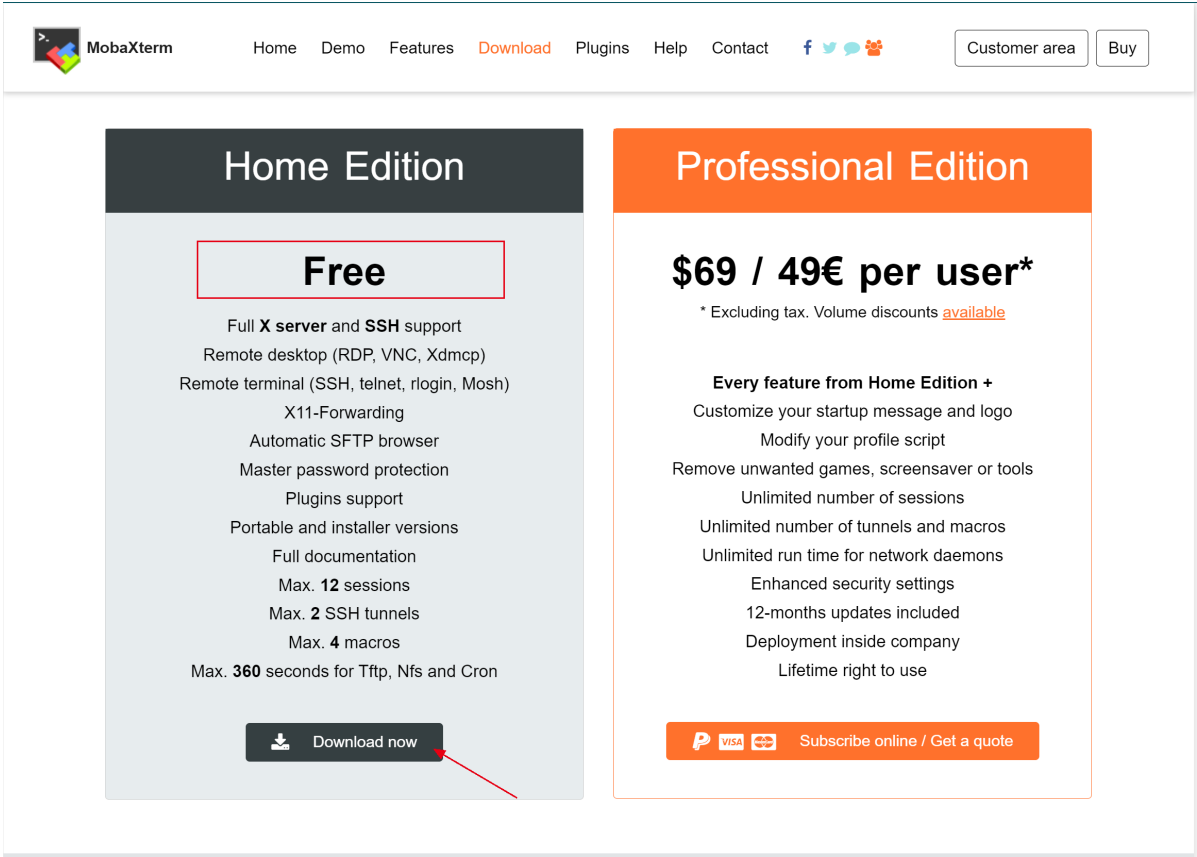


基因组组装挂载注释

远程登录服务器

1. 下载SSH客户端

MobaXterm: <https://mobaxterm.mobatek.net/download.html>

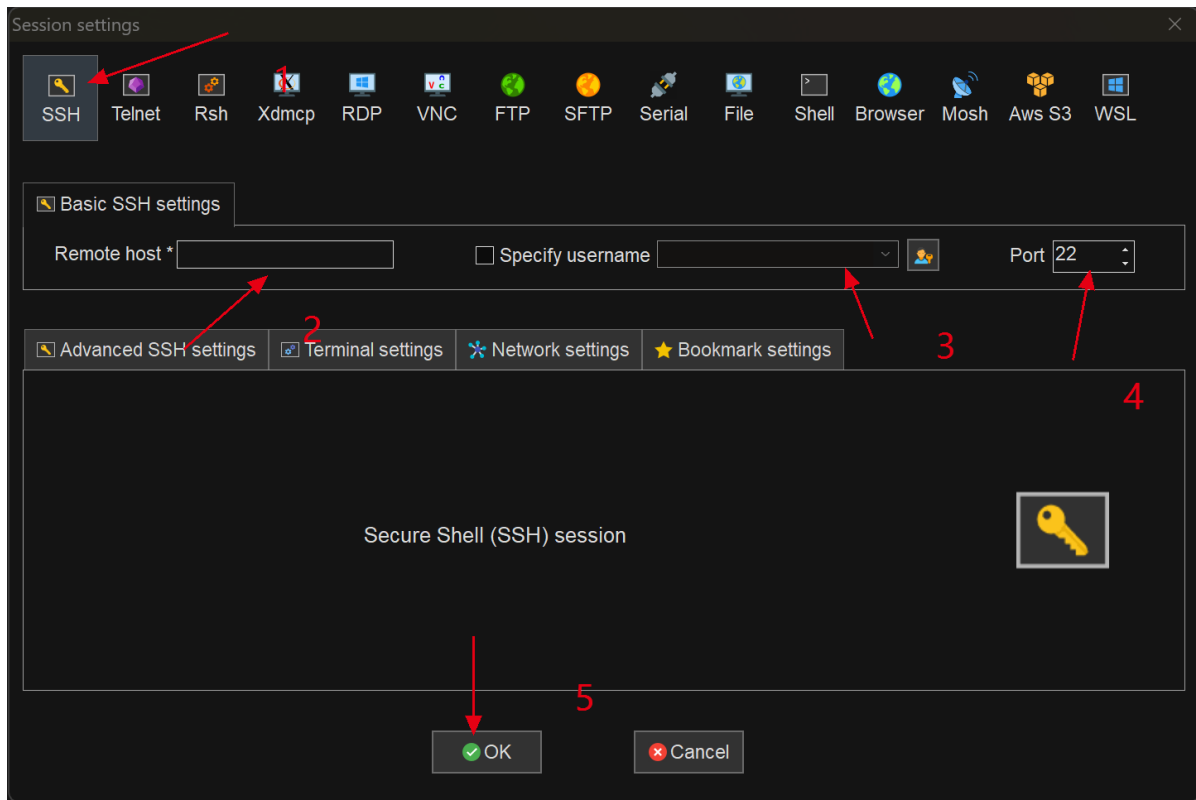


The screenshot shows the MobaXterm website. The navigation bar includes links for Home, Demo, Features, Download, Plugins, Help, and Contact, along with social media icons and buttons for 'Customer area' and 'Buy'. The main content area is divided into two columns. The left column, titled 'Home Edition', features a 'Free' badge and lists features such as Full X server and SSH support, Remote desktop (RDP, VNC, Xdmcp), Remote terminal (SSH, telnet, rlogin, Mosh), X11-Forwarding, Automatic SFTP browser, Master password protection, Plugins support, Portable and installer versions, Full documentation, Max. 12 sessions, Max. 2 SSH tunnels, Max. 4 macros, and Max. 360 seconds for Tftp, Nfs and Cron. A 'Download now' button is at the bottom, with a red arrow pointing to it. The right column, titled 'Professional Edition', shows a price of '\$69 / 49€ per user*' and lists additional features like Customizable startup message, Profile script modification, Removal of unwanted games, Unlimited sessions, Unlimited tunnels and macros, Unlimited run time for network daemons, Enhanced security settings, 12-month updates, Company deployment, and Lifetime right to use. A 'Subscribe online / Get a quote' button is at the bottom.

和普通软件一样安装到自己的电脑上就好。

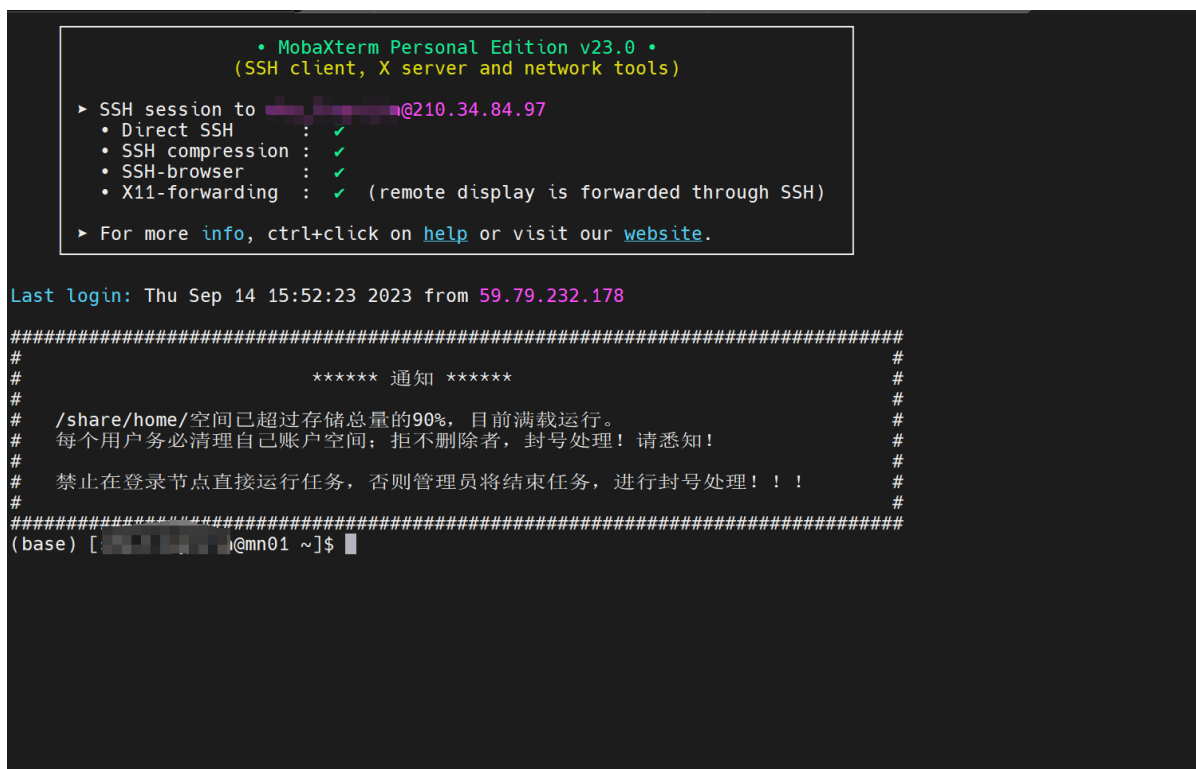
2. 登录服务器

打开软件：选择左上角的[Sessions]--->[New session]



选择 SSH 连接，分别在2输入服务器的域名,例如 210.34.81.54；在3输入服务器的登录名，例如 off_zhangsan;在第四步为端口选择，默认22端口，不变。最后点击 OK。

提示输入密码，正常输入密码回车即可，在linux界面中，输入密码的行为，不会在页面上显示任何东西。



看到这样的画面即服务器登录成功！可以开始使用，开始下面的操作。

Linux基础

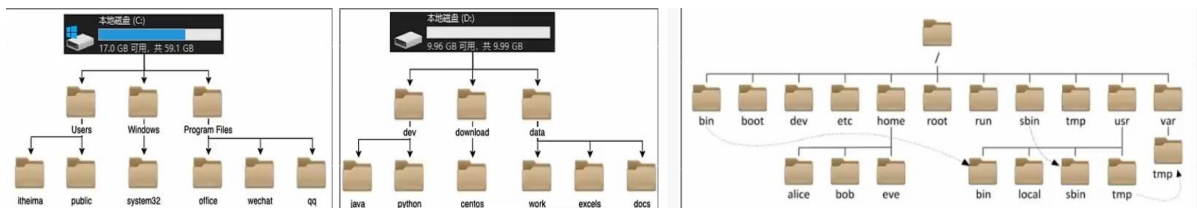
Linux的目录结构

Windows系统：

- (1) 多个并列的树状结构
- (2) 拥有多个盘符，如C盘、D盘、E盘。
- (3) 路径之间的层级关系，使用：\来表示，如：D: \data\work\hello.txt

Linux系统：

- (1) 是一个树状结构
- (2) 没有盘符的概念，只有一个根目录/，所有文件都在它下面
- (3) 路径之间的层级关系，使用：/来表示，如：/user/local/hello.txt



Linux基础命令

1.Linux命令基础格式：

command [-options] [parameter]

• command：命令本身

-options：[可选，非必填]命令的选项，控制命令的行为细节

parameter：[可选，非必填]命令的参数，多数用于命令的指向目标

注：语法中的[]表示可选的意思；命令、选项、参数之间用空格隔开。

示例

```
ls -l /home/work #ls是命令本身，-l是选项， /home/work是参数
```

目录处理命令

ls、mkdir、pwd、cd、cp、mv、rm、chmod

文件处理命令

内容1：touch、cat、more、less、head、tail、sort、uniq、cut、wc

文本编辑器

vim

压缩文件命令

zip&unzip、gzip&gunzip、tar

2. Linux常用命令—目录处理命令

ls命令

格式：命令 [选项] [参数]

选项：-a 列出目录下的所有文件，包括以 . 开头的隐含文件。

-l 以长格式的形式查看当前目录下可见文件的详细属性

-t 以时间排序。

-h 以容易理解的格式列出文件大小，例如K、M、G

注：当不使用选项和参数，直接使用ls，表示：以平铺形式，列出当前工作目录下的内容。

home目录：

(1) 是每个Linux用户在Linux系统的个人账户目录，路径：/home/用户名

(2) Linux系统的命令终端在启动的时候会默认进入home目录

mkdir命令

语法：mkdir [选项] 参数

选项：-p 递归创建

```
#创建一个biosoft的文件夹
mkdir biosoft
#创建一个workspace, 并且在workspace下再创建一个gene_family
mkdir -p workspace/gene_family
```

pwd命令

语法：pwd

cd命令

语法：cd [参数]

用法：

(1) cd 命令不加选项参数，表示进入用户家目录，cd 即 cd ~

(2) cd 命令只写参数，表示进入相应的目录

(3) 绝对路径：以根目录为起点来描述路径，以/开头

相对路径：以当前目录为起点来描述路径，无需以/开头。 .. 表示上级目录 . 表示当前目录。

```
#进入hisat2-2.1.0文件夹
cd biosoft/hisat2-2.1.0/
#返回上一级，到biosoft文件夹
cd ../
```

mv命令

语法：mv 参数

描述：剪切，可改名

rm命令

语法：rm [选项] [参数]

选项：-r 删除目录

cp命令

语法: cp [选项] 参数 原文件或目录可以累加

选项: -r 复制目录

-L 复制链接指向的文件

描述: 复制过程中可以改名字

touch命令

语法: touch 参数

描述: 创建一个新文件

• cat命令 适合查看短文件, 合并文件

语法: cat [选项] 参数

选项: -n 显示行号

-A 显示所有字符

cat 文件1 文件2 > 文件3 按行合并文件

paste 命令 按列合并文件

语法: paste [选项] 参数

选项: -d 指定分隔符

less命令 适合长文件

语法: less [文件名]

f 或空格向下翻页

page up键向上翻页

回车 向下换行

↑ 键向上换行

q或Q 退出

描述: 可以向上翻页, 退出后不显示文件内容。

#一般使用, 表示不换行且展示行号打开

```
less -SN file.txt
```

ln命令

语法: ln 选项 参数

选项: -s 软链接

描述: 链接时可以改名

软链接特点:

(1) 文件类型是 l

(2) 权限为rwxrwxrwx, 但权限取决于源文件

(3) 类似Windows的快捷方式, 链接指向原文件。如果源文件删除, 软链接无法访问

(4) 链接可以指向绝对路径, 也可以指向相对路径

sort命令

语法: sort [选项] 参数

选项: -t 指定分隔字符

-k 按指定的列进行排序

-r 默认是升序排列, 加上-r就是降序

- o 将排序后的结果存入指定的文件
- n 依照数值的大小排序

uniq命令

语法: uniq [选项] 参数

选项: -c 在每列旁边显示该行重复出现的次数

补充: 当重复的行并不相邻时, uniq 命令是不起作用的, 所以一般都与sort命令一起用。

例如: sort -k2nr 文件|uniq

cut命令

语法: cut 选项 参数

选项: -f 指定显示的字符段

-d 自定义分隔符

补充:

(1) cut -f3,2,1 与cut -f1,2,3输出的字符段顺序是一样的

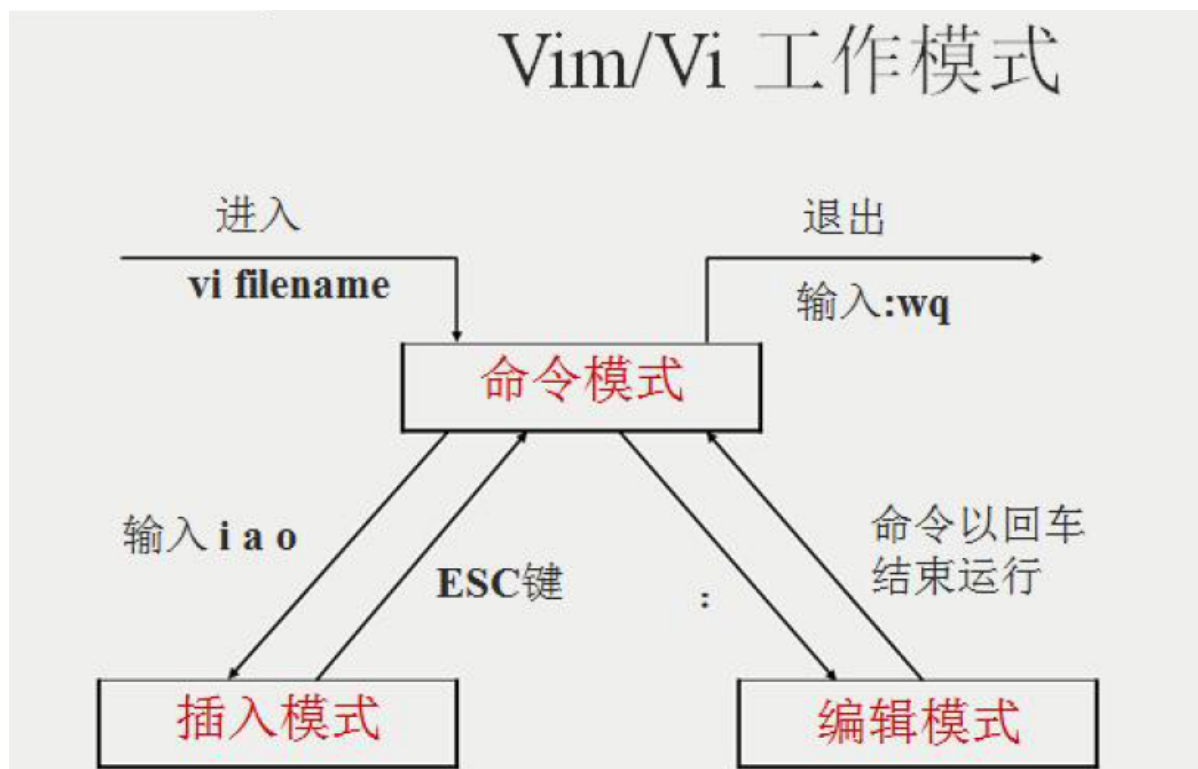
(2) 当分割符为tab键、逗号、分号等符号时可正常使用, 但当文件的分割符为空格时, 无法使用cut

wc命令

语法: wc[选项参数] [文件名]

选项: -l 显示行数

vim命令



插入命令:

- a 在光标后附加文本
- A 在本行行末附加文本
- i 在光标前插入文本
- I 在本行开始插入文本
- o 在光标下插入新行
- O 在光标上插入新行

定位命令:			
h、方向左键	保存和退出命令:		
j、方向下键	:w 保存修改		
k、方向上键	:wnew_filename 另存为指定文件		
\$. 移至行尾	:wq 保存修改并退出		
O、移至行首	ZZ 快捷键, 保存修改并退出		
:set nu 设置显示行号	:q! 不保存修改退出		
:set nonu 取消显示行号	:wq! 保存修改并退出 (文件所有者 可忽略文件的只读属性)		
G 到最后一行			
gg 到第一行			

服务器配置

一般我们拿到一台服务器之后, 里面都是空的, 什么软件都没有, 就需要我们去配置 服务器。为了规范化配置和使用服务器, 我们一般都是在自己的家目录下进行操作。

1.在家目录位置创建关键性文件夹

一般登录服务器后就是出于家目录下, 或者可以使用 `cd ~` 进入

```
#存储软件安装包的文件夹
mkdir software
#安装软件包的文件夹
mkdir biosoft
#工作目录
mkdir workspace
```

2. base 环境部署

我们都会在服务上面安装一个类似于手机应用商店的软件, 前几年使用 conda 分为 miniconda 和 anaconda 两种, 区别在于 miniconda 只是一个应用商店而 anaconda 是内置了许多软件的, 我们用的话 miniconda 就够用了。conda 软件是基于 python 语言开发的, 使用一段时间或者安装的软件较多后就会出现卡和慢的现象。所以现在推出了 C + 语言 编写的 mamba 软件。mamba 和 conda 是一样的使用效果, 也同样有两种, 我们安装 micromamba 即可。

```
#mamba官网
https: /github.com/conda-forge/miniforge#mamba
#mambaforge下载
wget -c http: /ftp.genek.cn:8888/Share/linux_software/Mambaforge-Linux-x86_64.sh
-P ~/software/
#mamba安装
bash ~/software/Mambaforge-Linux-x86_64.sh -b
#初始化
./mambaforge/bin/mamba init
#加载环境变量，加载后名称前出现（base）即成功
source ~/.bashrc
#.bashrc这个文件是非常重要的一个软件，后面的手动安装的软件都需要在里面进行添加路
径，他相当于自己服务器的软件目录快捷键
```

给 mamba 添加镜像源，这样下载软件就是使用国内镜像

```
vim ~/.condarc

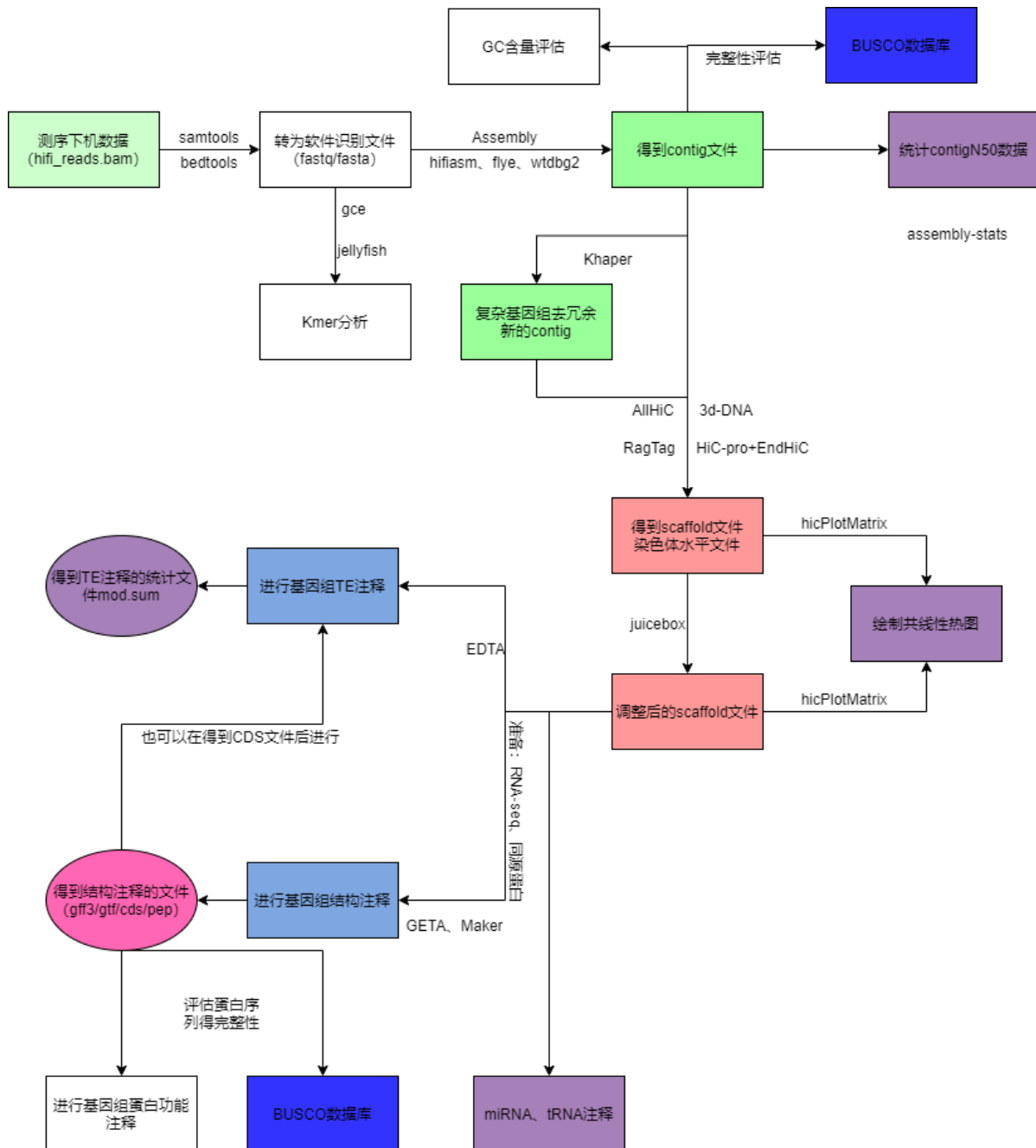
channels:
- https: /mirrors.ustc.edu.cn/anaconda/pkgs/main/
- https: /mirrors.ustc.edu.cn/anaconda/cloud/conda-forge/
- https: /mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
- defaults
show_channel_urls: true
```

3.基础软件的安装

```
mamba install gxx_linux-64 biopython fastqc fastp blast samtools -y
mamba install bwa bedtools seqtk seqkit -y
mamba install diamond hmmer trf minimap2 bowtie bowtie2 star -y
mamba install assembly-stats spaln -y
mamba install vcftools hisat2 subread htseq gffread -y
```

这样的话 生物信息 的基础软件我们已经配置好了！

基因组组装



1.软件的安装

```
#####
#以下的软件安装都需要将/share/home/off/替换为自己的目录 #
#####
用`pwd`查看自己所在的位置

# Installing GCE (https://arxiv.org/abs/1308.2012
|ftp://ftp.genomics.org.cn/pub/gce)下载到software
wget ftp://ftp.genomics.org.cn/pub/gce/gce-1.0.0.tar.gz -P ~/software/
#解压到biosoft
tar xzf ~/software/gce-1.0.0.tar.gz -C ~/biosoft/
echo "PATH=/share/home/off/biosoft/gce-1.0.0:$PATH" >> ~/.bashrc

# Installing Genomescope 2.0 + jellyfish
#(https://github.com/tbenavi1/genomescope2.0) and Jellyfish
(https://github.com/gmarcais/Jellyfish)
```

```

wget https://github.com/gmarcais/Jellyfish/releases/download/v2.3.0/jellyfish-2.3.0.tar.gz -P ~/software/
tar xzf ~/software/jellyfish-2.3.0.tar.gz -C ~/biosoft/
cd ~/biosoft/jellyfish-2.3.0/
./configure --prefix=~/biosoft/jellyfish-2.3.0
make -j 8 && make install
echo "PATH=/share/home/off/biosoft/jellyfish-2.3.0/bin:$PATH" >> ~/.bashrc

##Installing hifiasm
cd ~/biosoft
git clone https://github.com/chhy1p123/hifiasm
cd hifiasm && make
echo "export PATH=/share/home/off/biosoft/hifiasm:$PATH" >> ~/.bashrc

# Installing Canu (https://canu.readthedocs.io/en/latest/|https://github.com/marbl/canu/releases)
wget https://github.com/marbl/canu/releases/download/v2.1.1/canu-2.1.1.tar.xz -P ~/software/
tar Jxf ~/software/canu-2.1.1.tar.xz -C ~/biosoft
cd ~/biosoft/canu-2.1.1/src/
make -j 8
echo "export PATH=/share/home/off/biosoft/canu-2.1.1/build/bin:$PATH" >> ~/.bashrc

# install NextDenovo (https://github.com/Nextomics/NextDenovo)
wget https://github.com/Nextomics/NextDenovo/releases/download/v2.4.0/NextDenovo.tgz -P ~/software/
tar xzf ~/software/NextDenovo.tgz -C ~/biosoft
echo "export PATH=/share/home/off/biosoft/NextDenovo:$PATH" >> ~/.bashrc
echo "export PATH=/share/home/off/biosoft/NextDenovo/bin:$PATH" >> ~/.bashrc

##Installing verkko
mamba install -c conda-forge -c bioconda -c defaults verkko -y
##Installing flye
mamba install -c conda-forge -c bioconda -c defaults flye -y

#Installing yahs
mamba create -n hic-scaffolding chromap samtools yahs samtools assembly-stats openjdk

#最后要加载环境变量
source ~/.bashrc

```

2.Kmer分析

```
hifi_read=  
name=  
kmer=21  
mkdir 00.kmer && cd 00.kmer  
jellyfish count <(zcat ${hifi_read} ) -m ${kmer} -s 100M -t 10  
jellyfish stats mer_counts.jf -o kmer_coun  
jellyfish histo -v -t 10 -h 10000 mer_counts.jf > kmer.histo  
Rscript genomescope.R -i kmer.histo -o ${name} -k ${kmer} -p 3 -n ${name}
```

3.基因组组装

1.hifiasm组装

a.有HIFI无HiC数据模式

```
mkdir 01.hifiasm && cd 01.hifiasm  
hifi_fq=  
name=  
threads=  
hifiasm -o ${name} -t ${threads} ${hifi_fq} 2>${name}.log  
awk '/^S/{print ">"$2;print $3}' ${name}.bp.p_ctg.gfa >  
${name}.bp.p_ctg.fa 2>2.log  
awk '/^S/{print ">"$2;print $3}' ${name}.bp.hap1.p_ctg.gfa >  
${name}.bp.hap1.p_ctg.fa 2>2.log  
awk '/^S/{print ">"$2;print $3}' ${name}.bp.hap2.p_ctg.gfa >  
${name}.bp.hap2.p_ctg.fa 2>2.log
```

b.有HIFI数据和HiC数据

```
mkdir 01.hifiasm && cd 01.hifiasm  
hifi_fq=  
hic_fq1=  
hic_fq2=  
name=  
  
hifiasm -o ${name}.asm -t ${threads} --h1 ${hic_fq1} --h2  
${hic_fq2} ${hifi_fq} 2>${name}.log  
awk '/^S/{print ">"$2;print $3}' ${name}.asm.hic.p_ctg.gfa >  
${name}.asm.hic.p_ctg.fa 2>2.log  
awk '/^S/{print ">"$2;print $3}' ${name}.asm.hic.hap1.p_ctg.gfa >  
${name}.asm.hic.hap1.p_ctg.fa 2>2.log  
awk '/^S/{print ">"$2;print $3}' ${name}.asm.hic.hap2.p_ctg.gfa >  
${name}.asm.hic.hap2.p_ctg.fa 2>2.log
```

c.参数解析

- o : 输出文件前缀
- t : 调用线程数目
- h1 : hic一端数据
- h2 : hic另一端数据

2.verkko组装

a.有HIFI无HiC数据模式

```
hifi_fq=  
name=  
threads=  
verkko -d 02.verkko.${name} \  
    --hifi ${hifi_fq} \  
    --threads ${threads} 2>${name}.log
```

b.有HIFI数据和HiC数据

```
hifi_fq=  
hic_fq1=  
hic_fq2=  
name=  
threads=  
verkko -d 02.verkko.${name} \  
    --hifi ${hifi_fq} \  
    --hic1 ${hic_fq1} \  
    --hic2 ${hic_fq2} \  
    --threads ${threads} 2>${name}.log
```

c.参数解析

- d : 输出文件夹名称
- hifi : HiFi数据
- threads : 调用线程数目
- hic1 : hic一端数据
- hic2 : hic另一端数据

3.canu组装

```
##HiFi数据模式  
hifi_fq=  
name=  
size=  
canu -p ${name} -d 03.canu.${name} maxThreads=20\  
    genomeSize=${size} \  
    "batOptions=-eg 0.0 -sb 0.001 -dg 0 -db 3 -dr 0 -ca 200 -cp 200" \  
    -pacbio-hifi $hifi 2>canu.log  
  
##参数解释  
-p : 输出文件前缀
```

-d : 输出文件夹名称
maxThreads : 调用的最大线程, 可并行
genomeSize : 预估的基因组大小 (单位: m, g)
-pacbio-hifi : canu的HiFi数据模式

4.wtdbg2组装

```
hifi_fq=  
name=  
size=  
threads=  
mkdir 04.wtdbg2 && cd 04.wtdbg2  
wtdbg2 -t ${threads} -x ccs \  
    -g ${size} -L 2000 -l 2048 \  
    -e 3 -i ${hifi_fq} -o ${name}  
  
wtpoa-cns -t 20 -i ${name}.ctg.lay.gz \  
    -f -o ${name}.wtpoa-cns.fa
```

##参数解释

-x : 调用数据类型
-g : 预估的基因组大小 (单位: m, g)
-L : 选择最长的子读并删除比它短的子读
-l : 最小对准长度
-e : Min read depth of a valid edge, [3]
-i : 输入文件
-o : 输出文件前缀

5.nextDenovo组装

```
hifi_fq=  
name=  
size=  
threads=  
  
mkdir 05.nextDenovo && cd 05.nextDenovo  
  
echo "  
[General]  
job_type = pbs # local, slurm, sge, pbs, lsf  
job_prefix = nextDenovo  
task = all # all, correct, assemble  
rewrite = yes # yes/no  
deltmp = yes # 删除临时文件  
parallel_jobs = 2 # 并行任务数, 默认10 建议: 内存/(32~64G)  
input_type = raw # raw, corrected  
read_type = hifi # clr, ont, hifi  
input_fofn = input.fofn # reads文件路径列表  
workdir = 05.nextDenovo.${name}  
[correct_option]  
read_cutoff = 2k  
genome_size = ${size} # estimated genome size  
pa_correction = 4 # correct步骤并行任务数, 内存/C数据量
```

```

*1.2/4)
sort_options = -m 2g -t 5 # -m 数据量*1.2/4g -t 总线程
数/pa_correction
minimap2_options_raw = -t 5 # 总线程数 P/parallel_jobs
correction_options = -p 5 # 总线程数 P/pa_correction
[assembly_option]
minimap2_options_cns = -t ${threads} # 总线程数 -t P/parallel_jobs
nextgraph_options = -a 1 # 输出fasta格式
" >run.cfg

#ont数据路径输出到文件
echo $hifi >input.fofn

#####
#运行命令
#####
nextDenovo run.cfg

```

6.flye组装

```

hifi_fq=
name=
size=
threads=
flye --pacbio-hifi ${hifi_fq} --out-dir ./06.flye.${name} --genomesize \
    ${size} --threads ${threads}

##参数解释
--pacbio-hifi : 使用HiFi数据
--out-dir : 输出路径及文件夹
--genome-size: 预估基因组的大小
--threads : 调用的线程

```

基因组去冗余

软件安装

```

#Installing khaper
wget https://github.com/lardo/khaper/archive/refs/heads/khapermaster.zip -P
~/software/
cd biosoft && unzip ~/software/khaper-master.zip
mv khaper-master khaper
cd khaper/Bin && chmod 755 *
echo "export PATH=/share/home/off/biosoft/khaper/Bin:$PATH" >> ~/.bashrc

#Installing purge_dups
cd ~/biosoft
git clone https://github.com/dfguan/purge_dups.git
cd purge_dups/src && make
echo "export PATH=/share/home/off/biosoft/purge_dups/bin:$PATH" >> ~/.bashrc
cd ~/biosoft
git clone https://github.com/dfguan/runner.git
cd runner && python3 setup.py install --user ##会自动添加到快捷方式

```

1.khaper

```
##Prepare input files
HiFi_path=
contig=

##Build the kmer frequency table
ls ${HiFi_path}/*.gz > fq.list
perl Graph.pl pipe -i fq.list -m 2 -k 17 -s 1,3 -d Kmer_17
##Compress the assembly file
#perl ~/biosoft/khaper/Bin/remDup.pl ${contig} --kbit
Kmer_17/02.Unique_bit/kmer_17.bit --kmer 17 Compress 0.3
##0.3为设置数值，默认的cutoff为0.7，我们可以先依次设置0.6、0.5、0.4再看最后的结果
##Compress为输出文件夹的名称，可自行修改
perl Graph.pl ${contig} --kbit Kmer_17/02.Unique_bit/kmer_17.bit --kmer 17
cutoff_0.6 0.6
perl Graph.pl ${contig} --kbit Kmer_17/02.Unique_bit/kmer_17.bit --kmer 17
cutoff_0.5 0.5
perl Graph.pl ${contig} --kbit Kmer_17/02.Unique_bit/kmer_17.bit --kmer 17
cutoff_0.4 0.4
```

2.purge_dugs

```
#!/bin/bash
mkdir Purge_Dups
cd Purge_Dups
##1.数据准备
contig=
ln -s ${contig} pri_asm.fa
pri_asm=pri_asm.fa
minimap2 -xasm20 -t 10 $pri_asm $hifi | gzip -c - > hifi.paf.gz
pbcstat hifi.paf.gz #(produces PB.base.cov and PB.stat files)
calcuts PB.stat > cutoffs 2>calcuts.log
split_fa $pri_asm > $pri_asm.split
minimap2 -x asm5 -DP $pri_asm.split $pri_asm.split | gzip -c - >
$pri_asm.split.self.paf.gz
purge_dups -2 -T cutoffs -c PB.base.cov $pri_asm.split.self.paf.gz > dups.bed 2>
purge_dups.log
get_seqs -e dups.bed $pri_asm
```

基因组挂载

软件安装

```
##Installing ALLHiC
cd ~/biosoft
git clone https://github.com/tangerzhang/ALLHiC
cd ALLHiC
chmod +x bin/*
chmod +x scripts/*
echo "export PATH=/share/home/off/biosoft/ALLHiC/bin:$PATH" >> ~/.bashrc
echo "export
```

```

PATH=/share/home/off_wenhao/biosoft/ALLHiC/scripts:$PATH" >> ~/.bashrc

##Installing HiC-Pro (这个比较难安装)
cd ~/biosoft
git clone https://github.com/nservant/HiC-Pro.git
cd HiC-Pro
mamba env create -f environment.yml -p ~/mambaforge/envs/hic-pro
mamba activate ~/mambaforge/envs/hic-pro
cd ~/biosoft/HiC-Pro
make configure && make install
#测试是否成功安装
$ ./bin/HiC-Pro
    usage : HiC-Pro -i INPUT -o OUTPUT -c CONFIG [-s ANALYSIS_STEP] [-p] [-h] [-v]
    Use option -h|--help for more information
echo "export PATH=/share/home/off/biosoft/HiC-Pro_3.1.0/bin:$PATH" >> ~/.bashrc

#Installing EndHiC
git clone git@github.com:fanagislab/EndHiC.git
echo "export PATH=/share/home/off/biosoft/EndHiC:$PATH" >> ~/.bashrc

#Installing chromap + yahs
mamba create -n hic-scaffolding -c bioconda -c conda-forge chromap samtools yahs
samtools assembly-stats openjdk

```

1. AllHiC挂载

```

#数据变量
name=
hic_fq1=
hic_fq2=
contig=
#设置变量
threads=
enzyme=
K=11#染色体条数

mkdir 02.allhic && cd 02.allhic
darft=${contig}
#1.将contig数据软链接到现在的目录下
ln -s $darft ./darft.asm.fasta

#2.对原始contig构建索引
bwa-mem2 index -p darft.asm.fasta darft.asm.fasta

#3.将HiC数据比对到原始contig
bwa-mem2 mem -SP5M -t ${threads} darft.asm.fasta ${hic_fq1} ${hic_fq2} | \
    sambamba view /dev/stdin -S -f bam -t ${threads} | \
    sambamba sort /dev/stdin -t ${threads} -o sorted.bam

#4.correct contig
ALLHiC_corrector -m sorted.bam -r darft.asm.fasta -o seq.HiCcorrected.fasta -t
${threads}

#5.对矫正后的contig构建索引

```



```

bwa-mem2 index -p seq.HiCcorrected.fasta seq.HiCcorrected.fasta

#6. 将HiC数据比对到矫正后contig
bwa-mem2 mem -SP5M -t ${threads} seq.HiCcorrected.fasta ${hic_fq1} ${hic_fq2} | \
    sambamba view /dev/stdin -S -f bam -t ${threads} | \
    sambamba sort /dev/stdin -t ${threads} -o sample.bwa_mem.bam

#7. filter bam
samtools view -bq ${threads} sorted.bam | \
    samtools view -bt seq.HiCcorrected.fasta.fai > sample.unique.bam

reprocessSAMS.pl sample.unique.bam seq.HiCcorrected.fasta ${enzyme}

#8. partition
ALLHiC_partition -r seq.HiCcorrected.fasta -e ${enzyme} -k ${K} -b
sample.unique.REFduced.paired_only.bam

#9. optimize
for k in `seq 1 $K`
do
    allhic optimize
    sample.unique.REFduced.paired_only.counts_${enzyme}.${K}g${K}.txt
    sample.unique.REFduced.paired_only.clm
done

#10. build
ALLHiC_build seq.HiCcorrected.fasta

```

2.hicPro+EndHiC挂载

a.hicpro

```

mkdir 01.hicpro
#在工作目录下创建一个rawdata的文件夹
mkdir -p rawdata/sre && cd rawdata/sre
ln -s /YOU_PATH/*.hic.fastq.gz ./

#通过digest_genome.py和HiC的内切酶生成bed文件
~/biosoft/HiC-Pro/bin/utils/digest_genome.py -r HINDIII -o HINDIII.bed contig.fa

#统计contig长度
getChrLength.py contig.fa > contig.size

#构建contig的bowtie2的索引文件
bowtie2-build --threads 20 contig.fa contig.fa

##配置文件修改
echo "
# Please change the variable settings below if necessary
#####
#####
## Paths and Settings - Do not edit !
#####
#####
TMP_DIR = tmp #默认

```

```

LOGS_DIR = logs #默认
BOWTIE2_OUTPUT_DIR = bowtie_results #默认
MAPC_OUTPUT = hic_results #默认
RAW_DIR = rawdata #默认
#####
####
## SYSTEM AND SCHEDULER - Start Editing Here !!
#####
####
N_CPU = 20 #设置cpu使用量
LOGFILE = hicpro.log #设置日志文件
JOB_NAME = contig #设置任务名称
JOB_MEM = 128g #设置需要的内存
JOB_WALLTIME = 200000 #默认
JOB_QUEUE = all.q #默认
JOB_MAIL = *****@163.com #默认,或设置自己的邮箱
#####
#####
## Data
#####
#####
PAIR1_EXT = _1
PAIR2_EXT = _2
#####
####
## Alignment options
#####
####
FORMAT = phred33 #默认
MIN_MAPQ = 0 #默认
BOWTIE2_IDX_PATH
=/share/home/work/Genome_assembly/Assembly_hifi_hic/09.EndHiC/01.hicpro #设置工作目录
BOWTIE2_GLOBAL_OPTIONS = --very-sensitive -L 30 --score-min L,-0.6,-0.2 --end-to-end --reorder #默认
BOWTIE2_LOCAL_OPTIONS = --very-sensitive -L 20 --score-min L,-0.6,-0.2 --end-to-end --reorder #默认
#####
####
## Annotation files
#####
####
REFERENCE_GENOME =contig.fa #contig文件
GENOME_SIZE =
/share/home/work/Genome_assembly/Assembly_hifi_hic/09.EndHiC/01.hicpro/contig.size #前期准备的统计文件
#####
####
## Allele specific analysis
#####
####
ALLELE_SPECIFIC_SNP = #默认
#####
####
## Digestion Hi-C
#####

```

```

####
GENOME_FRAGMENT =
/share/home/work/Genome_assembly/Assembly_hifi_hic/09.EndHic/01.hic
pro/HINDIII.bed #前期准备的bed文件
LIGATION_SITE = AAGCTT #设置酶切序列
MIN_FRAG_SIZE = 100 #默认
MAX_FRAG_SIZE = 100000 #默认
MIN_INSERT_SIZE = 100 #默认
MAX_INSERT_SIZE = 600 #默认
#####
####
## Hi-C processing
#####
####
MIN_CIS_DIST = #默认
GET_ALL_INTERACTION_CLASSES = 1 #默认
GET_PROCESS_SAM = 0 #默认
RM_SINGLETON = 1 #默认
RM_MULTI = 1 #默认
RM_DUP = 1 #默认
#####
####
## Contact Maps
#####
####
BIN_SIZE = 100000 500000 #默认
MATRIX_FORMAT = upper #默认
#####
####
## Normalization
#####
####
MAX_ITER = 100 #默认
FILTER_LOW_COUNT_PERC = 0.02 #默认
FILTER_HIGH_COUNT_PERC = 0 #默认
EPS = 0.1 #默认
" > config-hicpro.txt

```

```
HiC-Pro -i rawdata -o dlo_outdir_new -c config-hicpro.txt
```

#参数解释

#-i 添加rawdata文件夹

#-o 输出文件夹名称

#-c 配置文件

b.hicpro

```
mkdir 02.endhic
```

```
cat work.sh
```

```
contig=
```

##contig文件，一定要和HiC-Pro中的contig保持一致

name= ##物种名称，也要和HiC-Pro设置的保持一致，也就是hic-pro的输出文件夹`*_outdir_new`

```

##get contig length
perl fastaDeal.pl -attr id:len ${contig} > contigs.fa.len

##draw contig Hi-C heatmaps with 10*100000 (1-Mb) resolution
#更换自己的hicpro文件结果的所在位置
hic_pro_dir=/share/home/off/Work/Genome_assembly/Assembly/08.EndHiC/\
    01.hicpro/${name}_outdir_new/hic_results/matrix/${name}

matrix2heatmap.py ${hic_pro_dir}/raw/100000/${name}_100000_abs.bed \
    ${hic_pro_dir}/raw/100000/${name}_100000.matrix 10

##Run one round, when the contig assembly is quite good
perl endhic.pl contigs.fa.len \
    ${hic_pro_dir}/raw/100000/${name}_100000_abs.bed \
    ${hic_pro_dir}/raw/100000/${name}_100000.matrix \
    ${hic_pro_dir}/iced/100000/${name}_100000_iced.matrix

ln Round_A.04.summary_and_merging_results/z.EndHiC.A.results.summary.cluster* ./

##convert cluster file to agp file
perl cluster2agp.pl
Round_A.04.summary_and_merging_results/z.EndHiC.A.results.summary.cluster \
    contigs.fa.len > scaffolds.agp
##get final scaffold sequence file
perl agp2fasta.pl scaffolds.agp ${contig} > ${name}.scaffolds.fa

##draw HiC heatmaps for scaffolds with 10*100000 (1-Mb) resolution
cluster2bed.pl ${hic_pro_dir}/raw/100000/${name}_100000_abs.bed \

Round_A.04.summary_and_merging_results/z.EndHiC.A.results.summary.cluster\
    > clusterA_100000_abs.bed 2> clusterA.id.len

matrix2heatmap.py clusterA_100000_abs.bed
${hic_pro_dir}/raw/100000/${name}_100000.matrix 10

```

真正需要的就只有 scaffolds.agp 和 prefix.scaffolds.fa 两个，一个是scaffold文件，一个是map文件。

3.chromap + yahs挂载

```

contigsFasta=
hic_fq1=
hic_fq2=

# 建立索引
samtools faidx $contigsFasta
chromap -i -r $contigsFasta -o contigs.index
# alignment
chromap --preset hic -r $contigsFasta -x contigs.index --remove-pcr-duplicates \
    -1 $hic_fq1 -2 $hic_fq2 -o aligned.sam --SAM -t 40

# 排序
samtools view -bh aligned.sam | samtools sort -@ 40 -n > aligned.bam
rm aligned.sam

```

```

yahs $contigsFasta aligned.bam

juicer pre -a -o out_JBAT yahs.out.bin yahs.out_scaffolds_final.agp
$contigsFasta.fai
# -o out_JBAT表示输出文件名的前缀

#out_JBAT.txt就作为下游的输入
wget
https://github.com/aidenlab/Juicebox/releases/download/v2.20.00/juicer_tools.2.20.00.jar
wget
https://s3.amazonaws.com/hicfiles.tc4ga.com/public/juicer/juicer_tools_1.19.02.jar

asm_size=$(awk '{s+=$2} END{print s}' $contigsFasta.fai)
java -Xmx20G -jar juicer_tools.2.20.00.jar pre out_JBAT.txt out_JBAT.hic <(echo
"assembly ${asm_size}")
post -o out_JBAT out_JBAT.review.assembly out_JBAT.liftover.agp $contigsFasta

```

4.基因组在juicebox中调整

```

#将ctg文件和agp文件进行*.hic和*.assembly文件转换
hic1=
hic2=
contig=
agp=

chromap -i -r $contig -o index 2>1.log
chromap --preset hic -t 40 -x index -r $contig -1 ${hic1} -2 ${hic2} -o aln.pairs
2>2.log

grep -v '#' aln.pairs |awk '{if($6!="+") $6=16; else $6=0; if($7!="+") $7=16;
else $7=0} \
    $2<=$4{print $6, $2, $3, 0, $7, $4, $5, 1, "1 - - 1 - - -" } \
    $4<$2{print $7, $4, $5, 0, $6, $2, $3, 1, "1 - - 1 - - -" }' >
out.links.txt

#agp2assembly.py可以到juice的github下载
python ~/software/juicebox_scripts/juicebox_scripts/agp2assembly.py $agp
out.assembly 2>3.log

#run-assembly-visualizer.sh到3d-dna的github下载
bash ~/software/3d-dna-201008/visualize/run-assembly-visualizer.sh -p false
out.assembly out.links.txt 2>4.log

```

基因组评估

软件安装

```
###Installing BUSCO 依赖环境相当多，建议mamba安装
cd ~/biosoft
git clone https://gitlab.com/ezlab/busco.git
cd ~/biosoft/busco
python setup.py install
###Installing augustus 依赖环境相当多，建议mamba安装
cd ~/biosoft
git clone https://github.com/Gaius-Augustus/Augustus.git
cd augustus-3.3.1
make augustus
echo "export PATH=/share/home/off/biosoft/augustus-3.2.3/bin:$PATH"
>> ~/.bashrc
echo "export PATH=/share/home/off/biosoft/augustus-3.2.3/scripts:$PATH" >>
~/.bashrc
echo "export AUGUSTUS_CONFIG_PATH=/share/home/off/biosoft/augustus-3.2.3/config/"
>> ~/.bashrc
mamba create -n busco -c conda-forge -c bioconda busco=5.3.2
#BUSCO数据库下载
https://busco-data.ezlab.org/v5/data/lineages/
```

1.完整性评估

```
contig=./genome.fa
db1=embryophyta_odb10
db2=eudicots_odb10
db3=eukaryota_odb10
db4=viridiplantae_odb10

mkdir busco && cd busco

busco -i ${contig} -r -o embryophyta -l ${db1} -m genome -c 20 --offline --config \
~/biosoft/busco-master/config/config.ini
busco -i ${contig} -r -o eudicots -l ${db2} -m genome -c 20 --offline --config \
~/busco-master/config/config.ini
busco -i ${contig} -r -o eukaryota -l ${db3} -m genome -c 20 --offline --config \
~/biosoft/busco-master/config/config.ini
busco -i ${contig} -r -o viridiplantae -l ${db4} -m genome -c 20 --offline --
config \
~/busco-master/config/config.ini

mkdir BUSCO_summaries
cp */short_summary*.txt BUSCO_summaries
generate_plot.py -wd BUSCO_summaries
Rscript BUSCO_summaries/busco_figure.R
```

2.GC含量评估

```
#定义变量
genome=genome.fasta ## 基因组文件
hifi_fq=
prefix=gcdep ## 输出结果前缀
window=500 ## 窗口大小
step=250 ## 步长
minimap2 -ax map-hifi ${genome} ${hifi_fq} | samtools sort -@ 20 - -o
aln_sort.bam
#计算基因组序列长度
seqtk comp $genome | awk '{print $1"\t"$2}' > $prefix.len
#划分窗口 生成bed文件
bedtools makewindows -w $window -s $step -g $prefix.len > $prefix.window.bed
#按窗口提取序列并计算gc含量
seqtk subseq $genome $prefix.window.bed > $prefix.window.fasta
seqtk comp $prefix.window.fasta | \
    awk '{print $1 "\t" ($4+$5)/($3+$4+$5+$6) }' > $prefix.window.gc
#按窗口计算平均深度
bedtools coverage -b aln_sort.bam -a $prefix.window.bed -mean | \
    awk '{print $1:"$2+1"-"$3"\t"$4}' > $prefix.window.depth

#绘图
Rscript run_gcdep.R $prefix.window.gc $prefix.window.depth
$prefix.pdf 0 0.8 0 500
```

基因组注释

软件安装

```
#Installing EDTA(v2.0.0)
cd biosoft
git clone https://github.com/oushujun/EDTA.git
conda env create -f EDTA.yml
conda activate EDTA

#Installing geta

#Installing cmscan
mamba install cmscan -y

#Installing infernal
wget http://eddylib.org/software/infernal/infernal-1.1.1.tar.gz -P ~/software/
tar -zxvf ~/software/infernal-1.1.1.tar.gz -C ~/biosoft/

cd ~/biosoft/infernal-1.1.1

./configure --prefix=`pwd`
make && make install
cd ease1 && make install

echo "PATH=/share/home/off/biosoft/infernal-1.1.1/bin:$PATH" >> ~/.bashrc
```

基因组结构注释

#切换服务器

```
ssh gata@210.34.81.54
password:123456
```

```
cd /public2/geta/project
```

```
mkdir you_file
```

##数据准备

#1.RNA-seq数据准备

```
mkdir RNA_seq
```

```
cp /YOU-PATH/*.fq.gz ./RNA_seq
```

#2.同源蛋白准备(自己下载并合并到一个文件里面)

```
mkdir pep
```

```
cat *.fa > all.pep.fa
```

```
geta.pl --RM_species "Dimocarpus longan" \#设置物种拉丁名
--genome /public2/geta/project/47/Dlongan/dlo.scaffolds.fa \#加载调整好的基因组文件
-1 /public2/geta/project/47/Dlongan/RNAseq/*_1.clean.fq.gz \#加载RNA-seq文件
-2 /public2/geta/project/47/Dlongan/RNAseq/*_2.clean.fq.gz \#加载RNA-seq文件
--protein /public2/geta/project/47/Dlongan/pep/all.pep.fa \#加载同源蛋白文件
--augustus_species Dlongan \#设置物种名缩写
--out_prefix Dlo \#设置注释结果文件夹前缀
--config /public2/geta/project/45KBL/Cannabis_sativa/conf.txt \
--cpu 28 \ #设置调用线程数
--gene_prefix Dlo.M \#设置基因名前缀
--pfam_db /public2/geta/project/45KBL/Ap85-441/22.geta/Pfam-AB.hmm
```

基因组TE注释

#有CDS序列模式下的注释

```
source activate EDTA
```

```
EDTA=
```

```
curatedlib=
```

```
genome=
```

```
CDS=
```

```
perl $EDTA --genome $genome \
--cds $CDS \
--curatedlib $curatedlib \
--overwrite 1 --sensitive 1 --anno 1 --evaluate 1 --species others \
--step all --threads 20 2>1.log
```


#无CDS序列得TE注释

source activate EDTA

EDTA=~/.biosoft/EDTA/EDTA.pl

curatedlib=

genome=

```
perl $EDTA --genome $genome \  
            --curatedlib $curatedlib \  
            --overwrite 1 --sensitive 1 --anno 1 --evaluate 1 --species others \  
            --step all --threads 20 2>1.log
```

miRNA注释

wget ftp://ftp.ebi.ac.uk/pub/databases/Rfam/14.0/Rfam.cm.gz

gunzip Rfam.cm.gz

#使用infernall中的cmpress索引Rfam.cm

~/biosoft/infernal-1.1.1/bin/cmpress Rfam.cm

~/biosoft/infernal-1.1.1/bin/esl-seqstat genome.fasta

\$输出

Format:	FASTA
Alphabet type:	DNA
Number of sequences:	379
Total # residues:	467373171
Smallest:	18664
Largest:	46829937
Average length:	1233174.6

~/miniconda3/bin/cmscan -Z 934 \#该数据为2*Total

```
--cpu 20 \  
--cut_ga \  
--rfam \  
--nohmmonly \  
--tblout Dlongan.genome.tblout \  
--fmt 2 \  
--clanin Rfam.clanin Rfam.cm genome.fasta > genome.cmscan
```