STAT115 Homework 1

Peter Shen

2018-01-26

Part I– Introduction to R

This first question is an introduction to using RStudio and knitting your homework for final submission. We will be downloading a short nucleotide sequence and producing summary statistics associated with the string. The sequence originates from a mitochondrial control region from the homo sapiens genome and will be retrieved from NCBI.

1. Download and install the ape and seqinr packages. Include the installation commands in your R code but make sure the chunk does not get executed when you knit the document for final submission (hint: eval=FALSE).

```
install.packages("ape")
install.packages("seqinr")
```

2. Load the ape and seqinr packages into the current workspace. Hint: library().

```
library(ape)
```

```
## Warning: package 'ape' was built under R version 3.4.2
```

```
library(seqinr)
```

```
##
## Attaching package: 'seqinr'
```

```
## The following objects are masked from 'package:ape':
##
##     as.alignment, consensus
```

3. Use the read.GenBank() function available in the ape library to read in the following accession number: MF320132. Use the as.character=TRUE option to return the nucleotide sequence. Assign the result to a variable named seq_1.

```
seq_1 <- read.GenBank("MF320132", as.character = T)
```

4. What species do these reads come from? Hint: use the attributes() command to view available attributes for seq_1 and then use the attr() command to access that.

```
attributes(seq_1)
```

```
## $names
## [1] "MF320132"
##
## $species
## [1] "Homo_sapiens"
```

```
attr(seq_1, "species")
```

```
## [1] "Homo_sapiens"
```

The sequence is from Homo sapiens

5. How long is the nucleotide sequence? Hint: use the length function.

```
length(seq_1$MF320132)
```

```
## [1] 304
```

6. Report the overall nucleotide counts of the sequence. Hint: you may use the table function.

```
table(seq_1$MF320132)
```

```
##
##  a  c  g  n  t
## 77 86 46 20 75
```

7. Replace the sequence with an edited version in which the "n" nucleotides have been removed. What is the length of the sequence now?

```
seq_1_replaced <- seq_1
seq_1_replaced$MF320132 <- seq_1$MF320132[seq_1$MF320132!="n"]
```

7. Again report the overall nucleotide counts of the modified sequence but instead convert these counts to proportions. Hint: use the prop.table function. What is the overall GC-content of the sequence (i.e. what proportion of the total sequence is composed of either G or C bases?).

```
length(seq_1_replaced$MF320132)
```

```
## [1] 284
```

8. Use the set.seed() function to set a seed such that your results may be reproducible (set.seed(12345)). Generate a random uniform vector (over the range 0-1) of length equal to the answer you found in question 5. Call this vector x.random.

```
set.seed(5)
x.random <- runif(304, min=0, max=1)
```

9. Now create a logical vector indicating whether x.random is less than or equal to 0.5. Call this vector x.logical.

```
x.logical <- sapply(x.random, function(x) {ifelse(x <= 0.5, T, F)})
```

10. Finally print just those nucleotides corresponding to x.logical==TRUE. What is the GC-content of this new sequence?

```
seq_1$MF320132[x.logical]
```

```
##   [1] "t" "t" "t" "t" "g" "g" "g" "g" "g" "t" "c" "g" "c" "g" "a" "a" "c"
##  [18] "a" "t" "t" "g" "c" "g" "a" "g" "g" "g" "a" "c" "c" "c" "t" "t" "a"
##  [35] "c" "g" "c" "t" "t" "t" "t" "c" "c" "c" "c" "a" "t" "c" "t" "t" "t"
##  [52] "t" "c" "c" "a" "c" "a" "c" "g" "t" "t" "a" "a" "t" "a" "g" "g" "c"
##  [69] "g" "a" "c" "a" "t" "a" "c" "t" "a" "c" "t" "t" "t" "a" "t" "a" "n"
##  [86] "n" "n" "n" "n" "n" "n" "n" "n" "n" "a" "a" "t" "a" "t" "g" "t" "t"
## [103] "c" "a" "g" "c" "t" "c" "c" "a" "c" "a" "c" "g" "a" "c" "t" "a" "c"
## [120] "a" "a" "a" "t" "t" "a" "a" "a" "c" "c" "c" "c" "c" "t" "c" "c" "c"
## [137] "g" "c" "t" "t" "c" "t" "c" "c" "a" "g" "a" "t" "a" "a" "a" "a" "t"
## [154] "t" "c" "c" "a" "a" "a" "c" "a"
```

```
tbl <- table(seq_1$MF320132[x.logical])
gc <- as.numeric((tbl["g"] + tbl["c"])/length(seq_1$MF320132[x.logical]))
paste0("GC content is ", gc*100, "%")
```

```
## [1] "GC content is 41.6149068322981%"
```

Part II– Introduction to python

In this part, we will do a python programing exercise so that you start to learn and use python as soon as possible. The following code is intended to translate the DNA sequence to an equivalent amino acid sequence, please complete it and use the data below to test your result. The input is the dna_seq string and the intended output should match the protein_seq string.

```
dna_seq = 'GCGTTTGACCGCGCTTGGGTGGCCTGGGACCCTGTGGGAGGCTTCCCCGGCGCCGAGAGCCCTGGCTGACGGCTGATGGGGAGGAGCCGGCG

protein_seq = 'MGRSRRAEKATGSPVPSPARDRCGKPGGASAGPAERTSEVKSLVYLPLGAGLGPQPLP_'


def translate_DNA(sequence):
    start_codon = 'ATG'
    stop_codons = ('TAA', 'TAG', 'TGA')
    codontable = {
        'ATA': 'I', 'ATC': 'I', 'ATT': 'I', 'ATG': 'M',
        'ACA': 'T', 'ACC': 'T', 'ACG': 'T', 'ACT': 'T',
        'AAC': 'N', 'AAT': 'N', 'AAA': 'K', 'AAG': 'K',
        'AGC': 'S', 'AGT': 'S', 'AGA': 'R', 'AGG': 'R',
        'CTA': 'L', 'CTC': 'L', 'CTG': 'L', 'CTT': 'L',
        'CCA': 'P', 'CCC': 'P', 'CCG': 'P', 'CCT': 'P',
        'CAC': 'H', 'CAT': 'H', 'CAA': 'Q', 'CAG': 'Q',
        'CGA': 'R', 'CGC': 'R', 'CGG': 'R', 'CGT': 'R',
        'GTA': 'V', 'GTC': 'V', 'GTG': 'V', 'GTT': 'V',
        'GCA': 'A', 'GCC': 'A', 'GCG': 'A', 'GCT': 'A',
        'GAC': 'D', 'GAT': 'D', 'GAA': 'E', 'GAG': 'E',
        'GGA': 'G', 'GGC': 'G', 'GGG': 'G', 'GGT': 'G',
        'TCA': 'S', 'TCC': 'S', 'TCG': 'S', 'TCT': 'S',
        'TTC': 'F', 'TTT': 'F', 'TTA': 'L', 'TTG': 'L',
        'TAC': 'Y', 'TAT': 'Y', 'TAA': '_', 'TAG': '_',
        'TGC': 'C', 'TGT': 'C', 'TGA': '_', 'TGG': 'W'
    }

    start = sequence.find(start_codon)
    # Create a codon list to store codons generated from coding seq.
    codons = []

    for i in range(start, len(sequence), 3):
        codon = sequence[i:i + 3]
        if (codon in stop_codons):
            break
        else:
            codons.append(codon)

    # Translate condons to protein seq.
    protein_sequence = ''.join([codontable[codon] for codon in codons])
    return "{0}_".format(protein_sequence)

print(translate_DNA(dna_seq))
```

```
## MGRSRRAEKATGSPVPSPARDRCGKPGGASAGPAERTSEVKSLVYLPLGAGLGPQPLP_
```

Submission

Please submit your solution directly on the canvas website. Please provide your code (.Rmd) and a pdf file for your final write-up. Please pay attention to the clarity and cleanness of your homework. Page numbers and figure or table numbers are highly recommended for easier reference.

The teaching fellows will grade your homework and give the grades with feedback through canvas within one week after the due date. Some of the questions might not have a unique or optimal solution. TFs will grade those according to your creativity and effort on exploration, especially in the graduate-level questions.