732A51 Bioinformatics

Lab 2

Johannes Hedström, Mikael Montén

STIMA
Institutionen för datavetenskap
Linköpings universitet

2024-11-21

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.

```
library(ape)
library(seqinr)

## 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.

```
#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(lizards_sequences[i][1]) # extract base compositions
    seq_len <- length(lizards_sequences[[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.

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
  CpG <- counts['cg'] / sum(counts)

  base_comp <- base.freq(seq[1]) # base comp
  GC <- GC.content(seq[1]) # GC and AT content
  AT <- 1 - GC

  return(t(c('Species'=names(seq),base_comp[1],base_comp[2],base_comp[3],base_comp[4],GC,AT, 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','AT', '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	AT	CpG
JF806202	0.28987	0.20261	0.24373	0.26379	0.44634	0.55366	0.01304
ARTIFICAL_JF806202	0.27856	0.19739	0.24449	0.27956	0.44188	0.55812	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.

```
# 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.

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.

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().