

# 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

```
1 # 公共数据库database(db)位置, 如管理员设置/db, 个人下载至~/db
2 # Conda软件安装目录, `conda env list`查看, 如/anaconda3
3 soft=~/.miniconda3
4 db=/home/ubuntu/database/
5 # 设置工作目录work directory(wd), 如meta
6 wd=/home/ubuntu/project/test_mgs/
7 # 创建并进入工作目录
8 mkdir -p $wd && cd $wd
9 # 创建3个常用子目录: 序列, 临时文件和结果
10 mkdir -p seq temp result
11 #加载环境
12 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/

/mnt/data1/home/luodong3/project/smoke/data/mgs/result/					
Name	Size (KB)	Last modified	Owner	Group	Access
..					
metadata.txt	1	2024-02-23...	luodong3	luodong3	-rw-rw-r--

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/

/mnt/data1/home/luodong3/project/smoke/data/mgs/seq/					
Name	Size (KB)	Last modified	Owner	Group	Access
..					
C1_1.fq.gz	6 251	2024-02-23...	luodong3	luodong3	-rw-rw-r--
C1_2.fq.gz	6 243	2024-02-23...	luodong3	luodong3	-rw-rw-r--
C2_2.fq.gz	6 112	2024-02-23...	luodong3	luodong3	-rw-rw-r--
C2_1.fq.gz	5 410	2024-02-23...	luodong3	luodong3	-rw-rw-r--

查看序列大小

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

## kneaddata数据质控

```
1 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

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

## metaphlan物种组成

```
1 ##数据合并
2 time parallel -j 2 --xapply "cat temp/qc/{1}_1_kneaddata_paired_1.fastq te
   mp/qc/{1}_1_kneaddata_paired_2.fastq > temp/cat/{1}_1.fastq" ::: `tail -n+
   2 result/metadata.txt|cut -f1`
3 ##metaphlan
4 time parallel -j 2 --xapply "metaphlan temp/cat/{1}_1.fastq --input_type f
   astq -o result/metaphlan4/{1}_1.tsv --nproc 20 --no_map" ::: `tail -n+2 re
   sult/metadata.txt|cut -f1`
5 ##合并
6 cd ./result/metaphlan4/
7 merge_metaphlan_tables.py *.tsv > merged_abundance_table.txt
8 ##按照界门纲目科属种来进行分类
9 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_s
   pecies.txt
10 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_g
   enus.txt
11 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_f
   amily.txt
12 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_o
   rder.txt
13 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_c
   lass.txt
14 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_p
   hylum.txt
15
```

C1_1.tsv	12	2024-04-09...
merged_abundance_table.txt	10	2024-04-09...
C2_1.tsv	3	2024-04-09...
merged_abundance_table_sp...	1	2024-04-09...
merged_abundance_table_ge...	1	2024-04-09...
merged_abundance_table_fa...	1	2024-04-09...
merged_abundance_table_or...	1	2024-04-09...
merged_abundance_table_cl...	1	2024-04-09...
merged_abundance_table_ph...	1	2024-04-09...

## MEGAHIT拼接

```
1 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
```

```
(metagenome) [huodong3@cloud mgs]$ 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 ^C
(metagenome) [huodong3@cloud mgs]$ tail -n+2 result/metadata.txt|cut -f 1|sed 's/^/temp\qc\\';s/$/_1_kneaddata_paired_1.fastq/'| tr '\n' ','|sed 's/,,$//'\`
temp/qc/C1_1_kneaddata_paired_1.fastq,temp/qc/C2_1_kneaddata_paired_1.fastq(metagenome) [huodong3@cloud mgs]$
```

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

## Prodigal基因预测

```
1 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

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

```
(metagenome) [huodong3@cloud mgs]$ tail temp/prodigal/gene.log
Finding genes in sequence #873 (669 bp)...done!
Finding genes in sequence #874 (387 bp)...done!
Finding genes in sequence #875 (311 bp)...done!
Finding genes in sequence #876 (365 bp)...done!
Finding genes in sequence #877 (308 bp)...done!
Finding genes in sequence #878 (315 bp)...done!
Finding genes in sequence #879 (347 bp)...done!
Finding genes in sequence #880 (342 bp)...done!
Finding genes in sequence #881 (332 bp)...done!
Finding genes in sequence #882 (483 bp)...done!
```

命令运行

```
1 # 统计基因数量
2 grep -c '>' temp/prodigal/gene.fa
3 # 统计完整基因数量，数据量大可只用完整基因部分
4 grep -c 'partial=00' temp/prodigal/gene.fa
5 # 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)
```

```

6 grep 'partial=00' temp/prodigal/gene.fa | cut -f1 -d ' ' | sed 's/> //' > temp/prodigal/full_length.id
7 seqkit grep -f temp/prodigal/full_length.id temp/prodigal/gene.fa > temp/prodigal/full_length.fa
8 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

```

1 cd /mnt/data1/home/huodong3/Software/
2 wget https://github.com/weizhongli/cdhit/archive/refs/tags/V4.8.1.tar.gz
3
4 #解压文件
5 tar xvf V4.8.1.tar.gz
6
7 #进入文件夹，然后编辑
8 cd cdhit-4.8.1/
9 make

```

### 命令运行

```

1 # 输入文件：prodigal预测的基因序列 temp/prodigal/gene.fa
2 # 输出文件：去冗余后的基因和蛋白序列：result/NR/nucleotide.fa
3 #                                     result/NR/protein.fa
4 mkdir -p result/NR
5 # 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
6 # 2万基因2m，2千万需要2000h，多线程可加速
7 time cd-hit-est -i temp/prodigal/gene.fa \
8     -o result/NR/nucleotide.fa \
9     -aS 0.9 -c 0.95 -G 0 -g 0 -T 0 -M 0
10 # 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
11 grep -c '>' result/NR/nucleotide.fa
12 # 翻译核酸为对应蛋白序列，emboss
13 conda install emboss
14 transeq -sequence result/NR/nucleotide.fa -outseq result/NR/protein.fa -tr
    im Y
15 # 序列名自动添加了_1，为与核酸对应要去除
16 sed -i 's/_1 / /' result/NR/protein.fa

```

## 基因定量salmon

```

1 mkdir -p temp/salmon
2
3 # 建索引，-t序列，-i 索引，10s
4 time salmon index -t result/NR/nucleotide.fa -p 20 -i temp/salmon/index
5

```

```

6 # 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
7 # 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
8 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`
9
10 # 合并
11 mkdir -p result/salmon
12 salmon quantmerge --quants temp/salmon/*.quant -o result/salmon/gene.TPM
13
14 salmon quantmerge --quants temp/salmon/*.quant --column NumReads -o result
   /salmon/gene.count
15
16 sed -i '1 s/.quant//g' result/salmon/gene.*
17
18 # 预览结果表格
19 head -n3 result/salmon/gene.*

```

## 功能基因注释

```

1 # 输入数据：上一步预测的蛋白序列 result/NR/protein.fa
2 # 中间结果：temp/eggnoG/protein.emapper.seed_orthologs
3 #           temp/eggnoG/output.emapper.annotations
4 #           temp/eggnoG/output
5
6 # COG定量表：result/eggnoG/cogtab.count
7 #           result/eggnoG/cogtab.count.spf（用于STAMP）
8
9 # KO定量表：result/eggnoG/kotab.count
10 #          result/eggnoG/kotab.count.spf（用于STAMP）
11
12 # CAZy碳水化合物注释和定量：result/dbcan2/cazytab.count
13 #                             result/dbcan2/cazytab.count.spf（用于STAMP）
14
15 # 抗生素抗性：result/resfam/resfam.count
16 #             result/resfam/resfam.count.spf（用于STAMP）
17
18 # 这部分可以拓展到其它数据库

```

## eggNOG基因注释

```

1 db=/mnt/data1/home/huodong3/database/
2 ### eggNOG直接安装
3   # 新建环境并进入
4   conda create -n eggnog -y
5   conda activate eggnog
6   # 安装eggnoG比对工具emapper
7   conda install eggnog-mapper -y -c bioconda -c conda-forge

```

```

8  ### eggNOG安装测试
9      emapper.py --version # 2.1.12
10     # Expected eggNOG DB version: 5.0.2 / Installed eggNOG DB version: 5.
11     0.2 /
12     # Diamond version found: diamond version 2.0.15 / MMseqs2 version fou
13     nd: 13.45111
14     ### eggNOG数据库安装
15     # 下载常用数据库, 注意设置下载位置
16     mkdir -p ${db}/eggnog && cd ${db}/eggnog
17     # -y默认同意, -f强制下载, eggnog.db.gz 6.3G+4.9G, 解压后48G
18     download_eggnog_data.py -y -f --data_dir ${db}/eggnog

```

## 运行软件

```

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

```

```

# emapper-2.1.12
# emapper.py --data_dir /home/ubuntu/database/eggnog -i result/NR/protein.fa --cpu 3 -m diamond --override -o temp/eggnog/output
/home/ubuntu/miniconda3/envs/eggnog/bin/diamond blastp -d /home/ubuntu/database/eggnog/eggnog_proteins.dmnd -q /home/ubuntu/project/test_mgs/res
ult/NR/protein.fa --threads 3 -o /home/ubuntu/project/test_mgs/temp/eggnog/output.emapper.hits --tmpdir /home/ubuntu/project/test_mgs/emappertmp_
dmnd_l1zn2ogr --sensitive --iterate -e 0.001 --top 3 --outfmt 6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send eval bitscor
e gcovhsp scovhsp
Functional annotation of hits...
500 3.466031789779663 144.26 q/s (% mem usage: 1.60, % mem avail: 98.40)
955 4.813011646270752 198.42 q/s (% mem usage: 1.60, % mem avail: 98.43)
Done
Result files:
/home/ubuntu/project/test_mgs/temp/eggnog/output.emapper.hits
/home/ubuntu/project/test_mgs/temp/eggnog/output.emapper.seed orthologs
/home/ubuntu/project/test_mgs/temp/eggnog/output.emapper.annotations

=====
CITATION:
If you use this software, please cite:

[1] eggNOG-mapper v2: functional annotation, orthology assignments, and domain
prediction at the metagenomic scale. Carlos P. Cantalapiedra,
Ana Hernandez-Plaza, Ivica Letunic, Peer Bork, Jaime Huerta-Cepas. 2021.
Molecular Biology and Evolution, msab293, https://doi.org/10.1093/molbev/msab293

[2] eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated
orthology resource based on 5090 organisms and 2502 viruses. Jaime
Huerta-Cepas, Damian Szklarczyk, Davide Heller, Ana Hernandez-Plaza,
Sofia K Forslund, Helen Cook, Daniel R Mende, Ivica Letunic, Thomas
Rattet, Lars J Jensen, Christian von Mering and Peer Bork. Nucleic Acids
Research, Volume 47, Issue D1, 8 January 2019, Pages D309-D314,
https://doi.org/10.1093/nar/gky1085

[3] Sensitive protein alignments at tree-of-life scale using DIAMOND.
Buchfink B, Reuter K, Drost HG. 2021.
Nature Methods 18, 366-368 (2021). https://doi.org/10.1038/s41592-021-01101-x

e.g. Functional annotation was performed using eggNOG-mapper (version emapper-2.1.12) [1]
based on eggNOG orthology data [2]. Sequence searches were performed using [3].

```

```

1  # 格式化结果并显示表头
2  grep -v '^##' temp/eggnog/output.emapper.annotations | sed '1 s/^#//' > te
3  mp/eggnog/output
4
5  csvtk -t headers -v temp/eggnog/output
6
7  # 生成COG/KO/CAZy丰度汇总表
8  mkdir -p result/eggnog
9
10 # 显示帮助
11 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
                        Separator(s) for each group. Default <,> represents
                        each group would be split by comma. If each group
                        using special separators, 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
```

```
1 # 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
2 summarizeAbundance.py -i result/salmon/gene.TPM -m temp/egglog/output --dr
  opkeycolumn -c '7,12,13, 19' -s '+,+, ' -n raw -o result/egglog/egglog
3 sed -i 's#^ko:##' result/egglog/egglog.KEGG_ko.raw.txt
4 sed -i '/^-/d' result/egglog/egglog*
5 head -n3 result/egglog/egglog*
6 # egglog.CAZy.raw.txt  egglog.COG_category.raw.txt  egglog.KEGG_ko.raw.txt
```

KO--KEGG(整理的KO表如下)

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

```
1 mkdir ./result/kegg
2 summarizeAbundance.py -i result/egglog/KO.txt -m /home/ubuntu/project/test
  _mgs/EasyMicrobiome/kegg/K01-4.txt -c 2,3,4 -s '+,+, ' -n raw -o result/ke
  gg/KEGG
```

参数: -o 文件前缀