

Linux-based Essential Bioinformatics

LEB23G4：期末报告

作者：高大可，邓昆月，唐明川，吴航锐

时间：2023年6月30日

指导老师：罗静初



这是我们一学期的成果，如果 Harry 看到了，他也会感到高兴的吧

作者信息



- 高大可
- 北京大学生命科学学院 2022 级博士生
- 邮箱: gaodk@stu.pku.edu.cn
- GitHub 主页: <https://github.com/DrinaG>



- 邓昆月
- 北京大学化学与分子工程学院 2022 级博士生
- 邮箱: 879158131@qq.com
- GitHub 主页: <https://github.com/dkyyyyyyyyyyyy>



- 唐明川
- 北京大学元培学院 2020 级本科生
- 邮箱: 1418767078@qq.com
- GitHub 主页: <https://github.com/Tangmc-kawa>



- 吴航锐
- 北京大学化学与分子工程学院 2022 级博士生
- 邮箱: hrywu@qq.com
- GitHub 主页: <https://github.com/Alchemiiist>

前言

这本册子是 2023 年春季学期《Linux 生物信息技术基础》课程的期末总结。Linux 生物信息技术基础 (Linux-based Essential Bioinformatics, LEB) 是由罗静初教授¹开设的课程，罗老师以生物信息为主题，以 Linux 系统为载体，以上课和小组讨论为形式，带领初学者步入生物信息的大门，让有基础的学习者也能有所收获。

万分有幸的是，我们处在一个和谐快乐的小组，特色鲜明的各位组员让我们能各取所长，共同进步。高大可正从事蛋白质翻译调控相关研究；邓昆月；作为本科生，唐明川在空间转录组方向进行科研实践，打磨科研技能，发展科学思维，同时正在积极探索开拓自己的能力边界；吴航锐。

学期末，虽无考试压力，但一同回顾半年所得，梳理知识技能，也是一件令人愉悦的事情；为了使报告不只是考核，我们力图增加它的交流性，因而你可以看到每位成员都留下了自己的照片和联系方式，期待能够和大家建立联系，以我们所学帮助到大家。从课程整体来看，课程内容比较灵活，与生物信息相关的主线在于蛋白质和核酸序列处理，包括序列比对和 ChIP-seq、RNA-seq 等过程的上下游分析。主线之外，根据同学需要，老师安排了 GitHub，TBtools 使用介绍和基于 conda、vsCode、zshell 等的环境配置，这让我们受益匪浅。

我们选择从熟悉 Linux 系统到各个主题应用的逻辑将所学内容分为如下几个章节：

Linux 基础：事实上，在书写各类项目和在 GitHub 或各种开发者平台冲浪的过程中，你只需要很基础的 Linux 知识就能慢慢进化成大神，因此，本章力求简约，意在总结 Linux 的基础操作，介绍 Linux 的基本特性，让初学者以此为基础，探索属于自己的 Linux 编程之路。

序列比对和数据库介绍：序列比对和数据库使用是课程的重点，作为后续课程的前置知识，我们在此处简单向大家介绍序列比对原理和常见数据库使用方法介绍。

EMBOSS、BLAST 和 HMMER：作为生物信息的重点和 Linux 实践，我们将分三章，从易到难介绍当前常用的三种序列比对方法。当然，由于 EMBOSS 是一个软件包，除序列比对外它还有许多功能，不过其他功能不是我们的介绍重点，留给大家自由探索。

ChIP-seq 和 RNA-seq：我们通过 ChIP-seq 和 RNA-seq 两个项目向大家介绍测序数据上下游处理的全流程，这既是一个新的主题，也是此前所学内容的综合，你可以尝试用 shell 编程等方式将所学知识融会贯通。

TBtools：作为“番外篇”，我们在此介绍由个人开发者开发的生物信息整合工具——TBtools。

除了 Linux 和生物信息的主线，我们特将以下主题放置于附录中，此举主要出于便于大家随用随查的考量，并不意味着它们不重要：

环境配置：你可以在这里看到 Linux 环境配置，conda 环境管理器，docker 容器技术和 vsCode 代码编辑器的介绍。你当然可以选择在此处将日后可能需要的环境一并配好，提供便利；不过这部分内容设置的本意是作为“字典”，随用随查，不一定需要从头看到尾，在学习和实践的过程中，你或许会找到你最喜欢的环境配置方法。

Git 和 GitHub：Git 是一个版本控制神器，让我们摆脱手动进行版本控制的烦恼，同时方便与其他人的合作，本身已是一个足够好用的工具，基于 Git 的 GitHub 网站则是将这个工具用到了极致。作为开发者，个人开发的能力固然重要，但了解已有的开发成果、快速学习使用，加入社群与其他优秀的开发者建立合作关系才是人类生为群体动物最大的优势。GitHub 是世界上最大的开发者社群，它常常能给你很多惊喜！

“君子生非异也，善假于物也。”——《荀子·劝学》

最后，小组的 GitHub 仓库 [LEB23_G4](#) 记录了我们所有的报告和大部分代码，本期末报告的 Latex 源代码也在仓库中。预祝大家学习愉快！也希望能和大家一起不断完善此文档，和罗老师一起将《Linux 生物信息技术基础》这门课越办越好！

¹罗静初，北京大学生命科学院教授，博士生导师，欧洲分子生物学网络组织中国节点负责人，英国 Briefings in Bioinformatics 杂志编委。1947 年生，1970 年毕业于北京大学生物系。1986 年起从事 DNA 和蛋白质序列计算机分析。1987-1989 年赴美国马里兰大学进修访问，从事蛋白质分子模型和计算机在分子生物学中的应用研究。1991-1999 年先后 5 次赴英国帝国癌症研究所合作研究，从事蛋白质分子模型、蛋白质结构域分析和数据库构建、蛋白质回环数据库构建等研究。1996 年起主持和参加 863、973、211、985，以及自然科学基金委、教育部、北京市科委等项目。发表学术论文 30 多篇，主持翻译《生物信息学概论》、合作编写《生物信息学》、《分子生物学生前沿技术》，开设“实用生物信息技术”研究生课程。

目录

第一章 Linux 基础	1
1.1 Linux 及操作系统简介	1
1.1.1 Linux 发行版	1
1.2 Linux 教程网站	1
1.3 Linux 基本命令	1
1.3.1 服务器有关命令	1
1.3.2 文件操作指令	2
第二章 Alignment and database	5
2.1 比对	5
2.1.1 打分矩阵	5
2.1.2 全局比对	6
2.1.3 局部比对	6
2.2 数据库	7
2.2.1 核酸数据库	7
2.2.2 蛋白数据库	7
第三章 EMBOSS	10
3.1 EMBOSS 简介	10
3.2 软件功能实例	10
3.2.1 needle	10
3.2.2 water	11
3.2.3 edalig	12
第四章 BLAST	14
4.1 背景	14
4.2 BLAST	14
4.2.1 BLAST 的基本想法	14
4.2.2 Word size 和积分矩阵	14
4.3 BLAST 比对工具	15
4.4 blast 本地化安装配置	16
4.4.1 准备工作	16
4.4.2 进行本地 blast	17
第五章 HMMER	19
5.1 HMMER 简介	19
5.2 HMMER 网站的四种搜索方法	19
5.2.1 phmm	19
5.2.2 hmmscan	19
5.2.3 hmmsearch	19
5.2.4 jackhmmer	20
5.3 Hmmer 的网站使用实例：以 jackhmmer 为例	20
5.4 在 python 上使用 HMMER	23

第六章 TBtools	24
6.1 TBtools 简介	24
6.2 软件功能实例	25
6.2.1 ORF 预测	25
6.2.2 基因富集分析	26
6.2.3 其他常见功能	28
6.2.4 TBtools 插件安装	28
第七章 ChIP-Seq 项目实操	31
7.1 背景介绍	31
7.1.1 ChIP-Seq 原理	31
7.1.2 CTCF 简介	31
7.1.3 ChIP-Seq 分析前的准备工作	31
7.2 上游分析流程	33
7.2.1 准备工作	33
7.2.2 质量控制	34
7.2.3 数据清晰和筛选	34
7.2.4 比对	35
7.2.5 ChIP-Seq 分析	37
7.3 ChIP-Seq 下游分析	41
7.3.1 call motif	41
7.3.2 计算与 TSS 的共定位情况	43
第八章 RNA Seq	46
8.1 软件的安装	46
8.2 RNA-seq 上游分析	46
8.2.1 质控	46
8.2.2 使用 cutadapt 去接头	46
8.2.3 比对	46
8.2.4 定量	46
8.3 RNA-seq 下游分析	47
A 环境配置	53
A.1 Linux 环境配置	53
A.1.1 Windows Subsystem for Linux, WSL	53
A.1.2 WSL 配置	53
A.2 conda: 环境管理系统	53
A.2.1 conda 安装	53
A.2.2 conda 安装	54
A.2.3 conda 运行	54
A.2.4 环境	54
A.3 VSCode: 轻量化集成代码编辑器	55
A.3.1 VSCode 安装及中文环境配置	55
A.3.2 VSCode 连接 WSL、SSH	56
A.3.3 在 VSCode 上通过 SSH 连接远程 Linux 服务器	56
A.4 实例: VSCode 中使用 JupyterNotebook 运行 R	59

A.4.1 用 conda 配置 R 环境	59
A.4.2 安装 package	59
A.4.3 在 vsCode 中使用 JupyterNotebook 运行 R 程序	60
附录 B Git 和 GitHub 简介	61
B.1 Git	61
B.1.1 Git 的基本概念	61
B.1.2 Git 的命令操作	63
B.1.3 Git 的重要功能——分支	64
B.2 Github	67
B.2.1 实例：通过 GitHub 创建小组仓库并同步小组资料	68
B.3 实例：VSCode 中利用 Git 和 GitHub 管理文件	69
参考文献	71

第一章 Linux 基础

1.1 Linux 及操作系统简介

Linux 是一种免费的开源操作系统。它基于 Unix 操作系统，由 Linus Torvalds 于 1991 年创建。Linux 用于许多不同的设备，例如服务器，超级计算机，智能手机甚至汽车。它以其稳定性，安全性和灵活性而闻名。Linux 有许多不同的发行版或“distros”，例如 Ubuntu, Fedora, CentOS, Gentoo, Arch Linux 等。

Linux 是基于命令行的操作系统，这意味着用户使用文本命令与之交互，而不是像 Windows 或 Mac OS X 那样使用图形用户界面（GUI）。它使用户对其系统拥有更多的控制权，并使他们更容易自动化任务。

1.1.1 Linux 发行版

Linux 发行版是一个由 Linux 内核、GNU 工具、附加软件和软件包管理器组成的操作系统。它也可能包括显示服务器和桌面环境，以用作常规的桌面操作系统。这个术语之所以是“Linux 发行版”，是因为像 Debian、Ubuntu 这样的机构“发行”了 Linux 内核以及所有必要的软件及实用程序（如网络管理器、软件包管理器、桌面环境等），使其可以作为一个操作系统使用。

一些流行的 Linux 发行版包括 Ubuntu, Fedora, CentOS, Gentoo, Arch Linux 等。这些发行版都有自己的特点和优点，列举如下：

- **Fedora:** 面向开发人员，新版本的软件包更新频繁。
- **Debian:** 稳定，易于使用，适合服务器。
- **Ubuntu:** 易于使用，适合桌面用户。
- **CentOS:** 企业级操作系统，稳定性高。
- **Gentoo:** 高度可定制，适合高级用户。
- **Arch Linux:** 轻量级和自定义。

在本学期中，我们使用的是 Ubuntu。

1.2 Linux 教程网站

1. Data Science at the Command Line <https://datascienceatthecli.com/2e/>
2. 菜鸟 <https://www.runoob.com/linux/linux-tutorial.html>
3. 自由的 GNU/Linux 发行版 <https://www.gnu.org/distros/free-distros.html>
4. Ubuntu 问答社区 <https://askubuntu.com/>
5. 软件开发者 <https://www.csdn.net/>

1.3 Linux 基本命令

1.3.1 服务器有关命令

1.3.1.1 登陆服务器

windows 系统: window powershell

mac 系统: 终端 (我的系统)

使用 ssh 指令来进行用户名的登陆, 如: ssh leb4b@117.78.18.116。登陆后，需要输入密码

注：输入密码时，密码会进行保密，因此在界面上看不到输入的密码

1.3.1.2 查看服务器状态

- 查看服务器此时登陆用户名名单

指令 1:

```
w | less
```

指令 2:

```
who | less
```

- 查看我的用户名

指令:

```
whoami  
\end{listing}  
\item 修改自己用户的密码  
\begin{listing}  
指令: passwd
```

- 显示系统运行状态

指令:

```
top
```

1.3.2 文件操作指令

1.3.2.1 list 指令

- 查看下属文件

```
ls
```

- 查看下属文件时，同时查看文件的属性信息

```
ls -l
```

- 按照文件名大小排序

```
ls -s
```

- 按照时间排序

```
ls -t
```

- 按照字节大小排序

```
ls -h
```

- 显示目录下以.FASTA 结尾的文件

```
ls *. FASTA
```

- 显示目录下所有文件，并且一行只显示一个

```
ls -1
```

8. 列出根目录

```
ls ~
```

9. 逐级显示当前目录下所有子目录和文件

```
ls -lR
```

10. 逐屏显示当前目录下所有子目录和文件详细信息

```
ls -lR | less
```

1.3.2.2 cd 指令

1. 进入 ABC 子目录

```
cd ABC
```

2. 返回根目录

```
cd
```

3. 进入子目录 ABC 下面的二级子目录 EFG

```
cd ABC/EFG
```

4. 返回上级目录

```
cd ..
```

1.3.2.3 新建、复制、删除、移动文件操作

1. 新建文件

```
mkdir          #建立文件名  
mkdir 0306 0313 0320 #建立以0306 0313 0320命名的多个文件夹  
mkdir 0227/HB    #在子目录0227下建立二级子目录HB
```

```
cat > 文件夹名  
control c #并且可以输入信息,退出该文件指令为:  
cat 文件名 #此外还可以通过查看指令来看文件内的信息
```

```
nano 文件名
```

2. 复制文件

```
cp a b #将a复制,命名为b
```

3. 重命名文件名

```
mv a b #将a的名字改成b
```

4. 删除文件

```
rm b #将b文件夹删掉  
#注意:chmod -w 文件 , 该指令能够改变文件属性, 使之不能够被删除
```

5. 建立软连接

```
ln -s #源文件 目标文件
```

1.3.2.4 其他常用的指令

1. 查找文件的位置(例如查找 needle 的位置)

```
which needle  
whereis needle  
local needle
```

2. 显示系统存储空间不同分区名称、容量和使用状态

```
df
```

3. 显示当前目录下所有子目录和文件占用空间

```
du
```

4. 按 MB 为单位显示/tmp 目录占用空间

```
du -m /tmp
```

5. 将文件 209HBA.FASTA 压缩

```
gzip 209HBA.FASTA
```

6. 将子目录 0307 中及其子目录中所有文件生成归档文件

```
tar -cvf 0307.tar 0307
```

7. 输入路径自动补齐文件名:tab

8. 重复上一次使用的指令: 上箭头

第二章 Alignment and database

2.1 比对

当我们得到某个基因的序列，想要了解这段序列的基因比如：该序列属于哪个物种，该物种和其他物种的亲缘关系，该序列能够行使怎样的功能等等。这时候我们需要对该条序列进行序列比对，也就是在已知的数据库中，找到相似或者一致的序列。关于比对的话，这里我们介绍对比的基本原理：包括打分矩阵和原理算法。

2.1.1 打分矩阵

在序列比对算法中的替换矩阵又称为打分矩阵，其数学本质是统计权重。在序列比对中，我们一般需要给出一个定量的数值来描述两者的一致性和相似性。在此过程中，替换矩阵用来评价碱基或残基之间的相似性，在长期实践中，人们发现一些特定的碱基替换或者残基替换的频率是要高于另一些替换的，因此人们可以通过统计方法或者基于进化的突变模型来给每一种替换定义不同的分值，来体现出不同碱基或残基之间发生替换的可能性。其可以分成核酸序列替换矩阵和蛋白质序列替换矩阵。

1. 等价矩阵

其是最简单的记分矩阵。其相同核苷酸的匹配得分为 1，不同为 0。由于不含有碱基理化性质和不区别对待的替换，较少使用。

①

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

图 2.1：等价矩阵

2. 转换-颠换矩阵

核酸的碱基按照环结构划分两类：嘌呤（腺嘌呤 A、鸟嘌呤 G），有两个环；与嘧啶（胞嘧啶 C、胸腺嘧啶 T），只有一个环。如果环数发生变化，则称为颠换（嘌呤 \leftrightarrow 嘧啶），得 -5 分；如果环数不变，（C \leftrightarrow T，A \leftrightarrow G），称为转换，得 -1 分。而一般在进化过程中，转换发生的频率比颠换高。而相同碱基，得 1 分。

(2)

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

图 2.2: 转换-颠换矩阵

3. BLAST 矩阵

最早应用于 BLAST 软件，得名。经过大量比对后总结出的矩阵：相同核酸为 +5，不同为-4，这个矩阵被广泛使用。

(3)

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

图 2.3: blast 矩阵

2.1.2 全局比对

利用动态规划模型做序列的全局比对是有上个世纪七十年代芝加哥的两位科学家提出的一种算法，用来寻找全局最优解，该算法也叫做 Needleman-Wunsch algorithm。事实上，如果对两条序列做比对，我们又有打分规则，理论上我们可以对两条序列采用枚举法进行比对，最后选择比对分数最大。

例如有两个序列，GCATGCU 以及 GATTACA。NW 算法的具体步骤如下：

1. 构造一个如下形式的表格
2. 设计得分矩阵。例如：第 i 行第 j 列字母匹配 +1，不匹配-1，插入和删除（字母与空白对比）操作-1。
3. 将以上表格的第二行第二列的初始得分设为 0，通过公式
4. 从表格的右下角开始，根据之前记录的路径逐级返回，并通过对应的操作得到最优匹配的序列。

2.1.3 局部比对

Needleman-Wunsch algorithm 算法在早期比对给定的两条序列时受到广泛使用。然而，随着越来越多序列数据的产生，该算法对两条序列所有残基进行比对的算法也遇到了问题。首先，有些功能相近的蛋白，虽然在整体上的序列相似性不高，但是在局部某些功能域序列上会有很高的相似性，这些功能域常常能够独立的发挥作用，但仅靠全局比对是无法发现这些相似的蛋白质，另一方面，在核酸序列比对上，我们需要处理由于内含子导致的大片段的插入缺失造成的比对得分偏差，因而，我们需要新的方法来发现相似的局部序列。

1981 年由 Smith 和 Waterman 两人提出了一个 SmithWaterman 算法，简单来说，相较于全局比对，通过对比对公式迭代算法的一个简单调整，来引入了一个止损下线，SmithWaterman 算法实质上提供了在差异区域扩大之

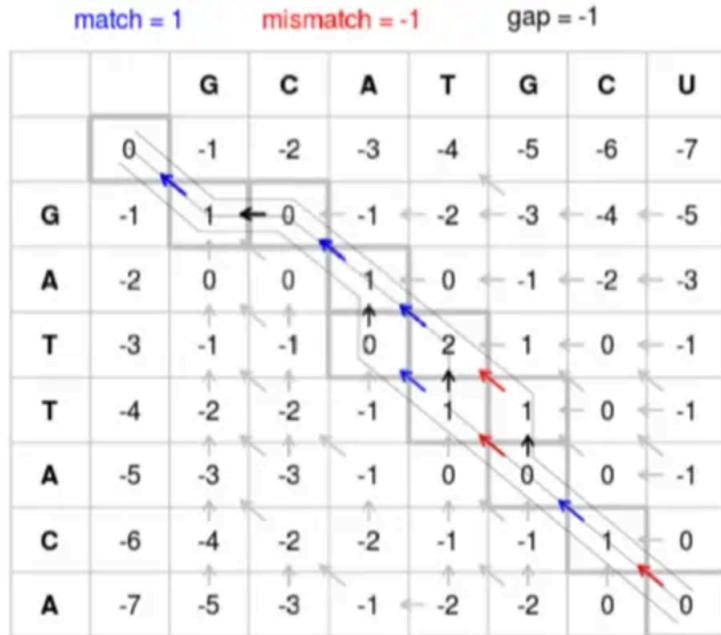


图 2.4: NW 算法

后，重启比对的一个方法，从而可以有效发现局部水平上的相似性。

2.2 数据库

用于生信分析的数据库大体上可以分为核酸数据库以及蛋白数据库。这里浅举核酸方面以及蛋白方面几个经典的数据库介绍。

2.2.1 核酸数据库

核酸数据库主要包括三个不同的机构建立的三大核酸数据库:NCBI, ENA, DDBJ

1. NCBI|Genbank:National Center for Biotechnology Information

NCBI 是由美国国家生物技术信息中心 (National Center for Biotechnology Information) 开发并负责维护, 隶属于美国国立卫生研究院 (National Institutes of Health, NIH)。这是一个功能非常强大的生物数据库, 里面可以做 DNA、RNA 甚至蛋白质的序列比对, 查找相关序列的注释信息。

2. ENA:ENA Browser

ENA: 欧洲核苷酸序列数据库 (European Nucleotide Archive), 由欧洲分子生物学研究室 (European Molecular Biology Laboratory, EMBL) 开发并维护。

3. DDBJ:DNA Data Bank of Japan

由日本国立遗传学研究所 (National Institute of Genetics, NIG) 开发并负责维护。以上三个数据库共同组成了国际核酸序列数据库合作联盟 (International Nucleotide Sequence Database Collaboration, INSDC)。即这个数据库的信息可以相互交换, 同步更新, 共享。

2.2.2 蛋白数据库

蛋白质一级结构的序列数据库也主要由三个数据库 (Swissprot, TrEMBL, PIR) 组成, 以上三个蛋白质序列数据库相关的机构共同成立了大家熟知的联合蛋白质序列数据库 (Universal Protein Resource, UniProt)。

UniProt 的主要功能包括高级检索、帮助文档、数据下载、统计报表等等。

National Library of Medicine
National Center for Biotechnology Information

All Databases Search

Welcome to NCBI

The National Center for Biotechnology Information advances science and health by providing access to biomedical and genomic information.

[About the NCBI](#) | [Mission](#) | [Organization](#) | [NCBI News & Blog](#)

Submit
Deposit data or manuscripts into NCBI databases

Download
Transfer NCBI data to your computer

Learn
Find help documents, attend a class or watch a tutorial

Develop
Use NCBI APIs and code libraries to build applications

Analyze
Identify an NCBI tool for your data analysis task

Research
Explore NCBI research and collaborative projects

Popular Resources

- PubMed
- Bookshelf
- PubMed Central
- BLAST
- Nucleotide
- Genome
- SNP
- Gene
- Protein
- PubChem

NCBI News & Blog

- GenBank Release 256.0 is Available! 28 Jun 2023
- Genbank release 256.0 (6/21/2023) is now available on the NCBI FTP site. This release has 24.45 trillion bases and 3.68
- NCBI Virus: Mutation-Based Search for SARS-CoV-2 Data 27 Jun 2023
- Millions of SARS-CoV-2 samples from around the world have been made
- NCBI SARS-CoV-2 Resources Page Will Redirect Soon 21 Jun 2023
- End of the COVID-19 Public Health Emergency During the COVID-19

[More...](#)

COVID-19 Information

[Public health information \(CDC\)](#) | [Research information \(NIH\)](#) | [SARS-CoV-2 data \(NCBI\)](#) | [Prevention and treatment information \(HHS\)](#) | [Español](#)

图 2.5: NCBI 网站

FOLLOW NCBI

UniProt BLAST Align Peptide search ID mapping SPARQL

Release 2023_02 | Statistics Help

Proteins UniProt Knowledgebase
Reviewed: 569,516 Unreviewed: 1,071,349 Total: 249,306,459

Species Proteomes
Protein sets for species with sequenced genomes from across the tree of life

Protein Clusters UniRef
Clusters of protein sequences at 100%, 90% & 50% identity

Sequence Archive UniArchive
Non-redundant archive of publicly available protein sequences seen across different databases

ProtNLM Predictions
Browse all the entries annotated with Google's ProtNLM predictions

UniProt COVID-19 portal
UniProt portal for the latest SARS-CoV-2 coronavirus protein entries and receptors, updated independent of the general UniProt release cycle

Supporting Data

Taxonomy	Keywords	Literature Citations
Human diseases	Cross-referenced databases	Subcellular locations
Automatic annotations: UniRule & ARBA		

A shrewd tweak protein spotlight
The chairs were rickety. So I rummaged around the kitchen drawer, extracted an old knife, and used its tip to drive a few screws back into the wood. The knife kept on losing grip and I kept on swearing.

#UsingUniProt – Dis...
In recent years a wealth of information has become available about genetic...

How artificial intellig...
A conversation with machine learning engineer Andreea Gare. At UniProt we...

Latest News

Forthcoming changes
Planned changes for UniProt
UniProt release 2023_03
The fair price of an (art) lunch | Changes to the controlled vocabulary of human disease...
UniProt release 2023_02
Leveraging the DNA | Changes in prokaryotic taxonomy | Changes to the controlled...

Analysis Tools

- BLAST**
Search with a sequence to find homologs through pairwise sequence alignment
- Align**
Align two or more protein sequences with UniProt to find conserved regions
- Search with Lists Map IDs**
Find protein/peptide lists of UniProt IDs or convert from/to other database IDs
- Search Peptides**
Search with a peptide sequence to find all UniProt proteins that contain exact matches

图 2.6: uniprot 网站

关于 Uniprot 的具体介绍，这里推荐罗静初老师在《生物信息学》杂志上发表的《UniProt 蛋白质数据库简介》一文，该文详细介绍了该数据库的发展历史、主要内容、网站功能模块、统计报表等部分。

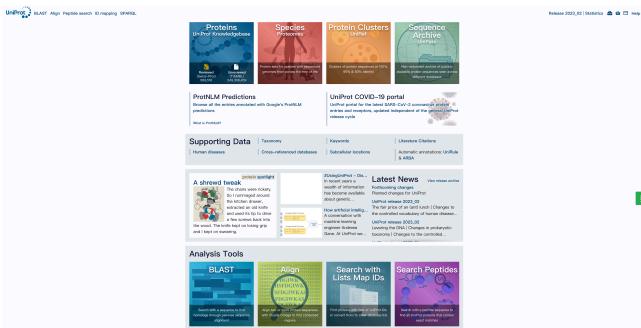


图 2.7: uniprot 蛋白数据库简介

第三章 EMBOSS

3.1 EMBOSS 简介

EMBOSS 为整合了多个生物信息学软件的软件包（软件包主页 <http://emboss.open-bio.org>），其作者为以 Peter Rice 为领导的来自欧洲生物信息学研究所的团队。

EMBOSS 包含多种功能的生物信息学工具，如基于 Needleman-Wunsch 动态规划算法的全局比对程序 needle，基于 Smith-Waterman 算法的局部比对程序 water 等。

EMBOSS 有三种常见的运行方式：命令行界面、图形界面、浏览器界面。其中，浏览器界面依赖于通过互联网访问装有 EMBOSS 的服务器进行；图形界面、命令行界面主要依赖于 Linux 系统完成。本小节以命令行界面演示代码为主，参数式运行所有命令。

3.2 软件功能实例

3.2.1 needle

needle 程序用于两序列的全局比对。以人类 CEAM3、CEAM4 蛋白质序列比对为例，使用 needle 程序进行分析：

```
#!/bin/usr
path="/rd1/home/public/EMBOSS/input"
needle ${path}/CEAM3_HUMAN.fasta ${path}/CEAM4_HUMAN.fasta CEAM3-CEAM4.NEEDLE -gapo 20 -gape 2
```

-gapo: 起始空位罚分，默认为 10。

-gape: 延伸空位罚分，默认为 0.5。

CEAM3、CEAM4 序列：

```
(base) leb4a@bbt:~/0227/CEA$ cat /rd1/home/public/EMBOSS/input/CEAM3_HUMAN.fasta
>CEAM3_HUMAN | P40198 | J Luo 2020-07-29
MGPPSASPHECIPWQGLLLTASLLNFWNPPTTAKLTIESMPLSVAEGKEVLLLVHNLPQ
HLFGYSWYKGGERVDGNSLIVGYVIGTQQATPGAAAYSGRETIYTNASLLIQNVQTQNDIGFY
TLQVIKSDLVNEEATGQFHVYQENAPGLPVGAVAGIVTGVLVGVALVAALVCFLLLAKTG
RTSIQRDLKEQQPQALAPGRGPSSHSAFSMSPLSTAQAPLPNPRTAASIYEELLKHDTNI
YCRMMDHKAEVAS
(base) leb4a@bbt:~/0227/CEA$ cat /rd1/home/public/EMBOSS/input/CEAM4_HUMAN.fasta
>CEAM4_HUMAN | 075871 | J Luo 2020-07-29
MGPPSAAPRGGHRPWQGLLITASLLTFWHPPTTVQFTIEALPSSAAEGKDVLLLACNISE
TIQAYYYWHKGKTAEGSPLIAGYITDIQANIPGAAYSGRETVPNGSLLFQNITLEDAGSY
TLRTINASYDSDQATGQLHVHQNVPGLPVGAVAGIVTGVLVGVALVAALVCFLLLSRTG
RASIQRDLREQPPPASTPGHGPSHRSTFSAPLPSRPTATPIYEELLYSDANIYCQIDHKA
DVVS
```

输出结果：

```

# Aligned_sequences: 2
# 1: CEAM3_HUMAN
# 2: CEAM4_HUMAN
# Matrix: EBL0SUM62
# Gap_penalty: 20.0
# Extend_penalty: 2.0
#
# Length: 252
# Identity: 158/252 (62.7%)
# Similarity: 188/252 (74.6%)
# Gaps: 8/252 ( 3.2%)
# Score: 801.0
#
#=====
CEAM3_HUMAN      1 MGPPSASPHRECIPWQGLLTASLLNFWNPPPTAKLTIESMPLSVAEGKE      50
| | | | : | . . . . | | | | : | | | | . | | : | | | . . | | | : | . | | | :
CEAM4_HUMAN      1 MGPPSAAPRGGHRPWQGLLITASLLTFWHPPTTVQFTIEALPSSAAEGKD      50
|
CEAM3_HUMAN      51 VLLLVLHNLPGQHLFGYSWYKGGERVDGNSLIVGYVIGTQQATPGAAAYSRET      100
| | | . | : . . . . | : | | . . . | . | | . . . | . | | | | | | | | |
CEAM4_HUMAN      51 VLLLACNISSETIQAYYWHKGKTAEGSPLIAGYITDIQANIPGAAYSGRET      100
|
CEAM3_HUMAN      101 IYTNASLLIQNVQTQNDIGFYTLQVIKSDLVNNEATGQFHVYQENAPGLPV      150
: | . | | | . | : | . . | . | | : | . . . . : | | | . | | : | . | | | |
CEAM4_HUMAN      101 VYPNGSLLFQNITLEDAGSYTLRTINASYDSDQATGQLHVHQNNVPGLPV      150
|
CEAM3_HUMAN      151 GAVAGIVTVGVLVGVVALVAALVCFLLLAKTGRTSIQRDLKEQQPQALAPGR      200
| | | | | | | | | | | | | | | | | | | | : | | | . | | | | | | | | . | . | . | .
CEAM4_HUMAN      151 GAVAGIVTVGVLVGVVALVAALVCFLLSRTGRASIQRDLREQPPPASTPGH      200
|
CEAM3_HUMAN      201 GPSHSSAFSMSPLSTAQAPLPNPRTAASIYEELLKHDTNIYCRMDHKAEV      250
| | | . | . | | | | : | | | . | | | | | | . | | | : | | | : | | | : |
CEAM4_HUMAN      201 GPSHRSTFS-----APLPSRPTATPIYEELLYSDANIYCQIDHKADV      242
|
CEAM3_HUMAN      251 AS      252
. |
CEAM4_HUMAN      243 VS      244

```

3.2.2 water

water 程序用于序列的局部比对。以人类 CEAM3、CEAM5 蛋白质序列比对为例，使用 water 程序进行分析：

```

#!/bin/usr
path="/rd1/home/public/EMBOSS/input"
water ${path}/CEAM3_HUMAN.fasta -sbegin 35 -send 142 \
${path}/CEAM5_HUMAN.fasta -sbegin 35 -send 142 \
./CEAM3-CEAM5.WATER -gapo 20 -gape 2

```

- sbegin: 比对起始位点。
- send: 比对终止位点。
- gapo: 起始空位罚分，默认为 10。
- gape: 延伸空位罚分，默认为 0.5。

CEAM3、CEAM5 序列：

```
(base) leb4a@bbt:~/0227/CEA$ cat /rd1/home/public/EMBOSS/input/CEAM3_HUMAN.FASTA
>CEAM3_HUMAN | P40198 | J Luo 2020-07-29
MGPPSASPHRECIPWQGLLLTASLLNFWNPPTAKLTIESMPLSVAEGKEVLLLvhNLPQ
HLFGYSWYKGERVDGNSLIVGYVIGTQQATPGAAAYSGRETIYTNASLLIQNVTQNDIGFY
TLQVIKSDLVNEEATGQFHVYQENAPGLPVGAVAGIVTGVLVGVALVAALVCFLLAKTG
RTSIQRDLKEQQPQALAPGRGPSSHSAFSMSPLSTAQAPLPNRTAACIYEELLKHDTNI
YCRMDHKAEVAS
(base) leb4a@bbt:~/0227/CEA$ cat /rd1/home/public/EMBOSS/input/CEAM5_HUMAN.FASTA
>CEAM5_HUMAN | P06731 | J Luo 2020-07-29
MESPSAPPHRWCIPWQRLLLTASLLTFWNPPTTAKLTIESPFNVAEGKEVLLLvhNLPQ
HLFGYSWYKGERVDGMRQIIGYVIGTQQATPGPAYSGREIIYPNASHLLIQNIIQNDIGFY
TLHVIKSDLVNEEATGQFRVYPELPKPSISSNNSKPVEDKDAVAFTCEPETQDATYLWVV
NNQSLPVSPRLQLSNGNRRTLTLFNVRNDTASYKCETQNPVSARRSDSVILNVLYGPDAP
TISPLNTSYRSGENLNLSCHAASNPPAQYSWFVNNGTFQQSTQELFIPNITVNNSGSYTCQ
AHNSDTGLNRTTVTTITVYAEPPKPFITSNNSNPVEDEDAVALTCEPEIQTTLWVVNN
QSLPVSPRLQLSNDNRTLTLFSVTRNDVGPYECGIQNELSVHSDPVILNVLYGPDPTI
SPSYTYYRPGVNLNSLSCHAASNPQAQYSWLLIDGNIQQHTQELFISNITEKNSGLYTCQAN
NSASGHRSRTTVKTITVSAELPKPSISSNNSKPVEDKDAVAFTCEPEAQNTTTLWVVNGQS
LPVSPRLQLSNGNRRTLTLFNVRNDARAYVCGIQNSVSANRSDPVTLDVLYGPDPTIISP
PDSSYLSGANLNLSCHASASPSPQYSWRINGIPQQHTQVLFIAKITPNNNGTYACFVSNL
ATGRNNNSIVKSITVSASGTSPGLSAGATVGIMIGVVLVVALI
```

输出结果：

```
=====
#
# Aligned_sequences: 2
# 1: CEAM3_HUMAN
# 2: CEAM5_HUMAN
# Matrix: EBLOSUM62
# Gap_penalty: 20.0
# Extend_penalty: 2.0
#
# Length: 107
# Identity: 93/107 (86.9%)
# Similarity: 96/107 (89.7%)
# Gaps: 0/107 ( 0.0%)
# Score: 483.0
#
#
=====

CEAM3_HUMAN      35 KLTIESMPLSVAEGKEVLLLvhNLPQHFGYSWYKGERVDGNSLIVGYVI    84
                  |||||.:.|||||:|||||:|||||:|||||:|||||:|||||:|||||:|||:|||
CEAM5_HUMAN      35 KLTIESPFNVAEGKEVLLLvhNLPQHFGYSWYKGERVDGMRQIIGYVI    84
                  |||||.|||||.|||.|||||:||||.||||.|||||:|||||:|||||:|||
CEAM3_HUMAN      85 GTQQATPGAAAYSGRETIYTNASLLIQNVTQNDIGFYTLQVIKSDLVNEEA  134
                  |||||.|||||.|||.|||||:||||.||||.|||||:|||||:|||||:|||
CEAM5_HUMAN      85 GTQQATPGPAYSGREIYPNASLLIQNIIQNDTGFYTLHVIKSDLVNEEA  134
                  |||||.|||||.|||.|||||:||||.||||.|||||:|||||:|||||:|||
CEAM3_HUMAN      135 TGQFHVY     141
                  ||||.|||
CEAM5_HUMAN      135 TGQFRVY     141
                  ||||.|||
```


3.2.3 edialign

edialign 程序用于序列的多重比对。以人类所有 CEA 蛋白质序列比对为例，使用 edialign 程序进行分析：

```
#!/bin/usr
path="/rd1/home/public/EMBOSS/input"
edialign ${path}/12HUMAN_CEA.FASTA 12HUMAN_CEA.EDIA 12HUMAN_CEA.ALN
```

12 个人类 CEA 氨基酸序列：

```
(base) leb4a@bbt:~/0227/CEA$ cat /rd1/home/public/EMBOSS/input/12HUMAN_CEA.FASTA
>CEAM3_HUMAN | P40198 | J Luo 2020-07-29
MGPPSAPHRECIPWQGLLLTASLLNFWNPPTTAKLTIIESMPLSVAEGKEVLLVHNLPQ
HLFGYSWYKGERVDGNSLIVGYVIGTQQATPGAAYSGRETIYTNASLLIQNVTQNDIGFY
TLQVIKSDLVNEEATGQFHVYQENAPGLPVGAVAGIVTGVLVGVALVAALVCFLLLAKTG
RTSIQRDLKEQQPQALAPRGPSHSSAFSMSPLSTAQAPLPNPRTAASIYEELLKHDNTI
YCRMDHKAEVAS
>CEAM4_HUMAN | 075871 | J Luo 2020-07-29
MGPPSAAPRGGHRPWQGLLITASLLTFWHPPPTVQFTIEALPSSAAEGKDVLLLACNISE
TIQAYYWHKGKTAEGSPLIAGYITDIQANIPGAAYSGRETVYPNGSLLFQNTLEDAGSY
TLRTINASYDSDQATGQLHVHQNNVPGLVGAVAGIVTGVLVGVALVAALVCFLLSRTG
RASIQRDLREQPPPASTPGHGPSHRSTFSAPLPSPRTATPIYEELLYSDANIYCQIDHKA
DVVS
>CEA19_HUMAN | Q7Z692 | J Luo 2020-07-29
MEIPMGTQGCFSKSLLLSASILVLWMLQGSQAALYIQLIKEPQPQKNQDLLSVQGPDTF
QDFNWYLGEETYGGTRLFTYIPGIQRPQRDGSAMGQRDIVGFPNGSMILLRAQPTDSGY
QVAITINSEWTMKAKTEVQVAEKNKELPSTHLPTNAGILAATIIGSLAAGALLISCIAYL
LVTRNWRQSHRLPAPRGQGSLSILCSAVSPVPSVTPSTWMATTEKPELGAHDAGDNNI
YEVMPSPVLLVSPISDTRSIINPARPLPTPHLQAEPENHQYQQDLLNPDPAPYCQLVPTS
>CEAM7_HUMAN | Q14002 | J Luo 2020-07-29
MGSPSACPYRVCIPWQGLLLTASLLFWNLPNQAQTNIIDVVVPFNVAEGKEVLLVHNESQ
NLYGYNWYKGERVHANYRIIGYVKNISQENAPGPAHNGRETIYPNGTLLIQNVTNDAGF
YTLLHVIKENLVEEVTRQFYVFSEPPKPSITSNNFNVPENKDIVVLTCPETQNTTYLW
VNNQSLLVSPRLLLSTDNRLLVLLSATKNDIGPYECEIQNPNVGASRSDPVTLNVRYESVQ
ASSPDLSAGTAVSIMIGVLAGMALI
>CEA21_HUMAN | Q3KPI0 | J Luo 2020-07-29
```

部分输出结果:

```
(base) leb4a@bbt:~/0227/CEA$ head 12HUMAN_CEA.ALN
>CEAM3_HUMAN
MG-PPSAPHRECIPWQGLLLTASLLNFWNP-TTAKLTIESMPLSVAEGKEVLLVHN
PQHLFGYSWYKGERVDGNSLIVGYVIG-TQQATP-GAAYSGRETI-YTNASLLIQNVTQ
N-DIGFYTLQVIKSDLVNEEATGQFHVYQE-----
```


参考资料

- [1] 罗静初.EMBOSS 和 EMNet[J]. 生物信息学,2021,19(04):223-231.
- [2] 罗静初.EMBOSS 软件包序列分析程序应用实例 [J]. 生物信息学,2021,19(01):1-25.

第四章 BLAST

4.1 背景

对于研究序列同源性，序列的相似程度问题，序列比对算法是非常重要的工具。双序列比对可以采用基于动态规划算法的 Needleman-Wunsch 和 Smith-Waterman 算法，虽然精度高，但是计算消耗大，因此当与数据库进行比对时，这两种算法就显得力不从心。Blast 采用启发式算法，通过丢失灵敏度来减少运行时间，以实现序列与大规模数据库的比对。

4.2 BLAST

BLAST (Basic Local Alignment Search Tool) 算法的基本思想是通过产生数量更少的但质量更好的增强点来提高比对的速度。算法的原理主要分为以下五步：(1) 过滤：首先过滤掉低复杂度区域，即含有大量重复的序列；(2) Seeding：将 Query 序列中每 k 个字组合成一个表，即将一个序列拆分成多个连续的‘seed words’（通常蛋白质 $k=3$, 核酸 $k=11$ ）；(3) 比对：列出我们所关心的所有可能的字组，再配合置换矩阵给出高分值的字组并组织成快速搜索树结构或者哈希索引，因此此步骤可以快速搜索出大数据集中的所有匹配序列，找到每个 seed words 在参考序列中的位置；(4) 延伸：当找到 seed words 的位置后，接下来需要将 seed word 延伸成长片段，延伸过程中，得分值也在变化，当得分值小于阈值时即停止延伸，最后得到的片段成为高分片段对，HSP (High-scoring segment pair)；(5) 显著性分析，最后我们使用如下公式计算 E 值， E 值衡量了在随机情况下，数据库存在的比当前匹配分数更好的比对的数目，因此可以用该值作为指标评价 HSP 比对序列的可信度

$$E = kmne^{-\lambda S}$$

其中， m 是数据库长度， n 是 query 的长度， S 是 HSP 分数，其他两个参数是修正系数。

4.2.1 BLAST 的基本想法

1. 首先将输入序列切分成若干小段——seed words，对于一个给定的字长 w (usually 3 for proteins and 11 for nucleotides), 将输入序列分成许多连续的 seed words
2. 通过事先建立的索引表，在数据库中快速定位候选序列，以及在候选序列中的具体位置（通过正确设计索引结果，这一步可以在线性甚至常数时间范围内完成，从而提高效率）
3. 通过对所有的 seed 重复上述操作，就可以得到查询序列与候选序列之间的 hit map —— 最优比对对应的路径，应当平行于主对角线。
4. 进一步去掉零散的 hits，仅保留沿对角线方向上，有两个及以上连续 hits 的 hit clusters，以便进一步缩小搜索空间
5. 以 hit cluster 为基础，向左右两个方向，延伸扩展，知道总分数的下降超过一个给定的 x 值之后，停止延伸
6. 在扩展后的区域，应用动态规划算法，确定最终的比对

4.2.2 Word size 和积分矩阵

Word size 和积分矩阵是 BLAST 算法中的两个重要参数。Word size (W) 指的是在查询序列中用于匹配数据库序列的单词长度。积分矩阵 (scoring matrix) 用于计算序列比对的得分。调整这些参数可以优化 BLAST 搜索。例如，如果 T 与 W 成比例缩放，则较小的 word size 会增加灵敏度并降低速度。 W , T 和积分矩阵之间的相互作用至关重要，明智地选择它们是控制 BLAST 速度和灵敏度最有效的方法

4.3 BLAST 比对工具

BLASTP 是蛋白序列到蛋白库中的一种查询。库中存在的每条已知序列将逐一地同每条所查序列作一对一的序列比对。

BLASTX 是核酸序列到蛋白库中的一种查询。先将核酸序列翻译成蛋白序列（一条核酸序列会被翻译成可能的六条蛋白），再对每一条作一对一的蛋白序列比对。

BLASTN 是核酸序列到核酸库中的一种查询。库中存在的每条已知序列都将同所查序列作一对一对地核酸序列比对。

TBLASTN 是蛋白序列到核酸库中的一种查询。与 BLASTX 相反，它是将库中的核酸序列翻译成蛋白序列，再同所查序列作蛋白与蛋白的比对。

TBLASTX 是核酸序列到核酸库中的一种查询。此种查询将库中的核酸序列和所查的核酸序列都翻译成蛋白（每条核酸序列会产生 6 条可能的蛋白序列），这样每次比对会产生 36 种比对阵列。

Table 1. Key features of the BLAST search pages in the “Basic BLAST” category

Search page	Query & database	Alignment	Programs & functions (default program in bold)
nucleotide blast	nucleotide vs nucleotide	Nucleotide	megablast : for sequence identification, intra-species comparison discontiguous megablast : for cross-species comparison, searching with coding sequences blastn : for searching with shorter queries, cross-species comparison
protein blast	Protein vs protein	protein	blastp : general sequence identification and similarity searches Quick BLASTP: with a kmer match to accelerate search speed for very similar proteins DETA-BLAST [3]: protein similarity search with higher sensitivity than blastp PSI-BLAST: iterative search for position-specific score matrix (PSSM) to identify distant relatives for a protein family PHI-BLAST: protein alignment with input pattern as anchor/constraint
blastx	nucleotide (tr) vs protein	protein	blastx : for identifying potential protein products encoded by a nucleotide query
tblastn	protein vs nucleotide (tr)	protein	tblastn : for identifying database sequences encoding proteins similar to the query
tblastx	nucleotide (tr) vs nucleotide (tr)	protein	tblastx : for identifying nucleotide sequences similar to the query based on their coding potential

其中 blastp 中又有一些细分的方法，适用于比对不同相似程度的序列。quickblastp 适用于快速比较非常相似的蛋白序列；delta-blast 相比普通的 blastp 具有更高的灵敏度；PSI-blast 使用 PSSM 打分矩阵，可以比较蛋白家族中关系较远的序列；PHI-blast 将输入模式作为约束的蛋白比对，找到与查询序列具有一样的表达模式且具有同源性的蛋白序列。

其中 blastn 中也有一些细分的方法，具有不同的特点。除去标准方法 blastn，magablast 使用模糊搜索加快比对速度，用于优化非常相似的序列比较，或鉴定某段核酸序列是否在数据库中；DiscontiguousMagablast 比 blastn 灵敏度更高，用于精确比对，适用于跨物种间的同源比对。

此外还有一些特殊功能的 blast:

1. Primer-BLAST: 使用 primer3 算法设计引物，并使用 BLAST 比对所选序列的模板特异性
2. SmartBlast: 可对使用者的蛋白查询进行处理，从数据库中选出 5 个最佳的蛋白匹配项，并给出一个简明的摘要
3. IgBLAST: 针对种系数据库搜索免疫球蛋白或 T 细胞受体序列以注释输入的免疫球蛋白序列，可以报告 D/J 区的相似对象；注释数据中的不同结构域
4. MOLE-BLAST: 从选定目标数据库中识别输入核苷酸序列的相邻序列（使用 blast），然后使用多重序列比对（MUSCLE）根据序列相似性聚类。
5. VecScreen: 根据已知载体库和其他人工序列筛选输入的核酸序列，确定污染
6. CD-search: 与保守结构域数据库做比较，根据数据库检索蛋白序列进行功能分析
7. CDART: 识别输入蛋白序列中的保守结构域，然后寻找含有这些保守结构域的其他序列。
8. Targeted loci: 从细菌和古菌中的 16S rRNA，或真菌的 18S, 28S 和 ITS 中搜索仔细挑选的核苷酸序列，用于物种鉴定的需要

4.4 blast 本地化安装配置

配置 local BLAST 可以有效增加 BLAST 稳定性和比对速度。local BLAST 由 ncbi BLAST 和数据库组成，用户可根据自己需要的数据库选择性下载，如 rna_reference 数据库 300GB 左右，rna_select 数据库不超过 50GB。

4.4.1 准备工作

源码安装 BLAST

```
# 复制安装包到目标目录
cp /rd1/home/public/BLAST/BLAST_TOOL/ncbi-blast-2.13.0+-x64-linux.tar.gz ~/install

# 解压解包
gunzip ncbi-blast-2.13.0+-x64-linux.tar.gz
tar xvf ncbi-blast-2.13.0+-x64-linux.tar

# 复制可执行程序
cp -rf install/ncbi-blast-2.13.0+/bin/ ~/bin/blast_bin/

# 用户根目录隐藏文件.ncbirc用于指定数据库文件位置
# Specifies the path where BLAST databases are installed
[BLAST]
BLASTDB=/rd1/home/public/blast_db
# 自己下载的数据库或自己构建的数据库也需要添加到此配置文件中

# 环境配置
#### BLAST环境变脸配置
#### .bashrc文件
# add the following code to the ~/.profile using vim/nano or other editor
# "~/.app/BLAST/ncbi-blast-2.13.0+/bin" is customized dir of your program's bin

# set blast bin to PATH TangMingchuan 20230320
if [ -d "$HOME/.app/BLAST/ncbi-blast-2.13.0+/bin" ]; then
    PATH="$HOME/.app/BLAST/ncbi-blast-2.13.0+/bin:$PATH"
fi

#### .profile文件
# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
```

```
source ~/.bashrc
```

4.4.2 进行本地 blast

数据库可以从 NCBI 上下载构建好的数据库，也可以自己构建数据库，这里选择自己构建数据库

```
##### step0 下载NCBI数据库
# 查看可以下载的数据库
update_blastdb.pl --showall
# 如欲下载指定的数据库，如swissprot
update_blastdb.pl --decompress swissprot

##### step1 创建索引文件，建立索引所需数据库

ln -s /rd1/home/public/blast_data/sbp/ZMTF_PEP.fasta .
ln -s /rd1/home/public/blast_data/sbp/ZMTF_CDS.fasta .
makeblastdb -dbtype prot -in ZMTF_PEP.fasta -out ZMTF_PEP
makeblastdb -dbtype nucl -in ZMTF_CDS.fasta -out ZMTF_CDS

### -in: 待格式化的序列文件
### -dbtype: 数据库类型，prot或nucl
### -out: 个人数据库命名，之后使用数据库使用这个名称

##### step2 进行blast比对
## 根据不同的需求选择合适的blast工具
## 不同的blast工具的应用范围虽然不同，但是基本参数都是一致的，这里以blastp为例。
```

blastp 的一般参数

1. -query 指定查询序列
2. -db 指定使用的数据库
3. -out 指定输出文件名
4. -evalue 指定可接受的最高假阳性期望
5. -outfmt 指定输出文件的格式
6. -word_size 指定词长
7. -matrix 指定计分矩阵
8. -query_loc 指定查询序列中要查询片段的起止位置
9. -subject 指定使用的被查询的序列文件
10. -subject_loc 指定被查询序列文件的起始位置
11. -max_target_seqs 指定搜索结果中显示的最大序列数目

outfmt 的类型共有 18 个

- 0 = Pairwise,
- 1 = Query-anchored showing identities
- 2 = Query-anchored no identities
- 3 = Flat query-anchored showing identities
- 4 = Flat query-anchored no identities,
- 5 = BLAST XML,
- 6 = Tabular,

- 7 = Tabular with comment lines,
 - 8 = Seqalign (Text ASN.1),
 - 9 = Seqalign (Binary ASN.1),
 - 10 = Comma-separated values,
 - 11 = BLAST archive (ASN.1),
 - 12 = Seqalign (JSON),
 - 13 = Multiple-file BLAST JSON,
 - 14 = Multiple-file BLAST XML2,
 - 15 = Single-file BLAST JSON,
 - 16 = Single-file BLAST XML2,
 - 17 = Sequence Alignment/Map (SAM),
 - 18 = Organism Report
- 其中 6, 7, 10, 17 可以自定输出格式。

```
blastq -query ZMTF_PEP.fasta -db swissprot -out ZMTF_SW.txt -eval 0.01 -outfmt 7 -word_size 3 -matrix PAM250
```

第五章 HMMER

5.1 HMMER 简介

HMMER is used for searching sequence databases for sequence homologs, and for making sequence alignments. It implements methods using probabilistic models called profile hidden Markov models (profile HMMs).

HMMER is often used together with a profile database, such as Pfam or many of the databases that participate in Interpro. But HMMER can also work with query sequences, not just profiles, just like BLAST. For example, you can search a protein query sequence against a database with phmmmer, or do an iterative search with jackhmmer.

HMMER is designed to detect remote homologs as sensitively as possible, relying on the strength of its underlying probability models. In the past, this strength came at significant computational expense, but as of the new HMMER3 project, HMMER is now essentially as fast as BLAST.

HMMER can be downloaded and installed as a command line tool on your own hardware, and now it is also more widely accessible to the scientific community via new search servers at the European Bioinformatics Institute.

<http://hmmer.org/>

5.2 HMMER 网站的四种搜索方法

5.2.1 phmmmer

single protein sequence against protein sequence database

使用蛋白质序列在蛋白质数据库中搜索

phmmmer is used to search one or more query protein sequences against a protein sequence database. For each query sequence in seqfile, use that sequence to search the target database of sequences in seqdb, and output ranked lists of the sequences with the most significant matches to the query.

<https://www.mankier.com/1/phmmmer>

5.2.2 hmmsecan

single protein sequence against profile HMM library (Pfam, CATH-Gene3D, PIRSF Superfamily and TIGRFAMs).

序列搜序列谱（以 HMM 构建的序列谱）

hmmsecan is used to search protein sequences against collections of protein profiles. For each sequence in seqfile, use that query sequence to search the target database of profiles in hmmdb, and output ranked lists of the profiles with the most significant matches to the sequence.

<https://www.mankier.com/1/hmmsecan>

5.2.3 hmmsearch

either multiple sequence alignment or profile HMM against protein sequence database.

序列谱搜序列

hmmsearch is used to search one or more profiles against a sequence database. For each profile in hmmfile, use that query profile to search the target database of sequences in seqdb, and output ranked lists of the sequences with the most significant matches to the profile.

<https://www.mankier.com/l/hmmsearch>

5.2.4 jackhmmer

iterative searches. Initiated with a single sequence, a profileHMM or a multiple sequence alignment against a target sequence database.

迭代搜索

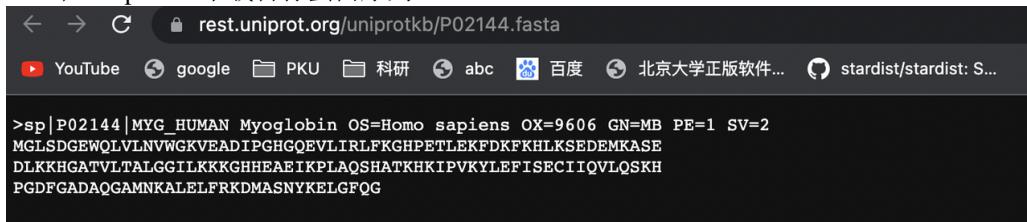
jackhmmer iteratively searches each query sequence in seqfile against the target sequence(s) in seqdb. The first iteration is identical to a phmm search. For the next iteration, a multiple alignment of the query together with all target sequences satisfying inclusion thresholds is assembled, a profile is constructed from this alignment (identical to using hmmbuild on the alignment), and profile search of the seqdb is done (identical to an hmmsearch with the profile).

<https://www.mankier.com/l/jackhmmer>

5.3 Hmmer 的网站使用实例：以 jackhmmer 为例

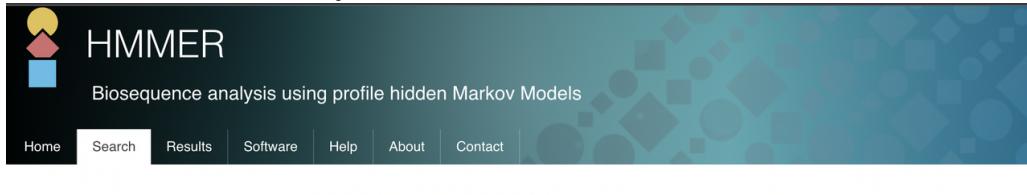
WSL 提供了一个完整的 Linux 内核，但它不是一个虚拟机。相反，WSL 提供了一个 Linux 系统调用兼容层，以便可以在 Windows 上运行原生 Linux 二进制文件。这意味着您可以在 Windows 上运行 Linux 命令行工具、脚本和应用程序，而无需使用虚拟机或双引导设置。

(1) 在 uniprot 上下载目标蛋白序列



```
< → C rest.uniprot.org/uniprotkb/P02144.fasta
YouTube google PKU 科研 abc 百度 北京大学正版软件... stardist/stardist: S...
>sp|P02144|MYG_HUMAN Myoglobin OS=Homo sapiens OX=9606 GN=MB PE=1 SV=2
MGLSDGEWQLVNVWGKVEADIPGHQEVILRLFKGHPETLEKFDKFHLKSEDEMKA
DLKKHGATVLTALGGILKKKGHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVLQSKH
PGDFGADAQGAMNKALELFRKDMASNYKELGFQG
```

(2) 登陆 hmmer 网址-search-jackhmmer，输入序列



iterative search vs protein sequence database

Paste a Sequence or an Alignment | Upload a File | Accession Search

Paste in your sequence (example), HMM (example) or multiple sequence alignment (example)◎

Submit Reset

(3) 选择比对数据库与评估标准

The screenshot shows the 'Sequence Database' section of the Hmmer website. It includes a dropdown menu for 'Current database selection' set to 'Reference Proteomes'. Below it is a 'Cut-Offs' section with two tabs: 'E-value' (selected) and 'Bit score'. Under 'E-value', there are fields for 'Significance E-values' (0.01) and 'Report E-values' (1). There is also a 'Restrict by Taxonomy' button.

(4) 得到比对结果

The screenshot shows the 'Sequence Matches and Features' results page. At the top, it displays 'Pfam' and 'Globin' (154 hits). Below are filters for 'disorder', 'coiled-coil', and 'tm & signal peptide'. A 'Distribution of Significant Hits' chart shows the distribution of significant hits from most to least significant. The main table lists 'Significant Query Matches (4590)' with columns for Row, Target, Secondary Accessions & IDs, Description, Kingdom, Phylum, Species, Cross-references, E-value, and a 'Customise' button. The first four rows are shown:

Row	Target	Secondary Accessions & IDs	Description	Kingdom	Phylum	Species	Cross-references	E-value
> 1	MYG_HUMAN	P02144	Myoglobin	Eukaryota	Chordata	Homo sapiens		1.9e-99 <input checked="" type="checkbox"/>
> 2	G1RW45_NOMLE	G1RW45	Myoglobin	Eukaryota	Chordata	Nomascus leucogenys		1.2e-98 <input checked="" type="checkbox"/>
> 3	A0A2R9BL13_PANPA	A0A2R9BL13	Myoglobin	Eukaryota	Chordata	Pan paniscus		1.2e-98 <input checked="" type="checkbox"/>
> 4	MYG_PANTR	P02145	Myoglobin	Eukaryota	Chordata	Pan troglodytes		1.2e-98 <input checked="" type="checkbox"/>

(5) 右上角 customise 可以选择需要列举的属性

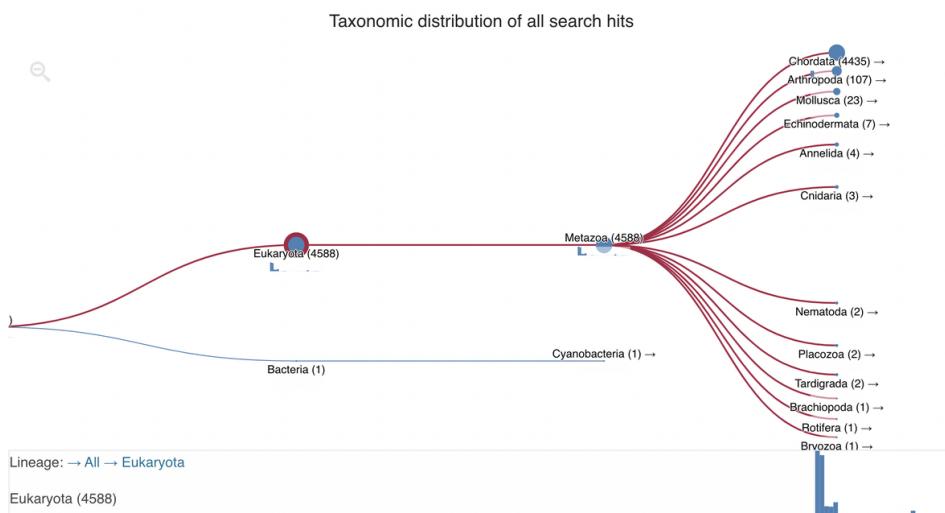
Customise Results

The screenshot shows the 'Customise Results' interface. On the left, under 'Select Visible Columns', checkboxes are checked for 'Row Count', 'Secondary Accessions and IDs', 'Description', 'Species', 'Cross-references', 'Kingdom', and 'Phylum'. On the right, under 'Rows Per Page', the '100' option is selected. Buttons for 'Update' and 'Restore Defaults' are at the bottom.

(6) 点击序列可以看到详细的对比情况

5.3 Hmmer 的网站使用实例：以 jackhmmer 为例

(7) 点击 taxonomy 可以得到分类图，其中的每一个节点都可以当作筛选的标准



(8) 点击一轮迭代的右下角的“next iteration”，进行下一轮的迭代

Iteration 1

[Return to the Results Summary](#) | [Jump to threshold](#)

Sequence selection ?

above threshold ?

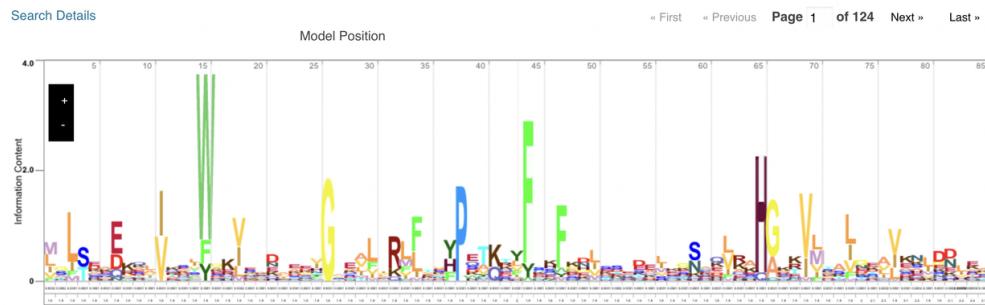
unselect all ?

Rerun iteration 2 Warning, this will replace iteration 2 and delete iterations that rely on the data produced by it.

[next iteration »](#)

(9) 得到 E-value 极低的比对结果，此外还能得到 Model Position 图

Significant Query Matches (10846) in uniprotrefprot (v.2021_04)								Customise
Row	Target	Secondary Accessions & IDs	Description	Kingdom	Phylum	Species	Cross-references	E-value
> 1	A0A7M7G143_STRPU	A0A7M7G143	Uncharacterized protein	Eukaryota	Echinodermata	Strongylocentrotus purpuratus		0.0e+00
> 2	A0A7M7LT58_STRPU	A0A7M7LT58	Uncharacterized protein	Eukaryota	Echinodermata	Strongylocentrotus purpuratus		0.0e+00
> 3	A0A7M7HNK1_STRPU	A0A7M7HNK1	Uncharacterized protein	Eukaryota	Echinodermata	Strongylocentrotus purpuratus		0.0e+00
> 4	A0A7M7HID4_STRPU	A0A7M7HID4	Uncharacterized protein	Eukaryota	Echinodermata	Strongylocentrotus purpuratus		0.0e+00
> 5	A0A7M7HLE9_STRPU	A0A7M7HLE9	Uncharacterized protein	Eukaryota	Echinodermata	Strongylocentrotus purpuratus		0.0e+00



5.4 在 python 上使用 HMMER

将下列代码引入 python 中，并下载 HMMER 对应的数据库

```
import pyhmmer

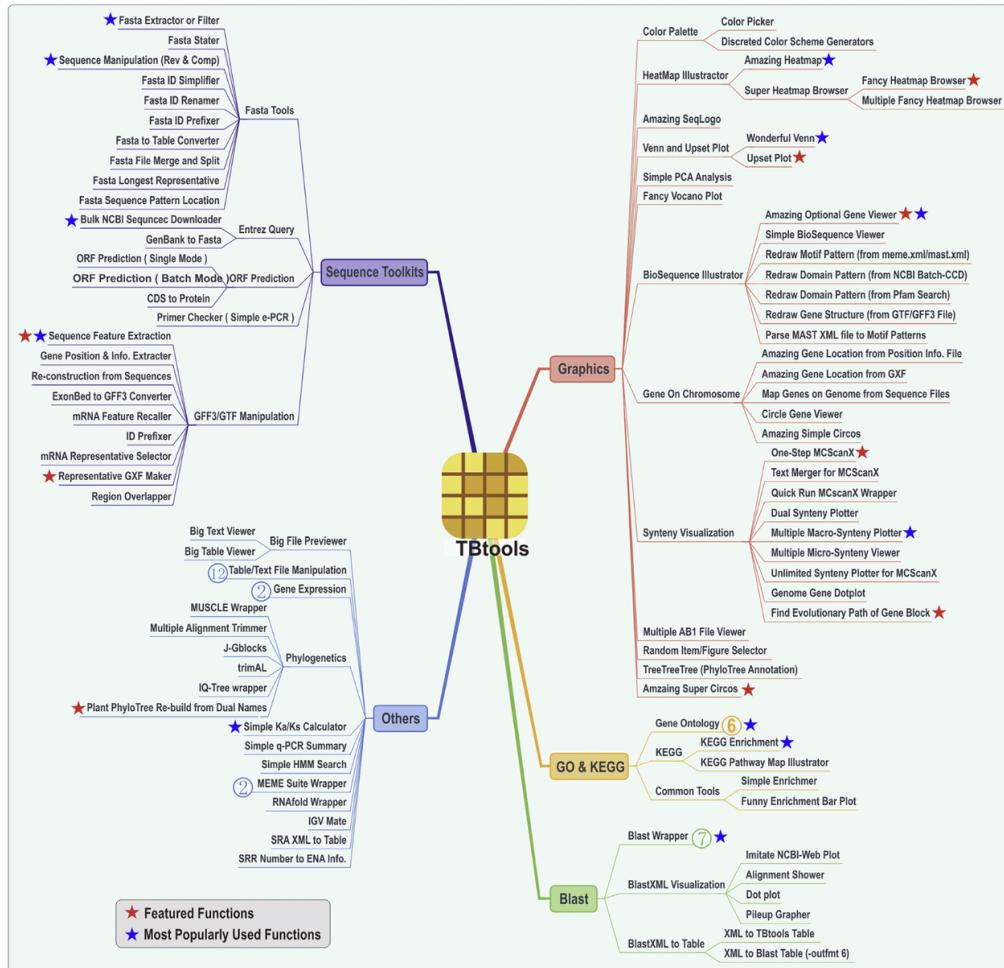
with pyhmmer.easel.SequenceFile("pyhmmer/tests/data/seqs/938293.PRJEB85.HG003687.faa",
                               digital=True) as seq_file:
    sequences = list(seq_file)
with pyhmmer.plan7.HMMFile("pyhmmer/tests/data/hmms/txt/t2pk.hmm") as hmm_file:
    for hits in pyhmmer.hmmsearch(hmm_file, sequences, cpus=4):
        print(f"HMM {hits.query_name.decode()} found {len(hits)} hits in the target sequences")
```

第六章 TBtools

6.1 TBtools 简介

(a Toolkit for Biologists integrating various biological data-handling tools), 是一款集成了 blast、KEGG 等模块的生信分析软件（软件包主页 <https://github.com/CJ-Chen/TBtools/releases>）。

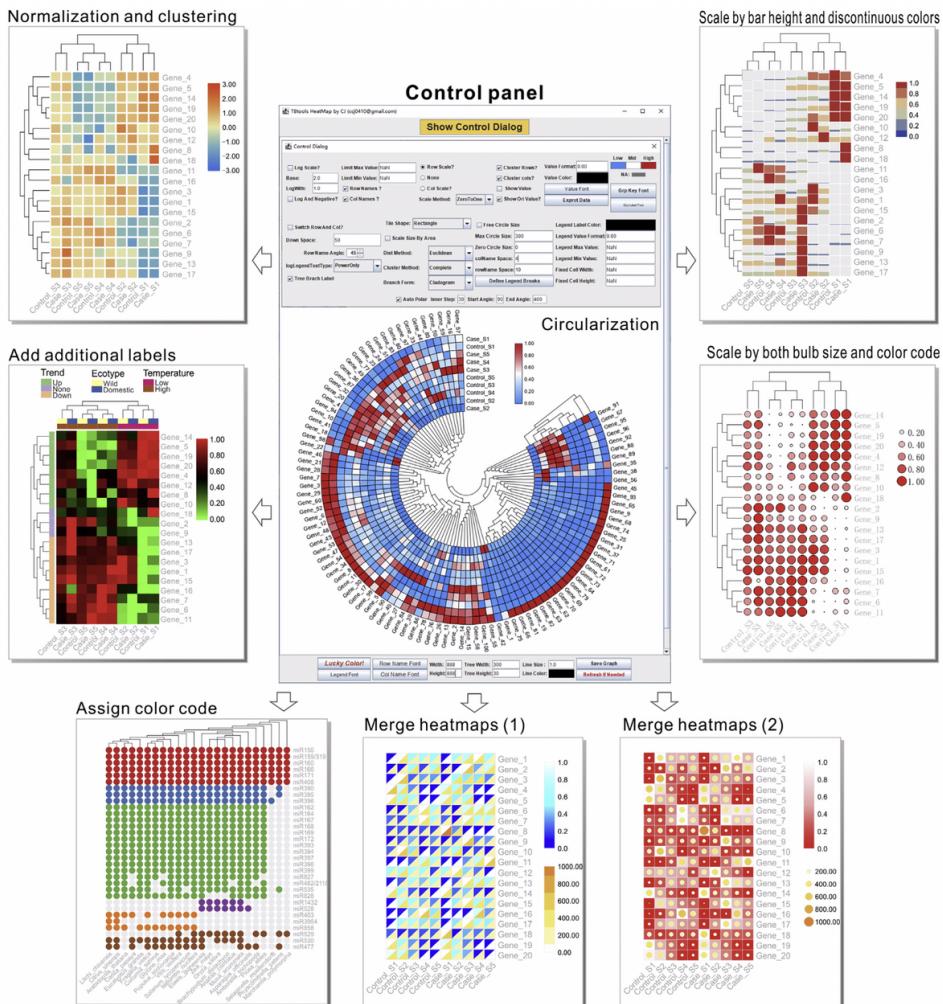
TBtools 有共 130 余种不同功能，是一款强大的生信分析-可视化软件。



功能总览, 图源: <https://www.yuque.com/cjchen/hirv8i>

TBtools 是为交互式数据表示而开发的，而数据可视化和表示是生物信息分析不可或缺的部分。与通常生成不可编辑图形的常规图形生成器不同，TBtools 生成充满可编辑特征的交互图形。TBtools 中集成了一个名为“JIGplot”（Java 交互式图形）的新开发的绘图引擎，可以快速修改各种图形功能。

例如：可以在 control panel 上快速的调节绘画时的各种参数，修改生成的可视化图片，为用户提供了很大的方便性。



本小节中，只对 TBtools 的功能进行简要介绍，不包含详细的案例分析。

6.2 软件功能实例

6.2.1 ORF 预测

6.2.1.1 背景

预测 ORF(Open Reading Frame) 时，从 mRNA 角度（起始密码子，canonical 或 non-canonical 等），蛋白质角度（蛋白产物稳定性、是否具有功能等）进行预测。

6.2.1.2 常用的 ORF 预测软件——OrfM

以往寻找终止密码子的方式：将原始序列切成 6 种不同的 frame，逐个扫描获得终止密码子。

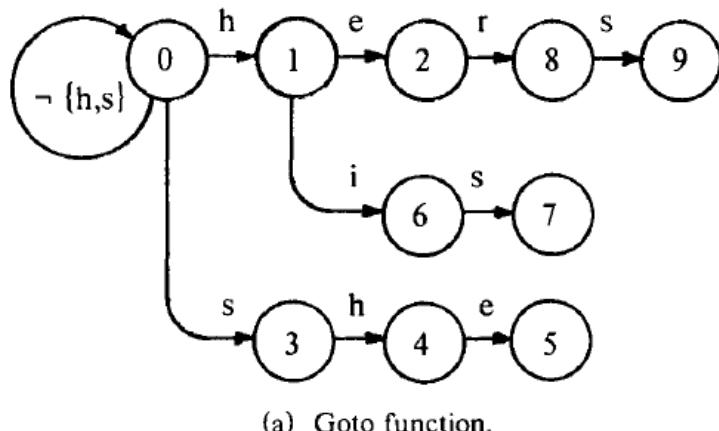
OrfM(*fast open reading frame predictor for metagenomic data*) 直接在原始的核苷酸序列种寻找终止密码子，使用 Aho-Corasick 算法

Aho-Corasick：

A. 根据已有的数据建立有限的目标 pattern 和模式匹配机 B. 使用模式匹配机对文本字符串进行单遍处理。

构建模式匹配机所需的时间与 pattern 长度的总和成正比。

模式匹配机在处理文本字符串时进行状态转换的次数与关键字的数量无关。

Fig. 1. Pattern matching machine.

(a) Goto function.

i	1	2	3	4	5	6	7	8	9
$f(i)$	0	0	0	1	2	0	3	0	3

(b) Failure function.

i	$output(i)$
2	{he}
5	{she, he}
7	{his}
9	{hers}

(c) Output function.

6.2.1.3 TBtools 中的 ORF prediction

可进行单条序列中 ORF 的预测

批量序列中 ORF 的预测

批量 CDS 对蛋白质的转化（是否考虑密码子偏好性）

6.2.2 基因富集分析

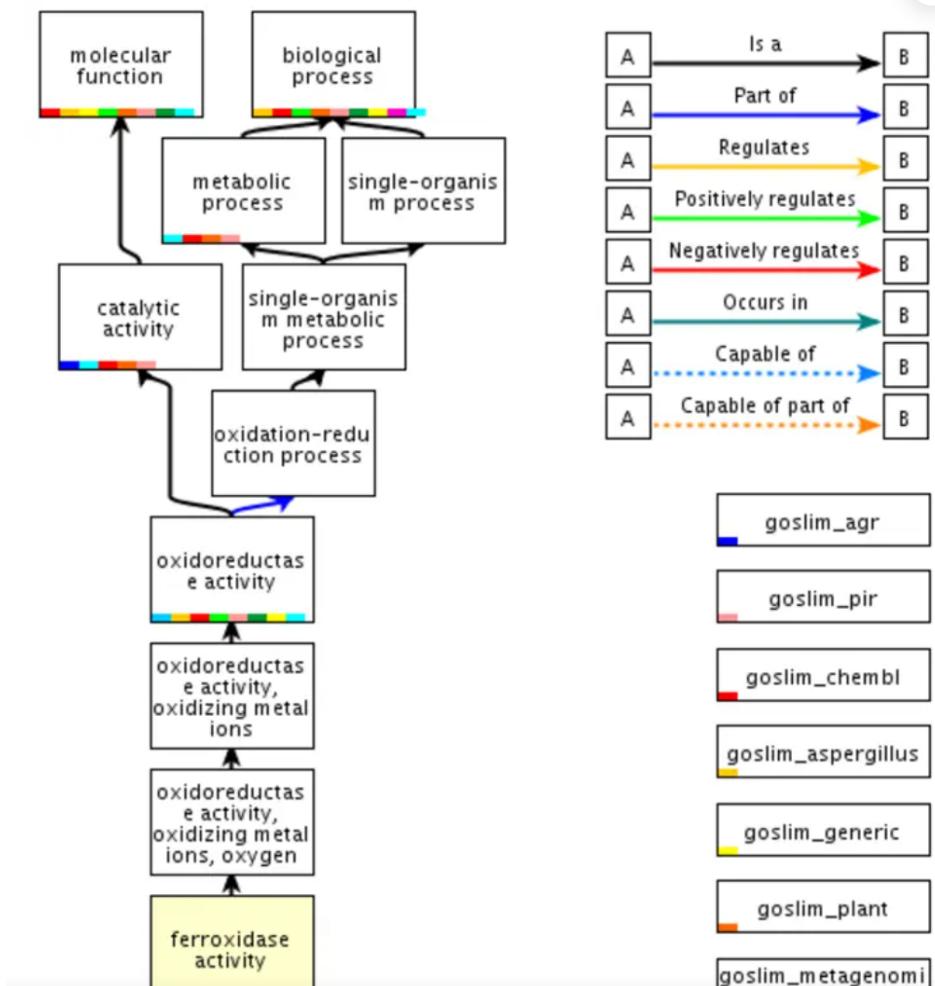
6.2.2.1 背景

GO (gene ontology, 基因本体论) 的主要目的是归类，统一生物学方言（不同的生物学数据库可能会使用不同的术语）。

GO 是一个有向无环图(DAG)本体，主要形式是 term 标记，每个 GO term 代表一种功能描述，都属于 ontology。

GO 总共分成三个 ontology:molecular function(MF), cellular component(CC) 和 biological process(BP)。

不同 GO term 之间存在多种关系，常见的主要是 is_a 和 part_of 和两种关系。



6.2.2.2 基本原理介绍

如已有一基因集合（如差异表达基因集合或 ChIP-seq 的 Peaks 或 GWAS 定位的系列区间），及有一个功能标签（例如如生长素信号转导相关）。

假定某物种共有 100 个基因，其中 20 个基因与生长素信号转导相关，80 个基因未注释到与生长素信号转导相关（在该注释库下被认为无关）。对植株进行处理，通过差异表达分析鉴定到 10 个差异表达基因，其中 2 个与生长素信号转导相关，而另外 8 个则没注释到生长素信号转导相关。

结果如下：

	基因全集	差异表达
生长素响应相关	20	2
未注释到生长素响应相关	80	8

检测结果中用于差异分析的基因集合中与生长素相应相关的基因比例与基因全集中与生长素相应相关的基因比例（背景比例相同），说明没有富集。

注意：

1. 区别“富集”和“富集显著”：上述按理，若实验组基因集合中具有感兴趣标签的基因的比例超过背景比例，那么这种情况类比上，就是“富集”，因为偏离了背景。但是通过检验，如果偏离程度不大，则不能排除这是一种随机波动；而如果显著偏离了背景分布，就是“富集显著”。

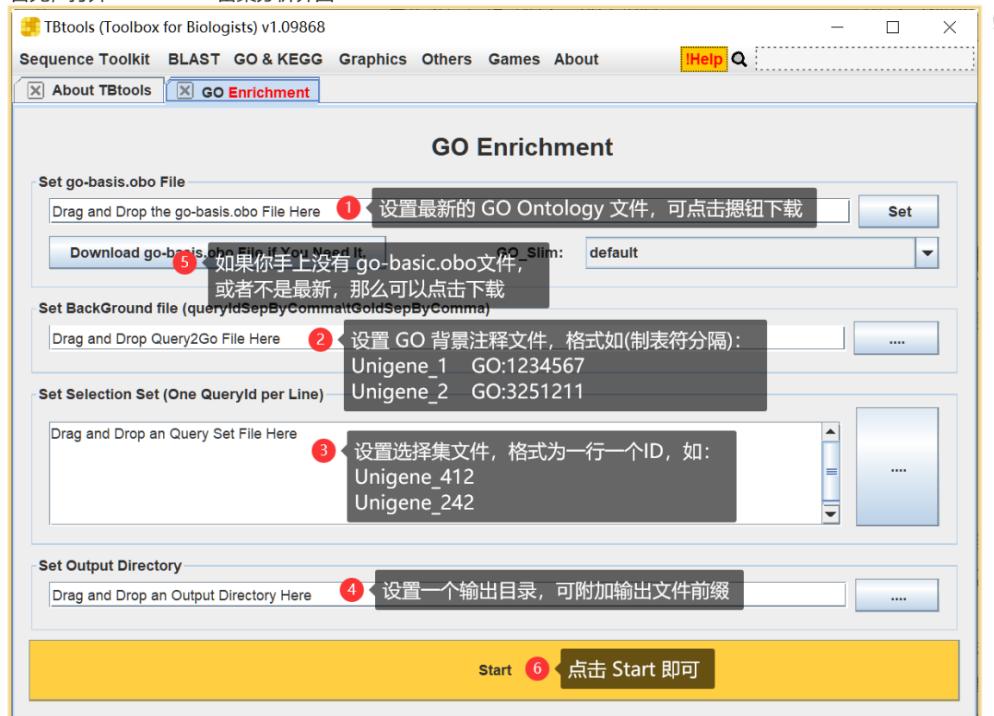
2. 富集分析时，很多新接触的，搞错的往往就是没搞清楚原理，背景（基因全集）和实验组基因集合（基因选择集合）（如差异表达基因集合）。一定要注意，做基因功能富集分析是，背景注释指的是这个物种所有基因的功能注释信息而不是选择集的基因功能注释。比如，做拟南芥的，大概有2w+个基因的功能注释，拿这个做背景；而不是拿差异表达的几百上千个基因的注释做背景。

6.2.2.3 TBtools 实现 GO 富集分析

需要准备三个文件：

- go-basic.obo 文件，可以从 <http://purl.obolibrary.org/obo/go-basic.obo> 下载
 - 一个物种所有基因的 GO 注释文件
 - 一个基因选择集合，如实验组基因集合，如差异基因集合，或 GWAS 筛选出来的基因集合等
- 具体实现过程：

首先，打开 TBtools GO 富集分析界面

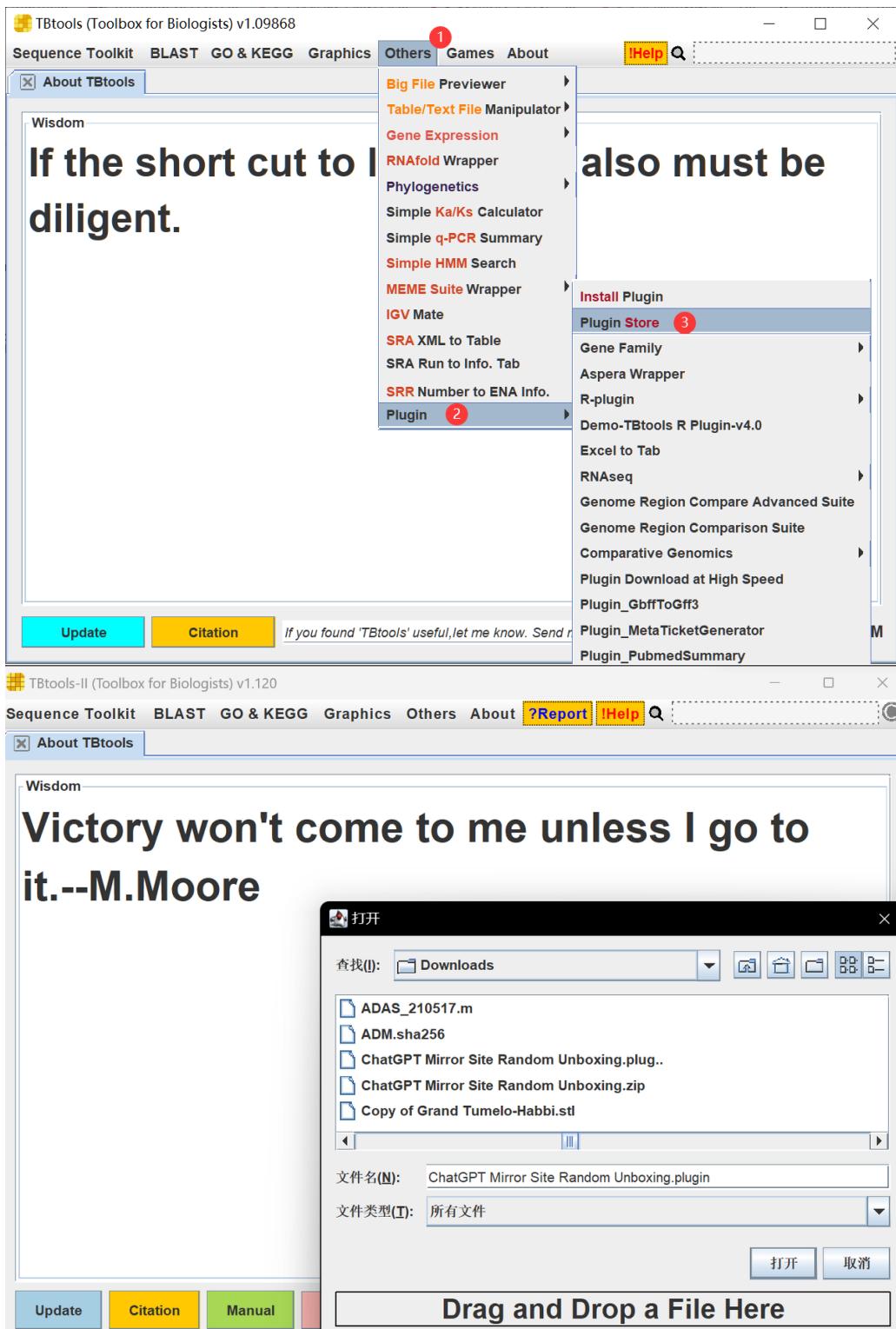


6.2.3 其他常见功能

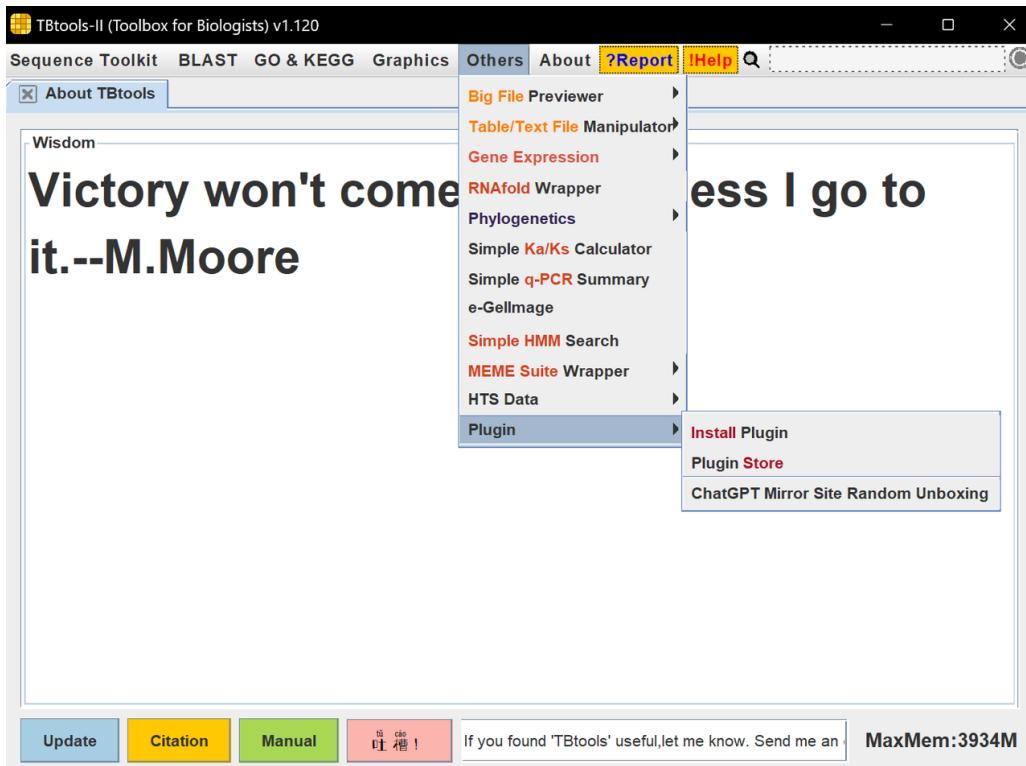
- eFP Browser 能够提供一个很简单便捷的方法用来可视化基因在宏观结构上的表达情况
 - interactive heatmap：提供基因互作关系的程度关系
 - simple circos：一种常见的基因比较分析可视化方法
-

6.2.4 TBtools 插件安装

TBtools 的插件模式允许用户在 TBtools 核心功能外按照自己需要安装特定插件，从而使用对应功能。这些插件可以直接在 TBtools 的插件商店中找到。



安装完成后重启软件即可看见插件。



参考资料

- <https://www.yuque.com/cjchen/hirv8i>
- <https://zhuanlan.zhihu.com/p/144873860>
- https://www.youtube.com/watch?v=O7_w001f58c

第七章 ChIP-Seq 项目实操

本章将介绍 ChIP-Seq 的基本分析流程，包括上游分析和下游分析。

其中上游分析流程主要包括：

1. fastq 文件的质控、过滤、比对
2. 比对文件的排序、过滤、去重、索引
3. 查看 ChIP-Seq 分析的质量

下游分析主要包括：

1. call motif 分析
2. 与 TSS 的共定位分析

本章涉及的软件较多，不能一一介绍其算法原理和详细的使用方法，因此对该项目代码中具体参数的调整需要针对不同的项目自行斟酌。但该项目代码依然具备较高的参考价值，可作为某些生信分析项目的通用分析模板。感谢饶希晨师兄提供的大部分代码，我们在此基础上完成了后续代码的修改，并成功运行。

感谢李帅师兄的总结报告，为了该总结的完整性，我们在本章部分引用了李帅师兄的总结报告中的部分内容，如果对湿实验部分感兴趣，还请参考李帅的《生物信息技术期末总结》。

7.1 背景介绍

在此简要介绍该 ChIP-Seq 项目中使用到的一些基础概念。

7.1.1 ChIP-Seq 原理

染色质免疫共沉淀 (Chromatin Immunoprecipitation) 实验主要用于目标蛋白会与基因组 DNA 中那些序列结合。其实验原理为，利用甲醛交联固定后，打断染色质 DNA，与目标蛋白结合的 DNA 会与目标蛋白一起被目标蛋白的抗体免疫沉淀。染色质免疫共沉淀测序技术 (ChIP-Seq) 则是将上述免疫沉淀出的 DNA 接上测序接头，进行二代测序。在此不再赘述湿实验的具体流程，本章主要聚焦该实验所得数据的生物信息学分析。ChIP-Seq 的基本分析流程也可拓展应用到其他任何与寻找目标核酸序列在基因组中的位置相关的生信分析中。

7.1.2 CTCF 简介

本章节使用的测序数据为 CTCF 蛋白的 ChIP-Seq 数据。

CCCTC-结合因子 (CCCTC-binding factor, CTCF) 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 可以与染色质结构域边界结合的同时招募其他转录因子，并帮助真核基因组构建三维结构。

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。目前 CTCF 与 CBS 复合物的晶体结构已被解出。

7.1.3 ChIP-Seq 分析前的准备工作

本项目的上游分析流程是许多生信分析共有的一般分析流程，包括质量控制、去接头、比对；比对文件的排序、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、富集峰的寻找。最后根据研究项目的目的可进行进一步下游分析，本项目进行了初步的下游分析，包括富集峰与转录起始位点 (TSS) 的共定位、峰基序分析 (call motif)

7.1.3.1 环境配置

```

conda create -n chip-seq
conda activate chip-Seq
conda install fastqc
conda install cutadapt
conda install bowtie2
conda install samtools ## 操作比对结果文件的软件
conda install macs2 ## 用于寻峰的软件
conda install -c bioconda deeptools
conda install -c idr ## 用于多样本合并的软件

# meme套件和homer软件需要perl环境，建议编译安装perl环境后再编译安装上述软件
# 下载meme套件安装包
wget https://meme-suite.org/meme/meme-software/5.4.1/meme-5.4.1.tar.gz
tar zxf meme-5.4.1.tar.gz
cd meme-5.4.1
./configure --prefix=$HOME/meme --enable-build-libxml2 --enable-build-libxslt \par
make
make test
make install

wget http://homer.ucsd.edu/homer/configureHomer.pl
perl configureHomer.pl -install
Homer\_HOME=/Users/zhaohuanan/rxc/homer
echo PATH="${Homer\_HOME}/bin:$PATH"
chmod +x configureHomer.pl
./configureHomer.pl -install hg38

# piacrd需要Java环境，建议安装openjdk
sudo apt update
sudo apt install openjdk-17-jdk
# 下载picard包，并添加至环境变量，http://broadinstitute.github.io/picard/
java -jar /path/to/picard.jar -h
java -jar $PICARD -h
http://broadinstitute.github.io/picard/

```

7.1.3.2 测序数据和参考文件的下载

本项目使用的公共数据来自数据库 ENCODE(DNA 元件百科全书, Encyclopedia of DNA Elements)

项目数据的登录号为 ENCSR617IFZ。实验数据总共有四个文件，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。

其中生物学重复 1 测序数据注册号为 ENCFF631JSV, ENCFF715KYL; 生物学重复 2 的数据注册号为 ENCFF002OWA, ENCFF833BQT。

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.

Raw sequencing data (4 Files)									
			File type	Run type	Read	Output type	Lab	Date added	File size
1	ENCLB048DBS	ENCFF631JS	fastq	PE10int	1	reads	Michael Snyder, Stanford	2016-04-01	3.37 GB
		ENCFF715KY	fastq	PE10int	2	reads	Michael Snyder, Stanford	2016-04-01	3.43 GB
2	ENCLB541OLX	ENCFF002OWA	fastq	PE10int	1	reads	Michael Snyder, Stanford	2016-04-01	3.2 GB
		ENCFF833BQI	fastq	PE10int	2	reads	Michael Snyder, Stanford	2016-04-01	3.26 GB

7.2 上游分析流程

ChIP-Seq 的一般分析流程 ChIP-Seq 测序数据需要完成基本的 NGS 分析流程。该项目中，实验数据有两个生物学重复；此外为了 call peak，另外需要一个未进行富集操作的测序数据作为参考。此三份数据采取完全相同的一般分析流程。

以本分析项目为例，作者希望建立一套尽可能自动化，规范化的流程。

7.2.1 准备工作

在分析开始之前，设置好自己的储存分配逻辑

```
#!/bin/bash

#####
# code for chipseq
#####

##### 当前工作路径为 #####
### 建议新建项目文件夹（此处为practice），并将该文件夹设置为项目的工作路径
# mkdir practice
# cd practice
# pwd
# /home/wuhangrui/practice/

##### step0 创建原始数据与参考文件的软连接 #####
### 建议使用专门的存储空间或存储服务器存储原始数据和参考文件，并建议将其软链接到自己的工作路径下
## 在该项目的例子中，本章作者将下载的rawdata存放于nas中/nas\_data/whr/practice/chipseq/rawdata,
## 由于存储空间问题，本章作者在NAS中创建了存储中间结果的文件夹，并将该文件夹也软连接到工作目录下，以将
## 中间结果文件也保存在NAS中
ln -s /nas\_data/whr/practice/chipseq/ /home/wuhangrui/practice/chipseq
cd ./chipseq

# 创建存储中间结果的文件夹
mkdir 01\_fastqc 02\_cutadapt 03\_mapping 05\_peak\_calling

# 读取样本名字
cd ./rawdata
```

```
# 把样本名字信息存放在config文件
ls *_r1.fastq.gz > config

# 去掉后缀，仅保留名字信息
sed -i "s/_r1.fastq.gz//g" config

cd /home/wuhangrui/practice/chipseq
```

7.2.2 质量控制

```
##### step1 fastqc 质量控制 #####
## 从nas里直接读取数据进行fastqc，速度有点慢，建议在同一局域网下进行

# 指示开始运行该软件的时间
echo "fastq started at $(date)"

# 对所有实验样本以及对照样本进行质控
cat rawdata/config | while read id
do
fastqc -t 10 -o 01\_fastqc/ rawdata/${id}\_r1.fastq.gz rawdata/${id}\_r2.fastq.gz > 01\_fastqc/
    fastqc.log 2>&1
done

# 指示结束运行该软件的时间
echo "fastq finished at $(date)"

### 软件参数说明
### fastqc [options] fastq1 [fastq2 ...]
### -t 指定使用的线程数
### -o 指定输出文件夹
```

7.2.3 数据清晰和筛选

```
##### step2 cutadapt 数据清洗和筛选 #####
# 指示去除接头操作的开始时间
echo "cutadapt started at $(date)"

# 对所有实验组样本和对照组进行去接头和除去低质量读段的操作
cat rawdata/config | while read id
do
cutadapt -j 10 \
--time 1 -e 0.1 -O 3 --quality-cutoff 25 -m 55 \
-a AGATCGGAAGAGCACACGTCTGAACCCAGTCA -A AGATCGGAAGAGCGTCGTAGGGAAAGAGTGT \
-o 02\_cutadapt/fixed\_${id}\_read1.fastq.gz -p 02\_cutadapt/fixed\_${id}\_read2.fastq.gz rawdata
    /\${id}\_r1.fastq.gz rawdata/\${id}\_r2.fastq.gz > 02\_cutadapt/cutadapt.log 2>&1
done
```

```

# 指示去接头操作的结束时间
echo "cutadapt finished at $(date)"

### 程序参数说明
### -j 2 #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quality的碱基去除
### -m 55 # 去除接头之后低于该值的一对reads丢弃，一般要大于40
### -a AGATCGGAAGAGCACACGTCTGAAGTCAGTCA # read1 3'的adaptor,请根据您使用的试剂盒进行接头序列的调整
### -A AGATCGGAAGAGCGTCGTAGGGAAAGACTGT # read2 3'的adaptor, 请根据您使用的试剂盒进行接头序列的调整
### -o fix.fastq/test\_R1\_cutadapt.temp.fq.gz read1的输出路径和输出文件名
### -p fix.fastq/test\_R2\_cutadapt.temp.fq.gz read2的输出路径和输出文件名
### Read1的输入文件 (这里没有形参指示)
### Read2的输入文件 (这里没有形参指示)

```

7.2.4 比对

```

##### step3 Mapping 比对到参考基因组 #####
### build index 在初次进行比对时，需要建立比对所使用的index，比对软件将基于建立的index进行比对
## 建立目录专门存放各种比对软件的index
# cd /home/wuhangrui/database/ref/hg38
# mkdir bowtie\_index\_gencode
# cd bowtie\_index\_gencode

# bowtie2-build -f ../../GRCh38.p13.genome.fa hg38\_human\_index > build\_index.log 2>&1

### bowtie2-build [options]* <reference\_in> <bt2\_base>
### 第二个参数[hg38\_human\_index]为index前缀，在比对时-x参数需要输入到该前缀
### -f 基因组为fasta格式

# 重新前往工作目录，进行比对
cd /home/wuhangrui/practice/chipseq

## mapping
# 指示比对操作的开始时间
echo "bowtie2 mapping started at $(date)"

# 对每条read依次进行比对
cat rawdata/config | while read id
do
bowtie2 -x /home/wuhangrui/database/ref/hg38/bowtie\_index\_gencode/hg38\_human\_index \
-1 ./02\_cutadapt/fixed\_${id}\_read1.fastq.gz \
-2 ./02\_cutadapt/fixed\_${id}\_read2.fastq.gz \
-p 10 \
-S ./03\_mapping/\${id}.sam > ./03\_mapping/bowtie2\_mapping.log 2>&1
done

```

```

# 指示比对操作结束时间
echo "bowtie2 mapping finished at $(date)"

# bowtie2参数说明
### -x index file
### -1 read1 fastq file
### -2 read2 fastq file
### -p threads
### -S out SAM file
### --very-sensitive-local SNP比对时需要的参数

##### step4 比对结果的排序、过滤、去重、索引#####
## 将SAM文件转换为BAM文件，并排序
# 按染色体坐标序列排序，这是对BAM文件建立索引前的必要步骤
cat rawdata/config | while read id
do
samtools sort -O BAM -o 03\_mapping/${id}.bam -@ 10 -m 10G 03\_mapping/${id}.sam
done

# 参数说明
### -O 指定输出文件的格式
### -o 指定输出文件
### -@ 线程数
### -m 为每个线程分配的内存大小
### 03\_mapping/${id}.sam 未排序的输入文件

## BAM 文件过滤
cat rawdata/config | while read id
do
samtools view -bS -q 30 -@ 10 -m 10G -f 1 -f 2 -o 03\_mapping/${id}\_filtered.bam 03\_mapping/${id}.bam
done

# 参数说明
### -bS 输出BAM文件，并忽略与以前的samtools版本的兼容性。
### -q 跳过MAPQ小于该值的比对（MAPQ值位于SAM文件的第五列，描述比对的质量）
### -@ 线程数
### -m 为每个线程分配的内存大小
### -o 指定输出文件
### 03\_mapping/${id}.bam 未过滤的输入文件

## 去重，去除PCR重复
echo "deduplicate started at $(date)"
cat rawdata/config | while read id
do

```

```

picard MarkDuplicates -REMOVE\_DUPLICATES true -I ./03\_mapping/${id}\_filtered.bam \
-O ./03\_mapping/${id}\_filtered\_dedup.bam -M deduplication.txt > picard.log 2>&1
done
echo "deduplicate finished at $(date)"

# 参数说明
### MarkDuplicates 指定使用的Picard工具
### -REMOVE\_DUPLICATES=true 要求去除重复序列，如不指定则只会将重复序列标记出来
### -I 指定输入文件
### -O 指定输出文件
### -M 重复序列的一些重要统计信息

## 索引
# 比对软件产生的序列通常是随机的。然而，比对后的分析步骤通常要求sam/bam文件被进一步处理，例如在IGV查看
# 比对结果时，常需要输入的bam文件已经被index
echo "index bam files started at $(date)"
cat rawdata/config | while read id
do
samtools index -@ 10 ./03\_mapping/${id}\_filtered\_dedup.bam
done
echo "index bam files finished at $(date)"

# 参数说明
### 与samtools view类似，不再赘述

```

至此我们完成了基本的 ChIP-Seq 上游分析得到的基本文件结构如下图所示：

上述目录中有许多中间文件不会在下游分析中使用，因此可以直接删去。

7.2.5 ChIP-Seq 分析

7.2.5.1 ChIP-Seq 质量控制

在进行进一步的下游分析前，我们需要考察本次 ChIP-Seq 实验是否达到效果，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。这一步骤实际并不依赖于 call peak 的结果这里使用 plotFingerprint 程序（deeptools 包）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，在一个指定长度的窗口中 (bin) 中，所有重叠的 reads 都被计数，所有计数按大小排序，对累计和绘图。该方法被称为累计富集方法，也称为 BAM 指纹

```

##### step4 chipseq qc #####
## 对chipseq进行质量控制
cd /home/wuhangrui/practice/chipseq

mkdir 04\_chipseq\_qc
cd 04\_chipseq\_qc

echo "chip-seq quality control started at $(date)"

# 对实验组数据进行质量控制
cat ../rawdata/config2 | while read id

```

```

do
plotFingerprint -b ../../_mapping/${id}_filtered_dedup.bam ../../_mapping/ctcf_chip_input_
    _filtered_dedup.bam \
--labels rep1 rep2 input --plotFileFormat svg --plotTitle CTCF_Enrichment -o CTCF_Enrichment.svg
    -p 10
done

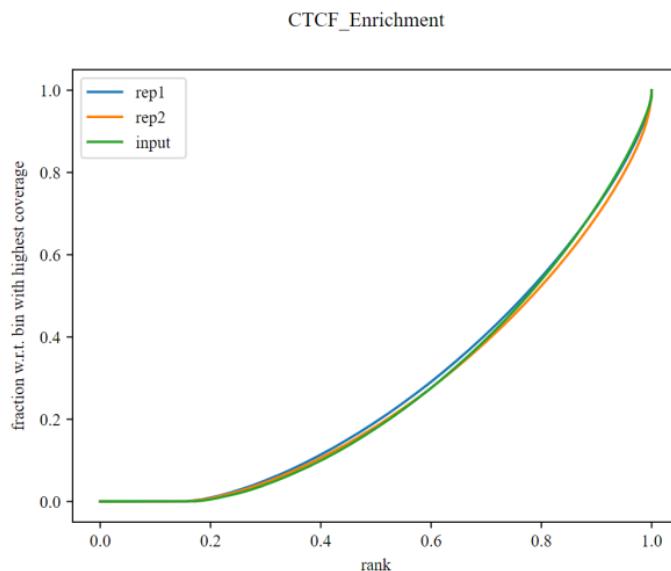
# 参数说明
### -b 输入BAM文件列表，也可以是一个或多个BAM文件
### --labels 输出图像中使用的标签
### --plotFileFormat 输出文件格式
### --plotTitle 输出文件标题
### -o 指定输出文件
### -p 使用的线程数

cd /home/wuhangrui/practice/chipseq

echo "chip-seq quality control finished at $(date)"

```

得到的结果如下图所示



理论上，对于没有富集的对照组，其累计曲线应接近与对角线，这与我们的结果符合。对于转录因子的强富集，理想的情况下曲线应该会先缓慢上升，到 rank 较高的区域后再急剧上升，这表明大量的 reads 分布在较少的特定窗口中，即蛋白质的结合位点存在显著富集。

然而本实验中实验组和对照的累计曲线却相当接近，我们并没有进一步向下探索造成这种现象的原因。

7.2.5.2 寻峰

CTCF 是一种转录因子，针对 CTCF 的 ChIP-Seq 能够将与之有相互作用的 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。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。

本流程使用 MACS (基于模型的 ChIP-Seq 分析, Model-based analysis of ChIP-Seq) 进行 call peak 分析。该软件假设 DNA 上的片段在随机状态下服从泊松分布, macs2 根据不满足泊松分布的地方寻找富集点。

```
##### step5 call peaks #####
echo "call peaks started at $(date)"

## 将实验组数据提取出来
cp rawdata/config rawdata/config2
cat rawdata/config2 | sed "/input/d" > rawdata/config2

## 寻峰, 对实验组的所有已经过滤、去重、索引的bam文件寻峰
cat rawdata/config2 | while read id
do
macs2 callpeak -t ./03\_mapping/${id}\_filtered\_dedup.bam \
-c ./03\_mapping/ctcf\_chip\_input\_filtered\_dedup.bam \
-f BAMPE -g hs -n CTCF\_ChIP -B -q 0.01 \
--outdir ./05\_peak\_calling > ./05\_peak\_calling/peak\_calling.log 2>&1
done
echo "call peaks finished at $(date)"

### 参数说明
### -t 处理组的比对文件
### -c 对照组的比对文件
### -f 输入文件的格式, 设为auto则表示自动识别输入文件的格式; 双端测序用BAMPE
### -g 有效基因组大小, hs表示人的有效基因组大小, 约为2.7E9
### -n name string, 会成为输出文件的prefix
### --outdir 输出文件路径
### -B/--BGD 保存堆积对齐片段到bedGraph文件里
### -q FDR Cutoff

## 备注
# broad peak常用于组蛋白修饰的peak calling, 因为组蛋白修饰往往是一大片; 用另外一个程序来call peak
# narrow peak用于转录因子的peak calling, 因此本课题应该使用narrow peak calling。
# narrowPeak 也可以放进igv里
```

macs2 的输出结果如下:

NAME_peaks.xls 保存了每个 peak 的信息:

1. chromosome name: 染色体名
2. start position of peak: peak 开始的位置

3. end position of peak: peak 结束的位置
4. length of peak region: peak 区域的长度
5. absolute peak summit position: peak 峰出现的位置
6. pileup height at peak summit: peak 峰的堆积高度
7. -log10(pvalue) for the peak summit (e.g. pvalue = 1e-10, then this value should be 10)
8. fold enrichment for this peak summit against random Poisson distribution with local lambda: 相对于随机选取的片段, peak 峰高度的变化倍数, 即富集倍数
9. -log10(qvalue) at peak summit

NAME_peaks.narrowPeak 也同样储存了 peak 的信息:

1. 1: 染色体号
2. 2: peak 起始位点
3. 3: 结束位点
4. 4: peak name
5. 5: int(-10*log10qvalue)
6. 6: 正负链 (如果不区分, 则记为":")
7. 7: fold change
8. 8: -log10pvalue
9. 9: -log10qvalue

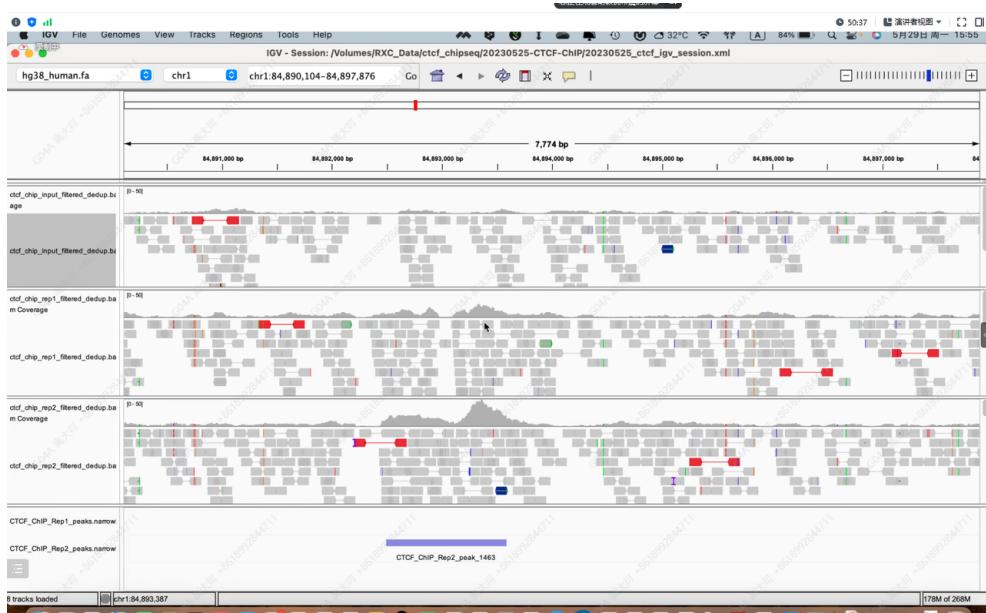
10. 10: peak 的峰相对于 peak 的位置

NAME_summits.bed 文件记录了所有 peak 峰的位置和可信度, 下游可进一步利用该文件寻找 motif

NAME_treat_pileup.bdg 文件用于存放区间的坐标轴信息和相关的评分文件, 用于储存稀疏, 不连续的数据, 并可转化为 bigwig 文件 (二进制, 有索引, 更高效)

NAME_control_lambda.bdg 文件记录了由对照组中估计得到的局部偏好, 可转化为 bigwig 文件。

下载 igv 软件后, 可将排序并索引后的 bam 文件与 narrowPeak 文件导入 IGV 进行可视化查看。



从上图中我们可以可视化地看见 narrowpeak 的位置, 以及那些染色体位置上存在峰的富集。从该图中还能看见部分 read 太长, 只测得了两端 (这种情况下 macs2 会用-f BAMPE 进行矫正), 部分 read 太短, 二代测序将其测穿, 部分位点存在 PCR 错配或 SNP。

如果仔细对比 rep1 和 rep2 的 bam 文件在 IGV 上的可视化图像, 可以发现, 有的 reads 的富集在 rep2 上明显比 rep1 更多, 这可能是因为建库时 rep2 的富集效果更好, 或 rep1 中某些 reads 的质量不满足被清洗掉了, 或者

没有通过假设检验，可以尝试将 q 值从 0.01 提高到 0.05。

7.2.5.3 生物学重复的合并

在进行下游分析时，可以将生物学重复样本合并后进行分析。可以使用 IDR 软件评估生物学重复样本的相关性，并根据阈值筛选出最终的一组 peak，即该软件可从多个生物学重复样本的 peak 结果中提取高一致性 peak 区间。

使用 idr 前首先需要对 MACS2 的结果文件 narrowPeak 按照 p-value 排序；排序完成后进行合并。

```
cat rawdata/config2 | while read id
do
sort -k8,8nr ./05\_peak\_calling/${id}\_peak.narrowPeak > ./05\_peak\_calling/${id}\_sorted\_peaks
    .narrowPeak
done

### -k8,8 选项使输入文件按其第八列进行排序
### -nr 选项指定按整个数字排序（而非一个字符一个字符去比较）

idr --samples ctcf\_rep1\_sorted\_peaks.narrowPeak ctcf\_rep2\_sorted\_peaks.narrowPeak --input-
    file-type narrowPeak --rank p.value --output-file sample-idr --plot --log-output-file sample-
    idr.log
### --samples: narrowPeak的输入文件（生物学重复样本）
### --input-file-type: 输入文件格式为narrowPeak
### --rank p.value: 指定输入文件以p.value排序的
### --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。包含了整合后结果与原本样品 1 和 2 的信息。sample-idr.png 是结果输出图，描述了两个生物学重复样本中峰的保留情况。

在项目的后续分析中没有使用合并的样本，只使用了 rep1 进行后续下游分析。

7.3 ChIP-Seq 下游分析

7.3.1 call motif

使用 meme 套件进行该任务。

可以使用 meme-chip 寻找 motif，并可以详细查看某个转录因子的信息。

```
mkdir 06\_downstream

##### step6. call motif, peak注释和motif search #####
prefix=CTCF\_ChIP # 该prefix与peak calling的prefix相同
### 首先提取用于寻找motif的fasta文件
## 取峰值前后100个碱基，第二列-50得到第六列，第三列+49得到第七列，print第1（染色体列），6，7，4列
awk -v OFS="\t" '{$6=$2-50;$7=$3+49;print $1','$6','$7','$4}' ./04\_peak\_calling/${prefix}\_
    summits.bed > ./06\_downstream/${prefix}\_motif.bed
bedtools getfasta -fi /home/wuhangrui/database/ref/hg38/GRCh38.p13.genome.fa -bed ./06\_downstream
    /${prefix}\_motif.bed > ./06\_downstream/${prefix}\_motif.fasta
```

```

cd /home/wuhangrui/practice/chipseq/06\downstream

echo "calling motif started at $(date)"

mkdir memechip\_out

meme-chip -db /home/wuhangrui/database/motif\_databases/HUMAN/HOCOMOCOv11\_full\_HUMAN\_mono\_meme
\_format.meme -meme-p 10 -dna ${prefix}\_motif.fasta -oc /home/wuhangrui/practice/chipseq/06\
downstream/memechip\_out

echo "calling motif finished at $(date)"
cd /home/wuhangrui/practice/chipseq

```

可以在输出文件夹下找到输出结果 html 文件,meme-chip 中可以看到软件找出的 motif，点击后可在 tomtom 页面看到该 motif 的详细信息。

注意 html 结果文件必须和其他 meme-chip 的输出文件在同一目录下。

MEME-ChIP
Motif Analysis of Large Nucleotide Datasets

For further information on how to interpret these results please access <https://meme-suite.org/meme/doc/meme-chip-output-format.html>.
To get a copy of the MEME software please access <https://meme-suite.org>.

If you use MEME-ChIP in your research, please cite the following paper:
Philip Machanick and Timothy L. Bailey, "MEME-ChIP: motif analysis of large DNA datasets", Bioinformatics 27(12):1696-1697, 2011. [full text]

[MOTIFS](#) | [PROGRAMS](#) | [INPUT FILES](#) | [PROGRAM INFORMATION](#) | [SUMMARY IN TSV FORMAT](#) | [MOTIFS IN MEME TEXT FORMAT](#)

MOTIFS

The significant motifs (E-value ≤ 0.05) found by the programs MEME, STREME and CentriMo; clustered by similarity and ordered by E-value.

Expand All Clusters | Collapse All Clusters

Motif Found	Discovery/Enrichment Program	E-value	Known or Similar Motifs	Distribution	SpaMo & FIMO
	MEME	4.0e-616	CTCF_HUMAN.H11MO.0.A CTCF_HUMAN.H11MO.0.A ZIC3_HUMAN.H11MO.0.B		Motif Spacing Analysis Motif Sites In GFF3
	CentriMo from STREME	2.8e-104	CTCF_HUMAN.H11MO.0.A CTCF_HUMAN.H11MO.0.A MYCN_HUMAN.H11MO.0.A		Motif Spacing Analysis Motif Sites In GFF3
	CentriMo Group	4.3e-0	CPEB1_HUMAN.H11MO.0.D		Motif Spacing Analysis

Reverse Complement ⚡ Show 10 More ⚡ CentriMo Group ↗

Tomtom
Motif Comparison Tool

For further information on how to interpret these results please access <https://meme-suite.org/meme/doc/tomtom-output-format.html>.
To get a copy of the MEME software please access <https://meme-suite.org>.

If you use Tomtom in your research, please cite the following paper:
Shubhit Gupta, JA Stamatoyanopolous, Timothy Bailey and William Stafford Noble, "Quantifying similarity between motifs", Genome Biology, 8(2):R24, 2007. [full text]

[QUERY MOTIFS](#) | [TARGET DATABASES](#) | [MATCHES](#) | [SETTINGS](#) | [PROGRAM INFORMATION](#) | [RESULTS IN TSV FORMAT](#) | [RESULTS IN XML FORMAT](#)

QUERY MOTIFS

Database	ID	Alt. ID	Preview	Matches	List
meme.xml	ACACACACACACACA	MEME-3		0	
meme.xml	SYGCCMYCTRSTG	MEME-1		6	CTCF_HUMAN.H11MO.0.A, CTCFL_HUMAN.H11MO.0.A, ZIC3_HUMAN.H11MO.0.B, ZIC2_HUMAN.H11MO.0.D, SNAI1_HUMAN.H11MO.0.C, MYC_HUMAN.H11MO.0.A
meme.xml	TTTTTTKTTTTTTT	MEME-2		15	CPEB1_HUMAN.H11MO.0.D, PRDM6_HUMAN.H11MO.0.C, FOXG1_HUMAN.H11MO.0.D, PUBP1_HUMAN.H11MO.0.D, FOXJ3_HUMAN.H11MO.0.A, ANDR_HUMAN.H11MO.0.A

Next Top

TARGET DATABASES

Database	Used	Matched
HOCOMOCOv11_full_HUMAN_mono_meme_format	769	21

Previous Next Top

MATCHES TO SYGCCMYCTRSTG (MEME-1)

Previous Next Top

7.3.1.1 peak 注释

使用 Homer 进行 Peak 的注释。Homer 包含一个用于峰注释的程序 annotatePeaks.pl。它可以将峰与其他附近的基因关联起来。

annotatePeaks.pl 在默认情况下，会寻找距离 peak 最近的 TSS，并根据 TSS 所属的基因进行注释。此外该脚本还会根据 peak 占据的位置，注释其基因组信息

```
##### step7. peakannotation #####
cd /home/wuhangrui/practice/chipseq/06\_downstream

echo "peak annotation started at $(date)"
mkdir peak\_annotation
annotatePeaks.pl /home/wuhangrui/practice/chipseq/04\_peak\_calling/${prefix}\_peaks.narrowPeak hg
    38 > peak\_annotation/peak\_annotation.tsv
echo "peak\_annotation finished at $(date)"
```

注释 peak 得到的信息是表格形式，不方便查看和进一步研究。可以将该输出文件下载到本地用，使用 R 进行可视化。例如 ChIPseeker，GO 富集分析，KEGG 富集分析等。该部分是得到的 peak 注释的进一步分析，和 linux 本身关系不大，此处省略该部分的代码。可在本组 [github](#) 仓库上寻找相关代码。

7.3.2 计算与 TSS 的共定位情况

1. TSS bed 文件可 UCSC 网站上直接下载，下载选项只需要 5' UTR 的位置
2. 需要将索引好的 BAM 文件转化为 bigwig 文件
3. 使用 deeptools 包下的软件完成该项任务

```
##### step8. 不适合看motif的共定位分析 #####
##### 8.1 与TSS, TES的共定位情况 #####
# 使用bamCoverage进行 bam -> bigwig 文件转换（一种压缩方式：将基因组压缩为很多个bin区域）
# 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
# 实验组对照组都需要此步骤

cd /home/wuhangrui/practice/chipseq/06\_downstream
mkdir co\_location

# 将input和rep1都进行bam -> bw 文件的转换

cat /home/wuhangrui/practice/chipseq/rawdata/config | while read id
do
bamCoverage --bam /home/wuhangrui/practice/chipseq/03\_mapping/${id}\_filtered\_dedup.bam \
-o ./co\_location/${id}\_filtered\_dedup.bw \
--binSize 10 \
--normalizeUsing RPGC \
--effectiveGenomeSize 2913022398 \
--extendReads \
-p 10
done

### --bam输入的bam文件
### -o 输出的bigwig文件
```

```

#### --binSize 计算coverage时的bin的大小
#### --normalizeUsing 选择normalization的方法 RPGC按照bin和测序量大小归一化
#### --effectiveGenomesize 如果需要normalization则需有设置有效基因组大小
#### --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
#### -p 使用的线程数

##### 与其他区域的共定位情况（高甲基化区域，低甲基化区域，5'UTR，3'UTR等）
# 使用“computeMatrix”计算指定位置的信号矩阵
cd /home/wuhangrui/practice/chipseq/06\_downstream

computeMatrix reference-point \
-S ./co\_location/ctcf\_chip\_input\_filtered\_dedup.bw ./co\_location/ctcf\_chip\_rep1\_filtered \
\_dedup.bw ./co\_location/ctcf\_chip\_rep2\_filtered\_dedup.bw \
-R /home/wuhangrui/database/ref/hg38/uscs\_refseq.bed \
-a 2500 -b 2500 \
--samplesLabel input ctcf\_rep1 ctcf\_rep2 \
--sortRegions descend \
-o ./co\_location/Rep1\_Input\_TSS\_Matrix.gz \
-p 10 > computeMatrix\_input.log 2>&1

# 使用reference-point模式，reference位点为TSS位置
### -S 输入的bigwig文件
### -R 记录reference位点信息的bed文件
### -a 起始位点上游的距离
### -b 起始位点下游的距离
### --sortRegions 在矩阵中，信号按什么顺序排列？
### --samplesLabel 在矩阵中，样品如何命名
### -o 输出的矩阵名
### -p 线程数
### 如果热图中出现黑色条状色块，可以尝试增加--missingDataAsZero选项

# 使用plotHeatmap作图
cd /home/wuhangrui/practice/chipseq/06\_downstream
plotHeatmap -m ./co\_location/Rep1\_Input\_TSS\_Matrix.gz \
-out ./co\_location/Rep1\_Input\_TSS\_Matrix.svg \
--plotFileFormat svg --yAxisLabel RPGC --regionsLabel all\_tss \
--legendLocation none

### -m 输入的矩阵文件
### -out 输出的图片文件
### --plotFileFormat 指定输出文件的格式
### --yAxisLabel 指定y轴标签

```

如图7.1，可见处理组在 TSS 上有非常明显的富集，且呈现双峰模式。颜色表示 reads 密度。

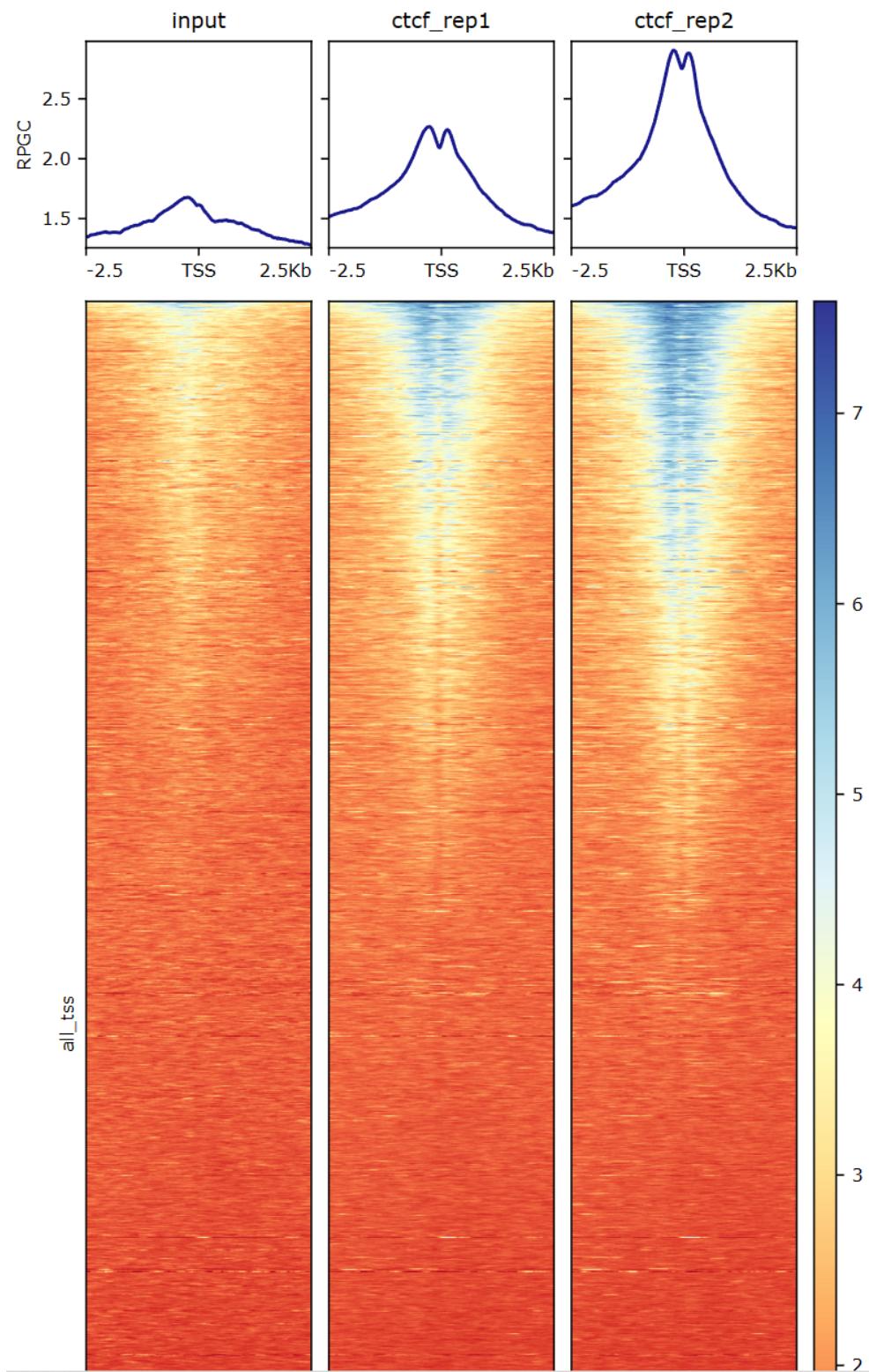


图 7.1: 热图

第八章 RNA Seq

8.1 软件的安装

主要使用到了 cutadapt、STAR、fastqc、stringtie，首先使用在 conda config 中添加 bioconda，再创建一个环境，添加会使用到的各种包。

```
conda config --add channels bioconda
conda create -n rnaseq
conda activate rnaseq
conda install cutadapt STAR fastqc stringtie
```

```
[(rnaseq) leb04b@bbt-enhance:~$ conda install cutadapt STAR fastqc stringtie
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.3.1
  latest version: 23.5.0

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

  conda install conda=23.5.0

# All requested packages already installed.
```

8.2 RNA-seq 上游分析

RNA-seq 上游分析与 ChIP-Seq 类似，参见。故此处仅简单提及。

8.2.1 质控

原始数据为 fastq.qz 文件，一般是双端配套的比对结果，有两个文件使用 fastqc 进行质控，最后得到两个文件，其中一个为 html 文件，可用网页浏览，也可以在 VSCode 中安装 html 插件查看。

8.2.2 使用 cutadapt 去接头

8.2.3 比对

下载参考基因组，star 建立索引。比对后，可以得到 bam、log、log out、log final out 文件

8.2.4 定量

使用 stringtie，输入 bam 文件，得到 gtf 文件（包含基因、数量）

使用 stringtie 的自带脚本 prepDE.py 运行得到的 gtf 文件得到表达矩阵 csv 文件

```
(rnaseq) leb04b@bbt-enhance:~$ fastqc -o . SRR5859403.fastq
null
Started analysis of SRR5859403.fastq
Approx 5% complete for SRR5859403.fastq
Approx 10% complete for SRR5859403.fastq
Approx 15% complete for SRR5859403.fastq
Approx 20% complete for SRR5859403.fastq
Approx 25% complete for SRR5859403.fastq
Approx 30% complete for SRR5859403.fastq
Approx 35% complete for SRR5859403.fastq
Approx 40% complete for SRR5859403.fastq
Approx 45% complete for SRR5859403.fastq
Approx 50% complete for SRR5859403.fastq
Approx 55% complete for SRR5859403.fastq
Approx 60% complete for SRR5859403.fastq
Approx 65% complete for SRR5859403.fastq
Approx 70% complete for SRR5859403.fastq
Approx 75% complete for SRR5859403.fastq
Approx 80% complete for SRR5859403.fastq
Approx 85% complete for SRR5859403.fastq
Approx 90% complete for SRR5859403.fastq
Approx 95% complete for SRR5859403.fastq
Analysis complete for SRR5859403.fastq
```

SRR5859403.fastq
SRR5859403_fastqc.html
SRR5859403_fastqc.zip

8.3 RNA-seq 下游分析

表达矩阵每一列是一个样本，每一行是一个基因。（注意基因 symbol 和 ensembl id 的转换）
clusterprofiler 包中的 bitr 函数可以进行 symbol 到 ensembl id 的转化

8.3.0.1 差异表达基因分析（在 R 的环境下运行）

引入相应的包

```
library(tidyverse)
library(pheatmap)
library(DESeq2)
library(readxl)
library(EnhancedVolcano)
library(clusterProfiler)
library(org.Hs.eg.db)

#加载常用的包
#如未下载，可使用类似代码进行安装
#
#if (!require("BiocManager", quietly = TRUE))
#  install.packages("BiocManager")
#BiocManager::install("EnhancedVolcano")
#安装后，必须使用library命令激活
```

\begin{lstlisting}

```
getwd()
setwd('leb/rnaseq')
#切换工作目录
```

读取生成的文件

```
rna_seq <- read_excel(
  "RNAseq.xlsx",
  sheet = "Readcounts"
)
#读入excel文件
#注意！数据类型未tibble, 不是data.frame!
```

进行 id 转化

```
genid <- bitr(rna_seq$Geneid, fromType = "ENSEMBL", toType = "SYMBOL",
  OrgDb = org.Hs.eg.db)
# org.Hs.eg.db是一个homo sapiens数据库, 根据该数据库中的注释进行id转换
```

整理数据格式

```
rna_seq <- rna_seq %>%
  mutate(Gene = ...1) %>%
  dplyr::select(-...1) %>%
  group_by(Gene) %>%
  summarise_all(sum) %>% # 这里的sum是对每一列进行求和, 其实这个矩阵应该是已经整理好的不用。
  as.data.frame()

# 使用管道符

rownames(rna_seq) <- rna_seq$Gene
rna_seq <- rna_seq[, -1] # remove Gene column
print(head(rna_seq))
rna_seq <- rna_seq[-c(1:20), ] # 去掉前20行, 这里面是一些不是基因的东西
rna_seq <- rna_seq[rowSums(rna_seq) >= 10, ] # 去掉在所有样本里面表达量小于10的基因
```

直接生成图片的时候要把 plot 框调到合适大小

```
> biplot(p)
Error in grid.Call(C_convert, x, as.integer(whatfrom), as.integer(whattov),
  Viewport has zero dimension(s)
```

8.3.0.2 数据质控

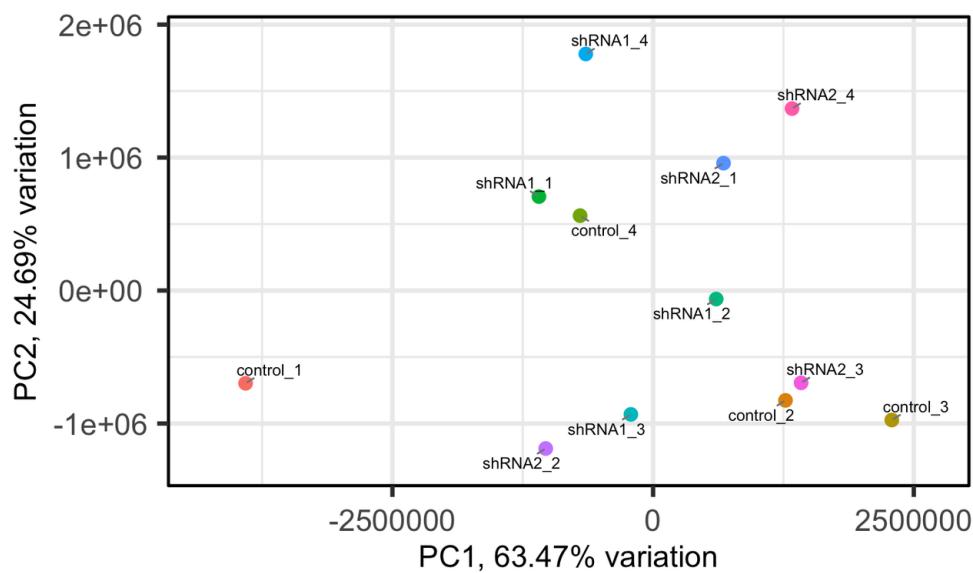
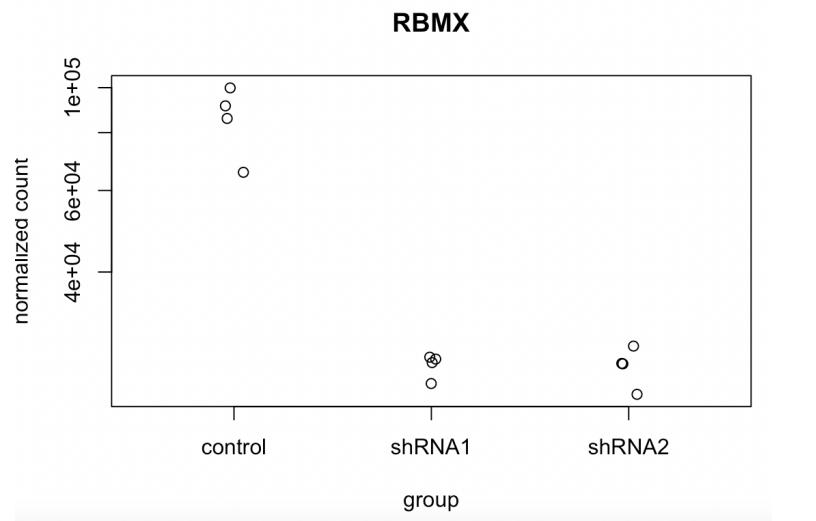
Plotcount 可以查看目标基因的表达情况

```
plotCounts(dds, gene = "RBMX", intgroup = "type")
```

使用 PCA 降维聚类

```
library(PCAtools)
vst <- rna_seq
meta <- data.frame(row.names = colnames(rna_seq), type = id$type)
p <- pca(vst, metadata=meta, removeVar = 0.05)
biplot(p)
```

火山图的绘制, 如图8.1。



```

res1 <- results(dds, name="type_shRNA1_vs_control")
res2 <- results(dds, name="type_shRNA2_vs_control")
EnhancedVolcano(res1,
  lab = rownames(res1),
  x = "log2FoldChange",
  y = "pvalue"
)
EnhancedVolcano(res2,
  lab = rownames(res2),
  x = "log2FoldChange",
  y = "pvalue"
)

```

pheatmap 对差异基因绘制热图，如图8.2。

```
res_filtered <- res2 %>%
```

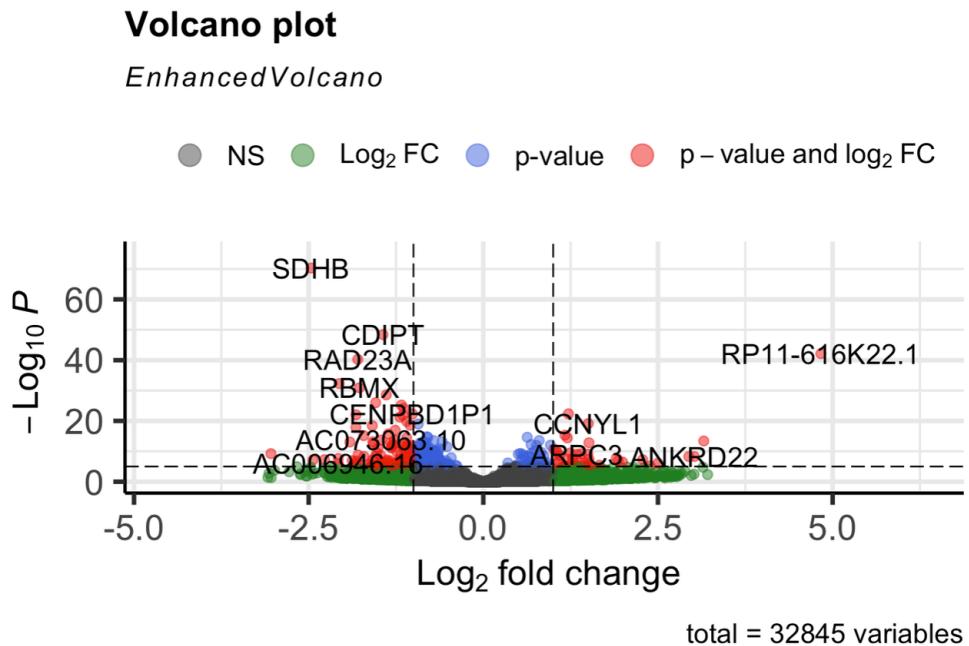


图 8.1: 火山图

```
as.data.frame() %>%
  rownames_to_column("Gene") %>%
  filter(padj < 0.05 & abs(log2FoldChange) > 1)
  dim(res_filtered)
  print(head(res_filtered))
  differential_genes <- res_filtered$Gene
  diff_matrix <- rna_seq[sample(differential_genes, 50), ]
  RBMX <- rna_seq["RBMX",]
  diff_matrix <- rbind(diff_matrix, RBMX)
  pheatmap(diff_matrix, scale = "row")
  ggsave("heatmap.png", width = 10, height = 10)
```

8.3.0.3 GO 富集分析

使用 clusterprofile 包可以进行 go 富集分析，如图8.3。

```
library(clusterProfiler)
library(org.Hs.eg.db)
#gene ontology
go_bp <- enrichGO(gene = differential_genes,
OrgDb = org.Hs.eg.db,
keyType = "SYMBOL",
ont = "BP", # BP: biological process, MF: molecular function, CC: cellular component, ALL
pAdjustMethod = "BH",
pvalueCutoff = 0.05,
qvalueCutoff = 0.05)
dotplot(go_bp, title = "GO Biological Pathway", showCategory = 10)
```

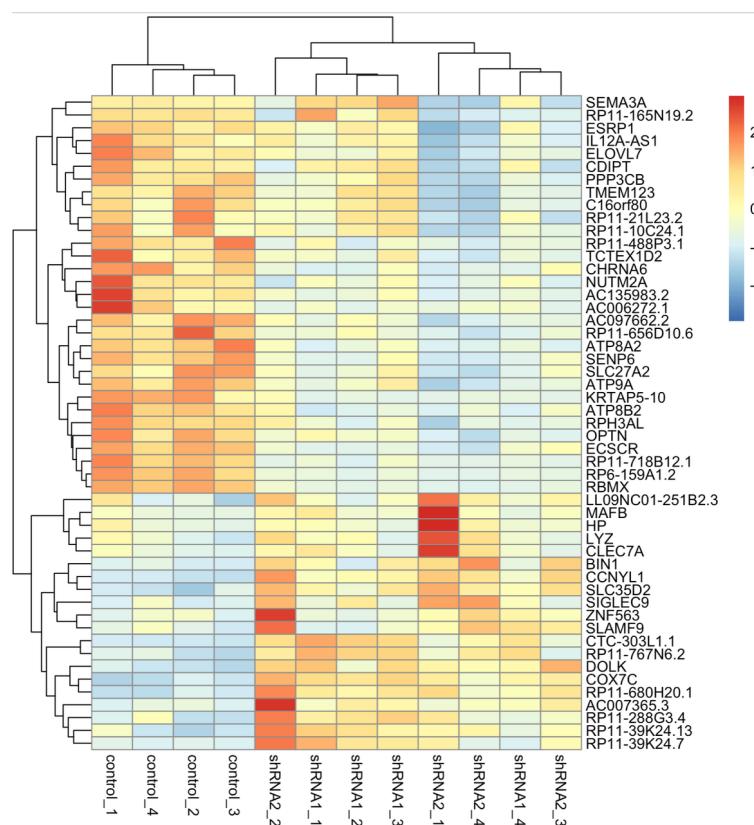


图 8.2: 基因差异表达结果

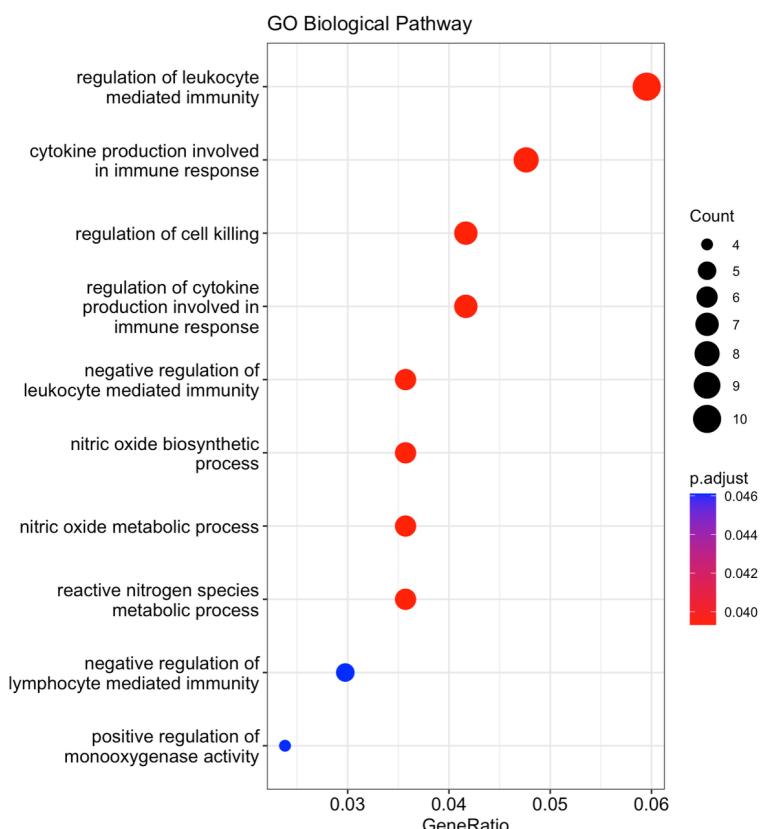


图 8.3: GO 富集分析结果

致谢

首先，非常感谢罗静初老师。罗老师非常认真地规划了这门课程，从课前预习部分、课上理论与实践、课后讨论，罗老师都会通过邮件和大家交流。

饶希晨师兄作为助教，在组织课程中投入了很多的时间与精力。在课程中，老师还邀请了吕钰麟、颜林林等学长为我们讲解相应部分的知识，也邀请了易鼎程、王沛宇、张任子墨等同学分享他们的经验。

罗老师为我们推荐的**Applied Bioinformatics Course**中包含大量的参考资料，包括生信分析常见软件、文献与常用的数据库网站，是非常实用的网站。

感谢教学中心为课程提供的教学服务器、高性能服务器，这保证我们在课程进行时能够顺利地同步运行代码。

在这门课程中，我们不仅了解、学习了生信分析相关的原理与软件，还学习了撰写代码的逻辑。

对于能够选上这门课程，我们感到十分幸运。

附录 A 环境配置

A.1 Linux 环境配置

Windows 和 MacOS 是当前最流行的两个操作系统，其中，MacOS 是类 Unix 系统，与 Linux 共享编程语言，因此你可以在 Terminal 中直接使用 Linux 相关指令。如果你是 Windows 系统，你需要安装 Linux 子系统或虚拟机。

A.1.1 Windows Subsystem for Linux, WSL

WSL 是 Windows Subsystem for Linux 的缩写，是一个在 Windows 10 上能够运行原生 Linux 二进制可执行文件（ELF 格式）的兼容层。它是由微软与 Canonical 公司合作开发，其目标是使纯正的 Ubuntu、Debian 等映像能下载和解压到用户的本地计算机，并且映像内的工具和实用工具能在此子系统上原生运行。

WSL 提供了一个完整的 Linux 内核，但它不是一个虚拟机。相反，WSL 提供了一个 Linux 系统调用兼容层，以便可以在 Windows 上运行原生 Linux 二进制文件。这意味着您可以在 Windows 上运行 Linux 命令行工具、脚本和应用程序，而无需使用虚拟机或双引导设置。

A.1.2 WSL 配置

[3] Windows 提供了方便的 WSL 配置流程，在管理员模式下打开 PowerShell 或 Windows 命令提示符，方法是右键单击并选择“以管理员身份运行”，输入 `wsl --install` 命令，然后重启计算机，即可完成安装。

```
wsl --install
```

若要更改安装的发行版，输入 `wsl --install -d <Distribution Name>`。将 `<Distribution Name>` 替换为要安装的发行版的名称。

若要查看可通过在线商店下载的可用 Linux 发行版列表，可输入： `wsl --list --online` 或 `wsl -l -o`。

安装完成后即可在终端（terminal）看到 Ubuntu 或直接在应用列表中搜索到 Ubuntu，如图A.1所示：

A.2 conda：环境管理系统

Conda 是一个开源的包管理系统和环境管理系统，可在 Windows、macOS 和 Linux 上运行。Conda 可快速安装、运行和更新包及其依赖项。Conda 可以轻松地在计算机上创建、保存、加载和切换环境。它是为 Python 程序而创造的，但它可以打包和分发任何语言的软件。

A.2.1 conda 安装

conda 分为 anaconda 和 miniconda 两种，前者包含一个基础的 python 环境，其中预装了常用的包；而后者较为精简，后续所有需要的东西可自行安装。这里展示 miniconda 的下载和安装过程，anaconda 类似，只需要更改下载链接即可。

A.2.1.1 conda 下载

以下是使用 wget 通过 conda 官网或者清华镜像源下载 miniconda 的代码，根据需要选择其一即可。

```
wget -c https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

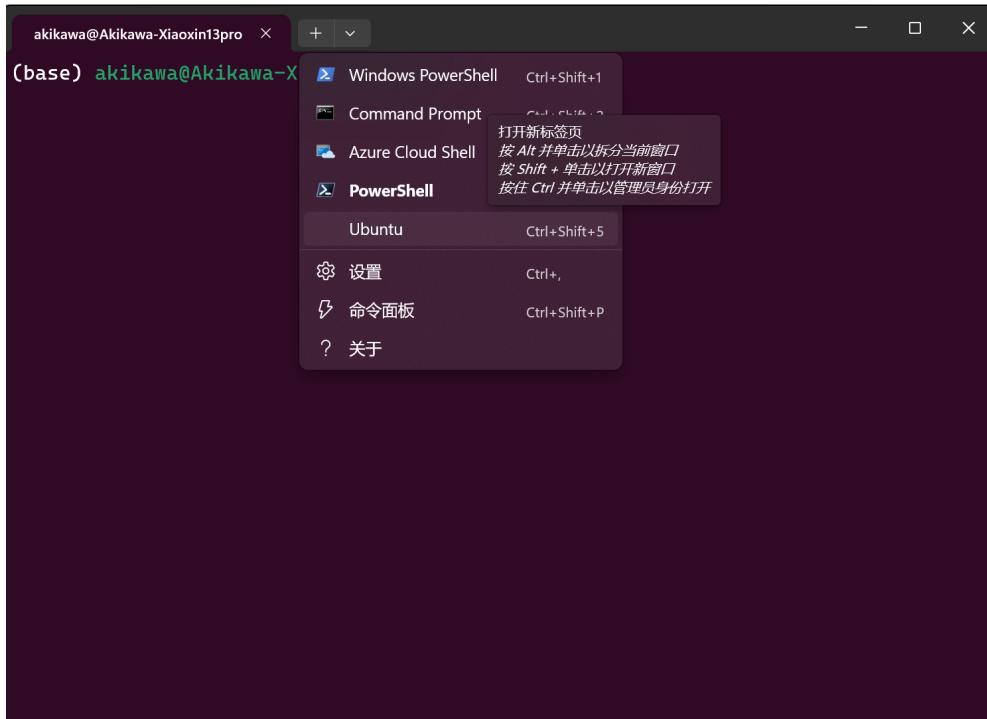


图 A.1: 在终端中打开 wsl

```
wget -c https://mirrors.bfsu.edu.cn/anaconda/miniconda/Miniconda3-latest-Linux-x86_64.sh #清华的镜像源latest的版本会一直更新到最新的版本
```

其中，-c 表示断点续传。一般来说，如果没有 VPN，清华镜像源下载速度较快，更为推荐。

A.2.2 conda 安装

使用 bash 运行下载好的Miniconda3-latest-Linux-x86_64.sh文件，如下：

```
bash Miniconda3-latest-Linux-x86_64.sh      #运行.sh
```

安装过程中某一步会让你指定安装路径，如果有特殊需要可以改为你想安装到的路径（我将 linux 上安装的软件都放入了~/.app中），否则全部选择 yes 即可。

A.2.3 conda 运行

安装完成后，在终端中直接运行 conda 是无法启动的，你可以在~/.bashrc中看见 conda 的路径，需要运行.bashrc 文件进行初始化后方可进入 conda 环境，.bashrc文件内容如下图：

A.2.4 环境

环境是 conda 的重要概念，conda 可以创建各种环境，每个环境可以指定具体的 python 版本，可以在指定的环境下安装管理自己所需的包，并且环境之间相互隔离，互相不影响，类似命名空间的作用，对于不同需求场景下可以进行环境的自由切换，以下是与环境相关的一些简单命令，可在终端中打开 Ubuntu 后输入以下命令执行对应操作。

```
# 创建环境
conda create -n forfun python=3.6
# 列出所有环境
```

```

# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup=$(('/home/akikawa/.app/anaconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/akikawa/.app/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/home/akikawa/.app/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/akikawa/.app/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<

```

图 A.2: .bashrc 文件中的 conda 初始化部分

```

conda env list
# 删除环境
conda env remove -n forfun
# 激活环境
source activate forfun
# 退出环境
source deactivate

```

在某个环境下，可以通过下述命令管理包

```

# 检索可以下载的包
conda search numpy
# 下载包
conda install numpy
# 移除包
conda remove numpy
# 列出所有安装包
conda list

```

A.3 VSCode：轻量化集成代码编辑器

VSCode 全称 Visual Studio Code，是微软出的一款轻量代码编辑器，免费、开源而且功能强大。它支持几乎所有主流的程序语言的语法高亮、智能代码补全、自定义热键、括号匹配、代码片段、代码对比 Diff、GIT 等特性，支持插件扩展，并针对网页开发和云端应用开发做了优化。软件跨平台支持 Win、Mac 以及 Linux。

A.3.1 VSCode 安装及中文环境配置

打开 [VSCode 官网](#)，点击下载链接即可 s 下载安装。下载简体中文插件后即可将语言切换为中文。此后，如需配置对应的环境，如 Python、R、Latex 环境，只需下载主程序后¹在 VSCode 中安装对应的包即可。

注 VSCode 除了各类代码，还可以方便地打开 xml、csv、xlsx、docx、tif、html、svg 等常见文件，只要安装对应的插件即可；也可以直接对文本文件进行编辑，因而 VSCode 为没有图形化界面的 WSL 增添了许多便利。

¹因为本质上 VSCode 只是一个壳，和记事本类似，程序的运行是在对应终端中进行的

A.3.2 VSCode 连接 WSL、SSH

安装好 WSL 后，在 VSCode 左侧侧边栏的 Remote Explorer 分区，你可以在上方选择 WSL Target 直接连接到子系统，方便完成文本编辑，文件预览，文件传输等功能（可将 Windows 上文件直接拖动到 VSCode 的资源管理器中完成上传，将 WSL 上文件通过右键点击完成下载）等功能

A.3.3 在 VSCode 上通过 SSH 连接远程 Linux 服务器

A.3.3.1 在 VSCode 中配置 SSH

本部分参考 CSDN 上部分文章 [2]，结合我们自身配置过程书写而成。

在 win10 中安装 open-SSH 客户端，可使用“应用与程序”或“windows power shell”，如图A.3。

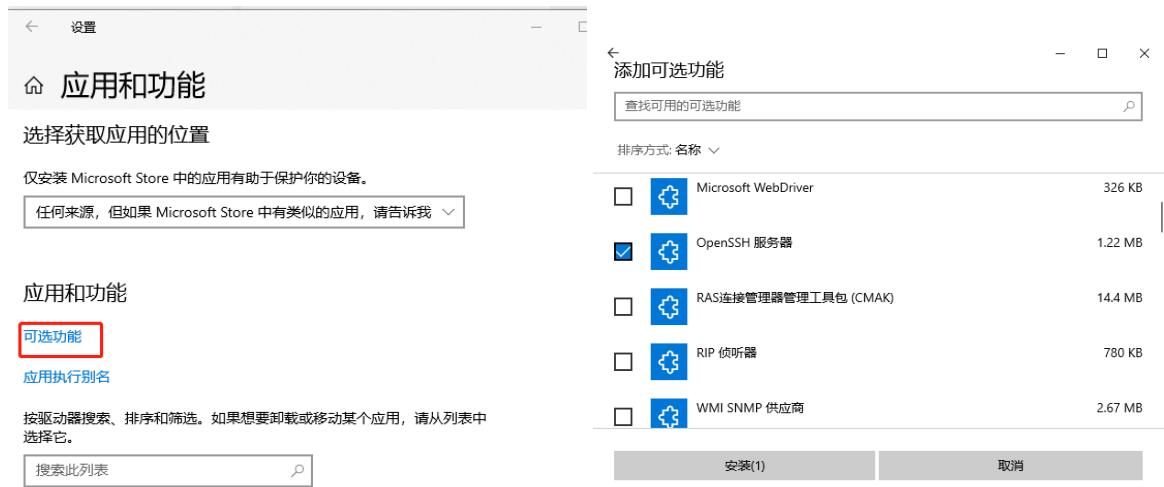
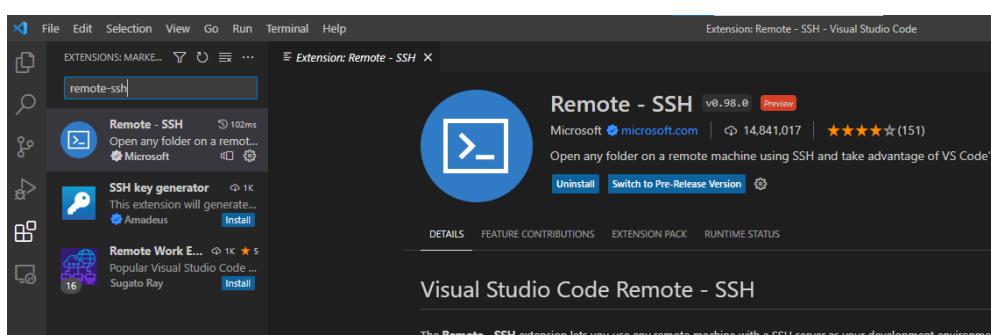


图 A.3: Win10 中安装 OpenSSH

在 vs-code 的 extension 中搜索 remote-ssh 并安装



配置.config 文件

```
Host LEB
HostName 117.78.18.116
User leb4c

Host LEB_enhanced
HostName 114.115.164.221
User leb04c
```

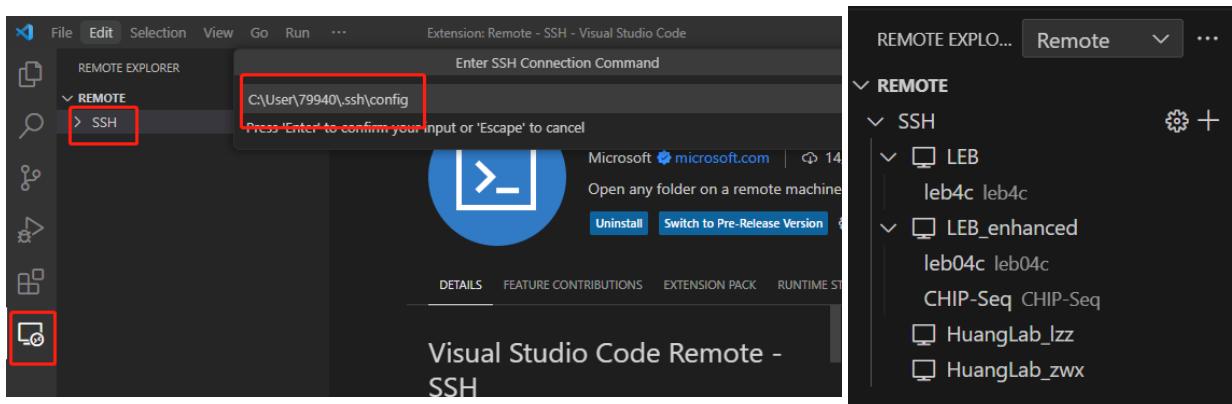
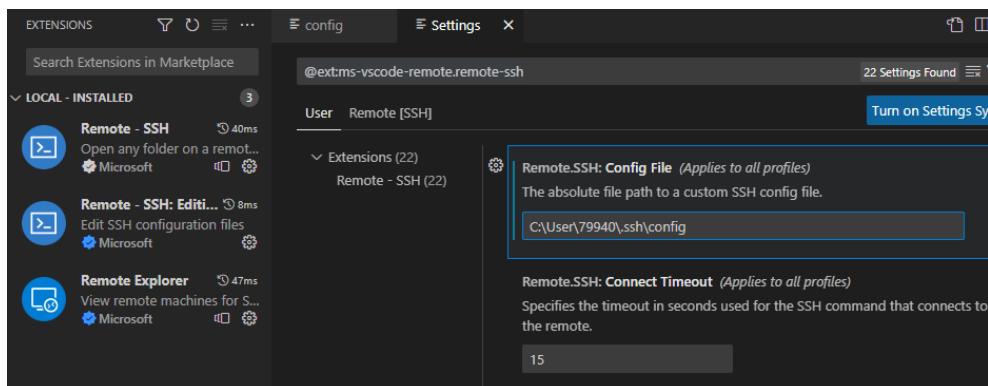


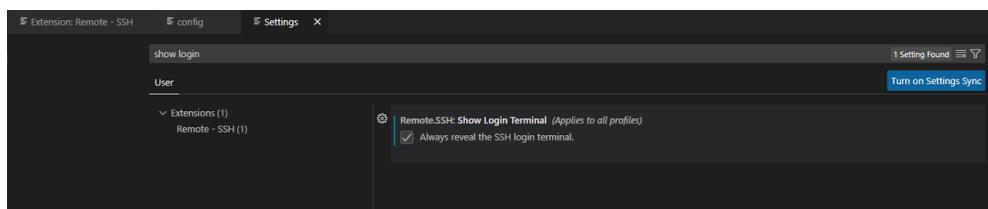
图 A.4: VSCode 中配置 SSH 远程连接

其中，Host、HostName 和 User 分别表示你希望显示在本地的服务器名称，服务器 SSH 地址和你在服务器上登录的用户名。保存后可直接点击列表内的服务器登录，如图A.4右所示。

右键添加扩展设置。



勾选首选项中显示登录窗口，以输入密码



在 VSCode 中登录并在提示后输入密码，如图A.5。

可以通过 VSCode 清晰展示各个文件、路径，可在展示 terminal 的同时修改文件内容，如图A.6。

A.3.3.2 配置 SSH 免密码登录

本部分参考 CSDN 上部分文章 [5]，结合我们自身配置过程书写而成。该部分主要针对 Windows 用户，MacOS 用户配置过程类似，不过在获取密钥和复制密钥部分有所不同，需要留心。

首先，使用win+R进入 cmd，在 cmd 中输入ssh-keygen -t rsa获取密钥，如图A.7，文件中生活蹭了公钥和私钥。

然后，在配置文件中添加密钥地址，注意为不含.pub 后缀的文件

```
# Read more about SSH config files: https://linux.die.net/man/5/ssh_config
Host leb4a
```

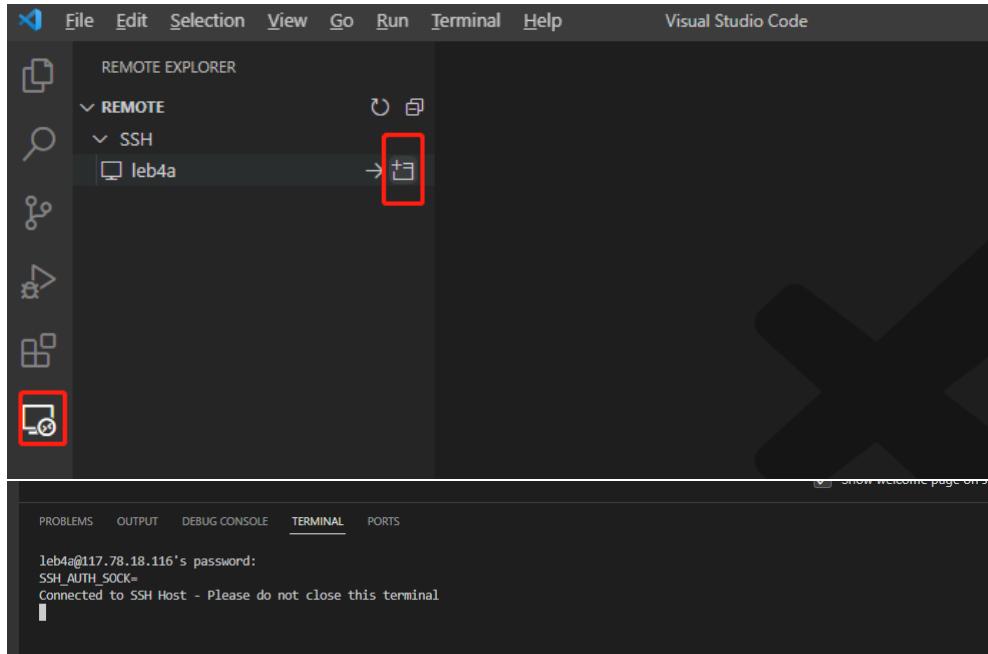


图 A.5: VSCode 中登录

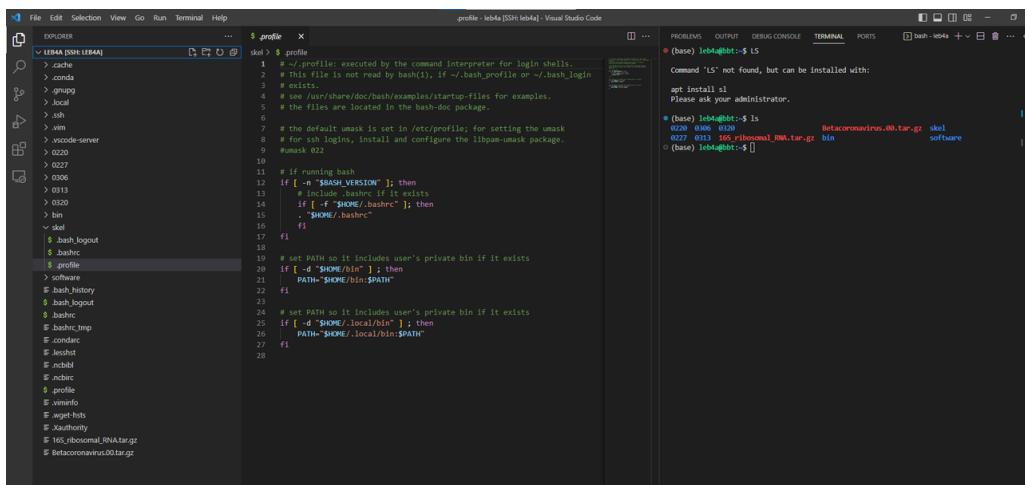


图 A.6: VSCode 连接服务器后的页面

```
HostName 117.78.18.116
# 服务器地址
User leb4a
# 登录服务器时的用户名
PreferredAuthentications publickey
IdentityFile "C:\Users\79940\.ssh\id_rsa"
# 本地密钥文件地址
```

在服务器下.ssh 文件夹中，使用 touch 新建 authorized_keys 许可文件，将公钥.pub 文件中的内容复制到 authorized_keys 中，如图A.8，即可完成不输入密码登录。

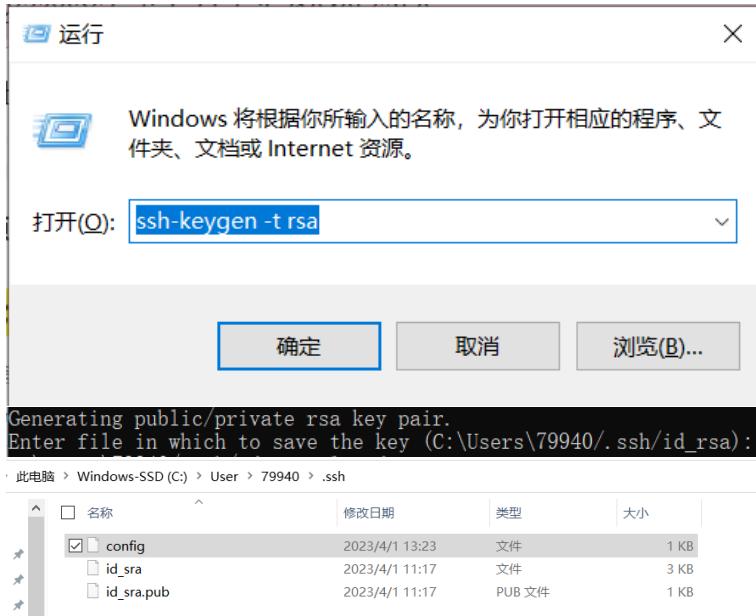


图 A.7: 在 CMD 中生成密钥

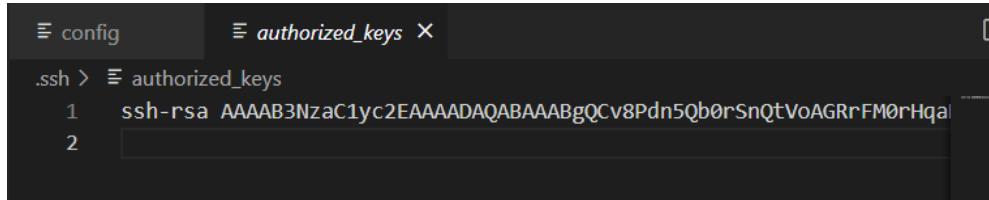


图 A.8: 将.pub 文件中内容复制到服务器

A.4 实例：VSCode 中使用 JupyterNotebook 运行 R

Conda 除了可以配置 python 环境，也可以用于配置和管理 R 环境，以此将不同项目所用的 R 环境隔开，减少 package 加载时间，同时防止 package 冲突。这里，我们利用 conda 在 WSL 中配置 R 的 JupyterNotebook 环境并在 VSCode 中编辑与运行。

A.4.1 用 conda 配置 R 环境

```
conda create -n r_env r-base=4.3.0 r-essentials
# -n 指定r环境名称，此处为"r_env"
# r-base=xx，设置r版本
# r-essentials，基础Rpackages，选装
conda activate r_env
```

A.4.2 安装 package

如果用 conda 安装 r 包，需要切换对应的到 r 环境下，如 (r_env)，且包名前需要加“r”，如 conda install r-package_name

但更推荐的是使用 r 的包管理应用安装包，如 BiocManager。

在终端中切换到 r_env，并启动 R，然后使用 BiocManager 安装需要的包。需要注意的是此步需要在终端内完成，不能在 JupyterNotebook 中完成，否则无法连接网络。

```
# 启动R环境
R

# 以下已经进入R环境
# 如果没有BiocManager就安装BiocManager
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

# 用BiocManager安装对应包
BiocManager::install("DESeq2")
BiocManager::install("readxl")
BiocManager::install("EnhancedVolcano")
```

A.4.3 在 vsCode 中使用 JupyterNotebook 运行 R 程序

利用 conda 下载 JupyterNotebook for R 的核心插件 irkernel

```
conda install -c r r-irkernel
```

然后就可以使用网页端 JupyterNotebook 了！命令如下：

```
conda activate r_env
jupyter notebook
```

或者也可以在 VSCode 中选择 kernel（安装 irkernel 后需要先重启 VSCode 才能看到内核）

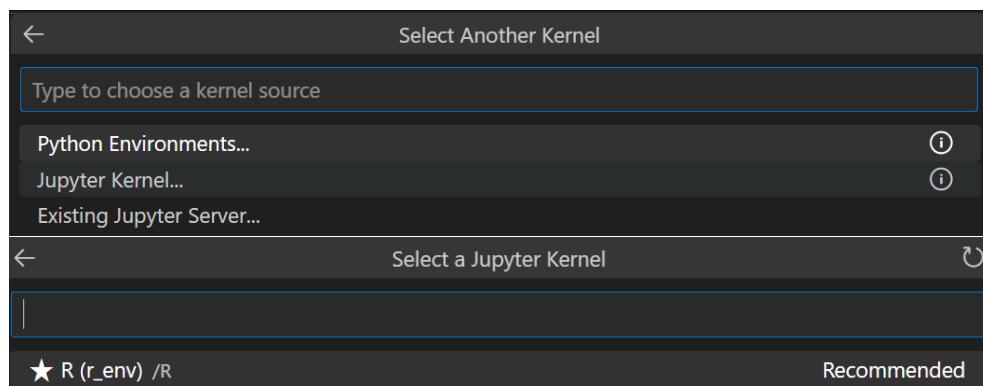


图 A.9：在 VSCode 中选择 JupyterNotebook 的内核

这样，就可以在 vscode 中愉快地使用 R 了！

附录 B Git 和 GitHub 简介

在本章的最开始，我们先做一个本质的区分：

Git: 一个用于版本控制的软件

Github: 运营的网站，协作式源代码托管网站

相信在阅读过程中，你能逐渐体会到两者的差异和联系。

B.1 Git

Git 是一个开源免费的分布式版本管理系统，由芬兰程序员 Linus Torvalds 开发。Git 的版本控制可以对代码、文档和日常工作所用到的文件进行方便的管理，而不用多次对源代码进行复制。还可以在多人合作的项目中可以记录所有的版本信息，同步不同来源的改动并在不同的地方备份整个项目。

B.1.1 Git 的基本概念

本部分参考了 [1] 的内容。

B.1.1.1 版本控制系统

Git 是目前世界上最优秀的分布式版本控制系统。版本控制系统是能够随着时间的推进记录一系列文件的变化以便于你以后想要的退回到某个版本的系统。版本控制系统分为三大类：本地版本控制系统，集中式版本控制系统和分布式版本控制系统。

本地版本控制（Local Version Control Systems）是将文件的各个版本以一定的数据格式存储在本地的磁盘（有的 VCS 是保存文件的变化补丁，即在文件内容变化时计算出差量保存起来），如图B.1左所示。这种方式在一定程度上解决了手动复制粘贴的问题，但无法解决多人协作的问题。

集中式版本控制（Centralized Version Control Systems）相比本地版本控制没有什么本质的变化，只是多了一个中央服务器，各个版本的数据存储在中央服务器，管理员可以控制开发人员的权限，而开发人员也可以从中央服务器拉取数据，如图B.1右所示。集中式版本控制虽然解决了团队协作问题，但缺点也很明显：所有数据存储在中央服务器，服务器一旦宕机或者磁盘损坏，会造成不可估量的损失。

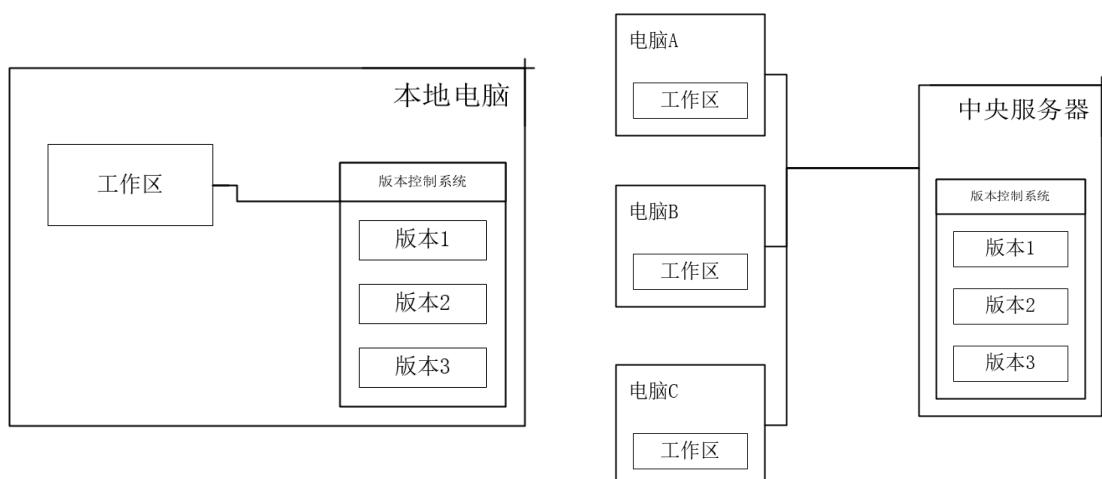


图 B.1: 本地式与集中式版本控制系统

分布式版本控制（Distributed Version Control System）与前两者均不同，如图B.2所示。首先，在分布式版本控制系统（如 Git, Mercurial, Bazaar 及 Darcs 等）中，系统保存的不是文件变化的差量，而是文件的快照，即把文件的整体复制下来保存，而不关心具体的变化内容。其次，最重要的是分布式版本控制系统是分布式的，当你从中央服务器拷贝下来代码时，你拷贝的是一个完整的版本库，包括历史纪录，提交记录等，这样即使某一台机器宕机也能找到文件的完整备份。

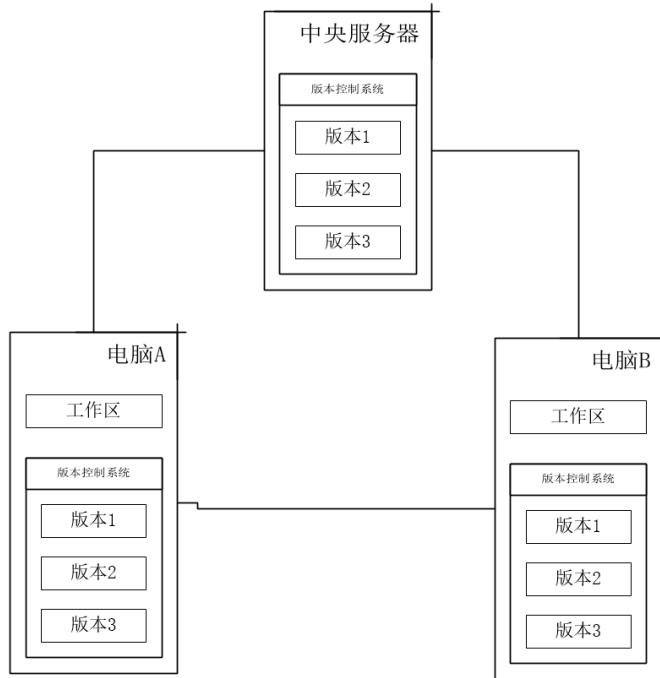


图 B.2: 分布式版本控制系统

B.1.1.2 Git 介绍

Git 是一个分布式版本控制系统，保存的是文件的完整快照，而不是差异变化或者文件补丁。

Git 每一次提交都是对项目文件的一个完整拷贝，因此你可以完全恢复到以前的任一个提交而不会发生任何区别。这里有一个问题：如果我的项目大小是 10M，那 Git 占用的空间是不是随着提交次数的增加线性增加呢？我提交（commit）了 10 次，占用空间是不是 100M 呢？答案是否，如果文件没有变化，Git 只会保存指向一个版本的文件的指针，即对于一个特定版本的文件，Git 只会保存一个副本，但可以有多个指向该文件的指针。如图B.3右所示。

注 Git 最适合保存文本文件，如各种语言的源代码，因为 Git 可以对文本文件进行很好的压缩和差异分析。而针对视频、图片等二进制文件，Git 版本管理并不能取得较好的效果（压缩比率低，不能差异分析）。对于二进制文件，Git 压缩率非常小，其占用空间随提交次数几乎线性增长。

Git 有三个工作区域：工作目录，暂存区域，以及本地仓库，三者关系如图B.4左所示。工作目录是你当前进行工作的区域；暂存区域是你运行 git add 命令后文件保存的区域，也是下次提交将要保存的文件（注意：Git 提交实际读取的是暂存区域的内容，而与工作区域的文件无关，这也是当你修改了文件之后，如果没有添加 git add 到暂存区域，并不会保存到版本库的原因）；本地仓库就是版本库，记录了你工程某次提交的完整状态和内容，这意味着你的数据永远不会丢失。file 相应的，文件也有三种状态：已提交（committed），已修改（modified）和已暂存（staged）。已提交表示该文件已经被安全地保存在本地版本库中了；已修改表示修改了某个文件，但还没有提交保存；已暂存表示把已修改的文件放在下次提交时要保存的清单中，即暂存区域。所以使用 Git 的基本工作流程如图B.4右所示，即：

1. 在工作区域增加，删除或者修改文件。

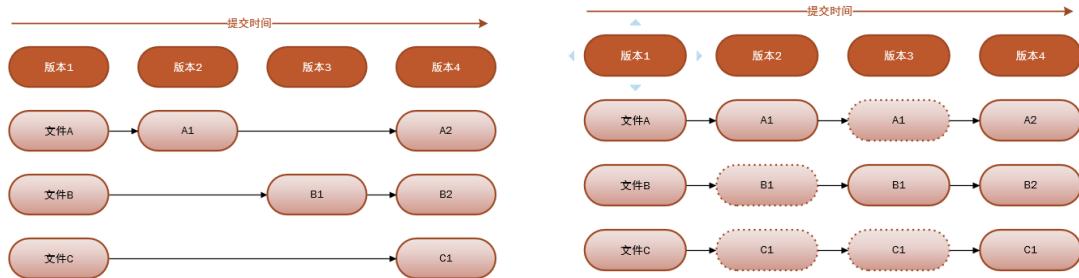


图 B.3: Git 的文件管理方式

2. 用 Git 将文件快照保存到暂存区域。
3. 提交更新，将文件永久版保存到版本库中。

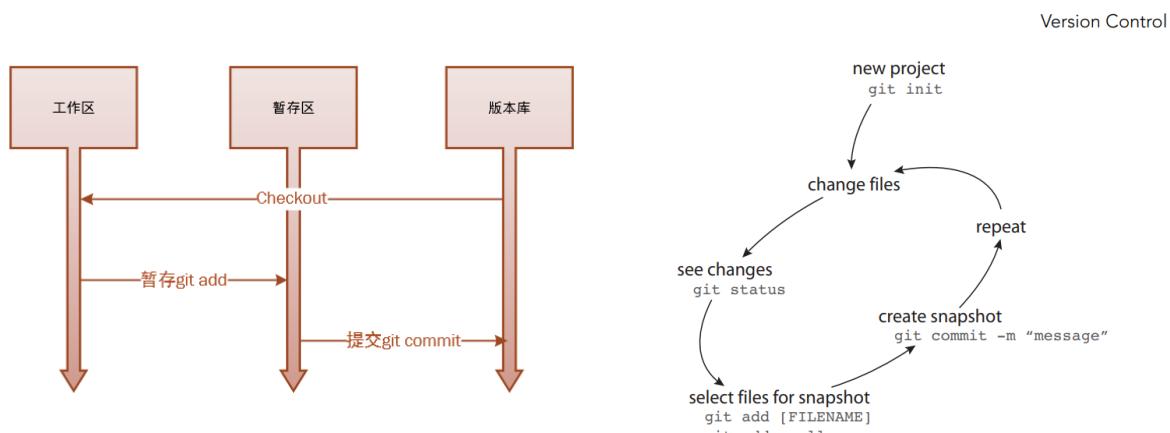


图 B.4: Git 分区及各分区关系和使用 Git 的工作流程

B.1.2 Git 的命令操作

Git 的常用命令分为本地和远程两部分。本地命令主要是在本地进行对代码的版本控制和分支，远程命令则包含了与远程存储库通信的方法。

B.1.2.1 Git 环境配置

为了协同和历史查询的方便，Git 在操作过程中会记录操作者的有关信息，这些有关信息都可以在.gitconfig 文件中进行配置。值得一提的是，为了后续与 GitHub 传递文件时网络稳定，建议为 Git 配置代理，Clash for windows 默认监听 127.0.0.1:7890 端口，故配置命令如下：

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
git config --global http.proxy http://127.0.0.1:7890
```

Windows 系统的.gitconfig 文件位于用户文件夹下，是一个隐藏文件，配置完成后.gitconfig 文件内容如下：

```
[user]
email = 1418767078@qq.com
name = MingchuanTang
```

```
[http]
proxy = http://127.0.0.1:7890
```

B.1.2.2 Git 本地命令

```
## 版本控制的基本操作
# 首先需要在工作文件夹下初始化git，生成本地git仓库
git init
# 查看git状态：是否有未提交的文件和处在缓存状态下的文件
git status
# 提交该文件夹下的所有文件进入缓存状态
git add .
# 提交该文件夹下的某个文件
git add <the file>
# 将缓存状态下的所有文件提交到本地git仓库
git commit
# 输入此次提交的说明信息
git commit -m "message for the upload"
# 查看提交的历史记录
git log
```

除此之外，你可以将不需要跟踪/更新的文件加入`.gitignore`文件中，这样在`git add`和`git commit`操作均不会改变`.gitignore`中文件的状态。

B.1.2.3 Git 远程命令

```
# 将远程仓库中的内容克隆到本地
git clone <weblink>
# 从远程仓库获取更新合并到本地
git pull
# 将本地更新提交到远程仓库
git push
# 从远程仓库获取更新
git fetch
# 查看远程仓库
git remote
```

B.1.3 Git 的重要功能——分支

分支（branch）是 Git 最重要的功能之一¹，使用 Git 可以在工作流程中频繁使用分支与合并，因为 Git 分支非常轻量级，不像其他的版本控制，创建分支意味着要把项目完整的拷贝一份，而 Git 创建分支是在瞬间完成的，与工程的复杂程度无关。

与分支有关的操作如下，可通过网站[Learn Git Branching](#)练习分支有关操作。

```
# 创建分支
git branch new_branch_name
```

¹该部分内容较进阶，在开发新功能、合作完成项目时能起到巨大作用。初学者略读，了解就好，可在实际使用中不断加深对分支的理解。

```
# 转到某个分支，之后的 add ``commit 创建子分支等操作将在此基础上进行
git checkout exist_branch_name
# 创建并转到某个分支
git checkout -b new_branch_name
# 将当前分支和 branch_name 分支合并
git merge branch_name
# 将文件放入暂存区
git add
# 将暂存区内容添加到本地仓库
git commit
# 合并两个分支，最为双方的子节点
git merge another_branch
# 合并分支并将该分支节点作为 another branch 单独的子节点
git rebase another_branch
```

B.1.3.1 分支的创建与切换

Git 的默认分支是 master (因为某些原因现在改为了 main)，存储在 .git\refs\heads\master (main) 文件中，假设你在 master 分支运行 git branch dev 创建了一个名字为 dev 的分支，那么 Git 所做的实际操作是：

1. 在 .git\refs\heads 文件夹下新建一个文件名为 dev (没有扩展名) 的文本文件。
2. 将 HEAD 指向的当前分支 (当前为 master) 的 40 位 SHA-1 校验和外加一个换行符写入 dev 文件。
3. 结束。

.git > refs > heads			
名称	修改日期	类型	大小
dev	2016/10/3 15:09	文件	1 KB
master	2016/10/3 15:10	文件	1 KB

图 B.5: branch 在文件夹中的存储方式

如图 B.5 所示。

对于切换分支，Git 实际完成了如下操作：

1. 修改 .git 文件下的 HEAD 文件为 ref: refs/heads/< 分支名称 >。
2. 按照分支指向的提交记录将工作区的文件恢复至一模一样。
3. 结束。

B.1.3.2 分支合并 merge

创建多条分支后，最终都会面临合并到一个主线的问题。合并分支首先是 Fast-forward，换句话说，如果顺着一个分支走下去可以到达另一个分支的话，那么 Git 在合并两者时，只会简单地把指针右移，因为这种单线的历史分支不存在任何需要解决的分歧，所以这种合并过程可以称为快进 (Fast forward)。如图 B.6 上：注 注意箭头方向，因为每一次提交都有一个指向上次提交的指针，所以箭头方向向左

当在 master 分支合并 dev 分支时，因为他们在一条线上，这种单线的历史分支不存在任何需要解决的分歧，所以只需要 master 分支指向 dev 分支即可，所以很快。

当分支出现分叉时，就有可能出现冲突，而这时 Git 会要求你去解决冲突，如图 B.6 中所示：

因为 master 分支和 dev 分支不在一条线上，即 v7 不是 v5 的直接祖先，Git 不得不进行一些额外处理。就此例而言，Git 会用两个分支的末端（v7 和 v5）以及它们的共同祖先（v3）进行一次简单的三方合并计算。合并之后会生成一个合并提交 v8。

注 合并提交有两个祖先（v7 和 v5）

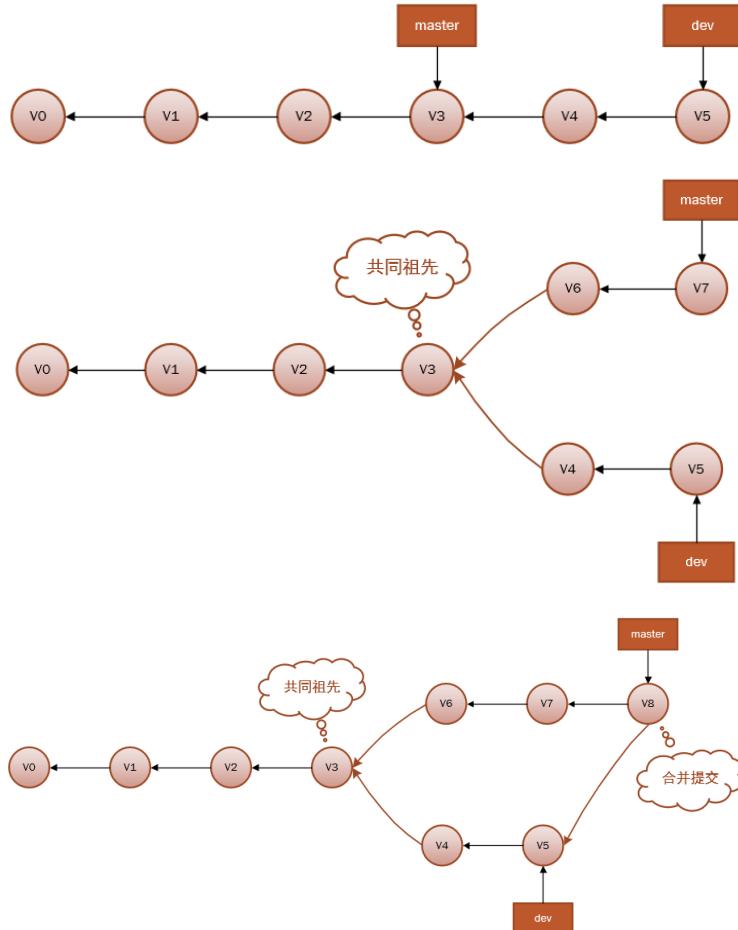


图 B.6: Git merge 的可能情况

B.1.3.3 分支变基 rebase

把一个分支中的修改整合到另一个分支的办法有两种：merge 和 rebase。merge 和 rebase 最终的结果是一样的，但 rebase 能产生一个更为整洁的提交历史。仍然以图B.7上为例，如果简单的 merge，会生成一个提交对象 v8，现在我们尝试使用变基合并分支，切换到 dev：

```
git checkout dev
git rebase master
# First, rewinding head to replay your work on top of it...
# Applying: added staged command
```

这段代码的意思是：回到两个分支最近的共同祖先 v3，根据当前分支（也就是要进行变基的分支 dev）后续的历次提交对象（包括 v4, v5），生成一系列文件补丁，然后以基底分支（也就是主干分支 master）最后一个提交对象（v7）为新的出发点，逐个应用之前准备好的补丁文件，最后会生成两个新的合并提交对象（v4', v5'），从而改写 dev 的提交历史，使它成为 master 分支的直接下游，如图B.7中：

现在，就可以回到 master 分支进行快速合并 Fast-forward 了，因为 master 分支和 dev 分支在一条线上，如图B.7下：

```
git checkout master
git merge dev
```

现在的 v5' 对应的快照，其实和普通的三方合并，即上个例子中的 v8 对应的快照内容一模一样。虽然最后整合得到的结果没有任何区别，但变基能产生一个更为整洁的提交历史。如果视察一个变基过的分支的历史记录，看起来会更清楚：仿佛所有修改都是在一根线上先后进行的，尽管实际上它们原本是同时并行发生的。

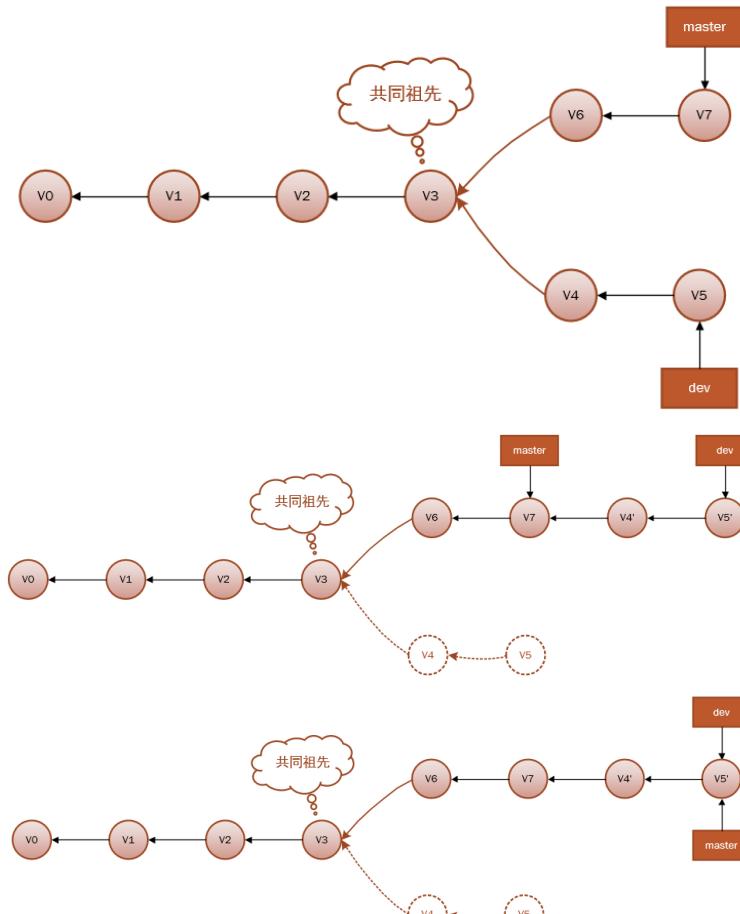


图 B.7: Git rebase 过程

B.2 Github

GitHub 是一个基于 Git 的远程代码托管仓库，目前由微软公司所有。它提供了一个用户友好的 Web 界面，让开发者可以在浏览器中轻松地管理 Git 仓库。GitHub 不仅提供公共仓库，让开发者可以共享和协作开源项目，还提供私有仓库，供团队内部协作使用。GitHub 的功能包括：代码审查、问题跟踪、团队协作工具、代码托管等。从此前对分布式版本管理系统的描述，你可以看出 GitHub 的角色就是其中的中央服务器，类似 GitHub 的平台还有 GitLab, Gitolite 等。

注 Git 和 GitHub 的关系在于，GitHub 是一个基于 Git 的服务。Git 本身是一个独立的分布式版本控制系统，可以在本地和任何远程服务器上使用，而 GitHub 则为 Git 提供了一个便捷的在线平台，使得开发者可以轻松地共享、协作和托管代码。

与 GitHub 相关的 Git 基本操作见 B.1.2.3，除此之外，为了方便写作 GitHub 提供了如下功能：

- Issue: 写出项目的 bug 等，发起话题供大家讨论
- Pull Request & Merge: 远程协作，将自己的代码整合到项目中
- Comment: 对代码等提出评论
- Follow: 对项目、开发者、组织等进行关注和跟进
- Star: “收藏”功能

B.2.1 实例：通过 GitHub 创建小组仓库并同步小组资料

我们曾在前言中提到小组的 GitHub 仓库 **LEB23_G4**。该仓库的维护过程正是本节的内容。

1. 安装 Git 本部分参考了 CSDN 上的文章 [4]。通过git-scm.com、gitforwindows.org或者[阿里镜像](#)下载系统对应的安装包，然后跟随安装指南一步步安装即可。
2. 配置本地 Git（设置用户名和邮箱）

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

3. 使用令牌登录

参考文章：[6]。可在 GitHub 中获取仓库 token（令牌），如图B.8。

```
git remote set-url origin https://<your_token>@github.com/<USERNAME>/<REPO>.git
```

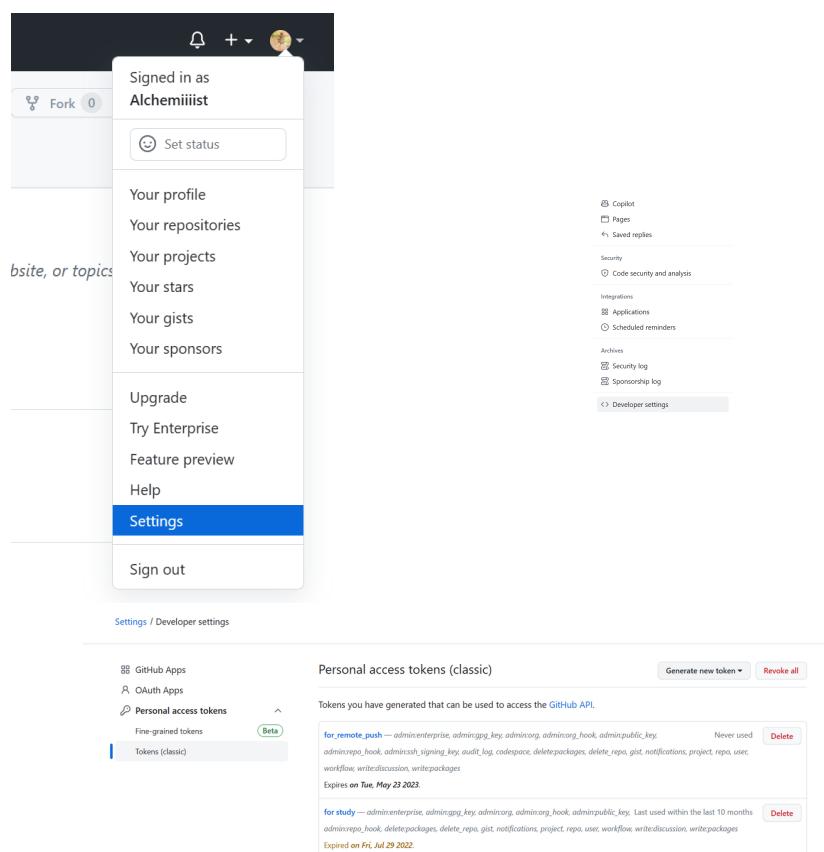


图 B.8: 在 GitHub 中设置 token

4. 注册 GitHub 账号并且建立仓库

在主页点击repository-new 即可创建仓库，如图B.9

得到仓库地址：<https://github.com/dkyyyyyyyyyy/dkys-repository.git>

5. 本地新建测试文件夹，并且建立与指定远程仓库相同步的本地仓库

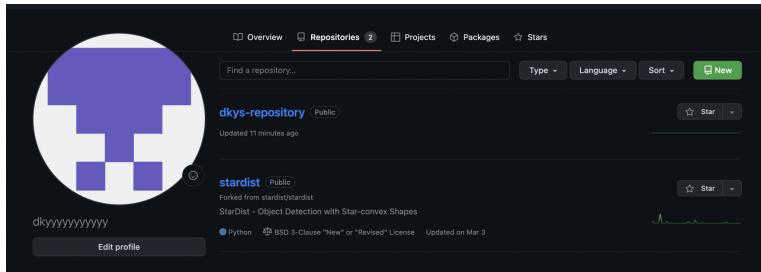


图 B.9: GitHub 仓库

```
touch 1234.txt
git clone https://github.com/dkyyyyyyyyyy/dkys-repository.git
```

6. 暂存并提交代码

```
git add *
git commit -m 'messages'
git branch -M main
git push origin main (本地) :main (远程)
```

VSCode 会自动连接到 github 询问授权信息，点击同意即可。

7. 传输完成

如其他人上传到你的仓库，会收到邮件，最后在网页上同意 pull request 即可，如图B.10。

B.2.1.1 注意事项

一般来说，针对自己的 GitHub 仓库，直接 push 即可将本地文件与云端同步，但是对于其他人的 GitHub 仓库，若没有 token，则协作的一般过程是在网页端将他人的仓库 fork 到自己的仓库，然后在本地克隆 fork 的仓库。本地进行的修改将和自己 fork 的仓库同步，最后在网页端向原仓库发起 pull request（即请求将自己的修改 merge 到原仓库中），由原仓库端消除冲突后即可完成对原项目的贡献，你的 GitHub 账户也会出现在原仓库的 contributor 列表中。

若有 token，则可按照上述步骤直接对原仓库进行修改，所以切记，保护好自己的 token，不要轻易泄露，以防对自己的项目造成不可估量的损失。

B.3 实例：VSCode 中利用 Git 和 GitHub 管理文件

VSCode 自带 Source control 功能，可以在此进行 commit、push 等行为。特别的是，如果需要对其他项目进行协作，vscode 能自动完成创建 fork（为用户创建一个原文件的复制）和 pull and push（将本地和云端仓库同步）的工作，省去很多麻烦。

如图B.11中所示，如果用 GitHub 打开一个含.git 文件夹的目录（经过 git init 的目录），则可在 Source Control 页面查看当前的文件状态，其中 M 表示修改，U 表示新加，D 表示删除。同时可右键点击某文件对其进行单独处理，如图B.11右所示。

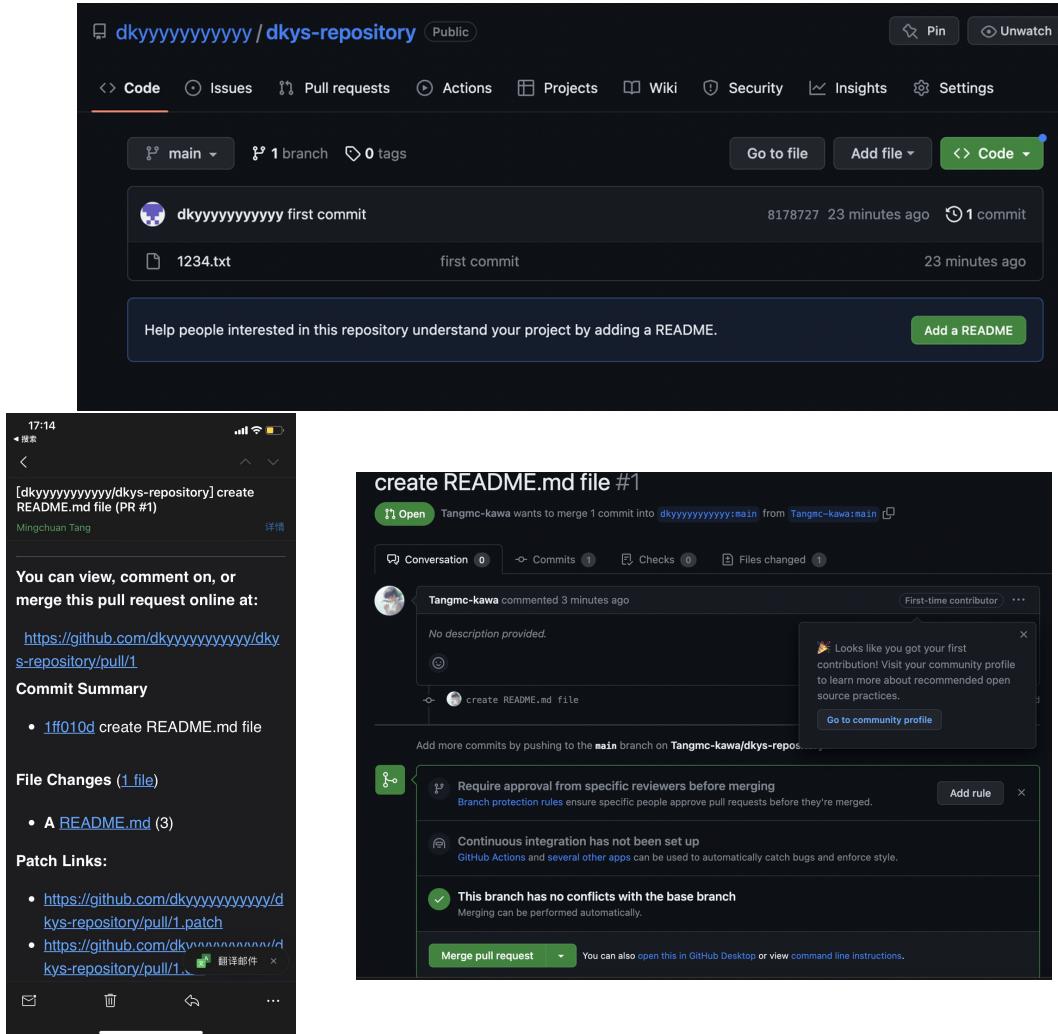


图 B.10: git pull request

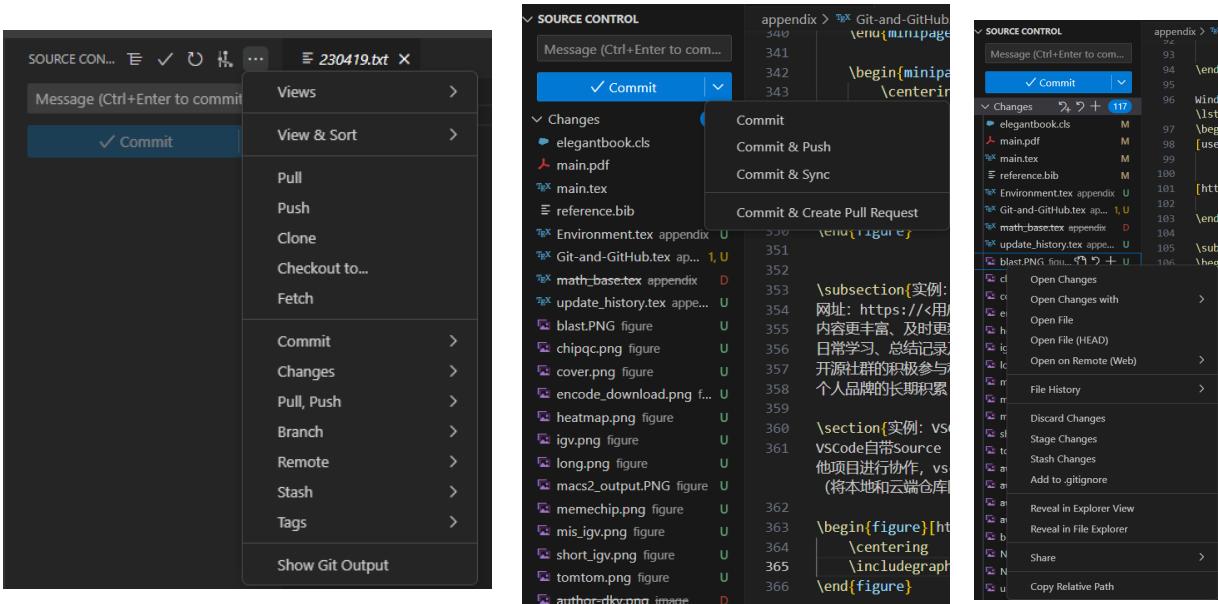


图 B.11: VSCode 的 Source Control 功能实现 Git 所有操作

参考文献

- [1] Congcong Li. *Git* 的核心概念. <https://blog.lufficc.com/the-core-conception-of-git/>. 2016.
- [2] der.der. *VSCCode* 远程登录 *Linux*, 实现 *Windows* 下进行 *Linux* 编程. <https://blog.csdn.net/qq42669026/article/details/10405635>. 2020.
- [3] Microsoft. 使用 *WSL* 在 *Windows* 上安装 *Linux*. <https://learn.microsoft.com/zh-cn/windows/wsl/install>. 2023.
- [4] mukes. *Git* 详细安装教程(详解 *Git* 安装过程的每一个步骤). <https://blog.csdn.net/mukes/article/details/115693833>. 2023.
- [5] ninding. 【软件教程】如何让 *vscode* 连接 *ssh* 时免密登录. <https://blog.csdn.net/zhangjiuding/article/details/129478438>. 2023.
- [6] 前端历劫之路. 2021.8.13 起, *Github* 要求使用基于令牌的身份验证. <https://zhuanlan.zhihu.com/p/401978754>. 2021.