Lab 2 - Gr. 14 - Bioinformatics (732A93)

Andreas Stasinakis (andst745), Hector Plata (hecpl268), Julius Kittler (julki092), Mim Kemal Tekin (mimte666), Stefano Toffol (steto820)

Assignment 1

Question 1.1

Starting from 33 DNA sequence of various species of casque-headed lizard (Basiliscus basiliscus), other 33 sequences of nucleotides have been generated. The sampling probabilities are the same of the real proportions of the original dataset.
After the artificial DNA has been created, the base frequencies are compared in Table 1. As expected, the observed proportions of the generated data closely resamble the theoretical ones.

Question 1.2

- Created one phylogenetic tree with 33 tips
- For each original DNA sequence of the 33 available, used the function `simSeq(.)` from package `phangorn` to simulate the sequences.
- Result: 33 phylogenetic tree, one for DNA sequence, each with 33 tips.

Is the result Krzysztof wanted? I've read the suggested materials and the lecture slides, but I am still not sure how to use the tree in order to simulate the sequences.

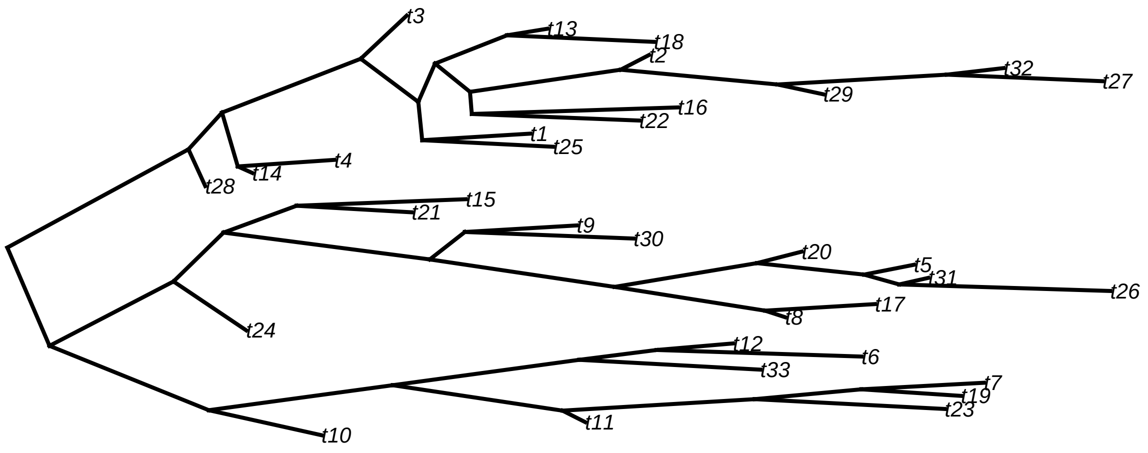

Table 1: Base frequencies of the 33 original and generated DNA sequences.

Base	Original frequency	Simulated frequency
a	0.3121	0.3120
c	0.2052	0.2045
g	0.2307	0.2329
t	0.2519	0.2505

Assignment 2

Note that, by convention, a *coding strand* is used when displaying a DNA sequence. "A coding strand, is the segment within double-stranded DNA that runs from 5' to 3', and which is complementary to the antisense strand of DNA, or template strand, which runs from 3' to 5"' (https://en.wikipedia.org/wiki/Sense_strand).

https://www.ncbi.nlm.nih.gov/nuccore/CU329670

2.1

2.2

2.3

Assignment 3

3.1

3.2

Appendix

```r
knitr::opts_chunk$set(fig.width = 7, fig.height = 3, echo = FALSE)

library(dplyr)
library(tidyr)
library(magrittr)
library(ape)           # This is a general R-package for phylogenetics and comparative methods
library(seqinr)        # This is an specialized package for nucleotide sequence management
library(phangorn)
library(kableExtra)

source("732A51_BioinformaticsHT2018_Lab02_GenBankGetCode.R")
# ------------------------------------------------------------------------------
# Question 1.1
# ------------------------------------------------------------------------------

lizards_format_sequences <- read.fasta(file = "lizard_seqs.fasta")  # Alternative version of the file
# Useful in some ways?

n <- length(lizards_accession_numbers)  # Number of sequences to reproduce
p <- base.freq(lizards_sequences)  # Probability of the base sequences
simulated_lizards <- list()  # Object that will contain our simulated data

# The names of the simulated data are the original names + "_sim"
# NOTE: it does not follow the format from GenBank
simulated_names <- paste(lizards_accession_numbers, "_sim", sep = "")

set.seed(1535)  # Set seed in order to reproduce the experiment
for(i in 1:n) {  # Cycle through every single object of the lizard_sequences
```

```r
  len_seq <- length(lizards_sequences[[i]])  # Lenght of each sequence
  simulated_lizards[[ simulated_names[i] ]] <-
    sample(c("a", "c", "g", "t"), len_seq, replace = T, prob = p)
  # Creating the artificial sequence sampling with probabilities p equal to the original ones
  # NOTE: we use the general distribution for every single sequence
}

# Save as fasta file
write.dna(simulated_lizards, file = "simulated_lizards.fasta", format = "fasta", append = F,
          nbcol = 6, colsep = " ", colw = 10)

# Table with simulated base frequency
df_table <- data.frame("Base" = c("a", "c", "g", "t"),
                       "Original\nfrequency" = p,
                       "Simulated\nfrequency" = base.freq(as.DNAbin(simulated_lizards)),
                       row.names = NULL)
kable(df_table, booktabs = T, align = c("r", "l", "l"), digits = c(NA, 4, 4),
      col.names = c("Base", "Original\nfrequency", "Simulated\nfrequency"), format = "latex",
      caption = "Base frequencies of the 33 original and generated DNA sequences.")
# ------------------------------------------------------------------------------
# Question 1.2
# ------------------------------------------------------------------------------

################################################################################
# Create the tree
################################################################################

# Set the random seed for reproducibility once again
set.seed(1545)
# Simulate the tree: it has to be with 33 tips. We try to generate a tree with uniform distribution
# (default function)
simulated_tree <- rtree(33)

# Plot the tree:
par(lend = 2)
par(mar = rep(2, 4))
par(cex = 0.8)

plot(simulated_tree, type = "cladogram", edge.width = 2)  # Cladogram template, the easiest to
                                                          # understand for this tree


################################################################################
# Simulate the sequences
################################################################################

# Object where to store the randomic sequences
simulated_tree_seq <- list()
# The names of the simulated data are the original names + "_sim_tree"
# NOTE: it does not follow the format from GenBank
simulated_names_tree <- paste(lizards_accession_numbers, "_sim_tree", sep = "")
```

```r
# We now define a function to compute a partially random transiction matrix:
# It first calculate first-order Markov transition matrix where each *row*
# corresponds to a single run of the Markov chain. It then add some random noise.

trans.matrix <- function(DNA_seq) {

  # Retrieve unique elements
  DNA_unique <- unique(DNA_seq)

  # Create an empty matrix
  matrix <- matrix(0, ncol = length(DNA_unique), nrow=length(DNA_unique))

  # Fill it: to count i and element i + 1 and add one in the corresponding cell of the matrix.
  for (i in 1:(length(DNA_seq) - 1)) {
    index_of_i <- DNA_unique == DNA_seq[i]
    index_of_i_plus_1 <- DNA_unique == DNA_seq[i + 1]
    matrix[index_of_i, index_of_i_plus_1] = matrix[index_of_i, index_of_i_plus_1] + 1
  }

  # Normalize it
  matrix <- matrix / rowSums(matrix)

  # Add random noise, keeping the rowSum equal to 1
  # Proceed row by row
  for(i in nrow(matrix)) {
    noise <- rnorm(1, mean = 0,  sd = 0.01)   # Random quantity to ADD/SUBtract
    ind_add <- floor(runif(1, 1, nrow(matrix)+1))  # Column index where to add
    ind_sub <- floor(runif(1, 1, nrow(matrix)+1))  # Column index where to subtract
    matrix[i,ind_add] <- matrix[i,ind_add] + noise  # Add noise to one prob...
    matrix[i,ind_sub] <- matrix[i,ind_sub] - noise  # ...and subtract from another
  }

  return(matrix)
}


# Time to generate! Set the seed again (better safe than sorry)
set.seed(1545)

# Create one sequence for each one of the originals
for(i in 1:n) {
  # Problem: original sequences include unknown nucleotides --> delete them
  known_nucleotides <- lizards_sequences[[i]][lizards_sequences[[i]] %in% c(18, 28, 48, 88)]

  # Decide lenght randomically: we take the lenght of the original known sequence +
  # uniform random number equal max to 10% of the original lenght
  len_seq <- length(known_nucleotides)
  len_seq <- len_seq + floor(runif(1, -floor(len_seq/10), floor(len_seq/10)))

  # We re-use the same probabilities computed before, referring to the original dataset.
  # We also compute the transiction matrix using the function defined above.
  simulated_tree_seq[[ simulated_names_tree[i] ]] <-
    simSeq(simulated_tree, l = len_seq, bf = p, Q = trans.matrix(known_nucleotides))
}
# ------------------------------------------------------------------------------
# Question 2.1
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# Question 2.2
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# Question 2.2
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# Question 3.1
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# Question 3.2
# ------------------------------------------------------------------------------
```