**Bash commands used in   
“Draft genomes of the corallimorpharians *Amplexidiscus fenestrafer* and *Discosoma* sp.”**

**1. Genome assembly**

**Filtering raw reads**

# Remove adapters, low quality sequences ((< Q30 and < 30 bp) and duplicates reads using Trimmomatic v0.32

*java -jar trimmomatic.jar PE -threads 40 -trimlog trimmoatic.log Raw\_final\_R1.fastq Filter\_R1.fastq unpaired\_R1.fastq Raw\_final\_R2.fastq Filter\_R2.fastq unpaired\_R2.fastq ILLUMINACLIP:TruSeq3-SE.fa:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:30 MINLEN:30*

**Mapping to *S. minutum* and *S. microadriaticum***

# Removing potential contamination from two *Symbiodinium* sources using Bowtie2 v2.1.0.

# Combine two *Symbiodinium* genomes and build the index

*bowtie2-build –f genome.fa genome\_index*

# Map different libraries to two genomes

*bowtie2 –x genome\_index -1 Filtered\_R1.fastq -2 Filtered\_R2.fastq -q -t --un mapresult\_un --al mapresult\_al --un-conc mapresult\_un\_conc --al-conc mapresult\_al\_conc -p 10 –S mapping.sam*

**Genome size estimate**

# Estimate genome size using KmerFreq\_AR under different K-mer

*KmerFreq\_AR -k 17 -t 4 -c -1 -p test test\_read.lst >kmerfreq.cout 2>kmerfreq.cerr*

**Digital normalization** (Khmer v1.4)

# Normalize everything to a coverage of 20 for paired-end reads

*interleave-reads.py Filter.1.fastq Filter.2.fastq > Lall\_B\_filter.fastq*

*normalize-by-median.py -k 20 -C 20 -N 4 -x 4e9 -p -s Lall\_B\_filter\_norm\_hash.kh Lall\_B\_filter.fastq*

# Trim off any k-mer that are abundant in high-coverage reads

*filter-abund.py Lall\_B\_filter\_norm\_hash.kh Lall\_B\_filter.fastq.keep*

*extract-paired-reads.py Lall\_B\_filter.fastq.keep.abundfilt*

# Normalize down to C=10

*normalize-by-median.py -k 20 -C 10 -N 4 -x 4e9 -p Lall\_B\_filter.fastq.keep.abundfilt.pe*

*extract-paired-reads.py Lall\_B\_filter.fastq.keep.abundfilt.pe.keep*

*split-paired-reads.py Lall\_B\_filter.fastq.keep.abundfilt.pe.keep.pe*

**ALLPATH-LG assembly**

# Prepare data for ALLPATH

*PrepareAllPathsInputs.pl DATA\_DIR=~/Allpath\_assemble/data PLOIDY=2 IN\_GROUPS\_CSV=in\_groups.csv IN\_LIBS\_CSV=in\_libs.csv GENOME\_SIZE=400000000 OVERWRITE=True | tee prepare2.out*

# Run ALLPATH-LG using HAPLOIDIFY and OVERWRITE parameters

*RunAllPathsLG \*

*PRE=$PWD\*

*REFERENCE\_NAME=norm\*

*DATA\_SUBDIR=data\*

*RUN=run\*

*SUBDIR=FINAL\*

*OVERWRITE=True\*

*TARGETS=standard\*

*HAPLOIDIFY=True | tee -a assemble2.out*

**Genome correction (correct ambiguous bases)**

# Ambiguous and erroneous bases were identified through mapping of filtered, pre-normalized paired-end reads against the assembled genome using Bowtie2 v2.1.0 and subsequently corrected using SAMtools v1.2

# Map the paired end reads to assembled genome

*bowtie2-build -f Final.fasta Final\_index*

*bowtie2 -q --fr --phred33 --end-to-end --very-sensitive -I 0 -X 500 -p 40 -x Final\_index -S Final.pe.sam -1 paired\_R1.fastq -2 paired\_R2.fq*

# Transfer sam to bam then sort.

*samtools view -bS -o Final.bam Final.pe.sam*

*samtools sort Final.bam Final.sorted*

# Retain the unique mapping results

*samtools rmdup -S Final.sorted.bam Final.sorted.uniq.bam*

*samtools mpileup -f Final.fasta Final.sorted.uniq.bam > alllib.variants.vcf*

awk '{if ($3 !~ /[ACGT]/) print}' alllib.variants.vcf > alllib.variants.flt.vcf

# Correct the ambiguous and erroneous bases.

*perl* [*correct.ambiguous.pl*](http://correct.ambiguous.pl/) *alllib.variants.flt.vcf Final.fasta Final.corrected.fasta*

**Gap-filling**

## Close gaps with Gapcloser v1.12

*GapCloser –a Final.corrected.fasta –b library.txt –o Gapcloser.fasta –l 125 –p 31*

**Genome estimation**

# Estimate basic genome information, including insert size and mapping rate

# Map mate pair reads to evaluate the insert size

*bowtie2-build -f Final\_allpath.fasta Final\_index*

*bowtie2 -q --fr --phred33 --end-to-end --very-sensitive -I 0 -X 500 -p 40 -x Final\_index -S Final.pe.sam -1* *mate\_R1.fastq -2 mate\_R2.fq*

# Calculate statistics for mapping information

*samtools view -bS -o Final.bam Final.pe.sam*

*samtools sort Final.bam Final.sorted*

*samtools stats Final.sorted.bam*

# Obtain basic genome information

*perl data\_statatistic.pl –i Final.fasta –l 100 –o Final.stat*

*perl scaffold\_gap.pl –I Final.fasta –o scaffold.stat*

**Removing bacterial and virus contamination using BLAST**

# Identify and remove scaffolds originating from bacterial or virus contaminants

*blastn –db Bac\_vir –query Final.fasta -seg yes –out blast.out -evalue 1e-5 -outfmt 7 -num\_threads 40*

**Genome validation**

# Validate the genome using CEGMA v2.5 with core genes

*cegma -g Final.fasta -o species -T 25 -p core.fa*

# Evaluate the ratio of putative artificially duplicate contigs

*blastn -query CONTIG.fa -db CONTIG\_DB -out OUT.blast.tbl -num\_threads 64 -outfmt '6 qseqid sseqid pident length mismatch gapopen qstart qend qlen sstart send slen bitscore evalue' -max\_target\_seqs 2*

**2. Genome annotation**

**Repeat annotation**

# Identify repeat boundaries and construct the consensus models using RepeatModeler v1.0.8

*BuildDatabase -name Final\_builddata Final.fasta -engine ncbi*

*RepeatModeler -database Final\_builddata -engine ncbi -pa 15 -recoverDir Final\_fail*

# Identify and classify different repetitive elements

*RepeatMasker -e ncbi -pa 12 -s -lib consensi.fa.classified -gccalc -noisy -a -inv -gff Final\_gapcloser.fa*

*perl juke-distance\_distribiution.pl –i repeat\_information –o repeat\_JD.tsv –t genome\_size*

**Assembly of transcriptome**

# Remove adapter, low quality sequences ((< Q20 and < 20 bp) and duplicates reads using Trimmomatic v0.32

*java -jar trimmomatic.jar PE -threads 40 -trimlog trimmoatic.log RNA \_final\_R1.fastq RNA\_filter\_R1.fastq RNA\_unpaired\_R1.fastq RNA\_final\_R2.fastq RNA\_filter\_R2.fastq RNA\_unpaired\_R2.fastq ILLUMINACLIP:TruSeq3-SE.fa:2:20:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:20 MINLEN:20*

# Map RNA-seq reads to assembly using Tophat v2.0.9

*bowtie2-build Final.fasta FINAL\_bowtie\_index*

*tophat --microexon-search --library-type fr-unstranded -o ./tophat\_out FINAL\_bowtie\_index RNA\_filter\_1.fastq RNA\_filter\_2.fastq*

# Assemble using Trinity v2.02

*~/software/trinityrnaseq-2.0.2/Trinity --left RNA\_filter\_1.fastq --right RNA\_filter\_2.fastq --genome\_guided\_bam tophat\_out/accepted\_hits.bam --genome\_guided\_max\_intron 2000 --max\_memory 100G --seqType fq --CPU 18*

*TransDecoder -t trinity\_out\_dir/Trinity-GG.fasta -m 50 --CPU 10*

# Assemble using Cufflinks v2.1.1

*cufflinks -p 15 -o RNA\_cufflinks -L RNA tophat\_out/accepted\_hits.bam*

**Genome annotation**

# Prediction with Augustus v3.0.3

*perl selectComplete.pl Trinity-GG.fasta.transdecoder.geno,m.gff3 Trinity-GG.fasta.transdecoder.gff3> selected\_models.gff3*

*perl autoAug.pl -g Final\_genome.fa.masked --species=Speices --cdna=Trinity-GG.rename.fasta --pasa --useexisting -v --noutr --singleCPU --optrounds=5*

# Prediction with SNAP v2013-02-16

*perl gff2snap\_train.pl -i trainingSetComplete.gff -o PASA\_SNAP.training -g Final.fasta.masked -f 500 -type cds*

*fathom genome.ann genome.dna -gene-stats >gene.stat*

*fathom genome.ann genome.dna -validate >stat.validate*

*fathom genome.ann genome.dna -categorize 1000*

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

*hmm-assembler.pl Amplex\_snap\_trained params >SNAP\_train.hmm*

*perl auto\_SNAP.pl -i Final\_genome.fa.masked -hmm SNAP\_train.hmm -o Species\_snap\_out -gff -aa -tx -p 15*

# Prediction with GlimmerHMM v3.02

*perl gff2glimmer\_train.pl -i ../autoAug/trainingSet/training/trainingSetComplete.gff -checkorf -o glimmer.train -nosingle -genome Final\_genome.masked*

*trainGlimmerHMM Final\_genome.masked glimmer.train.mmtrain -d glimmer*

*perl auto\_glimmerhmm.pl -g Final\_genome.masked -d glimmerM -w GlimmerM\_2/glimmer\_ouput -o glimmer\_result -t 20 -gff -f*

# Prediction with Spaln v2.14

*perl auto\_spaln.pl -p prot2genome -q uniprot\_sprot\_withclosespecies.dat.list.fasta -d Final.fasta.masked -o spaln -t 16 -w ./ -b 1000 -- -Q7 -O0 -M10 –LS*

# Combine all the annotated files with EVidenceModeler v1.1.1

*partition\_EVM\_inputs.pl --genome Final.fasta.masked --gene-predictions gene\_prediction.gff3 --protein\_alignments protein\_alignments.gff3 --transcript\_alignments transcript\_alignments.gff3 --overlapSize 10000 --partition\_listing partitions\_list.out*

*write\_EVM\_commands.pl --genome Final.fasta.masked --weights $PWD/weights.txt --gene\_predictions gene\_prediction.gff3 --protein\_alignments protein\_alignments.gff3 --transcript\_alignments transcript\_alignments.gff3 --output\_file\_name evm.out --partitions partitions\_list.out > commands.list*

*execute\_EVM\_commands.pl evm.comandsal |tee runl.log*

*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 Final.fasta.masked*

*find ./ -name "evm.out.gff3" -exec cat {} \; > ../EVM\_out.gff3*

**tRNA and rRNA prediction**

# Predict rRNA based rfam v12.0

*perl rfam\_scan.pl\_v\_1.0.4 -bt 1e-5 -o Final.rfam -blastdb ~/database/rRNAScan/rfam/12.0/Rfam.fasta ~/database/rRNAScan/rfam/12.0/Rfam.cm Final.fasta*

#Predict tRNAusing tRNAscan-SE v1.3.1

*tRNAscan-SE -G -o.tRNA.out –f tRNA.structure -m tRNA.summary Final.fasta*

**Functional annotation using InterProScan v5.9**

*~/software/interproscan-5.9-50.0/interproscan.sh -i Final.fasta -appl CDD,COILS,Gene3D,HAMAP,MobiDBLite,PANTHER,Pfam,PIRSF,PRINTS,ProDom,PROSITEPATTERNS,PROSITEPROFILES,SFLD,SMART,SUPERFAMILY,TIGRFAM -goterms -pa -o butterfly\_fish.intepro.out -t p*

**3. Gene family comparisons and phylogenetic analysis**

**Ortholog identification**

# Ortholog identification using OrthoMCL v2.0.9

*orthomclAdjustFasta amp Amplex.evm.pep 1*

*orthomclAdjustFasta dis Dis.evm.pep 1*

*orthomclAdjustFasta jgi Nem.evm.pep 1*

*orthomclAdjustFasta aip Apa.evm.pep 1*

*orthomclAdjustFasta adi Adi.evm.pep 1*

*orthomclAdjustFasta gnl Hma.evm.pep 1*

*mv aip.fasta amp.fasta dis.fasta jgi.fasta spi.fasta acr.fasta my\_orthomcl*

*orthomclFilterFasta my\_othomcl 10 20 good bad*

*makeblastdb -in orthomcl.fasta -dbtype prot -title orthomcl -parse\_seqids -out orthomcl -logfile orthomcl.log*

*blastp -db orthomcl -query goodProteins.fasta -seg yes -out orthomcl.blastout -evalue 1e-5 -outfmt 7 -num\_threads 24*

*orthomclBlastParser blastresult compliantFasta > similarSequences.txt*

*orthomclLoadBlast orthomcl.config.template similarSequences.txt*

*orthomclPairs orthomcl.config.template orthomcl\_pairs.log cleanup=no*

*orthomclDumpPairsFiles orthomcl.config.template*

*mcl mclInput --abc -I 1.5 -o mclOutput*

*orthomclMclToGroups led 1 < mclOutput > groups.txt*

**Phylogenetic analysis**

# Annotate each group with pfam and blast function, and then select single copy orthologs

*perl group\_family\_into\_cafe\_format\_blastannotat.pl –i mclOutput.grp –f all\_blast.tsv –o mclOutput\_blast.grp*

*perl group\_family\_into\_cafe\_format\_pfam.pl –i mclOutput.grp –f all\_pfam –o mclOutput.grp.cafe\_all\_pfam.tsv*

*perl -ne '{chomp; my @array=split/\t/,$\_; my $num=0; foreach my $i (2..$#array){next if ($i==7);if($array[$i]==1){$num++}} print "$\_\n" if ($num==6)}' mclOutput.grp.cafe\_all\_pfam.tsv |grep "NA" -v | grep ";" -v >single\_copy\_gene.tsv*

*perl select\_sinlge\_copy\_group.pl -i single\_copy\_gene\_without\_spi.tsv -q mclOutput.grp -o single\_copy\_gene.gr*

# Alignment of single orthologs using muscle v3.8.31

*muscle –in each\_ortholog.fasta –out each\_ortholog\_muscle.fasta*

#Use Gblock v0.91 to trim poorly-aligned regions and then concatenate all the orthologs

*Gblock each\_ortholog\_muscle.fasta –t=p –b4=5*

*perl combine\_glock\_file.pl -i gblock -o combine\_glock.fa*

**Choosing best protein substitution model**

# Testing best model using Prottest v2.4

*java –jar prottest-3.4.jar -i combine\_glock.fa -t1 -all-matrices -all-distributions -F -S 2 -o prott.out -threads 30 -tc 0.5*

**Constructing the phylogeny**

# Maximum-likelihood tree using RAxMLv8.1.22

*raxmlHPC-PTHREADS-SSE3 -m PROTGAMMAILGF -p 12345 -s combine\_glock.fa -T 30 -n final\_best*

*raxmlHPC-PTHREADS-SSE3 -m PROTGAMMAILGF -p 12345 –b -# 1000 -s combine\_glock.fa -T 30 -n final\_bootstrap*

*raxmlHPC-PTHREADS-SSE3 -f b -m PROTGAMMAILGF -p 12345 –t RAxML\_bestTree.final\_best -z RAxML\_bootstrap.final\_bootstrap -n Bootstrap\_best*

# Bayesian inference using Mrbayes v3.2.6

*Execute combine\_out.nex*

*prset aamodelpr=fixed(lg)*

*prset statefreqpr=fixed(empirical); lset rates=invgamma;*

*mcmc ngen=5000000 samplefreq=500 nruns=4 printfreq=500 nchains=4 temp=0.2 savebrlens=yes burnin=200 starttree=random;*

**4. Estimation of duplication rates**

**Duplication rate measurement**

# Duplication rates estimated using MUMmer v3.23

*nucmer --mum --delta -l 100 --coords --optimize -p Final\_itself Final.fasta Final.fasta*

*delta-filter -i 90 -l 30 -1 Final\_itself.delta >Final\_itself.delta.filter*

*show-coords -I 90 -l -L 20 -o -c -T -H -r Final\_itself.delta.filter >Final\_itself.delta.filter.coords*

*perl -ne '{chomp;s/\r//; if (/>(\S+)/){$id=$1;}else{$string{$id}.=$\_;}} END {my $start=1; foreach my $i(sort keys %string){ my $length=length $string{$i}; my $end=$start +$length-1; print "$i\t$start\t$end\t$length\n";$start +=$length;}}' Final.fasta >Final.rearrangement.tsv*

*perl HCE\_extraction.pl -i Final.rearrangement.tsv -q Final\_L3K\_I80.coord -o Final\_L3K\_I80.coord\_partial.tsv*