Burrows Bay Octopus Phylogeny

Kirt Onthank

12/7/2020

Libraries

```
library(ape)
library(xlsx)
library(insect)
library(aphid)
library(DECIPHER)
```

```
## Loading required package: Biostrings
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following object is masked from 'package:insect':
##
##     expand
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
```

```
## Loading required package: IRanges
```

```
##
## Attaching package: 'IRanges'
```

```
## The following object is masked from 'package:insect':
##
##     trim
```

```
## Loading required package: XVector
```

```
## Loading required package: GenomeInfoDb
```

```
##
## Attaching package: 'Biostrings'
```

```
## The following object is masked from 'package:ape':
##
##     complement
```

```
## The following object is masked from 'package:base':
##
##     strsplit
```

```
## Loading required package: RSQLite
```

```
library(magrittr)
library(phangorn)
library(rwty)
```

```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(stringi)
library(Biostrings)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:Biostrings':
##
##     collapse, intersect, setdiff, setequal, union
```

```
## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect
```

```
## The following object is masked from 'package:XVector':
##
##     slice
```

```
## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union
```

```
## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

This function to convert DNAbin objects used by the ape package to DNAStringSet objects used by the DECIPHER package was written by Joel Nitta, available on his github (https://gist.github.com/joelnitta/6f30a7c0f1c83d78c76a5469e935d56f)

```
# Convert ape::DNAbin format to Biostrings::DNAStringSet format,
# optionally removing gaps
DNAbin_to_DNAstringset <- function (seqs, remove_gaps = TRUE) {
  if(isTRUE(remove_gaps)) {
  seqs %>% as.list() %>% as.character %>%
      lapply(.,paste0,collapse="") %>%
      lapply( function (x) gsub("-", "", x)) %>%
      unlist %>% Biostrings::DNAStringSet()
  } else {
    seqs %>% as.list() %>% as.character %>%
      lapply(.,paste0,collapse="") %>%
      unlist %>% Biostrings::DNAStringSet()
  }
}
```

Preparing the dataset

Reading in accession numbers

```
access=read.xlsx("Enteroctopodidae_and_Outgroup.xlsx",sheetIndex = 1)
#preaccess=preaccess[complete.cases(preaccess[,1]),1:6]
```

Here I am paring down the data...

```
#access=preaccess[preaccess$two_genes=="y"&preaccess$solo_rep=="y",]
```

I use ape's read.GenBank command to go out and get the specific sequences.

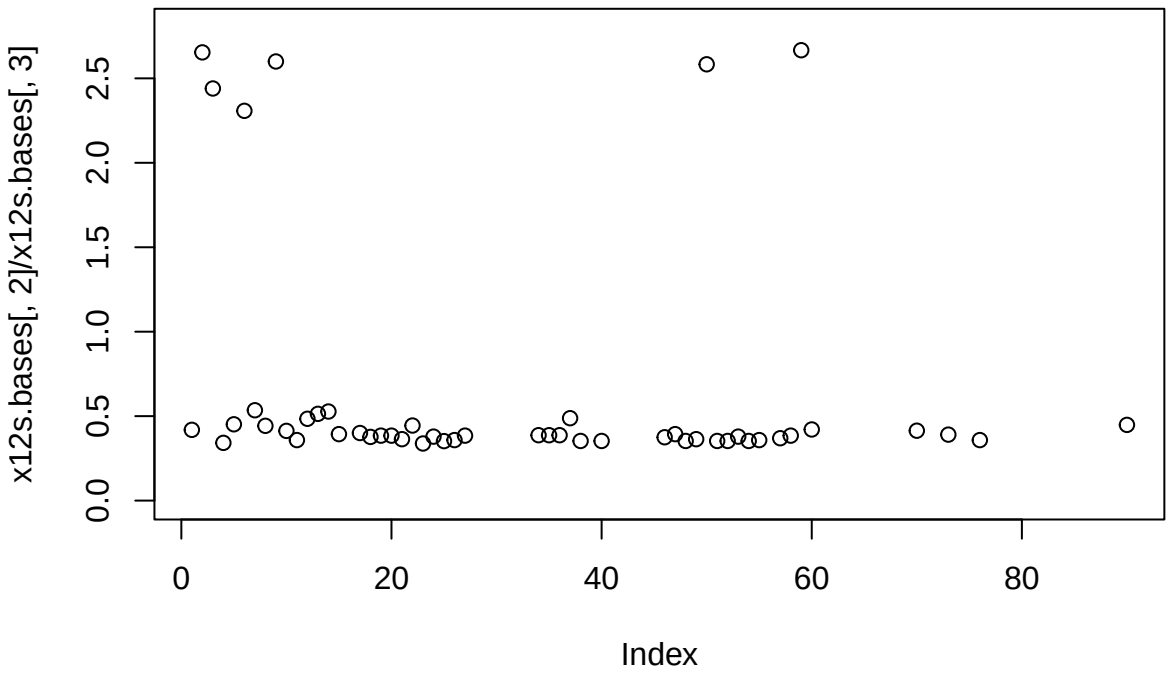
```
x12s.accession=access$X12s
x12s.accession[is.na(x12s.accession)]="DJ078208"
x12s=read.GenBank(x12s.accession)
x12s[names(x12s)=="DJ078208"]=as.DNAbin("-")
names(x12s)=access$NA.
```

As is the case with with a lot of the sequences you get from Genbank, all of the sequences are not same strand, and the reverse complement will need to be taken for those sequences. To determine which sequences

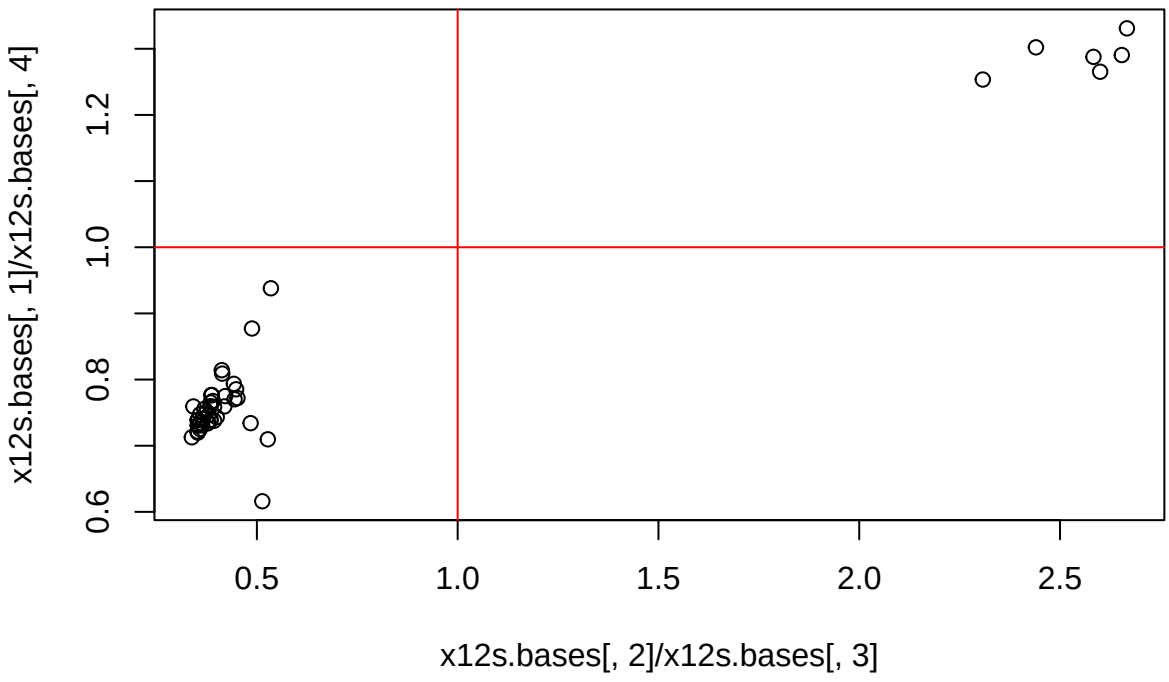
need this done, I am looking at the pattern of base frequencies for each sequence. I probably could have built the object with some apply family function instead of a for loop if I knew that group of functions better...

```
x12s.bases=base.freq(x12s[1])
for (i in 2:length(x12s)){
  x12s.bases=rbind(x12s.bases,base.freq(x12s[i]))
}
rownames(x12s.bases)=names(x12s)
```

```
plot(x12s.bases[,2]/x12s.bases[,3],ylim=c(0,2.8))
```



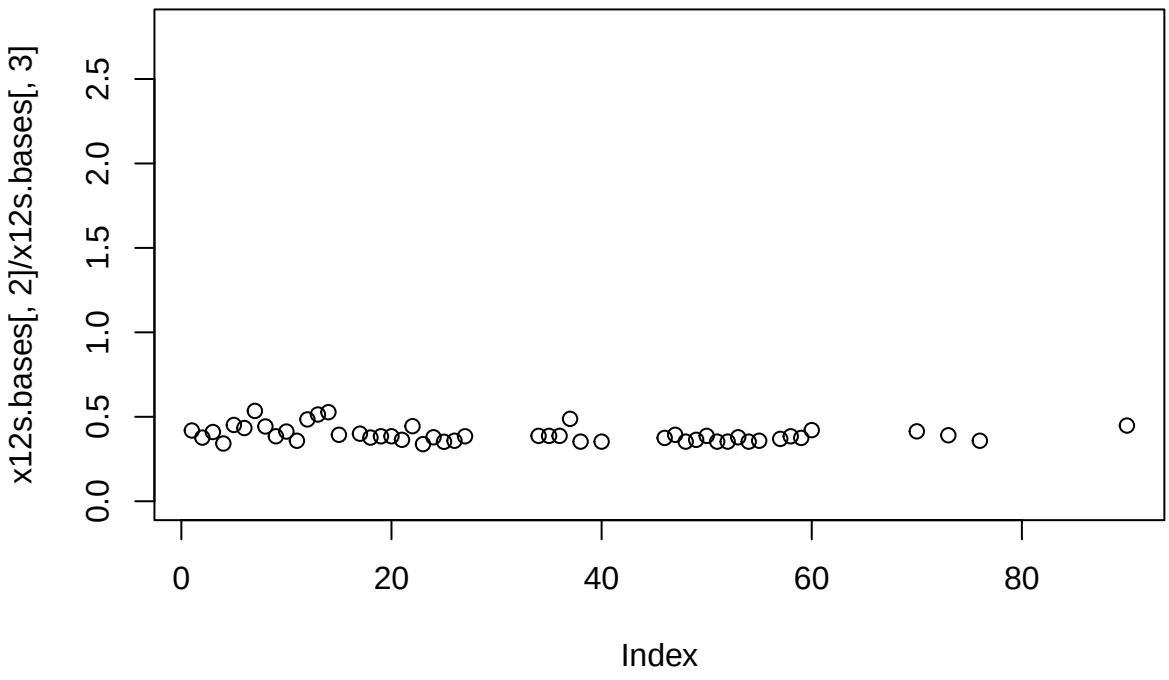

```
plot(x12s.bases[,2]/x12s.bases[,3],x12s.bases[,1]/x12s.bases[,4])
abline(h=1,col="red")
abline(v=1,col="red")
```




```
revcomp=which(x12s.bases[,2]/x12s.bases[,3]>=1)
```

```
for (i in revcomp){
  x12s[i]=ape::complement(x12s[i])
}
```

```
x12s.bases=base.freq(x12s[1])
for (i in 2:length(x12s)){
  x12s.bases=rbind(x12s.bases,base.freq(x12s[i]))
}
rownames(x12s.bases)=names(x12s)
plot(x12s.bases[,2]/x12s.bases[,3],ylim=c(0,2.8))
```




```
write.FASTA(x12s[!is.na(access$X12s)],"12s.fasta")
#write.FASTA(x12s,"12s.fasta")
```

```
mkdir 12s
cp 12s.fasta 12s
cp dotbracket2indexPairs.pl 12s
```

```
## mkdir: cannot create directory '12s': File exists
```

```
cd 12s
#mlocarna --probabilistic --consistency-transformation --cpus=20 --stockholm --write-structure 12s.fast
mlocarna --free-endgaps --keep-sequence-order --cpus=20 --stockholm --write-structure 12s.fasta > LocAR
#reliability-profile.pl 12s.out
```

```
cd 12s
############################
# step 1: extraction of RNAalifold consensus structure
############################

cat LocARNA.output | awk 'BEGIN{p=0;}{if(p!=0){printf$1;if(NF>1){p=0;}}else{if($1=="alifold"){p=1;printf

############################
# step 1: convert dot-bracket to index information
############################

perl dotbracket2indexPairs.pl `cat LocARNA.RNAalifold.consensus` > LocARNA.RNAalifold.consensus.bp
```

```
x12s.txt=readLines("12s/12s.out/results/result.aln")
sp=length(x12s[!is.na(access$X12s)])
txt=length(x12s.txt)
spaces=max(nchar(names(x12s[!is.na(access$X12s)])))+1
string=paste(paste(rep(" ",spaces),sep="",collapse=""),"*",sep="")
rep.lines=seq(from=sp+4,to=txt,by=sp+1)
x12s.txt[rep.lines]=string
x12s.txt=gsub("U","T",x12s.txt)
writeLines(x12s.txt,"12s_result_mod.aln")
```

```
x12s.align=read.dna("12s_result_mod.aln",format="clustal")
```

```
sed -i -e '$a\' ./12s/LocARNA.RNAalifold.consensus.bp
```

```
base.pairs.12s=readLines("12s/LocARNA.RNAalifold.consensus.bp")
base.pairs.12s=gsub("^ ","",base.pairs.12s)
stems.12s=gsub(":",",",base.pairs.12s)
stems.12s=gsub(" ",",",stems.12s)
writeLines(stems.12s,"basepairs_12s.csv")
stems.12s=as.vector(t(read.csv("basepairs_12s.csv",header=F)[1,]))
stems.12s=sort(stems.12s)
```

```
stems.12s.align=x12s.align[,stems.12s]
stems.12s.align=DNAbin_to_DNAstringset(stems.12s.align)
stems.12s.align=AlignSeqs(stems.12s.align)
```

```
## Determining distance matrix based on shared 9-mers:
## ================================================================================
##
## Time difference of 0.01 secs
##
## Clustering into groups by similarity:
## ================================================================================
##
## Time difference of 0.02 secs
##
## Aligning Sequences:
## ================================================================================
##
## Time difference of 0.31 secs
##
## Iteration 1 of 2:
##
## Determining distance matrix based on alignment:
## ================================================================================
##
## Time difference of 0 secs
##
## Reclustering into groups by similarity:
## ================================================================================
##
## Time difference of 0.02 secs
##
## Realigning Sequences:
## ================================================================================
```

```
##
## Time difference of 0.25 secs
##
## Iteration 2 of 2:
##
## Determining distance matrix based on alignment:
## ================================================================================
##
## Time difference of 0 secs
##
## Reclustering into groups by similarity:
## ================================================================================
##
## Time difference of 0.02 secs
##
## Realigning Sequences:
## ================================================================================
##
## Time difference of 0.03 secs
```

```
stems.12s.align=as.DNAbin(stems.12s.align)
```

```
loops.12s=1:length(x12s.align[1,])
loops.12s=loops.12s[-stems.12s]
loops.12s.align=x12s.align[,loops.12s]
loops.12s.align=DNAbin_to_DNAstringset(loops.12s.align)
loops.12s.align=AlignSeqs(loops.12s.align)
```

```
## Determining distance matrix based on shared 9-mers:
## ================================================================================
##
## Time difference of 0.01 secs
##
## Clustering into groups by similarity:
## ================================================================================
##
## Time difference of 0.02 secs
##
## Aligning Sequences:
## ================================================================================
##
## Time difference of 0.31 secs
##
## Iteration 1 of 2:
##
## Determining distance matrix based on alignment:
## ================================================================================
##
## Time difference of 0 secs
##
## Reclustering into groups by similarity:
## ================================================================================
##
## Time difference of 0.01 secs
##
```

```
## Realigning Sequences:
## ================================================================================
##
## Time difference of 0.18 secs
##
## Iteration 2 of 2:
##
## Determining distance matrix based on alignment:
## ================================================================================
##
## Time difference of 0 secs
##
## Reclustering into groups by similarity:
## ================================================================================
##
## Time difference of 0.01 secs
##
## Realigning Sequences:
## ================================================================================
##
## Time difference of 0.08 secs
##
## Refining the alignment:
## ================================================================================
##
## Time difference of 0.03 secs
```

```
loops.12s.align=as.DNAbin(loops.12s.align)
```

```
x12s.stems.dist=dist.dna(stems.12s.align)
x12s.stems.nj=NJ(x12s.stems.dist)
x12s.stems.pd=as.phyDat(stems.12s.align)
x12s.stems.mT=modelTest(x12s.stems.pd,x12s.stems.nj)
```

```
## [1] "JC+I"
## [1] "JC+G"
## [1] "JC+G+I"
## [1] "F81+I"
## [1] "F81+G"
## [1] "F81+G+I"
## [1] "K80+I"
## [1] "K80+G"
## [1] "K80+G+I"
## [1] "HKY+I"
## [1] "HKY+G"
## [1] "HKY+G+I"
## [1] "SYM+I"
## [1] "SYM+G"
## [1] "SYM+G+I"
## [1] "GTR+I"
## [1] "GTR+G"
## [1] "GTR+G+I"
```

```
x12s.stems.mT=x12s.stems.mT[order(x12s.stems.mT$AICc),]
x12s.stems.mT$Model[1]
```

```
## [1] "GTR+G+I"
```

```
mt=character()
mt[1]=x12s.stems.mT$Model[1]
```

```
x12s.loops.dist=dist.dna(loops.12s.align)
x12s.loops.nj=NJ(x12s.loops.dist)
x12s.loops.pd=as.phyDat(loops.12s.align)
x12s.loops.mT=modelTest(x12s.loops.pd,x12s.loops.nj)
```

```
## [1] "JC+I"
## [1] "JC+G"
## [1] "JC+G+I"
## [1] "F81+I"
## [1] "F81+G"
## [1] "F81+G+I"
## [1] "K80+I"
## [1] "K80+G"
## [1] "K80+G+I"
## [1] "HKY+I"
## [1] "HKY+G"
## [1] "HKY+G+I"
## [1] "SYM+I"
## [1] "SYM+G"
## [1] "SYM+G+I"
## [1] "GTR+I"
## [1] "GTR+G"
## [1] "GTR+G+I"
```

```
x12s.loops.mT=x12s.loops.mT[order(x12s.loops.mT$AICc),]
x12s.loops.mT$Model[1]
```

```
## [1] "HKY+G"
```

```
mt[2]=x12s.loops.mT$Model[1]
```

```
x12s.proto=matrix(ncol=length(x12s.align[1,]),nrow=nrow(access),data="-")
rownames(x12s.proto)=access$NA.
x12s.final=as.DNAbin(x12s.proto)

for (i in 1:length(labels(x12s.align))){
  x12s.final[which(access$NA.==labels(x12s.align)[i]),]=x12s.align[i,]
}

write.nexus.data(x12s.final,"12s.nxs",charsperline = 1000)
```

```
length(x12s.final[1,])+49
```

```
## [1] 545
```

```
sed -i -E 's/(^.{545}).*/\1/g' 12s.nxs
```

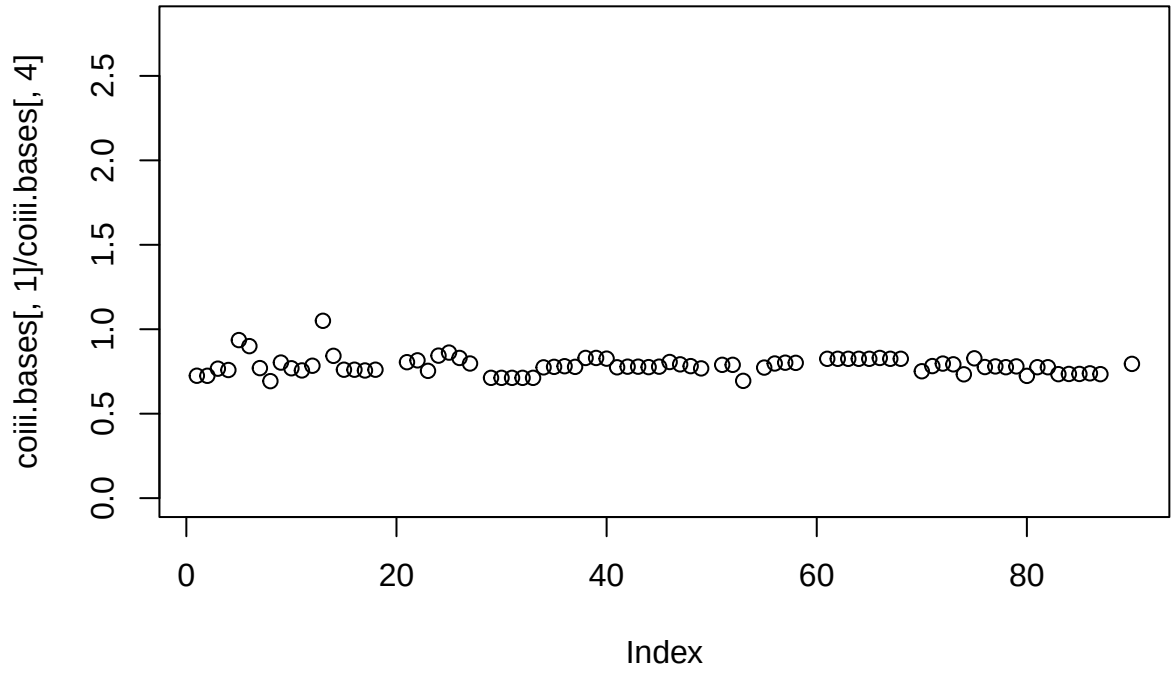
COIII

```
coiii.accession=access$COIII
coiii.accession[is.na(coiii.accession)]="DJ078208"
```

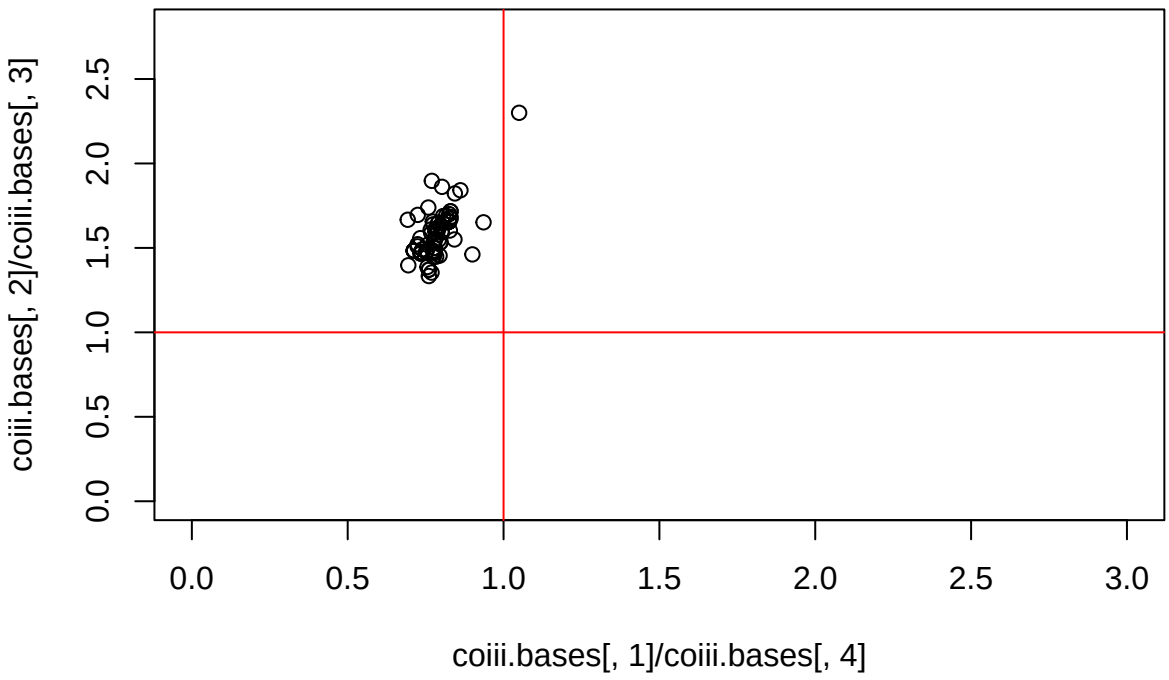
```
coiii=read.GenBank(coiii.accession)
coiii[names(coiii)=="DJ078208"]=as.DNAbin("-")
names(coiii)=access$NA.
```

```
coiii.bases=base.freq(coiii[1])
for (i in 2:length(coiii)){
  coiii.bases=rbind(coiii.bases,base.freq(coiii[i]))
}
rownames(coiii.bases)=names(coiii)
```

```
plot(coiii.bases[,1]/coiii.bases[,4],ylim=c(0,2.8))
```




```
plot(coiii.bases[,1]/coiii.bases[,4],coiii.bases[,2]/coiii.bases[,3],ylim=c(0,2.8),xlim=c(0,3))
abline(v=1,col="red")
abline(h=1,col="red")
```




```
coiii2=DNAbin_to_DNAstringset(coiii)
coiii.align=AlignSeqs(coiii2)
```

```
## Determining distance matrix based on shared 9-mers:
## ============================================================================
##
## Time difference of 0.12 secs
##
## Clustering into groups by similarity:
## ============================================================================
##
## Time difference of 0.03 secs
##
## Aligning Sequences:
## ============================================================================
##
## Time difference of 0.77 secs
##
## Iteration 1 of 2:
##
## Determining distance matrix based on alignment:
## ============================================================================
##
## Time difference of 0.01 secs
##
## Reclustering into groups by similarity:
```

```
## ================================================================================
##
## Time difference of 0.02 secs
##
## Realigning Sequences:
## ================================================================================
##
## Time difference of 0.55 secs
##
## Alignment converged - skipping remaining iteration.
##
## Refining the alignment:
## ================================================================================
##
## Time difference of 0.09 secs
```

```
coiii.stag=StaggerAlignment(coiii.align)
```

```
## Calculating distance matrix:
## ================================================================================
##
## Time difference of 0.01 secs
##
## Constructing neighbor-joining tree:
## ================================================================================
##
## Time difference of 0.02 secs
##
## Staggering insertions and deletions:
## ================================================================================
##
## Time difference of 0.21 secs
```

```
coiii.final=as.DNAbin(coiii.stag)
```

```
coiii.final=as.DNAbin(coiii.stag)
write.nexus.data(coiii.final,"coiii.nxs",charsperline = 1000)
```

trimming written dataset

```
length(coiii.final$Octopus_vulgaris)+49
```

```
## [1] 713
```

```
sed -i -E 's/(^.{713}).*/\1/g' coiii.nxs
```

Model Test

```
coiii.dist=dist.dna(coiii.final[!is.na(access$COIII)])
coiii.nj=NJ(coiii.dist)
coiii.pd=as.phyDat(coiii.final[!is.na(access$COIII)])
```

```
coiii.mT=modelTest(coiii.pd,coiii.nj)
```

```
## [1] "JC+I"
## [1] "JC+G"
```

```
## [1] "JC+G+I"
## [1] "F81+I"
## [1] "F81+G"
## [1] "F81+G+I"
## [1] "K80+I"
## [1] "K80+G"
## [1] "K80+G+I"
## [1] "HKY+I"
## [1] "HKY+G"
## [1] "HKY+G+I"
## [1] "SYM+I"
## [1] "SYM+G"
## [1] "SYM+G+I"
## [1] "GTR+I"
## [1] "GTR+G"
## [1] "GTR+G+I"
```

```
coiii.mT=coiii.mT[order(coiii.mT$AICc),]
coiii.mT$Model[1]
```

```
## [1] "GTR+G+I"
```

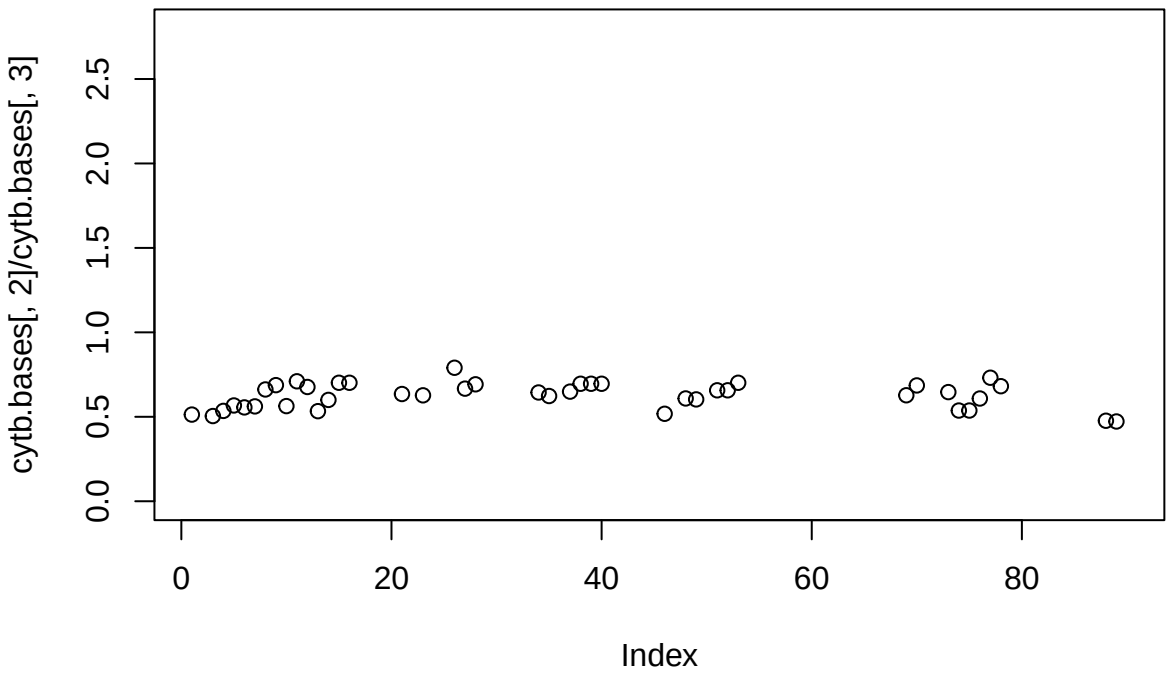
```
mt[3]=coiii.mT$Model[1]
```

Cytb

```
cytb.accession=access$Cytb
cytb.accession[is.na(cytb.accession)]="DJ078208"
cytb=read.GenBank(cytb.accession)
cytb[names(cytb)=="DJ078208"]=as.DNAbin("-")
names(cytb)=access$NA.
```

```
cytb.bases=base.freq(cytb[1])
for (i in 2:length(cytb)){
  cytb.bases=rbind(cytb.bases,base.freq(cytb[i]))
}
rownames(cytb.bases)=names(cytb)
```

```
plot(cytb.bases[,2]/cytb.bases[,3],ylim=c(0,2.8))
```




```
cytb2=DNAbin_to_DNAstringset(cytb)
cytb.align=AlignSeqs(cytb2)
```

```
## Determining distance matrix based on shared 9-mers:
## ================================================================================
##
## Time difference of 0.03 secs
##
## Clustering into groups by similarity:
## ================================================================================
##
## Time difference of 0.03 secs
##
## Aligning Sequences:
## ================================================================================
##
## Time difference of 0.82 secs
##
## Iteration 1 of 2:
##
## Determining distance matrix based on alignment:
## ================================================================================
##
## Time difference of 0 secs
##
## Reclustering into groups by similarity:
```

```
## ================================================================================
##
## Time difference of 0.03 secs
##
## Realigning Sequences:
## ================================================================================
##
## Time difference of 0.56 secs
##
## Iteration 2 of 2:
##
## Determining distance matrix based on alignment:
## ================================================================================
##
## Time difference of 0 secs
##
## Reclustering into groups by similarity:
## ================================================================================
##
## Time difference of 0.02 secs
##
## Realigning Sequences:
## ================================================================================
##
## Time difference of 0.03 secs
##
## Refining the alignment:
## ================================================================================
##
## Time difference of 0.27 secs
```

```
cytb.stag=StaggerAlignment(cytb.align)
```

```
## Calculating distance matrix:
## ================================================================================
##
## Time difference of 0 secs
##
## Constructing neighbor-joining tree:
## ================================================================================
##
## Time difference of 0.02 secs
##
## Staggering insertions and deletions:
## ================================================================================
##
## Time difference of 0.23 secs
```

```
cytb.final=as.DNAbin(cytb.stag)
write.nexus.data(cytb.final,"cytb.nxs",charsperline = 1000)
```

```
length(cytb.final$Octopus_vulgaris)+49
```

```
## [1] 815
```

```
sed -i -E 's/(^.{815}).*/\1/g' cytb.nxs
```

Model Test Cytb

```
cytb.dist=dist.dna(cytb.final[!is.na(access$Cytb)])
cytb.nj=NJ(cytb.dist)
cytb.pd=as.phyDat(cytb.final[!is.na(access$Cytb)])
```

```
cytb.mT=modelTest(cytb.pd,cytb.nj)
```

```
## [1] "JC+I"
## [1] "JC+G"
## [1] "JC+G+I"
## [1] "F81+I"
## [1] "F81+G"
## [1] "F81+G+I"
## [1] "K80+I"
## [1] "K80+G"
## [1] "K80+G+I"
## [1] "HKY+I"
## [1] "HKY+G"
## [1] "HKY+G+I"
## [1] "SYM+I"
## [1] "SYM+G"
## [1] "SYM+G+I"
## [1] "GTR+I"
## [1] "GTR+G"
## [1] "GTR+G+I"
```

```
cytb.mT=cytb.mT[order(cytb.mT$AICc),]
cytb.mT$Model[1]
```

```
## [1] "GTR+G+I"
```

```
mt[4]=cytb.mT$Model[1]
```

Combining data together.

```
write.table(t(stems.12s),"stems12s",sep=" ",row.names = F,col.names = F)
write.table(t(loops.12s),"loops12s",sep=" ",row.names = F,col.names = F)
```

Length of the whole dataset

```
total.len=length(x12s.final[1,])+
  length(coiii.final$Octopus_vulgaris)+
  length(cytb.final$Octopus_vulgaris)
total.len
```

```
## [1] 1926
```

The start of the coiii dataset:

```
coiii.start=length(x12s.final[1,])+1
coiii.start
```

```
## [1] 497
```

The start of the cytb dataset:

```
cytb.start=length(x12s.final[1,])+
  length(coiii.final$Octopus_vulgaris)+1
cytb.start
```

```
## [1] 1161
```

```
models=read.csv("model_selection.csv")
models.muus=character()
for (i in 1:length(mt)){
  models.muus[i]=paste("      lset applyto=(",i,") ",models$lset[models$model==mt[i]],";",sep="")
}
for (i in 1:length(mt)){
  models.muus[i+length(mt)]=paste("      prset applyto=(",i,") ",models$prset[models$model==mt[i]],";",s
}
models.muus=models.muus[-grep("NA",models.muus)]
write.table(models.muus,"models_to_write",row.names = F,col.names = F,quote = F)
```

Making basepairs

```
write.table(t(c(as.vector(t(read.csv("basepairs_12s.csv",header=F)[1,])))),"basepairs",row.names = F,co
```

```
sed -i -E 's/([0-9]{2,4}) ([0-9]{2,4})/\1:\2/g' basepairs
```

Writing length variable lines

```
write.table(paste("  DIMENSIONS NTAX=",nrow(access)," NCHAR=",total.len,";",sep=""),"dimensions",
            row.names = F,col.names = F,quote = F)

write.table(paste("      charset coiii =  ",coiii.start," - ",cytb.start-1,";",sep=""),"charset.coii",
            row.names = F,col.names = F,quote = F)

write.table(paste("      charset cytb =  ",cytb.start," - ",total.len,";",sep=""),"charset.cytb",
            row.names = F,col.names = F,quote = F)
```

Generating nexus file with MrBayes command block.

```
rm complete.nxs #just cleaning out previous iteration if running multiple times
rm -r mb
touch complete.nxs
echo "#NEXUS" >> complete.nxs
echo "BEGIN DATA;" >> complete.nxs
sed -n 1p dimensions >> complete.nxs
#echo "  DIMENSIONS NTAX=24 NCHAR=1725;" >> complete.nxs
echo "  FORMAT DATATYPE=DNA MISSING=? GAP=- INTERLEAVE=YES;
  MATRIX

  [12s]" >> complete.nxs
sed -n 7,96p 12s.nxs >> complete.nxs
echo "
   [COIII]" >> complete.nxs
sed -n 7,96p coiii.nxs >> complete.nxs
echo "
```

```
   [Cytb]" >> complete.nxs
sed -n 7,98p cytb.nxs >> complete.nxs
echo "
begin mrbayes;
    [Define pairs for the doublet model]" >> complete.nxs
cat basepairs | sed '1s/^/    pairs /' | sed -E 's/(:[0-9]{,4})/\1,/g' | sed -E 's/,$/;/g' | sed -r 's/
echo " " >> complete.nxs
cat stems12s | sed '1s/^/       charset  12s-stems = /' | sed -E 's/$/;/g' | sed -r 's/(.{73,76} )/\1\n

echo " " >> complete.nxs
cat loops12s | sed '1s/^/       charset  12s-loops = /' | sed -E 's/$/;/g' | sed -r 's/(.{73,76} )/\1\n
echo " " >> complete.nxs
sed -n 1p charset.coii >> complete.nxs
sed -n 1p charset.cytb >> complete.nxs
#echo "     charset coiii =  479 - 1118;" >> complete.nxs
#echo "     charset cytb =  1119 - 1725;" >> complete.nxs
echo " " >> complete.nxs
echo "      partition parts = 4:12s-stems,12s-loops,coiii,cytb;" >> complete.nxs
echo "      set partition = parts;

        lset applyto=(1)   nucmodel=doublet;
        lset applyto=(2,3,4)   nucmodel=4by4;
        prset ratepr=variable;
    " >> complete.nxs

cat models_to_write >> complete.nxs

echo "
     mcmcp ngen=100000000 printfreq=1000 samplefreq=10000 stoprule=yes stopval=0.01 nruns=5 nchains=2 bu
     mcmc;
     sump;
     sumt filename=mb/complete.nxs contype=allcompat conformat=simple;

end;
" >> complete.nxs

mkdir mb
cp complete.nxs mb/complete.nxs
```

```
mb-mpi mb/complete.nxs > mb_log
```

```
octo.trees=load.multi("mb/", format = "mb")
```

```
## [1] "complete.nxs.run1.t"
## [1] "Reading trees..."
## [1] "10000 generations per tree..."
## [1] "Trees are unrooted..."
## [1] "Reading parameter values from complete.nxs.run1.p"
## [1] "complete.nxs.run2.t"
## [1] "Reading trees..."
## [1] "10000 generations per tree..."
## [1] "Trees are unrooted..."
## [1] "Reading parameter values from complete.nxs.run2.p"
## [1] "complete.nxs.run3.t"
```

```
## [1] "Reading trees..."
## [1] "10000 generations per tree..."
## [1] "Trees are unrooted..."
## [1] "Reading parameter values from complete.nxs.run3.p"
## [1] "complete.nxs.run4.t"
## [1] "Reading trees..."
## [1] "10000 generations per tree..."
## [1] "Trees are unrooted..."
## [1] "Reading parameter values from complete.nxs.run4.p"
## [1] "complete.nxs.run5.t"
## [1] "Reading trees..."
## [1] "10000 generations per tree..."
## [1] "Trees are unrooted..."
## [1] "Reading parameter values from complete.nxs.run5.p"
```

```
check.chains(octo.trees)
```

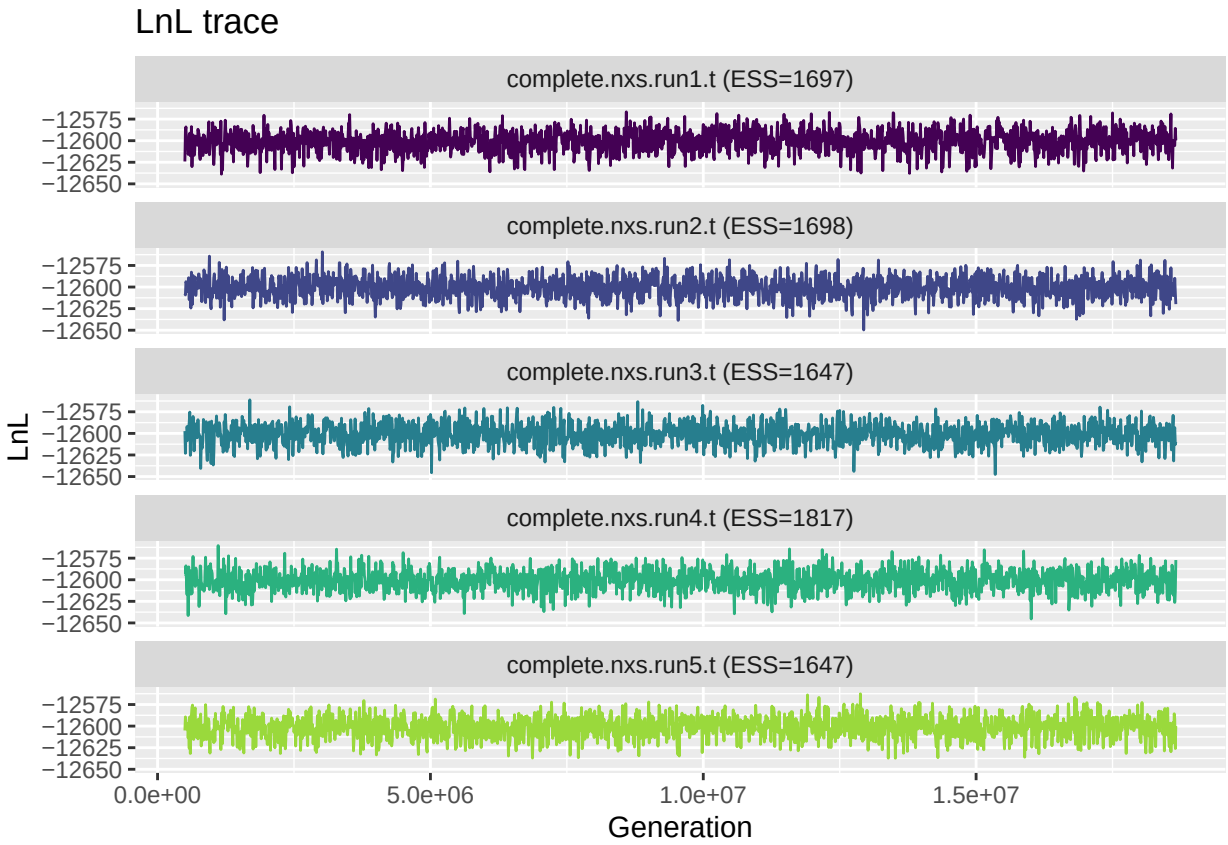
```
## $complete.nxs.run1.t
##               Length Class      Mode
## trees         1867   multiPhylo list
## ptable          38   data.frame list
## gens.per.tree    1   -none-     numeric
##
## $complete.nxs.run2.t
##               Length Class      Mode
## trees         1867   multiPhylo list
## ptable          38   data.frame list
## gens.per.tree    1   -none-     numeric
##
## $complete.nxs.run3.t
##               Length Class      Mode
## trees         1867   multiPhylo list
## ptable          38   data.frame list
## gens.per.tree    1   -none-     numeric
##
## $complete.nxs.run4.t
##               Length Class      Mode
## trees         1867   multiPhylo list
## ptable          38   data.frame list
## gens.per.tree    1   -none-     numeric
##
## $complete.nxs.run5.t
##               Length Class      Mode
## trees         1867   multiPhylo list
## ptable          38   data.frame list
## gens.per.tree    1   -none-     numeric
```

```
rwty.processors <<- 15
octo.rwty=analyze.rwty(octo.trees,filename="octo_rwty.pdf")
```

```
makeplot.param(octo.trees, burnin = 50, "LnL")
```

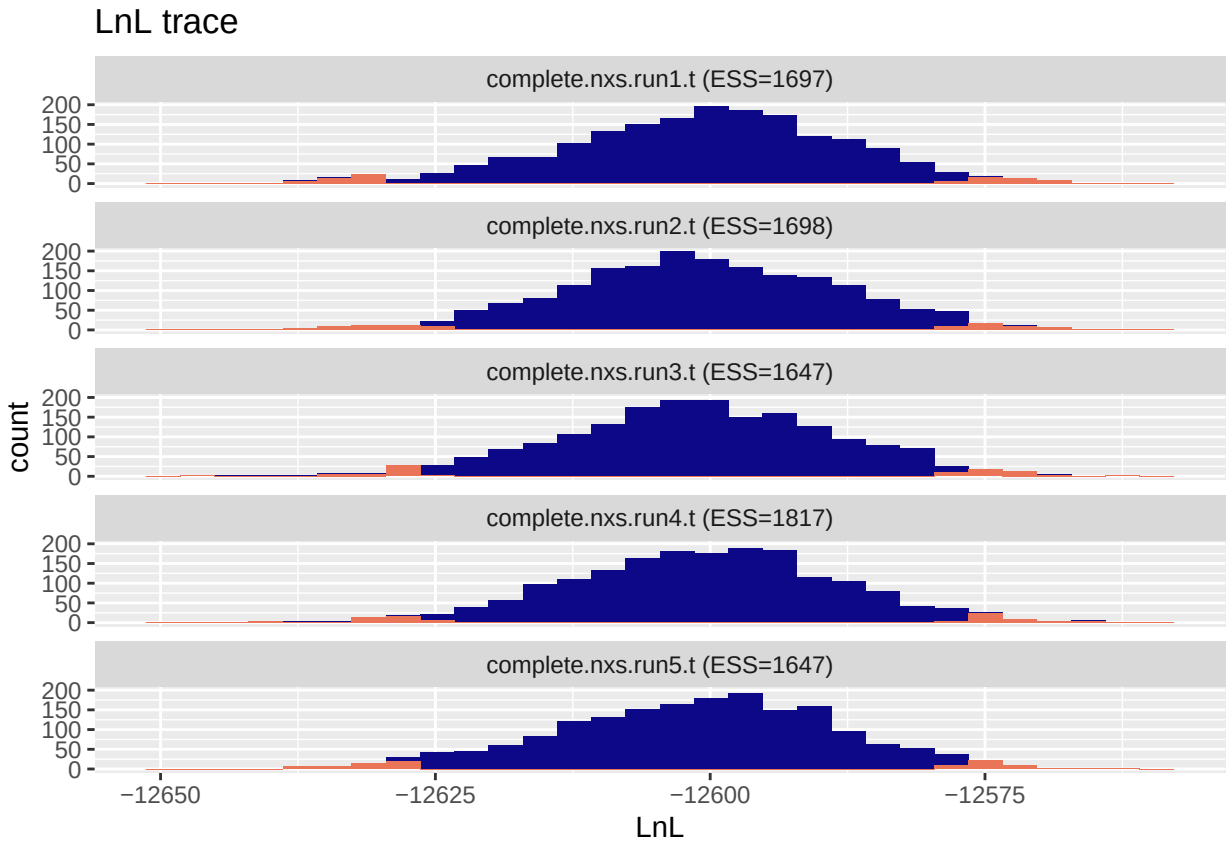
```
## [1] "Creating trace for LnL"
```

```
## $trace.plot
```




```
##
## $density.plot
```

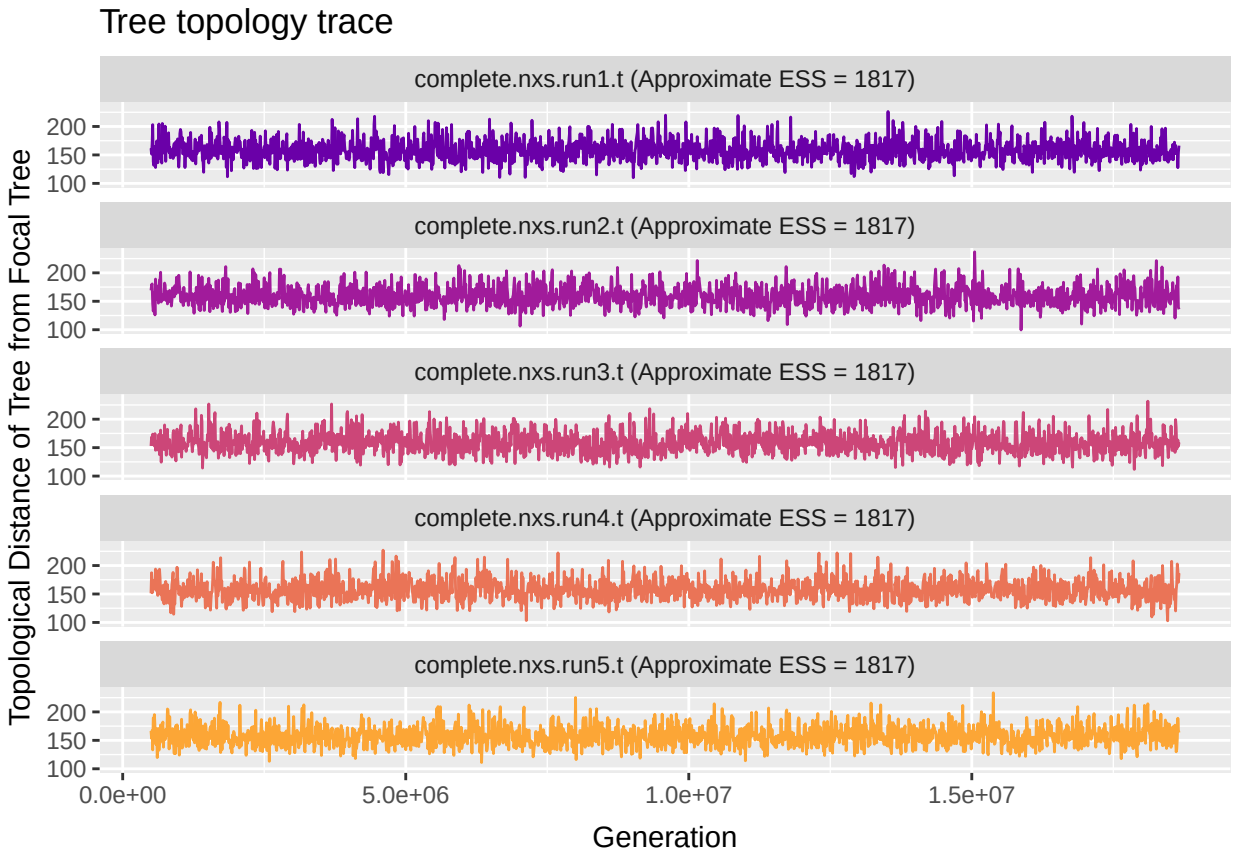
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



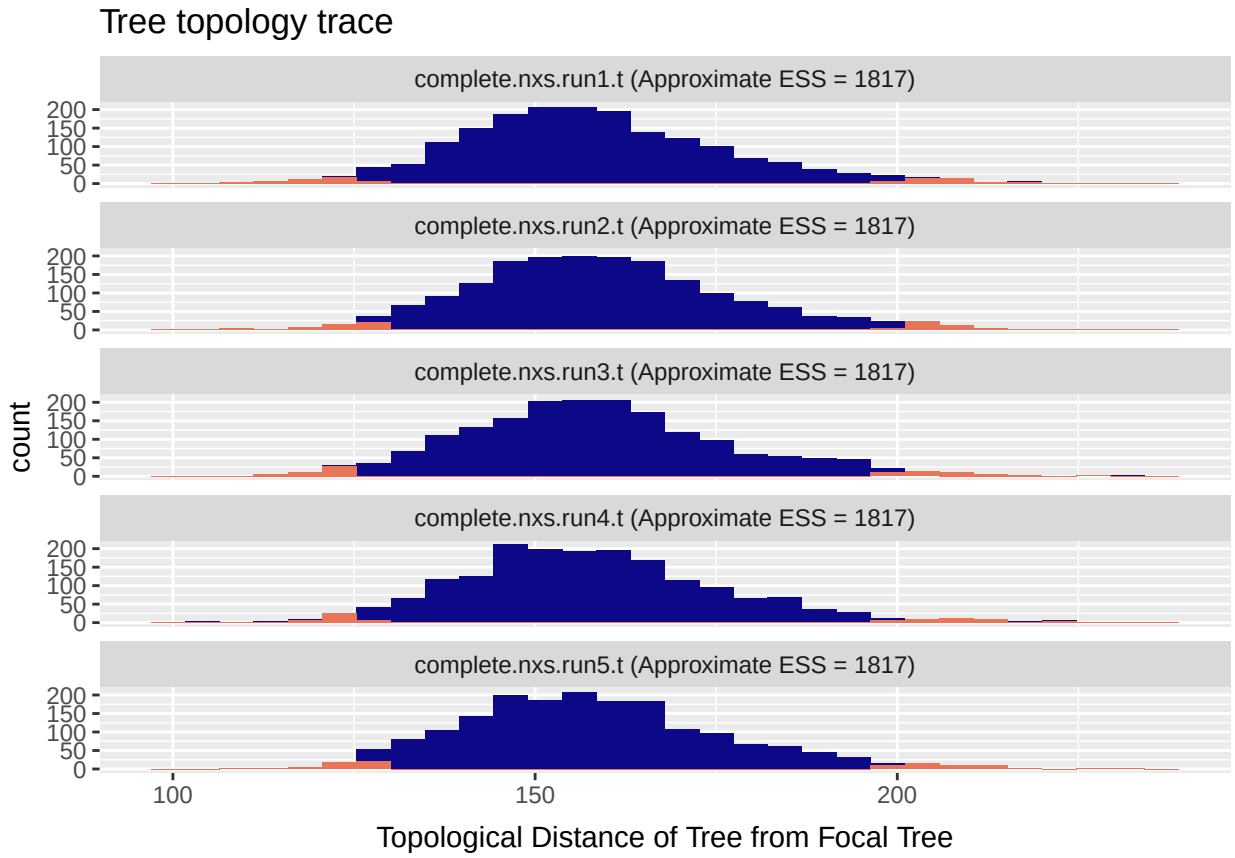

```
makeplot.topology(octo.trees, burnin = 50)
```

```
## [1] "Creating trace for tree topologies"
## [1] "Calculating approximate ESS with sampling intervals from 1 to 100"
```

```
## $trace.plot
```



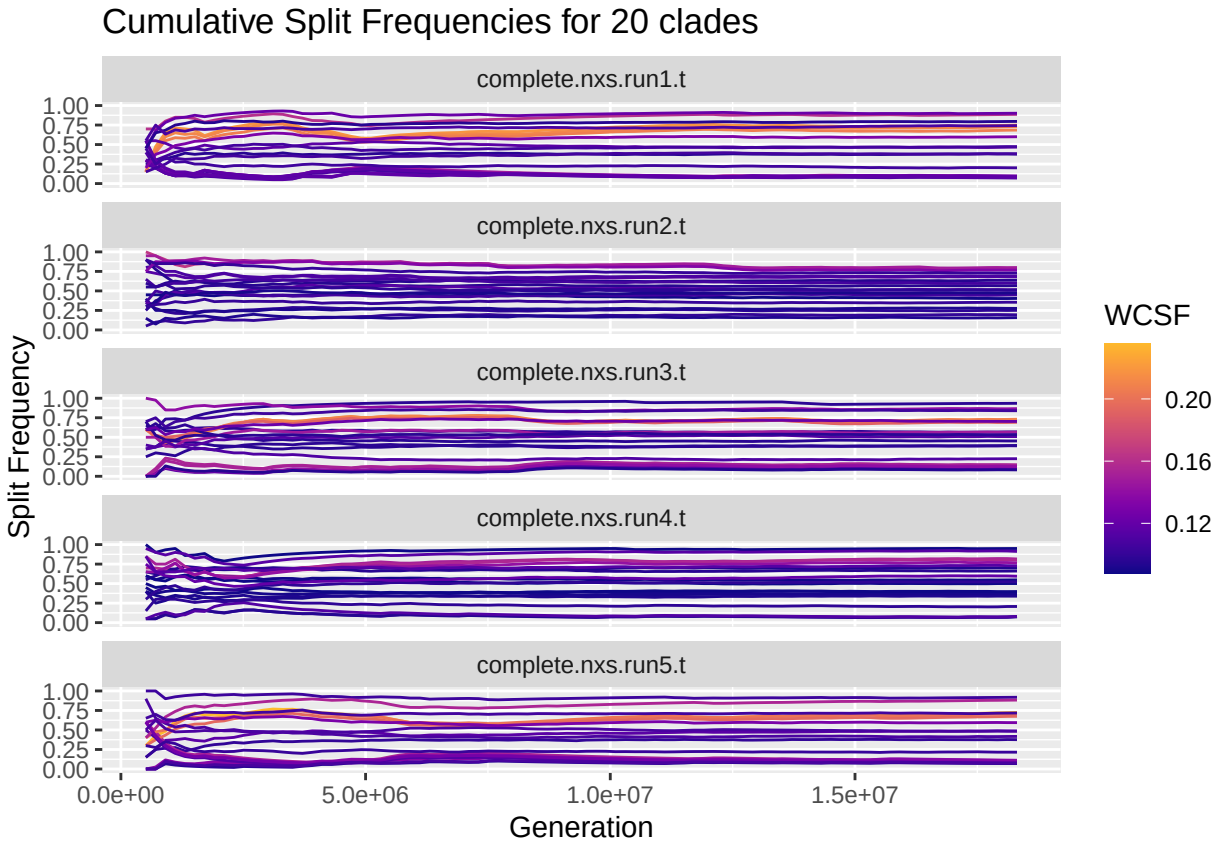

```
##
## $density.plot

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
makeplot.splitfreqs.cumulative(octo.trees, burnin = 50)
```

```
## [1] "Creating cumulative split frequency plot for 20 clades"
```

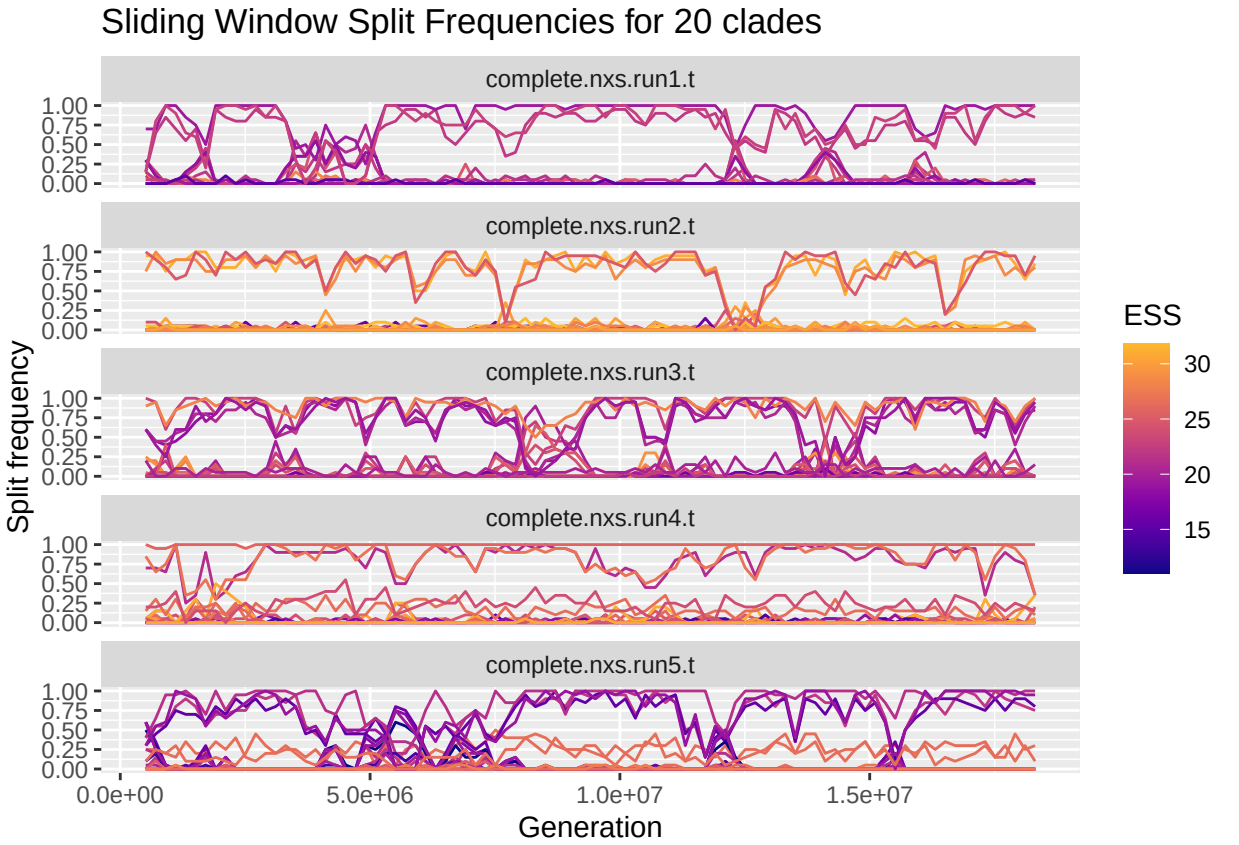
```
## $splitfreqs.cumulative.plot
```




```
makeplot.splitfreqs.sliding(octo.trees, burnin = 50)
```

```
## [1] "Creating sliding window split frequency plot for 20 clades"
```

```
## $splitfreqs.sliding.plot
```



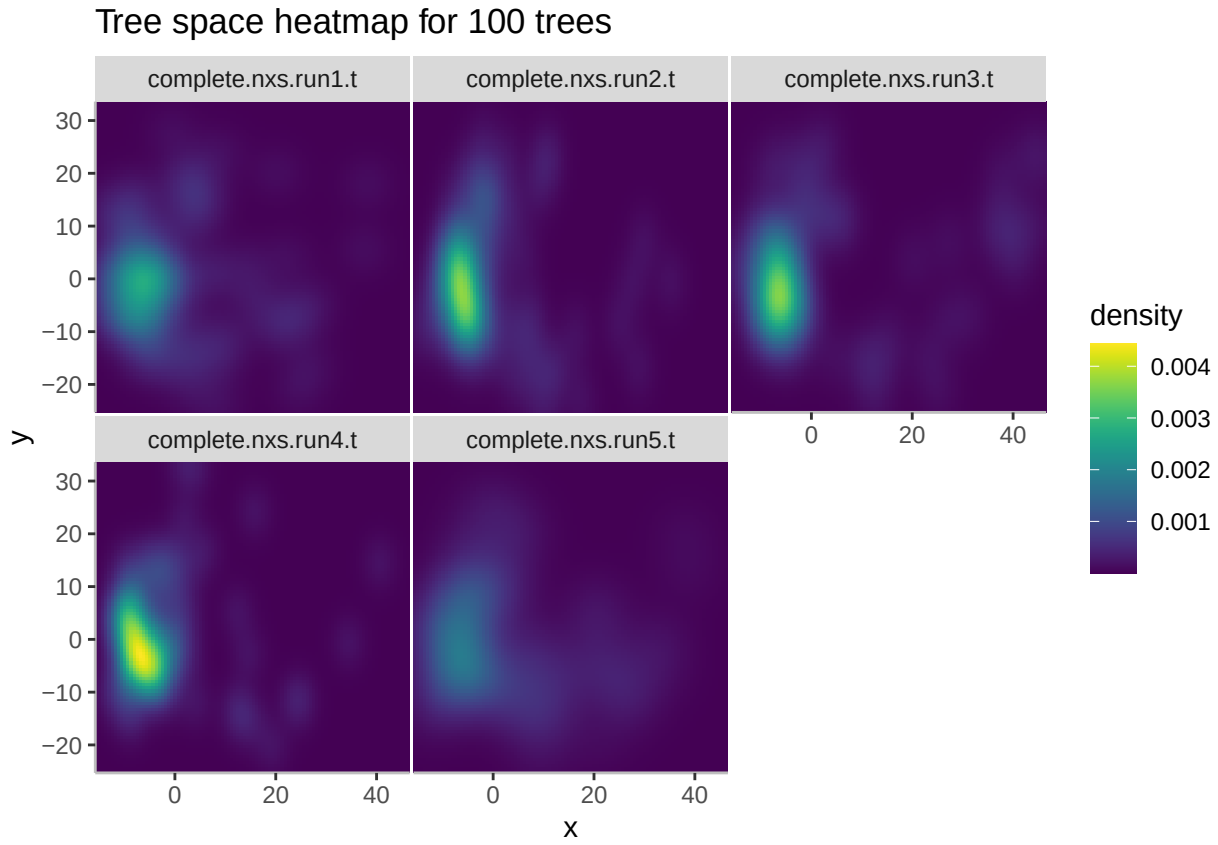

```
makeplot.treespace(octo.trees, burnin =50, fill.color = "LnL")
```

```
## [1] "Creating treespace plots"
```

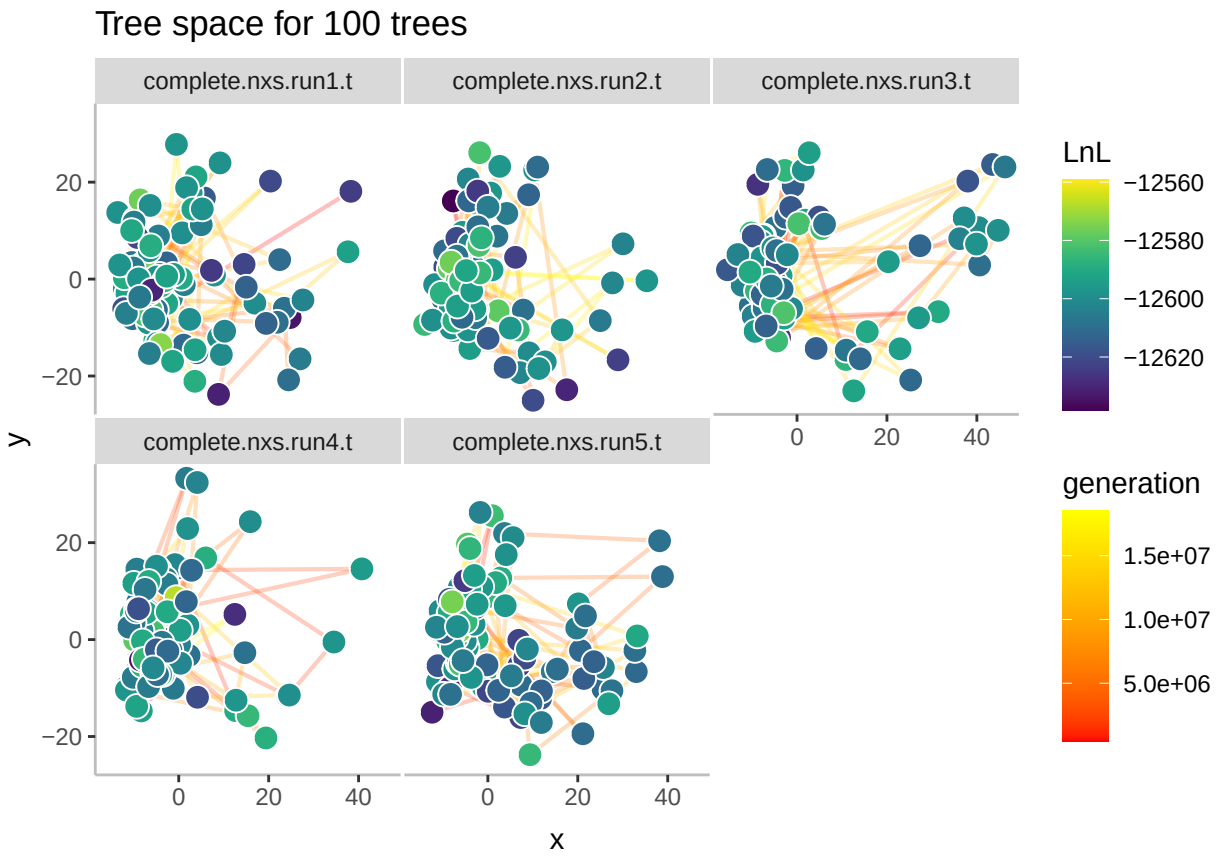
```
## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```

```
## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```

```
## $treespace.heatmap
```




```
##
## $treespace.points.plot
```



Reading into R the resulting trees

The file that contains the consensus trees is "complete.nxs.con.tre". This file actually contains two trees. One contains no branch probability information, and the other contains the branch probabilities. The following code deletes the tree without probabilities from file. If both trees are present, R opens the tree as a multi-phylo class, and manipulation of the tree becomes much harder.

```
cp ./mb/complete.nxs.con.tre ./mb/completeBACKUP.nxs.con.tre
sed -i '/Note: This tree contains information only on the topology/d' ./mb/complete.nxs.con.tre
sed -i '/and branch lengths (median of the posterior probability density)/d' ./mb/complete.nxs.con.tre
sed -i '/):/d' ./mb/complete.nxs.con.tre
```

Now I read the tree into R using the following code.

```
mrbayes.tree=read.nexus("mb/complete.nxs.con.tre",tree.names=NULL)
```

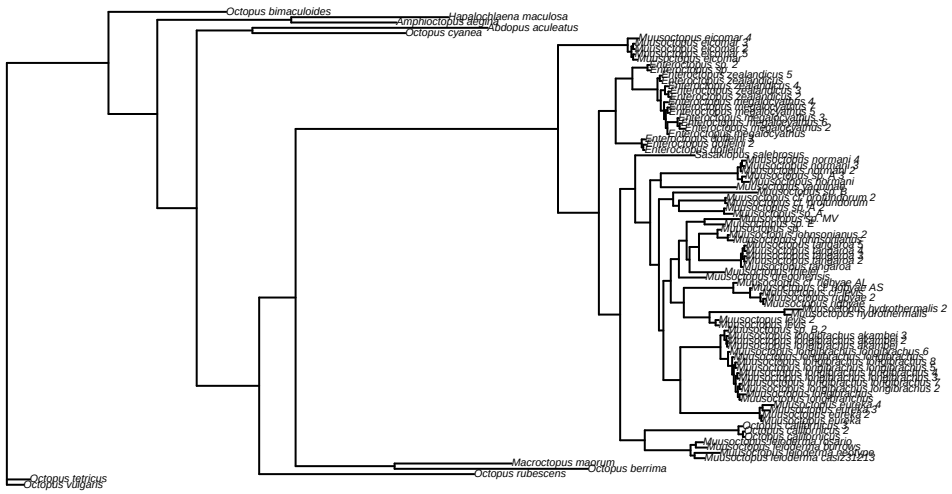
Taking a quick look at the tree to make sure it looks right.

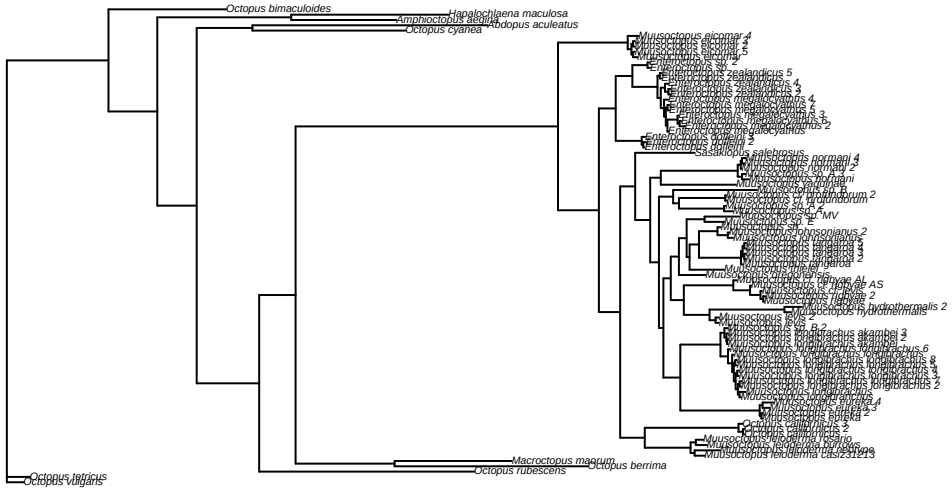
```
svg(filename = "tree_Expanded_Enterocopodidae.svg",height=14,width=7)
plot(mrbayes.tree, show.node.label=F,cex=0.3)
nodelabels(mrbayes.tree$node.label,bg=NULL,cex=0.3,frame="none",adj=c(1.1,-0.1))
dev.off()
```

```
## pdf
##   2
```

```
as.numeric(mrbayes.tree$node.label)
```

```
plot(mrbayes.tree, show.node.label=F,cex=0.3)
```




Octopus bimaculoides
Hapalochlaena maculosa
Amphioctopus aegina
Abdopus aculeatus
Octopus cyanea
Muusoctopus eiconae 4
Muusoctopus eiconae 3
Muusoctopus eiconae 5
Muusoctopus eiconae
Enteroctopus sp. 2
Enteroctopus zealandicus 5
Enteroctopus zealandicus
Enteroctopus zealandicus 4
Enteroctopus megalocyathus 4
Enteroctopus megalocyathus 2
Enteroctopus megalocyathus 3
Enteroctopus megalocyathus 2
Enteroctopus megalocyathus
Enteroctopus dofleini 3
Enteroctopus dofleini 2
Enteroctopus dofleini
Casakopus calebrosus
Muusoctopus normani 4
Muusoctopus normani 3
Muusoctopus normani 2
Muusoctopus normani
Muusoctopus rapanui
Muusoctopus sp. B
Muusoctopus cf. hydrothermalis 2
Muusoctopus sp. A 2
Muusoctopus sp. A
Muusoctopus sp. MV
Muusoctopus sp. E
Muusoctopus sp.
Muusoctopus longibrachus 2
Muusoctopus longibrachus 5
Muusoctopus longibrachus 4
Muusoctopus longibrachus 3
Muusoctopus longibrachus 2
Muusoctopus longibrachus
Muusoctopus thielei
Muusoctopus profundorum
Muusoctopus hydrothermalis AL
Muusoctopus hydrothermalis AS
Muusoctopus cf. thielei
Muusoctopus rapanui 2
Muusoctopus rapanui
Muusoctopus hydrothermalis 2
Muusoctopus hydrothermalis
Muusoctopus leioderma
Muusoctopus leioderma 2
Muusoctopus sp. B 2
Muusoctopus longibrachus akambei 3
Muusoctopus longibrachus akambei 2
Muusoctopus longibrachus akambei
Muusoctopus longibrachus longibrachus 6
Muusoctopus longibrachus longibrachus
Muusoctopus longibrachus longibrachus 5
Muusoctopus longibrachus longibrachus 4
Muusoctopus longibrachus longibrachus 3
Muusoctopus longibrachus longibrachus 2
Muusoctopus longibrachus
Muusoctopus eiconae 4
Muusoctopus eiconae 3
Muusoctopus eiconae 2
Muusoctopus eiconae
Muusoctopus eureka 3
Muusoctopus eureka 2
Muusoctopus eureka
Octopus californicus 3
Octopus californicus 2
Muusoctopus berryi Vestigo
Muusoctopus leioderma Vestigo
Muusoctopus leioderma cas231213
Muusoctopus leioderma cas231211
Macroctopus maorum
Octopus rubescens
Octopus berrima
Octopus tetricus
Octopus vulgaris