《生物信息技术基础》期末总结

Notes of *Linux-based Essential Bioinformatics* taught by Prof. Luo JC

李 帅

2022 年 9 月 3 日

简介

这本小册子是《生物信息技术基础》课程的期末总结。

《生物信息技术基础》（英文：**L**inux-based **E**ssential **B**ioinformatics，或 The **B**asics of **B**ioinformation **T**echnology）（课程号：01139371）是由北京大学生命科学学院罗静初教授开设的一门生物信息学研本合上课程。

我在 2022 年春季学期选修了这门课（2022 年 2 月 21 日-2022 年 6 月 23 日），杨德昌学长担任课程助教，饶希晨 (师兄)、刘沛瑶 (学姐)、钱柏言和我是第一小组成员，饶希晨 (师兄) 是我们的组长，《转录因子 CTCF 的 ChIP-Seq 数据初步分析》是本组的期末项目。

课堂上，罗老师丁宁周至，言提其耳，每事指斥，不尚浮辞[1]。课后的学习，也堪称如切如磋，如琢如磨[2]，很有收获。

今为固定学习的成果，完成了这份笔记，它的主要内容的包括：

- **第一章 Linux 初步**、**第二章 Linux 基础** 介绍 Linux 系统的基本知识和使用技巧，帮助熟悉命令行操作；
- **第三章 序列比对基本方法** 补充全局比对和局部比对的理论，虽然不需要实际操作，但能够为后面的内容准备基础；
- **第四章 EMBOSS 软件包** 是利用 EMBOSS 软件包进行序列分析的使用教程；
- **第五章 BLAST 原理与实践** 和**第六章 HMMER 原理与实践** 聚焦两种原理迥异的常用序列比对软件及其操作；
- **第七章 InterPro** 简单介绍了 InterPro 数据库；
- **第八章 Web 开发与 MySQL 初步**、**第九章 RNA-seq 实验与分析** 是本课程其他组同学的期末项目，旨在获得初步网页构建、数据库操作及 RNA-seq 分析经验；
- **第十章 ChIP-seq 项目笔记** 详细记载了一次 ChIP-seq 分析的基本流程及相关的原理、工具和分析。

内容中的所有代码均已运行，所有的配置方法均无遗漏，相信它会对初学者有所帮助。如果有见识的人愿意给我一些指教，我的邮箱是 `li_shuai@pku.edu.cn` ，来信受欢迎。

在简介的最后和内容的开始，我想感谢罗静初老师的教导和帮助，无论是在课堂中，还是在课堂之外。

李帅

2022 年 9 月 3 日

[1] 《齐民要术 · 自序》【北魏】贾思勰。大意：反复嘱咐，揪着耳朵说话，每件事情都给予指导，不崇尚空洞无物的言辞。

[2] 《诗经 · 卫风 · 淇奥》，比喻互相商讨砥砺。

目录

第一章 Linux 初步 **1**
1.1 GNU/Linux 与 Linux 发行版 . . . 1
1.2 登陆服务器 . . . 1
1.3 壳 (shell) . . . 2
1.4 文件、目录与链接 . . . 2
1.5 监测 . . . 3
1.6 环境变量和配置文件 . . . 4
1.7 文件权限 . . . 4
1.8 vim 编辑器 . . . 5
1.9 conda . . . 6
1.10 终端多路复用器 . . . 8
参考文献 . . . 8

第二章 Linux 基础 **9**
2.1 基本脚本 . . . 9
2.1.1 Hello World . . . 9
2.1.2 使用变量 . . . 9
2.1.3 输入输出重定向 . . . 10
2.1.4 管道 . . . 11
2.1.5 退出代码 . . . 11
2.2 结构化命令 . . . 12
2.2.1 if语句和test命令 . . . 12
2.2.2 while、until和for语句 . . . 13
2.2.3 case语句 . . . 14
2.3 正则表达式与grep . . . 15
2.4 sed和awk . . . 17
参考文献 . . . 18

第三章 序列比对基本方法 **19**
3.1 相似性与同源性 . . . 19
3.2 打分矩阵 . . . 20

3.2.1 可接受点突变 (PAM) 矩阵 . . . 21
3.2.2 区块替换矩阵 (BLOSUM) . . . 21
3.3 全局比对算法 . . . 22
3.3.1 约定记号 . . . 22
3.3.2 Needleman-Wunsch 算法的基本思想 . . . 22
3.3.3 一个例子 . . . 23
3.4 局部比对算法 . . . 25
3.4.1 Smith-Waterman 算法的基本思想 . . . 25
3.5 空位罚分 . . . 26
3.5.1 仿射罚分模型 . . . 26
3.5.2 仿射罚分模型下的 Needleman-Wunsch 算法 . . . 26
参考文献 . . . 27

第四章 EMBOSS 软件包 28
4.1 欧洲分子生物学开放软件套装 . . . 28
4.1.1 历史的注记 . . . 28
4.1.2 什么是 EMBOSS? . . . 29
4.1.3 `wossname`: 第一个 EMBOSS 应用 . . . 29
4.1.4 `tmf`:EMBOSS 手册 . . . 32
4.2 处理序列 . . . 33
4.2.1 从数据库获得序列 . . . 33
4.2.2 从文件中读入序列 . . . 34
4.2.3 获知关于序列的信息 . . . 35
4.3 双序列比对 . . . 42
4.3.1 点阵图 . . . 42
4.3.2 `needle`进行全局比对 . . . 44
4.3.3 `water`进行局部比对 . . . 44
4.4 蛋白质分析 . . . 45
4.4.1 鉴定 ORF . . . 45
4.4.2 翻译序列 . . . 48
4.4.3 二级结构预测 . . . 48
4.4.4 其他的蛋白质分析软件 . . . 52
4.5 模式分析和多序列比对 . . . 52
4.5.1 模式匹配 . . . 53
4.5.2 蛋白质指纹 . . . 56
4.5.3 多序列分析 . . . 58
4.5.4 谱分析 . . . 60
参考文献 . . . 63

第五章 BLAST 原理与实践 **64**
5.1 BLAST: 1990 年 . . . 64
5.1.1 启发式算法 . . . 64
5.1.2 最大片段对（MSP）度量 . . . 65
5.1.3 MSP Score 的快速估计 . . . 65
5.1.4 BLAST 的步骤 . . . 66
5.1.5 衡量 BLAST 的表现 . . . 67
5.2 BLAST: 1997 年 . . . 68
5.2.1 两次命中法 . . . 69
5.2.2 空位处理能力 . . . 69
5.2.3 结果的统计显著性评价 . . . 70
5.2.4 迭代 BLAST . . . 71
5.3 用接近实践的例子来理解启发性 . . . 71
5.4 本地 BLAST 实践 . . . 78
5.4.1 数据库的获取或构建 . . . 78
5.4.2 `blastp` 和 `blastn` . . . 80
5.4.3 `psiblast` . . . 84
5.4.4 `blastx` 、`tblastn` 和 `tblastx` . . . 84
5.4.5 PHI-BLAST 与 `deltablast` . . . 86
5.4.6 命令行上的在线服务 `-remote` . . . 87
5.5 在线 BLAST 实践 . . . 88
参考文献 . . . 88

第六章 HMMER 原理与实践 **90**
6.1 谱 HMM 基础 . . . 90
6.1.1 谱方法及其优点 . . . 90
6.1.2 谱 HMM . . . 91
6.1.3 数学意味 . . . 92
6.2 本地 HMMER . . . 93
6.2.1 HMMER 软件包中的部分软件 . . . 93
6.2.2 `hmmsearch` 用一个 profile 搜索一个序列数据库 . . . 93
6.2.3 `phmmer` 用单个蛋白质序列搜索蛋白质序列数据库 . . . 100
6.2.4 `jackhmmer` 进行迭代的蛋白质搜索 . . . 100
6.2.5 用一个序列搜索一个 profile 数据库 . . . 101
6.2.6 搜索 DNA 序列 . . . 103
6.3 在线 HMMER . . . 104
参考文献 . . . 105

第七章 InterPro **106**
7.1 蛋白质标识 . . . 106
7.2 InterPro 与 InterProScan . . . 107
7.3 本地 InterProScan . . . 108
7.3.1 软件安装和依赖 . . . 108
7.3.2 使用示例数据运行 . . . 111
7.3.3 选项设置 . . . 114
7.3.4 输出格式 . . . 114
7.3.5 自定义数据运行 . . . 117
7.4 在线 InterProScan . . . 119
参考文献 . . . 122

第八章 Web 开发与 MySQL 初步 **123**
8.1 网页的组成部分 . . . 123
8.2 超文本标记语言 (HTML) . . . 123
8.2.1 HTML 文件的结构 . . . 123
8.2.2 标签和元素 . . . 124
8.2.3 属性 . . . 125
8.2.4 列表 . . . 126
8.2.5 块 . . . 127
8.3 层叠样式表 (CSS) . . . 128
8.3.1 CSS 文件的结构 . . . 128
8.3.2 设计与布局——以 ABC 主页为例 . . . 131
8.4 网站搭建、ABC 主页与 PHP . . . 138
8.4.1 Web 开发技术的历史 . . . 138
8.4.2 搭建静态网站 . . . 139
8.4.3 ABC 主页的结构与 PHP 初步 . . . 140
8.5 Javascript 简介 . . . 145
8.5.1 数据类型和运算符 . . . 145
8.5.2 条件和循环 . . . 146
8.5.3 函数、回调函数、匿名函数 . . . 147
8.5.4 对象与类 . . . 148
8.6 MySQL 初步 . . . 149
8.6.1 数据库相关概念 . . . 149
8.6.2 MySQL 及其安装配置 . . . 150
8.6.3 MySQL 数据库操作 . . . 152
8.6.4 MySQL 数据表操作 . . . 154
参考文献 . . . 157

第九章 RNA-seq 实验与分析 **158**
9.1 下一代测序 . . . 158
9.1.1 桥式扩增 . . . 158
9.1.2 循环可逆终止 . . . 159
9.1.3 Illumina 测序平台 . . . 159
9.2 RNA-seq . . . 161
9.2.1 一般原理 . . . 161
9.2.2 文库的结构 . . . 162
9.2.3 方法变体 . . . 162
9.3 概述和准备工作 . . . 164
9.3.1 数据来源及实验设计 . . . 164
9.3.2 配置环境 . . . 165
9.3.3 下载参考文件和测序数据 . . . 165
9.4 RNA-seq 分析 I . . . 168
9.4.1 序列修剪 . . . 168
9.4.2 质量控制 . . . 171
9.4.3 比对和转换 . . . 180
9.4.4 转录本组装 . . . 186
9.4.5 生成计数矩阵 . . . 188
9.5 RNA-seq 分析 II . . . 188
9.5.1 R 与 Bioconductor 简介 . . . 188
9.5.2 用 DESeq2 进行差异表达分析 . . . 189
9.5.3 GO 富集分析 . . . 194
参考文献 . . . 196

第十章 ChIP-seq 项目笔记 **198**
10.1 原理、方法和背景 . . . 198
10.1.1 ChIP-seq 原理 . . . 198
10.1.2 ChIP-seq 实验方案 . . . 198
10.1.3 CTCF 简介 . . . 200
10.2 概述和准备工作 . . . 201
10.2.1 分析流程 . . . 201
10.2.2 配置环境 . . . 201
10.2.3 下载测序数据和参考文件 . . . 202
10.3 一般的分析 . . . 204
10.3.1 测序数据质量控制 . . . 205
10.3.2 去接头 . . . 206
10.3.3 比对 . . . 207
10.3.4 比对结果的排序、过滤、去重、索引 . . . 209

10.4 ChIP-seq 分析 214
10.4.1 ChIP-seq 质量控制 214
10.4.2 寻找富集峰 214
10.4.3 合并两个生物学重复 (可选项) 220
10.5 ChIP-seq 结果的分析 221
10.5.1 计算与 TSS 的共定位 221
10.5.2 Peak 注释 225
10.5.3 寻找 motif 232
10.6 页边集 235
参考文献 235

第一章　Linux 初步

本章介绍 Linux 命令行的初步知识，但不包括编写脚本。本章的部分内容整理自个人总结 1，此外补充了关于 `conda` 、终端多路复用器、监测等内容。

1.1　GNU/Linux 与 Linux 发行版

Linux 是一个操作系统，它包括 Linux 内核、GNU 工具、图形化桌面环境和应用软件。

- **Linux 内核**由社区维护，第一个 Linux 内核由 Linux Torvald 编写。内核负责管理系统内存、软件程序、硬件设备和文件系统
- **GNU 工具**是可执行一些标准功能，包括处理文件、处理文本、管理进程、与用户交互。GNU 工具包括 coreutils 软件包和 shell
- **图形化桌面环境**包括 XWindow、KDE 和 GNOME 等
- **应用软件**是由用户安装的软件

Linux 发行版一般包括上述四个部分，方便安装。常见发行版包括 Ubuntu、Mint、CentOS 等。

1.2　登陆服务器

Linux 服务器与普通 PC 不同，用户难以直接接触，因此，一般使用客户端进行远程登录。最常用的登陆方式是使用 SSH 服务。SSH 是安全外壳协议（Secure Shell）的简称，是一种加密的网络传输协议，可在不安全的网络中为网络服务提供安全的传输环境。

如果是 Mac 系统，可在 Terminal 中使用 `ssh` 命令；如果是 Windows 系统，可安装 Ubuntu Linux 子系统，在子系统中使用 `ssh` 命令。Windows 更方便的登录方式是使用支持 SSH 的终端模拟器以实现命令行界面 (Command line interface, CLI) 交互，常用的终端模拟器有 Xshell 和 PuTTY。

以 Xshell 为例：首先下载并安装Xshell，启动 Xshell，然后选择“新建会话”，填写主机 IP、端口号、用户名和密码，即可保存并连接。

`exit` 命令可从终端登出，Xshell 中可直接关闭窗口退出。

首次登陆需要用 `passwd` 修改登录密码，输入密码时，屏幕不会打印输入的字符。

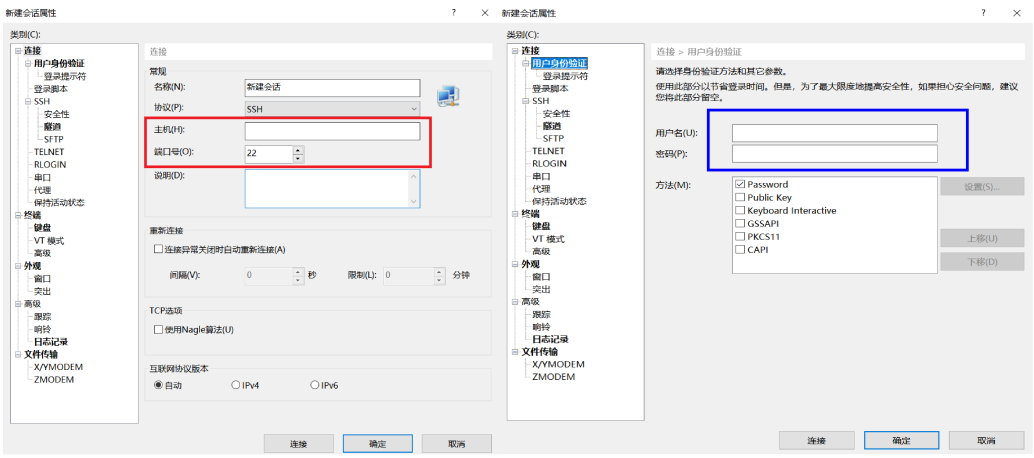


图 1.1: Xshell 登陆服务器。左侧红框填写主机 IP、端口号，右侧蓝框填写用户名和密码。随后按指示操作即可。

1.3　壳 (shell)

Linux 系统中，常常有几种 shell，包括 GNU bash、ash、zsh、但发行版默认的 shell 都是 GNU Bash Shell。GNU bash shell 能提供对 Linux 系统的交互式访问，`/etc/passwd` 文件对应用户名的第 7 个冒号分隔字段指定了用户使用的 shell 程序。

bash shell 位于 `/bin` 目录下，是一个可执行程序。在一个 shell 中运行 `/bin/bash` 将创建一个子 shell，子 shell 可以嵌套，运行脚本时，也是创建了子 shell。`exit` 命令可用来逐步退出子 shell，乃至于关闭命令行界面。

不属于 bash shell 的程序是**外部命令**，或文件系统命令，通常位于 `/bin` 、`/usr/bin` 、`/sbin` 或 `/usr/sbin` 。可用 `which <command>` 查找一个命令的位置。外部命令执行会创造子进程，这个过程称为 fork。

内建命令（builtin）不需要使用子进程来进行，它们是 shell 的一部分，可使用 `type <command>` 来查看一个命令是否内建的。内建命令执行的速度更快，效率更高。

bash 内置了手册，可以作为使用时的参考，可用 `man <command>` 来查看某个命令的手册。手册是逐页显示的，阅读完毕后按 q 退出，使用方法类似 `less` 命令。除了 `man` 意外，`info` 命令也可以提供命令的参考资料。此外，一般命令都可以接受 `-help` 或 `--help` 选项，并打印使用帮助。

1.4　文件、目录与链接

`pwd` 打印当前工作目录。

`mkdir` 创建目录。

`cd` 在目录间进行切换，`cd ~` 返回家目录，`cd ..` 返回父目录。

`tree <dirname>` 命令以树状图列出目录内容，十分清晰。

`ls` 显示文件/目录列表。常用 `ls` 参数显示如下：

`cp <source> <destination>` 用于复制文件。`mv <source> <destination>` 用于移动文

参数	含义
-a	列出所有文件，包括隐藏文件
-l	长格式显示
-h	以易读的方式显示文件大小
-S	按文件大小排列
-t	按修改时间排列

件，也可用于重命名文件。rm <filename> 用于删除文件。如果上述三者的作用对象是整个目录，则必须递归地 -r 执行。

此外，rmdir 可以删除空目录。

touch <filename> 用于创建一个空文件，touch -a <filename> 可更改文件的时间戳。

file 命令可以查看文件类型，cat 命令可以在屏幕上打印文本文件中的所有内容，cat -n 会打印行号。less 命令能够一次一屏地显示文本，并具备高级搜索功能。

tail 命令可以查看文件最后几行的内容，默认是最后 10 行，可以通过 -n 选项来修改要显示的行数。

head 命令可以查看文件最后几行的内容，默认是最后 10 行，可以通过 -n 选项来修改要显示的行数。

链接是文件的虚拟副本，是目录中指向文件真实位置的占位符。链接有两种：

- 硬链接：默认情况下，每个文件有一个硬链接。硬链接和文件本身没有区别，当删除一个文件的所有硬链接时，一个文件才算删除了。硬链接有两个缺点：1. 不能引用与该链接不在同一磁盘分区的文件。2. 无法引用目录。
- 符号链接：创建一个特殊类型的文献，该文件包含了指向所引用文件或目录的文本指针。当向符号链接中写入内容时，也向文件中写入了内容。删除符号链接不影响文件。删除文件将使对应的符号链接变成坏链接，再 ls 命令下显示红色。

ln <file> <link> 创建硬链接，ln -s <file> <link> 创建软链接。

1.5 监测

ps 命令可监测进程。默认运行 ps 讲输出运行在当前控制台下的属于当前用户的进程，显示进程 ID(PID)、运行的终端 (TTY) 及进程已用的 CPU 时间。

ps -ef 可以查看系统上运行的所有进程的详细信息。如增加 -l 参数，则会输出更多信息。

top 命令能实时显示进程信息，top 命令输出非常丰富，但也好懂。

kill <PID> 能够终止特定进程，但前提是该进程属于你。

df 命令能查看所有已挂载磁盘的使用情况，一般添加 -h 参数，输出成易读的形式。

du 命令能查看当前目录的磁盘使用情况，-h 参数输出成易读的形式，-c 显示所有已列出文件的大小。

1.6 环境变量和配置文件

环境变量能够持久地存储关于 shell 会话和工作环境的信息，它能够被程序或脚本访问到。全局环境变量对于 shell 会话和所有生成的子 shell 都是可见的，但局部变量只对创建它们的 shell 可见。

`printenv` 命令可打印所有全局环境变量及其值，可以打印指定的环境变量如 `printenv HOME` ，也可以通过引用打印环境变量，如 `echo $HOME` 。

`set` 命令可显示全局环境变量，局部环境变量及用户定义的变量。

设置局部用户定义变量时，变量名、等号和值之间不能有空格，否则 bash shell 就会把值当作一个单独的命令，引发 `command not found` 错误。用户自己的变量定义后，也可以通过 `$varname` 的形式来调用。

设置全局环境变量的方法是先创建一个局部环境变量再将其导出到全局环境中，通过 `export` 命令完成。`export` 以后，子 shell 中改变局部变量将不会影响父 shell 中的全局变量，即使再次 `export` 。

`unset <varname>` 可以删除已经存在的局部环境变量，但它无法影响全局环境变量。

一个特殊的环境变量是 `PATH` ，它定义了用于进行命令和程序查找的目录，使用冒号分隔。放在这些目录下的文件不需要使用绝对路径即可执行。如果新安装的程序不处于 `PATH` 定义的目录下，需要增补 `PATH` 变量的内容。

登录时，**登录 shell** 将自动执行 `/etc/profile` 、`~/.bash_profile` 、`~/.bashrc` 、`~/.bash_login` 和 `~/.profile` `/etc/profile` 是主启动文件，登录时就会执行，包含了系统环境变量。`~/.profile` 将负责执行 `~/.bashrc` 文件，目的是提供一个用户专属的启动文件来定义该用户用到的环境变量。如欲在 `PATH` 中添加目录，或定义额外的环境变量，需要更改 `~/.profile` 文件；其他的更改则只需在 `~/.bashrc` 中进行。修改配置文件之前必须备份。修改配置文件后不会立即生效，需要使用 `source .bashrc` 再次读入环境。

当用 `bash` 命令启动 shell 时，这个 shell 是**交互式 shell**，它不会访问 `/etc/profile` ，而只会访问 `~/.bashrc` 。

当运行 shell 脚本时，系统用的 shell 是**非交互式 shell**，它将继承当前 shell 的变量。

1.7 文件权限

使用 `ls -l` 查看文件信息时，第 2-10 个字符分别代表文件所有者（user）权限（3 位，读 + 写 + 执行）、同组用户（group）权限（3 位，读 + 写 + 执行）、其他用户（others）权限（3 位，读 + 写 + 执行）。- 代表没有该项权限。这 9 个字符合称**文件模式**。

`chmod` 命令用于更改文件模式。该命令有两种使用方式：

- 八进制数字表示法：每个八进制数字可以表示 3 位二进制数字，正好对应三种权限的有无。通过三位八进制数字可以很方便地表示所有者，组成员和其他所有用户的文件模式。常用的对应关系：7：rwx，6：rw-，5：r-x，4：r–，0：—。
- 符号表示法：u，g，o，a 表示影响对象，分别对应：user，group，others 及 all=u+g+o。如果没有指定字符，则默认是 all。+，-，= 分别表示增加权限，减少权限以及“只有指

定的权限可用，其他所有权限都被删除”。权限由 r，w，x 指定。

```
1  $ ls -l
2  total 4
3  -rw-r--r-- 1 leb1c leb 5 Mar 14 21:54 test.txt
4
5  $ chmod 731 test.txt # 八进制数字表示法更改文件模式
6  $ ls -l
7  total 4
8  -rwx-wx--x 1 leb1c leb 5 Mar 14 21:54 test.txt
9
10 $ chmod u-x test.txt # 符号表示法更改文件模式
11 $ ls -l
12 total 4
13 -rw--wx--x 1 leb1c leb 5 Mar 14 21:54 test.txt
```

1.8　vim 编辑器

vim 是 vi improved 的缩写，是一个功能强大且复杂的文本编辑器。`vi` 后接文件名（如此文件不存在将创建），即可打开 vim。此时 vim 处于命令模式。

在 vim 的命令模式下，输入 `:q` 退出 vim。输入 `:q!` 可强制退出 vim，这样不会保留修改。

如果不能确定 vim 处于何种模式，按两次 esc 键以退回命令模式。

命令模式下，按 `i` 进入插入模式，此时底部将显示 `--INSERT--` ，即可开始输入。

按 esc 从插入模式退出，而后输入 `:w` 保存文件，之后输入 `:q` 退出。也可以输入 `:wq` ，代表保存后退出。如果在保存时重命名了文件，则执行的是“另存为”功能。

命令模式下，有一些光标移动快捷键，列出如下：

先输入数字，再按上述按键，代表将对应动作重复做多次，例如输入 `2` ，再按 `j` 就代表光标向下移动 2 行。

数字后接 shift+g 代表移动到文件的特定行。

命令模式下，一些基本编辑命令列出如下：

下面这一系列命令称为 `d` 命令，它的本质是剪切命令，但常常用作删除。

`d` 命令删除的文本并不会消失，而是进入缓存，用户可继续使用 `p` 命令将其粘贴到光标之后，也可以使用 shift+p 将其粘贴到光标之前。

与 `d` 命令对应地，有一系列 `y` 命令，它们的用法相似，只是功能从剪切变成了复制。

shift+j 合并当前行与下面一行。

`f` 命令实现行内搜索。光标定位以后，输入 ; 可使 vim 重复上一次搜索（也即找下一个）。

按键	光标动作
h 或左方向键	向左移动
j 或下方向键	向下移动
k 或上方向键	向上移动
l 或右方向键	向右移动
0	到本行开头
^	到本行的第一个非空字符
$	到本行尾
crtl+F 或 page down 键	向下翻页
ctrl+B 或 page up 键	向上翻页
shift+G	到文件的最后一行

按键	操作
o	在当前行下方插入空行，并进入插入模式
shift+o	在当前行上方插入空行，并进入插入模式
u	撤销上一步操作
x	删除当前字符
number x	删除自当前字符（含）开始向后共计 number 个字符

按键	操作
dd	删除当前行
number dd	删除自此行（含）开始向后共计 number 行
d $	删除当前字符到行尾的字符
d0	删除当前字符到行首的字符
dG	删除当前字符到文件末尾的所有内容
d number G	删除当前字符到文件第 number 行的所有内容

/ 命令可以搜索整个文件中的单词和短语，输入 n 可以重复上一次搜索。搜索可以利用正则表达式。

如果 vim 太过复杂，也可从简单的 nano 编辑器开始，直接使用 nano 命令即可启动，常用快捷键被列出在屏幕下方。

1.9 conda

conda 是一个软件管理工具能够方便地进行软件下载，软件更新，软件降级和环境管理。它的安装和配置方法如下：

先下载 miniconda 安装包：网页，选择 Linux Installer-Python 3.9-Miniconda3 Linux 64-bit，下载，上传到服务器家目录下。(或直接 wget https://docs.conda.io/en/latest/miniconda.html##linux-installers)

而后在家目录下执行命令：`bash ./Miniconda3-py39_4.11.0-Linux-x86_64.sh` 等待滚屏，滚屏停止则不断按 Enter，有询问 `[yes/no]` 时输入 no。安装完成。conda 的使用方法如下：

```
# 更改权限并激活
$ cd ~/miniconda3/bin
$ chmod u+x activate
$ source ./activate
# 此时提示符之前应该出现(base)

# 增加频道:
$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/
$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda/
$ conda config --add channels bioconda
$ conda config --add channels conda-forge

# 创建并进入环境
$ conda create -n <env_name>
# 按提示操作即可创建
$ source activate <env_name>
# 激活环境，此时提示符前应该出现(环境名)
$ conda deactivate
# 退出当前环境

# 安装软件
$ conda search <software>
# 查询是否能安装软件
$ conda install <software>
# 安装支持conda安装的软件
```

1.10 终端多路复用器

命令行的典型使用方式是，打开一个终端窗口 (terminal window)，在里面输入命令。用户与计算机的这种临时的交互，称为一次会话 (session)。会话的特点是，窗口与其中启动的进程是连在一起的。打开窗口，会话开始；关闭窗口，会话结束，会话内部的进程也会随之终止，不管有没有运行完。

一个典型的例子就是，SSH 登录远程计算机，打开一个远程窗口执行命令。这时，网络突然断线，再次登录的时候，是找不回上一次执行的命令的。因为上一次 SSH 会话已经终止了，里面的进程也随之消失了。

为了解决这个问题，应该把会话与窗口" 解绑": 窗口关闭时，会话并不终止，而是继续运行，等到以后需要的时候，再让会话" 绑定" 其他窗口。

终端多路复用器 (Terminal Multiplexer) 实现了这个 "解绑" 功能，常用的终端复用器包括 Screen、Tmux、Byobu 等。下面这里介绍 Tmux。

`tmux new -s <session-name>` 可新建一个指定名称的会话，并进入执行任务。

`tmux detach` 可将当前会话与窗口分离，会话中进行的任务不会中断，即使在此时断开与服务器的连接。

`tmux attach -t <session-name>` 重新接入已经存在的会话，可进入其中查看任务是否已经完成。

`tmux kill-session -t <session-name>` 可用于杀死某个会话，当它已不再被需要时。

`tmux switch -t <session-name>` 可直接切换到指定会话，即使当前处于另一个会话中。

`tmux rename-session` 用于重命名会话。

`exit` 可退出当前的 Tmux 窗口。

此外，Tmux 可以将窗口分成多个窗格（pane），每个窗格运行不同的命令。

`tmux split-window` 将窗口划分为上下两个窗格，而如果加上 `-h` 选项则划分为左右两个窗格。

`tmux select-pane -U` 可将光标切换到上方窗格，`-D` 、`-L` 、`-R` 分别代表切换到下方、左侧和右侧窗格。

上述是 Tmux 的基础功能，更加详细的用法介绍，如窗口管理和快捷键，可参考其开发者提供的入门指南。

参考文献

[1] 绍茨 (Shotts, W. E.)., 郭光伟, & 郝记生. (2013). *linux* 命令行大全. 人民邮电出版社.

[2] 布卢姆 (Blum, R.)., 布雷斯纳汉 (Bresnahan, C.)., 门佳, & 武海峰. (2016). *linux* 命令行与 *shell* 脚本编程大全 (第 3 版 ed.). 人民邮电出版社.

[3] Tmux 使用教程

[4] 罗静初. Linux 生物信息技术基础第 1 讲 30 个常用 Linux 命令. 课堂讲义.

第二章　Linux 基础

本章介绍 Linux 的基础知识，包括脚本、结构化命令、正则表达式以及文本处理的实用工具 sed 、awk 等。本章的主要内容整理自个人总结 1，补充了关于结构化命令、test 以及 sed 和 awk 的初步使用。

2.1　基本脚本

2.1.1　Hello World

编写第一个脚本：

```
1 #!/bin/bash
2 #This is my first script.
3 echo 'Hello world!'#可以增加注释
```

脚本第一行的 ##! 是特殊结构，称为 Shebang，它告知操作系统后面的脚本该使用哪个 shell，每个脚本都应当以此作为第一行，并且可以指定 bash shell 或其他 shell。执行该脚本需要指定其路径，如果其所在路径已经被添加到 PATH 环境变量中，则不需要指定路径

编写脚本的一些提示包括：不要随意增加空格；Tab 键缩进可以使脚本结构清晰；vim 有语法高亮模式，在命令模式输入 :syntax on 以打开。

echo 命令是输出的重要手段，echo 输出文本字符串，一般不需要引号，但亦可用引号划定。注意，如果输出的文本包含双引号，则需使用单引号划定输出范围。echo -n 可取消字符串后的换行符。

2.1.2　使用变量

脚本中，环境变量可以通过 $ 来引用，但如果只是想输出美元符号 $ 本身，则需用 \ 转义。

创建用户变量时，一定要注意变量、等号和值之间不能出现空格。用户变量也可用美元符号引用。

shell 对于变量的管理是宽松的，当其遇到一个不存在的变量时就立刻创建它。因此注意用花括号圈定扩展内容，以免误解。shell 不区分常量和变量，但约定用大写字母表示常量。shell 赋值时，如无特殊声明，一概作字符串处理。

命令替换是一种复杂的参数扩展，是将一个命令的输出作为一个扩展模式使用，或者说，是将一个命令的输出作为另一个命令的参数列表。

命令替换用两种使用形式，第一种是美元符号 + 括号 $(<command>)，第二种是反引号 ``，注意，变量、等号和命令替换字符之间不能有空格。

```
$ # 命令替换：which cp的输出作为ls -l的参数

$ ls -l $(which cp)
-rwxr-xr-x 1 root root 141528 Jan 18 2018 /bin/cp

$ ls -l which cp
-rwxr-xr-x 1 root root 141528 Jan 18 2018 /bin/cp
```

2.1.3 输入输出重定向

标准输入（stdin）、标准输出（stdout）、标准输出（stderr）是特殊文件，它们在 shell 中的内部文件描述符分别是 0、1、2。stdin 一般链接到硬盘，程序自此得到输入。stdout、stderr 直接连接到屏幕，不会保存。程序运行的结果会发送到 stdout，默认显示在屏幕上。程序运行产生的错误信息会发送到 stderr，默认也显示在屏幕上。如有需要，可将其重定向到其他文件。

stdout 重定向示例：

```
$ ls
0224  0302d 0307  0311  install  skel     x-y.svg
0228  0303  0310  0314  MyEMBOSS tutorial

$ ls -l ~ > ls_output.txt # 将ls的输出重定向到ls_output.txt中
$ ls
0224  0302d 0307  0311  install      MyEMBOSS tutorial
0228  0303  0310  0314  ls_output.txt skel    x-y.svg

$ ls / >> ls_output.txt
# 注意，>>将这次的标准输出重定向到ls_output.txt，
# 这样做不会覆写它，而是增添内容在其末尾
```

stderr 重定向示例：

```
$ ls /03 2> ls_error.txt # 并不存在/03这个目录，因此产生错误
$ ls
0224  0302d 0307  0311  install     ls_output.txt skel    x-y.svg
```

```
4 0228 0303  0310 0314 ls_error.txt MyEMBOSS    tutorial
```

stdout 和 stderr 同时重定向：

```
1 $ ls /03 &> ls_output_and_error.txt
```

stdin 重定向示例：

```
1 $ cat > new.txt # 将标准输入定向到new.txt，这实际上在新建文件
2 hello world!
3 ^C
4
5 $ cat new.txt
6 hello world!
7
8 $ cat new.txt > brand_new.txt # 将new.txt定向到brand_new.txt
9 $ cat brand_new.txt
10 hello world!
```

除此以外，还有一种使用 << 的内联输入重定向，其基本格式为：

```
1 command << marker
2 data
3 marker
```

data 是指定用于输入重定向的数据，marker 为文本标记（可用任何字符串作为文本标记），用于划分输入数据的开始和结尾。

内联输入重定向最常被用于 Shell 自动交互，或可将多行命令换行书写。

2.1.4 管道

管道 | 的作用是将一个命令的标准输出作为另一个命令的标准输入。

管道中的每个命令是过滤器，它们接受输入，按照某种方式对输入进行改变，然后输出。

```
1 $ ls . | sort | less
2 # 先列出，而后排序，最后输出
```

管道符两侧的命令实际上是同时执行的，在第一个命令产生输出的同时，立即被送给第二个命令，不会用到任何中间文件或缓冲区。

2.1.5 退出代码

shell 中运行的每个命令在运行完毕时都会向 shell 传递一个退出状态码，退出状态码是一个 0-255 的整数值，保存在 $? 中。但要注意的是，$? 仅仅保存 shell 所执行的最后一条命

令的状态码。

状态码的一些参考：

状态码	描述
0	命令成功结束
1	一般性未知错误
2	不适合 shell 的命令
126	命令不可执行 (没有权限)
127	找不到命令
128	无效的退出参数
128+x	与 Linux 信号 x 相关的严重错误
130	ctrl+C 终止的命令
255	正常范围之外的退出状态码

2.2 结构化命令

2.2.1 if语句和test命令

if-then 结构判断紧随其后的命令的返回状态，如果返回 0 则执行 then 之后的命令，其基本结构如下：

```
1 if <commands>; then
2         <commands>
3 [elif <commands>; then
4         <commands>...]
5 [else
6         <command>...]
7 fi
```

test <expression> 命令可检测函数或表达式的退出状态，当表达式为真，则返回 0 退出状态。

[expression] 与 test <expression> 等价。

直接决定是否运行 then 之后的命令是 if 后命令的返回状态，而非某一表达式是否为真，表达式的值需要被 test 命令转换为返回状态。

基于 test 命令还有整数判断操作、字符串比较操作。以下逐一简介：

正则表达式匹配操作 [[string1~=regex]] ，如果 string1 匹配正则表达式 regex 则返回真。

[[expression]] 实际上是一种更现代的 test 命令，不仅用于正则表达式的支持。

整数计算操作 (()) ，其中的计算结果非 0 时返回真。

字符串表达式	为真的条件
string	不为空
-n string	长度大于 0
-z string	长度等于 0
string1==string2	二者相等
string1!=string2	二者不相等
string1\\>string2	排序时，string1 在 string2 后

整数判断表达式	为真的条件
int1 -eq int2	int1 与 int2 相等
int1 -ne int2	int1 与 int2 不相等
int1 -le int2	int1 小于或等于 int2
int1 -lt int2	int1 小于 int2
int1 -ge int2	int1 大于或等于 int2
int1 -gt int2	int1 大于 int2

组合表达式：

对应于 `test` 命令的逻辑运算符（AND,OR,NOT）分别是 `-a` , `-o` , `!` 。

对应于 `[[]]` 或 `(())` 的的逻辑运算符（AND,OR,NOT）分别是 `&&` , || , `!` 。

`test` 中所有操作符均被 shell 看作命令参数，因此在 bash 中有特殊含义的字符都必须进行转义或者用引号引起来。但 `[[]]` , `(())` 则不需要。

2.2.2 while、until和for语句

`while` 循环的基本结构：

```
1 while commands; do
2         commands
3 done
```

同样的，循环体是否会继续被执行，仍然依赖于 `while` 后命令的退出状态（退出状态为 0 时进行循环），因此依然可以使用 `test` 命令。

`break` 命令可立即终止循环（跳出）。

`continue` 命令可跳过本次循环（跳到下一次判断）。

`until` 循环的基本结构：

```
1 until commands; do
2         commands
3 done
```

循环体是否会被执行，依赖于 `until` 后命令的退出状态（退出状态为 0 时终止循环）。

for 也是循环命令，推荐使用类似 C 语言表达方式：

```
for (( expression1; expression2; expression3 )); do
        commands
done
#上面格式等价于
(( expression1 ))
while (( expression2 )); do
        commands
                (( expression3 ))
done
```

可以以一个简单的数数程序来说明：

```
#!/bin/bash
#simple counter: demo for C style of "for"

for (( i=0; i<5; i=i+1 )); do
    echo $i
done
```

2.2.3 case语句

case 是一个多选项命令，其基本结构为：

```
case word in
pattern1)
    command1
    command2
    ...
    commandN
    ;;
pattern2)
    command1
    command2
    ...
    commandN
    ;;
esac
```

当 case 第一次发现吻合的模式后将不再比对剩下的模式。

一般而言，最后一个模式应设定为 * ，来兜底不吻合前面所有模式的情况。

2.3 正则表达式与grep

正则表达式是一种符号表示法，用于识别文本模式，可以看作用途更广泛的“通配符”。

正则表达式有一些元字符，除了元字符以外都是普通字符，元字符有特殊含义。

- . 匹配任意字符，但不能不匹配。
- ^ 是开头锚定符。
- $ 是结尾锚定符。
- [] 匹配字符集中的某个字符或某一字符类，元字符在中括号中失去其特殊含义，但在中括号中首位的 ^ 表示否定，在中括号中非首位的 - 表示字符范围。
- {} 、- 、? 、* 、+ 、() 、| 、\ 也是元字符。

正则表达式有一定的规范，例如 POSIX 规范，它将正则表达式的实现分为基本正则表达式（BRE）和扩展正则表达式（ERE）。

基本正则表达式 (BRE) 只承认 ^ 、$ 、. 、[] 、* 为元字符，ERE 则增添了 () 、{} 、? 、| 、+ 。

这容易造成问题，如欲使用 ERE 定义的元字符，请将 grep 替换为 egrep 或 grep -E 。

一些元字符可指定匹配的次数，称为限定符。

- * 限定符表示匹配某元素 0-多次。
- ? 限定符表示前面的元素是可选的，即匹配某元素 0 次或 1 次。
- + 限定符表示匹配某元素一次或多次。
- {} 以指定次数匹配某元素，{n} 匹配前面的元素 n 次，{n,m} 匹配前面的元素出现 n-m 次之间，{n,} 匹配前面的元素出现超过 n 次，{,m} 匹配前面的元素不超过 m 次。

下面看一些例子:

```
^\(?[[0-9][0-9][0-9]\)? [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$
# 上面的表达式将会匹配(nnn)nnn-nnnn或nnn nnn-nnnn。斜杠表示此括号为文字字符

[[:upper:]][[:upper:][:lower:] ]*\.
# 上述表达式将会匹配一个句子（以大写字母开头，以句号结尾）
# 结构：两个中括号表达式 and *元字符 and 用反斜杠转义的.符号
# 第一个中括号表达式：大写字母开头
# 第二个中括号表达式：含两个字符类及一个空格，中间可以是大小写字母和空格
# *：第二个中括号表达式中的内容出现任意次
# 转义.：点结尾

^([[:alpha:]]+ ?)+$ #匹配由单个空格分隔的一个或多个字符组成的行
```

```
13
14 ^\(?[[0-9]{3}\)? [0-9]{3}-[0-9]{4}$ #与第一个表达式同义
```

ERE 支持以 | 表示或选项，但为了避免 shell 将其理解为管道符，需要在单引号中使用。

grep 命令在文件中搜索与指定正则表达式相匹配的行并将结果输送到标准输出，其含义为全局正则表达式打印（global regular expression print）。

常用选项如下：

选项	功能
i	忽略大小写
v	输出不匹配行
c	输出匹配到的项目数目
l	输出匹配项文件名
L	输出不含匹配项的文件名
n	在每个匹配行之前加上改行在文件内的行号
h	多文件搜索时抑制文件名输出

用 12HUMAN_CEA.FASTA 演示如下：摘取此 FASTA 文件中的所有信息行

```
 1 $ grep -E '^>CEA.*$' 12HUMAN_CEA.FASTA # 第一种方式
 2 >CEAM3_HUMAN | P40198 | J Luo 2020-07-29
 3 >CEAM4_HUMAN | O75871 | J Luo 2020-07-29
 4 >CEA19_HUMAN | Q7Z692 | J Luo 2020-07-29
 5 >CEAM7_HUMAN | Q14002 | J Luo 2020-07-29
 6 >CEA21_HUMAN | Q3KPI0 | J Luo 2020-07-29
 7 >CEA16_HUMAN | Q2WEN9 | J Luo 2020-07-29
 8 >CEAM6_HUMAN | P40199 | J Luo 2020-07-29
 9 >CEAM8_HUMAN | P31997 | J Luo 2020-07-29
10 >CEA18_HUMAN | A8MTB9 | J Luo 2020-07-29
11 >CEAM1_HUMAN | P13688 | J Luo 2020-07-29
12 >CEA20_HUMAN | Q6UY09 | J Luo 2020-07-29
13 >CEAM5_HUMAN | P06731 | J Luo 2020-07-29
14
15 $ grep -E '[0-9]{4}-[0-9]{2}-[0-9]{2}$' 12HUMAN_CEA.FASTA # 第二种方式
16 >CEAM3_HUMAN | P40198 | J Luo 2020-07-29
17 >CEAM4_HUMAN | O75871 | J Luo 2020-07-29
18 >CEA19_HUMAN | Q7Z692 | J Luo 2020-07-29
19 >CEAM7_HUMAN | Q14002 | J Luo 2020-07-29
20 >CEA21_HUMAN | Q3KPI0 | J Luo 2020-07-29
21 >CEA16_HUMAN | Q2WEN9 | J Luo 2020-07-29
```

```
22 >CEAM6_HUMAN | P40199 | J Luo 2020-07-29
23 >CEAM8_HUMAN | P31997 | J Luo 2020-07-29
24 >CEA18_HUMAN | A8MTB9 | J Luo 2020-07-29
25 >CEAM1_HUMAN | P13688 | J Luo 2020-07-29
26 >CEA20_HUMAN | Q6UY09 | J Luo 2020-07-29
27 >CEAM5_HUMAN | P06731 | J Luo 2020-07-29
```

find 命令在指定文件夹下查找文件名，添加 -regex 选项以应用正则表达式。

例子：寻找差文件名

```
1 $ find . -regex '.*[^-_./0-9a-zA-Z].*'
2
3 ./install/EMBOSS-6.6.0/m4/lt~obsolete.m4
4 ./0314/worse$name.txt
5 ./0314/bad name.txt
```

2.4 sed和awk

sed 编辑器是一种流编辑器 (stream editor)，它不同于 vim 等交互式编辑器，需要在处理数据之前基于预先提供的一组规则来编辑数据流。

对于数据流，sed 编辑器会执行如下操作：

- 一次从输入中读取一行数据
- 根据所提供的编辑器命令匹配数据
- 按照命令修改流中的数据
- 将新的数据输出到标准输出

sed 编辑器的命令可在命令行内定义，如：

```
1 $ echo "This is a test" | sed 's/test/new test/'
2 This is a new test
```

这里使用了 s 命令，指定斜线间的第二个文本字符串来替换斜线间的第一个文本字符串模式。

sed 编辑器亦可指定文件输入，格式为 sed options script file 。-

sed 编辑器并不会修改文本文件的数据，而是将修改后的数据发送到标准输出，原来的数据仍然保持原状。

如要在 sed 命令行上执行多个 sed 命令时，可使用 -e 选项，命令之间用分号分开，命令末尾和分号中间不能有空格，每一条命令都会作用到文件中的每行。

如果有大量要处理的 sed 命令，可将其放到一个单独的文件，如 script.sed ，而后以 -f 选项指定对应的 sed 脚本文件。

```
1 $ sed -f script.sed data.txt
```

gawk 是 awk 程序的 GNU 版本，它提供了一种编程语言，通过它可以定义变量保存数据、使用算术和字符串操作来处理数据、使用结构化的编程概念以及格式化。

gawk 常用于从大文件中提取数据元素，并将其格式化为报告。

gwak 程序的脚本用一对花括号来定义，脚本必须放在一对花括号中，如下所示的命令：

```
1 $ gawk '{print "Hello World!"}'
```

此时，在标准输入 (键盘) 输入任意字符串并回车，gwak 程序就会在标准输出打印 Hello World! ，直到按下 Ctrl-D 向程序发送 EOF 字符，即终止 gawk 的运行。

gawk 会自动为每一行中的每个数据字段分配一个变量，每个数据字段都是通过字段分隔符划分的，默认的字段分隔符为空白字符，如空格和制表符。这些字段中，$0 代表整个文本行，$n 代表文本行中的第 n 个数据字段。

要在命令行上的 gawk 运行多条命令，只需要在命令之间放一个分号即可；gawk 也支持将程序存储到文件中，可使用 -f 指定 gawk 脚本文件。

gawk 还有一些关键字，例如 BEGIN 关键字可强制 gawk 在读入数据前执行其后指定的程序脚本；而 END 关键字可强制 gawk 在读入数据后执行其后指定的程序脚本

sed 和 gwak 详细教程篇幅很大，已经超出了 “Linux 基础” 的范围，推荐阅读参考文献 [2] 第 19-22 章。

参考文献

[1] 绍茨 (Shotts, W. E.)., 郭光伟, & 郝记生. (2013). *linux 命令行大全*. 人民邮电出版社.

[2] 布卢姆 (Blum, R.)., 布雷斯纳汉 (Bresnahan, C.)., 门佳, & 武海峰. (2016). *linux 命令行与 shell 脚本编程大全* (第 3 版 ed.). 人民邮电出版社.

[3] 罗静初. Linux 生物信息技术基础第 1 讲 30 个常用 Linux 命令. 课堂讲义.

第三章　序列比对基本方法

本章介绍序列比对的基本方法，围绕经典的 Needleman-Wunsch 算法和 Smith-Waterman 算法展开，旨在为后面讨论 EMBOSS、BLAST 和 HMMER 等实用工具准备基础。本章的内容是根据第一小组第三次讨论重新整理的，是新增的。

3.1　相似性与同源性

如果两条生物序列是从同一个共同祖先序列进化而来的，则二者之间具有**同源性**。然而，同源性是一个非量化的、动态的、模糊的概念。因此判断两条生物序列是否同源的主要依据就是它们之间的**相似性**。然而不幸的是，没有一种单一、精确或普遍适用的相似性概念。

计算机科学中，字符串的相似性可以用**汉明距离** (Hamming distance) 来衡量。两个等长字符串之间的汉明距离是指两个字符串对应位置的不同字符的个数，是将一个字符串变换成另外一个字符串所需要替换的字符个数。

因为插入与缺失的存在，生物序列可能并非完全等长，汉明距离并不实用，因而引入**编辑距离** (Edit distance)。编辑距离是指需要多少次的处理才能将一个字符串变成另一个字符串，处理包括插入 (Insertion)、删除 (Deletion) 和替换 (Substitution)。

相似性的一种定性展示是**点阵图**，将两条序列 s、t 分别置于平面直角坐标系的两个坐标轴上，当 s_i 和 t_j 的字符相同时就在坐标 (i, j) 处绘制一个圆点，如果两条序列间存在公共子串，则会在图中形成平行于对角线的线段。

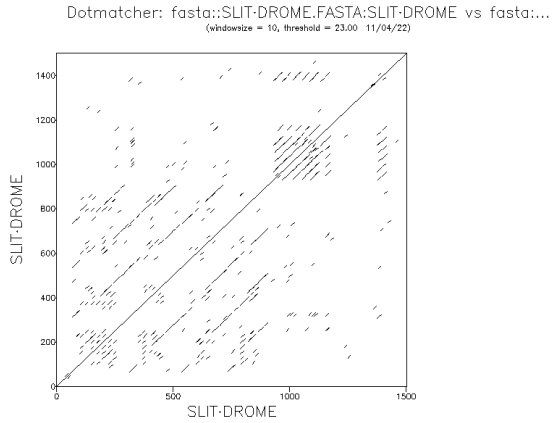
使用 EMBOSS 软件包中的 `dotmatcher` 程序来绘制点阵图，这里运行的是自身比对，旨在寻找重复区域，运行情况如图 3.1。

```
1 dotmatcher SLIT_DROME.FASTA SLIT_DROME.FASTA -graph png -goutfile
    SLIT_DROME_W10T23 # 默认情况下, -window 10 -threshold 23
2 dotmatcher SLIT_DROME.FASTA SLIT_DROME.FASTA -graph png -goutfile
    SLIT_DROME_W24T20 -window 24 -threshold 20
3 dotmatcher SLIT_DROME.FASTA SLIT_DROME.FASTA -graph png -goutfile
    SLIT_DROME_W38T20 -window 38 -threshold 20
4 dotmatcher SLIT_DROME.FASTA SLIT_DROME.FASTA -graph png -goutfile
    SLIT_DROME_W38T30 -window 38 -threshold 30
```

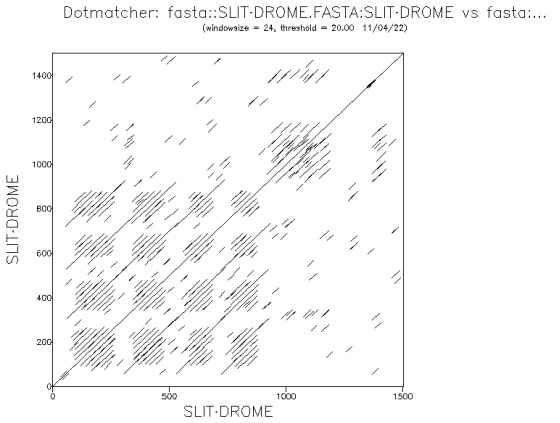
`dotmatcher` 程序默认地使用了**滑动窗口方法**，`-window` 指定了窗口的大小，`-threshold` 指定了窗口内得分的阈值。该方法在决定 (i, j) 位置是否需要绘制点时，会将 $s_i s_{i+1} \cdots s_{i+w-1}$

与 $t_j t_{j+1} \cdots t_{j+w-1}$ 作一比较，并根据某种内置的规则给出分值，只有当分值大于阈值 T 时，程序才会在图中 (i, j) 位置绘制点。

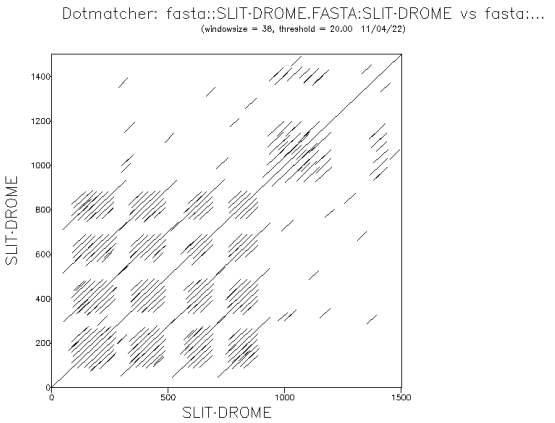
滑动窗口方法使点阵图较为简洁，改变窗口大小和阈值参数将产生对点阵图产生影响：阈值设得越大，背景噪声越小，窗口大小与重复单元大小相近时，结果清晰。



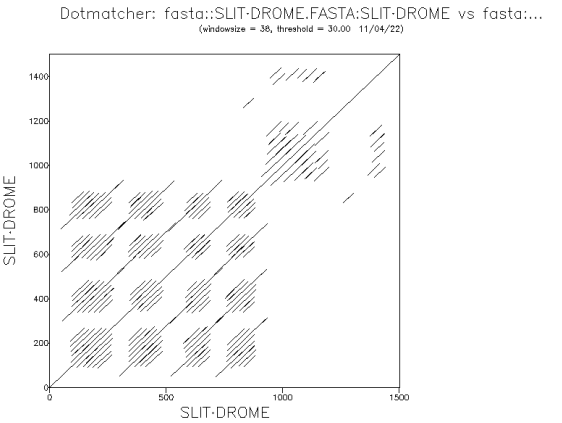
(a) 窗口 =10, 阈值 =23



(b) 窗口 =24, 阈值 =20



(c) 窗口 =38, 阈值 =20



(d) 窗口 =38, 阈值 =30

图 3.1: 点阵图方法寻找同源区域

3.2　打分矩阵

对于任意两条序列，很容易给出它们的比对。比对的数目很多，为了找出最好的比对，需要有一种统一的，量化的**打分规则**。为了简单，一般假设比对中的各个字符对可以分别打分，而整个比对的得分是各个字符对得分的和。打分规则应该是简单的，同时应该是有生物学意义的。

考虑到各种替换情况，打分规则常常以矩阵的形式展示，如蛋白质序列比对中常用的**可接受点突变** (point accepted mutations, PAM) **矩阵**或**区块替换矩阵** (block substitution matrix, BLOSUM)。

3.2.1　可接受点突变 (PAM) 矩阵

可接受点突变矩阵由 Dayhoff 及同事建立。构建 PAM 矩阵的模型是：序列中氨基酸的改变是一个马尔可夫过程，每个位点的历次氨基酸改变是独立的，与该位点之前的改变历史没有关系，并假设更远的变化反映了多次发生的短期变化。那么自然地，我们就可以通过从这些短期变化外推得出可用于比较更远相关蛋白质的矩阵。

PAM 的字面含义 “可接受的点突变”，就是指不会严重破坏蛋白质功能的，能够被遗传的突变。

PAM 单位：两个蛋白质序列 s、t，使用一系列的可接受点突变将 s 变为 t，若平均每 100 个残基发生一次点突变，则称这两条序列相距 1 个 PAM 单位。

第一个 PAM 矩阵的构建方法如下：

1. 寻找相似性在 85% 以上的保守氨基酸序列
2. 根据匹配计分进行多重比对，比对结果排除空格
3. 以比对结果构建进化树，树中能反映氨基酸的替换关系
4. 计算每种氨基酸转换成其他氨基酸的次数
5. 计算每种氨基酸的突变率
6. 计算每对氨基酸的突变率，得到突变概率矩阵
7. 将此矩阵按需要自乘若干次
8. 将此矩阵中，氨基酸 $i \to$ 氨基酸 j 的突变率除以氨基酸 i 的出现频度，取常用对数后乘以 10，即得到 PAM 矩阵

3.2.2　区块替换矩阵 (BLOSUM)

BLOSUM 矩阵由 Henikoff 等提出，其原理是直接利用多序列比对分析亲缘关系较远的蛋白质序列。

Henikoff 等研究了 Prosite 数据库中的大量氨基酸序列，发现了 1961 个**区块**，一个区块是从 k 条序列中选取高度保守的区域并经过多序列比对形成的长度为 b 的区域。从区块开始构建 BLOSUM 矩阵的方法是：

1. 将每个区块中有一定相似性的序列合并成同一序列
2. 统计各个区块中的氨基酸对的数量 f
3. 计算氨基酸对的出现频率 q
4. 计算每种氨基酸的期望频率 p
5. 计算氨基酸对出现的期望频率 e
6. 计算氨基酸对的出现频率与期望频率的比值 $\frac{q}{e}$
7. 将此比值取 2 的对数，乘以 2，即得到 BLOSUM 矩阵

BLOSUM 矩阵后的数字表示构建矩阵的区块的最小差别程度。

3.3　全局比对算法

3.3.1　约定记号

- S：打分函数，$S(a,b)$ 为 a 和 b 匹配的得分（由指定的矩阵给出），$S(-,a)$ 或 $S(a,-)$ 为插入或删除状态的得分，这里规定 $S(-,a)=S(a,-)=-g$，$-g$ 是空位得分
- Σ：字符表，所有可能出现的字符，对于蛋白质是 20 种氨基酸
- $s=s_1s_2s_3\cdots s_i\cdots s_m$ 是一条长度为 m 的字符串
- $t=t_1t_2t_3\cdots t_j\cdots t_n$ 是一条长度为 n 的字符串
- $prefix(s,i)$ 表示字符串 s 的长度为 i 的前缀子串，即 $prefix(s,i)=s_1s_2s_3\cdots s_i$
- $prefix(t,j)$ 表示字符串 t 的长度为 j 的前缀子串，即 $prefix(t,j)=t_1t_2t_3\cdots t_j$
- $\boldsymbol{H}$：二维动态规划辅助矩阵，有 $m+1$ 行 (编号为 $0,1,\cdots,m$),$nx+1$ 列 (编号为 $0,1,\cdots,n$)
- $H_{i,j}$ 表示矩阵 $\boldsymbol{H}$ 中的元素，代表 $prefix(s,i)$ 与 $prefix(t,j)$ 的最佳比对得分

3.3.2　Needleman-Wunsch 算法的基本思想

Needleman-Wunsch 算法采用了**动态规划**思想，即“把多阶段过程转化为一系列单阶段问题，利用各阶段之间的关系，逐个求解”，换言之，“为了找一个问题的最优解，首先找到更小规模的子问题的最优解，然后将这些子问题的最优解进行组合，从而获得原始问题的最优解”。

双序列的比对需要填充辅助矩阵 $\boldsymbol{H}$，并标明路径，具体步骤如下：

第一步：初始化辅助矩阵，填充第 0 行与第 0 列，规定 $H_{0,0}=0$，其余元素按照固定空位罚分给分。

$$\boldsymbol{H}=\begin{bmatrix} 0 & -g & \cdots & -ng \\ -g & & & \\ \vdots & & & \\ -mg & & & \end{bmatrix}$$

第二步：递推计算每个 $H_{i,j}$，即计算 $prefix(s,i)$ 与 $prefic(t,j)$ 的最佳比对得分，对于每一个位点，其得分仅有三种情况，可写成递推公式：

$$H_{i,j}=\max\begin{cases} H_{i-1,j}+S(s_i,-) \\ H_{i,j-1}+S(-,t_j) \\ H_{i-1,j-1}+S(s_i,t_j) \end{cases} \tag{3.1}$$

在计算得分时，也同时需要记录最佳得分是从哪一种（或几种）情形得到的（匹配、插入、缺失），对应于三种箭头：↖　←　↑或它们的组合。

第三步：根据填充 $\boldsymbol{H}$ 的同时记录的箭头，写出最优比对的路径。注意，最优路径可能有多条。$H_{m,n}$ 的数值即最优比对的得分。

3.3.3　一个例子

为了进一步说明 Needleman-Wunsch 算法的运行方式，这里举一个核酸序列比对的简单例子，并予以逐步解释：

- 打分函数 S：匹配得分为 1，空位得分为-1，错配罚分为-2
- 待比对序列 s 为 `GAAGACAA`
- 待比对序列 t 为 `CAGAAA`

第一步，初始化辅助矩阵：由于每个空位给分都是-1，因此长度为 k 的片段与空字符串比对，得分为 $-k$。

$$\boldsymbol{H} = \begin{bmatrix} 0 & -1 & -2 & -3 & -4 & -5 & -6 \\ -1 & & & & & & \\ -2 & & & & & & \\ -3 & & & & & & \\ -4 & & & & & & \\ -5 & & & & & & \\ -6 & & & & & & \\ -7 & & & & & & \\ -8 & & & & & & \end{bmatrix}$$

第二步，计算 $prefix(s,i)$ 与 $prefic(t,j)$ 的最佳比对得分，例如对于 $H_{1,1}$：

$$H_{1,1} = \max \begin{cases} H_{0,1} + S(G,-) \\ H_{1,0} + S(-,C) \\ H_{0,0} + S(G,C) \end{cases} \tag{3.2}$$

其中：$H_{0,0} = 0$，$H_{1,0} = -1$，$H_{0,1} = -1$，$S(G,-) = S(-,C) = -1$，$S(G,C) = -2$，则，

$$H_{1,1} = \max \begin{cases} -1 + (-1) \\ -1 + (-1) \\ 0 + (-2) \end{cases} = -2 \tag{3.3}$$

将其填入矩阵 $\boldsymbol{H}$：

$$\boldsymbol{H} = \begin{bmatrix} 0 & -1 & -2 & -3 & -4 & -5 & -6 \\ -1 & -2 & & & & & \\ -2 & & & & & & \\ -3 & & & & & & \\ -4 & & & & & & \\ -5 & & & & & & \\ -6 & & & & & & \\ -7 & & & & & & \\ -8 & & & & & & \end{bmatrix}$$

注意到，无论是配对、插入还是缺失的情况，$H_{1,1}$ 均等于-2，因此，在这一点上应该同时记录 $\nwarrow$ $\leftarrow$ $\uparrow$。

接下来就可以计算 $H_{i,2}$ 和 $H_{2,1}$ 了，下面的矩阵是填充完毕的结果：

$$\boldsymbol{H} = \begin{bmatrix} 0 & -1 & -2 & -3 & -4 & -5 & -6 \\ -1 & -2 & -3 & -1 & -2 & -3 & -4 \\ -2 & -3 & -1 & -2 & 0 & -1 & -2 \\ -3 & -4 & -2 & -3 & -1 & 1 & 0 \\ -4 & -5 & -3 & -1 & -2 & 0 & -1 \\ -5 & -6 & -4 & -2 & 0 & -1 & 1 \\ -6 & -4 & -5 & -3 & -1 & -2 & 0 \\ -7 & -5 & -3 & -4 & -2 & 0 & -1 \\ -8 & -6 & -4 & -5 & -3 & -1 & 1 \end{bmatrix}$$

同时，也把回溯指针（箭头）记录在了另一个 $m \times n$ 的矩阵中：

$$\begin{bmatrix} \nwarrow\leftarrow\uparrow & \nwarrow\leftarrow\uparrow & \nwarrow & \leftarrow & \leftarrow & \leftarrow \\ \nwarrow\leftarrow\uparrow & \nwarrow & \leftarrow\uparrow & \nwarrow & \nwarrow\leftarrow & \nwarrow\leftarrow \\ \nwarrow\leftarrow\uparrow & \nwarrow\uparrow & \nwarrow\leftarrow\uparrow & \nwarrow\uparrow & \nwarrow & \nwarrow\leftarrow \\ \nwarrow\leftarrow\uparrow & \uparrow & \nwarrow & \leftarrow\uparrow & \uparrow & \nwarrow\leftarrow\uparrow \\ \nwarrow\leftarrow\uparrow & \nwarrow\uparrow & \uparrow & \nwarrow & \nwarrow\leftarrow\uparrow & \nwarrow \\ \nwarrow & \leftarrow\uparrow & \uparrow & \uparrow & \nwarrow\leftarrow\uparrow & \uparrow \\ \uparrow & \nwarrow & \leftarrow\uparrow & \nwarrow\uparrow & \nwarrow & \nwarrow\leftarrow\uparrow \\ \uparrow & \nwarrow\uparrow & \nwarrow\leftarrow\uparrow & \nwarrow\uparrow & \nwarrow\uparrow & \nwarrow \end{bmatrix}$$

从右下到左上，按照箭头回溯，可行的路径为：

$$\begin{bmatrix} \nwarrow & & & & & & \\ \uparrow & \nwarrow & & & & & \\ & \nwarrow\uparrow & & & & & \\ & & \nwarrow & & & & \\ & & & \nwarrow & & & \\ & & & \uparrow & & & \\ & & & & \nwarrow & & \\ & & & & & \nwarrow & \end{bmatrix}$$

这里的两条路径，最终得分均是 1:

路径	$\nwarrow$	$\nwarrow$	$\uparrow$	$\nwarrow$	$\nwarrow$	$\uparrow$	$\nwarrow$	$\nwarrow$
s	G	A	A	G	A	C	A	A
t	C	A	-	G	A	-	A	A

路径	↖	↑	↖	↖	↖	↑	↖	↖
s	G	A	A	G	A	C	A	A
t	C	-	A	G	A	-	A	A

比对完毕。

3.4　局部比对算法

3.4.1　Smith-Waterman 算法的基本思想

局部比对是指将两个部分区域高度相似的序列进行比对，旨在寻找两条较长序列的最佳匹配的子区域。局部比对的经典算法是 Smith-Waterman 算法，它的主要考虑是，最终获得的最佳比对不必完全包含整条序列

局部比对同样需要构建 $(m+1)\times(n+1)$ 的辅助矩阵 $\boldsymbol{H}$，但其中元素的含义发生了改变，$H_{i,j}$ 代表 $prefix(s,i)$ 的所有后缀串与 $prefix(s,j)$ 的所有后缀串的最佳比对得分。因为局部比对中，最佳比对不必包含整条完整序列。

比对计算中，如果发现继续延长原有的比对片段反而导致得分小于 0 时，就应该放弃延长而启用一个新的局部比对。基于这种思想，Smith-Waterman 算法分为以下几步。

第一步：初始化辅助矩阵，填充第 0 行与第 0 列，规定所有第 0 行或第 0 列的元素为 0。$H_{i,0}=0$ 表明，$prefix(s,i)$ 与长度为 i 的空位比对时未参与计分，即允许最终形成最佳比对区域中忽略掉 $prefix(s,j)$。

$$\boldsymbol{H}=\begin{bmatrix}0 & 0 & \cdots & 0\\ 0 & & & \\ \vdots & & & \\ 0 & & & \end{bmatrix}$$

第二步：递推计算每个 $H_{i,j}$，对于每一个位点，其得分有 4 种情况，可写成递推公式：

$$H_{i,j}=\max\begin{cases}0\\ H_{i-1,j}+S(s_i,-)\\ H_{i,j-1}+S(-,t_j)\\ H_{i-1,j-1}+S(s_i,t_j)\end{cases}\tag{3.4}$$

在计算得分时，也同时需要记录最佳得分是从哪一种（或几种）情形得到的（匹配、插入、缺失），对应于三种箭头：↖　←　↑或它们的组合。

第三步：根据填充 $\boldsymbol{H}$ 的同时记录的箭头，写出最优比对的路径。与全局比对不同的是，局部比对并不总是终结于两条序列的末尾。因此，回溯起点可能在矩阵中的任何位置。因此，回溯需要在整个 $\boldsymbol{H}$ 矩阵中寻找最大的元素，并以此作为起点，而一旦回溯到达得分为 0 的元素，则终止该路径（如果此处仍有箭头指处，则亦可继续进行，这两种情形的得分相同）。同样地，最有比对可能并不唯一。

3.5　空位罚分

3.5.1　仿射罚分模型

前述的讨论中，对每个空位都取固定的分值，这种模型称为**固定空位罚分模型**。

在生物进化的过程中，核苷酸的插入与缺失都需要断开核酸链，这是需要能量的，对此进行罚分是合理的；但是，插入或删除碱基的数目却可多可少，应该根据数目的不同赋不同的罚分，且连续插入或缺失的情况，罚分应该更轻一些。

固定空位罚分模型可以表示为：

$$w(k) = g \cdot k, \quad k = 1, 2, \ldots$$

则一种更符合进化理念的模型可表示为：

$$w(k) = g_o + g_e \cdot (k-1), \quad k = 1, 2, \ldots$$

其中，g_o 为打开空位罚分，g_e 为延长空位罚分，在形式上是一种仿射变换，因此称为**仿射空位罚分模型**。

如果考虑仿射空位罚分模型，全局和局部比对算法将大大变得复杂，下面考虑仿射空位罚分模型下的全局比对算法：

3.5.2　仿射罚分模型下的 Needleman-Wunsch 算法

引用仿射罚分模型寻找双序列最优全局比对需要借助三个动态规划辅助矩阵：

$$\boldsymbol{H}_{(m+1)\times(n+1)}, \quad \boldsymbol{F}_{m\times n}, \quad \boldsymbol{E}_{m\times n}$$

其中，

- $H_{i,j}$ 表示在最右端为 s_i 与 t_j 对准 (匹配或替换) 时，$prefix(s,i)$ 与 $prefix(t,j)$ 的最佳比对得分
- $E_{i,j}$ 表示在最右端为 s_i 与一个空位对准 (删除) 时，$prefix(s,i)$ 与 $prefix(t,j)$ 的最佳比对得分
- $F_{i,j}$ 表示在最右端为一个空位与 t_j 对准 (插入) 时，$prefix(s,i)$ 与 $prefix(t,j)$ 的最佳比对得分

根据上述定义，可写出如下递推关系式：

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + S(s_i, t_j) & i = 1, 2, \ldots, m; \\ E_{i-1,j-1} + S(s_i, t_j) & j = 1, 2, \ldots, n; \\ F_{i-1,j-1} + S(s_i, t_j) & \end{cases} \tag{3.5}$$

$$E_{i,j} = \max \begin{cases} H_{i-1,j} - g_o & i = 1, 2, \ldots, m; \\ E_{i-1,j} - g_e & j = 0, 1, \ldots, n; \\ F_{i-1,j} - g_o & \end{cases} \tag{3.6}$$

$$F_{i,j} = \max \begin{cases} H_{i,j-1} - g_o & i = 0, 1, \ldots, m; \\ E_{i,j-1} - g_o & j = 1, 2, \ldots, n; \\ F_{i,j-1} - g_e \end{cases} \tag{3.7}$$

同样地，先进行初始化，设置 $H_{0,0} = 0$，并从上到下从左到右计算所有元素值，最佳得分比对取值为：$\max\{H_{m,n}, E_{m,n}, F_{m,n}\}$。回溯时从最高分元素开始，不仅按照记录的指针向上和向左，还需要在矩阵之间进行“跳跃”。

而考虑仿射罚分模型下的局部比对算法只需在全局比对算法中的三个递推公式中均加入一个 0 项，回溯时从所有三个矩阵的最大元素出发寻找最优路径即可，这里不再赘述。

参考文献

[1] Mount D. W. (2008). Comparison of the PAM and BLOSUM Amino Acid Substitution Matrices. *CSH protocols*, 2008, pdb.ip59.

[2] 王勇献, & 王正华. (2011). *生物信息学导论: 面向高性能计算的算法与应用*. 清华大学出版社.

[3] 杨晶. (2010). *生物计算: 生物序列的分析方法与应用*. 科学出版社.

[4] 琼斯 (Jones, N. C.)., 帕夫纳 (Pevzner, P. A.)., & 王翼飞. (2007). *生物信息学算法导论*. 化学工业出版社生物 · 医药出版分社.

[5] 许忠能. (2008). *生物信息学*. 清华大学出版社.

第四章　EMBOSS 软件包

EMBOSS Gentle Tutorial[1]最初由 Val Curwen 和其他人撰写，现在由 David Martin 维护。它会介绍一些有用和泛用的程序，引导读者使用 EMBOSS，特别是以命令行的形式。

本章以 EMBOSS Gentle Tutorial 为基础，涵盖了文档的大部分内容，并运行了所有的代码，不能运行的代码被替换或删除了。

文档还根据罗老师的文章[2]和课堂讲授补充了一些内容。

原文档的说明抄录如下：

> Introduction to Sequence Analysis using EMBOSS
>
> This document was generated using the LaTeX2HTML translator Version 98.1p1 release (March 2nd, 1998) Copyright © 1993, 1994, 1995, 1996, 1997, Nikos Drakos, Computer Based Learning Unit, University of Leeds.
>
> The command line arguments were: `latex2html -no_math -html_version 3.2, math -t EMBOSS tutorial -split +1 emboss_tutorial.tex.` The translation was initiated by Gary Williams on 2003-04-29

本章主要分析的序列文件是: 非洲爪蟾视紫红质基因序列、非洲爪蟾视紫红质 mRNA 序列，可点击在 NCBI 下载。

4.1　欧洲分子生物学开放软件套装

4.1.1　历史的注记

在 1988 年，序列分析软件包 EGCG 开始为占主流的商业序列分析软件包 GCG 提供扩展。EGCG 的发展是小组合作的结晶，小组成员来自 EMBnet 及 EMBnet 以外。

EGCG 曾为 Sanger 中心的关键测序活动提供支持，构成了其内部使用的新的序列分析软件的基础。它同时也为大约 150 个站点和 EMBnet 国家服务的 10000 余用户提供了高级功能。该项目达到了基于 GCG 软件包所能实现的极限。

可由于环境的变化，分发使用了 GCG 库的学术软件源代码已不再被允许，分发二进制文件也变得越来越困难。因此，前 EGCG 开发者开始着手创造一个全新的学术序列分析软件，这就是今天 EMBOSS 项目的诞生。

[1] 参考文献 [2]

[2] 参考文献 [3]

4.1.2 什么是 EMBOSS？

欧洲分子生物学开放软件套装[3](The European Molecular Biology Open Software Suite, **EMBOSS**) 是一个新的，自由的开源软件分析包，它为满足分子生物学使用者社区（例如 EMBnet）的需要而开发。

该软件自动处理多种格式的数据，亦允许网络上序列数据的透明检索（transparent retrieval）。

不仅如此，它实际上构建了一个遵循真正开源精神的，允许其他科学家开发和发布软件的平台，与软件包同时提供的扩展库就是很好的例子。

EMBOSS 也将当前可用的序列分析软件包和工具包无缝地整合为了一个整体。EMBOSS 拦腰阻断了学术工具走向商业软件包的历史趋势。

EMBOSS 软件包具有如下特性：

- 提供一个完整的序列分析工具集（超过 150 种工具）
- 提供一系列核心软件库（AJAX 和 NUCLEUS）
- 整合了其他的公共软件包
- 鼓励在序列分析训练中使用 EMBOSS
- 鼓励其他方面的开发者使用 EMBOSS 库
- 支持所有的常见 Unix 平台，包括 Linux，Digital Unix，Irix 及 Solaris

在 EMBOSS 中你将找到超过 150 个程序（应用软件），它们涵盖但不仅涵盖以下这些方面：

- 序列比对
- 在数据库中快速搜索序列模式
- 蛋白基序鉴定，包括结构域分析
- 表达序列标签分析
- 核酸序列模式分析，例如 CpG 岛的鉴定
- 简单的和物种特异性的重复序列鉴定
- 小基因组的密码子使用分析
- 在大尺度序列集中快速鉴定序列特征
- 面向出版的展示工具
- 更多...

获取更多关于 EMBOSS 的信息请访问EMBOSS 主页。

4.1.3 `wossname`: 第一个 EMBOSS 应用

所有的 EMBOSS 程序都在 Unix 命令行上开始运行。我们将用一个特定的例子介绍一些基础操作：EMBOSS 实用程序 `wossname` 将产生一个包含所有 EMBOSS 应用的列表。

[3] 参考文献 [1]

```
$ wossname
Find programs by keywords in their short description
Text to search for, or blank to list all programs: protein
SEARCH FOR 'PROTEIN'
antigenic    Find antigenic sites in proteins
backtranambig Back-translate a protein sequence to ambiguous
   nucleotide sequence
backtranseq  Back-translate a protein sequence to a nucleotide
   sequence
cathparse    Generate DCF file from raw CATH files
. . .
topo         Draw an image of a transmembrane protein
tranalign    Generate an alignment of nucleic coding regions from
   aligned proteins
wordcount    Count and extract unique words in molecular sequence(s)
```

这一命令将返回 EMBOSS 程序名称，每个名称后紧跟着单行描述文档。

很多 EMBOSS 程序有提供更多功能性的附加/可选参数。一般来说，您可以通过在程序名称后附加 `-opt` 标签来强制其显示这些信息，就像下面这样。

```
$ wossname -opt
Find programs by keywords in their short description
Text to search for, or blank to list all programs: protein
Use the expanded group names [N]:
Match all words in the search string [Y]: N
Show keywords with program documentation [N]: N
Tool metadata output file [stdout]: myfile
Format the output for HTML [N]: Y
Output only the group names [N]:
Output an alphabetic list of programs [N]: Y
```

你将会看到很多附加选项，方括号内是该选项的默认值，你可以按 return 来接受默认值，也可以键入你需要的值。

上述操作的含义是，令 `wossname` 输出一个 HTML 格式的文件 `myfile` ，程序名按照字母顺序排列。

如欲得到所有 EMBOSS 程序的列表，启动 `wossname` 后不必指定任何值，按下 return。所有程序的列表将滚入你的屏幕，这些程序根据功能分组。滚动以查看全部。您能想到如何将这些数据转入一个文档吗?（提示：使用 `-opt` ）

如果你在任何 EMBOSS 程序名后附加 `-help` 标签，你将看到该程序所能识别的所有标签的列表。例如：

```
$ wossname -help
Find programs by keywords in their short description
Version: EMBOSS:6.6.0.0

   Standard (Mandatory) qualifiers:
   [-search]          string    Enter a word or words here and a
                                case-independent search for it will be
                                   made
                                in the one-line documentation, group
                                   names
                                and keywords of all of the EMBOSS
                                   programs.
                                If no keyword is specified, all
                                   programs
                                will be listed. (Any string)

   Additional (Optional) qualifiers:
   -explode           boolean   [N] The groups that EMBOSS applications
                                belong to have two forms, exploded and
                                   not
                                exploded. The exploded group names are
                                   more
                                numerous and often vaguely phrased than
                                    the
                                non-exploded ones. The exploded names
                                   are
                                formed from definitions of the group
                                   names
                                that start like NAME1:NAME2 and which
                                   are
                                then expanded into many combinations of
                                    the
                                names as: 'NAME1', 'NAME2', 'NAME1
                                   NAME2',
                                NAME2 NAME1'. The non-expanded names
                                   are
                                    simply like: 'NAME1 NAME2'.
    . . .
```

```
26     -gui                boolean    [N] This option is intended to help
       those
27                                     who are designing Graphical User
       Interfaces
28                                     to the EMBOSS applications. Some
       EMBOSS
29                                     programs are inappropriate for
       running in a
30                                     GUI, these include other menu
       programs and
31                                     interactive editors. This option
       allows you
32                                     to only report those programs that
        can be
33                                     run from a GUI
34
35     General qualifiers:
36     -help               boolean    Report command line options and
       exit. More
37                                     information on associated and
       general
38                                     qualifiers can be found with -help
        -verbose
```

4.1.4 tmf:EMBOSS 手册

EMBOSS 网站有很多有益的知识，软件的用法，参数的含义以及报错和处理方法，可以查阅。

此外，EMBOSS 有内置的手册 `tfm` 。

```
1 $ tfm wossname
2 Display full documentation for an application
3                                wossname
4
5 Wiki
6
7    The master copies of EMBOSS documentation are available at
8    http://emboss.open-bio.org/wiki/Appdocs on the EMBOSS Wiki.
9
```

```
10    Please help by correcting and extending the Wiki pages.
11
12  Function
13
14    Find programs by keywords in their short description
15
16  Description
17
18   . . . . . .
```

4.2 处理序列

整个教程中，我们将研究 G 蛋白偶联受体中视紫红质家族的成员。当然，基本的原理对于任何你想要分析的序列都是适用的。我们将使用从 EMBL 和 SwissProt 取得的序列，但你也可以使用 EMBOSS 分析文本文档中的序列。

我们将从两个 EMBL 中取得的序列开始，其标识符是 XL23808 和 XLRHODOP；它们分别是非洲爪蟾（*Xenopus laevis*）视紫红质的基因组序列和对应的 cDNA 序列。

你需要告诉 EMBOSS 到何处去读取你想要分析的序列。EMBOSS 可以从文本文件读取序列，也可以直接从序列数据库读取。例子是最好的说明：

4.2.1 从数据库获得序列

EMBOSS 程序可以从各种序列数据库中读取序列，只要这些序列是以 database:entry 的形式被引用的。这种格式被称为 USA（Uniform Sequence Address）。更多关于 USA 的信息可在 EMBOSS 网站上找到。你可以用 `showdb` 程序看到我们为你设置好的数据库。

4.2.1.1 showdb

`showdb` 显示你所在地区可用的数据库，这取决于当地的 EMBOSS 维护者设置了哪些。

```
1  $ showdb
2  Display information on configured databases
3  # Name         Type    Comment
4  # ============= ======== =======
5  taxon          Taxonomy NCBI taxonomy
6  drcat          Resource Data Resource Catalogue
7  chebi          Obo     Chemical Entities of Biological Interest
8  eco            Obo     Evidence code ontology
9  edam           Obo     EMBRACE Data and Methods ontology
10 edam_data      Obo     EMBRACE Data and Methods ontology (data)
```

```
11 edam_format   Obo     EMBRACE Data and Methods ontology (formats)
12 edam_identifier Obo   EMBRACE Data and Methods ontology (identifiers)
13 edam_operation Obo    EMBRACE Data and Methods ontology (operations)
14 edam_topic    Obo     EMBRACE Data and Methods ontology (topics)
15 go            Obo     Gene Ontology
16 go_component  Obo     Gene Ontology (cellular components)
17 go_function   Obo     Gene Ontology (molecular functions)
18 go_process    Obo     Gene Ontology (biological processes)
19 pw            Obo     Pathways ontology
20 ro            Obo     Relations ontology
21 so            Obo     Sequence ontology
22 swo           Obo     Software ontology
```

showdb 给出一个表格，显示数据库的名称，内容和访问方法。

- **ID** 允许程序从数据库中提取一个被精确命名的条目，例如：`embl:x13776`
- **Query** 代表程序可以提取一个符合通配符的条目名称的集合，例如：`swissprot:pax*_human`
- **All** 允许程序连续分析在数据库中的所有条目，例如：`embl:*`

你可以通过标识符（例如，xlrhodop）或者登录号（例如，L07770）来访问 EMBL。

4.2.1.2 seqret

seqret 读入一个序列，并将其写出，实际上相当于 readseq 在 EMBOSS 中的等价物。它可能是最常用的 EMBOSS 程序。

由于未知的原因，从数据库获取序列并不可行，因此后文的数据全部是预先下载的。

```
1 $ seqret
2 Read and write (return) sequences
3 Input (gapped) sequence(s): embl:xlrhodop
4 Output sequence [xlrhop.fasta]:
5 # 在这里会卡住
```

4.2.2 从文件中读入序列

EMBOSS 也可以从文件中读入序列。例如，如果想要将 fasta 文件转换为 gcg 格式的话，可以使用：

```
1 $ seqret xlrhodop_gene.fasta -outseq gcg::xlrhodop_gene.gcg
2 Read and write (return) sequences
3 $ seqret xlrhodop_gene.fasta -outseq xlrhodop_gene.gcg -osformat gcg
```

```
4 Read and write (return) sequences
```

将 GenBank 格式转换为 fasta 格式:

```
1 $ seqret xlrhodop_gene.gb xlrhodop_gene.fasta
2 Read and write (return) sequences
```

4.2.3 获知关于序列的信息

4.2.3.1 infoseq

infoseq 是一个列出序列的 USA、名称、注册号、类型（核酸或蛋白）、长度、GC 百分比（核酸序列）及描述的实用工具。我们可用它看看上述序列的信息。

```
1 $ infoseq xlrhodop_mRNA.gb
2 Display basic information about sequences
3 USA                    Database Name         Accession     Type Length %
     GC  Organism          Description
4 genbank::xlrhodop_mRNA.gb:XELRHODOP -         XELRHODOP     L07770
            N    1684  45.72 Xenopus laevis    Xenopus laevis rhodopsin
     mRNA, complete cds.
```

4.2.3.2 序列注释

序列数据库中不仅仅包含序列，也包括了很多与序列条目有关的信息（注释）。默认情况下使用 seqret ，EMBOSS 将不会向您返回所有这些信息。

为了获得数据库存储的形式的完整条目，你可以使用 entret （并不好用）。应当可以输出 .entret 文件，包含很多信息。在接近底部的序列上方是一个序列特征的列表。

序列特征是指任意有相关特殊描述的区域，我们可以用 showfeat 来给出一个简单的图形化展示。

```
1 $ showfeat xlrhodop_gene.gb
2 Display features of a sequence in pretty format
3 Output file [xlu23808.showfeat]:
4
5 $ cat xlu23808.showfeat
6 XLU23808
7 Xenopus laevis rhodopsin gene, complete cds.
8 |=========================================================| 8914
9 |---------------------------------------------------------> source
10                                |->                      misc_feature
```

```
11                                  >                       misc_feature
12                                  |>                      misc_feature
13                                  >                      misc_feature
14                                  >                      misc_feature
15                                  >                      misc_feature
16                                  >                      misc_feature
17                                  |-->                   mRNA
18                                      |->                 mRNA
19                                              |>            mRNA
20                                              |->          mRNA
21                                                      |---> mRNA
22                                  |-->                    CDS
23                                      |->                 CDS
24                                              |>            CDS
25                                              |->          CDS
26                                                      |>    CDS
```

如欲在获取一个序列时也取得其所有序列特征，可使用 `seqret -feature` 。此处有一些需要注意的特性。

```
1  $ seqret -feature xlrhodop_mRNA.gb
2  Read and write (return) sequences
3  output sequence(s) [xelrhodop.fasta]: xlrhodop_mRNA.fasta
4
5  $ cat xlrhodop_mRNA.gff
6  ##gff-version 3
7  ##sequence-region XELRHODOP 1 1684
8  #!Date 2022-04-11
9  #!Type DNA
10 #!Source-version EMBOSS 6.6.0.0
11 XELRHODOP     EMBL   databank_entry 1     1684   .      +      .
12 ID=XELRHODOP.1;organism=Xenopus laevis;mol_type=mRNA;
13 db_xref=taxon:8355;tissue_type=retina;dev_stage=adult;tissue_lib=
       lambda-ZAPII
14 XELRHODOP     EMBL   CDS    110    1174   .      +      0      I
15 D=XELRHODOP.2;note=gene accession number U23808;
16 codon_start=1;product=rhodopsin;protein_id=AAC42232.1;
17
18 translation=MNGTEGPNFYVPMSNKTGVVRSPFDY
19     PQYYLAEPWQYSALAAYMFLLILLGLPINFMTL
```

```
20      FVTIQHKKLRTPLNYILLNLVFANHFMVLCGFT
21      VTMYTSMHGYFIFGQTGCYIEGFFATLGGEVAL
22      WSLVVLAVERYMVVCKPMANFRFGENHAIMGVA
23      FTWIMALSCAAPPLFGWSRYIPEGMQCSCGVDY
24      YTLKPEVNNESFVIYMFIVHFTIPLIVIFFCYG
25      RLLCTVKEAAAQQQESATTQKAEKEVTRMVVIM
26      VVFFLICWVPYAYVAFYIFTHQGSNFGPVFMTV
27      PAFFAKSSAIYNPVIYIVLNKQFRNCLITTLCC
28      GKNPFGDEDGSSAATSKTEASSVSSSQVSPA
29
30  XELRHODOP      EMBL    sequence_feature      189     1684    .       +
        .       ID=XELRHODOP.3;note=sequenced from clone pXOP71
31  XELRHODOP      EMBL    sequence_variant      1224    1224    .       +
        .       ID=XELRHODOP.4;note=clone pXOP5 contained deletion from bp
        1224-1534
```

这是一个以 GFF（通用特征格式）格式呈现的数据库条目中的序列列表。你可以在 EMBOSS 网站上了解早更多关于特征格式的信息。为了改变特征格式和文件名，我们需要在使用 `seqret -feature` 时添加相关限定符。让我们将特征以 EMBL 格式保存在 `rhodop.features` 文件中。

```
1  $ seqret -feature xlrhodop_mRNA.gb -offormat embl -ofname
       xlrhodop_mRNA.features
2  Read and write (return) sequences
3  output sequence(s) [xelrhodop.fasta]: xlrhodop_mRNA.fasta
4
5  $ cat xlrhodop_mRNA.features
6  FH   Key             Location/Qualifiers
7  FH
8  FT   source          1..1684
9  FT                   /organism="Xenopus laevis"
10 FT                   /mol_type="mRNA"
11 FT                   /db_xref="taxon:8355"
12 FT                   /tissue_type="retina"
13 FT                   /dev_stage="adult"
14 FT                   /tissue_lib="lambda-ZAPII"
15 FT   CDS             110..1174
16 FT                   /note="gene accession number U23808"
17 FT                   /codon_start=1
18 FT                   /product="rhodopsin"
```

```
19 FT                   /protein_id="AAC42232.1"
20 FT                   /translation="
   MNGTEGPNFYVPMSNKTGVVRSPFDYPQYYLAEPWQYSALAAYMF
21 FT
   LLILLGLPINFMTLFVTIQHKKLRTPLNYILLNLVFANHFMVLCGFTVTMYTSMHGYFI
22 FT
   FGQTGCYIEGFFATLGGEVALWSLVVLAVERYMVVCKPMANFRFGENHAIMGVAFTWIM
23 FT
   ALSCAAPPLFGWSRYIPEGMQCSCGVDYYTLKPEVNNESFVIYMFIVHFTIPLIVIFFC
24 FT
   YGRLLCTVKEAAAQQQESATTQKAEKEVTRMVVIMVVFFLICWVPYAYVAFYIFTHQGS
25 FT
   NFGPVFMTVPAFFAKSSAIYNPVIYIVLNKQFRNCLITTLCCGKNPFGDEDGSSAATSK
26 FT                     TEASSVSSSQVSPA"
27 FT   misc_feature    189..1684
28 FT                   /note="sequenced from clone pXOP71"
29 FT   variation       1224
30 FT                   /note="clone pXOP5 contained deletion from bp
   1224-1534"
```

4.2.3.3 更多序列处理软件

coderet 可提取 mRNA 序列、编码区序列、翻译产物蛋白质序列和非编码区序列。

```
1  $ coderet xlrhodop_mRNA.gb
2  Extract CDS, mRNA and translations from feature tables
3  Output file [xelrhodop.coderet]:
4  Coding nucleotide output sequence(s) (optional) [xelrhodop.cds]:
5  Messenger RNA nucleotide output sequence(s) (optional) [xelrhodop.mrna
   ]:
6  Translated coding protein output sequence(s) (optional) [xelrhodop.
   prot]:
7  Non-coding nucleotide output sequence(s) (optional) [xelrhodop.
   noncoding]:
8  Warning: No sequences written to output file 'xelrhodop.mrna'
9
10 $ ls
11 xelrhodop.cds xelrhodop.coderet xelrhodop.mrna xelrhodop.noncoding
   xelrhodop.prot xlrhodop_mRNA.gb
```

extractfeat 提取指定的序列。

```
$ extractfeat xlrhodop_gene.gb xlrhodop_gene.CDS -type "CDS"
Extract features from sequence(s)

$ cat xlrhodop_gene.CDS
>XLU23808_5470_5830 [CDS] Xenopus laevis rhodopsin gene, complete cds.
atgaacggaacagaaggtccaaatttttatgtccccatgtccaacaaaactggggtggta
cgaagcccattcgattaccctcagtattacttagcagagccatggcaatattcagcactg
gctgcttacatgttcctgctcatcctgcttgggttaccaatcaacttcatgaccttgttt
 . . . . . .
>XLU23808_8210_8338 [CDS] Xenopus laevis rhodopsin gene, complete cds.
ttccgtaactgcttgatcaccaccctgtgctgtggaaagaatccattcggtgatgaagat
ggctcctctgcagccacctccaagacagaagcttcttctgtctcttccagccaggtgtct
cctgcataa
```

revseq 将序列转换成其反向互补序列。

```
$ revseq xlrhodop_mRNA.fasta
Reverse and complement a nucleotide sequence
output sequence(s) [xelrhodop.rev]: xlrhodop_mRNA.rev.fasta
```

msbar 对序列进行随机突变。

```
$ msbar xlrhodop_mRNA.fasta xlrhodop_mRNA.mut.fasta
Mutate a sequence
Number of times to perform the mutation operations [1]: 100
Point mutation operations
        0 : None
        1 : Any of the following
        2 : Insertions
        3 : Deletions
        4 : Changes
        5 : Duplications
        6 : Moves
Types of point mutations to perform [0]: 1
Block mutation operations
        0 : None
        1 : Any of the following
        2 : Insertions
        3 : Deletions
```

```
18          4 : Changes
19          5 : Duplications
20          6 : Moves
21  Types of block mutations to perform [0]: 1
22  Codon mutation operations
23          0 : None
24          1 : Any of the following
25          2 : Insertions
26          3 : Deletions
27          4 : Changes
28          5 : Duplications
29          6 : Moves
30  Types of codon mutations to perform [0]: 1
```

不妨画点阵图看一看突变效果：

```
1  $ dottup xlrhodop_mRNA.mut.fasta xlrhodop_mRNA.fasta -graph png -
       goutfile mut_out -gtitle 'xlrhodop_mRNA.mut vs xlrhodop_mRNA' -word
        5
2  Display a wordmatch dotplot of two sequences
3  Created mut_out.1.png
```

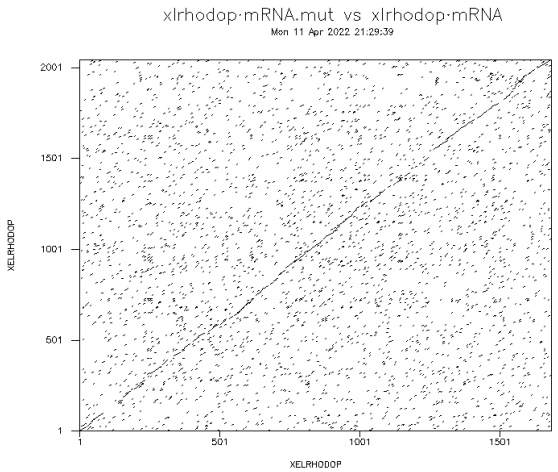


图 4.1:　`msbar` 随机突变序列。

4.2.3.4　序列组分统计

`wordcount` 和 `compseq` 统计字符串出现频率。

`freak` 和 `geecee` 分析 GC 含量，下面分别演示。

```
1 $ freak xlrhodop_mRNA.fasta -letters "GC" -plot Y -graph png -window
    100 -step 10
2 Generate residue/base frequency table or plot
3 Created freak.1.png
```

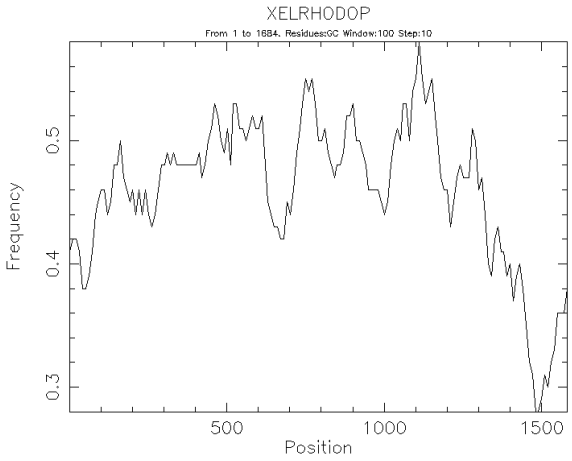


图 4.2: XELRHODOP 序列的 GC 分布。

4.2.3.5　CpG 岛识别

cpgplot 程序预测 DNA 序列中的 CpG 岛。

以人 α 血红蛋白基因簇序列为例:

```
1  $ cpgplot Z84721.FASTA
2  Identify and plot CpG islands in nucleotide sequence(s)
3  Window size [100]: 5000^H
4  Error: Invalid integer value '500'
5  Window size [100]: 500
6  Minimum length of an island [200]: 500
7  Minimum observed/expected [0.6]: 0.65
8  Minimum percentage [50.]: 55
9  Output file [z84721.cpgplot]:
10 Graph type [png]:
11 Features output [z84721.gff]:
12 Created cpgplot.1.png
```

4.2.3.6　重复序列寻找

palindrome 寻找端片段回文结构，einverted 寻找较长的倒转重复。

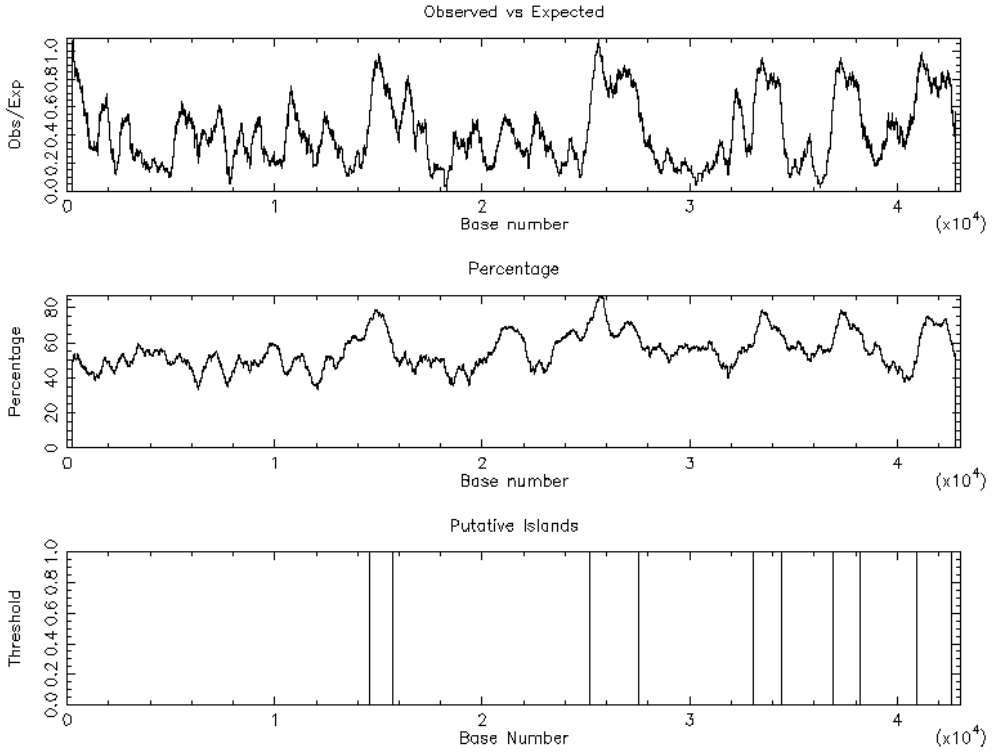


图 4.3: 人 α 血红蛋白基因簇 CpG 岛预测结果。

4.2.3.7　密码子分析

`cusp` 用于统计核酸序列中的 64 种密码子使用频率和期望值。

`chip` 用于计算有效密码子数（ENC），ENC 越小，密码子使用偏好性越强。

4.3　双序列比对

本节是关于序列相似性的。让我们先重温第二章中的一个重要论断：“相似性”并不是一个单一的、精确的或普适的概念。一个比对指的是对这两个序列的一种排列，该排列可以显示两者的相似之处及不同之处。最优比对能显示最显著的相似性，以及最少的不同之处。泛泛而谈，序列比较有三类方法。

- **分段方法**: 将一个序列的所有窗口（一个确定长度的部分重叠片段）依次与另一个序列的所有片段对比，这是在散点图中应用的方法。
- **最优全局比对方法**: 在考虑空位的情况下实现两序列之间比较的最高分数。
- **最优局部比对算法**: 追求最好的局部相似性，然而，不同于分段方法的是，此法考虑了空位。

4.3.1　点阵图

两序列间比较最直观的展示是使用点阵图。序列表示为各个轴，明显配对的区域分布在矩阵的对角线上。

dottup 程序可绘制点阵图：

```
1 $ dottup xlrhodop_gene.gb xlrhodop_mRNA.gb -graph png -goutfile
    dottup_out -gtitle 'xlrhodop_gene vs xlrhodop_mRNA' -word 10
2 Display a wordmatch dotplot of two sequences
3 Created dottup_out.1.png
```

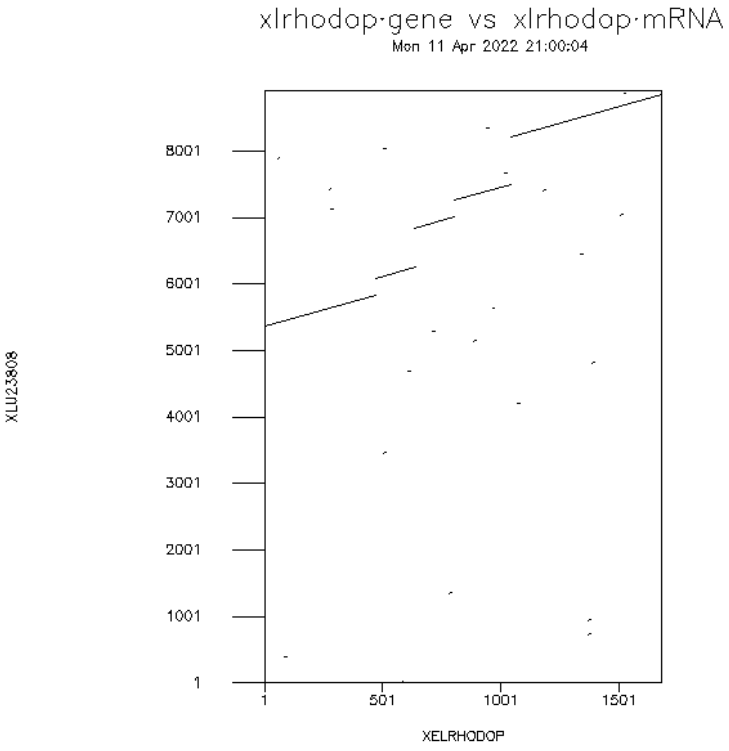


图 4.4: 非洲爪蟾完整基因序列和 cDNA 序列的点阵图。

对角线上的直线代表连个序列完美匹配的区域，这张图中可以看到 5 段对角线。这代表着非洲爪蟾视紫红质基因的 5 个外显子，因为作点阵图的两个文件分别是完整基因序列和 cDNA 序列。

本例中使用的设置是该情况下最佳的。dottup 查找序列之间的精确匹配。由于我们期望基因组序列中的外显子区域与 cDNA 序列完全匹配，我们可以使用更长的字长，即使这样仍然应该得到完全匹配。这给出了一个非常干净的点阵图。

如果要将非洲爪蟾视紫红质 cDNA 序列与相关序列（例如来自小鼠的视紫红质）进行匹配，那么我们不会期望长的完全匹配，因此应该使用较短的字长。

点阵图没有给我们任何详细的序列信息。为此，我们需要使用不同的程序。

下面将介绍的算法比用于搜索数据库的算法更严格；因此，即使您已使用 BLAST 之类的工具从数据库中检索到序列，在之后执行仔细的成对比对也是非常值得的。

序列比对程序背后的基本思想是以产生最高分数的方式比对两个序列——评分矩阵用于将每个匹配的分数添加到每个匹配的分数中，并为每个错配指定扣分。用于核酸比对的矩阵往往具有比较简单的匹配/错配评分方案，而通常用于对蛋白质比对进行评分的矩阵更复杂，

分数旨在反映不同氨基酸之间的相似性，而不是简单地对同一性进行评分。

随着时间的推移，各种突变会按顺序发生；评分矩阵试图处理突变，同时，为了在比对中引入间隙，需要一些额外的参数。

空位打开和空位扩展都会受到惩罚，空位罚分有经验值，但也应该尝试不同的罚分力度。

dotmatcher 可为蛋白质序列绘制点阵图，详见第二章第 1 节。

4.3.2 needle进行全局比对

全局比对在整个长度上比较两个序列的比对，并且适用于全长都具有相似性的序列。比对按照给定的评分矩阵和空位罚分参数使相似区域最大化并使间隙最小化。EMBOSS 程序 needle 是全局比对的 Needleman-Wunsch 算法的一个实现；它的计算是严格的，因此如果序列很长，运行 needle 可能会很耗时。

```
1 $ needle xlrhodop_gene.gb xlrhodop_mRNA.gb -gapo 10 -gape 0.5 -outfile
      xlrhodop.needle
2 Needleman-Wunsch global alignment of two sequences
```

虽然可以找出 5 个配对区域，但这个全局比对是不合理的，输入的两个序列长度差别巨大，因此必须要开很长的空位。

生物学知识告诉我们，内含子/外显子边界有一个保守的 GT-AG 结构限定剪接位点。needle 难以正确比对这些边界，因为它未考虑基因结构模型，它使用的打分方法没有专门处理剪接位点。est2genome 程序对此有一个专门的打分因子，可以更好地比对内含子/外显子边界。

4.3.3 water进行局部比对

water 使用 Smith-Waterman 算法（为了提高速度，并不是严格的实现）来计算序列与一个或多个其他序列的局部比对。它指定了用于计算比对的空位插入罚分、空位延伸罚分和替换矩阵。输出是一个标准的 EMBOSS 比对文件。

```
1 $ water xlrhodop_gene.gb xlrhodop_mRNA.gb -gapo 10 -gape 0.5 -outfile
     xlrhodop.water
2 Smith-Waterman local alignment of sequences
```

EMBOSS 还包含其他双序列比对程序：stretcher （全局比对程序）和 matcher （局部比对程序）。它们不像 needle 和 water 那样苛刻，因此运行速度也更快。

supermatcher 是为超长序列的局部比对而设计的，在算法实现上甚至也不那么严格。

4.4 蛋白质分析

本节将介绍一些可用于分析蛋白质序列的 EMBOSS 程序。显然，前面介绍的核酸序列的双序列比对方法也可以用到蛋白质序列上。

4.4.1 鉴定 ORF

这一小节会介绍一些将 cDNA 序列转换为蛋白质序列的 EMBOSS 程序。基因结构的预测是一个困难的问题，基因组序列中的内含子/外显子边界识别并不容易。为避免这些难题，目前在练习中均使用 cDNA 序列。首先，我们需要鉴定开放阅读框 (ORF)。使用 EMBOSS 程序 `plotorf` 可以快速显示六种可能读码方式中的 ORF 分布。

4.4.1.1 plotorf

```
1 $ plotorf xlrhodop_mRNA.fasta -graph png -goutfile plotorf
2 Plot potential open reading frames in a nucleotide sequence
3 Created plotorf.1.png
```

输出结果显示如下，包括六种可能读码方式中的开放阅读框分布。

最长的 ORF 在第二种读码方式中，大约位于第 100 位到第 1200 位。为获得翻译的精确开始和结束位置，可使用 EMBOSS 程序 `getorf` 。

4.4.1.2 getorf

```
1 $ getorf -options
2 Find and extract open reading frames (ORFs)
3 Input nucleotide sequence(s): xlrhodop_mRNA.fasta
4 Genetic codes
5        0 : Standard
6        1 : Standard (with alternative initiation codons)
7        2 : Vertebrate Mitochondrial
8        3 : Yeast Mitochondrial
9         . . . . . .
10       22 : Scenedesmus obliquus
11       23 : Thraustochytrium Mitochondrial
12 Code to use [0]:
13 Minimum nucleotide size of ORF to report [30]:
14 Maximum nucleotide size of ORF to report [1000000]:
15 Type of sequence to output
16           0 : Translation of regions between STOP codons
```

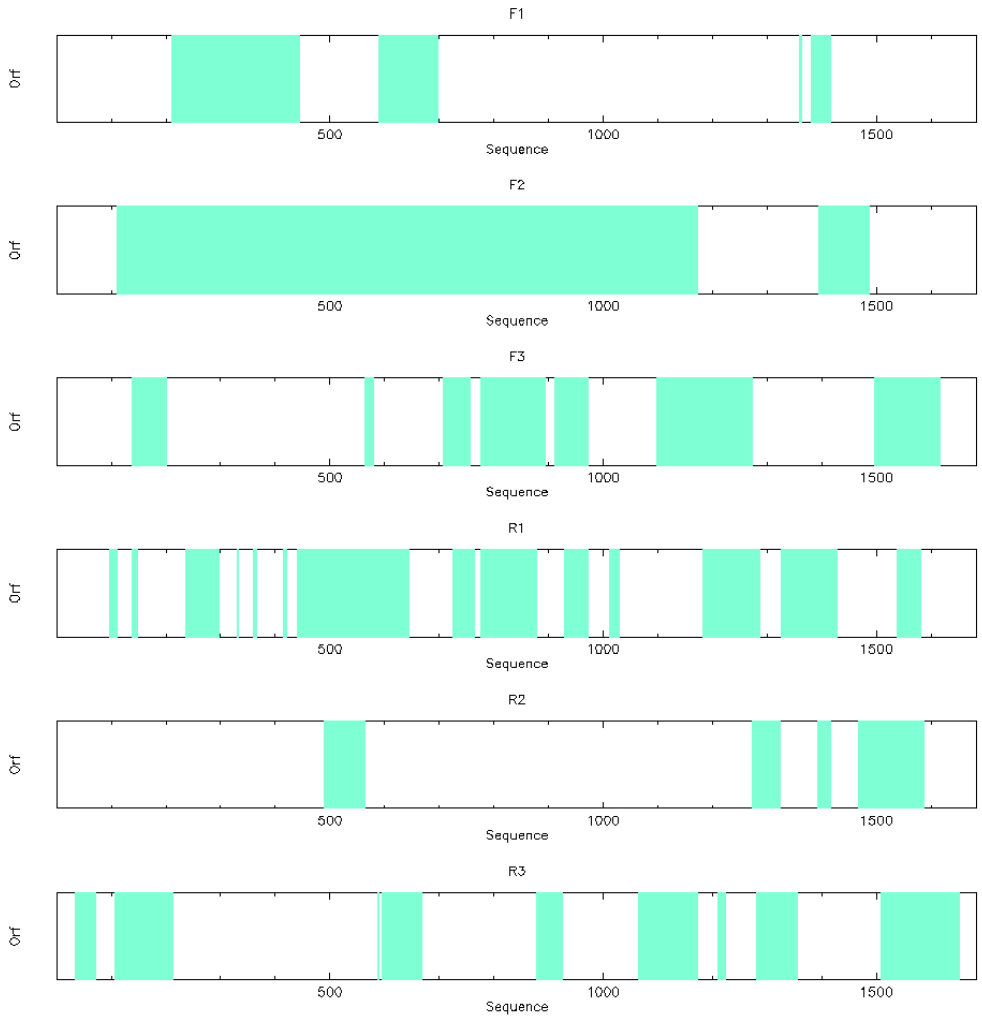


图 4.5:　`plotorf` 鉴定可能的开放阅读框。

```
17            1 : Translation of regions between START and STOP codons
18            2 : Nucleic sequences between STOP codons
19            3 : Nucleic sequences between START and STOP codons
20            4 : Nucleotides flanking START codons
21            5 : Nucleotides flanking initial STOP codons
22            6 : Nucleotides flanking ending STOP codons
23 Type of output [0]: 3
24 protein output sequence(s) [l07770.orf]:
25 (class) leb1c@bbt:~/tutorial$ cat l07770.orf
26 >L07770.1_1 [138 - 200] Xenopus laevis rhodopsin mRNA, complete cds
27 ATGTCCCCATGTCCAACAAAACTGGGGTGGTACGAAGCCCATTCGATTACCCTCAGTATT
28 ACT
29  . . . . . .
30 >L07770.1_7 [110 - 1171] Xenopus laevis rhodopsin mRNA, complete cds
31 ATGAACGGAACAGAAGGTCCAAATTTTTATGTCCCCATGTCCAACAAAACTGGGGTGGTA
32  . . . . . .
33 CTGTGCTGTGGAAAGAATCCATTCGGTGATGAAGATGGCTCCTCTGCAGCCACCTCCAAG
34 ACAGAAGCTTCTTCTGTCTCTTCCAGCCAGGTGTCTCCTGCA
35 >L07770.1_8 [1098 - 1271] Xenopus laevis rhodopsin mRNA, complete cds
36 ATGAAGATGGCTCCTCTGCAGCCACCTCCAAGACAGAAGCTTCTTCTGTCTCTTCCAGCC
37 AGGTGTCTCCTGCATAAGAGCTTCACCAGGGCTGTCTCAGGGTCCGCTGCCTCACACAAT
38 TCCCATCACTTAAGCCCTGTCTACTTGTTGCGAAGGCAAAGAATTCCACAGTTT
39  . . . . . .
40 >L07770.1_29 [71 - 36] (REVERSE SENSE) Xenopus laevis rhodopsin mRNA,
      complete cds
41 ATGCCTTCTGTGTTTCTTTTTTGCCCAAAGGATCCC
```

注意，这里可以选择根据你的序列选择不同生物的遗传密码表（Genetic codes），也可以选择输出信息的形式（Type of sequence to output）。这里只对序列中起始密码子和终止密码子的位置感兴趣，因此选择了输出 Nucleic sequences between START and STOP codons 。

plotorf 仅仅是对 getorf 返回信息的一个图形化显示。可以用 plotorf 粗略查看可能的开放读码框，而后对开放读码框的最大、最小长度进行限制，这样能够让输出的文件更加简单些。从输出的 l07770.orf 中可以看到，具体的 ORF 开始于 110 位，终止于 1171 位。

4.4.1.3 其他 ORF 分析软件

sixpack 输出正链和互补链共计 6 条序列，并给出 6 条 ORF 所编码的氨基酸。

showorf 输出读码框序列和对应氨基酸序列。

4.4.2 翻译序列

前面的练习已经鉴定了 cDNA 序列中需要翻译的区域，现在可以用 transeq 来将其翻译成蛋白质序列作进一步分析了。

用命令行来运行，这里有新的参数 -sbegin 和 -send ，用于指定序列中的某一段区域作为输入。这里，我们指定 xlrhodop.fasta 中的我们认为是编码区域的一段作为输入。

```
1  $ transeq xlrhodop_mRNA.fasta -sbegin 110 -send 1171 -outseq xlrhodop.
       pep
2  Translate nucleic acid sequences
3
4  $ cat xlrhodop.pep
5  >XELRHODOP_1 Xenopus laevis rhodopsin mRNA, complete cds.
6  MNGTEGPNFYVPMSNKTGVVRSPFDYPQYYLAEPWQYSALAAYMFLLILLGLPINFMTLF
7  VTIQHKKLRTPLNYILLNLVFANHFMVLCGFTVTMYTSMHGYFIFGQTGCYIEGFFATLG
8  GEVALWSLVVLAVERYMVVCKPMANFRFGENHAIMGVAFTWIMALSCAAPPLFGWSRYIP
9  EGMQCSCGVDYYTLKPEVNNESFVIYMFIVHFTIPLIVIFFCYGRLLCTVKEAAAQQQES
10 ATTQKAEKEVTRMVVIMVVFFLICWVPYAYVAFYIFTHQGSNFGPVFMTVPAFFAKSSAI
11 YNPVIYIVLNKQFRNCLITTLCCGKNPFGDEDGSSAATSKTEASSVSSSQVSPA
```

4.4.3 二级结构预测

DNA 序列如何决定了特定蛋白的结构？这一问题自提出始即不断引发各种猜测，显示其无尽的魅力。这仍然是一个很困难的领域；一般将其称作“折叠问题”，这是当今分子生物学中主要的前沿问题之一。从序列预测蛋白质三级结构的尝试，大多归于以下两种类型：

- 建立一个接近真实的机制模型，并用它模拟蛋白质折叠过程
- 经验主义地，从已知的三维结构推测未知的蛋白结构

基于机制模型的方法具有天生的吸引力，因其理论上不需要任何关于蛋白结构的先验知识，如果成功，就能立刻被应用到所有的蛋白质序列上。另一方面，任何从已知结构推断未知结构的方法都会不断受到应用范围的限制；这一方法仅仅适合于哪些与已知结构相似的蛋白结构的预测。幸运的是，通常有生物物理或生化线索可以帮助做出这个选择，这些线索通常被整合地运用在结构预测的方法中。

现在，最好的获得合理二级结构预测的方法是运行很多种预测算法，而后寻找结果中的有“共识”的预测。很多服务器能提供这样的多重分析，包括 HGMP 的 PIX，以及 Dundee 大学的 Jpred。(网址也许已经迁移)

目前来看，EMBOSS 内的二级结构预测算法仍然有限。为了运行上述的寻找“共识”的方法，还需要添加新的算法。接下来将介绍一写目前在 EMBOSS 中可用的预测工具。

4.4.3.1 pepinfo

`pepinfo` 产生有关氨基酸性质的信息 (大小，极性，方向性，电荷等)。在定位 trun，可能的抗原肽段和跨膜螺旋时，蛋白的疏水性谱是有用的。

`pepinfo` 采用多种算法，其中包括 Kyte-Doolittle 亲水性评估——该曲线是在 9 个残基的窗口上的残基特异性疏水性指数的平均值。当结果中的线处在框的上半部分时，它代表这部分是疏水的，反之则代表这部分是亲水的。

```
1 $ pepinfo xlrhodop.pep
2 Plot amino acid properties of a protein sequence in parallel.
3 Graph type [png]:
4 Output file [xelrhodop_1.pepinfo]:
5 Created pepinfo.1.png
6 Created pepinfo.2.png
```

图 4.6: 蛋白质序列的基本信息绘图。

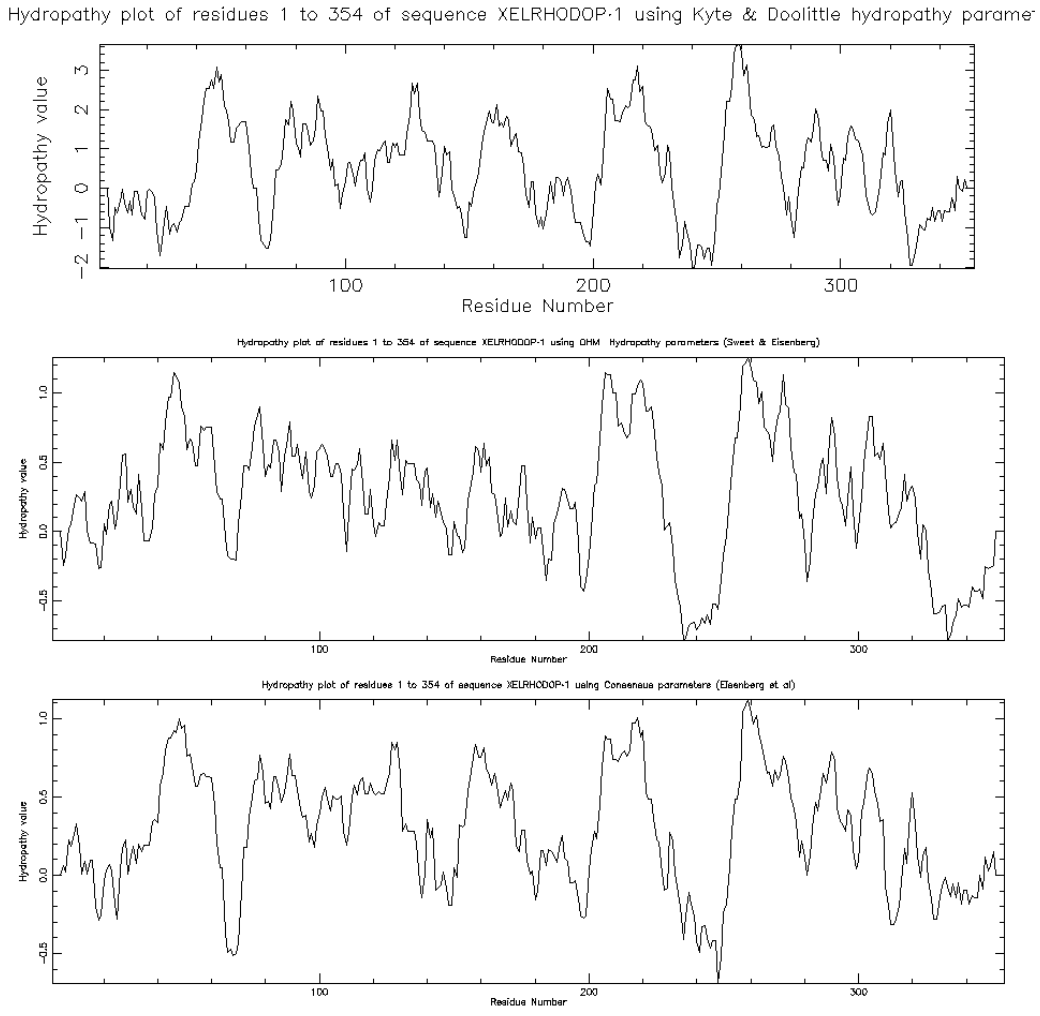


图 4.7: 蛋白质序列的疏水性绘图。

4.4.3.2 garnier预测二级结构

garnier 是用于预测蛋白质二级结构的 Garnier Osguthorpe Robson 算法 (GOR I) 的实现。它读取输入蛋白质序列并写入具有预测二级结构的标准 EMBOSS 报告文件。Garnier 方法不被认为是最准确的预测，但在大多数工作站上很容易计算。

```
$ garnier xlrhodop_mRNA.fasta
Predict protein secondary structure using GOR method
Output report [xelrhodop.garnier]:

$ cat xelrhodop.garnier
########################################
# Program: garnier
# Rundate: Mon 11 Apr 2022 22:25:17
# Commandline: garnier
#    [-sequence] xlrhodop_mRNA.fasta
# Report_format: tagseq
# Report_file: xelrhodop.garnier
########################################

#=======================================
#
# Sequence: XELRHODOP  from: 1  to: 1684
# HitCount: 256
#
# DCH = 0, DCS = 0
#
#  Please cite:
#  Garnier, Osguthorpe and Robson (1978) J. Mol. Biol. 120:97-120
#
#
#=======================================

           .   10    .   20    .   30    .   40    .   50
       ggtagaacagcttcagttgggatcacaggcttctagggatcctttgggca
helix                                                H
sheet     EE    EEEE      EEE     EEE     EEEE
turns T     TTTT    TTTTTT    TTTTT   TTTTTT    TTTT
    coil CCCC              C                      CC
           .   60    .   70    .   80    .   90    .  100
```

```
35         aaaaagaaacacagaaggcattctttctatacaagaaaggactttataga
36  helix HHHHHHHHHHHHHH           HHHHHHHHH
37  sheet                  E       E           EEEEE
38  turns             TTTTTT TTTTTTT        TTT     TTTT
39      coil
40              .  110    .  120    .  130    .  140    .  150
41         gctgctaccatgaacggaacagaaggtccaaatttttatgtccccatgtc
42  helix                H
43  sheet     EEEE        EEE     EEEEEEE
44  turns TTTTT   TTTTTTTT   TTTTTT      TTTTTTTTTTTTTTT
45      coil
46   . . . . . .
```

4.4.3.3　tmap预测跨膜区域

pepinfo 的结果显示在 xlrhodop.pep 中存在 7 个高度疏水的区域。这些区域会不会是跨膜区域呢？可使用 EMBOSS 程序 tmap 来检验这种可能。

```
1  $ tmap xlrhodop.pep
2  Predict and plot transmembrane segments in protein sequences
3  Graph type [png]:
4  Output report [xlrhodop.tmap]:
5  Created tmap.1.png
```

此外 tmap 还输出了文件 xlrhodop.tmap ，内含预测的 7 个可能的跨膜区域的的起止位点和蛋白序列。结合前面 pepinfo 的分析结果，可以推测在这个蛋白中可能含有 7 个跨膜螺旋区域。这一信息与常识吻合，作为一种 GPCR，非洲爪蟾视紫红质含有 7 重跨膜螺旋是符合事实的。

4.4.4　其他的蛋白质分析软件

pepstat 统计 20 种氨基酸出现的频率。
wordcount 分析短片段重复序列的出现频率。
antigenic 可预测抗原决定簇。
fuzzpro 可寻找序列模体。

4.5　模式分析和多序列比对

这一教程并未包含 BLAST 或 FASTA 搜索的内容，因为它们尚不是 EMBOSS 的一部分，世界范围内很多网站提供了这种服务。然而，数据库搜索是生物信息学家军械库的重要组成

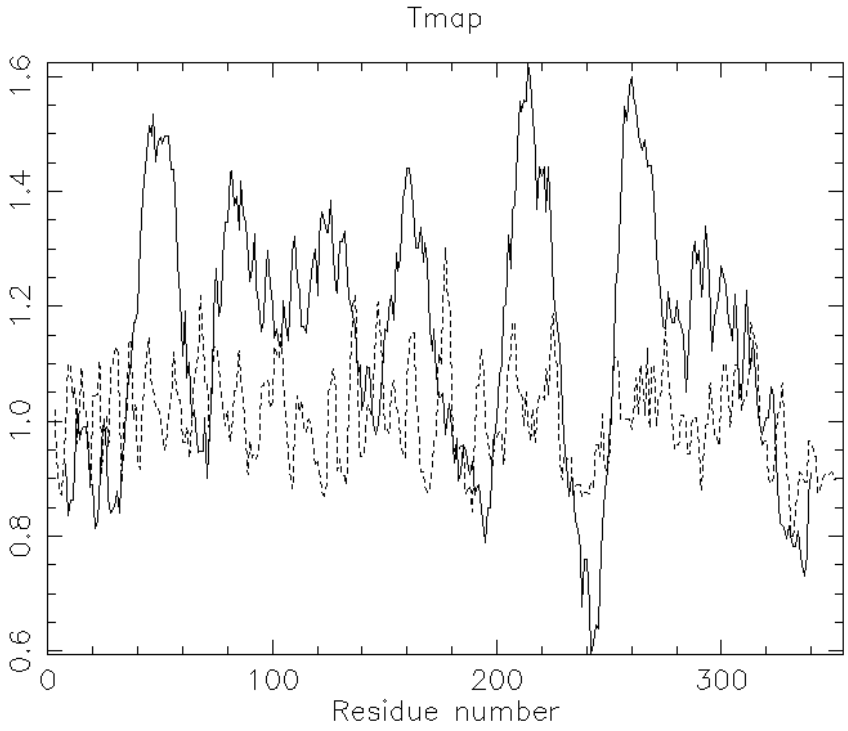


图 4.8: 预测 `xlrhodop.pep` 序列的跨膜区域。

部分。在一个已知序列的数据库中搜索一个新序列时，我们实际上在尝试回答下列问题：

- 是否存在任何与未知蛋白质的序列具有足够的相似性的已知结构的蛋白质，以表明家族关系？
- 如果没有，那么哪个已知蛋白质的序列与未知蛋白质的序列最相似？

如果我们可以确定与已知结构蛋白质的关系，就可以推断出新蛋白质与其相关蛋白质具有共同的结构，也可以据此确定共有的折叠方式。但如果同源物结构也是未知的，又该怎么办呢？如果其功能已被坚定的话，我们可以期待已知序列与其有相似的功能。然而，总有例外。一个经典的例子是溶菌酶，它与 -乳清蛋白具有 50% 的序列一致性与 70% 的序列相似性；二者具有类似的折叠，但其功能完全不同：溶菌酶中的两个关键催化残基在 -乳清蛋白中并不保守；而对于乳清蛋白功能很重要的酸性钙离子结合基序也不存在于大多数溶菌酶中。因此，任何计算预测都必须被湿实验验证。

如果仅靠序列相似性不能令人满意地识别亲缘关系近的序列，应该怎么办？本章将介绍 EMBOSS 中的更多应用程序，它们可以预测序列的功能。

4.5.1 模式匹配

在一些情况下，蛋白的活性位点可以根据一些特定的“指纹”或“模板”（指一小段残基，但它仅仅在某一家族的蛋白中存在）被识别。例如，蛋白基序 GXGXXG，代表着一个 GTP 结合位点。在一个序列中寻找一个预先定义的字符串就是模式匹配。

4.5.1.1 patmatmotifs进行模式搜寻

EMBOSS 程序 patmatmotifs 通过一种模式搜寻算法，在给定的蛋白质序列中查找在 PROSITE 数据库（由 Geneva 大学的 Dr. Amos Bairoch 构建）中定义的模式以寻找基序。

PROSITE 是一个蛋白家族和结构域的数据库，它基于这样一种理念：蛋白虽然多种多样，但都是基于其序列相似性被分到有限的家族中。属于特定家族的蛋白质或蛋白质结构域通常具有相似功能并且源自共同的祖先。

需要首先下载 prosite.dat 和 prosite.doc 文件 (下载地址)，并在其存放目录下运行 prosextract 。但如果 EMBOSS 并非自己安装的，可能会遇到权限不足的问题。

```
$ prosextract
Process the PROSITE motif database for use by patmatmotifs
PROSITE database directory [.]:

$ patmatmotifs xlrhodop.pep -outfile xlrhodop.patmatmotifs
Scan a protein sequence with motifs from the PROSITE database

$ cat xlrhodop.patmatmotifs
########################################
# Program: patmatmotifs
# Rundate: Sun 10 Apr 2022 15:27:06
# Commandline: patmatmotifs
#    [-sequence] xlrhodop.pep
#    -outfile xlrhodop.patmatmotifs
# Report_format: dbmotif
# Report_file: xlrhodop.patmatmotifs
########################################

#=======================================
#
# Sequence: L07770.1_1  from: 1  to: 354
# HitCount: 2
#
# Full: No
# Prune: Yes
# Data_file: /rd1/home/leb1c/miniconda3/envs/class/bin/../share/EMBOSS
    /data/PROSITE/prosite.lines
#
#=======================================

```

```
Length = 17
Start = position 123 of sequence
End = position 139 of sequence

Motif = G_PROTEIN_RECEP_F1_1

TLGGEVALWSLVVLAVERYMVVCKPMA
        |              |
    123             139

Length = 17
Start = position 290 of sequence
End = position 306 of sequence

Motif = OPSIN

PVFMTVPAFFAKSSAIYNPVIYIVLNK
        |              |
    290             306


#---------------------------------------
#---------------------------------------
```

这个例子中，我们已经事先知道了非洲爪蟾视紫红质的功能。然而在未知功能的蛋白研究中，patmatmotifs 给出的结果也许能指导未来的研究方向。

4.5.1.2　报告格式

许多 EMBOSS 程序都会生成报告作为输出，具有各种可选格式。继续前面的例子，如果需要一个简单列表文件，而不是在前面的练习中看到的图形显示，可以使用 -rformat 选项指定报告格式。

```
$ patmatmotifs xlrhodop.pep -outfile xlrhodop.patmatmotifs -rformat
   listfile
Scan a protein sequence with motifs from the PROSITE database

$ cat xlrhodop.patmatmotifs
########################################
# Program: patmatmotifs
```

```
# Rundate: Sun 10 Apr 2022 15:34:03
# Commandline: patmatmotifs
#    [-sequence] xlrhodop.pep
#    -outfile xlrhodop.patmatmotifs
#    -rformat2 listfile
# Report_format: listfile
# Report_file: xlrhodop.patmatmotifs
#######################################

#=======================================
#
# Sequence: fasta::xlrhodop.pep:L07770.1_1 from: 1 to: 354
# HitCount: 2
#
# Full: No
# Prune: Yes
# Data_file: /rd1/home/leb1c/miniconda3/envs/class/bin/../share/EMBOSS
    /data/PROSITE/prosite.lines
#
#=======================================

fasta::xlrhodop.pep:L07770.1_1[123:139]
fasta::xlrhodop.pep:L07770.1_1[290:306]

#---------------------------------------
#---------------------------------------
```

由于给出了序列起止位置，可以很方便地用 `seqret` 截取这几段序列。其他的可用格式参考 EMBOSS 网页。

4.5.2　蛋白质指纹

PRINTS 是一个定义功能性蛋白质家族的数据库，通过一些短的、特别保守的序列来识别每个结构域。以正确的顺序匹配所有相关的短序列（“指纹”）的序列，记录完全匹配。如果“指纹”缺少一些部分或指纹出现的顺序不正确，记录部分匹配。

4.5.2.1　`pscan` 搜索蛋白质指纹

使用 EMBOSS 中提供的 `pscan` 程序搜索 PRINTS 数据库。

需要首先下载 `prints41_1.dat` 文件 (下载地址)，并在其存放目录下运行 `printsextract`

。但如果 EMBOSS 并非自己安装的，可能会遇到权限不足的问题。

```
$ printsextract
Extract data from PRINTS database for use by pscan
PRINTS database file: prints41_1.dat

$ pscan xlrhodop.pep -outfile xlrhodop.pscan
Scan protein sequence(s) with fingerprints from the PRINTS database
Minimum number of elements per fingerprint [2]:
Maximum number of elements per fingerprint [20]:

$ cat xlrhodop.pscan
CLASS 1
Fingerprints with all elements in order

Fingerprint RHODOPSIN Elements 6
    Accession number PR00579
    Rhodopsin signature
    Element 1 Threshold 80% Score 100%
              Start position 3 Length 19
    Element 2 Threshold 76% Score 94%
              Start position 22 Length 17
    Element 3 Threshold 53% Score 90%
              Start position 85 Length 17
    Element 4 Threshold 71% Score 100%
              Start position 191 Length 17
    Element 5 Threshold 56% Score 97%
              Start position 271 Length 19
    Element 6 Threshold 81% Score 95%
              Start position 319 Length 14


CLASS 2
All elements match but not all in the correct order
 . . . . . .
```

结果提示，这里的序列属于视紫红质家族，这是正确的判断。

4.5.3 多序列分析

多个核苷酸或氨基酸序列的比对是分子生物学中的重要工具。多重比对用于：寻找判断模式来表征蛋白质家族；检测或证明新序列与现有序列家族之间的同源性；帮助预测新序列的二级和三级结构；为用于 PCR 的寡核苷酸引物提供参考；分子进化分析。

最流行的多序列比对程序之一是 clustalw。EMBOSS 有一个 `clustal` 接口，`emma` clustal（简写为 `emma` ），它使用逐步的成对比对从一组相关序列中创建多序列比对。它还可以生成一个树状图，显示用于生成比对的聚类关系。

树状图显示了成对序列比对和序列聚类的顺序，它们共同产生最终的比对；尽管分支的长度与序列的相对距离有关，但树状图并非一棵进化树。

`clustal` 寻找全局最优比对。比对过程从两个最相似序列的成对比对开始，产生两个比对序列的簇。然后可以将该簇与下一个最相关的序列或比对序列簇进行比对。两个序列簇的比对可以通过两个单独序列的成对比对的简单扩展来实现。最终的比对是通过一系列渐进的成对比对来得到的，这些比对包括越来越不同的序列和簇，直到所有序列都包含在最终的成对比对中。当将空位插入序列中以产生比对时，它们将插入到簇的所有序列中的相同位置。

下面尝试进行多序列比对。

4.5.3.1 `emma` 进行多序列比对

使用 9 个人源血红蛋白进行演示：

```
$ emma
Multiple sequence alignment (ClustalW wrapper)
Input (gapped) sequence(s): 9HUMAN_HB.FASTA
(aligned) output sequence set [hbaz_human.aln]:
Dendrogram (tree file) from clustalw output file [hbaz_human.dnd]:

 CLUSTAL 2.1 Multiple Sequence Alignments

Sequence type explicitly set to Protein
Sequence format is Pearson
Sequence 1: HBAZ_HUMAN 142 aa
Sequence 2: HBM_HUMAN 141 aa
 . . . . . .
Sequence 9: HBE_HUMAN 147 aa
Start of Pairwise alignments
Aligning...

Sequences (1:2) Aligned. Score: 47
Sequences (1:3) Aligned. Score: 59
```

```
20
21  . . . . . .
22 Sequences (7:8) Aligned. Score: 99
23 Sequences (7:9) Aligned. Score: 80
24 Sequences (8:9) Aligned. Score: 79
25 Guide tree file created: [00013467C]
26
27 There are 8 groups
28 Start of Multiple Alignment
29
30 Aligning...
31 Group 1: Sequences: 2    Score:2044
32 Group 2: Sequences: 3    Score:1205
33  . . . . . .
34 Group 8: Sequences: 9    Score:903
35 Alignment Score 16402
36
37 GCG-Alignment file created  [00013467B]
```

不妨查看一下以 fasta 格式显示的比对结果：

```
1  >HBA_HUMAN
2  -MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF------DLSHGS
3  AQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAH
4  LPAEFTPAVHASLDKFLASVSTVLTSKYR
5  >HBAT_HUMAN
6  -MALSAEDRALVRALWKKLGSNVGVYTTEALERTFLAFPATKTYFSHL------DLSPGS
7  SQVRAHGQKVADALSLAVERLDDLPHALSALSHLHACQLRVDPASFQLLGHCLLVTLARH
8  YPGDFSPALQASLDKFLSHVISALVSEYR
9  . . . . . .
10 >HBE_HUMAN
11 MVHFTAEEKAAVTSLWSKMN--VEEAGGEALGRLLVVYPWTQRFFDSFGNLSSPSAILGN
12 PKVKAHGKKVLTSFGDAIKNMDNLKPAFAKLSELHCDKLHVDPENFKLLGNVMVIILATH
13 FGKEFTPEVQAAWQKLVSAVAIALAHKYH
```

emma 是一种全局比对，因此需要将所有序列补齐至同一长度。

emma 的结果并不容易看出错配，可使用 prettyplot 来更好地显示结果。

4.5.3.2 prettyplot 为多序列比对绘图

```
1 $ prettyplot hbaz_human.aln
2 Draw a sequence alignment with pretty formatting
3 Graph type [png]:
4 Created prettyplot.1.png
```

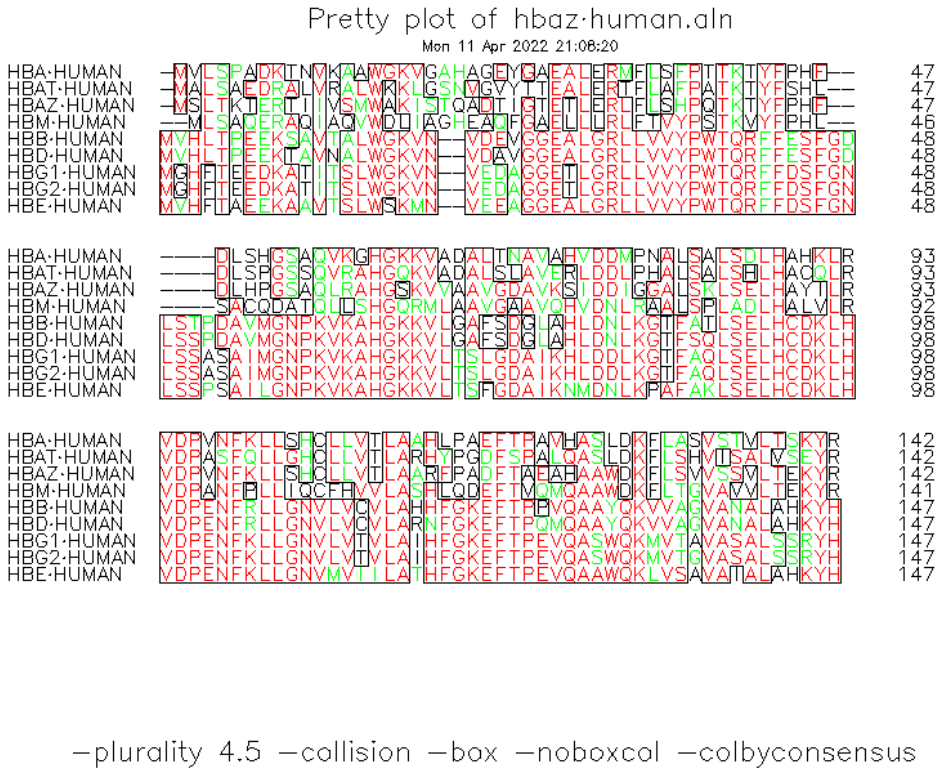


图 4.9: 9 个人血红蛋白的多序列比对。一致的残基用红色表示，相似的残基用绿色表示。

4.5.3.3　其他多序列比对软件

`edialign` 采用全局比对加局部比对的方法，适合寻找具有局部相似性的区域。

```
1 $ edialign 12HUMAN_CEA.FASTA 12HUMAN_CEA.EDIA 12HUMAN_CEA.ALN
2 Local multiple alignment of sequences
```

4.5.4　谱分析

谱分析 (Profile analysis) 是一种表征序列的推定结构和功能强大的技术，本质上是一种用于查找和比对远相关序列的序列比较方法。

这种比较允许将新序列与相似序列家族进行最优化比对，它同时利用评分矩阵和多个类似蛋白质序列的既有比对结果。

属于某一“家族”的类似序列首先被比对在一起以创建多序列比对，然后将多序列比对中的信息表示为由位置特异的符号比较值和空位罚分值组成的表格。

这个表格称为**谱** (Profile)。基于一个谱，新的序列的相似性能够以这样的方式被检验：使用改进的 Smith-Waterman 算法将每个新序列和存在的谱进行比较。

4.5.4.1 prophecy 从多重比对中创建谱

EMBOSS 程序 prophecy 可以从一个多重比对中创建一个谱，下面以 hbaz_human.aln 作为演示的例子。

```
$ prophecy hbaz_human.aln
Create frequency matrix or profile from a multiple alignment
Profile type
          F : Frequency
          G : Gribskov
          H : Henikoff
Select type [F]: g
Scoring matrix [Epprofile]:
Enter a name for the profile [mymatrix]: HB_sequences
Gap opening penalty [3.0]:
Gap extension penalty [0.3]:
Output file [hbaz_human.prophecy]:
```

4.5.4.2 prophet 基于谱进行比对

下面用 prophet ，基于 prophecy 生成的谱，将 HBA_MOUSE.FASTA 比对到人源的血红蛋白组。

```
$ prophet HBA_MOUSE.FASTA
Scan one or more sequences with a Gribskov or Henikoff profile
Profile or weight matrix file: hbaz_human.prophecy
Gap opening coefficient [1.0]:
Gap extension coefficient [1.0]:
Output alignment [hba_mouse.prophet]:

$ cat hba_mouse.prophet
########################################
# Program: prophet
# Rundate: Sun 10 Apr 2022 16:59:34
# Commandline: prophet
#    [-sequence] HBA_MOUSE.FASTA
#    -infile hbaz_human.prophecy
```

```
# Align_format: simple
# Report_file: hba_mouse.prophet
########################################

#=======================================
#
# Aligned_sequences: 2
# 1: HB_sequences
# 2: HBA_MOUSE
# Matrix: EBLOSUM62
#
# Length: 149
# Identity:    71/149 (47.7%)
# Similarity:  92/149 (61.7%)
# Gaps:         7/149 ( 4.7%)
# Score: 1340.43
#
#
#=======================================

HB_sequences     1 MMHLTAEEKAAVTALWGKVNAHVDEAGGEALGRLLVVYPWTQRFFDSFGN
    50
                   |: |:.|:|:.:.|.|||:..|..|.|.|||.|:...:|.|:.:|..|
HBA_MOUSE        1 MV-LSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF--
    47

HB_sequences    51 LSSPDAIMGNPKVKAHGKKVLTALGDAVKHLDDLKGAFSQLSELHCDKLH
    100
                       |...|:.:||.|||||..||..|..|||||.||.|.||:||..||.
HBA_MOUSE       48 ----DVSHGSAQVKGHGKKVADALASAAGHLDDLPGALSALSDLHAHKLR
    93

HB_sequences   101 VDPENFKLLGNVLVVVLAAHFGKEFTPEVQAAWQKFVAGVASALASKYH
    149
                      |||.|||||.:.:|:|.||:|...:|||.|.|:..||:|.|::.|.|||.
HBA_MOUSE       94 VDPVNFKLLSHCLLVTLASHHPADFTPAVHASLDKFLASVSTVLTSKYR
    142

```

```
48 #---------------------------------------
49 #---------------------------------------
```

| 代表 *HBA_MOUSE* 与 *9HBA_HUMAN* 的共同序列一致的残基，：代表保守的替换。一个家族内序列的比对可能显示对结构或功能起重要作用的保守性区域。

参考文献

[1] Rice, P., Longden, I., & Bleasby, A. (2000). EMBOSS: the European Molecular Biology Open Software Suite. *Trends in genetics : TIG*, 16(6), 276–277.

[2] EMBOSS Tutorial(Online)

[3] 罗静初.(2021).EMBOSS 软件包序列分析程序应用实例. 生物信息学 (01),1-25.

[4] 罗静初.(2021).EMBOSS 和 EMBnet. 生物信息学 (04),223-231.

第五章　BLAST 原理与实践

BLAST 是基本局部比对搜索工具 (Basic Local Alignment Search Tool) 的缩写，是 Smith-Waterman 算法的启发性近似。

BLAST 的目标是快速的序列比较，它的基本思想是：直接获得使局部相似性最高的比对，而局部相似性的度量方式为最大片段对分数 (Maximal Segment Pair Score, MSP Score)。MSP 也称高分片段对 (High-score Segment Pair, HSP)。

本章首先介绍结合两篇原始文献，简略介绍 BLAST 原理及其统计；而后用接近实践的例子，说明 BLAST 的启发性方法；最后分别在命令行和网页执行了 BLAST 程序。本章的基础是第三章：序列比对的基本方法。

这一章的内容，曾与罗老师多次讨论过，相比于个人总结 3 有些补充。

BLAST 的理论由数学家证明，其软件为程序员设计，但却最常由生物学研究者使用。了解一些算法背后的数学原理，对于使用者而言，当然有好处：可以增进对于工具原理的理解，启发人们创造、改进或更好地使用工具。但，除此之外的意义，我总觉得寥寥。对于那些，追求严格的数学证明的读者，我更推荐参考文献 [10]，而非这里的简单介绍。

5.1　BLAST: 1990 年

1990 年，David J. Lipman 等在 *Journal of Molecular Biology* 以 *Basic Local Alignment Search Tool* 为题首次发表 BLAST，设计了一种新的启发式序列相似性搜索算法，是 BLAST 的最早文献。

5.1.1　启发式算法

第三章所述的双序列比对方法 (Needleman-Wunsch 或 Smith-Waterman 算法)（可参考第一组第三次讨论报告），为插入、缺失和替换赋分，而后计算出对应于最小扣分代价的序列比对。

它们的基本思想是最小化两个被比对序列之间的进化距离，或者最大化两个序列之间的相似性。这些算法的主要开销在于计算相似性。它们输出的结果在给定的打分规则下一定是最优的。

然而，因为需要的计算资源太大了，这类方法对于搜索大型数据库而言是不可行的。

启发式算法 (Heuristic Algorithms) 是相对于最优化算法而言的，它根据一些直觉或经验，在可接受的开销下给出待解决问题的可行解，该可行解并不一定最优。贪心算法就是一种启

发式算法。

启发式算法中，相似性的衡量标准已经不是显式定义的最小扣分代价，而是隐含于算法本身。

早期采用启发式算法的程序包括 FASTP：该方法先根据连两个序列之间的**一致性部分**（identities）来找出局部相似性区域（这一过程是不考虑缺口的，一致性部分要求完全的相同，因此不需要矩阵），而后再对这些区域按相似性（similarities）进行加分（根据打分矩阵）。

显然，这种方法估计最小进化距离是很不直接的，但这并不妨碍它的广泛运用，主要原因是**快**。

5.1.2 最大片段对（MSP）度量

序列数据库的搜索一般是应用局部比对方法（而非全局比对方法），这具有生物学的理由：cDNA 序列只会与基因的某个局部匹配，远缘的蛋白质只会在特殊位置附近（例如活性位点）有局部的相似性。

相似性的度量需要一个预先定义好的替换矩阵，BLAST 也不例外。**但 HMMER 在大多数时候不需要**，因其不直接度量相似性。

对于氨基酸序列，原始文献中使用的是 PAM-120 矩阵，实际操作中，矩阵可以由用户选择。

先明确一些概念：

- 序列片段（sequence segment）指一段任意长度的连续的残基串。
- 两个等长序列片段的比对分数是每一个比对残基的相似性打分的和。（打分由矩阵指定）
- **最大片段对**（MSP）是从两个序列中选出的相等长度的片段对中打分最高的一对。MSP 的边界总是使其分数最高，故 MSP 可以是任意长度。
- 当一个片段对的得分既不能因为将其向周边扩展而提高，也不能因为将其缩短而提高时，我们定义它为**局部最大片段对**（local maximal segment pair）。

BLAST 可以找出所有得分高于某一 cutoff 的所有局部 MSP。

MSP score 的计算可以利用动态规划算法在线性时间内解决，与 MSP 的长度成正比。MSP 度量的优点之一是，在 BLAST 发表前，已有文章提供了在合适的随机序列模型下的 MSP Score 统计显著性估测方法，数学处理容易。MSP 度量的优点之二是，对于任何打分矩阵都适用。

5.1.3 MSP Score 的快速估计

在一个数据库中搜索某一查询序列的同源物，得到的结果肯定是有限的，且不会太多（也许现在已经够多了）。应当设置一个 MSP Score 的 cutoff，S，这样保证得到的目标序列与查询序列的相似性高于一个临界值。

高度相似的序列，固然可据此推断其同源性；相似性接近临界值的序列也并非无意义，它们可能与查询序列有一些有趣的生物学上的关系，尽管并不一定是同源性。

为了加快数据库搜索的速度，BLAST 的主要策略是设置一个固定的词对（word pair）长度 w，只寻找得分大于阈值（threshold）T 的长度为 w 的词对，找到的称为 hit。然后，对于每个 hit 向其周边扩展，检查其是否属于一个 $Score > S$ 的片段对。

显然，T 越低，一个 $Score \geq S$ 的片段对就**更有可能**包含一个 $Score \geq T$ 的词对。说的更明白些，就更有可能找到潜在的同源序列。但是同时会增加找到的 hit 数，并增加算法的执行时间。T 的选择要考虑这两方面的平衡。

5.1.4 BLAST 的步骤

1990 年发表的 BLAST，其基本步骤包括：

- 根据查询序列，编纂一个高分词表
- 用高分词表扫描数据库，找到得分高于阈值的 hits
- 扩展这些 hits，得到得分高于 cutoff 的局部最大片段对

这些步骤在搜索 DNA 和蛋白质序列时有细节上的不同，下面分步论述：

5.1.4.1 蛋白质序列 BLAST Step 1：编纂词表

一个查询序列可以按给定的词长 w 划分为很多词，记这个集合为 Q。(蛋白质序列的 w 一般取 3) 根据 20 种氨基酸残基的全排列，很容易生成所有的 w-mer（mer 的含义为 monomerie unit，单体单元，w-mer 代表该序列由 w 个单体组成），记这个集合为 A。

并不需要，也不应该在数据库中搜索 A 中的所有词。因此 BLAST 的做法是根据一个打分矩阵（如 PAM-120），从 A 中找出，能够与 Q 中的任意一个词组成词对，并且该词对得分高于 T 的所有词，放入集合 H。

H 就是高分词表，下一步只需要用 H 中的词去搜索数据库即可，这大大减少了计算量。获得 H 的过程，经过算法的优化，可以在线性时间复杂度下实现。这里有两个问题：
1. 是否有可能，一个 Q 中的词 q，在 A 中没有高分对应？是有可能的，有人或许争辩：q 本身必然存在于 A 中，q 与自身对应，肯定是高分的。但是，打分的依据是打分矩阵，如果你的矩阵比较奇怪，使得 $q-q$ 的词对打分也低于阈值 T，那可能出现没有高分对应的情况。
2. w 和 T 应该如何确定？好的取值大约使 Q 集合的元素个数 $=$ 查询序列碱基数 $\times 50$

5.1.4.2 蛋白质序列 BLAST Step 2: 扫描数据库

问题的本质是在一个长序列中找到一个短序列的所有出现位置，这里有两种方式。

方法一：假设 $w = 4$，我们将每个词映射到一个 $[1, 20^4]$ 中的整数，一个词就可被视为一个长度为 $20^4 = 160000$ 的嵌套数组的下标。这个嵌套数组的每个元素都是一个数组。它的第 i 个元素中存放着查询序列中第 i 个词出现的所有记录。扫描数据库，数据库的每个词被立即指向对应的 hits。(未必真要这样死板，散列也是好的)

方法二：使用确定有限状态自动机 (deterministic finite automaton)，特点是快，大约能达到 500,000 残基/秒。内容涉及较深的计算理论，略，有兴趣的读者可参考周益民, 陈文宇, & 程伟. (2018). *有限自动机理论*. 电子科技大学出版社.。

5.1.4.3 蛋白质序列 BLAST Step 3: Hits 的延伸

Hits 延伸的目标是找出局部 MSP。延伸是双向的，但为了节约时间，在某一方向的延伸导致整体分数相对短片段有一定水平的下降以后，延伸就会终止。这样找出的局部 MSP 并不准确，但这些准确的牺牲在速度面前无关紧要。

5.1.4.4 DNA 序列 BLAST 的特殊之处

DNA 序列作 BLAST 的 w 可以设得稍大，一般设定为 12。

DNA 序列数据库可以将 4 个核苷酸压缩进 1 个 byte 中，以 byte 为单位搜索数据库可以快 4 倍，典型速度为 2,000,000 碱基/秒。DNA 序列是高度非随机的，有些地方是高度重复的或者碱基偏倚的，因此搜索工具也该考虑对策。

在构建压缩版本的序列数据库时，程序会找出并存储那些出现次数大大超过预期的词，并从查询序列词表中过滤掉这些词。在搜索普通数据库时，程序会搜索一个重复序列的子库，如果发现查询序列的某些位置明显匹配，则会从搜索词表中删除来自这些位置的词。

5.1.5 衡量 BLAST 的表现

5.1.5.1 BLAST 在随机序列上的表现

MSP Score 的统计显著性由两个参数参与计算：K 和 λ。当两个长度分别为 m 和 n 的序列作比对时，找到一个分数高于或等于 S 的片段对的**概率**是:

$$1 - \mathrm{e}^{-y} \tag{5.1}$$

其中 $y = Kmn\mathrm{e}^{-\lambda S}$

那么，找到 c 个或更多不连续的片段对，它们的分数总和大于或等于 S 的概率是:

$$1 - \mathrm{e}^{-y} \sum_{i=0}^{c-1} \frac{y^i}{i!} \tag{5.2}$$

使用公式 (5.2)，共享多个分离的相似区域的两个序列够被找出来，尽管每一个相似区域的打分都不够公式 (5.1) 的标准。

尽管在比对两个序列时，得到一个 $p - value < 0.001$ 的 MSP 是很罕见的。然而用一条序列搜索一个包含 10000 条序列的数据库时，仅仅是随机地，就会有 10 个满足条件的 MSP 被找到。运用公式 (5.1)，可以将真正找到的 MSP Score 满足要求的结果同偶然出现的结果区分开。

BLAST 算法仅仅把注意力放在那些包含一个长为 w 的，打分大于 T 的词对的片段对。那么就要知道，分数等于一个固定值的片段对包含这样的词对的比例。尚没有数学理论给出一个得分为 S 的片段对**不**包含一个打分大于等于 T 的词对的概率（记为 q），q 也可以理解为 **BLAST 忽略一个真正的 MSP 的概率**。

一种观点认为，q 应该指数地依赖于 MSP Score。MSP 中成对字母出现的频率接近一个 limiting distribution（极值分布），期望的 MSP 长度与其打分成线性关系。因此，一个 MSP

越长，它就越可能包含一个打分高于 T 的词对；随 MSP Score S 的增加，q（不包含的概率）呈指数下降。模拟实验的结果支持这个指数关系。

5.1.5.2 选择词长和阈值参数

按照前面阐述的 BLAST 三个步骤，第三步所需的时间与 hits 数线性地增长，而 hits 数直接依赖于 w 和 T。

通过随机序列的模拟实验，其他条件固定时，找到 hits 数目与 w 负相关，这是符合直觉的。词对长 w 越长，每次搜索的获得关于潜在 MSP 的信息就越多，这将节约 BLAST 在第三步的时间。但是，w 太大也会带来问题，一方面，所有可能的词有 20^w 个，搜索空间很大，其次，按照给定的灵敏度，产生的高分词会指数地增加。这将在第一步耗费更多时间，同时增加内存需求。实践表明，对于**蛋白质序列搜索，最佳的词长是 4**。

阈值 T 越低，BLAST 给出的 MSP Score 估计越准确，但同时会增加算法的执行时间。具体地说，进入 “高分词表” 的门槛变低了，导致高分词表太臃肿，找到的 hits 更多，第二、第三步所需的时间会大大增加。阈值的设定存在灵敏度和执行时间的权衡。

基于模拟实验和算法分析，BLAST 的期望计算时间复杂度大约是:

$$aW + bN + \frac{cNW}{20^w}$$

其中，W 是高分词表的词数，N 是整个数据库的大小（以残基总数计算），a, b, c 是常数。上式第一项代表了词表编纂，第二项代表了扫描数据库，第三项代表了 hits 的延伸过程。W 随 T 的减小而指数增长，随查询序列长度的增加而线性增加。

实践表明，对于**蛋白质序列搜索，最佳的阈值是 17**（这是在随机序列上得到的），进一步降低阈值对寻找同源物没有帮助。

BLAST 在时间和准确度上作出了权衡，详见图 5.1。

5.1.5.3 BLAST 在同源序列上的表现

在特定参数组合下，真实生物学序列数据库搜索的结果可能比预测的好或者坏。其原因是，突变在序列上的出现并不完全符合泊松过程的描述。

泊松过程基本假设包括:

- 一个突变的发生并不影响另一个突变的发生。
- 突变在任何区域发生的概率是相等的
- 两个突变不会同时发生

而事实是，突变有时成簇，存在突变热点。BLAST 在真实数据上仍然保持了它的速度，并相较于之前的方法，例如 FASTP，有更低的假阳性率，且速度快一个数量级。

5.2 BLAST: 1997 年

1997 年，同一团队（Stephen F. Altschul 作为通讯作者）发表了第二版 BLAST，文章以 *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs* 为题

Table 2
The central processing unit time required to execute BLAST as a function of the approximate probability q *of missing an MSP with score* S

q (%)	CPU time (s)			
2	39	25	17	12
5	25	17	12	9
10	17	12	9	7
20	12	9	7	5
S:	44	55	70	90
p-value	1·0	0·8	0·01	10^{-5}

Times are for searching the PIR database (Release 23·0) with a random query sequence of length 250 using a SUN4-280. CPU, central processing unit.

图 5.1: BLAST 时间和准确度的权衡。q 代表了丢掉打分为 S 的 MSP 的概率，p-value 代表了对结果的统计显著性，S 代表了 hits 延伸后的 cutoff。为了尽可能减少 MSP 的丢失，必须付出计算时间的代价。在给定的 q 下，cutoff S 设得越高，结果的统计显著性越好，耗费时间越少。

发表在 *Nucleic Acids Research*，对 BLAST 作出了重要改进，并开发了 PSI-BLAST。

5.2.1 两次命中法

为了提高速度，更改了延伸 hits 的标准。

从前的方式是：如果一个高分词对出现在了序列中，则延伸。这一步骤消耗了大量的计算时间。

新的方式 (两次命中法，"two-hit" method) 是：如果两个高分词对互不重叠地紧邻着出现序列中，且间隔小于 A，则延伸。显然，这样做减少了需要延伸的 hits 的数目，节约了时间；但同时造成了灵敏度的下降。

为了补偿这一灵敏度的下降，必须下调 T 值，以获得足够的 hits。

5.2.2 空位处理能力

第一版 BLAST 并不具备处理空位的能力，因此会找出一堆邻近的短片段，它们分别与查询序列配对，却不能拼合起来（拼合起来后也是显著的）。

改进的核心思想是如果一个 HSP 的分数超过某个 S_g，则在其周边执行一个精确的，带空位的比对计算过程。S_g 在蛋白质序列中一般被设为 22，以保证大约数据库中的每 50 条序列才会启动一次延伸，被延伸的序列只占数据库总数的 2%。

虽然这种带空位的比对是精确的，但仍然不保证是结果是最优的，这与第一版 BLAST 一样。这种比对很耗时间，但是由于需要延伸的序列数目并不多，因此也可以接受。

形象地说，这有些像教育考试的选拔机制。

5.2.3　结果的统计显著性评价

这一点并非 1997 年文献中才提出，在 BLAST 的数学理论建立时，结果的显著性评价已经被考虑了。

5.2.3.1　归一化分数 S'

如第一版 BLAST 所做的，用 s_{ij} 作为比对中一个残基对的打分，这个打分由替换矩阵指定。

按照统计理论，假设一个简单的蛋白质序列模型：所有的氨基酸在所有位置随机出现，一个氨基酸在任意位置处出现的概率都相同，是 P_i。要求两个随机序列进行比较的总分是负值，即

$$\sum_{i,j} P_i P_j s_{ij} < 0$$

给定 P_i 和 s_{ij}，该理论可以给出两个参数 λ、K。按下述公式，将名义得分 (nominal score)S 转换为归一化得分 (normalized score)(S')。

$$S' = \frac{\lambda S - \ln K}{\ln 2} = \log_2 \left(\frac{\mathrm{e}^{\lambda S}}{K} \right)$$

这样，所有的打分系统都能够被一视同仁地比较了，S 是没有单位的，S' 的单位为 bit。

注记：这里的 bit 或许或让人想到蛋白质序列图标的纵轴单位 bit，它们没有直接的关系，但由于它们都在计算的过程中采用了信息熵的概念，故碰巧单位相同。信息论中约定，将以自然对数表述的某种分数（Score）附以单位 nat，而将以 2 为底的对数表示的分数附以单位 bit。这里的 s' 计算时底数为 2，因此单位为 bit。更多内容可阅读参考文献 [11]、[12]。

5.2.3.2　E 值

当比较两个长度为 m，n 的序列进行比对时 (则 $m \times n$ 是搜索空间的大小)，定义一个 HSP 的 E 为：

$$E = \frac{mn}{2^{S'}}$$

E 的含义是，随机出现的，归一化得分大于等于 S' 的，不同于当前 HSP 的，HSP 个数的期望值。

当搜索一个数据库时，n 为数据库的大小（以残基数计算）。

上式很容易写成：$S' = \log_2(\frac{mn}{E})$，据此可以计算为了达到设定的 E 值，一个合格的 HSP 需要多高的归一化分数。

这样定义的公式在考虑空缺的 BLAST 中仍然成立，但是参数 λ 和 K 需要调整。记传统的 BLAST 对应参数为 λ_u、K_u；考虑空缺的 BLAST 对应参数为 λ_g、K_g

5.2.3.3　P 值

按极值分布模型，P 值定义为在完全随机分布模型下，一个局部比对得分不小于当前得分的概率。或者说，表示” 得分不低于当前的比对纯粹是因为偶然性造成的” 这一事件的概率。

极值分布模型下，P **与** E **的关系**为：

$$P = 1 - \mathrm{e}^{-E} = E - \frac{E^2}{2} + \frac{E^3}{6} - \frac{E^4}{24} + \cdots$$

按泰勒展开式：$\mathrm{e}^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$

$$\begin{aligned} P &= 1 - \sum_{n=0}^{\infty} \frac{(-E)^n}{n!} \\ &= 1 - (1 - E + \frac{E^2}{2} - \frac{E^3}{6} + \frac{E^4}{24} - \cdots) \\ &= E - \frac{E^2}{2} + \frac{E^3}{6} - \frac{E^4}{24} + \cdots \end{aligned}$$

忽略高阶项，可见 P 与 E 一般处于同一数量级。值得一提的是 $P = 0.05$ 时，$E = 0.051293$。

5.2.4　迭代 BLAST

基序或 Profile 搜索方法采用多种不同的工具，收集一个基序或 Profile 的集合，而后在一个数据库中搜索其家族成员，这是早已实现的，只是涉及多种软件，比较麻烦。

位置特异性迭代 BLAST(Position-Specific Iterated BLAST) 的核心思想是将第 i 轮找到的显著性的比对收集起来，生成一个位置特异性的打分矩阵，并将其应用到第 $i+1$ 轮的搜索中去，以替代常规 BLAST 所用的 PAM 或 BLOSUM 等矩阵。

PSI-BLAST 与专门的基序搜寻软件相比，或许不够灵敏，然而其快速和易用仍然使其大受欢迎。

5.3　用接近实践的例子来理解启发性

前两节的描述，或许并不容易理解。读者可以深入阅读原始文件，但因为课程的实践导向，我更推荐从实际的例子中学习。为了简单，我们使用核酸序列 BLAST 来演示。

启发性方法的灵魂是经验，并不要求有严格数学公式，如同动物的试错行为一样，在多次尝试中就能习得正确的解决方法，这个过程恐怕没有复杂的数学推导。启发性方法的反面是严格的算法。

BLAST 用到的启发性方法体现在 seed-and-extend(确定种子-和-延伸) 策略上。

BLAST 的问题实际上是一个局部比对问题，考虑如下的简单例子：

```
1 Query Sequence(查询序列):
2 GACAGC
3
```

```
Database Sequence(目标数据库):
ACGGATTCCATAT

Scoring Scheme(打分策略):
Match:1
Mismatch:-1
Gap Insertion:-1
```

如果我们使用第三章所述的 Smith-Waterman 算法，结合上述打分策略，计算辅助矩阵而后回溯，将会得到图 5.2 的结果。

得分最高 (2 分) 的局部比对包括:

```
GA
GA

CA
CA

AC-G
ACAG

ACGG
ACAG
```

对应图 5.2 中的 4 条红线。

该法能够取得最优局部比对，但如第三章所说的，其时间复杂度是二次方的，即与 (数据库总长度 × 查询序列长度) 成正比。Smith-Waterman 算法为了精确性而牺牲了速度，BLAST 则恰好相反。那么这个例子如何使用 BLAST 来寻找局部比对呢?

BLAST 所用到的核心技术（也被 NGS 比对工具应用）的是**构建索引** (Indexation)。

打个比方，如果你要从一本乱序词汇书 (图 5.3(a)) 中查找某个单词，你很可能失败。因为这种书的单词并不按特定顺序排列 (图 5.3(b))，且你的大脑内存无法把书中的所有内容记下来并快速查找。

但如果你获得了该书的一份索引 (图 5.3(c))，那这件事情就异常简单了。索引中的单词按照字母顺序排列，且标注了每个单词出现的页码。

现在回到 BLAST，事实上，BLAST 也是在查字典，这本字典就是要查找的数据库，我们将一个一个的 k-mer 视为“单词”，k 是“字典”中的词长，这里取 3。数据库不是单词书，没有为我们提供索引，所以需要将所有单词的出现位置找到，自行编纂一份数据库的索引。

上面例子中的数据库非常简短: `ACGGATTCCATAT` ，现在为该数据库中的所有 3-mer，编纂索引:

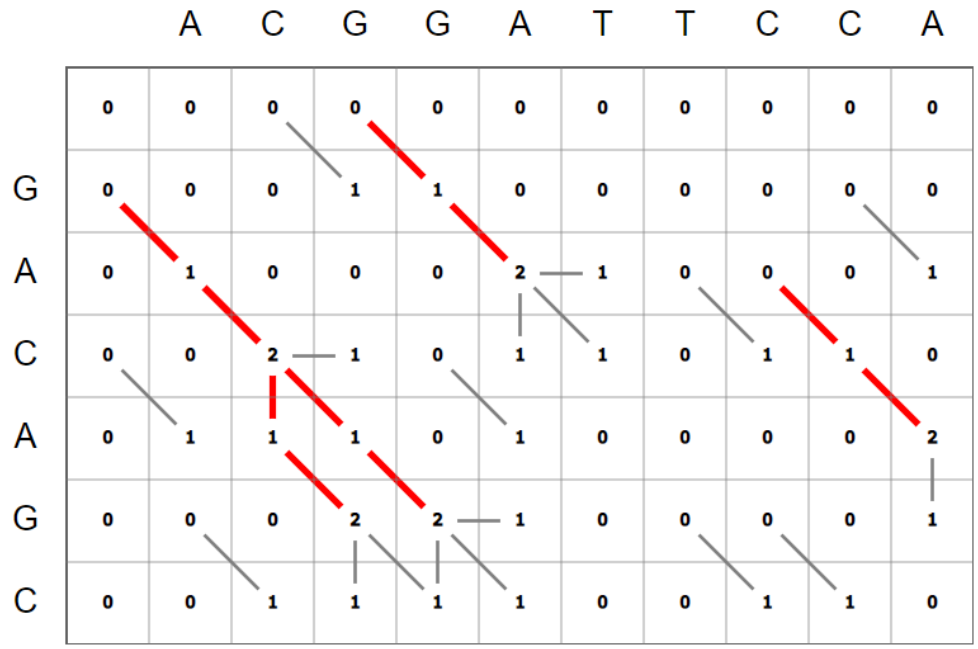


图 5.2: 标准的 Smith-Waterman 局部比对结果，红色线为找出的可能局部比对。

图 5.3: 一本颇流行的乱序词汇书，封面、内页、索引。

```
Database Sequence(目标数据库):
ACGGATTCCATAT

拆分3-mer，并标记其对应位置（编号），共有11个:
ACG.............1
 CGG............2
  GGA...........3
   GAT..........4
    ATT.........5
     TTC........6
      TCC.......7
       CCA......8
        CAT.....9
         ATA....10
          TAT...11
```

我们也将查询序列拆分为 3-mer:

```
Query Sequence(查询序列):
GACAGC

拆分3-mer，并标记其对应位置（编号），共有11个:
GAC.............1
 ACA............2
  CAG...........3
   AGC..........4
```

将查询序列拆出的 3-mer 与数据库序列拆出的 3-mer 进行比较，按找前面的打分规则，进行打分:

共有 $4 \times 11 = 44$ 个词对，这里设定高分词对 (HSSP,High Scoring Segments Pairs) 的阈值为 1，则这 44 个词对中，符合高分词对标准的共有 6 个，在表格中用 * 标出。

进一步总结出高分词对表:

“启发”是一种被相信的经验，BLAST 启发性就在于：我们相信一**个显著的局部比对应该包含高分词对**。

因此，上述高分词对被作为“种子”(seed)，在其附近寻找局部比对，如图 5.4。注意，上面共计找出了 6 个高分词对，但此处仅有 4 个种子，这是为什么?。这是因为种子 1 和种子 3 都是由 2 个高分词对重叠构成的。根据 1997 年版 BLAST 使用的两次命中法，种子 1 和种子 3 都具有两次命中，应该被延伸，而种子 2 和种子 4 都只有一次命中，我们还没有足够的信心确认这些位置有显著的局部比对。

本例中，我们对 4 个种子都进行延伸。延伸使用的是考虑空位的 Smith-Waterman 算法，

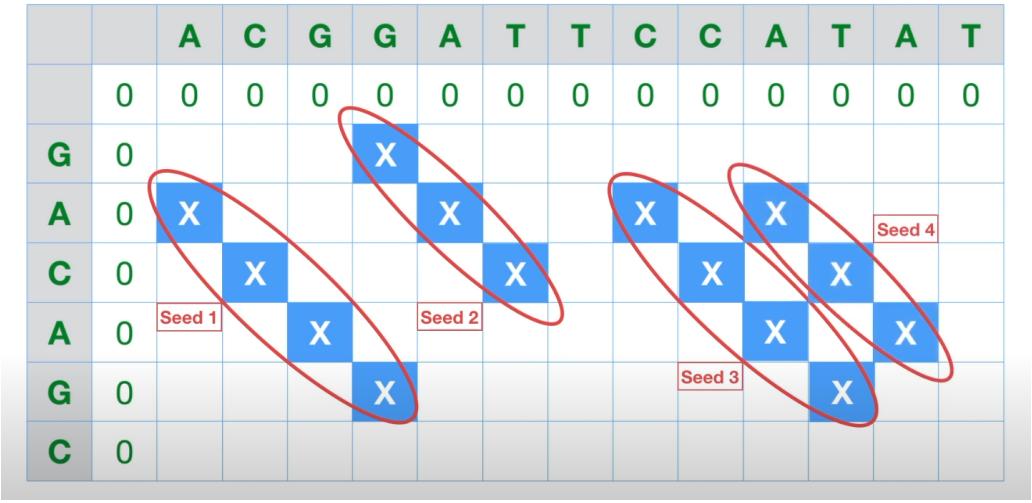
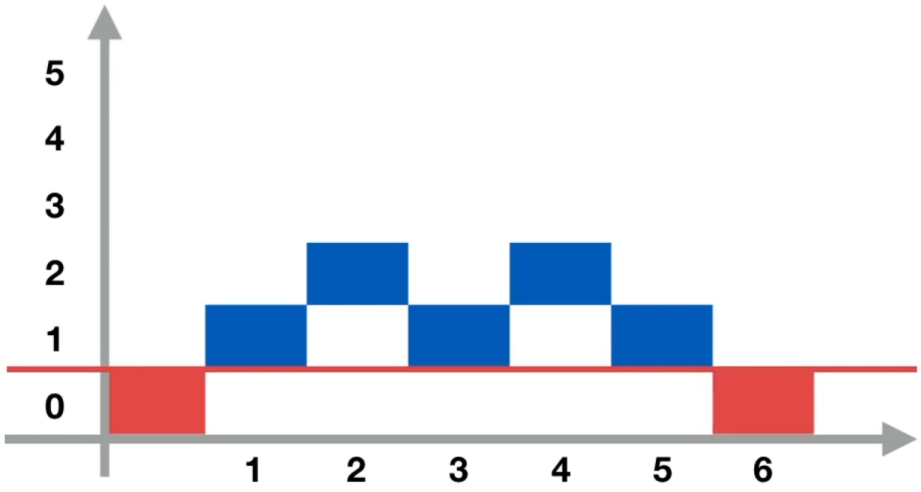


图 5.4: 以 HSSP 作为种子，在其附近寻找局部比对



		A	C	G	G	A	T	T
	0	0	0	0	0	0	0	0
G	0	0	0	1	1	0	0	0
A	0	1	0	0	0	2	1	0
C	0	0	2	1	0	1	1	0
A	0	1	1	1	0	1	0	0
G	0	0	0	2	2	1	0	0
C	0	0	1	1	1	1	0	0

图 5.5: 种子 1 的延伸

表 5.1: 所有词对得分表，按照前述打分策略进行给分。

	GAC (1)	ACA (2)	CAG (3)	AGC (4)
ACG (1)	-3	1*	-1	-1
CGG (2)	-3	-3	1*	-1
GGA (3)	-1	-1	-3	-1
GAT (4)	1*	-3	-1	-3
ATT (5)	-3	-1	-3	-1
TTC (6)	-3	-3	-3	-1
TCC (7)	-1	-3	-3	-1
CCA (8)	-3	1*	-1	-3
CAT (9)	-1	-3	1*	-3
ATA (10)	-3	1*	-3	-1
TAT (11)	-1	-3	-1	-3

表 5.2: 高分词对列表

数据库的 3-mer	序列坐标	数据库坐标
GAT	1	4
ACG	2	1
CCA	2	8
ATA	2	10
CGG	3	2
CAT	3	9

这里设定了一个分数阈值 1，如果延伸导致整个片段的分数低于此阈值，延伸即告终止。(图 5.5)

以种子 1 的延伸为例：延伸从左上方的 0 开始，向右下方扩展，每一步完成后片段的得分展示在上方的坐标系中。当延伸到达右下方的红色方块时，片段分数低于片段分数阈值 1，延伸结束。

其他种子的延伸依此类推，值得注意的是，第 4 个种子无法向周边延伸 (图 5.6)，这说明上述启发性方法并不保证正确，即高分词对的出现也许是偶然。

如图 5.7，将上述延伸结果汇总即可得到对应的局部比对，这就是运行一次 BLAST 后返回的结果，有兴趣的读者可以将其于 Smith-Waterman 算法的结果作一个比较。在上面这个简短的例子里，BLAST 似乎并没有什么优势，但面对巨大的数据库，速度快就是很宝贵的品质。

在一次 BLAST 搜索中，有三个重要参数：词长 (Word Length)、高分词对阈值 (HSSP Threshold)、延伸片段分数阈值 (Extension Threshold)。它们的意义已经在本节和上几节中叙述过了。这里只是提醒，作为使用者，选择合适的参数至关重要。

实际的 BLAST 程序远比上述的例子复杂，如图 5.8 所示，包括对序列的处理、掩蔽重

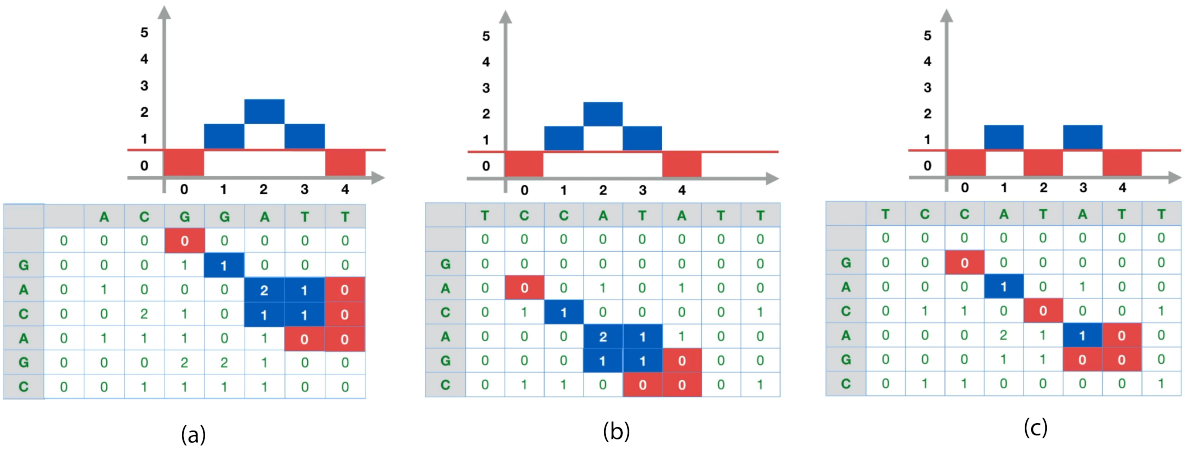


图 5.6: 种子 2、3、4 的延伸，分别对应图 (a)、(b)、(c)

		A	C	G	G	A	T	T	C	C	A	T	A	T	T
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
A	0	1	0	0	0	2	1	0	0	0	1	0	1	0	0
C	0	0	2	1	0	1	1	0	1	1	0	0	0	0	1
A	0	1	1	1	0	1	0	0	0	0	2	1	1	0	0
G	0	0	0	2	2	1	0	0	0	0	1	1	0	0	0
C	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1

A	C	G	G	A		A	C	-	G	-		G	A	T		C	A	T
A	C	A	G	C		A	C	A	G	C		G	A	-		G	A	-
A	C	G	G	A		A	C	-	G	G		G	A	T		C	A	T
A	C	A	G	-		A	C	A	G	C		G	A	C		C	A	G
A	C	G	G	-		A	C	G	G	A		G	A	-		C	A	-
A	C	A	G	C		A	C	A	G	C		G	A	C		C	A	G

图 5.7: 延伸结果汇总 (上) 和对应的局部比对 (下)

复片段、构建搜索树、寻找 hits、回溯、报告等复杂的过程。有兴趣的读者可以阅读参考文献 [8]、[14]、[15]、[16]。

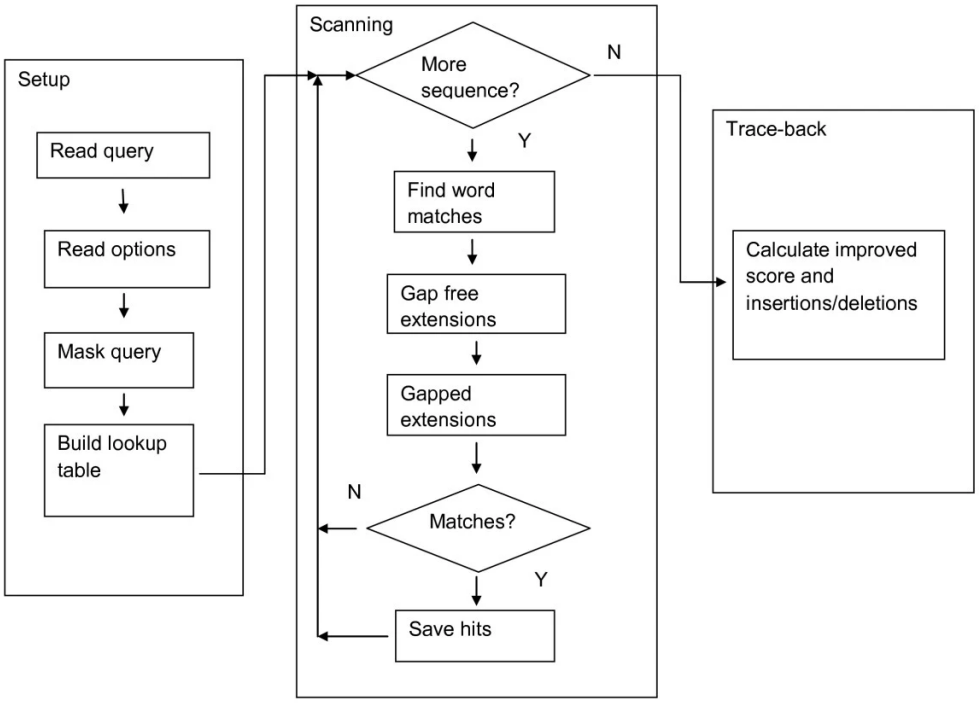


图 5.8: 一次 BLAST 的实际运行流程，引自参考文献 [8]

5.4　本地 BLAST 实践

blast 版本：2.12.0, 安装自 bioconda

5.4.1　数据库的获取或构建

使用者可从 NCBI 下载已经构建好的数据库，这是最常用的方式。

`update_blastdb.pl --showall` 可以查看所有能下载的数据库，这些数据库的含义参考NCBI 的说明。

```
1 $ update_blastdb.pl --showall
2 Connected to AWS
3 16S_ribosomal_RNA
4 18S_fungal_sequences
5  . . .
6 refseq_select_prot
7 refseq_select_rna
8 split-cdd
9 swissprot
```

```
taxdb
tsa_nr
tsa_nt
```

如欲下载指定的数据库，如 swissprot，可运行：

```
update_blastdb.pl --decompress swissprot
```

```
$ mkdir swissprot
$ cd swissprot/
$ update_blastdb.pl --decompress swissprot
Connected to AWS
$ ls
swissprot.pdb swissprot.phr swissprot.pin swissprot.pog swissprot.pos
    swissprot.pot swissprot.ppd swissprot.ppi swissprot.psq swissprot.
    ptf swissprot.pto taxdb.btd taxdb.bti
```

也可以用自己的本地数据构建数据库，例如使用 LEB 服务器上的 12HUMAN_CEA.FASTA 文件：

```
$ makeblastdb -in 12HUMAN_CEA.FASTA -out 12human_cea -dbtype prot


Building a new DB, current time: 05/07/2022 19:14:57
New DB name: /rd1/home/leb1c/tutorial/ch7/12human_cea
New DB title: 12HUMAN_CEA.FASTA
Sequence type: Protein
Keep MBits: T
Maximum file size: 1000000000B
Adding sequences from FASTA; added 12 sequences in 0.000636101 seconds
    .
```

注意此时需要在主目录下修改 .ncbirc 配置文件：

```
# the path where BLAST databases are installed
[BLAST]
BLASTDB=/rd1/home/leb1c/tutorial/ch7/swissprot
BLASTDB=/rd1/home/leb1c/tutorial/ch7/12human_cea
```

随后 blast 即可正常找到数据库。

5.4.2　blastp 和 blastn

blastp 用蛋白质序列搜索蛋白质数据库，一些常用参数列出如下：

选项	含义
-query	指定查询序列
-db	指定使用的数据库
-out	指定输出文件名，包括指定格式为 txt、xls
-evalue	指定可接受的最高假阳性期望
-outfmt	指定输出格式
-word_size	指定词长
-matrix	指定计分矩阵
-query_loc	指定查询序列中要查询片段的起止位置
-subject	指定使用的被查询的序列文件（当不查询数据库时）
-subject_loc	指定被查询序列文件的起止位置
-max_target_seqs	指定搜索结果中显示的最大序列数目

```
1 $ blastp -query HBA_HUMAN.FASTA -db swissprot -out HBA_SW.TXT -evalue
    0.01 -outfmt 7 -word_size 3 -matrix PAM250
2 $ less HBA_SW.TXT
```

输出文件如下：

```
1  # BLASTP 2.12.0+
2  # Query: HBA_HUMAN - P69905, Hemoglobin subunit alpha, HBA1; J Luo,
       2016-08-21
3  # Database: swissprot
4  # Fields: query acc.ver, subject acc.ver, % identity, alignment length
       , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
       , bit score
5  # 500 hits found
6  HBA_HUMAN     P69905.2      100.000 142   0      0      1      142
       1      142    6.67e-82      229
7  HBA_HUMAN     P06635.2      97.887 142    3      0      1      142
       1      142    3.17e-81      227
8  HBA_HUMAN     P01923.1      99.291 141    1      0      2      142
       1      141    6.73e-81      226
9  HBA_HUMAN     Q9TS35.2      98.592 142    2      0      1      142
       1      142    1.09e-80      226
10 HBA_HUMAN     P63107.2      97.183 142    4      0      1      142
```

```
      1      142    2.26e-80      225
11  HBA_HUMAN     P01924.1      97.872 141    3      0      2      142
      1      141    2.48e-80      225
12  HBA_HUMAN     P67817.2      96.479 142    5      0      1      142
      1      142    3.23e-80      225
13  HBA_HUMAN     P01928.2      95.775 142    6      0      1      142
      1      142    4.01e-80      225
14  HBA_HUMAN     P01926.2      96.479 142    5      0      1      142
      1      142    4.39e-80      224
15  HBA_HUMAN     P01929.1      96.454 141    5      0      2      142
      1      141    8.64e-80      224
16  HBA_HUMAN     Q9TS34.2      96.479 142    5      0      1      142
      1      142    1.22e-79      223
17  HBA_HUMAN     P18972.1      97.163 141    4      0      2      142
      1      141    1.77e-79      223
18  HBA_HUMAN     P21767.1      97.163 141    4      0      2      142
      1      141    2.15e-79      223
19  . . .
```

结果表格从左至右分别代表了：查询序列名，目标序列名，一致性百分比，比对长度，错配位点数目，起始空位数，查询序列起始位点，查询序列终止位点，目标序列起始位点，目标序列终止位点，假阳性期望数目（E-value），打分（bit 单位）。

如前所述，设置更低的 E-value 将使得到的结果减少，只有那些相似性很确定的结果被保留下来。

设置更小的词长将使找到的结果增加，并可能找出更多远缘序列，这涉及 1997 版 BLAST 的几个改进。

尝试在自己构建的数据库 `12human_cea` 中查询：

```
1 $ blastp -query CEAM5_HUMAN.FASTA -db 12human_cea -out CEA_CEA.txt -
    evalue 0.01 -outfmt 7
```

输出结果为：

```
1 # BLASTP 2.12.0+
2 # Query: CEAM5_HUMAN P06731 Human CEA5 IgC:145-675, J Luo, 2021-11-15
3 # Database: 12human_cea
4 # Fields: query acc.ver, subject acc.ver, % identity, alignment length
    , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
    , bit score
5 # 38 hits found
```

```
6  CEAM5_HUMAN   CEAM5_HUMAN   99.858 702   1     0     1     702
       1     702   0.0   1428
7  CEAM5_HUMAN   CEAM1_HUMAN   72.600 427   112   2     1     422
       1     427   0.0   614
8  CEAM5_HUMAN   CEAM1_HUMAN   61.736 311   115   2     284   591
       107   416   8.78e-123     367
9  CEAM5_HUMAN   CEAM1_HUMAN   64.953 214   74    1     462   675
       107   319   1.24e-84      267
10 CEAM5_HUMAN   CEAM1_HUMAN   24.234 359   223   16    375   702
       104   444   6.89e-11      54.3
11 CEAM5_HUMAN   CEAM6_HUMAN   83.901 323   52    0     1     323
       1     323   0.0   558
12 CEAM5_HUMAN   CEAM6_HUMAN   66.390 241   78    2     462   702
       107   344   2.03e-97      295
13 CEAM5_HUMAN   CEAM6_HUMAN   69.266 218   66    1     284   501
       107   323   1.20e-96      293
14 CEAM5_HUMAN   CEAM8_HUMAN   76.752 314   73    0     5     318
       5     318   4.57e-173     489
15 . . . . . .
16 CEAM5_HUMAN   CEA18_HUMAN   30.120 166   105   6     505   662
       141   303   3.81e-11      54.7
17 CEAM5_HUMAN   CEA19_HUMAN   30.357 112   75    3     16    124
       13    124   2.12e-12      57.8
18 # BLAST processed 1 queries
```

注意到，生成数据库的原始数据仅有 12 条，`blastp` 却找出了 38 个结果，这很好的说明了，BLAST 是一种局部比对，因为每个癌胚抗原蛋白可能有多个相互之间相似的结构域。

由此性质，我们可以指定搜索 CEA 中的某个结构域，已知 CEA5 中，第 35-144 为可变型免疫球蛋白结构域（Ig-like V-type），那我们就搜索它：

```
1 $ blastp -query CEAM5_HUMAN.FASTA -db 12human_cea -out CEA5_35-144_CEA
      .txt -evalue 0.01 -outfmt 7 -query_loc 35-144
```

结果更有针对性：

```
1 # BLASTP 2.12.0+
2 # Query: CEAM5_HUMAN P06731 Human CEA5 IgC:145-675, J Luo, 2021-11-15
3 # Database: 12human_cea
4 # Fields: query acc.ver, subject acc.ver, % identity, alignment length
      , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
```

```
      , bit score
5  # 12 hits found
6  CEAM5_HUMAN   CEAM5_HUMAN   100.000 110   0      0      35     144
       35     144    3.22e-77      231
7  CEAM5_HUMAN   CEAM6_HUMAN   89.091 110    12     0      35     144
       35     144    6.08e-71      206
8  CEAM5_HUMAN   CEAM1_HUMAN   89.091 110    12     0      35     144
       35     144    5.62e-70      208
9  CEAM5_HUMAN   CEAM3_HUMAN   86.239 109    15     0      35     143
       35     143    1.39e-67      194
10 CEAM5_HUMAN   CEAM8_HUMAN   71.560 109    31     0      35     143
       35     143    4.71e-55      165
11 CEAM5_HUMAN   CEAM7_HUMAN   66.355 107    35     1      38     143
       38     144    5.71e-49      147
12 CEAM5_HUMAN   CEAM4_HUMAN   46.729 107    57     0      35     141
       35     141    2.09e-32      104
13 CEAM5_HUMAN   CEA21_HUMAN   50.943 106    52     0      36     141
       36     141    5.50e-32      104
14 CEAM5_HUMAN   CEA16_HUMAN   35.514 107    69     0      35     141
       21     127    6.90e-21      75.9
15 CEAM5_HUMAN   CEA16_HUMAN   37.624 101    61     1      42     142
       325    423    1.30e-18      69.7
16 CEAM5_HUMAN   CEA18_HUMAN   34.043 94     60     2      47     140
       41     132    2.06e-13      54.7
17 CEAM5_HUMAN   CEA19_HUMAN   27.103 107    72     3      36     140
       34     136    2.19e-10      45.8
18 # BLAST processed 1 queries
```

此外，使用 -subject 选项可以指定一个序列文件而非数据库作为查询对象：

```
1  $ blastp -query CEAM5_HUMAN.FASTA -subject CEAM7_HUMAN.FASTA -out
      CEA5_CEA7.txt -evalue 0.01 -outfmt 7 -query_loc 35-144
2  $ less CEA5_CEA7.txt
```

```
1  # BLASTP 2.12.0+
2  # Query: CEAM5_HUMAN P06731 Human CEA5 IgC:145-675, J Luo, 2021-11-15
3  # Database: User specified sequence set (Input: CEAM7_HUMAN.FASTA)
4  # Fields: query acc.ver, subject acc.ver, % identity, alignment length
      , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
```

```
      , bit score
5 # 1 hits found
6 CEAM5_HUMAN   CEAM7_HUMAN   66.355 107    35     1      38     143
      38     144    3.23e-50      147
7 # BLAST processed 1 queries
```

5.4.3　psiblast

psiblast 可以迭代地进行 BLAST 搜索，-comp_based_stats 选项指定 profile 计分矩阵的校正方法:

```
1 $ psiblast -query HBA_HUMAN.FASTA -db swissprot -evalue 0.0001 -outfmt
      7 -out HBA_psi_SW.txt -num_iterations 2 -comp_based_stats 0
```

结果如下:

```
1  # PSIBLAST 2.12.0+
2  # Iteration: 1
3  # Query: HBA_HUMAN - P69905, Hemoglobin subunit alpha, HBA1; J Luo,
      2016-08-21
4  # Database: swissprot
5  # Fields: query acc.ver, subject acc.ver, % identity, alignment length
      , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
      , bit score
6  # 550 hits found
7  HBA_HUMAN     P69905.2      100.000 142    0      0      1      142
      1      142    1.80e-100     286
8  HBA_HUMAN     P01923.1      99.291 141    1      0      2      142
      1      141    2.89e-99      283
9  HBA_HUMAN     Q9TS35.2      98.592 142    2      0      1      142
      1      142    6.03e-99      283
10  . . . . . .
```

相比无迭代的 blastp ，找出的 Hits 更多。

blastn 用核酸序列搜索核酸序列库，其用法与 blastp 大同小异，不过在一些词长、阈值的设置上，仍然要注意，它们的默认值未必相同。可自行查阅手册，这里略。

5.4.4　blastx 、tblastn 和 tblastx

它们统称为翻译后 BLAST（translated BLAST），三者的区别是:

这里将有一个问题，为什么要作这种麻烦的处理?

程序	检测序列	数据库	检索方法
‘blastx‘	核酸	蛋白质	将查询序列按 6 条编码链翻译成蛋白质序列，用翻译后的序列搜索蛋白质序列数据库
‘tblastn‘	蛋白质	核酸	将核酸数据库按 6 条编码链翻译成蛋白质序列，用蛋白质序列搜索翻译所得蛋白质序列
‘tblastx‘	核酸	核酸	将检测序列和数据库按 6 条编码链翻译成蛋白质序列再进行搜索

因为一般而言，蛋白质序列比核酸序列更加保守（考虑同义突变的存在），这三种程序所进行的比较都是发生在蛋白水平的，它们比核酸水平的比较更加敏感。`blastx` 和 `tblastn` 常用于注释核苷酸序列上的编码区，也可用于检测编码区域中的移码。`tblastx` 能在不知道任何翻译蛋白质的情况下，将转录本与基因组序列进行比对，并且十分灵敏。但 1 次 `tblastx` 相当于要进行 6×6 次 `blastp` ，因此其计算量也很大。

下面演示以拟南芥转录因子 SPL3 蛋白质序列为检索序列，用 `tblastn` 搜索玉米转录因子编码区序列，其他的程序可查阅手册。

```
1 $ tblastn -query 17SPL_ARATH.FASTA -subject ZMTF_CDS.FASTA -out
    SPL_tblastn_ZMTF.txt -outfmt 7
2 $ less SPL_tblastn_ZMTF.txt
```

结果如下：

```
1  # TBLASTN 2.12.0+
2  # Query: SPL1_ARATH
3  # Database: User specified sequence set (Input: ZMTF_CDS.FASTA)
4  # Fields: query acc.ver, subject acc.ver, % identity, alignment length
       , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
       , bit score
5  # 6 hits found
6  SPL1_ARATH    PTZm00605.1   68.919 74    23    0    106   179
       727   948   2.77e-29      117
7  SPL1_ARATH    PTZm00608.1   68.919 74    23    0    106   179
       549   328   7.84e-29      117
8  SPL1_ARATH    PTZm00607.1   64.583 48    17    0    106   153
       1279  1422  6.34e-16      76.6
9  SPL1_ARATH    PTZm00606.1   41.333 75    40    1    74    148
       546   758   7.92e-14      67.8
10 SPL1_ARATH    PTZm00562.1   34.426 61    35    3    707   766
       1539  1369  3.3   26.2
11 SPL1_ARATH    PTZm00177.1   37.500 32    20    0    293   324
```

```
      240    145    9.6    24.6
12 # TBLASTN 2.12.0+
13 # Query: SPL2_ARATH
14 # Database: User specified sequence set (Input: ZMTF_CDS.FASTA)
15 # Fields: query acc.ver, subject acc.ver, % identity, alignment length
      , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
      , bit score
16 # 8 hits found
17 SPL2_ARATH    PTZm00608.1   40.996 261    111    7      104    346
      738    31     7.38e-46      161
18 SPL2_ARATH    PTZm00605.1   40.613 261    112    7      104    346
      538    1245   1.88e-45      158
19 SPL2_ARATH    PTZm00606.1   54.545 110    47     2      103    211
      435    758    1.46e-29      110
20 SPL2_ARATH    PTZm00607.1   53.509 114    49     2      104    216
      1090   1422   1.71e-28      112
21 SPL2_ARATH    PTZm00060.1   26.866 67     39     2      211    277
      1215   1045   0.23   28.5
22 SPL2_ARATH    PTZm00636.1   28.571 35     25     0      7      41
      58     162    1.8    25.8
23 SPL2_ARATH    PTZm00154.1   34.615 26     17     0      244    269
      747    670    5.5    24.3
24 SPL2_ARATH    PTZm00166.1   38.710 31     19     0      301    331
      436    528    8.1    23.9
25
26  . . . . . .
```

5.4.5 PHI-BLAST 与 `deltablast`

PHI-BLAST (Pattern-Hit Initiated BLAST, 模式搜寻起始的 BLAST) 将正则表达式匹配与匹配位置周边的局部比对结合。

在给定的蛋白序列 S 和一个正则表达式 P 时，PHI-BLAST 帮助找出其他的包含 P 的且在 P 出现的位点周边与 S 同源的蛋白质。但既有正则表达式，那么为什么不直接搜索呢?

因为直接搜索的结果可能是随机地产生的。而 PHI-BLAST 保证了找出的位点都有一定同源性，这样的蛋白才具有真正的相同基序。

DELTA-BLAST (Domain Enhanced Lookup Time Accelerated BLAST, 结构域增强加速查找时间的 BLAST) 使用位点特异性的打分矩阵 (Position Specific Scoring Matrix, PSSM) 搜索蛋白质序列数据库，与 PSI-BLAST 异曲同工。

DELTA-BLAST 首先搜索 NCBI CDD(NCBI Conserved Domains Database) 构建 PSSM，而后基于此 PSSM 来进行数据库搜索。

DELTA-BLAST 检测远缘同源物的灵敏度与 PSI-BLAST 相当，且它不需要迭代，速度更快。

使用 `deltablast` 需要先安装CDD-Delta，该数据库大小为 1.5G，为节约服务器空间，这里没有下载，也没有运行。但下面将介绍一种无需下载的“曲线救国”方法：

5.4.6　命令行上的在线服务 -remote

一些搜索囿于服务器存储空间或计算能力的限制，无法本地运行，那么可以为搜索添加 `-remote` 选项，在 NCBI 服务器上执行搜索，所有其他选项与本地运行相同。

例如：

```
$ blastp -query HBA_HUMAN.FASTA -db swissprot -out HBA_blastp_remote.
    txt -outfmt 7 -remote
```

大约花费了五分钟，得到的结果为：

```
# BLASTP 2.6.0+
# Query: HBA_HUMAN - P69905, Hemoglobin subunit alpha, HBA1; J Luo,
    2016-08-21
# RID: 7CYEGRJK016
# Database: swissprot
# Fields: query acc.ver, subject acc.ver, % identity, alignment length
    , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
    , bit score
# 500 hits found
HBA_HUMAN     P69905.2      100.000 142   0      0      1      142
    1      142    1.94e-100     286
HBA_HUMAN     P01923.1      99.291 141    1      0      2      142
    1      141    1.04e-98      282
HBA_HUMAN     Q9TS35.2      98.592 142    2      0      1      142
    1      142    2.33e-98      281
HBA_HUMAN     P06635.2      97.887 142    3      0      1      142
    1      142    3.49e-98      281
 . . . . . .
```

相比于本地运行，网络传输可能需要一些额外的时间，注意到结果开头多出了一行 `RID`，这正如网页版一样，是一个任务的请求编号（Request ID）。

“曲线救国” `deltablast` 如下：

```
$ deltablast -query HBA_HUMAN.FASTA -db swissprot -evalue 0.01 -outfmt
    7 -out HBA_delta_remote.txt -remote
$ less HBA_delta_remote.txt
```

结果：

```
# DELTABLAST 2.12.0+
# Query: HBA_HUMAN - P69905, Hemoglobin subunit alpha, HBA1; J Luo,
    2016-08-21
# RID: 7CY34N52013
# Database: swissprot
# Fields: query acc.ver, subject acc.ver, % identity, alignment length
    , mismatches, gap opens, q. start, q. end, s. start, s. end, evalue
    , bit score
# 1000 hits found
HBA_HUMAN     P81024.1      61.702 141    54     0      2      142
    1      141    4.50e-57      177
HBA_HUMAN     P08257.1      59.574 141    57     0      2      142
    1      141    4.65e-57      177
HBA_HUMAN     P68059.1      60.993 141    55     0      2      142
    1      141    1.73e-56      175
HBA_HUMAN     P02001.1      59.574 141    57     0      2      142
    1      141    2.18e-56      175
HBA_HUMAN     Q9XSN3.3      85.211 142    21     0      1      142
    1      142    2.59e-56      175
```

5.5　在线 BLAST 实践

多个组织都提供了在线 BLAST，其中 NCBI BLAST 最为常用，可访问网页并在线搜索。NCBI BLAST 提供的How to BLAST Guide 是学习在线 BLAST 的最佳指南。

此外，The new BLAST results page 介绍了如何解读 NCBI BLAST 的结果并使用附加展示功能。

参考文献

[1] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3), 403–410.

[2] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database

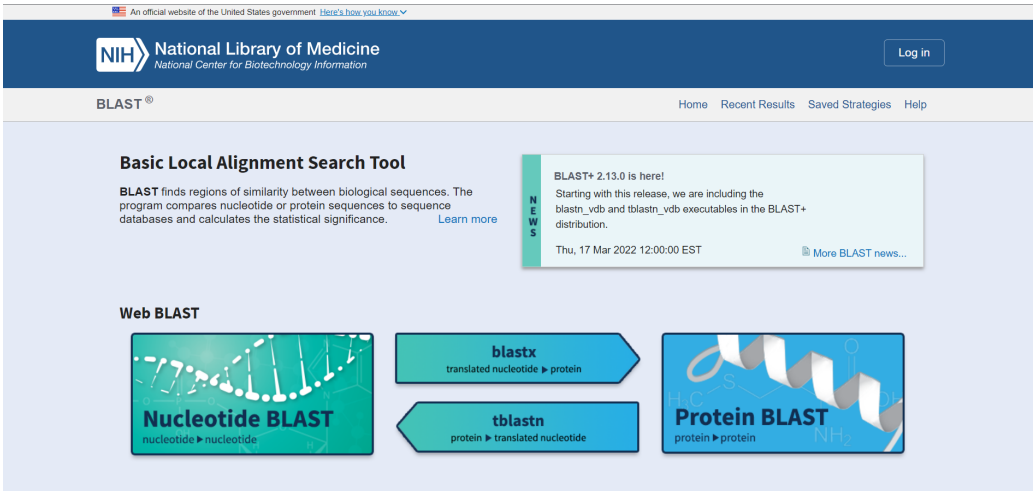


图 5.9: NCBI BLAST 界面。

search programs. *Nucleic acids research*, 25(17), 3389–3402.

[3] BLAST® Command Line Applications User Manual [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2008-.

[4] How to BLAST guide

[5] The new BLAST results page

[6] 罗静初. Linux 生物信息技术基础 第 4 讲 BLAST 数据库相似性搜索. 课堂讲义.

[7] 罗静初. 双序列比对基础和应用实例. 未发表.

[8] Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., & Madden, T. L. (2009). BLAST+: architecture and applications. *BMC bioinformatics*, 10, 421.

[9] Wheeler D, Bhagwat M. BLAST QuickStart: Example-Driven Web-Based BLAST Tutorial. In: Bergman NH, editor. *Comparative Genomics: Volumes 1 and 2*. Totowa (NJ): Humana Press; 2007. Chapter 9.

[10] Ewens, W. J., & Grant, G., R. (2005;2006;). *Statisical methods in bioinformatics: An introduction* (2nd ed.). Springer.

[11] Altschul S. F. (1993). A protein alignment scoring system sensitive at all evolutionary distances. *Journal of molecular evolution*, 36(3), 290–300.

[12] Schneider, T. D., & Stephens, R. M. (1990). Sequence logos: a new way to display consensus sequences. *Nucleic acids research*, 18(20), 6097–6100.

[13] Video: Heuristic of Blast. Puerto Rico IDeA Network Biomedical Research Excellence Bioinformatics Resources Core.

[14] Video: Blast The Algorithm. Puerto Rico IDeA Network Biomedical Research Excellence Bioinformatics Resources Core.

[15] Mount, D. W., & Cao, Z.(2006). *Bioinformatics: Sequence and genome analysis* = 生物信息学: 序列与基因组分析 (2nd ed.). 科学出版社.

[16] Once Upon A BLAST

第六章　HMMER 原理与实践

本章介绍谱隐马尔可夫模型及 HMMER 工具，这一章的内容与个人总结 4 相同，没有特别增补内容。

6.1　谱 HMM 基础

6.1.1　谱方法及其优点

谱 (Profile) 是一个重要但难以翻译的概念，它本身有几个意思（轮廓、剖面、概况、档案），又有特殊情境下的含义（配置、谱）。

本章中，谱被理解为对一个数据集中所有成员蛋白质具有的共同特征的统计描述。具体地说，谱描述了在一个多序列比对中每个位置的元素的出现情况，如下图：从一个序列比对中可以统计出每个位置的每种碱基的出现次数，这就是一个谱矩阵；根据谱矩阵的信息，选出每个位置出现频率最高的碱基，就可以构成一条共有序列（consensus）。

Alignment		A	T	C	C	A	G	C	T
		G	G	G	C	A	A	C	T
		A	T	G	G	A	T	C	T
		A	A	G	C	A	A	C	C
		T	T	G	G	A	A	C	T
		A	T	G	C	C	A	T	T
		A	T	G	G	C	A	C	T
Profile	A	5	1	0	0	5	5	0	0
	T	1	5	0	0	0	1	1	6
	G	1	1	6	3	0	1	0	0
	C	0	0	1	4	2	0	6	1
Consensus		A	T	G	C	A	A	C	T

图 6.1: 比对-谱-共识序列。

4.5.4 中介绍了一些 EMBOSS 中的谱分析工具，可以参考。

使用多序列比对中关于每个位点的信息来搜索数据库中的同源序列，这是符合直觉的。谱方法就是做了这件事，它根据多序列比对的结果生成了打分模型。

传统的比对，全局的，局部的，或广泛使用的 BLAST，都有很好的数学基础，并能够作出确定的打分估计。因为根据知识或者经验，研究者已经开发了满足各种要求的精妙矩阵，这些打分矩阵是固定的，事先确定的。但谱方法不一样，它使用的打分系统是临时的/一事一用的 (ad hoc)。隐马尔可夫模型 (HMM) 为 profile 方法提供了理论支撑。

6.1.2　谱 HMM

谱隐马尔可夫模型 (Profile HMM) 是一类概率模型，它封装了在一个相关序列集合中的序列在进化过程中发生过的改变（一个多序列比对）。谱 HMM 对一个多序列比对结果建模，捕捉关于比对的每个列中每种氨基酸的保守程度，如图 6.2。

这一模型还捕捉一些关键信息：例如空位和插入出现位置和几率的变化程度。不同于其他的序列同源性检测算法，profile HMM 使用位置依赖性的空位罚分和替换概率，这更好地反映了生物学上的真实情况。即，我们不应该相信，自然界具有通用的空位罚分和替换矩阵，即使它们的参数都是精心优化的。

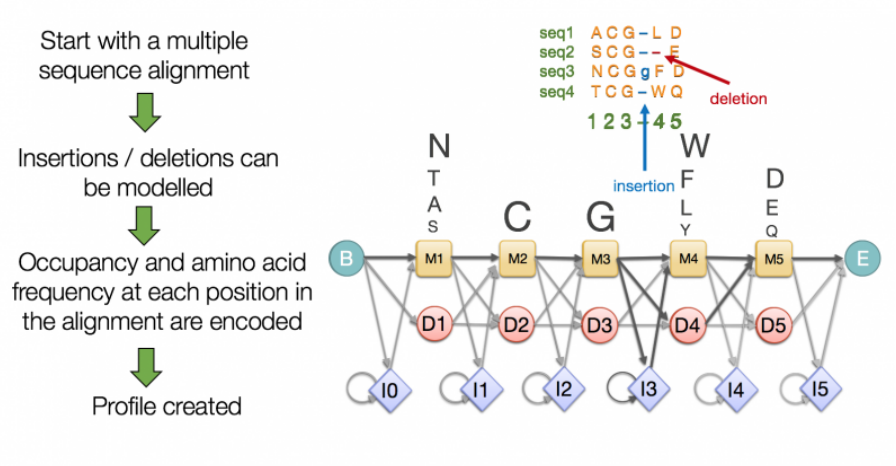


图 6.2: 谱的产生和隐马尔可夫模型。

如图所示，存在三种状态：

- 匹配状态（M）：M 状态的概率分布是在该位置上氨基酸出现的频率（不同大小的单字母氨基酸符号）
- 插入状态（I）：插入状态可以由自身转换到自身，因为可能在某个位置发生长串插入
- 删除状态（D）：删除状态不能由自身转换到自身，因为某个位置删除以后，下一个删除的必然是下一个位置

最终的概率模型描述了在每个位置上所观察到的氨基酸频率的估计，以及在多序列比对中所观察到的占据每个位置的氨基酸之间的转换频率的估计。

6.1.3　数学意味

6.1.3.1　一些概念

隐马尔可夫模型 (HMM) 是关于时序的概率模型，描述由一个隐藏的马尔科夫链随机生成不可观测的状态随机序列，再由各个状态生成一个观测，从而产生观测随机序列的过程。

所有可能的状态 $Q = \{q_1, q_2, \cdots, q_N\}$; 所有可能的观测 $V = \{v_1, v_2, \cdots, v_M\}$。

隐藏的马尔科夫链生成的状态的序列称为状态序列 $I = (i_1, i_2, \cdots, i_T)$。

每个状态生成一个观测，由此产生的观测的随机序列称为观测序列 $O = (o_1, o_2, \cdots, o_M)$。

状态转移矩阵定义为:

$A = [a_{ij}]_{N\times N}$，其中 $a_{ij} = P(i_{t+1} = q_j | i_t = q_i), \quad i = 1, 2, \cdots, N; j = 1, 2, \cdots, N$

观测概率矩阵定义为:

$B = [b_j(k)]_{N\times M}$，其中 $b_j(k) = P(o_t = v_k | i_t = q_j), \quad k = 1, 2, \cdots, M; j = 1, 2, \cdots, N$。

初始状态向量定义为: $\pi = (\pi_i)$，其中 $\pi_i = P(i_1 = q_i), i = 1, 2, \cdots, N$。

隐马尔可夫模型由初始状态向量、状态转移矩阵、观测概率矩阵决定，因此 HMM 可以表示为 $\lambda = (A, B, \pi)$。

6.1.3.2　HMM 的基本问题

1. 概率计算问题: 给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = (o_1, o_2, \cdots, o_T)$，计算在该模型 λ 下观测序列 O 出现的概率 $P(O|\lambda)$。采用前向算法、后向算法解决。(固定一个 HMM，将序列数据库中的每一条序列，都据此概率排序，概率超过阈值者，即判断可能是该模型产生的，进而判断可能是进化上来自该家族的，对应于 `hmmsearch`)
2. 学习问题: 已知观测序列 $O = (o_1, o_2, \cdots, o_T)$，估计模型 $\lambda = (A, B, \pi)$ 的参数，使得在该模型下观测序列概率 $P(O|\lambda)$ 最大。采用 Baum-Welch 算法解决。(实际上是“建立模型”的过程，对应于 `hmmbuild`)
3. 预测问题 (解码问题): 给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = (o_1, o_2, \cdots, o_T)$，求对给定观测序列条件概率 $P(I|O)$ 最大的状态序列 $I = (i_1, i_2, \cdots, i_T)$。采用维特比算法解决。(固定一条序列，对于每个 HMM 计算条件概率最大的序列，按照这个条件概率排序，概率超过阈值者，即判断该 HMM 可以解释该序列，进而判断该序列上具有此 HMM 代表的结构域，对应于 `hmmscan`)

6.1.3.3　HMM 应用于序列比对

基本步骤包括:

1. 首先要根据多序列比对结果建立 Profile 隐马氏模型 (学习问题)
2. 而后应用谱 HMM 对某一特定序列进行打分，判断查询的序列是否属于谱 HMM 代表的家族 (概率计算问题或解码问题)

对于蛋白质序列比对问题，所有可能的状态为 $Q = \{M, I, D, start, end\}$，所有可能的观测为 $V = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$。

隐马尔科夫链可以看成蛋白质序列上的运动，状态上，从 *start* 开始，随后以某种概率选择 $\{M, I, D\}$ 中的某一个：当选择 M 时，则根据观测概率矩阵给出 V 中的任何一个字母；当选择 I 时，观测概率由特定残基的全局出现概率决定；当选择 D 时，没有字母被观测到；这样，直到走到 *end* 状态。

这个过程将产生状态序列 I 和观测序列 O。比对结果，本质上是一个状态序列 I。蛋白质序列，就是观测序列 O。

无论是预测问题还是概率计算问题，无论是 `hmmscan` 还是 `hmmsearch` ，均能给出查询序列/profile HMM 的比对结果，这是因为它们都会得到 I，尽管是采取不同的思路。

6.2　本地 HMMER

6.2 的演示文件在 HMMER project 的GitHub 页面获取。UniProtKB/Swiss-Prot FASTA 数据库在 LEB 公共目录下获取。HMMER 版本：3.3.2 (Nov 2020)。

6.2.1　HMMER 软件包中的部分软件

软件名	用途
`hmmbuild`	输入多序列比对，构建 profile
`hmmalign`	使用 profile 产生一个多序列比对
`hmmsearch`	**在序列数据库中搜索 profile**
`hmmscan`	**在 profile 数据库中搜索序列**
`hmmpress`	为 `hmmscan` 准备 profile 数据库
`phmmer`	在序列数据库中搜索单个序列
`jackhmmer`	迭代地在数据库中搜索序列
`nhmmer`	在 DNA 序列库中搜索 DNA 查询条目
`nhmmscan`	在 DNA profile 数据库中搜索 DNA 序列
`hmmfetch`	从 profile 文件中获取 profile(s)
`hmmstat`	显示一个 profile 文件的统计摘要
`hmmemit`	从一个 profile 中生成示例序列
`hmmlogo`	从一个 profile 生成一个保守性图标图
`hmmconvert`	在不同的 profile 文件格式中进行转换
`makehmmerdb`	准备一个 `nhmmer` 二进制数据库

6.2.2　`hmmsearch` 用一个 profile 搜索一个序列数据库

6.2.2.1　`hmmbuild` 构建一个 profile

输入：比对文件，偏好 `.sto` ， `.aln` 亦可。

```
$ cat globins4.sto

# STOCKHOLM 1.0

HBB_HUMAN  ........VHLTPEEKSAVTALWGKV....
   NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVL
HBA_HUMAN  .........VLSPADKTNVKAAWGKVGA..HAGEYGAEALERMFLSFPTTKTYFPHF.
   DLS.....HGSAQVKGHGKKVA
MYG_PHYCA  .........VLSEGEWQLVLHVWAKVEA..
   DVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEAEMKASEDLKKHGVTVL
GLB5_PETMA PIVDTGSVAPLSAAEKTKIRSAWAPVYS..
   TYETSGVDILVKFFTSTPAAQEFFPKFKGLTTADQLKKSADVRWHAERII

HBB_HUMAN  GAFSDGLAHL...D..NLKGTFATLSELHCDKL..
   HVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANAL
HBA_HUMAN  DALTNAVAHV...D..DMPNALSALSDLHAHKL..
   RVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVL
MYG_PHYCA  TALGAILKK....K.GHHEAELKPLAQSHATKH..
   KIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDI
GLB5_PETMA NAVNDAVASM..DDTEKMSMKLRDLSGKHAKSF..QVDPQYFKVLAAVIADTVAAG
   .........DAGFEKLMSMICILL

HBB_HUMAN  AHKYH......
HBA_HUMAN  TSKYR......
MYG_PHYCA  AAKYKELGYQG
GLB5_PETMA RSAY.......
//
```

指定输出 profile 文件名，`globins4.hmm`：

```
$ hmmbuild globins4.hmm globins4.sto

# hmmbuild :: profile HMM construction from multiple sequence
   alignments
# HMMER 3.3.2 (Nov 2020); http://hmmer.org/
# Copyright (C) 2020 Howard Hughes Medical Institute.
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - -
```

```
# input alignment file:         globins4.sto
# output HMM file:              globins4.hmm
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    - -

# idx name              nseq  alen  mlen eff_nseq re/pos description
#---- -------------------- ----- ----- ----- -------- ------
    -----------
1     globins4              4   171   149    0.96 0.589

# CPU time: 0.10u 0.00s 00:00:00.10 Elapsed: 00:00:00.10
```

输出信息很好理解：

- nseq 代表 globins4.sto 是一个 4 条序列的多序列比对
- 比对长度 alen 是 171
- 共有序列（consensus）长度 mlen 为 149
- 实际有效序列 eff_nseq 为 0.96，这意味着 4 条序列之间非常相似，“近乎是同样的序列”
- re/pos 代表每个位置的相对熵值

6.2.2.2　hmmsearch 用一个 profile 文件搜索序列数据库

如果要搜索序列数据库，那么必然要事先下载得到这个数据库。这里使用 uniprot_sprot.fasta 。这里重定向了输出。

```
$ hmmsearch globins4.hmm uniprot_sprot.fasta > globins4.search.out
$ less globins4.search.out
```

输出结果为：

```
# hmmsearch :: search profile(s) against a sequence database
# HMMER 3.3.2 (Nov 2020); http://hmmer.org/
# Copyright (C) 2020 Howard Hughes Medical Institute.
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    - -
# query HMM file:             globins4.hmm
# target sequence database:   uniprot_sprot.fasta
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    - -

```

```
Query:       globins4 [M=149]
Scores for complete sequences (score includes all domains):
   --- full sequence --- --- best 1 domain --- -#dom-
    E-value score bias   E-value score bias   exp  N  Sequence
                  Description
    ------- ------ ----- ------- ------ ----- ---- -- --------
                  -----------
    4.9e-65 223.2  0.1   5.5e-65 223.0  0.1   1.0  1  sp|P02024|
       HBB_GORGO Hemoglobin subunit beta OS=Gorilla gor
    6.9e-65 222.7  0.1   7.6e-65 222.6  0.1   1.0  1  sp|P68871|
       HBB_HUMAN Hemoglobin subunit beta OS=Homo sapien
    6.9e-65 222.7  0.1   7.6e-65 222.6  0.1   1.0  1  sp|P68872|
       HBB_PANPA Hemoglobin subunit beta OS=Pan paniscu
    6.9e-65 222.7  0.1   7.6e-65 222.6  0.1   1.0  1  sp|P68873|
       HBB_PANTR Hemoglobin subunit beta OS=Pan troglod

 . . . . . .
```

当然，也可以使用经过挑选过的更小一些的序列数据库，甚至单条序列。

```
$ hmmbuild fn3.hmm fn3.sto
 . . . . . .
$ hmmsearch fn3.hmm 7LESS_DROME > fn3.search.out
```

这个例子的结果更简洁，下面将结果拆开解读。

输出：

```
$ cat fn3.search.out

# hmmsearch :: search profile(s) against a sequence database
# HMMER 3.3.2 (Nov 2020); http://hmmer.org/
# Copyright (C) 2020 Howard Hughes Medical Institute.
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - -
# query HMM file:             fn3.hmm
# target sequence database:   7LESS_DROME
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - -
```

第一部分：记录运行的程序，输入输出的文件

```
Query:       fn3  [M=85]
Accession:   PF00041.20
Description: Fibronectin type III domain
Scores for complete sequences (score includes all domains):
   --- full sequence --- --- best 1 domain --- -#dom-
    E-value score bias   E-value score bias   exp  N  Sequence
        Description
    ------- ------ ----- ------- ------ ----- ---- -- --------
        -----------
    5.6e-57 176.4  0.9   2.3e-16  46.2  0.9   9.8  9  7LESS_DROME
        RecName: Full=Protein sevenless;    EC=2.7
```

第二部分：序列查询结果，按 E-value 从小到大排序，最小的（最显著的）排在前面，以类似 BLAST 的风格展示：

- 第 1-3 列：描述全序列
  - E-value： score 比当前条目更高的假阳性结果的个数期望，是显著性的度量，一般小于 10^{-3} 即可认为显著；值得注意的是，E-value 的大小依赖数据库的大小，因为大数据库意味着更大的假阳性率
  - score：完整序列的 log-odds 分数，score 的大小与不依赖于数据库大小
  - bias： score 的偏差，只有当它大到与 score 数量级接近的时候才需要注意
- 第 4-6 列：描述序列中一个得分最高的结构域，E-value 、score 、bias 的含义类似，但这个例子中，所有的蛋白都只有一个结构域，因此这几列的意义不大
  - 如果 full sequence 和 best 1 domain 的 E-value 均显著的话，该序列就很可能是查询 profile 的同源物
  - 如果 full sequence 显著，但 best 1 domain 的 E-value 不显著的话，该序列可能是一个远缘的多结构域同源物
- 第 7-8 列：
  - exp：目标序列中的结构域的期望数，这是按照 HMMER 的模型计算的，并不一定是个整数
  - N：经过后处理后，HMMER 最终决定鉴定到的结构域的数目
  - 这两列应当是接近的，但当目标序列是高度重复时，它们未必接近

```
Domain annotation for each sequence (and alignments):
>> 7LESS_DROME RecName: Full=Protein sevenless;    EC=2.7.10.1;
   #    score bias  c-Evalue  i-Evalue hmmfrom hmm to  alifrom ali to
       envfrom env to    acc
 ---   ------ ----- --------- --------- ------- ------- ------- -------
       ------- ------- ----
```

```
 5     1 ?   -1.9   0.0      0.24      0.24      60      72 ..     396     408 ..
                395     410 .. 0.87
 6     2 !   41.9   0.0   5.1e-15   5.1e-15       2      83 ..     440     520 ..
                439     521 .. 0.95
 7     3 !   15.1   0.0   1.2e-06   1.2e-06      14      84 ..     838     913 ..
                827     914 .. 0.74
 8     4 !    4.6   0.0    0.0022    0.0022       6      36 ..    1206    1236 ..
               1203    1251 .. 0.83
 9     5 !   22.1   0.0   7.6e-09   7.6e-09      13      79 ..    1313    1380 ..
               1305    1384 .. 0.82
10     6 ?    0.4   0.0     0.045     0.045      57      72 ..    1754    1769 ..
               1742    1769 .. 0.88
11     7 !   46.2   0.9   2.3e-16   2.3e-16       1      82 [.    1800    1888 ..
               1800    1891 .. 0.91
12     8 !   21.0   0.0   1.7e-08   1.7e-08       5      74 ..    1904    1967 ..
               1901    1977 .. 0.90
13     9 !    8.9   0.0   9.8e-05   9.8e-05       1      85 []    1994    2107 ..
               1994    2107 .. 0.81
```

第三部分：对每个序列的结构域注释。7LESS_DROME 中包含一条序列，其中有 9 个结构域。

- ##：序号，按在原始序列中的出现顺序，并非按显著性顺序
- !：代表该结构域满足每个序列的和每个结构域的阈值
- ?：与 ! 相反
- score 和 bias 的含义与第二部分相同，对于单个结构域作计算
- c-Evalue：条件 E-value，事先知晓目标序列是一个真的同源序列时，判断每个结构域的显著性
- i-Evalue：独立 E-value，如果该结构域是唯一被确定的结构域，整条序列的显著性；这个例子中，两个 Evalue 没有区别，因为被搜索的数据库中仅有 1 条序列。一般而言，i-Evalue 显著即可判定同源性，如果有多个 i-Evalue 显著的结构域，那么可以通过 c-Evalue 寻找那些不那么显著的结构域
- hmmfrom 、hmm to 指出局部比对在 profile 中的起止点
- alifrom 、ali to 指出局部比对在目标序列中的起止点
- envfrom 、env to 指出结构域在目标序列上定位的范围，其范围并不确定
- acc：代表了比对上的目标序列的残基的平均后验概率

```
1  Alignments for each domain:
2  == domain 1 score: -1.9 bits; conditional E-value: 0.24
3                EESSSTTEEEEEE CS
4          fn3  60 ltgLkpgteYevr 72
5                 l+ L p+t+Y++r
6  7LESS_DROME 396 LEALIPYTQYRFR 408
```

```
7                6789*******98 PP
8    . . . . . .
9    == domain 8 score: 21.0 bits; conditional E-value: 1.7e-08
10                EEEEEEESSSEEEEEEE-CSSSSSTECEEEEEEEETTSSSTEEEEEEESTCSEEEEESSSTTEEEEEEEE CS
11          fn3   5 nlsvtevtstsltvsWtppedgngpitgYeveyrpknegeewneitvpgtttsvtltgLkpgteYevrVq 74
12                +l++ e +++sl+++W ++p+ + ++e+r++ e+ +++v g++t ++++ L+p t+Y++r+
13   7LESS_DROME 1904 ELQLLELGPYSLSLTWAGT---PDPLGSLQLECRSSAEQ---LRRNVAGNHTKMVVEPLQPRTRYQCRLL 1967
14                6889999***********...9**********655...7**********************9985 PP
15
16   == domain 9 score: 8.9 bits; conditional E-value: 9.8e-05
17                TSBCEEEEEEESSSEEEEEEE-CSSSSSTECEEEEEEEETTSSS.....TEEEEEEESTCS.EEEEESSSTTEE CS
18          fn3   1 saPsnlsvtevtstsltvsWtppedgngpitgYeveyrpknege.....ewneitvpgttt.svtltgLkpgte 68
19                s+P+ ++++ + + ++v+Wt+ + +++pi Y++e ++++    e+++ + +tt+ s++++ L+ ++
20   7LESS_DROME 1994
          SQPGKPQLEHIAEEVFRVTWTAARGNGAPIALYNLEALQARSDIrrrrrrrrrnsggsleqlpwaeepvvvEDQWLDFCNTTEISCIVKSLHSSRL
          2089
21                689999*****************9999********9888666668888888888888888777777777777777777777888888888888 PP
22
23                EEEEEEEEETTE.EEEEE CS
24          fn3   69 YevrVqavnggg.egpes 85
25                  +rV+a++ + +gp+s
26   7LESS_DROME 2090 LLFRVRARSLEHgWGPYS 2107
27                888888888544477665 PP
```

第四部分：每个结构域范围的具有最优后验概率的比对，以 domian8 为例:

- 第 1 行是一些基本信息，与第三部分一致
- 第 2 行：多序列比对注释
- 第 3 行：查询的 profile 的共有序列（consensus）
- 第 4 行：查询序列和目标序列的比对情况：字母代表匹配，+ 代表该残基有正的 log-odds 发射分数
- 第 5 行：目标序列
- 第 6 行：每个残基的后验概率，9 代表 85%-95%，以此类推

```
1  Internal pipeline statistics summary:
2  -------------------------------------
3  Query model(s):                            1  (85 nodes)
4  Target sequences:                          1  (2554 residues searched)
5  Passed MSV filter:                         1  (1); expected 0.0 (0.02)
6  Passed bias filter:                        1  (1); expected 0.0 (0.02)
7  Passed Vit filter:                         1  (1); expected 0.0 (0.001)
8  Passed Fwd filter:                         1  (1); expected 0.0 (1e-05)
9  Initial search space (Z):                  1  [actual number of targets]
10 Domain search space (domZ):                1  [number of targets reported
       over threshold]
11 # CPU time: 0.00u 0.00s 00:00:00.00 Elapsed: 00:00:00.00
12 # Mc/sec: 49.19
13 //
14 [ok]
15 (class) leb1c@bbt:~/tutorial/ch6$
```

第五部分：一份简单的统计总结，代表目标条目逐个通过不同的算法过滤，最后得到匹配序列的过程。本例很特殊，因为使用的序列数据库是单条序列。

6.2.3　phmmer 用单个蛋白质序列搜索蛋白质序列数据库

phmmer 功能类似 BLASTP 或 FASTA，你只需提供要查询的蛋白质序列文件和序列数据库文件，不需要提供 profile 文件。

本质上，HMMER 在内部为单个序列构建了一个 profile，这是通过位置无关的打分矩阵（BLOSUM62）和罚分参数计算的。

```
1 $ phmmer HBB_HUMAN uniprot_sprot.fasta > HBB_HUMAN.phmmer.out
```

其输出格式与 hmmsearch 别无二致，这里不再重复。

6.2.4　jackhmmer 进行迭代的蛋白质搜索

jackhmmer 功能类似 PSI-BLAST，用单个查询序列迭代地搜索序列数据库。

其第一轮搜索等同 phmmer ，将会得到一些匹配结果。这些结果被收集起来，作为一个多序列比对，基于此产生一个 profile。该 profile 被用于第二轮搜索同一序列数据库，第二轮搜索如同 hmmsearch 。新的结果又被收集起来，构建新的 profile，继续搜索，如此循环，直到无法检测到新的匹配了，或者达到最大迭代次数（默认为 5，以 -N 参数修改）。

这里设置了阈值 -E 0.001 和 --noali ，以避免输出太多结果和比对。

```
1 $ jackhmmer -E 0.001 --noali -N 6 HBB_HUMAN uniprot_sprot.fasta >
    HBB_HUMAN.jackhmmer.outdir
```

输出结果类似多个 hmmsearch 输出结果的拼合，但不同在于，第二部分的每个条目前都会有一个标记：+ 代表在本轮迭代中新发现的，- 代表在本轮迭代中被丢失的。

每一次迭代的结果后会有一个简单总结，将其摘出来看，含义很明白。

```
1 $ grep -E "^@@.*$" HBB_HUMAN.jackhmmer.out | less
```

```
1 @@ New targets included: 954
2 @@ New alignment includes: 955 subseqs (was 1), including original
    query
3 @@ Continuing to next round.
4 @@
5 @@ Round:               2
6 @@ Included in MSA:     955 subsequences (query + 954 subseqs from 954
     targets)
7 @@ Model size:          146 positions
8 @@
```

```
 9  @@ New targets included: 154
10  @@ New alignment includes: 1111 subseqs (was 955), including original
       query
11  @@ Continuing to next round.
12  @@
13  @@ Round:               3
14  @@ Included in MSA:     1111 subsequences (query + 1110 subseqs from
       1108 targets)
15  @@ Model size:          146 positions
16  @@
17  @@ New targets included: 58
18  @@ New alignment includes: 1170 subseqs (was 1111), including original
        query
19  @@ Continuing to next round.
20  @@
21  @@ Round:               4
22  @@ Included in MSA:     1170 subsequences (query + 1169 subseqs from
       1166 targets)
23  @@ Model size:          146 positions
24  @@
25  @@ New targets included: 0
26  @@ New alignment includes: 1170 subseqs (was 1170), including original
        query
27  @@
28  @@ CONVERGED (in 4 rounds).
29  @@
```

随着迭代的不断进行，第 5 轮迭代已经无法找出新序列了，因此停止，即使这里指定了 `-N 6` 。

6.2.5　用一个序列搜索一个 profile 数据库

用 profile 去搜索序列数据库，目的是找到与查询的 profile 相关的序列，作深入研究。用序列去搜索 profile 数据库，目的是要对该序列进行注释，找到它可能属于哪一个结构域家族。常用的 profile 数据库包括 Pfam、SMART 或 TIGRFams。

6.2.5.1　构建 profile 数据库文件

profile 数据库就是多个 profile 文件的合并，既可以先逐一构建后合并，也可以直接合并多序列比对文件，而后一次性构建。

构建小型的 profile 数据库：

```
$ hmmbuild globins4.hmm globins4.sto >/dev/null
$ hmmbuild fn3.hmm fn3.sto >/dev/null
$ hmmbuild Pkinase.hmm Pkinase.sto >/dev/null
$ cat globins4.hmm fn3.hmm Pkinase.hmm > minifam
```

大型 profile 数据库，例如 Pfam，如果下载其 `Pfam-A.seed` 自行构建 profile 数据库，将非常耗时，这时需考虑根据任务调度系统进行并行计算，或下载构建好的 `.hmm` 文件。

6.2.5.2　hmmpress 对平文件进行压缩和索引编制

平文件十分笨重，不利于快速处理，因此必须对其进行二进制压缩和索引编制。

```
$ hmmpress minifam

Working...  done.
Pressed and indexed 3 HMMs (3 names and 2 accessions).
Models pressed into binary file: minifam.h3m
SSI index for binary model file: minifam.h3i
Profiles (MSV part) pressed into: minifam.h3f
Profiles (remainder) pressed into: minifam.h3p
```

压缩和索引完毕。

6.2.5.3　hmmscan 搜索 profile 数据库

已知 7LESS_DROME 是一个受体酪氨酸激酶，具有 fibronectin type III 结构域和蛋白激酶结构域。

`minifam` profile 数据库包含了 `fn3` 、`Pkinase` 和 `globins4` 的 profile。

```
$ hmmscan minifam 7LESS_DROME > 7LESS_DROME.hmmscan.minifam.out
```

结果的含义与 `hmmsearch` 类似，差别在于，搜寻的目标已经从序列变成了 profile。

需要注意的是，HMMER 在计算中采用了随机抽样算法，因此不能保证同样的命令每次运行的结果一致。如果要求一致，需要设定 `--seed` 。

6.2.5.4　hmmstat 统计 profile 数据库的信息

```
$ hmmstat minifam

# hmmstat :: display summary statistics for a profile file
# HMMER 3.3.2 (Nov 2020); http://hmmer.org/
```

```
# Copyright (C) 2020 Howard Hughes Medical Institute.
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#
# idx name                 accession      nseq eff_nseq    M relent  info p relE compKL
# ---- -------------------- ------------ -------- -------- ------ ------ ------ ------ ------
1     globins4             -                 4     0.96   149   0.59   0.59   0.52   0.03
2     fn3                  PF00041.20       98    12.20    85   0.67   0.64   0.60   0.04
3     Pkinase              PF00069.24       38     2.57   259   0.59   0.57   0.50   0.02
```

结果中部分条目的含义为：

- `idx`：序号
- `name`：profile 的名称
- `accession`：登记号
- `nseq`：profile 是从多少个序列的比对中构建的
- `eff_nseq`：profile 是从多少个“有效序列”的比对中构建的，衡量这些序列的相似性
- `M`：以共有序列的长度计算 profile 的大小

6.2.6　搜索 DNA 序列

HMMER 最初为处理蛋白质序列而设计，但也可用于长 DNA 序列的 profile 搜索。仍然是需要先构建 profile 文件，而后 `nhmmer` 用 profile 搜索对应的 DNA 序列数据库。

`nhmmer` 的功能类似 `hmmsearch` ，`nhmmscan` 的功能类似 `hmmscan` 。

这里使用的例子中，`MADE1.sto` 是 100 个人类 MADE1 转座元件的多序列比对，而 `dna_target.fa` 是人类 1 号染色体上提取的长度为 330Kb DNA 序列。

6.2.6.1　`hmmbuild` 构建 profile

```
$ hmmbuild MADE1.hmm MADE1.sto

# hmmbuild :: profile HMM construction from multiple sequence alignments
# HMMER 3.3.2 (Nov 2020); http://hmmer.org/
```

```
# Copyright (C) 2020 Howard Hughes Medical Institute.
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# input alignment file:         MADE1.sto
# output HMM file:              MADE1.hmm
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# idx name                 nseq  alen  mlen    W eff_nseq re/pos
description
#---- -------------------- ----- ----- ----- ----- -------- ------
-----------
1     MADE1                 100   304    80   278    4.23 0.708 MADE1 (
MAriner Derived Element 1), a TcMar-Mariner DNA transposon

# CPU time: 0.02u 0.00s 00:00:00.02 Elapsed: 00:00:00.02
```

6.2.6.2　nhmmer 用 profile 搜索 DNA 序列数据库

```
$ nhmmer MADE1.hmm dna_target.fa > MADE1.nhmmer.out
```

输出结果的格式类似 `hmmsearch` 。

6.3　在线 HMMER

EBI-EMBL 将 HMMER 部署在其服务器上，可访问网页并在线搜索。
可选的序列数据库：

数据库	特点
UniProtKB	全面的蛋白质序列集合
Swiss-Prot	人工审核、带有注释
PDB	具有结构信息
Reference Proteomes	广泛覆盖自然界中的蛋白序列，反映支序多样性，是 HMMER Web 的默认选项
Ensembl Genomes	可选定特定物种的蛋白质

可选的 Profile HMM 数据库：
在线 HMMER 的输入格式：

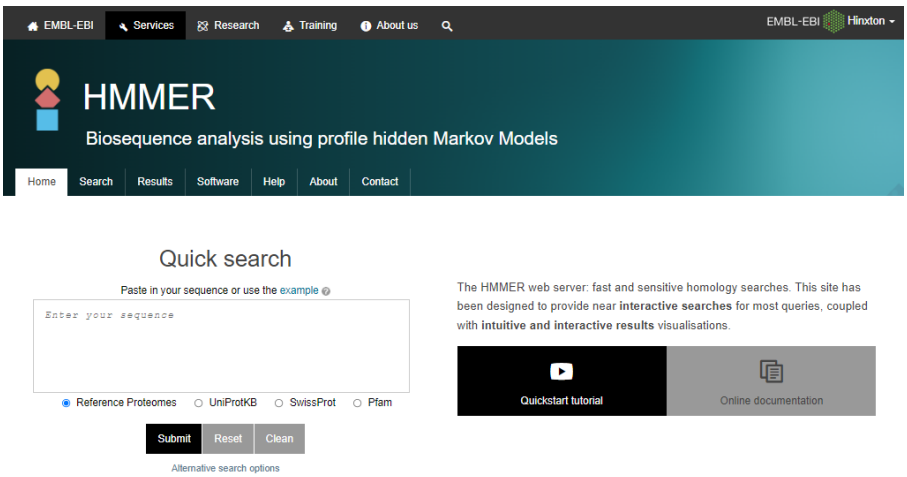


图 6.3: HMMER 网页工具。

数据库	特点
Pfam	大型蛋白质家族集合，HMMER Web 的默认选项
TIGRFAMS	为自动序列注释而设计的模型
PIRSF	提供 UniProtKB 的全面非重叠聚类模型

程序	可接受输入
phmmer	单个蛋白质氨基酸序列（仅序列或 FASTA）
hmmscan	单个蛋白质氨基酸序列（仅序列或 FASTA）
jackhmmer	单个蛋白质氨基酸序列（仅序列或 FASTA）、多序列比对文件、profile HMM
hmmsearch	多序列比对文件、profile HMM

详细使用指南可查阅参考文献 [3]，对于任意一种搜索方式，HMMER Web 均给出了示例，使用者不妨先运行示例，再调试各种参数。

参考文献

[1] EBI 关于 Pfam 数据库和隐马氏模型的网络教程

[2] HMMER Web Server Online Docs

[3] HMMER Software Userguide

[4] HMMER: Fast and sensitive sequence similarity searches

[5] Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics (Oxford, England)*, 14(9), 755-763.

[6] 李航. (2019). 统计学习方法第 2 版. 清华大学出版社.

[7] 罗静初. Linux 生物信息技术基础. 课堂讲义.

第七章　InterPro

本章介绍 InterPro 数据库及其搜索工具 InterProScan，这一章的内容与个人总结 5 相同，没有特别增补内容。

7.1　蛋白质标识

蛋白质分类的三种基本依据：所属的家族（family），包含的结构域（domain），具有的序列特征（sequence feature）。

- 蛋白质家族：一组具有共同进化起源的蛋白质，它们或许具有相关的功能、相似的结构或序列。家族的描述是等级制的：
  - 超家族（super family）
  - 家族（family）
  - 亚家族（subfamily）（从上至下，成员的亲缘关系逐渐接近）
- 结构域：蛋白质中能被较为清晰地分开的结构或功能单元，往往负责特定的相互作用，或者组成特定的结构。
- 序列特征：赋予蛋白质某些特征的氨基酸组，例如活性位点、结合位点、转录后修饰位点、重复序列；不同于结构域，序列特征通常仅有短短几个氨基酸，它常常被包含在结构域中。

蛋白质标识（protein signature）指用于蛋白质分类的可计算的预测模型，因计算方法的不同，标识有很多不同表达方式：

- 模式（patterns）：用于识别一些保守的序列特征，可用正则表达式表示，PROSITE 是基于模式的数据库。
- 谱（profiles）：由多序列比对统计每个位点的氨基酸出现概率，并转换成的位点特异性打分矩阵，CDD、HAMAP、PROSITE 是 profile 数据库。
- 指纹（fingerprints）：由多序列比对得到多个保守的短基序，每个基序被转换成 profile，整体构成一个指纹。PRINTS 数据库是一个指纹数据库。
- 隐马尔可夫模型（HMMs）：如个人总结 4 所述，HMM 是复杂且强大的统计模型，可以描述蛋白序列的整体特征，考虑插入和缺失。Pfam、SMART、TIGRFAM、PIRSF、PANTHER、SFLD、Superfamily 和 Gene3D 都运用了 HMM。

各种蛋白质标识都从一些相似序列的比对开始构建，而后以迭代方式搜索数据库，通过不断将新找出的蛋白质纳入考虑而不断使模型精细化。构建好的成熟模型可用于未知蛋白质

分析。因为各个数据库采用不同的蛋白质标识，所以查询并不方便。

7.2 InterPro 与 InterProScan

Interpro 是一个数据库，它将各种数据库中的各种蛋白质标识整合起来，消除冗余，让用户方便地查询蛋白可能属于的家族，包含的结构域和功能位点。EMBL-EBI 内部用 InterPro 来帮助注释 UniProtKB 中的蛋白序列。InterPro 包含 13 个成员数据库，它们使用不同的方式分类蛋白质，简述如下：

- CATH-Gene3D (Class-Architecture-Topology/fold-Homologous superfamily-Gene3D): CATH 是对 PDB 中的蛋白结构分类，Gene3D 使用 CATH 的信息来预测公共序列数据库的蛋白质结构域，由 UCL 维护
- CDD (Conserved Domains Database, 保守结构域数据库): 蛋白质结构单元的注释，由 NCBI 维护一个结构域数据集，并使用 3D 结构来预测序列-结构-关系之间的关系
- HAMAP (High-quality Automated and Manual Annotation of Proteins, 高质量的蛋白质自动和手工注释): 蛋白序列的分类和注释系统，由 SIB 维护
- MobiDB Lite: 蛋白质内在无序区域注释
- PANTHER (Protein ANalysis THrough Evolutionary Relationships，蛋白质进化关系分析): 人工注释的被细分为功能相关亚家族的蛋白家族集合，由 USC 维护
- Pfam: 很多常见蛋白结构域的多序列比对和隐马尔可夫模型数据库，由 EMBL-EBI 维护
- PIRSF (Protein Information Resource SuperFamily classification system，PIR 超家族分类系统): 一个具有从超家族到亚家族的多层次序列多样性的网络，反映了全长蛋白质和结构域的进化关系，由 Georgetown University Medical Centre 维护
- PRINTS: 蛋白质指纹数据库，University of Manchester 维护
- Prosite: 蛋白结构域、家族和功能位点数据库，还包含用于识别这些蛋白质的模式或谱，由 SIB 维护
- SFLD (Structure-Function Linkage Database, 结构-功能联系数据库): 酶的层级分类数据，将特定序列结构特征与特定催化功能相关联，由 UCSF 维护
- SMART (Simple Modular Architecture Research Tool，简单模块组织研究工具): 遗传可变结构域的识别和注释、结构域组织的分析，由 EMBL-EBI 维护
- SUPERFAMILY: 代表了所有已知结构蛋白的 profile 隐马尔可夫模型库，University of Bristol 维护
- TIGRFAMs: 人工维护的蛋白质家族数据库，包括隐马尔可夫模型 (HMM) 、多序列比对、注释，由 NCBI 维护

InterPro 的管理团队手动地整合不同成员数据库中的蛋白签名，并把那些代表相同蛋白家族、结构域或位点的蛋白标识合并为一个 InterPro 条目。如有可能，团队也会追溯条目之间的生物学关系。团队会检查标识的准确性，并向它们附加额外的相关信息，包括组成的条

目名、描述性的摘要、文献链接和 GO 条目，同时也提供指向 UniProt、ENZYME 和 PDBe 的链接。图 7.1 展示了 InterPro 的基本结构：

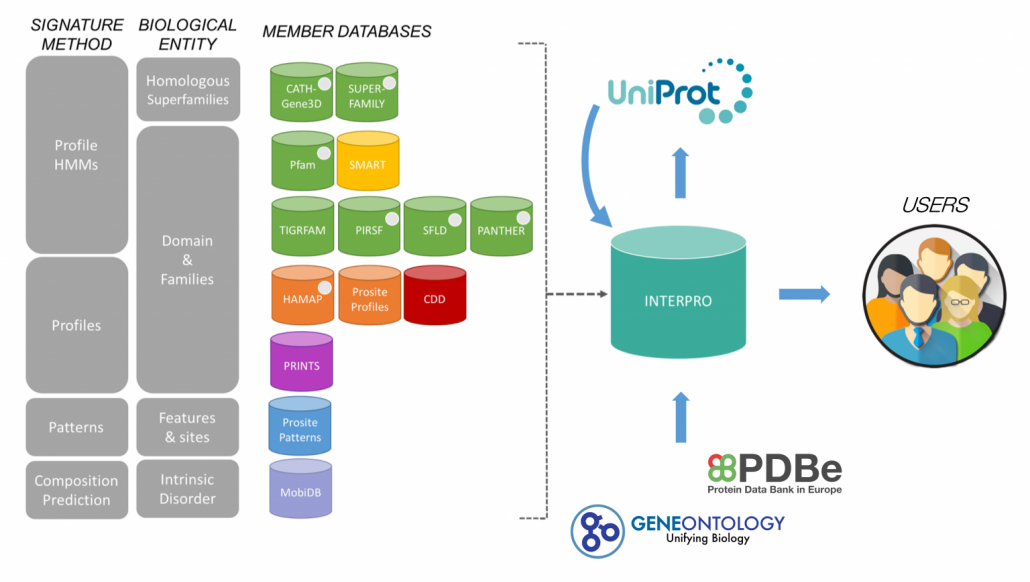


图 7.1: InterPro 数据库的基本结构。

InterPro 擅长发现蛋白的家族、功能和并提示可能的结构，但它不能进行结构比对、也不能分析基因组序列（例如预测外显子或转录起始位点）。

InterProScan 是一个 InterPro 搜索工具，提供 FASTA 格式文件，InterProScan 将在各个数据库中查询匹配，并以各种格式输出结果。InterProScan 的目标是初步确定未知序列的功能或家族所属。

7.3 本地 InterProScan

7.3.1 软件安装和依赖

InterPro 数据库很大，因此并不整合在 InterProScan 软件中。可经 `conda` 安装 InterProScan 软件，安装完毕后，屏幕上将打印出下载数据库的方法，可以照做下载。

下面使用公共服务器上已经安装好的 InterProScan 及数据库创建软链接，并尝试运行：

```
1 $ ln -s /rd1/home/public/interproscan-5.51-85.0
2 $ ls
3 interproscan-5.51-85.0
4
5 $ ./interproscan-5.51-85.0/interproscan.sh
6 Java version 11 is required to run InterProScan.
7 Detected version 1.8.0_152-release
8 Please install the correct version.
9
```

```
$ java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-8u312-b07-0ubuntu1~18.04-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
```

注意到，该版本的 InterProScan 需要 Java 11，解决办法：conda 创建新环境，并安装对应的 openjdk：

```
$ conda create -n iprscan python=3.7
$ conda install openjdk=11.0.13
$ java -version

openjdk version "11.0.13" 2021-10-19
OpenJDK Runtime Environment JBR-11.0.13.7-1751.21-jcef (build 11.0.13+7-b1751.21)
OpenJDK 64-Bit Server VM JBR-11.0.13.7-1751.21-jcef (build 11.0.13+7-b1751.21, mixed mode)
```

再次测试 InterProScan，无参数运行将返回使用指南，内容包括参数设置方法、可做的分析、未激活的分析。

```
(iprscan) leb1c@bbt:~/iprscan$ ./interproscan-5.51-85.0/interproscan.sh
23/05/2022 21:53:04:757 Welcome to InterProScan-5.51-85.0
23/05/2022 21:53:04:765 Running InterProScan v5 in STANDALONE mode... on Linux
usage: java -XX:+UseParallelGC -XX:ParallelGCThreads=2 -XX:+AggressiveOpts -XX:+UseFastAccessorMethods -Xms128M
       -Xmx2048M -jar interproscan-5.jar


Please give us your feedback by sending an email to

interhelp@ebi.ac.uk

 -appl,--applications <ANALYSES>                    Optional, comma separated list of analyses. If this option
                                                    is not set, ALL analyses will
```

```
                                                    be run.
14  -b,--output-file-base <OUTPUT-FILE-BASE>        Optional, base
       output filename (relative or absolute path).
15                                                  Note that this
                                                    option, the --
                                                    output-dir (-d)
                                                     option and the
16                                                  --outfile (-o)
                                                    option are
                                                    mutually
                                                    exclusive. The
17                                                  appropriate file
                                                    extension for
                                                    the output
                                                    format(s) will
                                                    be
18                                                  appended
                                                    automatically.
                                                    By default the
                                                    input file path
                                                    /name
19                                                  will be used.
20     . . . . . .
21  -T,--tempdir <TEMP-DIR>                         Optional, specify
       temporary file directory (relative or
22                                                  absolute path).
                                                    The default
                                                    location is
                                                    temp/.
23  -verbose,--verbose                              Optional, display
       more verbose log output
24  -version,--version                              Optional, display
       version number
25  -vl,--verbose-level <VERBOSE-LEVEL>             Optional, display
       verbose log output at level specified.
26  -vtsv,--output-tsv-version                      Optional, includes
        a TSV version file along with any TSV
27                                                  output (when TSV
                                                    output
```

```
                                                    requested)
28 Copyright © EMBL European Bioinformatics Institute, Hinxton, Cambridge
     , UK. (http://www.ebi.ac.uk) The InterProScan
29 software itself is provided under the Apache License, Version 2.0 (
     http://www.apache.org/licenses/LICENSE-2.0.html).
30 Third party components (e.g. member database binaries and models) are
     subject to separate licensing - please see the
31 individual member database websites for details.
32
33 Available analyses:
34                     TIGRFAM (15.0) : TIGRFAMs are protein families
                          based on hidden Markov models (HMMs).
35                       . . . . . .
36                  MobiDBLite (2.0) : Prediction of intrinsically
                       disordered regions in proteins.
37                       PIRSF (3.10) : The PIRSF concept is used as a
                            guiding principle to provide comprehensive and
                             non-overlapping clustering of UniProtKB
                            sequences into a hierarchical order to reflect
                             their evolutionary relationships.
38
39 Deactivated analyses:
40                       TMHMM (2.0c) : Analysis TMHMM is deactivated,
                            because the resources expected at the
                            following paths do not exist: bin/tmhmm/2.0c/
                            decodeanhmm, data/tmhmm/2.0c/TMHMM2.0c.model
41                        . . . . . .
42                       Phobius (1.01) : Analysis Phobius is deactivated,
                             because the resources expected at the
                            following paths do not exist: bin/phobius
                            /1.01/phobius.pl
43 (iprscan) leb1c@bbt:~/iprscan$
```

7.3.2 使用示例数据运行

```
1 $ interproscan-5.51-85.0/interproscan.sh -i interproscan-5.51-85.0/
     test_all_appl.fasta -f tsv -dp
2 24/05/2022 13:19:35:514 Welcome to InterProScan-5.51-85.0
```

```
3  24/05/2022 13:19:35:517 Running InterProScan v5 in STANDALONE mode...
       on Linux
4  24/05/2022 13:19:45:921 RunID: bbt_20220524_131945460_717v
5  24/05/2022 13:20:01:093 Loading file /rd1/home/leb1c/iprscan/
       interproscan-5.51-85.0/test_all_appl.fasta
6  24/05/2022 13:20:01:094 Running the following analyses:
7  [CDD-3.18,Coils-2.2.1,Gene3D-4.3.0,Hamap-2020_05,MobiDBLite-2.0,
       PANTHER-15.0,Pfam-33.1,PIRSF-3.10,PIRSR-2021_02,PRINTS-42.0,
       ProSitePatterns-2019_11,ProSiteProfiles-2019_11,SFLD-4,SMART-7.1,
       SUPERFAMILY-1.75,TIGRFAM-15.0]
8  Pre-calculated match lookup service DISABLED. Please wait for match
       calculations to complete...
9  24/05/2022 13:20:35:702 26% completed
10 24/05/2022 13:21:12:937 52% completed
11 24/05/2022 13:22:27:049 75% completed
12 24/05/2022 13:30:41:603 92% completed
13 24/05/2022 13:31:00:143 100% done: InterProScan analyses completed
14
15 $ ls
16 interproscan-5.51-85.0 temp test_all_appl.fasta.tsv
```

注意到，这个计算过程花费了 11 分钟，interproscan 的计算很耗费计算资源。

为了部分解决这个问题，interproscan 默认开启了 Precalculated match lookup（预先计算匹配查询）服务。InterProScan 的分析是完全基于序列的，因此，同样的查询序列必定产生同样的结果。利用这个特性，InterProScan 在收到查询序列后会先计算查询序列的 MD5 值，然后利用预先计算匹配查询服务：如果该序列曾经被查询过，则可直接输出查询结果（无需再次运行查找）；如果该序列没有在服务中，则运行一次查找。

interproscan 默认使用的是 EBI 托管的查找服务，但用户也可以下载本地副本进行查找。如欲关闭预先计算匹配查询服务，需使用 -dp/--disable-precalc 选项，上面的运行就使用了这一选项，下面尝试省略之。

```
1  $ interproscan-5.51-85.0/interproscan.sh -i interproscan-5.51-85.0/
       test_all_appl.fasta -f tsv
2
3  24/05/2022 13:37:10:362 Welcome to InterProScan-5.51-85.0
4  24/05/2022 13:37:10:371 Running InterProScan v5 in STANDALONE mode...
       on Linux
5  24/05/2022 13:37:23:661 RunID: bbt_20220524_133723317_hhnj
6  24/05/2022 13:37:38:386 Loading file /rd1/home/leb1c/iprscan/
       interproscan-5.51-85.0/test_all_appl.fasta
```

```
24/05/2022 13:37:38:392 Running the following analyses:
[CDD-3.18,Coils-2.2.1,Gene3D-4.3.0,Hamap-2020_05,MobiDBLite-2.0,
    PANTHER-15.0,Pfam-33.1,PIRSF-3.10,PIRSR-2021_02,PRINTS-42.0,
    ProSitePatterns-2019_11,ProSiteProfiles-2019_11,SFLD-4,SMART-7.1,
    SUPERFAMILY-1.75,TIGRFAM-15.0]
Available matches will be retrieved from the pre-calculated match
    lookup service.

Matches for any sequences that are not represented in the lookup
    service will be calculated locally.
2022-05-24 13:37:44,947 [Thread-6] [uk.ac.ebi.interpro.scan.business.
    sequence.BerkeleyPrecalculatedProteinLookup:748] WARN -

The version of InterProScan you are using is 5.51-85.0
The version of the lookup service you are using is 5.55-88.0
As the data in these versions is not the same, you cannot use this
    match lookup service.
InterProScan will now run locally
If you would like to use the match lookup service, you have the
    following options:
i) Download the newest version of InterProScan5 from our FTP site by
    following the instructions on:
   https://www.ebi.ac.uk/interpro/interproscan.html
ii) Download the match lookup service for your version of InterProScan
     from our FTP site and install it locally.
    You will then need to edit the following property in your
       configuration file to point to your local installation:
    precalculated.match.lookup.service.url=

In the meantime, the analysis will continue to run locally.


24/05/2022 13:38:15:489 25% completed
24/05/2022 13:39:00:380 50% completed
24/05/2022 13:40:07:681 75% completed
24/05/2022 13:48:08:760 90% completed
24/05/2022 13:48:21:841 100% done: InterProScan analyses completed
```

运行出错，因为版本不合：InterProScan 版本是 5.51-85.0，而 EBI 托管的查询服务的版

本是 5.55-88.0。所以此次运行仍然是本地计算完成的，耗时约 11 分钟。

解决方法：

1. 下载安装最新版本的 InterProScan。
2. 下载符合当前版本 InterProScan 的预先计算查询服务，并修改配置文件，以本地服务替代 EBI 提供的服务。

二者都需要下载大文件，因此不再演示。

7.3.3 选项设置

如无参数运行 `interproscan.sh` 所打印的指南，interproscan 有众多选项，下面择要介绍：

选项	描述
`-dp / --disable-precalc`	停用预先计算查找服务，这会导致 InterProScan 运行变慢
`-appl / --applications`	指定要做的分析，实际上是指定数据库，可以指定 13 个成员数据库中的一个或多个（逗号分隔）；未指定则搜索全部数据库；如欲搜索非成员数据库，需要安装相应服务
`-i / --input`	指定输入文件，FASTA 格式
`-goterms / --goterms`	启用相应的 GO 注释查询
`-b / --output-file-base`	指定输出文件的路径和文件名（basename），该命令不会覆盖已有文件，较为安全，不推荐用 `-o` 指定输出文件
`-t / --seqtype`	指定输入文件的序列类型，只有查询核酸序列是，需要指定参数 `n`
`-f / --formats`	指定输出文件的格式（可指定多种，逗号分隔），默认为 `TSV, XML, GFF3, JSON`
`-pa / --pathways`	启用相应的通路注释查询

7.3.4 输出格式

输出格式包括 TSV(Tab-separated values，制表符分隔值)，XML(可扩展标记语言，Extensible Markup Language)，JSON，GFF3(Generic Feature Format Version 3, 通用特征格式第 3 版)。其中，XML、JSON 信息最丰富，但不适合直接阅读；GFF3 的信息较为丰富；TSV 信息最少。

7.3.4.1 TSV 格式

TSV 输出文件共有 13-15 列，以 5.2.3 中的示例输出为例，分别代表：

```
1  UPI0004FABBC5 92e4b89dd86f8ab828f57121f6d7d460   257    PRINTS PR01914
       Neurotrophin-3 signature   81     95     2.0E-26 T     24-05-2022
         IPR015578     Neurotrophin-3
```

```
2 UPI0004FABBC5 92e4b89dd86f8ab828f57121f6d7d460 257 Pfam PF00243
      Nerve growth factor family 145 254 7.7E-52 T 24-05-2022
        IPR002072 Nerve growth factor-related
3 UPI0004FABBC5 92e4b89dd86f8ab828f57121f6d7d460 257 PANTHER
      PTHR11589 NERVE GROWTH FACTOR NGF -RELATED 1 257 5.6E
      -150 T 24-05-2022 IPR020408 Nerve growth factor-
      like
```

列	条目	例子
1	蛋白登录号	UPI0004FABBC5
2	序列 MD5 值	92e4b89dd86f8ab828f57121f6d7d460
3	序列长度	257
4	分析类型（数据库）	PRINTS
5	标识登录号（Signature accession）	PR01914
6	标识描述	Neurotrophin-3 signature
7	起始位点	81
8	终止位点	95
9	分数或 e-value，不同的成员数据库计算方法不同	2.0E-26
10	状态，匹配的真假	T
11	运行查询的日期	24-05-2022
12	InterPro 注释-登录号	IPR015578
13	InterPro 注释-描述	Neurotrophin-3
14	GO 注释，如果启用了 `-goterms` 选项	-
15	通路注释，如果启用了 `--pathways` 选项	-

7.3.4.2 GFF3 格式

GFF3 有较为丰富的内容和成熟的规范，它允许用户从匹配回溯到预测的蛋白质和核酸序列，并包含了预测的蛋白质序列及其匹配的 FASTA 序列。示例：

```
1 ##gff-version 3
2 ##feature-ontology http://song.cvs.sourceforge.net/viewvc/song/
     ontology/sofa.obo?revision=1.269
3 ##interproscan-version 5.51-85.0
4 ##sequence-region UPI0004FABBC5 1 257
5 UPI0004FABBC5 . polypeptide 1 257 . + .
     ID=UPI0004FABBC5;md5=92e4b89dd86f8ab828f57121f6d7d460
```

```
6  UPI0004FABBC5 Gene3D protein_match 139    257    8.2E-53 +      .
       date=24-05-2022;Target=UPI0004FABBC5 139 257;ID=match$1_139_257;
       Name=G3DSA:2.10.90.10;status=T;Dbxref="InterPro:IPR029034"
7  . . . . . .
8  UPI0004FABBC5 SUPERFAMILY   protein_match 137    253    3.76E-51
       +      .      date=24-05-2022;Target=UPI0004FABBC5 137 253;ID=
       match$12_137_253;Name=SSF57501;status=T;Dbxref="InterPro:IPR029034"
9  ##sequence-region UPI0002E0D40B 1 370
10 UPI0002E0D40B .      polypeptide   1      370    .      +      .
       ID=UPI0002E0D40B;md5=f91cb3cf61f2d7c7f5aaf6ea04e07868
11 UPI0002E0D40B ProSitePatterns protein_match 54   62     .      +
       .      date=24-05-2022;Target=UPI0002E0D40B 54 62;ID=match$13_54_62
       ;signature_desc=Asparaginase / glutaminase active site signature
       1.;Name=PS00144;status=T;Dbxref="InterPro:IPR020827"
12 . . . . . .
13 ##FASTA
14 >O31533
15 MGNYLSVAVELVCGLGILFIILKLLGKTQFSQITPFDFISALILGELVGNAVYDHEIKIK
16 EIIFASLLWGVLIYIIEFITQKMKSSRKFLEGEPNIVIRKGELQYKVMKKNKIDINQLQS
17 LLRQAGSFSIQEVEYAILETNGMVSVLPKSDFDKPTNKDLQIPSKSVSLPITLIIDGEIV
18 RDNLKEAGVDEQWLKQELKKKNIDKTEDVLFAEWHKNKPLYTVTYEQSRST
19 >UPI0000167F57
20 MKAQGETEESEKLSKMSSLLERLHAKFNQNRPWSETIKLVRQVMEKRVVMSSGGHQHLVS
21 CLETLQKALKVTSLPAMTDRLESIARQNGLGSHLSASGTECYITSDMFYVEVQLDPAGQL
22 . . . . . .
```

####FASTA 部分包含了预测的蛋白质序列。####sequence-region 部分包含很多列，它们的含义是：

列	条目	例子
1	序列 ID	UPI0004FABBC5
2	来源	Gene3D
3	特征类型	protein_match
4	起始位点	139
5	终止位点	257
6	分数，不同的成员数据库计算方法不同	8.2E-53
7	链信息（正义链或反义链）	+
8	phase(仅对特征类型为 CDS 的行适用)	.
9	属性	date=24-05-2022...

7.3.4.3 JSON 格式的网页展示

JSON 格式虽然不适合直接阅读，但 EBI 提供了方便的网页查看工具。图 7.2-7.5 是 JSON 格式网页展示的方法。

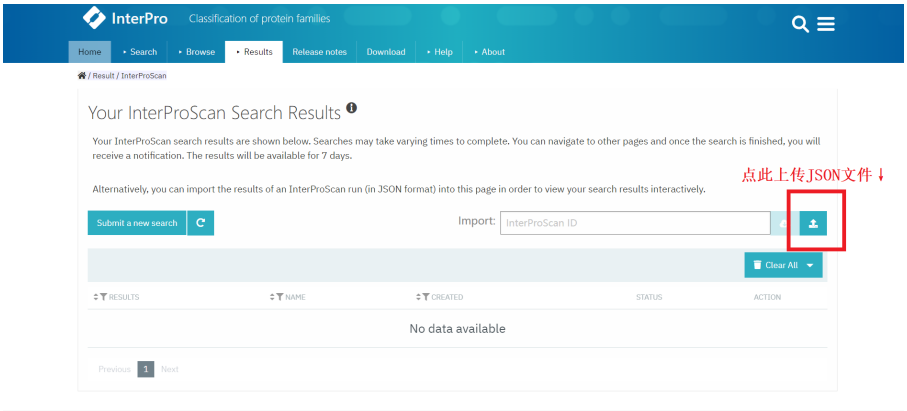


图 7.2: 在页面红框处上传 JSON 文件，虽然提示版本不合，但仍可上传，选择 “OK”。

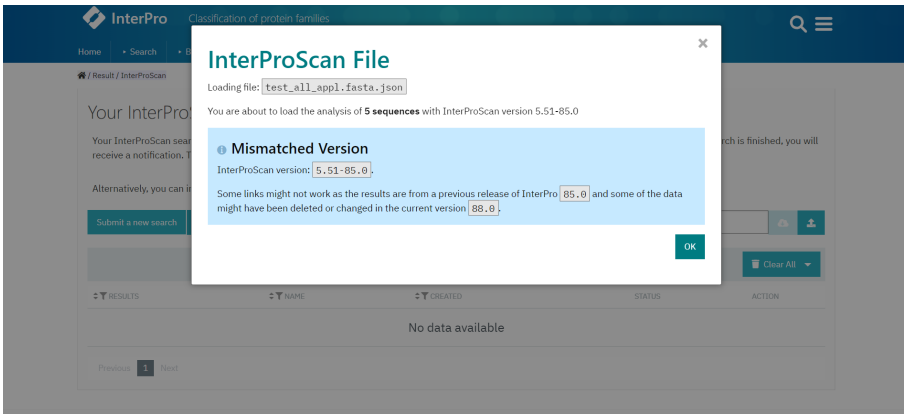


图 7.3: 在页面红框处上传 JSON 文件，虽然提示版本不合，但仍可上传，选择 “OK”。

7.3.5 自定义数据运行

选择 HBA_HUMAN.FASTA 运行 interproscan：

```
1 $ cd iprscan/
2 $ ./interproscan-5.51-85.0/interproscan.sh -i HBA_HUMAN.FASTA -goterms
      -pathways -dp
3 24/05/2022 15:36:50:247 Welcome to InterProScan-5.51-85.0
4 24/05/2022 15:36:50:251 Running InterProScan v5 in STANDALONE mode...
     on Linux
5 24/05/2022 15:37:00:591 RunID: bbt_20220524_153700090_kmlj
6 24/05/2022 15:37:16:514 Loading file /rd1/home/leb1c/iprscan/HBA_HUMAN
     .FASTA
```

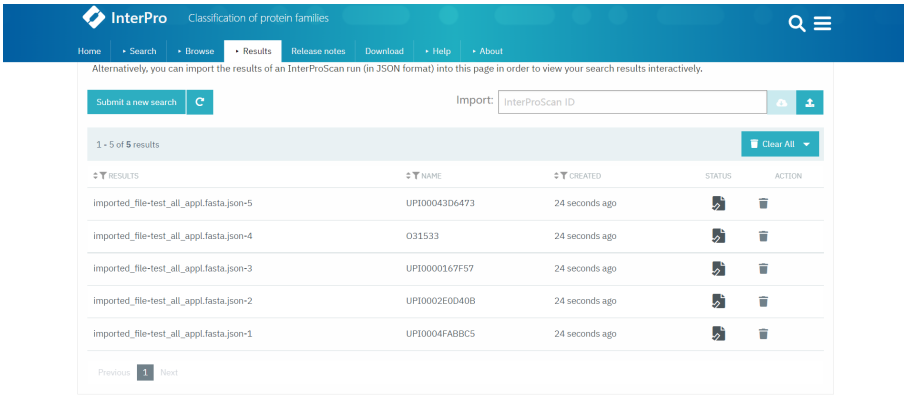


图 7.4: 选择特定结果查看，查询的 FASTA 文件包含 5 条序列，因此会分成 5 个结果，选择 UPI0004FABBC5。

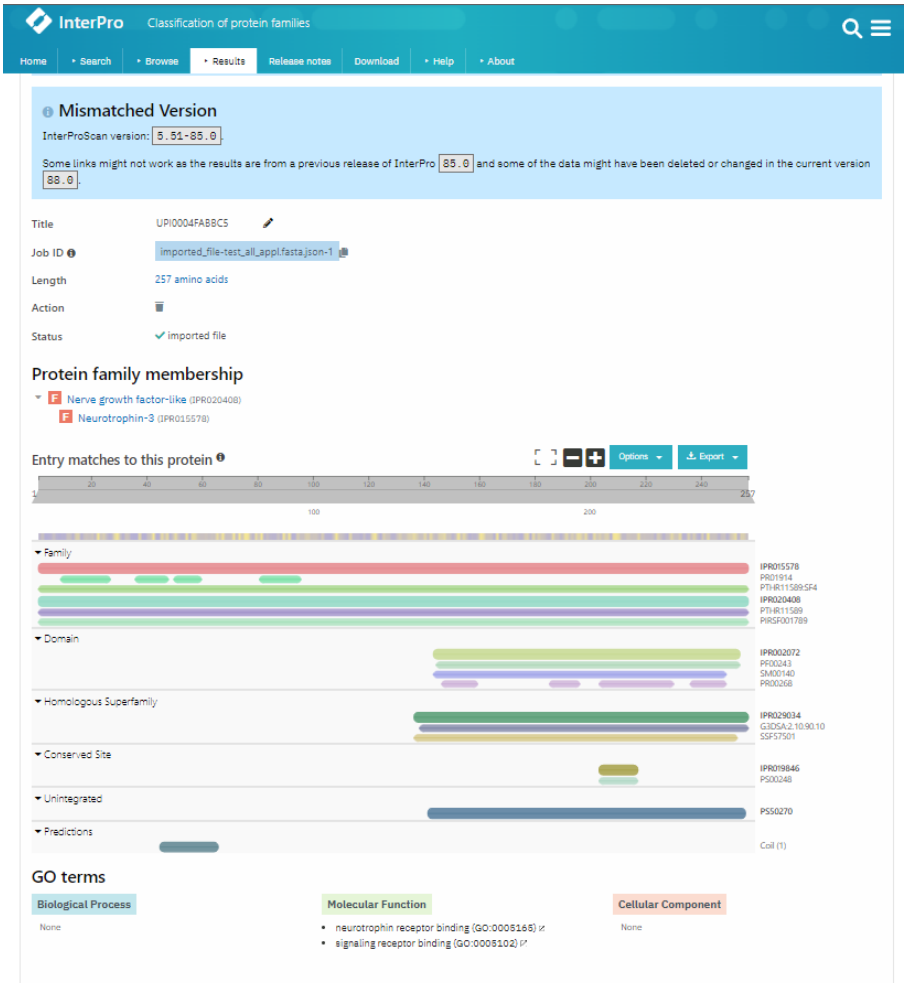


图 7.5: 随后，如同在线 InterProScan 一样解读结果。

```
24/05/2022 15:37:16:515 Running the following analyses:
[CDD-3.18,Coils-2.2.1,Gene3D-4.3.0,Hamap-2020_05,MobiDBLite-2.0,
    PANTHER-15.0,Pfam-33.1,PIRSF-3.10,PIRSR-2021_02,PRINTS-42.0,
    ProSitePatterns-2019_11,ProSiteProfiles-2019_11,SFLD-4,SMART-7.1,
    SUPERFAMILY-1.75,TIGRFAM-15.0]
Pre-calculated match lookup service DISABLED. Please wait for match
    calculations to complete...
24/05/2022 15:37:36:183 25% completed
24/05/2022 15:37:54:073 51% completed
24/05/2022 15:39:20:223 75% completed
24/05/2022 15:39:39:260 90% completed
24/05/2022 15:39:55:692 100% done: InterProScan analyses completed
```

将输出的 HBA_HUMAN.FASTA.json 下载到本地，适用前述方法进行展示，逐条目解读如图 7.6-7.10。

图 7.6: 概览。

7.4 在线 InterProScan

EBI-EMBL 将 InterProScan 部署在其服务器上，可访问网页并在线搜索。

网页搜索十分方便，不必考虑是否预先计算匹配服务、GO 搜索、pathway 搜索或输出文件格式等等问题，也不会遇到软件-数据库版本不合的问题。

仍以 HBA 为进行网页版 InterPro 搜索，搜索结果与本地搜索、网页展示的几乎一致，同时没有版本报错，这体现了 InterPro 数据库是完全基于序列的，同样的序列，必定输出同样的结果。

常用蛋白质数据库 UniProt 提供了指向 InterPro 的链接，当查找已知蛋白质时，可直接定位到 InterPro 条目，例如 HUMAN HBA 的页面。

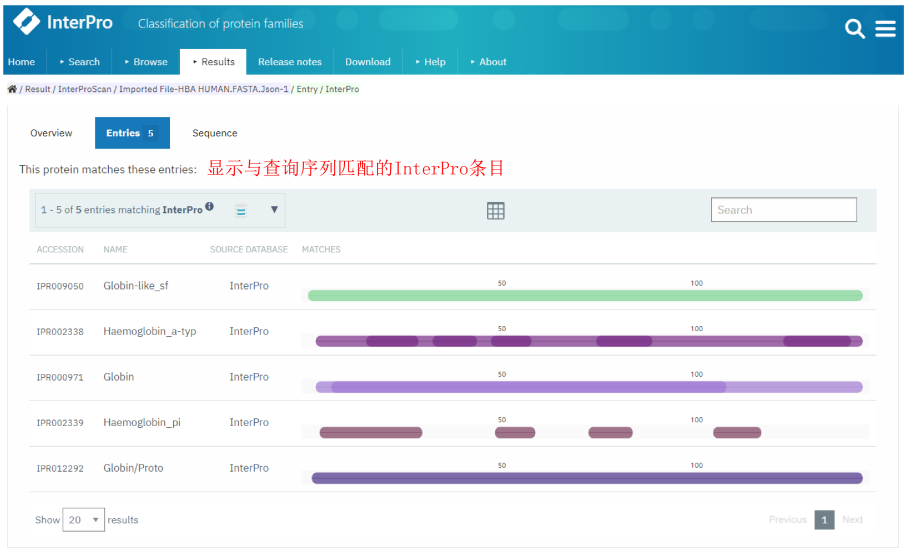


图 7.7: 条目。

图 7.8: GO 条目。

图 7.9: 蛋白质家族信息。

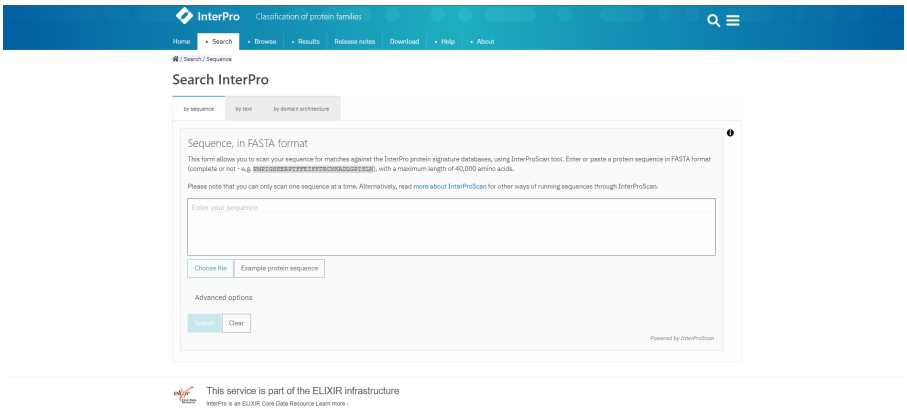


图 7.10: 在线 InterProScan 界面。

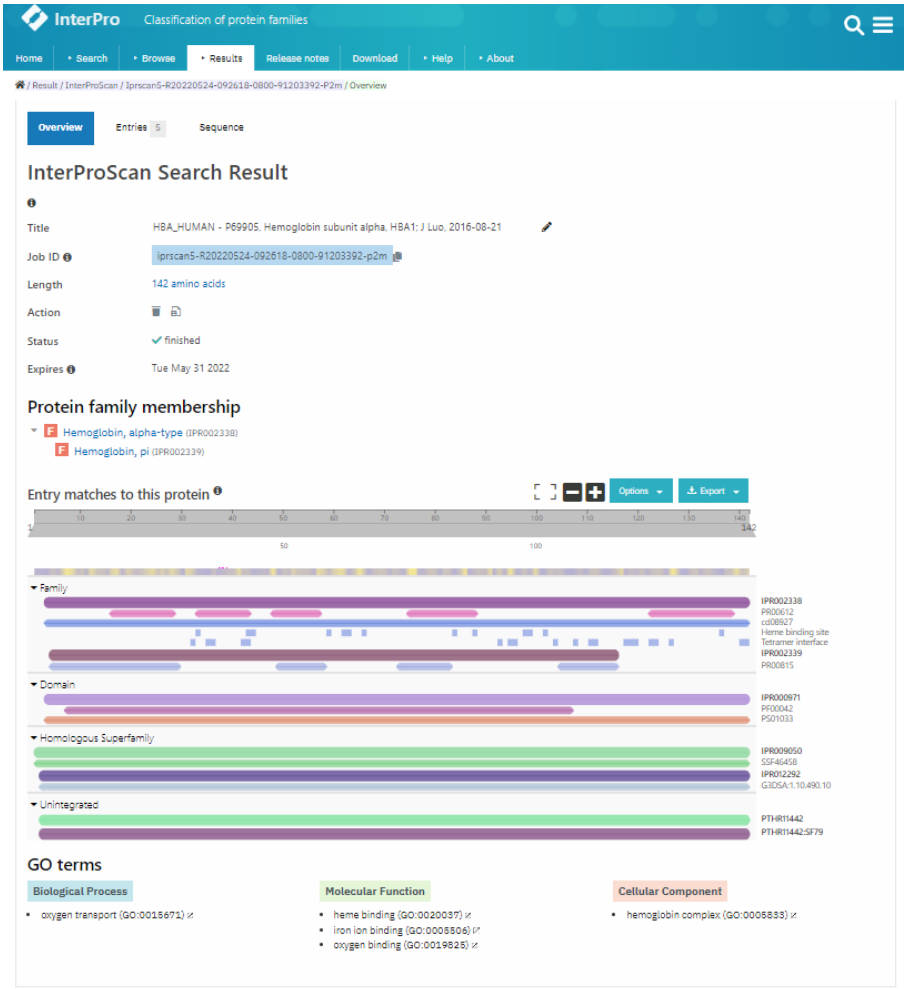


图 7.11: HBA 网页搜索结果。

参考文献

[1] Blum, M., Chang, H. Y., Chuguransky, S., Grego, T., Kandasaamy, S., Mitchell, A., Nuka, G., Paysan-Lafosse, T., Qureshi, M., Raj, S., Richardson, L., Salazar, G. A., Williams, L., Bork, P., Bridge, A., Gough, J., Haft, D. H., Letunic, I., Marchler-Bauer, A., Mi, H., ⋯ Finn, R. D. (2021). The InterPro protein families and domains database: 20 years on. *Nucleic acids research*, 49(D1), D344–D354.

[2] Jones, P., Binns, D., Chang, H. Y., Fraser, M., Li, W., McAnulla, C., McWilliam, H., Maslen, J., Mitchell, A., Nuka, G., Pesseat, S., Quinn, A. F., Sangrador-Vegas, A., Scheremetjew, M., Yong, S. Y., Lopez, R., & Hunter, S. (2014). InterProScan 5: genome-scale protein function classification. *Bioinformatics (Oxford, England)*, 30(9), 1236–1240.

[3] InterProScan Documentation

[4] InterPro Quick Tour

[5] 罗静初. Linux 生物信息技术基础. 课堂练习.

第八章　Web 开发与 MySQL 初步

这一章是新增的，以 ABC 网站作为例子，分析一个网页的组成部分，随后介绍如何制作一个简单的网页，以及 MySQL 的基本命令。

8.1　网页的组成部分

当我们访问一个网站时，浏览器（如 Chrome）将给服务器发送一个请求，随后，服务器将回复，回复的内容包含一些描述网页的文件，浏览器解析、渲染这些文件，于是形成我们能够看见的网页。

描述网页的文件主要有以下几种，它们的作用分别是：

- 超文本标记语言（Hypertext Markup Language, HTML）文件，是网页的基本组成部分。
- 层叠样式表（Cascading Style Sheet, CSS）文件，让网页变好看。
- Javascript 文件，控制动态或交互部件。
- Assets（直译为财产），存放静态资源，例如一张图片。

接下来依次做介绍。

8.2　超文本标记语言 (HTML)

8.2.1　HTML 文件的结构

HTML 文件是网页的骨架，或者结构。它是浏览器用于描述网页内容和结构的语言 (HTML 不是一门编程语言)。直观地说，HTML 是一系列嵌套的盒子。

下面给出一个简单的 HTML 文件，并解释它各个部分的含义：

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Test Page</title>
5     </head>
6     <body>
7         <h1>Heading!</h1>
```

```
 8         <p>Paragraph!</p>
 9     </body>
10 </html>
```

- `<!DOCTYPE html>` 是一个声明，让浏览器采取最新的 HTML 标准解释这个网页。
- `<html>` 和 `</html>` 是一对标签，是这个 HTML 文档的根（root）。
- `<head>` 和 `</head>` 是头部标签对，这两个标签之间可以存放网页的元数据（meta-data）。元数据包括网页的标题、脚本、风格、概括信息。
- `<title>Test Page</title>` 是页面标题标签对，这两个标签之间的是该网页的标题，它将在浏览器中的页面标签上显示出来。HTML 文档中，除了标签之外的东西都是文本，这里，“Test Page” 就是文本。
- `<body>` 和 `</body>` 是主体标签对，这两个标签之间是该网页的主体内容。
- `<h1>` 和 `</h1>` 是一级标题标签对，这两个标签包围了一级标题。
- `<p>` 和 `</p>` 是段落标签对，这两个标签包围了一个段落。

8.2.2　标签和元素

HTML 中的基本标签包括：

<html></html>	HTML 文档根
<head></head>	头部标签，存放有关此文档的数据
<body></body>	文档主体标签
	一级标题
	二级标题，可直到六级标题
	段落标签
	通用块分区标签
	通用行内分区标签

HTML 中用于文本格式化的标签包括：

	粗体文本
<code></code>	计算机代码文本
	强调文本
	斜体文本

此外，HTML 中的注释格式为：`<!-- 注释 -->`。注释中的内容会被浏览器忽略，但适当的注释会使代码更好理解。

初步认识以后，还需要厘清一些概念。是什么组成了 HTML？并不是标签，而是**元素**（elements），标签是元素的一部分。

我们以一个段落元素为例[1]。

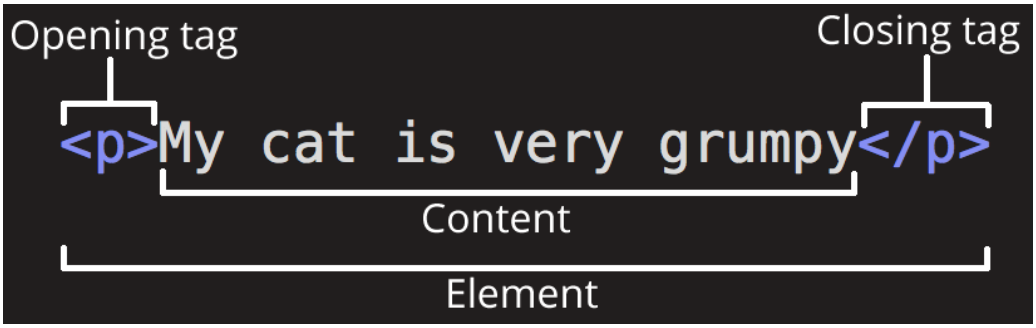


图 8.1: 一个段落元素

一个元素的主要部分包括：

1. 开始标签（Opening tag）：包含元素的名称（本例为 p），被大于号、小于号所包围。表示元素从这里开始或者开始起作用——在本例中即段落由此开始。
2. 结束标签（Closing tag）：与开始标签相似，只是其在元素名之前包含了一个斜杠。这表示着元素的结尾——在本例中即段落在此结束。初学者常常会犯忘记包含结束标签的错误，这可能会产生一些奇怪的结果。
3. 内容（Content）：元素的内容，本例中就是所输入的文本本身。
4. 元素（Element）：开始标签、结束标签与内容相结合，便是一个完整的元素。

元素与元素是可以嵌套，犹如一个盒子里可以包含更小的盒子，例如，`<p>My cat is <strong>very</strong> grumpy.</p>` 。

但是 `<p>My cat is <strong>very grumpy.</p></strong>` 是错误的，因为嵌套的元素必须有正确的层次，犹如两个盒子不能部分交叉（或者用时髦的说法，两个盒子不能穿模）。

8.2.3　属性

元素可以有属性，属性包含了关于元素的一些额外信息，这些信息本身不应显现在内容中，但会被应用到所有内容上。

通过元素属性可以实现带链接的文本，例如：`<a href="http://www.ncbi.nlm.nih.gov/">NCBI</a>` 。这里”a” 的含义是锚（anchor），其超链接的功能由 `href` 属性实现。

注意：属性与元素名称或上一个属性之间应该有空格，属性名称和属性值用等号连接，属性值用双引号包围。

通过元素的属性也可以实现插入图像，并指定其替代文字、宽度、高度和排列方式，例如：`<img src="./gif/abcs.gif"alt="abc0"width="280"height="210"align="bottom">` 。但此元素的特殊之处在于其只有一个开始标签而无内容或结束标签，因为图像元素并不需要内容来产生效果，这样的元素被称为**空元素**。

[1]引自MDN Web Docs

8.2.4 列表

某些时候，你需要在网页中列举一些条目，例如 ABC 网站中的这一部分：

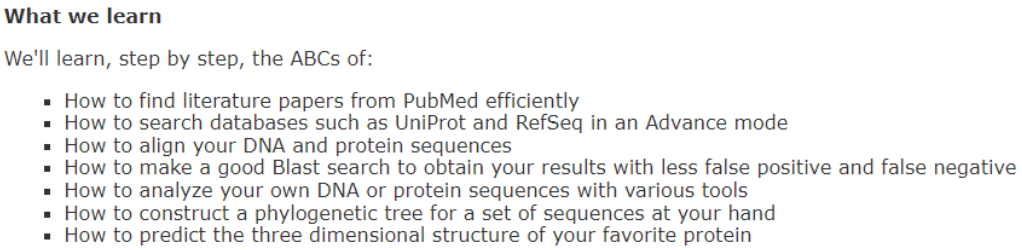


图 8.2: ABC 主页截图

这部分对应的代码是：

```
<b>What we learn</b>
<p>We'll learn, step by step, the ABCs of:</p>

<ul>
    <li>
    How to find literature papers from PubMed efficiently
    </li>
    <li>
    How to search databases such as UniProt and RefSeq in an Advance mode
    </li>
    <li>
    How to align your DNA and protein sequences
    </li>
    <li>
    How to make a good Blast search to obtain your results with less false positive and false negative
    </li>
    <li>
    How to analyze your own DNA or protein sequences with various tools
    </li>
    <li>
    How to construct a phylogenetic tree for a set of sequences at your hand
    </li>
    <li>
```

```
24      How to predict the three dimensional structure of your favorite
       protein
25      </li>
26  </ul>
```

这里， 是一对代表无序列表 (unordered list) 的标签，其中，每一个列表条目 (list item) 都被写在 中。

如果想要实现一个有序列表 (ordered list)，则可将 替换为 。

亦可实现一个定义列表（definition list）：

```
1  <b>你将学到什么？</b>
2  <dl>
3      <dt>计算机技能</dt>
4        <dd>基础Linux命令行操作</dd>
5      <dt>生物信息学技能</dt>
6        <dd>序列比对、BLAST、HMMER、InterPro</dd>
7      <dt>基于项目的技能</dt>
8        <dd>RNA-seq数据分析流程、ChIP-seq数据分析流程、数据库构建</dd>
9      <dt>更多</dt>
10       <dd>你有兴趣尝试的，或与同学探讨的</dd>
11 </dl>
```

效果如下（没有启用 CSS）：

你将学到什么？

计算机技能
　　基础Linux命令行操作
生物信息学技能
　　序列比对、BLAST、HMMER、InterPro
基于项目的技能
　　RNA-seq数据分析流程、ChIP-seq数据分析流程、数据库构建
更多
　　你有兴趣尝试的，或与同学探讨的

图 8.3: 定义列表演示

8.2.5　块

块分区（block section）与行内分区（inline section）分别通过 和 实现。

回顾本节首先给出的简单 HTML 文件，下面对它做些改动，将一级标题和段落放到一个块分区中。

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test Page</title>
    </head>
    <body>
        <div>
            <h1>Heading!</h1>
            <p>Paragraph!</p>
        </div>
    </body>
</html>
```

浏览器打开以后，你将发现，没有任何改变。但实际上，我们获得了一个块。这个块暂时没有任何影响，但如果为它添加一些属性，并配合 CSS 文件，即可使它变得有用起来。块几乎是万能的，但不要在任何地方都使用块，代码要讲究一点语法，其含义应该是浅显易懂的。

8.3　层叠样式表 (CSS)

8.3.1　CSS 文件的结构

CSS 文件是网页的皮肤，是为网页添加样式的代码。它是一套用以描述 HTML 里的元素应该如何表现的规则集 (rule set)，是一系列描述性指令。

HTML 文件应该采用哪一个 CSS 文件装饰自己，需要在头部被指明，如 ABC 网站就指定 `<link rel="stylesheet"href="./css/abc.css"type="text/css">` 。

注意 `href` 属性的内容，就是该网页要使用的 CSS 文件的路径。

我们不妨新建一个 `style` 文件夹，在其中新建一个 `style.css` 文件，其内容为：

```
div {
    color: red;
    font-family: Arial;
    font-size: 24pt;
}
```

保存，而后在上一小节最后的示例 HTML 文件的头部放入 `<link rel="stylesheet" href="./style/style.css"/>` 。

这样，浏览器打开 HTML 文件以后，你将看到被 `<div></div>` 包围的部分会变为红色，Arial 字体，24pt 大小。

我们稍微解释一下这段代码：

- `div`：选择器，指定这个样式的适用范围。
- `color`：属性（property），指定要改变哪方面的样式，在这里是颜色，下面分别是字体族、字体大小。
- `red`：属性的值（value）。
- `color: red;`：属性和属性的值组成一条声明（declaration）。

一条声明是一个单独的规则，整段代码被称为一个规则集。每个规则集（除了选择器的部分）都应该包含在成对的大括号里。在每个声明里要用冒号将属性与属性值分隔开。在每个规则集里要用分号将各个声明分隔开。

CSS 被称为层叠样式表，顾名思义地，这些样式应有层次，有些样式是普遍的（general），而有些样式是特别的（special）。

上面的代码中，选择器指定了一个元素 (element)，但选择器还可以指定一个类 (class)，或者一个 ID 属性，或者一个行内样式 (inline style)。

从最普遍到最特别，排序是：元素、类、ID 属性、行内样式。

特别的规则将覆盖比它广泛的规则，我们仍然以上面的 HTML 示例作为演示：

HTML 文件如下，注意，主体部分有 3 个块，其中第 1 个块没有属性，第 2 个块指定了其类为 `info` ，第 3 个块指定了其 ID 为 `unique` 。

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test Page</title>
        <meta charset="UTF-8">
        <link rel="stylesheet" href="./style.css" />
    </head>
    <body>
        <h1>Heading!</h1>
        <div>Paragraph!</div>
        <div class="info">Info</div>
        <div classs="info" id="unique">Test</div>
    </body>
</html>
```

同路径下，放置 `style.css` 文件，内容如下：

```
div {
    color: red;
    font-family: Arial;
    font-size: 24pt;
    }
```

```
6
7  .info {
8    color: blue;
9    font-family: Arial;
10   font-size: 24pt;
11 }
12
13 #unique {
14   color: green;
15   font-family: Arial;
16   font-size: 24pt;
17 }
```

浏览器打开 HTML 文件，显示如下：

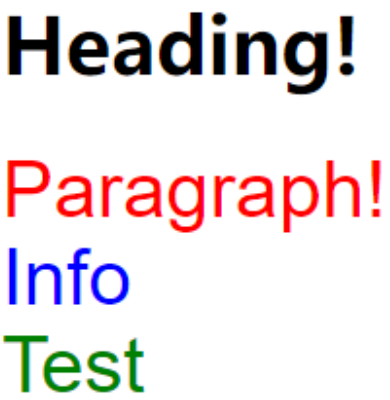


图 8.4: CSS 演示

这里，CSS 文件定义了 3 套规则，分别对应于 `div` 元素、`info` 类、和 `unique` ID。

注意到，HTML 主体部分的第一个块是一个普通的 `div` 元素，没有属性，因此，其样式由 CSS 文件中第一个规则集定义，显示红色字体。第二个块具有属性 `class="info"` ，则，其同时受到 CSS 文件中第一、二个规则集的控制，但由于针对于类的规则集更加具体，因此，两个规则集不同之处，以针对 `info` 类的规则集为准，故显示蓝色字体。第三个块具有两个属性，它属于 `info` 类，但又具有 ID `unique` 。可以看到，CSS 文件中的第三个规则集就是针对 `unique` ID 的，而且是最特殊的，因此这个块的样式将被第三个规则集定义，显示为绿色。

在一个 HTML 文件中，可能有很多元素属于一类，每个元素也可以属于很多类；但每个 ID 都必须是独一无二的，也就是说，一个 ID 只能对应一个元素，一个元素也最多只能对应一个 ID。

由于 ID 很特殊，过多使用 ID 指定样式会让代码变得杂乱，难以管理。因此，最好只使用类来指定样式。

有些 HTML 元素是有默认样式的，如一级标题 `<h1>Heading!</h1>`，你可以在 CSS 中覆盖这些默认样式。

`<div>` 和 `<span>` 是没有默认样式的，可以自由发挥。

CSS 中指定类的样式：

```
.classname {
    property1: value1;
    property2: value2;
}
```

CSS 中指定 ID 的样式：

```
#id {
    property1: value1;
    property2: value2;
}
```

CSS 中的注释用 `/*` 和 `*/` 包围，注释中的内容会被浏览器忽略，但适当的注释会使代码更好理解。特别注意，CSS 中的注释不可嵌套，也不能用 `//` 进行注释。（这不是 C 语言）

CSS 中可以定义变量，下举一例：

```
:root{
    --primary: #396dff;
    --grey: #f7f7f7;
    --white: #fff;
}
.navTitle{
    color:var(--primary);
    font-size: 20px;
    margin: 0;
    font-weight: normal;
}
```

`:root` 中并没有为任何元素或类指定样式，只是为了定义变量而存在。CSS 中的变量以双横线 `--` 开头，此后可以 `var(--varName)` 的形式调用。这个例子中用变量定义颜色，这样就不必每次使用十六进制代码表示颜色，减轻记忆负担。

8.3.2　设计与布局——以 ABC 主页为例

CSS 的重要作用之一是负责网页布局，前面说到，HTML 是嵌套的盒子，CSS 则围绕这些盒子进行装饰。

8.3.2.1　块的属性

一个页面由很多“块”组成，这些块都有如下属性[2]：

- `padding`：内边距，围绕着内容（比如段落）的空间。
- `border`：边框，紧接着内边距的线。
- `margin`：外边距，围绕元素外部的空间。

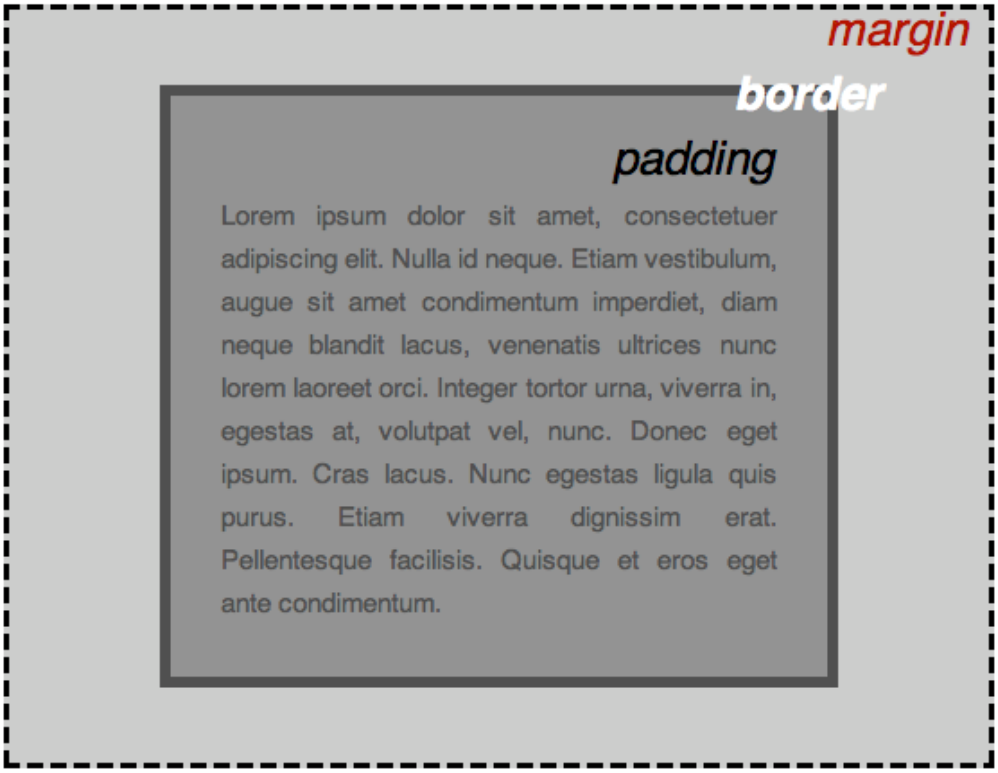


图 8.5: 块布局

此外，还有一些其他属性：

- `width`：元素的宽度
- `background-color`：元素内容和内边距底下的颜色
- `color`：元素内容（通常是文本）的颜色
- `text-shadow`：为元素内的文本设置阴影
- `display`：设置元素的显示模式（暂略）

8.3.2.2　HTML 标签样式

下面尝试以 ABC 主页的 CSS 文件（`abc.css`）选段为例，解读它的页面设计。

```
1  body{
2      font-family: Verdana,Arial,sans-serif;
```

[2]引自MDN Web Docs

```
3     color: #333333;
4     line-height: 1.166;
5     margin: 0px;
6     padding: 0px;
7 }
```

上述代码是文档体格式设置。前两行指定了字体族、默认字体颜色。十六进制颜色代码可以在一些设计网站上方便地查询，这些网站通常提供一整套配色方案，可以按需选用。

margin 和 padding 均设置为 0pt 意味着不设置任何内外边距。

```
1 a:link, a:visited, a:hover {
2     color: #006699;
3     text-decoration: none;
4 }
5
6 a:hover {
7     text-decoration: underline;
8 }
```

上述代码设置了链接的格式，其中：a:link 代表未访问过的链接，a:visited 代表用户已访问过的链接，a:hover 鼠标悬停其上时的链接，a:active 代表正在被点击的链接。

第一个规则集将普通链接、已访问链接、鼠标悬停时的链接定义为深蓝色，并不加任何文本修饰。

第二个规则集将鼠标悬停时的链接增加下划线修饰。综合来看，就得到了如下的效果。

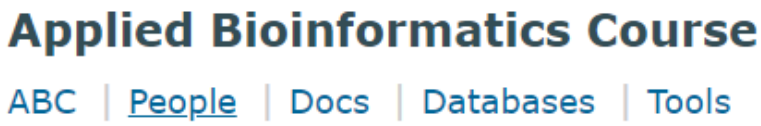


图 8.6: 此时，鼠标正悬停在 People 上

```
1 h1, h2, h3, h4, h5, h6 {
2     font-family: Verdana, Arial,sans-serif;
3     margin: 0px;
4     padding: 0px;
5 }
6
7 h1{
8     font-family: Verdana,Arial,sans-serif;
9     font-size: 120%;
```

```
10      color: #334d55;
11  }
12
13  h2{
14      font-size: 114%;
15      color: #006699;
16  }
17
18  h3{
19      font-size: 100%;
20      color: #334d55;
21  }
22
23  h4{
24      font-size: 100%;
25      font-weight: normal;
26      color: #333333;
27  }
28
29  h5{
30      font-size: 100%;
31      color: #334d55;
32  }
```

上述代码指定了各级标题的样式，各级标题有不同的颜色、大小，但其字体是相同的。注意对字体系列的定义 `font-family: Verdana, Arial,sans-serif;` ，这里有三种字体，当浏览器找不到或不支持第一种字体时，则会尝试第二种字体、第三种字体。如果字体名称有空格，则需要用双引号包围，如经典的 `"Times New Roman"` 。

这里的代码有些冗余，例如第二个规则集中的字体系列声明是不必要的，因为第一个规则集已经指定了同样的字体系列。

```
1  ul{
2      list-style-type: square;
3  }
4
5  ul ul{
6      list-style-type: disc;
7  }
8
9  ul ul ul{
```

```
10      list-style-type: none;
11  }
```

这段代码定义了无序列表和嵌套无序列表的样式，所谓嵌套无序列表，就是一个以无序列表作为更高级无序列表的条目。这里第一级无序列表条目以实心方块标记，第二级以实心圆点标记，第三级没有标记。

```
1  label{
2      font: bold 100% Verdana, Arial,sans-serif;
3      color: #334d55;
4  }
```

这里定义了 <label> 元素的属性，HTML 中的 <label> 元素表示对用户界面中某个元素的说明，一般与 <input> 元素相关联。

8.3.2.3　界面样式

ABC 主页使用了 ID 选择器来指定界面样式，因为界面样式往往不需要复用。

```
1  #masthead {
2    margin: 0;
3    padding: 10px 0px;
4    border-bottom: 1px solid #cccccc;
5    width: 100%;
6  }
7
8  #mastbottom {
9    margin: 0;
10   padding: 10px 0px;
11   border-top: 1px solid #cccccc;
12   width: 100%;
13 }
14
15 #navBar {
16   margin: 0 0 0 79%;
17   padding: 0px;
18   background-color: #eeeeee;
19   border-left: 1px solid #ccc;
20   border-bottom: 1px solid #ccc;
21 }
22
```

```
#content {
  float: left;
  width: 100%;
  margin: 0;
  padding: 0 3% 0 0;
}
```

四个规则集分别指定了页面头、页面底部、导航栏、和内容的布局格式。

注意，如果 padding 或 margin 属性具有 4 个值，分别指代 4 边的内边距，顺序为，上-右-下-左；

但是，如果 padding 或 margin 属性仅有 2 个值，第一个值指代“上方和下方”的边距(上下相同)；第二个值指代“左方和右方”的边距，左右对称。

由于这些样式是由于 ID 选择器指定的，因此在一个 HTML 文档中只能独一无二地运用一次。

8.3.2.4　部件样式

ABC 网页中有很多部件，CSS 文件中也为它们指定了样式。

```
#siteName{
    margin: 0px;
    padding: 0px 0px 10px 10px;
}

#pageName{
        padding: 0px 0px 10px 10px;
}

#globalNav{
    color: #cccccc;
    padding: 0px 0px 0px 10px;
    white-space: nowrap;
}

/*white-sapce: nowrap 指定空格不换行，因此无论浏览器窗口有多小，导航栏
    总是在一行中显示，而不会换行，当无法完全展示时，页面下方会出现横向滚
    动条*/

#globalNav img{
    display: block;
```

```
20 }
21
22 #globalNav a {
23         font-size: 90%;
24         padding: 0px 4px 0px 0px;
25 }
26
27 #breadCrumb{
28         font-size: 80%;
29         padding: 5px 0px 5px 10px;
30     vertical-align: baseline
31 }
```

上述第一个规则集指定站名 “Applied Bioinformatics Course” 的样式。第二个规则集指定页面名的样式，但这一 ID 似乎并未在现在的页面使用。第三个规则集指定全局导航栏 “ABC | People | Docs | Databases | Tools | PDB | UniProt | ExPASy | EBI | NCBI | BIGD ” 的样式，这一样式在 `head.html` 中被用到。第四个规则集指定 `globalNav` ID 中所有 `img` 元素的样式。第五个规则集指定 `globalNav` ID 中锚（链接）的样式。

第六个规则集比较有趣，指定的是面包屑（bread crumb）导航的样式。《格林童话》有汉泽尔与格莱特的故事，这两个小孩试图在进入森林时将面包屑洒在地上以避免迷路。可惜他们失败了，因为面包屑被林中的鸟儿吃掉了。面包屑导航即希望给用户一条 “回家的路”，格式类似于 `首页>分类页>次级分类页` ，但这一部件并不很实用，或许曾在历史版本中有功能，但现在仅仅负责指定网站底部导航栏的样式。

8.3.2.5　类选择器指定的样式

对于一些页面中要使用多次的样式，用 ID 选择器是不明智的，因为 ID 在单个页面中不能重复出现。此时，类选择器就是一个很好的选择，只需要在 HTML 标签的属性中指定 `class=classname` ，即可应用类的样式。下面择要介绍 ABC 网站 CSS 文件中的几个类选择器。

```
1 .feature{
2     padding: 0px 0px 10px 10px;
3     font-size: 80%;
4 }
5
6 .feature h3{
7     padding: 10px 0px 5px 0px;
8     text-align: center;
9 }
```

```
10
11 .feature img{
12     float: right;
13     padding: 0px 10px 0px 0px;
14     margin: 10px 5px 10px 5px;
15 }
16
17 .story{
18     clear: both;
19     padding: 5px 0px 0px 10px;
20     font-size: 80%;
21 }
22
23 .story p{
24     padding: 0px 0px 5px 0px;
25 }
```

feature 类控制了 ABC 网页的主要内容，这里第一个规则指定了主体的排版；第二个规则集指定了其中三级标题的格式，使其居中；第三个规则集指定了其中图片的格式，控制为向右浮动对齐，正如 ABC 主页上的照片。

story 类在网页中常被用到，大段文字一般都放在此类中。第四个规则集指定了其排版，clear: both; 使其必须位于浮动元素的下方，在这里，导致主页 story 框中的文字不会与图片混排。第五个规则集指定了 story 类中段落元素的排版。

如果希望阅读更加详细的介绍，读者可以在互联网上查找相关教程。值得一提的是，由于不断增添内容、不断重写，ABC 网页有些代码或许不那么简洁，有兴趣的读者可以尝试简化 ABC 网页的 CSS 文件。

8.4　网站搭建、ABC 主页与 PHP

8.4.1　Web 开发技术的历史

网络应用开发分为几个阶段，下面对有关历史作一简单介绍[3]。

- 最初的网页仅仅是一个 HTML 文件，只能展示静态内容。为了增加网页的动态效果，javascript 被设计出来，使用户能够与浏览器进行交互。此后，为了对网页进行美化，CSS 应运而生。至此，一个基本的，具有一定美感的，能够执行一定交互功能的网页即告诞生。
- 1993 年，为了实现客户端与服务端的数据交互，通用网关接口（Common Gateway Interface, CGI）出现，它定义了 Web 服务器与外部应用程序之间的通信接口标准。Web

[3]引自参考文献 [1]

服务器通过 CGI 执行外部程序，让外部程序根据 Web 请求内容生成动态的内容。CGI 的主要编写语言是 Perl，但所有符合接口标准的语言均可编写 CGI 程序，CGI 程序运行在服务器上。

- 后来，为了更好地组织 Web 应用的内容，并提高执行效率，人们想到将模板和动态内容进行分别标记。动态内容（程序）被嵌入模板（HTML）中，执行完毕后返回给浏览器。PHP 语言承载了其中的动态内容，并可以与 HTML 混写，这是**第一代 Web 应用框架**的代表。CSS 通过外联方式控制 HTML 标签样式，使代码更容易维护。ASP.net、Java servlets 也属于此类。
- 为了构建复杂 Web 应用，模型-视图-控制器 (Model-View-Controler, MVC) 设计模式被引入 Web 开发，其含义是，将应用进行解耦合，分为模型、视图、和控制器三部分。模型负责封装与业务逻辑相关的数据和数据处理方法，视图是数据的 HTML 展现，控制器负责响应请求及模式和试图间的协调。这就是**第二代框架**，其代表包括著名的 Django（基于 Python 的服务端编程）、Ruby on Rails。
- **第三代框架**以 AngularJS 为代表，是一个在浏览器内运行的 Javascript 框架，并不依赖于特定的服务器端功能（如数据库或 Node.js），且沿用了前代技术的一些概念：MVC、模板（HTML-CSS 描述视图）。

有兴趣的读者还可阅读相关教程，下面介绍几种搭建静态网站的方案。

8.4.2　搭建静态网站

最简单的方式是在现有的博客网站、平台（如知乎、简书）上创建账号。这种方法基本不用写代码，可以直接发布内容。但是，这种方法也会受到一些限制，例如某些平台访客需要登陆后才能查看文章，你的内容并非自由的。

如要追求一定的自主性，同时又不愿意花太多时间在处理网站细节上，可以选择较为成熟的快速网站生成技术。例如Hugo。

Hugo 是一个用 Go 语言编写的静态网站生成器，它使用起来非常简单，仅需要一个二进制文件 hugo(hugo.exe) 即可轻松生成静态页面。

生成页面后，你可以将其托管到 GitHub，并通过 GitHub Pages 发布。具体地，你需要创建一个名为 `username.github.io` 的仓库，其中 `username` 是你的 GitHub 用户名。然后将网页文件放进仓库中，并提交，随后即可访问 username.github.io 来查看你的网页。

然而，由于众所周知的原因，GitHub 使用起来并不很方便。

第三种方案是使用内容管理系统，如 WordPress。这类系统功能丰富，但需要为网站购买服务器及域名以发布。

第四种方案较为原始，即自己手写 HTML、CSS、Javascript 文件，此后亦可通过 GitHub Pages 发布。适合于不需要处理请求的、不需要使用数据库的网站，例如 ABC 主页。ABC 主页是一个静态网站，也即，浏览器请求后收到的是一个 HTML 文件，PHP 代码的运行过程都在服务器上进行。

第五种方案是服务端编程，例如使用 Django 框架，可以自定义前端页面、后台请求处理、数据库结构。同样地，也需要购买服务器及域名。

有兴趣的读者可以自己尝试搭建一个简单的个人博客，Google 可以是你的朋友。

8.4.3　ABC 主页的结构与 PHP 初步

这份笔记不准备详细复述 PHP 语言的语法，而是希望能够在解读 ABC 主页代码的过程中逐步体会 PHP 的使用方式。

关于 PHP 的基础知识，可以参考W3school 教程（中文）或PHP Manual (in English)、参考文献 [7]。

稍微提一下，**LAMP**，缩写自 Linux-Apache-MySQL-PHP，是一组用来运行动态网站或服务器的自由软件（软件栈）。ABC 主页采取的似乎也是这个软件栈。其中，

- **Linux** 是操作系统，第一二章已有介绍。
- **Apache** 是 Web 服务器，通过处理 HTTP 请求而提供服务。
- **MySQL** 是数据库，可将所有信息以易于查询的方式进行存储。
- **PHP** 是编程语言，用于创建动态化的页面（这一过程在服务端执行，最终发送出 HTML 文件）。

LAMP 具有典型的分层架构，Linux 处于最低级别。接下来的层次是 Apache 和 MySQL，最上面是 PHP。虽然 PHP 名义上是顶层，也就是表示层，但 PHP 组件位于 Apache 中。

在服务器启动后端服务 (如 Apache 或 Nginx 后)，在对应的内容文件夹下放置 `index.html` 或 `index.php` ，而后使用 IP 地址或者域名进行访问，对应的文件即可被浏览器解析。这里以 ABC 主页的 `index.php` 为例，我们先忽视一些内容，以求看清楚基本结构：

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http
       ://www.w3.org/TR/html4/loose.dtd">
2  <html>
3
4  <head>
5      <meta http-equiv="Content-Type" content="text/html; charset=GB2312"
           >
6      <title>abc</title>
7      <link rel="stylesheet" href="./css/abc.css" type="text/css">
8      <style type="text/css">
9      </style>
10 </head>
11
12 <body>
13     <?php include './ssi/top0.html'; ?>
14
15     <div id="content">
16         <div class="feature">
```

```
17            <img src="./gif/abcs.gif" alt="abc0" width="280" height="210
                 " align="bottom">
18            <p>
19               <b>Welcome to ABC - the Applied Bioinformatics Course!</
                    b>
20            </p>
21
22            <!--省略段落-->
23
24            <b>What we learn</b>
25            <p>
26               We'll learn, step by step, the ABCs of:
27             </p>
28             <ul>
29                <!--省略列举条目-->
30             </ul>
31             And lots more!
32         </div>
33
34         <div class="story">
35             <h3>How we learn </h3>
36             <!--省略段落-->
37         </div>
38
39         <div class="story">
40             <h3>What you need before the course</h3>
41             <ul>
42                <!--省略列举条目-->
43             </ul>
44         </div>
45         <div class="story">
46             <h3>What you gain from the course</h3>
47             <!--省略段落-->
48         </div>
49     </div>
50
51     <?php include './ssi/bottom.html'; ?>
52     <?php include './ssi/foot.html'; ?>
53 </body>
```

```
54 </html>
```

重点关注主体（body），其中有 4 个部分，这里，PHP include 语句会获取指定文件中存在的所有文本/代码/标记，并复制到使用 include 语句的文件中。特别适合在多张页面上引用相同的 PHP、HTML 或文本。

- <?php include './ssi/top0.html'; ?>：这一部分引用了 top0.html文件 ，这是 ABC 主页的页眉。
- ：这一部分是内容块，应用 CSS 文件中 ##content 规则集，这一块中又包含 4 个块。
  - ：欢迎词及照片。
  - ：介绍学习内容。
  - ：介绍学习方式。
  - ：介绍先修要求。
- <?php include './ssi/bottom.html'; ?>：引用 bottom.html 文件，这是底部导航栏。
- <?php include './ssi/foot.html'; ?>：引用 foot.html 文件，这是页脚。

这里 ssi 目录的含义或许是 Server Side Include。

下面分析 top0.html 文件，

```
1  <div id="masthead">
2     <h1 id="siteName">Applied Bioinformatics Course </h1>
3     <div id="globalNav">
4        <a href="./index.php">ABC</a> |
5        <a href="./people.php">People</a> |
6        <a href="./literatures.php">Docs</a> |
7        <a href="./databases.php">Databases</a> |
8        <a href="./tools.php">Tools</a> |
9        <a href="http://www.rcsb.org/">PDB</a> |
10       <a href="http://www.uniprot.org/">UniProt</a> |
11       <a href="http://www.expasy.org/">ExPASy</a> |
12       <a href="http://www.ebi.ac.uk/">EBI</a> |
13       <a href="https://www.ncbi.nlm.nih.gov/">NCBI</a> |
14       <a href="http://bigd.big.ac.cn/">BIGD</a>
15    </div>
16 </div>
```

这是一个块，包含：

- <h1 id="siteName">Applied Bioinformatics Course </h1>：一级标题（Applied Bioinformatics course），该标题具备 ID，在 CSS 中有对应的规则集。

- `<div id="globalNav">`：块，其角色是全局导航栏，包括 11 个以竖线分隔的超链接，该块具备 ID，在 CSS 中有对应的规则集。

`bottom.html` 文件内容如下：

```
<div id="siteInfo">
    <a href="./about.php">About</a> |
    <a href="./practices.php">Practices</a> |
    <a href="./projects.php">Projects</a> |
    <a href="./talks.php">Talks</a> |
    <a href="./seminars.php">Seminars</a> |
    <a href="./manuals.php">Manuals</a>
</div>
```

包含一个块，其角色是底部全局导航栏，包括 6 个以竖线分隔的超链接，该块具备 ID，在 CSS 中有对应的规则集。

`foot.html` 文件内容如下：

```
16 August 2022,
<a href="./jcl.php">J Luo</a>,
<a href="http://www.cbi.pku.edu.cn/">CBI</a>,
<a href="http://www.pku.edu.cn/">PKU</a>, Beijing, China
```

包含一串文本（网页最后更新时间）和三个超链接（指向网页作者介绍页、北京大学生物信息中心主页、北京大学主页）。

当我们通过链接转向其他页面，例如 People 页时，页眉、底部导航栏、页脚不会改变，但页面内容，即 `<div id="content">` 标签包围的部分改变了。

由于主页（`index.php` ）与 People 页（`people.php` ）是两个文件，因此 `content` 这个 ID 可以复用。

至此，我们应该已经了解了 ABC 主页的基本结构，并且能够基于它改造一个小组主页/个人主页了。

值得一提的是，虽然 ABC 网站并不复杂，也难以与很多精美的个人博客相比较，但是它丰富的内容（相信有心寻找的人会在其中获得很大的收获。）和悠久的历史，已经足够珍贵。

图 8.7: ABC 主页的历史。左上，2005 年 12 月 8 日；右上，2008 年 6 月 20 日；左下，2011 年 8 月 30 日；右下，2014 年 6 月 19 日。存档于互联网档案馆

8.5　Javascript 简介

前面提到，HTML+CSS 可以实现一个静态网站，却不能进行互动。JavaScript 可以实现这一点。

JavaScript(简称 JS) 是一种操纵网页内容的编程语言，被绝大多数网站和 Web 应用所采用。JS 与 Java 没有什么关系。

我们可以在浏览器控制台执行 JS 代码，在 Chrome 浏览器中，按 Ctrl + Shift + J 可以打开控制台。

8.5.1　数据类型和运算符

JS 有五种基本数据类型：

- 布尔型
- 数值
- 字符串，由引号包围，可以为空字符串
- null 型，没有值，可以使用 nulls 型来清空一个变量
- 未定义型，声明但不赋值

JS 的基本运算符包括：

- 算数运算符：`+ - * /`
- 比较运算符：相等 `===` ；不等 `!==` ；小于 `<` ；大于 `>` ；小于等于 `<=` ；大于等于 `>=`

JS 的注释以 `//` 开始。

JS 每行语句需要以分号结尾。

JS 会忽略空格。

8.5.1.1　定义变量和常量

JS 需要先声明变量，才能使用变量，可用 `=` 对变量进行赋值：

```
let myBoolean = true;
let myNumber = 12;
let myString = "Hello World!";

myBoolean = false;
myNumber = -5.6;
myString = "";
```

为了代码的安全，可以定义常量，常量是不能修改的：

```
const answerToLife = 6.148;
answerToLife = 42 //无效
```

可声明未定义变量，而后对其赋值，可以 null 清空一个变量。

```
let firstName; //未定义类型
firstName = "Albert" //字符串
firstName = null; //清空变量
```

不要使用 var 去声明变量。

8.5.1.2　输出和警告

有时我们希望能够输出些中间结果来检验程序的正确性，类似于 Python 中的 print ，JS 使用 console.log() 将文本输出到控制台。

alert() 将生成一个包含内容的弹出型通知，也可用于 debug。

8.5.2　条件和循环

JS 条件结构示例 (问好程序)：

```
if (hour < 12){
    console.log("Good morning!");
}else if (hour < 16){
    console.log("Good afternoon!");
}else if (hour < 20){
    console.log("Good evening!");
}else{
    console.log("Good night!");
}
```

JS 循环结构示例：

```
let z=1;
while (z<1000) {
    z = z*2;
    console.log(z);
}
```

迭代示例 (列举宠物)：

```
const pets = ["cat", "dog", "guinea pig", "bird"];
for (let i=0;i< pets.length; i++){
    const phrase = "I love my " + pets[i];
    console.log(phrase);
}
```

8.5.3　函数、回调函数、匿名函数

JS 中函数的基本结构为 (parameters)=> { body }; 。我们可以看一个简单的例子 (摄氏度转华氏度):

```
1  const celsiusToFahrenheit = tempC => {
2      const tempF = tempC * 1.8 + 32;
3      return tempF;
4  }
5
6  console.log(celsiusToFahrenheit(10);) //50
```

JS 中函数可以像变量那样作为一种参数传递传递，例如下面这个程序:

```
1  const addTwo = x =>{
2      return x+2;
3  }
4
5  const modifyArray = (array, callback) => {
6      for (let i=0; i < array.length; i++){
7          array[i] = callback(array[i]);
8      }
9  }
10 let myArray = [5,10,15,20];
11 modifyArray(myArray, addTwo); //[7,12,17,22]
```

其中，addTwo 是一个函数，而 addTwo(x) 是 addTwo 函数以 x 为输入时的返回值。modifyArray 需要的第二个参数是一个函数，而非一个返回值，这一点需要特别注意。

```
1  const modifyArray = (array, callback) => {
2      for (let i=0; i < array.length; i++){
3          array[i] = callback(array[i]);
4      }
5  }
6  let myArray = [5,10,15,20];
7  modifyArray(myArray, x =>{
8      return x+2;
9  }); //[7,12,17,22]
```

当一个函数实现的功能太过简单时，不必要先定义、命名一个函数而后使用，而是可以直接创建并使用它，这种函数称为匿名函数。

JS 包含一些较复杂数据结构，例如列表，它们具有一些内建的函数 (或者说是方法)，例

如实现栈的压入和弹出：

```
let pets = ["cat", "dog", "guinea pig", "bird"];
console.log(pets[3]); //"bird"
pets[2] = "hamster";
pets.pop(); // remove from the end
pets.push("rabbit"); //add to the end
```

map() 方法可以将一个回调函数应用到一个列表中的每个元素，例如：

```
let myArray = [1,2,3,4,5];
let modifiedArray = myArray.map(x => x*3);
// modifiedarray === [3,6,9,12,15]
```

filter() 方法可以将旧列表中符合条件的元素挑出来构建一个新列表：

```
let values = [3,-6,9,-12,15];
let positiveValues = values.filter(x => x>0);
// modifiedarray === [3,9,15]
```

8.5.4　对象与类

对象是一些属性的集合，每个属性由有一个名字，每个属性对应一个值。我们可以创建一个 myCar 类：

```
const myCar = {
    make : "Ford",
    model: "Mustang",
    year : 2005,
    color: "red"
};
```

可以指定输出或调用对象的某些属性：

```
console.log(myCar.model);
console.log(myCar["color"]);
```

可以像使用变量那样使用对象，但要注意，对象本身实际上是一个索引，指向数据存储的位置。即使两个对象的所有属性和值都相同，但由于它们是分别创建的，因此存储在不同的位置，故在用 === 判断时，输出的结果仍然是 false 。

一种拷贝对象的方式是：

```
let arr = [1,2,3];
```

```
let copyArr = [...arr];
```

JS 的类与 Python 相似，每个类都办好一个构建器，能够创建该类对应的实例。例如以 Rectangle 类为例，指定其宽和高就可以创建一个长方形的实例。

```
class Rectangle{
    constructor(width, height){
    this.width = width;
    this.height = height;
}
}
const samllRect = new Rectangle(3,4);
const bigRect = new Rectangle(15,11);
```

实例的属性用 this 指定，类似于 python 中的 self 。

类中可以包含方法，例如上述长方形类中即可包含面积计算的方法：

```
class Rectangle{
    constructor(width, height){
        this.width = width;
        this.height = height;
    }
    getArea = () => {
        return this.width * this.height;
    };
}
const samllRect = new Rectangle(3,4);
const bigRect = new Rectangle(15,11);
console.log(smallRect.getArea()); // 48
```

类中的方法可以被该类的所有实例所使用。

至此，JS 的基本语法介绍完毕，更多在 Web 开发中的应用，请参考JavaScript —Dynamic client-side scripting。

8.6　MySQL 初步

8.6.1　数据库相关概念

作为初学者，首先应该明确一些概念：

数据库（database）是保存有组织的数据的容器，通常是一个或一组文件。这里应该注意区分数据库和**数据库管理系统**（database management system, DBMS），后者是创建和操纵数

据库的工具。

表（table）是某种特定类型数据的结构化清单。数据库中每个表都有一个唯一的名字。

表由一个或多个**列**（column）构成，每个列都有相应的**数据类型**（datatype），数据类型限制了该列中存储的数据。

行（row）是表中的一个记录，一行有时也称为一个数据库记录（record）。每个行都有可以唯一标识自己的一列，这一（组）列称为主键（primary key），主键用来表示一个特定的行。表中的任何列都可以作为主键，只要其任意两行都不具有相同的主键值，且每行都有一个主键值。

模式（schema）是关于数据库和表的布局及特性的信息，定义了数据在表中如何存储。

SQL，即**结构化查询语言**（Structure Query Language），是专门用来与数据库通信的语言。

8.6.2　MySQL 及其安装配置

MySQL 是一个数据库管理系统，分为客户端-服务器两部分。

客户端部分用于提交请求给服务器软件，服务器软件部分直接处理数据库文件。客户端可能是 MySQL Client，编程语言、Web 开发语言（如 PHP）等，服务器软件为 MySQL DBMS。

安装 MySQL 的命令如下：

```
1 sudo apt install mysql-server # 需要root权限
```

安装完毕后，MySQL 将自动启动。MySQL 自带了一个简易的客户端——MySQL Client，我们以后都用它登陆 MySQL 服务器软件。注意区分，MySQL Server 和 MySQL Client 的是不同的程序。MySQL Client 的可执行程序是 `mysql` ，MySQL Server 的可执行程序是 `mysqld` 。我们进入的是 Client。

验证安装，检查服务器版本：

```
1 $ mysqladmin --version
2 # 输出结果：mysqladmin Ver 8.0.30-0ubuntu0.22.04.1 for Linux on x86_64
      ((Ubuntu))
```

初次登陆，首先利用 MySQL Client 连接到 MySQL 服务器，执行下面的命令（*root 用户初始密码为空*）：

```
1 $ sudo mysql
2 # 以下是输出
3 Welcome to the MySQL monitor. Commands end with ; or \g.
4 Your MySQL connection id is 9
5 Server version: 8.0.30-0ubuntu0.22.04.1 (Ubuntu)
6
7 Copyright (c) 2000, 2022, Oracle and/or its affiliates.
```

```
8
9  Oracle is a registered trademark of Oracle Corporation and/or its
10 affiliates. Other names may be trademarks of their respective
11 owners.
12
13 Type 'help;' or '\h' for help. Type '\c' to clear the current input
       statement.
14
15 mysql>
```

此时即已进入 MySQL Client，注意到提示符已经不再是 `$` ，而是 `mysql>` 。尝试列出所有数据库：

```
1  mysql> SHOW DATABASES;
2  +--------------------+
3  | Database           |
4  +--------------------+
5  | information_schema |
6  | mysql              |
7  | performance_schema |
8  | sys                |
9  +--------------------+
10 4 rows in set (0.00 sec)
```

在 MySQL Client 中输入 `exit` 可退出 MySQL Client。

创建 root 用户的密码：

```
1  sudo mysqladmin -u root password "YOUR_PASSWD"
```

再次进入时，就需要输入密码：

```
1  $ sudo mysql -u root -p
2  Enter password:
```

创建新用户：

```
1  mysql> CREATE USER 'leb1c'@'localhost' IDENTIFIED BY 'my_passwd';
```

授予权限 (这里对用户 `leb1c` 授予了所有数据库的所有权限，但只能在服务器上本地登陆)：

```
1  mysql> GRANT ALL ON *.* TO 'leb1c'@'localhost';
```

此时退出，即可以新用户的身份登陆了，新身份无需 `sudo` 获得超级用户权限：

```
$ mysql -u leb1c -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 8.0.30-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    statement.

mysql>
```

8.6.3　MySQL 数据库操作

在 MySQL Client 中，输入 \h 可以显示帮助。MySQL Client 中的命令都以分号 ; 结束。

MySQL Client 支持按上箭头查看上一条命令，支持制表键补全命令。

MySQL 的关键字、函数名、列名不区分大小写，但运行在 Linux 上的 MySQL 服务器对数据库名和表名区分大小写，插入到表中的数据也是区分大小写的。

在 SQL 语句中直接书写的字符串、日期或者数字等称为常数。字符串和日期需要用英文单引号 '' 括起来，数字可以直接书写，多个单词之间需要以空格或换行符分隔。

SQL 语句中不能出现中文标点符号和全角空格。

MySQL 单行注释以 ## 开始，## 后面直接添加注释内容。多行注释格式为:

```
/*
注释内容
*/
```

SHOW DATABASES 可以查看当前用户权限范围内的数据库:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
```

```
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

SHOW DATABASES 可以使用 LIKE 子句，可显示符合要求的数据库，如 SHOW DATABASES LIKE 'db%' 命令可显示所有名称以 db 开头的数据库，这里 % 的作用类似于 Linux 命令行中的 * 通配符。在管理的数据库较多时，该命令比较有用。

上述数据库都是首次启动 MySQL 时自动创建的，初学者不应该随便乱动。我们创建一个新的数据库用于演示，CREATE DATABASE name_of_database 用于创建数据库：

```
mysql> CREATE DATABASE leb_students;
Query OK, 1 row affected (0.04 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| leb_students       |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

MySQL 不允许在同一系统下创建两个相同名称的数据库，因此，同一条创建数据库的命令第二次运行时就会出错。

为了避免这种情况，可为使用 IF NOT EXISTS 子句，即 CREATE DATABASE IF NOT EXISTS leb_students 。

CREATE DATABASES 还可选其他子句，如 DEFAULT CHARACTER SET utf8 子句可指定字符集，默认为系统的字符集。如要处理的数据库中包含中文信息，应该指定为 utf8 。

MySQL 管理许多数据库，用户在使用前需要选择一个数据库进行操作，USE 命令可以指定数据库，实现在不同数据库中跳转：

```
mysql> USE leb_students
Database changed
```

出现 Database changed，表示选择数据库成功。

8.6.4 MySQL 数据表操作

当选择一个数据库后，SHOW TABLES 可以展示数据库中的数据表：

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

我们的课程小组组长 R 君想要建立一个表来管理本组同学的一些信息，但当前数据库中并没有表，故他使用 CREATE TABLE 创建：

```
mysql> CREATE TABLE group1 (No CHAR(4), Name CHAR(20), Year INT(4));
Query OK, 0 rows affected, 1 warning (0.14 sec)

mysql> SHOW TABLES;
+------------------------+
| Tables_in_leb_students |
+------------------------+
| group1                 |
+------------------------+
1 row in set (0.00 sec)
```

括号内的内容是表定义选项，包含列名，列定义或更多内容。这里 R 君想要记录每位同学的编号（字符）、姓名（字符）和入学年份（数字）。

如要删除表，可使用 DROP TABLE group1 ，应注意，用户必须有执行此命令的权限才能删除数据表。

此时 group1 表中没有任何内容，R 君现在尝试向其中插入几条数据：

```
mysql> INSERT INTO group1 (No, Name, Year)
-> VALUES
-> ("G1A", "RAO XC", 2020);
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO group1 (No, Name, Year) VALUES ("G1B", "LIU PY",
    2020);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO group1 (No, Name, Year) VALUES ("G1C", "LI S",
    2019);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO group1 (No, Name, Year) VALUES ("G1D", "QIAN BY",
    2019);
```

```
13 Query OK, 1 row affected (0.04 sec)
14
15 mysql> INSERT INTO group1 (No, Name, Year) VALUES ("G1D", 'LLLL',
       2019);
16 Query OK, 1 row affected (0.02 sec)
```

INSERT INTO 命令需要指定要插入的表格名、该表格的列名、以及对应的值。MySQL 会用 VALUES 列表中的相应值填入列表中的对应项。VALUES 按其指定的测序匹配列名，不一定要按每个列实际出现在表中的次序，其优点是，即使表的结构改变了，该语句仍然可以正常工作。

可以使用 SELECT 语句检索数据：

```
1  mysql> SELECT * FROM group1; #显示整个表，*是通配符代表选择所有列
2  +------+---------+------+
3  | No   | Name    | Year |
4  +------+---------+------+
5  | G1A  | RAO XC  | 2020 |
6  | G1B  | LIU PY  | 2020 |
7  | G1C  | LI S    | 2019 |
8  | G1D  | QIAN BY | 2019 |
9  | G1D  | LLLL    | 2019 |
10 +------+---------+------+
11 5 rows in set (0.00 sec)
12 mysql> SELECT No from group1; #只显示No列
13 +------+
14 | No   |
15 +------+
16 | G1A  |
17 | G1B  |
18 | G1C  |
19 | G1D  |
20 | G1D  |
21 +------+
22 5 rows in set (0.00 sec)
```

课程开始一段时间后，有人提出，本组似乎只有 4 名同学，并不存在一位 LLLL 。

R 君听说以后大惊，连忙检查了一遍，发现果真如此，上表中的第 5 行需要删除。他用 DELETE 命令删除特定的行：

```
1  mysql> DELETE FROM group1 WHERE Name="LLLL";
```

```
2  Query OK, 1 row affected (0.04 sec)
3
4  mysql> SELECT * from group1;
5  +------+---------+------+
6  | No   | Name    | Year |
7  +------+---------+------+
8  | G1A  | RAO XC  | 2020 |
9  | G1B  | LIU PY  | 2020 |
10 | G1C  | LI S    | 2019 |
11 | G1D  | QIAN BY | 2019 |
12 +------+---------+------+
13 4 rows in set (0.00 sec)
```

这里 DELETE FROM 指定了从哪个表格中删除，而 WHERE 子句指定了要删除的行。如果没有 WHERE 子句，该命令将删除每一行。

上课一段时间后，R 君发现 QIAN BY 同学格外年长，他是 2009 年入学的，并不是 2019 年，因此想要将第 4 行数据的 Year 条目值改为 2001 ，此时应使用 UPDATE 语句。

```
1  mysql> UPDATE group1 SET Year=2001 WHERE Name="QIAN BY";
2  Query OK, 1 row affected (0.03 sec)
3  Rows matched: 1 Changed: 1 Warnings: 0
4
5  mysql> SELECT * from group1;
6  +------+---------+------+
7  | No   | Name    | Year |
8  +------+---------+------+
9  | G1A  | RAO XC  | 2020 |
10 | G1B  | LIU PY  | 2020 |
11 | G1C  | LI S    | 2019 |
12 | G1D  | QIAN BY | 2001 |
13 +------+---------+------+
14 4 rows in set (0.00 sec)
```

这里 UPDATE 制定了要更改的表，SET 指定了要更改的值及其所在的列，WHERE 子句指定了要更改的行。如果没有 WHERE 子句，该命令将更改每一行的 Year 。

至此，基本 MySQL 用法介绍完毕，读者应该可以自己在 MySQL 中创建数据库、表格并管理数据，更多的 SQL 语句和深入的 MySQL 使用方法，请阅读参考文献 [4]、[5]、[6]。

MySQL 在用 PHP 进行的 Web 开发中应用较为广泛，PHP 脚本可以连接 MySQL、查询、修改数据。但我并不参与相关期末项目，因此本章只介绍基本语法，有兴趣的读者可以阅读参考文献 [5]，其代码均有 PHP 语言的实现，这里不抄录。

参考文献

[1] Web 开发技术的演变，技术发展史
[2] 如何搭建个人博客
[3] MDN Web Docs
[4] 福尔塔 (Forta, B.), 刘晓霞, & 钟鸣. (2009). *mysql* 必知必会. 人民邮电出版社.
[5] 菜鸟教程-MySQL 教程
[6] MySQL Documentation of ORACLE
[7] PHP 语言参考
[8] 罗静初. Linux 生物信息技术基础. 课堂练习.
[9] 朱擎国. Linux 生物信息技术基础课程期末个人总结.

第九章　RNA-seq 实验与分析

基于下一代测序的 **RNA 测序** (RNA sequencing, RNA-seq) 能够揭示 RNA 的存在和数量，本身是一种非常强大的技术。而实验方法的不断改进和浩浩荡荡的的商业化过程又使得 RNA-seq" 飞入寻常百姓家"，不再复杂神秘。

近年来，由于某些热门领域的影响，掌握一定的测序数据分析技能已经成为了一种时尚，而如果能绘制一点图片就更好了。因此，培训班和教程层出不穷，它们是更好的学习资源。

但即使不期望在信息学上有所洞见，也不希图能够构建或读懂软件，只是学习一点实验原理和流程想必也是有好处的。抱着这样的想法，本章先介绍下一代测序、文库构建实验，然后完成一个完整的生物信息学分析流程和结果绘图。

本章的代码是课程助教提供的，有修改。

9.1　下一代测序

长核酸序列测定的首个实用方法由 Sanger 在 1975 年建立，称为 "加减法" [1]；1977 年，Maxam 和 Gilbert 发明了化学法，其基本原理是用特异化学试剂[2]作用于 4 种碱基，使末端标记的 DNA 分子被切成不同长度的末端都是特异的碱基的片段，变性电泳后读出序列。同年，Sanger 等对于 "加减法" 作出重要改进，提出了 "终止法"，即以双脱氧核苷三磷酸在随机位置终止反应，而后以变性凝胶电泳直接读出 DNA 序列的方法，这是第一代测序技术的基本原理。

下一代测序（Next Generation Sequencing, NGS）是相对于第一代测序技术而言的，尽管这一概念已经提出十几年，但它仍然是描述高通量测序的流行语。2006 年，Genome Analyzer（第一代 Solexa 测序仪）上市，随后主宰了高通量测序市场，次年，Illumina 收购了 Solexa。Illumina/Solexa 测序 (以下简称 Illumina 测序) 的基本原理是：桥式扩增（bridge amplification）和循环可逆终止（cyclic reversible termination, CRT）测序。Illumina 提供了形象的讲解视频，清晰易懂。

9.1.1　桥式扩增

测序载玻片上共价连接了很多正向和反向引物 (图 9.1 中红色和蓝色短片段)，密度很高。测序文库被变性后与这些固定的引物退火，开始第一轮延伸。第一个循环完毕后，即得到一

[1]详见参考文献 [1]、[2]。

[2]详见参考文献 [1]。

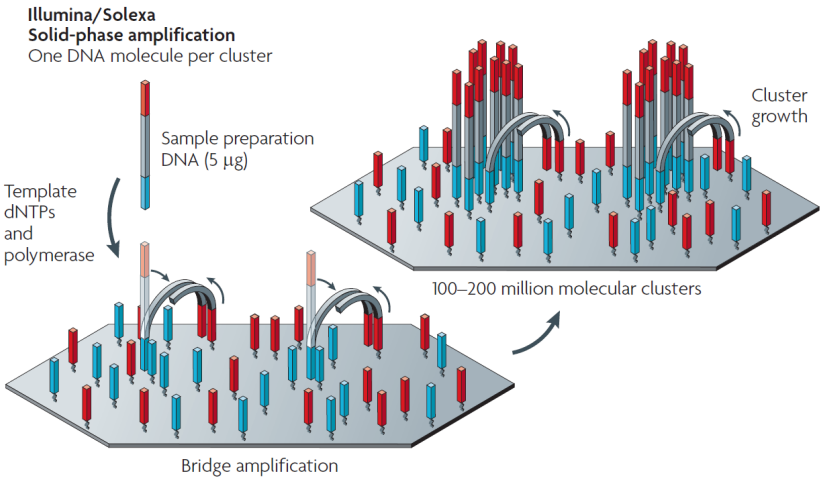


图 9.1: 桥式 PCR 示意图。引自参考文献 [4]。

条固定在测序载玻片上的文库片段，该文库片段再经变性，其 3' 端与被固定的引物退火-延伸（延伸产物形似桥），又得到第二条固定在载玻片上的文库片段。如此反复进行多轮桥式 PCR，得到很多片段簇（cluster），一个簇中包含很多序列相同的带相同接头的片段。

桥式 PCR 使测序模板被固定在了一个固相表面上，同时，不同的片段被分散在测序载玻片上，成很多空间上相互分隔的簇，这使高通量的检测成为可能。

9.1.2　循环可逆终止

循环可逆终止（CRT）包括 3 个步骤：核苷酸掺入、荧光成像、切除荧光染料基团和终止基团。

第一步：聚合酶在测序引物后添加 1 个核苷酸，由于核苷酸终止基团的存在，聚合酶无法添加第 2 个核苷酸。

第二步：洗去未被掺入的核苷酸，荧光成像，记录。

第三步：切除被掺入的核苷酸的荧光染料基团，移除终止基团解放 3'-OH，洗去残余试剂，准备下一次掺入。

如何做到循环可逆终止是一个关键问题，Illumina 测序使用了染料标记的修饰核苷酸（dye-labelled modified nucleotides）。如右图所示。红色基团代表 3' OH 的终止基团 (3'-*O*-叠氮甲基，3'-*O*-azidomethyl，可使用还原剂 Tris(2-carboxyethyl)phosphine (TCEP) 来释放 3'-OH)，dye 代表了荧光染料基团，蓝色部分代表一段 linker，黑色箭头指示了在第三步切割位置。

Illumina 测序在每次测序循环中使用 4 种染料分别标记 4 种核苷酸，因此每次循环将得到一张 4 色图像，如下图所示。

9.1.3　Illumina 测序平台

十几年来，Illumina 推出了多种型号的 NGS 测序平台，它们在原理上大同小异，但又各有所长。最早的测序平台是 Genome Analyzer (GA)，目前被广泛采用的测序平台包括：HiSeq、

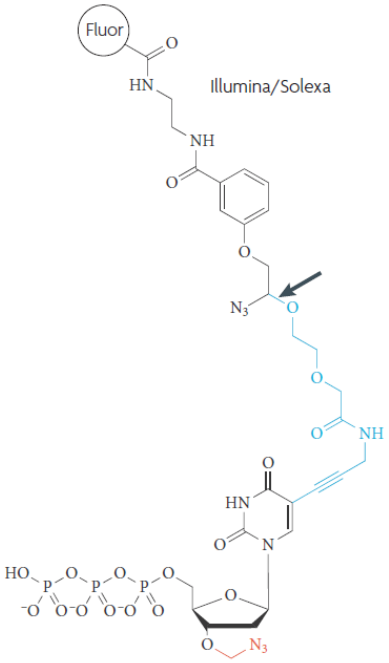


图 9.2: 染料标记的修饰核苷酸

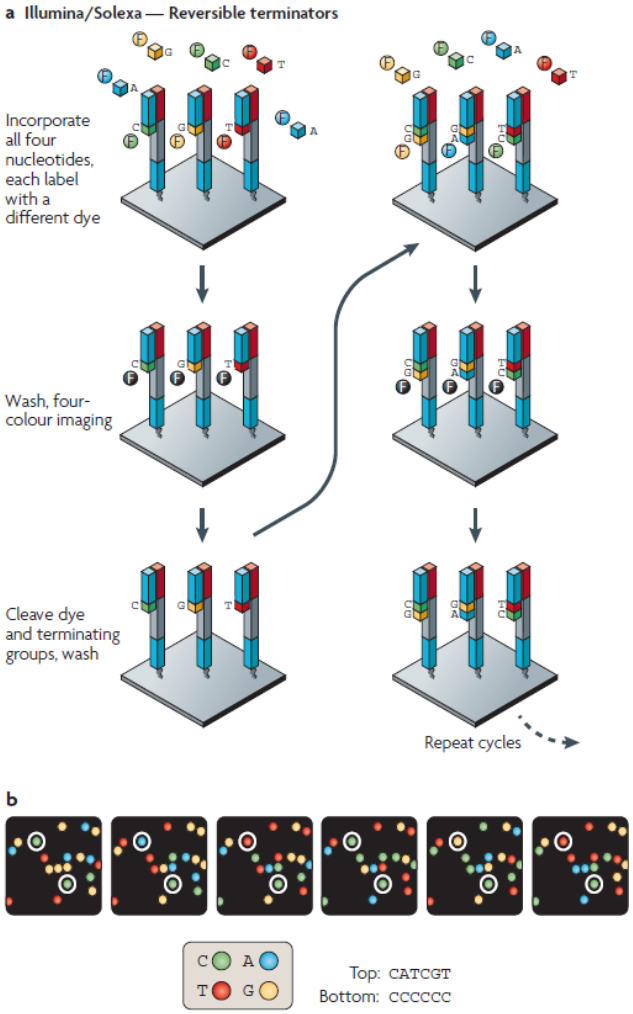


图 9.3: Illumina 采用的四色循环可逆终止方法。引自参考文献 [4]。

HiSeq X Ten、MiSeq、NextSeq 和 NovaSeq。

其中，NovaSeq 发布于 2017 年，有两个较大的改进：

- 可运行 S1、S2、S3 和 S4 四种不同类型的图案化流动槽，通量上面具有较大的灵活性，运行一次可产生 0.5-6 Tb 数据, 但是一条 lane 的数据量较大，增加了 pooling 和数据拆分的难度
- 采用改进的 2 色边合成变成测序 (SBS) 化学 (T= 绿色, C= 红色, A= 绿色/红色, G= 无信号) 和 3 代实时分析 (Real-Time Analysis) 系统，读写速度更快

具体的技术细节不是课程的核心内容，更加详细的介绍可以访问 Illumina 公司或测序服务提供商的网页。

9.2 RNA-seq

9.2.1 一般原理

RNA 测序（RNA sequencing, RNA-seq）出现于 2008 年，由多个研究组同时发表。时至今日，RNA-seq 已经发展出多种变体，无论方法如何变化，使用 NGS 的 RNA-seq 都需要用逆转录酶将 RNA 转换成 DNA。

逆转录酶发挥作用需要引物，根据使用随机引物或 Oligo dT 引物，RNA-seq 分为两个分支。随机引物可能结合到任何 RNA 的任何位置，因此一般用与测序总 RNA；Oligo dT 结合到 mRNA 的 polyA 尾，一般用于测序 mRNA，即转录本。

在得到 cDNA 以后，下一个技术上的问题是，从样品中提取的 RNA 并不多，因此 cDNA 的量也较少，在测序之前，先要进行扩增。通用的扩增方法是聚合酶链式反应（PCR），然而，RNA 及其对应的 cDNA 序列多种多样，无法一一设计引物（事实上也不可能）。所以，需要在扩增之前为所有 RNA 添加统一的两种接头。

但此时又有新的问题，即 NGS 读长短，但 RNA 较长（m RNA 长度一般超过 1000bp），无法一次测得，因此需要被破碎。

综合考虑以上这些问题，技术开发者提出的方案是（mRNA 测序）：

1. 使用连有 Oligo dT 的磁珠富集带有 PolyA 尾的 mRNA
2. 使用超声打断 mRNA
3. 使用 Oligo dT 引物逆转录 mRNA，合成 cDNA 第一链
4. 合成 cDNA 第二链
5. 使用连接酶为 cDNA 两端加上不同的接头
6. PCR 扩增文库
7. 上机测序

很明显，这个流程有些问题。因为超声打断在逆转录之前，故只有 mRNA3‘端最接近 polyA 尾的一段能够被测序，因此，该流程只能对转录本进行大致的定量，却忽略选择性剪接等需要全长转录本信息才能了解的变异。

聪明的读者可以自己思考构建全长文库的方法。

技术本身更新换代很快，这是一件令人高兴又叹惋的事情。一旦新的替代品出现，旧的技术就会无人问津，成为一些文章 *Introduction* 部分论资排辈报出的菜名。研究老的方法，宛如考古。

目前，最广泛采用的文库构建解决方案是 Illumina Truseq RNA 建库方法，可以称为” 事实上的” 标准方法，其基本步骤如图 9.5 所示。

9.2.2　文库的结构

上述” 标准方法” 的接头连接步骤细节及构建的文库结构如图 9.4 所示。

完整文库结构包括”P5-Index2-Rd1 SP-DNA Insert-Rd2 SP-Index1-P7”。

其中，Index1 与 Index2 是文库的**标签**，为了多路复用（在一次上机，一个芯片中同时测多个文库）而设计，所有的文库都具有相同的结构，但由于其 Index 不同，因此数据下机后可按 Index 进行拆分。

P5 和 P7 是**流动槽结合接头**，位于文库的两端，因此也是 PCR（无论是为了扩增而进行的 PCR，还是为了定量而进行的 qPCR）使用的引物。上一节提到，NGS 需要先进行桥式扩增形成序列簇，序列簇正是靠这一段序列首次结合到固定在流动槽底部的接头上的。

Rd1 SP 和 Rd2 SP 分别是 Read 1 Sequencing Primer 和 Read 2 Sequencing Primer 的缩写，与真正的测序引物序列相同，因此测序引物可以结合在此序列的互补序列上，进行测序反应，最终会得到 Read1 和 Read2 的数据。

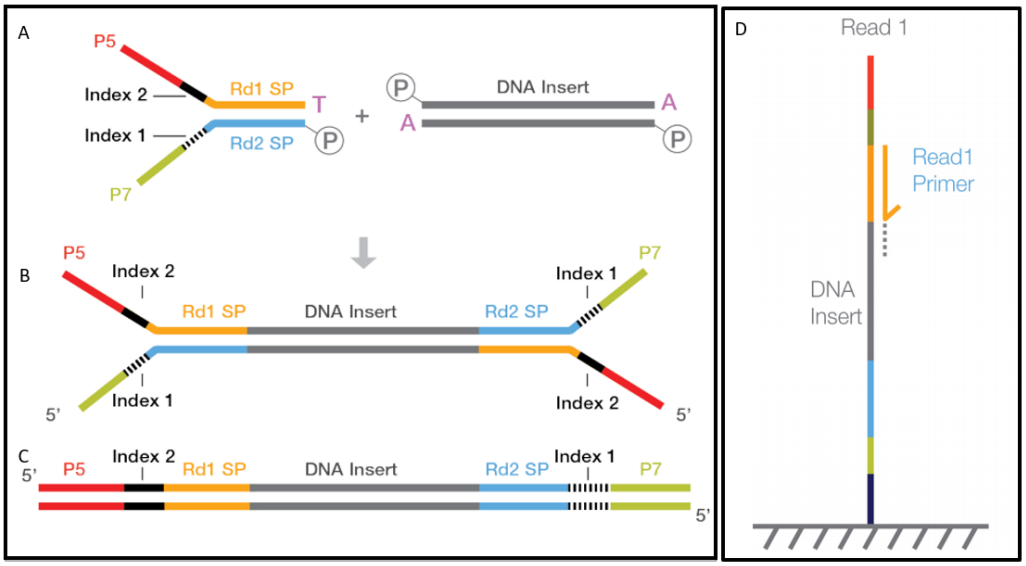


图 9.4: 接头连接和文库结构。接头连接依赖于末端加 A 和磷酸化构成的黏性末端。引自参考文献 [13]

9.2.3　方法变体

许多年过去，RNA-seq 已经发展出很多变体，这些变体往往服务于某种特定的需求，常常具有较为有趣的缩写。下面给出一些方法，有兴趣的读者可以自行了解。

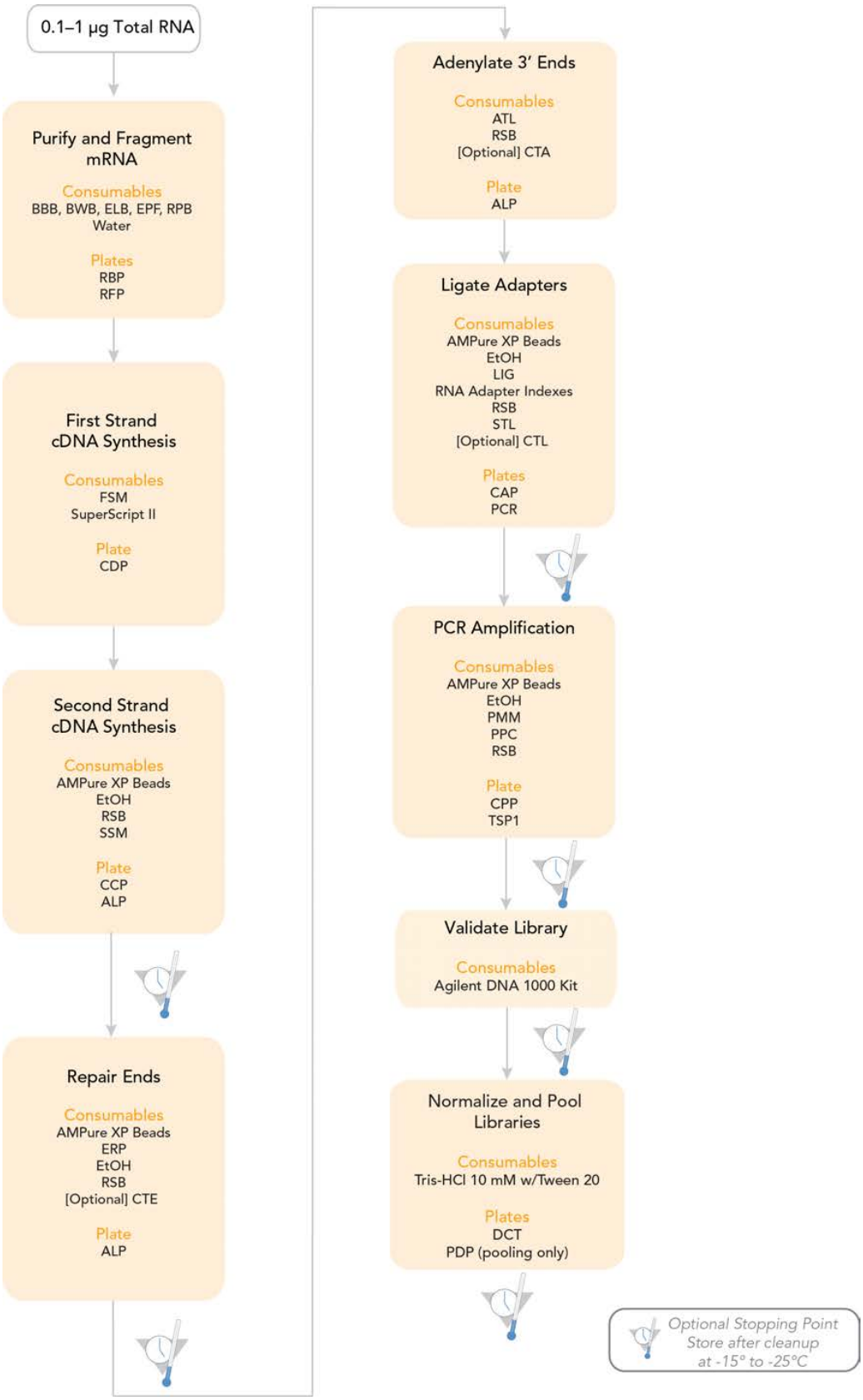


图 9.5: Illumina TruSeq RNA Library Preparation v2 基本流程。引自参考文献 [12]

针对 RNA-seq 本身进行优化，或使它在转录本检测方面具有更多功能的变体：TAIL-seq, PAL-seq, GRO-seq, PRO-seq, CAGE。

为研究 RNA-蛋白质相互作用而开发的变体方法：Ribo-seq, CLIP-seq, eCLIP, DLAF, CLASH。

为鉴定或研究 RNA 上的化学修饰而开发的变体方法：MeRIP-seq, PSI-seq, Pseudo-seq, ICE。

为研究 RNA 结构而开发的变体方法：SHAPE-seq, PARS-seq, Cap-seq。

低起始量 RNA 建库方法：TRACE-seq, scRNA-seq, CEL-seq, Smart-seq, Smart-seq2, Smart-seq3, Drop-seq。

9.3　概述和准备工作

9.3.1　数据来源及实验设计

本分析示例数据来源于文献 Zhao, Y., Zhang, Z., Gao, J., Wang, P., Hu, T., Wang, Z., Hou, Y. J., Wan, Y., Liu, W., Xie, S., Lu, T., Xue, L., Liu, Y., Macho, A. P., Tao, W. A., Bressan, R. A., & Zhu, J. K. (2018). Arabidopsis Duodecuple Mutant of PYL ABA Receptors Reveals PYL Repression of ABA-Independent SnRK2 Activity. *Cell reports*, 23(11), 3340–3351.e5.。

这是一项植物生理学研究工作，我们重点关注其部分 RNA-seq 实验，在 SRA 中对应的实验编号为：SRP145580，该实验研究的是 *pyl112458* 和 *379101112* 突变体对脱落酸和渗透胁迫的基因表达反应。

该实验的整体设计是：野生型和突变体幼苗在处理前均垂直生长 9 天，分别进行：不处理、100 µM ABA 处理、在含 300mM 甘露醇的滤纸上处理 24 小时。转录组用 HiSeq2500 平台进行深度测序进行分析，每组进行三个重复。

几乎所有在野生型籽苗中响应 ABA 的基因表达反应在上述两种突变体中都减弱了。然而响应渗透胁迫的的转录本水平变化在突变体中却很少消失，大多数的响应仍然保持着，即使突变体中很多转录本的上调和下调幅度都减弱了。

在SRA Run Selector页面我们可以找到每一份数据对应的实验条件和基本信息(包括 SRR 号，GEO 登录号，碱基数，基因型，处理条件)，这里只节录分析时用到的 Run。

Run	Bases	Genotype	GEO_Accession	Treatment
SRR7160928	5.27 G	wildtype	GSM3140728	300 mM mannitol for 24 hours
SRR7160929	4.80 G	wildtype	GSM3140729	300 mM mannitol for 24 hours
SRR7160930	5.03 G	wildtype	GSM3140730	300 mM mannitol for 24 hours
SRR7160931	4.58 G	wildtype	GSM3140731	1/2 MS for 24 hours
SRR7160932	5.00 G	wildtype	GSM3140732	1/2 MS for 24 hours
SRR7160933	4.71 G	wildtype	GSM3140733	1/2 MS for 24 hours

上表中可见，6 份 RNA-seq 数据中，SRR7160928、SRR7160929，SRR7160930 是野生型幼苗渗透胁迫处理，SRR7160931、SRR7160932、SRR7160933 是野生型幼苗对照组。

据此，我们预期最后分析得到的差异表达基因应该富集在渗透胁迫反应功能上。

9.3.2　配置环境

为了不污染服务器环境，应该在虚拟环境中进行后来的操作。关于建立虚拟环境，有多种方案，此流程中使用 conda ，安装 conda 的方法请参考第一章。

执行下面的命令创建和激活一个名为 rnaseq 的环境，并安装需要用到的软件。

```
conda create -n rnaseq
conda activate rnaseq
conda install -c bioconda sra-tools
conda install -c bioconda fastqc
conda install -c bioconda trimmomatic
conda install -c bioconda hisat2
conda install -c bioconda samtools
conda install -c bioconda stringtie
```

9.3.3　下载参考文件和测序数据

新建 ./genome 目录，在其中下载参考文件：

```
# 参考基因组TAIR10
wget -b -c http://ftp.ensemblgenomes.org/pub/plants/release-54/fasta/
    arabidopsis_thaliana/dna/Arabidopsis_thaliana.TAIR10.dna.toplevel.
    fa.gz
# GTF文件
wget -b -c http://ftp.ensemblgenomes.org/pub/plants/release-54/gtf/
    arabidopsis_thaliana/Arabidopsis_thaliana.TAIR10.54.gtf.gz
# GFF3文件
wget -b -c http://ftp.ensemblgenomes.org/pub/plants/release-54/gff3/
    arabidopsis_thaliana/Arabidopsis_thaliana.TAIR10.54.gff3.gz

# 选项
# -c 设置wget自动断点续传
# -b 启动下载后转入后台执行
```

下载完毕后，使用 gunzip 命令解压缩。

接下来还要获得接头和引物序列，根据实际使用的引物、文献列出的引物或者测序平台提供的引物确定，这里已经有现成的文件 Truseq-PE.fa ：

```
>PrefixPE/1
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT
>PrefixPE/2
CAAGCAGAAGACGGCATACGAGATCGGTCTCGGCATTCCTGCTGAACCGCTCTTCCGATCT
>PCR_Primer1
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT
>PCR_Primer1_rc
AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT
>PCR_Primer2
CAAGCAGAAGACGGCATACGAGATCGGTCTCGGCATTCCTGCTGAACCGCTCTTCCGATCT
>PCR_Primer2_rc
AGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCGTCTTCTGCTTG
>FlowCell1
TTTTTTTTTTAATGATACGGCGACCACCGAGATCTACAC
>FlowCell2
TTTTTTTTTTCAAGCAGAAGACGGCATACGA
```

接下来下载测序数据文件：

```
SRR7160928_1.fastq SRR7160929_1.fastq SRR7160930_1.fastq SRR7160931_1.
    fastq SRR7160932_1.fastq SRR7160933_1.fastq
SRR7160928_2.fastq SRR7160929_2.fastq SRR7160930_2.fastq SRR7160931_2.
    fastq SRR7160932_2.fastq SRR7160933_2.fastq
```

根据 SRR 编号，指定下载路径，用 wget 下载到 ./data 目录下，此处使用循环下载所有的文件：

执行以下脚本：

```
#! /bin/bash
mkdir data
for i in {28..33}
do
    access="SRR71609${i}"
    wget -c -b -O ./data/${access}.sra "https://sra-pub-run-odp.s3.
        amazonaws.com/sra/${access}/${access}"
done

# 选项
# -O 指定下载文件所在的文件夹和文件名
```

注意到，下载的文件是 SRA 格式（Sequence Read Archive），我们需要将其转换为 fastq 格式，并拆分为 Read1 与 Read2。这里需要用到 SRA Toolkit，建议手动安装并配置路径，也可以使用 conda 安装。随后，执行下面的脚本转换格式并拆分数据：

```
#! /bin/bash
for i in {28..33}
do
    access="SRR71609${i}"
    fastq-dump --split-3 -O . data/${access}.sra &
done

# fastq-dump 是sra-tools中的程序，可用于下载或拆分SRA数据，这里只用它来拆分数据
# --split-3 将双端测序分为两份，放在不同的文件，但是对于一方有而一方没有的reads会单独放在一个文件夹里
# -O 指定输出的文件夹
# data/${access}.sra 要拆分的文件
```

此时，文件目录为：

```
.
    data
        SRR7160928_1.fastq
        SRR7160928_2.fastq
        SRR7160928.sra
        SRR7160929_1.fastq
        SRR7160929_2.fastq
        SRR7160929.sra
        SRR7160930_1.fastq
        SRR7160930_2.fastq
        SRR7160930.sra
        SRR7160931_1.fastq
        SRR7160931_2.fastq
        SRR7160931.sra
        SRR7160932_1.fastq
        SRR7160932_2.fastq
        SRR7160932.sra
        SRR7160933_1.fastq
        SRR7160933_2.fastq
        SRR7160933.sra
```

```
21      genome
22          Arabidopsis_thaliana.TAIR10.54.gff3
23          Arabidopsis_thaliana.TAIR10.54.gtf
24          Arabidopsis_thaliana.TAIR10.dna.toplevel.fa
25          Truseq-PE.fa
```

得到这些文件以后，就可以开始分析了。

9.4　RNA-seq 分析 I

RNA-seq 的分析，一般分为所谓“上游”和“下游”，前者指在 Linux 终端进行的，使用不同的工具从测序数据产生表达计数矩阵的分析过程；后者指拿到表达计数矩阵后，用 R 或 Python 等脚本语言进行绘图的分析过程。这里只是一提，区分这些概念没有什么意义，只需全部完成一遍即可。

9.4.1　序列修剪

首先创建工作目录：`mkdir workdir`，并在其中为每个实验组创建子目录。

目录结构如下：

```
1  workdir/
2      SRR7160928
3      SRR7160929
4      SRR7160930
5      SRR7160931
6      SRR7160932
7      SRR7160933
```

第一步是序列修剪，因为测序平台的不同，测序的接头的序列也会不同，用户需要自己指定所使用的接头种类（可能污染的接头种类）。

常用的去接头软件包括 `trimmomatic` 和 `cutadapt`，这里使用前者，并以一个样品结果的处理为例详细解读。

`trimmomatic` 有两种修剪，第一种，如果一个序列来源于技术而非样品，显然是应该去除的 (这种修剪较为复杂，分为简单模式和回文模式，后者适合于检出“接头通读”式 reads)；第二种，如果一个序列质量太差，则即使它来源于样品，也是应该去除的 (通过滑动窗口过滤和最大信息过滤)。

```
1  trimmomatic PE -threads 9 data/SRR7160928_1.fastq data/SRR7160928_2.
       fastq -baseout workdir/SRR7160928/SRR7160928 ILLUMINACLIP:./
       genome_annotation/Truseq-PE.fa:2:30:10 LEADING:5 TRAILING:5
       SLIDINGWINDOW:5:20 MINLEN:36 AVGQUAL:20
```

```
2
3  # 选项
4  # PE 指定双端测序模式
5  # -threads 8 指定线程数
6  # data/SRR7160928_1.fastq data/SRR7160928_2.fastq 指定两个输入文件
7  # -baseout workdir/SRR7160928/SRR7160928 指定输出文件的存储路径和基本文
       件名
8  # ILLUMINACLIP:./genome_annotation/Truseq-PE.fa:2:30:10 下文详细解释
9  # LEADING:5 设定碱基质量阈值，从reads起始端开始切除质量低于此的碱基
10 # TRAILING:5 设定碱基质量阈值，从reads末端开始切除质量低于此的碱基
11 # SLIDINGWINDOW:5:20 滑动窗口剪切，第一个值设定窗口大小，第二个值设定平
       均质量阈值，这里检测整个reads，如某个窗口中平均碱基质量低于20，则将这
       个窗口整体切除
12 # MINLEN:36 设定reads长度阈值，如果剪切后reads长度低于此则丢弃整条reads
13 # AVGQUAL:20 设定reads平均碱基质量阈值，如果剪切后reads平均碱基质量低于
       此则丢弃整条reads
```

ILLUMINACLIP 选项中，第一个参数 ./genome_annotation/Truseq-PE.fa 代表要过滤的接头和引物序列，因为这些序列并不是来自于生物样品，而是实验过程中引入的。

类似于一些序列比对软件，trimmomatic 先将接头和引物序列片段切成种子，与 reads 进行比对；得到匹配良好的词对后（如 BLAST 中的 HSP），再进行全长比对。

第二个参数代表种子搜索允许的错配碱基数，这里设置为 2。

第三个参数针对 PE 的回文修剪模式[3]下，需要 R1 和 R2 之间至少多少比对分值，才会进行接头切除，这里取 30。

第四个参数是切除接头序列的最低比对分值，这里设置为 10。

为了对每个样品都进行一样的处理，可使用循环：

```
1 #! /bin/bash
2 for i in {28..33}
3 do
4     access="SRR71609${i}"
5     trimmomatic PE -threads 8 data/${access}_1.fastq data/${access}_2.
          fastq -baseout workdir/${access}/${access} ILLUMINACLIP:./
          genome/Truseq-PE.fa:2:30:10 LEADING:5 TRAILING:5 SLIDINGWINDOW
          :5:20 MINLEN:36 AVGQUAL:20 &
6 done
```

trimmomatic 将输出一些运行信息，如下所示，稍作解释：

[3]请阅读参考文献 [1][2][3]

```
# 命令
TrimmomaticPE: Started with arguments:
-threads 56 data/SRR7160928_1.fastq data/SRR7160928_2.fastq -baseout workdir/SRR7160928/SRR7160928 ILLUMINACLIP:./genome_annotation/Truseq-PE.fa:2:30:10 LEADING:5 TRAILING:5 SLIDINGWINDOW:5:20 MINLEN:36 AVGQUAL:20
# 输出文件，其中P代表Paired，即两端都保留的reads，U代表Unpaired，即仅有一段保留者
Using templated Output files: workdir/SRR7160928/SRR7160928_1P workdir/SRR7160928/SRR7160928_1U workdir/SRR7160928/SRR7160928_2P workdir/SRR7160928/SRR7160928_2U
# 欲切除的引物和接头
Using PrefixPair: 'AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT' and 'CAAGCAGAAGACGGCATACGAGATCGGTCTCGGCATTCCTGCTGAACCGCTCTTCCGATCT'
Using Long Clipping Sequence: 'AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT'
Using Long Clipping Sequence: 'AGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCGTCTTCTGCTTG'
Using Long Clipping Sequence: 'TTTTTTTTTTAATGATACGGCGACCACCGAGATCTACAC'
Using Long Clipping Sequence: 'TTTTTTTTTTCAAGCAGAAGACGGCATACGA'
Using Long Clipping Sequence: 'CAAGCAGAAGACGGCATACGAGATCGGTCTCGGCATTCCTGCTGAACCGCTCTTCCGATCT'
Using Long Clipping Sequence: 'AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT'
ILLUMINACLIP: Using 1 prefix pairs, 6 forward/reverse sequences, 0 forward only sequences, 0 reverse only sequences
# trimmomatic会自动识别碱基质量编码方式，因此无需在命令中指定
Quality encoding detected as phred33
# 给出简单的统计结果，两端保留的，前向保留的，反向保留的，丢弃的reads的条目和占比
Input Read Pairs: 20924237 Both Surviving: 18744203 (89.58%) Forward Only Surviving: 1832502 (8.76%) Reverse Only Surviving: 128862 (0.62%) Dropped: 218670 (1.05%)
TrimmomaticPE: Completed successfully
```

此时，workdir 目录的结构为：

```
workdir/
```

```
2     SRR7160928
3         SRR7160928_1P
4         SRR7160928_1U
5         SRR7160928_2P
6         SRR7160928_2U
7     SRR7160929
8         SRR7160929_1P
9         SRR7160929_1U
10        SRR7160929_2P
11        SRR7160929_2U
12    SRR7160930
13        SRR7160930_1P
14        SRR7160930_1U
15        SRR7160930_2P
16        SRR7160930_2U
17    SRR7160931
18        SRR7160931_1P
19        SRR7160931_1U
20        SRR7160931_2P
21        SRR7160931_2U
22    SRR7160932
23        SRR7160932_1P
24        SRR7160932_1U
25        SRR7160932_2P
26        SRR7160932_2U
27    SRR7160933
28        SRR7160933_1P
29        SRR7160933_1U
30        SRR7160933_2P
31        SRR7160933_2U
```

`trimmomatic` 输出的文件中，后缀为 `1P` 的是正向配对 reads，后缀为 `1U` 的是正向未配对 reads，后缀为 `2P` 的是反向配对 reads，后缀为 `2U` 的是反向未配对 reads。

9.4.2　质量控制

9.4.2.1　`fastq` 文件结构

当前数据文件是以 `fastq` 文件格式保存的，其中的 `q` 即代表 Quality，它比 `fasta` 更有用的地方就在于整合了碱基质量的信息，这也是质量控制的基础。因此，了解一点 `fastq` 文

件的结构是很有必要的。

fastq 文件中，一个序列由 4 行构成，我们以 SRR7160928_1.fastq 的最后 4 行为例：

```
1 @SRR7160928.20924237 20924237 length=126
2 CTGAATCACAGGGATAGTGTGAAGATCGACCAAAGTGAATTTATCAGAAGCCAAATACTTGGACTCAC
    CAAGCCTGTGTTCGTAAACATCGAGGACCTTGGCTAGCTTAGCCTCTTCTTCTTCAAC
3 +SRR7160928.20924237 20924237 length=126
4 BBBB/B/F/FFFFFFFFFFFFFFFFF<FF/<//FF<FFFFFFFFFFFFFFF/<FFFFFFFFF/BBBF<<
    FF/<<<FFBBFFFF<FFFFFF/BFFB//FFFF/BF/FFFF//FFBBBFFFB<FFFF
```

- 第一行以 @ 开头，之后为序列的标识符以及描述信息（与 FASTA 格式的描述行类似）
- 第二行为序列信息
- 第三行以 + 开头，之后可以再次加上序列的标识及描述信息（可选）
- 第四行为质量得分信息，与第二行的序列相对应，长度必须与第二行相同

这里面最复杂的是碱基质量得分信息，它有两种编码方式：phred33 和 phred64 （碱基质量得分是由 phred 程序开发者定义的）。相比于这些字符编码，更加直观的碱基质量表示是**碱基错误率**（P_{error}）（因为 CCD 捕捉的信号荧光信号并不永远清晰易分辨），错误率可以从信号本身的特征计算出来。

碱基错误率 (P_{error}) 与碱基质量得分 (Q) 的关系是：

$$Q = -10 \log_{10} P_{error}$$

如果 $P_{error} = 1$，则 $Q = 0$；如果 $P_{error} = 0figure.001$，则 $Q = 30$。

出于节约存储空间的考虑，需要将可能的碱基质量得分对应于单个符号，而非像 30 这样的两位数字。

美国信息交换标准代码（American Standard Code for Information Interchange, ASCII）就是一个简便的方案，ASCII 是一套字符编码方案，1963 年被作为通过电话线传送文本信息的标准而发明，其单个字符由 7 个 bit 表示，因此总共可以表示 $2^7 = 128$ 个字符，编号为 0-127。修订后的 ASCII 表如下：

表中，0-31 号字符和 127 号字符控制码，用于控制电传打字机，无法在屏幕上显示。剩余的字符是可打印字符，包括大写字母、小写字母、数字和基本标点。

像上面所说的，Q 的取值一般在 0-几十这个范围活动，如果直接对应，很多质量值都会对应到控制码，无法显示。因此，必须为 Q 加上某个数值，使得可能的取值都能对应上可打印字符。phred33 方案选择 $Q = Q + 33$，而 phred64 方案选择 $Q = Q + 64$。

如上面例子中的碱基质量值 F，查 ASCII 表知，F 编号为 70，又由于该文件采用 phred33 方案，则原始 Q 值为 $70 - 33 = 37$。则对应的 $P = 10^{\frac{Q}{-10}} = 10^{-3.7} \approx 0.0002$。碱基错误率仅仅为 0.02%，其质量是非常好的。

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	″	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	’	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

图 9.6: ASCII 表

9.4.2.2　fastqc 实践

进行质量控制的脚本如下：

```
#! /bin/bash
for i in {28..33}
do
    input_file="SRR71609${i}"
    fastqc -t 4 workdir/${input_file}/${input_file}_1P workdir/${
        input_file}/${input_file}_2P -o workdir/${input_file}/
done
```

参数较为简单，-t 指定线程数，-o 指定输出目录，还需要给程序提供一个实验组的 fastq 文件（即 read1 和 read2）。

fastqc 的结果以 HTML 文件的形式呈现，下载到本地使用浏览器查看；如果想要自行绘图，则需要在 fastqc.zip 中寻找所需的数据段。

9.4.2.3　fastqc 结果

下面以 SRR7160928_1P_fastqc.html 为例，解读 fastqc 结果 (打开 HTML 文件)，报告分为 10 个部分，在左侧有导航栏和图标提示（绿色为通过，黄色为警告，红色为未通过）：

1. 基本统计量 (Basic Statistics)

如图 9.7，该部分给出被质控数据的基本信息和统计值。包括：

- 文件名
- 文件类型
- 质量值编码方式
- 总序列数：处理的序列总数的计数
- 被标记的低质量序列的计数
- 序列长度：提供集合中最短和最长序列的长度，如果所有序列的长度相同，则只报告一个值
- %GC：所有序列中所有碱基的总体 GC 百分比

该部分永远不会发出警告或报告不通过。

2. 每个碱基序列质量 (Per base sequence quality)

如图 9.8，该部分显示 FASTQ 文件中每个位置的所有碱基的质量值范围，x 轴代表碱基在 read 中的位置，y 轴代表碱基的质量分数，分数越高质量越好。FastQC 将自动检测数据使用的碱基质量编码方法，无需用户指定。

图中，红线是碱基质量中位数，黄色框代表四分位数范围（25-75%），上下横线分别代表 10% 和 90% 点，蓝色曲线代表平均质量。

底纹中，红色区域代表质量差，橙色区域代表质量尚可，绿色区域代表质量很好。

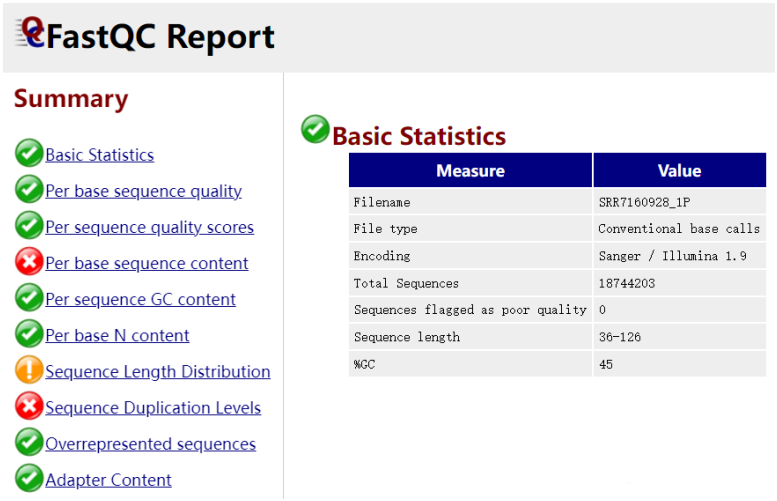


图 9.7: FastQC 报告 (1)-基本统计。

一般而言，测序的质量会随着在 read 上位置推后而下降。本例测序数据质量很好。蓝线一直位于绿色部分。

如果任何碱基质量的下四分位数小于 10，或任何碱基质量的中位数小于 25，将发出警告。如果任何碱基质量的下四分位数小于 5 或 任何碱基质量的中位数小于 20，将不予通过。

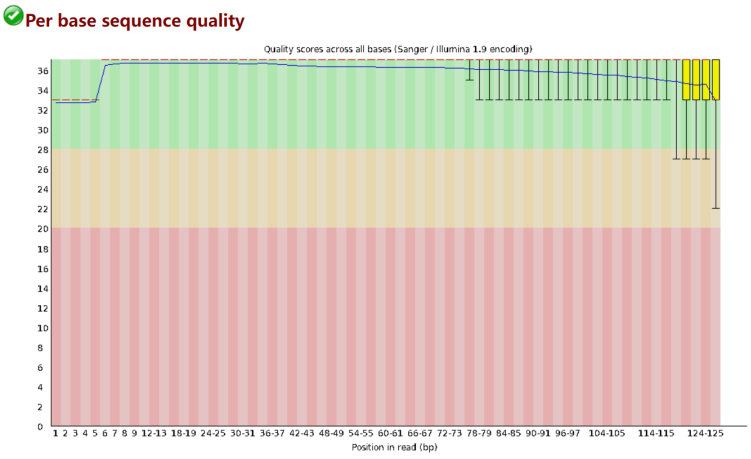


图 9.8: FastQC 报告 (2)-每碱基序列质量。

3. 每条序列质量分数 (Per sequence quality scores)

如图 9.9，该部分显示 FASTQ 文件中序列的质量分布。横轴代表每一条序列中碱基的平均质量分数，纵轴代表有多少碱基获得了该分数。该图还会标出所有序列的平均质量分数的平均值。

如果该值低于 27——这相当于 0.2% 的错误率，则会发出警告；如果低于 20——这相当于 1% 的错误率，则不予通过。

本例中该值高达 35-36，测序数据的整体质量是很好的。

4. 每个碱基序列组成 (Per base sequence content)

如图 9.10，该部分给出 reads 中每个位置的碱基组成，四种颜色分别代表四种碱基的百分比。

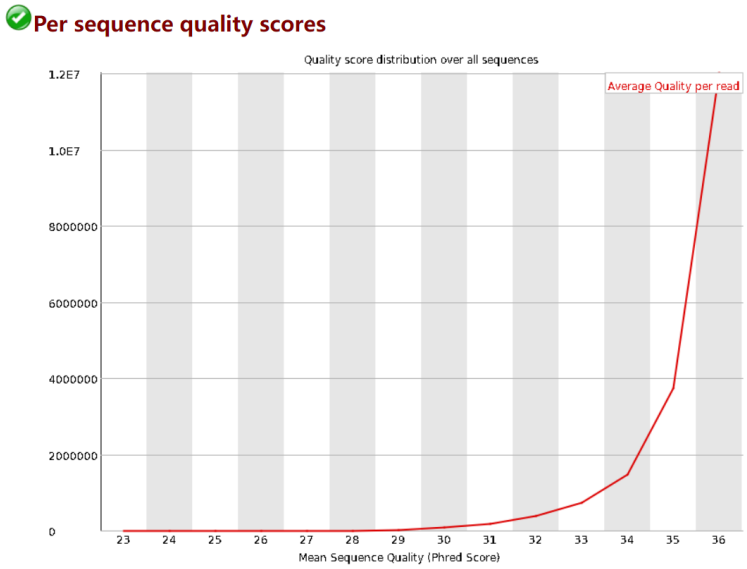


图 9.9: FastQC 报告 (3)-每条序列的质量分数。

如果一个文库内容是随机的，不同碱基的占比应该从头到尾都是相同的。真实情况可能有些波动，但绝不应该产生巨大差距。

本例中，reads 的开头部分各种碱基的比例差别较大。这或许是技术造成，某些类型的文库总是会产生有偏差的序列组成，通常在读取开始时。使用随机六聚体引发的文库（包括几乎所有的 RNA-Seq 文库）和使用转座酶片段化的文库读取开始位置会有内在偏差。

如果 A 和 T 或 G 和 C 之间的差异在任何位置大于 10%，此模块会发出警告；如果大于 20 %，则不予通过。

本例中，数据未通过这个检验，这或许是因为片段化过程引入的偏差（bias），这无伤大雅。

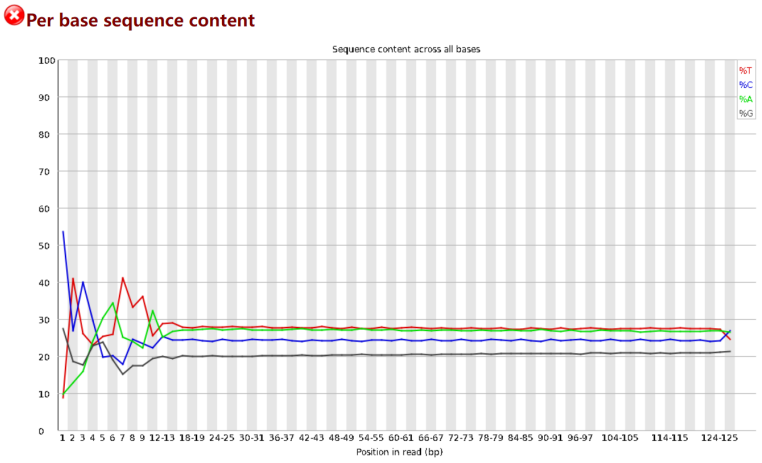


图 9.10: FastQC 报告 (4)-每碱基序列含量。

5. 每条序列 GC 含量 (Per sequence GC content)

如图 9.11，该部分描述所有序列的 GC 含量情况。x 轴代表 GC 百分比，纵轴代表有多少序列的 GC 含量在该百分比。

在正常的随机文库中，用户会期望看到大致正态分布的 GC 含量，其中中心峰对应于基础基因组的整体 GC 含量。由于我们不知道基因组的 GC 含量，因此理论曲线 GC 含量是根据观察到的数据计算的，并用于构建参考分布。

如果实际分布与正态分布的偏差总和超过 15%，则会发出警告；如果超过 30% 的读数，则不予通过。

本例中，实际 GC 含量分布与正态分布符合得较好。

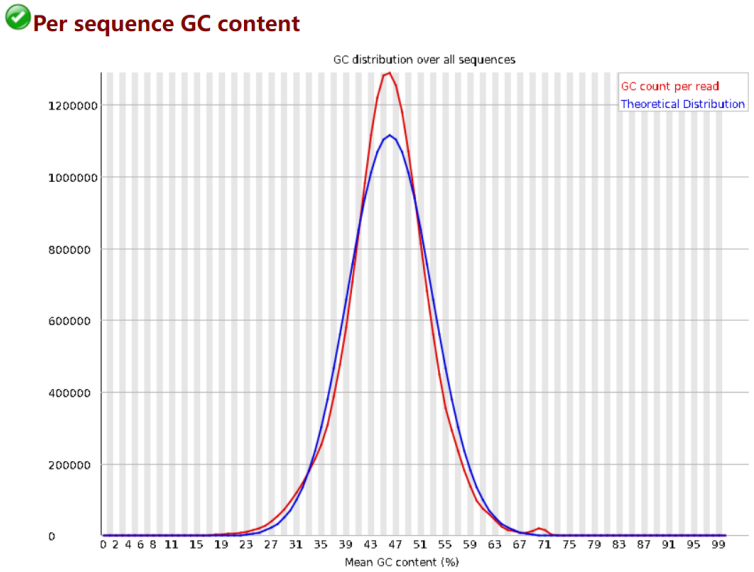


图 9.11: FastQC 报告 (5)-每条序列 GC 含量。

6. 每个碱基位置的 N 含量 (Per base N content)

如图 9.12，该图表示在每个位置 N 碱基的百分比。当测序仪无法就某个位置作出明确判断时，就会将其记为 N。

如果任何位置显示 N 含量 >5%，此模块会发出警告；如果 >20%，将不予通过。大量的 N 代表着该数据整体质量低，在本例中，极少有（或无）N 出现。

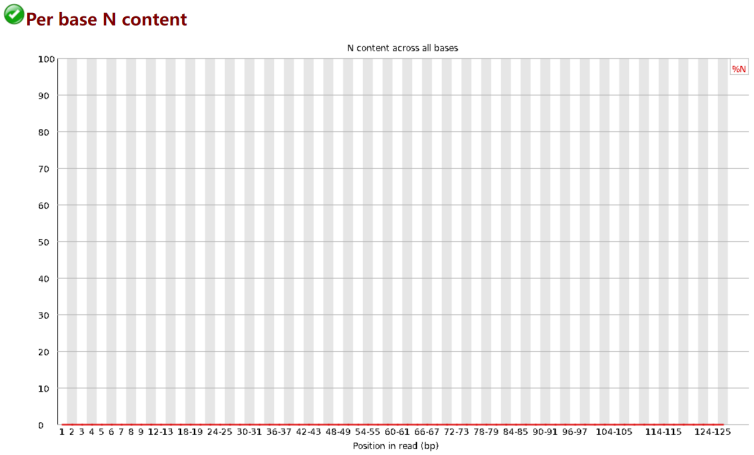


图 9.12: FastQC 报告 (6)-每条序列 N 含量。

7. 序列长度分布 (Sequence Length Distribution)

如图 9.13，显示序列长度的分布。该部分的警告或不通过不必要很担心，有些测序平台所有的 reads 都有相同的长度，有些平台并不，后者会被发给警告，但无需为此担忧。

当任何序列的长度为 0 时，不予通过。

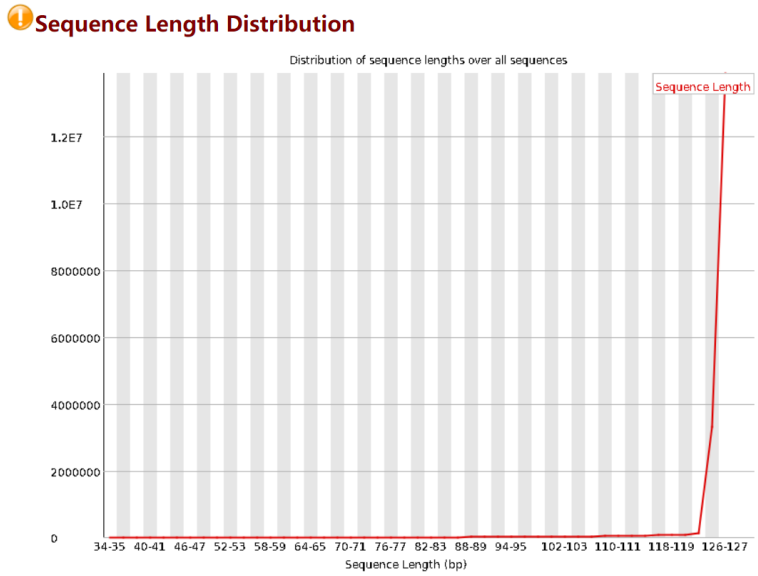


图 9.13: FastQC 报告 (7)-序列长度分布。

8. 序列重复水平 (Sequence Duplication Levels)

如图 9.14，该图显示具有不同重复水平的序列的相对数量。注意这是一个折线图，且越到右边，x 轴划定分组的范围就越大。

图中的蓝色线展示全序列集合中重复水平的分布，红色线展示去重后的序列集的重复水平的分布。重点关注蓝色线。

如果非唯一序列占总数的 20% 以上，此模块将发出警告；如果超过 50%，将不予通过。很遗憾，本例数据没有通过此项质控。读者可以自己寻找原因，推测可能是过度 PCR 或者过度测序。

9. 过度代表的序列 (Overrepresented sequence)

如图 9.15，本例中没有过度代表的序列。

正常的文库应该是多样的，如果发现单个序列在数据中的代表性过高，要么意味着它具有高度的生物学意义，要么表明文库被污染。

如果发现任何序列占总数的 0.1% 以上，此模块将发出警告；如果超过 1%，将不予通过。

10. 接头含量 (Adapter Content)

如图 9.15，该图显示每个位置看到每个适配器序列的文库比例的累积百分比计数。一旦在 reads 中看到一个序列，它就被视为一直存在到 reads 结束，因此百分比只会随着 reads 长度的增加而增加。

如果任何序列存在于超过 5% 的 reads 中，此模块将发出警告；如存在于超过 10 % 的 reads 中，此模块将不予通过。

本例中的数据中很少有接头序列。

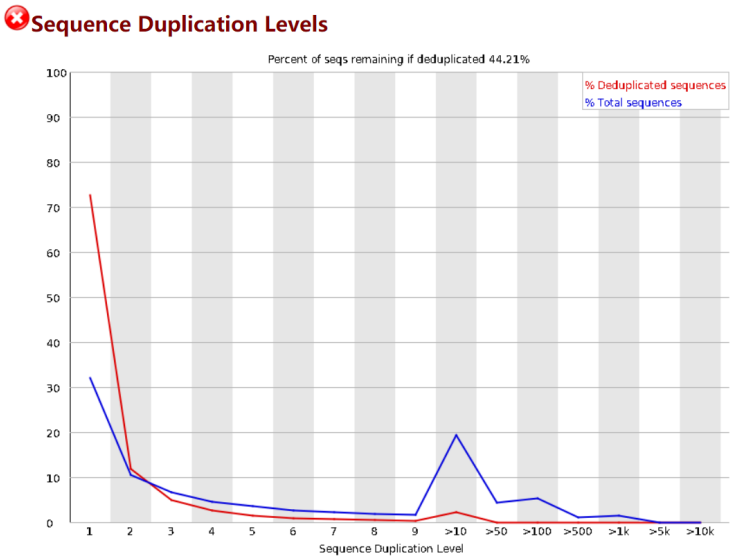


图 9.14: FastQC 报告 (8)-序列重复水平。

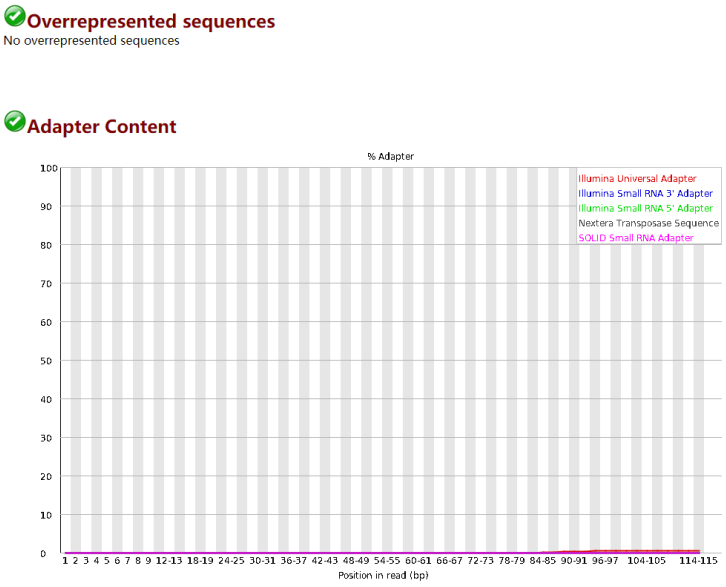


图 9.15: FastQC 报告 (9)(10)-过度代表的序列和接头内容。

至此，FastQC 报告内容已经完毕。读者或许有一种感觉，这份测序数据的质量未免**太好了**。然而，应该注意到，这是经过 `trimmomatic` 处理之后的质控结果! 因此，并不能说明原始测序数据的质量，读者可以对原始测序数据也进行一次质控，并作类似的解读。

9.4.3　比对和转换

如果质控通过，数据就可以进入比对分析；但如果没有通过，可以采用更严格的质量过滤处理一遍，重新质控，也可以重新测序，还可以忽略质控警告，直接进行下一步分析。

9.4.3.1　比对软件的介绍

第三章介绍了序列比对的基本方法，这里结合高通量测序的读段比对，做一点补充。

据参考文献 [14] 统计，从 1988 年到发表前 (2021 年)，已经至少有107种比对工具问世。这里摘取一部分：

表 9.1: 几种常用比对工具

工具名称	出版年份	用途	索引方法	配对比对方法
FASTA	1988	DNA	Hashing	SW and NW
BLAST	1990	DNA	Hashing	Non-DP Heuristic
Gapped BLAST	1997	DNA	Hashing	SW
BLAT	2002	DNA	Hashing	Non-DP heuristic
BWT-SW	2008	DNA	BWT	SW
BWA	2009	DNA	BWT-FM	Semi-Global
Bowtie	2009	DNA	BWT-FM	HD
TopHat	2009	RNA-Seq	BWT-FM	HD
BWA-SW	2010	DNA	BWT-FM	SW
Bowtie2	2012	DNA	BWT-FM	SW & NW
BWA-MEM	2013	DNA	BWT-FM	SW & NW
STAR	2013	RNA-Seq	Suffix array	SW
TopHat2	2013	RNA-Seq	BWT-FM	SW & NW
HISAT	2015	RNA-Seq	BWT-FM	Non-DP Heuristic
BWA-MEM2	2019	DNA	BWT-FM	SW
HISAT2	2019	DNA	BWT-FM	Non-DP Heuristic

如要介绍所有软件的原理和实现，既不必要，也不可能，因此，这里推荐有兴趣的读者自行查找相关文献或软件文档。

本章我们使用的比对工具是 HISAT2，下一章 ChIP-seq 项目笔记中使用的比对工具是 Bowtie2。

HISAT2，发表于 2019 年（参考文献 [15]），是一个快速、灵敏的比对程序，可将 NGS reads（DNA 或 RNA）比对到人类基因组或其他参考基因组。基于图的 BWT 的扩展，HISAT2 首次实现了一个图 FM 索引（GFM）。除了使用一个代表人类基因组的全局 GFM 索引之外，HISAT2 还使用很多加起来可覆盖整个基因组的小的 GFM 索引。这些小的索引（称为局部索引）与多种比对策略一起，实现了测序 reads 的快速精确比对。新的索引策略被称为层次性图 FM 索引（Hierarchical Graph FM index，HGFM）。

Bowtie2，发表于 2012 年（参考文献 [16]），是一个将测序 reads 快速比对到长参考序列的程序，它的一大优点是省内存。Bowtie2 特别擅长于将大约 50-100nt 的 reads 比对到相对较长的基因组上。它使用 FM 索引来索引基因组以维持较少的内存开销，对于人类基因组，其一般所需内存为 3.2G。Bowtie2 支持有空位的、局部的以及端到端比对模式。[4]

9.4.3.2　构建索引

比对之前先要为基因组构建索引，使用 `hisat2-build` 程序。某些基因组在网络上已有构建完成的索引可供下载，但拟南芥基因组不属于此类，因此需要自己构建。不同于 Bowtie2 的索引只有基因组序列信息，HISAT2 的索引包含序列信息和转录组信息。转录组信息需要自己从 GTF 注释文件中提取，HISAT2 提供了两个程序 `hisat2_extract_splice_sites.py`、`hisat2_extract_exons.py` 可以完成这项工作：

```
hisat2_extract_splice_sites.py genome/Arabidopsis_thaliana.TAIR10.54.gtf > genome/splice_sites.txt
hisat2_extract_exons.py genome/Arabidopsis_thaliana.TAIR10.54.gtf > genome/exons.txt
```

构建索引：

```
hisat2-build -p 50 --ss genome/splice_sites.txt --exon genome/exons.txt genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.fa genome/genome > genome/hisat2-build.log 2>&1 &

# 选项
# -p 使用的线程数
# --ss 指定一个剪接位点的列表，注意，必须是hisat2自己的格式
# --exon 指定一个外显子列表，注意，必须是hisat2自己的格式
# 注意，--ss选项和--exon选项必须同时使用，或同时不用
# genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.fa 基因组FASTA文件
# genome/genome 输出文件的位置和前缀
# genome/hisat2-build.log 构建索引日志
```

生成的文件如下：

[4]这些话无用，只是鹦鹉学舌，弄清楚原理需要大量时间，讲清楚原理就更难了。

```
genome
    Arabidopsis_thaliana.TAIR10.54.gff3
    Arabidopsis_thaliana.TAIR10.54.gtf
    Arabidopsis_thaliana.TAIR10.dna.toplevel.fa
    exons.txt
    genome.1.ht2 # 索引文件都以ht2结束，共8个
    genome.2.ht2
    genome.3.ht2
    genome.4.ht2
    genome.5.ht2
    genome.6.ht2
    genome.7.ht2
    genome.8.ht2
    splice_sites.txt
    Truseq-PE.fa
```

索引构建完成以后，即可开始比对。

9.4.3.3　比对

执行下面的脚本:

```
#! /bin/bash
for i in {28..33}
do
    access="SRR71609${i}"
    hisat2 -p 50 --rna-strandness RF --known-splicesite-infile genome/
        splice_sites.txt -x genome/genome --dta -1 workdir/${access}/${
        access}_1P -2 workdir/${access}/${access}_2P > workdir/${access
        }/${access}.sam
done

# 选项
# -p 线程数
# --rna-strandness 指定链特异性信息，默认是没有链特异性的，对于单端测
    序，选择F或R，F代表读段对应于转录本，R代表读段是转录本的反向互补序
    列；对于双端测序，指定FR或RF，使用这个选项让每个比对获得一个XS属性标
    签，+标签表示该读段属于一个在基因组+链上的转录本，-标签表示该读段属于
    一个在基因组-链上的转录本
# --known-splicesite-infile 指定已知剪接位点信息，在本例这一选项是不需
```

```
    要的，因为在构建索引的过程中已经考虑了剪接位点，而这里指定的剪接位点
    与之前指定的剪接位点相同
12  # -x 指定索引文件的位置和前缀
13  # --dta 即--downstream-transcriptome-assembly，如果下游需要进行转录本组
    装，就应该增加这个选项，使用此选项将报告为转录本组装程序而额外剪裁过
    的比对，hisat2需要更长的锚定长度才能从从头发现剪接位点，到这于短锚的
    比对减少，帮助提高转录组组装程序的计算和内存使用率
14  # -1 指定read1文件
15  # -2 指定read2文件
16  # workdir/${access}/${access}.sam 指定输出文件的位置和文件名
```

SAM 文件一般较大，在存储空间有限的情况下应该使用管道符 | 直接传递给 samtools 。HISAT2 将产生一些打印在屏幕上的输出结果，节录如下：

```
1   16389493 reads; of these: # 总reads数目
2     16389493 (100.00%) were paired; of these: # 可配对的reads，因为是用
        trimmomatic输出的_1P和_2P做的比对，当然是100%
3     337606 (2.06%) aligned concordantly 0 times # 没有比对上的reads
4     15292177 (93.30%) aligned concordantly exactly 1 time # 唯一比对的
        reads
5     759710 (4.64%) aligned concordantly >1 times # 不唯一比对的reads
6     ----
7     337606 pairs aligned concordantly 0 times; of these: # 没有合理比对
        的
8       137938 (40.86%) aligned discordantly 1 time # 只有不合理的比对的
9     ----
10    199668 pairs aligned 0 times concordantly or discordantly; of these
        :
11      399336 mates make up the pairs; of these:
12      199796 (50.03%) aligned 0 times
13      179862 (45.04%) aligned exactly 1 time # 作为单端比对，可以唯一
          比对
14      19678 (4.93%) aligned >1 times # 不唯一比对
15  99.39% overall alignment rate # 总体比对率=唯一比对+不唯一比对+不能比对
    中可以单端比对的
```

本次数据的比对结果普遍很好。

9.4.3.4 SAM 文件排序和转换

SAM 格式，全称 Sequence Alignment/Map，是一个用于存储比对到参考序列上的生物序列的文本格式。BAM 格式，全称 Binary Alignment/Map，是无损失压缩的二进制 SAM 文件。SAM，BAM，及配套分析工具 SAMtools 是中国科学家李恒主导开发的 (参考文献 [19])。

SAM 文件由文件头 (header) 和比对区 (alignment section) 组成。

文件头部分以 @ 开头，截取 SRR7160928.sam 的文件头如下：

```
1 @HD    VN:1.0 SO:unsorted
2 @SQ    SN:1   LN:30427671
3 @SQ    SN:2   LN:19698289
4 @SQ    SN:3   LN:23459830
5 @SQ    SN:4   LN:18585056
6 @SQ    SN:5   LN:26975502
7 @SQ    SN:Mt  LN:366924
8 @SQ    SN:Pt  LN:154478
9 @PG    ID:hisat2     PN:hisat2     VN:2.2.1      CL:"/home/lishuai/
    Software/miniconda3/envs/jejune/bin/hisat2-align-s --wrapper basic
    -0 -p 50 --rna-strandness RF --known-splicesite-infile genome/
    splice_sites.gtf -x genome/genome --dta -1 workdir/SRR7160928/
    SRR7160928_1P -2 workdir/SRR7160928/SRR7160928_2P"
```

比对区由很多行构成，每行包括 11 个固定的字段，以及几个可选字段。这 11 个强制性字段包括：

列	字段	类型	描述
1	QNAME	String	read 名字
2	FLAG	Int	比对 FLAG 数字之和
3	RNAME	String	参考序列名，比对到参考序列的染色体号
4	POS	Int	read 比对到参考序列上，第一个碱基所在的位置
5	MAPQ	Int	比对的质量分数，越高说明比对越唯一
6	CIGAR	String	CIGAR 值
7	RNEXT	String	mate 或者下一个序列参考序列名（染色体号）
8	PNEXT	Int	mate 或者下一个序列的位置
9	TLEN	Int	模板长度观测值
10	SEQ	String	片段序列
11	QUAL	String	Phred33 方法表示的碱基质量

截取 SRR7160928.sam 比对区的一个条目如下：

```
1  SRR7160928.3  99     3      16908211      60     100M1D26M     =
       16908271      186
       AGCCATTCCCGATGCCGCGTTTGCCTGTTATCATATCAATACACATAACAAAACACTGAGAAGAT
       GCACATCTCTTCAAAAACACATTTTCTAAAACAAAATCACAACTTAACACAGAAATTGGAT
       BBBBBFFFFFFFFFFFFFFFFFFFFFFFF<FFFFFFFF<FFFFF/FFF<BFFFFFFFFF<FF<
       FFFFFFBBFF//<FFBB/<<BBFFBFF/<FFFFFF<FBFFFFFFFFFFFBFFBFFFFFF<//< AS:
       i:-8 XN:i:0 XM:i:0 XO:i:1 XG:i:1 NM:i:1 MD:Z:100^G26 YS:i:-8 YT:Z:
       CP XS:A:- NH:i:1
2
3  # 固定字段
4  # SRR7160928.3: read名字，在一个SAM文件中可能出现多个同read名的条目，这
       是因为一条read可能被比对到多个位置
5  # 99: 以bit形式表示的FLAGs，这里99=64+32+2+1分别代表"这条序列是read1"、
       "配对序列反向比对"、"配对序列正负链完美比对"、"read是成对序列中的一条
       "
6  # 3: 该read被比对到3号染色体
7  # 16908211: 该read比对上的起始位点是16908211
8  # 60: 该read的比对质量，计算方法为-10*log10(Pr{mapping position is
       wrong})，对最近的整数取整
9  # 100M1D26M: CIGAR字符串，100个匹配、1个Deletion、26个匹配
10 # =: mate比对到的参考序列名，也是3，标记为=
11 # 16908271: mate比对上的起始位点是16908271
12 # 186: 成对reads比对起点到比对终点的距离，或者说是比对的长度或模板序列的
       长度
13 # AGCCATTCCCGATGCCGCGTTTGCCTGTTATCATATCAATACACATAACAAAACACTGAGAAGAT
       GCACATCTCTTCAAAAACACATTTTCTAAAACAAAATCACAACTTAACACAGAAATTGGAT read
       的序列
14 # BBBBBFFFFFFFFFFFFFFFFFFFFFFFF<FFFFFFFF<FFFFF/FFF<BFFFFFFFFF<FF<
       FFFFFFBBFF//<FFBB/<<BBFFBFF/<FFFFFF<FBFFFFFFFFFFFBFFBFFFFFF<//<:
       Phred33编码表示的碱基质量
15
16 # 可选字段
17 # 可选字段遵循统一的格式: TAG:TYPE:VALUE
18 # AS:i:-8 比对分数，带符号整数型，得分为-8
19 # XN:i:0 在参考序列上模糊碱基的个数
20 # XM:i:0 错配的个数
21 # XO:i:1 gap open的个数
22 # XG:i:1 gap 延伸的个数
23 # NM:i:1 到参考序列的编辑距离，包括模糊的碱基，但不包括剪切，带符号整数
```

```
      型，编辑距离为1，因为仅有1个deletion；关于编辑距离可参考第3章
24 # MD:Z:100^G26 错配位置字符串，可能是所有的可打印字符，100个碱基后出现一
      个gap
25 # YS:i:-8 成对序列比对得分
26 # YT:Z:CP 比对的类型，CP 表示合理的比对
27 # XS:A:- 第二好的匹配的得分，这里没有
28 # NH:i:1 在本记录中已经报告的包含此查询序列的比对数目
```

SAM 格式是很复杂的，如果想要尝试挖掘更多的比对信息，SAM 文件是很好的入手之处。但本章的介绍就到此为止，聊为一点引入。SAM 文件以可直接读取的方式保存比对结果，方便人类阅读，但不利于计算机处理，因此需要转换成二进制的 BAM 文件，执行如下脚本转换：

```
1  #! /bin/bash
2  for i in {28..33}
3  do
4      access="SRR71609${i}"
5      samtools sort -l 5 -O BAM -T workdir/${access}/tmp -o workdir/${
          access}/${access}.bam workdir/${access}/${access}.sam -@ 50 -m
          1G > workdir/${access}/${access}.sort.log 2>&1
6  done
7
8  # 选项
9  # -l 指定压缩水平，0为不压缩，1为最低压缩，9为最高压缩，较高的压缩可以缩
      减文件大小但会拖慢处理速度
10 # -O 指定输出文件格式
11 # -T 指定临时文件存放位置和前缀
12 # -o 指定输出文件
13 # -@ 线程数
14 # -m 为每个线程分配的内存大小
```

注意，转换后的 BAM 文件是不能用 `less` 、`cat` 等命令直接读取的。

9.4.4　转录本组装

获得 BAM 文件后，即可进行转录本的组装。不过，为什么要组装 (assemble)?

因为 NGS 获得的小片段不能直接用于后续的生物学分析过程，需要将这些小片段拼接称较长的片段。本次分析使用 StringTie 进行。

StringTie 发表于 2015 年（参考文献 [21]）是一款专为 RNA-seq 设计的转录本组装与定量软件，与后续的分析软件如 DESeq2、EdgeR 等有良好的兼容性。关于它的详细介绍，请阅读参考文献 [7]。

执行下面的脚本：

```
#! /bin/bash
for i in {28..33}
do
    access="SRR71609${i}"
    stringtie workdir/${access}/${access}.bam --rf -p 50 -o workdir/${
        access}/transcript.gtf -G genome/Arabidopsis_thaliana.TAIR10
        .54.gtf -e -b workdir/${access} > workdir/${access}/stringtie.
        log 2>&1
done

# 选项
# -rf 指定链特异性建库方式：fr-firststrand
# -p 组装转录本的线程数
# -o 设置StringTie组装转录本的输出GTF文件的路径和文件名
# -G 使用参考注释基因文件指导组装过程，格式GTF/GFF3
# -e 限制reads比对的处理，仅估计和输出与用-G选项给出的参考转录本匹配的组
    装转录本，使用该选项会跳过处理与参考转录本不匹配的组装转录本，将大大
    提升处理速度
# -b 指定.ctab 文件的输出路径
# workdir/${access}/${access}.bam 指定输入的BAM文件，注意该输入文件必须
    按其基因组位置排序，HISAT2的输出文件则需经过samtools sort生成的bam文
    件才可当做输入文件
```

StringTie 的主要输出文件包括（以 SRR7160928 为例）：

```
SRR7160928
    e2t.ctab # 所有.ctab文件都是用于下游Ballgown软件差异表达分析的输入文
    件
    e_data.ctab
    i2t.ctab
    i_data.ctab
    stringtie.log # 组装日志
    t_data.ctab
    transcript.gtf # 包含已组装转录本的GTF文件
```

9.4.5　生成计数矩阵

在 Linux 终端使用各种软件进行的操作已经接近尾声，接下来我们需要从 StringTie 产生的 GTF 文件中生成计数矩阵，以供基因差异表达分析使用。

StringTie 软件包提供了一个 Python 脚本 `prepDE.py` ，可从 `stringtie` 输出的 GTF 文件中方便地提取 reads 计数映射到特定基因组区域的矩阵，供 `DESeq2` 使用。

执行下面的命令：

```
1 prepDE.py -i workdir -l 126
2
3 # 选项
4 # -i 指定输入文件，需要给一个包含所有样本子目录的目录，在这里我们直接给
    workdir/目录
5 # -l 平均reads长度，可从fastqc结果中获知
```

该命令将产生两个文件：`gene_count_matrix.csv` 和 `transcript_count_matrix.csv` 。

使用 FileZilla 或 Xftp 将它们下载到本地，RNA-seq 分析第一阶段至此结束，接下来的分析可在个人电脑中完成。

9.5　RNA-seq 分析 II

Linux-based Essential Bioinformatics(LEB) 是这门课程的缩写，既然如此，为什么不把这一部分分析也在 Linux 平台上完成呢?

因为在 Linux 上安装和编译 R 包太容易出错，且 R 软件对于 Windows 平台还算友好。

9.5.1　R 与 Bioconductor 简介

R 是科学计算中重要的开源软件，软件包中涉及多种算法和应用，而生物相关的软件都收集在Bioconductor平台下。CRAN(综合性 R 存档网络，Comprehensive R Archive Network)是由 R 基金支持的 R 语言的中心软件仓库。要在计算机上安装 R，可访问 CRAN 下载 R 软件安装包。

Bioconductor 项目旨在开发、支持和传播免费的开源软件，促进对生物数据的严格的和可重复的分析。

为编写 R 程序，我们需要一个好用的集成开发环境 (IDE)，这里推荐RStudio，可安装 RStudio Desktop Open Source Edition。软件的安装和配置可参考互联网教程。顺便一提，安装完毕后记得到 RStudio>Tools>Global Options>Packages>Management 中，将 Primary CRAN repository 改为 China (Beijing) [https] - TUNA Team, Tsinghua University，这将显著提升下载包的速度。

一切准备就绪后，即可进行分析。

9.5.2　用 DESeq2 进行差异表达分析

差异基因表达 (Differential Gene Expression) 分析是 RNA-seq 分析的重要一环，顾名思义，该分析能找出在对照组和实验组中具有差异表达的基因。这些基因表达水平在实验条件下的上调或下调，如果足够显著的话，是很可能有生物学意义的。

DESeq2 是分析差异表达的常用 R 包，用于高维计数数据的归一化、可视化和差异分析。它利用经验贝叶斯技术来估计对数倍数变化和离散程度的先验估计值，并计算这些量的后验估计值。

安装和载入 DESeq2：

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager") # 检查BiocManager是否安装，如无，则安装

BiocManager::install("DESeq2") # 从bioconductor安装DESeq2
library("DESeq2") # 载入DESeq2包
```

在 R 控制台输入 `vignette("DESeq2")` 可以查看 DESeq2 的使用说明 *Analyzing RNA-seq data with DESeq2*，十分值得一读。DESeq2 的基本使用方法如下（引自 DESeq2 使用说明），先留下一个印象，之后会结合本次分析的例子详细解释：

```
dds <- DESeqDataSetFromMatrix(countData = cts,
                              colData = coldata,
                              design= ~ batch + condition)
dds <- DESeq(dds)
resultsNames(dds)
res <- results(dds, name="condition_trt_vs_untrt")
res <- lfcShrink(dds, coef="condition_trt_vs_untrt", type="apeglm")
```

DESeq2 预期输入的计数数据是一个整数构成的矩阵。该矩阵的第 i 行第 j 列记录了在第 j 个样品数据中有多少条 reads 可以被分配到第 i 个基因头上。DESeq2 将在内部校正文库大小，因此输入的数据无需也不应该事先归一化。

我们不妨检查一下 `gene_count_matrix.csv` 文件的内容，可使用 RStudio 的内置工具打开（不要使用 Excel）。截取前几行：

```
gene_id,SRR7160928,SRR7160929,SRR7160930,SRR7160931,SRR7160932,SRR7160933
AT1G01010|NAC001,643,445,523,285,335,293
AT1G03987|AT1G03987,0,0,0,0,0,0
AT1G01030|NGA3,115,122,123,106,100,116
AT1G01020|ARV1,451,421,440,390,454,477
AT1G01060|LHY,673,643,654,928,1044,853
```

```
AT1G01070|UMAMIT28,614,546,529,207,204,179
AT1G01080|AT1G01080,1002,819,843,1878,2073,2119
AT1G03997|AT1G03997,0,0,0,0,0,0
```

第 1 列是基因编号，第 2-7 列分别对应 SRR7160928 到 SRR7160933 共 6 个样品，数据都是整数，应该符合 DESeq2 的要求。

DESeq2 中重要的数据结构是 DESeqDataSet(在代码中常以 dds 表示该数据结构的实例)，构建 dds 需要一个计数矩阵 cts ，一个关于样品信息的表格 coldata 和设计公式。

DESeqDataSet 对象必须具有关联的设计公式（design formula)。设计公式表达了建模中用到的变量，应该以波浪线 (~) 开头，后面是以加号 + 连接的多个变量。设计公式用来估计离散程度和估计模型的 log2 倍数的变化。

首先构建 dds 对象：

```
# 在之前同一环境中执行
# gene_count_matrix.csv 需放在当前工作目录下
countData <- as.matrix(read.csv("gene_count_matrix.csv", row.names="
    gene_id")) # 载入csv文件，指定第一列为列名而非数据，countData内容如
    图9.16
condition <- factor(c(rep("Osmotic",3),rep("Control",3))) # 将字符串向
    量转换成因子，colData数据框内容如下一个代码框中所示
colData <- data.frame(row.names=colnames(countData), condition) # 以
    countData的列名为行名，以因子为第二列内容
dds <- DESeqDataSetFromMatrix(countData = countData, # 计数矩阵
                              colData = colData, # 样品信息，已经实现样品
                                 名(SRR71609..)和处理条件因子("Osmotic","
                                 Control")的对应
                              design = ~ condition) # 仅仅研究实验条件对表
                                 达的影响
```

```
> colData # colData内容
           condition
SRR7160928  Osmotic
SRR7160929  Osmotic
SRR7160930  Osmotic
SRR7160931  Control
SRR7160932  Control
SRR7160933  Control
```

生成 dds 对象后，即可进行差异表达分析。差异表达分析的标准步骤都被封装在了函数 DESeq 中。可以在 R 控制台输入 ?DESeq 查看帮助。执行下面的命令进行分析：

	SRR7160928	SRR7160929	SRR7160930	SRR7160931	SRR7160932	SRR7160933
AT1G01010|NAC001	643	445	523	285	335	293
AT1G03987|AT1G03987	0	0	0	0	0	0
AT1G01030|NGA3	115	122	123	106	100	116
AT1G01020|ARV1	451	421	440	390	454	477
AT1G01060|LHY	673	643	654	928	1044	853
AT1G01070|UMAMIT28	614	546	529	207	204	179
AT1G01080|AT1G01080	1002	819	843	1878	2073	2119
AT1G03997|AT1G03997	0	0	0	0	0	0
AT1G01040|DCL1	1683	1615	1603	1662	1882	1744
AT1G03993|AT1G03993	1	1	1	1	1	0
AT1G01046|ath-MIR838	42	34	43	43	52	51
AT1G01050|PPa1	2293	2459	2362	2292	2494	2371
AT1G01090|PDH-E1 ALPHA	3884	3496	3702	4019	4499	4082
AT1G01110|IQD18	80	71	75	130	108	95
AT1G01120|KCS1	2217	1712	1913	2583	2829	2544
AT1G01100|AT1G01100	4722	3955	4381	5020	5159	4861
AT1G01180|AT1G01180	416	409	424	451	469	458
AT1G01200|RABA3	35	28	41	76	65	59
AT1G01190|CYP78A8	156	172	246	155	204	180
AT1G01130|AT1G01130	48	47	37	33	29	51
AT1G01140|CIPK9	2925	2605	2481	1791	1998	1615

图 9.16: countData 矩阵的内容（截取）。

```
dds <- DESeq(dds)
```

而后用 `results` 对处理条件对比构建结果表：

```
res <- results(dds, contrast = c("condition","Osmotic","Control"))
```

不妨看看 `res` 对象（在 R 控制台输入 `res` 后回车即可），这是一个 32833 行 6 列的表格。其中的信息包括 $\log_2$ 倍数改变（log 2 Fold Change, LFC），p 值和校正后的 p 值。在 `results` 函数没有特殊参数，$\log_2$ 倍数改变和 Wald 测验 p 值是针对设计公式的最后一项而言的；如果该项是一个因子，那么就会在作该因子的最后一个水平与参考水平之间的比较。

本例指定参数 `contrast = c("condition","Osmotic","Control")`，即指定 $\log_2$ 倍数改变 $= \log_2 \frac{Osmotic}{Control}$。Wald test 的 p 值也是针对 Osmotic vs Control 而言的。

```
> res
log2 fold change (MLE): condition Osmotic vs Control
Wald test p-value: condition Osmotic vs Control
DataFrame with 32833 rows and 6 columns
                     baseMean log2FoldChange  lfcSE      stat
                        pvalue       padj
                   <numeric>     <numeric> <numeric> <numeric> <numeric>
                       <numeric>
AT1G01010|NAC001    414.226     0.6991045 0.1378251 5.072404 3.92821e
```

```
      -07 1.19671e-06
8  AT1G03987|AT1G03987  0.000           NA       NA        NA         NA
             NA
9  AT1G01030|NGA3     113.442     0.0418131 0.2049195 0.204047 8.38317e
      -01 8.79056e-01
10 AT1G01020|ARV1     438.232    -0.1280266 0.1179763 -1.085189 2.77838e
      -01 3.57643e-01
11 AT1G01060|LHY      802.978    -0.6359010 0.0917137 -6.933542 4.10432e
      -12 1.76098e-11
12 ...                    ...          ...      ...       ...        ...
            ...
13 ATCG00010          0.000000          NA       NA        NA         NA
             NA
14 ATCG00060          0.501972     0.796135  3.05109 0.2609347   0.794143
            NA
15 ATCG00230          0.337939    -0.118069  3.91989 -0.0301205 0.975971
            NA
16 ATCG00260          0.177490    -1.079850  4.08047 -0.2646386 0.791288
            NA
17 ATCG01030          0.337939    -0.118069  3.91989 -0.0301205 0.975971
            NA
```

我们可以按校正后 p 值从小到大对 res 表格进行排序，而后写入文件中。

```
1  resOrdered <- res[order(res$padj), ] # 对res根据padj进行排序
2  write.table(x = as.data.frame(resOrdered), file = "./
      Osmotic_Control_all", quote = F, sep = "\t", row.names = TRUE, col.
      names = TRUE) # 将排序后的结果写入文件
3
4  # 选项
5  # x = as.data.frame(resOrdered) 要写入文件的对象，这里进行强制类型转换
      成数据框
6  # file = paste0(".","Osmotic","_","Control","_all")
7  # quote = F 字符或因子不会被双引号包围
8  # sep = "\t" 指定字段分隔符，x中每一行都会被这个符号分开，这里选择的是制
      表符
9  # row.names = TRUE 存在行名列
10 # col.names = TRUE 存在列名行
11 # file = "./Osmotic_Control_all" 指定导出的文件路径及其路径
```

此后，对 resOrdered 中的条目作一些筛选，选择其中变化较为显著的。这里的关键问题是：什么样的变化“显著”？本例中采取的标准是要求：校正后 p 值非空（NA），且，校正后 $p < 0.05$，且，$\log_2$ 倍数变化的绝对值 >= 1（这一位置至少发生了 2 倍以上的上调或下调）。代码如下：

```
resSig <- resOrdered[! is.na(resOrdered$padj) & resOrdered$padj < 0.05
    & abs(resOrdered$log2FoldChange)>=1, ] # 筛选显著变化的基因存入
    resSig中
write.table(x = as.data.frame(resSig), file = "./Osmotic_Control",
    quote = F, sep = "\t", row.names = TRUE,col.names = TRUE) # 写入文
    件中
```

此时，当前工作目录下将得到两个文件 Osmotic_Control_all 和 Osmotic_Control 。后者是在渗透处理后表达水平显著变化的基因，我们可以用此表格进行富集分析。

查看 Osmotic_Control 文件的前几行：

```
    baseMean  log2FoldChange lfcSE stat  pvalue padj
AT1G16850|AT1G16850  1715.99374481034  5.49670350388015
    0.139677664645622  39.352773529153 0  0
AT1G17020|SRG1 3114.9192729861 4.06382582468953  0.0788026125440306
    51.5696839672528  0  0
AT1G54570|PES1 6540.59765028929  3.65917578065995
    0.0904564945713295  40.4523279174223  0  0
AT1G54575|AT1G54575  3949.00276631203  4.4835805051715
    0.0889960098312112 50.3795677320254  0  0
AT1G58360|AAP1 7140.71482107642  2.6659669727083 0.0629809894512548
    42.3297092652326  0  0
```

该表格的内容很丰富，但我们只需要其中的基因名，不需要 baseMean 、log2FoldChange 、lfcSE 、stat 、pvalue 或 padj 等信息，因此需要稍做处理，在 Linux 中会比较方便：

```
sed -n '2,$p' ./Osmotic_Control | cut -f 1 > ./DE_genes

# 解释
# sed -n '2,$p' ./Osmotic_Control 打印该文件第2到最后一行的内容
# | 管道符，sed命令处理完的行传输给cut命令
# cut -f 1 提取该行的第一个字段
# > ./DE_genes文件 处理完毕的结果重定向到./DE_genes文件
```

查看 DE_genes 文件前几行：

```
AT1G16850|AT1G16850
```

```
AT1G17020|SRG1
AT1G54570|PES1
AT1G54575|AT1G54575
AT1G58360|AAP1
AT1G64110|DAA1
AT1G75040|PR5
AT2G19900|NADP-ME1
AT2G22990|SNG1
AT2G37870|AT2G37870
AT2G38530|LTP2
```

提取基因名成功。

9.5.3　GO 富集分析

在第七章中我们曾简单介绍 InterPro 数据库中的 GO 条目，可以回顾。

GO 是基因本体（Gene Ontology）的缩写，它的数据库是世界上最大的基因功能信息知识库，是大规模分子生物学和遗传学实验的计算分析的基础。其知识是人类可读的，也是机器可读的。GO 协作团队的目标是建立一个全面的生物系统计算模型，纵向囊括从分子到有机体水平，横向跨越生命之树的多样。

基因本体是一种系统地对物种基因及其产物属性进行注释的方法和过程。它的目标是：

- 维护和发展有限的基因及其产物属性描述的词汇
- 注释基因及其产物，同化和传播注释数据
- 提供方便的工具访问数据
- 实现在实验数据的基础上，使用 GO 进行解析，例如基因富集组分分析

GO 主要包括三个分支：细胞组分、分子功能和生物过程。

不应该被”Ontology” 这个词所迷惑，它既不是哲学意义上的本体论，也不是信息学意义上的本体。在这里，Ontology 指代具有层次的术语表，提供一种规范的注释方式。[5]

本次分析使用 `clusterProfiler` 进行 GO 富集。首先进行准备工作。使用 clusterProfiler 这个包进行 GO 分析时，通常并不需要转换 ID，所有的 ID 类型都可以由 `OrgDb` 文件提供。

Bioconductor 维护了常见物种的 `OrgDb` 文件，可以通过安装包使用。

```
# 安装BiocManager，如果本没有安装的话
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

# 安装包
```

[5]读者可以看一看：*The Ontology of the Gene Ontology*，这是一篇有点意思的文章，可以消除 GO 的神秘感，或者说，对它进行“祛魅”。

```
BiocManager::install("AnnotationHub")
BiocManager::install("org.At.tair.db")
BiocManager::install("clusterProfiler")
BiocManager::install("dplyr") # 如果安装不顺利，可尝试添加force=TRUE参数
BiocManager::install("ggplot2")
# 载入包
library("AnnotationHub")
library("org.At.tair.db")
library("clusterProfiler")
library("dplyr")
library("ggplot2")
```

读入表达水平显著改变的基因名，并用 enrichGO 进行富集分析，随后作图。富集结果如图所示 9.17，可见差异表达基因被富集在呼吸电子传递链、电子传递链、细胞呼吸、氧化有机化合物供能、ATP 合成偶联的电子传递、氧化磷酸化、无氧呼吸、ATP 代谢过程、前体代谢物和能量生成方面。

对此可以进行一些解释：渗透胁迫可能导致细胞损伤，因此可能激活非正常的呼吸链或无氧呼吸，导致线粒体膜破坏进而引发电子传递链断裂，相关基因表达上调；渗透胁迫可能导致细胞内盐离子浓度偏高，导致需要额外的能量将多余的盐离子排出细胞，引发氧化有机化合物功能、氧化磷酸化、ATP 代谢的变化；渗透胁迫还可能促使植物在细胞内积聚代谢中间体以抵抗渗透压力，这导致了前体代谢物的积累。

总而言之，正如我的一位植物专业室友[6]所说的，“解释都可以解释，但真正研究还要仔细甄别”。

```
gene_all_tair <- read.table(file="./DE_genes") # 读入文件
gene_all_tair <- as.list(gene_all_tair)$V1 # 取其第一列作为列表

go_all <- enrichGO ( gene_all_tair, OrgDb=org.At.tair.db , ont = "BP" , pvalueCutoff = 0.05 , keyType="TAIR")

# enrichGO选项
# gene_all_tair 输入的基因列表
# OrgDb=org.At.tair.db 指定转换基因名所用的数据库
# ont = "BP" 在所有三个水平进行富集
# pvalueCutoff = 0.05 指定pvalue截断值
# keyType="TAIR" 标明输入的数据

```

[6]Zhang TY

```
# pdf("./DE_genes_GO.pdf",, width = 10, height = 10,compress=F) # 如要生成pdf文件，取消本行注释
barplot(go_all,showCategory=20) + scale_fill_gradient( low = "#ffc080", high = "#55a0fb", space = "Lab", aesthetics = "fill", guide= guide_colorbar(reverse= TRUE)) # 如要生成pdf文件，取消本行注释
# dev.off()
```

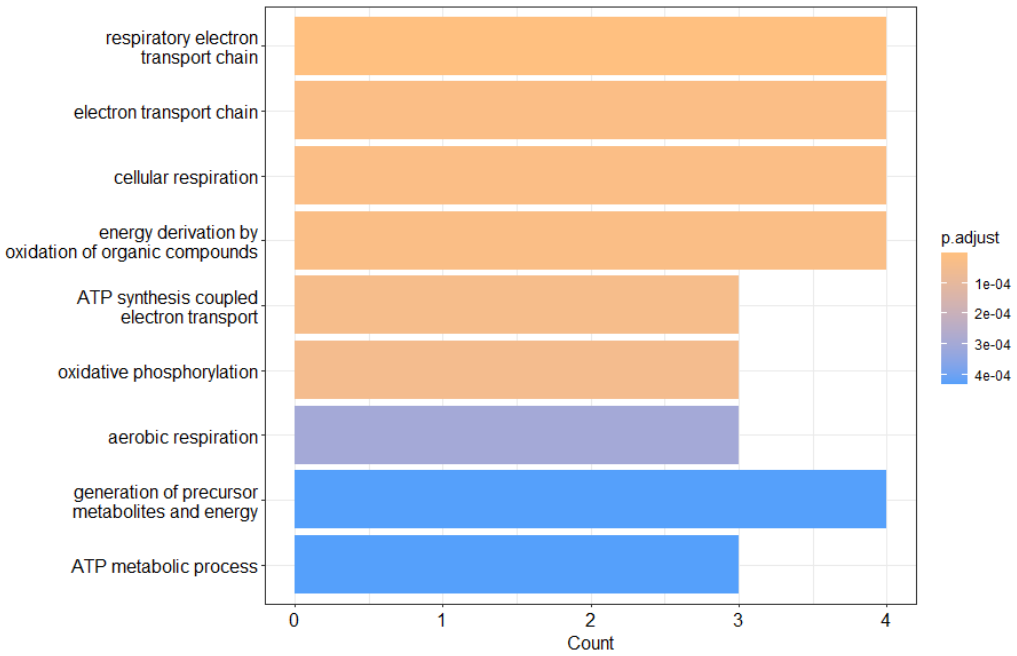


图 9.17: GO 富集分析气泡图-生物过程层次。这张图做了一些美化。

至此，全部 RNA-seq 分析已经结束。这只是一个简单演示，其结果是否正确，我没有什么把握，但本章的代码均已运行成功，或许对于读者有所帮助。

如果读者想要绘制热图、火山图，或者进行 KEGG 通路富集，可以阅读参考文献 [18] 的系列文章。参考文献 [18] 是本班同学的期末项目成果，相当详尽，是本章的内容所难以达到的，我自请避贤路。

参考文献

[1] NGS 数据过滤之 Trimmomatic 详细说明

[2] Trimmomatic Manual: V0.32

[3] Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics (Oxford, England)*, 30(15), 2114–2120.

[4] SAMtools Documentation

[5] FastQC Documentation

[6] NGS 文库制备-Illumina 技术页

[7] StringTie Manual

[8] Sultan, M., Schulz, M. H., Richard, H., Magen, A., Klingenhoff, A., Scherf, M., ... & Yaspo, M. L. (2008). A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome. *Science*, 321(5891), 956-960.

[9] Kukurba, K. R., & Montgomery, S. B. (2015). RNA sequencing and analysis. *Cold Spring Harbor Protocols*, 2015(11), pdb-top084970.

[10] mRNA 文库构建

[11] Illumina TruSeq RNA Library Prep Kit v2 Data Sheet

[12] TruSeq Sample Preparation Best Practices and Troubleshooting Guide

[13] How short inserts affect sequencing performance

[14] Alser, M., Rotman, J., Deshpande, D., Taraszka, K., Shi, H., Baykal, P. I., Yang, H. T., Xue, V., Knyazev, S., Singer, B. D., Balliu, B., Koslicki, D., Skums, P., Zelikovsky, A., Alkan, C., Mutlu, O., & Mangul, S. (2021). Technology dictates algorithms: recent developments in read alignment. *Genome biology*, 22(1), 249.

[15] HISAT2 主页

[16] Kim, D., Paggi, J. M., Park, C., Bennett, C., & Salzberg, S. L. (2019). Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature biotechnology*, 37(8), 907–915.

[17] Langmead, B., & Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4), 357–359.

[18] 李扬. 等. RNA-seq 数据分析原理及流程详解.

[19] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., & 1000 Genome Project Data Processing Subgroup (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, 25(16), 2078–2079.

[20] SAM 格式文档

[21] Pertea, M., Pertea, G. M., Antonescu, C. M., Chang, T. C., Mendell, J. T., & Salzberg, S. L. (2015). StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nature biotechnology*, 33(3), 290–295.

[22] Bioconductor Project

[23] DESeq2 Manual

[24] 罗静初. Linux 生物信息技术基础. 课堂练习.

第十章　ChIP-seq 项目笔记

转录因子 CTCF 的 ChIP-Seq 数据初步分析是 2022 年春季 LEB 课程第一小组和第四小组的期末项目，这一章以此为基础，适当补充一些内容，介绍 ChIP-seq 分析的基本流程。本章涉及的工具较多，不能一一详细介绍其原理和使用方法，只为关键工具作简单介绍，其余的只解释代码中用到的功能和参数。

感谢饶希晨[1]师兄提供大部分代码（有增改）、项目总结文件和研讨会幻灯片，并感谢两组的所有同学。

本章代码已被验证可以成功运行。

10.1　原理、方法和背景

10.1.1　ChIP-seq 原理

染色质免疫共沉淀 (Chromatin Immunoprecipitation) 能够找出在细胞基因组中哪些特定的 DNA 序列能够与目的蛋白结合。它的基本原理是，甲醛交联后并打断后，与目的蛋白结合的 DNA 会与目的蛋白一起被目的蛋白的抗体免疫沉淀。

而如果此时再解除交联，将被沉淀的 DNA 释放、加上测序接头，即可进行下一代测序，这就是**染色质免疫共沉淀测序技术** (ChIP-seq)[2]。

10.1.2　ChIP-seq 实验方案

下面简单叙述 ChIP-seq 的湿实验方案[3]：

- 交联
  - 以终浓度 1% 甲醛在 37°C 孵育 10 分钟以固定细胞
  - 在细胞培养板上加入甘氨酸至终浓度为 125 mM 以终止固定，室温下孵育 5 分钟
  - 用冷 PBS 洗涤两次后，在完全蛋白酶抑制剂和 PMSF 存在的条件下，刮取细胞并在 4°C 下以 300 g 离心 5 分钟

[1]饶希晨 (1997 -)，湖北武汉人，北京大学生命科学学院博士研究生，2022 春季 LEB 课程第一组组长，2022 秋季 ABC 课程助教，ORCID ID Profile of Xichen Rao

[2]参考文献 [1]

[3]参考文献 [2][3]

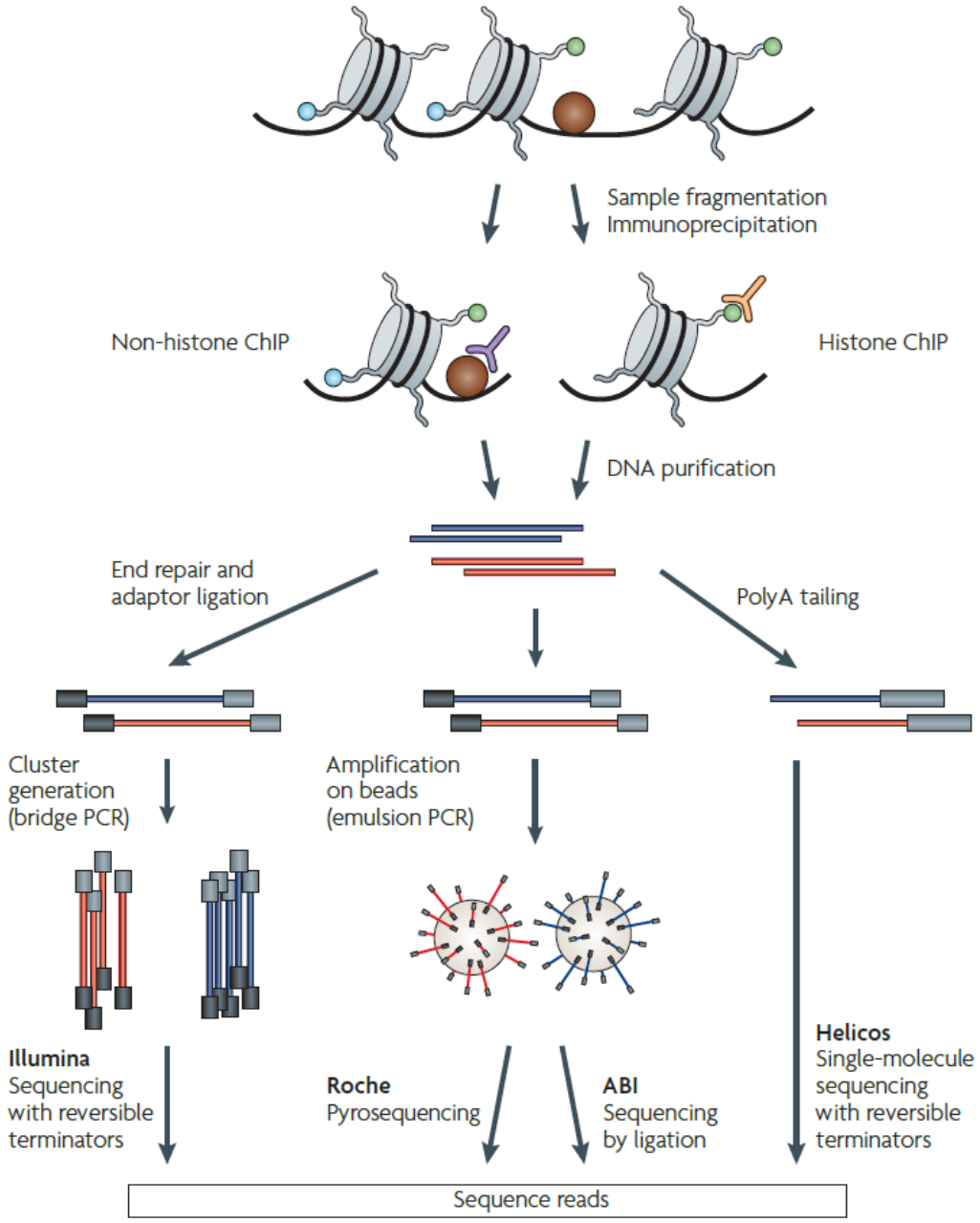


图 10.1: ChIP-seq 的基本原理。用甲醛处理细胞，这使得 DNA 结合蛋白在体内与 DNA 交联，随后染色质被超声波打断成长度为 200-600bp 的小片段。接下来使用目标蛋白质特异的抗体免疫沉淀 DNA-蛋白质复合物并逆转交联反应以释放的 DNA。最后按照各个测序平台的要求构建文库，测序，鉴定 ChIP-seq 富集的 DNA 序列。

- 细胞裂解和染色质超声破碎
  - 加入 2.5 ml 0.5% NP-40 裂解缓冲液以裂解细胞沉淀以进行细胞核分离，孵育 10 分钟。
  - 4°C 下以 3000 g 离心 5 min
  - 在 1 ml SDS 1% 裂解缓冲液中悬浮细胞沉淀
  - 使用 Bioruptor (Diagenode, UCD-200) 以高强度对裂解缓冲液中的细胞进行三个循环，每次 10 分钟，超声处理 30 秒，暂停 30 秒
  - 4°C 下以 17000g 离心 10 分钟来使溶液澄清
- 免疫沉淀
  - 将 30 μg 染色质和 4 μg 抗体在 IP 缓冲液（0.1% SDS、1% 161 TX-100、2 mM EDTA、20 mM Tris-HCl pH8、150 mM NaCl）中 4°C 旋转孵育过夜
  - 添加 25 μl 预封闭 (1 mg/ml BSA) Dynabeads 蛋白 A 和 Dynabeads 蛋白 G，孵育 4 小时
  - 用 IP 缓冲液清洗磁珠 10 分钟
  - 用增加盐浓度 (500 mM NaCl) 的 IP 缓冲液清洗磁珠 10 分钟
  - 用 LiCl 缓冲液（0.25 M LiCl、1% NP40、1% NaDoc、20 165 mM Tris-HCl pH8 和 1 mM EDTA）清洗磁珠 10 分钟
  - 使用标记 DNA 酶进行 ChIPmentation(打断、加接头)
  - 通过在 100 ul 洗脱缓冲液（1% SDS 和 100 mM $NaHCO_3$）中 50 °C 孵育 30 分钟，从磁珠中洗脱 DNA。
- 去交联和纯化
  - 将洗脱的样品与 200 mM NaCl 和 100 μg/ml 蛋白酶在 65 °C 下孵育过夜，以解除交联
  - 使用 PCR 纯化柱纯化 DNA
- 文库构建
  - 使用测序平台要求的文库构建试剂盒生成完整文库
  - 使用磁珠纯化和选择大小在 100-500 bp 范围内的文库
- NGS 测序

注意，ChIP-seq 实验一般需要一个或多个对照组，构建不使用抗体进行富集的文库。只有将实验组与对照组数据进行比较，才能发现目标蛋白富集的 DNA 位点。

10.1.3 CTCF 简介

CCCTC-结合因子 (CCCTC-binding factor, CTCF, Uniprot 主页) 是一种高度保守的锌指蛋白，可以作为转录激活因子、转录阻遏因子，此外，还可以作为绝缘因子阻断增强子和启动子之间的通讯。

CTCF 可以在与染色质结构域边界结合的同时招募其他转录因子。真核基因组的三维结

构决定了它的功能，CTCF 是帮助建立这种组织的核心结构蛋白之一。

不同物种中 CTCF 结合位点的作图表明，基因组被 CTCF 结合位点覆盖，结合其上的 CTCF 有各种各样的功能。

CTCF 通过其 11 个锌指 (ZF) 定向结合众多 CTCF 结合位点 (CBS) 内的四个模块的特定序列元件，确定适当的全基因组染色质环相互作用。

CTCF 与 CBS 复合物的晶体结构已被解出。

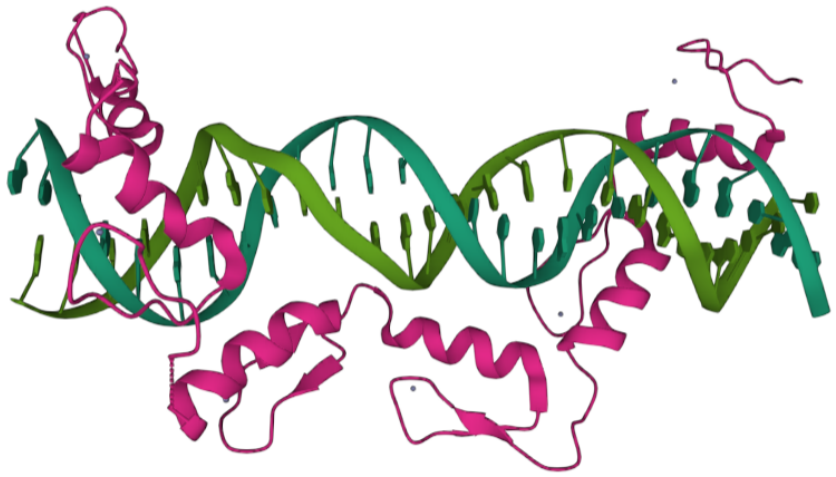


图 10.2: CTCF ZFs6-11-gb7CSE 晶体结构。PDB 编号：5YEL

10.2　概述和准备工作

10.2.1　分析流程

本项目首先对测序数据进行一般的分析，包括质量控制、去接头、比对、排序、转换、过滤、去重、索引；随后进行 ChIP-seq 特有的分析，包括 ChIP-seq 质量控制、富集峰寻找；最后根据研究目的进行一些初步分析，包括与寻找转录起始位点的共定位、峰基序（Peak motif）。

10.2.2　配置环境

本项目需要的环境和软件如下：

执行下面的命令：

```
conda create -n chip-seq
conda activate chip-seq
conda install -c bioconda fastqc
conda install -c bioconda cutadapt
conda install -c bioconda bowtie2
```

```
conda install -c bioconda samtools
conda install -c bioconda picard
conda install -c bioconda macs2
conda install -c bioconda deeptools
conda install -c bioconda idr
conda install -c bioconda homer
conda install -c bioconda meme
```

10.2.3 下载测序数据和参考文件

10.2.3.1 DNA 原件百科全书

本项目使用的公共数据从 **DNA 元件百科全书** (Encyclopedia of DNA Elements, ENCODE) 下载。

ENCODE Consortium 产生高质量的数据，并以整合的方式分析数据。ENCODE 百科全书将最突出的分析结果组织成注释，并提供搜索和可视化它们的工具。

ENCODE 的注释内容包括包括：

- 综合水平注释：整合了多种类型的实验数据和基本水平注释
  - 候选顺式调控元件 (Candidate cis-egulatory elements, cCREs) 登记表
  - cCRE 的搜索和可视化工具：SCREEN (Search Candidate cis-Regulatory Elements by ENCODE)
  - 染色质状态注释
  - 变异注释
- 基本水平注释：直接从实验数据中以统一处理流程产生的注释
  - 开放染色质 (DNase-seq, ATAC-seq)
  - 组蛋白标记富集 (ChIP-seq)
  - 转录因子结合 (TF ChIP-seq)
  - 基因表达 (RNA-seq)
  - 转录起始位点 (TSS) 活动性分析 (RAMPAGE)
  - RNA 结合蛋白的结合情况 (eCLIP-seq)
  - DNA 甲基化 (RRBS, WGBS)
  - 三维染色质相互作用 (ChIA-PET)
  - 拓扑相关结构域 (Hi-C)

10.2.3.2　数据描述

项目数据的 ENCODE 登录号是ENCSR617IFZ，相关文章是：TopicNet: a framework for measuring transcriptional regulatory network change[4]。

这是一份人 HEK293 细胞中的 eGFP-CTCF 的 ChIP-seq 实验数据，该细胞系使用靶向人 CTCF 的位点特异性重组插入了 eGFP。

如图 10.1 所示，实验数据总共有 4 个文件，分别是两个生物学重复的 Read1 和 Read2，测序类型为双端 101nt，文件格式为 `fastq` 。

生物学重复 1 测序数据注册号为ENCFF631JSV, ENCFF715KYL；

生物学重复 2 测序数据注册号为ENCFF002OWA, ENCFF833BQT。

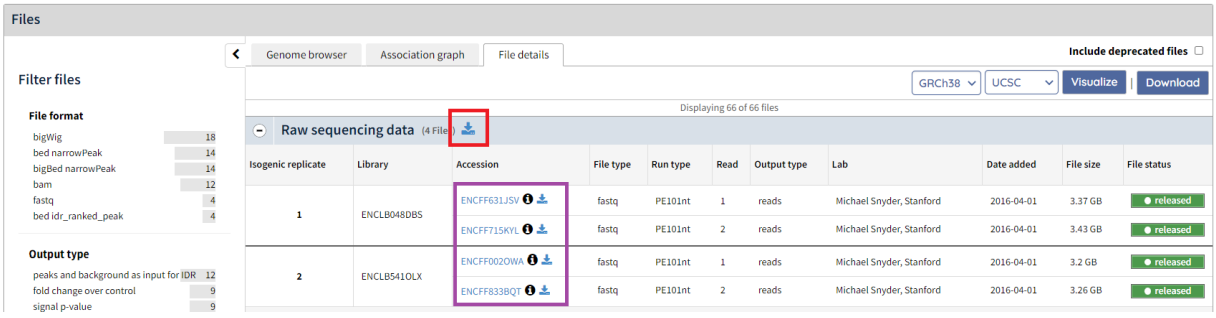


图 10.3: ENCODE实验页面

此外，在 ChIP-seq 数据中寻找富集峰需要一份未富集文库测序数据作为参考（Input），本项目选用的数据注册号为：ENCFF873ZZU，ENCFF611URR。

10.2.3.3　下载方法

方法一：点击图 10.1 页面中的紫色方框中的蓝色下载图案，即可下载数据文件到个人电脑，随后可用 Filezilla 或 Xftp 上传服务器。

方法二：点击图 10.1 页面中的红色方框中的蓝色下载图案，可下载此实验的 `files.txt` 文件，内容如下：

```
1 "https://www.encodeproject.org/metadata/?type=Experiment&%40id=%2
    Fexperiments%2FENCSR617IFZ%2F&files.output_type=reads&files.
    output_category=raw+data"
2 https://www.encodeproject.org/files/ENCFF631JSV/@@download/ENCFF631JSV
    .fastq.gz
3 https://www.encodeproject.org/files/ENCFF002OWA/@@download/ENCFF002OWA
    .fastq.gz
4 https://www.encodeproject.org/files/ENCFF833BQT/@@download/ENCFF833BQT
    .fastq.gz
```

[4]参考文献 [5]

```
5 https://www.encodeproject.org/files/ENCFF715KYL/@@download/ENCFF715KYL
      .fastq.gz
```

files.txt 文件包含了一个指向实验元数据的 URL 以及下载文件的链接，将此文件复制到服务器，并运行命令：

```
1 xargs -n 1 curl -O -L < files.txt
```

即可下载。其中，curl 是一个利用 URL 规则在命令行下工作的文件传输工具，它支持文件的上传和下载。xargs 单独使用可以读取文件并将每一行数据作为参数传给指定的命令执行。

方法三：使用 wget 进行下载，这里以对照数据进行演示：

```
1 wget https://www.encodeproject.org/files/ENCFF873ZZU/@@download/
     ENCFF873ZZU.fastq.gz
2 wget https://www.encodeproject.org/files/ENCFF611URR/@@download/
     ENCFF611URR.fastq.gz
```

下载完成后，把文件命名为更容易理解的名字：

```
1 mv ENCFF631JSV.fastq.gz ctcf_rep1_read1.fastq.gz
2 mv ENCFF715KYL.fastq.gz ctcf_rep1_read2.fastq.gz
3 mv ENCFF002OWA.fastq.gz ctcf_rep2_read1.fastq.gz
4 mv ENCFF833BQT.fastq.gz ctcf_rep2_read2.fastq.gz
5 mv ENCFF873ZZU.fastq.gz input_read1.fastq.gz
6 mv ENCFF611URR.fastq.gz input_read2.fastq.gz
```

此外，还需下载参考基因组文件：

```
1 wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.
     gz
2 gunzip hg38.fa.gz
```

10.3　一般的分析

ChIP-seq 测序数据需要完成基本的 NGS 分析流程。注意，实验数据有两个生物学重复，此外，为了鉴定富集峰，另外需要一份未富集的测序数据作为参考。此三份数据应该采取完全相同的一般分析处理流程。由于样品数并不多，且分析流程自动化不是课程目标，因此采用较为呆板的方式将命令运行 3 次，避免使用循环、通道或组织脚本。

10.3.1 测序数据质量控制

首先创建工作目录：`mkdir workdir`，并在其中为 2 个实验组和对照组创建子目录。目录结构如下：

```
.
    data
        ctcf_rep1_read1.fastq.gz
        ctcf_rep1_read2.fastq.gz
        ctcf_rep2_read1.fastq.gz
        ctcf_rep2_read2.fastq.gz
        input_read1.fastq.gz
        input_read2.fastq.gz
    genome
        hg38.fa
    workdir
        input
        rep1
        rep2
```

如 RNA-seq 分析一样，仍然使用 FastQC 作质量控制。FastQC 可以多线程运行，因此可先检测服务器的线程数。

查看服务器的线程数：

```
grep 'processor' /proc/cpuinfo | sort -u | wc -l
```

质控命令如下：

```
fastqc -t 20 -o workdir/rep1 data/ctcf_rep1_read1.fastq.gz data/
    ctcf_rep1_read2.fastq.gz > workdir/rep1/fastqc.log 2>&1 &
fastqc -t 20 -o workdir/rep2 data/ctcf_rep2_read1.fastq.gz data/
    ctcf_rep2_read2.fastq.gz > workdir/rep2/fastqc.log 2>&1 &
fastqc -t 20 -o workdir/input data/input_read1.fastq.gz data/
    input_read2.fastq.gz > workdir/input/fastqc.log 2>&1 &

# 选项
# -t 指定使用的线程数
# -o 指定输出文件夹
```

质控完毕，除 `input_read2.fastq.gz` 的 Per tile sequence quality 一项未通过以外，其余数据和样本均顺利通过质控（有警告，但没有失败）。

Per tile sequence quality 只会出现在保留其原始序列标识符的 Illumina 文库的分析结果

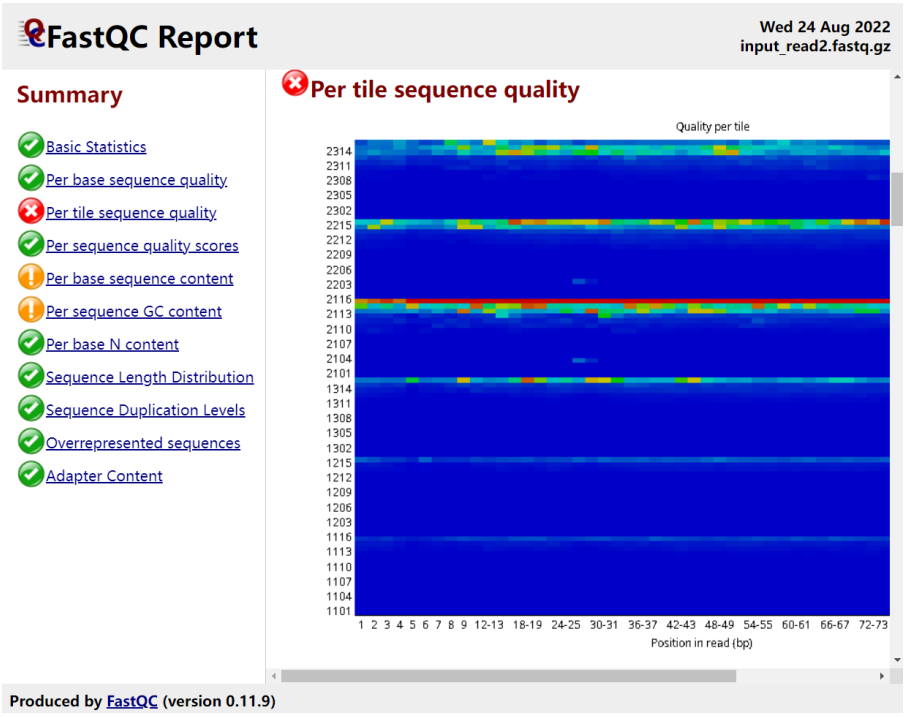


图 10.4:　`input_read2.fastq.gz` 质控报告 Per tile sequence quality 部分。

中。该图展示来自每个 tile(flowcell 中的最小单元) 所有碱基的质量分数，以查看是否存在仅与 flowcell 的某一特定部分相关的质量损失。该图显示了与每个 tile 的碱基质量相对于所有 tile 平均碱基质量的偏差。冷色 tile 是数据质量达到或高于平均水平的位置，而较暖的颜色表明该 tile 产生数据的质量比其他 tile 更差。

造成图中这种错误的原因一般是由于测序的序列有偏差 (biased)。这些高亮的区域代表 flowcell 边缘，因为这些区域拍照系统识别 read 的信号的能力下降，导致无论在 reads 的什么位置，该 tile 处产生的碱基质量都较差。一般而言，这些数据还不是太糟糕，常常还是能用的[5] [6]。因此，我们继续分析。

10.3.2　去接头

使用 `cutadapt` 去接头，Cutadapt 从高通量测序 reads 中查找并删除接头序列、引物、poly-A 尾和其他类型的不需要的序列。

去接头命令如下：

```
1  cutadapt -j 20 --time 1 -e 0.1 -O 3 --quality-cutoff 25 -m 55 -a
       AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A
       AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o workdir/rep1/
       ctcf_rep1_read1_cut.fastq.gz -p workdir/rep1/ctcf_rep1_read2_cut.
       fastq.gz data/ctcf_rep1_read1.fastq.gz data/ctcf_rep1_read2.fastq.
       gz > workdir/rep1/cutadapt.log 2>&1 &
```

[5] Per tile sequence quality

[6] FastQC 文档

```
cutadapt -j 20 --time 1 -e 0.1 -O 3 --quality-cutoff 25 -m 55 -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o workdir/rep2/ctcf_rep2_read1_cut.fastq.gz -p workdir/rep2/ctcf_rep2_read2_cut.fastq.gz data/ctcf_rep2_read1.fastq.gz data/ctcf_rep2_read2.fastq.gz > workdir/rep2/cutadapt.log 2>&1 &
cutadapt -j 20 --time 1 -e 0.1 -O 3 --quality-cutoff 25 -m 55 -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o workdir/input/input_read1_cut.fastq.gz -p workdir/input/input_read2_cut.fastq.gz data/input_read1.fastq.gz data/input_read2.fastq.gz > workdir/input/cutadapt.log 2>&1 &

# 选项，以第一条命令为例
# -j 20 设置线程数
# --times 1 只去处一次接头，因为接头只出现一次
# -e 0.1 去除的序列与adaptor相比，missmatch率低于该值，则认为是adaptor，一般设置为0.1
# -O 3 当与adaptor overlap的碱基数大于等于该值时，才进行去除
# --quality-cutoff 25 小于等于该qulity的碱基去除
# -m 55 trim之后低于该值的一对reads丢弃，一般要大于40
# -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA read1 3'的adaptor
# -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT read2 3'的adaptor
# -o workdir/rep1/ctcf_rep1_read1_cut.fastq.gz read1的输出文件
# -p workdir/rep1/ctcf_rep1_read2_cut.fastq.gz read2的输出文件
# data/ctcf_rep1_read1.fastq.gz Read1输入文件
# data/ctcf_rep1_read2.fastq.gz Read2输入文件
# workdir/rep1/cutadapt.log 日志文件，标准输出和标准错误重定向到此
```

10.3.3 比对

采用 Bowtie2 软件进行比对。

10.3.3.1 构建索引

bowtie2-build 构建索引：

```
bowtie2-build --threads 40 -f genome/hg38.fa genome/hg38_human_index > genome/build_index.log 2>&1 &
# 选项
```

```
# --threads 40 线程数，按前面查到的服务器线程数选择
# -f 基因组为fasta格式，指定基因组文件位置
# genome/hg38_human_index 指定索引文件输出位置和前缀
# genome/build_index.log 日志文件，标准输出和标准错误重定向到此
```

10.3.3.2　比对

ChIP-seq 的比对比较简单，因为文库从基因组构建，无需考虑内含子或外显子，同时也无需考虑正负链信息，可以使用默认参数。

```
bowtie2 -x genome/hg38_human_index -1 workdir/rep1/ctcf_rep1_read1_cut
    .fastq.gz -2 workdir/rep1/ctcf_rep1_read2_cut.fastq.gz -p 40 -S
    workdir/rep1/ctcf_rep1_mapped.sam > workdir/rep1/bowtie2_mapping.
    log 2>&1 &
bowtie2 -x genome/hg38_human_index -1 workdir/rep2/ctcf_rep2_read1_cut
    .fastq.gz -2 workdir/rep2/ctcf_rep2_read2_cut.fastq.gz -p 40 -S
    workdir/rep2/ctcf_rep2_mapped.sam > workdir/rep2/bowtie2_mapping.
    log 2>&1 &
bowtie2 -x genome/hg38_human_index -1 workdir/input/input_read1_cut.
    fastq.gz -2 workdir/input/input_read2_cut.fastq.gz -p 40 -S workdir
    /input/input_mapped.sam > workdir/input/bowtie2_mapping.log 2>&1 &
# 选项
# -x 索引文件
# -1 read1的fastq文件
# -2 read2的fastq文件
# -p 线程数
# -S 输出的SAM文件
# workdir/rep1/bowtie2_mapping.log 日志文件
```

我们不妨查看一番比对日志，以 workdir/rep1/bowtie2_mapping.log 为例：

```
38244291 reads; of these:
38244291 (100.00%) were paired; of these:
    798008 (2.09%) aligned concordantly 0 times
    31773460 (83.08%) aligned concordantly exactly 1 time
    5672823 (14.83%) aligned concordantly >1 times
    ----
    798008 pairs aligned concordantly 0 times; of these:
    210694 (26.40%) aligned discordantly 1 time
    ----
```

```
10      587314 pairs aligned 0 times concordantly or discordantly; of these
            :
11      1174628 mates make up the pairs; of these:
12          633374 (53.92%) aligned 0 times
13          244314 (20.80%) aligned exactly 1 time
14          296940 (25.28%) aligned >1 times
15  99.17% overall alignment rate
```

可见整体比对率很高。

10.3.4　比对结果的排序、过滤、去重、索引

10.3.4.1　SAM 文件排序生成 BAM 文件

使用 samtools sort 程序按照染色体坐标排序并转换为 BAM 文件（如果需要构建 BAM 文件的 index 文件，必须按照染色体坐标排序）：

```
1  samtools sort -O BAM -o workdir/rep1/ctcf_rep1_mapped.bam -@ 50 -m 1G
       workdir/rep1/ctcf_rep1_mapped.sam > workdir/rep1/samtools_sort.log
       2>&1 &
2  samtools sort -O BAM -o workdir/rep2/ctcf_rep2_mapped.bam -@ 50 -m 1G
       workdir/rep2/ctcf_rep2_mapped.sam > workdir/rep2/samtools_sort.log
       2>&1 &
3  samtools sort -O BAM -o workdir/input/input_mapped.bam -@ 50 -m 1G
       workdir/input/input_mapped.sam > workdir/input/samtools_sort.log
       2>&1 &
4  # 选项
5  # -O 指定输出文件格式
6  # -o 指定输出文件
7  # -@ 线程数
8  # -m 为每个线程分配的内存大小
9  # workdir/rep2/ctcf_rep2_mapped.sam 未排序的输入文件
10 # workdir/rep1/samtools_sort.log 日志文件
```

10.3.4.2　BAM 文件过滤

使用 samtools view 程序进行 BAM 文件过滤：

```
1  samtools view -bS -q 30 -@ 50 -m 1G -o workdir/rep1/
       ctcf_rep1_mapped_filtered.bam workdir/rep1/ctcf_rep1_mapped.bam >
       workdir/rep1/samtools_view.log 2>&1 &
```

```
2  samtools view -bS -q 30 -@ 50 -m 1G -o workdir/rep2/
       ctcf_rep2_mapped_filtered.bam workdir/rep2/ctcf_rep2_mapped.bam >
       workdir/rep2/samtools_view.log 2>&1 &
3  samtools view -bS -q 30 -@ 50 -m 1G -o workdir/input/
       input_mapped_filtered.bam workdir/input/input_mapped.bam > workdir/
       input/samtools_view.log 2>&1 &
4  # 选项
5  # -b 输出BAM文件
6  # -S 忽略与以前的samtools版本的兼容性。从前这个选项被用来指定输入文件为
       SAM格式，现在程序会自动检测输入文件的格式
7  # -q 跳过MAPQ小于该值的比对（MAPQ值位于SAM文件的第5列，描述比对的质量）
8  # -@ 线程数
9  # -m 为每个线程分配的内存大小
10 # -o 指定输出文件
11 # workdir/rep1/ctcf_rep1_mapped.bam 未过滤的输入文件
12 # workdir/rep1/samtools_view.log 日志文件
```

10.3.4.3　去重

NGS 数据中有两类重复：因扩增文库引起的 PCR 重复和因测序仪将一个 cluster 区分成多个 cluster 的光学重复。

去重的原则是，如果一个重复是来源于样本的，那么就不该去；如果是来源于技术的，就应该去重。普通的 RNA-seq 起始量大，重复很可能是来源于样本；但 ChIP-seq 的起始量小，重复一般是技术原因。

因为聚合酶的偏好性，在 PCR 循环次数较多时，一些序列将比另一些序列更受聚合酶的喜爱，因而具有更高的扩增倍数。这种情况会严重影响 ChIP-seq 的定量，因此 ChIP-seq 数据一般需要去重[7]。

本项目使用 Picard 软件去重：

通过 conda 安装的最新 picard 已经无需使用 java -jar PATH/picard.jar ... 方式运行，且选项参数的指定格式也发生了改变。目前可用的命令格式是：

```
1  picard PicardToolName \
2         OPTION1=value1 \
3         OPTION2=value2
```

执行下面的命令：

```
1  picard MarkDuplicates REMOVE_DUPLICATES=true I="workdir/rep1/
       ctcf_rep1_mapped_filtered.bam" O="workdir/rep1/
```

[7] NGS 去重知多少

```
    ctcf_rep1_mapped_filtered_dedup.bam" M="workdir/rep1/
    marked_dup_metrics.txt" > workdir/rep1/picard.log 2>&1 &
picard MarkDuplicates REMOVE_DUPLICATES=true I="workdir/rep2/
    ctcf_rep2_mapped_filtered.bam" O="workdir/rep2/
    ctcf_rep2_mapped_filtered_dedup.bam" M="workdir/rep2/
    marked_dup_metrics.txt" > workdir/rep2/picard.log 2>&1 &
picard MarkDuplicates REMOVE_DUPLICATES=true I="workdir/input/
    input_mapped_filtered.bam" O="workdir/input/
    input_mapped_filtered_dedup.bam" M="workdir/input/
    marked_dup_metrics.txt" > workdir/input/picard.log 2>&1 &

# MarkDuplicates 指定使用的Picard工具
# 选项
# REMOVE_DUPLICATES=true 要求去除重复序列，如不指定只会将重复序列标记出
    来
# I="workdir/rep1/ctcf_rep1_mapped_filtered.bam" 指定输入文件
# O="workdir/rep1/ctcf_rep1_mapped_filtered_dedup.bam" 指定输出文件
# M="workdir/input/marked_dup_metrics.txt" 重复序列的一些统计信息
# workdir/input/picard.log 日志文件
```

10.3.4.4　BAM 文件索引

使用 samtools index 程序索引 bam 文件

执行下面的命令：

```
samtools index -@ 40 workdir/rep1/ctcf_rep1_mapped_filtered_dedup.bam
    > workdir/rep1/samtools_index.log 2>&1 &
samtools index -@ 40 workdir/rep2/ctcf_rep2_mapped_filtered_dedup.bam
    > workdir/rep2/samtools_index.log 2>&1 &
samtools index -@ 40 workdir/input/input_mapped_filtered_dedup.bam >
    workdir/input/samtools_index.log 2>&1 &
# 选项和参数含义与samtools view类似，不再赘述
```

至此，工作目录结构应如下所示，对每个文件作了一些解释，请对照检查。

```
.
    data # 存放原始数据
        ctcf_rep1_read1.fastq.gz # 重复1
        ctcf_rep1_read2.fastq.gz
        ctcf_rep2_read1.fastq.gz # 重复2
```

```
6         ctcf_rep2_read2.fastq.gz
7         input_read1.fastq.gz # 对照组
8         input_read2.fastq.gz
9     genome # 存放参考文件
10        build_index.log # 参考基因组构建索引日志
11        hg38.fa # 参考基因组文件
12        hg38_human_index.1.bt2 # 参考基因组索引
13        hg38_human_index.2.bt2
14        hg38_human_index.3.bt2
15        hg38_human_index.4.bt2
16        hg38_human_index.rev.1.bt2
17        hg38_human_index.rev.2.bt2
18    workdir # 工作目录
19        input # 对照组
20            bowtie2_mapping.log # 比对日志
21            cutadapt.log # 去接头日志
22            fastqc.log # 质控日志
23            input_mapped.bam # 比对BAM文件
24            input_mapped_filtered.bam # 过滤后的BAM文件
25            input_mapped_filtered_dedup.bam # 去重后的BAM文件
26            input_mapped_filtered_dedup.bam.bai # 去重后的BAM文件的索引
          文件
27            input_mapped.sam # 比对SAM文件
28            input_read1_cut.fastq.gz # read1切除的序列
29            input_read1_fastqc.html # read1质控报告
30            input_read1_fastqc.zip # read1质控报告原始数据
31            input_read2_cut.fastq.gz # read2切除的序列
32            input_read2_fastqc.html # read2质控报告
33            input_read2_fastqc.zip # read2质控报告原始数据
34            marked_dup_metrics.txt # 去重去掉的序列的统计
35            picard.log # 去重日志
36            samtools_index.log # 索引日志
37            samtools_sort.log # 排序日志
38            samtools_view.log # 过滤日志
39        rep1 # 重复1，文件含义与input相同，不再赘述
40            bowtie2_mapping.log
41            ctcf_rep1_mapped.bam
42            ctcf_rep1_mapped_filtered.bam
43            ctcf_rep1_mapped_filtered_dedup.bam
```

```
44            ctcf_rep1_mapped_filtered_dedup.bam.bai
45            ctcf_rep1_mapped.sam
46            ctcf_rep1_read1_cut.fastq.gz
47            ctcf_rep1_read1_fastqc.html
48            ctcf_rep1_read1_fastqc.zip
49            ctcf_rep1_read2_cut.fastq.gz
50            ctcf_rep1_read2_fastqc.html
51            ctcf_rep1_read2_fastqc.zip
52            cutadapt.log
53            fastqc.log
54            marked_dup_metrics.txt
55            picard.log
56            samtools_index.log
57            samtools_sort.log
58            samtools_view.log
59        rep2 # 重复2，文件含义与input相同，不再赘述
60            bowtie2_mapping.log
61            ctcf_rep2_mapped.bam
62            ctcf_rep2_mapped_filtered.bam
63            ctcf_rep2_mapped_filtered_dedup.bam
64            ctcf_rep2_mapped_filtered_dedup.bam.bai
65            ctcf_rep2_mapped.sam
66            ctcf_rep2_read1_cut.fastq.gz
67            ctcf_rep2_read1_fastqc.html
68            ctcf_rep2_read1_fastqc.zip
69            ctcf_rep2_read2_cut.fastq.gz
70            ctcf_rep2_read2_fastqc.html
71            ctcf_rep2_read2_fastqc.zip
72            cutadapt.log
73            fastqc.log
74            marked_dup_metrics.txt
75            picard.log
76            samtools_index.log
77            samtools_sort.log
78            samtools_view.log
```

很明显，上述文件目录中有很多中间文件并不会再下游继续使用，因此可以删除；当然，更好的方法是使用管道，不保留中间文件，以节约存储空间。

到这里，一般分析已经完成。

10.4 ChIP-seq 分析

10.4.1 ChIP-seq 质量控制

ChIP-seq 实验中的一个重要问题是 “ChIP 有效果吗?”，即，抗体处理是否导致了足够程度的富集以使 ChIP 信号可以从背景信号中分离出来。这一步骤应该在 call peak 之前进行，它并不依赖于 call peak 的结果。

`deeptools` 程序包中的 `plotFingerprint` 程序可以对有索引的 BAM 文件进行采样，并对其聚集性 read 分布情况进行绘图。这一方法称为累积富集（Cumulative enrichment），也称 BAM 指纹（BAM fingerprint）。

在一个指定长度的窗口 (bin) 中，所有交叠的 reads 都被计数，所有这些计数按大小排序，其累积和被绘图。

```
1 plotFingerprint -b workdir/rep1/ctcf_rep1_mapped_filtered_dedup.bam
     workdir/rep2/ctcf_rep2_mapped_filtered_dedup.bam workdir/input/
     input_mapped_filtered_dedup.bam --labels rep1 rep2 input --
     plotFileFormat svg --plotTitle CTCF_Enrichment -o workdir/
     CTCF_Enrichment.svg -p 50
2
3 # 参数
4 # -b 输入BAM文件列表，可以是一个或多个BAM文件
5 # --labels 输出图像中使用的标签
6 # --plotFileFormat 输出文件的格式
7 # --plotTitle 输出文件的标题
8 # -o 指定输出文件
9 # -p 使用的线程数
```

输出文件图 10.5 所示。理论上，对于没有富集的对照组，其累积富集曲线应接近于对角线，这与我们的结果符合。对于转录因子的强富集，理想的情形应该是曲线先缓慢上升，到了 rank 较高的区域后急剧上升，这种情形代表大量的 reads 分布在较少的特定窗口中，也即，在蛋白因子的结合位点存在显著的富集。

但由于我不知道的原因，本实验 ChIP-seq 与对照组的累积富集曲线相当接近，有兴趣的读者可以自己查明原因，或者使用其他的方法验证本项目 ChIP-seq 数据的质量。我们暂且进入下一步。

10.4.2 寻找富集峰

CTCF 是一种转录因子，针对 CTCF 的 ChIP-seq 能够将与其有相互作用的 DNA 片段富集起来。Peak calling 是一种 ChIP-seq 特有的分析流程，用于鉴定基因组中 reads 的富集部位。

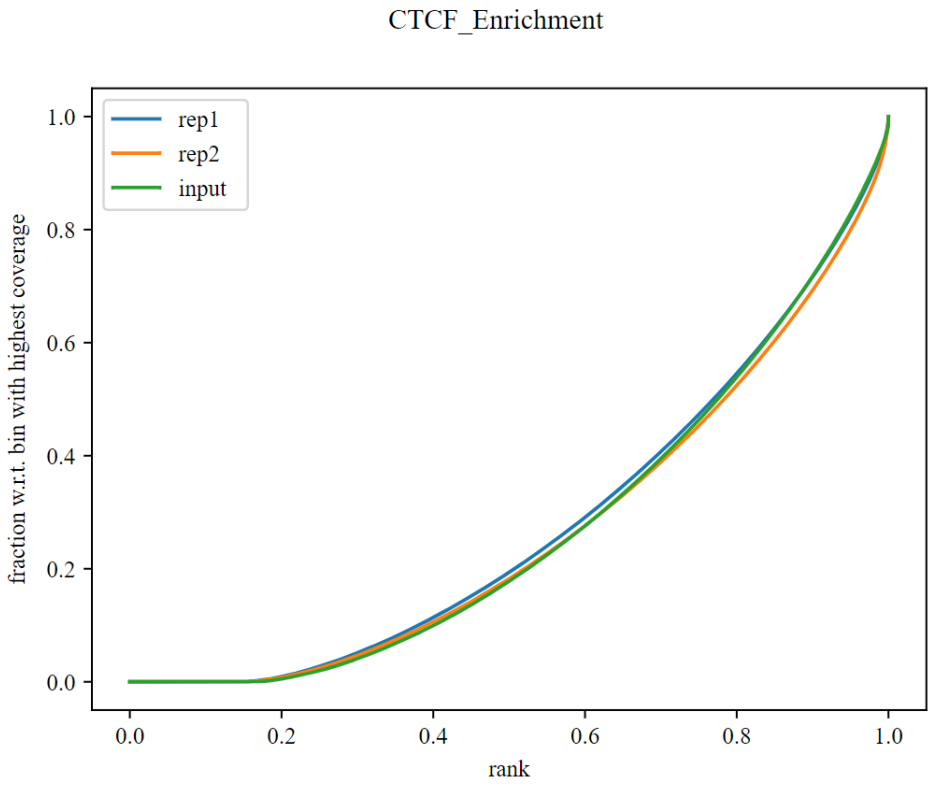


图 10.5: CTCF-ChIP-seq 的两个重复与 input 的累积富集绘图。

在 ChIP-seq 实验中，比对文件 reads 的分布是链不对称的：reads 以转录因子结合位点为中心，在正义链和反义链各自的 5’ 端形成富集。

根据这种特征，可使用统计方法评估这些 reads 的分布，并与背景（输入或模拟 IP 样本）进行比较，以确定富集位点是否可能是真正的结合位点。

10.4.2.1　MACS 原理

基于模型的 ChIP-seq 分析（Model-based Analysis of ChIP-seq, MACS）是进行 peak calling 的常用工具，程序名：`macs2` 。

可以使用单独的 ChIP-seq 样本 或 ChIP-seq 样本和对照样本来进行富集峰寻找，其基本流程如图所示。

注意到，无论是处理组还是对照组，都需要进行去重操作。去重步骤在我们的项目中已经提前完成了，但也可以通过 `macs2` 指定参数完成去重[8]。

横移大小建模及 reads 横移

MACS 利用 reads 分布的双峰模式对横移（shift）大小进行经验建模，以更好地定位精确的结合位点。

为了找到成对的峰来构建模型，MACS 首先扫描整个数据集以搜索高度显着的富集区域。给定一个破碎片段大小 (bandwidth) 和一个高置信度富集倍数 (mfold)，MACS 在基因组上滑动 `bandwidth` 宽的两个窗口，寻找 reads 数相对于随机分布富集程度高 `mfold` 倍的区域。这

[8]详见参考文献 [10]

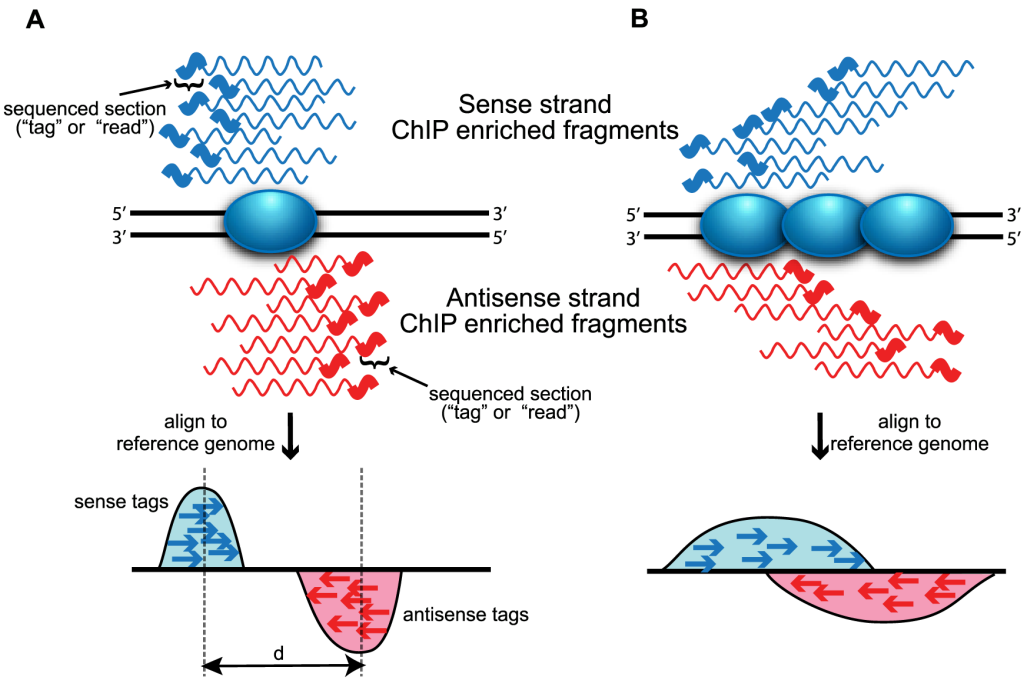


图 10.6: ChIP-seq 数据 reads 在转录因子结合位点周边呈现双峰富集模式。引自参考文献 [11]，A，窄峰，常见于特定转录因子的 ChIP-seq；B，宽峰，常见于组蛋白修饰的 ChIP-seq。

些被找出的区域称为高质量峰。

MACS 随机抽取 1000 个高质量的峰，按所在的是正、负链分类，并找出这两个峰的中点，及其峰顶到中点的距离。

两个峰的模式之间的距离定义为 d，表示估计的转录因子结合片段长度。MACS 将所有 reads 向 3' 末端移动 $\frac{d}{2}$，就可得到最可能的蛋白质-DNA 相互作用位点。

缩放文库

如果对照组样本和处理样本之间测序深度不同，MACS 将按比例缩小较大的样本，以使两个样本的 reads 数一样多。

有效基因组大小

MACS2 需要知道有效基因组大小以计算接下来要用到的 λ_{BG} 值，对于常见生物，这个值已经预先计算好。

有效基因组大小即可比对的基因组大小。可比对性（mappability）与基因组特定位置的 k-mer 的唯一性相关。重复片段区域具有较低的唯一性，也就有较低的可比对性。有效基因组大小用来校正在难比区域的真实信号丢失。

检测峰

在完成 reads 横移的基础上，我们只需要检测单峰即可鉴定真实的转录因子结合位点。此时以大小为 $2d$ 的窗口沿着基因组滑动来寻找可能的峰。

基因组上的 reads 分布可用泊松分布（Poisson distribution）来建模。

泊松分布模型是一个单参数的模型，λ 是唯一的参数，是该分布的期望和方差。

泊松分布对随机事件的发生次数进行建模，其概率密度函数 (随机事件恰好发生 x 次的概率) 是：

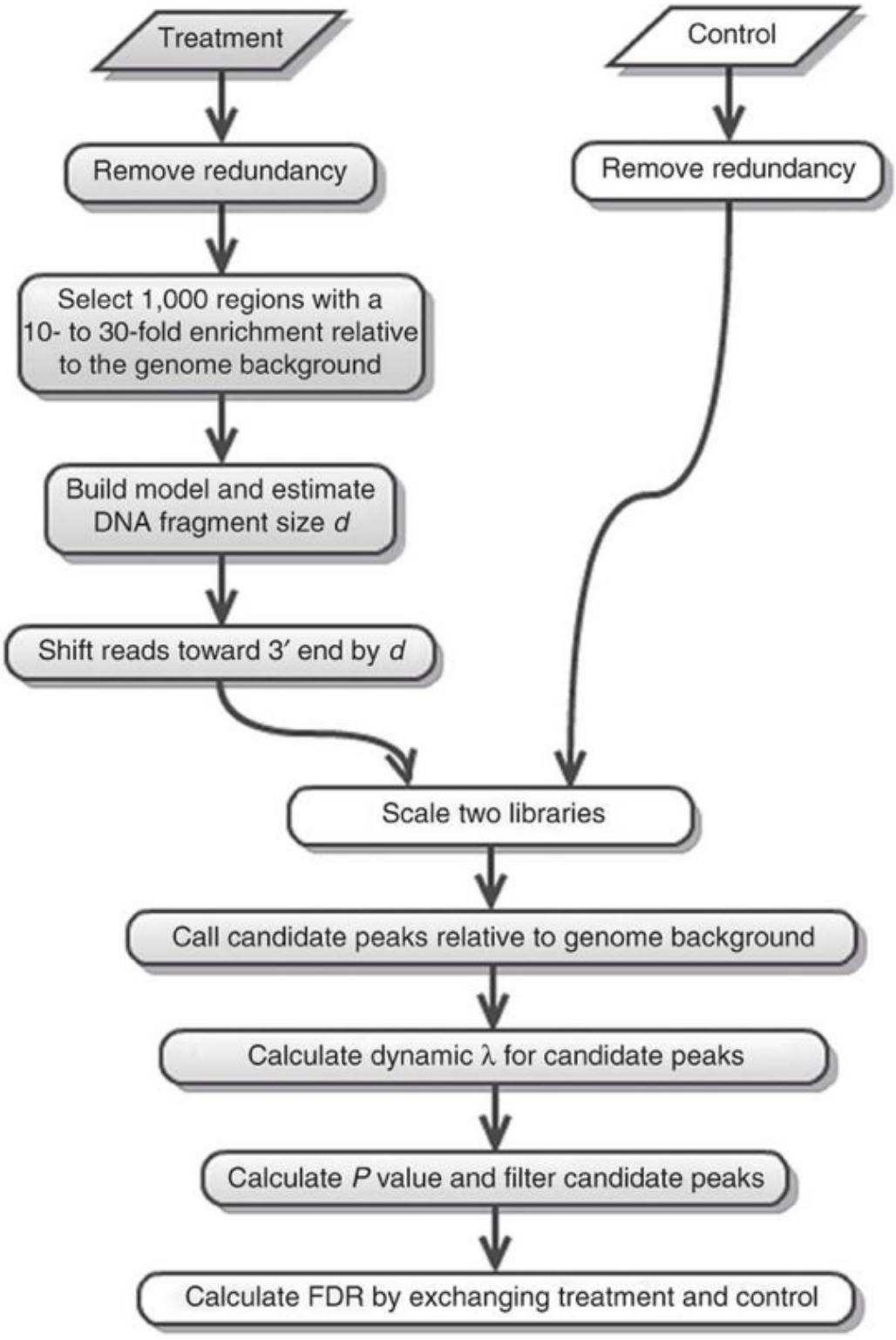


图 10.7: MACS 的工作流程。

$$f(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}; \text{其中：} \ x = 0, 1, 2, \ldots, \infty$$

其累积分布函数（随机事件发生小于等于 x 次的概率）是：

$$F(x|\lambda) = e^{-\lambda} \sum_{i=0}^{floor(x)} \frac{\lambda^i}{i!}$$

在这个应用场景下，前述“随机事件”是指“在该窗口中的 reads 数”，此时 λ 即为该窗口中 reads 数的期望。

MACS 并不对基因组的任何部分使用同样的 λ 值，而是从对照组中估计多个值后取最大值：

$$\lambda_{local} = \max\{\lambda_{BG}, \lambda_{1k}, \lambda_{5k}, \lambda_{10k}\}$$

其中，λ_{BG} 是对全基因组的计算，$\lambda_{1k}, \lambda_{5k}, \lambda_{10k}$ 则是来自对照组中以候选区域为中心，1kb，5kb，10kb 范围内区域进行计算。

估计假阳性率

如果一个窗口中的 reads 数 x 大大超过了 λ_{local}，以至于使得 $p = 1 - F(x|\lambda_{local}) < 1 \times 10^5$（单尾检验），即可以有足够的把握，断定如此多的 reads 出现在一个窗口中，并不是随机事件，而是 ChIP-seq 富集所导致的。

由于每个峰是一个独立的假设检验，在一个样品中我们或许要同时处理数千个峰，因此必须对 p 值进行校正，MACS 使用 Benjamini-Hochberg 校正，该法将各 p 值从小到大排序，生成顺序数。排第 k 的校正 p 值为 $\frac{p \times n}{k}$。

10.4.2.2　MACS 实践

`macs2` 程序中有等多种子命令，可通过 `macs2 <command> -h` 来查看其使用方法，这里只介绍用到的 `callpeak` 命令。

```
1  macs2 callpeak -t workdir/rep1/ctcf_rep1_mapped_filtered_dedup.bam -c
       workdir/input/input_mapped_filtered_dedup.bam -f BAMPE -g hs -n
       ctcf_rep1 -B -q 0.01 --outdir workdir/rep1/call_peak > workdir/rep1
       /peak_calling.log 2>&1 &
2
3  macs2 callpeak -t workdir/rep2/ctcf_rep2_mapped_filtered_dedup.bam -c
       workdir/input/input_mapped_filtered_dedup.bam -f BAMPE -g hs -n
       ctcf_rep2 -B -q 0.01 --outdir workdir/rep2/call_peak > workdir/rep2
       /peak_calling.log 2>&1 &
4
5  # 选项
6  # -t ChIP-seq比对数据文件 (BAM)
```

```
7  # -c 对照组（无富集）比对数据文件
8  # -f 指定数据文件的格式，这里指定为双端BAM文件，该选项的默认值是AUTO，即
       macs2自己识别输入文件的格式
9  # -g 有效基因组大小，hs表示人的有效基因组大小，约为2.7E9
10 # -n name string,会成为输出文件的prefix
11 # --outdir 输出文件路径
12 # -B/--BDG 保存堆积对齐片段到bedGraph文件里
13 # p-value cutoff
14 # workdir/rep2/peak_calling.log 日志文件
```

运行完毕后，以 rep1 为例，检查 workdir/rep1/call_peak 目录下生成的文件：

```
1 call_peak/
2     ctcf_rep1_control_lambda.bdg
3     ctcf_rep1_peaks.narrowPeak
4     ctcf_rep1_peaks.xls
5     ctcf_rep1_summits.bed
6     ctcf_rep1_treat_pileup.bdg
```

其中，ctcf_rep1_peaks.xls 保存了每个 peak 的信息，截取一段：

```
1  chr    start  end    length abs_summit    pileup -log10(pvalue)
       fold_enrichment -log10(qvalue) name
2  chr1   778618 779016 399    778803 46     19.3857 4.96015 15.7596
       ctcf_rep1_peak_1
3  chr1   912845 913104 260    913017 24     9.92487 4.21776 6.81581
       ctcf_rep1_peak_2
4  chr1   920977 921268 292    921181 37     9.68755 3.14401 6.60121
       ctcf_rep1_peak_3
5  chr1   924770 925246 477    925062 40     17.9764 5.16729 14.4152
       ctcf_rep1_peak_4
6
7  # 按列说明
8  # chr: 染色体名
9  # start: peak开始的位置
10 # end:peak结束的位置
11 # length; peak区域的长度
12 # abs_summit=absolute peak summit position: peak峰出现的位置
13 # pileup: peak峰的堆积高度
14 # -log10(pvalue): 该峰p-value的负对数值
```

```
# fold_enrichment: local lambda下，相对于泊松分布（随机事件），peak峰高度的变化倍数，即富集倍数
# -log10(qvalue): 该峰q-value的负对数值
# name: 该峰命名
```

ctcf_rep1_peaks.narrowPeak 是 BED6+4 文件，同样存储了 Peak 信息，截取一部分:

```
chr1   778617 779016 ctcf_rep1_peak_1     157     .       4.96015     19.3857 15.7596 185
chr1   912844 913104 ctcf_rep1_peak_2     68      .       4.21776     9.92487 6.81581 172
chr1   920976 921268 ctcf_rep1_peak_3     66      .       3.14401     9.68755 6.60121 204

# 每列分别是(含义与xls文件中相同):
# 染色体号
# peak起始位点
# peak结束位点
# peak命名
# int(-10*log10(qvalue))
# 正负链（如果不区分，则记为".")
# fold change
# -log10pvalue
# -log10qvalue
# peak的峰相对于peak的位置
```

ctcf_rep1_summits.bed ： 记录了所有 peak 峰的位置和可信度，这个文件可用于寻找 motif。

ctcf_rep1_treat_pileup.bdg：用于存放区间的坐标轴信息和相关评分 (score) 的文件，存储稀疏，不连续的数据。本文件记录了处理组的堆积信号；可以转化为 bigwig 文件（二进制，有索引，更高效）。

ctcf_rep1_control_lambda.bdg：记录了由对照组中估计得到的局部偏好；可以转化为 bigwig 文件。

10.4.3 合并两个生物学重复 (可选项)

本项目 ChIP-seq 实验组有 2 个生物学重复，在进行接下来的分析时，需要将两个生物学重复合并。

IDR 软件评估生物学重复样本间的相关性，并根据阈值筛选出最终的一组 peak。IDR (rreproducible Discovery Rata)，不可重复性率，是一个专门用于从多个生物学重复样本的 peak 结果中提取高一致性 peak 区间的软件，

使用 IDR 需要先对 MACS2 的结果文件 narrowPeak 根据-log10(p-value) 进行排序:

```
mkdir workdir/merge/
sort -k8,8nr workdir/rep1/call_peak/ctcf_rep1_peaks.narrowPeak >
    workdir/merge/sorted_ctcf_rep1_peaks.narrowPeak
sort -k8,8nr workdir/rep2/call_peak/ctcf_rep2_peaks.narrowPeak >
    workdir/merge/sorted_ctcf_rep2_peaks.narrowPeak

# 选项
# -k 8,8 选项使输入文件按其第8列进行排序
# -nr 选项指定按整个数字排序（而非一个一个字符比较）
```

而后进行合并:

```
cd workdir/merge/
idr --samples sorted_ctcf_rep1_peaks.narrowPeak sorted_ctcf_rep2_peaks
    .narrowPeak --input-file-type narrowPeak --rank p.value --output-
    file sample-idr --plot --log-output-file sample-idr.log

# 选项
# --samples: narrowPeak的输入文件（重复样本）
# --input-file-type: 输入文件格式为narrowPeak
# --rank p.value: 指定输入文件是以p-value排序
# --output-file: 指定输出文件路径
# --plot: 输出IDR度量值的结果
```

输出文件中，sample-idr 是共有 peaks 的结果输出文件，格式与输入文件格式类似，只是多了几列信息。前 10 列是标准的 narrowPeak 格式文件，包含重复样本整合后的 peaks 信息；后面 10 列包含 local 和 global IDR 值，样本 1 的信息和样本 2 的信息。

sample-idr.png 是程序输出的结果图，描述了两个重复中峰的保留情况，其解释详见图注。

10.5 ChIP-seq 结果的分析

10.5.1 计算与 TSS 的共定位

10.5.1.1 下载 TSS bed 文件

到网站下载 UCSC RefSeq bed 文件并上传到服务器，放在 genome 文件夹下。

下载文件的选项如图，只需要 5'UTR 序列位置:

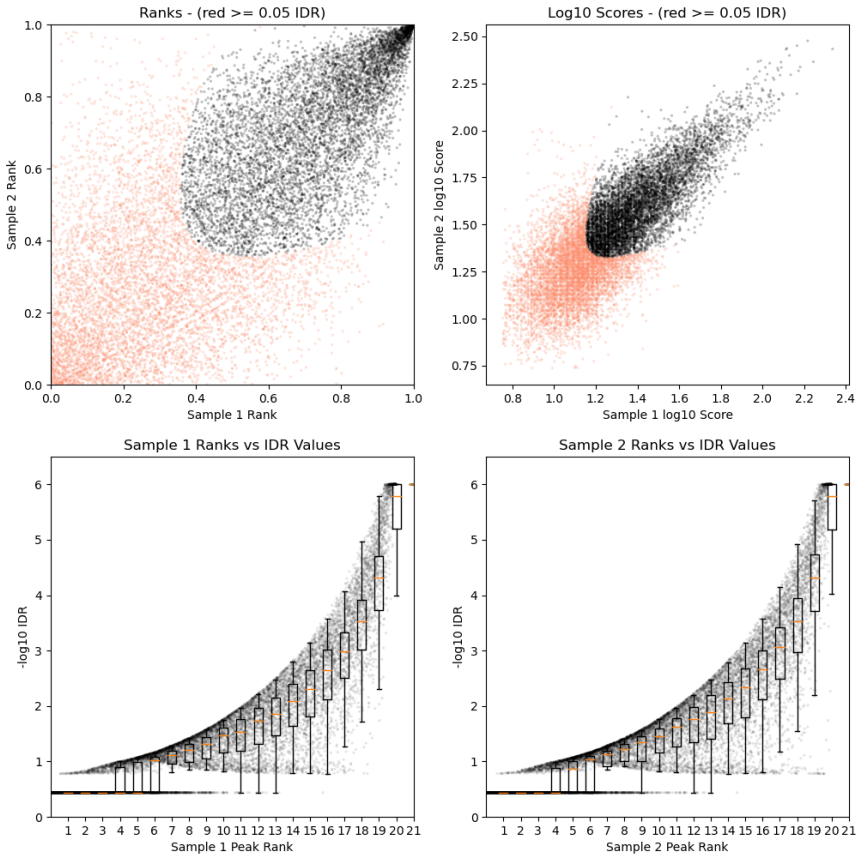


图 10.8: IDR 程序输出文件之一。左上，重复 1 的 peak rank(峰排名) 对比重复 2 的 peak rank(峰排名)，未通过 IDR 阈值的峰以红色显示；右上，重复 1 的 log10(peak score) 对比重复 2 的 log10(peak score)，未通过的 IDR 阈值的峰以红色显示；下方，黑色点图显示 peak rank 对比 IDR 分数，箱线图显示 IDR 值在每个 5% 部分中的分布。IDR 阈值默认设为 1e-6。

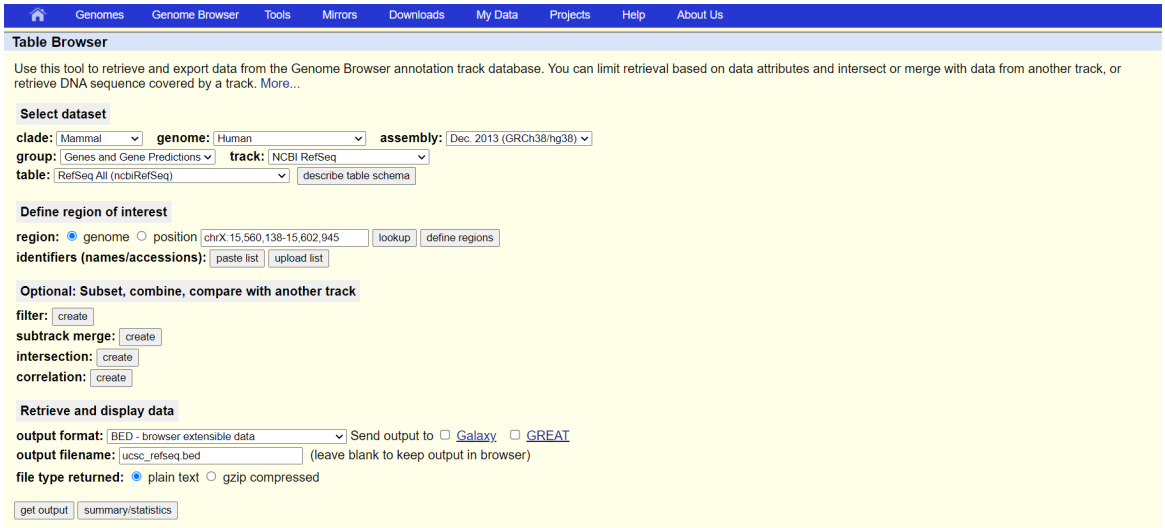


图 10.9: UCSC 下载选项（1），点击”get output”

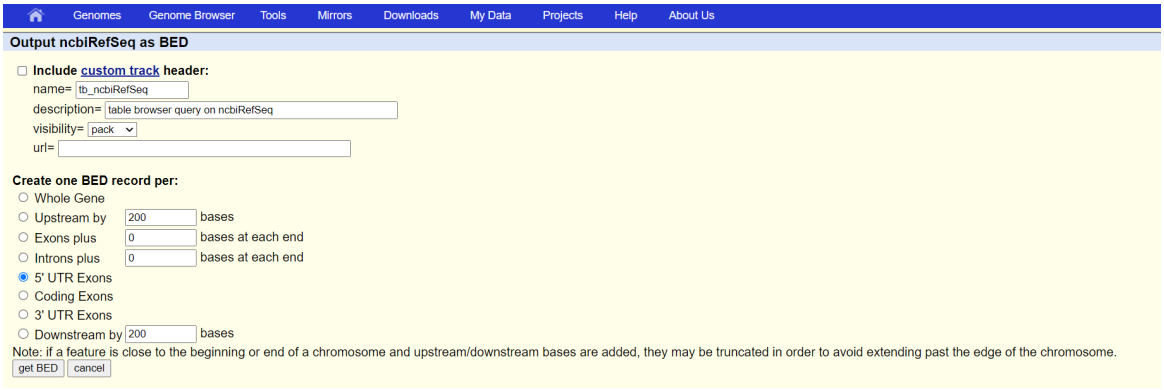


图 10.10: UCSC 下载选项（2），点击"get BED"

10.5.1.2 将 BAM 文件转换为 bigwig 文件

bigwig 文件记录的是“每个碱基”的 coverage 情况，可以理解为 ChIP-Seq 的信号，处理组和对照组都需要进行这个步骤。

使用 deeptools 软件包中的 bamCoverage 程序来完成。

```
1  bamCoverage --bam workdir/rep1/ctcf_rep1_mapped_filtered_dedup.bam -o
       workdir/rep1/ctcf_rep1_mapped_filtered_dedup.bw --binSize 10 --
       normalizeUsing RPGC --effectiveGenomeSize 2913022398 --extendReads
       -p 50 > workdir/rep1/bam_coverage.log 2>&1 &
2
3  bamCoverage --bam workdir/rep2/ctcf_rep2_mapped_filtered_dedup.bam -o
       workdir/rep2/ctcf_rep2_mapped_filtered_dedup.bw --binSize 10 --
       normalizeUsing RPGC --effectiveGenomeSize 2913022398 --extendReads
       -p 50 > workdir/rep2/bam_coverage.log 2>&1 &
4
5  bamCoverage --bam workdir/input/input_mapped_filtered_dedup.bam -o
       workdir/input/input_mapped_filtered_dedup.bw --binSize 10 --
       normalizeUsing RPGC --effectiveGenomeSize 2913022398 --extendReads
       -p 50 > workdir/input/bam_coverage.log 2>&1 &
6
7  # 选项
8  # --bam 输入的bam文件
9  # -o 输出的bigwig文件
10 # --binSize 计算coverage时的bin的大小
11 # --normalizeUsing 选择normalization的方法
12 # --effectiveGenomesize 如果需要normalization则需有设置有效基因组大小
13 # --extendReads 设置该值以拓展reads的长度，以计算出用于测序的DNA片段的
       长度；pair-end的测序可以自动估计DNA片段的长度
```

```
14 # -p 使用的线程数
```

10.5.1.3 计算信号矩阵

使用 computeMatrix 计算指定位置的信号矩阵:

```
1  computeMatrix reference-point -S workdir/input/
       input_mapped_filtered_dedup.bw workdir/rep1/
       ctcf_rep1_mapped_filtered_dedup.bw workdir/rep2/
       ctcf_rep2_mapped_filtered_dedup.bw -R genome/ucsc_refseq.bed -a
       2500 -b 2500 --samplesLabel input ctcf_rep1 ctcf_rep2 --sortRegions
        descend -o workdir/merge/tss_matrix.gz -p 50 > workdir/merge/
       computeMatrix.log 2>&1 &
2
3  # 选项
4  # reference-point 选择参考点模式，参考点是指 BED 区域内的位置（例如，起
       点）。在这种模式下，只会绘制参考点之前（上游）和/或之后（下游）的特定
       基因组位置
5  # -S 输入的bigwig文件
6  # -R 记录reference位点信息的bed文件
7  # -a 起始位点上游的距离
8  # -b 起始位点下游的距离
9  # --sortRegions 在矩阵中，信号按什么顺序排列
10 # --samplesLabel 在矩阵中，样品如何命名
11 # -o 输出的矩阵
12 # -p 线程数
13 # 如果最后的热图中出现黑色条状色块，可尝试增加--missingDataAsZero选项
```

10.5.1.4 绘图

```
1  plotHeatmap -m workdir/merge/tss_matrix.gz -out workdir/merge/
       ctcf_tss_matrix.svg --plotFileFormat svg --yAxisLabel RPGC --
       regionsLabel all_tss --legendLocation none
2
3  # 参数
4  # -m 输入的矩阵文件
5  # -out 输出的图片文件
6  # --plotFileFormat 指定输出文件格式
7  # --yAxisLabel 指定y轴标签
```

```
8 # --regionsLabel 指定区域标签
9 # --legendLocation 指定图例位置
```

10.5.2 Peak 注释

10.5.2.1 Homer

这里，我们首先使用 Homer 来进行峰的注释。

Homer 包含一个用于峰注释的程序 annotatePeaks.pl 。它可以将峰与其附近的基因关联起来，执行 GO 分析，基因组特征关联分析，将峰与基因表达数据关联起来，计算 ChIP-seq reads 密度，寻找峰中出现的 motif。该程序还可以创建直方图和热图。

annotatePeaks.pl 脚本在默认情况下，会寻找距离 peak 最近的 TSS（根据 RefSeq 注释），并根据该 TSS 所属的 gene 进行注释；此外，该脚本还会根据 peak 占据的位置，注释其基因组信息（处于 TSS, TTS, CDS, 5' UTR, 3' UTR, CpG islands, repeats, introns, intergenic）。

```
1 mkdir workdir/analysis
2 # 配置homer所需的基因组文件，注意替换USER_NAME及conda中configureHomer.
    pl的路径
3 perl /home/USER_NAME/Software/miniconda3/envs/ENV_NAME/share/homer/.//
    configureHomer.pl -install hg38
4 # 注释
5 annotatePeaks.pl workdir/merge/sample-idr hg38 > workdir/analysis/
    peak_annotation.txt &
6 # 注意这里使用的peak文件为sample-idr，这是两个重复合并以后的文件，类似
    BED格式
```

Homer 注释结果为表格文件，而非图像，缺乏一点直观展示。接下来尝试将 sample-idr 下载至本地，而后用 R 中的 ChIPseeker 包进行分析并绘图。

10.5.2.2 ChIPseeker

ChIPseeker 可以进行 Peak 注释，这里用它来展示 peak 相关的基因组区域的类别。

执行下面的 *R* 脚本：

```
1 # 安装BiocManager
2 if (!requireNamespace("BiocManager", quietly = TRUE))
3 install.packages("BiocManager")
4
5 # 安装ChIPseeker，如果遇到困难，可尝试增加force=TRUE
6 BiocManager::install("ChIPseeker")
7 BiocManager::install("GO.db")
```

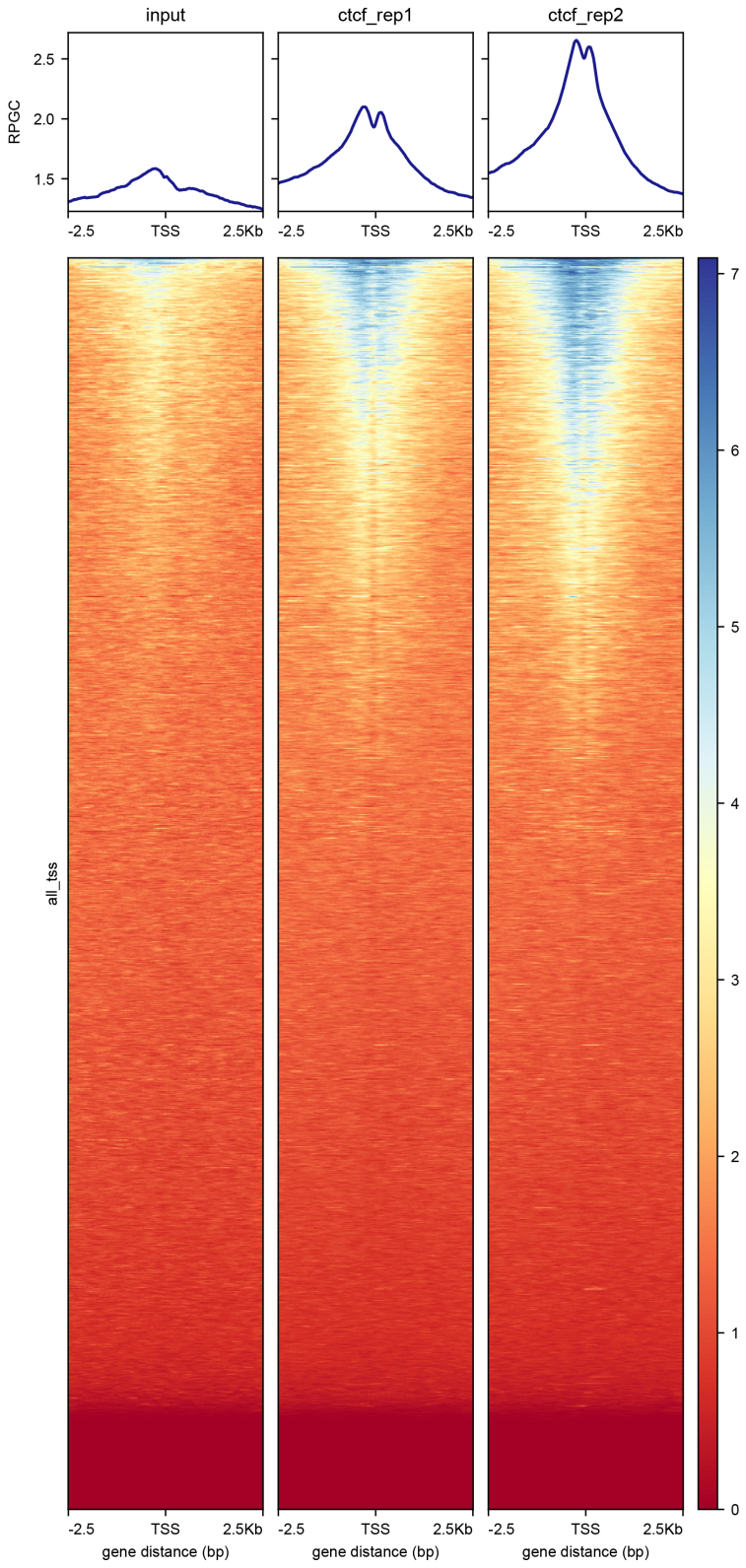


图 10.11: CTCF 结合位点与 TSS 的共定位热图。可见两个处理组均在 TSS 处有明显的富集，且呈现双峰模式。颜色表示 reads 密度。

```
BiocManager::install("DO.db")
BiocManager::install("org.Hs.eg.db")
BiocManager::install("TxDb.Hsapiens.UCSC.hg38.knownGene")

# 载入库
library("ChIPseeker")
library("TxDb.Hsapiens.UCSC.hg38.knownGene")
library("org.Hs.eg.db")

# 改变工作目录注意替换USER_NAME
setwd("C:/Users/USER_NAME/Desktop/LEBR")

# 注释
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
peak <- readPeakFile("sample-idr") # 注意这里使用的peak文件为sample-idr，这是两个重复合并以后的文件，类似BED格式
peakAnno <- annotatePeak(
peak,
tssRegion = c(-2500, 2500),
TxDb = txdb,
annoDb = "org.Hs.eg.db")

# 绘饼图
plotAnnoPie(peakAnno)
```

10.5.2.3　GO 富集分析

GO 分析的相关基础知识在第七章和第九章中都有提及，因此这里不再赘述，本项目使用 clusterProfiler 进行 GO 富集分析。

继续上一小节的分析，应该在相同环境中执行：

```
gene = as.data.frame(peakAnno)$geneId # 从峰注释的数据结构中取出基因列表
ego_mf <- enrichGO(gene = gene,
                   OrgDb = org.Hs.eg.db, # 选择人类基因组数据库
                   ont = "MF", # 选择GO注释分子功能
                   pAdjustMethod = "BH",
                   qvalueCutoff = 0.05,
                   readable = TRUE)
```

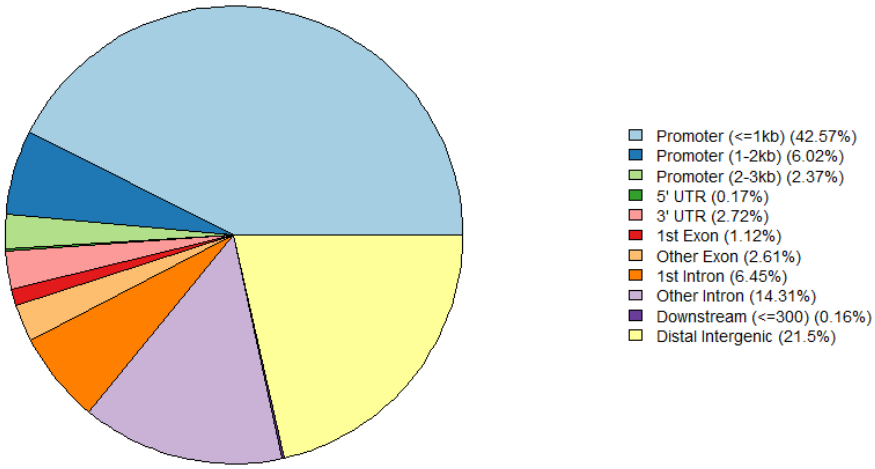


图 10.12: CTCF 结合位点相关的基因组区域，绝大部分结合位点（Peak）属于启动子，这与之前的 TSS 共定位分析一致，也与 CTCF 转录因子的本质一致。

```
ego_cc <- enrichGO(gene = gene,
                   OrgDb = org.Hs.eg.db,
                   ont = "CC", # 选择GO注释细胞组分
                   pAdjustMethod = "BH",
                   qvalueCutoff = 0.05,
                   readable = TRUE)
ego_bp <- enrichGO(gene = gene,
                   OrgDb = org.Hs.eg.db,
                   ont = "BP", # 选择GO注释生物过程
                   pAdjustMethod = "BH",
                   qvalueCutoff = 0.05,
                   readable = TRUE)

dotplot(ego_mf, showCategory=10) # 作图
dotplot(ego_cc, showCategory=10)
dotplot(ego_bp, showCategory=10)
```

CTCF_HUMAN 的 UniProt 主页给出了 CTCF 的参考 GO 注释。与我们的富集结果有相同和不同。抛砖引玉，我猜测这是因为 CTCF 并非某类基因特定的转录因子，这样的注释结果体现的是它作为染色质结构组分的功能，而不是激活某种特殊通路或反应的功能。但也可能有更高明的解释。

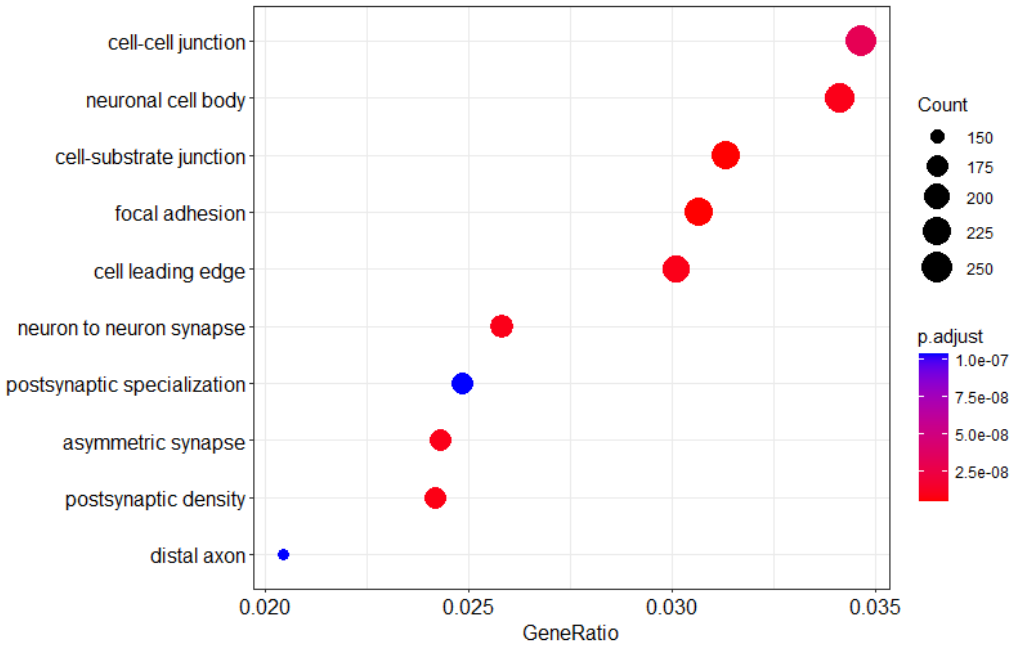


图 10.13: GO 富集分析气泡图 (1)-细胞组分层次。可见 CTCF 结合位点周边基因在细胞组分方面与细胞间连接、神经细胞胞体、细胞-基质链接、黏着斑相关。

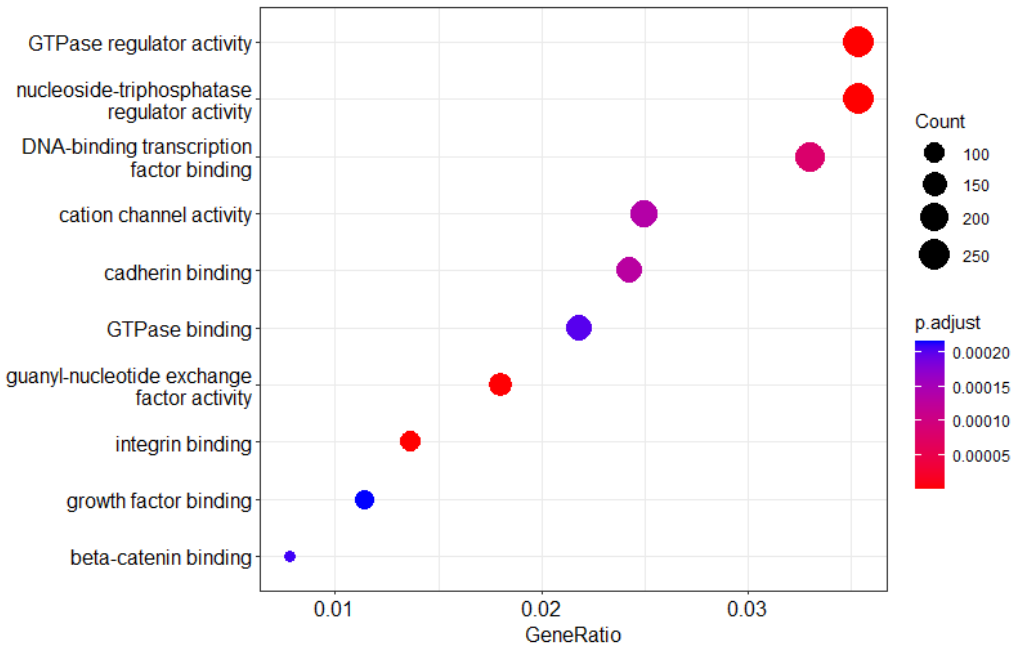


图 10.14: GO 富集分析气泡图 (2)-分子功能层次。可见 CTCF 结合位点周边基因在分子功能方面富集于 GTP 酶活性调控、核苷三磷酸酶活性调控、DNA 结合转录因子结合、钙黏蛋白结合等。

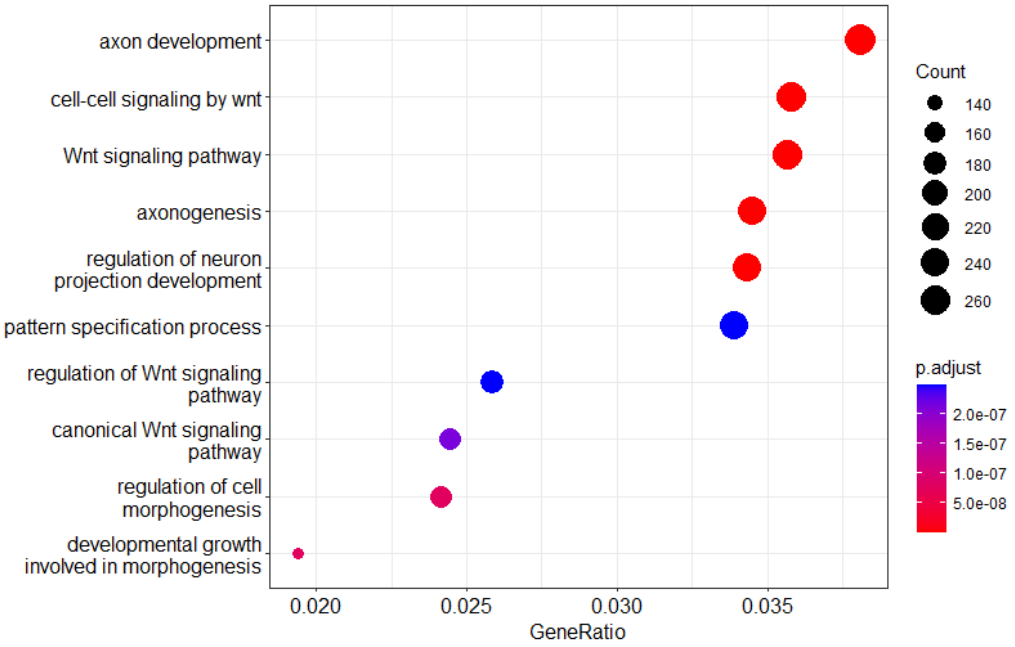


图 10.15: GO 富集分析气泡图 (3)-生物过程层次。可见 CTCF 结合位点周边基因富集于轴突发育、Wnt 信号介导的细胞信号转导、Wnt 信号通路、轴突发生。

10.5.2.4　KEGG 富集分析

KEGG是京都基因与基因组百科全书（Kyoto Encyclopedia of Genes and Genomes）的缩写。

KEGG 是一系列数据库，通过分子水平的信息，尤其是基因组测序和其他高通量实验技术产生的大规模分子数据集，了解细胞、生物体和生态系统等生物系统的高级功能和效用。

KEGG 中，最常用的数据库是 KEGG Pathway，我们使用这个数据库为 ChIP-seq Peak 作通路注释。

继续上一小节的分析，应该在相同环境中执行：

```
kegg <- enrichKEGG(gene, organism = 'hsa',
                   keyType = 'kegg',
                   pvalueCutoff = 0.05,
                   pAdjustMethod = 'BH',
                   minGSSize = 10, # 需要检测的最小基因长度
                   maxGSSize = 500, # 需要检测的最大基因长度
                   qvalueCutoff = 0.2,
                   use_internal_data = FALSE) # 选择使用在线数据
barplot(kegg, showCategory=20) # 绘制柱状图
```

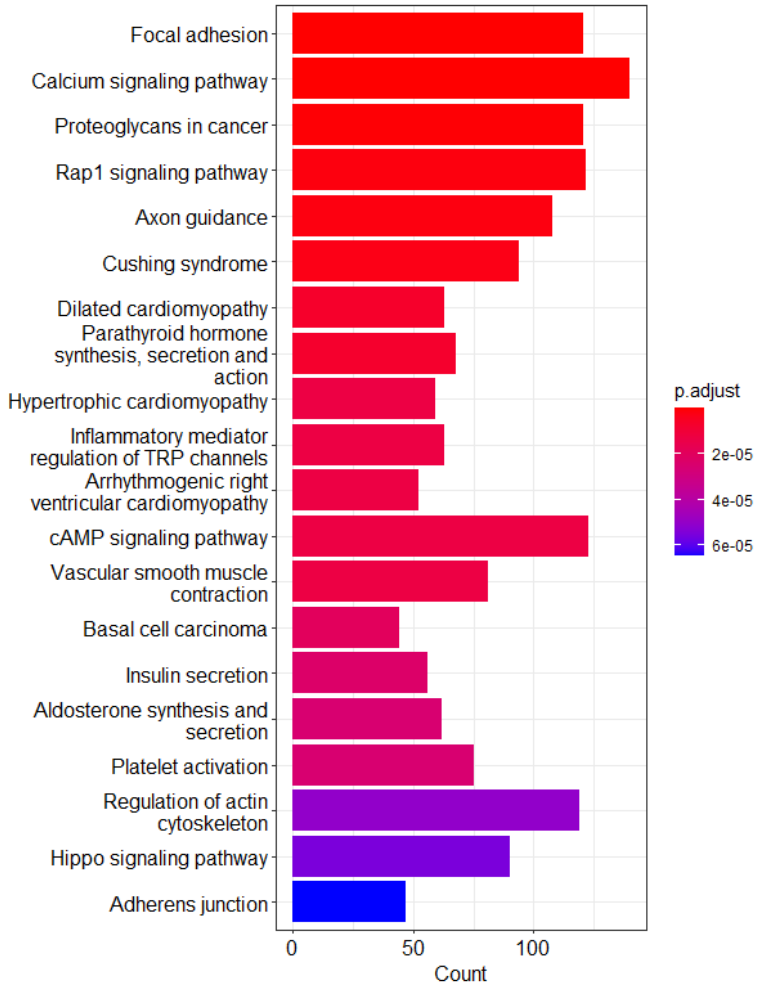


图 10.16: KEGG 富集结果柱状图。可见 CTCF 结合序列附近基因被富集到黏着斑、钙信号、癌症蛋白多糖、Rap1 信号、轴突导向等通路。结果与 GO 富集类似。

10.5.3 寻找 motif

CTCF 全称为 CCCTC 结合因子，因此，针对 CTCF 的 ChIP 理应发现结合位点具有相应的 motif。

这里我们使用 Homer 来寻找基序（motif），首先提取 peak 的位置信息，而后搜寻基序。

HOMER 最初是为了在 ChIP-Seq 峰中寻找基序而开发的。HOMER 读取基因组位置信息，不仅限于 ChIP-Seq 峰，来获得被富集的基序。用户只需要提供一个包含基因组坐标的文件（即 BED 文件或类似格式）。

分析基因组基序前必须配置适当的基因组，HOMER 可以处理自定义/任意基因组。用户可以指定包含 FASTA 格式的基因组序列的文件或目录的路径，而不是安装/配置基因组。基因组可以在单个 FASTA 文件中，或者指定一个目录，其中每个染色体可以在一个单独的文件中。

HOMER 可以执行已知 motif 搜索和从头 motif 搜索。

已知 motif 搜索：HOMER 将数据库中的可信 motif 对富集目标文件进行搜索，返回富集的 p-value 小于 0.05 的 motif。该种搜索较快，结果可在 `knownResults.html` 中查看。一般来说，一次成功的 ChIP-seq 应该能找到一些已经被报道过的可信的 motif。

从头（de novo）motif 搜索：不依赖于现有知识，完全通过计算在输入数据中搜寻，结果保存在 `homerResults.html` 中。

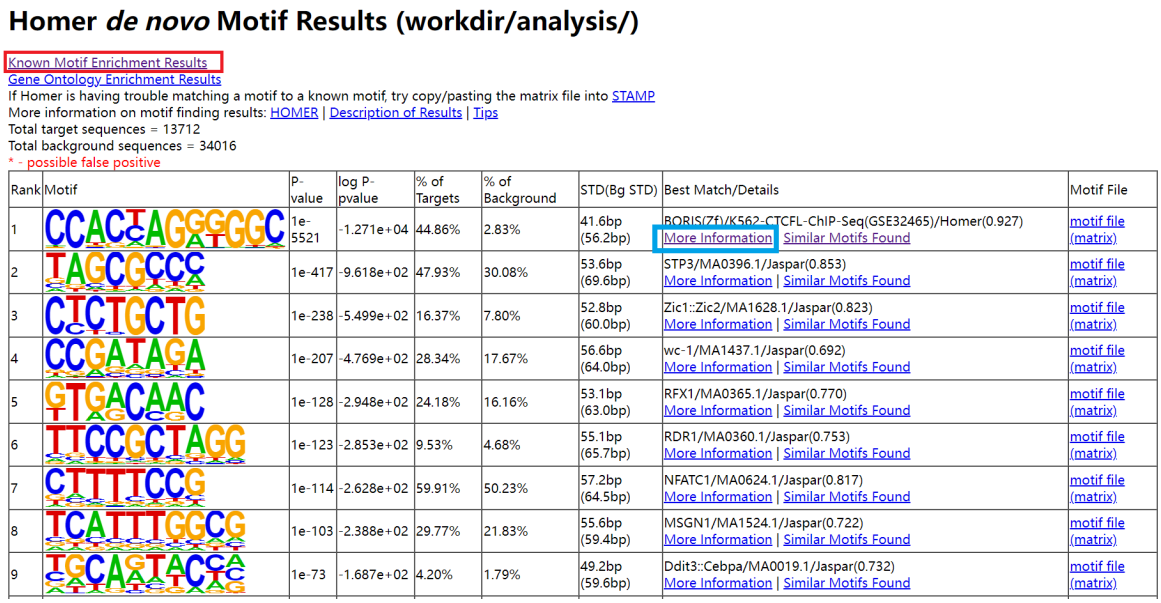
```
1  # 调整peak文件的格式
2  awk '{print $4"\t"$1"\t"$2"\t"$3"\t+"}' workdir/merge/sample-idr >
       workdir/analysis/sample-idr.homer
3  # sample-idr是两个重复的peak合并后的文件
4  # 这里awk程序将每一行的第4、1、2、3列挑出，并放入一个新文件
5
6  # 寻找motif
7  findMotifsGenome.pl workdir/analysis/sample-idr.homer genome/hg38.fa
       workdir/analysis -size 200 -len 8,10,12 -p 40
8  # 选项和参数
9  # workdir/analysis/sample-idr.homer，输入文件，有关基因组位置信息
10 # genome/hg38.fa 基因组文件，务必使用与比对时相同的基因组
11 # workdir/analysis 输出文件的存放目录
12 # -size 用于搜寻motif的区域大小，如果是转录因子的ChIP-seq峰，寻找基本
       motif推荐50bp，寻找基本motif和共富集的motif推荐200bp
13 # -len 指定要寻找的motif长度，可以指定多个值，HOMER将按每个长度寻找
       motif，最后综合起来；指定的长度越长，所花费的时间越多
14 # -p 线程数
```

输出结果中包括两个 HTML 文件：`homerResults.html` 、`knownResults.html` 。可将其下载到本地用浏览器打开。

`homerResults.html` 中列出了从头找出的 motif，点击图 10.17 中的红框中的链接可以跳转到 `knownResults.html` ，点击其中蓝框中的链接可以查看每个 motif 的详细信息。

`knownResults.html` 给出了从输入数据中找到的可信 motif，每个 motif 都给出了参考命名。

而从头发现 motif 的详细信息页面（如图 10.19 所示）将显示该 motif 及其反向互补 motif，一些基本信息，及其与已知 motif 的对应关系。可以看到，此 motif 与已知 motif 具有良好对应。本 ChIP-seq 实验是基本成功的。

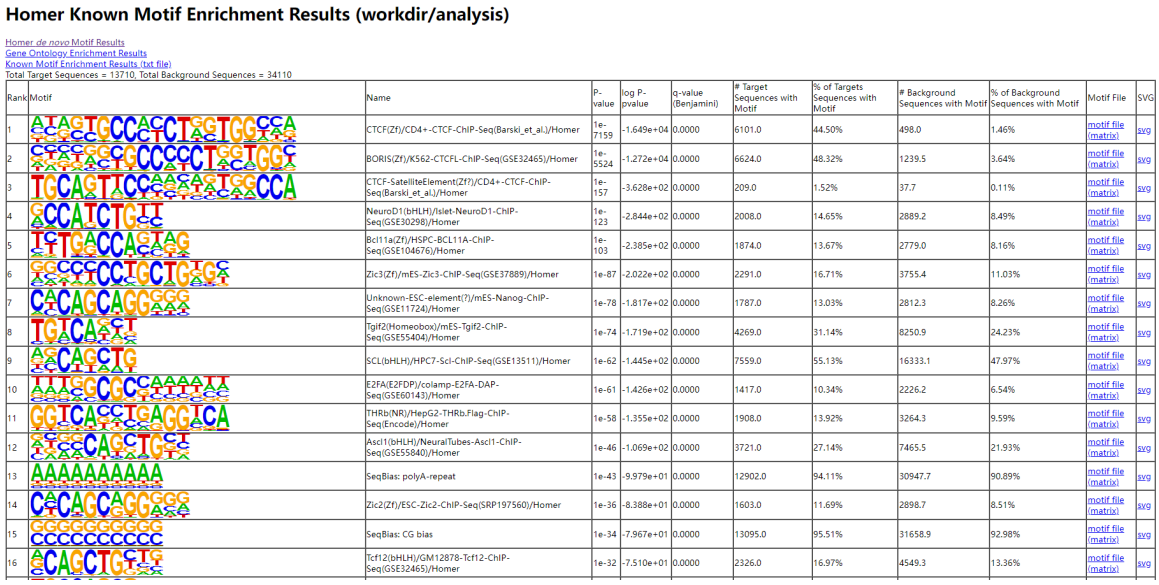


Homer *de novo* Motif Results (workdir/analysis/)

Known Motif Enrichment Results
Gene Ontology Enrichment Results
If Homer is having trouble matching a motif to a known motif, try copy/pasting the matrix file into STAMP
More information on motif finding results: HOMER | Description of Results | Tips
Total target sequences = 13712
Total background sequences = 34016
* - possible false positive

Rank	Motif	P-value	log P-pvalue	% of Targets	% of Background	STD(Bg STD)	Best Match/Details	Motif File
1		1e-5521	-1.271e+04	44.86%	2.83%	41.6bp (56.2bp)	BORIS(Zf)/K562-CTCFL-ChIP-Seq(GSE32465)/Homer(0.927) More Information | Similar Motifs Found	motif file (matrix)
2		1e-417	-9.618e+02	47.93%	30.08%	53.6bp (69.6bp)	STP3/MA0396.1/Jaspar(0.853) More Information | Similar Motifs Found	motif file (matrix)
3		1e-238	-5.499e+02	16.37%	7.80%	52.8bp (60.0bp)	Zic1::Zic2/MA1628.1/Jaspar(0.823) More Information | Similar Motifs Found	motif file (matrix)
4		1e-207	-4.769e+02	28.34%	17.67%	56.6bp (64.0bp)	wc-1/MA1437.1/Jaspar(0.692) More Information | Similar Motifs Found	motif file (matrix)
5		1e-128	-2.948e+02	24.18%	16.16%	53.1bp (63.0bp)	RFX1/MA0365.1/Jaspar(0.770) More Information | Similar Motifs Found	motif file (matrix)
6		1e-123	-2.853e+02	9.53%	4.68%	55.1bp (65.7bp)	RDR1/MA0360.1/Jaspar(0.753) More Information | Similar Motifs Found	motif file (matrix)
7		1e-114	-2.628e+02	59.91%	50.23%	57.2bp (64.5bp)	NFATC1/MA0624.1/Jaspar(0.817) More Information | Similar Motifs Found	motif file (matrix)
8		1e-103	-2.388e+02	29.77%	21.83%	55.6bp (59.4bp)	MSGN1/MA1524.1/Jaspar(0.722) More Information | Similar Motifs Found	motif file (matrix)
9		1e-73	-1.687e+02	4.20%	1.79%	49.2bp (59.6bp)	Ddit3::Cebpa/MA0019.1/Jaspar(0.732) More Information | Similar Motifs Found	motif file (matrix)

图 10.17: HOMER 从头（de novo）motif 搜索报告页面。



Homer Known Motif Enrichment Results (workdir/analysis)

Homer *de novo* Motif Results
Gene Ontology Enrichment Results
Known Motif Enrichment Results (txt file)
Total Target Sequences = 13710, Total Background Sequences = 34110

Rank	Motif	Name	P-value	log P-pvalue	q-value (Benjamini)	# Target Sequences with Motif	% of Targets Sequences with Motif	# Background Sequences with Motif	% of Background Sequences with Motif	Motif File	SVG
1		CTCF(Zf)/CD4+-CTCF-ChIP-Seq(Barski_et_al.)/Homer	1e-7159	-1.649e+04	0.0000	6101.0	44.50%	498.0	1.46%	motif file (matrix)	svg
2		BORIS(Zf)/K562-CTCFL-ChIP-Seq(GSE32465)/Homer	1e-5524	-1.272e+04	0.0000	6624.0	48.32%	1239.5	3.64%	motif file (matrix)	svg
3		CTCF-SatelliteElement(Zf?)/CD4+-CTCF-ChIP-Seq(Barski_et_al.)/Homer	1e-157	-3.628e+02	0.0000	209.0	1.52%	37.7	0.11%	motif file (matrix)	svg
4		NeuroD1(bHLH)/Islet-NeuroD1-ChIP-Seq(GSE30298)/Homer	1e-123	-2.844e+02	0.0000	2008.0	14.65%	2889.2	8.49%	motif file (matrix)	svg
5		Bcl11a(Zf)/HSPC-BCL11A-ChIP-Seq(GSE104676)/Homer	1e-103	-2.385e+02	0.0000	1874.0	13.67%	2779.0	8.16%	motif file (matrix)	svg
6		Zic3(Zf)/mES-Zic3-ChIP-Seq(GSE37889)/Homer	1e-87	-2.022e+02	0.0000	2291.0	16.71%	3755.4	11.03%	motif file (matrix)	svg
7		Unknown-ESC-element(?)/mES-Nanog-ChIP-Seq(GSE11724)/Homer	1e-78	-1.817e+02	0.0000	1787.0	13.03%	2812.3	8.26%	motif file (matrix)	svg
8		Tgif2(Homeobox)/mES-Tgif2-ChIP-Seq(GSE55404)/Homer	1e-74	-1.719e+02	0.0000	4269.0	31.14%	8250.9	24.23%	motif file (matrix)	svg
9		SCL(bHLH)/HPC7-Scl-ChIP-Seq(GSE13511)/Homer	1e-62	-1.445e+02	0.0000	7559.0	55.13%	16333.1	47.97%	motif file (matrix)	svg
10		E2FA(E2FDP)/colamp-E2FA-DAP-Seq(GSE60143)/Homer	1e-61	-1.426e+02	0.0000	1417.0	10.34%	2226.2	6.54%	motif file (matrix)	svg
11		THRb(NR)/HepG2-THRb.Flag-ChIP-Seq(Encode)/Homer	1e-58	-1.355e+02	0.0000	1908.0	13.92%	3264.3	9.59%	motif file (matrix)	svg
12		Ascl1(bHLH)/NeuralTubes-Ascl1-ChIP-Seq(GSE55840)/Homer	1e-46	-1.069e+02	0.0000	3721.0	27.14%	7465.5	21.93%	motif file (matrix)	svg
13		SeqBias: polyA-repeat	1e-43	-9.979e+01	0.0000	12902.0	94.11%	30947.7	90.89%	motif file (matrix)	svg
14		Zic2(Zf)/ESC-Zic2-ChIP-Seq(SRP197560)/Homer	1e-36	-8.388e+01	0.0000	1603.0	11.69%	2898.7	8.51%	motif file (matrix)	svg
15		SeqBias: CG bias	1e-34	-7.967e+01	0.0000	13095.0	95.51%	31658.9	92.98%	motif file (matrix)	svg
16		Tcf12(bHLH)/GM12878-Tcf12-ChIP-Seq(GSE32465)/Homer	1e-32	-7.510e+01	0.0000	2326.0	16.97%	4549.3	13.36%	motif file (matrix)	svg

图 10.18: HOMER 已知 motif 搜索报告页面。

Information for 1-CCACYAGGGGGC (Motif 1)

Reverse Opposite:

p-value:	1e-5521
log p-value:	-1.271e+04
Information Content per bp:	1.776
Number of Target Sequences with motif	6150.0
Percentage of Target Sequences with motif	44.86%
Number of Background Sequences with motif	964.7
Percentage of Background Sequences with motif	2.83%
Average Position of motif in Targets	100.6 +/- 41.6bp
Average Position of motif in Background	99.6 +/- 56.2bp
Strand Bias (log2 ratio + to - strand density)	-0.0
Multiplicity (# of sites on avg that occur together)	1.05
Motif File:	file (matrix) reverse opposite
SVG Files for Logos:	forward logo reverse opposite

Matches to Known Motifs

BORIS(Zf)/K562-CTCFL-ChIP-Seq(GSE32465)/Homer

Match Rank: 1
Score: 0.93
Offset: -1
Orientation: reverse strand
Alignment: -CCACYAGGGGGC-------
GCCASCAGGGGGCGCYVNNG

图 10.19: 从头发现 motif 编号 1 的详细信息。反向 motif 中可见 CCCTC 片段，这符合 CTCF 的特征。

10.6　页边集

本节是这份笔记的最后一节，是做期末项目一点感想，应该用笔写在书籍页边空白处。

《转录因子 CTCF 的 ChIP-Seq 数据初步分析》是 2022 年春季 LEB 课程第一组和第四组的期末项目，完成时间于 2022 年 5-6 月。2022 年 6 月 21 日，在课程期末网络研讨会上，两组向大家作了汇报。此后，我在 8 月重复了流程。

本项目帮助我们理解并运用所学到的 Linux 操作方法、理解二代测序数据分析流程中的重要概念、掌握 ChIP-Seq 的分析流程与基本原理。它作为一个生物信息学课程的练习项目，是非常好的。

很多时候，由于流程不断丰富，软件更新迭代，操作者容易将大量时间花费在安装和调试程序参数上，却疏于掌握计算原理、很少解读输出结果。有限的时间，以及应用上的急迫需求，容易使学习技术的过程陷入“理解要执行，不理解也要执行”的混沌。

但我组长饶希晨，神文圣武，廓开大业，英明决断，领导大家脱离了窠臼，拨开了迷雾。本项目从介绍 ChIP-seq 的原理到基本的分析，最后到结果作图和解读，无不贯彻理论与实际结合的理念，在执行中加深了理解，是好的，也是值得纪念的。

参考文献

[1] Park, P. J. (2009). ChIP–seq: advantages and challenges of a maturing technology. *Nature reviews genetics, 10*(10), 669-680.

[2] Kim, T. H., & Dekker, J. (2018). ChIP. *Cold Spring Harbor protocols, 2018*(4), 10.1101/pdb.prot082610.

[3] Kim, T. H., & Dekker, J. (2018). ChIP-seq. *Cold Spring Harbor Protocols, 2018*(5), pdb-prot082644.

[4] Kim, S., Yu, N. K., & Kaang, B. K. (2015). CTCF as a multifunctional protein in genome regulation and gene expression. *Experimental & molecular medicine*, 47(6), e166.

[5] Lou, S., Li, T., Kong, X., Zhang, J., Liu, J., Lee, D., & Gerstein, M. (2020). TopicNet: a framework for measuring transcriptional regulatory network change. *Bioinformatics (Oxford, England)*, 36(Suppl_1), i474–i481.

[6] Yin, M., Wang, J., Wang, M., Li, X., Zhang, M., Wu, Q., & Wang, Y. (2017). Molecular mechanism of directional CTCF recognition of a diverse range of genomic sites. *Cell research*, 27(11), 1365–1377.

[7] Cutadapt Documentation

[8] Picard Documentation

[9] GitHub Page of Irreproducible Discovery Rate (IDR)

[10] MACS2 PyPI Page

[11] Wilbanks, E. G., & Facciotti, M. T. (2010). Evaluation of algorithm performance in ChIP-seq peak detection. *PloS one*, 5(7), e11471.

[12] Landt, S. G., Marinov, G. K., Kundaje, A., Kheradpour, P., Pauli, F., Batzoglou, S., Bernstein, B. E., Bickel, P., Brown, J. B., Cayting, P., Chen, Y., DeSalvo, G., Epstein, C., Fisher-Aylor, K. I., Euskirchen, G., Gerstein, M., Gertz, J., Hartemink, A. J., Hoffman, M. M., Iyer, V. R., …Snyder, M. (2012). ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome research*, 22(9), 1813–1831.

[13] HOMER 主页

[14] MEME 软件包主页

[15] bioconductor:ChIPseeker 主页

[16] 用 ChIPseeker 进行 peak 注释和可视化

[17] 罗静初. Linux 生物信息技术基础. 课堂练习.

[18] 饶希晨. LEB 课程期末个人总结.

结语

"*I' ve learned that I still have a lot to learn.*"

—Maya Angelou