732A51 Bioinformatics

Lab 2

Johannes Hedström, Mikael Montén

STIMA
Institutionen för datavetenskap
Linköpings universitet

2024-11-25

Contents

1 Question 1: DNA sequence acquisition and simulation

In this exercise you will perform statistical analysis of three nucleotide data sets. First download the sequences from GenBank and save them in a fasta file. For this use the provided R script, 732A51 BioinformaticsHT2023 Lab02 GenBankGetCode.R. This is a dataset of the RAG1 gene sequences from 33 lizard species. You are encouraged to read in detail the references in the script as they indicate many useful tools. Explore the dataset using the tools provided by the ape and seqinr packages. Take note of the lengths of all the sequences and the base composition.

```r
library(ape)
library(seqinr)

library(phangorn)



## Gene bank accession numbers taken from http://www.jcsantosresearch.org/Class_2014_Spring_Comparative/
lizards_accession_numbers <- c("JF806202", "HM161150", "FJ356743", "JF806205",
                               "JQ073190", "GU457971", "FJ356741", "JF806207",
                               "JF806210", "AY662592", "AY662591", "FJ356748",
                               "JN112660", "AY662594", "JN112661", "HQ876437",
                               "HQ876434", "AY662590", "FJ356740", "JF806214",
                               "JQ073188", "FJ356749", "JQ073189", "JF806216",
                               "AY662598", "JN112653", "JF806204", "FJ356747",
                               "FJ356744", "HQ876440", "JN112651", "JF806215",
                               "JF806209")
lizards_sequences<-ape::read.GenBank(lizards_accession_numbers)
print(lizards_sequences)
```

```
## 33 DNA sequences in binary format stored in a list.
##
## Mean sequence length: 1982.879
##    Shortest sequence: 931
##     Longest sequence: 2920
##
## Labels:
## JF806202
## HM161150
## FJ356743
## JF806205
## JQ073190
## GU457971
## ...
##
## Base composition:
##     a     c     g     t
## 0.312 0.205 0.231 0.252
## (Total: 65.44 kb)
```

1.1 Question 1.1

Simulate an artificial DNA sequence dataset. It should contain 33 sequence. The lengths of the sequences should be the same as in the lizard dataset, i.e. for each real sequence simulate an artificial one. The simulation rule is as follows, each nucleotide is to be independently and randomly drawn from the distribution given by the base composition (frequencies) in the true lizard sequences. Save your dataset in a fasta format file. Remember to give unique names to your sequences. Report on the base composition in your simulated data.

```r
#library(help = ape)
#library(help = seqinr)
set.seed(123456790)
#ape::as.character.DNAbin(lizards_sequences[[1]])


artif_dna_seq <- function(org_seq){
  new_seq <- list()

  for(i in 1:length(org_seq)){
    comp_dist <- base.freq(org_seq[i][1]) # extract base compositions
    seq_len <- length(org_seq[[i]]) # extract length of sequence
    # sample according to individual base comp distribution and length
    new_seq[[i]] <- sample(c("a", "c", "g", "t"), size = seq_len, replace = TRUE, prob = comp_dist)
  }
  # set unique names according to original sequence
  names(new_seq) <- paste0("ARTIFICAL_",names(org_seq))
  return(new_seq)
}

artif_liz_seq <- artif_dna_seq(lizards_sequences)
# save dataset as fasta
#seqinr::write.fasta(sequences =artif_liz_seq,
#                    names = names(artif_liz_seq),
#                    file.out = "artifical_lizard_seqs.fasta")

# report on base composition
artif_liz_seq_DNAbin <- as.DNAbin(artif_liz_seq)

seq_compare <- rbind(base.freq(lizards_sequences),base.freq(artif_liz_seq_DNAbin))
rownames(seq_compare) <- c("Lizards sequences", "Artif. lizards sequences")
knitr::kable(seq_compare, caption="Base compositions")
```

Table 1: Base compositions

	a	c	g	t
Lizards sequences	0.3121454	0.2052325	0.2307222	0.2518999
Artif. lizards sequences	0.3074960	0.2068006	0.2329029	0.2528005

Comparing the base compositions for the original lizards_sequences FASTA file and the artificially created lizards_sequences, we can see that the base compositions are nearly identical eachother with minimal differences between the sequences.

1.2 Question 1.2

Simulate a second artificial DNA sequence dataset. It should contain 33 sequence. The lengths of the sequences should be the same as in the lizard dataset, i.e. for each real sequence simulate an artificial one. First simulate a phylogenetic tree with 33 tips in phylo format (i.e. ape). Plot your resulting tree. For simulating the tree explore the functions of the ape, TreeSim or other Rpackages. Choose a simulation function and model yourself.Now simulate sequences on this using e.g. phangorn::simSeq(). Choose the sequence length yourself, but try to make it so that it will be comparable with the original lizards dataset. You need to also specify the Q matrix/the transition rate matrix. Choose one yourself, however try to make the stationary distribution equal to the base composition (frequencies) of the lizard sequences (look at EG Ch. 14:3:3). If you cannot obtain such a transition matrix, choose some another one. Save your dataset in a fasta format file. Remember to give unique names to your sequences. Report on the base composition in your simulated data. Comment on if it is whatyou expect.

```r
# simulate second artifical DNA seq with 33 sequences and same lengths
a_seq2 <- artif_dna_seq(lizards_sequences)

# simulate a phylogenetic tree with 33 tips in phylo format
a_seq_tree <- rtree(n = length(a_seq2))
# set node names to match sequences
a_seq_tree$tip.label <- names(a_seq2)
```

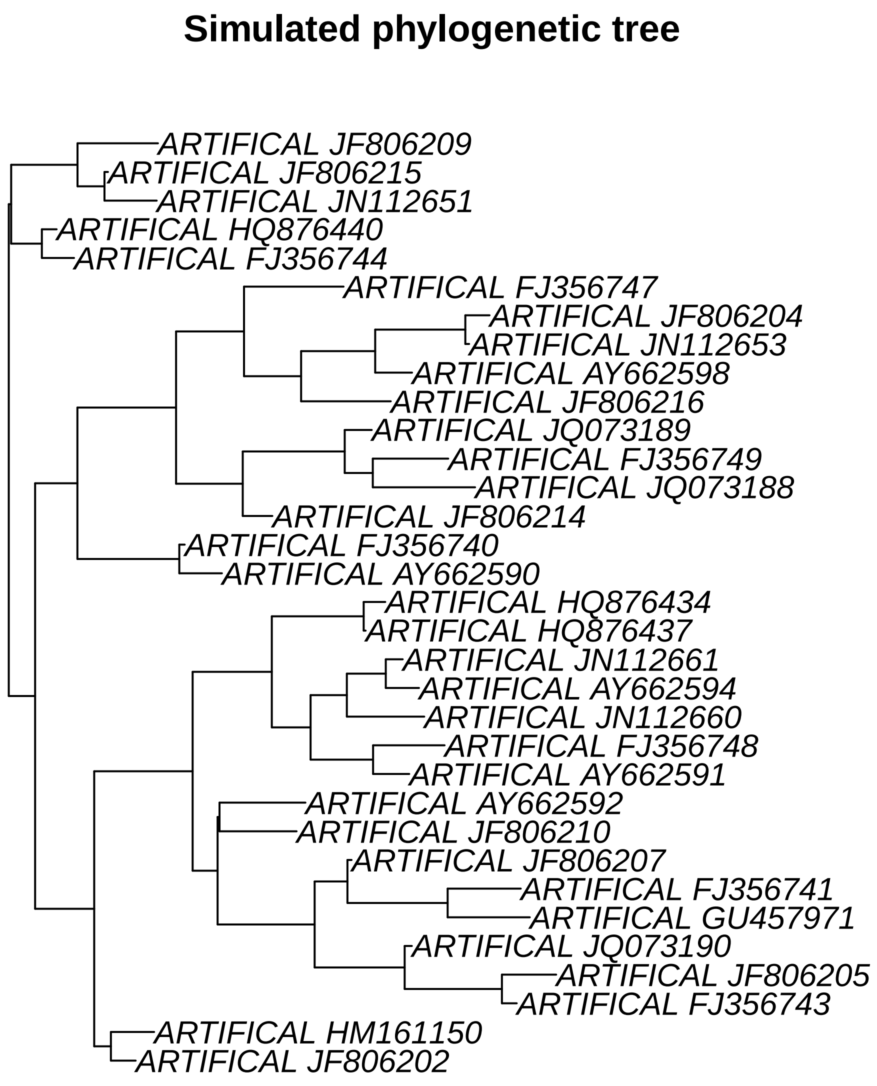


Figure 1: Phylogenetic tree of artifical DNA sequences

```r
as.DNAbin(a_seq2)
```

```
## 33 DNA sequences in binary format stored in a list.
##
## Mean sequence length: 1982.879
##    Shortest sequence: 931
##     Longest sequence: 2920
##
## Labels:
## ARTIFICAL_JF806202
## ARTIFICAL_HM161150
## ARTIFICAL_FJ356743
## ARTIFICAL_JF806205
## ARTIFICAL_JQ073190
## ARTIFICAL_GU457971
## ...
##
## Base composition:
##     a     c     g     t
## 0.310 0.203 0.234 0.252
## (Total: 65.44 kb)
```

As the mean DNA sequence length is 1982.879, this will be used to simulate sequences of length 1983. The base compositions shown above is what will be used as stationary distribution for the Q-matrix as well.

$$\vec{\phi}_0 = (0.310, 0.203, 0.234, 0.254)^T$$

$$\mathbf{Q} = \begin{bmatrix} q_{AA} & q_{AC} & q_{AG} & q_{AT} \\ q_{CA} & q_{CC} & q_{CG} & q_{CT} \\ q_{GA} & q_{GC} & q_{GG} & q_{GT} \\ q_{TA} & q_{TC} & q_{TG} & q_{TT} \end{bmatrix}$$

So that diagonal elements satisfy $q_{ii} = -\sum_{i \neq j} q_{ij}$, which means row sums = 0. According to https://en.wikipedia.org/wiki/Models_of_DNA_evolution#Ergodicity, a stationary distribution π satisfies $\pi Q = 0 \implies \vec{\phi}_0 Q = 0$ which is the result to look for when constructing the Q-matrix.

Creating a symmetric Felsenstein model according to https://en.wikipedia.org/wiki/Models_of_DNA_evolution#F81_model_(Felsenstein_1981).

$$\mathbf{Q} = \begin{bmatrix} * & \phi_0^C & \phi_0^G & \phi_0^T \\ \phi_0^A & * & \phi_0^G & \phi_0^T \\ \phi_0^A & \phi_0^C & * & \phi_0^T \\ \phi_0^A & \phi_0^C & \phi_0^G & * \end{bmatrix}$$

```r
# sequence length
n <- 1983

# base composition
phi <- base.freq(as.DNAbin(a_seq2))

# initialize Q
Q <- matrix(0, nrow = 4, ncol = 4)
colnames(Q) <- c("a","c","g","t")
rownames(Q) <- c("a","c","g","t")

# fill each non-diagonal with the base comp
for (i in 1:4) {
  for (j in 1:4) {
    if (i != j) {
      Q[i,j] <- phi[j]
    }
  }
}

# fill diagonal to ensure row sums to 0
diag(Q) <- -rowSums(Q)

print(Q)
```

```
##            a         c         g         t
## a -0.6899213  0.2031482  0.2343089  0.2524643
## c  0.3100787 -0.7968518  0.2343089  0.2524643
## g  0.3100787  0.2031482 -0.7656911  0.2524643
## t  0.3100787  0.2031482  0.2343089 -0.7475357
```

```
# confirm the assumption holds
phi %*% Q
```

```
##                 a            c             g t
## [1,] 1.387779e-17 6.938894e-18 -6.938894e-18 0
```

The Q-matrix constructed according to the Felsenstein model works good as a transition matrix, which is evident that the result is 0 or nearly 0 for all bases when computing $\vec{\phi}_0 Q$.

```
# simulate sequences
phyl_seq <- simSeq(a_seq_tree, l = n, Q = Q, bf = phi, type = "DNA")
# save as FASTA
#write.phyDat(phyl_seq, file = "simulated_phylogeny_sequences.fasta", format = "fasta")

as.DNAbin(phyl_seq)
```

```
## 33 DNA sequences in binary format stored in a matrix.
##
## All sequences of same length: 1983
##
## Labels:
## ARTIFICAL_JF806202
## ARTIFICAL_HM161150
## ARTIFICAL_FJ356743
## ARTIFICAL_JF806205
## ARTIFICAL_JQ073190
## ARTIFICAL_GU457971
## ...
##
## Base composition:
##     a     c     g     t
## 0.313 0.202 0.233 0.252
## (Total: 65.44 kb)
```

```
seq_compare <- rbind(base.freq(as.DNAbin(phyl_seq)),base.freq(as.DNAbin(a_seq2)))
rownames(seq_compare) <- c("Phylogeny simulated sequence", "Second simulated sequence")
knitr::kable(seq_compare, caption="Base compositions")
```

Table 2: Base compositions

	a	c	g	t
Phylogeny simulated sequence	0.3132383	0.2017757	0.2329956	0.2519904
Second simulated sequence	0.3100787	0.2031482	0.2343089	0.2524643

Comparing the base compositions of these simulated data, they are closely aligned each other. This is the result that we expected so the phylogeny simulations seem to be working well.

2 Sequence analysis

```
# file from Ying
codon_ORF <- read.csv('codon_and_ORF.csv')
```

2.1 Question 2.1

Report some basic statistics on each sequence dataset: individual base composition, GC content,CG, AT content. Also translate your sequences into protein sequences (see Lab 1) and report on the amino acid composition. In your simulated sequences, how many times did you observe a stop codon inside your sequence? Does this occur in your true sequences? Comment.

```
library(stringr)

# function to calculate basic stats from a DNA sequence
dna_stats <- function(seq){

  alf_seq <- as.character(seq) # from binary to letters
  sequence_str <- paste(unlist(alf_seq), collapse = "")# 1 long sequence

  # Create the dinucleotides for the CpG calculations
  dinucleotides <- substring(sequence_str, 1:(nchar(sequence_str) - 1), 2:nchar(sequence_str))
  # Counting the dinucleotide
  counts <- table(dinucleotides)

  # Calculate CpG content (C followed by G)
  if('cg' %in% names(counts)){ # to be sure that 'cg' is in the sequence
    CpG <- counts['cg'] / sum(counts)
  } else {
    CpG = 0
  }
  base_comp <- base.freq(seq[1]) # base comp
  GC <- GC.content(seq[1]) # GC

  if('at' %in% names(counts)){ # to be sure that 'at' is in the sequence
    ApT <- counts['at'] / sum(counts)
  } else {
    ApT = 0
  }


  return(t(c('Species'=names(seq),base_comp[1],base_comp[2],base_comp[3],base_comp[4],GC,ApT, CpG)))
}

df <- as.data.frame(dna_stats(lizards_sequences[1]))

df <- rbind(df,dna_stats(artif_liz_seq_DNAbin[1]))
colnames(df)[6:8] <-c('GC','ApT', 'CpG')
df[ , 2:8] <- sapply(df[ , 2:8], as.numeric)
```

```
df[ , 2:8] <- round(df[ , 2:8], digits = 5)
knitr::kable(df)
```

Species	a	c	g	t	GC	ApT	CpG
JF806202	0.28987	0.20261	0.24373	0.26379	0.44634	0.07723	0.01304
ARTIFICAL_JF806202	0.27856	0.19739	0.24449	0.27956	0.44188	0.07422	0.05517

The proportions for the different stats looks to be close to the same for the specie JF806202 except for the CpG which is much higher for the artificial.

```
df1 <- matrix(0,ncol=8,nrow=33)
df2 <- matrix(0,ncol=8,nrow=33)
for(i in 1:33){
df1[i,] <- dna_stats(lizards_sequences[i])
df2[i,] <- dna_stats(artif_liz_seq_DNAbin[i])
}

df1 <- as.data.frame(df1)
df2 <- as.data.frame(df2)

colnames(df1)[2:8] <-c('a','c','g','t','GC','ApT', 'CpG')
df1[ , 2:8] <- sapply(df1[ , 2:8], as.numeric)
df1[ , 2:8] <- round(df1[ , 2:8], digits = 5)
colnames(df2)[2:8] <-c('a','c','g','t','GC','ApT', 'CpG')
df2[ , 2:8] <- sapply(df2[ , 2:8], as.numeric)
df2[ , 2:8] <- round(df2[ , 2:8], digits = 5)


mean1 <- colMeans(df1[,2:8])
mean2 <- colMeans(df2[,2:8])
df3 <- rbind(mean1, mean2)

rownames(df3) <- c('Mean of true sequences', 'Mean of sampled sequences')

knitr::kable(df3)
```

	a	c	g	t	GC	ApT	CpG
Mean of true sequences	0.3078355	0.2049530	0.2321824	0.2550285	0.4371358	0.0685985	0.0106855
Mean of sampled sequences	0.3033052	0.2065394	0.2348561	0.2552997	0.4413964	0.0763303	0.0474115

```
# df3[,2]*df3[,3] expected cG
```

```
# Translate DNA sequences to protein sequences

translate_sequences <- function(seq) {
  alf_seq <- as.character(seq) # from binary to letters
  sequence_str <- paste(unlist(alf_seq), collapse = "") # 1 long sequence

  name <- names(seq) # picking out the name to get correct start pos
  start <- codon_ORF[codon_ORF$accession==name,2] -1 # 0-2 in the function
  protein <- translate(s2c(sequence_str), frame=start) # Translation table 1 (Standard Code)

  return(list(name=name,protein=protein))
}


# Translate all true sequences
protein_sequences <- lapply(1:33, function(i) translate_sequences(lizards_sequences[i]))
print(protein_sequences[1])
```

```
## [[1]]
## [[1]]$name
## [1] "JF806202"
##
## [[1]]$protein
##   [1] "H" "A" "L" "R" "T" "A" "E" "K" "S" "L" "L" "P" "G" "Y" "H" "P" "F" "E"
##  [19] "W" "K" "P" "P" "L" "K" "N" "V" "S" "S" "N" "T" "E" "V" "G" "I" "I" "D"
##  [37] "G" "L" "S" "G" "I" "Q" "H" "L" "V" "D" "D" "Y" "P" "V" "D" "T" "I" "A"
##  [55] "K" "R" "F" "R" "Y" "D" "A" "A" "L" "V" "S" "A" "L" "M" "D" "M" "E" "E"
##  [73] "D" "I" "L" "E" "G" "L" "K" "S" "Q" "D" "M" "D" "D" "Y" "L" "K" "X" "P"
##  [91] "F" "T" "V" "V" "I" "K" "E" "S" "C" "D" "G" "M" "G" "D" "V" "S" "E" "K"
## [109] "H" "G" "C" "G" "P" "A" "V" "P" "E" "K" "A" "V" "R" "F" "S" "F" "T" "L"
## [127] "M" "S" "I" "S" "V" "T" "H" "G" "N" "A" "S" "I" "R" "I" "F" "E" "E" "N"
## [145] "K" "P" "N" "S" "E" "L" "C" "C" "K" "P" "L" "C" "L" "M" "L" "A" "D" "E"
## [163] "S" "D" "H" "E" "T" "L" "T" "A" "I" "L" "S" "P" "L" "V" "A" "E" "R" "E"
## [181] "A" "M" "K" "D" "S" "V" "L" "I" "L" "D" "M" "A" "G" "I" "P" "R" "M" "F"
## [199] "K" "F" "I" "F" "R" "G" "T" "G" "Y" "D" "E" "K" "L" "V" "R" "E" "V" "E"
## [217] "G" "L" "E" "A" "S" "G" "S" "T" "Y" "I" "C" "T" "L" "C" "D" "A" "T" "R"
## [235] "L" "E" "A" "S" "Q" "N" "L" "I" "L" "H" "S" "I" "T" "R" "N" "H" "V" "E"
## [253] "N" "L" "E" "R" "Y" "E" "V" "W" "R" "S" "N" "P" "Y" "R" "E" "T" "V" "D"
## [271] "E" "L" "R" "D" "R" "V" "K" "G" "V" "S" "A" "K" "P" "F" "I" "E" "T" "V"
## [289] "P" "S" "I" "D" "A" "L" "H" "C" "D" "I" "G" "N" "A" "A" "E" "F" "Y" "K"
## [307] "I" "F" "Q" "F" "E" "I" "G" "E" "V" "Y" "K" "N" "R" "D" "A" "S" "K" "E"
## [325] "E" "R" "K" "R" "W" "Q" "S" "A"
```

```
sim <- read.fasta('sim.fasta') # simulated proteins sequances

table(sim$ARTIFICAL_JF806202_1)
```

```
##
##  *  a  c  d  e  f  g  h  i  k  l  m  n  p  q  r  s  t  v  w  x  y
## 20 19 14 11 13 11 20 10 22 12 28  5 13  7  6 33 33 19 16  7  1 13
```

The number of stop codons in the simulated sequence for JF806202 after translations to proteins are 20, which can be seen as the number below the star(*) in the table above. The true sequence have 0 stop codons, the difference here is that in the simulated sequence there is a possibility to sample nucleotides that create stop codons(TAA, TAG, and TGA) without any constraints which there is likely to be for the true sequence where the coding regions are complete and translated into functional proteins.

```r
freq_all <- sapply(protein_sequences, function(x) x$protein) # picking out only protein

freq_true <- table(unlist(freq_all))

knitr::kable(t(freq_true), caption='Amino acid frequency for all true sequences')
```

Table 5: Amino acid frequency for all true sequences

A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	X	Y
1330	773	1167	1785	857	1061	764	1152	1828	2043	504	747	1020	615	1295	1548	1081	1403	186	36	597

```r
freq_sim <- table(unlist(sim))

knitr::kable(t(freq_sim), caption='Amino acid frequency for all simulated sequences')
```

Table 6: Amino acid frequency for all simulated sequences

*	a	c	d	e	f	g	h	i	k	l	m	n	p	q	r	s	t	v	w	x	y
1356	1026	589	711	887	647	1151	658	1400	1170	1834	390	947	919	783	1880	1835	1354	1223	285	19	760

2.2 Question 2.3

Align your sequences using software of your choice (a starter for R: https://stackoverflow.com/questions/4497747/how-to-perform-basic-multiple-sequence-alignments-in-r, you can also look what Biopython, BioPerl offer, use the Clustal family of programs or something else of your choice). Choose a distance measure between sequences, calculate for each alignment the distances between all pairs of sequences. Then, plot heatmaps visualizing the distances. Comment on what you can observe.

```r
#BiocManager::install("msa")
library(msa)
library(stringdist)
library(Biostrings)
```

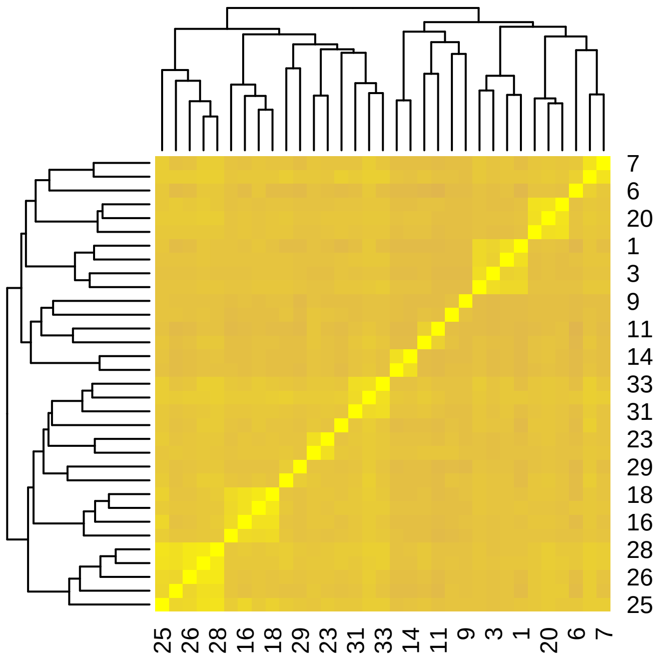
Identity measure:

$$Distance = 1 - \frac{\text{Number of identical positions}}{\text{Total number of positions}}$$

where a distance of 0 = identical and 1 is no identical positions.

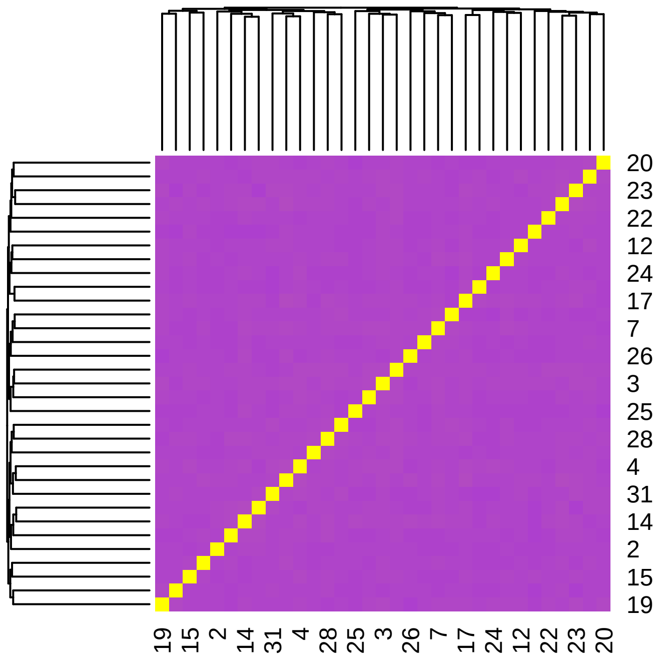
```r
# align with MUSCLE as its quite fast and good for dna sequenecs
true_msa <- msa('lizard_seqs.fasta','Muscle'  ,type='dna')
sim_msa <-  msa('artifical_lizard_seqs.fasta','Muscle', type='dna')

# Convert MultipleAlignment object to a character vector
```

```r
#true_seqs <- as.character(true_msa)
#sim_seqs <- as.character(sim_msa)

# this took too much time (Levenshtein Distance)
#true_dist <- stringdistmatrix(true_seqs, true_seqs, method = "lv")
#sim_dist <- stringdistmatrix(sim_seqs, sim_seqs, method = "lv")

# converting to seqinr object of class alignment
true_seqs <- msaConvert(true_msa,type='seqinr::alignment')
sim_seqs <- msaConvert(sim_msa,type='seqinr::alignment')

#  identity for distance matrices and not similarity
# and identity is also then the measure
true_dist <- as.matrix(dist.alignment(true_seqs,'identity'))
sim_dist <-   as.matrix(dist.alignment(sim_seqs,'identity'))


# changing color scale so its easier to compare the graphs
color_palette <- colorRampPalette(c('yellow', 'purple'))(100)

heatmap(true_dist,col=color_palette, breaks=seq(0,1,length.out=101), scale='none')
```



Low scores overall, small clusters of a bit more similar sequences like sequence 25,26 and maybe 28(hard to see when not all axis labels are presented).

```r
# Create a heatmap
heatmap(sim_dist,col=color_palette, breaks=seq(0,1,length.out=101), scale='none')
```



The simulated sequences are only similar to themselves and have a quite high score to every other sequences, the score is close to the same for all pairwise comparisons(drawn from the same probabilities so this is expected). It shows the lack of evolutionary complexity and diversity that naturally occur in biological data, the consistently low similarity scores within the true sequences highlight the genetic variability and evolutionary history that shape actual biological sequences.

3 Phylogeny reconstruction

3.1 Question 3.1

Construct (using algorithm and software of your choice) phylogenetic trees from the three multiple alignments (or distance matrices) done in Question 2.3. You might want to look at the functions offered by ape, phangorn (https://cran.r-project.org/web/packages/phangorn/vignettes/Trees.pdf) or go for some completely different software. Plot the inferred trees. Are the two based on the simulated data similar to expected? Perform a phylogenetic bootstrap analysis and report the bootstrap support for the individual clades, you can look at ape::boot.phylo().

```r
# construct phylogenetic trees from the sequences in question 2.3
true_dna <- as.DNAbin(true_seqs)
sim_dna <- as.DNAbin(sim_seqs)
true_dna_dist <- dist.dna(true_dna)
sim_dna_dist <- dist.dna(sim_dna, model="raw")

# using neighbor-join tree from ape
true_tree <- nj(true_dna_dist)
sim_tree <- nj(sim_dna_dist)
```

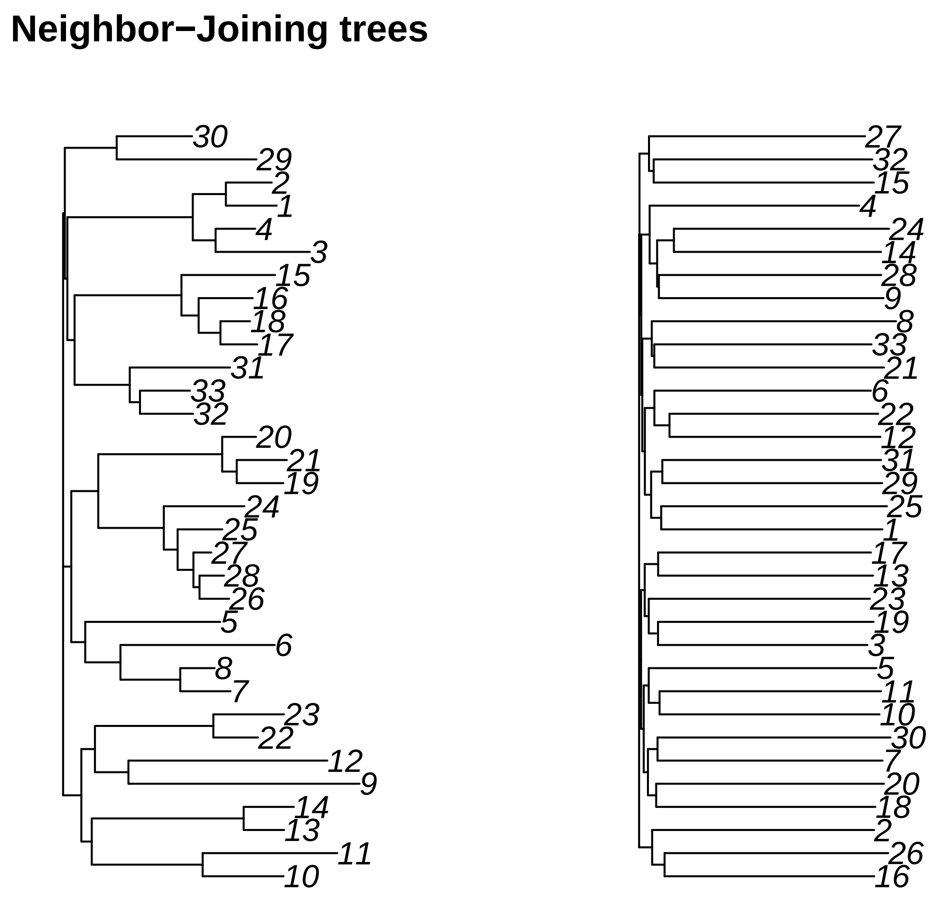


Figure 2: True sequence left, simulated sequence right

The simulated tree on the right from the simulated sequence is very different from the tree generated from the true sequence. This is expected as when calculating neighbor join trees, you construct new clustered nodes from just distance between tips. This differs from the true sequence which are derived from real evolutionary scenarios, which makes the sequences shaped by natural selection, regional constraints and other real world evolutionary settings.

```r
# phylogenetic bootstrap analysis
f <- function(x) nj(dist.dna(x, model = "raw"))
true_boot <- boot.phylo(true_tree,true_dna,f,B=1000)
```

```
## Running bootstraps:       100 / 1000Running bootstraps:       200 / 1000Running bootstraps:       300
## Calculating bootstrap values... done.
```

```r
sim_boot <- boot.phylo(sim_tree,true_dna,f,B=1000)
```

```
## Running bootstraps:       100 / 1000Running bootstraps:       200 / 1000Running bootstraps:       300
## Calculating bootstrap values... done.
```

```r
par(mfrow=c(1,2))
plot.phylo(true_tree, main="Bootstrap trees")
nodelabels(true_boot)
plot.phylo(sim_tree)
nodelabels(sim_boot)
```

```r
boots <- rbind(true_boot,sim_boot)
rownames(boots) <- c("T", "S")
colnames(boots) <- 1:31
knitr::kable(boots, caption = "Bootstrapped values for true and simulated sequences")
```

Table 7: Bootstrapped values for true and simulated sequences

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
T	NA	0	0	0	0	0	0	0	0	0	28	2	24	1	25	0	0	2	20	25	0	2	28	0	26	20	27	0	22	23	26
S	NA	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1	25	0	17	28	30	20	28	0	21	25	27	34	15	14	24

The bootstrap values show the proportion where specific group of specific clade appear in the bootstrapped trees. Generally the values are low for both sequences which indicates there is low support for the evolutionary relationship computed.

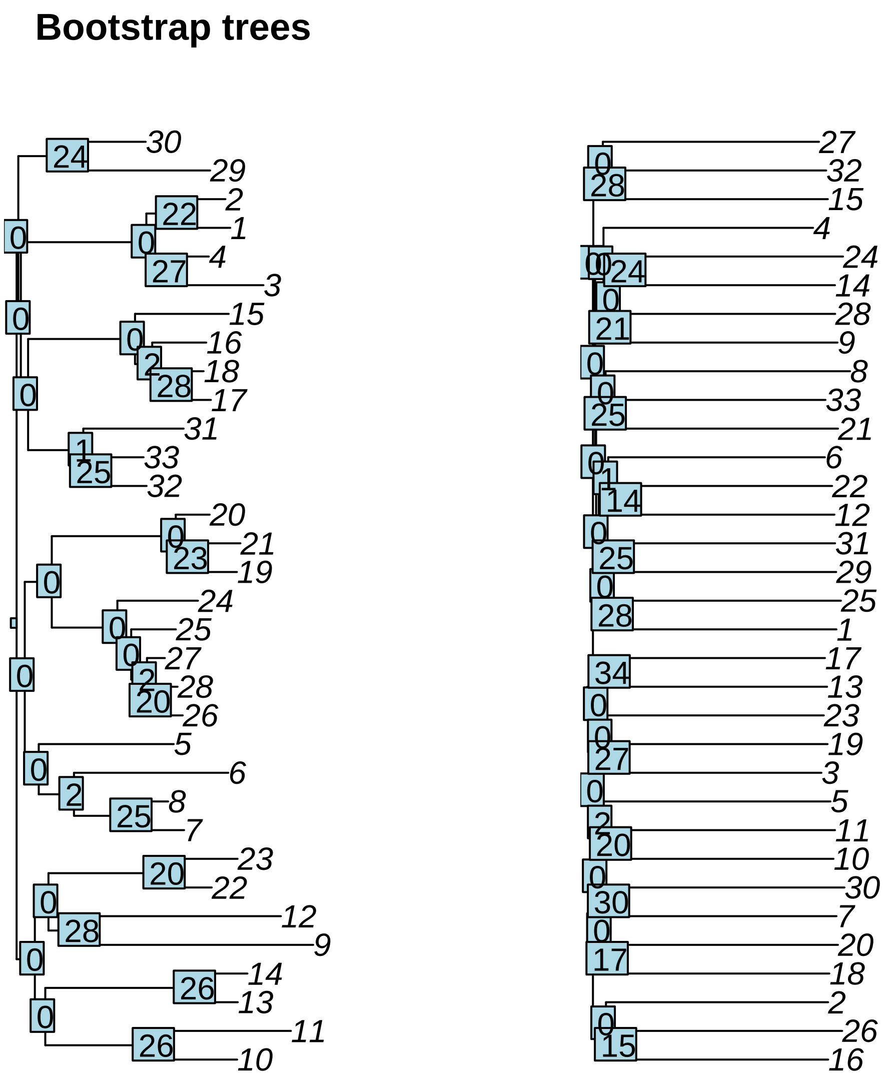


Figure 3: True sequence left, simulated sequence right