Basal Asteraceae - R Markdown

**TABLE OF CONTENTS**

[**Introduction 3**](#_nljmw28focnj)

[**Miniconda or Anaconda installation (if needed) 4**](#_khihxbdv2kly)

[**Prepare sequence data 5**](#_gsujzeciv73y)

[1. Check sequence quality with FastQC 5](#_92m6q2c5j06h)

[2. Trim sequence data with Trimmomatic 5](#_kf32tramzs7d)

[3. Run SPAdes Genome Assembler 6](#_3m2dr4wjwepo)

[**Generate species phylogenies 7**](#_mfmlgm278dwz)

[PHYLUCE 7](#_n4mr1chwvq75)

[1. Install PHYLUCE 7](#_xkcqkl93soy9)

[2. Run PHYLUCE 7](#_xm0pkya2xzii)

[3. (Optional) Extracting data from the sqlite database produced by PHYLUCE 10](#_cjtzacvpxkja)

[PartitionFinder for model selection 12](#_f2xyrshuk8a7)

[1. Install PartitionFinder 12](#_5cwo6zsun5df)

[2. Run PartitionFinder 12](#_kcbtcw2775x3)

[Concatenated phylogeny 14](#_ormi1pbw8ejb)

[1. Install RAxML 14](#_8d9mgo97w9z5)

[2. Run RAxML on concatenated matrix 14](#_l6xpd2gzmmmq)

[Pseudo-coalescent phylogeny 15](#_y301khdkej95)

[RAxML to generate gene trees 15](#_e7d1q0apy5xf)

[1. Install RAxML 15](#_496u9u53wxut)

[2. Prepare folders/files 15](#_mydsbhi71j4g)

[3. Run RAxML 17](#_alern8hhr1nx)

[ASTRAL-III 18](#_myntjettj9oi)

[1. Install ASTRAL-III 18](#_t2sjavcea9ob)

[2. Prepare folders/files 18](#_e4ddm77i8qpj)

[3. Run ASTRAL-III 19](#_a9mcgmw73h8j)

[**Check for discordance 21**](#_r9hc1uar22wz)

[PhyParts 21](#_9axan7oanae4)

[1. Installation 21](#_yfwd06x9gb7v)

[2. Data preparation 21](#_5y7nmcoyl6eu)

[3. Run PhyParts 22](#_t9q1us64opiq)

[4. Run PhyPartsPieCharts 23](#_9tjp8s54hy3m)

[Quartet Sampling 24](#_cing1b6g88y5)

[1. Installation 24](#_ubd40ak4ynx8)

[2. Data preparation 24](#_vf1y6ldtj7g5)

[3. Run Quartet Sampling 25](#_o1lzuq8qnlpk)

[**WGDv2 - Ks plots 26**](#_rdfx5usuim6y)

[1. Prepare transcriptomic data 26](#_9jk21fwkz1i6)

[1. Installation 26](#_z3am1je0q7ga)

[2. Run TransDecoder 26](#_2k4ne8swq6zy)

[2. Generate Ks plots using wgd v2 27](#_31eujop93s3u)

[1. Installation 27](#_pfc5wr62ojpa)

[2. Run wgd v2 27](#_fwphh94oarj7)

[3. Visualize the results in R 29](#_fr3262c7yri)

[**Investigate ancient hybridizations 30**](#_amy4e9ddgv74)

[PhyloNetworks 30](#_3q15ipb4d862)

[1. Installation 30](#_ujmt1sul9vcu)

[2. Data preparation 30](#_xfubfhuwe0mk)

[(Optional) Prune species and gene tree(s) 30](#_9khpw6u0ne6c)

[Prune the species tree 30](#_t6z7fskwgb3l)

[Prune the underlying gene trees 31](#_3elghoejk0w)

[3. Run PhyloNetworks 32](#_4mroa9bdzvva)

[1. With no bootstraps 32](#_3hnsk0127c15)

[2. Determine best number of hybridizations 33](#_70cz2eh2zqgc)

[3. With bootstraps 34](#_ep2nqg4hl7iw)

[4. Visualize networks 35](#_4m7fha90ft5l)

[**Plastomes 35**](#_kyrfvmed2xr)

[**References 37**](#_mry7bu7wlred)

[*To include as an example file on GitHub 38*](#_pbzs8hbvnwio)

# 

# **Introduction**

Need to include an introduction of the project and goals.

Need to include that there are two data types we are using: target-enrichment (n=\_) and transcriptome data (n=69).

Need to designate which analyses use which data types.

# **Miniconda or Anaconda installation (if needed)**

Many of the softwares in this pipeline can be installed via Conda environments. Conda[<https://docs.anaconda.com/>] is amazingly powerful and simple to use. There is an entire collection of biology-related software that has been deposited into a “channel” of conda called bioconda. Check out all the available software packages you can install here — more than 7,000 and growing:

[anaconda.org/bioconda/repo](http://anaconda.org/bioconda/repo)

Depending on what works best for you, you should install **anaconda**[<https://docs.anaconda.com/anaconda/>] or **miniconda**[<https://docs.anaconda.com/miniconda/>]. To determine which installer would work best for you, refer here[<https://docs.anaconda.com/distro-or-miniconda/>].

**Example:** miniconda3 installation

To install miniconda3, follow the steps below:

cd /home/USER/ # go into your home directory

wget <https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh> # copy the bash script needed to install miniconda3

bash ~/Miniconda3-latest-Linux-x86\_64.sh # run the bash script to actually install

source ~/miniconda3/bin/activate # activate your installer

conda init --all # has miniconda3 automatically load every time you join your HPC

Close out of the terminal and restart your session for conda to be active and ready to use.

# **Prepare sequence data**

## 1. Check sequence quality with FastQC

[FastqQC](https://github.com/s-andrews/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.

1. Installation with a new conda environment ‘fastqc’

conda create --name fastqc

conda activate fastqc

conda install -c bioconda fastqc

1. Run samples through fastqc. The only input it needs are your sequence files in fastq or bam format. If you have many files, you can run it as a loop. For example, with fastq files:

cd /PATH/TO/FASTQ/FILES/

for filename in \*.fastq

do

fastqc $filename

done

Note: The fastq files can be zipped or unzipped!

1. The result should be an html file for each sequence run through the loop. This html file can be opened through any web browser. A great resource to better understand the report can be found [here](https://hbctraining.github.io/Intro-to-rnaseq-hpc-salmon/lessons/qc_fastqc_assessment.html).

## 2. Trim sequence data with Trimmomatic

[Trimmomatic](https://github.com/usadellab/Trimmomatic) 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.

1. Installation instructions can be found [here](http://www.usadellab.org/cms/?page=trimmomatic).
2. After installation, the java script needed to run Trimmomatic should be in a folder such as ‘Trimmomatic-0.36’.
3. Go into your folder containing the sequence files and run the below script.

trimmomatic.sh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cd /PATH/TO/FASTQ/FILES/

for fileR1 in \*R1.fastq

do

fileR2=`echo ${fileR1} | sed 's/R1/R2/'`

java -jar /home/USER/Trimmomatic-0.36/trimmomatic-0.36.jar PE $fileR1 $fileR2 $fileR1.tp.fastq $fileR1.tunp.fastq $fileR2.tp.fastq $fileR2.tunp.fastq ILLUMINACLIP:/home/USER/Trimmomatic-0.36/adapters/TruSeq3-PE.fa:2:30:10 LEADING:20 TRAILING:20 SLIDINGWINDOW:5:20 MINLEN:36

done

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note: This script is assuming that your paired-end reads have R1 or R2 endings. If that is not the case with your sequence data, then please alter the script.

1. One common change made to the script is changing the values for ILLUMINACLIP, LEADING, TRAILING, SLIDINGWINDOW, and MINLEN. Please refer to the [GitHub](https://github.com/usadellab/Trimmomatic) or primary literature to determine which values best fit your data.
2. Results of Trimmomatic will be two types of files: \*.tp.fastq or \*.tunp.fastq. The \*.tunp.fastq files contain sequence data removed from trimming, while \*.tp.fastq files have the resulting trimmed data. **Future work will continue with the \*.tp.fastq files!!**
3. [Optional] Re-run [FastQC](#_92m6q2c5j06h) on the trimmed sequence data (\*.tp.fastq) to see if the quality improved from trimming!

## 3. Run SPAdes Genome Assembler

SPAdes is a versatile toolkit designed for assembly and analysis of sequencing data. We need to assemble the data with SPAdes before running [PHYLUCE](#_n4mr1chwvq75) (detailed below).

1. Installation instructions can be found [here](https://github.com/ablab/spades).
2. Run SPAdes as a script with your trimmed sequence data from [Trimmomatic](#_kf32tramzs7d) (\*.fastq.tp.fastq):

spades.sh  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# run SPADES for all the samples with a loop

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

# 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

# **Generate species phylogenies**

## PHYLUCE

PHYLUCE 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](https://github.com/mossmatters/HybPiper)) since it removes any loci that is considered paralogous.

Helpful links:

https://phyluce.readthedocs.io/en/latest/index.html

https://jasonleebrown.github.io/UCE\_phyluce\_pipeline/

https://github.com/jasonleebrown/UCE\_phyluce\_pipeline

https://phyluce.readthedocs.io/en/latest/daily-use/list-of-programs.html

### 1. Install PHYLUCE

Follow the [installation instructions](https://phyluce.readthedocs.io/en/latest/installation.html) as designated by PHYLUCE given your operating system.

### 2. Run PHYLUCE

Now that we have PHYLUCE installed, we can now run it on our trimmed and assembled sequence data from [Step 3](#_3m2dr4wjwepo). The scripts below can be run on the command line individually or all together through a bash script (preferred).

NOTE: The scripts should be dependent upon your samples. Please refer to the [PHYLUCE ReadTheDocs](https://phyluce.readthedocs.io/en/latest/) page for more information.

Folder structure should be:

* datasets.conf - This is the name list. When using this, you need to reference which species you want to refer to by indicating it with the ‘taxon-group’ flag. An example datasets.conf file can be found on [GitHub](#_pbzs8hbvnwio).
* cos\_probes.fasta - This is the probe file with gene names and sequences from targeted genes. An example probe file can be found on [GitHub](#_pbzs8hbvnwio)
* contigs/ - This folder contains all the trimmed and assembled sequence data from [Step 3](#_3m2dr4wjwepo).
* (if desired) phyluce.sh - A script with all the below code

PHYLUCE will create ./log, ./exploded-fastas, and ./taxon-set-all folders folders throughout

\*\*Make sure to change the file paths throughout and change the number of taxa to the appropriate number at each '--taxa' flag when needed.

# Remember to always activate the environment before running PHYLUCE!

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/

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

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 /home/USER/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 /home/USER/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 /home/USER/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 /home/USER/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 /home/USER/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 /home/USER/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 /home/USER/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 /home/USER/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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Resulting files/folders from these runs are briefly summarized below:

* fasta\_lengths.csv - This file has summary stats on all of your sequences
* ./Exploded\_fastas/ - This folder contains your unaligned sequence data used to generate the summary statistics (fasta\_lengths.csv)
* ./Taxon\_set\_all/
  + all.conf - This file logs which taxa and genes were recovered with PHYLUCE
  + all.fasta - This file has all gene sequences that matched the probe set for each taxon as a multifasta file
  + all.incomplete - This file lists all genes that did not match the probe set for each taxon
  + mafft-nexus-trimmed/ - This folder contains all aligned sequences without internal trimming BUT does edge trim
  + mafft-nexus-trimmed-clean/ - This folder is similar to ‘mafft-nexus-trimmed/’ but removed the locus name from the alignment
  + mafft-nexus-trimmed-clean-genes/ - This folder contains gene files that will be used for pseudo-coalescent analysis
  + mafft-nexus-trimmed-raxml/ - This folder contains the concatenated gene matrix of all genes and taxa along with a ‘charsets’ file that will be used in [Step 3: PartitionFinder](#_f2xyrshuk8a7)
  + mafft-fasta/ - This folder is similar to ‘mafft-nexus-trimmed-raxml/’ but has the files as phylip-relaxed instead of nexus

### 3. (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.

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

1. You should now have sqlite> in your working line.

\*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

1. 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.

## 

## PartitionFinder for model selection

These next few steps will detail how to run [PartitionFinder](https://doi.org/10.1093/molbev/mss020) to determine which evolutionary model best fits our data to then generate gene trees.

### 1. Install PartitionFinder

Reference the [PartitionFinder website](https://www.robertlanfear.com/partitionfinder/) for installation instructions.

### 2. Run PartitionFinder

1. In your main working directory (e.g., /home/USER/phyluce/), create a new folder called ‘partition\_finder’.

cd /home/USER/phyluce/

mkdir partition\_finder/

cd partition\_finder/

1. Copy the \*.phylip and \*.charsets file from mafft-nexus-trimmed-raxml/ into partition\_finder

cp ../taxon-set-all/mafft-nexus-trimmed-raxml/mafft-nexus-trimmed-raxml.\* .

1. Using the charsets file, we need to create a configuration (.cfg) file with information about the matrix. An example of what the charsets file would generally look like:

head mafft-nexus-trimmed-100p-raxml.charsets

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

begin sets;

charset 'uce-100.nexus' = 1-529;

charset 'uce-1000.nexus' = 530-1433;

charset 'uce-1001.nexus' = 1434-2275;

charset 'uce-1004.nexus' = 2276-2736;

charset 'uce-1005.nexus' = 2737-3439;

charset 'uce-1008.nexus' = 3440-3912;

charset 'uce-1009.nexus' = 3913-4528;

charset 'uce-1015.nexus' = 4529-5310;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. The .cfg file needs to look something like this:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

alignment = mafft-nexus-trimmed-raxml.phylip;

branchlengths = linked;

models = all;

model\_selection = AICc;

[data\_blocks]

uce-100 = 1-529;

uce-1000 = 530-1433;

uce-1001 = 1434-2275;

uce-1004 = 2276-2736;

..........................

uce-991 = 343100-343597;

uce-993 = 343598-344362;

uce-994 = 344363-344838;

uce-995 = 344839-345496;

[schemes]

search = rcluster;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. You can edit the .cfg file to match your data manually using tools like Microsoft Excel, sed, etc.
2. Once you have your partition\_finder.cfg file (example [here](#_pbzs8hbvnwio)), you can run PartitionFinder either on the command line or as a script (preferred):

/PATH/TO/PartitionFinder.py /home/USER/phyluce/partition\_finder/partition\_finder.cfg --raxml --rcluster-max 100 --no-ml-tree

PartitionFinder creates a log file (log.txt) and an analysis/ folder which contains best\_scheme.txt. This file shows you the loci and their corresponding best model.

Record the results and counts of each loci model choice:

GTR: \_\_\_; GTR+G: \_\_\_; GTR+I+G: \_\_\_

## 

## Concatenated phylogeny

To generate a concatenated phylogeny using RAxML, you will only need the concatenated and aligned matrix from PHYLUCE, mafft-nexus-trimmed-raxml.phylip, which is the same file used for Partition Finder.

### 1. Install RAxML

Install RAxML following the installation instructions by the [Exelixis Lab](https://cme.h-its.org/exelixis/web/software/raxml/cluster.html).

### 2. Run RAxML on concatenated matrix

You should already know what the best fitting model is given the results of [Partition Finder](#_f2xyrshuk8a7). With concatenated phylogenetic analyses, you only need to run the overall best fitting model and do not have to run each locus separately like you would for a pseudo-coalescent analysis (discussed below in [Pseudo-coalescent tree](#_e7d1q0apy5xf)).

1. First, make new folder in your main working folder to indicate where all your files will go:

cd /home/USER/phyluce/

mkdir concat\_raxml/

cd concat\_raxml/

1. Copy your concatenated and aligned matrix to concat\_raxml/

cp ../taxon-set-all/mafft-nexus-trimmed-raxml/mafft-nexus-trimmed-raxml.phylip .

1. Then, you want to generate a script to run RAxML on the concatenated matrix, which I named ‘concat-raxml.sh’:

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

concat-raxml.sh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The results of this are five files, somewhat resembling these names:

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

The RAxML\_bestTree.out, RAxML\_bipartitionsBranchLabels.out, and RAxML\_bootstrap.out trees can be put into [FigTree](http://tree.bio.ed.ac.uk/software/figtree/) (a free, downloadable software) to visualize the evolutionary relationships within the tree.

## Pseudo-coalescent phylogeny

### RAxML to generate gene trees

#### 1. Install RAxML

Install RAxML following the installation instructions by the [Exelixis Lab](https://cme.h-its.org/exelixis/web/software/raxml/cluster.html).

#### 2. Prepare folders/files

When running a pseudo-coalescent (hereafter referred to as “ASTRAL”) analysis, you should separate the individual loci by their most appropriate model as indicated by PartitionFinder ([Step 3](#_f2xyrshuk8a7)).

1. In your main working directory (e.g., /home/USER/phyluce/), create a new folder called ‘astral\_raxml’ (or similar).

cd /home/USER/phyluce/  
mkdir astral\_raxml

cd astral\_raxml

1. To help prepare for the next few steps, we will also generate new files within astral\_raxml/ that designate which model choices will be run (i.e., folders for GTR, GTR+G, and GTR+I+G).

mkdir GTR

mkdir GTRG

mkdir GTRIG

1. In each model choice folder, create two folders: batch\_exports/ and output/

Example:

cd GTR

mkdir batch\_exports

mkdir output

1. Then, you will want to copy all your loci files into the batch\_exports/ folder, depending on their designated model choice folder. For example, if uce-727’s best model choice was GTR, you will copy the associated file into the /home/USER/phyluce/astral\_raxml/GTR/batch\_export/ folder.

\*The files you are looking for are in the /home/USER/phyluce/taxon-set-all/mafft-nexus-trimmed-clean-genes/ folder as \*.nexus files.

Example:

cd /home/USER/phyluce/astral\_raxml/GTR/batch\_export/

cp /home/USER/phyluce/taxon-set-all/mafft-nexus-trimmed-clean-genes/uce-727.nexus .

1. Once you have all the gene files in their appropriate model choice’s batch\_export/ folder, we will then need to change the file type and ending from .nexus to .fasta. I use the tool [ElConcatenero](https://github.com/ODiogoSilva/ElConcatenero).

python /home/USER/ElConcatenero/ElConcatenero.py -c -if nexus -of fasta -in \*.nexus  
You can then remove the old nexus files in the batch\_exports/ folder

rm \*.nexus

1. Finally, you want to generate a script to run RAxML on the individual, aligned gene files, which I named script\_GTR.sh (name changes depending on model choice). This is usually a script because depending on the number of genes/taxa, this may take a few hours to a few days.

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

Example:

script\_GTR.sh  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DIR\_I=/home/USER/phyluce/astral\_raxml/GTR/batch\_exports/\*.\_\_\_

#DIR\_O=/home/USER/phyluce/astral\_raxml/GTR/output

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 /home/USER/phyluce/astral\_raxml/GTR/output

done

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. At the end of steps 1-6, your folder/file structure should resemble this:

/home/USER/phyluce/astral\_raxml/

GTR/  
 batch\_exports/

uce-727.fasta  
 … # remaining fasta gene files for GTR

output/

script\_GTR.sh

GTRG/

batch\_exports/

… # fasta gene files for GTR+G

output/

script\_GTRG.sh

GTRIG/

batch\_exports/

… # fasta gene files for GTR+I+G

output/

#### 3. Run RAxML

1. Then, run the script:

sbatch script\_GTR.sh

\*Results of RAxML will be in the output/ folders within each model choice!

### ASTRAL-III

#### 1. Install ASTRAL-III

Install ASTRAL-III (or newer) following the installation instructions on [GitHub](https://github.com/smirarab/ASTRAL).

#### 2. Prepare folders/files

Once the output files are produced from RAxML, you will need to copy the RAxML\_bestTree\* and RAxML\_bootstrap\* files into their own folders.

1. In the astral\_raxml/ folder, make new folders called bestTree/ and bootTree/.

cd /home/USER/phyluce/astral\_raxml/

mkdir bestTree

mkdir bootTree

1. Then copy all RAxML\_bestTree\* and RAxML\_bootstrap\* files into the bestTree/ and bootTree/ folders, respectively:

cp ./GTR/output/RAxML\_bootstrap\* ./bootTree/

cp ./GTR/output/RAxML\_bestTree\* ./bestTree/

\*Change GTR to GTRG or GTRIG to get all gene trees!

1. Re-record the number of loci in each folder as the number may be changed since RAxML removes loci that have 3 or less taxa!
2. Next, change the \*.fasta extension to a \*.tre extension in bestTree/ and bootTree/. Code does not need to be changed in the loops unless the file ending (e.g., .fasta) differs:

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

1. Now concatenate all the bestTree/\*.tre files within their designated folders using the command cat.

cd bestTree

cat RAxML\_bestTree\* > concat\_best.tre

1. Next, make a new folder called Astral\_try/ in the astral\_raxml/ folder and copy the concat\_best.tre file into Astral\_try/.

cd /home/USER/phyluce/astral\_raxml/

mkdir Astral\_try

cd Astral\_try

cp ../bestTree/concat\_best.tre .

1. (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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/home/USER/phyluce/astral\_raxml/bootTree/RAxML\_bootstrap.uce-1000.tre

/home/USER/phyluce/astral\_raxml/bootTree/RAxML\_bootstrap.uce-1005.tre

/home/USER/phyluce/astral\_raxml/bootTree/RAxML\_bootstrap.uce-1007.tre

/home/USER/phyluce/astral\_raxml/bootTree/RAxML\_bootstrap.uce-1008.tre

.... # remaining bootstrap file locations

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Some code to do that:

cd /home/USER/phyluce/astral\_raxml/bootTree/

ls \*.tre > bs-files

pwd #copy the directory!

sed -i 's\RAxML\/home/USER/phyluce/astral\_raxml/bootTree/RAxML\g' bs-files #paste your working directory where it is highlighted!

mv bs-files ../Astral-try/

#### 3. Run ASTRAL-III

Now we are ready to run ASTRAL-III. To do this, you just need a script (run\_astral\_script.sh [below]), the concatenated matrix (concat\_best.tre), and (optional) bs-files.

1. Create the script.

run\_astral\_script.sh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Run ASTRAL

java -jar /home/USER/ASTRAL/astral.5.7.3.jar -i concat\_best.tre -o Astral\_best.tre 2> astral\_best.log

#Generate LPP values

java -jar /home/USER/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 /home/USER/ASTRAL/astral.5.7.3.jar -i concat\_best.tre -b bs-files -o Astral\_bootcount.tre 2> astral\_bootcount.log

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Run the script!

sbatch run\_astral\_script.sh

1. The output of this script will be \*.log and \*.tre files. The Astral\_lpp.tre is the final tree with LPP support values. This final tree can be put into [FigTree](http://tree.bio.ed.ac.uk/software/figtree/) (a free, downloadable software) to visualize the evolutionary relationships within the tree.

# 

# **Check for discordance**

## PhyParts

[PhyParts](https://bmcecolevol.biomedcentral.com/articles/10.1186/s12862-015-0423-0), along with [PhypartsPieCharts](#_9tjp8s54hy3m), 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.

PhyParts only needs a rooted species tree and rooted gene trees in newick format.

### 1. Installation

General installation instructions can be found [here](https://bitbucket.org/blackrim/phyparts/src/master/). However, code is below to make a conda environment called ‘phyparts’.

conda create -n phyparts python=2.7

pip install ete3

pip install matplotlib

Additionally, we will need to clone the GitHub to obtain the python scripts to run PhyParts.

cd ~ #go to home directory

git clone <https://bitbucket.org/blackrim/phyparts.git>

### 2. Data preparation

[Phyx](https://github.com/FePhyFoFum/phyx) 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.

1. First, we need to make a new folder.

cd /home/USER/phyluce/

mkdir PhyParts

cd PhyParts

1. To run pxrr on the species tree, you first need to copy over the species tree from ASTRAL-III

cp ../astral\_raxml/Astral\_best.tre .

1. Then, you can run the below script but designate which taxon/taxa is/are the outgroups and root from there

pxrr -t Astral\_best.tre -g outgroup\_taxon > Astral\_best\_rooted.tre #for a single taxon rooting

pxrr -t Astral\_best.tre -g outgroup\_taxon1 outgroup\_taxon2 outgroup\_taxon3 … > Astral\_best\_rooted.tre #for rooting to multiple taxa

The output should now be your tree that is rooted to the designated outgroup!

1. To run pxrr on the gene trees, we will first need to copy over the gene trees from RAxML into a new folder called ‘root’.

mkdir root

cp ../astral\_raxml/bestTree/\*.tre ./root/

1. Then, you can run pxrr on the gene trees in a loop, changing the outgroup\_taxon to your designated outgroup.

for t in \*.tre

do

pxrr -t $t -g outgroup\_taxon > $t.rooted.tre

done

1. Note: 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

Now, you should have a rooted species tree (Astral\_best\_rooted.tre) and rooted gene trees in a new folder (./root/$t.rooted.tre) in your new folder ‘PhyParts’! We are now ready to run PhyParts!

### 3. Run PhyParts

Now we are ready to run PhyParts. To do this, you just need a script (phyparts.sh [below]), rooted species tree (Astral\_best\_rooted.tre) and a folder containing rooted gene trees (./root/).

1. Create the script.

phyparts.sh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note: the flags here are default values that can change depending on your project. Additionally, there are other flags you can add. Please refer to the [manual](https://bitbucket.org/blackrim/phyparts/src/master/) to get more information there.

1. Run the script!

sbatch phyparts.sh

1. The output of this script will be multiple files that start with ‘out’, unless specified otherwise:
   * 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
2. Each file has great information about the concordant/discordant nodes as indicated by PhyParts (detailed [here](https://bitbucket.org/blackrim/phyparts/src/master/)).

### 4. Run PhyPartsPieCharts

The results of PhyParts are not very figure ready. Fortunately, Matt Johnson (Texas Tech University) has generated a [visualization script](https://github.com/mossmatters/phyloscripts/tree/master/phypartspiecharts), called ‘phypartspiecharts.py’, that creates an svg image that can be used in manuscripts.

1. First, make sure you are in your phyparts conda environment

conda activate phyparts

1. Download the python script from GitHub. Personally, I had issues with git cloning the script, so I simply copied and pasted the script from [here](https://github.com/mossmatters/MJPythonNotebooks/blob/master/phypartspiecharts.py).

nano phypartspiecharts.py  
#paste the script from GitHub

1. Next, you will want to combine all your outfiles. You can do this using ‘cat’.

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

1. Now we are ready to run phypartspiecharts.py. To do this, 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).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/home/USER/phyluce/PhyParts/phypartspiecharts.py /home/USER/phyluce/PhyParts/Astral\_best-rooted.newick phyparts 1000 --svg\_name phyparts\_piechart.svg

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note: the flags here are default values that can change depending on your project. Additionally, there are other flags you can add. Please refer to [GitHub](https://github.com/mossmatters/phyloscripts/tree/master/phypartspiecharts) for additional information.

1. The result should be an svg file that contains 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

## 

## Quartet Sampling

[Quartet Sampling](http://dx.doi.org/10.1002/ajb2.1016) (QS) 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).

Quartet Sampling only needs an alignment file and a species tree without support values (suggested). Manual with additional information can be found [here](https://github.com/FePhyFoFum/quartetsampling/blob/master/quartetsampling.pdf).

### 1. Installation

General installation instructions can be found [here](https://github.com/FePhyFoFum/quartetsampling/blob/master/quartetsampling.pdf). However, code is below to make a conda environment called ‘quartetsampling’

conda create -n quartetsampling numpy

conda activate quartetsampling

conda install -c bioconda python

conda install -c bioconda raxml-ng #raxml-ng is suggested in the manual

conda install -c bioconda raxml #I included regular raxml as well.

Additionally, we will need to clone the GitHub to obtain the python scripts to run Quartet Sampling.

cd ~ #go to home directory

git clone <https://github.com/FePhyFoFum/quartetsampling.git>

### 2. Data preparation

Quartet Sampling only needs a species tree in newick format and an alignment file in Relaxed Phylip format. We already have an alignment file from PHYLUCE (mafft-nexus-trimmed-raxml.phylip), so we can simply copy that to a new folder that we will call ‘QuartetSampling’.

cd /home/USER/phyluce/

mkdir QuartetSampling

cd QuartetSampling

cp ../taxon-set-all/mafft-nexus-trimmed-raxml/mafft-nexus-trimmed-raxml.phylip .

Next, we will need our species tree. The manual suggests that support scores be removed from the tree prior to running Quartet Sampling. Fortunately, when we ran ASTRAL-III, we generated a species tree without support values (Astral\_best.tre), so we can just copy that over to ‘QuartetSampling’.

cp ../astral\_raxml/Astral\_try/Astral\_best.tre .

Now, you should have an alignment file (mafft-nexus-trimmed-raxml.phylip) and a species tree (Astral\_best.tre) in your new folder ‘QuartetSampling’! We are now ready to run Quartet Sampling!

### 3. Run Quartet Sampling

Now we are ready to run Quartet Sampling. To do this, you just need a script (quartetsampling.sh [below]), the alignment file (mafft-nexus-trimmed-raxml.phylip) and a species tree (Astral\_best.tre)

1. Create the script.

quartetsampling.sh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

python /home/USER/quartetsampling/pysrc/quartet\_sampling.py --tree /home/USER/phyluce/QuartetSampling/Astral\_best.tre --align /home/USER/phyluce/QuartetSampling/mafft-nexus-trimmed-raxml.phylip --reps 100 --threads 4 --lnlike 2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note: the flags here are default values that can change depending on your project. Additionally, there are other flags you can add. Please refer to the [manual](https://github.com/FePhyFoFum/quartetsampling/blob/master/quartetsampling.pdf) to get more information there.

1. Run the script!

sbatch quartetsampling.sh

1. The output of this script will be multiple files that start with ‘RESULT’, unless specified otherwise:
   * 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
2. 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](http://tree.bio.ed.ac.uk/software/figtree/) (a free, downloadable software) to visualize discordance in the tree.

# 

# **WGDv2 - Ks plots**

[wgd v2](https://doi.org/10.1093/bioinformatics/btae272) is a package that can be used to infer and date ancient whole-genome duplication (WGD) events. Even so, we will be using wgd v2 exclusively to generate Ks plots to determine the evolutionary age of WGD events across our 69 transcriptome species.

## 1. Prepare transcriptomic data

Prior to running wgd v2, we need to identify candidate coding regions within our transcriptome data using the tool [TransDecoder](https://github.com/TransDecoder/TransDecoder/wiki).

### 1. Installation

General installation instructions for TransDecoder can be found [here](https://github.com/TransDecoder/TransDecoder/wiki). However, some code is below on how to install TransDecoder with singularity:

cd ~

mkdir TransDecoder

cd TransDecoder

wget https://data.broadinstitute.org/Trinity/CTAT\_SINGULARITY/MISC/TransDecoder/transdecoder.v5.7.1.simg

NOTE: To run the commands with singularity:

singularity exec -e ~/TransDecoder/transdecoder.v5.7.1.simg TransDecoder.Predict …

### 2. Run TransDecoder

1. Make a new working directory

cd ~

mkdir transdecoder

cd transdecoder

1. Upload or copy the transcriptome data files to your new folder, ‘transdecoder’.
2. TransDecoder can be run as a loop using a namelist.txt if you have a lot of samples:

namelist.txt

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Aci\_ML\_Trinity  
Ainsliaea\_aptera\_SRR12832696

…

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Run TransDecoder as a loop to increase efficiency:

while read name;

do

singularity exec -e ~/TransDecoder/transdecoder.v5.7.1.simg TransDecoder.LongOrfs -t /home/USER/transdecoder/”$name”.fasta --output\_dir /home/USER/transdecoder/output/;

singularity exec -e ~/TransDecoder/transdecoder.v5.7.1.simg TransDecoder.Predict -t /home/USER/transdecoder/”$name”.fasta --output\_dir /home/USER/transdecoder/output/;

done < namelist.txt

Results can be found in the ‘output’ folder. ‘output’ will contain a folder (ending with ‘\_dir’) containing multiple files, including \*.bed, \*.cds, \*.gff3, and \*.pep files.

## 2. Generate Ks plots using wgd v2

### 1. Installation

We will need to clone the GitHub page prior to installation.

cd ~ #go to home directory

git clone <https://github.com/heche-psb/wgd>

cd wgd

General installation instructions can be found [here](https://github.com/heche-psb/wgd). However, some code is below to make a conda environment called ‘wgdv2’.

conda create -n wgdv2 python=3.8 #tried python 3.6 but it did not work for me

conda activate wgdv2

pip install numpy==1.19.0

pip install wgd==2.0.38

pip install .

conda install diamond #future dependencies

conda install bioconda::mafft #future dependencies

conda install bioconda::mcl #future dependencies

### 2. Run wgd v2

For the purposes of what we need to use wgd v2 for, 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.

1. Make a new working directory

cd ~

mkdir wgdv2

cd wgdv2

1. Copy over the \*.cds files from Transdecoder

cp ../transdecoder/\*.cds .

1. Then, you can run wgd v2 on each individual sample. For example, the code for ‘Aci\_ML\_Trinity’ would be:

wgd dmd Aci\_ML\_Trinity.fasta.transdecoder.cds -o wgd\_dmd

wgd ksd wgd\_dmd/Aci\_ML\_Trinity.fasta.transdecoder.cds.tsv Aci\_ML\_Trinity.fasta.transdecoder.cds -o wgd\_ksd

wgd viz -d wgd\_ksd/Aci\_ML\_Trinity.fasta.transdecoder.cds.tsv.ks.tsv -o wgd\_ELMM

1. Alternatively, the code can be run as a loop using a namelist.txt if you have a lot of samples:

namelist.txt

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Aci\_ML\_Trinity  
Ainsliaea\_aptera\_SRR12832696

…

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Run wgd v2:

while read name;

do

wgd dmd “$name”.fasta.transdecoder.cds -o wgd\_dmd;

wgd ksd wgd\_dmd/“$name”.fasta.transdecoder.cds.tsv “$name”.fasta.transdecoder.cds -o wgd\_ksd;

wgd viz -d wgd\_ksd/“$name”.fasta.transdecoder.cds.tsv.ks.tsv -o wgd\_ELMM;

done < namelist.txt

1. Final results will be in the wgd\_ELMM folder.

/home/USER/wgdv2/

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

\*\*Please refer to [GitHub](https://github.com/heche-psb/wgd) for a full explanation on what each file is.

1. The main results of our study are the elmm\*\_best\_models\_node\_averaged.pdf/svg files.

### 3. Visualize the results in R

# 

# **Investigate ancient hybridizations**

## PhyloNetworks

\_\_

### 1. Installation

To run PhyloNetworks, you will need to have [Julia](https://docs.julialang.org/en/v1/) installed. General installation instructions can be found [here](https://julialang.org/downloads/).

Example code to get julia-1.8.3:

wget <https://julialang-s3.julialang.org/bin/linux/x64/1.8/julia-1.8.3-linux-x86_64.tar.gz>

tar zxvf julia-1.8.3-linux-x86\_64.tar.gz

To activate julia:

/home/USER/julia-1.8.3/bin/julia #After this, the Julia prompt should pop up.

### 2. Data preparation

To run PhyloNetworks, you need a gene tree matrix and a species tree. We have already generated those through running ASTRAL-III, so we can make a new folder called ‘phylonetworks’ and copy those files over.

cd ~

mkdir phylonetworks

cd phylonetworks

cp ../phyluce/astral\_raxml/Astral\_try/Astral\_lpp.tre . #species tree

cp ../phyluce/astral\_raxml/Astral\_try/concat\_best.tre . #concatenated gene tree matrix

#### (Optional) Prune species and gene tree(s)

PhyloNetworks is computationally very intense. Given this, it is often suggested to subset your species and gene trees into smaller trees and run those individually if you have a large tree, such as in our case (n = 262 taxa).

To do this, we use [Phyx](https://github.com/FePhyFoFum/phyx), which was already introduced in the [PhyParts section](#_5y7nmcoyl6eu) above. For the purposes of subsetting our data to run PhyloNetworks, we only need to use the ‘pruning trees’ function, pxrmt.

##### Prune the species tree

1. To prune the species tree (Astral\_lpp.tre), you will want to run Phyx. If you have only a few taxa to prune, you can simply list the sequence names (e.g., taxon1, taxon2, etc.) in the command line as a comma-delimited list with the ‘-n’ flag. For example:

pxrmt -t Astral\_lpp.tre -n taxon1,taxon2,taxon3 > Astral\_lpp\_pruned.tre

If you have a lot of taxa to remove, you can also create a file with the sequence names that you want to prune, with one sequence name per line with the flag ‘-f’. For example:

to\_remove.txt:

\_\_\_\_\_

taxon1

taxon2

taxon3

…

\_\_\_\_\_\_

Then run Phyx with the file:

pxrmt -t Astral\_lpp.tre -f to\_remove.txt > Astral\_lpp\_pruned.tre

1. The result of this should be a new species tree file called Astral\_lpp\_pruned.tre that only contains the taxa you want to run PhyloNetworks on!

##### Prune the underlying gene trees

You will use the above commands also on the gene trees.

1. To prune the underlying gene trees, you will need to first copy over the gene trees from your ASTRAL-III analysis into a new folder.

cd /home/USER/phylonetworks/tree\_pruning

cp -r ../phyluce/astral\_raxml/bestTree .

cd bestTree

1. Now, we will want to run Phyx on all of the gene trees. The code is very similar to pruning the species tree (above), but differs in that we will instead make a loop for efficiency. For example, if we wanted to prune the gene trees with a file, we can run:

for file in \*.tre;

do

pxrmt -t "$file" -f toremove.txt > "$file".pruned.tre

done

**NOTE:** Your number of gene trees may decrease as some gene trees may only contain the pruned taxa, or by pruning the gene tree, the phylogeny only has two tips. This is normal, and it is good to record which trees were removed.

1. Then, I would create a new folder to move these new, pruned trees to.

cd ..

mkdir bestTree\_pruned

mv bestTree/\*.pruned.tre bestTree\_pruned

cd bestTree\_pruned

1. From there, you want to concatenate all the gene trees into a matrix with the ‘cat’ function.

cat \*.pruned.tre > concat\_best\_pruned.tre

1. Then, move the concat\_best\_pruned.tre file into your main PhyloNetworks folder.

mv concat\_best\_pruned.tre ../

1. The result of this loop should be a new concatenated gene tree file with only the taxa you want to run PhyloNetworks on!

Now, going forward, you should have a pruned species tree (Astral\_lpp\_pruned.tre) and gene tree matrix (concat\_best\_pruned.tre)!

### 3. Run PhyloNetworks

Using either the [full trees](#_xfubfhuwe0mk) (Astral\_lpp.tre and concat\_best.tre) or the [pruned trees](#_9khpw6u0ne6c) (Astral\_lpp\_pruned.tre and concat\_best\_pruned.tre), we can now run PhyloNetworks. PhyloNetworks will be run in two parts: 1) running the analysis without bootstraps to find the optimal number of hybridization events ([Step 1](#_3hnsk0127c15) below), and 2) running the analysis with bootstraps only with the optimal number of hybridization events ([Step 3](#_ep2nqg4hl7iw) below).

#### 1. With no bootstraps

1. Activate julia:

/home/USER/julia-1.8.3/bin/julia #After this, the Julia prompt should pop up.

1. Set working directory to where your species tree and gene tree matrix is:

cd("/home/USER/phylonetworks/")

1. Add required packages (if applicable):

using Pkg

Pkg.add("PhyloNetworks")

Pkg.add("RCall")

Pkg.add("PhyloPlots")

Pkg.add("CSV")

Pkg.add("DataFrames")

Pkg.add("StatsModels")

Pkg.add("Gadfly")

Pkg.add("Plots")

1. Open packages:

using PhyloNetworks

using PhyloPlots

using CSV

using DataFrames

using StatsModels

using Gadfly

using RCall

using Plots

1. Load your gene tree matrix and species tree (using pruned data as input for example)

genetrees = readMultiTopology("concat\_best\_pruned.tre")

q,t = countquartetsintrees(genetrees);

df = writeTableCF(q,t)

CSV.write("tableCF.csv", df);

raxmlCF = readTableCF("tableCF.csv")

astraltree = readMultiTopology("Astral\_lpp\_pruned.tre")

T=readTopologyLevel1("Astral\_lpp\_pruned.tre")

1. Start with 0 to 10 potential hybridizations on your initial PhyloNetworks run.

net0 = snaq!(astraltree[1], raxmlCF, hmax=0, filename="net0", seed=1234)

net1 = snaq!(net0, raxmlCF, hmax=1, filename="net1", seed=2345)

net2 = snaq!(net1, raxmlCF, hmax=2, filename="net2", seed=3456)

net3 = snaq!(net2, raxmlCF, hmax=3, filename="net3", seed=4567)

net4 = snaq!(net3, raxmlCF, hmax=4, filename="net4", seed=5678)

net5 = snaq!(net4, raxmlCF, hmax=5, filename="net5", seed=6789)

net6 = snaq!(net5, raxmlCF, hmax=6, filename="net6", seed=6789)

net7 = snaq!(net6, raxmlCF, hmax=7, filename="net7", seed=6789)

net8 = snaq!(net7, raxmlCF, hmax=8, filename="net8", seed=6789)

net9 = snaq!(net8, raxmlCF, hmax=9, filename="net9", seed=6789)

net10 = snaq!(net9, raxmlCF, hmax=10, filename="net10", seed=6789)

1. Results of each run will create an \*.err, \*.log, and \*.out file designated with the name under “filename” (e.g., net0.err, etc.).
2. Typically, this can take a little bit, so it is advised to run as a script!

#### 2. Determine best number of hybridizations

Once the no bootstrap runs are done, we then want to find the optimal number of hybridizations. To do this, you want to look for the lowest log-likelihood (-loglik) score, found in the \*.out files. A common way to discover the lowest -loglik is by plotting them. You can do this easily with julia.

1. First, reload your \*.out files (if applicable)

net0 = readTopology("net0.out")

net1 = readTopology("net1.out")

net2 = readTopology("net2.out")

net3 = readTopology("net3.out")

net4 = readTopology("net4.out")

net5 = readTopology("net5.out")

1. Then, make a list of the scores

scores = [net0.loglik, net1.loglik, net2.loglik, net3.loglik, net4.loglik, net5.loglik]

1. Finally, plot “scores” and save as a pdf

R"pdf"("score-vs-h.pdf", width=4, height=4);

R"plot"(x=0:5, y=scores, type="b", xlab="number of hybridizations h", ylab="network score");

R"dev.off"();

1. From there, you can open the pdf file, ‘score-vs-h.pdf’, and find the lowest value and consider that h-value the optimal number of hybridizations for your data. If your data plateaus, then you want to choose the first value at the plateau.

#### 3. With bootstraps

Once you have decided on the best number of hybridizations without bootstraps, you will then run PhyloNetworks with bootstraps. This is the computationally limiting step and can take days to weeks to months to finish. Given this, I always run it as a script.

1. Before running the script, you need a bootstrap file, similar to the one run with ASTRAL-III (bs-files). If you already have that file, you can simply copy it over to the ‘phylonetworks’ folder. If not, please follow the steps from [ASTRAL-III](#_e4ddm77i8qpj).
2. Once you have that file, you can run PhyloNetworks with bootstraps as a script. To run julia text as a script, it requires two files. First, a julia script containing the code needed to run julia/PhyloNetworks. Next, a bash script that will run the julia script.
3. Example julia script with h = 2 as optimal number of hybridization events:

scriptBS.jl

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#!/home/USER/julia-1.8.3/bin/julia

cd("/home/USER/phylonetworks/")

using PhyloNetworks

using PhyloPlots

using CSV

using DataFrames

using StatsModels

using Gadfly

using RCall

using Plots

net2 = readTopology("net2.out")

bootTrees = readBootstrapTrees("bs-files")

bootnet2 = bootsnaq(net2, bootTrees, hmax=2, nrep=100, runs=10, filename="bootsnaq2\_raxmlboot", seed=2345)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Example bash script to run scriptBS.jl:

phylonetworksBS.sh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#!/bin/bash

… #whatever input you use for bash scripts

/home/USER/julia-1.8.3/bin/julia scriptBS.jl

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Run PhyloNetworks as a bash script!

sbatch phylonetworksBS.sh

### 4. Visualize networks

1. There are multiple ways to visualize your networks. One example is using the PhyloPlots tool:

PhyloPlots.plot(bootnet2, showgamma=true);

1. Alternatively, you can look at the plots in [Dendroscope](https://uni-tuebingen.de/fakultaeten/mathematisch-naturwissenschaftliche-fakultaet/fachbereiche/informatik/lehrstuehle/algorithms-in-bioinformatics/software/dendroscope/). To do this, you need to first create dendro-coded trees with julia:

writeTopology(bootnet2,di=true)

1. Then, you will open Dendroscope, click “File” in the top left, go down to “Enter Trees or Networks…”, and then copy and paste the output from writeTopology there.

# **Plastomes**

## 

# **References**

# *To include as an example file on GitHub*

* cos\_probes.fasta (PHYLUCE)
* datasets.conf (PHYLUCE)
* partition\_finder.cfg (PartitionFinder)
* Pdf with -loglik values (PhyloNetworks)
* Pdf of PhyloPlots output (PhyloNetworks)
* Pdf of Dendroscope output (PhyloNetworks)