Bioinformatics Lab2

Milda Poceviciute, Fanny Karelius, Rab Nawaz Jan Sher and Saman Zahid

24 November 2018

Question 1: DNA sequence acquisition and simulation

In this question an analysis of 3 nucleotide sequences is performed. One sequence is downloaded from GenBank database, and saved into fatsa file. Here we load thev data set:

```
lizards_sequences<-ape::read.FASTA("lizard_seqs.fasta")
```

Q1.1

In this part we simulate DNA sequences with the same base composition and sequences length as the one from the GenBank:

```
len_list <- lengths(lizards_sequences)
new_seq_list <- list()
new_seq <- c()
for (i in 1:length(lizards_sequences)){
    # extract the base composition of each sequence
    freqx <- ape::base.freq(lizards_sequences[i])
    new_seq <- sample(c("a","c","g","t"),len_list[i],rep=TRUE,prob=freqx)
    new_seq_list[i] <- list(new_seq)
}

# convert result to the DNAbin class
simulated_seq1 <- as.DNAbin(new_seq_list)
names(simulated_seq1) <- names(lizards_sequences)
#ape::write.dna(simulated_seq1, file ="sim1_seq.fasta", format = "fasta", colsep = "")
```

```
## Base composition of the data from GenBank:
```

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

```
## Base composition of the simulated sequences:
```

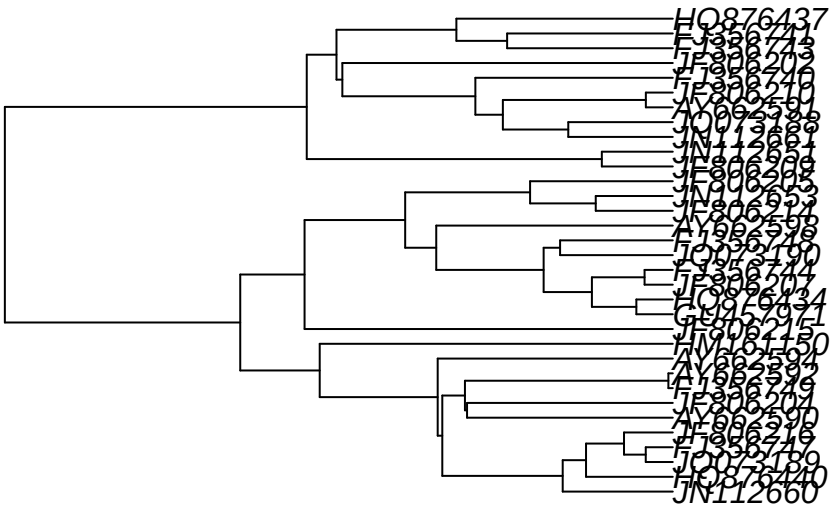
```
##         a         c         g         t
## 0.3103538 0.2059143 0.2334072 0.2503247
```

We can see that the original data, and the simulated data have quite similar base compositions.

Q1.2

In this part we simulate another set of DNA data with similar characteristics as the original one. Now we use a simulated tree with 33 nodes from the original data and package phangorn to do this:

```
# Simulate a tree with 33 nodes
tree<- TreeSim::sim.bd.taxa(n=33, numbsim=1, lambda=1, mu=0)[[1]]
tree$tip.label <- names(lizards_sequences)
plot(tree, yaxt = "n")
```



We need to create a transition matrix, so we use the full dataset to compute that:

```
data <- c()
simulated_seq_tree <- list()

DNA_list <- unlist(as.character(lizards_sequences),use.names = FALSE)
DNA_unique <- c("a","c","g","t")
matrix <-  matrix(0, ncol = length(DNA_unique),
                  nrow = length(DNA_unique))

for (i in 1:(length(DNA_list) - 1)) {
    index_of_i <- DNA_unique == DNA_list[i]
    index_of_i_plus_1 <- DNA_unique == DNA_list[i + 1]
    matrix[index_of_i, index_of_i_plus_1] = matrix[index_of_i, index_of_i_plus_1] + 1
}

# Transition matrix is finally computed
Q <- matrix / rowSums(matrix)
#Q_lower <- Q[lower.tri(Q)]
# Other parameters that are needed for phangorn package function
bf <-  ape::base.freq(lizards_sequences)
size <- round(mean(len_list))
```

```
# Generating the data based on the tree

simulated_seq_tree <-  phangorn::simSeq(
        tree,
        l = size,
        type = "DNA",
        bf = bf,
        Q = Q)

# change numbers into letters
simulated_seq_tree2 <- lapply(simulated_seq_tree,function(seq){
    #seq = a
    states = c("a","c","g","t")
    seq[which(seq==1)] <- states[1]
    seq[which(seq==2)] <- states[2]
```

```
    seq[which(seq==3)] <- states[3]
    seq[which(seq==4)] <- states[4]
    return(seq)
})

simulated_seq2 <- as.DNAbin(simulated_seq_tree2)
# Write the simulated sequence to a fasta file
ape::write.dna(simulated_seq2, file ="sim2_seqs.fasta", format = "fasta", colsep = "")
```

```
## Base composition of the data from GenBank:
```

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

```
## Base composition of the simulated sequences Q1.1:
```

```
##         a         c         g         t
## 0.3103538 0.2059143 0.2334072 0.2503247
```

```
## Base composition of the simulated sequences Q1.2:
```

```
##         a         c         g         t
## 0.3119241 0.2066505 0.2321551 0.2492703
```

The base compositions of all simulated sequences are quite similar to the original one.

Question 2: Sequence analysis

Q2.1

```
## Individual base composition of sequence 1
## The data from the GenBank:
##         a         c         g         t
## 0.2898696 0.2026078 0.2437312 0.2637914
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3156313 0.1863727 0.2494990 0.2484970
## The simulated data in Q1.2:
##         a         c         g         t
## 0.2939990 0.2259203 0.2400403 0.2400403
##
## Individual base composition of sequence 2
## The data from the GenBank:
##         a         c         g         t
## 0.3115541 0.2122554 0.2314507 0.2447398
## The simulated data in Q1.1:
##         a         c         g         t
## 0.2894057 0.2284976 0.2340347 0.2480620
## The simulated data in Q1.2:
##         a         c         g         t
## 0.2970247 0.2188603 0.2380232 0.2460918
##
## Individual base composition of sequence 3
## The data from the GenBank:
##         a         c         g         t
```

```
## 0.3130405 0.2099620 0.2348668 0.2421308
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3064684 0.2127292 0.2369422 0.2438603
## The simulated data in Q1.2:
##         a         c         g         t
## 0.2909733 0.2097832 0.2471004 0.2521432
##
## Individual base composition of sequence 4
## The data from the GenBank:
##         a         c         g         t
## 0.2842942 0.2097416 0.2445328 0.2614314
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3002973 0.2081269 0.2517344 0.2398414
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3096319 0.1956631 0.2395361 0.2551689
##
## Individual base composition of sequence 5
## The data from the GenBank:
##         a         c         g         t
## 0.3059701 0.1987788 0.2360923 0.2591588
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3052917 0.2062415 0.2204885 0.2679783
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3257690 0.2067574 0.2163389 0.2511346
##
## Individual base composition of sequence 6
## The data from the GenBank:
##         a         c         g         t
## 0.2988084 0.2071494 0.2364803 0.2575619
## The simulated data in Q1.1:
##         a         c         g         t
## 0.2914757 0.1998167 0.2483960 0.2603116
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3126576 0.1976803 0.2349975 0.2546646
##
## Individual base composition of sequence 7
## The data from the GenBank:
##         a         c         g         t
## 0.3137323 0.2068488 0.2338291 0.2455898
## The simulated data in Q1.1:
##         a         c         g         t
## 0.2940159 0.2047734 0.2421308 0.2590799
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3050933 0.2138174 0.2239032 0.2571861
##
## Individual base composition of sequence 8
## The data from the GenBank:
```

```
##         a         c         g         t
## 0.2968127 0.2001992 0.2340637 0.2689243
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3157371 0.2041833 0.2191235 0.2609562
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3146747 0.2052446 0.2395361 0.2405446
##
## Individual base composition of sequence 9
## The data from the GenBank:
##         a         c         g         t
## 0.2998084 0.1992337 0.2346743 0.2662835
## The simulated data in Q1.1:
##         a         c         g         t
## 0.2969349 0.2088123 0.2471264 0.2471264
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3217347 0.1880988 0.2244075 0.2657590
##
## Individual base composition of sequence 10
## The data from the GenBank:
##         a         c         g         t
## 0.3157534 0.2071918 0.2304795 0.2465753
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3092466 0.2071918 0.2404110 0.2431507
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3308119 0.2042360 0.2264246 0.2385275
##
## Individual base composition of sequence 11
## The data from the GenBank:
##         a         c         g         t
## 0.3121449 0.2041903 0.2325994 0.2510653
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3110795 0.1942472 0.2485795 0.2460938
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3282905 0.1916288 0.2344932 0.2455875
##
## Individual base composition of sequence 12
## The data from the GenBank:
##         a         c         g         t
## 0.3177408 0.2061677 0.2293832 0.2467082
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3157168 0.2107081 0.2300518 0.2435233
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3202219 0.2072617 0.2314675 0.2410489
##
## Individual base composition of sequence 13
```

```
## The data from the GenBank:
##         a         c         g         t
## 0.3187773 0.2034207 0.2241630 0.2536390
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3122271 0.2041485 0.2361718 0.2474527
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3257690 0.1986889 0.2299546 0.2455875
##
## Individual base composition of sequence 14
## The data from the GenBank:
##         a         c         g         t
## 0.3146384 0.2035273 0.2320988 0.2497354
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3027846 0.2051463 0.2382799 0.2537892
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3177005 0.2128089 0.2223903 0.2471004
##
## Individual base composition of sequence 15
## The data from the GenBank:
##         a         c         g         t
## 0.3196064 0.2033528 0.2255831 0.2514577
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3209607 0.2066958 0.2147016 0.2576419
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3287948 0.1916288 0.2233989 0.2561775
##
## Individual base composition of sequence 16
## The data from the GenBank:
##         a         c         g         t
## 0.2921236 0.2123629 0.2382851 0.2572283
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3227092 0.2201195 0.2091633 0.2480080
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3292990 0.1906203 0.2319718 0.2481089
##
## Individual base composition of sequence 17
## The data from the GenBank:
##         a         c         g         t
## 0.2902903 0.2052052 0.2412412 0.2632633
## The simulated data in Q1.1:
##     a     c     g     t
## 0.319 0.208 0.254 0.219
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3005547 0.2072617 0.2385275 0.2536561
##
```

```
## Individual base composition of sequence 18
## The data from the GenBank:
##         a         c         g         t
## 0.3203540 0.2042478 0.2269027 0.2484956
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3238938 0.1918584 0.2417699 0.2424779
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3267776 0.2052446 0.2269289 0.2410489
##
## Individual base composition of sequence 19
## The data from the GenBank:
##         a         c         g         t
## 0.3170475 0.2003515 0.2274165 0.2551845
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3078273 0.2004212 0.2306072 0.2611443
## The simulated data in Q1.2:
##         a         c         g         t
## 0.2768533 0.2118003 0.2304589 0.2808875
##
## Individual base composition of sequence 20
## The data from the GenBank:
##         a         c         g         t
## 0.2946429 0.2043651 0.2341270 0.2668651
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3015873 0.1875000 0.2380952 0.2728175
## The simulated data in Q1.2:
##         a         c         g         t
## 0.2955119 0.2052446 0.2370146 0.2622289
##
## Individual base composition of sequence 21
## The data from the GenBank:
##         a         c         g         t
## 0.3172554 0.1956522 0.2248641 0.2622283
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3077446 0.1908967 0.2384511 0.2629076
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3151790 0.1976803 0.2445789 0.2425618
##
## Individual base composition of sequence 22
## The data from the GenBank:
##         a         c         g         t
## 0.3219225 0.2074689 0.2285615 0.2420470
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3256135 0.2035949 0.2305565 0.2402351
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3318205 0.2042360 0.2244075 0.2395361
```

```
##
## Individual base composition of sequence 23
## The data from the GenBank:
##         a         c         g         t
## 0.3144044 0.1966759 0.2250693 0.2638504
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3047685 0.1983414 0.2218383 0.2750518
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3126576 0.2183560 0.2370146 0.2319718
##
## Individual base composition of sequence 24
## The data from the GenBank:
##     a     c     g     t
## 0.296 0.214 0.237 0.253
## The simulated data in Q1.1:
##     a     c     g     t
## 0.304 0.212 0.218 0.266
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3121533 0.2133132 0.2329803 0.2415532
##
## Individual base composition of sequence 25
## The data from the GenBank:
##         a         c         g         t
## 0.3175890 0.2054988 0.2284103 0.2485019
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3225238 0.2051463 0.2305252 0.2418047
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3101362 0.2087746 0.2213817 0.2597075
##
## Individual base composition of sequence 26
## The data from the GenBank:
##         a         c         g         t
## 0.3144453 0.2110934 0.2306464 0.2438148
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3028731 0.2174781 0.2362330 0.2434158
## The simulated data in Q1.2:
##         a         c         g         t
## 0.2914776 0.2143217 0.2506304 0.2435703
##
## Individual base composition of sequence 27
## The data from the GenBank:
##         a         c         g         t
## 0.3094527 0.2059701 0.2218905 0.2626866
## The simulated data in Q1.1:
##         a         c         g         t
## 0.2885572 0.2089552 0.2278607 0.2746269
## The simulated data in Q1.2:
##         a         c         g         t
```

```
## 0.3146747 0.2087746 0.2233989 0.2531518
##
## Individual base composition of sequence 28
## The data from the GenBank:
##         a         c         g         t
## 0.3160083 0.2065142 0.2286902 0.2487872
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3095815 0.2144587 0.2244898 0.2514701
## The simulated data in Q1.2:
##         a         c         g         t
## 0.2965204 0.2107917 0.2370146 0.2556732
##
## Individual base composition of sequence 29
## The data from the GenBank:
##         a         c         g         t
## 0.3207612 0.2020761 0.2280277 0.2491349
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3133864 0.2078865 0.2251816 0.2535455
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3177005 0.2062532 0.2334846 0.2425618
##
## Individual base composition of sequence 30
## The data from the GenBank:
##         a         c         g         t
## 0.2943396 0.2037736 0.2377358 0.2641509
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3103774 0.1981132 0.2537736 0.2377358
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3146747 0.2198689 0.2249117 0.2405446
##
## Individual base composition of sequence 31
## The data from the GenBank:
##         a         c         g         t
## 0.3200585 0.2071611 0.2232371 0.2495433
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3323594 0.2173119 0.2158510 0.2344777
## The simulated data in Q1.2:
##         a         c         g         t
## 0.2990419 0.2168432 0.2309632 0.2531518
##
## Individual base composition of sequence 32
## The data from the GenBank:
##         a         c         g         t
## 0.2961117 0.2033898 0.2352941 0.2652044
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3130608 0.2013958 0.2243270 0.2612164
## The simulated data in Q1.2:
```

```
##         a         c         g         t
## 0.3187090 0.1931417 0.2324760 0.2556732
##
## Individual base composition of sequence 33
## The data from the GenBank:
##         a         c         g         t
## 0.2932331 0.2030075 0.2406015 0.2631579
## The simulated data in Q1.1:
##         a         c         g         t
## 0.3071966 0.1815252 0.2502685 0.2610097
## The simulated data in Q1.2:
##         a         c         g         t
## 0.3066062 0.2188603 0.2309632 0.2435703
```

The CG contents are:

```
## In the data from the GenBank:
```

```
## [1] 0.4359547
```

```
## In the simulated data in Q1.1:
```

```
## [1] 0.4393215
```

```
## In the simulated data in Q1.2:
```

```
## [1] 0.4388056
```

A,C,G,T contents are:

```
# A,C,G,T content
cat("In the data from the GenBank:")
```

```
## In the data from the GenBank:
```

```
cat("\n")
```

```
print(ape::base.freq(lizards_sequences))
```

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

```
cat("In the simulated data in Q1.1:")
```

```
## In the simulated data in Q1.1:
```

```
cat("\n")
```

```
print(ape::base.freq(simulated_seq1))
```

```
##         a         c         g         t
## 0.3103538 0.2059143 0.2334072 0.2503247
```

```
cat("In the simulated data in Q1.2:")
```

```
## In the simulated data in Q1.2:
```

```
cat("\n")
```

```
print(ape::base.freq(simulated_seq2))
```

```
##         a         c         g         t
## 0.3119241 0.2066505 0.2321551 0.2492703
```

The six versions of the nucleotide sequences of the lizard (all sequences starting from nucleotide 1, nucleotide 2 and nucleotide 3, as well as the reversed and complimented versions of them) were transformed into amino acid sequences using the EMBOSS Transeq function, and saved tofasta files:

```
# start from the second nucleotide
lizard <- as.character(lizards_sequences)
lizard_seq2 = lapply(lizard, function(a){return(as.character(a[-1]))})
lizard_seq3 = lapply(lizard, function(a){return(as.character(a[c(-1,-2)]))})
ape::write.dna(lizard_seq2, file ="lizard_seq2.fasta", format = "fasta", colsep = "")
ape::write.dna(lizard_seq3, file ="lizard_seq3.fasta", format = "fasta", colsep = "")

# Create 3 versions of lizard sequences

amino_lizard_seq <- read.fasta("lizard_amino_seqs.fasta")
amino_lizard_seq2 <- read.fasta("amino_lizard2.fasta")
amino_lizard_seq3 <- read.fasta("amino_lizard3.fasta")


amino_rev_com_lizard <- read.fasta("rev_com_lizard_amino_acid.fasta")
amino_rev_com_lizard2 <- read.fasta("amino_reversed_lizard2.fasta")
amino_rev_com_lizard3 <- read.fasta("amino_reversed_lizard3.fasta")
```

```
stop_code_count <- function(amino_acid_seq){

  stop_count = c()
  for (i in 1:length(amino_acid_seq)) {

    sequence = amino_acid_seq[[i]]
    stop <- sequence[which(sequence == "*")]
    stop_count[i] = length(stop)
  }
  return(stop_count)
}
df = data.frame("lizard_seq" = sum(stop_code_count(amino_lizard_seq)),
           "lizard_seq2" = sum(stop_code_count(amino_lizard_seq2)),
           "lizard_seq3" = sum(stop_code_count(amino_lizard_seq3)),
           "rev_comp_lizard_seq" = sum(stop_code_count(amino_rev_com_lizard)),
           "rev_comp_lizard2" = sum(stop_code_count(amino_rev_com_lizard2)),
           "rev_comp_lizard3" = sum(stop_code_count(amino_rev_com_lizard3))
          )

knitr::kable(df)
```

lizard_seq	lizard_seq2	lizard_seq3	rev_comp_lizard_seq	rev_comp_lizard2	rev_comp_lizard3
1269	645	1418	897	897	897

The above table shows the total number of stop codans in each of the fasta files of lizard sequences. We can see that the number of stop codans is very high in general (this is due to the fact that the sequences is yet not alligned, hence the same parts of the real RNA strand is repeated multiple times). However, when we started to translate into amino acids from the second nucleotide, we have considerably less stop codans than if we translated from the first or third nucleotide. The all reversed and complimented sequences have exactly same amount of stop codans. This is reasonable, as they all are exactly the same sequences apart from the last 1-3 nucleotides (the length of the sequences are different, and the first nucleotides became the last ones

after this operation).

Therefore, it is more likely that the true sequence starts from nucleotide 2. However, to find out the true underlying sequence, we'd need to do much more work. We'd have to check all possible combinations of the different starts of all the sequences in the data set. Also probably we would need to query the databases and try to find if there is some information about the exact sequence.

Q2.2

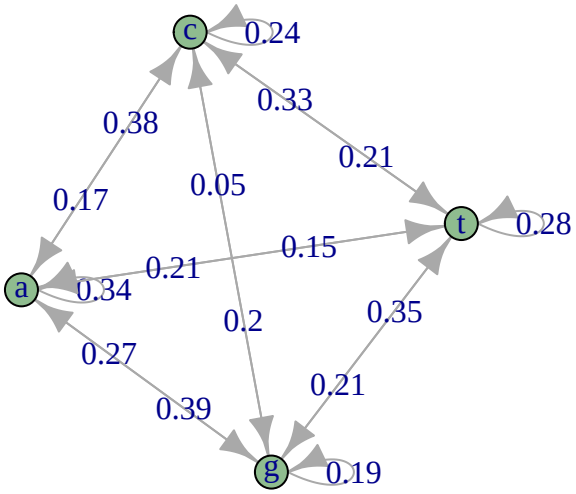
In this part we do markov chain analysis of the data sets. We combine all the sequences from each dataset to seperate lists, and use the *markovchainFit* function with bootstrapping to estimate the Markov models. This function fits the first order markov model to all three datasets.

```
lizard <- as.character(lizards_sequences)
sim1 <- as.character(simulated_seq1)
sim2 <- as.character(simulated_seq2)
flat_lizard <- unlist(lizard,use.names = FALSE)
remove_values = c("y","m","r","s")
flat_lizard <- flat_lizard[!flat_lizard %in% remove_values]
mc_lizard <-  markovchainFit(flat_lizard,method = "bootstrap", nboot = 3,
                             name = "Bootstrap Mc",
                             parallel = TRUE)

flat_sim1 <- unlist(sim1,use.names = FALSE)
flat_sim1 <- flat_sim1[!flat_sim1 %in% remove_values]
mc_sim1 <-  markovchainFit(flat_sim1,method = "bootstrap", nboot = 3,
                           name = "Bootstrap Mc",
                           parallel = TRUE
                           )

flat_sim2 <- unlist(sim2,use.names = FALSE)
flflat_sim2at_sim1 <- flat_sim2[!flat_sim2 %in% remove_values]
mc_sim2 <-  markovchainFit(flat_sim2,method = "bootstrap", nboot = 3,
                           name = "Bootstrap Mc",
                           parallel = TRUE
                           )
```

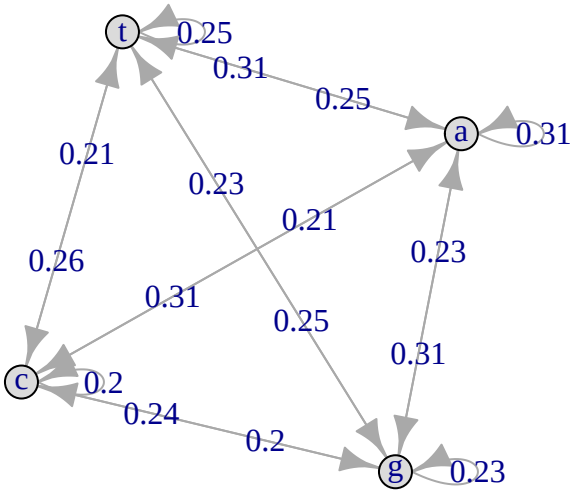
We see that the original data set from GenBank contains a small amount of additional "nucleotides": they are called "y", "m", and "r".

These are the nucleotides which we are not sure about where Y indicates unsurity in A or G while R shows ambiguity in C or T and M appears when there is no surity at all.

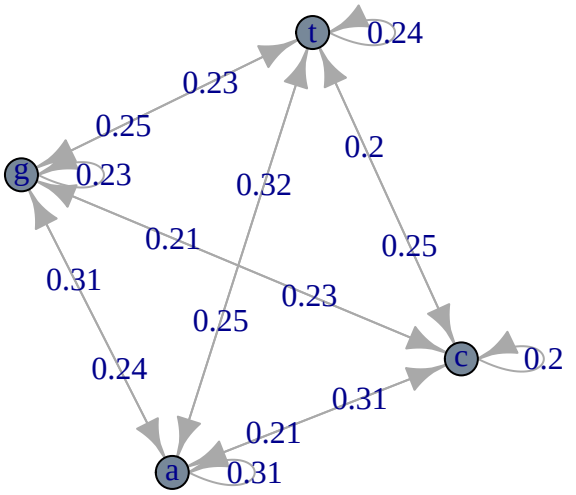
```
plot(mc_lizard$estimate)
```




```
plot(mc_sim1$estimate)
```




```
plot(mc_sim2$estimate)
```



In the first order Markov chains we assume that the nucleotide X_n only depends on a previous nucleotide, namely X_{n-1}. This is not a reasonable assumption, as DNA (or RNA) sequences contain the codons: the triplets of the nulceotides that determine what amino acids will be produced from it. Hence, it may be more

reasonable that the most important dependences are between nucleotides in these triplets. In the lecture it was mentioned that there is some research supporting that a sixth order Markov Chain models perform better. That implies that the nth nucleotide depends on what were the previous 5 nucleotides.

Q2.3

In this part the three DNA collections of the sequences are aligned using the *msa* package. Furthermore the distances between the alignments are calculated. They are used to produce the heatmaps

```
aligment_lizard <- msa("lizard_seqs.fasta", type="dna")
```

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

```
aligment_sim1 <- msa("sim1_seq.fasta", type="dna")
```

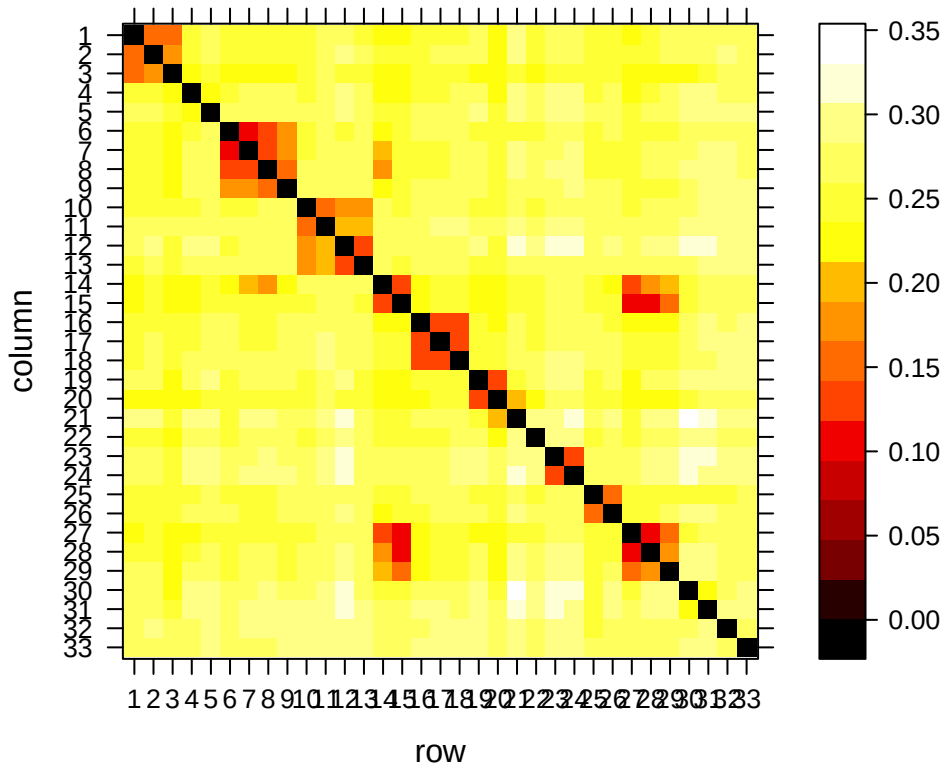
```
## use default substitution matrix
```

```
aligment_sim2 <- msa("sim2_seqs.fasta", type="dna")
```

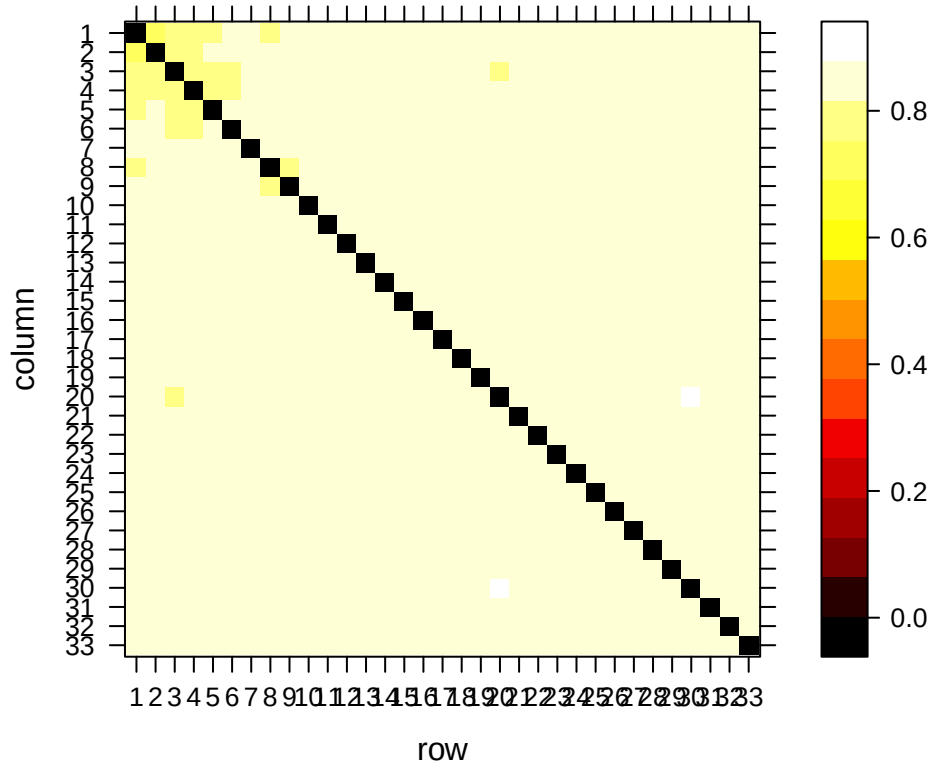
```
## use default substitution matrix
```

```
aligment_lizard2 <- msaConvert(aligment_lizard, type="seqinr::alignment")
dist_lizard <- seqinr::dist.alignment(aligment_lizard2,matrix="identity")

aligment_sim1_2 <- msaConvert(aligment_sim1, type="seqinr::alignment")
dist_sim1 <- seqinr::dist.alignment(aligment_sim1_2,matrix="identity")

aligment_sim2_2 <- msaConvert(aligment_sim2, type="seqinr::alignment")
dist_sim2 <- seqinr::dist.alignment(aligment_sim2_2,matrix="identity")

# the seqinr::dist.alignment computes the square root distance
dm_lizard <- as.matrix(dist_lizard)
dm_sim1 <- as.matrix(dist_sim1)
dm_sim2 <- as.matrix(dist_sim2)
```

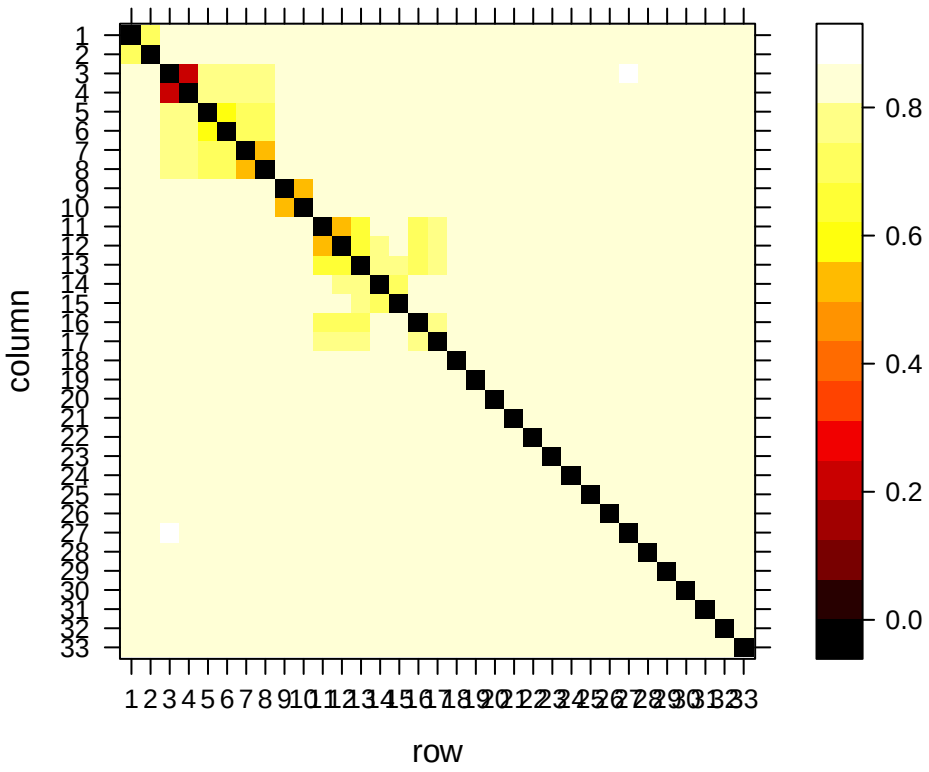
```
# heatmaps
new.palette=colorRampPalette(c("black","red","yellow","white"),space="rgb")
levelplot(dm_lizard[1:ncol(dm_lizard),ncol(dm_lizard):1],col.regions=new.palette(20))
```




```
levelplot(dm_sim1[1:ncol(dm_sim1),ncol(dm_sim1):1],col.regions=new.palette(20))
```




```
levelplot(dm_sim2[1:ncol(dm_sim2),ncol(dm_sim2):1],col.regions=new.palette(20))
```



The plots illustrate how much overlaying the sequences have in each dataset. The smaller the value (the darker the colour), the more nucleotides are matching between the corresponding sequences of the DNA. Therefore, the sequences from the data from GenBank has much more overlaying parts then the two randomly generated sequences. This result is expected as sequencing of a DNA produces "fragments" of the real DNA sequence, and those "fragments" do contain the same bits of the original DNA. The artificially simulated DNA sequences did not actually go through a process of the sequencing, hence they are way less likely to randomly contain long matching parts in their sequences.

Question 3: Phylogeny reconstruction

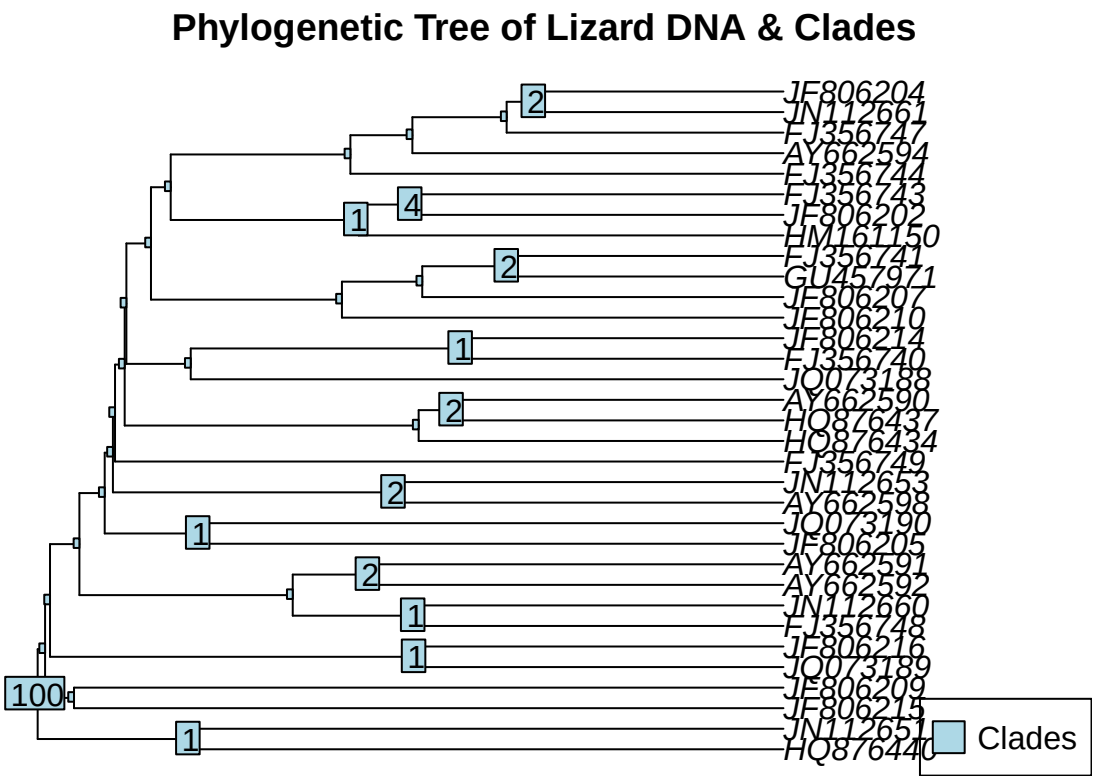
Q3.1

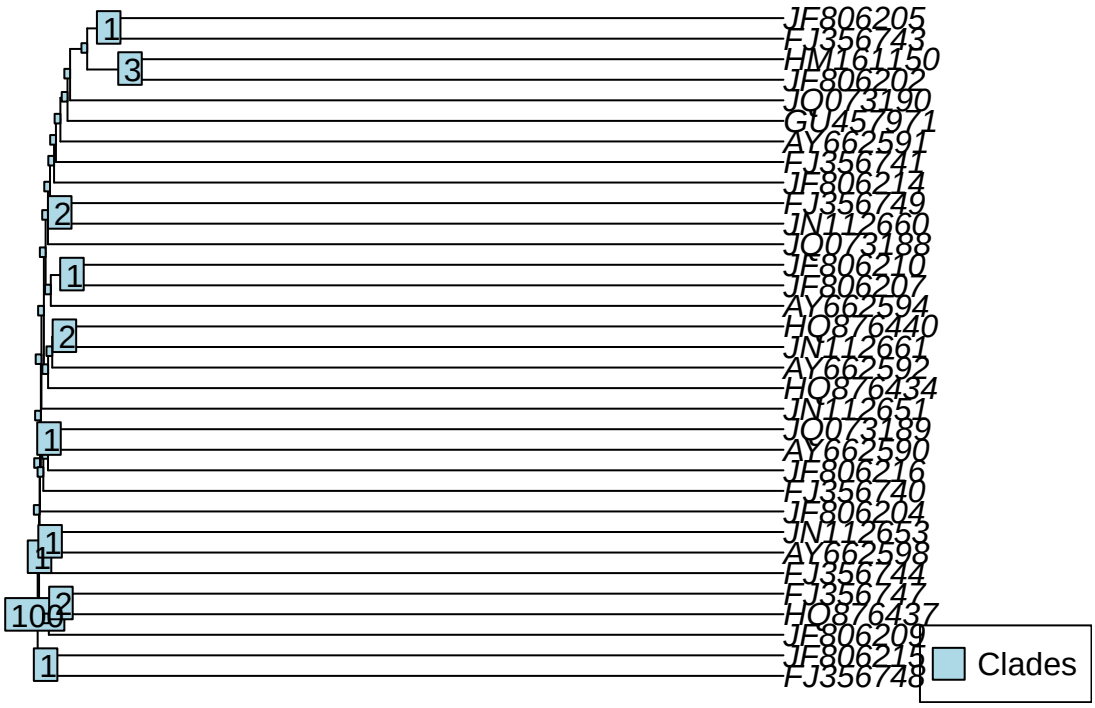
```
library(phangorn)


upgma_tree <- function(x){
  tree <- upgma(x)
  tree$tip.label <- names(lizards_sequences)
  return(tree)
}


lizard_seq_tree <- upgma_tree(dm_lizard)
#plot(lizard_seq_tree, main="Phylogenetic Tree of Lizard DNA")
```

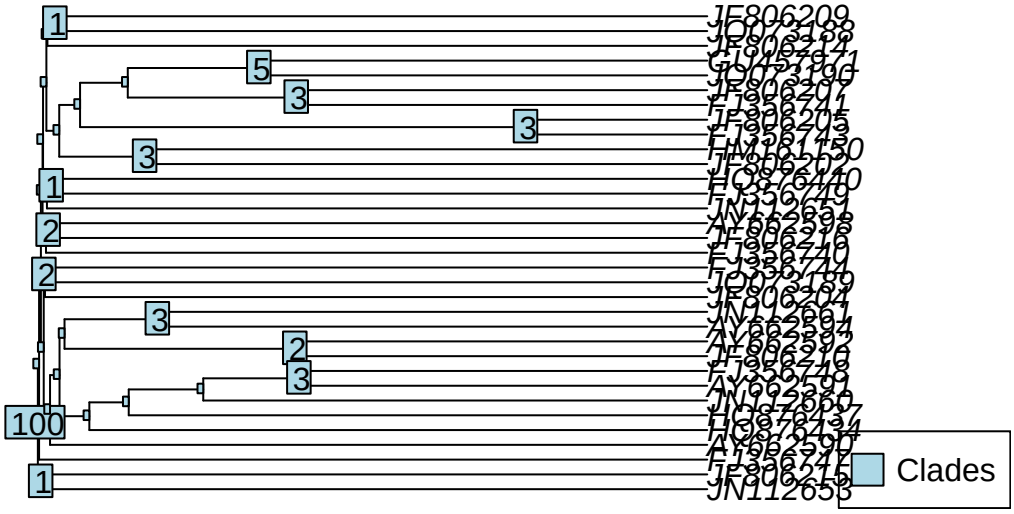
```
sim1_seq_tree <- upgma_tree(dm_sim1)
#plot(sim1_seq_tree, main="Phylogenetic Tree of Lizard DNA sim1")


sim2_seq_tree <- upgma_tree(dm_sim2)
#plot(sim2_seq_tree, main="Phylogenetic Tree of Lizard DNAs sim2")


boot_lizard <-  boot.phylo(lizard_seq_tree, dm_lizard, FUN=upgma_tree, trees  = TRUE,quiet = TRUE)
sim1_boot     <- boot.phylo(sim1_seq_tree, dm_sim1, FUN=upgma_tree, trees  = TRUE,quiet = TRUE)
sim2_boot     <- boot.phylo(sim2_seq_tree, dm_sim2, FUN=upgma_tree, trees  = TRUE,quiet = TRUE)
```

Extracting Clades

```
lizard_clade = prop.clades(lizard_seq_tree,boot_lizard$trees,rooted = TRUE)
sim1_clade = prop.clades(sim1_seq_tree,sim1_boot$trees,rooted = TRUE)
sim2_clade = prop.clades(sim2_seq_tree,sim2_boot$trees,rooted = TRUE)
```







Q3.2

```
library(phangorn)
path_diff_sim1_lizard <- path.dist(sim1_seq_tree,lizard_seq_tree)
path_diff_sim2_lizard <- path.dist(sim2_seq_tree,lizard_seq_tree)
path_diff_sim1_sim2 <- path.dist(sim1_seq_tree,sim2_seq_tree)
```

```
felenstein_diff_sim1_lizard <- KF.dist(sim1_seq_tree,lizard_seq_tree)
felenstein_diff_sim2_lizard <- KF.dist(sim2_seq_tree,lizard_seq_tree)
felenstein_diff_sim1_sim2 <- KF.dist(sim1_seq_tree,sim2_seq_tree)

result = data.frame(path_difference=c(path_diff_sim1_lizard,
                                      path_diff_sim2_lizard,
                                      path_diff_sim1_sim2),
                    felenstein_distance=c(felenstein_diff_sim1_lizard,
                                          felenstein_diff_sim2_lizard,
                                          felenstein_diff_sim1_sim2)
                    )
rownames(result) = c("sim1 vs lizard","sim2 vs lizard","sim1 vs sim2")
knitr::kable(result)
```

	path_difference	felenstein_distance
sim1 vs lizard	118.4314	1.9094493
sim2 vs lizard	98.6509	1.6802872
sim1 vs sim2	103.8268	0.6809785

Geodesic Distance (Distance between edeges)

```
library(distory)
#distance between edeges using geodesic
geodesic_dist = dist.multiPhylo(list(lizard_seq_tree,sim1_seq_tree,sim2_seq_tree), method = "geodesic")
geodesic_dist = as.matrix(geodesic_dist)
geodesic_dist
```

```
##          1         2         3
## 1 0.000000 1.9122684 1.7132242
## 2 1.912268 0.0000000 0.6896505
## 3 1.713224 0.6896505 0.0000000
```

It can be seen from the path distance, Kuhner Felensteins distance and edge distance that tree from simulation 1 differs more from expected lizard tree than simulation 2. It can also be observed that both simulations are closer to each other that is in terms of distance.

```
comp1 <- comparePhylo(lizard_seq_tree,sim1_seq_tree)
comp2 <- comparePhylo(lizard_seq_tree,sim2_seq_tree)
comp12 <- comparePhylo(sim1_seq_tree,sim2_seq_tree)
```

```
comp1$messages
```

```
## [1] "=> Comparing lizard_seq_tree with sim1_seq_tree."
## [2] "Both trees have the same number of tips: 33."
## [3] "Both trees have the same tip labels."
## [4] "Both trees have the same number of nodes: 32."
## [5] "Both trees are rooted."
## [6] "Both trees are ultrametric."
## [7] "30 clades in lizard_seq_tree not in sim1_seq_tree."
## [8] "30 clades in sim1_seq_tree not in lizard_seq_tree."
## [9] "Branching times of clades in common between both trees: see ..$BT\n(node number in parentheses)
```

```
comp2$messages
```

```
## [1] "=> Comparing lizard_seq_tree with sim2_seq_tree."
## [2] "Both trees have the same number of tips: 33."
## [3] "Both trees have the same tip labels."
## [4] "Both trees have the same number of nodes: 32."
## [5] "Both trees are rooted."
## [6] "Both trees are ultrametric."
## [7] "31 clades in lizard_seq_tree not in sim2_seq_tree."
## [8] "31 clades in sim2_seq_tree not in lizard_seq_tree."
## [9] "Branching times of clades in common between both trees: see ..$BT\n(node number in parentheses)
```

```
comp12$messages
```

```
## [1] "=> Comparing sim1_seq_tree with sim2_seq_tree."
## [2] "Both trees have the same number of tips: 33."
## [3] "Both trees have the same tip labels."
## [4] "Both trees have the same number of nodes: 32."
## [5] "Both trees are rooted."
## [6] "Both trees are ultrametric."
## [7] "29 clades in sim1_seq_tree not in sim2_seq_tree."
## [8] "29 clades in sim2_seq_tree not in sim1_seq_tree."
## [9] "Branching times of clades in common between both trees: see ..$BT\n(node number in parentheses)
```

The comparision in terms of tips,nodes,label and clades also implies that simulation 2 is more similar to lizard tree as it has 27 different clades from expected lizard tree while simulation 1 has 28 different clades

```
knitr::opts_chunk$set(echo = TRUE)

library(ape)
library(phangorn)
library(markovchain)
library(seqinr)
library(msa)
library(lattice)
lizards_sequences<-ape::read.FASTA("lizard_seqs.fasta")
len_list <- lengths(lizards_sequences)
new_seq_list <- list()
new_seq <- c()
for (i in 1:length(lizards_sequences)){
    # extract the base composition of each sequence
    freqx <- ape::base.freq(lizards_sequences[i])
    new_seq <- sample(c("a","c","g","t"),len_list[i],rep=TRUE,prob=freqx)
    new_seq_list[i] <- list(new_seq)
}

# convert result to the DNAbin class
simulated_seq1 <- as.DNAbin(new_seq_list)
names(simulated_seq1) <- names(lizards_sequences)
#ape::write.dna(simulated_seq1, file ="sim1_seq.fasta", format = "fasta", colsep = "")

cat("Base composition of the data from GenBank:")
cat("\n")
ape::base.freq(lizards_sequences)
cat("\n")
```

```
cat("Base composition of the simulated sequences:")
cat("\n")
ape::base.freq(simulated_seq1)

# Simulate a tree with 33 nodes
tree<- TreeSim::sim.bd.taxa(n=33, numbsim=1, lambda=1, mu=0)[[1]]
tree$tip.label <- names(lizards_sequences)
plot(tree, yaxt = "n")

data <- c()
simulated_seq_tree <- list()

DNA_list <- unlist(as.character(lizards_sequences),use.names = FALSE)
DNA_unique <- c("a","c","g","t")
matrix <-  matrix(0, ncol = length(DNA_unique),
                  nrow = length(DNA_unique))

for (i in 1:(length(DNA_list) - 1)) {
    index_of_i <- DNA_unique == DNA_list[i]
    index_of_i_plus_1 <- DNA_unique == DNA_list[i + 1]
    matrix[index_of_i, index_of_i_plus_1] = matrix[index_of_i, index_of_i_plus_1] + 1
}

# Transition matrix is finally computed
Q <- matrix / rowSums(matrix)
#Q_lower <- Q[lower.tri(Q)]
# Other parameters that are needed for phangorn package function
bf <-  ape::base.freq(lizards_sequences)
size <- round(mean(len_list))
# Generating the data based on the tree

simulated_seq_tree <-  phangorn::simSeq(
        tree,
        l = size,
        type = "DNA",
        bf = bf,
        Q = Q)

# change numbers into letters
simulated_seq_tree2 <- lapply(simulated_seq_tree,function(seq){
    #seq = a
    states = c("a","c","g","t")
    seq[which(seq==1)] <- states[1]
    seq[which(seq==2)] <- states[2]
    seq[which(seq==3)] <- states[3]
    seq[which(seq==4)] <- states[4]
    return(seq)
})

simulated_seq2 <- as.DNAbin(simulated_seq_tree2)
# Write the simulated sequence to a fasta file
ape::write.dna(simulated_seq2, file ="sim2_seqs.fasta", format = "fasta", colsep = "")
```

```r
cat("Base composition of the data from GenBank:")
cat("\n")
ape::base.freq(lizards_sequences)
cat("\n")
cat("Base composition of the simulated sequences Q1.1:")
cat("\n")
ape::base.freq(simulated_seq1)
cat("\n")
cat("Base composition of the simulated sequences Q1.2:")
cat("\n")
ape::base.freq(simulated_seq2)

for (i in 1:length(lizards_sequences)){
    cat(paste0("Individual base composition of sequence ",i))
    cat("\n")
    cat("The data from the GenBank:")
    cat("\n")
    print(ape::base.freq(lizards_sequences[i]))
    cat("The simulated data in Q1.1:")
    cat("\n")
    print(ape::base.freq(simulated_seq1[i]))
    cat("The simulated data in Q1.2:")
    cat("\n")
    print(ape::base.freq(simulated_seq2[i]))
    cat("\n")
}
# CG content
comp_true <- ape::base.freq(lizards_sequences)
cg_true <- sum(comp_true[2:3])/sum(comp_true)
comp_s1 <- ape::base.freq(simulated_seq1)
cg_s1 <- sum(comp_s1[2:3])/sum(comp_s1)
comp_s2 <- ape::base.freq(simulated_seq2)
cg_s2 <- sum(comp_s2[2:3])/sum(comp_s2)

cat("In the data from the GenBank:")
cat("\n")
print(cg_true)
cat("In the simulated data in Q1.1:")
cat("\n")
print(cg_s1)
cat("In the simulated data in Q1.2:")
cat("\n")
print(cg_s2)
cat("\n")
# A,C,G,T content
cat("In the data from the GenBank:")
cat("\n")
print(ape::base.freq(lizards_sequences))
cat("In the simulated data in Q1.1:")
cat("\n")
print(ape::base.freq(simulated_seq1))
cat("In the simulated data in Q1.2:")
cat("\n")
```

```r
print(ape::base.freq(simulated_seq2))
# start from the second nucleotide
lizard <- as.character(lizards_sequences)
lizard_seq2 = lapply(lizard, function(a){return(as.character(a[-1]))})
lizard_seq3 = lapply(lizard, function(a){return(as.character(a[c(-1,-2)]))})
ape::write.dna(lizard_seq2, file ="lizard_seq2.fasta", format = "fasta", colsep = "")
ape::write.dna(lizard_seq3, file ="lizard_seq3.fasta", format = "fasta", colsep = "")

# Create 3 versions of lizard sequences

amino_lizard_seq <- read.fasta("lizard_amino_seqs.fasta")
amino_lizard_seq2 <- read.fasta("amino_lizard2.fasta")
amino_lizard_seq3 <- read.fasta("amino_lizard3.fasta")


amino_rev_com_lizard <- read.fasta("rev_com_lizard_amino_acid.fasta")
amino_rev_com_lizard2 <- read.fasta("amino_reversed_lizard2.fasta")
amino_rev_com_lizard3 <- read.fasta("amino_reversed_lizard3.fasta")

stop_code_count <- function(amino_acid_seq){

  stop_count = c()
  for (i in 1:length(amino_acid_seq)) {

    sequence = amino_acid_seq[[i]]
    stop <- sequence[which(sequence == "*")]
    stop_count[i] = length(stop)
  }
  return(stop_count)
}
df = data.frame("lizard_seq" = sum(stop_code_count(amino_lizard_seq)),
           "lizard_seq2" = sum(stop_code_count(amino_lizard_seq2)),
           "lizard_seq3" = sum(stop_code_count(amino_lizard_seq3)),
           "rev_comp_lizard_seq" = sum(stop_code_count(amino_rev_com_lizard)),
           "rev_comp_lizard2" = sum(stop_code_count(amino_rev_com_lizard2)),
           "rev_comp_lizard3" = sum(stop_code_count(amino_rev_com_lizard3))
          )

knitr::kable(df)
lizard <- as.character(lizards_sequences)
sim1 <- as.character(simulated_seq1)
sim2 <- as.character(simulated_seq2)
flat_lizard <- unlist(lizard,use.names = FALSE)
remove_values = c("y","m","r","s")
flat_lizard <- flat_lizard[!flat_lizard %in% remove_values]
mc_lizard <-  markovchainFit(flat_lizard,method = "bootstrap", nboot = 3,
                             name = "Bootstrap Mc",
                             parallel = TRUE)

flat_sim1 <- unlist(sim1,use.names = FALSE)
flat_sim1 <- flat_sim1[!flat_sim1 %in% remove_values]
mc_sim1 <-  markovchainFit(flat_sim1,method = "bootstrap", nboot = 3,
                           name = "Bootstrap Mc",
```

```
                            parallel = TRUE
                            )

flat_sim2 <- unlist(sim2,use.names = FALSE)
flflat_sim2at_sim1 <- flat_sim2[!flat_sim2 %in% remove_values]
mc_sim2 <-  markovchainFit(flat_sim2,method = "bootstrap", nboot = 3,
                            name = "Bootstrap Mc",
                            parallel = TRUE
                            )


plot(mc_lizard$estimate)
plot(mc_sim1$estimate)
plot(mc_sim2$estimate)
aligment_lizard <- msa("lizard_seqs.fasta", type="dna")
aligment_sim1 <- msa("sim1_seq.fasta", type="dna")
aligment_sim2 <- msa("sim2_seqs.fasta", type="dna")

aligment_lizard2 <- msaConvert(aligment_lizard, type="seqinr::alignment")
dist_lizard <- seqinr::dist.alignment(aligment_lizard2,matrix="identity")

aligment_sim1_2 <- msaConvert(aligment_sim1, type="seqinr::alignment")
dist_sim1 <- seqinr::dist.alignment(aligment_sim1_2,matrix="identity")

aligment_sim2_2 <- msaConvert(aligment_sim2, type="seqinr::alignment")
dist_sim2 <- seqinr::dist.alignment(aligment_sim2_2,matrix="identity")

# the seqinr::dist.alignment computes the square root distance
dm_lizard <- as.matrix(dist_lizard)
dm_sim1 <- as.matrix(dist_sim1)
dm_sim2 <- as.matrix(dist_sim2)
# heatmaps
new.palette=colorRampPalette(c("black","red","yellow","white"),space="rgb")
levelplot(dm_lizard[1:ncol(dm_lizard),ncol(dm_lizard):1],col.regions=new.palette(20))
levelplot(dm_sim1[1:ncol(dm_sim1),ncol(dm_sim1):1],col.regions=new.palette(20))
levelplot(dm_sim2[1:ncol(dm_sim2),ncol(dm_sim2):1],col.regions=new.palette(20))

library(phangorn)


upgma_tree <- function(x){
  tree <- upgma(x)
  tree$tip.label <- names(lizards_sequences)
  return(tree)
}


lizard_seq_tree <- upgma_tree(dm_lizard)
#plot(lizard_seq_tree, main="Phylogenetic Tree of Lizard DNA")



sim1_seq_tree <- upgma_tree(dm_sim1)
```

```
#plot(sim1_seq_tree, main="Phylogenetic Tree of Lizard DNA sim1")


sim2_seq_tree <- upgma_tree(dm_sim2)
#plot(sim2_seq_tree, main="Phylogenetic Tree of Lizard DNAs sim2")


boot_lizard <-  boot.phylo(lizard_seq_tree, dm_lizard, FUN=upgma_tree, trees  = TRUE,quiet = TRUE)
sim1_boot     <- boot.phylo(sim1_seq_tree, dm_sim1, FUN=upgma_tree, trees  = TRUE,quiet = TRUE)
sim2_boot     <- boot.phylo(sim2_seq_tree, dm_sim2, FUN=upgma_tree, trees  = TRUE,quiet = TRUE)


lizard_clade = prop.clades(lizard_seq_tree,boot_lizard$trees,rooted = TRUE)
sim1_clade = prop.clades(sim1_seq_tree,sim1_boot$trees,rooted = TRUE)
sim2_clade = prop.clades(sim2_seq_tree,sim2_boot$trees,rooted = TRUE)
layout(1)
par(mar = rep(2, 4))
plot(lizard_seq_tree, main = "Phylogenetic Tree of Lizard DNA & Clades")
nodelabels(lizard_clade)
legend("bottomright", legend = c("Clades"), pch = 22,
       pt.bg = c("lightblue"), pt.cex = 2.5)


layout(1)
par(mar = rep(2, 4))
plot(sim1_seq_tree, main = "Sim 1 Phylogenetic Tree of Lizard DNA & Clades")
nodelabels(sim1_clade)
legend("bottomright", legend = c("Clades"), pch = 22,
       pt.bg = c("lightblue"), pt.cex = 2.5)



plot(sim2_seq_tree, main = "Sim 2 Phylogenetic Tree of Lizard DNA & Clades")
nodelabels(sim2_clade)
legend("bottomright", legend = c("Clades"), pch = 22,
       pt.bg = c( "lightblue"), pt.cex = 2.5)


library(phangorn)
path_diff_sim1_lizard <- path.dist(sim1_seq_tree,lizard_seq_tree)
path_diff_sim2_lizard <- path.dist(sim2_seq_tree,lizard_seq_tree)
path_diff_sim1_sim2 <- path.dist(sim1_seq_tree,sim2_seq_tree)


felenstein_diff_sim1_lizard <- KF.dist(sim1_seq_tree,lizard_seq_tree)
felenstein_diff_sim2_lizard <- KF.dist(sim2_seq_tree,lizard_seq_tree)
felenstein_diff_sim1_sim2 <- KF.dist(sim1_seq_tree,sim2_seq_tree)

result = data.frame(path_difference=c(path_diff_sim1_lizard,
                                      path_diff_sim2_lizard,
                                      path_diff_sim1_sim2),
                    felenstein_distance=c(felenstein_diff_sim1_lizard,
                                          felenstein_diff_sim2_lizard,
```

```
                                        felenstein_diff_sim1_sim2)
                )
rownames(result) = c("sim1 vs lizard","sim2 vs lizard","sim1 vs sim2")
knitr::kable(result)
library(distory)
#distance between edeges using geodesic
geodesic_dist = dist.multiPhylo(list(lizard_seq_tree,sim1_seq_tree,sim2_seq_tree), method = "geodesic")
geodesic_dist = as.matrix(geodesic_dist)
geodesic_dist

comp1 <- comparePhylo(lizard_seq_tree,sim1_seq_tree)
comp2 <- comparePhylo(lizard_seq_tree,sim2_seq_tree)
comp12 <- comparePhylo(sim1_seq_tree,sim2_seq_tree)

comp1$messages
comp2$messages
comp12$messages
```