Methods for detecting hybridization and phylogenetic discordance

Bioinformatic Tutorial

Paige Ellestad, Erika R. Moore-Pollard, Carolina M. Siniscalchi, Ramhari Thapa, Linda E. Watson, J. Mauricio Bonifacino, Jennifer R. Mandel

2025-03-25

Table of Contents

# 1 Introduction

Recent advances in phylogenetic network methods now allow for the estimation of ancient hybridization. Using Hyb-Seq and transcriptomic data from ~300 species within the family Asteraceae, the sunflower family, this study explores how these reticulations intersect with other evolutionary processes, such as polyploidy and whole-genome duplication events, and assesses the impact of taxon selection on these analyses. Given the complexity of these bioinformatic approaches, a tutorial was developed to guide researchers through these methodological processes, making it applicable to other study systems. Alongside a growing body of research, this study underscores the role of gene flow, hybridization, and introgression in speciation. Further advancements in bioinformatic approaches will continue to enhance theoretical studies in the field of plant evolution.

This tutorial may serve as a template for researchers aiming to investigate the complex evolutionary processes that influence the diversity of life. Researchers may contact [ermoore3@memphis.edu](mailto:ermoore3@memphis.edu) or [pllestad@memphis.edu](mailto:pllestad@memphis.edu) regarding questions in methodology. This pipeline implements code in multiple languages, mainly bash, R, python, and julia, and often uses conda as a way to quickly and cleanly install packages on a HPC.

To use conda environments on a Linux system, install miniconda3 following the steps below.

#bash  
# copy the bash script needed to install miniconda3  
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh   
 # run the bash script to actually install  
bash ~/Miniconda3-latest-Linux-x86\_64.sh  
# activate your installer  
source ~/miniconda3/bin/activate   
# make miniconda3 automatically load every time you join your HPC  
conda init --all

# 2 Preparing sequence data

## 2.1 Assessing raw sequence quality

FastqQC (<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>) is a tool that can spot potential problems in high throughput sequencing datasets. Input can be raw or trimmed sequence files in fastq or bam format. The result of FastQC is an html report which summarises the sequence quality of your samples.

### 2.1.1 Installing FastQC

Install FastQC using bioconda with the following code:

#bash  
# Install program with Bioconda  
conda create --name fastqc  
conda activate fastqc  
conda install -c bioconda fastqc

### 2.1.2 Running FastQC

**INPUT**

* Raw sequence files in fastq of bam format, zipped or unzipped.

If you have many files, you can run it as a loop. For example, with fastq files:

#bash  
for filename in \*.fastq  
do  
fastqc $filename  
done

**OUTPUT**

* html file

This html file can be opened through any web browser. A great resource to better understand the report can be found at <https://hbctraining.github.io/Intro-to-rnaseq-hpc-salmon/lessons/qc_fastqc_assessment.html>.

## 2.2 Trimming sequence data

Trimmomatic ([Bolger et al., 2014](#ref-Bolger2014)) is a read trimming tool that removes Illumina adapters from low-quality bases and adapter contamination. It’s important to do this step first to make sure you have cleaned and trimmed data going forward.

### 2.2.1 Installing Trimmomatic

Installation instructions can be found at <http://www.usadellab.org/cms/?page=trimmomatic>.

### 2.2.2 Running Trimmomatic

**INPUT**

* Paired end raw sequence files in fastq format.

After installation, the java script needed to run Trimmomatic should be in a folder such as ‘Trimmomatic-0.36’. Go into your folder containing the sequence files and run Trimmomatic in a loop on all paired end fastq files (here ending in R1.fastq and R2.fastq) using the following code:

#bash  
for fileR1 in \*R1.fastq  
do  
fileR2=`echo ${fileR1} | sed 's/R1/R2/'`  
java -jar /PATH/TO/Trimmomatic-0.36/trimmomatic-0.36.jar PE $fileR1 $fileR2 $fileR1.tp.fastq $fileR1.tunp.fastq $fileR2.tp.fastq $fileR2.tunp.fastq ILLUMINACLIP:/PATH/TO/Trimmomatic-0.36/adapters/TruSeq3-PE.fa:2:30:10 LEADING:20 TRAILING:20 SLIDINGWINDOW:5:20 MINLEN:36  
done

One common change made to the script is changing the values for ILLUMINACLIP, LEADING, TRAILING, SLIDINGWINDOW, and MINLEN. Please refer to the GitHub or primary literature to determine which values best fit your data.

**OUTPUT**

* \*.tp.fastq, the trimmed sequence data
* \*.tunp.fastq, the sequence data removed from trimming

[Optional] Re-run FastQC on the trimmed sequence data to see if the quality improved from trimming!

# 3 Assembling Sequence Data

This tutorial will cover different sequence assembly methods including SPAdes ([Prjibelski et al., 2020](#ref-Prjibelski2020)) for genome assembly using target-enriched sequence data, GetOrganelle ([Jin et al., 2020](#ref-Jin2020)) for chloroplast assembly using off-target reads from target-enriched sequence data, and Trinity ([Grabherr et al., 2011](#ref-Grabherr2011)) for transcriptome assembly of RNA sequence data.

## 3.1 Assembling genomes from target-enriched sequencing data

SPAdes ([Prjibelski et al., 2020](#ref-Prjibelski2020)) is a versatile toolkit designed for assembly and analysis of sequencing data and was primarily developed for Illumina sequencing data.

### 3.1.1 Installing SPAdes

Installation instructions can be found at <https://github.com/ablab/spades>.

### 3.1.2 Running SPAdes

**INPUT**

* \*.tp.fastq, the trimmed sequence data

#bash  
for trimR1 in `find ./ -name "\*R1.fastq.tp.fastq"`   
do  
trimR2=`echo ${trimR1} | sed 's/R1/R2/'`  
/PATH/TO/bin/spades.py -k 21,33,55,77,99 --only-assembler --pe1-1 $trimR1 --pe1-2 $trimR2 -o ./$trimR1.spades\_output   
done  
  
PAIGE- does this renaming need to be here  
# Rename spade output folder making it short having only sample name  
for i in \*;  
do  
mv "./$i" "./$(echo $i |grep ".fastq.spades\_output"| awk '{split($0,a,"\_");print a[1]}')"  
done  
  
# copy all the contig files with the sample name in a folder named ‘contigs’  
mkdir ../contigs  
for f in \*;  
do  
cp "$f/contigs.fasta" "./contigs/$f.fasta";  
done

**OUTPUT** PAIGE add more info here

* \*.fasta: contig files

## 3.2 Assembling transcriptomes from RNA-seq data

Trinity ([Grabherr et al., 2011](#ref-Grabherr2011)) represents a novel method for the efficient and robust de novo reconstruction of transcriptomes from RNA-seq data.

### 3.2.1 Installing Trinity

Installation instructions can be found at <https://github.com/trinityrnaseq/trinityrnaseq/wiki>.

### 3.2.2 Running Trinity

**INPUT**

* \*.tp.fasta, trimmed RNA-sequence data PAIGE need to add trimming for rna seq data into pipeline

#bash  
 Trinity --seqType fq --left reads\_1.fq --right reads\_2.fq --CPU 6 --max\_memory 20G   
PAIGE find out what the code actually was

**OUTPUT**

* \*.fasta: assembled transcripts in fasta format

## 3.3 Assembling chloroplast genomes from target-enriched sequencing data

Organellar sequences may be extracted from off-target sequence reads and assembled using GetOrganelle (<https://github.com/Kinggerm/GetOrganelle>) ([Jin et al., 2020](#ref-Jin2020)). Often, there will not be enough off-target sequence data for complete organellar assembly. The integrity of complete chloroplast assemblies may be assessed using Bandage ([Wick et al., 2015](#ref-Wick2015)), then annotated using GeSeq ([Tillich et al., 2017](#ref-Tillich2017)).

### 3.3.1 Installing GetOrganelle

#bash  
conda create --name getorganelle  
conda activate getorganelle  
conda install -c bioconda getorganelle

### 3.3.2 Running GetOrganelle

**INPUT**

* \*.tp.fastq, the trimmed sequence data

#bash  
conda activate getorganelle  
get\_organelle\_from\_reads.py -1 forward.fq -2 reverse.fq -o plastome\_output -R 15 -k 21,45,65,85,105 -F embplant\_pt

Other parameters may be adjusted to support the successs of a complete plastome assembly. I prefer to increase the number of rounds (-R), double the disentangle time limit (–disentangle-time-limit 7200), then possibly decrease the word size (-w) based on the default word size found in the log file. Refer to <https://github.com/Kinggerm/GetOrganelle> for more parameter adjustments.

**OUTPUT**

* \*.path\_sequence.fasta, the plastome assembly
* \*.selected\_graph.gfa, the organelle-only assembly graph
* Additional output is described at <https://github.com/Kinggerm/GetOrganelle>

### 3.3.3 Installing Bandage

Check integrity of plastome assembly with Bandage ([Wick et al., 2015](#ref-Wick2015)) using the file “\*.selected\_graph.gfa” as input. Bandage can be downloaded as a GUI executable for Mac or Windows but can also be run command-line on Linux systems using the following code from <https://github.com/rrwick/Bandage/wiki/Command-line>.

#bash  
#Upload to your HPC  
wget https://github.com/rrwick/Bandage/releases/download/v0.8.1/Bandage\_Ubuntu\_dynamic\_v0\_8\_1.zip  
  
unzip Bandage\_Ubuntu\_dynamic\_v0\_8\_1.zip

### 3.3.4 Running Bandage

**INPUT**

* \*.selected\_graph.gfa, output from GetOrganelle

#bash  
for file in \*.p\_ctg.gfa;  
do  
~/Bandage image $file $file.jpg;  
done

**OUTPUT**

* visualizations may be exported in multiple formats

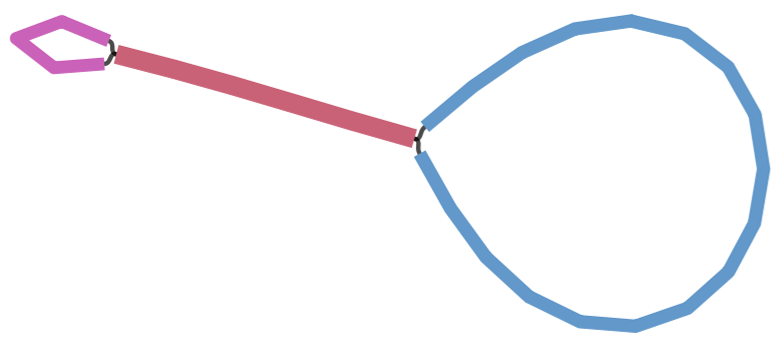


Figure 3.1: Bandage visualization of contig structure of plastome assembly for Amphoricarpus autariatus. A complete plastome assembly should show this structure representing the 4 quadripartite regions of the plastome: a large circle representing the large single copy region, a small circle representing the small single copy region, and and a connecting line representing the two inverted repeat regions.

### 3.3.5 Running GeSeq

GeSeq ([Tillich et al., 2017](#ref-Tillich2017)) is a web-based applicaion that was developed for the rapid and accurate annotation of organelle genomes, in particular chloroplast genomes. Complete circular plastome assemblies may be annotated using GeSeq ([Tillich et al., 2017](#ref-Tillich2017)) and visualized using OGDraw at <https://chlorobox.mpimp-golm.mpg.de/geseq.html>.

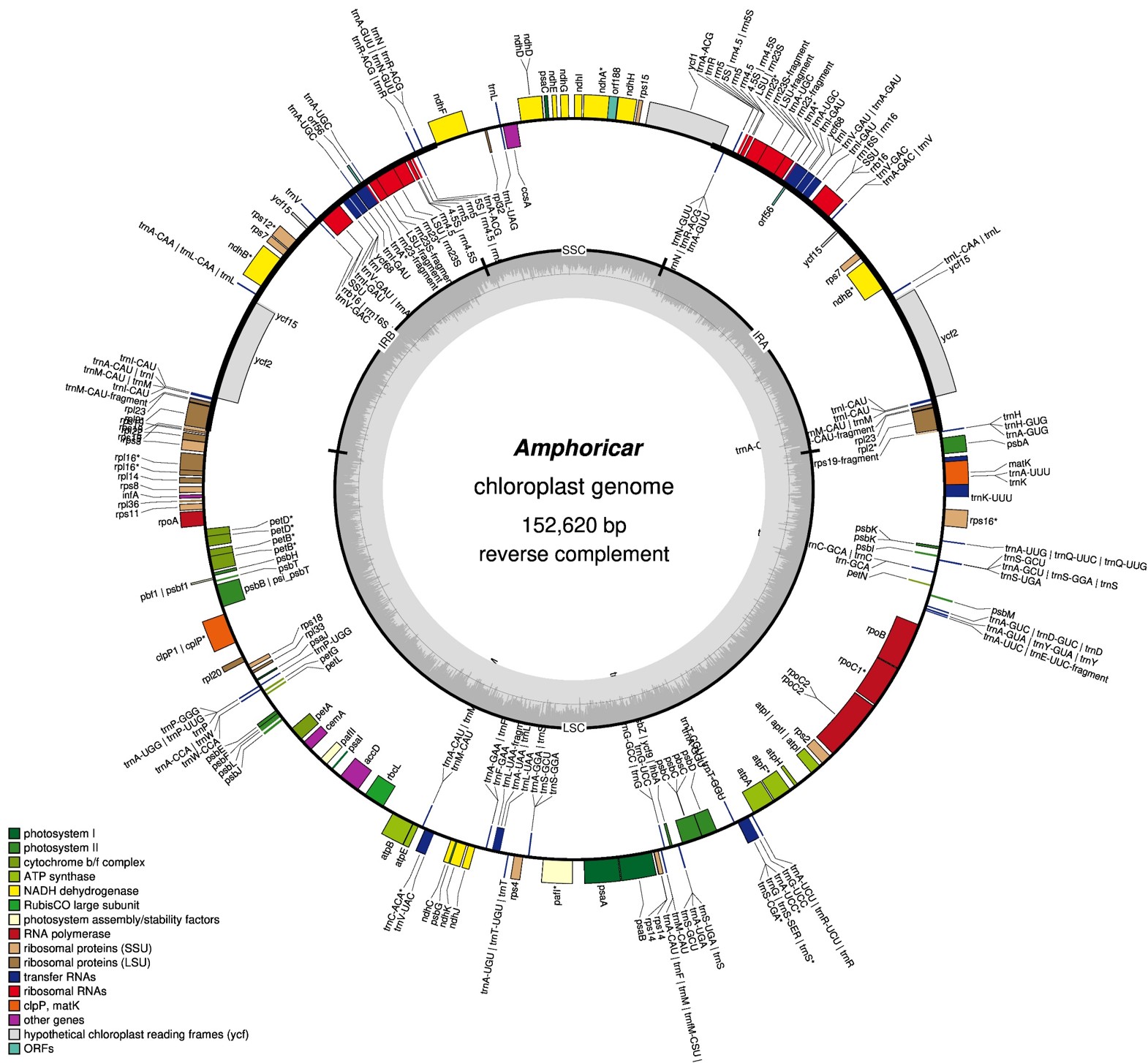


Figure 3.2: Annotated chloroplast of Amphoricarpos autariatus produced by OGDraw.

# 4 Generating Species Phylogenies

## 4.1 Identifying ultra-conserved elements from target-enriched data

PHYLUCE ([Faircloth, 2016](#ref-Faircloth2016)) is a software that determines orthology from targeted loci/genes (e.g., from target-enrichment data). It is considered ‘ultra-conserved’ compared to other orthology determination softwares (e.g., HybPiper) since it removes any loci that is considered paralogous. Other poeple use hybpiper blah blah , skip down to blah model selection section use your concatenate gene file . PAIGE

### 4.1.1 Installing Phyluce

#bash  
conda create --name phyluce  
conda activate phyluce  
conda install bioconda::phyluce

### 4.1.2 Running PHYLUCE

**INPUT**

* contigs/\*.fasta, folder containing all contig files (output from SPAdes)
* datasets.conf, a file containing a list of species used in the analysis. Species names should be separated by “\_“. When using this, you need to reference which species you want to refer to by indicating it with the ‘taxon-group’ flag.
* phyluce\_cos1061\_probes.fasta: the probe file with gene names and sequences from targeted genes. PAIGE for our probe set, this can be downloaded from erikas github….

NOTE: Make sure to change the file paths throughout and change the number of taxa to the appropriate number at each ‘–taxa’ flag when needed.

#bash  
#activate the conda environment  
conda activate phyluce  
  
# Create empty log folder  
mkdir ./log  
  
# Generate \*.lastz files for each contig from SPAdes.  
python /home/USER/miniconda3/envs/phyluce/bin/phyluce\_assembly\_match\_contigs\_to\_probes --contigs ./contigs --probes cos\_probes.fasta --output ./output --log-path ./log  
  
#Resulting files are \*.lastz files in the ‘output’ folder. These files only need to be made once and are only probe–not species–specific.  
  
# Create empty output directory, in this case it is called ‘taxon-set-all’  
mkdir ./taxon-set-all/  
  
#PAIGE what is done here  
python /home/USER/miniconda3/envs/phyluce/bin/phyluce\_assembly\_get\_match\_counts --locus-db ./output/probe.matches.sqlite --taxon-list-config datasets.conf --taxon-group 'subset4' --output ./taxon-set-all/all.conf --incomplete-matrix --log-path ./log  
  
#PAIGE what is done here  
python /home/USER/miniconda3/envs/phyluce/bin/phyluce\_assembly\_get\_fastas\_from\_match\_counts --contigs ./contigs --locus-db ./output/probe.matches.sqlite --match-count-output ./taxon-set-all/all.conf --incomplete-matrix ./taxon-set-all/all.incomplete --output ./taxon-set-all/all.fasta --log-path ./log  
  
# Explode the monolithic FASTA  
python /PATH/TO/miniconda3/envs/phyluce/bin/phyluce\_assembly\_explode\_get\_fastas\_file --input ./taxon-set-all/all.fasta --output exploded-fastas  
  
# Then run the below code to get stats  
for i in exploded-fastas/\*.fasta;  
do  
python /PATH/TO/miniconda3/envs/phyluce/bin/phyluce\_assembly\_get\_fasta\_lengths --input $i --csv;  
done > fasta\_lengths.csv  
#the resulting csv has summary stats on the FASTAS  
#the column headers are: samples,contigs,total bp,mean length,95 CI length,min length,max length,median length,contigs >1kb  
  
# Alignment without internal trimming BUT does edge trim   
#CHANGE TAXON NUMBER  
python /PATH/TO/miniconda3/envs/phyluce/bin/phyluce\_align\_seqcap\_align --input ./taxon-set-all/all.fasta --output ./taxon-set-all/mafft-nexus-trimmed/ --taxa 11 --aligner mafft --cores 4 --incomplete-matrix --log-path ./log  
  
# Get basic stats on the alignments   
python /PATH/TO/miniconda3/envs/phyluce/bin/phyluce\_align\_get\_align\_summary\_data --alignments ./taxon-set-all/mafft-nexus-trimmed/ --cores 4 --log-path ./log  
  
# Remove locus name from alignments   
python /PATH/TO/miniconda3/envs/phyluce/bin/phyluce\_align\_remove\_locus\_name\_from\_files --alignments ./taxon-set-all/mafft-nexus-trimmed/ --output ./taxon-set-all/mafft-nexus-trimmed-clean/ --log-path ./log  
  
# Generates individual gene matrix files that will be used for the pseudo-coalescent analysis  
#CHANGE TAXON NUMBER  
python /PATH/TO/miniconda3/envs/phyluce/bin/phyluce\_align\_get\_only\_loci\_with\_min\_taxa --alignments ./taxon-set-all/mafft-nexus-trimmed-clean --taxa 11 --percent 0 --output ./taxon-set-all/mafft-nexus-trimmed-clean-genes --cores 4 --log-path ./log  
  
# Build the total concatenated data matrix from the gene matrices  
python /PATH/TO/miniconda3/envs/phyluce/bin/phyluce\_align\_concatenate\_alignments --alignments ./taxon-set-all/mafft-nexus-trimmed-clean-genes --output ./taxon-set-all/mafft-nexus-trimmed-raxml --phylip --log-path ./log  
  
# Converts nexus to phylip-relaxed file format  
python /PATH/TO/miniconda3/envs/phyluce/bin/phyluce\_align\_convert\_one\_align\_to\_another --alignments ./taxon-set-all/mafft-nexus-trimmed-clean-genes --output ./taxon-set-all/mafft-fasta --input-format nexus --output-format phylip-relaxed --cores 1 --log-path ./log

**OUTPUT**

* fasta\_lengths.csv, summary stats on all sequences
* ./Exploded\_fastas/, folder containing unaligned sequence data used to generate the summary statistics (fasta\_lengths.csv)
* ./Taxon\_set\_all/
* all.conf, logs which taxa and genes were recovered with PHYLUCE
* all.fasta, all gene sequences that matched the probe set for each taxon as a multifasta file
* all.incomplete, lists all genes that did not match the probe set for each taxon
* mafft-nexus-trimmed/, folder containing all aligned sequences without internal trimming BUT does edge trim
* mafft-nexus-trimmed-clean/, folder similar to ‘mafft-nexus-trimmed/’ but removed the locus name from the alignment
* mafft-nexus-trimmed-clean-genes/, folder containing gene files that will be used for pseudo-coalescent analysis
* mafft-nexus-trimmed-raxml/, folder containing the concatenated gene matrix of all genes and taxa along with a ‘charsets’ file that will be used for model selection
* mafft-fasta/, folder similar to ‘mafft-nexus-trimmed-raxml/’ but has the files as phylip-relaxed instead of nexus

(Optional) Extracting data from the sqlite database produced by PHYLUCE PHYLUCE produces a sqlite database that contains two tables: “matches” and “match\_map”. “matches” contains which loci were recovered for each taxa, while “match\_map” has the name of the contig that matches each loci in each taxa. We can use the following commands to save these tables as csv files.

PAIGE make following navigable

1. First, navigate to the PHYLUCE output/ folder. The sqlite file ( probe.matches.sqlite) should be there.
2. Next, you want to open the file: sqlite3 probe.matches.sqlite
3. You should now have sqlite> in your working line. NOTE: With the code below, the “sqlite>” bit is the prompt, you don’t need to type it down. Type one command at a time and hit enter. sqlite> .headers on  
   sqlite> .mode csv sqlite> .output matches.csv #creates the first file sqlite> SELECT \* FROM matches; #populate the file with the contents of the matches table sqlite> .output match\_map.csv #creates the second file sqlite> SELECT \* FROM match\_map; #populate the file with the contents of the match\_map sqlite> .quit #exits sqlite
4. Now you should have two files, matches.csv and match\_map.csv, in your output folder. These files can be downloaded to get more stats on your PHYLUCE run.

## 4.2 Model Selection

These next few steps will detail how to run PartitionFinder ([Lanfear et al., 2016](#ref-Lanfear2016)) to determine which evolutionary model best fits our data to then generate gene trees.

### 4.2.1 Installing PartitionFinder

Installation instructions can be found at www.robertlanfear.com/partitionfinder/tutorial.

### 4.2.2 Running PartitionFinder

**INPUT**

* \*.phylip, tree file (output from Phyluce)
* \*.charsets, character set file (output from Phyluce)

Using the charsets file, create a configuration (.cfg) file with information about the matrix. You may use the following command to make this file.

#bash  
#code to change charsets file to cfg file PAIGE erika did it

Using the newly made .cfg file, run the following code:

#bash  
/PATH/TO/PartitionFinder.py /PATH/TO/partition\_finder.cfg --raxml --rcluster-max 100 --no-ml-tree

**OUTPUT**

* log.txt, log file
* analysis/best\_scheme.txt, file listing loci along with their corresponding best model.

## 4.3 Concatenated Phylogeny

When analyzing target-enriched sequencing data for phylogenetic analyses, multiple approaches may be implemented. For a concatenated approach all gene sequences are combined into a single matrix. This method assumes that all trees share the same evolutionary history, and therefore may not always be ideal for ancient divergences where gene trees are highly divergent. This tutorial presents instructions for a concatenated phylogeny using RAxML ([Stamatakis, 2014](#ref-Stamatakis2014)), a maximum likelihood phylogenetic inference tool. Phylogenetic trees may be visualized by an array of tools, eg. FigTree (<http://tree.bio.ed.ac.uk/software/figtree/>).

### 4.3.1 Installing RAxML

Installation instructions can be found at <https://cme.h-its.org/exelixis/web/software/raxml/cluster.html>.

### 4.3.2 Running RAxML on a concatenated matrix

**INPUT**

* \*.phylip, tree file (output from Phyluce)

Note: Change the -m function depending on the best fitting model (GTR model: GTRCAT; GTR+G model: GTRGAMMA; GTR+I+G model: GTRGAMMAI).

#bash  
/PATH/TO/raxmlHPC-PTHREADS-SSE3 -T $SLURM\_CPUS\_PER\_TASK -f a -p 12345 -x 12345 -m GTRGAMMAI -# 100 -s /PATH/TO/mafft-nexus-trimmed-raxml.phylip -n out

**OUTPUT**

*RAxML\_bipartitionsBranchLabels.out* RAxML\_bootstrap.out *RAxML\_info.out* RAxML\_bestTree.out \*RAxML\_bipartitions.out

### 4.3.3 Installing FigTree

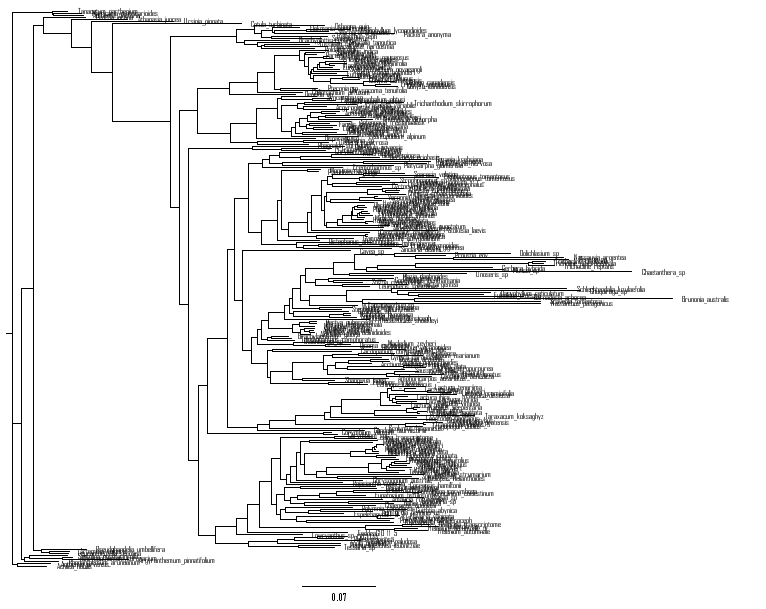
Installation instructions can be found at <https://evomics.org/resources/software/molecular-evolution-software/figtree/>.

**INPUT**

* RAxML\_bipartitionsBranchLabels.out, output from RAxML

**OUTPUT**

* visualizations may be exported in multiple formats



(#fig:concat\_tree)Phylogenetic tree produced from RAxML analysis on a concatenated matrix visualized using FigTree.

## 4.4 Coalescent-based Phylogeny

In contrast to a concatenated approach, a coalescent phylogenetic approach analyzes individual gene tree, accounting for the fact that different genes may have different evolutionary histories. This method may be better suited for resolving phylogenetic relationships in the presence of incomplete lineage sorting and hybridization, but is more computationally demanding than concatenated methods. One method to generate a coalescent phylogeny is by using Astral-III ([Zhang et al., 2018](#ref-Zhang2018)) on a suite of RAxML ([Stamatakis, 2014](#ref-Stamatakis2014)) gene trees constructed under different models. Resulting phylogenetic trees may be visualized by an array of tools, eg. FigTree (<http://tree.bio.ed.ac.uk/software/figtree/>).

### 4.4.1 Installing RAxML

Installation instructions can be found at <https://cme.h-its.org/exelixis/web/software/raxml/cluster.html>.

### 4.4.2 Running RAxML for coalescent analyses

When running RAxmL for a pseudo-coalescent (hereafter referred to as “ASTRAL”) analysis, you should separate the individual loci by their most appropriate model as indicated by the results of PartitionFinder and run each loci separately.

**INPUT**

* uce-\*.nexus, individual loci file in nexus format (output from Phyluce)
* analysis/best\_scheme.txt, file denoting the best model for each loci (output from Phyluce)

To organize all loci files into new directories pertaining to each model type, you can use the following code. In this code, loci files (eg. uce-\*.nexus) will be moved into folders labeled “batch\_exports”.

#bash  
#make folders of each model type in main working directory  
mkdir astral\_raxml  
cd astral\_raxml  
  
mkdir bestTree  
mkdir bootTree  
mkdir Astral\_try  
  
mkdir GTR  
mkdir GTRG  
mkdir GTRIG  
cd GTR  
mkdir GTR\_batch\_exports  
mkdir output  
cd ../GTRG  
mkdir GTRG\_batch\_exports  
mkdir output  
cd ../GTRIG  
mkdir GTRIG\_batch\_exports  
mkdir output  
  
#edit the best\_scheme.txt file to be used for designating loci files to proper model folder  
tail -n +22 best\_scheme.txt| sed -n '/^$/q;p' > best\_scheme\_uce\_table.txt

Move each loci folder to its designated model folder using the following R code. Change the uce location paths to match your own directory organization.

#R  
  
df <- read.table(file = "best\_scheme\_uce\_table.txt", header = TRUE, sep = "|", na.strings = "", comment.char = "", quote = "\"", fill = FALSE, nrows = 200000)  
  
df$Best.Model <- gsub(" ", "", df$Best.Model, fixed = TRUE)  
df$Partition.names <- gsub(" ", "", df$Partition.names, fixed = TRUE)  
  
GTR <- c(unlist(strsplit(df$Partition.names[which(df$Best.Model == "GTR")],",")))  
GTRG <- c(unlist(strsplit(df$Partition.names[which(df$Best.Model == "GTR+G")],",")))  
GTRIG <- c(unlist(strsplit(df$Partition.names[which(df$Best.Model == "GTR+I+G")],",")))  
  
model\_GTR <- cbind(GTR, rep("GTR", n= length(GTR)))  
model\_GTRG <- cbind(GTRG, rep("GTRG", n= length(GTRG)))  
model\_GTRIG <- cbind(GTRIG, rep("GTRIG", n= length(GTRIG)))  
  
model\_df <- as.data.frame(rbind(model\_GTR, model\_GTRG, model\_GTRIG))  
colnames(model\_df) <- c("uce", "model")  
  
  
uce\_original\_location <- "/PATH/TO/taxon-set-all/mafft-nexus-trimmed-clean-100p/"  
uce\_model\_location <- "/PATH/TO/astral\_raxml/"  
files <- list.files(uce\_original\_location)  
  
for(i in 1:length(files)) {  
   
 file\_name <- files[i]  
 file\_uce <- gsub(".nexus", "", file\_name)  
 file\_model <- model\_df$model[which(model\_df$uce %in% file\_uce)]  
 file.copy(from = paste(uce\_original\_location, file\_name, sep = ""),to = paste(uce\_model\_location, file\_model, "/", file\_model, "\_batch\_exports", sep = ""))  
   
}

Once all the gene files ARE in their appropriate model choice’s batch\_export/ folder, the file type and ending will need to be changed from .nexus to .fasta. This can be doe using the tool ElConcatenero (<https://github.com/ODiogoSilva/TriFusion>) using the following code.

#bash  
python /home/USER/ElConcatenero/ElConcatenero.py -c -if nexus -of fasta -in \*.nexus

RAxML can then be run on the individual, aligned gene files in a loop. The -m flag will need to be adjusted depending on the best fitting model (GTR model: GTRCAT; GTR+G model: GTRGAMMA; GTR+I+G model: GTRGAMMAI).

#bash  
DIR\_I=/PATH/TO/astral\_raxml/GTR/batch\_exports/\*  
  
for f in $DIR\_I  
do  
echo "Processing $f"  
file\_name=$(basename $f)  
/PATH/TO/raxmlHPC-PTHREADS-SSE3 -T 4 -f a -p 12345 -x 12345 -m GTRCAT -# 100 -s $f -n $file\_name -w /PATH/TO/astral\_raxml/GTR/output  
done

**OUTPUT**

* /output/RAxML\_bootstrap\*, bootstrap files for each model type
* /output/RAxML\_bestTree\*, best tree files for each model type

Output from these RAxML ([Stamatakis, 2014](#ref-Stamatakis2014)) analyses will be used as input for the Astral-III ([Zhang et al., 2018](#ref-Zhang2018)) analysis.

### 4.4.3 Installing ASTRAL-III

Installation instructions can be found at <https://github.com/smirarab/ASTRAL>.

### 4.4.4 Running ASTRAL-III

**INPUT**

* RAxML\_bestTree\*, tree files (output from RAxML)
* RAxML\_bootstrap\*, bootstrap files (output from RAxML)

After the output files are produced from RAxML, copy the RAxML\_bestTree\* and RAxML\_bootstrap\* files into their own folders. The file endings will then need to be changed and they will need to be concatenated. The following code may be used to prepare files for the ASTRAL analysis.

#bash  
#make new folders  
cd /PATH/TO/astral\_raxml/  
mkdir bestTree  
mkdir bootTree  
  
#copy files to new folders  
cp /PATH/TO/GTR/output/RAxML\_bootstrap\* ./bootTree/  
cp /PATH/TO/GTR/output/RAxML\_bestTree\* ./bestTree/  
  
#change fasta extension to .tre extension  
cd bestTree  
for f in \*.fasta; do  
mv -- "$f" "${f%.fasta}.tre"  
done  
cd bootTree  
for f in \*.fasta; do  
mv -- "$f" "${f%.fasta}.tre"  
done  
  
#concatenate the .tre files  
cd bestTree  
cat RAxML\_bestTree\* > concat\_best.tre   
  
#make a new folder named Astral and copy the concatenated best tree file into it  
cd /PATH/TO/astral\_raxml/  
mkdir Astral\_try  
cd Astral\_try  
cp ../bestTree/concat\_best.tre .

(Optional) We have stopped doing bootstrap analyses since the creators of ASTRAL-III said that LPP support values are more reliable than bootstrap, but the code is provided if you would like to do it. If you are running a bootstrap analysis, you also need to make a bs-files file which designates where the RAxML\_bootstrap\*.tre files are (e.g., bootTree/).

#example bs\_files  
  
/PATH/TO/astral\_raxml/bootTree/RAxML\_bootstrap.uce-1000.tre  
/PATH/TO/astral\_raxml/bootTree/RAxML\_bootstrap.uce-1005.tre  
/PATH/TO/astral\_raxml/bootTree/RAxML\_bootstrap.uce-1007.tre  
/PATH/TO/astral\_raxml/bootTree/RAxML\_bootstrap.uce-1008.tre  
# remaining bootstrap file locations

Code to make this bs-files is below.

#bash  
cd /PATH/TO/astral\_raxml/bootTree/  
ls \*.tre > bs-files  
pwd #copy the directory!  
sed -i 's\RAxML\==/PATH/TO/astral\_raxml/bootTree/==RAxML\g' bs-files #paste your working directory where it is highlighted!  
mv bs-files ../Astral\_try/

ASTRAL-III can now be run with the following code:

#bash  
#Run ASTRAL  
java -jar /PATH/TO/ASTRAL/astral.5.7.3.jar -i concat\_best.tre -o Astral\_best.tre 2> astral\_best.log  
  
#Generate LPP values  
java -jar /PATH/TO/ASTRAL/astral.5.7.3.jar -q Astral\_best.tre -i concat\_best.tre -o Astral\_lpp.tre 2> astral\_lpp.log  
  
#Get bootstrap values (optional)  
java -jar /PATH/TO/ASTRAL/astral.5.7.3.jar -i concat\_best.tre -b bs-files -o Astral\_bootcount.tre 2> astral\_bootcount.log

\*\*OUTPUT

* \*.log, log file
* Astral.lpp.tre, final tree with LPP support values.

### 4.4.5 Installing FigTree

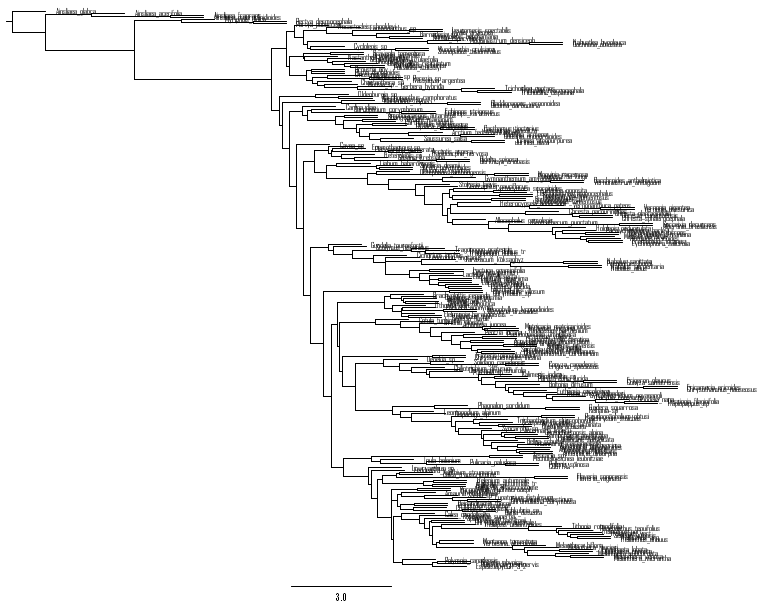
Installation instructions can be found at <https://evomics.org/resources/software/molecular-evolution-software/figtree/>.

**INPUT**

* Astral.lpp.tre, output from ASTRAL-III

**OUTPUT**

* visualizations may be exported in multiple formats



(#fig:astral\_tree)Phylogenetic tree produced from ASTRAL-III analysis visualized using FigTree.

## 4.5 Orthology-based Phylogeny

Another tool for generating phylogenetic trees within a coalescent framework is OrthoFinder ([Emms and Kelly, 2019](#ref-Emms2019)). This program identifies orthogroups, infers rooted trees for all orthogroups, identifies gene duplication events, and infers a rooted species tree.

PAIGE this whole section needs to be done

### 4.5.1 Installing OrthoFinder

#bash  
# Install program with Bioconda  
conda create --name orthofinder  
conda activate orthofinder  
conda install -c bioconda orthofinder

### 4.5.2 Running OrthoFinder

**INPUT**

#bash

**OUTPUT**

Refer to <https://davidemms.github.io/orthofinder_tutorials/exploring-orthofinders-results.html> for more detailed description of results.

## 4.6 Chloroplast Phylogeny

PAIGE also add cophylo to the visualization section

Phylogenetic trees from chloroplast sequencing data can be generated in a number of different ways. This tutorial will describe a method for plastome phylogenetic reconstruction using target-enriched sequencing data. As mentioned above in the plastome assembly seciton (PAIGE figure out how to link to dif sections), there will often not be enough off-target sequence data for complete organellar assembly.

### 4.6.1 Installing Bowtie2, Samtools, BCFtools, seqtk

#bash  
# Install program with Bioconda  
conda create --name mapping  
conda activate mapping  
conda install bioconda::bowtie2  
conda install bioconda::samtools  
conda install bioconda::bcftools  
conda install bioconda::seqtk

**INPUT**

* \*.tp.fastq, the trimmed sequence data

### 4.6.2 Running Bowtie2, Samtools, BCFtools, seqtk

This tutorial will describe a process of mapping off-target sequence reads to a reference plastome in order to create consensus sequences which may then be aligned and used as input for phylogenetic analyses.

**INPUT**

* reference.fasta, a reference plastome can be downloaded from GenBank or used from previous GetOrganelle analyses
* \*.tp.fastq, the trimmed sequence data for all samples

First, a reference database will need to be constructed using Bowtie2. This can be done using the following code:

#bash  
conda activate mapping  
  
bowtie2-build reference\_plastome.fasta reference\_plastome

After a reference database has been built, the paired end target-enriched sequence reads may be mapped on to the reference plastome. To do this in a loop with all reads with file endings “\_1\_tp.fastq” and “\_2\_tp.fastq”, use the following code:

#bash  
conda activate mapping  
  
#in your working directory, make folders for output of this pipeline  
mkdir sam\_files  
mkdir bam\_files  
mkdir fastq  
mkdir consensus\_fastas  
  
#run the pipeline in a loop on all paired files  
for fileR1 in \*\_1\_tp.fastq  
do  
fileR2=`echo ${fileR1} | sed 's/\_1\_/\_2\_/'`  
filename=`echo ${fileR1} | sed 's/\_1\_.\*//'`  
  
#map reads to reference plastome  
bowtie2 --very-sensitive -p 24 -x /PATH/TO/reference\_plastome -1 "$fileR1" -2 "$fileR2" -S /PATH/TO/sam\_files/"$filename".sam  
  
#convert sam files to bam files  
cat /PATH/TO/sam\_files/"$filename".sam | samtools view -bS - | samtools sort -o /PATH/TO/bam\_files/"$filename".bam  
  
#Get consensus fastq file  
bcftools mpileup -f /PATH/TO/reference\_plastome.fasta /PATH/TO/bam\_files/"$filename".bam | bcftools call -c | vcfutils.pl vcf2fq > /PATH/TO/fastq/"$filename"\_cns.fastq  
  
#vcfutils.pl is part of bcftools  
  
#Convert .fastq to .fasta   
seqtk seq -a /PATH/TO/fastq/"$filename"\_cns.fastq > /PATH/TO/consensus\_fastas/"$filename"\_cns.fastq  
done

The scaffold names need to be changed to match the name of the sample.

#R  
files <- list.files(pattern= "fasta")  
  
for (i in 1:length(files)) {  
   
 # 2. Set file name  
 f <- files[i]  
   
 # 3. Load fasta   
 fasta <- readLines(f)   
   
 # replace scaffold name with sample id  
   
 id <- gsub("\_cns.fasta", "", f)  
   
 new\_scaf\_name <- paste0(">", id)  
   
 fasta[1] <- new\_scaf\_name  
   
 # 7. Overwrite consensus file with new scaffold names  
 write.table(fasta, paste0(id, "\_cns.fasta"), row.names = F, col.names=F, quote = F)  
}   
 # 8. Quit R  
 quit(save="no")

PAIGE add how to calc missing data Now all of the consensus files can be concatenated and aligned for phylogenetic analysis. In this tutorial, MAFFT ([Katoh and Standley, 2013](#ref-Katoh2013)) will be used to align sequences.

### 4.6.3 Installing MAFFT

Installation instructions can be found at <https://mafft.cbrc.jp/alignment/software/source.html>.

### 4.6.4 Running MAFFT

Concatenate all sequences and align them using mafft.

#bash  
cd /PATH/TO/consensus\_fastas  
  
cat \*.fasta > all\_cns.fasta  
  
mafft all\_cns.fasta > all\_cns\_aligned.fasta

Phylogenetic analyses may be run on the aligned sequences in multiple ways. One user-friendly way is by usi IQTREE ([Minh et al., 2020](#ref-Minh2020)). Comprehensive instructions on how to run IQTREE can be found at <http://www.iqtree.org/doc/iqtree-doc#beginners-tutorial>.

### 4.6.5 Installing IQTREE

#bash  
conda create --name iqtree  
conda activate iqtree  
conda install -c bioconda iqtree

### 4.6.6 Running IQTREE

**INPUT**

* all\_cns\_aligned.fasta

#bash  
conda activate iqtree  
  
iqtree -s all\_cns\_aligned.fasta -B 1000

**OUTPUT**

* \*.phy.iqtree: the main report file
* \*.phy.treefile: the ML tree
* \*.phy.log: log file

### 4.6.7 Installing FigTree

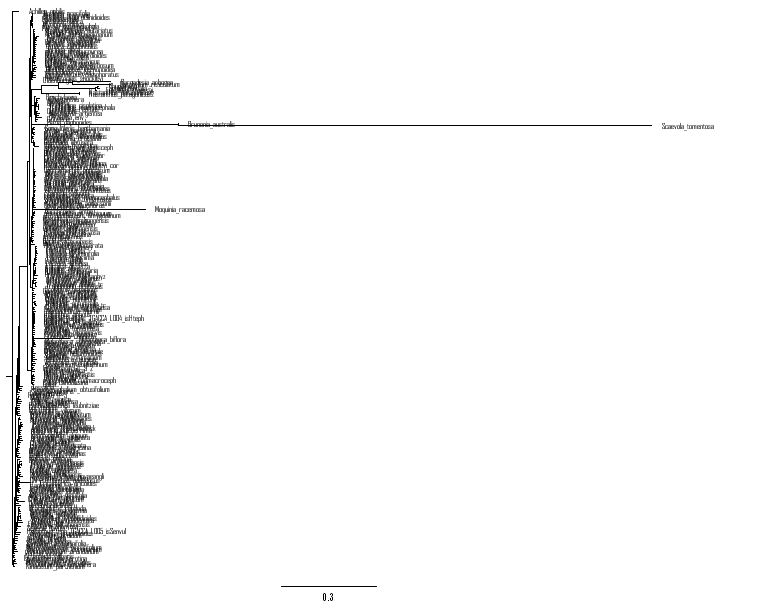
Installation instructions can be found at <https://evomics.org/resources/software/molecular-evolution-software/figtree/>.

**INPUT**

* \*.phy.treefile: ML tree file from IQTREE

**OUTPUT**

* visualizations may be exported in multiple formats



(#fig:cp\_tree)Phylogenetic tree produced from RAxML analysis on a concatenated matrix visualized using FigTree.

# 5 Identifying phylogenetic discordance

ovreview of phylo discordance PAIGE

Phylogenomic incongruence is constantly seen in phylogenomic studies and typically results from underlying gene tree discordance. Below are tutorials on how to investigate the extent of gene tree discordance among phylogenies using PhyParts (Smith et al., 2015), which summarizes how many gene trees agree/disagree with the final species topology, and Quartet Sampling (Pease et al., 2018), which uses quartets to investigate the cause of low support at each node.

## 5.1 PhyParts

PhyParts ([Smith et al., 2015](#ref-Smith2015)), along with PhypartsPieCharts, are tools that indicate the percentage of concordant gene trees, percentage in the top alternative bipartition, other conflicting topologies, and uninformative genes as pie charts on the nodes of a species phylogeny. In doing so, you gain a better visualization of discordant nodes caused by gene tree conflict. To prepare the input data for PhyParts, Phyx ([Brown et al., 2017](#ref-Brown2017)) can be used to root the species and gene trees produced from Astral-III ([Zhang et al., 2018](#ref-Zhang2018)).

## 5.2 Installing Phyx

Isntallation instructions can be found at <https://github.com/FePhyFoFum/phyx>.

## 5.3 Running Phyx

**INPUT**

* Astral\_best.tre: species tree output from Astral-III
* /astral\_raxml/bestTree/\*.tre: gene tree files output from Astral-III

Phyx ([Brown et al., 2017](#ref-Brown2017)) is a very helpful phylogenetic tool that has many functions. For the purposes of running PhyParts, we only need to use the ‘rerooting and unrooting tree’ function, pxrr. Designating the proper taxa/taxon as outgroups for rooting, the following code may be used:

#bash  
#for a single taxon rooting  
pxrr -t Astral\_best.tre -g outgroup\_taxon > Astral\_best\_rooted.tre   
#for rooting to multiple taxa  
pxrr -t Astral\_best.tre -g outgroup\_taxon1 outgroup\_taxon2 outgroup\_taxon3 … > Astral\_best\_rooted.tre   
  
#running pxrr on the gene trees in a loop   
for t in \*.tre  
do  
pxrr -t $t -g outgroup\_taxon > $t.rooted.tre  
done  
  
#Not all gene trees may have your outgroup taxon, so the number of gene trees may decrease from what you originally started with. Make sure to remove those empty gene trees. A quick, one liner code you can run to do that is:  
  
find . -maxdepth 1 -type f -empty -print -delete

**OUTPUT**

* Astral\_best\_rooted.tre: rooted species tree
* \*.rooted.tre: rooted gene trees

### 5.3.1 Installing PhyParts

General installation instructions can be found at <https://bitbucket.org/blackrim/phyparts/src/master/>, however, code is below to make a conda environment called ‘phyparts’. Additionally, the GitHub repository needs to be cloned to obtain the python scripts to run PhyParts.

#bash  
conda create -n phyparts python=2.7  
pip install ete3  
pip install matplotlib  
  
git clone https://bitbucket.org/blackrim/phyparts.git

### 5.3.2 Running PhyParts

**INPUT**

* Astral\_best\_rooted.tre: rooted species tree from Phyx
* \*.rooted.tre: rooted gene trees from Phyx

#bash  
java -jar /PATH/TO/phyparts/target/phyparts-0.0.1-SNAPSHOT-jar-with-dependencies.jar -a 1 -v -d /PATH/TO/phyluce/PhyParts/root/ -m /PATH/TO/Astral\_best\_rooted.tre -o output

**OUTPUT**

* out.concon.tre
* out.concord.node.1
* out.conflict.node.0
* out.conflict.node.2
* out.hist.alts
* out.concord.node.0
* out.concord.node.2
* out.conflict.node.1
* out.hist
* out.node.key

Detailed information on the ouput of PhyParts can be found at <https://bitbucket.org/blackrim/phyparts/src/master/>.

### 5.3.3 Running PhyPartsPieCharts

The results of PhyParts are not very figure ready. Fortunately, Matt Johnson (Texas Tech University) has generated a visualization script, called ‘phypartspiecharts.py’, that creates an svg image that can be used in manuscripts.

**INPUT**

#bash  
#copy python code from github  
git clone https://github.com/mossmatters/MJPythonNotebooks.git  
#activate conda environment  
conda activate phyparts  
  
#combine all outfiles from PhyParts  
cat \*.concord.node.\* > phyparts.concord.tre  
cat \*.conflict.node.\* > phyparts.conflict.tre  
cat \*.hist > phyparts.hist  
cat \*.hist.alts > phyparts.hist.alts  
cat \*.node.key > phyparts.node.key  
cat \*.  
concon.tre > phyparts.concon.tre  
  
#Now you just need the rooted species tree (Astral\_best\_rooted.tre), what the different outfiles can be referred to as (e.g., phyparts), and knowledge on how many gene trees were used (e.g., 1000 gene trees).  
  
/PATH/TO/phyluce/PhyParts/phypartspiecharts.py /Path/TO/phyluce/PhyParts/Astral\_best-rooted.newick phyparts 1000 --svg\_name phyparts\_piechart.svg

**OUTPUT**

* \*.svg, file containing your tree with pie charts at each node indicating:
* Blue: concordant
* Green: the top alternative bipartition
* Red: all other alternative bipartitions
* Black: uninformative for that node

## 5.4 Quartet Sampling

Quartet Sampling (QS) ([Pease et al., 2018](#ref-Pease2018)) is a quartet-based method that was designed to assess the confidence, consistency, and informativeness of internal tree relationships, and the reliability of each terminal branch. QS determines if a phylogeny has a lack of support due to not enough information, discordance from ILS or introgression, or misplaced or erroneous taxa. Three scores are calculated (QC, QD, QI) for each internal branch of the focal tree which together allows different interpretations of the branches, as well as a Quartet Fidelity (QF) score that reports how frequently a taxon is included in concordant topologies. QS efficiently synthesizes phylogenetic tests and offers more comprehensive and specific information on branch support than conventional measures (i.e., bootstrap).

### 5.4.1 Installing Quartet Sampling

#bash  
#create a conda environment for installation  
conda create -n quartetsampling numpy  
conda activate quartetsampling  
conda install -c bioconda python  
conda install -c bioconda raxml-ng   
  
#clone the GitHub to obtain the python scripts to run Quartet Sampling  
git clone https://github.com/FePhyFoFum/quartetsampling.git

### 5.4.2 Running Quartet Sampling

**INPUT**

* mafft-nexus-trimmed-raxml.phylip: alignment file (output from Phyluce)
* Astral\_best.tre: best tree file without support values (output from Astral-III)

Quartet sampling may be run using the following code with the default parameter values. For more information on adjusting the parameters go to <https://github.com/FePhyFoFum/quartetsampling/blob/master/quartetsampling.pdf>.

#bash  
python /PATH/TO/quartetsampling/pysrc/quartet\_sampling.py --tree /PATH/TO/Astral\_best.tre --align /PATH/TO/mafft-nexus-trimmed-raxml.phylip --reps 100 --threads 4 --lnlike 2

**OUTPUT**

* RESULT.labeled.tre
* RESULT.labeled.tre.freq
* RESULT.labeled.tre.qd
* RESULT.node.counts.csv
* RESULT.run.stats
* RESULT.labeled.tre.figtree
* RESULT.labeled.tre.qc
* RESULT.labeled.tre.qi
* RESULT.node.scores.csv

Each file has great information about the different scores generated by Quartet Sampling. The final tree file that shows all values (QC, QI, QD, and QF) is RESULT.labeled.tre.figtree. This tree file can be put into FigTree to visualize discordance in the tree.

### 5.4.3 Installing FigTree

Installation instructions can be found at <https://evomics.org/resources/software/molecular-evolution-software/figtree/>.

**INPUT**

* RESULT.labeled.tre.figtree: output from Quartet Sampling

**OUTPUT**

* visualizations may be exported in multiple formats

# 6 Investigating Ancient Hybridization

## 6.1 Phylogenetic networks

# 7 Inferring Whole Genome Duplication

PAIGE overview of ways to identify wGD

## 7.1 WGDv2

wgd v2 ([Chen et al., 2024](#ref-Chen2024)) is a package that can be used to infer and date ancient whole-genome duplication (WGD) events. This tutorial will describe using wgd v2 exclusively to generate Ks plots to determine the evolutionary age of WGD events using transcriptomic data which are first translated to coding sequences using TransDecoder (<https://github.com/TransDecoder/TransDecoder>).

### 7.1.1 Installing TransDecoder

Installation instructions can be found at <https://github.com/TransDecoder/TransDecoder>. PAIGEadd the singularity installation? it is the only one in the whole pipeline.

### 7.1.2 Running TransDecoder

**INPUT**

* assembled transcripts in fasta format (output from Trinity)

#bash  
#extract the long open reading frames  
TransDecoder.LongOrfs -t assembled\_transctipts.fasta --output\_dir output  
  
#predict the likely coding regions  
TransDecoder.Predict -t assembled\_transctipts.fasta --output\_dir output

**OUTPUT**

* \*.fasta.transdecoder.pep: peptide sequences for the final candidate ORFs; all shorter candidates within longer ORFs were removed.
* \*.fasta.transdecoder.cds: nucleotide sequences for coding regions of the final candidate ORFs
* \*.fasta.transdecoder.gff3: positions within the target transcripts of the final selected ORFs
* \*.fasta.transdecoder.bed: bed-formatted file describing ORF position.

### 7.1.3 Installing WGDv2

The wgdV2 repository can be cloned from github then a conda environment can be created.

#bash  
#clone repository   
git clone https://github.com/heche-psb/wgd   
  
#create conda environment and install programs  
conda create -n wgdv2 python=3.8  
conda activate wgdv2  
pip install numpy==1.19.0  
pip install wgd==2.0.38  
conda install diamond #future dependencies  
conda install bioconda::mafft #future dependencies  
conda install bioconda::mcl #future dependencies

### 7.1.4 Running WGDv2

To produce Ks plots from out transcriptomic data, we followed “Pipeline 3: Construction of age distribution and ELMM analysis from GitHub” (<https://github.com/heche-psb/wgd>). Given this, we will be running three commands–‘dmd’, ‘ksd’, and ‘viz’–on the \*.cds files generated from TransDecoder with the transcriptome data.

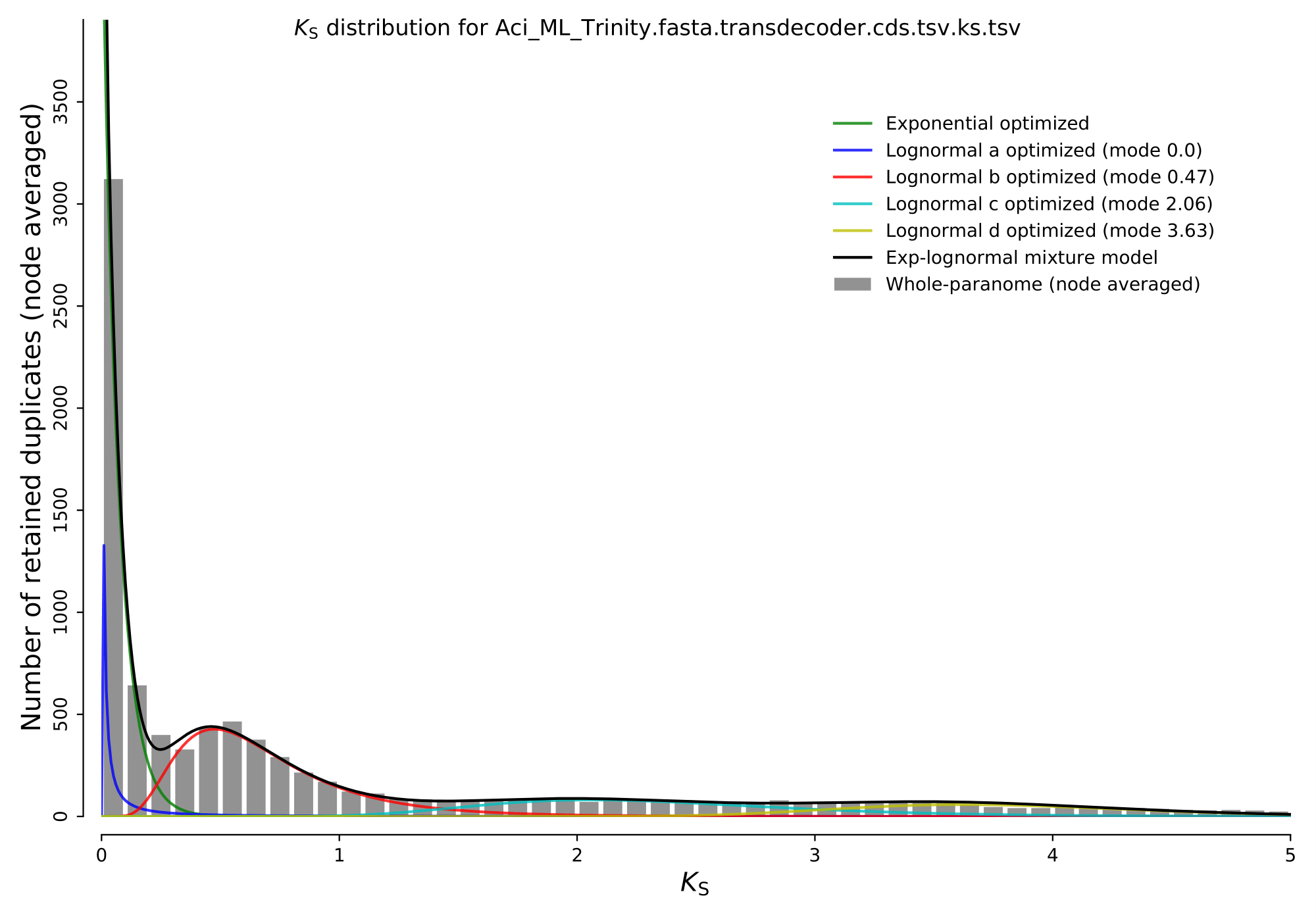
**INPUT**

* \*cds, coding sequence files (output from TransDecoder)

#bash  
#Construction of age distribution and ELMM analysis  
wgd dmd transcripts.transdecoder.cds -o wgd\_dmd  
wgd ksd wgd\_dmd/transcripts.transdecoder.cds.tsv transcripts.transdecoder.cds -o wgd\_ksd  
wgd viz -d wgd\_ksd/transcripts.transdecoder.cds.tsv.ks.tsv -o wgd\_ELMM

**OUTPUT**

* wgd\_ELMM/
  + \*.tsv.ks.tsv.ksd.pdf/svg
  + \*.tsv.ks.tsv.spline\_node\_averaged.pdf/svg
  + \*.tsv.ks.tsv.spline\_weighted.pdf/svg
  + \*.tsv.ks.tsv\_peak\_detection\_node\_averaged.pdf/svg
  + \*.tsv.ks.tsv\_peak\_detection\_weighted.pdf/svg
  + elmm\_\*.tsv.ks.tsv\_best\_models\_node\_averaged.pdf/svg
  + elmm\_\*.tsv.ks.tsv\_best\_models\_weighted.pdf/svg
  + elmm\_\*.tsv.ks.tsv\_models\_data\_driven\_node\_averaged.pdf/svg
  + elmm\_\*.tsv.ks.tsv\_models\_data\_driven\_weighted.pdf/svg
  + elmm\_\*.tsv.ks.tsv\_models\_random\_node\_averaged.pdf/svg
  + elmm\_\*.tsv.ks.tsv\_models\_random\_weighted.pdf/svg
  + elmm\_BIC\_\*.tsv.ks.tsv\_node\_averaged.pdf/svg
  + elmm\_BIC\_\*.tsv.ks.tsv\_weighted.pdf/svg

 # References

Bolger, A.M., M. Lohse, and B. Usadel. 2014. [Trimmomatic: A flexible trimmer for illumina sequence data](https://doi.org/10.1093/BIOINFORMATICS/BTU170). *Bioinformatics* 30: 2114–2120.

Brown, J.W., J.F. Walker, and S.A. Smith. 2017. [Phyx: Phylogenetic tools for unix](https://doi.org/10.1093/bioinformatics/btx063). *Bioinformatics* 33: 1886–1888.

Chen, H., A. Zwaenepoel, and Y.V. de Peer. 2024. [Wgd v2: A suite of tools to uncover and date ancient polyploidy and whole-genome duplication](https://doi.org/10.1093/bioinformatics/btae272). *Bioinformatics* 40:

Emms, D.M., and S. Kelly. 2019. [OrthoFinder: Phylogenetic orthology inference for comparative genomics](https://doi.org/10.1186/s13059-019-1832-y). *Genome Biology* 20:

Faircloth, B.C. 2016. [PHYLUCE is a software package for the analysis of conserved genomic loci](https://doi.org/10.1093/bioinformatics/btv646). *Bioinformatics* 32: 786–788.

Grabherr, M.G., B.J. Haas, M. Yassour, J.Z. Levin, D.A. Thompson, I. Amit, X. Adiconis, et al. 2011. [Full-length transcriptome assembly from RNA-seq data without a reference genome](https://doi.org/10.1038/nbt.1883). *Nature Biotechnology* 29: 644–652.

Jin, J.J., W.B. Yu, J.B. Yang, Y. Song, C.W. Depamphilis, T.S. Yi, and D.Z. Li. 2020. GetOrganelle: A fast and versatile toolkit for accurate de novo assembly of organelle genomes. *Genome Biology* 21: 1–31. Available at: <https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-02154-5>.

Katoh, K., and D.M. Standley. 2013. MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Molecular Biology and Evolution* 30: 772–780. Available at: </pmc/articles/PMC3603318/ /pmc/articles/PMC3603318/?report=abstract https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3603318/>.

Lanfear, R., P.B. Frandsen, A.M. Wright, T. Senfeld, and B. Calcott. 2016. [PartitionFinder 2: New methods for selecting partitioned models of evolution for molecular and morphological phylogenetic analyses](https://doi.org/10.1093/molbev/msw260). *Molecular Biology and Evolution*msw260.

Minh, B.Q., H.A. Schmidt, O. Chernomor, D. Schrempf, M.D. Woodhams, A.V. Haeseler, R. Lanfear, and E. Teeling. 2020. IQ-TREE 2: New models and efficient methods for phylogenetic inference in the genomic era. *Molecular Biology and Evolution* 37: 1530–1534. Available at: <https://dx.doi.org/10.1093/molbev/msaa015>.

Pease, J.B., J.W. Brown, J.F. Walker, C.E. Hinchliff, and S.A. Smith. 2018. [Quartet sampling distinguishes lack of support from conflicting support in the green plant tree of life](https://doi.org/10.1002/ajb2.1016). *American Journal of Botany* 105: 385–403.

Prjibelski, A., D. Antipov, D. Meleshko, A. Lapidus, and A. Korobeynikov. 2020. [Using SPAdes de novo assembler](https://doi.org/10.1002/cpbi.102). *Current Protocols in Bioinformatics* 70:

Smith, S.A., M.J. Moore, J.W. Brown, and Y. Yang. 2015. [Analysis of phylogenomic datasets reveals conflict, concordance, and gene duplications with examples from animals and plants](https://doi.org/10.1186/s12862-015-0423-0). *BMC Evolutionary Biology* 15: 150.

Stamatakis, A. 2014. RAxML version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* 30: 1312–1313. Available at: <https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu033>.

Tillich, M., P. Lehwark, T. Pellizzer, E. Ulbricht-Jones, A. Fischer, R. Bock, and S. Greiner. 2017. GeSeq- versatile and accurate annotation of organelle genomes. *Nucleic Acids Research* 45:

Wick, R.R., M.B. Schultz, J. Zobel, and K.E. Holt. 2015. [Bandage: Interactive visualization of de novo genome assemblies](https://doi.org/10.1093/bioinformatics/btv383). *Bioinformatics* 31: 3350–3352.

Zhang, C., M. Rabiee, E. Sayyari, and S. Mirarab. 2018. ASTRAL-III: Polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics* 19: 15–30. Available at: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2129-y>.

# 8 Appendix 1

Citations of all R packages used to generate this report.

[1] J. Allaire, Y. Xie, C. Dervieux, et al. *rmarkdown: Dynamic Documents for R*. R package version 2.29. 2024. <https://github.com/rstudio/rmarkdown>.

[2] H. Bengtsson. *R.methodsS3: S3 Methods Simplified*. R package version 1.8.2. 2022. <https://github.com/HenrikBengtsson/R.methodsS3>.

[3] H. Bengtsson. *R.oo: R Object-Oriented Programming with or without References*. R package version 1.27.0. 2024. <https://github.com/HenrikBengtsson/R.oo>.

[4] H. Bengtsson. *R.utils: Various Programming Utilities*. R package version 2.12.3. 2023. <https://henrikbengtsson.github.io/R.utils/>.

[5] H. Bengtsson. “The R.oo package - Object-Oriented Programming with References Using Standard R Code”. In: *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. Ed. by K. Hornik, F. Leisch and A. Zeileis. Vienna, Austria, Mar. 2003. <https://www.r-project.org/conferences/DSC-2003/Proceedings/Bengtsson.pdf>.

[6] H. Bengtsson. “The R.oo package - Object-Oriented Programming with References Using Standard R Code”. In: *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. Ed. by K. Hornik, F. Leisch and A. Zeileis. Vienna, Austria, Mar. 2003. <https://www.r-project.org/conferences/DSC-2003/Proceedings/Bengtsson.pdf>.

[7] C. Boettiger. *knitcitations: Citations for Knitr Markdown Files*. R package version 1.0.12. 2021. <https://github.com/cboettig/knitcitations>.

[8] D. Charif and J. Lobry. “SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis.” In: *Structural approaches to sequence evolution: Molecules, networks, populations*. Ed. by U. Bastolla, M. Porto, H. Roman and M. Vendruscolo. Biological and Medical Physics, Biomedical Engineering. ISBN : 978-3-540-35305-8. New York: Springer Verlag, 2007, pp. 207-232.

[9] D. Charif and J. R. Lobry. *seqinr: Biological Sequences Retrieval and Analysis*. R package version 4.2-36. 2023. <https://seqinr.r-forge.r-project.org/>.

[10] J. Cheng, B. Schloerke, B. Karambelkar, et al. *leaflet: Create Interactive Web Maps with the JavaScript Leaflet Library*. R package version 2.2.2. 2024. <https://rstudio.github.io/leaflet/>.

[11] M. C. Koohafkan. *kfigr: Integrated Code Chunk Anchoring and Referencing for R Markdown Documents*. R package version 1.2.1. 2021. <https://github.com/mkoohafkan/kfigr>.

[12] J. Ooms. *magick: Advanced Graphics and Image-Processing in R*. R package version 2.8.5. 2024. <https://docs.ropensci.org/magick/>.

[13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2023. <https://www.R-project.org/>.

[14] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. <https://ggplot2.tidyverse.org>.

[15] H. Wickham, J. Bryan, M. Barrett, et al. *usethis: Automate Package and Project Setup*. R package version 3.1.0. 2024. <https://usethis.r-lib.org>.

[16] H. Wickham, W. Chang, L. Henry, et al. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.5.1. 2024. <https://ggplot2.tidyverse.org>.

[17] H. Wickham, R. François, L. Henry, et al. *dplyr: A Grammar of Data Manipulation*. R package version 1.1.4. 2023. <https://dplyr.tidyverse.org>.

[18] H. Wickham, J. Hester, W. Chang, et al. *devtools: Tools to Make Developing R Packages Easier*. R package version 2.4.5. 2022. <https://devtools.r-lib.org/>.

[19] D. Winter. *pafr: Read, Manipulate and Visualize Pairwise mApping Format Data*. R package version 0.0.2. 2020. <https://dwinter.github.io/pafr/>.

[20] Y. Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*. Boca Raton, Florida: Chapman and Hall/CRC, 2016. ISBN: 978-1138700109. <https://bookdown.org/yihui/bookdown>.

[21] Y. Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.41. 2024. <https://github.com/rstudio/bookdown>.

[22] Y. Xie. *Dynamic Documents with R and knitr*. 2nd. ISBN 978-1498716963. Boca Raton, Florida: Chapman and Hall/CRC, 2015. <https://yihui.org/knitr/>.

[23] Y. Xie. *formatR: Format R Code Automatically*. R package version 1.14. 2023. <https://github.com/yihui/formatR>.

[24] Y. Xie. “knitr: A Comprehensive Tool for Reproducible Research in R”. In: *Implementing Reproducible Computational Research*. Ed. by V. Stodden, F. Leisch and R. D. Peng. ISBN 978-1466561595. Chapman and Hall/CRC, 2014.

[25] Y. Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.49. 2024. <https://yihui.org/knitr/>.

[26] Y. Xie and J. Allaire. *tufte: Tufte’s Styles for R Markdown Documents*. R package version 0.13. 2023. <https://github.com/rstudio/tufte>.

[27] Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman and Hall/CRC, 2018. ISBN: 9781138359338. <https://bookdown.org/yihui/rmarkdown>.

[28] Y. Xie, C. Dervieux, and E. Riederer. *R Markdown Cookbook*. Boca Raton, Florida: Chapman and Hall/CRC, 2020. ISBN: 9780367563837. <https://bookdown.org/yihui/rmarkdown-cookbook>.

[29] H. Zhu. *kableExtra: Construct Complex Table with kable and Pipe Syntax*. R package version 1.4.0. 2024. <http://haozhu233.github.io/kableExtra/>.