**CCPRD: A novel analytical framework for comprehensive proteomic reference database construction of** **non-model organisms**

\* Contributors

\* What is CCPRD?

\* Dependencies

\* Pipelines

\* Examples

\* Citing CCPRD and software called by CCPRD

\* License

**Contributors**

Qingxiang Guo1, 2, Dan Li1, 2, Yanhua Zhai1, 2, Zemao Gu1, 2\*

1Department of Aquatic Animal Medicine, College of Fisheries, Huazhong Agricultural University, Wuhan, Hubei province, 430070, PR China.

2Hubei Engineering Technology Research Center for Aquatic Animal Diseases Control and Prevention, Wuhan, 430070, PR China.

**What is CCPRD?**

Here we propose an efficient framework for constructing the comprehensive protein reference database, "customized comprehensive proteomic reference database (CCPRD)", which incorporated the draft genomes and transcriptomes. Compared with previous protocols, our method has superiorities in peptide and protein identification numbers, number of entries database size, and completeness.

**Dependencies**

Perl version >= 5.10, Python version = 2.7, acc2tax version 0.4, Anvi’o version 2.0.2, Augustus version 3.2.2, BLAST version 2.2.25, Blobtools version 0.9.19, Bowtie2 version 2.2.9, CD-HIT version 4.6.6, Diamond version 0.8.31.93, EMBOSS version 6.6.0, Ete 3, EVidenceModeler version 1.1.1, GeneMarkS-T version 5.1, GeneMark-ES-ET version 2.5m, PASA version 2.2.0, RepeatMasker version 4.0.5, RepeatModeler version 1.0.4, Samtools version 1.3.1, SNAP version 2006-07-28, SOAPdenovo version 2.04, TGICL version 2.1, Tophat version 2.1.1, TransDecoder version 3.0.0, Trinity version 2.2.0, Wise version 2.4.1

**Pipelines**

1. **Assembly the transcriptomes and genomes**

**1.1** **Assembly the transcriptomes from clean data**

Trinity --seqType fq --left M.wulii\_1\_fixed.fastq --right M.wulii\_2\_fixed.fastq --max\_memory 200G

--min\_contig\_length 200 --min\_glue 3 --group\_pairs\_distance 250 --path\_reinforcement\_distance 85 --min\_kmer\_cov 2 --jaccard\_clip --normalize\_reads --CPU 32

mv Trinity.fasta wulii.fasta

**1.2 Remove redundancy in transcriptomes with CD-HIT**

cd-hit-est -i wulii.fasta -o wulii.fasta\_cdhit -c 0.95 -n 10 -d 0 -M 16000 -T 8 &

**1.3 Further cluster the transcriptomes with TGICL**

tgicl -l 40 -c 10 -v 25 -O '-repeat\_stringency 0.95 -minmatch 35 -minscore 35' -F wulii.fasta\_cdhit

**1.4** **Process the cd-hit results to get unigenes**

fast\_extract\_seq\_from\_fasta.pl wulii.fasta\_cdhit wulii.fasta\_cdhit.singletons > extracted.fasta

cat asm\_1/contigs asm\_2/contigs asm\_3/contigs asm\_4/contigs > all.contigs

Unigene\_generator.pl -s extracted.fasta -c all.contigs -t WL

# This will output WL\_Unigene.fasta for downstream analyses.

**1.5 Assembly the genomes from clean data**

SOAPdenovo all -s lib.cfg -K 51 -D 1 -o WL >> soap.log

mv WL.scafSeq genome.fasta

Gapcloser -a genome.fasta -b config.txt -o gapcloser.fasta -t 32

mv gapcloser.fasta WL\_genome.fasta

1. **Remove potential host and bacterial contamination in transcriptome data with "conservative reciprocal best blast hit" method**
   1. **Collected all available proteins and nucleotide sequences for constructing host or close-related species no-redundant database**

# Host database contained (proteins or transcripts): *Cyprinus carpio* genome predicted proteins (GCF\_000951615.1), *Danio rerio* genome predicted proteins (GCA\_000002035.3), *Carassius* proteins in NCBI (4087), *Carassius auratus* EST (11307), *Carassius auratus* transcriptomes, *Cyprinus carpio* EST (497380); close-related no-redundant database contained (proteins or transcripts): *Anemonia viridis* proteins in NCBI (346)，*Clytia hemisphaerica* proteins in NCBI (529), *Exaiptasia pallida* genome predicted proteins (GCA\_001417965.1), *Hydra vulgaris* genome predicted proteins (Hydra\_RP\_1.0), *Hydractinia echinata* proteins in NCBI (121), *Metridium senile* proteins in NCB (123), *Nematostella vectensis* genome predicted protein (GCA\_000209225.1), *Thelohanellus kitauei* genome predicted proteins (GCA\_000827895.1), *Buddenbrockia plumatellae* EST (765), *Tetracapsuloides bryosalmonae* EST (308), *Polypodium hydriforme* transcriptomes, *Myxobolus pendula* transcriptome, *Myxobolus cerebralis* transcriptomes, *Chironex fleckeri* transcriptomes.

* 1. **Remove redundancy in each database**

cd-hit-est -i nucl\_host.fasta -o nucl\_host\_cdhit.fasta -c 0.9 -n 8 -T 8

cd-hit-est -i nucl\_myxo.fasta -o nucl\_myxo\_cdhit.fasta -c 0.9 -n 8 -T 8

cd-hit -i prot\_host.fasta -o prot\_host\_cdhit.fasta -c 0.9

d-hit -i prot\_myxo.fasta -o prot\_myxo\_cdhit.fasta -c 0.9

* 1. **Add tags in databases**

replace\_header\_for\_cdhit.pl -c nucl\_host\_cdhit.fasta -t HN

replace\_header\_for\_cdhit.pl -c prot\_host\_cdhit.fasta -t HP

replace\_header\_for\_cdhit.pl -c nucl\_myxo\_cdhit.fasta -t MN

replace\_header\_for\_cdhit.pl -c prot\_myxo\_cdhit.fasta -t MP

# Sequence number: HN (45486), HP (59764), MN (103090), MP (78786).

* 1. **Start hybridization**

# For proteins:

tblastn -query HP\_cdhit.fasta -db /home/gqx/transcriptome/assembly/M.wulii/clustering/MWdb -max\_hsps 1 -out result\_HP -evalue 1e-5 -outfmt 6 -num\_threads 8

tblastn -query MP\_cdhit.fasta -db /home/gqx/transcriptome/assembly/M.wulii/clustering/MWdb -max\_hsps 1 -out result\_MP -evalue 1e-5 -outfmt 6 -num\_threads 8

# For nucleotides:

tblastx -query MN\_cdhit.fasta -db /home/gqx/transcriptome/assembly/ M.wulii/clustering/HHdb -max\_hsps 1 -out result\_MN -evalue 1e-5 -outfmt 6 -num\_threads 8

tblastx -query HN\_cdhit.fasta -db /home/gqx/transcriptome/assembly/ M.wulii/clustering/HHdb -max\_hsps 1 -out result\_HN -evalue 1e-5 -outfmt 6 -num\_threads 8

* 1. **Process above results and remove the transcripts that only matched to host databases**

# Extract non-redundant subjects for each comparison result.

cat result\_MP | cut -f 2 > 1 | remove\_duplicate.pl 1 | mv duplicate\_remove MP

cat result\_HP | cut -f 2 > 1 | remove\_duplicate.pl 1 | mv duplicate\_remove HP

cat result\_MN | cut -f 2 > 1 | remove\_duplicate.pl 1 | mv duplicate\_remove MN

cat result\_HN | cut -f 2 > 1 | remove\_duplicate.pl 1 | mv duplicate\_remove HN

# Integrating and compare non-redundant lists.

mkdir combo | cd combo/

cp ../1\_host/nucl\_host/hybrid\_HN/HN ./

cp ../1\_host/prot\_host/hybrid\_HP/HP ./

cp ../2\_myxo/nucl\_myxo/hybrid\_MN/MN ./

cp ../2\_myxo/prot\_myxo/hybrid\_MP/MP ./

cat HN HP > H\_all

cat MN MP > M\_all

remove\_duplicate.pl H\_all | mv duplicate\_remove H

remove\_duplicate.pl M\_all | mv duplicate\_remove M

list\_compare.pl M H

# Process the comparison results.

mkdir 1\_only\_to\_host 2\_only\_to\_myxo 3\_both\_match 4\_neither\_match 5\_delete\_host

cd 1\_only\_to\_host | cp ../../3\_combo/H\_only ./Host\_only\_list

cd ../2\_only\_to\_myxo/ | cp ../../3\_combo/M\_only ./Myxo\_only\_list

cd ../3\_both\_match/ | cp ../../3\_combo/inter\_of\_M\_and\_H ./both\_match\_list

cd ../4\_neither\_match | cp ../../3\_combo/union\_of\_M\_and\_H ./

ln -s ~/transcriptome/3\_1\_assembly/M.wulii/WL\_Unigene.fasta ./

remove\_contaminant\_by\_ID.pl WL\_Unigene.fasta union\_of\_M\_and\_H

mv survive.fasta M.wulii\_neither\_match.fasta

extract\_fasta\_header.pl M.wulii\_neither\_match.fasta

mv header neither\_match\_list

cd 5\_delete\_host

remove\_contaminant\_by\_ID.pl /home/gqx/transcriptome/assembly/WL\_Unigene.fasta ../1\_only\_to\_host/Host\_only\_list

mv survive.fasta WL\_host\_delete\_Unigene.fasta

* 1. **Collected proteins for constructing** **bacterial no-redundant database**

# Bacterial no-redundant database contains: Genome predicted proteins from, *Aeromonas caviae*, *Aeromonas hydrophila*, *Aeromonas sobria*, *Aeromonas veronii B565*, *Citrobacter freundii CFNIH1*, *Escherichia coli*, *Flavobacterium columnare*, *Klebsiella pneumoniae*, *Lactobacillus sakei*, *Pseudomonas aeruginosa*, *Salmonella enterica*, *Shewanella putrefaciens*, *Plesiomonas shigelloides*.

* 1. **Remove redundancy in bacterial database and blast the host-seq-removed transcriptomes**

cd-hit -i bac\_protein.fasta -o bac\_protein\_cdhit.fasta -c 0.95 -n 5 -T 8

# 45530 proteins retained in bacterial databases after redundancy removal.

makeblastdb -in bac\_protein\_cdhit.fasta -out BAC -dbtype prot -parse\_seqids -hash\_index

blastx -query WL\_host\_delete\_Unigene.fasta -db BAC -out result\_bac\_e10 -evalue 1e-10 -outfmt 6 -num\_threads 32

* 1. **First round removement of bacterial and confirmation**

cat result\_bac\_e10 | cut -f1 > 1

remove\_duplicate.pl 1

mv duplicate\_remove bacteria\_contam\_list

extract\_seq\_from\_fasta.pl WL\_host\_delete\_Unigene.fasta bacteria\_contam\_list

mv extracted.fasta bacteria\_contam\_first.fasta

# Confirm those sequences by blasting with nr database.

blastx -query bacteria\_contam\_first.fasta -db nr -out bacteria\_contam\_first\_nr\_result -evalue 1e-5 -max\_target\_seqs 1 -num\_threads 56 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms"

grep "Bacteria" bacteria\_contam\_first\_nr\_result | cut -f 1 > 2

remove\_duplicate.pl 2 | rm 2

mv duplicate\_remove true\_bacteria\_contam.list

remove\_contaminant\_by\_ID.pl WL\_host\_delete\_Unigene.fasta true\_bacteria\_contam.list

# This step will produce the final clean transcriptome with host and bacterial contamination conservatively removed (WL\_all\_filter\_Unigene.fasta), however, if the proportion of true bacterial contamination in bacteria\_contam\_list is too large, a second-round decontamination is recommended.

* 1. **Second round removement of bacterial contamination**

# Blast transcriptomes in last step with uniref90 (with diamond) and nt database (with megablast).

diamond makedb --in uniref90.fasta --db uniref90

diamond blastx -q WL\_host\_delete\_Unigene.fasta --sensitive -k 20 -c 1 --threads 32 --db uniref90 --out diamond\_result

# Add taxonomy information.

perl -lne '

BEGIN{open UT, "<uniref90.taxlist" or die $!; while (<UT>) { $ut{$1}=$2 if /^(\S+)\t(\S+)$/ } }

{print "$\_\t$ut{$1}" if /^\S+\t(\S+)/ and exists $ut{$1}}' \

< assembly\_diamond\_10.out \

> assembly\_diamond\_10.out\_taxid

blastn -task megablast -query WL\_host\_delete\_Unigene.fasta -db nt -culling\_limit 5 -outfmt '6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle staxids sscinames sskingdoms' -num\_threads 48 -evalue 1e-25 -out assembly\_megablast\_25.out

# Add taxonomy information.

perl -lne '

BEGIN{open UT, "<acc2tax\_nucl\_all.txt" or die $!; while (<UT>) { $ut{$2}=$3 if /^(\S+)\t(\S+)\t(\S+)/ } }

{print "$\_\t$ut{$1}" if /^\S+\t(\S+)/ and exists $ut{$1}}' \

< assembly\_megablast\_25.out \

> assembly\_megablast\_25.out\_taxid

# Get the species distribution in transcriptomes.

cat ../diamond/assembly\_diamond\_10.out\_taxid ../megablast/assembly\_megablast\_25.out\_taxid > all

export LANG=C; export LC\_ALL=C; sort -k1,1 -k12,12gr -k11,11g -k3,3gr all | sort -u -k1,1 --merge > bestHits

cat bestHits | rev | cut -f 1 | rev > 2

# Transfer taxid to species name by ete3.

tax2name.py > result\_1

get\_species\_name\_from\_ete3.pl

species\_distribution.pl names

# Redownload host and bacterial sequences according to the species distribution results.

cat ./\* > fish\_protein.fasta

cd-hit -i fish\_protein.fasta -o fish\_protein\_cdhit.fasta -c 0.95 -n 5 -T 8

makeblastdb -in fish\_protein\_cdhit.fasta -out FISH -dbtype prot -parse\_seqids -hash\_index

cat ./\* > bac\_protein.fasta

cd-hit -i bac\_protein.fasta -o bac\_protein\_cdhit.fasta -c 0.95 -n 5 -T 8

makeblastdb -in bac\_protein\_cdhit.fasta -out BAC -dbtype prot -parse\_seqids -hash\_index

# Reblast with the new databases.

blastx -query WL\_all\_filter\_Unigene.fasta -db BAC -out result\_bac\_1 -evalue 1 -outfmt 6 -num\_threads 48

blastx -query WL\_all\_filter\_Unigene.fasta -db FISH -out result\_fish\_1 -evalue 1 -outfmt 6 -num\_threads 48

# Extract the potiential contaminations.

cat result\_bac\_1 | cut -f1 > 1

remove\_duplicate.pl 1

mv duplicate\_remove bacteria\_contam\_list

cat result\_fish\_1 | cut -f1 > 1

remove\_duplicate.pl 1

mv duplicate\_remove fish\_contam\_list

extract\_seq\_from\_fasta.pl WL\_all\_filter\_Unigene.fasta bacteria\_contam\_list

mv extracted.fasta bacteria\_contam\_first.fasta

extract\_seq\_from\_fasta.pl WL\_all\_filter\_Unigene.fasta fish\_contam\_list

mv extracted.fasta fish\_contam\_first.fasta

# Blast extracted potiential sequences with nr database.

diamond makedb --in nr -d nr -p 24

diamond blastx -q fish\_contam\_first.fasta --sensitive -k 20 -c 1 --evalue 1e-5 --threads 48 --db nr.dmnd --out fish\_diamond\_5.out

diamond blastx -q bacteria\_contam\_first.fasta --sensitive -k 20 -c 1 --evalue 1e-5 --threads 48 --db nr.dmnd --out bacteria\_diamond\_5.out

export LANG=C; export LC\_ALL=C; sort -k1,1 -k12,12gr -k11,11g -k3,3gr bacteria\_diamond\_5.out | sort -u -k1,1 --merge >> diamond\_bestHits

export LANG=C; export LC\_ALL=C; sort -k1,1 -k12,12gr -k11,11g -k3,3gr fish\_diamond\_5.out | sort -u -k1,1 --merge >> diamond\_bestHits

# Add taxonomy information, this step requires software acc2tax.

give\_tax\_2\_diamond\_blastx.pl diamond\_bestHits

# Extract contamination keys.

grep "Bacteria" diamond\_blastx\_with\_tax > list\_2

grep " Teleostomi" diamond\_blastx\_with\_tax > list\_2

cat bacteria/list\_2 fish/list\_2 > bad\_list

cat bad\_list | cut -f1 > 1

remove\_duplicate.pl 1

mv duplicate\_remove true\_bad\_list

# Get the second round decontamination results, WL\_second\_all\_filter\_Unigene.fasta.

remove\_contaminant\_by\_ID.pl WL\_all\_filter\_Unigene.fasta true\_bad\_list

1. **Remove contamination from genomes by TAGC methods**
   1. **Install Blobtools, details see** [**https://blobtools.readme.io/docs**](https://blobtools.readme.io/docs)
   2. **Process the genomes and get the mapping results**

filter\_fasta\_by\_length.pl WL\_genome.fasta 200 200000 WL\_genome\_200.fasta

bowtie2-build WL\_genome\_200.fasta index --threads 8

bowtie2 -p 24 -x index -1 M.wulii\_1.fq -2 M.wulii\_2.fq -k 1 --very-fast-local -S out.sam

samtools view -bS out.sam > out.bam

* 1. **Blast against NCBI Nucleotide database using megablast and against UniRef90 using diamond BLASTX**

# Megablast:

blastn -task megablast -query WL\_genome\_200.fasta -db nt -culling\_limit 5 -outfmt '6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle staxids sscinames sskingdoms' -num\_threads 48 -evalue 1e-25 -out assembly\_megablast\_25.out

# Add taxonomy information.

perl -lne '

BEGIN{open UT, "<acc2tax\_nucl\_all.txt" or die $!; while (<UT>) { $ut{$2}=$3 if /^(\S+)\t(\S+)\t(\S+)/ } }

{print "$\_\t$ut{$1}" if /^\S+\t(\S+)/ and exists $ut{$1}}' \

< assembly\_megablast\_25.out \

> assembly\_megablast\_25.out\_taxid

# Change the result style for Blobtools.

awk -v OFS="\t" -F"\t" '{print $1,$17,$12}' assembly\_megablast\_25.out\_taxid > mega.out

# Diamond:

diamond makedb --in uniref90.aa -d uniref90

diamond blastx -q WL\_genome\_200.fasta --sensitive -k 20 -c 1 --evalue 1e-10 --threads 48 --db uniref90.dmnd --out assembly\_diamond\_10.out

# Add taxonomy information.

perl -lne '

BEGIN{open UT, "<uniref90.taxlist" or die $!; while (<UT>) { $ut{$1}=$2 if /^(\S+)\t(\S+)$/ } }

{print "$\_\t$ut{$1}" if /^\S+\t(\S+)/ and exists $ut{$1}}' \

< assembly\_diamond\_10.out \

> assembly\_diamond\_10.out\_taxid

# Change the result style for Blobtools.

awk -v OFS="\t" -F"\t" '{print $1,$13,$12}' assembly\_diamond\_10.out\_taxid > diamond.out

* 1. **Process above results by the blobtools script to annotate each scaffold**

# Sort the blast results.

cat mega.out diamond.out > blast.out

sort\_blast\_by\_query\_name.pl blast.out

mv sorted\_output blast.out

# Create blob database.

python2.7 blobtools create -i WL\_genome\_200.fasta -b out.bam -t blast.out -o M.wulii\_1\_blob --names names.dmp --nodes nodes.dmp

# Process the blob database.

python2.7 ../blobtools view -i M.wulii\_1\_blob.blobDB.json -o ./

* 1. **Visualize the annotation results into Taxon-Annotated-Gc-Coverage plot (TAGC)**

python2.7 blobtools blobplot -i M.wulii\_1\_blob.blobDB.json -o ./ --format pdf --colours colours.txt

* 1. **Potential contamination is inspected manually and compared against NCBI Nucleotide database**

# Reformat the result.

format.sh > result

# Inspect scaffolds with a bit-score ≥200.

blob\_result\_seq\_extract.pl M.wulii \_1\_blob.blobDB.table.txt

# Extract potential contamination sequences.

extract\_seq\_from\_fasta.pl WL\_genome\_200.fasta seq\_for\_blast

mv extracted.fasta contam\_candidate.fa

# Blast nt database.

blastn -query contam\_candidate.fa -db nt -evalue 1e-5 -max\_target\_seqs 20 -num\_threads 24 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms" -out nt\_result

# Add taxonomy information.

export LANG=C; export LC\_ALL=C; sort -k1,1 -k12,12gr -k11,11g -k3,3gr nt\_result | sort -u -k1,1 --merge > bestHits

cat bestHits | cut -f2 > acc

perl -p -i -e 's/\.(\d)//g' acc

acc2tax -i acc -o result -d accession2taxid

# Extract contamination sequences according to customized key words and get the accession number.

Teleostomi\_Bacteria\_extract.pl result

# Get the contamination sequence header (final\_contam\_header) according to the accession number.

cat bestHits | cut -f1,2 > header

perl -p -i -e 's/\.(\d)//g' header

get\_contam\_from\_accesion.pl contam\_accession header

# Also add sequences whose bam0 lower than 20 (seq\_remove\_by\_bam0) to removal lists.

another\_blob\_result\_seq\_extract.pl M.wulii\_1\_blob.blobDB.table.txt

cat seq\_remove\_by\_bam0 final\_contam\_header > true\_bad\_list

remove\_duplicate.pl true\_bad\_list

mv duplicate\_remove true\_bad\_list

**3.7 Remove contamination in genomes**

remove\_contaminant\_by\_ID.pl WL\_genome\_200.fasta true\_bad\_list

mv survive.fasta genome.fasta

**4. Genome gene prediction**

**4.1 GeneMark-ET provides** **species HMM file to** **Augustus for training**

# Map transcriptome reads to genome.

tophat -o TophatOutput -p 4 --no-novel-juncs ~/transcriptome/myxobolus/carp\_remov/index/genome ~/transcriptome/myxobolus/processing/combinedF.fastq ~/transcriptome/myxobolus/processing/combinedR.fastq

# Make hints.

bet\_to\_gff.pl --bed junctions.bed -gff introns.gff --label tophat2 --seq genome.fasta

gmes\_petap.pl --sequence genome.fasta --ET introns.gff --et\_score 10 --cores 4 --min\_contig 2000

/opt/biosoft/PASApipeline-2.0.2/misc\_utilities/gtf\_to\_gff3\_format.pl genemark.gtf genome.fasta > genemark.gff3

# Filter good models from GeneMark-ET results.

filterGenemark.pl genemark.gtf introns.gff

/opt/biosoft/PASApipeline-2.0.2/misc\_utilities/gtf\_to\_gff3\_format.pl genemark.f.good.gtf genome.fasta >genemark.f.good.gff3

mv genemark.f.good.gff3 best\_candidates.gff3

# Extract protein sequences.

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/gff3\_file\_to\_proteins.pl best\_candidates.gff3 genome.fasta prot > best\_candidates.fasta

remove\_redundant\_high\_identity\_genes.pl best\_candidates.gff3 best\_candidates.fasta 4 0.70 > best\_candidates.lowIdentity.gff3 2> remove\_redundant\_high\_identity\_genes.log

# best\_candidates.lowIdentity.gff3 can be used for augustus training.

**4.2 Run Augustus gene prediction**

# Change the style.

gff2gbSmallDNA.pl best\_candidates.lowIdentity.gff3 genome.fasta 800 genes.raw.gb

# Remove the bad models in genes.raw.gb.

new\_species.pl --species=for\_bad\_genes\_removing

etraining --species=for\_bad\_genes\_removing --stopCodonExcludedFromCDS=false genes.raw.gb 2> train.err

cat train.err | perl -pe 's/.\*in sequence (\S+): .\*/$1/' > badgenes.lst

filterGenes.pl badgenes.lst genes.raw.gb > genes.gb

# First training with training set.

randomSplit.pl genes.gb 100

new\_species.pl --species=myxobolus\_wulii

etraining --species=myxobolus\_wulii genes.gb.train > train.out

# First gene prediction with test set.

augustus --species=myxobolus\_wulii genes.gb.test | tee firsttest.out

# Optimize the parameters.

optimize\_augustus.pl --species=myxobolus\_wulii --cpus=8 genes.gb.train

# Second training with training set.

etraining --species=myxobolus\_wulii genes.gb.train

# Second gene prediction with test set.

augustus --species=myxobolus\_wulii genes.gb.test | tee secondtest.out

# Mask genome with RepeatMasker and RepeatModeler.

mkdir repeatMasker

cd repeatMasker

fasta\_no\_blank.pl genome.fasta > genome2.fasta

rm genome.fasta

mv genome2.fasta genome.fasta

RepeatMasker -pa 24 -e ncbi -species cnidaria -gff -dir repeatMasker genome.fasta

cd ..

mkdir repeatModeler

cd repeatModeler

/opt/biosoft/RepeatModeler/BuildDatabase -name wulii -engine ncbi genome.fasta

/opt/biosoft/RepeatModeler/RepeatModeler -database wulii -pa 8

/opt/biosoft/RepeatMasker/RepeatMasker -pa 4 -e ncbi -lib RM\_37443.MonJan231456212017/consensi.fa.classified -dir ./ -gff genome.fasta

# Combine RepeatMasker and RepeatModeler results.

cd ..

merge\_repeatMasker\_out.pl repeatMasker/genome.fasta.out repeatModeler/genome.fasta.out > genome.repeat.stats

maskedByGff.pl genome.repeat.gff3 genome.fasta hardmaskN > genome.hardmaskN.fasta

# Map RNA-seq reads to masked genome.

mv genome.hardmaskN.fasta genome\_db.fa

bowtie2-build genome\_db.fa genome\_db --threads 8

tophat2 -N 3 --read-edit-dist 3 -p 32 -i 20 -I 4000 --min-segment-intron 20 --max-segment-intron 4000 --min-coverage-intron 20 --max-coverage-intron 4000 --coverage-search --microexon-search -o result genome\_db M.wulii\_1\_fixed.fastq M.wulii\_2\_fixed.fastq

# Make RNA-seq hints.

bam2hints --intronsonly --in=result/accepted\_hits.bam --out=hints.gff

# Formal gene prediction.

augustus --species=myxobolus\_wulii\_1 --extrinsicCfgFile=extrinsic.cfg --alternatives-from-evidence=true --allow\_hinted\_splicesites=atac --hintsfile=hints.gff --gff3=on genome.fasta > aug.gff3

perl -p -i -e 's/\ttranscript\t/\tmRNA\t/' aug.gff3

**4.3 GeneMark-ET provides species HMM file to SNAP for training**

# Use some functions in Maker.

maker2zff genemark.gff3

extract\_header\_for\_snap.pl genome.dna

# Establish index.

fastaindex genome.fasta genome.idx

fastafetch -f genome.fasta -i genome.idx -Fq <(sort -u header) > out

mv out genome.dna

# Breakdown the genome.

fathom -categorize 1000 genome.ann genome.dna

fathom uni.ann uni.dna -export 1000 -plus

mkdir params

cd params/

forge ../export.ann ../export.dna

cd ..

hmm-assembler.pl species params/ > species.hmm

# SNAP training finish.

**4.4 Run SNAP gene prediction**

snap species.hmm genome.fasta -gff -quiet > snap.gff

snap2gff3.pl snap.gff > snap\_ture.gff

**4.5 Homology-based gene prediction by Genewise**

# Prepare non-redundant close-related proteins.

rename\_fasta\_by\_numeber.pl all\_cdhit.fa

# Start the annotation.

/opt/biosoft/homolog\_genewise/homolog\_genewise.pl rename\_all.fasta genome.hardmaskN.fasta 8 0.1 1e-9

# Filter the result.

/opt/biosoft/homolog\_genewise/genewise\_filter.pl genewise.gff genome.hardmaskN.fasta 15 90 1 1e-6 0.30 4 > genewise.filter.gff 2> genewise.filter.stats

# Evaluate the completeness of predicted genes and filter the sequences containing stop codons.

/opt/biosoft/homolog\_genewise/genewise2EVM\_input.pl genewise.filter.gff genome.hardmaskN.fasta filterMiddleStopCodon=yes > evm\_protein\_alignment.gff3 2> genewise\_gene\_models\_completeness\_check.txt

# Process the output style

perl -p -i -e 's/^#.\*//; s/^\s\*$//' genewise.gff

make\_evm\_recognize\_gff.pl genewise.gff

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/misc/SNAP\_to\_GFF3.pl evm\_wise.gff3 > final\_evm\_wise.gff3

**4.6** **Gene prediction by PASA**

# Prepare the transcriptome file.

perl -e 'while (<>) { print "$1\n" if />(\S+)/ }' /home/train/00.incipient\_data/data\_for\_gene\_prediction\_and\_RNA-seq/Trinity.fasta > tdn.accs

# End-trimming of the transcriptome.

seqclean Trinity.fasta -v /opt/biosoft/PASApipeline-2.0.2/seqclean/UniVec

# Produce config file.

cp /opt/biosoft/PASApipeline-2.0.2/pasa\_conf/pasa.alignAssembly.Template.txt alignAssembly.config

DATE=`date +%Y%m%e%k%M%S | perl -pe 's/\s+//'`

echo "perl -p -i -e 's/MYSQLDB=.\*/MYSQLDB=pasa\_$DATE/' alignAssembly.config" | sh

# Produce mysql databases and tables.

/opt/biosoft/PASApipeline-2.0.2/scripts/create\_mysql\_cdnaassembly\_db.dbi -r -c alignAssembly.config -S /opt/biosoft/PASApipeline-2.0.2/schema/cdna\_alignment\_mysqlschema

# Start mapping transcripts.

/opt/biosoft/PASApipeline-2.0.2/scripts/Launch\_PASA\_pipeline.pl -c alignAssembly.config -R -g genome.fasta -t Trinity.fasta.clean -T -u Trinity.fasta --ALIGNERS gmap,blat --CPU 8 --stringent\_alignment\_overlap 30.0 --TDN tdn.accs --MAX\_INTRON\_LENGTH 20000 --TRANSDECODER &> pasa.log

# This will produce pasa\_\*.pasa\_assemblies.gff3 for downstream analyses.

**4.7 Combine above gff3 file by EVM**

# Augustus gff file.

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/misc/augustus\_GFF3\_to\_EVM\_GFF3.pl aug.gff3 > evm\_augustus.gff3

perl -p -i -e 's/^#.\*//; s/^\s\*$//' evm\_augustus.gff3

gff3\_gene\_prediction\_file\_validator.pl evm\_augustus.gff3

# Genemar-ET gff file.

ln -s ../genemark-et/genemark.gff3 evm\_genemark-et.gff3

gff3\_gene\_prediction\_file\_validator.pl evm\_genemark-et.gff3

# Snap gff file.

# Change the style.

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/misc/SNAP\_output\_to\_gff3.pl snap.zff genome.fasta > snap.gff3

gff3\_gene\_prediction\_file\_validator.pl snap.gff3

perl -p -i -e 's/^(\S+)\t(\.)\t/$1\tSNAP\t/g' snap.gff3

# PASA gff file.

cp ../new\_pasa/pasa\*.pasa\_assemblies.gff3 ./transcript\_alignments.gff3

perl -p -i -e 's/\t\S+/\tpasa\_transcript\_alignments/' transcript\_alignments.gff3

gff3\_gene\_prediction\_file\_validator.pl transcript\_alignments.gff3

# Genewise gff file.

ln -s ../genewise/wise/evm\_protein\_alignment.gff3 ./protein\_alignments.gff3

gff3\_gene\_prediction\_file\_validator.pl protein\_alignments.gff3

cat evm\_augustus.gff3 evm\_genemark-et.gff3 snap.gff3 | perl -pe 's/^#.\*//; s/^\s\*$//' > gene\_predictions.gff3

# Create weights file.

echo -e "ABINITIO\_PREDICTION\tAugustus\t6

ABINITIO\_PREDICTION\tSNAP\t2

ABINITIO\_PREDICTION\tGeneMark.hmm\t1

PROTEIN\tGeneWise\t5

TRANSCRIPT\tpasa\_transcript\_alignments\t10" > weights.txt

# Split the data.

partition\_EVM\_inputs.pl --genome genome.fasta --gene\_predictions gene\_predictions.gff3 --protein\_alignments protein\_alignments.gff3 --transcript\_alignments transcript\_alignments.gff3 --repeats genome.repeat.gff3 --segmentSize 500000 --overlapSize 10000 --partition\_listing partitions\_list.out

# Create and run commands.

write\_EVM\_commands.pl --genome genome.fasta --gene\_predictions gene\_predictions.gff3 --protein\_alignments protein\_alignments.gff3 --transcript\_alignments transcript\_alignments.gff3 --repeats genome.repeat.gff3 --weights `pwd`/weights.txt --partitions partitions\_list.out --output\_file\_name evm.out > commands.list

ParaFly -c commands.list -CPU 4

# Combine the results and reformat to gff3.

recombine\_EVM\_partial\_outputs.pl --partitions partitions\_list.out --output\_file\_name evm.out

convert\_EVM\_outputs\_to\_GFF3.pl --partitions partitions\_list.out --output\_file\_name evm.out --genome genome.fasta

find . -regex ".\*evm.out.gff3" -exec cat {} \; > EVM.all.gff3

# Extract proteins from gff3.

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/gff3\_file\_to\_proteins.pl EVM.all.gff3 ../evm/genome.fasta prot > genome\_protein.fasta

**5. Transcriptome gene prediction**

**5.1** **De novo prediction by** **TransDecoder**

# Predict ORF.

TransDecoder.LongOrfs -t WL\_second\_all\_filter\_Unigene.fasta -m 20

TransDecoder.Predict -t WL\_second\_all\_filter\_Unigene.fasta --cpu 12

# Get the Transdecoder result WL\_second\_all\_filter\_Unigene.fasta.transdecoder.pep.

**5.2 De novo prediction by GeneMarkS-T**

/opt/biosoft/GeneMarkS-T/gmst.pl --output M.wulii\_gmst --fnn --faa -clean 1 WL\_second\_all\_filter\_Unigene.fasta

# Get the GeneMarkS-T result M.wulii\_gmst.fasta.

**5.3 Homolog-based prediction by Hercules (https://github.com/qingxiangguo/hercules-v.1.0)**

# Blast transcriptome with nr, swiss, egg and kog database.

blastx -query WL\_second\_all\_filter\_Unigene.fasta -db nr -max\_target\_seqs 20 -out nr\_result -evalue 1e-5 -num\_threads 48 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms"

blastx -query WL\_second\_all\_filter\_Unigene.fasta -db kog -max\_target\_seqs 20 -out kog\_result -evalue 1e-5 -num\_threads 48 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms"

blastx -query WL\_second\_all\_filter\_Unigene.fasta -db swiss -max\_target\_seqs 20 -out swiss\_result -evalue 1e-5 -num\_threads 48 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms"

blastx -query WL\_second\_all\_filter\_Unigene.fasta -db eggnog -max\_target\_seqs 20 -out eggnog\_result -evalue 1e-5 -num\_threads 48 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms"

# Process blast results.

for next in $(cut -f1 nr\_result | sort -u); do grep -w -m 20 "$next" nr\_result; done > tmp

sort\_blast\_by\_query\_name.pl tmp

mv sorted\_output nr | rm tmp

for next in $(cut -f1 eggnog\_result | sort -u); do grep -w -m 20 "$next" eggnog\_result; done > tmp

sort\_blast\_by\_query\_name.pl tmp

mv sorted\_output eggnog | rm tmp

for next in $(cut -f1 swiss\_result | sort -u); do grep -w -m 20 "$next" swiss\_result; done > tmp

sort\_blast\_by\_query\_name.pl tmp

mv sorted\_output swiss | rm tmp

for next in $(cut -f1 kog\_result | sort -u); do grep -w -m 20 "$next" kog\_result; done > tmp

sort\_blast\_by\_query\_name.pl tmp

mv sorted\_output kog | rm tmp

hercules nr swiss egg kog

# Hercules requires the installation of anvi’o, for details, see https://github.com/merenlab/anvio/releases/v5.5.

# Hercules will combine blast results from different databases with customized priority and produce a gene-call file which can be fed into Anvi'o.

anvi-gen-contigs-database -f contigs.fa -o contigs.db --external-gene-calls gene\_call

# Get the proteins.

anvi-get-aa-sequences-for-gene-calls -c contigs.db -o homolog.fa

**5.4 Six-frame translation**

# Those transcripts that were translated neither by de novo nor homolog-based method are translated into amino acid sequences using the Transeq script from the EMBOSS.

# Get those sequences that were translated neither by de novo nor homolog-based method.

cp ../trans\_homolog/gene\_call ./

grep ">" ../trans\_denovo/M.wulii\_gmst.faa > gmst

grep ">" ../trans\_denovo/WL\_second\_all\_filter\_Unigene.fasta.transdecoder.pep > transdecoder

cat gene\_call | cut -f2 > gene\_call\_header

perl -p -i -e "s/\s.\*//" transdecoder

perl -p -i -e "s/>Gene.\d+:://" transdecoder

perl -p -i -e "s/:.\*//" transdecoder

perl -p -i -e "s/\s.\*//" gmst

perl -p -i -e "s/>//" gmst

cat gene\_call\_header gmst transdecoder > annotation\_list

remove\_duplicate.pl annotation\_list

rm annotation\_list

mv duplicate\_remove annotation\_list

remove\_contaminant\_by\_ID.pl WL\_second\_all\_filter\_Unigene.fasta annotation\_list

mv survive.fasta trans\_left.fasta

# Six-frame translation using EMBOSS.

transeq -sequence trans\_left.fasta -outseq out -frame 6

flat\_the\_fasta\_seq.pl out

rm out

# Collect the amino acid sequence between stop codons (represented by \*) and discard the sequence contained ambiguous amino acids (represented by X) or its total length is less than 30.

get\_seq\_between\_asterisk.pl flated 30

mv between\_asterisk trans\_left.pep

**6. Combine all the proteins predicted from genomes and transcriptomes**

**6.1 Collect all predicted proteins**

cat genome\_protein.fasta WL\_second\_all\_filter\_Unigene.fasta.transdecoder.pep M.wulii\_gmst.fasta homolog.fa trans\_left.pep > all.fasta

**6.2 Remove redundancy and filter by length**

cd-hit -i all.fasta -o all\_cdhit.fasta -c 1 -T 4 -M 0

filter\_fasta\_by\_length.pl all\_cdhit.fasta 30 1000000 filtered.fasta

**6.****3 Give name tag to CCPRD**

replace\_fasta\_header\_by\_number.pl filtered.fasta WL\_MC

mv ordered.fasta CCPRD

# Get the CCPRD here.

**7. Create alternative databases for comparison**

**7.1 Transcriptome six-frame translation**

transeq -sequence WL\_second\_all\_filter\_Unigene.fasta -outseq out -frame 6 -clean

remove\_no\_end\_asterisks.pl out

flat\_the\_fasta\_seq.pl out

rm out

get\_seq\_between\_asterisk.pl flated 30

cd-hit -i between\_asterisk -o out -c 1 -M 160000 -T 8

replace\_fasta\_header\_by\_number.pl out WL\_T6

mv ordered.fasta trans\_6\_frame

**7.2 Genome and transcriptome six-frame translation**

cat genome.fasta WL\_second\_all\_filter\_Unigene.fasta > all.fasta

transeq -sequence all.fasta -outseq out -frame 6 -clean

remove\_no\_end\_asterisks.pl out

flat\_the\_fasta\_seq.pl out

rm out

get\_seq\_between\_asterisk.pl flated 30

cd-hit -i between\_asterisk -o out -c 1 -M 160000 -T 8

replace\_fasta\_header\_by\_number.pl out WL\_A6

**7.3 CCPRD + contaminants**

# Add artificial host and bacteria sequences to CCPRD, and give them the name tag WL\_BAC, WL\_HOST respectively.

**7.4 CCPRD + sequences removed in decontamination process**

# Add back sequences that were removed during the de-contamination process, and give them the name tag WL\_RM.

**Examples**

# The CCPRD workflow was tested in a model organism, *Saccharomyces cerevisiae* and provided as example file. The public data of *S. cerevisiae* S288C were used, which included Ensembl annotated version of S288c genome R64-1-1, GenBank GCA\_000146045.2, and transcriptome shotgun assembly, GenBank GFJR01000000. Since the genome of S288c is a gold-standarded data, here we skipped the genome decontamination process by TAGC methods.

**1. Download and process the transcriptomes and genomes**

**1.1 Download the source data**

mkdir 1\_source\_data && cd 1\_source\_data

wget <https://sra-download.ncbi.nlm.nih.gov/traces/wgs03/wgs_aux/GF/JR/GFJR01/GFJR01.1.fsa_nt.gz>

gunzip GFJR01.1.fsa\_nt.gz

wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/146/045/GCF\_000146045.2\_R64/GCF\_000146045.2\_R64\_genomic.fna.gz

gunzip GCF\_000146045.2\_R64\_genomic.fna.gz

**1.2 Remove redundancy in transcriptomes with CD-HIT**

cd ..

mkdir 2\_cluster \_mRNA && cd 2\_cluster \_mRNA

ln -s ../1\_source\_data/GFJR01\_mRNA.fasta ./

cd-hit-est -i GFJR01\_mRNA.fasta -o yeast.fasta\_cdhit -c 0.95 -n 10 -d 0 -M 16000 -T 8

**1.3 Further cluster the transcriptomes with TGICL**

tgicl -l 40 -c 10 -v 25 -O '-repeat\_stringency 0.95 -minmatch 35 -minscore 35' -F yeast.fasta\_cdhit

**1.4 Process the cd-hit results to get unigenes**

cat asm\_1/contigs asm\_2/contigs asm\_3/contigs asm\_4/contigs asm\_5/contigs asm\_6/contigs asm\_7/contigs asm\_8/contigs asm\_9/contigs asm\_10/contigs > all.contigs

fast\_extract\_seq\_from\_fasta.pl yeast.fasta\_cdhit yeast.fasta\_cdhit.singletons > extracted.fasta

Unigene\_generator.pl -s extracted.fasta -c all.contigs -t YT

# This will output YT\_Unigene.fasta for downstream analyses.

**2. Remove potential bacterial contamination in transcriptome data**

#We use the same bacterial database used in above Pipeline section 2.6-2.9

**2.1 Collected proteins for constructing bacterial no-redundant database**

# Bacterial no-redundant database contains: Genome predicted proteins from, *Aeromonas caviae*, *Aeromonas hydrophila*, *Aeromonas sobria*, *Aeromonas veronii B565*, *Citrobacter freundii CFNIH1*, *Escherichia coli*, *Flavobacterium columnare*, *Klebsiella pneumoniae*, *Lactobacillus sakei*, *Pseudomonas aeruginosa*, *Salmonella enterica*, *Shewanella putrefaciens*, *Plesiomonas shigelloides*.

**2.2 Remove redundancy in bacterial database and blast the transcriptomes**

cd ..

mkdir 3\_decontam\_mRNA

cd 3\_decontam\_mRNA/

cd-hit -i bac\_protein.fasta -o bac\_protein\_cdhit.fasta -c 0.95 -n 5 -T 8

# 45530 proteins retained in bacterial databases after redundancy removal.

makeblastdb -in bac\_protein\_cdhit.fasta -out BAC -dbtype prot -parse\_seqids -hash\_index

blastx -query ../../2\_cluster\_mRNA/YT\_Unigene.fasta -db BAC -out result\_bac\_e10 -evalue 1e-10 -outfmt 6 -num\_threads 32

**2.3 First round removement of bacterial and confirmation**

cat result\_bac\_e10 | cut -f1 > 1

remove\_duplicate.pl 1

mv duplicate\_remove bacteria\_contam\_list

extract\_seq\_from\_fasta.pl ../../2\_cluster\_mRNA/YT\_Unigene.fasta bacteria\_contam\_list

mv extracted.fasta bacteria\_contam\_first.fasta

# Confirm those sequences by blasting with nr database.

blastx -query bacteria\_contam\_first.fasta -db nr -out bacteria\_contam\_first\_nr\_result -evalue 1e-5 -max\_target\_seqs 1 -num\_threads 56 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms"

grep "Bacteria" bacteria\_contam\_first\_nr\_result | cut -f 1 > 2

remove\_duplicate.pl 2 | rm 2

mv duplicate\_remove true\_bacteria\_contam.list

remove\_contaminant\_by\_ID.pl WL\_host\_delete\_Unigene.fasta true\_bacteria\_contam.list

# This step will produce the final clean transcriptome with bacterial contamination conservatively removed, however, if the proportion of true bacterial contamination in bacteria\_contam\_list is too large, a second-round decontamination is recommended. In present case, we are ok with the yeast transcriptome, almost no bacteria contamination was detected.

**3. Genome gene prediction**

# Since *S. cerevisiae* is a model organism with good species HMM file, we used the HMM file in prediction software instead of making our own.

mkdir 4\_MS\_database

cd 4\_MS\_database

mkdir all\_6\_frame CCPRD CCPRD\_contam trans\_6\_frame

cd 4\_MS\_database/CCPRD

mkdir genome\_denovo genome\_homolog genome\_pasa trans\_denovo trans\_homolog trans\_left\_six\_frame

**3.1 GeneMark-ES prediction**

cd 4\_MS\_database/CCPRD/ genome\_denovo/GeneMark-ES

ln -s ~/examples/1\_source\_data/GCF\_000146045.2\_R64\_genomic.fna ./genome.fasta

gmes\_petap.pl –sequence genome.fasta –ES –fungus –cores 4

gtf\_to\_gff3\_format.pl genemark.gtf genome.fasta > genemark.gff3

**3.2 Augustus gene prediction**

cd 4\_MS\_database/CCPRD/ genome\_denovo/AUGUSTUS

ln -s ~/examples/1\_source\_data/GCF\_000146045.2\_R64\_genomic.fna ./genome.fasta

augustus --species=saccharomyces\_cerevisiae\_S288C --gff3=on genome.fasta > aug.gff3

**3.3 SNAP gene prediction**

cd 4\_MS\_database/CCPRD/ genome\_denovo/SNAP

ln -s ../GeneMark-ES/genome.fasta ./genome.fasta

ln -s ../GeneMark-ES/genemark.gff3 ./

# Use some functions in Maker.

maker2zff genemark.gff3

extract\_header\_for\_snap.pl genome.dna

# Establish index.

fastaindex genome.fasta genome.idx

fastafetch -f genome.fasta -i genome.idx -Fq <(sort -u header) > out

mv out genome.dna

# Breakdown the genome.

fathom -categorize 1000 genome.ann genome.dna

fathom uni.ann uni.dna -export 1000 -plus

mkdir params

cd params/

forge ../export.ann ../export.dna

cd ..

hmm-assembler.pl species params/ > species.hmm

# SNAP training finish and start prediction

snap species.hmm genome.fasta -gff -quiet > snap.gff

snap2gff3.pl snap.gff > snap\_true.gff

**3.4 Homology-based gene prediction by Genewise**

# We use the *S. cerevisiae* proteins in UniProtKB/Swiss-Prot as close-related proteins.

# Start the annotation.

/opt/biosoft/homolog\_genewise/homolog\_genewise.pl rename\_all.fasta genome.hardmaskN.fasta 8 0.1 1e-9

# Filter the result.

/opt/biosoft/homolog\_genewise/genewise\_filter.pl genewise.gff genome.hardmaskN.fasta 15 90 1 1e-6 0.30 4 > genewise.filter.gff 2> genewise.filter.stats

# Evaluate the completeness of predicted genes and filter the sequences containing stop codons.

/opt/biosoft/homolog\_genewise/genewise2EVM\_input.pl genewise.filter.gff genome.hardmaskN.fasta filterMiddleStopCodon=yes > evm\_protein\_alignment.gff3 2> genewise\_gene\_models\_completeness\_check.txt

# Process the output style

perl -p -i -e 's/^#.\*//; s/^\s\*$//' genewise.gff

make\_evm\_recognize\_gff.pl genewise.gff

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/misc/SNAP\_to\_GFF3.pl evm\_wise.gff3 > final\_evm\_wise.gff3

**3.5 Gene prediction by PASA**

cd 4\_MS\_database/CCPRD/genome\_pasa

# Prepare the transcriptome file.

perl -e 'while (<>) { print "$1\n" if />(\S+)/ }' YT\_Unigene.fasta > tdn.accs

# End-trimming of the transcriptome.

seqclean YT\_Unigene.fasta -v /opt/biosoft/PASApipeline-2.0.2/seqclean/UniVec

# Produce config file.

cp /opt/biosoft/PASApipeline-2.0.2/pasa\_conf/pasa.alignAssembly.Template.txt alignAssembly.config

DATE=`date +%Y%m%e%k%M%S | perl -pe 's/\s+//'`

echo "perl -p -i -e 's/MYSQLDB=.\*/MYSQLDB=pasa\_$DATE/' alignAssembly.config" | sh

# Produce mysql databases and tables.

/opt/biosoft/PASApipeline-2.0.2/scripts/create\_mysql\_cdnaassembly\_db.dbi -r -c alignAssembly.config -S /opt/biosoft/PASApipeline-2.0.2/schema/cdna\_alignment\_mysqlschema

# Start mapping transcripts.

/opt/biosoft/PASApipeline-2.0.2/scripts/Launch\_PASA\_pipeline.pl -c alignAssembly.config -R -g ../genome\_denovo/GeneMark-ES/genome.fasta -t YT\_Unigene.fasta.clean -T -u ../../../2\_cluster\_mRNA/YT\_Unigene.fasta --ALIGNERS gmap,blat --CPU 8 --stringent\_alignment\_overlap 30.0 --TDN tdn.accs --MAX\_INTRON\_LENGTH 20000 --TRANSDECODER &> pasa.log

# This will produce pasa\_\*.pasa\_assemblies.gff3 for downstream analyses.

**3.6 Combine above gff3 file by EVM**

# Augustus gff file.

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/misc/augustus\_GFF3\_to\_EVM\_GFF3.pl aug.gff3 > evm\_augustus.gff3

perl -p -i -e 's/^#.\*//; s/^\s\*$//' evm\_augustus.gff3

gff3\_gene\_prediction\_file\_validator.pl evm\_augustus.gff3

# Genemar-ES gff file.

ln -s 4\_MS\_database/CCPRD/genome\_denovo/GeneMark-ES/genemark.gff3 evm\_genemark-es.gff3

gff3\_gene\_prediction\_file\_validator.pl evm\_genemark-es.gff3

# Snap gff file.

# Change the style.

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/misc/SNAP\_output\_to\_gff3.pl snap.zff genome.fasta > snap.gff3

gff3\_gene\_prediction\_file\_validator.pl snap.gff3

perl -p -i -e 's/^(\S+)\t(\.)\t/$1\tSNAP\t/g' snap.gff3

# PASA gff file.

cp ../new\_pasa/pasa\*.pasa\_assemblies.gff3 ./transcript\_alignments.gff3

perl -p -i -e 's/\t\S+/\tpasa\_transcript\_alignments/' transcript\_alignments.gff3

gff3\_gene\_prediction\_file\_validator.pl transcript\_alignments.gff3

# Genewise gff file.

ln -s ../genewise/wise/evm\_protein\_alignment.gff3 ./protein\_alignments.gff3

gff3\_gene\_prediction\_file\_validator.pl protein\_alignments.gff3

cat evm\_augustus.gff3 evm\_genemark-es.gff3 snap.gff3 | perl -pe 's/^#.\*//; s/^\s\*$//' > gene\_predictions.gff3

# Create weights file.

echo -e "ABINITIO\_PREDICTION\tAugustus\t6

ABINITIO\_PREDICTION\tSNAP\t2

ABINITIO\_PREDICTION\tGeneMark.hmm\t1

PROTEIN\tGeneWise\t5

TRANSCRIPT\tpasa\_transcript\_alignments\t10" > weights.txt

# Split the data.

partition\_EVM\_inputs.pl --genome genome.fasta --gene\_predictions gene\_predictions.gff3 --protein\_alignments protein\_alignments.gff3 --transcript\_alignments transcript\_alignments.gff3 --repeats genome.repeat.gff3 --segmentSize 500000 --overlapSize 10000 --partition\_listing partitions\_list.out

# Create and run commands.

write\_EVM\_commands.pl --genome genome.fasta --gene\_predictions gene\_predictions.gff3 --protein\_alignments protein\_alignments.gff3 --transcript\_alignments transcript\_alignments.gff3 --repeats genome.repeat.gff3 --weights `pwd`/weights.txt --partitions partitions\_list.out --output\_file\_name evm.out > commands.list

ParaFly -c commands.list -CPU 4

# Combine the results and reformat to gff3.

recombine\_EVM\_partial\_outputs.pl --partitions partitions\_list.out --output\_file\_name evm.out

convert\_EVM\_outputs\_to\_GFF3.pl --partitions partitions\_list.out --output\_file\_name evm.out --genome genome.fasta

find . -regex ".\*evm.out.gff3" -exec cat {} \; > EVM.all.gff3

# Extract proteins from gff3.

/opt/biosoft/EVidenceModeler-1.1.1/EvmUtils/gff3\_file\_to\_proteins.pl EVM.all.gff3 ../evm/genome.fasta prot > genome\_protein.fasta

**4. Transcriptome gene prediction**

**4.1 De novo prediction by TransDecoder**

# Predict ORF.

cd 4\_MS\_database/CCPRD/trans\_denovo

TransDecoder.LongOrfs -t ../../../2\_cluster\_mRNA/YT\_Unigene.fasta -m 20

TransDecoder.Predict -t ../../../2\_cluster\_mRNA/YT\_Unigene.fasta --cpu 12

# Get the Transdecoder result YT\_Unigene.fasta.transdecoder.pep

**4.2 De novo prediction by GeneMarkS-T**

gmst.pl --output YT\_gmst --fnn --faa -clean 1 ../../../2\_cluster\_mRNA/YT\_Unigene.fasta

**4.3 Homolog-based prediction by Hercules (**[**https://github.com/qingxiangguo/hercules-v.1.0**](https://github.com/qingxiangguo/hercules-v.1.0)**)**

cd 4\_MS\_database/CCPRD/trans\_homolog

# Blast transcriptome with swiss and kog database.

blastx -query ../../../2\_cluster\_mRNA/YT\_Unigene.fasta -db KOG -max\_target\_seqs 20 -out kog\_result -evalue 1e-5 -num\_threads 48 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms"

blastx -query ../../../2\_cluster\_mRNA/YT\_Unigene.fasta -db swiss -max\_target\_seqs 20 -out swiss\_result -evalue 1e-5 -num\_threads 48 -outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore stitle frames sscinames sskingdoms"

# Process blast results.

for next in $(cut -f1 swiss\_result | sort -u); do grep -w -m 20 "$next" swiss\_result; done > tmp

sort\_blast\_by\_query\_name.pl tmp

mv sorted\_output swiss | rm tmp

for next in $(cut -f1 kog\_result | sort -u); do grep -w -m 20 "$next" kog\_result; done > tmp

sort\_blast\_by\_query\_name.pl tmp

mv sorted\_output kog | rm tmp

hercules nr swiss egg kog

# Hercules requires the installation of anvi’o, for details, see https://github.com/merenlab/anvio/releases/v5.5.

# Hercules will combine blast results from different databases with customized priority and produce a gene-call file which can be fed into Anvi'o.

anvi-gen-contigs-database -f ../../../2\_cluster\_mRNA/YT\_Unigene.fasta -o contigs.db --external-gene-calls gene\_call

# Get the proteins.

anvi-get-aa-sequences-for-gene-calls -c contigs.db -o homolog.fa

**4.4 Six-frame translation**

# Those transcripts that were translated neither by de novo nor homolog-based method are translated into amino acid sequences using the Transeq script from the EMBOSS.

cd 4\_MS\_database/CCPRD/trans\_left\_six\_frame

# Get those sequences that were translated neither by de novo nor homolog-based method.

cp ../trans\_homolog/gene\_call ./

grep ">" ../trans\_denovo/YT\_gmst.faa > gmst

grep ">" ../trans\_denovo/YT\_Unigene.fasta.transdecoder.pep > transdecoder

cat gene\_call | cut -f2 > gene\_call\_header

perl -p -i -e "s/\s.\*//" transdecoder

perl -p -i -e "s/>Gene.\d+:://" transdecoder

perl -p -i -e "s/:.\*//" transdecoder

perl -p -i -e "s/\s.\*//" gmst

perl -p -i -e "s/>//" gmst

cat gene\_call\_header gmst transdecoder > annotation\_list

remove\_duplicate.pl annotation\_list

rm annotation\_list

mv duplicate\_remove annotation\_list

remove\_contaminant\_by\_ID.pl ../../../2\_cluster\_mRNA/YT\_Unigene.fasta annotation\_list

mv survive.fasta trans\_left.fasta

# Six-frame translation using EMBOSS.

transeq -sequence trans\_left.fasta -outseq out -frame 6

flat\_the\_fasta\_seq.pl out

rm out

get\_seq\_between\_asterisk.pl flated 30

mv between\_asterisk trans\_left.pep

**5. Combine all the proteins predicted from genomes and transcriptomes**

**5.1 Collect all predicted proteins**

cd 4\_MS\_database/CCPRD/final

cat genome\_protein.fasta YT\_Unigene.fasta.transdecoder.pep YT\_gmst.fasta homolog.fa trans\_left.pep > all.fasta

**5.2 Remove redundancy and filter by length**

cd-hit -i all.fasta -o all\_cdhit.fasta -c 1 -T 4 -M 0

filter\_fasta\_by\_length.pl all\_cdhit.fasta 30 1000000 filtered.fasta

**5.3 Give name tag to CCPRD**

replace\_fasta\_header\_by\_number.pl filtered.fasta YT\_CC

mv ordered.fasta CCPRD

# Get the CCPRD here.

**6. Create alternative databases for comparison**

**6.1 Transcriptome six-frame translation**

cd 4\_MS\_database/ trans\_6\_frame

transeq -sequence ../../2\_cluster\_mRNA/YT\_Unigene.fasta -outseq out -frame 6

remove\_no\_end\_asterisks.pl out

flat\_the\_fasta\_seq.pl out

rm out

get\_seq\_between\_asterisk.pl flated 30

cd-hit -i between\_asterisk -o out -c 1 -M 160000 -T 8

replace\_fasta\_header\_by\_number.pl out YT\_T6

mv ordered.fa trans\_6\_frame

**6.2 Genome and transcriptome six-frame translation**

cd 4\_MS\_database/ all\_6\_frame

cat ~/examples/1\_source\_data/GCF\_000146045.2\_R64\_genomic.fna ../../2\_cluster\_mRNA/YT\_Unigene.fasta > all.fasta

transeq -sequence all.fasta -outseq out -frame 6

flat\_the\_fasta\_seq.pl out

rm out

get\_seq\_between\_asterisk.pl flated 30

cd-hit -i between\_asterisk -o out -c 1 -M 160000 -T 8

replace\_fasta\_header\_by\_number.pl out YT\_A6

mv ordered.fa all\_6\_frame

**6.3 CCPRD + contaminants**

# Add artificial bacteria sequences to CCPRD, and give them the name tag YT\_BAC respectively.

cat CCPRD BAC.fa > CCPRD\_contam

**Citing CCPRD and software called by CCPRD**

Since CCPRD is a pipeline that depends several Bioinformatics tools, publication of results obtained by CCPRD requires that not only CCPRD is cited, but also the tools that are used by CCPRD:

Please cite:

1. Besemer, J., Lomsadze, A., & Borodovsky, M. (2001). Genemarks: a self-training method for prediction of gene starts in microbial genomes. implications for finding sequence motifs in regulatory regions. Nucleic Acids Research, 29(12), 2607-2618.
2. Birney, E., Clamp, M., Durbin, R. (2004). GeneWise and Genomewise. Genome Research, 14, 988-995.
3. Eren, A. M., Özcan C. Esen, Quince, C., Vineis, J. H., Morrison, H. G., & Sogin, M. L., et al. (2015). Anvi’o: an advanced analysis and visualization platform for ‘omics data. Peerj, 3.
4. Grabherr, M. G. , Haas, B. J. , Yassour, M. , Levin, J. Z. , Thompson, D. A. , & Amit, I. , et al. (2011). Full-length transcriptome assembly from rna-seq data without a reference genome. Nature Biotechnology, 29(7), 644-652.
5. Haas, B. J. , Delcher, A. L. , Mount, S. M. , Wortman, J. R. , & White, O. . (2003). Improving the arabidopsis genome annotation using maximal transcript alignment assemblies. Nucleic Acids Research, 31(19), 5654-5666.
6. Haas, B. J. , Salzberg, S. L. , Zhu, W. , & Mihaela Pertea…. (2008). Automated eukaryotic gene structure annotation using evidencemodeler and the program to assemble spliced alignments. Genome biology, 9(1).
7. Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., & Bowden, J., et al. (2013). De novo transcript sequence reconstruction from rna-seq using the trinity platform for reference generation and analysis. Nature Protocols, 8(8), 1494-1512.
8. Huerta-Cepas, J., Serra, F., & Bork, P. (2016). Ete 3: reconstruction, analysis, and visualization of phylogenomic data. Molecular Biology & Evolution, 33(6), 1635-1638.
9. Korf, I. (2004). Gene finding in novel genomes. Bmc Bioinformatics, 5(1), 59.
10. Laetsch, D. R., Blaxter, M. L. (2017). Blobtools: Interrogation of genome assemblies. F1000Research, 6, 1287.
11. Li, W., & Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics, 22(13), 1658.
12. Lomsadze, A. , Burns, P. D. , & Borodovsky, M. . (2014). Integration of mapped rna-seq reads into automatic training of eukaryotic gene finding algorithm. Nucleic Acids Research, 42(15), e119-e119.
13. Pertea, G., Huang, X., Liang, F., Antonescu, V., Sultana, R., & Karamycheva, S., et al. (2003). Tigr gene indices clustering tools (tgicl): a software system for fast clustering of large est datasets. Bioinformatics, 19(5), 651-652.
14. Stanke, M., Keller, O., Gunduz, I., Hayes, A., Waack, S., & Morgenstern, B. (2006). Augustus: ab initio prediction of alternative transcripts. Nucleic Acids Research, 34 (Web Server issue), 435-9.

**License**

All source code, i.e. scripts/\*.pl, scripts/\*.sh or scripts/\*.py are under the MIT license.