基因课宏基因组数据分析专题教材

王莹

2022-01-15

很多事情从你决定开始的一瞬间起

最困难的时刻就已经过去了

目录

前言

关于基因课

成立于 2017 年 7 月，专注于生物信息学培训。

基因课网校：已完成 20 多门视频课程的开发，累计学员 5000 余人。

线下培训班：武汉、成都、西安、杭州、广州、北京等地举办快速入门班 30 余期。

基因课网站：www.genek.cn

关于作者

主讲老师王莹

曾任华大基因、贝纳基因高级生物信息工程师。

擅长基因组拼接、比较基因组和进化分析、重测序分析。

发表文章：甲藻基因组（Science）、穿心莲基因组（The Plant Journal）、野生大豆基因组（Nature Communications）、菊花基因组（Molecular Plant）、鹰嘴豆基因组计划（Nature Biotechnology）、芝麻基因组计划（Genome Biology）、猕猴桃群体研究（New Phytologist）。

第一章　宏基因组容器使用及相关软件安装

宏基因组数据分析涉及较多软件，安装繁琐，因此课程用的软件都安装到了 singularity 容器中。

本节主要包含 singularity 容器使用方法和宏基因组容器介绍。

大家如果需要自己手动安装软件，可以参考官网说明，软件官网和安装方式在本节最后一部分。

1.1　Singularity 安装和使用

1.1.1　安装

1. 安装依赖

```
sudo apt-get update && sudo apt-get install -y \
    build-essential \
    uuid-dev \
    libgpgme-dev \
    squashfs-tools \
    libseccomp-dev \
    wget \
    pkg-config \
```

```
    git \
    cryptsetup-bin
```

2. 安装 GO 语言

发现 singularity 不支持最新版 1.16.2 的 go，1.14.12 则没有问题。

下载地址：https://golang.org/dl/

```
wget https://golang.org/dl/go1.14.12.linux-amd64.tar.gz
tar -C /pub/software -xzvf go1.14.12.linux-amd64.tar.gz
rm go1.14.12.linux-amd64.tar.gz
```

添加到环境变量

```
echo 'export PATH=/pub/software/go/bin:$PATH' >> /etc/profile.d/env.sh
```

3. 下载 singularity

apt install singularity-container 可以安装 2.4.2 版本的 Singularity, 但没有新版本。所以需要手动安装。

下载地址：https://github.com/hpcng/singularity/releases

```
$ wget https://github.com/hpcng/singularity/releases/
  download/v3.7.2/singularity-3.7.2.tar.gz
$ tar -xzf singularity-3.7.2.tar.gz
$ cd singularity
```

4. 安装 singularity，，安装好以后记得添加到 PATH。

```
$ ./mconfig
$ cd builddir
$ make
$ sudo make install
```

安装过程中可能会报这样的信息，根据经验，不用管它。

```
go: github.com/Netflix/go-expect@v0.0.0-20190729225929-0e00d9168667:\
Get "https://proxy.golang.org/github.com/...7.mod": dial tcp 172.217.27.1
i/o timeout
```

1.1.2 快速上手

1. 下载 images

可以从 Container Library（https://cloud.sylabs.io/library ）或 Docker Hub（https://hub.docker.com/ ）下载 images。

```
singularity pull --arch amd64 library://library/default/ubuntu:20.04
```

如果下载速度太慢，可以直接使用基因课提供 ubuntu_20.04.sif

2. 创建沙箱

刚下载下来的 ubuntu_20.04.sif 只是一个纯净的系统，我们需要创建一个沙箱，给里面装软件。

```
singularity build --sandbox blast ubuntu_20.04.sif
```

3. 进入容器

默认会自动挂载 $HOME, $PWD，/tmp，/proc，/sys，/dev 目录。

```
$ singularity shell --writable --fakeroot blast
Singularity> apt install ncbi-blast+
```

跟平时一样，在容器中安装软件，建议不要使用 anaconda 安装，而是手动安装。我们要尽量保持容器轻量，Anaconda 会安装太多不必要的东西。

4. 打包

```
singularity build blast.sif blast
```

5. 运行程序

```
singularity exec blast.sif makeblastdb -h
```

6. 下载镜像

可以从 Container Library（https://cloud.sylabs.io/library Docker Hub（https://hub.docker.com/ images。

```
# 从 Container Library 下载 sif 格式镜像
singularity pull library://cenat/default/blast.sif:latest
# 从 DockerHub 下载 sif 格式镜像
singularity pull docker://ncbi/blast
```

7. 运行容器

```
# 交互式运行
$ singularity shell blast.sif bash
Singularity> blastp
BLAST query/options error: Either a BLAST database or subject sequence(s)
Please refer to the BLAST+ user manual.
Singularity>
# 直接运行
$ singularity exec blast.sif blastp
BLAST query/options error: Either a BLAST database or subject sequence(s)
Please refer to the BLAST+ user manual.
```

8. 文件夹挂载

默认会自动挂载 $HOME, $PWD , /tmp , /proc , /sys , /dev 目录。

```
# 通过 --bind 挂载文件夹
$ singularity shell --bind /pub/software:/mnt blast.sif
Singularity> ls /mnt/samtools-1.9.tar.bz2
/mnt/samtools-1.9.tar.bz2
Singularity>

# 不写挂载点则和本地目录一直
$ singularity shell --bind /pub/software blast.sif
Singularity> ls /pub/software/samtools-1.9.tar.bz2
/pub/software/samtools-1.9.tar.bz2
Singularity>
```

9. 用户和权限

使用容器不得不考虑安全性，安全性来自两个方面，一个是使用了不被信任的容器，这个就像你在电脑上安装了不被信任的软件一样，Singularity 提供了签名机制来验证；另一方面是容器会不会越权对 Host 做一些不该做的事情，这个是需要考虑的。

singularity 的解决办法是会在容器内动态创建一个用户，该用户与 Host 里的用户名、组名、权限等都保持一致。这样你在 Host 中做不了的事情，在容器里也干不了。

```
# Host 用户
$ id
uid=1000(genek) gid=1000(genek) groups=1000(genek),4(adm),24(cdrom),27(sudo),3
(base) genek@gs30:~/workspace/singularity$ singularity shell blast.sif
# 容器内用户 完全一致
Singularity> id
uid=1000(genek) gid=1000(genek) groups=1000(genek),4(adm),24(cdrom),27(sudo),3
Singularity>
```

但是有时候我们需要以 root 权限（–fakeroot）登录容器，这时也不用担心，因为 root 只对容器内有效。

需要注意的是，Singularity 默认是绑定 $PWD 和 $HOME 目录的，对这些目录的操作都会修改 Host 文件。

1.2 查看和使用创建好的容器

本次课程给大家创建好了容器，里面包含了课程里所需的所有软件，容器内软件均为手动安装进去的，容器里不包含 conda。

1.2.1 Metagenome.sif

Metagenome.sif 内包含了本期培训使用的大部分软件以及调用的所有 R 包。容器中大部分软件存放在容器内的/opt 目录下。

查看容器内/opt 目录可以使用如下命令：

```
## 使用shell命令进入交互模式
$   singularity  shell MetaGenome.sif
```

```
## 交互模式下查看/opt目录
Singularity> ls /opt
ANIcalculator_v1                    hisat2-2.2.1
Bracken-2.6.2                       idba-1.1.3
CONCOCT-1.1.0                       iqtree-1.6.12-Linux
FastQC                              kraken2
FastTree                            mash-Linux64-v2.3
KrakenTools                         metaWRAP-1.3
MEGAHIT-1.2.9-Linux-x86_64-static   metabat
MaxBin-2.2.7                        mummer-4.0.0rc1
MetaGeneMark_linux_64               picard.jar
SPAdes-3.15.3-Linux                 pplacer-Linux-v1.1.alpha19
barrnap                             prodigal
bedtools2                           prodigal.linux
bwa                                 prokka
cd-hit-v4.8.1-2019-0228             quast
centrifuge-1.0.4                    raxml-ng_v1.0.3
checkm_data_2015_01_16              salmon-1.6.0_linux_x86_64
epstopdf.pl                         samtools-1.13
fastANI                             seqkit
fastp                               usearch11.0.667_i86linux32
gtfToGenePred
```

1.2.2 lefse_latest.sif

lefse_latest.sif 容器只包含容 lefse 软件，创建方式是直接使用软件官网提供的 docker 容器。

```
## 使用shell命令进入交互模式
$ singularity shell lefse_latest.sif
Singularity> which  run_lefse.py
/home/linuxbrew/.linuxbrew/bin/run_lefse.py
```

```
Singularity>
```

1.2.3 run_dbcan_latest.sif

run_dbcan_latest.sif 容器包含用于 CAZy 注释的 run_dbcan 软件，创建方式是直接使用软件官网提供的 docker 容器。

1.2.4 rgi.sif

rgi.sif 容器包含用于 CARD 注释的 RGI 软件，创建方式是直接使用软件官网提供的 docker 容器。

1.2.5 gtdbtk_latest.sif

gtdbtk_latest.sif 容器包含 gtdbtk 软件，创建方式是直接使用软件官网提供的 docker 容器。

1.3 R 包安装

```
install.packages(c("pheatmap","argparser", "cowplot","ggalluvial))
install.packages(c("tidyverse", "formattable", "seqinr","pkgbuild"))
install.packages("vegan")

BiocManager::install("AnnotationForge")
BiocManager::install("clusterProfiler")
BiocManager::install("phyloseq")
BiocManager::install("PCAtools")
```

1.4 conda 安装

conda 主要用来创建不同环境，来安装依赖较多的软件。

从 conda 官网下载最新版 conda 进行安装，网址如下：

https://repo.anaconda.com/archive/

```
# 下载 Anaconda
$ wget https://repo.anaconda.com/archive/Anaconda3-2021.11-\
    Linux-x86_64.sh

# 安装anaconda
$ bash Anaconda3-2021.11-Linux-x86_64.sh

#  配置conda channels
$ cat  ~/.condarc
channels:
- bioconda
- conda-forge
- r
- defaults
```


如果网速慢，可以使用清华的 channels:

```
channels:
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/menpo/
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda/
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2/
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
- defaults
show_channel_urls: true
```

使用 conda 安装和管理 python 模块，不建议使用 conda 安装 perl 和 R 相关软件

```
conda install biopython
```

1.5 其他软件安装

1.5.1 比对、数据质控及 fasta 序列处理

- fastp

网址：https://github.com/OpenGene/fastp

```
## 下载即可
wget http://opengene.org/fastp/fastp
chmod a+x ./fastp
```

- fastqc

网 址：https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

```
## 下载解压即可
wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/\
    fastqc_v0.11.9.zip

unzip  fastqc_v0.11.9.zip
```

- multiQC

网址：https://multiqc.info/

```
pip3 install multiqc
```

- bwa

网址：https://github.com/lh3/bwa

```
git clone https://github.com/lh3/bwa.git
cd bwa
make
```

- bowtie2

网址：http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

```
$ wget https://phoenixnap.dl.sourceforge.net/project/bowtie-bio/\
      bowtie2/2.4.4/bowtie2-2.4.4-linux-x86_64.zip
$ unzip  bowtie2-2.4.4-linux-x86_64.zip
```

- samtools

网址：https://github.com/samtools/samtools

```
$ wget https://github.com/samtools/samtools/releases/download/1.13/\
        samtools-1.13.tar.bz2
$ tar -xvf samtools-1.13.tar.bz2
$ cd  samtools-1.13
$ ./configure
$ make
```

- seqtk

```
## 有管理员权限可以直接apt安装
$ apt install seqtk

## 普通用户安装
$ git clone https://github.com/lh3/seqtk.git;
$ cd seqtk
$ make
```

- seqkit

网址：https://bioinf.shenwei.me/seqkit/

```
## 选择服务器系统对应版本，下载后直接解压即可
$ wget  https://github.com/shenwei356/seqkit/releases/download/\
    v2.0.0/seqkit_linux_arm64.tar.gz
$ tar -zxvf seqkit_linux_arm64.tar.gz
```

1.5.2 物种组成分析

- kraken2

网址：https://ccb.jhu.edu/software/kraken2/

数据库：https://benlangmead.github.io/aws-indexes/k2

```
## 依赖ncbi blast+ suite
$ sudo apt-get install ncbi-blast+

$ wget https://github.com/DerrickWood/kraken2/archive/refs/\
    tags/v2.1.2.tar.gz
$ tar -zxvf  v2.1.2.tar.gz
$ cd kraken2-2.1.2
```

```
$ mkdir   /path/to/installkraken2
$ ./install_kraken2.sh    /path/to/installkraken2

## 安装kraken-biom
$ pip install kraken-biom
```

- bracken

网址：https://ccb.jhu.edu/software/bracken/index.shtml

```
$ wget https://github.com/jenniferlu717/Bracken/archive/refs/\
    tags/v2.6.2.tar.gz
$ mv  v2.6.2.tar.gz Bracken-2.6.2.tar.gz
$ tar -zxvf  Bracken-2.6.2.tar.gz
$ bash install_bracken.sh
```

- lefse

网址：https://github.com/SegataLab/lefse

```
# 方法1
$ conda install -c bioconda lefse

# 方法2
$ docker run -it biobakery/lefse bash

# 方法3
$ singularity  pull  lefse_latest.sif docker:://biobakery/lefse
```

- krona

网址：https://github.com/marbl/Krona/wiki

```
wget https://github.com/marbl/Krona/releases/download/v2.8.1/\
   KronaTools-2.8.1.tar

tar -xvf  KronaTools-2.8.1.tar
cd KronaTools-2.8.1
## 安装
./install.pl --prefix /path/to/install
## 下载NCBI taxonomy
 updateTaxonomy.sh
```

1.5.3 组装及注释及定量

- megahit

网址：https://github.com/voutcn/megahit

```
## 免安装，下周直接解压缩
$ wget https://github.com/voutcn/megahit/releases/download/\
      v1.2.9/MEGAHIT-1.2.9-Linux-x86_64-static.tar.gz
$ tar zvxf MEGAHIT-1.2.9-Linux-x86_64-static.tar.gz
$ cd MEGAHIT-1.2.9-Linux-x86_64-static/bin/
## 测试
$ ./megahit --test  # run on a toy dataset
```

- metaspades

网址：https://cab.spbu.ru/software/meta-spades/

```
## 免安装，下载后直接解压缩即可
$ wget https://cab.spbu.ru/files/release3.15.3/SPAdes-3.15.3-Linux.tar.gz
$ tar -zxvf SPAdes-3.15.3-Linux.tar.gz
```

```
$ cd  SPAdes-3.15.3-Linux/bin
## 测试
$ ./spades.py -h
```

- quast

网址：http://quast.sourceforge.net/quast

```
## 方法1
$ pip install quast

## 方法2
$ git clone https://github.com/ablab/quast.git
## 测试并自动安装依赖
$ cd quast
$ ./quast.py  --test
```

- prodigal

网址：https://github.com/hyattpd/Prodigal

```
$ wget https://github.com/hyattpd/Prodigal/releases/download/\
    v2.6.3/prodigal.linux

$ mv prodigal.linux prodigal
$ chmod 755 prodigal
```

- prokka

网址：

```
## 提前安装好依赖
## sudo apt-get install libdatetime-perl libxml-simple-perl \
##      libdigest-md5-perl git default-jre bioperl
## sudo cpan Bio::Perl

$ git clone https://github.com/tseemann/prokka.git
$ ./prokka/bin/prokka --setupdb
```

- CD-hit

网址：http://weizhong-lab.ucsd.edu/cd-hit/

```
$ wget https://github.com/weizhongli/cdhit/releases/download/V4.8.1/\
      cd-hit-v4.8.1-2019-0228.tar.gz
$ tar -zxvf cd-hit-v4.8.1-2019-0228.tar.gz
$ cd cd-hit-v4.8.1-2019-0228
$ make
```

- salmon

网址：https://github.com/COMBINE-lab/salmon

```
## 解压后直接使用
$ wget https://github.com/COMBINE-lab/salmon/releases/download/v1.6.0/\
      salmon-1.6.0_linux_x86_64.tar.gz
$ tar  -zxvf salmon-1.6.0_linux_x86_64.tar.gz
```

- RGI

网址：https://github.com/arpcard/rgi

```
## 方法1
$ git clone https://github.com/arpcard/rgi
$ conda env create -f conda_env.yml
$ conda activate rgi
$ conda install rgi

## 方法2
$ singularity  pull docker://finlaymaguire/rgi:latest


##  Load CARD Reference Data
$ wget https://card.mcmaster.ca/latest/data
$ tar -xvf data ./card.json
```

- run_dbcan

网址：https://github.com/linnabrown/run_dbcan

```
# 方法1
$ conda create -n run_dbcan python=3.8 diamond hmmer prodigal
$ conda activate run_dbcan
$ pip install dbcan==3.0.1

# 方法2
$ singularity pull docker://haidyi/run_dbcan:latest
```

1.5.4 分箱相关

- metabat2

网址：https://bitbucket.org/berkeleylab/metabat/src

```
## 下载解压即可
$ wget https://bitbucket.org/berkeleylab/metabat/downloads/\
      metabat-static-binary-linux-x64_v2.12.1.tar.gz
$ tar -zxvf metabat-static-binary-linux-x64_v2.12.1.tar.gz
```

- maxbin2

网址：http://sourceforge.net/projects/maxbin/

```
$ wget https://iweb.dl.sourceforge.net/project/maxbin/MaxBin-2.2.7.tar.gz
$ tar -zxvf MaxBin-2.2.7.tar.gz
$ cd MaxBin-2.2.7/src
$ make
$ cd ../
$ ./autobuild_auxiliary
```

- concoct

网址：https://concoct.readthedocs.io/en/latest/#

```
## 可使用conda安装
$ conda create -n concoct_env python=3 concoct

## 可以从源码安装
$ wget https://github.com/BinPro/CONCOCT/archive/refs/tags/1.1.0.tar.gz
$ mv 1.1.0.tar.gz CONCOCT-1.1.0.tar.gz
$ tar -zxvf CONCOCT-1.1.0.tar.gz
$ cd CONCOCT-1.1.0
##安装依赖
$ pip install -r requirements.txt
## 安装concoct
$python setup.py install  --user
```

- checkM

网址：https://ecogenomics.github.io/CheckM/

```
## 安装依赖
$ conda create -n checkm python=3.9
$ conda activate checkm
$ conda install numpy matplotlib pysam
$ conda install hmmer prodigal pplacer

## 安装checkM
$ pip3 install checkm-genome

## checkM database
$ mkdir MY_CHECKM_FOLDER
$ wget https://data.ace.uq.edu.au/public/CheckM_databases/\
      checkm_data_2015_01_16.tar.gz
$ tar -xvf *.tar.gz
$ cd ../
## 设置database路径
$ checkm data setRoot /path/to/your/dir/MY_CHECKM_FOLDER
```

- metawrap

网址：https://github.com/bxlab/metaWRAP

database 构建参考以下网址:

https://github.com/bxlab/metaWRAP/blob/master/installation/database_installation.md

```
## 创建环境
$ conda create -y -n metawrap-env python=2.7
$ conda activate metawrap-env
```

```
## 下载metaWRAP
$ wget https://github.com/bxlab/metaWRAP/archive/refs/\
      tags/v1.3.tar.gz
$ mv  v1.3.tar.gz metaWRAP-1.3.tar.gz
$ tar -zxvf metaWRAP-1.3.tar.gz
## 添加环境
$ export PATH=/yourpath/metaWRAP-1.3/bin/:$PATH
```

- dRep

网址：https://github.com/MrOlm/drep

```
## 提前安装好以下依赖软件
### Near Essential:
## Mash   - Makes primary clusters (v1.1.1 confirmed works)
## MUMmer - Performs default ANIm comparison method (v3.23 confirmed works)
### Optional:
## fastANI - A fast secondary clustering algorithm
## CheckM  - Determines contamination and completeness of genomes (v1.0.7 conf
## gANI (aka ANIcalculator) - Performs gANI comparison method (v1.0 confirmed
## Prodigal - Used be both checkM and gANI (v2.6.3 confirmed works)
## NSimScan - Only needed for goANI algorithm (open source version of gANI)

$ pip install drep
```

- GTDB-tk

网址：https://github.com/Ecogenomics/GTDBTk

```
# 方法1
$ python -m pip install gtdbtk
```

```
## 设置数据库路径
$ export GTDBTK_DATA_PATH=/path/to/release/package/

# 方法2
$ conda create -n gtdbtk -c conda-forge -c bioconda gtdbtk
$ conda create -n gtdbtk-1.3.0 -c conda-forge -c bioconda gtdbtk=1.3.0
## 设置数据库路径
$ echo "export GTDBTK_DATA_PATH=/path/to/release/package/"  \
      > /miniconda3/envs/gtdbtk-1.3.0/etc/conda/activate.d/gtdbtk.sh

# 方法3 直接下载容器,数据库通过--bind挂载
singularity  pull  docker://ecogenomic/gtdbtk
```


基因课

第二章　测序数据质控过滤

本部分讲解二代宏基因组测序数据的质控和过滤。

过滤包括以下两个步骤：

1. 测序数据质量过滤，包括测序质量、adapter 等过滤
2. 去除污染数据，一般是宿主数据。该步骤对于水体、土壤等样品可以跳过。

测试数据下载自 NCBI，为水体测序数据，对应 project 编号为 PRJNA723368。

使用软件：fastqc、fastp 、bowtie2 、samtools 等。

2.1　数据质控

原始测序数据下机后，可能会包含低质量、接头等，我们可以使用 fastqc 对数据质量进行检查.

```
## 创建输出目录
mkdir  ./qc

## 运行fastqc进行质控
singularity exec ../../software/MetaGenome.sif fastqc \
  --outdir ./qc \   # 输出目录
  --threads 4  \    # 线程数
```

```
  ../../dataFQ/A1_1.fq.gz \  # 输入fq文件
  ../../dataFQ/A1_2.fq.gz \
  ../../dataFQ/A2_1.fq.gz \
  ../../dataFQ/A2_2.fq.gz
```

结果如下：

1. 网页版质控报告 *_fastqc.html
2. 报告对应数据 *_fastqc.zip

2.2 质量过滤

这里使用 fastp 进行低质量数据过滤。

```
## 创建输出目录
mkdir -p clean_data

## 每组数据分别运行fastp
singularity exec ../../software/MetaGenome.sif  fastp \
  --thread  4 \  # 线程数
  -i  ../../dataFQ/A1_1.fq.gz \  # 输入数据fq1
  -I ../../dataFQ/A1_2.fq.gz \  # 输入数据fq2
  -o  clean_data/A1_1.fq.gz \  # 输出数据fq1
  -O  clean_data/A1_2.fq.gz  \  # 输出数据fq2
  -j  clean_data/A1.fastp.json \  # json格式日志
  -h  clean_data/A1.fastp.html  # 网页版日志

singularity exec ../../software/MetaGenome.sif  fastp \
  --thread  4 \
  -i  ../../dataFQ/A2_1.fq.gz \
  -I ../../dataFQ/A2_2.fq.gz \
  -o  clean_data/A2_1.fq.gz \
```

```
  -O clean_data/A2_2.fq.gz  \
  -j  clean_data/A2.fastp.json \
  -h  clean_data/A2.fastp.html
```

输出结果：

过滤后 fq1 数据 clean_data/*.fq.gz

可以使用 muiltQC 对 fastp 日志进行汇总：

```
singularity exec ../../software/MetaGenome.sif  multiqc  \
  ./clean_data/
```

输出结果：

1. 网页版报告 multiqc_report.html
2. 报告对应数据目录 multiqc_data

2.3 去除宿主

对于宿主来源的宏基因组测序数据，需要对其中的宿主序列进行去除。

```
## 构建宿主基因组index

ln -s ../../data/genome.fa

singularity exec  ../../software/MetaGenome.sif bowtie2-build \
  genome.fa \  # 基因组序列
  genome.db  # 输出index前缀

## bowtie2比对
```

```
singularity exec  ../../software/MetaGenome.sif  bowtie2 \
  --threads 4 \  # 线程
  -x ./genome.db  \  # 基因组index名称
  -1 ../01.quality/clean_data/A1_1.fq.gz \  # 输入，fq1
  -2 ../01.quality/clean_data/A1_2.fq.gz \  # 输入，fq2
  -S  A1.sam  \  # 输出，sam格式比对结果
  2>A1.map.log

## 去除宿主数据
singularity exec ../../software/MetaGenome.sif samtools view \
    -f 12  \  # 去除比对上的reads
    A1.sam \  # 输入，sam文件
    >A1.unmap.bam  # 输出bam格式文件

## bam转换回fq格式
singularity exec  ../../software/MetaGenome.sif samtools fastq \
    -1  A1_1.clean.fq.gz \  # 输出，fq1
    -2 A1_2.clean.fq.gz  \  # 输出，fq2
    -s A1_singleton.clean.fq.gz \  # 输出，单端数据
    A1.unmap.bam  # 输入，bam文件


## 以上比对、去除宿主、转fq三个步骤可以串联起来一步完成
singularity exec  ../../software/MetaGenome.sif  bowtie2 --threads 4 \
  -x ./genome.db \
  -1 ../01.quality/clean_data/A1_1.fq.gz \
  -2 ../01.quality/clean_data/A1_2.fq.gz 2>A1.map.log | \
  singularity exec  ../../software/MetaGenome.sif samtools view -f 12 | \
  singularity exec  ../../software/MetaGenome.sif samtools fastq \
  -1  A1_1.clean.fq.gz -2 A1_2.clean.fq.gz \
  -s A1_singleton.clean.fq.gz
```

第三章　物种组成及多样性分析

该部分基于测序数据，使用 kraken2 进行 reads 的物种注释，使用 braken 进行丰度估计，之后基于丰度进行物种多样性分析及差异物种分析。

3.1　数据库构建

运行 kraken2 和 bracken 之前需要提前进行数据库构建。

kraken2 数据库地址：

https://benlangmead.github.io/aws-indexes/k2

```
### kraken2 database

### 方法1：直接下载官方提供的数据库（包括kraken2及bracken），解压即可
#wget https://genome-idx.s3.amazonaws.com/kraken/k2_pluspf_\
#    20210517.tar.gz
#wget https://genome-idx.s3.amazonaws.com/kraken/k2_pluspf_\
#    8gb_20210517.tar.gz
#wget https://genome-idx.s3.amazonaws.com/kraken/k2_standard\
#  _20210517.tar.gz

### 方法2：一步构建kraken标准数据库
## standard : archaea, bacteria, viral, plasmid, human1, UniVec_Core
singularity exec ../../software/MetaGenome.sif  kraken2-build \
    --standard  \
```

```
    --threads 4 \
    --db ./standardDB


### 方法3: 手动构建数据库
## step1 download taxonomy file from NCBI
singularity exec ../../software/MetaGenome.sif kraken2-build \
  --download-taxonomy \
  --db  ./K2db

## step2 download sequence data
singularity exec ../../software/MetaGenome.sif  kraken2-build \
  --download-library UniVec_Core  \
  --threads 4 \
  --db  ./K2db
singularity exec ../../software/MetaGenome.sif  kraken2-build \
  --download-library UniVec \
  --threads 4 \
  --db  ./K2db

## step3 build database
singularity exec ../../software/MetaGenome.sif  kraken2-build  \
  --build \
  --threads 4 \
  --db  ./K2db


### bracken database
singularity exec ../../software/MetaGenome.sif bracken-build \
    -d  K2db/ -t 4
```

kraken2 数据库:

```
hash.k2d
opts.k2d
taxo.k2d
```

braken 数据库：

```
database100mers.kmer_distrib
database100mers.kraken
database.kraken
```

3.2 物种组成及丰度估计

这里我们直接使用官方提供的数据库进行物种组成分析。

数据库下载自以下网址：

https://benlangmead.github.io/aws-indexes/k2

运行 kraken2，进行 reads 物种注释：

```
singularity exec ../../software/MetaGenome.sif kraken2 \
    --threads 4 \
    --quick \  # 快速模式
    --paired \  # 输入数据为双端模式
    --db ../../Database/K2/ \  # 数据库路径
    --report A1.kreport \  # 输出，report文件
    --output A1.kraken  \  # 输出，reads注释结果
    ../../P1.data_filter/02.contaminant/A1_1.clean.fq.gz \  # 输入，fq1
    ../../P1.data_filter/02.contaminant/A1_2.clean.fq.gz  # 输入，fq2
```

输出结果格式如下：

```
$ head  -n 5 A2.kraken
U       SRR14297775.1001        0       151|151 0:Q
U       SRR14297775.1002        0       151|151 0:Q
C       SRR14297775.1003        326522  151|151 326522:Q
U       SRR14297775.1004        0       151|151 0:Q
C       SRR14297775.1005        1879050 151|151 1879050:Q


$ head  -n 10  A1.kreport
 42.62  2245391 2245391 U       0       unclassified
 57.38  3022609 10470   R       1       root
 57.10  3008121 4341    R1      131567    cellular organisms
 56.83  2993672 62548   D       2           Bacteria
 35.80  1886075 75898   P       1224          Proteobacteria
 23.75  1251400 78611   C       1236            Gammaproteobacteria
 14.12  743761  624     O       72274             Pseudomonadales
 13.84  728900  4008    F       468                 Moraxellaceae
 13.37  704470  483073  G       469                   Acinetobacter
  1.81  95496   93935   S       40214                   Acinetobacter johnsoni
```

运行 bracken 进行各个分类水平物种丰度估计:

```
## 创建输出目录，这里为进行species水平分析
mkdir out_S

## 运行bracken
singularity exec ../../software/MetaGenome.sif bracken \
    -d ../../Database/K2/ \  # 数据库路径
    -i  A1.kreport \  # 输入，为kraken2 report文件
    -o A1.bracken.S \  #  输出，物种丰度估计表格
    -w A1.bracken.S.kreport \  # 输出，kraken2 report格式文件
    -l S \  # 分类水平，如 D,P,C,O,F,G,S
    -t 4  #   线程数
```

输出结果格式如下：

```
$ head -n 5 out_S/A1.bracken.S
name    taxonomy_id   taxonomy_lvl  kraken_assigned_reads   added_reads
new_est_reads   fraction_total_reads
Acinetobacter johnsonii       40214   S  10107 23754  33861  0.03461
Acinetobacter sp. NEB 394     2743575 S  1798  9414   11212  0.01146
Acinetobacter sp. WCHA45      2004644 S  978   2918   3896   0.00398
Acinetobacter sp. WCHAc010034 1879049 S  639   237    876    0.00090

$ head  -n 8 out_S/A1.kreport
100.00  978421  0   R     1       root
99.96   978023  0   R1    131567    cellular organisms
99.80   976502  0   D     2           Bacteria
65.48   640660  0   P     1224          Proteobacteria
29.08   284549  0   C     1236            Gammaproteobacteria
14.00   137022  0   O     72274             Pseudomonadales
13.51   132185  0   F     468                 Moraxellaceae
12.92   126457  0   G     469                   Acinetobacter
```

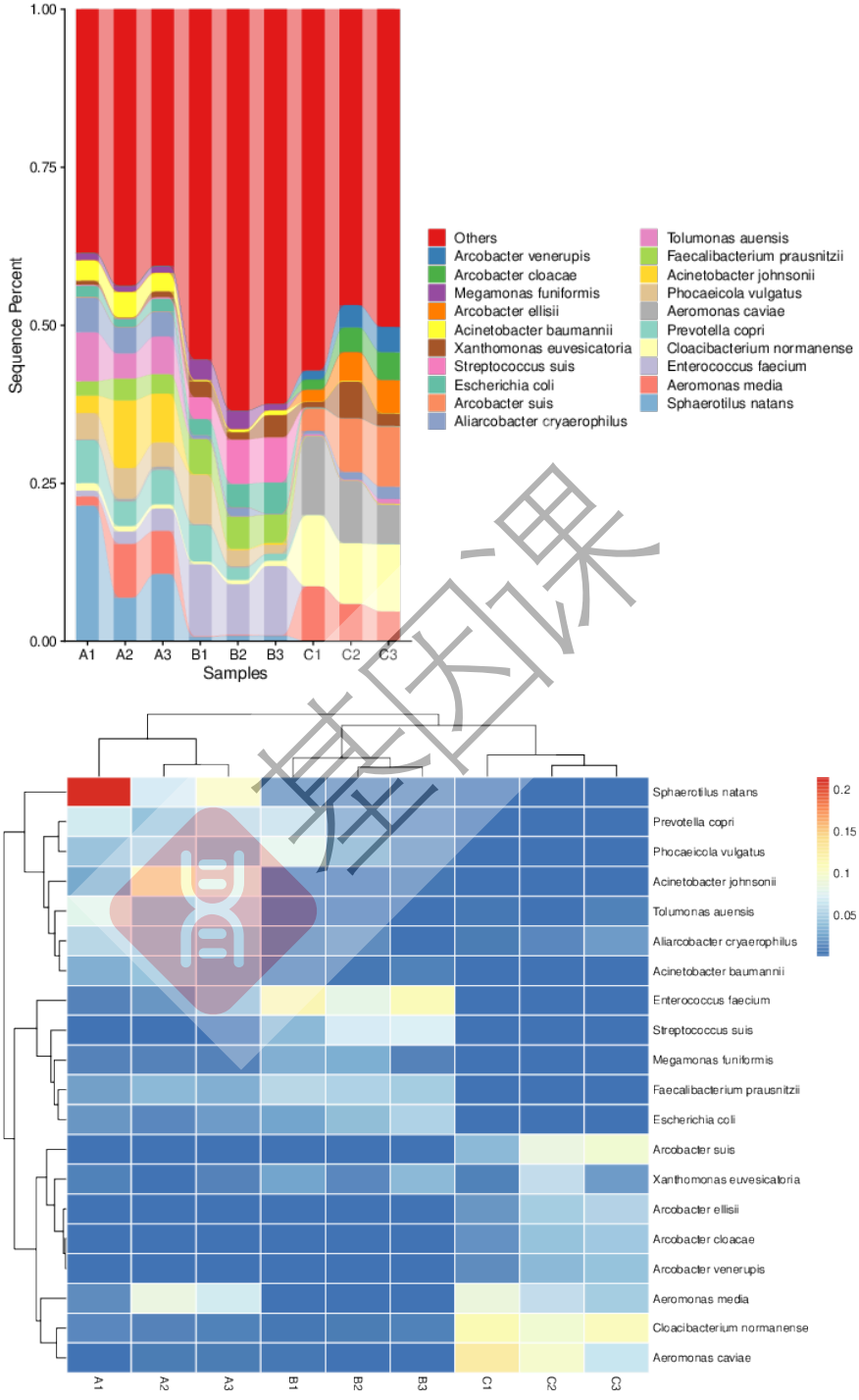
将各样品丰度结果汇总：

```
## report文件合并成biom格式
singularity exec ../../software/MetaGenome.sif kraken-biom  \
    ./out_S/*.kreport  \  # 输入,report文件
    --max D  \   # 最高物种分类
    -o  ./out_S/S.biom # 输出，  biom文件

## biom转count表格
singularity exec  ../../software/MetaGenome.sif biom convert \
    -i  ./out_S/S.biom \  # 输入，biom文件
    -o  ./out_S/S.count.tsv.tmp \  # 输出，丰度表格
    --to-tsv \  # 指定输出格式
```

```
    --header-key taxonomy  # 输出分类信息

## 输出文件格式调整，补全物种名
sed 's/; g__\([^;]\+\); s__/; g__\1; s__\1 /' ./out_S/S.count.tsv.tmp \
    >  ./out_S/S.taxID.count.tsv

##  taxonID 替换回拉丁名
sed  '/^#/! s/^[0-9]\+\t\(.*[A-Za-z]\+__\([^;]\+\)\)$/\2\t\1/'  \
    ./out_S/S.taxID.count.tsv  >  ./out_S/S.taxName.count.tsv

## 保留丰度信息，用于后续绘图
sed '1d; 2s/^#//' ./out_S/S.taxName.count.tsv | \
    awk  -F "\t" -v 'OFS=\t'  '{$NF = ""; { print $0 }}' | \
    sed 's/\t$//'  > ./out_S/S.count.tsv

## 绘制barplot 和 heatmap，输出相对丰度
singularity  exec  ../../software/MetaGenome.sif Rscript  \
      ../script/draw_taxonBarplot.R \
      out_S/S.count.tsv  20 \
      out_S/S.count.out
```

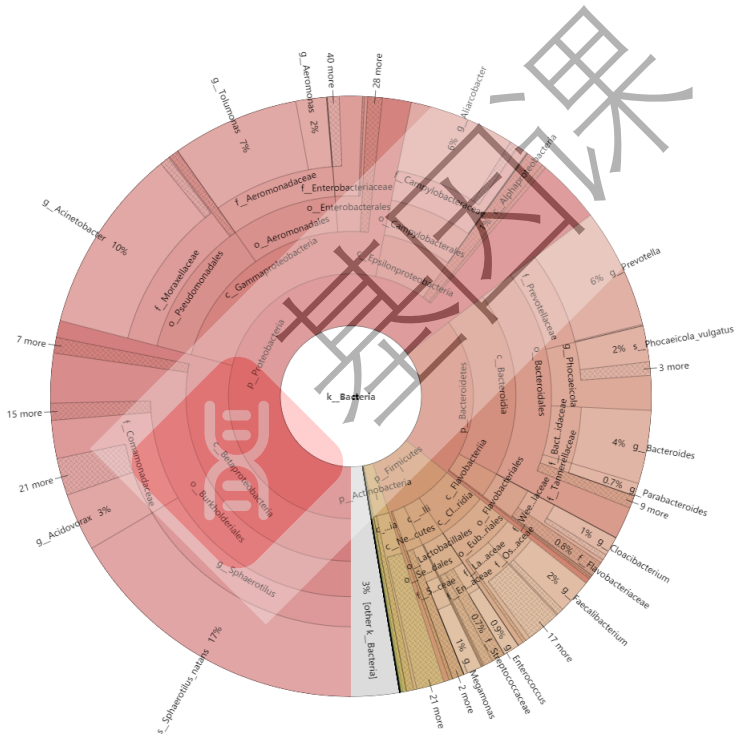
输出结果：

S.taxName.count.tsv 为包含物种分类信息的丰度文件
S.count.tsv 为去除物种分类信息的丰度文件
S.count.out.relative.table 相对丰度表
S.count.out.barplot.pdf 丰度barplot
S.count.out.heatmap_cluster.pdf 丰度热图
S.count.out.heatmap_original.pdf 丰度热图



kraken2 report 格式文件可以用来绘制 krona 环形图：

```
## 生成krona绘图文本数据
singularity exec ../../software/MetaGenome.sif kreport2krona.py \
    -r A1.kreport \   # 输入kraken report格式文件
    -o A1.kreport.krona   # 输出krona绘图数据

## 绘制krona环形图
singularity exec ../../software/MetaGenome.sif ktImportText \
    -o A1.kreport.krona.html \  # 出krona 网页版报告
    A1.kreport.krona  # 输入文件
```



3.3 多样性分析

这里使用 vegan 及 phyloseq 两个 R 包进行多样性分析。

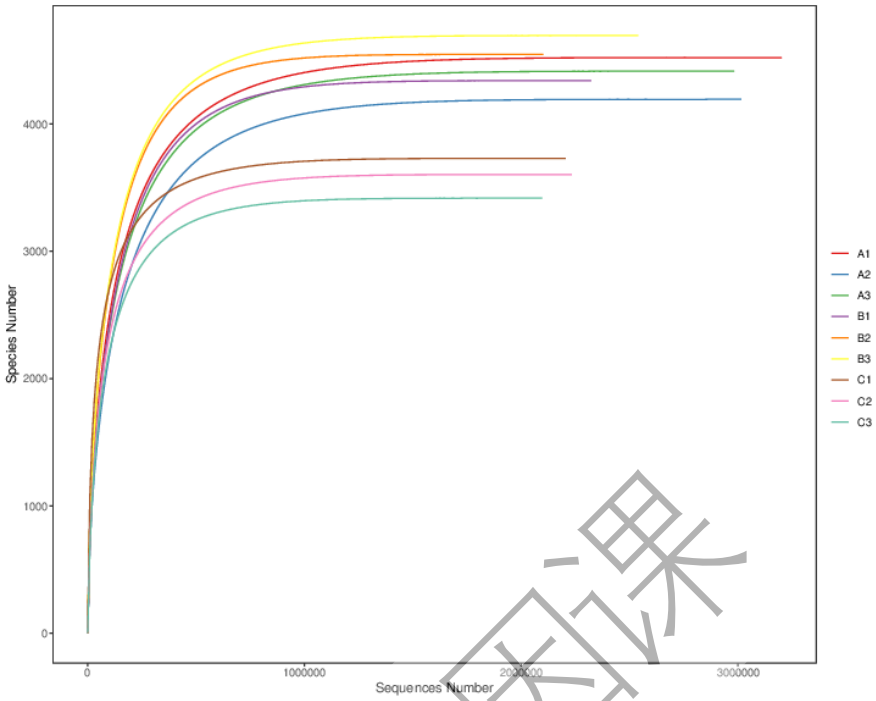
```
## 准备输入biom文件
ln -s ../01.taxon/out_S/S.biom

## 进行alpha多样性指数计算及绘制稀释曲线
singularity exec ../../software/MetaGenome Rscript \
    ../script/diversity_alpha.R  S.biom  S.alpha
```

输出结果：

```
S.alpha.alpha-diversity.table  alpha 多样性指数
S.alpha.Rarefaction_ggplot2.pdf  稀释曲线
S.alpha.Rarefaction_orginal.pdf  稀释曲线
```

结果格式如下：

```
$ cat S.alpha.alpha-diversity.table
Observed Chao1 se.chao1 ACE se.ACE Shannon Simpson InvSimpson
A1  1488  1488  0 1488  18.03  3.85  0.92 13.37
A2  4192  4192  0 4192  30.02  4.44  0.96 27.46
A3  4413  4413  0 4413  29.74  4.48  0.96 27.28
B1  4339  4339  0 4339  31.31  4.61  0.96 30.42
B2  4545  4545  0 4545  31.28  4.99  0.97 45.61
B3  4693  4693  0 4693  29.86  4.86  0.97 35.18
C1  3729  3729  0 3729  24.05  4.62  0.95 23.04
C2  3602  3602  0 3602  25.72  4.08  0.95 22.99
C3  3416  3416  0 3416  25.59  4.13  0.95 24.53
```



beta 多样性分析需要样品分组文件，格式如下：

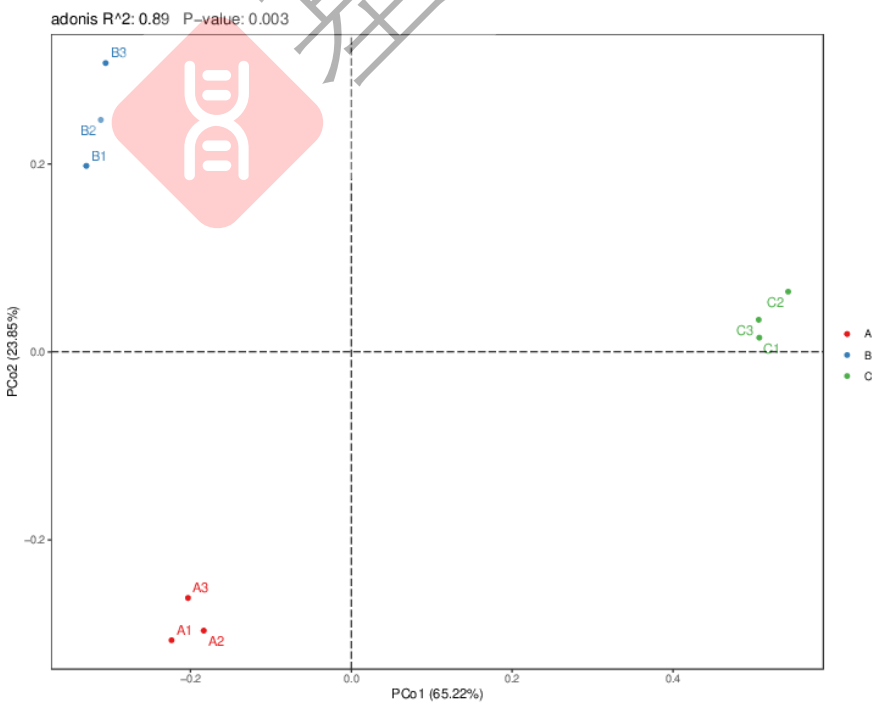
```
A       A1
A       A2
A       A3
C       C1
C       C2
C       C3
B       B1
B       B2
B       B3
```

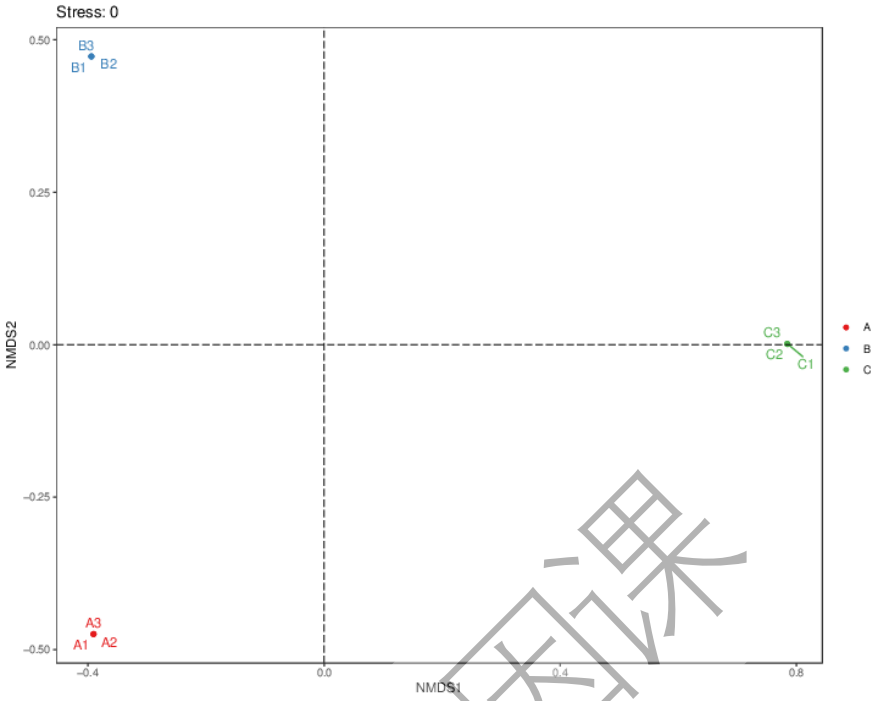
进行 beta 多样性分析

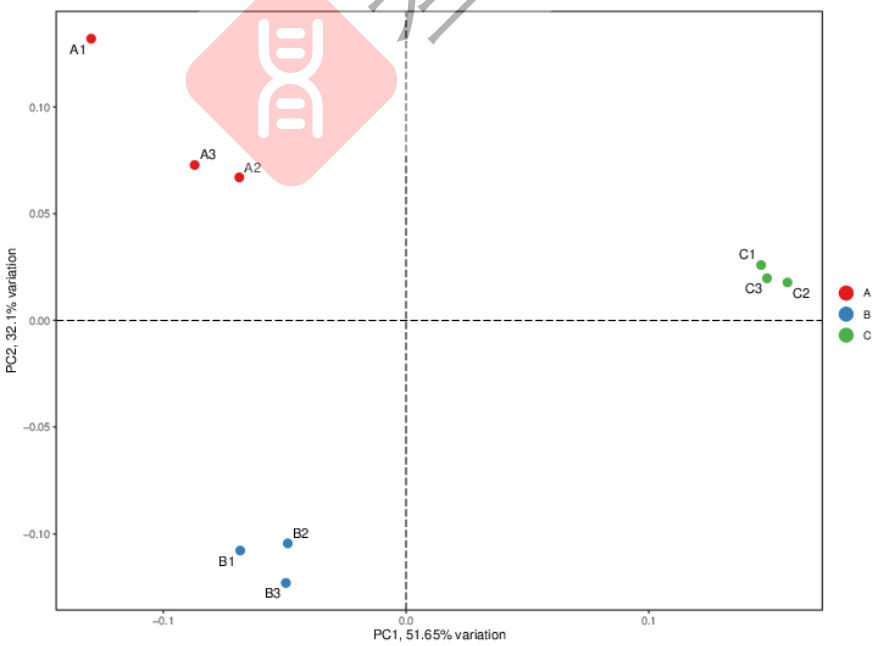
```
## 进行beta多样性分析
singularity exec ../../software/MetaGenome Rscript  \
    ../script/diversity_beta.R  S.biom  sample.txt  S.beta
```

输出结果：

S.beta.dist-braycurtis.table 距离矩阵
S.beta.NMDS.pdf NMDS-plot
S.beta.NMDS.table NMDS表格
S.beta.PCA.pdf PCA-plot
S.beta.PCA.table PCA表格
S.beta.PCoA.pdf PCoA-plot
S.beta.PCoA.table PCoA表格




Stress: 0
B3
B1
B2
C3
C2
C1
A3
A1
A2
NMDS2
NMDS1
0.50
0.25
0.00
−0.25
−0.50
−0.4
0.0
0.4
0.8
A
B
C


A1
A3
A2
C1
C3
C2
B2
B1
B3
PC2, 32.1% variation
PC1, 51.65% variation
0.10
0.05
0.00
−0.05
−0.10
−0.1
0.0
0.1
A
B
C

3.4 差异分析

差异物种分析主要基于物种丰度信息和样品分组，进行组间差异物种的检测。

这里使用 lefse 和 DESeq2/edgeR 进行差异物种分析。

lefse 用于发现两组或多组样本中最能解释组间差异的物种特征，以及这些特征对组间差异的影响程度. 该分析可以在线进行, 网址如下:

http://huttenhower.sph.harvard.edu/galaxy/

lefse 输入数据格式如图:

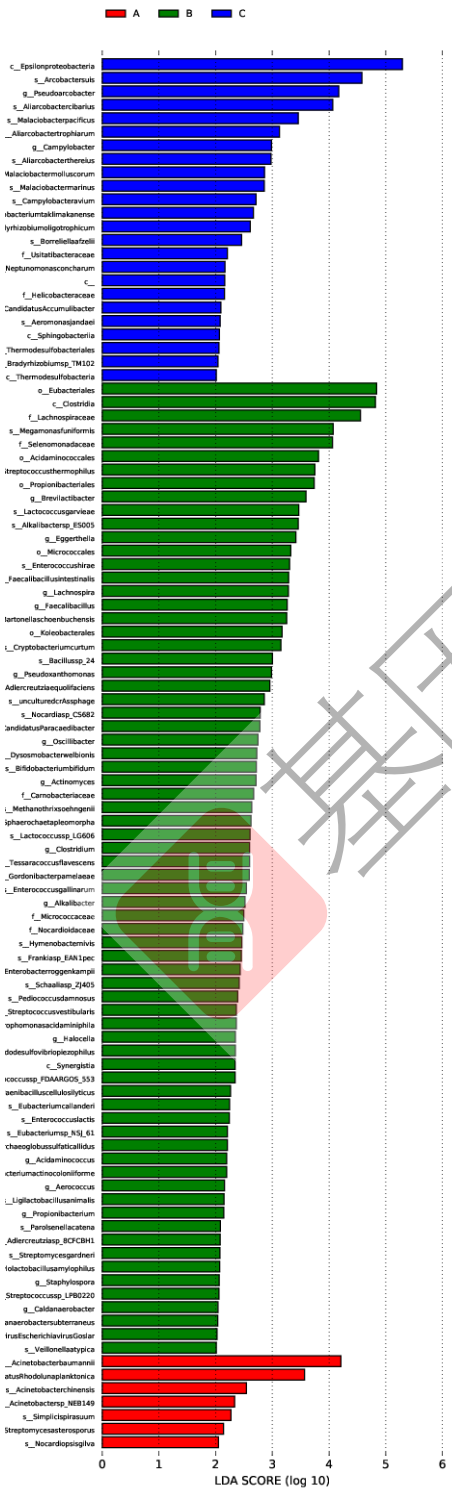
bodysite	mucosal	mucosal	mucosal	mucosal
subsite	oral	gut	oral	oral
id	1023	1023	1672	1876
Bacteria	0.99999	0.99999	0.999993	0.999989
Bacteria|Actinobacteria	0.311037	0.000864363	0.00446132	0.0312045
Bacteria|Bacteroidetes	0.0689602	0.804293	0.00983343	0.0303561
Bacteria|Firmicutes	0.494223	0.173411	0.715345	0.813046
Bacteria|Proteobacteria	0.0914284	0.0180378	0.265664	0.109549
Bacteria|Firmicutes|Clostridia	0.090041	0.170246	0.00483188	0.0465328

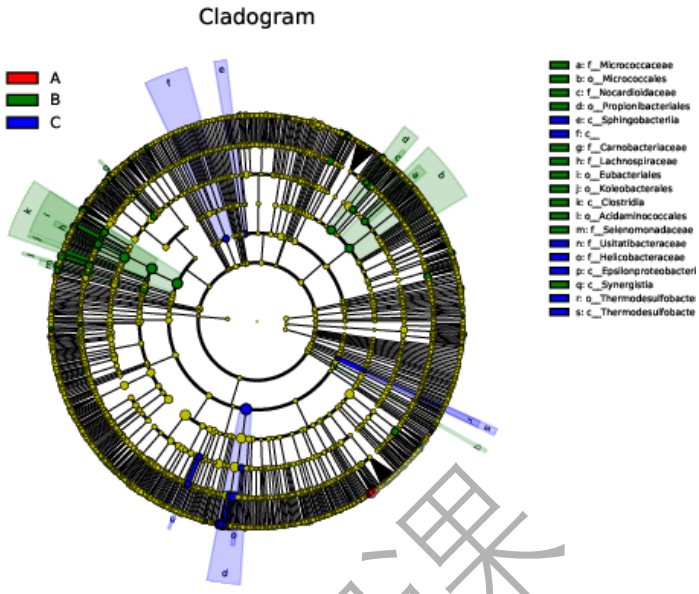
下面为本地运行 lefse 的示例:

```
## 准备输入文件
ln -s ../01.taxon/sample.txt
ln -s  ../01.taxon/out_S/S.taxName.count.tsv

## 数据准备-data部分
awk -F "\t" -v 'OFS=\t' '{$1=$NF; $NF=""; {print $0}}' \
   S.taxName.count.tsv | \
   sed 1d | \
   sed 's/; /|/g' > S.lefse.tmp

## 数据准备-class部分
head -n 1 S.lefse.tmp |sed 's/\t/\n/g' | sed '1d'  | \
  while read sp ;do awk '$2=="'$sp'" {printf "\t"$1 }' sample.txt ;done |
  awk '{print "class"$0 }' > S.lefse.header
```

```
## 合并
cat S.lefse.header S.lefse.tmp   >   S.lefse

## 格式转换
singularity exec  ../../software/lefse_latest.sif  format_input.py  \
    S.lefse \  # 输入
    S.lefse.in \  # 输出
    -c 1 \  # class所在行
    -s -1 \  # subclass所在行
    -u 2 \  # subject所在行
    -o 1000000  #  normalization 到1M

## 运行lefse
singularity exec  ../../software/lefse_latest.sif  run_lefse.py \
    S.lefse.in \  # 输入
    S.lefse.out \  # 输出
    -l 2  #  LDA阈值

##  LDA图
singularity exec  ../../software/lefse_latest.sif  plot_res.py \
    S.lefse.out \
    S.lefse.LDA.pdf \
    --format pdf

##  进化分支图
singularity exec  ../../software/lefse_latest.sif  plot_cladogram.py \
    S.lefse.out \
    S.lefse.cladogram.pdf \
    --format pdf \
    --labeled_start_lev 1
```

结果文件 S.lefse.out 为 5 列，第 1 列为分类信息，第 2 列为丰度对数，第 3 列为富集分组名称，第 4 列为 LDA 值，第 5 列为 P 值.


A
B
C
c__Epsilonproteobacteria
s__Arcobactersuis
g__Pseudoarcobacter
s__Allarcobactercibarius
s__Malaciobacterpacificus
__Allarcobactertrophilarum
g__Campylobacter
s__Allarcobacterbelreius
Malaciobactermolluscorum
s__Malaciobactermarinus
s__Campylobacteravium
obacteriumtaklimakanense
lyrhizobiumoligotrophicum
s__Borrelliellaafzelii
f__Usitatibacteraceae
Neptunomonasconcharum
c__
f__Helicobacteraceae
CandidatusAccumulibacter
s__Aeromonasjandaei
c__Sphingobacteriia
_Thermodesulfobacteriales
_Bradyrhizobiumsp_TM102
c__Thermodesulfobacteria
o__Eubacteriales
c__Clostridia
f__Lachnospiraceae
s__Megamonasfuniformis
f__Selenomonadaceae
o__Acidaminococcales
itreptococcusthermophilus
o__Propionibacteriales
g__Brevilactibacter
s__Lactococcusgarvieae
s__Alkalibactersp_ES005
g__Eggerthella
o__Micrococcales
s__Enterococcushirae
_Faecalibacillusintestinalis
g__Lachnospira
g__Faecalibacillus
lartonellaschoenbuchensis
o__Koleobacterales
i__Cryptobacteriumcurtum
s__Bacillussp_24
g__Pseudoxanthomonas
Adlercreutziaequolifaciens
s__unculturedcrAsxphage
s__Nocardiasp_CS682
CandidatusParacaedibacter
g__Oscillibacter
_Dysosmobacterwelbionis
s__Bifidobacteriumbifidum
g__Actinomyces
f__Carnobacteriaceae
i__Methanothrixsoehngenii
iphaerochaetapleomorpha
s__Lactococcussp_LG606
g__Clostridium
_Tessaracoccusflavescens
_Gordoniibacterpamelaeae
i__Enterococcusgallinarum
g__Alkalibacter
f__Micrococcaceae
f__Nocardioidaceae
s__Hymenobactemivis
s__Frankiasp_EAN1pec
Enterobacterroggenkampii
s__Schaaliasp_Zj405
s__Pediococcusdamnosus
_Streptococcusvestibularis
ophomonasacidaminiphila
g__Halocella
dodesulfovibriopiezophilus
c__Synergistia
ococcussp_FDAARGOS_553
aenibacilluscellulosilyticus
s__Eubacteriumcallanderi
s__Enterococcuslactis
s__Eubacteriumsp_NSJ_61
chaeoglobussulfaticallidus
g__Acidaminococcus
cteriumactinocoloniiforme
g__Aerococcus
i__Ligilactobacillusanimalis
g__Propionibacterium
s__Parolsenellacatena
_Adlercreutziasp_BCFCBH1
s__Streptomycesgardneri
tolactobacillusamylophilus
g__Staphylospora
_Streptococcussp_LPB0220
g__Caldanaerobacter
anaerobactersubterraneus
rustfischerichiavirusGoslar
s__Veillonellaatypica
__Acinetobacterbaumannii
atusRhodolunaplanktonica
s__Acinetobacterchinensis
_Acinetobactersp_NIEB149
s__Simplicispirasuum
Streptomycesasterosporus
s__Nocardiopsisgilva
0
1
2
3
4
5
6
LDA SCORE (log 10)

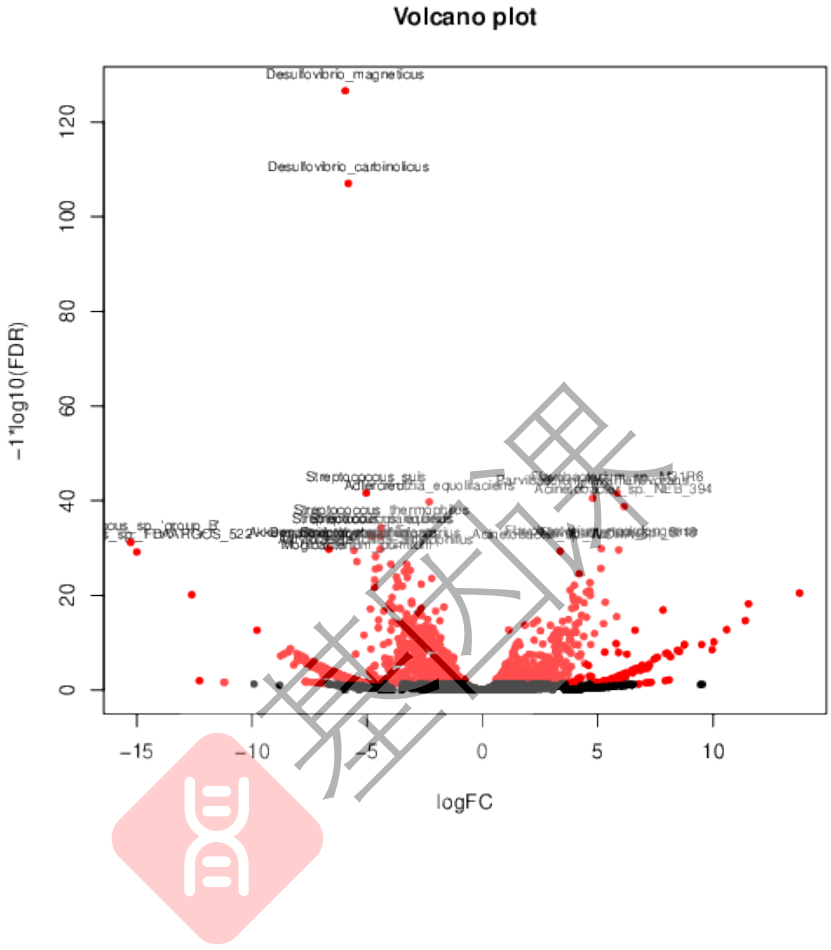


我们也可以使用 DESeq2/edgeR 这类差异分析软件进行差异物种分析:

```
## 准备count表格
ln -s ../01.taxon/out_S/S.count.tsv

## 替换掉空格
sed 's/ /_/g' S.count.tsv > S.count.matrix

## 进行差异分析
singularity exec ../../software/MetaGenome.sif perl \
    ../script/run_DE_analysis.pl \
    --matrix S.count.matrix \  # count表格
    --method DESeq2 \  # 分析方法，可设置为edgeR
    --samples_file sample.txt \  # 样品分组信息
    --output DESeq2_out  # 输出结果目录
```

输出结果文件为: S.count.matrix.A_vs_B.DESeq2.DE_results ; 可基于 FDR 和 Foldchange 进行差异物种筛选。常用标准为:

FDR < 0.05 && |log2(FC)|>1




基因课

第四章　宏基因组数据组装及注释

该部分讲解宏基因组测序数据的组装、注释及基因的差异分析，从功能水平研究分组之间的差异。

由于计算资源限制，一般会分样品进行组装注释。

4.1　组装

宏基因组组装常用软件为 megahit 及 metaspades。

其中 megahit 运行速度快、对内存要求低，为目前最常用软件。而 metaspedes 运行需要较大内存，请慎用。

使用 megahit 进行组装:

```
## 单组数据组装
singularity exec ../../software/MetaGenome.sif  megahit \
  -1 ../../dataFQ/A1_1.fq.gz \  # 输入，fq1
  -2 ../../dataFQ/A1_2.fq.gz \  # 输入，fq2
  --min-contig-len 1000 \  # contig最小长度
  --tmp-dir  ./ \  # 设置tmp目录
  --memory  0.4 \   # 内存占用
  --num-cpu-threads 4 \  # 线程数
  --out-dir A1_megahit \  # 输出目录
  --out-prefix  A1    # 输出前缀
```

```
## 多组数据组装，输入数据逗号分隔
singularity exec ../../software/MetaGenome.sif  megahit \
  -1 ../../dataFQ/A1_1.fq.gz,../../dataFQ/A2_1.fq.gz \
  -2 ../../dataFQ/A1_2.fq.gz,../../dataFQ/A2_2.fq.gz \
  --min-contig-len 1000 \
  --tmp-dir  ./ \
  --memory  0.4 \
  --num-cpu-threads 4 \
  --out-dir multi_megahit \
  --out-prefix  multi
```

输出结果文件: A1_megahit/A1.contigs.fa

使用 metaspades 进行组装:

```
## 单组数据组装
singularity  exec ../../software/MetaGenome.sif spades.py  \
    --meta \  # 宏基因组模式
    -t 4 \  #  线程
    -k 21,33 \  # kmer
    -1 ../../dataFQ/A1_1.fq.gz \  # 输入，fq1
    -2 ../../dataFQ/A1_2.fq.gz \  # 输入，fq2
    -o A1_metaspades  # 输出目录

##  多组数据组装
singularity  exec ../../software/MetaGenome.sif spades.py  --meta \
    -t 4 \
    -k 21,33,55,77 \
    --pe-1 1 ../../dataFQ/A1_1.fq.gz \  #输入，第1组fq1
    --pe-2 1 ../../dataFQ/A1_2.fq.gz \  #输入，第1组fq2
    --pe-1 2 ../../dataFQ/A2_1.fq.gz \  #输入，第2组fq1
    --pe-2 2 ../../dataFQ/A2_2.fq.gz \  #输入，第2组fq2
```

```
    -o multi_metaspades  #输出目录
```

输出结果文件：A1_metaspades/scaffolds.fasta

组装结果可以使用 quast 进行汇总统计：

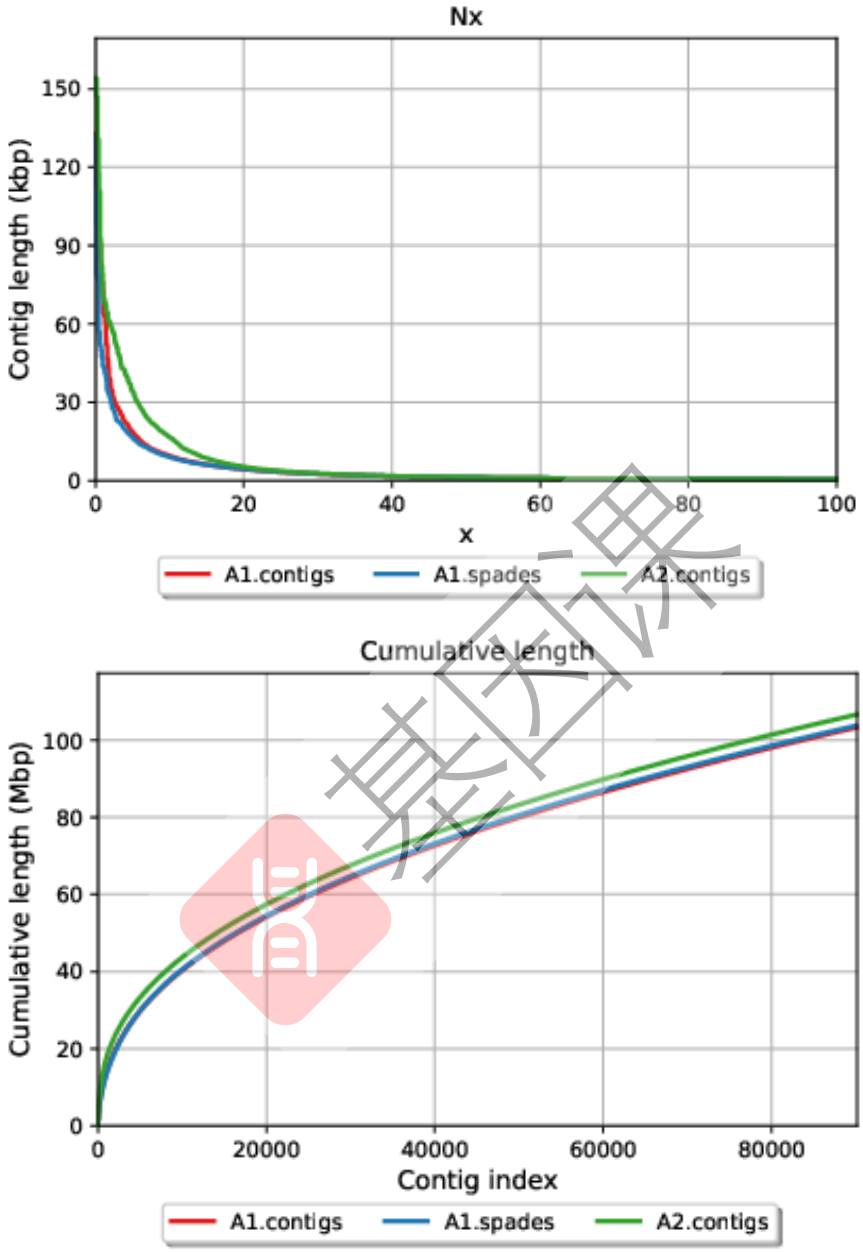
```
# 准备组装结果
ln -s ../01.megahit/A1_megahit/A1.contigs.fa
ln -s ../01.megahit/A1_megahit/A1.contigs.fa
ln -s ../01.metaspades/A1_metaspades/scaffolds.fasta  ./A1.spades.fa

# 使用quast进行汇总统计
singularity  exec ../../software/MetaGenome.sif quast.py ./*.fa
```

输出结果为网页版报告：report.html

Report

	A1.contigs	A1.spades	A2.contigs
# contigs (>= 0 bp)	90152	815775	90115
# contigs (>= 1000 bp)	25351	25898	25401
# contigs (>= 5000 bp)	1791	1907	1737
# contigs (>= 10000 bp)	512	518	646
# contigs (>= 25000 bp)	87	70	160
# contigs (>= 50000 bp)	21	11	46
Total length (>= 0 bp)	103373034	288979115	106698848
Total length (>= 1000 bp)	59857505	60794935	63184115
Total length (>= 5000 bp)	18409789	18598660	22002734
Total length (>= 10000 bp)	9758166	9263605	14589396
Total length (>= 25000 bp)	3704212	2794187	7148047
Total length (>= 50000 bp)	1635921	785191	3220767
# contigs	90152	89901	90115
Largest contig	132685	130981	153954
Total length	103373034	103729601	106698848
GC (%)	49.21	48.29	47.62
N50	1221	1238	1270
N90	572	571	578
auN	4625.822610856135	4053.6239567623516	6427.007100376566
L50	17916	17823	16605
L90	70823	70440	70239
# N's per 100 kbp	0.00	81.82	0.00

All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).



4.2 基因预测及去冗余

prokka 和 metaGeneMark 均为常用宏基因组注释软件，这里使用 prokka 进行基因预测。

```
# 准备输入文件
ln -s ../01.megahit/A1_megahit/A1.contigs.fa

# 基因预测
singularity exec ../../software/MetaGenome.sif  prokka  \
  --outdir A1_prokka \  # 输出目录
  --prefix A1 \  # 输出文件前缀
  --addgenes \  # 添加gene信息
  --addmrna \  # 添加mrna信息
  --locustag  A1 \  # 基因ID添加tag
  --kingdom  Bacteria \  #物种分类Archaea|Bacteria|Mitochondria|Viruses
  --metagenome \  # 宏基因组模式
  --cpus 8 \  # 线程数
  ./A1.contigs.fa  # 输入文件
```

主要结果:

```
A1.faa : 预测的氨基酸序列文件
A1.ffn : 预测出的转录本序列
A1.fna : 输入的基因组文件
A1.fsa : 添加Sequin tags的基因组文件
A1.gbf : Genbank格式注释结果
A1.gff : gff3格式注释结果
A1.sqn : ASN1 格式 Sequin文件，修改后可用于提交Genbank
A1.tbl : Feature Table，创建sequin文件使用
A1.tsv : features: locus_tag,ftype,len_bp,gene,EC_number,COG,product
A1.txt : 注释结果统计信息
```

对多个样品基因预测结果去冗余，得到 unigene:

```
## 将多样品转录本和氨基酸序列分别合并
cat  ../02.prokka/*_prokka/*.ffn > all.trans.fa
cat  ../02.prokka/*_prokka/*.faa > all.pep.fa
```

```
## 提取编码序列ID
sed -n "s/^>\(\S\+\).*$/\1/p"  all.pep.fa > all.cds.id

## 基于ID提取CDS序列
singularity exec  ../../software/MetaGenome.sif seqtk subseq \
    all.trans.fa all.cds.id > all.cds.fa

## 对CDS进行聚类
singularity exec  ../../software/MetaGenome.sif  cd-hit-est \
    -i all.cds.fa \
    -o  all.cds.cdhit \
    -c 0.95 \
    -aS 0.9 \
    -M 10000 \
    -T 4
```

输出结果文件:

```
all.cds.cdhit :  unigene 序列文件
all.cds.cdhit.clstr: 聚类cluster信息
```

由于 unigene 序列中使用原始的序列名称，这里我们可以将其统一修改成 clusterID:

```
cp  all.cds.cdhit  unigene_cds.fasta

# 提取原始unigene ID
awk '$1 ~/^>/{print $1}'  all.cds.cdhit | \
    sed 's/^>//' > unigene.id

# 基于unigene ID 提取其氨基酸序列文件
```

```
singularity exec  ../../software/MetaGenome.sif seqtk  subseq  \
    all.pep.fa unigene.id > unigene_pep.fasta

# 提取cluster ID和原始unigeneID 对应关系
awk '{if($1~/^>/){printf $1 $2} else if($NF ~ /*$/){print "\t"$3}}' \
    all.cds.cdhit.clstr | \
    sed 's/>//g; s/...$//'| \
    awk '{print $2"\t"$1}'  > map_id.txt

# 对原始unigeneID进行替换(原文件修改)
../script/map_fasta_ids  map_id.txt  unigene_cds.fasta
../script/map_fasta_ids  map_id.txt  unigene_pep.fasta
```

输出结果:

```
unigene_cds.fasta : unigene CDS序列文件
unigene_pep.fasta : unigene 氨基酸序列文件
```

4.3 功能注释

功能注释主要有两种方法，基于序列相似度以及结构相似度。

4.3.1 eggnog 注释

我们使用在线版 eggnog-mapper 进行 eggnog 数据库注释。

由于在线版 eggnog-mapper 单次输入序列条数不能超 10 万条，因此对氨基酸序列进行切分。

```
$ ln -s  ../03.cdhit/unigene_pep.fasta

$ seqkit split  --by-size 100000 --out-dir split  unigene_pep.fasta
```

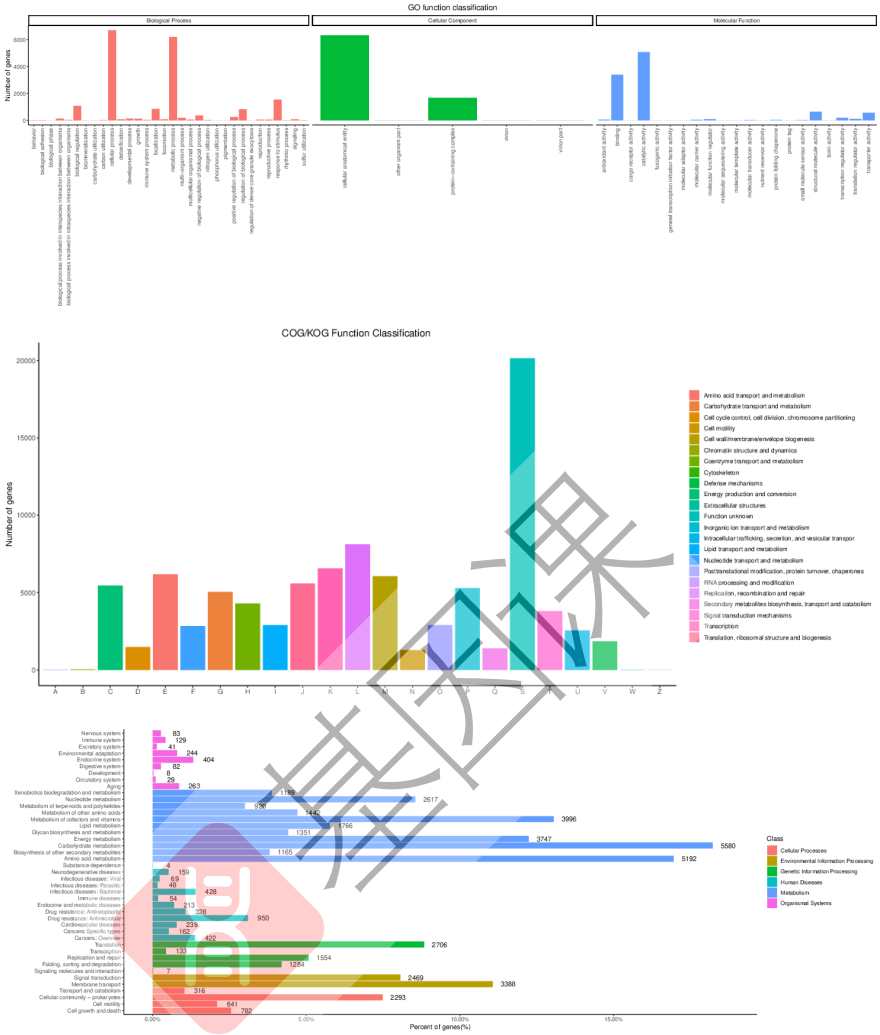
```
$ ls -l  split
unigene_pep.part_001.fasta
unigene_pep.part_002.fasta
```

数据准备好以后，登录 http://eggnog-mapper.embl.de/ 上传蛋白序列, 在线进行蛋白注释，完成后下载注释结果 *emapper.annotations，并将其合并。

```
awk '{if(NR==FNR || $1 !~ /^#/){print $0}}' ./*.emapper.annotations  \
    >  unigene.annotations

singularity exec ../../software/MetaGenome.sif Rscript \
    ../script/emcp/emapperx.R \
    ounigene.annotations unigene_pep.fasta
```

主要输出结果如下：

```
anno_stat.txt  : 注释结果统计
cog.pdf  # cog分类图
cog.txt  # cog分类统计表
pathway_stat.txt  # kegg pathway 统计表
pathway.txt # pathway 注释结果
pathway.pdf # pathway 分类图
go.pdf  # go分类图
go.txt  # go注释结果
R_Library # 构建好的R library, 用于后续富集分析
```



4.3.2 uniprot 注释

由于 eggnog 数据库对细菌的 GO 注释信息包含很少，这里我们使用 uniref90 数据库对氨基酸序列进行比对注释，并基于 idmapping 信息提取 GO 注释。

```
## 提前准备好uniref90数据库

### 下载uniref90序列文件31G
```

```
# wget  https://ftp.uniprot.org/pub/databases/uniprot/uniref/\
#    uniref90/uniref90.fasta.gz
### 下载id mapping文件 11G
# wget https://ftp.uniprot.org/pub/databases/uniprot/current_release/\
#    knowledgebase/idmapping/idmapping_selected.tab.gz

## 构建diamond database
# singularity exec ../../software/MetaGenome.sif diamond  makedb \
#    --in  ./uniref90.fasta.gz \
#    --db  uniref90

## 以上命令会生成uniref90.dmnd 用于后续diamond比对

ln -s ../03.cdhit/unigene_pep.fasta
## 进行diamond比对
singularity exec ../../software/MetaGenome.sif diamond blastp  \
    --db ../../Database/uniprot/uniref90 \  # db名称
    --query  unigene_pep.fasta \ # 输入，氨基酸fasta文件
    --out unigene_pep.uniref90.m6 \  # 输出，表格格式
    --threads 5 \  # 线程
    --outfmt 6 \  # 输出blast表格格式结果
    --max-target-seqs 5 \
    --evalue 1e-5

## 基于idmapping 提取GO注释信息
singularity exec ../../software/MetaGenome.sif  perl \
    ../script/uniref90_idmapping.pl \
    unigene_pep.uniref90.m6 \  # 比对结果
    ../../Database/uniprot/idmapping_selected.tab.gz \  # idmapping文件
    > unigene_pep.uniref90.GOanno  # 输出GO注释信息

## 构建orgdb，并统计GO 注释
singularity exec ../../software/MetaGenome.sif Rscript  \
```

```
    ../script/emcp/GOmapperx.R \
    unigene_pep.uniref90.GOanno
```

输出结果：

```
go.pdf  # level2 go柱状图
go.txt  # 绘图数据
org.My.eg.db_1.0.tar.gz #只包含GO注释的orgdb
R_Library/  # 用来后续富集分析的orgdb library
```

4.3.3 抗生素耐药基因注释

抗生素耐药基因基于 CARD 数据库进行注释，CARD 提供了 RGI 工具用于注释耐药基因。

RGI 可以网页版运行，也可以安装在本地服务器，下面是网页版地址：

https://card.mcmaster.ca/analyze/rgi

使用 RGI 进行抗生素耐药基因注释：

```
## 下载CARD数据库参考数据
# wget https://card.mcmaster.ca/latest/data
# tar -xvf data ./card.json

## 准备输入文件
ln -s ../03.cdhit/unigene_pep.fasta

## 加载数据库
singularity exec  ../../software/rgi.sif rgi load \
    -i ../../Database/CARD/card.json \  # 输入
    --local  # 数据库存放在当前目录

# 生成数据库目录localDB/
```

```
## 运行主程序
singularity exec ../../software/rgi.sif rgi main \
    --input_sequence  unigene_pep.fasta \  # 输入， 氨基酸序列
    --output_file rgi_out \  # 输出文件前缀
    --input_type protein \   # 输入数据类型contig/protein
    --alignment_tool DIAMOND \  # 比对软件
    --num_threads 5 \  # 线程数
    --local \  # 数据库在当前目录
    --clean \  # 删除临时文件
    --include_loose # 保留宽松比对结果

## 关注 1,6,9,11,15,16,17
cut -f 1,6,9,11,15,16,17 rgi_out.txt > rgi_out.txt.info
```

生成结果为:rgi_out.txt ， 结果格式参考以下网址：

https://github.com/arpcard/rgi#id49

```
1:      ORF_ID
2:      Contig
3:      Start
4:      Stop
5:      Orientation
6:      Cut_Off
7:      Pass_Bitscore
8:      Best_Hit_Bitscore
9:      Best_Hit_ARO
10:     Best_Identities
11:     ARO
12:     Model_type
13:     SNPs_in_Best_Hit_ARO
14:     Other_SNPs
15:     Drug Class
16:     Resistance Mechanism
17:     AMR Gene Family
```

```
18:     Predicted_DNA
19:     Predicted_Protein
20:     CARD_Protein_Sequence
21:     Percentage Length of Reference Sequence
22:     ID
23:     Model_ID
24:     Nudged
25:     Note
```

4.3.4 碳水化合物酶注释

由于 CAZy 数据库 (网址：http://www.cazy.org/) 不提供序列下载，数据库文件下载自 dbCAN2：

https://bcb.unl.edu/dbCAN2/download/

这里使用 dbCAN2 提供的 run_dbCAN 本地版进行碳水化合物酶注释，dbCNA 也可以使用在线版，网址如下：

https://bcb.unl.edu/dbCAN2/blast.php

```
ln -s ../03.cdhit/unigene_pep.fasta

##run_dbcan输入数据格式
# protein=proteome,
# prok=prokaryote,
# meta=metagenome/mRNA/CDSs/short DNA seqs

## 使用diamond比对方法
singularity exec ../../software/run_dbcan_latest.sif  run_dbcan \
  --out_dir cazy_diamond \  # 输出目录
  --db_dir /app/db  \  # 数据库路径，已经放入容器中
  --tools diamond  \  # 使用diamond比对进行注释
  --dia_cpu 5 \  # 线程数
  unigene_pep.fasta  \  # 输入，蛋白fasta序列
  protein   # 输入数据格式
```

```
## 使用hmmer结构域搜索方法
singularity exec ../../software/run_dbcan_latest.sif  run_dbcan \
  --out_dir cazy_hmmer \
  --db_dir /app/db  \  # 数据库路径，已经放入容器中
  --tools hmmer  \  # 使用结构域搜索进行注释
  --dia_cpu 5 \
  --hmm_cpu 5 \
  --eCAMI_jobs 5 \
  unigene_pep.fasta  \
  protein
```

主要输出结果如下：

```
 head overview.txt
Gene ID EC#     HMMER   eCAMI   DIAMOND #ofTools
Cluster28873    -       -       -       GH73    1
Cluster14137    -       -       -       GH109   1
Cluster34096    -       -       -       GT2     1
Cluster12158    -       -       -       PL29    1
Cluster1200     -       -       -       PL8     1
Cluster32711    -       -       -       GT58    1
Cluster41147    -       -       -       GH3     1
```

结果整理：

```
$ sed '1d' cazy_diamond/overview.txt | cut -f 1,5 | \
    sed  's/+/\t/g' | \
    awk '{for(i=2;i<=NF; i++){print $1"\t"$i}}' | \
    awk '$2 ~ /^[A-Z]/ ' | \
```

```
    sed 's/\t\(\([A-Z]\+\).*\)$/\t\1\t\2/' >  cazy_diamond.family.txt


$ sed '1d' cazy_hmmer/overview.txt |cut -f  1,3 | \
    sed  's/+/\t/g' | \
    awk '{for(i=2;i<=NF; i++){print $1"\t"$i}}' | \
    sed 's/([0-9]\+-[0-9]\+)$//g'| \
    awk '$2 ~ /^[A-Z]/ ' |  \
    sed 's/\t\(\([A-Z]\+\).*\)$/\t\1\t\2/' >  cazy_hmmer.family.txt


$ head -n 5 cazy_diamond.family.txt
Cluster28873    GH73    GH
Cluster14137    GH109   GH
Cluster34096    GT2     GT
Cluster12158    PL29    PL
Cluster1200     PL8     PL

}
```

4.4 基因丰度估计

这里使用 salmon 软件，以 unigene 作为 reference 进行基因丰度估计。

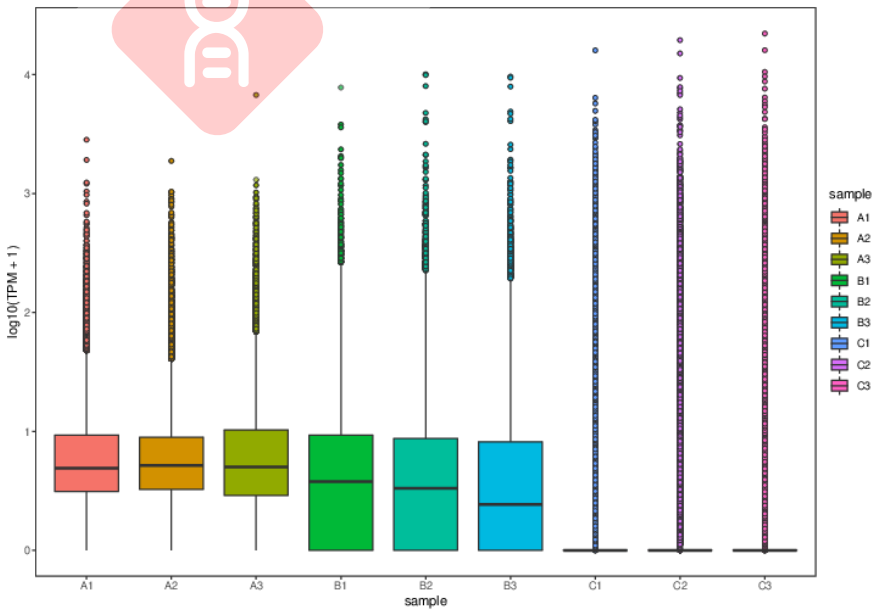
```
ln  -s   ../03.cdhit/unigene_cds.fasta

## 构建salmon index用于比对
singularity  exec  ../../software/MetaGenome.sif salmon index \
  -t  unigene_cds.fasta  \  # 输入，unigene序列文件
  -i  unigene_index  # 输出，index目录名称

## 每个样品分别定量，输出文件为quants/A1.quant/quant.sf
```

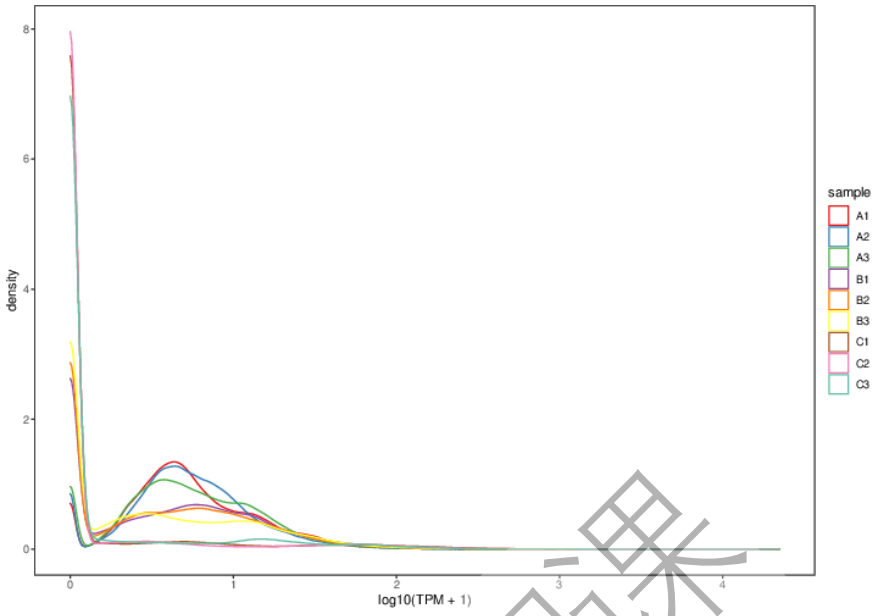
```
singularity  exec  ../../software/MetaGenome.sif salmon  quant \
  --validateMappings \  # 启用选择比对，提高灵敏度和特异性
  --meta \  # 启用宏基因组模式
  -p 4  \  # 线程
  -l IU  \  # 文库类型I表示inward , U表示unstranded
  -i  unigene_index \  # index目录
  -1 ../../dataFQ/A1_1.fq.gz \  # 输入fq1
  -2 ../../dataFQ/A1_2.fq.gz  \  # 输入fq2
  -o quants/A1.quant  # 输出目录名称

## 合并表格

# 生成每个样品定量结果列表
quants=` ls quants/ | awk '{print "quants/"$1 }' | tr '\n'  ' ' `

# 生成样品顺序
names=` ls quants/ | sed 's/.quant$//' | tr '\n'  ' ' `

## 合并生成TPM文件
singularity  exec  ../../software/MetaGenome.sif salmon quantmerge  \
    --quants $quants  \
    --names  $names  \
    --column tpm \
    -o unigenens.tpm

## 生成count文件
singularity  exec  ../../software/MetaGenome.sif salmon quantmerge  \
    --quants $quants \
    --names  $names  \
    --column numreads \
    -o unigenens.count

## 丰度结果绘图
```

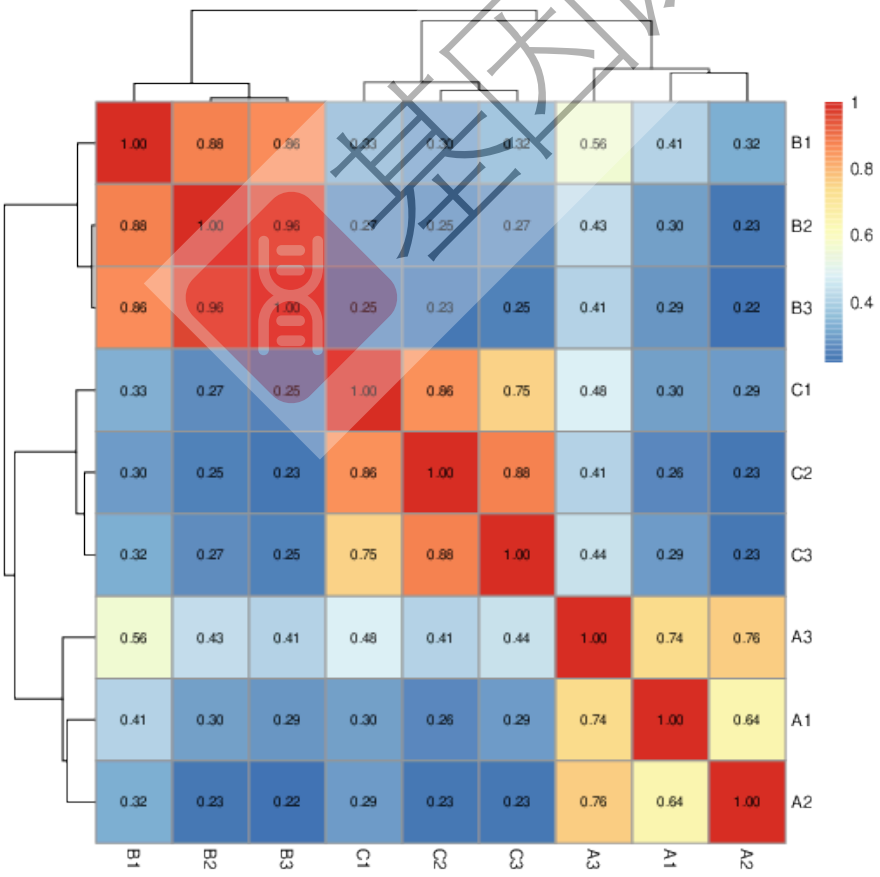
```
singularity exec ../../software/MetaGenome.sif  Rscript \
    ../script/draw_abundance.R \
    unigenens.tpm  \   # tmp表格
    ../../data/sample.txt \  # 样品分组信息文件
    unigenens.tpm_fig  # 输出图片前缀
```

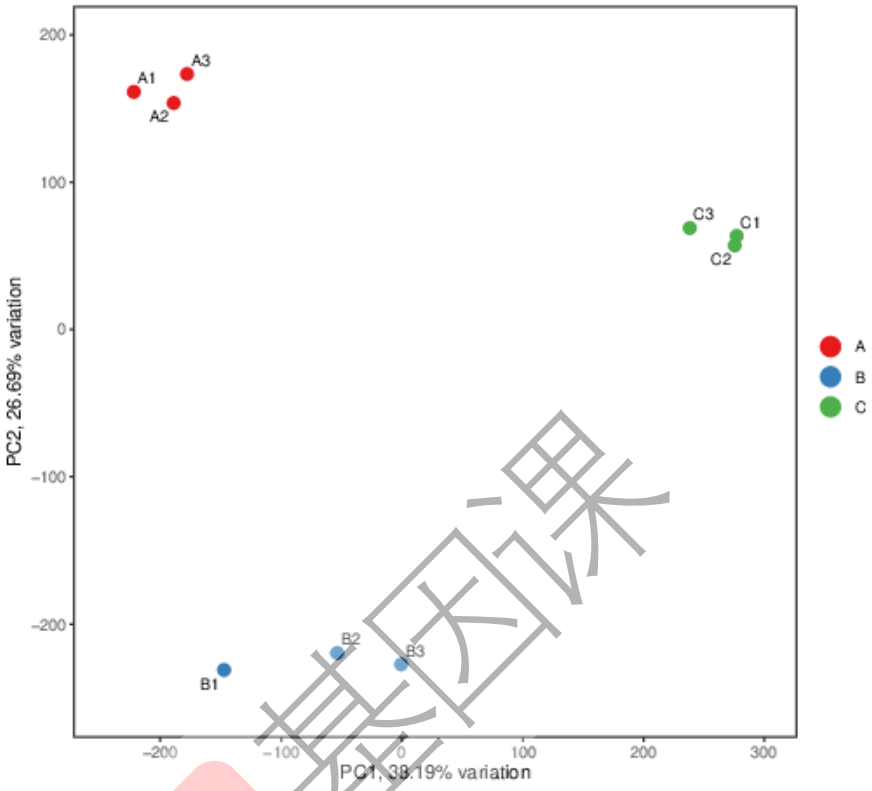
结果图片：

unigenens.tpm_fig.boxplot.pdf 箱线图
unigenens.tpm_fig.cor-cluster.pdf 相关性热图
unigenens.tpm_fig.cor.pdf 相关性热图
unigenens.tpm_fig.density.pdf 密度图
unigenens.tpm_fig.pca.pdf pca图




density
log10(TPM + 1)
sample
A1
A2
A3
B1
B2
B3
C1
C2
C3


1.00 0.88 0.86 0.33 0.30 0.32 0.56 0.41 0.32 B1
0.88 1.00 0.96 0.27 0.25 0.27 0.43 0.30 0.23 B2
0.86 0.96 1.00 0.25 0.23 0.25 0.41 0.29 0.22 B3
0.33 0.27 0.25 1.00 0.86 0.75 0.48 0.30 0.29 C1
0.30 0.25 0.23 0.86 1.00 0.88 0.41 0.26 0.23 C2
0.32 0.27 0.25 0.75 0.88 1.00 0.44 0.29 0.23 C3
0.56 0.43 0.41 0.48 0.41 0.44 1.00 0.74 0.76 A3
0.41 0.30 0.29 0.30 0.26 0.29 0.74 1.00 0.64 A1
0.32 0.23 0.22 0.29 0.23 0.23 0.76 0.64 1.00 A2
B1 B2 B3 C1 C2 C3 A3 A1 A2
1
0.8
0.6
0.4



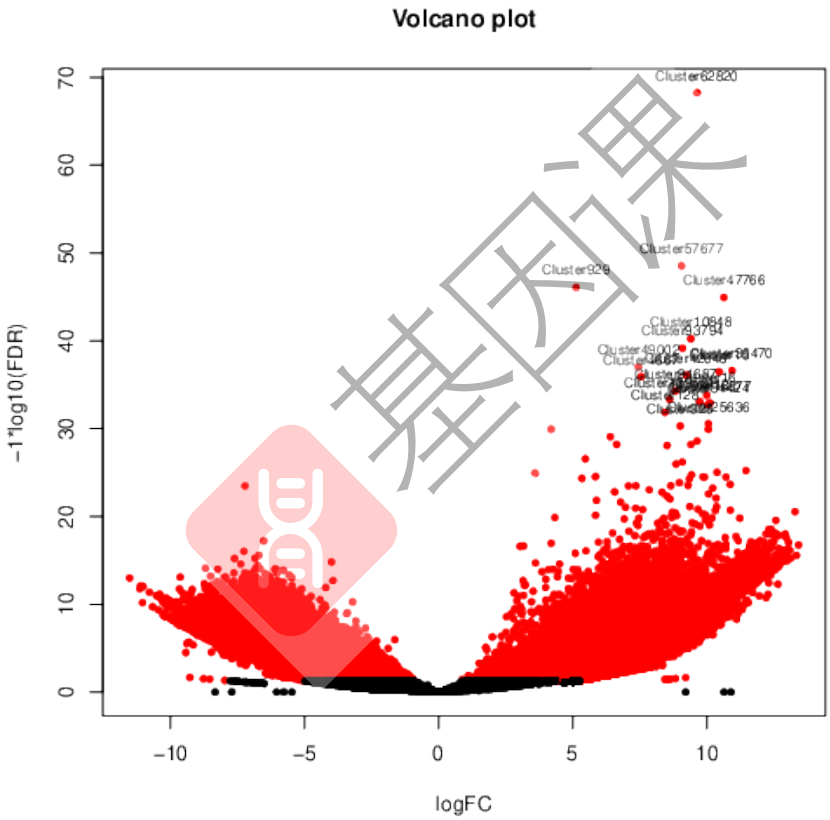
4.5 差异基因分析

这里使用 DESeq2 进行差异基因分析，输入数据为 gene count 矩阵文件和样品分组信息表。

```
# 格式转matrix
sed '1s/^Name\t//' unigenens.count > unigenens.count.matrix

## 差异表达分析 (DESeq2)
singularity  exec ../../software/MetaGenome.sif \
    ../script/run_DE_analysis.pl \
```

```
    --matrix unigenens.count.matrix  \ # 输入，count表格
    --method DESeq2 \  # 差异分析方法DESeq2/edgeR
    --samples_file  sample.txt \  # 样品分组信息表
    --contrasts contrasts.txt \  # 进行差异分析的组别
    --output DE_out  # 输出目录
```

输出结果：unigenens.count.matrix.A_vs_C.DESeq2.DE_results



4.6 差异基因富集分析

得到差异基因分析结果后，可以根据设置的差异基因标准对结果进行提取，并进行 GO 和 KEGG 富集分析。

```
## A vs C组差异基因进行筛选
## 标准为|log2FC| >1 && FDR <0.05

## 链接eggnog注释时创建的R_Library/
ln -s  ../05.uniprot/R_Library/

## 链接 eggnong注释结果表格
ln -s ../04.eggnog/unigene.annotations

singularity  exec ../../software/MetaGenome.sif Rscript \
 ../script/enrich_analysis.R \
 ./DE_out/unigenens.count.matrix.A_vs_C.DESeq2.DE_results \ #输入DE_resul
 ./unigene.annotations \  # 输入, eggnong表格
 ./DE_out/unigenens.count.matrix.A_vs_C.DESeq2.DE_results.enrich  #输出前
```


基因课

第五章　宏基因组分箱

对于宏基因组组装结果，可以基于覆盖度和序列碱基组成，使用分箱软件对其进行分箱操作。

5.1　分箱

Metabat2，maxbin2 和 concoct 是三款常用的分箱软件；下面为这三款软件的示例脚本，需要准备的输入数据为宏基因组拼接结果和质控过滤的测序数据。

- 使用 metabat2 进行分箱：

使用metabat2进行分箱：

```
## 构建index
singularity exec  ../../software/MetaGenome.sif  bowtie2-build \
    A1.contigs.fa \
    A1.contigs.db

## 比对并排序
singularity exec  ../../software/MetaGenome.sif  bowtie2 \
  --threads 4 -x ./A1.contigs.db \
  -1 ../../dataFQ/A1_1.fq.gz \
  -2  ../../dataFQ/A1_2.fq.gz  2>A1.map.log | \
  singularity exec  ../../software/MetaGenome.sif samtools sort \
```

```
  -o A1.sort.bam

## 运行分箱
singularity exec  ../../software/MetaGenome.sif runMetaBat.sh  \
    A1.contigs.fa \  # 输入， 组装结果
    A1.sort.bam \  # 输入，bam文件
    --minContig 1000  # contig最小长度
```

输出结果：A1.contigs.fa.metabat-bins/

- 使用 maxbin2 进行分箱：

```
## 创建目录
mkdir A1.bindir

## 该软件自动调用bowtie2进行比对
singularity  exec  ../../software/MetaGenome.sif run_MaxBin.pl \
  -contig A1.contigs.fa \  # 输入， 组装结果
  -reads  ../../dataFQ/A1_1.fq.gz \  # 输入，测序数据
  -reads2  ../../dataFQ/A1_2.fq.gz \  # 输入，测序数据
  -out A1.bindir/A1.bin \
  -min_contig_length 1000 \
  -thread 10
```

输出结果：A1.bindir/A1.bin*.fasta

- 使用 concoct 进行分箱：

```
## bowtie2 比对
singularity exec  ../../software/MetaGenome.sif  bowtie2-build \
    A1.contigs.fa \
```

```
    A1.contigs.db

singularity exec  ../../software/MetaGenome.sif  bowtie2 \
  --threads 4 -x ./A1.contigs.db  \
  -1 ../../dataFQt/A1_1.fq.gz \
  -2 ../../dataFQt/A1_2.fq.gz  2>A1.map.log | \
  singularity exec  ../../software/MetaGenome.sif samtools \
    sort -o A1.sort.bam

singularity exec  ../../software/MetaGenome.sif samtools \
  index  A1.sort.bam

## 数据划分
singularity  exec ../../software/MetaGenome.sif cut_up_fasta.py  \
    -c 10000 -o 0 \
    --merge_last \
    -b contigs_10K.bed  \
    A1.contigs.fa > contigs_10K.fa

## 生成coverage信息
singularity exec ../../software/MetaGenome.sif concoct_coverage_table.py
    contigs_10K.bed  ./*.sort.bam > coverage_table.tsv

## 运行分箱
singularity exec ../../software/MetaGenome.sif concoct \
    --composition_file contigs_10K.fa \  # 输入，fasta序列
    --coverage_file coverage_table.tsv \  # 输入，coverage信息
    -b concoct_output/  \  # 输出，结果目录
    --threads 5 \  # 线程数
    --read_length 150 \  # read长度
    --length_threshold 1000  # contig阈值

## subcontig 合并回原始contig
```

```
singularity exec ../../software/MetaGenome.sif merge_cutup_clustering.py \
    concoct_output/clustering_gt1000.csv \
    > concoct_output/clustering_merged.csv

## 构建bin输出目录
mkdir concoct_output/fasta_bins


## 提取bin序列
singularity exec ../../software/MetaGenome.sif extract_fasta_bins.py \
  A1.contigs.fa  \
  concoct_output/clustering_merged.csv \
  --output_path concoct_output/fasta_bins
```

5.2 分箱结果评估

分箱结果可以使用 checkM 检查完整性和污染度。

```
## 链接分箱结果到当前目录
ln -s  ../01.metabat2/A1.contigs.fa.metabat-bins/ ./bins

## 运行checkM
singularity  exec ../../software/MetaGenome.sif checkm lineage_wf \
    --threads 5 \  #  线程
    --tmpdir ./ \  # tmp目录路径
    --extension fa \  # 序列文件后缀
    bins \  # 输入，分箱结果目录
    bins_checkm \   # 输出目录
    > checkM.sh.log 2>&1 # 存储日志
```

Bin Id	Marker lineage	# genomes	# markers	# marker sets	0	1	2	3	4	5+	Completeness	Contamination	Strain heterogeneity
bin.3	o__Burkholderiales (UID4000)	193	427	214	30	391	6	0	0	0	94.67	1.79	33.33
bin.13	c__Gammaproteobacteria (UID4442)	899	312	185	28	284	0	0	0	0	94.59	0.00	0.00
bin.15	k__Bacteria (UID3060)	138	338	246	17	316	5	0	0	0	94.11	1.08	80.00
bin.5	o__Bacteroidales (UID2654)	163	485	265	54	332	97	2	0	0	94.05	20.65	34.95
bin.11	k__Bacteria (UID203)	5449	104	58	61	43	0	0	0	0	57.02	0.00	0.00
bin.7	k__Bacteria (UID203)	5449	104	58	65	36	3	0	0	0	53.78	5.17	66.67
bin.9	f__Flavobacteriaceae (UID2817)	81	511	283	239	264	8	0	0	0	52.35	1.27	0.00
bin.8	o__Burkholderiales (UID4000)	193	427	214	204	216	7	0	0	0	48.16	2.41	0.00
bin.12	k__Bacteria (UID203)	5449	104	58	75	26	3	0	0	0	44.83	5.17	66.67
bin.6	k__Bacteria (UID203)	5449	104	58	83	20	1	0	0	0	34.48	1.72	100.00
bin.16	k__Bacteria (UID203)	5449	104	58	82	22	0	0	0	0	29.31	0.00	0.00
bin.14	k__Bacteria (UID203)	5449	104	58	83	21	0	0	0	0	25.86	0.00	0.00
bin.2	k__Bacteria (UID203)	5449	103	57	86	17	0	0	0	0	25.44	0.00	0.00
bin.10	k__Bacteria (UID203)	5449	104	58	88	16	0	0	0	0	19.83	0.00	0.00
bin.4	root (UID1)	5656	56	24	53	3	0	0	0	0	12.50	0.00	0.00
bin.1	root (UID1)	5656	56	24	56	0	0	0	0	0	0.00	0.00	0.00

5.3 metaWRAP 使用

metaWARP 可以整合运行以上三款分箱软件，并可以自动运行 checkM，也可以对不同软件分箱结果进行整合。一步到位，操作简单。

```
## 准备宏基因组组装结果
ln -s  ../../P3.Assembly_annotation/01.megahit/A1_megahit/A1.contigs.fa

## 解压测序数据
gzip -dc ../../dataFQ/A1_1.fq.gz > A1_1.fastq
gzip -dc ../../dataFQ/A1_2.fq.gz > A1_2.fastq

## 使用metaWRAP进行分箱，并运行checkM
singularity exec ../../software/MetaGenome.sif  metaWRAP binning  \
    -t 6 \
    --metabat2 \  # 允许metabat2
    --maxbin2 \  # 允许maxbin2
    --run-checkm  \  # 对分箱结果运行checkM
    -a  A1.contigs.fa \  # 输入，组装结果
    -o metawrap_bin \  # 输出目录
    A1_1.fastq  \  # 输入fq1
    A1_2.fastq    # 输入fq2


## 多个软件分箱结果整合
singularity exec ../../software/MetaGenome.sif metaWRAP bin_refinement \
    -t  6 \
```

```
    -o metawrap_refine \  # 输出结果目录
    -A metawrap_bin/maxbin2_bins/ \  # 输入，分箱结果1
    -B metawrap_bin/metabat2_bins/  # 输入，分箱结果2

## refine结果为metawrap_refine/metawrap_70_10_bins*
```

5.4 分箱结果去冗余

计算资源充足的情况下，可以将所有测序数据一起进行组装，然后分箱。

但实际操作中，由于内存限制，一般会分样品或者将生物学重复放在一起进行拼接。这样就会存在多个组装和分箱结果，需要对其中重复的 bin 去冗余。这个操作可以使用 dRep 实现。

```
## 将所有分箱fasta文件存放在bin目录下

## 运行dRep去冗余
singularity exec ../../software/MetaGenome.sif dRep dereplicate  \
    out_dRep \
    --length 50000 \
    -comp 75 -con 25 \
    -g ./bins/*.fasta
```

结果存放于 out_dRep/dereplicated_genomes 目录

5.5 确定物种分类

当我们得到高质量 bin 以后，可以使用 GTDB-tk 对物种基于进化关系进行分类学注释。

gtdb-tk 运行时需要使用 50G 的 GTDB 数据库，要提前准备好。下载地址如下：

https://data.gtdb.ecogenomic.org/releases/latest/auxillary_files/gtdbtk_data.tar.gz

注意：该软件数据库大小为 50G，运行时需要消耗 200G 以上内存，请勿在基因课服务器运行。

```
## 将需要进行分类学注释的bin存放于genomes目录
## 运行gtdbtk
singularity  exec \
    -B ../../Database/GTDB/refdata/:/refdata \  # 挂载数据库到容器中
    ../../software/gtdbtk_latest.sif gtdbtk classify_wf \
    --genome_dir  ./genomes/  \  # 基因组文件目录
    --out_dir  out_gtdbtk \  # 输出结果目录
    --cpus 5 \  # 线程
    --extension fa \  # 基因组序列文件后缀
    --tmpdir ./
```

输出结果：

out_gtdbtk/gtdbtk.bac120.summary.tsv ： bin 物种分类信息

out_gtdbtk/gtdbtk.bac120.classify.tree ： 合并其他物种构建的 tree，可以使用 Dendroscope 查看


基因课