Supplementary data for ‘Population genetic structure and gene expression plasticity of the deep-sea vent and seep squat lobster *Shinkaia crosnieri’*

#####################

# commands used in this study

#####################

# filter raw reads with Trimmomatic

java -jar trimmomatic-0.36.jar PE rawdata\_1.fq.gz rawdata\_2.fq.gz data\_1p.fq data\_1up.fq data\_2p.fq data\_2up.fq ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 LEADING:10 TRAILING:10 SLIDINGWINDOW:4:15 MINLEN:25

# assemble reads with Trinity v.2.8.3

Trinity --seqType fq --max\_memory 100G --left data\_1p.fq --right data\_2p.fq --min\_contig\_length 300 --min\_kmer\_cov 2

# choose the longest isoform of each gene with a custom Python script

# -\*- coding: UTF-8 -\*-

f1 = open(r'./Trinity.fasta','r')

f2 = open(r'./list.txt','r+')

f3 = open(r'./temp\_list.txt','r+')

f4 = open(r'./list\_clean.txt','r+')

th = {}

lines = f1.readlines()

for line in lines:

if '>' in line:

a = str(line.split()[0:2])

f2.write(a + '\n')

lines = f2.readlines()

for line in lines[1:]:

b = line.split("'")[1]

c = line.split("'")[3]

th[b]=c

a = []

nu = []

c = []

th1 = sorted(th.iteritems(), key=lambda d:d[0])

for i in range(0, len(th1)):

b = str(th1[i])

a.append(b.split('i')[0])

tmp = b.split("'")[3]

nu.append(tmp.split('=')[1])

for i in a:

c = []

if a.count(i)>1:

m = a.index(i)

p = a.count(i)

for i in range(m,m+p):

c.append(int(nu[i]))

c.sort()

mark = nu.index(str(c[-1]))

s = str(th1[mark]).split("'")[1]

f3.write(s+'\n')

else:

n = a.index(i)

s = str(th1[n]).split("'")[1]

f3.write(s+'\n')

th = []

for line in open(r'./temp\_list.txt','r+'):

th.append(line)

l = []

[l.append(i) for i in th if not i in l]

F4 = open(r'./list\_clean.txt','r+')

for i in l:

f4.write(i)

f1.close()

f2.close()

f3.close()

f4.close()

# -\*- coding: utf-8 -\*-

#define the document names

cleanQuant = "list\_clean.txt"

orginalData = "Trinity.fasta"

outputfilename = "return\_cleanreads.fasta"

#Read all sequence names from clean\_quant

def get\_clean\_quant():

result = []

with open(cleanQuant) as fp:

while True:

line = fp.readline().strip()

if not line:

break

result.append(line)

#print(result[-1])

return result

#output new ducuments

def output\_file(total\_result):

with open(orginalData) as fp:

ofp = open(outputfilename,'w')

saveState = False #check one data is analyzing

startSave = False #if the name in the clean data then start save

while True:

line = fp.readline()

if not line:

break

if line.startswith('>'):

#check the name if in the clean data

name = line.split(" ")[0]

if name in total\_result:

ofp.write(line)

startSave = True

else:

startSave = False

else:

if startSave:

ofp.write(line)

ofp.close()

if \_\_name\_\_=="\_\_main\_\_":

total\_result = get\_clean\_quant()

output\_file(total\_result)

print("Done!!!")

# remove redundant unigenes with cd-hit-est

cd-hit-est -i return\_cleanreads.fasta -o return\_cleanreads\_cdhit95.fasta -c 0.95 -M 300000

# predict candidate ORF with Transdecoder

TransDecoder.LongOrfs -t target\_transcripts.fasta

TransDecoder.Predict -t target\_transcripts.fasta -- single\_best\_only

# check the transcriptome completeness with BUSCO

python run\_BUSCO.py -i cds.fasta -l /home/share/busco-master/db/metazoa\_odb9 -o trinity\_busco -m tran

# evaluate with Perl script

assemblathon\_stats.pl cds.fasta

# annotation

blastp -num\_threads 96 -query cds.fasta.transdecoder.pep -db nr\_database -out Blastp.m5 -outfmt 5 -evalue 1e-5 -word\_size 3 -num\_alignments 20 -max\_hsps 20

blastx -query cds.fasta -db uniprot\_sprot.fasta -outfmt 5 -out cds\_swissport.xml -evalue 1e-5

# quantify abundances of the assembled unigenes with Kallisto

kallisto index cds.fasta -I reference -k 31

kallisto quant -i reference -o ./output data\_1p.fq data\_2p.fq. –b 100

# normalize with edgeR

trinityrnaseq-Trinity-v2.8.3/util/abundance\_estimates\_to\_matrix.pl --est\_method kallisto --gene\_trans\_map none --cross\_sample\_norm none scs1\_abundance.tsv scs2\_abundance.tsv scs3\_abundance.tsv scs4\_abundance.tsv scs5\_abundance.tsv scs6\_abundance.tsv scs7\_abundance.tsv scs8\_abundance.tsv scs9\_abundance.tsv scs10\_abundance.tsv scs11\_abundance.tsv scs12\_abundance.tsv scs13\_abundance.tsv scs14\_abundance.tsv scs15\_abundance.tsv scs16\_abundance.tsv scs17\_abundance.tsv scs18\_abundance.tsv scs19\_abundance.tsv scs20\_abundance.tsv ot1\_abundance.tsv ot2\_abundance.tsv ot3\_abundance.tsv ot4\_abundance.tsv ot5\_abundance.tsv ot6\_abundance.tsv ot7\_abundance.tsv ot8\_abundance.tsv ot9\_abundance.tsv ot10\_abundance.tsv ot11\_abundance.tsv ot12\_abundance.tsv ot13\_abundance.tsv ot14\_abundance.tsv ot15\_abundance.tsv ot16\_abundance.tsv ot17\_abundance.tsv ot18\_abundance.tsv ot19\_abundance.tsv ot20\_abundance.tsv ot21\_abundance.tsv ot22\_abundance.tsv ot23\_abundance.tsv ot24\_abundance.tsv

trinityrnaseq-Trinity-v2.8.3/util/run\_DE\_analysis.pl --matrix kallisto.isoform.counts.matrix --method edgeR --samples\_file sample\_file.txt

# detect orthologous with proteinortho6

perl proteinortho6.pl -project=sl\_tran -p=blastn+ -cpus=2 -graph -clean scs1.Trinity.fasta scs2.Trinity.fasta scs3.Trinity.fasta scs4.Trinity.fasta scs5.Trinity.fasta scs6.Trinity.fasta scs7.Trinity.fasta scs8.Trinity.fasta scs9.Trinity.fasta scs10.Trinity.fasta scs11.Trinity.fasta scs12.Trinity.fasta scs13.Trinity.fasta scs14.Trinity.fasta scs15.Trinity.fasta scs16.Trinity.fasta scs17.Trinity.fasta scs18.Trinity.fasta scs19.Trinity.fasta scs20.Trinity.fasta ot1.Trinity.fasta ot2.Trinity.fasta ot3.Trinity.fasta ot4.Trinity.fasta ot5.Trinity.fasta ot6.Trinity.fasta ot7.Trinity.fasta ot8.Trinity.fasta ot9.Trinity.fasta ot10.Trinity.fasta ot11.Trinity.fasta ot12.Trinity.fasta ot13.Trinity.fasta ot14.Trinity.fasta ot15.Trinity.fasta ot16.Trinity.fasta ot17.Trinity.fasta ot18.Trinity.fasta ot19.Trinity.fasta ot20.Trinity.fasta ot21.Trinity.fasta ot22.Trinity.fasta ot23.Trinity.fasta ot24.Trinity.fasta

# call SNP

java -jar trimmomatic-0.36.jar PE scs5\_1.fq.gz scs5\_2.fq.gz scs5\_1p.fq scs5\_1up.fq scs5\_2p.fq scs5\_2up.fq ILLUMINACLIP:TruSeq3-PE-2.fa:2:30:10 LEADING:10 TRAILING:10 SLIDINGWINDOW:4:15 MINLEN:2

bowtie2 -D 20 -R 2 -N 1 -L 18 -i S,1,0.50 --maxins 1200 -x cds\_list -1 scs5\_1p.fq -2 scs5\_2p.fq 1>scs5\_map.sam 2>scs5\_mapping.err

samtools view -bS scs5\_map.sam >scs5\_map.bam

samtools sort scs5\_map.bam -m 5G >scs5\_map.sorted.bam

java -jar picard.jar MarkDuplicates I=scs5\_map.sorted.bam O=scs5\_picard.sorted.bam M=metrics.txt MAX\_FILE\_HANDLES=4000

samtools mpileup -q 10 -Q 20 -g -u -I -f cds\_list.fasta scs5\_picard.sorted.bam| bcftools call -mv -Ov >./scs5.vcf

bgzip -c scs5.vcf > scs5.vcf.gz

bcftools index scs5.vcf.gz

bcftools merge ot24.vcf.gz ot23.vcf.gz ot22.vcf.gz ot21.vcf.gz ot20.vcf.gz ot19.vcf.gz ot18.vcf.gz ot17.vcf.gz ot16.vcf.gz ot15.vcf.gz ot14.vcf.gz ot13.vcf.gz ot12.vcf.gz ot11.vcf.gz ot10.vcf.gz ot9.vcf.gz ot8.vcf.gz ot7.vcf.gz ot6.vcf.gz ot5.vcf.gz ot4.vcf.gz ot3.vcf.gz ot2.vcf.gz ot1.vcf.gz scs1.vcf.gz scs2.vcf.gz scs3.vcf.gz scs4.vcf.gz scs5.vcf.gz scs6.vcf.gz scs7.vcf.gz scs8.vcf.gz scs9.vcf.gz scs10.vcf.gz scs11.vcf.gz scs12.vcf.gz scs13.vcf.gz scs14.vcf.gz scs15.vcf.gz scs16.vcf.gz scs17.vcf.gz scs18.vcf.gz scs19.vcf.gz scs20.vcf.gz --force-samples > all.vcf

# filter SNP with VCFtool

vcftools --vcf all.vcf --recode --out out.vcf --min-alleles 2 --max-alleles 2 --maf 0.02 --remove-indels --hwe 0.01 –max-missing 0.05 --minDP 3 --maxDP 120

# detect outlier SNP with the R package pcadapt

> library(pcadapt)

> library(qvalue)

> filename <- read.pcadapt("out.vcf", type = "vcf")

> x <- pcadapt(input = filename, K = 20)

> qval <- qvalue(x$pvalues)$qvalues

> alpha <- 0.1

> outliers <- which(qval < alpha)

# population genetics analysis

plink --vcf file.vcf –recode --out file --double-id --allow-extra-chr

admixture --cv out\_plink.bed 2 | tee log2.out

admixture out\_plink.bed 2 -j20

> library(gdsfmt)

> library(SNPRelate)

> vcf.fn <-"out.vcf"

> snpgdsVCF2GDS(vcf.fn, "out.gds", method ="biallelic.only")

> genofile <- snpgdsOpen(snpgdsExampleFileName())

> RV <- snpgdsPCA(genofile, num.thread=5, autosome.only=FALSE)

> pc.percent <- RV$varprop\*100

> sample.id <- read.gdsn(index.gdsn(genofile, "sample.id"))

> pop\_code <- scan("pop\_code.txt", what=character())

> tab <- data.frame(sample.id = RV$sample.id, pop=factor(pop\_code)[match(RV$sample.id, sample.id)], EV1=RV$eigenvect[,1],EV2=RV$eigenvect[,2], stringsAsFactors = FALSE)

> plot(tab$EV2, tab$EV1, col=as.integer(tab$pop), xlab="eigenvector 1", ylab="eigenvector 2")

> library(diveRsity)

> data(Test\_data)

> div\_res <- divMigrate(infile = data, boots = 1000, stat = "d\_jost")