Using metatranscriptomics in estimating diversity and composition of zooplankton communities

Mark Louie D. Lopez, Ya-Ying Lin, Mitsuhide Sato, Chih-hao Hsieh, Fuh-Kwo Shiah, and Ryuji J. Machida

Biodiversity Program, Taiwan International Graduate Program, Academia Sinica, Nankang District, Taipei 11529, Taiwan, Republic of China.

Department of Life Science, National Taiwan Normal University, Wenshan District, Taipei 11677, Taiwan, Republic of China.

Biodiversity Research Center, Academia Sinica, Nankang District, Taipei 11529, Taiwan, Republic of China

Department of Environment and Fisheries Resources, Nagasaki University, Nagasaki City 852-8521, Nagasaki Prefecture, Japan.

Institute of Oceanography, National Taiwan University, Da'an District, Taipei 10617, Taiwan, Republic of China.

Environmental Change Research Center, Academia Sinica, Nankang District, Taipei 11529, Taiwan, Republic of China

Abstract

DNA metabarcoding is a rapid, high-resolution tool used for biomonitoring complex zooplankton communities. However, diversity estimates derived with this approach can be biased by the co-detection of sequences from environmental DNA and non-target taxa, nuclear mitochondrial (NUMT) pseudogene contamination, and PCR-inherent biases. To avoid these methodological uncertainties, we tested the use of metatranscriptomics as an alternative approach for characterizing zooplankton communities. Specifically, we compared metatranscriptomics with PCR-based methods using genomic (gDNA) and complementary DNA (cDNA) amplicons, using morphological data for estimating species diversity and composition for both mock communities and field-collected samples. Mock community analyses showed that the use of gDNA mitochondrial cytochrome c oxidase I amplicons inflates species richness due to environmental and non-target species sequence contamination. Significantly more amplicon sequence variants, nucleotide diversity, and indels were observed with gDNA amplicons than with cDNA, indicating the presence of putative NUMT pseudogenes. Moreover, PCR-based methods failed to detect the most abundant species in mock communities due to priming site mismatch. Overall, metatranscriptomics provided estimates of species richness and composition that closely resembled those derived from morphological data. The use of metatranscriptomics was further tested using field-collected samples, with the results showing consistent species diversity estimates among biological and technical replicates. Additionally, temporal zooplankton species composition changes could be monitored using different mitochondrial markers. These findings demonstrate the advantages of metatranscriptomics as an effective tool for monitoring diversity in zooplankton research.

Codes used for the Bioinformatics Processing

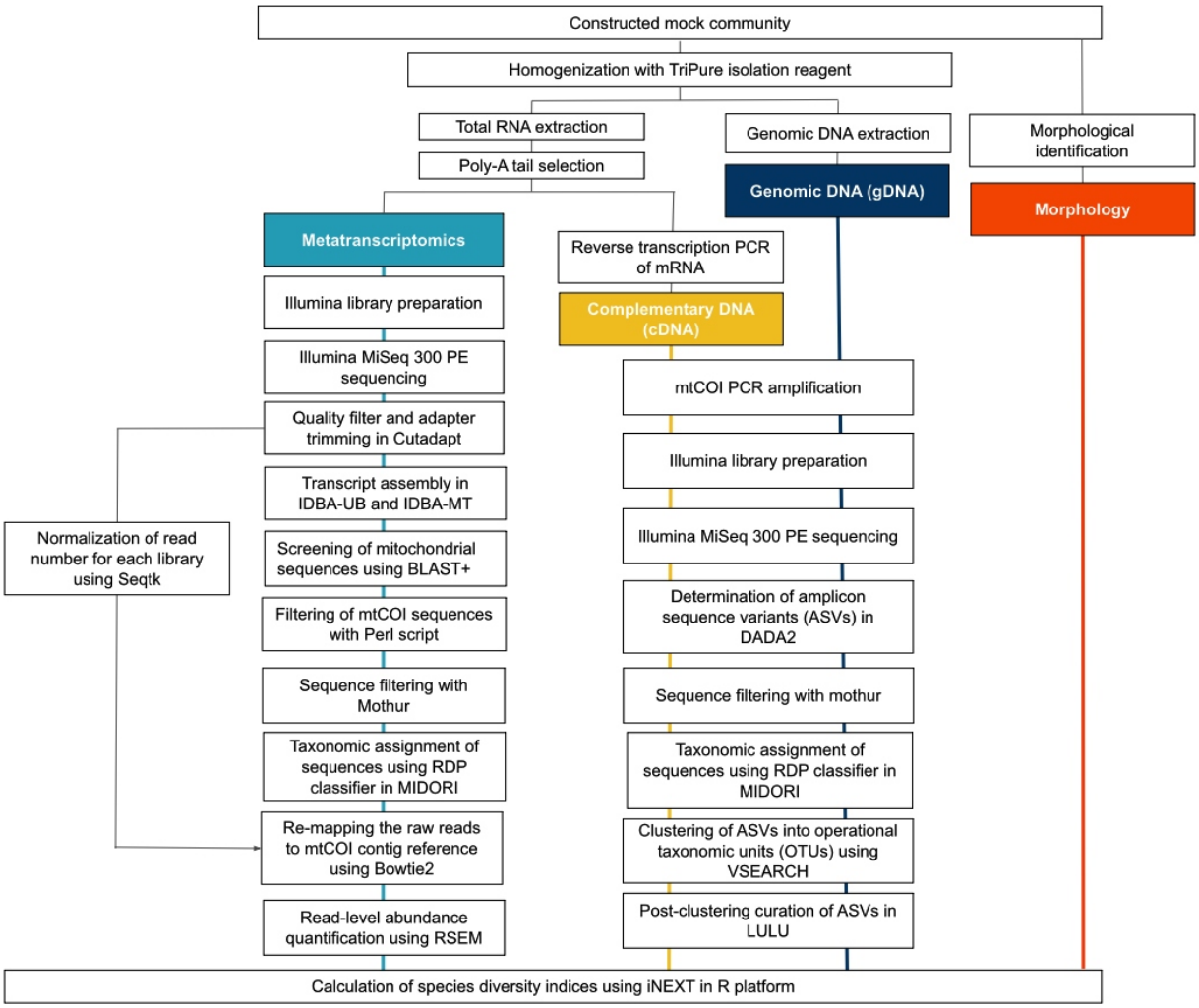

Figure 1. General workflow of the current study

Genomic and Complimentary DNA Amplicons Processing in Linux Terminal

Demultiplexing, Primer removal, Quality filtering, and Normalizing number of reads

```
# Adapter demultiplexing using Cutadapt
cutadapt -g file:Adapters_file.fasta --minimum-length 100 --untrimmed -o untrimmed_F.fastq.gz -o {name}.R1.fq 'path to R1.fastq.gz'
cutadapt -g file:Adapters_file.fasta --minimum-length 100 --untrimmed -o untrimmed_R.fastq.gz -o {name}.R2.fq 'path to R2.fastq.gz'

#Primer removal and quality filtering in Cutadapt
cutadapt -pair-filter=any -q 15 --cut 26 --minimum-length 100 -o trimmed1_R1.fastq -p trimmed2_R2.fastq 'path to R1.fastq.gz' 'path to R2.f

#Normalize read numbers by random subsampling using Seqtk
seqtk sample -s100 trimmed_R1.fastq 191000 > sub_R1.fastq
seqtk sample -s100 trimmed_R2.fastq 191000 > sub_R2.fastq
```

DADA2 Processing in R platform

```
#Install packages
install.packages("devtools")
install.packages("usethis")
install.packages("Rcpp")

#Load packages
library('usethis')
library("devtools") devtools::install_github("benjjneb/dada2", ref="v1.16")
library(dada2); packageVersion("dada2")
library('Rcpp')

#Set path to the files
path = setwd("path to file folder")

#Forward and reverse fastq filenames have format: SAMPLENAME_R1_001.fastq and SAMPLENAME_R2_001.fastq
fnFs <- sort(list.files(path, pattern="R1.fastq", full.names = TRUE))
fnRs <- sort(list.files(path, pattern="R2.fastq", full.names = TRUE))

#Extract sample names, assuming filenames have format: SAMPLENAME_XXX.fastq
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)

#Visualizing the quality profiles
plotQualityProfile(fnFs[1:2])
plotQualityProfile(fnRs[1:2])

# Place filtered files in filtered/ subdirectory
filtFs <- file.path(path, "filtered", paste0(sample.names, "_F_filt.fastq.gz"))
filtRs <- file.path(path, "filtered", paste0(sample.names, "_R_filt.fastq.gz"))
names(filtFs) <- sample.names
names(filtRs) <- sample.names

#Check for duplicates in filenames
any(duplicated(c(fnFs, fnRs)))
any(duplicated(c(filtFs, filtRs)))

#Filtering parameters depends on the results of the quality checking (optional).
out = filterAndTrim(fnFs, filtFs, fnRs, filtRs, truncLen=c(250,200), maxN=0, maxEE=c(2,2), truncQ=2, rm.phix=TRUE, compress=TRUE, multithre

head(out)

#Learn error rates
errF = learnErrors(filtFs, multithread=TRUE)
errR = learnErrors(filtRs, multithread=TRUE)

#Visualize error rates
plotErrors(errF, nominalQ=TRUE)

#Core sample inference algorithm to the filtered and trimmed sequence data
dadaFs = dada(filtFs, err=errF, multithread=TRUE)
dadaRs = dada(filtRs, err=errR, multithread=TRUE)

#Inspecting the returned dada-class object
dadaFs[[1]]
dadaRs[[1]]

#Merge pair ends
mergers = mergePairs(dadaFs, filtFs, dadaRs, filtRs, verbose=TRUE)

#Construct an amplicon sequence variant table (ASV) table
seqtab = makeSequenceTable(mergers)
dim(seqtab)

#Inspect distribution of sequence lengths
table(nchar(getSequences(seqtab)))

#Remove too short or too long sequences. Cutting band in-silico (optional)
seqtab2 = seqtab[,nchar(colnames(seqtab)) %in% 250:256]

#Remove chimeras
seqtab.nochim = removeBimeraDenovo(seqtab, method="consensus", multithread=TRUE, verbose=TRUE)
dim(seqtab.nochim)

sum(seqtab.nochim)/sum(seqtab)
```

Extract unique ASV sequences from the chimera table produced by DADA2

```
#Extraction of fasta from chimera free table
install.packages('openssl')
library(openssl)

#Where 'seqtab' is your chimera-free merged sequence tables
saveRDS(seqtab, "seqtab.nochim")

#We start by renaming the sequence variants using SHA1

seqtab_sha1 = seqtab
colnames(seqtab_sha1) = openssl::sha1(colnames(seqtab_sha1))
saveRDS(seqtab_sha1, "seqtab_SHA1.rds")
uniquesToFasta(seqtab, fout = "uniques.fasta") # we relabel this later using VSEARCH to also use SHA1 names

#Function modified from LULU manuscript to turn a sequence table into FASTA file with labels indicating sample & abundance in the header.
extrSamDADA2 <- function(my_table) {
  out_path <- file.path(getwd(), "DADA2_extracted_samples")
  if(!file_test("-d", out_path)) dir.create(out_path)
  for (sampleX in seq(1:dim(my_table)[1])){
    sinkname <- file.path(out_path, paste0(rownames(my_table)[sampleX],".fas"))
    {
      sink(sinkname)
      for (seqX in seq(1:dim(my_table)[2])) {
        if (my_table[sampleX,seqX] > 0) {
          header <- paste0(">",openssl::sha1(colnames(my_table)[seqX]),";size=",
                           my_table[sampleX,seqX],";barcodelabel=",rownames(my_table)[sampleX],";","\n")
          cat(header)
          seqq <- paste0(colnames(my_table)[seqX],"\n")
          cat(seqq)
        }
      }
      sink()
    }
  }
}
extrSamDADA2(seqtab) #This will start writing FASTA files to a subfolder (DADA2_extracted_samples).

#Proceed to terminal and concatenate the generated fasta files
cat *.fa > concatenated.fasta
```

Filtering the amplicon sequence variants (ASVs) to extract target taxa using Mothur in Linux terminal

```
#Set directory in Mothur
set.dir(input= path to the file folder)

#Taxonomic assignment of ASV sequences
classify.seqs(inputdir= path to the file folder, fasta=concatenated.fasta, reference=MIDORI_LONGEST_GB238_mtGene_MOTHUR.fasta, taxonomy=MID

#Extracting sequences of target species
get.lineage(inputdir= path to the file folder, taxonomy=input.MIDORI_LONGEST_GB238_mtGene_MOTHUR.wang.taxonomy, taxon=Eukaryota_2759(100);A
```

Clustering of amplicon sequence variants (ASVs) from DADA unto OTUs using VSEARCH in Linux terminal and construction of the input file for LULU post clustering curation in R platform

```
#Clustering of ASVs into OTUs with 94% similarity index
vsearch --cluster_fast concatenated.fasta --strand both -id 0.94 --uc vsearchcentroid.uc --centroids vsearchcentroid.fasta --otutabout vsea

#Construction of local blast database with the ASVs centroid fasta file
makeblastdb -in vsearchcentroid.fasta -parse_seqids -dbtype nucl

#Running blast using vsearchcentroid.fas against constructed local database
blastn -db vsearchcentroid.fasta -outfmt '6 qseqid sseqid pident' -out mixed.txt -qcov_hsp_perc 80 -perc_identity 84 -query concatenated.fa

#Produce matchlist file for LULU
vsearch --usearch_global vsearchcentroid.fasta --db vsearchcentroid.fasta --self --id .84 --iddef 1 --userout match_list.txt -userfields qu
```

Taxonomic assignment of the filtered mitochondrial transcript using RDP classifier function of the MIDORI server

```
http://reference-midori.info/server.php
```

Post-clustering sequence curation using LULU in R platform

```
#Install packages
install_github("tobiasgf/lulu")
install.packages('dplyr')

#Load the packages
library('dplyr')
library(devtools)
library('lulu')

#Read the prepared input files
setwd("path to the file folder")
otutab <- read.csv("vsearchOTU.txt",sep='\t',header=TRUE,as.is=TRUE, row.names = 1)
matchlist <- read.table("match_list.txt", header=FALSE,as.is=TRUE, stringsAsFactors=FALSE)

#Run curated data in LULU
curated_result = lulu(otutab, matchlist)

#Access the curated ASV table
curated_result$curated_table

#Access the original ASV table
curated_result$original_table

#Check the number of ASVs retained after sequence curation
curated_result$curated_count

#Save the ASV table generated by LULU
lulu = curated_result$curated_table
write.table(lulu, file = "lulu_mixed.txt", sep = "\t", row.names = TRUE, col.names = NA)
```

Metatranscriptomic Sequences Processing in Linux Terminal

Quality filtering and normalization of read numbers using Cutadapt and Seqtk

```
#Quality filtering of reads
cutadapt -pair-filter=any -q 15 --minimum-length 100 -a adapter_sequence -A adapter_se -o trimmed.R1.fastq.gz -p trimmed.R2.fastq.gz 'path

#Normalization of read number by random samplnig
seqtk sample -s100 trimmed_R1.fastq 1100000 > sub_R1.fastq
seqtk sample -s100 trimmed_R2.fastq 1100000 > sub_R2.fastq
```

Transcript assembly of the metatranscriptomic reads per community using IDBA-UD and IDBA-MT

```
#File conversion from fastq to fa
fq2fa --filter input.R1.fastq  input.R1.fa

#File conversion and merging of pair-end reads
fq2fa --merge --filter input.R1.fastq input.R2.fastq out.fa

#Assembly of pair-end reads using IDBA-UD
/home/louie/idba/bin/idba_ud' -r emptyfile.fa -l actualfile.fa --mink 20 --maxk 150 -o output_name --num_threads 5

#Chimera removal in assembled transcript using IDBA-MT
idba-mt -t input_R1.fa -f input_R2.fa -c contig.fa -O output_transcript_assembly.fa
```

Quality checking of assembled transcript using TrinityStats.pl and BUSCO

```
#Set the path to Trinity directory
export TRINITY_HOME=/path/to/trinity/installation/dir

#Use TrinityStats.pl for calculation of quality paramters
TRINITY_HOME/util/TrinityStats.pl  input.fasta > quality.txt

#Use BUSCO for transcript completion with arthropod lineage dataset
run_busco -i AC1_transcript.fa -l path_to_arthropoda_odb10 -o path_output_busco -m tran
```

Extracting selected mitochondrial transcripts from the assembly community metatranscriptomics in Linux terminal

```
#Construct local BLAST database using MIDORI (for mitochondrial genes) and SILVA for (rRNA)
makeblastdb -in MIDORI+SILVA.fasta -dbtype nucl -parse_seqids -out MIDORI+SILVA -title "MIDORI+SILVA"

#BLAST query using assembled community metatranscriptomics fasta against the constructed local database
blastn -query input.fasta -db MIDORI+SILVA -num_alignments 100 -word_size 11 -outfmt 7 -dust 'no' -soft_masking 'false' -out output_blastou

#Making list of mitochondrial transcript sequences using the assembled transcript fasta file and blast result with Perl script
perl 00ext_blast_result.pl

#Processing o the generated mitochondrial transcript sequences list
grep "^# Query" input_ext.txt > input_list.txt
sed -i -e "s/# Query: />/" input_list.txt
sed 's/\ /_/g' -i input_list.txt

#Extracting the mitochondrial transcript using the constructed list and original transcript assembly from trinity
awk 'NR==FNR{a[$0];next} $1 in a{c=4} c&&c--' input_list.txt input_trinity.fasta
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e9661397-46eb-460f-a10e-ebbe01ff733f/00ext_blast_result.pl

Filtering the mitochondrial transcript sequences of target species using Mothur in Linux Terminal

```
#Open mothur in Linux terminal
mothur

#Set directory in Mothur
set.dir(input= path to the file folder)

#Taxonomic assignment of ASV sequences
classify.seqs(inputdir= path to the file folder, fasta=input.fasta, reference=MIDORI_LONGEST_GB238_mtGene_MOTHUR.fasta, taxonomy=MIDORI_LON

#Extracting sequences of target species
get.lineage(inputdir= path to the file folder, taxonomy=input.MIDORI_LONGEST_GB238_mtGene_MOTHUR.wang.taxonomy, taxon=Eukaryota_2759(100);A
```

Clustering of the mitochondrial transcripts into OTUs using VSEARCH in Linux terminal

```
#Clustering of the mitochondrial transcript with 94% similarity index
vsearch -derep_prefix input.fasta --fasta_width 0 --minseqlength 500 -output output.fasta

vsearch --cluster_fast input.fa --strand both -id 0.94 --fasta_width 0 --sizeorder --uc vsearchcentroid.uc --centroids vsearchcentroid.fa -
```

Taxonomic assignment of the filtered mitochondrial transcript using RDP classifier function of the MIDORI server

```
http://reference-midori.info/server.php
```

Mapping back the normalize reads of community metatranscriptomic sequences to its respective assembled transcript using Bowtie2, then quantification of read number with RSEM

```
#Creating indexed reference for the alignment using bowtie2
'RSEM-1.3.3/rsem-prepare-reference' --bowtie2 --bowtie2-path 'path to bowtie' input_transcript_assembly.fa indexed_reference_name

#Alignment of raw reads using bowtie2 and normalization of the aligned read count using RSEM
'RSEM-1.3.3/rsem-calculate-expression' -p 8 --paired-end --bowtie2 --bowtie2-path 'bowtie2 path' --estimate-rspd input.R1.fastq input.R2.fa
```

Extracting the read-level abundance of the selected mitochondrial transcript from the RSEM results

```
#Create a list of all the selected mitochondrial sequences
grep -e ">" input_mtGene.pick.fasta | sed -E 's/((_*[^_]*){4}).*/\1/' | sed 's/.*>//' > input_mtGene_list.txt

#Construct the table with the read-level abundance using the constructed list and the RSEM gene count table
grep -w -f input_mtGene_list.txt RSEM.genes.results > input_mtGene_list_RSEM.txt
```