```
.libPaths(c("~/R/x86_64-pc-linux-gnu-library/4.4/"));
library(Biostrings)
```

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

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

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

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, 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, saveRDS, setdiff,
##     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:utils':
##
##     findMatches
```

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

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

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

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

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

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

```
library(ape)
```

```
##
## Attaching package: 'ape'
```

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

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:IRanges':
##
##     shift
```

```
## The following objects are masked from 'package:S4Vectors':
##
##     first, second
```

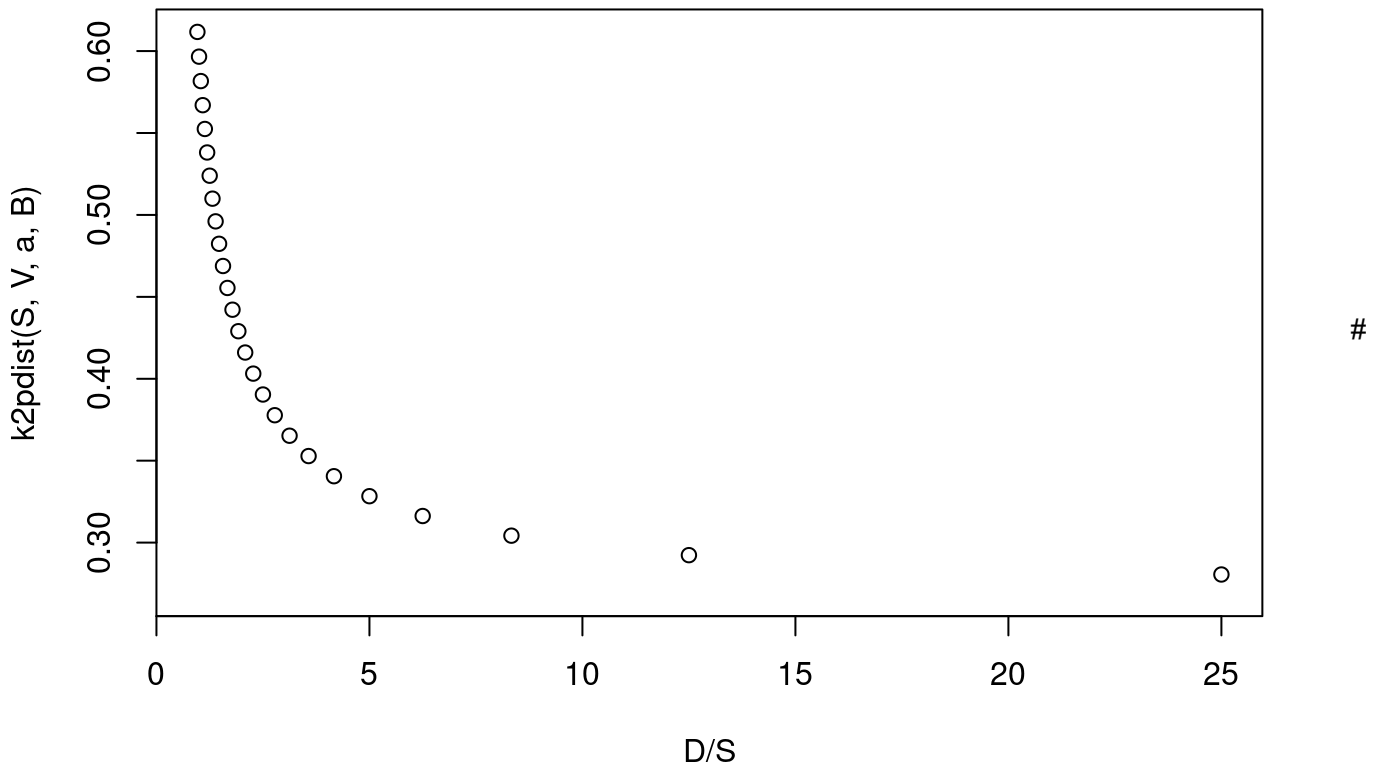
```
library(readr)
```

Part 1

Plotting K2P Distance

```
k2pdist <- function(S, V, a, B){
    #Given: S, the total number of transitions; V, the total number of transversions; a, the pro
bability of a transition; and B, the probability of a transversion
    #Returns: the K2P distance d.

    d <- (-0.5 * log(1 - (2*S) - V)) - (-0.25 * (1 - (2*V)))
    return(d)
}
```

```
# For a fixed D and set of parameters, plot how K2P distance changes as S and V change
D <-0.25 # D = S + V
a <- 0.5
B <- 0.5

step <- 0.01
S <- seq(0, D + step, step)
V <- D - S

plot(D / S, k2pdist(S, V, a, B));
```



#

Part 2 - Computing Phylogenetic Trees

2.1 - Aligning sequences

- Ran mafft_primates_msa.sh to align all sequences in primate_mtdna.fasta
- See runtime of MAFFT MSA below:

```
### COMPUTED MAFFT MSA FOR primate_mtdna.fasta and saved output to ALIGNED_primate_mtdna.fasta
### Ran sh mafft_primates_msa.sh
### Time:
# real    0m11.898s
# user    0m10.323s
# sys     0m1.538s
```

```
# Load in aligned strings
msa_path <- "~/amc6xa/comp_bio_a06/ALIGNED_primate_mtdna.fasta"
msa_seqs <- readDNAStringSet(msa_path)
```

```
msa_seqs # 17,289 columns in the MSA
```

```
## DNAStringSet object of length 7:
##     width seq                                               names
## [1] 17289 GTTTATGTAGCTTAAACCCCACC...CC---------------------- Macaca_sinica
## [2] 17289 GTTTATGTAGCTTA---CCTCCC...---------------------- Gorilla_gorilla
## [3] 17289 GTTTATGTAGCTTA--TTCTATC...---------------------- Pongo_abelii
## [4] 17289 GTTTATGTAGCTTA---CCCCCT...---------------------- Pan_troglodytes
## [5] 17289 GTTTATGTAGCTTA---CCTCCT...---------------------- Homo_sapiens
## [6] 17289 GTTAATGTAGCTTA---AATTAT...TCCATAATAATCACAAATCCCAA Lemur_catta
## [7] 17289 GTTTATGTAGCTTA---CCCCCT...---------------------- Pan_paniscus
```

2.2 - Computing Distance Matrix and Computing Trees

- Used given code to compute distance matrix D
- Wrote BuildUPGMA to return an UPGMA Phylogenetic tree in Newick format
- Compared BuildUPGMA() to ape::nj neighbor joining tree

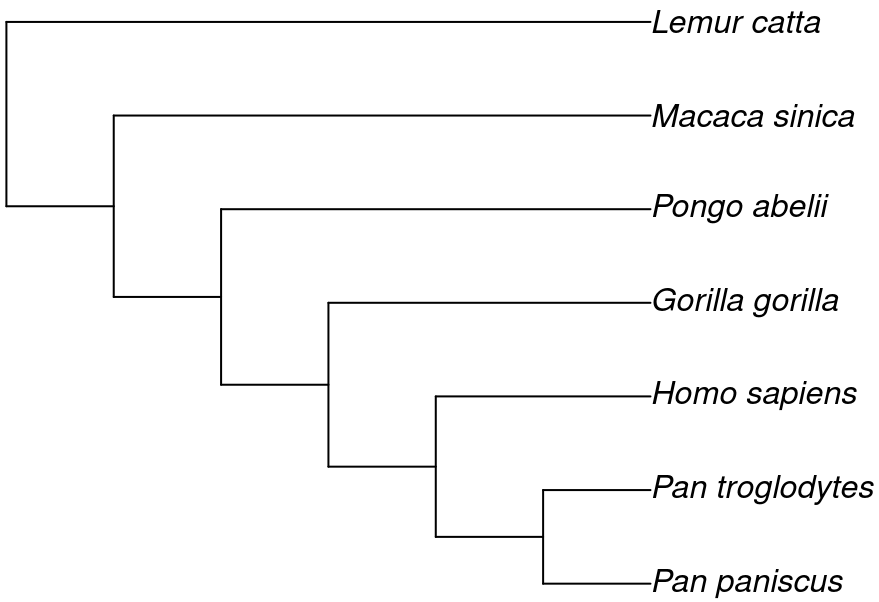
```
### Code given as part of assignment for computing pairwise alignment distances
msa <- Biostrings::readDNAMultipleAlignment(msa_path, "fasta")
D <- pwalign::stringDist(as(msa, "DNAStringSet"))
```

```
# Builds an UPGMA tree, given a distance matrix
# Ignore branch lengths.
# D = levenshtein distance matrix
BuildUPGMATree = function (D_mat) {
  # Assume D_mat is square matrix. Make its upper tri + diag = Inf
  # working only with lower triangular matrices b/c of min()
  D_mat <- as.matrix(D_mat)
  N <- dim(D_mat)[1]
  newick <- NULL

  # Create mapping bc index in D_mat and species name
  name_map_D_mat <- data.table(
    index = seq(1, N),
    name = rownames(D_mat)
  ); setkey(name_map_D_mat, index) # D_mat row/col index to row/col name lookup table

  # Rename all col and row names to be their indices in D_mat
  rownames(D_mat) <- seq(1, N); colnames(D_mat) <- seq(1, N)

  # Initialize UPGMA
  D_prev <- D_mat
  D_prev[upper.tri(D_prev, diag=T)] = Inf


  for(step in 1:N){
    # Get min of last iteration's matrix (indices of min)
    min_D_prev <- which(D_prev == min(D_prev), arr.ind = TRUE)[1, ] # get first if tie
    min_names <- rownames(D_prev)[min_D_prev]
    i <- min_D_prev[1]; i_name <- rownames(D_prev)[i]
    j <- min_D_prev[2]; j_name <- rownames(D_prev)[j]
    curr_clust_name <- paste("c(", i_name, ",", j_name, ")", sep="") # name + name / name + (nam
e, name)/ (name, name) + (name, name) / any combination or nesting

    # Get rid of rows i and j in D_prev to build D_next
    mask <- rep(T, dim(D_prev)[1])
    mask[min_D_prev] = F
    D_next <- as.matrix(D_prev[mask, mask]) # as.matrix prevents errors when D_prev[mask, mask]
is a single element
    if(sum(mask) == 1){
      # If we are here, then the masking removed the col name from the one elet in D_next. add i
t back
      rownames(D_next) = rownames(D_prev)[mask]; colnames(D_next) = colnames(D_prev)[mask];
    }

    # Add distance between everything in (i, j) and each element to D_next
    # Not the most efficient way, but I will use D_mat for this computation each time
    # i_D_mat <- name_map_D_mat[i_name]$name[1]
    # j_D_mat <- name_map_D_mat[j_name]$name[1]

    D_next_bottom_row <- c()
    curr_clust_all_nodes <- c(eval(parse(text = curr_clust_name)))
    for(index in 1:dim(D_next)[1]){
```

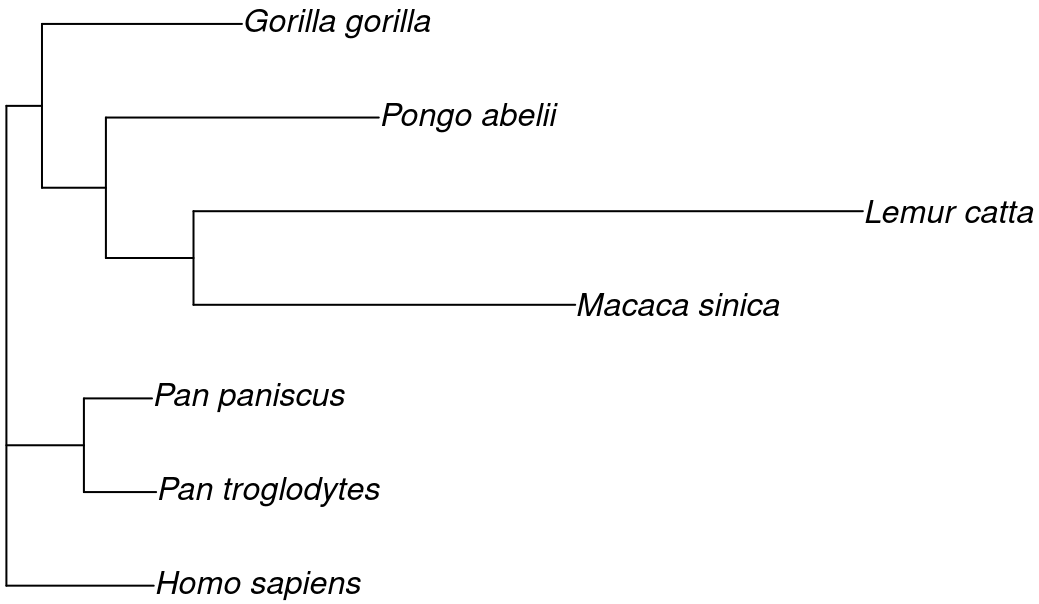
```
      # Compute dist between all nodes at index and all nodes in curr cluster
      index_all_nodes <- c(eval(parse(text = rownames(D_next)[index])))

      total_dist <- 0
      num_nodes <- 0
      for(outer in curr_clust_all_nodes){
        for(inner in index_all_nodes){
          total_dist <- total_dist + D_mat[outer,inner]
          num_nodes <- num_nodes + 1
        }
      }
      curr_dist <- total_dist / num_nodes
      D_next_bottom_row <- c(D_next_bottom_row, curr_dist)
    }

    D_next <- rbind(D_next, D_next_bottom_row);
    D_next <- cbind(D_next, Inf)

    rownames(D_next)[dim(D_next)[1]] = curr_clust_name
    colnames(D_next)[dim(D_next)[1]] = curr_clust_name

    if(dim(D_next)[1] == 2){
      # If here, then we can join the last two nodes
      newick <- paste("c(", rownames(D_next)[2], ",", rownames(D_next)[1], ")", sep = "")
      newick <- gsub('c','',newick)
      newick <- paste(newick, ";", sep = "")
      # replace each num in newick with name_map_D_mat[num]
      newick_list <- c()
      for(char in strsplit(newick, "")[[1]]){
        if(!grepl("[^0-9]", char) == T){
          char <- name_map_D_mat[as.numeric(char)]$name[1]
        }
        newick_list <- c(newick_list, char)
      }
      newick_names <- paste(newick_list, collapse='')
      return(newick_names) # this but with no "c"s
    }

    D_prev <- D_next
  }
}
```

In UPGMA, the branch length between nodes/clusters A and B and their branch point is 1/2 dist(A, B). I think this means all leaf nodes should be at the same position, whicih is what my function is doing without explicitly computing lengths…

```
upgmaTree = BuildUPGMATree(D)
njTree = ape::nj(D)
plot(read.tree(text=upgmaTree))
```


Lemur catta
Macaca sinica
Pongo abelii
Gorilla gorilla
Homo sapiens
Pan troglodytes
Pan paniscus

```
plot(njTree)
```

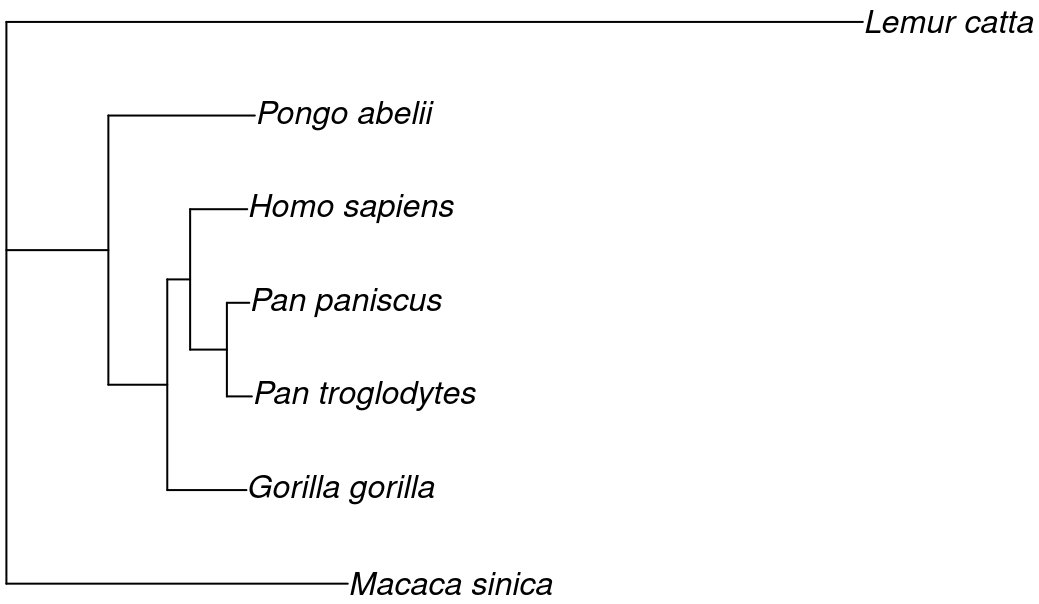


####

2.3 - Using iqtree to build a maximum likelihood tree from the same data.

Built using iqtree_slurm.sh

```
# Output from iqtree webserver:
iqtree_newick <- read_file("primate_mtdna_IQTREE_OUT/primate_mtdna_IQTREE_OUT_.treefile")
plot(read.tree(text=iqtree_newick))
```



2.4 - Compare Trees

Compare the trees. How are they similar and different, in terms of topologies, branch lengths, and rootedness? Do your mtDNA trees support the modern view that chimpanzees are more similar to humans than to other apes, or the older view that the apes form a monophyletic clade separate from humans?

Answer: My UPGMA and ape's neighbor-joining trees share similarities. In both trees, Lemur catta and Macaca sinica are both connected by the same most recent common ancestor (MRCA), as are both Pan species. However, the neighbor-joining tree assumes one MRCA between Homo sapiens, the Pan genus, and the remaining primates. UPGMA keeps this same idea that Gorilla, Homo, and Pan are more related to one another than the remaining primates, but its structure assumes more common ancestors in-between.

There are multiple distances in the matrix D that are very close to one another, but UPGMA always chooses one minimum distance and infers one MRCA between that pair, which may be a less accurate representation of relatedness in cases where three species all have similar distances between one another.

The maximum likelihood tree is more similar to the UPGMA tree, where Macaca sinica and Lemur catta, and the rest of the 5 species all descend from the same 1 or 2 MRCA(s). In the maximum likelihood tree, the same MRCA is shared between Macaca, Lemur, and the rest of the 5 primates, and in UPGMA, this same general relationship holds, with one additional common ancestor between Lemur catta and Macaca sinica. The maximum likelihood tree is much more different compared to the neighbor-joining tree, which splits apart the 5 remaining primates into two groups, from one of which descends the MRCA for Lemur and Macaca.

Part 3 - Larger-scale alignment using SLURM

3.1

iqtree_slurm.sh uses MAFFT to compute the MSA between all sequences in `16s.fasta`. Then, it uses iqtree to compute a maximum-likelihood tree for all sequences in 16s.

In the cell below, I use some code I found online to plot the tree to a PDF, which makes such a large tree more readable. See `16s_iqtree.pdf`

```
iqtree_16s <- read_file("16s_IQTREE_OUT/16s_IQTREE_OUT_.treefile")
pdf("16s_iqtree.pdf", width=115, height=150)
plot(read.tree(text=iqtree_16s))
dev.off()
```

```
## png
##   2
```

3.2

Question: Based on what you know about how evolutionary rates differ among different types of sequences, why do you think could be useful to use mtDNA for a phylogeny of primates, but 16S for a phylogeny of bacteria and archaea?

Answer: Research shows that that mutation rates in mtDNA are significantly higher (20-50x) than nuclear DNA mutation rates. Assuming the strength of selection on mutations is equal between the two genomes, I would expect mtDNA to evolve more quickly than nuclear/non-mt DNA, which includes the ribosomal 16s sequence. For organisms with a small number of offspring and a relatively large generation times, I can see how comparing mtDNA between groups might reveal more differences, and thus possibly providing more information about evolutionary relationships, than nuclear DNA.

On the otherhand, for an organism like bacteria, with very short generation times, there are simply more opportunities for mutations to arise in 16s sequences over a given timespan, possibly making the evolutionary rates between bacterial 16s dna and primate mtDNA more comparable. Analyzing bacterial mtDNA may show too many mutations, many of which are not fixed in populations, resulting in noisier data.