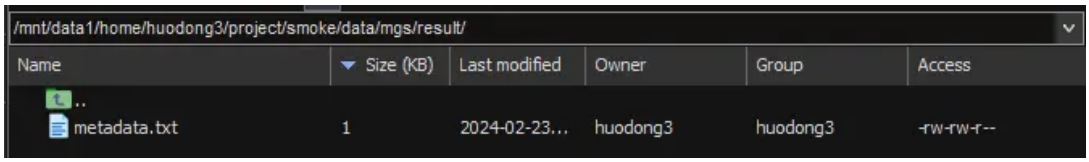
基于组装的宏基因组上游流程

数据文件夹以及数据库准备

```
# 公共数据库database(db)位置，如管理员设置/db，个人下载至~/db
# Conda软件安装目录，`conda env list`查看，如/anaconda3
soft=~/miniconda3
db=/home/ubuntu/database/
# 设置工作目录work directory(wd)，如meta
wd=/home/ubuntu/project/test_mgs/
# 创建并进入工作目录
mkdir -p $wd && cd $wd
# 创建3个常用子目录：序列，临时文件和结果
mkdir -p seq temp result
#加载环境
conda activate mgs
```

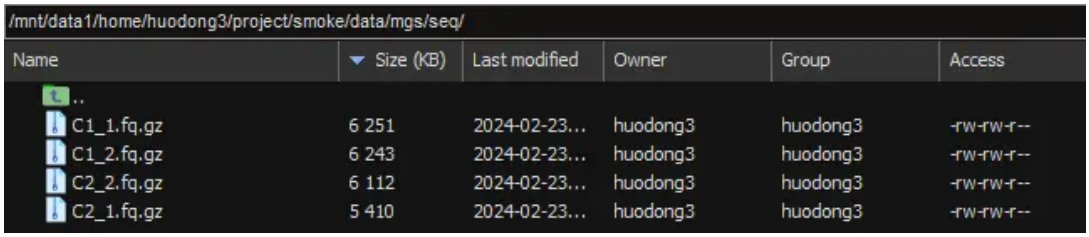
移动metadata到/home/ubuntu/project/test_mgs/result/ 文件夹下

metadata位置：/home/ubuntu/project/test_mgs/EasyMetagenome/result/



移动原始序列到/home/ubuntu/project/test_mgs/seq/

原始序列位置：/home/ubuntu/project/test_mgs/EasyMetagenome/seq/



查看序列大小

```
# 查看文件大小
ls -lsh seq
#显示文件结构
tree -L 2
```

kneaddata数据质控

```
time parallel -j 2 --xapply "kneaddata -i1 seq/{1}_1.fq.gz -i2 seq/{1}_2.fq.gz -o temp/qc -v -t 20 --remove-intermediate-output --trimmomatic /home/ubuntu/Software/Trimmomatic-0.39/ --reference-db /home/ubuntu/database/kneaddata/human_genome/" ::: `tail -n+2 result/metadata.txt|cut -f1`
```

```
Final output files created:
/mnt/data1/home/huodong3/project/smoke/data/mgs/temp/qc/C2_1_kneaddata_paired_1.fastq
/mnt/data1/home/huodong3/project/smoke/data/mgs/temp/qc/C2_1_kneaddata_paired_2.fastq
/mnt/data1/home/huodong3/project/smoke/data/mgs/temp/qc/C2_1_kneaddata_unmatched_1.fastq
/mnt/data1/home/huodong3/project/smoke/data/mgs/temp/qc/C2_1_kneaddata_unmatched_2.fastq
```

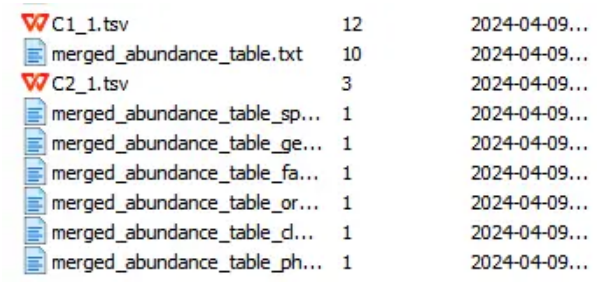
质控结果汇总

```
# 采用kneaddata附属工具kneaddata_read_count_table
kneaddata_read_count_table --input temp/qc --output temp/kneaddata.txt
# 筛选重点结果列
cut -f 1,2,4,12,13 temp/kneaddata.txt | sed 's/_1_kneaddata//' > result/qc/sum.txt
cat result/qc/sum.txt
```

metaphlan物种组成

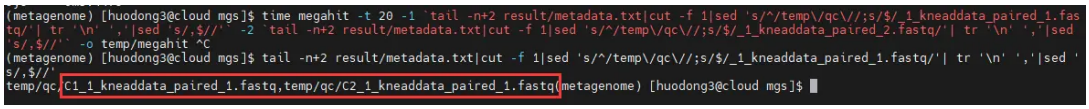
```
##数据合并
time parallel -j 2 --xapply "cat temp/qc/{1}_1_kneaddata_paired_1.fastq temp/qc/{1}_1_kneaddata_paired_2.fastq > temp/cat/{1}_1.fastq" ::: `tail -n+2 result/metadata.txt|cut -f1`
##metaphlan
time parallel -j 2 --xapply "metaphlan temp/cat/{1}_1.fastq --input_type fastq -o result/metaphlan4/{1}_1.tsv --nproc 20 --no_map" ::: `tail -n+2 result/metadata.txt|cut -f1`
##合并
cd ./result/metaphlan4/
merge_metaphlan_tables.py *.tsv > merged_abundance_table.txt
##按照界门纲目科属种来进行分类
grep -E '(s__)|(clade_name)' merged_abundance_table.txt |grep -v 't__'|sed 's/^.*s__//g'|sed 's/\ \ /\ /g'|sed 's/\ /\t/g' > merged_abundance_table_species.txt
grep -E '(g__)|(clade_name)' merged_abundance_table.txt |grep -v 's__'|sed 's/^.*g__//g'|sed 's/\ \ /\ /g'|sed 's/\ /\t/g' > merged_abundance_table_genus.txt
grep -E '(f__)|(clade_name)' merged_abundance_table.txt |grep -v 'g__'|sed 's/^.*f__//g'|sed 's/\ \ /\ /g'|sed 's/\ /\t/g' > merged_abundance_table_family.txt
grep -E '(o__)|(clade_name)' merged_abundance_table.txt |grep -v 'f__'|sed 's/^.*o__//g'|sed 's/\ \ /\ /g'|sed 's/\ /\t/g' > merged_abundance_table_order.txt
grep -E '(c__)|(clade_name)' merged_abundance_table.txt |grep -v 'o__'|sed 's/^.*c__//g'|sed 's/\ \ /\ /g'|sed 's/\ /\t/g' > merged_abundance_table_class.txt
grep -E '(p__)|(clade_name)' merged_abundance_table.txt |grep -v 'c__'|sed 's/^.*p__//g'|sed 's/\ \ /\ /g'|sed 's/\ /\t/g' > merged_abundance_table_phylum.txt

```



MEGAHIT拼接

```
time megahit -t 20 -1 `tail -n+2 result/metadata.txt|cut -f 1|sed 's/^/temp\/qc\//;s/$/_1_kneaddata_paired_1.fastq/'| tr '\n' ','|sed 's/,$//'` -2 `tail -n+2 result/metadata.txt|cut -f 1|sed 's/^/temp\/qc\//;s/$/_1_kneaddata_paired_2.fastq/'| tr '\n' ','|sed 's/,$//'` -o temp/megahit
```



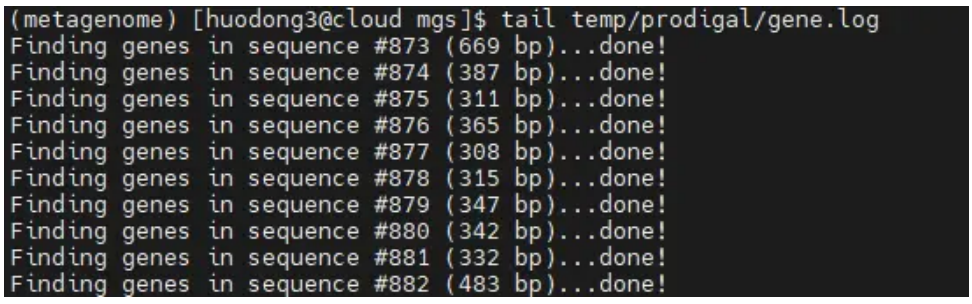

```
# 备份重要结果
mkdir -p result/megahit/
ln -f temp/megahit/final.contigs.fa result/megahit/
# 删除临时文件
rm -rf temp/megahit/intermediate_contigs/
```

Prodigal基因预测

```
time prodigal -i result/megahit/final.contigs.fa -d temp/prodigal/gene.fa -o temp/prodigal/gene.gff -p meta -f gff > temp/prodigal/gene.log 2>&1
```

查看日志

```
# 查看日志是否运行完成，有无错误
tail temp/prodigal/gene.log
```



命令运行

```
# 统计基因数量
grep -c '>' temp/prodigal/gene.fa
# 统计完整基因数量，数据量大可只用完整基因部分
grep -c 'partial=00' temp/prodigal/gene.fa
# 提取完整基因(完整片段获得的基因全为完整，如成环的细菌基因组)
```

```
grep 'partial=00' temp/prodigal/gene.fa | cut -f1 -d ' '| sed 's/>//' > temp/prodigal/full_length.id
seqkit grep -f temp/prodigal/full_length.id temp/prodigal/gene.fa > temp/prodigal/full_length.fa
seqkit stat temp/prodigal/full_length.fa
```

```
(metagenome) [huodong3@cloud mgs]$ seqkit grep -f temp/prodigal/full_length.id temp/prodigal/gene.fa > temp/prodigal/full_length.fa
[INFO] 33 patterns loaded from file
(metagenome) [huodong3@cloud mgs]$ seqkit stat temp/prodigal/full_length.fa
file                          format  type  num_seqs  sum_len  min_len  avg_len  max_len
temp/prodigal/full_length.fa  FASTA   DNA         33    9,834      114      298      537
```

基因聚类/去冗余cd-hit

下载软件

```
cd /mnt/data1/home/huodong3/Software/
wget https://github.com/weizhongli/cdhit/archive/refs/tags/V4.8.1.tar.gz

#解压文件
tar xvf V4.8.1.tar.gz

#进入文件夹，然后编辑
cd cdhit-4.8.1/
make
```

命令运行

```
# 输入文件：prodigal预测的基因序列 temp/prodigal/gene.fa
# 输出文件：去冗余后的基因和蛋白序列：result/NR/nucleotide.fa
#                                       result/NR/protein.fa
mkdir -p result/NR
# aS覆盖度，c相似度，G局部比对，g最优解，T多线程，M内存0不限制
# 2万基因2m，2千万需要2000h，多线程可加速
time cd-hit-est -i temp/prodigal/gene.fa \
    -o result/NR/nucleotide.fa \
    -aS 0.9 -c 0.95 -G 0 -g 0 -T 0 -M 0
# 统计非冗余基因数量，单次拼接结果数量下降不大，多批拼接冗余度高
grep -c '>' result/NR/nucleotide.fa
# 翻译核酸为对应蛋白序列，emboss
conda install emboss
transeq -sequence result/NR/nucleotide.fa -outseq result/NR/protein.fa -trim Y
# 序列名自动添加了_1，为与核酸对应要去除
sed -i 's/_1 / /' result/NR/protein.fa
```

基因定量salmon

```
mkdir -p temp/salmon

# 建索引，-t序列，-i 索引，10s
time salmon index -t result/NR/nucleotide.fa -p 20 -i temp/salmon/index

```

```
# 定量，l文库类型自动选择，p线程，--meta宏基因组模式，2个任务并行2个样
# 注意parallel中待并行的命令必须是双引号，内部变量需要使用原始绝对路径
time parallel -j 2 "salmon quant -i temp/salmon/index -l A -p 8 --meta -1 temp/qc/{1}_1_kneaddata_paired_1.fastq -2 temp/qc/{1}_1_kneaddata_paired_2.fastq -o temp/salmon/{1}.quant" ::: `tail -n+2 result/metadata.txt|cut -f1`

# 合并
mkdir -p result/salmon
salmon quantmerge --quants temp/salmon/*.quant -o result/salmon/gene.TPM

salmon quantmerge --quants temp/salmon/*.quant --column NumReads -o result/salmon/gene.count

sed -i '1 s/.quant//g' result/salmon/gene.*

# 预览结果表格
head -n3 result/salmon/gene.*
```

功能基因注释

```
# 输入数据：上一步预测的蛋白序列 result/NR/protein.fa
# 中间结果：temp/eggnog/protein.emapper.seed_orthologs
#           temp/eggnog/output.emapper.annotations
#           temp/eggnog/output

# COG定量表：result/eggnog/cogtab.count
#            result/eggnog/cogtab.count.spf （用于STAMP）

# KO定量表：result/eggnog/kotab.count
#           result/eggnog/kotab.count.spf （用于STAMP）

# CAZy碳水化合物注释和定量：result/dbcan2/cazytab.count
#                           result/dbcan2/cazytab.count.spf （用于STAMP）

# 抗生素抗性：result/resfam/resfam.count
#             result/resfam/resfam.count.spf （用于STAMP）

# 这部分可以拓展到其它数据库
```

eggNOG基因注释

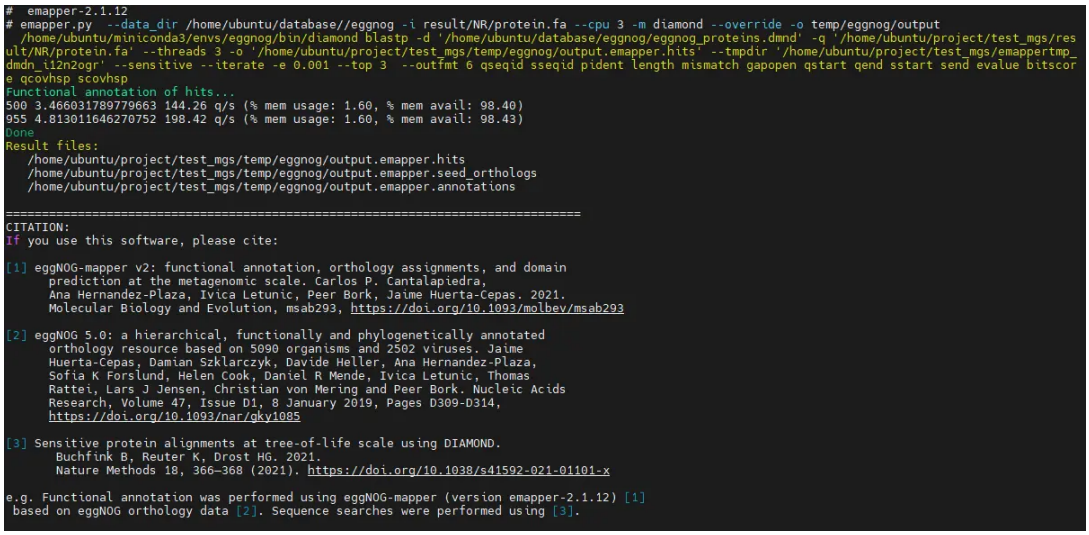
```
db=/mnt/data1/home/huodong3/database/
### eggNOG直接安装
    # 新建环境并进入
    conda create -n eggnog -y
    conda activate eggnog
    # 安装eggnog比对工具emapper
    conda install eggnog-mapper -y -c bioconda -c conda-forge
```

```
### eggNOG安装测试
    emapper.py --version # 2.1.12
    #  Expected eggNOG DB version: 5.0.2 / Installed eggNOG DB version: 5.0.2 /
    #  Diamond version found: diamond version 2.0.15 / MMseqs2 version found: 13.45111
### eggNOG数据库安装

    # 下载常用数据库，注意设置下载位置
    mkdir -p ${db}/eggnog && cd ${db}/eggnog
    # -y默认同意，-f强制下载，eggnog.db.gz 6.3G+4.9G，解压后48G
    download_eggnog_data.py -y -f --data_dir ${db}/eggnog
```

运行软件

```
# 运行emapper，18m，默认diamond 1e-3
    mkdir -p temp/eggnog
    time emapper.py --data_dir ${db}/eggnog \
      -i result/NR/protein.fa --cpu 3 -m diamond --override \
      -o temp/eggnog/output
```




```
# 格式化结果并显示表头
grep -v '^##' temp/eggnog/output.emapper.annotations | sed '1 s/^#//' > temp/eggnog/output
csvtk -t headers -v temp/eggnog/output

# 生成COG/KO/CAZy丰度汇总表
mkdir -p result/eggnog
# 显示帮助
summarizeAbundance.py -h
```

```
Usage: summarizeAbundance.py -i file

Options:
  -h, --help            show this help message and exit
  -i FILEIN, --input-file=FILEIN
                        Sub-item abundance file with format specified above
  -m MAP_FILE, --map-file=MAP_FILE
                        Map file containing group information
  -c GRP_COLUMN, --grpcolumn=GRP_COLUMN
                        The column(s) contains group information. Multiple
                        columns should be supplied as <2,3> represents the
                        second and third column (1-based).
  -s GRP_SEP, --grpsep=GRP_SEP
                        Separtor(s) for each group. Default <,> represents
                        each group would be split by comma. If each group
                        using special separtors, they should be joined by one
                        <+>. <,+*> represents group separators are <,> or any
                        character (*) (each alphabet would be treat as one
                        group)
  -k SUBITEM_KEEP, --subitem-keep=SUBITEM_KEEP
                        <unique> means only keep sub-items which map to only
                        one group. <all> (default) means keep sub items map to
                        all group.
  -e ABUNDANCE_KEEP, --abundance-keep=ABUNDANCE_KEEP
                        This parameter deals with groups containing multiple
                        sub-items. <median>  means using the median abundance
                        of all sub-items map to one group as final group
                        abundance. <min> means using the minimum abundance of
                        all sub-items map to one group as final group
                        abundance. <sum> (default) means using the sum
                        abundance of all sub-items map to one group as final
                        group abundance.
  -n NORM_TYPE, --norm-type=NORM_TYPE
                        Specify the output data type, accept <raw>, <cpm> or
                        both <raw,cpm>.
  -o OUTPUT_PREFIX, --output-prefix=OUTPUT_PREFIX
                        Output file prefix.
  -D, --dropkeycolumn   Show process information
  -v, --verbose         Show process information
  -d, --debug           Debug the program
```

```
# 汇总，7列COG_category按字母分隔，12列KEGG_ko和19列CAZy按逗号分隔，原始值累加
summarizeAbundance.py -i result/salmon/gene.TPM -m temp/eggnog/output --dr
opkeycolumn -c '7,12,13,19' -s '*+,+,' -n raw -o result/eggnog/eggnog
sed -i 's#^ko:##' result/eggnog/eggnog.KEGG_ko.raw.txt
sed -i '/^-/d' result/eggnog/eggnog*
head -n3 result/eggnog/eggnog*
# eggnog.CAZy.raw.txt  eggnog.COG_category.raw.txt  eggnog.KEGG_ko.raw.txt
```

KO--KEGG(整理的KO表如下)

KO.txt

	KEGG_ko	C1	C2
1	K00003	1741.81	0
2	K00052	430.182	0
3	K00067	1954.33	0
4	K00075	907.774	0
5	K00088	1470.24	0
6	K00121	0	2347.57
7	K00134	1426.23	0
8	K00265	0	2128.56
9	K00349	1175.01	0
10	K00383	0	7068.68
11	K00384	1483.169	0
12	K00385	722.646	0

```
mkdir ./result/kegg
summarizeAbundance.py -i result/eggnog/KO.txt -m /home/ubuntu/project/test
_mgs/EasyMicrobiome/kegg/KO1-4.txt -c 2,3,4 -s ',+,+,' -n raw -o result/ke
gg/KEGG
```

参数：–o 文件前缀