.12/bwa

或者设置环境变量（export PATH="/Pipeline/FIS.Traits/tools/:$PATH"，也可写入 ~/.bashrc 并 source）后仅使用程序名称。

四、数据准备

1.过滤后测序数据

less ./01_clean/test_clean.fq.gz

2.参考序列(FASTA 格式)

软件：BWA/Samtools/BCFtools

五、实验步骤

1. 为参考序列建立索引

/Pipeline/FIS.Traits/tools/bwa-mem2 index ./00_ref/MGI358.SNP.fa

```
(base) [geneu1@iZwz9ieukla9l29jthy267Z Tom]$ bwa index ./00_ref/MGI358.SN
[bwa_index] Pack FASTA... 0.00 sec
[bwa_index] Construct BWT for the packed sequence...
[bwa_index] 0.09 seconds elapse.
[bwa_index] Update BWT... 0.01 sec
[bwa_index] Pack forward-only FASTA... 0.00 sec
[bwa_index] Construct SA from BWT and Occ... 0.04 sec
[main] Version: 0.7.17-r1188
[main] CMD: /data/gene/bwa/bwa-0.7.17/bwa index ./00_ref/MGI358.SNP.fa
[main] Real time: 0.165 sec; CPU: 0.150 sec
```

2. 将测序 reads 比对到参考序列

/Pipeline/FIS.Traits/tools/bwa-mem2 mem -M -Y -t
1 ./00_ref/MGI358.SNP.fa ./01_clean/test_clean.fq.gz > ./02_align/test.clean.sam

```
(base) [geneu1@iZwz9ieukla9l29jthy267Z Tom]$ bwa mem -M -Y -t 1 ./00_ref/MGI358.SNP.fa ./01_clean/test_clean.fq
[M::bwa_idx_load_from_disk] read 0 ALT contigs
[M::process] read 200000 sequences (10000000 bp)...
[M::process] read 200000 sequences (10000000 bp)...
[M::mem_process_seqs] Processed 200000 reads in 3.836 CPU sec, 3.654 real sec
[M::process] read 73598 sequences (3679900 bp)...
[M::mem_process_seqs] Processed 200000 reads in 3.718 CPU sec, 3.529 real sec
[M::mem_process_seqs] Processed 73598 reads in 1.520 CPU sec, 1.430 real sec
[main] Version: 0.7.17-r1188
[main] CMD: /data/gene/bwa/bwa-0.7.17/bwa mem -M -Y -t 1 ./00_ref/MGI358.SNP.fa ./01_clean/test_clean.fq.gz
[main] Real time: 9.518 sec; CPU: 9.312 sec
```

3. 用 less 命令查看存储比对信息的序列文件 sam

less ./02_align/test.clean.sam

```
@SQ     SN:rs1490413    LN:2084
@SQ     SN:rs5745448    LN:2078
@SQ     SN:rs3737576    LN:2079
@SQ     SN:rs1698647    LN:2088
@SQ     SN:rs1343469    LN:2068
@SQ     SN:rs11239930   LN:2077
@SQ     SN:rs7554936    LN:2069
@SQ     SN:rs3829868    LN:2062
@SQ     SN:rs2814778    LN:2082
@SQ     SN:rs560681     LN:2088
@SQ     SN:rs10801520   LN:2088
@SQ     SN:rs1106201    LN:2082
@SQ     SN:rs2013162    LN:2075
@SQ     SN:rs2292564    LN:2080
```

Sam 文件格式详解见 https://en.wikipedia.org/wiki/SAM_%28file_format%29

4. 为节约存储，可将 sam 转换为二进制文件 bam，并对比对结果进行排序

samtools sort ./02_align/test.clean.sam -o ./02_align/test.clean.sort.bam

5．为排序后的 bam 文件创建索引

samtools index ./02_align/test.clean.sort.bam

```
(base) [geneu1@iZwz9ieukla9l29jthy267Z 02_align]$ ls
test.clean.sam  test.clean.sort.bam  test.clean.sort.bam.bai
```

6．samtools view -F 256 -
hb ./02_align/test.clean.sort.bam > ./02_align/test.clean.sort.uniq.bam

7．查看 bam 文件

samtools view -S ./02_align/test.clean.sort.bam|less -S

```
FS2000L1C001R001293843  0  rs1490413  978   60  3S47M  *  0  0  GGTGCTCAGAACTGCCTGGTGTGGACTGGGCTGATGTGGGTTCTTTGCAG  IIIIII
FS2000L1C002R004009003  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAGAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004012257  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAAAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004012842  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAGAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004017443  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAAAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004023488  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAGAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004028816  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAGAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004033135  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAAAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004042561  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAAAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004043471  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAAAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004044645  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAGAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
FS2000L1C002R004052152  0  rs1490413  1001  60  50M    *  0  0  TGGGCTGATGTGGGTTCTTTGCAAAACTGGCTGGCCTCAGAGCAGGGACA  IIIIII
```

8．统计覆盖深度

/Pipeline/FIS.Traits/tools/bamdst -p ./00_ref/target.358.SE50.subSNP.bed -
o ./02_align ./02_align/test.clean.sort.uniq.bam

9．查看覆盖深度统计情况

less ./02_align/depth.tsv.gz

#Chr	Pos	Raw Depth	Rmdup depth	Cover depth
rs1490413	1001	1123	1123	1123
rs1490413	1002	1131	1131	1131
rs1490413	1003	1131	1131	1131
rs1490413	1004	1131	1131	1131
rs1490413	1005	1131	1131	1131
rs1490413	1006	1131	1131	1131
rs1490413	1007	1131	1131	1132
rs1490413	1008	1132	1132	1132
rs1490413	1009	1132	1132	1132
rs1490413	1010	1132	1132	1132
rs1490413	1011	1131	1131	1132
rs1490413	1012	1132	1132	1132
rs1490413	1013	1132	1132	1132
rs1490413	1014	1132	1132	1132

六、实验后处理和预期结果

使用 IGV 可视化软件，将比对结果进行可视化展示，挑选感兴趣的变异位点，从图示中查看该位置的碱基变化情况。

IGV 下载、安装及可视化操作可参见以下链接：

https://www.jianshu.com/p/d7b2677fc134

预期结果图示：以 rs1490413 位点为例，从 IGV 软件视图界面可清晰查看该位点在参考序列上的位置、突变类型及突变引发氨基酸变化的情况。