Noah L. Schrick
1492657

Lab 11. Phylogenetic analysis of Human, Neanderthal and other primates using a mitochondrial DNA (mtDNA) gene. When studying closely related organisms, mtDNA is a useful tool because it has a high mutation rate. Again use a color to indicate your answers.

A. Fetch GenBank sequences and write a multiple fasta file. First manually find one of the sequences on Genbank by searching for the GenBank id X90314 (http://www.ncbi.nlm.nih.gov/) under nucleotide in the dropdown. 1. What is the name of the gene and the species?

1. Mitochondrial DNA for D-Loop (isolate WG+ice37+B) for Homo Sapiens.

Start a new R script with the following code to grab multiple species' mitochondrial DNA sequences from NCBI and load them into one object for later alignment. You may need to install the `"ape"` library, which stands for Analysis of Phylogenetics and Evolution. 2. From the output of the `names(...)` command, what are the species of the sequences? 3. How many nucleotides are in the Human sequence?

```
# setwd – set the working directory!
library(ape) # needed for read.GenBank, install.packages

# fetch the mtDNA sequences
mtDNA.MultiSeqs.list<-
read.GenBank(c("AF011222","AF254446","X90314","AF089820","AF176766",
            "AF451972", "AY079510","AF050738","AF176722","AF315498",
                  "AF176731","AF451964"), as.character=TRUE)
# look at species names
mtDNA.Species<-attr(mtDNA.MultiSeqs.list,"species")
# use species as name instead of genbank id
names(mtDNA.MultiSeqs.list)<-mtDNA.Species
# need to fix some names
names(mtDNA.MultiSeqs.list)[1] <- paste("German_Neanderthal",sep="")
names(mtDNA.MultiSeqs.list)[2] <- paste("Russian_Neanderthal",sep="")
names(mtDNA.MultiSeqs.list)[3] <- paste("Human")
names(mtDNA.MultiSeqs.list)[6] <- paste("Puti_Orangutan",sep="")
names(mtDNA.MultiSeqs.list)[12] <- paste("Jari_Orangutan",sep="")

length(mtDNA.MultiSeqs.list$Human)

# look at one of the sequences using $
mtDNA.MultiSeqs.list$Human
```

2. Initially, the names are:

```
> names(mtDNA.MultiSeqs.list)
 [1] "Homo_sapiens_neanderthalensis"  "Homo_sapiens_neanderthalensis"
"Homo_sapiens"
 [4] "Gorilla_beringei_beringei"      "Pan_troglodytes_troglodytes"
"Pongo_pygmaeus"
 [7] "Gorilla_gorilla_gorilla"        "Gorilla_beringei_graueri"
"Pan_troglodytes_schweinfurthii"
[10] "Pan_troglodytes_ellioti"        "Pan_troglodytes_verus"
"Pongo_pygmaeus"
```

A few of these names are then corrected, and the final names are:

```
> names(mtDNA.MultiSeqs.list)
 [1] "German_Neanderthal"             "Russian_Neanderthal"            "Human"
 [4] "Gorilla_beringei_beringei"      "Pan_troglodytes_troglodytes"
"Puti_Orangutan"
 [7] "Gorilla_gorilla_gorilla"        "Gorilla_beringei_graueri"
"Pan_troglodytes_schweinfurthii"
[10] "Pan_troglodytes_ellioti"        "Pan_troglodytes_verus"          "Jari_Orangutan"
```

3. There are 360 nucleotides in the Human sequence.

Add the code below to convert the list of sequences into a Biostrings object. This is needed to perform the multiple sequence alignment. 4. Which species has the longest sequence? 5. Explain what the rows of `proportion.nts` represent.

```
### Convert to Biostrings object for the sequences
#library(BiocManager)
#BiocManager::install("Biostrings")
library(Biostrings)
# loop through the list to create vector of strings for Biostrings input
Names.vec <- c() # initialize speices names string vector
Seqs.vec <- c()  # initialize sequence string vector
for (mtDNA.name in names(mtDNA.MultiSeqs.list))
{
```

```
 Names.vec <- c(Names.vec,mtDNA.name) # concatenate vector
 Seqs.vec <-c(Seqs.vec,paste(mtDNA.MultiSeqs.list[[mtDNA.name]],collapse=""))
}
mtDNA.multSeqs.bstr <- DNAStringSet(Seqs.vec) # convert to Biostring

# name the Biostring sequences and compute stats
names(mtDNA.multSeqs.bstr) <- Names.vec # count nucs and sequence lengths
num.nts <- alphabetFrequency(mtDNA.multSeqs.bstr)[,1:4]
mtDNA.lengths <- rowSums(num.nts)
proportion.nts <- num.nts/mtDNA.lengths
```

4.

```
> cbind(mtDNA.lengths, Names.vec)
      mtDNA.lengths Names.vec
 [1,] "379"         "German_Neanderthal"
 [2,] "345"         "Russian_Neanderthal"
 [3,] "360"         "Human"
 [4,] "374"         "Gorilla_beringei_beringei"
 [5,] "340"         "Pan_troglodytes_troglodytes"
 [6,] "354"         "Puti_Orangutan"
 [7,] "367"         "Gorilla_gorilla_gorilla"
 [8,] "374"         "Gorilla_beringei_graueri"
 [9,] "339"         "Pan_troglodytes_schweinfurthii"
[10,] "411"         "Pan_troglodytes_ellioti"
[11,] "339"         "Pan_troglodytes_verus"
[12,] "345"         "Jari_Orangutan"
```

```
nlengthnames <- cbind(mtDNA.lengths, Names.vec)
idx <- which.max(nlengthnames[,1])
nlengthnames[idx,]
```

```
> nlengthnames[idx,]
          mtDNA.lengths              Names.vec
                  "411" "Pan_troglodytes_ellioti"
```

5.

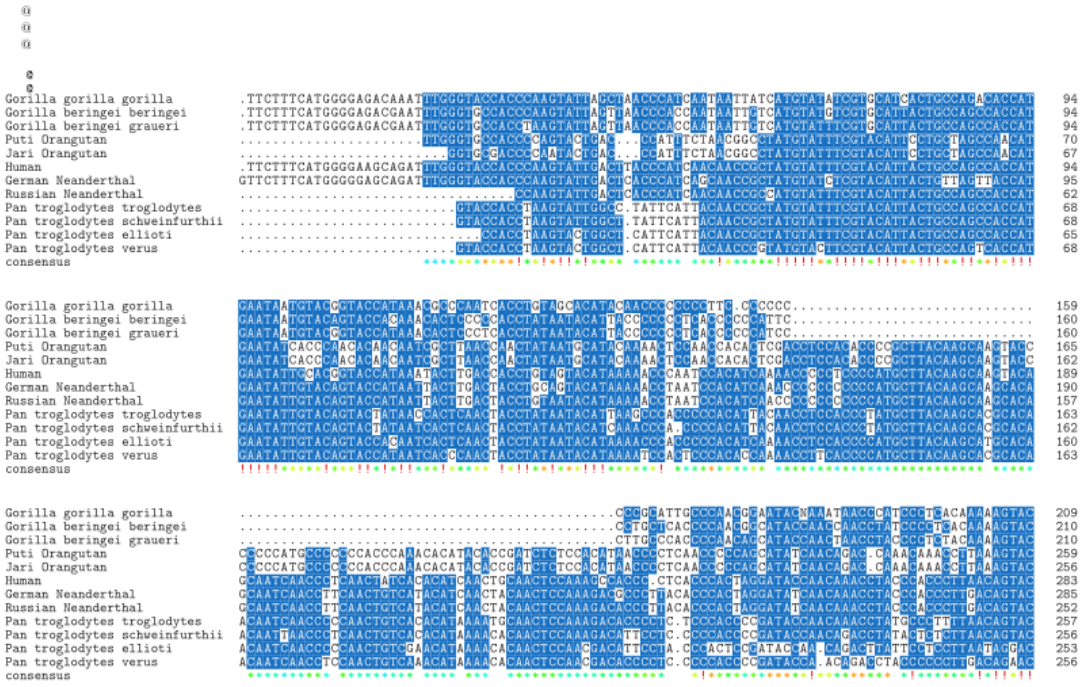
proportion.nts represents the proportion of each nucleotide in a sequence. For example, for the “Pan_troglodytes_ellioti”, “A” makes up 32.6% of the 411-length sequence and “C” takes up 36.3%.

B. Multiple Sequence Alignment with MSA library. Use `msa` to create the multiple sequence alignment of the mitochondrial primate sequences. You may need to install msa. If the msaPrettyPrint function works to create a pdf, paste the pdf into the Word document. If the pdf isn't created, you can use a tex complier like TexWorks + MacTex for Mac.

```
#library(BiocManager)
#BiocManager::install("msa") library(msa)
mtDNA.msa <- msa(mtDNA.multSeqs.bstr,method="ClustalW")
setw("set working directory")
msaPrettyPrint(mtDNA.msa, file="mtDNA.pdf", output="pdf", showNames="left",
               showLogo="none", askForOverwrite=FALSE, verbose=TRUE)
```

Inspect the MSA result above. 1. Looking at the alignment pdf, describe the alignments at the left and right ends. **If you cannot create the pdf with msaPrettyPrint, use the following code to answer 1.**

```
ncol(mtDNA.msa)
firstPart <- DNAMultipleAlignment(subseq(unmasked(mtDNA.msa),1,50))
firstPart  # first section of alignment
midPart <- DNAMultipleAlignment(subseq(unmasked(mtDNA.msa),300,350))
midPart    # a middle section of alignment
lastPart <- DNAMultipleAlignment(subseq(unmasked(mtDNA.msa),400,452))
lastPart   # end section of alignment
```



NOTE: I used ClustalOmega for this portion. ClustalW consistently caused crashes to the R session and it could not be resolved through numerous troubleshooting steps and with various approaches.

The beginning of the alignment (positions 1-50) displays many gaps between all the sequences. Where gaps are not present, there are very similar sequences of "TTCTTTCATGGGG". In the first portion after the initial gaps and alignment, when sequences do not exactly match, it is typically by only one nucelotide, and the sequences go back to matching.

The end of the alignment is comparatively worse than the beginning and middle of the alignment, but there is still quite a bit of similarity overall. In the last last 7 positions of the alignment, 5 are identical among all the sequences.

Add the following code to create the `align.stats.df` data.frame that contains information about the sequences and the alignments. 2. For the first sequence in `align.stats.df`, what is the species, number of gaps, the length without gaps, and the nucleotide frequencies?

```
## loop to make results data frame
num_seqs <- length(Names.vec)
# initialize data frame
align.stats.df <- data.frame(species=rep(NA,num_seqs), seqlen=rep(0,num_seqs),
                             numgaps=rep(0,num_seqs), nt_a=rep(NA,num_seqs),
                             nt_c=rep(NA,num_seqs), nt_g=rep(NA,num_seqs),
                             nt_t=rep(NA,num_seqs))
# DNAbin type required for dist.dna and helpful for other calculations
mtDNA.msa.DNAbin <- as.DNAbin(mtDNA.msa)
for (i in 1:num_seqs){
      seq_name <- Names.vec[i]
      seq.vec <- as.character(mtDNA.msa.DNAbin[i,])
      num.gaps <- sum(seq.vec=="-")
      prop.nt.i <- proportion.nts[i,]
      align.stats.df[i,] <- c(seq_name, mtDNA.lengths[i], num.gaps,
             round(prop.nt.i[1],digits=2), round(prop.nt.i[2],digits=2),
             round(prop.nt.i[3],digits=2), round(prop.nt.i[4],digits=2))
}

# write to file
write.table(align.stats.df,file="alignstats.tab",sep = "\t", row.names=FALSE,
quote=FALSE)

# you can use $ operator to grab a named column from a data.frame
# similar to grabbing a named variable from a list
align.stats.df$species
align.stats.df$nt_a             # strings by default
as.numeric(align.stats.df$nt_a) # convert to numeric
```

2.

```
> align.stats.df[1,]
          species seqlen numgaps nt_a nt_c nt_g nt_t
1 German_Neanderthal    379      81 0.31 0.32 0.13 0.24
```

The `write.table` function above writes the table to a tab delimited file (.tab). Open this file in Excel and paste it into your Word document. 3. Which are the most used and least used nucleotides across species? 4. Which species has the longest and shortest sequence and what are their lengths? 5. How might the long sequence affect the multiple sequence alignment – look at the alignment pdf? Sometimes it is advisable to trim ends of multiple sequence alignments because of lack of information leading to inaccurate alignments.

species	seqlen	numgaps	nt_a	nt_c	nt_g
German_Neanderthal	379	81	0.31	0.32	0.13
Russian_Neanderthal	345	76	0.33	0.34	0.11
Human	360	76	0.33	0.34	0.11
Gorilla_beringei_beringei	374	96	0.29	0.36	0.13
Pan_troglodytes_troglodytes	340	105	0.32	0.36	0.1
Puti_Orangutan	354	90	0.31	0.4	0.1
Gorilla_gorilla_gorilla	367	71	0.3	0.35	0.14
Gorilla_beringei_graueri	374	105	0.28	0.35	0.14
Pan_troglodytes_schweinfurthii	339	110	0.34	0.35	0.09
Pan_troglodytes_ellioti	411	111	0.33	0.36	0.1
Pan_troglodytes_verus	339	39	0.34	0.37	0.1
Jari_Orangutan	345	111	0.32	0.39	0.1

3.

Average a: 0.317
Average c: 0.358 → Most used
Average g: 0.113 → Least used
Average t: 0.22

4.

Longest sequence **(411)**: Pan_troglodyttes_ellioti
Shortest sequence **(339)**: Pan_troglodytes_schweinfurthii **and** Pan_troglodytes_verus

5.

Since there is a 19.2% difference between the longest and shortest sequences, there is a large number of gaps to pad the sequences. This large number of gaps

can be seen in the multiple sequence alignment in this example.

D. Create DNA distance matrices and phylogenetic trees. Trees are sometimes called dendrograms or hierarchical clusters. Run the two lines of code below. First we convert the Biostrings multiple sequence alignment object into a "DNAbin" data type as required by ape for use in dist.dna. Then we use the function dist.dna to compute the distance between all pairs of sequences, which we assign to the variable mtDNA.dist. mtDNA.dist is stored as a lower diagonal object because distances are symmetric and it saves memory. The default molecular evolution model for DNA distance calculations is Kimura's 1980 model (K80).

```
### Compute Distances
mtDNA.dist <- dist.dna(mtDNA.msa.DNAbin,model="K80")
```

Below we transform the lower-diagonal DNA distance into a full matrix and compute the minimum value in this matrix. Computing the min is not necessary for creating the tree; the idea is to give you some insight into how hclust works next to create the tree from the distance matrix. 1. What is the minimum value in the distance matrix and what does it represent? 2. What two sequences does the value correspond to?

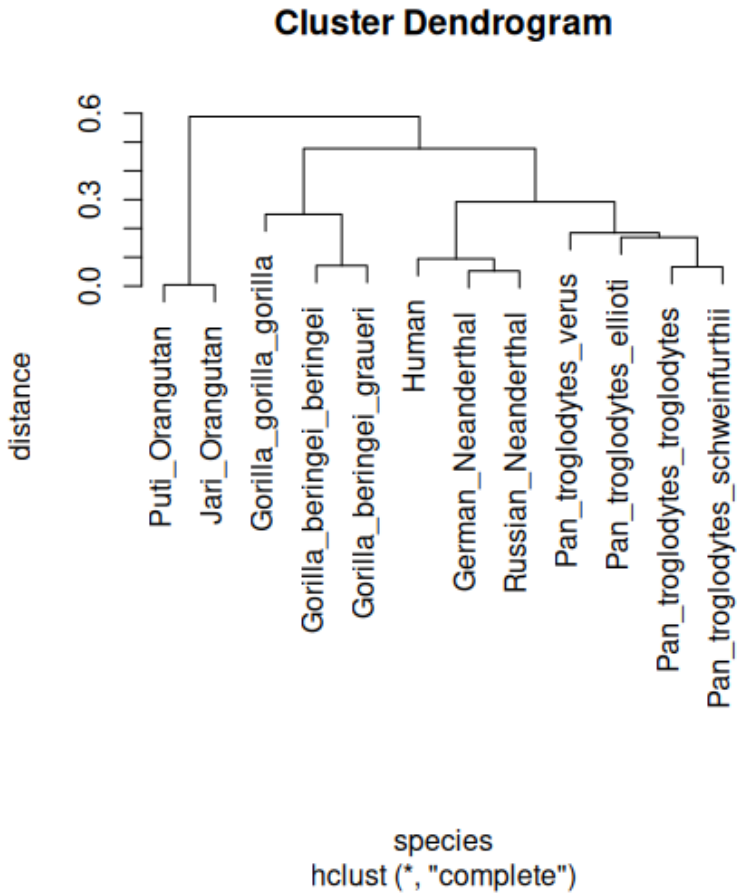
```
# manually find closest species
mtDNA.dist.mat <-as.matrix(mtDNA.dist)
diag(mtDNA.dist.mat)<-1  # force diagonal to be 1, not 0
which(mtDNA.dist.mat == min(mtDNA.dist.mat), arr.ind = TRUE)
```

1. > min(mtDNA.dist.mat)
[1] 0.004166691
This represents the distance of the closest two values in the matrix.

2.
Jari Orangutan and Puti Orangutan

Use `hclust` below to construct a tree from the distance matrix. The function `hclust` is a built-in R routine for computing hierarchical clusters from any distance matrix, not just from sequence alignments. 3. From the output of the code below, show your dendrogram and explain how the `which/min` calculation above is reflected in the plot.

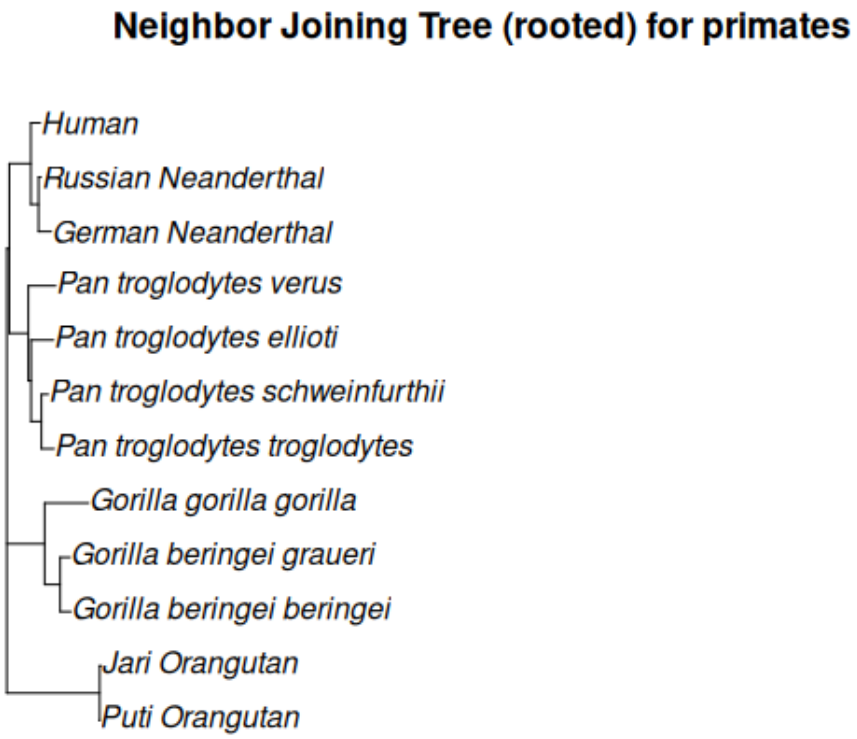
```
hc<- hclust(as.dist(mtDNA.dist.mat)) # transform to dist object first
plot(hc,xlab="species",ylab="distance")
```

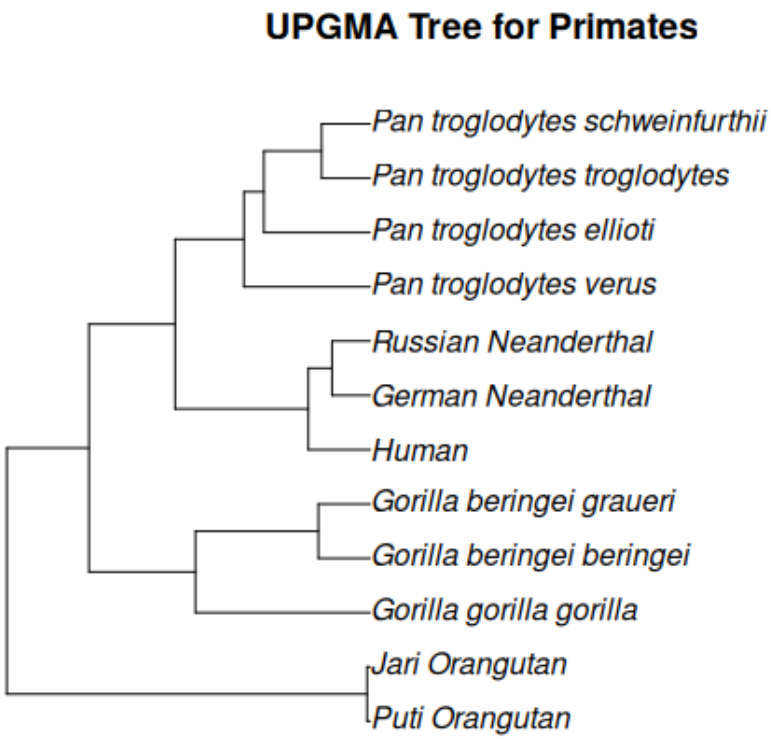


The which/min calculation is reflected in the plot via the connection (horizontal line on the plot) of species. For instance, the Jari and Puti Orangutans were identified as having the smallest distance, so they are connected on the plot at a height of their distance.

UPGMA, neighbor joining, and bootstrap support. Next we use UPGMA and NJ methods from the `phangorn` library. If you can't install phangon, you probably need to download and install a newer version of R and restart RStudio.

```
# Try this. If it doesn't work, upgrade R to the latest version
install.packages("phangorn")
#library(BiocManager)              # try if install.packages fails
#BiocManager::install("phangorn")
library(phangorn)
mtDNA.tree.nj <- NJ(mtDNA.dist) # phangorn function
plot(mtDNA.tree.nj, main="Neighbor Joining Tree (rooted) for primates")
# UPGMA
mtDNA.tree.upgma  <- upgma(mtDNA.dist)
plot(mtDNA.tree.upgma, show.node.label = TRUE, main="UPGMA Tree for Primates")
```





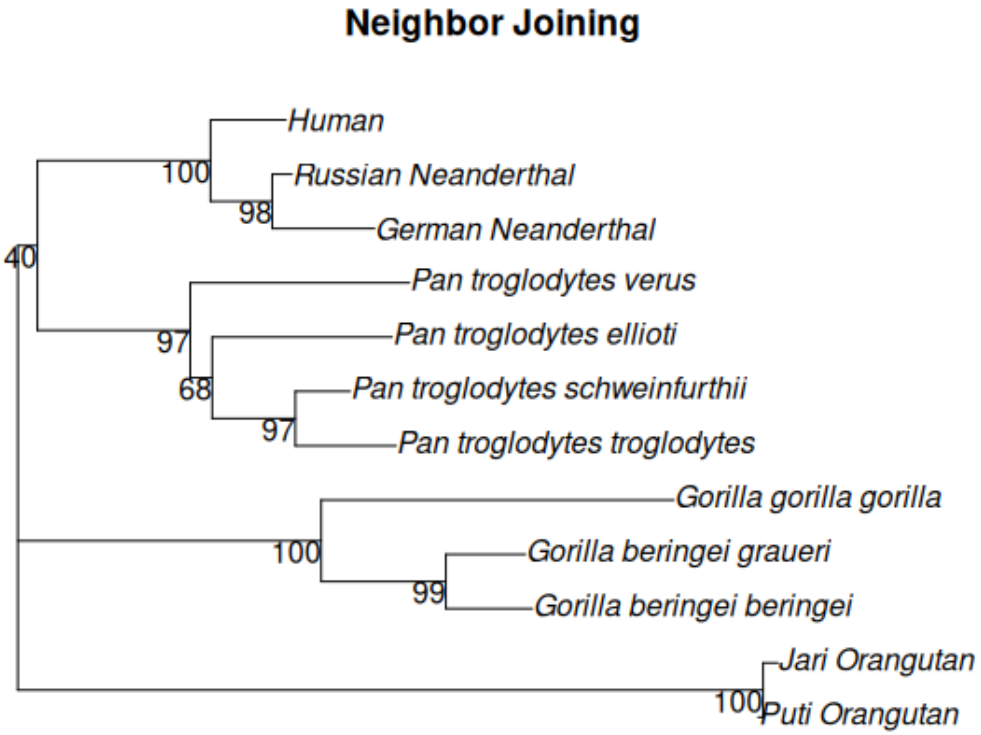
In order to do bootstrapping, we need to convert the MSA to a phangorn-friendly format with the `msaConvert` function, which is in the **download msaUtils.R from Harvey**.

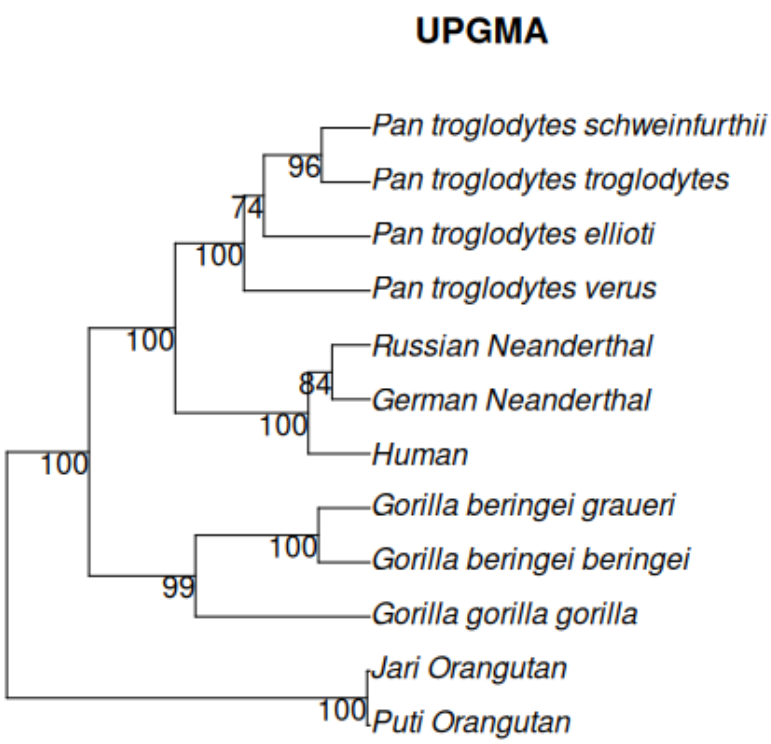
```
#setwd
source("msaUtils.R") # load msaConvert function into memory
mtDNA.msa.phangorn <-msaConvert(mtDNA.msa,type="phangorn::phyDat")
parsimony(mtDNA.tree.nj, mtDNA.msa.phangorn)
# bootstrap to show support for tree edges
# creates trees from bootstrap samples and checks how often
# that edge appears. Show consistency of tree edge.
bs.trees <- bootstrap.phyDat(mtDNA.msa.phangorn,
FUN=function(x)NJ(dist.dna(as.DNAbin(x),model="K80")), bs=100)
plotBS(mtDNA.tree.nj, bs.trees, "phylogram", main="Neighbor Joining")

parsimony(mtDNA.tree.upgma, mtDNA.msa.phangorn)

bs.upgma.trees <- bootstrap.phyDat(mtDNA.msa.phangorn,
FUN=function(x)upgma(dist.dna(as.DNAbin(x),model="K80")), bs=100)
plotBS(mtDNA.tree.upgma, bs.upgma.trees, "phylogram", main="UPGMA")
```

Show the bootstrap phylogenetic trees. 4. Based on these trees, which connections have the least bootstrap support? 5. Based on these data, which two species are humans most closely related to? Which groups of species are next closely related?





4.

From the Neighbor Joining tree, The grouping of Humans and Neanderthals with Troglodytes have the least bootstrap support with gorillas and orangutans.

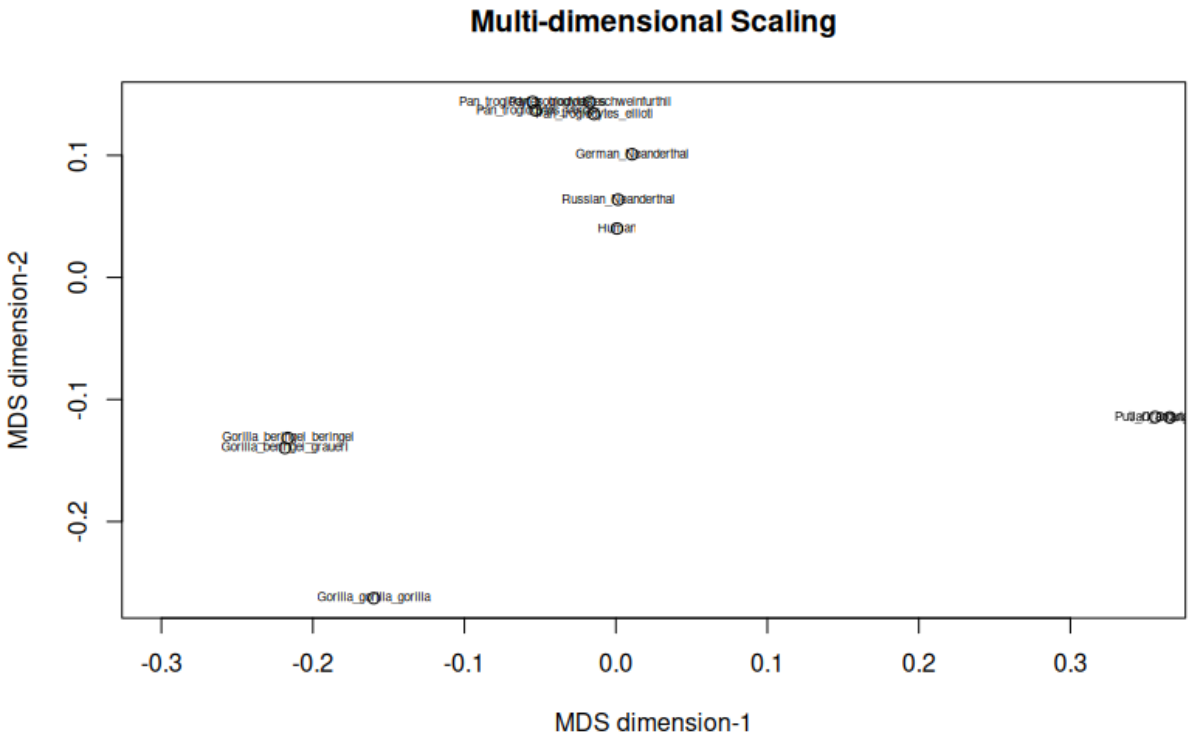
From UPGMA, Pan Troglodytes ellioti have the least bootstrap support with the grouping of pan troglodytes schweinfurthii and troglodytes.

5.

Humans are most closely related to Russian and German Neanderthals, and are next closest to Troglodytes.

E. Multidimensional Scaling. Now we visualize the relationships between the species in a 2d or 3d space using multidimensional scaling (MDS). MDS is a computational visualization technique that creates an artificial space such that the distance between the objects in the space is close to the distances in the original distance matrix.

```
# 2d MDS viz
locs<-cmdscale(as.dist(myDist))
x<-locs[,1]
y<-locs[,2]
plot(x,y,main="Multi-dimensional Scaling",xlab="MDS dimension-1",ylab="MDS
dimension-2", xlim=c(-.3,.35)
text(x,y,rownames(locs),cex=1.5)
```

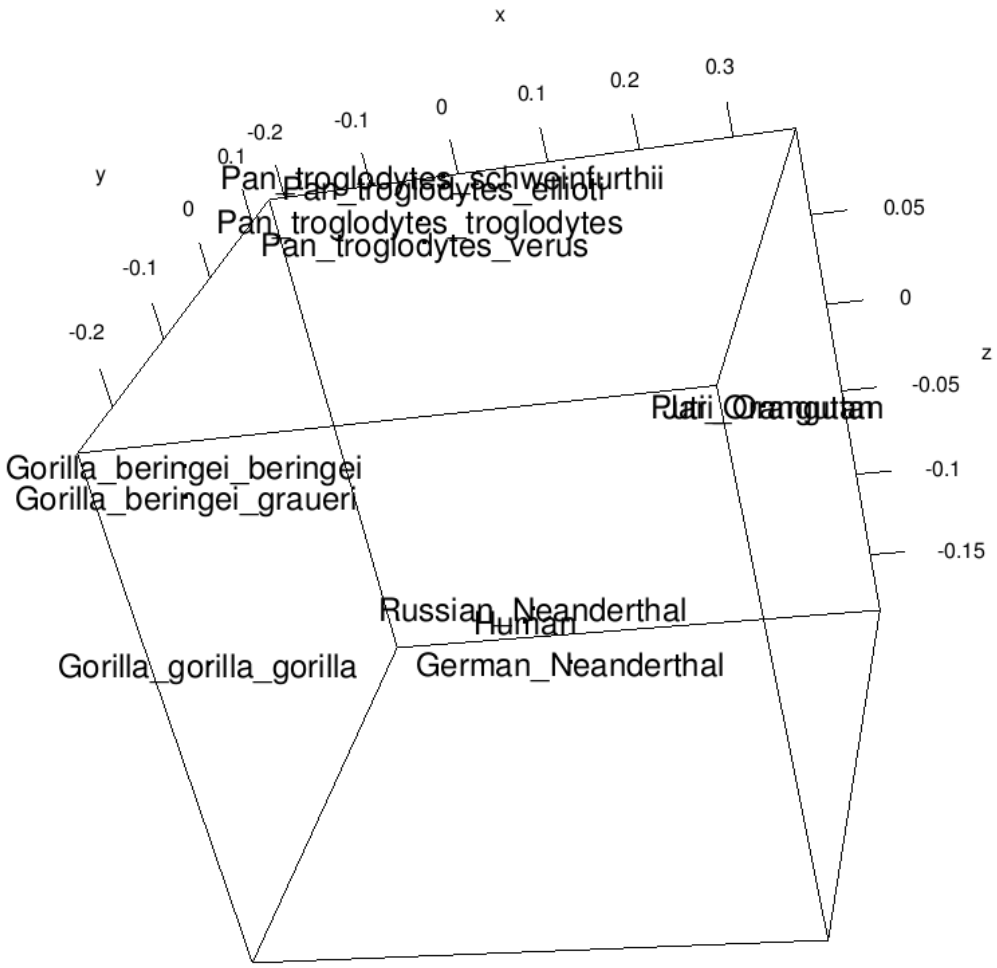


Note: Changed cex to 0.5 instead of 1.5

Do a 3d MDS analysis. Requires installation of rgl for plot3d. Show your plots. **1.** Do you think specifying 2 or 3 dimensions in MDS is better and why? 2. How do the relationships between species compare when 3d MDS is used versus sany of the tree-based visualizations?

```
library(rgl)
```

```
locs<-cmdscale(as.dist(myDist),k=3)
x<-locs[,1]
y<-locs[,2]
z<-locs[,3]
plot3d(x,y,z)
text3d(x=x,y=y,z=z,texts=rownames(locs),cex=1.5)
play3d(spin3d(axis=c(0,1,1), rpm=3), duration=30)
```



1.

In this example, specifying 3d is likely the better option. Overall, the 2d option works fine for this instance, since the two-dimensions does a well-enough job at capturing the distances between species. Where 3D has advantage in this case is that it more accurately displays the distance of humans and neanderthals with the other species.

2.

The 3D MDS view allows for a comparison between individual species to any other species. For the tree view, there is no direct comparison between Humans and Puti Orangutans, for example. The tree view allows for viewing the species in a hierarchical sense, and the 3D MDS view allows for a pairwise comparison between any species.