Bioinformatics Lab2, Group 3

Roshni Sundaramurthy, Prudhvi Peddmallu, Jiawei Wu, Zijie Feng

2018-11-26

Assignment 1

We start from the code in *732A51 BioinformaticsHT2018 Lab02 GenBankGetCode.R* and thereby create a fasta file with 33 original lizard sequences.

```
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)
ape::write.dna(lizards_sequences, file ="lizard_seqs.fasta", format = "fasta",
               append =FALSE, nbcol = 6, colsep = " ", colw = 10)

bin0 <- read.FASTA(file="lizard_seqs.fasta")
bin0
```

```
## 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

Based on the requirement, we write the following function to simulate the first artificial sequences in accordance to the original lizard sequences.

```
## simulation function
simulate2 <- function(raw){
  a <- as.data.frame(str_split(raw,""))
  nt <- unique(a)           # all exist nucleotides
  nt[,1] <- as.character(nt[,1])
  nm <- vector()            # corresponding numbers
  for (i in 1:nrow(nt)) {nm[i] <- str_count(raw,nt[i,1])}
  res <- sample(nt[,1],nrow(a),rep = TRUE, prob=nm/nrow(a))
  res
}
```

Creating the first 33 artificial DNA sequences depending on the raw 33 lizard sequences.

```
ss0 <- read.fasta(file="lizard_seqs.fasta",as.string=TRUE)
simulation <- list()
for (i in 1:33) {
  raw <- gsub(" ","", ss0[[i]])
  simulation[[i]] <- simulate2(raw)
}
n1 <- rep("ZJ",1,33)          # names ZJ11 ~ ZJ43
n2 <- 11:43; names <- vector()
for (i in 1:33) {names[i] <- paste0(n1[i],n2[i])}
names(simulation) <- names
ape::write.dna(simulation, file ="simulation.fasta", format = "fasta",
               append =FALSE, nbcol = 6, colsep = " ", colw = 10)
```

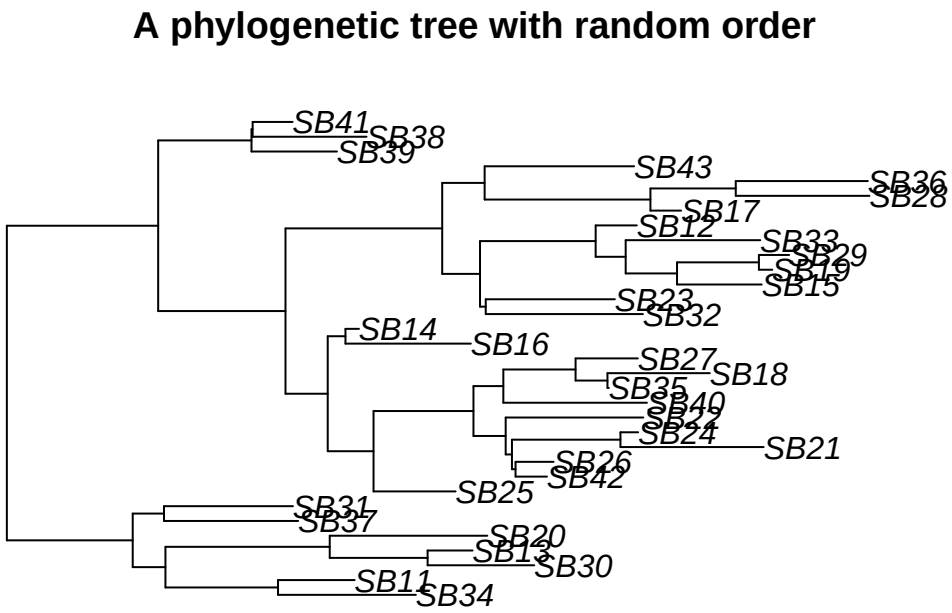
```
bin1 <- read.FASTA(file="simulation.fasta")
bin1
```

```
## 33 DNA sequences in binary format stored in a list.
##
## Mean sequence length: 1982.879
##    Shortest sequence: 931
##     Longest sequence: 2920
##
## Labels:
## ZJ11
## ZJ12
## ZJ13
## ZJ14
## ZJ15
## ZJ16
## ...
##
## Base composition:
##     a     c     g     t
## 0.312 0.205 0.230 0.252
## (Total: 65.44 kb)
```

1.2

Then create the second artificial DNA data with 33 sequence.

```
n1 <- rep("SB",1,33)            # names SB11 ~ SB43
n2 <- 11:43; names1 <- vector()
for (i in 1:33) {names1[i] <- paste0(n1[i],n2[i])}

set.seed(23333)           # create a phylogenetic tree with random order
mytree <- rtree(33, tip.label = names1)
plot(mytree)
title("A phylogenetic tree with random order")
```



Since there are totally 65408 nt in raw 33 lizard sequences, we create 33 sequences with length=2000 and the transition rate matrix from raw sequences.

```
## transition rate matrix from raw sequences
bin0 <- read.FASTA(file="lizard_seqs.fasta")
mcFit <- markovchainFit(data=as.character(bin0))
mat <- mcFit$estimate
a <-c(1,2,3,7)                  # avoid such nt as y, m, r, s
mat <- mat@transitionMatrix[a,a]

set.seed(23333)
simulation2 <- as.character(simSeq(mytree,l=2000,
         type = "DNA", Q = mat,
         # base composition of ACGT in raw sequences
         bf=c(0.312, 0.205, 0.231, 0.252))
```

```
                            )

ape::write.dna(simulation2, file ="simulation2.fasta", format = "fasta",
               append =FALSE, nbcol = 6, colsep = " ", colw = 10)

bin2 <- read.FASTA(file="simulation2.fasta")
bin2
```

```
## 33 DNA sequences in binary format stored in a list.
##
## All sequences of same length: 2000
##
## Labels:
## SB34
## SB11
## SB30
## SB13
## SB20
## SB37
## ...
##
## Base composition:
##     a     c     g     t
## 0.311 0.204 0.230 0.256
## (Total: 66 kb)
```

Afterwards, the following matrix contains the frequencies of all the nucleotides (a,c,g,t).

```
##                      a         c         g            t
## original    0.3120623 0.2049569 0.2306583 0.2517720334
## artifitial1 0.3120849 0.2050092 0.2302989 0.0002293472
## artifitial2 0.3112916 0.2035263 0.2294936 0.2556885881
```

Although the base compositions of three lizard sequences are different, these values are quite similar mutually. Additionally, since it takes the transition rate matrix into consideration as well, the second artificial sequences should be more similar to the original sequences than the first artificial sequences (each nucleotide frequency in *artifitial2* is closer to the *original* than *artifitial1*). Thus, it is reasonable to say that such two simulations are similar as what we expect.

Assignment 2

2.1

```
## Original 33 lizard sequences:

## [1] "The base individual base composition of lizard sequences is:"

##         a         c         g         t
## 0.3121454 0.2052325 0.2307222 0.2518999

## [1] "The GC content of lizard sequences is:"

## [1] 0.4359547

## [1] "The AT content of lizard sequences is:"

## [1] 0.5640453

## The first artifitial 33 lizard sequences:

## [1] "The base individual base composition of the first simulation sequences is:"

##         a         c         g         t
## 0.3123069 0.2050953 0.2304646 0.2521332

## [1] "The GC content of the first simulation sequences is:"

## [1] 0.4355598

## [1] "The AT content of the first simulation sequences is:"

## [1] 0.5644402

## The second artifitial 33 lizard sequences:

## [1] "The base individual base composition of the second simulation sequences is:"

##         a         c         g         t
## 0.3113182 0.2035303 0.2295152 0.2556364

## [1] "The GC content of the second simulation sequences is:"

## [1] 0.4330455

## [1] "The AT content of the second simulation sequences is:"

## [1] 0.5669545
```

We use the *transeq(https://www.ebi.ac.uk/Tools/st/emboss_transeq/)* to get the protein sequences from the three sequences we get before. Because we are not sure where is the start point of the translation and whether the DNA sequence we got is coding sequences or non-coding sequences, so we get all possible protein sequences results as it shows in the fasta files. There are sequences with the same names and different end numbers like "JF806202_1" to "JF806202_6". The sequences end with 1, 2 and 3 means these protein sequences are translated from the original DNA sequences with the beginning at the first, second and third nucleotide. The sequences end with 4, 5 and 6 means these protein sequences are translated from the reverse-complement of original DNA sequences with the beginning at the first, second and third nucleotide.

```
## [1] "The amino acid composition of lizard sequences (by mean):"

##     A     R     N     D     C     Q     E     G     H     I     L     K
##    31    35    20    19    24    30    28    32    25    29    72    38
##     M     F     P     S     T     W     Y     V     U     O     B     J
##    15    36    32    69    34    13    13    34     0     0     0     0
##     Z     X     *     -     +     . other
##     0     1    30     0     0     0     0
```

```
## [1] "The amino acid composition of the first simulation sequences (by mean):"
```

```
##     A     R     N     D     C     Q     E     G     H     I     L     K
##    31    51    26    20    20    20    21    31    20    40    67    28
##     M     F     P     S     T     W     Y     V     U     O     B     J
##    11    28    31    61    40     9    26    40     0     0     0     0
##     Z     X     *     -     +     . other
##     0     1    37     0     0     0     0
```

```
## [1] "The amino acid composition of the second simulation sequences (by mean):"
```

```
##     A     R     N     D     C     Q     E     G     H     I     L     K
##    31    53    26    21    20    20    21    31    20    41    68    28
##     M     F     P     S     T     W     Y     V     U     O     B     J
##    11    28    31    62    41     8    26    41     0     0     0     0
##     Z     X     *     -     +     . other
##     0     1    38     0     0     0     0
```

```
## [1] "The number range of stop codons in lizard sequences:"
```

```
## [1]  0 87
```

```
## [1] "The number range of stop codons in the first simulation sequences:"
```

```
## [1]  9 71
```

```
## [1] "The number range of stop codons in the second simulation sequences:"
```

```
## [1] 23 55
```

```
## [1] "The number of simulation sequences without stop coden in lizard sequences:"
```

```
## [1] 33
```

"y","m","r","s". They were treated as R(puRines) - A or G ; Y(pYrimidines) - C,T or U ; M(bases with aMino groups) - A or C and S(strong interaction) - C or G. As we shows in the result compositions of each sequence files, we calculate the mean of the frequencies for each amino acids. And we can see that there are *37 and 38 stop codon(marked as "*") on average in the two artificial sequences which are higher than that in the lizard sequence(only 30 stop codon on average). And as we can see the range of numbers of stop codon in each sequences, the lizard sequence have 33 sequences without any stop coden but others artificial sequences are all included the stop coden.

This situation is reasonable because the lizard sequences are natural sequences and they can translate into amino acids with reasonable length. So it will have fewer stop coden in their sequences. As the result shows, there are 33 sequences in lizard sequences have no stop codens, and we can assume that these 33 protein sequences are the original protein sequences that the lizard make in natural.

However, the artificial sequences is created randomly, and they only considered the frequencies of each nucleotides occurs and transition matrices. So they would not consider how they can be translated into amino acids. As a result, there will be more stop codens occurs in the translation result.

2.2

We fit Markov chains to our three data sets.

```
## Original 33 lizard sequences:
```

```
##           a         c          g         t
## a 0.3377604 0.1730948 0.27493261 0.2136731
## c 0.3793901 0.2477071 0.05010812 0.3222728
## g 0.3934372 0.2029168 0.19323832 0.2096122
## t 0.1508047 0.2115396 0.35718190 0.2801093
```

```
## The first artifitial 33 lizard sequences:
```

```
##           a         c         g         t
## a 0.3122030 0.2113359 0.2266693 0.2492529
## c 0.3157581 0.1973302 0.2311880 0.2549780
## g 0.3138153 0.2002921 0.2326894 0.2524729
## t 0.3073562 0.2078781 0.2317917 0.2524278
```

```
## The second artifitial 33 lizard sequences:
```

```
##           a         c         g         t
## a 0.3074563 0.2013831 0.2339648 0.2571957
## c 0.3150369 0.2054815 0.2284948 0.2509868
## g 0.3156782 0.1992471 0.2288997 0.2561749
## t 0.3090424 0.2084198 0.2253780 0.2571598
```

For lizard sequences and the artificial sequences, markov chains were fitted. From the transition matrices, we observed that it is of order 1 using markovchainFit function of markovchain package. We obtaibed the same results using other function called fitMarkovChain function of clickstream package. Since the orders of real and simulated sequences are 1's, the probability of a current state depends only on the probability of the nearest previous state. The higher order markov model gives greater accuracy as long as we have enough data.

The change of bases should not happen according to the principle of complementary base pairing, which means that the Markov chain has order 0. However, the genetic inheritance follows an 1-order Markov chain based on our samples, which would change the lizard gene sequences for generations. It might give both superiority and weakness to their children. But in general, such situation could results in the differences between species and the variety of creatures.

2.3

Alignments

The following alignment sequences is based on the second artificial sequences.

```
## use default substitution matrix
```

```
## CLUSTAL 2.1
##
## Call:
##    msaClustalW("lizard_seqs.fasta", type = "dna")
##
## MsaDNAMultipleAlignment with 33 rows and 2923 columns
##      aln
##  [1] CAAAGTAAAGTCACTTGAGAAGCCACTTCCTGA...AAGC----------------------------
##  [2] CAAAGTGAGATCACTTGAGAAGCCACTTCCTGA...AAGCACCATTGGCTATATACCTCAAAACATTTG
##  [3] ----------------------------------...----------------------------------
##  [4] ----------------------------------...----------------------------------
```

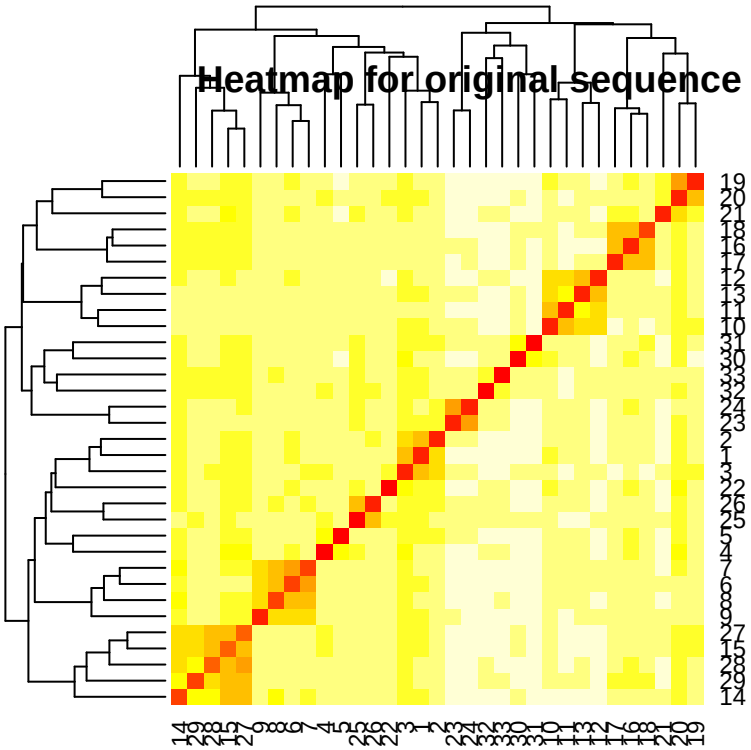
```
##  [5] CAAAGTAAGGTCACTTGAGAAGCCACTTCCTGA...AAGC----------------------------
##  [6] ----------------------------------...----------------------------------
##  [7] ----------------------------------...----------------------------------
##  [8] CAAAGTAAGATCACTTGAGAAGCCACTTTCTGA...AAGC----------------------------
##  [9] ----------------------------------...----------------------------------
##  ... ...
## [26] CAAAGTGAGATCACTTGAGAAGCCACTTTCTGA...AAGCAA--------------------------
## [27] ----------------------------------...----------------------------------
## [28] -AAAGTGAGATCATTTGAGAAGTCACTTCCTGA...----------------------------------
## [29] -AAAGTGAGATCATTTGAGAAGTCACTTCCTGA...----------------------------------
## [30] ----------------AAAGTGAGATCACTTGA...----------------------------------
## [31] ----------CACTTGAGAAGCCACTTCCTGA...----------------------------------
## [32] ----------------------------------...----------------------------------
## [33] ----------------------------------...----------------------------------
##  Con ----------------------------------...----------------------------------

## use default substitution matrix

## CLUSTAL 2.1
##
## Call:
##    msaClustalW("simulation.fasta", type = "dna")
##
## MsaDNAMultipleAlignment with 33 rows and 3034 columns
##      aln
##  [1] ----------------------------------...----------------------------------
##  [2] ----------------------------------...GGACCTACGTGGATGTCTTACTGTCTAATCTCT
##  [3] ----------------------------------...----------------------------------
##  [4] ----------------------------------...CGGCGAGAGAGTAA------------------
##  [5] ----------------------------------...----------------------------------
##  [6] ----------------------------------...----------------------------------
##  [7] ----------------------------------...----------------------------------
##  [8] ----------------------------------...----------------------------------
##  [9] ----------------------------------...----------------------------------
##  ... ...
## [26] ----------------------------------...----------------------------------
## [27] ----------------------------------...----------------------------------
## [28] ----------------------------------...----------------------------------
## [29] ----------------------------------...CAATCTTGAAAT----------------------
## [30] ----------------------------------...----------------------------------
## [31] ----------------------------------...----------------------------------
## [32] ----------------------------------...----------------------------------
## [33] ----------------------------------...----------------------------------
##  Con ----------------------------------...----------------------------------

## use default substitution matrix

## CLUSTAL 2.1
##
## Call:
##    msaClustalW("simulation2.fasta", type = "dna")
##
## MsaDNAMultipleAlignment with 33 rows and 2097 columns
##      aln
##  [1] -----------CGCACGCGGTTAAAAACCAAAC...CTAACATTCCGGGATTACTAGGCAT--------
##  [2] -----------CACACCCGGGTAAAAGGCACAC...CTAGCATCCCCGGATTACGAAGCAT--------
```

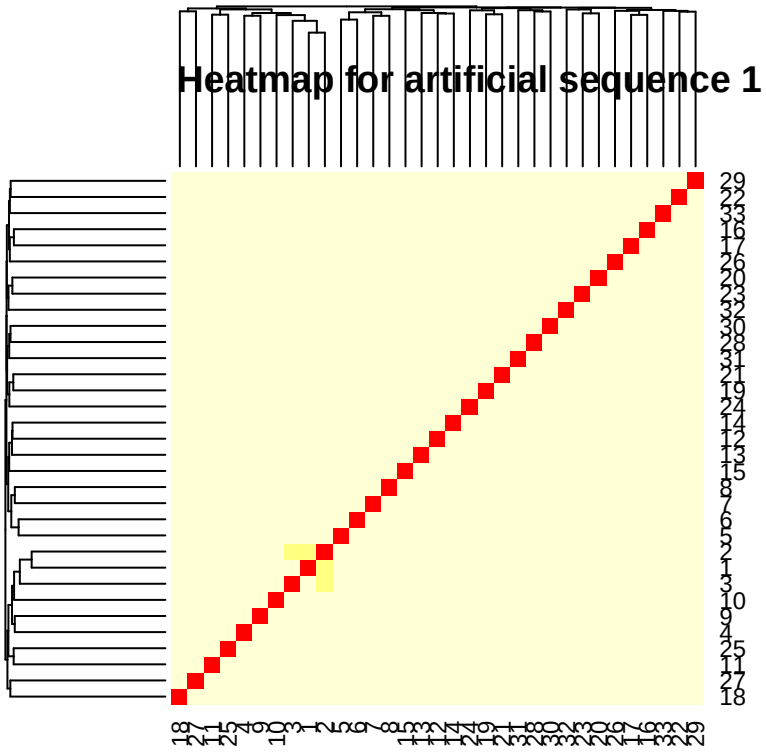
```
##  [3] --------ATAGGAATTGGTCAGGTTAGCTACC...TCTACATTGTTGACTCGGCTAACACGCGACATA
##  [4] --------AAGCTAATTATTCCGAAAGGCTGCA...TCGCCATTGGCGGCTAGTTTAGCCTGGCAAGCA
##  [5] ------GCTGAAAAAGATCTATAGACGATCGTA...TACTTGCGCGCGGTATATGAAGCATCAGA----
##  [6] ------GCTAAACACATTCTATACCCTGTCGGA...TTTATGCGCGCTATAAAGAAAGCTGGTGG----
##  [7] ------GCAGAACAATGTCTTTCGGCTCTCTTA...TAAAGTTGTACGCCATTTTCAATAACGGA----
##  [8] -----------------GCTTTATTCGAACG-G...ACCC-ATGTTCGTAGTAGGAGGTTC--------
##  [9] -----------------CAGTTATCCAAACA-G...TGGA-GTGTTCGATATAGAAGACAC--------
##  ... ...
## [26] ----------------------AACCGATATA...TAGT------------------------------
## [27] ATTAATCTTGGGGGACGCCTTGGCGAAATGTTT...---------------------------------
## [28] ---------------------------------...TTACGGAATTATTCTCAAACCGAGGA-------
## [29] ----------------------ACGCCACATTA...TGCCCACCCGAG---------------------
## [30] ----------------GCCCCCGAGATCCTTCT...---------------------------------
## [31] ----------------TAAACCCAGTTTCTTAT...---------------------------------
## [32] --------------------------------AT...TCAAC------------------------------
## [33] ----------ATCCACTTCATACCTATCATTTG...---------------------------------
##  Con -----------??????????????????????...T??????????????????????????--------
```

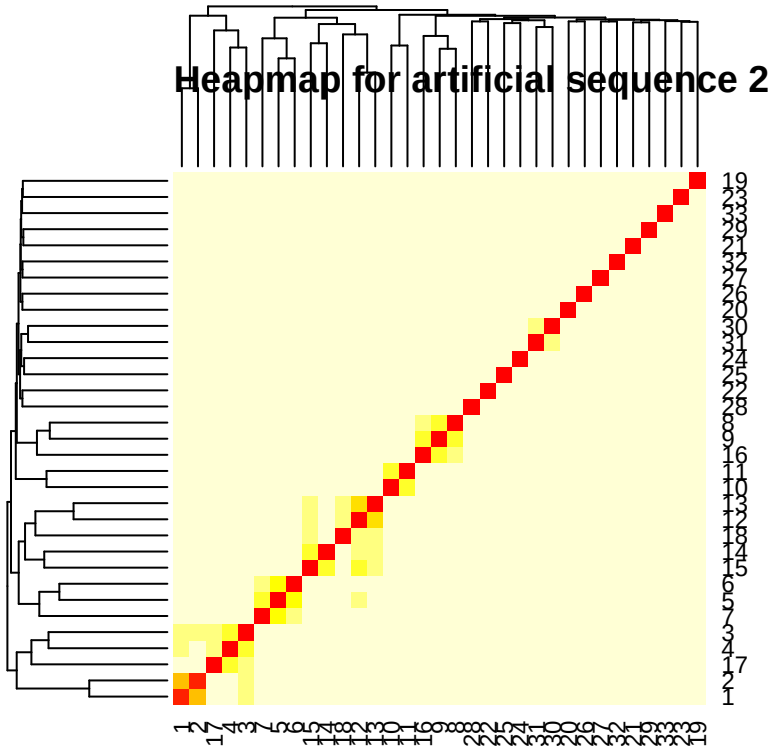
Compared with the alignment from the first artificial sequences, the first and third alignments seem better.

Heatmaps

According to these three alignments, we create the corresponding heat maps to find possible clusters in such three sequences.




Heatmap for artificial sequence 1



We can find that some sequences have strong connections in the original lizard sequences. It seems most of the *clusters appear in anti diagonal position* and *it represents species with similar traits*. For example, 14, 29, 28, 15 and 27 numbered tips seem to belong to one cluster. However, it is hard to find any clusters in the first and second artificial sequences. **Perhaps these situations arises because the sequences have no biological significance as they are randomly generated from the original sequence.**

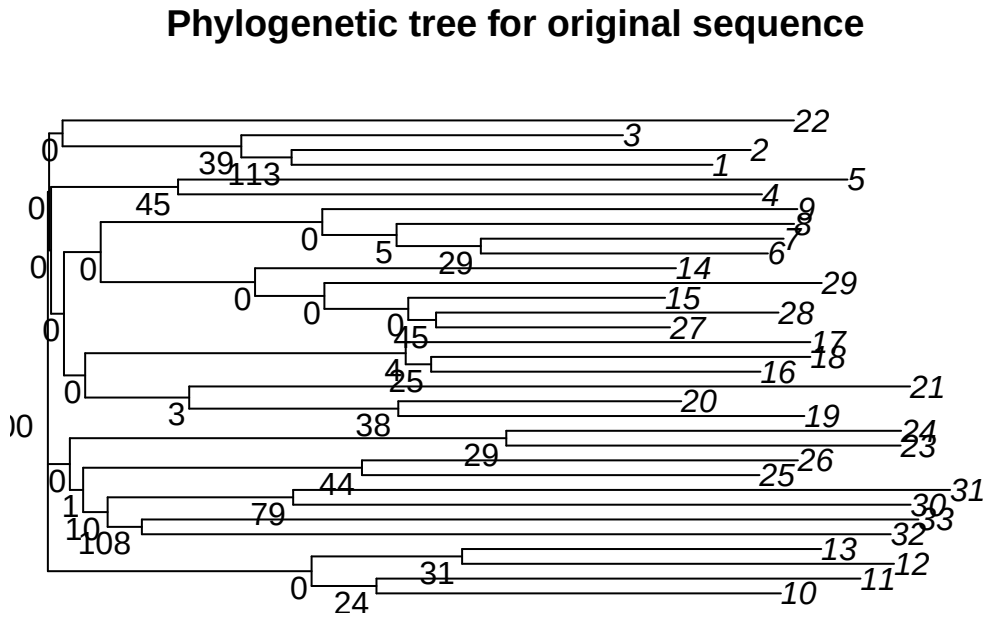
Assignment 3

3.1

We use bootstrap with 1000 replicates to validate our three sequences, and then show the result values on the corresponding NJ trees. Unfortunately, most of bootstrap values on branches are 0, and only the branches on the bottom (right) exist some positive values, which is still small compared with 1000 replicates.

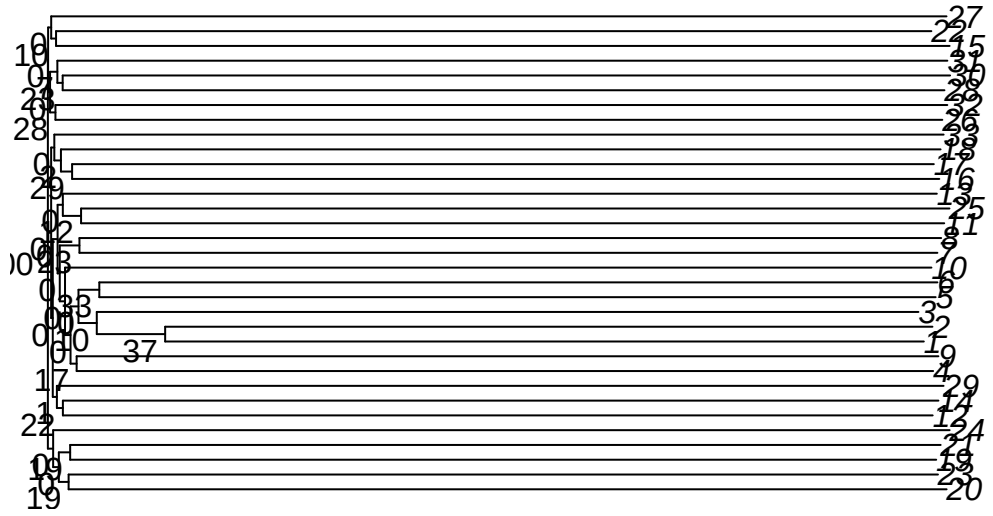
```
## [1] "The bootstrap results of original sequence is:"
```

```
##  [1] 1000    0    0    0    0    0    1    0    0   10  108    3   45    0
## [15]   39  113   79    0    0    0    0   45   44    5   24   38    4   25
## [29]   29   31   29
```



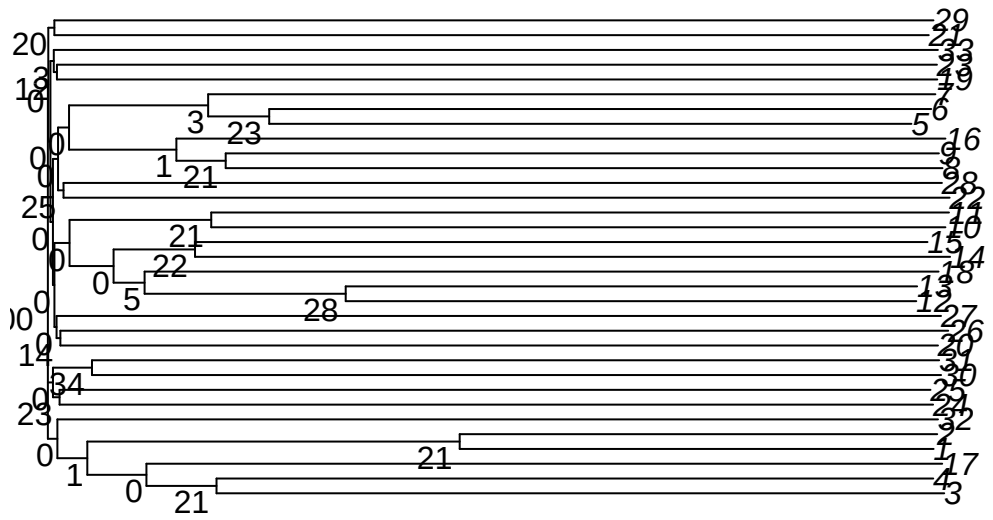

```
## [1] "The bootstrap results of artificial sequence 1 is:"
```

```
##  [1] 1000    0    0    0    0    0    0    0    1   10    0   28    0    7
## [15]    0   22    2    0    0   23    0   19   19   17    0   29   23   12
## [29]   10   33   37
```




```
## [1] "The bootstrap results of artificial sequence 2 is:"
```

```
##  [1] 1000    0    0    0    0    3    0   12    0    0   14   20    0   25
## [15]   23    0    0    1   34    0    5    0    1   22    3   21   21   21
## [29]   23   28   21
```

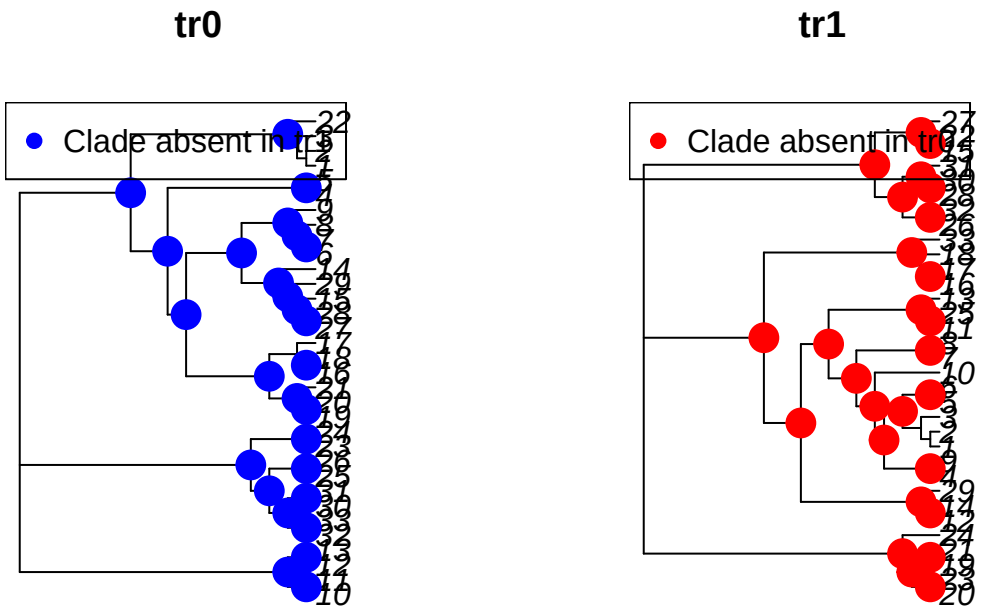


We used boot.phylo to boostrap the trees and get the number of bootstrap trees(out of 1000).

For instance, we can see the second artificial sequences, the end branch connected with 3 and 4 is not generated in original NJ tree. So the relation of sequences in each tree is totally different. We cannot dare to say that there is similarity between those two artificial trees and the original sequence tree.

3.2

```
#distances between the trees
Dist1 <- dist.topo(tr0,tr1)
Dist2 <- dist.topo(tr1,tr2)
Dist3 <- dist.topo(tr2,tr0)

# using ape, distory, phangorn, phyloTop, TotalCopheneticIndex or treespace R packages
#Using ape, compare between original and artificial 1 trees
ape::comparePhylo(tr0,tr1, plot = TRUE, force.rooted = TRUE)
```




```
## => Comparing tr0 with tr1.
## Both trees have the same number of tips: 33.
## Both trees have the same tip labels.
## Both trees have the same number of nodes: 31.
## Both trees are unrooted.
## Both trees are not ultrametric.
## 27 clades in tr0 not in tr1.
## 27 clades in tr1 not in tr0.
```

```
# using phyloTop package to count ladder size of trees
phyloTop::ladderSizes(tr0)
```

```
## $ladderSizes
## [1] 3 2 2
##
## $ladderNodes
## [1] 55 56 57 59 60 63 64
##
## $ladderEdges
## [1] 40 41 49 59
```

```
phyloTop::ladderSizes(tr1)
```

```
## $ladderSizes
## [1] 2
##
## $ladderNodes
## [1] 56 57
```

```
##
## $ladderEdges
## [1] 44
```

```
# using phangorn package
phangorn::treedist(tr0,tr1, check.labels = TRUE)
```

```
##      symmetric.difference   branch.score.difference
##                 54.000000                  1.912088
##           path.difference quadratic.path.difference
##                 82.036577                 13.500329
```

```
# using distory we have done dist.dna

# using TotalCopheneticIndex
TotalCopheneticIndex::tci(tr0)
```

```
## [1] 612
```

The Total Cophenetic Index is a measure of tree balance - i.e. whether a (phylogenetic) tree comprises symmetric pairs of nodes, or has a pectinate ' caterpillar' shape. The index has a greater resolution power than Sackin's and Colless' indices, and can be applied to trees that are not perfectly resolved. Full details are provided by Mir et al. (2013).

Finally, we save and read the trees using the code below.

```
# 3.2
# save tree to our computer
Tre1<-write.tree(mytree,file="tre1")
Treee1<- read.tree("Tre1")

Tre2<-write.tree(tr0,file="tre2")
Treee2<- read.tree("Tre2")

Tre3<-write.tree(tr1,file="tre3")
Treee3<- read.tree("Tre3")

Tre4<-write.tree(tr2,file="tre4")
Treee4<- read.tree("Tre4")
```

Appendix

```
Sys.setlocale(locale="english")
knitr::opts_chunk$set(echo = TRUE)
library(ape)
library(seqinr)
library(stringr)
library(markovchain)
library(phangorn)
library(msa)
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)
ape::write.dna(lizards_sequences, file ="lizard_seqs.fasta", format = "fasta",
               append =FALSE, nbcol = 6, colsep = " ", colw = 10)

bin0 <- read.FASTA(file="lizard_seqs.fasta")
bin0
## simulation function
simulate2 <- function(raw){
  a <- as.data.frame(str_split(raw,""))
  nt <- unique(a)           # all exist nucleotides
  nt[,1] <- as.character(nt[,1])
  nm <- vector()            # corresponding numbers
  for (i in 1:nrow(nt)) {nm[i] <- str_count(raw,nt[i,1])}
  res <- sample(nt[,1],nrow(a),rep = TRUE, prob=nm/nrow(a))
  res
}
ss0 <- read.fasta(file="lizard_seqs.fasta",as.string=TRUE)
simulation <- list()
for (i in 1:33) {
  raw <- gsub(" ","", ss0[[i]])
  simulation[[i]] <- simulate2(raw)
}
n1 <- rep("ZJ",1,33)          # names ZJ11 ~ ZJ43
n2 <- 11:43; names <- vector()
for (i in 1:33) {names[i] <- paste0(n1[i],n2[i])}
names(simulation) <- names
ape::write.dna(simulation, file ="simulation.fasta", format = "fasta",
               append =FALSE, nbcol = 6, colsep = " ", colw = 10)
bin1 <- read.FASTA(file="simulation.fasta")
bin1
n1 <- rep("SB",1,33)          # names SB11 ~ SB43
n2 <- 11:43; names1 <- vector()
for (i in 1:33) {names1[i] <- paste0(n1[i],n2[i])}
```

```r
set.seed(23333)          # create a phylogenetic tree with random order
mytree <- rtree(33, tip.label = names1)
plot(mytree)
title("A phylogenetic tree with random order")
## transition rate matrix from raw sequences
bin0 <- read.FASTA(file="lizard_seqs.fasta")
mcFit <- markovchainFit(data=as.character(bin0))
mat <- mcFit$estimate
a <-c(1,2,3,7)                 # avoid such nt as y, m, r, s
mat <- mat@transitionMatrix[a,a]

set.seed(23333)
simulation2 <- as.character(simSeq(mytree,l=2000,
         type = "DNA", Q = mat,
         # base composition of ACGT in raw sequences
         bf=c(0.312, 0.205, 0.231, 0.252))
                             )

ape::write.dna(simulation2, file ="simulation2.fasta", format = "fasta",
               append =FALSE, nbcol = 6, colsep = " ", colw = 10)

bin2 <- read.FASTA(file="simulation2.fasta")
bin2
a<- steadyStates(mcFit$estimate)[1,c(1,2,3,7)]
mc1 <- markovchainFit(data=as.character(bin1))
b<-steadyStates(mc1$estimate)[1,c(1,2,3,7)]
mc2 <- markovchainFit(data=as.character(bin2))
c<-steadyStates(mc2$estimate)
m <- rbind(original=a,artitcial1=b,artitcial2=c)
rownames(m) <- c("original","artifitial1","artifitial2")
m
cat("Original 33 lizard sequences:\n")
GC<-GC.content(bin0)
AT<-1-GC
BF<-base.freq(bin0)
paste("The base individual base composition of lizard sequences is:")
BF
paste("The GC content of lizard sequences is:")
GC
paste("The AT content of lizard sequences is:")
AT

cat("\n")
cat("The first artifitial 33 lizard sequences:\n")
bin1 <- read.FASTA(file="simulation.fasta")
GC<-GC.content(bin1)
AT<-1-GC
BF<-base.freq(bin1)
paste("The base individual base composition of the first simulation sequences is:")
BF
paste("The GC content of the first simulation sequences is:")
GC
paste("The AT content of the first simulation sequences is:")
```

```
AT

cat("\n")
cat("The second artifitial 33 lizard sequences:\n")
GC<-GC.content(bin2)
AT<-1-GC
BF<-base.freq(bin2)
paste("The base individual base composition of the second simulation sequences is:")
BF
paste("The GC content of the second simulation sequences is:")
GC
paste("The AT content of the second simulation sequences is:")
AT
z1 <- readAAStringSet("lizard_trans.fasta")
k1<-alphabetFrequency(z1)
summary1<-round(colMeans (k1, na.rm = FALSE, dims = 1))
paste("The amino acid composition of lizard sequences (by mean):")
summary1
cat("\n")
z2 <- readAAStringSet("simulation_trans.fasta")
k2<-alphabetFrequency(z2)
summary2<-round(colMeans (k2, na.rm = FALSE, dims = 1))
paste("The amino acid composition of the first simulation sequences (by mean):")
summary2
cat("\n")
z3 <- readAAStringSet("simulation2_trans.fasta")
k3<-alphabetFrequency(z3)
summary3<-round(colMeans (k3, na.rm = FALSE, dims = 1))
paste("The amino acid composition of the second simulation sequences (by mean):")
summary3
cat("\n")
paste("The number range of stop codons in lizard sequences:")
range(k1[,"*"])
paste("The number range of stop codons in the first simulation sequences:")
range(k2[,"*"])
paste("The number range of stop codons in the second simulation sequences:")
range(k3[,"*"])
paste("The number of simulation sequences without stop coden in lizard sequences:")
length(which(k1[,"*"]==0))

cat("Original 33 lizard sequences:\n")
mc0 <- markovchainFit(data=as.character(lizards_sequences))      # list(markovchain)
m0 <- mc0$estimate[c(1,2,3,7),c(1,2,3,7)]
m0
cat("\n")
cat("The first artifitial 33 lizard sequences:\n")
mc1 <- markovchainFit(data=as.character(bin1))
m1<-mc1$estimate[c(1,2,3,6),c(1,2,3,6)]
m1
cat("\n")
cat("The second artifitial 33 lizard sequences:\n")
mc2 <- markovchainFit(data=as.character(bin2))
m2<-mc2$estimate[c(1,2,3,4),c(1,2,3,4)]
```

```
m2
msa0 <- msaClustalW("lizard_seqs.fasta",type="dna")
print(msa0)
msa1 <- msaClustalW("simulation.fasta",type="dna")
print(msa1)
msa2 <- msaClustalW("simulation2.fasta",type="dna")
print(msa2)
align0 <- msaConvert(msa0, type="seqinr::alignment")
d0 <- seqinr::dist.alignment(align0, "identity")
Tree0 <- nj(d0)
Tree0$tip.label <- names(bin0)
align1 <- msaConvert(msa1, type="seqinr::alignment")
d1 <- seqinr::dist.alignment(align1, "identity")
Tree1 <- nj(d1)
Tree1$tip.label <- names(bin1)
align2 <- msaConvert(msa2, type="seqinr::alignment")
d2 <- seqinr::dist.alignment(align2, "identity")
Tree2 <- nj(d2)
Tree2$tip.label <- names(bin2)
heat0<-as.matrix(d0)
heatmap(heat0)
title("Heatmap for original sequence")
cat("\n")
heat1<-as.matrix(d1)
heatmap(heat1)
title("Heatmap for artificial sequence 1")
cat("\n")
heat2<-as.matrix(d2)
heatmap(heat2)
title("Heapmap for artificial sequence 2")
#boot strap and heat maps
y0 <- as.DNAbin(align0)
F = function(x) njs(x)
tr0<-F(d0)
v0 <- boot.phylo(phy =  tr0, x = y0,
                 FUN = F, quiet = T,
                 1000)
y1 <- as.DNAbin(align1)
tr1<-F(d1)
v1 <- boot.phylo(phy =  tr1, x = y1,
                 FUN = F, quiet = T,
                 1000)
y2 <- as.DNAbin(align2)
tr2<-F(d2)
v2 <- boot.phylo(phy =  tr2, x = y2,
                 FUN = F, quiet = T,
                 1000)
# construct the trees
paste("The bootstrap results of original sequence is:")
v0
p1<- plot(tr0, use.edge.length = T)
nodelabels(v0, adj = c(1.2, 1.2), frame = "none")
title("Phylogenetic tree for original sequence")
```

```r
paste("The bootstrap results of artificial sequence 1 is:")
v1
p2<-plot(tr1, use.edge.length = T)
nodelabels(v1, adj = c(1.2, 1.2), frame = "none")
title("Phylogenetic tree for artificial sequence 1")

paste("The bootstrap results of artificial sequence 2 is:")
v2
p3<-plot(tr2, use.edge.length = T)
nodelabels(v2, adj = c(1.2, 1.2), frame = "none")
title("Phylogenetic tree for artificial sequence 2")
#distances between the trees
Dist1 <- dist.topo(tr0,tr1)
Dist2 <- dist.topo(tr1,tr2)
Dist3 <- dist.topo(tr2,tr0)

# using ape, distory, phangorn, phyloTop, TotalCopheneticIndex or treespace R packages
#Using ape, compare between original and artificial 1 trees
ape::comparePhylo(tr0,tr1, plot = TRUE, force.rooted = TRUE)

# using phyloTop package to count ladder size of trees
phyloTop::ladderSizes(tr0)
phyloTop::ladderSizes(tr1)

# using phangorn package
phangorn::treedist(tr0,tr1, check.labels = TRUE)

# using distory we have done dist.dna

# using TotalCopheneticIndex
TotalCopheneticIndex::tci(tr0)


# 3.2
# save tree to our computer
Tre1<-write.tree(mytree,file="tre1")
Treee1<- read.tree("Tre1")

Tre2<-write.tree(tr0,file="tre2")
Treee2<- read.tree("Tre2")

Tre3<-write.tree(tr1,file="tre3")
Treee3<- read.tree("Tre3")

Tre4<-write.tree(tr2,file="tre4")
Treee4<- read.tree("Tre4")
```