## Phylogenetic analysis

GTDB-tk (0.3.2)

Create the GTDB-Tk environment and install

conda create -n gtdbtk -c conda-forge -c bioconda gtdbtk

download database:

download-db.sh

Verifying your installation:

gtdbtk check\_install

output be like:

[2020-11-04 09:35:16] INFO: GTDB-Tk v1.4.0

[2020-11-04 09:35:16] INFO: gtdbtk check\_install

[2020-11-04 09:35:16] INFO: Using GTDB-Tk reference data version r95: /release95

[2020-11-04 09:35:16] INFO: Running install verification

[2020-11-04 09:35:16] INFO: Checking that all third-party software are on the system path:

[2020-11-04 09:35:16] INFO: |-- FastTree OK

[2020-11-04 09:35:16] INFO: |-- FastTreeMP OK

[2020-11-04 09:35:16] INFO: |-- fastANI OK

[2020-11-04 09:35:16] INFO: |-- guppy OK

[2020-11-04 09:35:16] INFO: |-- hmmalign OK

[2020-11-04 09:35:16] INFO: |-- hmmsearch OK

[2020-11-04 09:35:16] INFO: |-- mash OK

[2020-11-04 09:35:16] INFO: |-- pplacer OK

[2020-11-04 09:35:16] INFO: |-- prodigal OK

[2020-11-04 09:35:16] INFO: Checking /release95

[2020-11-04 09:35:16] INFO: |-- pplacer OK

[2020-11-04 09:35:16] INFO: |-- masks OK

[2020-11-04 09:35:17] INFO: |-- markers OK

[2020-11-04 09:35:17] INFO: |-- radii OK

[2020-11-04 09:35:20] INFO: |-- msa OK

[2020-11-04 09:35:20] INFO: |-- metadata OK

[2020-11-04 09:35:20] INFO: |-- taxonomy OK

[2020-11-04 09:47:36] INFO: |-- fastani OK

[2020-11-04 09:47:36] INFO: |-- mrca\_red OK

[2020-11-04 09:47:36] INFO: Done.

check test file:

gtdbtk test --out\_dir /tmp/test --cpus 8

output be like:

[2020-04-13 09:50:58] INFO: GTDB-Tk v1.1.0

[2020-04-13 09:50:58] INFO: gtdbtk test --out\_dir /tmp/test --cpus 3

[2020-04-13 09:50:58] INFO: Using GTDB-Tk reference data version r89: /release89

[2020-04-13 09:50:58] INFO: Command: gtdbtk classify\_wf --genome\_dir /tmp/test/genomes --out\_dir /tmp/test/output --cpus 3

[2020-04-13 09:52:35] INFO: Test has successfully finished.

Output files: [prefix].warnings.log, output/, test\_execution.log

What’s in output/: gtdbtk.ar122.classify.tree, gtdbtk.ar122.summary.tsv, gtdbtk.ar122.markers\_summary.tsv, gtdbtk.ar122.msa.fasta, gtdbtk.ar122.user\_msa.fasta

Species annotation:

gtdbtk classify\_wf --genome\_dir <my\_genomes> --out\_dir <output\_dir> --cpus 64 -x fna --force

Upload \*.unrooted.tree to itol (https://itol.embl.de/) for visualization.

## RNA-seq

Install required software

It is strongly recommended to create a new conda environment for metatranscriptome before installation. The software required, the version used and installation commands are listed:

fastp, version 0.20.1

Bowtie2, version 2.4.4

samtools, version 1.14

featureCounts, version 2.0.1

conda create -n RNA-seq

conda activate RNA-seq

conda install -c bioconda fastp

conda install -c bioconda bowtie2

conda install -c bioconda samtools

# enter the direction to install featureCounts

wget https://jaist.dl.sourceforge.net/project/subread/subread-2.0.1/subread-2.0.1-Linux-x86\_64.tar.gz

tar -zxvf subread-2.0.1-Linux-x86\_64.tar.gz

count reads

Raw data was deposited in NCBI under the accession number of PRJNA799876. Four triplicate samples were collected during SMX degradation at 0, 4, 8 and 12 h.

Quality control:

fastp -i in.R1.fq.gz -I in.R2.fq.gz -o out.R1.fq.gz -O out.R2.fq.gz

Build sam database

bowtie2-build ASSEMBLY/megahit/final.contigs.fa smx

bowtie2 -x smx -1 example\_1.1.fastq -2 example\_1.2.fastq -S example.sam -p 32

sam to bam:

samtools view -bS smx.sam > smx.bam

samtools sort -n smx.bam -o smx.sorted.bam

samtools view -h smx.sorted.bam|less -N

Count mapped reads using the sorted bam file

featureCounts -p -t gene -g gene\_id -a total.gtf -o counts.txt smx.sorted.bam

Sort 12 counts files into 1 file manually, see detailed in files/total\_counts.txt.

Select degradation genes shown in Figure 5 and calculate their average counts, see detailed in files/degradation\_genes\_expression.txt.

Differential expression genes (based on R)

Install and library packages needed:

install.packages('BiocManager')

BiocManager::install('DESeq2')

install.packages('tidyverse')

library(tidyverse)

library(DESeq2)

read counts file (after set working directory where you put the counts file):

mycounts <- read.table(file="total\_counts.txt", sep = "\t", quote = "", header = T, row.names=1)

condition <- factor(c(rep("control",3),rep("treat1",3),rep("treat2",3),rep("treat3",3)), levels = c("control","treat1","treat2","treat3"))

colData <- data.frame(name = c("h0\_1","h0\_2","h0\_3","h4\_1","h4\_2","h4\_3","h8\_1","h8\_2","h8\_3","h12\_1","h12\_2","h12\_3"), condition)

dds <- DESeqDataSetFromMatrix(mycounts, colData=colData, design= ~ condition)

dds <- dds[ rowSums(counts(dds)) > 0, ] #filter genes that all counts are zero

dds <- DESeq(dds)

res <- results(dds, contrast=c("condition","treat1","control")，alpha=0.05) #result of treat1 and control comparison (h4 vs h0)

write.table(res,"h4vs0.txt",sep = '\t', col.names = T, row.names = T, quote = FALSE, na='') # save the result

res <- results(dds, contrast=c("condition","treat2","control")，alpha=0.05) #result of treat2 and control comparison (h8 vs h0)

write.table(res,"h8vs0.txt",sep = '\t', col.names = T, row.names = T, quote = FALSE, na='') # save the result

res <- results(dds, contrast=c("condition","treat3","control")，alpha=0.05) #result of treat3 and control comparison (h12 vs h0)

write.table(res,"h12vs0.txt",sep = '\t', col.names = T, row.names = T, quote = FALSE, na='') # save the result

resultsNames(dds)

[1] "Intercept" "bacth\_2\_vs\_I"

[3] "Group\_MF\_vs\_WTF" "Group\_MMF\_vs\_WTF"

[5] "Group\_WTM\_vs\_WTF"

res <- results(dds, name="Group\_MMF\_vs\_WTF")

res <- results(dds, contrast=c("Group"," MMF "," WTF ")) #后面的是对照

Heatmap (based on R)

# Install and library the package:

install.packages("pheatmap")

library(pheatmap)

# read data and transform to log2:

data <- read.table ("degradation\_genes\_expression.txt", header = T, sep ="\t", row.names = 1)

datalg = log2(data+1)

# draw heatmap:

pheatmap (datalg, cluster\_cols = F, legend = T, angle\_col = 0, clustering\_method = "complete", annotation\_legend = F, show\_rownames = T, cellwidth = 25, cellheight = 25, border\_color = "black")

## Plasmid

Plasflow (1.1.0)

Install:

conda create -n plasflow

conda install -c bioconda plasflow

filter contigs less than 1000 bp:

filter\_sequences\_by\_length.pl -input SDB1.fa -output SDB1filter.fasta -thresh 1000

run plasflow:

PlasFlow.py --input SDB1filter.fasta --output SDB1.tsv --threshold 0.7

The results contain four files: SDB1.tsv, SDB1.tsv\_chromosomes.fasta, SDB1.tsv\_plasmids.fasta, SDB1.tsv\_unclassified.fasta

Plascad

Install:

conda create -n plascad

conda install -c pianpianyouche plascad

take fasta file of plasmid sequences as input:

Plascad -i SDB1.tsv\_plasmids.fasta

Output

\*\_classification\_sum.txt: name, plasmid type, ARGs (type\_subtype)

\*\_conj\_plasimids\_loc\_sum.txt: name, marker genes, c-value & e-value (c-value for hmmsearch, e-value for blastp), genetic location, strand

A plasmid visualization component using AngularJS is integrated into our pipeline, all the plasmid maps are in HTML formats. In order to view the map locally, you need to download the js folder in addtion to the HTML files.

SPAdes

Require a 64-bit Linux system and Python (supported versions are Python 2.7, and Python3: 3.2 and higher) on it. Go to the directory in which you wish SPAdes to be installed and run:

wget http://cab.spbu.ru/files/release3.15.5/SPAdes-3.15.5-Linux.tar.gz

tar -xzf SPAdes-3.15.5-Linux.tar.gz

cd SPAdes-3.15.5-Linux/bin/

In case of successful installation the following files will be placed in the bin directory:

spades.py (main executable script)

metaspades.py (main executable script for metaSPAdes)

**plasmidspades.py (main executable script for plasmidSPAdes)**

metaplasmidspades.py (main executable script for metaplasmidSPAdes)

metaviralspades.py (main executable script for metaviralSPAdes)

rnaspades.py (main executable script for rnaSPAdes)

truspades.py (main executable script for truSPAdes, DEPRECATED)

rnaviralspades.py (main executable script for rnaviralSPAdes)

coronaspades.py (wrapper script for coronaSPAdes mode)

spades-core (assembly module)

spades-gbuilder (standalone graph builder application)

spades-gmapper (standalone long read to graph aligner)

spades-kmercount (standalone k-mer counting application)

spades-hammer (read error correcting module for Illumina reads)

spades-ionhammer (read error correcting module for IonTorrent reads)

spades-bwa (BWA alignment module which is required for mismatch correction)

spades-corrector-core (mismatch correction module)

spades-truseq-scfcorrection (executable used in truSPAdes pipeline)

adding SPAdes installation directory to the PATH variable.

export PATH=/data/home/a204a/SPAdes/SPAdes-3.15.5-Linux/bin:$PATH

Verifying your installation

spades.py --test

If the installation is successful, you will find the following information at the end of the log:

===== Assembling finished. Used k-mer sizes: 21, 33, 55

\* Corrected reads are in spades\_test/corrected/

\* Assembled contigs are in spades\_test/contigs.fasta

\* Assembled scaffolds are in spades\_test/scaffolds.fasta

\* Assembly graph is in spades\_test/assembly\_graph.fastg

\* Assembly graph in GFA format is in spades\_test/assembly\_graph\_with\_scaffolds.gfa

\* Paths in the assembly graph corresponding to the contigs are in spades\_test/contigs.paths

\* Paths in the assembly graph corresponding to the scaffolds are in spades\_test/scaffolds.paths

======= SPAdes pipeline finished.

========= TEST PASSED CORRECTLY.

SPAdes log can be found here: spades\_test/spades.log

Thank you for using SPAdes!

spades.py --metaplasmid -o /data/home/a204a/SPAdes/SPAdes-3.15.5-Linux/bin/smx\_spades -1 /data/home/a204a/BINNING-longlongago/CLEAN\_READS/ALL\_READS\_1.fastq -2 /data/home/a204a/BINNING-longlongago/CLEAN\_READS/ALL\_READS\_2.fastq --threads 32

# --threads default 16

Contigs and scaffolds format (scaffolds.fasta is recomended)

Contigs/scaffolds names in SPAdes output FASTA files have the following format: >NODE\_3\_length\_237403\_cov\_243.207 Here 3 is the number of the contig/scaffold, 237403 is the sequence length in nucleotides and 243.207 is the k-mer coverage for the last (largest) k value used. Note that the k-mer coverage is always lower than the read (per-base) coverage.

In order to distinguish contigs with putative plasmids detected at different cutoff levels we extend the contig name in FASTA file with cutoff value used for this particular contig (in format \_cutoff\_N). For metaplasmid mode we output **only circular putative plasmids**.

Plasmidfinder

conda create -n plasmidfinder

conda activate plasmidfinder

conda install -c bioconda plasmidfinder

cd anaconda3/envs/plasmidfinder/bin

download-db.sh

plasmidfinder.py -h

cd /data/home/a204a/BINNING-longlongago/ASSEMBLY/megahit

mkdir plasmidfinder

plasmidfinder.py -i final.contigs.fa -o plasmidfinder -tmp plasmidfinder -mp blastn -t 32 -p /home/a204a/anaconda3/envs/plasmidfinder/share/plasmidfinder-2.1.6/database/

Error: BLAST did not run as expected.

abricate

contain plasmidfinder database

conda create -n abricate

conda activate abricate

conda install -c bioconda abricate

stopped here

abricate --check

abricate xxx.fasta -db plasmidfinder

abricate --list # view all the databases

## Coverage

# 教程https://blog.csdn.net/Emmett\_Bioinfo/article/details/115705002

conda activate RNA-seq

cd /data/home/DXQ/SMX\_WQ

# 组装后没有分箱的所有contig

# 建库

bowtie2-build --thread 40 final.contigs.fa all\_contig\_build

# 生成sam文件，需要fastq，不能跑多个线程

bowtie2 -x all\_contig\_build -1 ALL\_READS\_1.fastq -2 ALL\_READS\_2.fastq -S all\_contig.sam --threads 1

# sam to bam

samtools view -bS all\_contig.sam > all\_contig.bam

# bam文件排序，生成bai文件

samtools sort all\_contig.bam -o all\_contig\_sorted.bam

samtools index all\_contig\_sorted.bam

# 将fasta, sorted.bam, sorted.bam.bai文件放入一个文件夹中checkM\_all

# 计算每个contig在宏基因中的覆盖率

checkm coverage checkM\_all/ all\_coverage.out checkM\_all/all\_contig\_sorted.bam -x fasta -m 20

# 将fasta文件换为bin文件，可计算bin中的contig在宏基因中的覆盖率，如

checkm coverage checkM\_all/ SDB6\_meta\_coverage.out checkM\_all/all\_contig\_sorted.bam -x fasta -m 20