尊敬的王老师：

展信佳！

我按照计划完成了本周的学习任务，现向您汇报我的学习收获、进度，以及下周规划。

*一、Bioinformatics Data Skills*阅读

1. Chapter 2

以已学完第二章的视角回顾全章内容，不难总结出本章主旨为：如何高效创建并管理生信项目。

首先聚焦于项目的创建，选择一个简短的、合适的项目名称，并创建一些基本的目录，如data、scripts、analysis等。命名时应注意仅使用字母，数字，下划线和连接号(-)，慎用空格，另外，脚本里尽量使用相对路径而非绝对路径，以便工程移植。

随后谈论记录工程。目标是：很长时间后回顾工程仍能知晓所有细节。需要清晰记录的有：流程与参数、数据来源与版本号、数据下载时间与方式、软件版本号。所有记录文件最好采用纯文本README的形式，且工程的每个主文件夹下都应存放一个README文件，简单方便可移植的同时，能在工程趋于复杂时雪中送炭。

接下来讨论文件命名与匹配。命名文件时最好采取统一的命名风格，且在序号前补空零是很好的习惯，如“0002024”，从而避免排序错误（如1后面不是2而是10）。匹配文件时要用到通配符：“\*”匹配任意长度字符；“?”匹配单个字符；“[]”只对一位字母或者一位数字（即[0-9]）有效，而对多位的数字无效；若未匹配到想要的文件，可采用“{}”尝试。

下面作者介绍了一种工程笔记的记录方式——Markdown。权衡多方观点，我下载了Typora这款工具进行markdown练习。基本格式如：

#后接文字可生成标题，需要几级标题则打几个#号，上限为6级。

\*文字\* 对应斜体。

\*\*文字\*\* 对应粗体。

>文字 对应文字引用。

`文字` 对应代码引用。

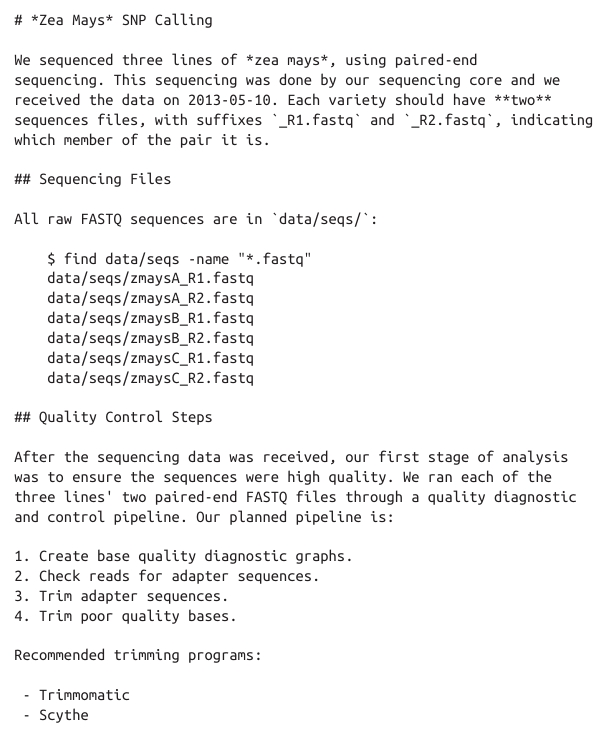
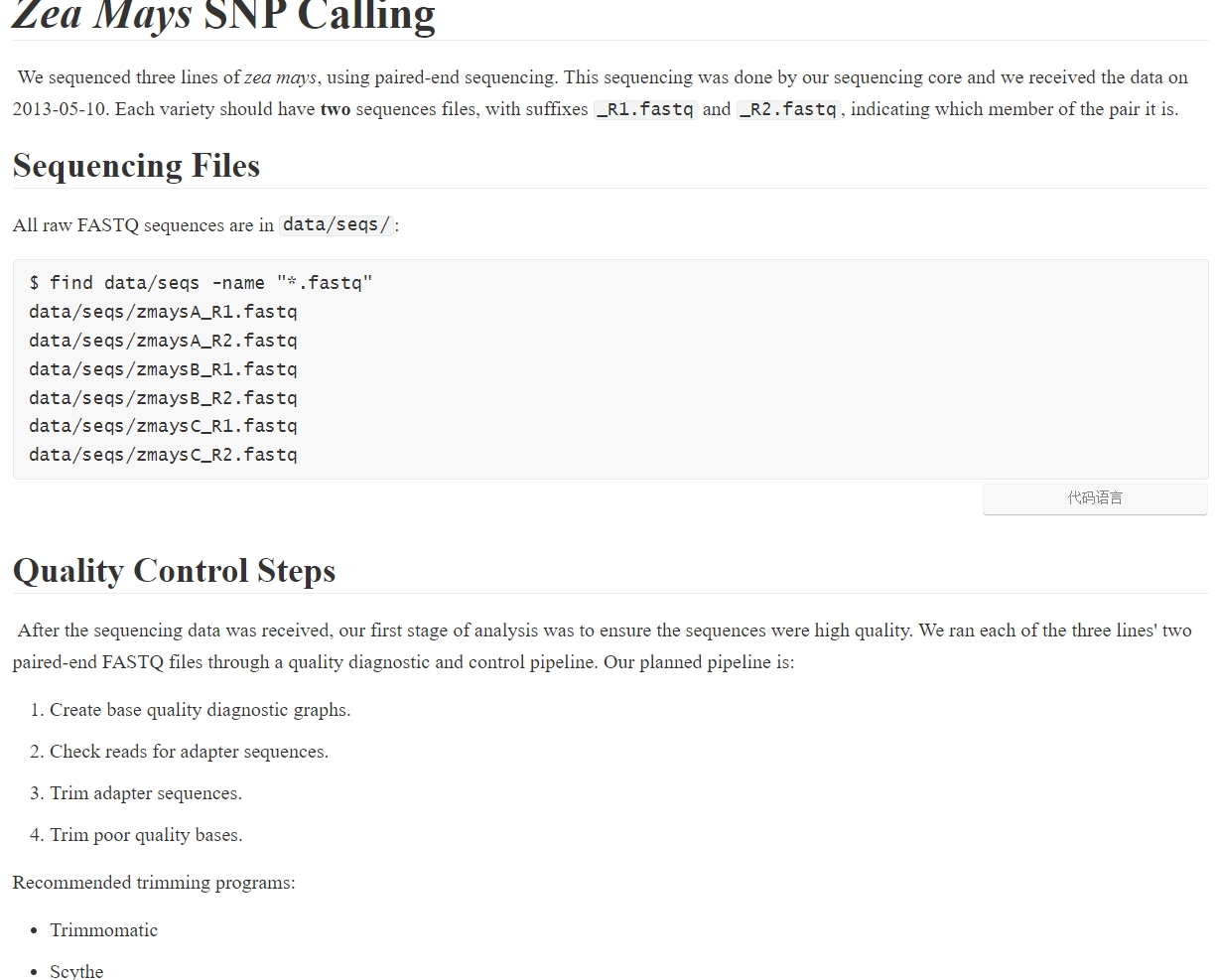
-文字 对应列表。

1.文字 对应有序列表。

[文字](链接) 对应超链接。

![文字](图片链接) 对应图片。

上述为核心格式，另外还有其他特殊格式，篇幅所限在此不列举。作者在文中提供了一大段原始字符作为示例（左图），在Typora中打开即可得到实际语段（右图），如下所示。



紧接着，作者提供了名为Pandoc的格式渲染工具，可将markdown文件导出成多种格式，如html、docx、pdf等等。例如下面这个即为上图文件导出的html。



2. Chapter 3

本章中，作者主要谈论Unix Shell这一工作环境。

Unix Shell具有两大哲学思想：模块化、重定向。模块化指的是一个程序只做好一件事情，然后将程序使用管道连接起来，这样便于定位bug、测试模块、在各模块采用不同工具；重定向指在不读取数据到内存的情况下进行处理，可以节省磁盘与内存空间，增加速度，便于处理文本数据量可能非常大的生物信息。

cat命令（在DOS窗口为type命令）可以将文件内容传输到标准化输出流，可以将单个文件或者多个文件重定向输出，也可合并多个文件内容。重定向符号为>和>>，前者会覆盖已存在的文件内容，后者会在已存在的内容后面追加。两者都会在文件不存在的情况下创建新文件。作者补充了ls -lrt这一语法快捷方式。l表示以列表形式展示；r表示倒序；t表示时间。将r去掉后最新的文件会出现在列表最上方。（DOS窗口中dir即为ls命令，但我没找到可等价lrt的命令）

程序的错误、警告或者其它信息也会直接显示在屏幕上，如果想要记录这些信息作为日志文件，可分别对标准化输出与标准化错误进行重定向，如：

ls 存在.txt 不存在 > 导出.txt 2> 错误.txt

像这样，存在.txt的内容将导入导出.txt，“不存在”对应的标准化错误输出将导入错误.txt。

另外，linux的标准化输入为<，但是这种方式不直观也并不常用，常用的方式是通过“管道”，类似cat inputfile | program > output file。

紧接着，作者就开始介绍起unix管道，称其兼具速度与美学。其作用是将一个程序的输出作为另一个程序的输入，而标准错误会打印在屏幕上。使用管道可以提高运行速度，节约磁盘空间。

作者在此举了一个管道处理文件的例子，在一大堆碱基字母中找出含有非碱基字母的一行。他使用了以下命令：

grep -v "^>" tb1.fasta | \

grep --color -i "[^ACTG]"

grep为常用的文本匹配命令；引号里面是待匹配的正则表达式（加引号是一个好习惯，否则>会被程序理解为重定向）；第一行中^>代表以“>”开头的行；-v表示反向匹配，意思是在tbl.fasta里面寻找非“>”开头的文本； \只是为了换行，来提高可读性；第二行中—color可对匹配结果加颜色，更醒目；-i代表不区分大小写；[^ACTG]中^表示取“非”，意思是不区分大小写地匹配包含非ACTG字符的行。

如果需要将程序的标准错误信息存储为日志文件，可以将管道与重定向结合，如：

program1 inputfile 2> program1.err | program2 2> program2.err > result.txt

如果想将标准出错信息同时重定向到文件中，可以使用“2>&1”：

program1 inputfile | tee intermediatefile.txt | program2 > result.txt

接下来，作者介绍了如何使进程在后台运行，终止进程与查看进程状态的基本操作。想要使程序在后台运行，只需要在命令后面添加&，而shell会返回该程序对应的ID。使用“jobs”命令可以查看后台进程。想要将后台程序重新调到前台，可以根据jobs返回的序号使用“fg %序号”命令（此序号与前面的ID没有关系）。若要终止后台运行的程序，最简单的操作是将后台程序调到前台，然后按ctrl + z组合键。

如果想知晓程序运行是否成功，使用“echo $?”命令可进行确认。0代表成功运行，其它数字代表不同类型的错误。另外，状态码是非常有用的，我们可以将多个程序连接起来。例如如果想要使用program1生成中间结果，然后使用program2读取中间结果作为输入，为了保证program1的结果生成完毕后再执行program2，我们可以采用短路运算符&&（意思是前面的程序返回成功运行的0才运行后面的程序）。与&&相对的是||运算符，意思是前面一个程序非成功运行（返回非0状态）时才运行后面一个程序。如果不论前面是成功还是失败都执行后一个程序的话可以采用“；”连接程序。

后续又出现一个技能，称为命令替换，指将一个行内表达式的运行结果作为另外一个程序的输入。以此可增加脚本的自动化程度，并减少人工错误。下面是例子。

第一例，用于统计FASTA序列数目：

grep -c "^>" example.fasta

# 416

echo " fasta文件里面一共有$(grep -c "^>" example.fasta)"条序列。

# fasta文件里面一共有416条序列。

其中，grep的-c参数代表返回匹配结果的行数；使用$()可以将表达式的运算结果嵌到字符串里面。

第二例，使用日期作为文件后缀：

假设我们因为某种原因需要重新运行程序，但是又不想覆盖前一次的结果。除了对结果进行备份外，在结果后面加上日期也是一个好的选择。人工去加日期的话较麻烦也可能犯错，故可以采用“date +%F”生成日期，嵌在创建结果文件夹的命令中。

总的来说，本周我学习了二三两章，为18~56页，大体符合我每周40页左右的初步规划。

二、python学习

有一个错误需要向您坦白，上周粗略浏览前三章时，我把markdown和unix shell的内容当成了编程语言，因此称学习python是燃眉之急。现在看来好像确实如您所说“可有可无”（笑）。不过目前我已经下载了python并找了网络课程，目前进度在“python语言的简介”。

三、一点问题

关于*Bioinformatics Data Skills*这本书，它指出Unix操作系统是生信领域主流，在学习过程中也能发现作者提供的语法和技能都是基于Unix的。请问在电脑上安装Unix就得先删除windows系统，是这样吗？如果确实是，我目前有必要这么做吗？另外windows的DOS窗口貌似功能不如Unix齐全，怎么更好地模拟书中的指令呢？对此我很犹豫，恳请老师指点。

四、下周规划

鉴于四级和期末即将来临，每周40页左右的规划似乎有些重了，下周尝试修正为35页左右，且看能否缓解压力。

Python的学习可能会缓慢推进，毕竟目前确实并非刚需。

感谢老师阅读我的学习报告，有错误之处还请多多斧正，格式需要改进之处也请不吝赐教，非常感谢！

丁加

2024年12月1日