Homework Assignment

Sequence Data

2023-02-09

Task A

```r
# Run the query
result <- query("query1", query = "SP=Chlamydia trachomatis AND M=DNA")
print(paste(result$nelem, "sequences were retrieved."))
```

```
## [1] "43564 sequences were retrieved."
```

Task B

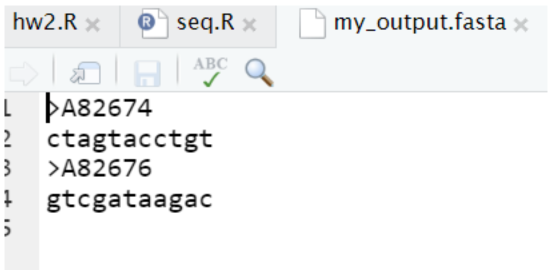
```r
# Get all the sequences
all_sequences <- result$req
# Get the length vector
length_vector <- getLength(all_sequences)

# Get the shortest sequence(s)
shortest_seq <- all_sequences[length_vector == min(length_vector)]

# Print the length of the shortest sequence(s)
print(paste("Length of shortest sequence:", min(length_vector)))
```

```
## [1] "Length of shortest sequence: 11"
```

```r
# Print the accession number(s) of the shortest sequence(s)
print(paste("Accession Numbers:", paste(getName(shortest_seq), collapse = ", ")))
```

```
## [1] "Accession Numbers: A82674, A82676"
```

```r
# Export the data to a FASTA file
write.fasta(getSequence(shortest_seq), names = getName(shortest_seq), file.out = "my_output.fasta")
```



```
>A82674
ctagtacctgt
>A82676
gtcgataagac
```

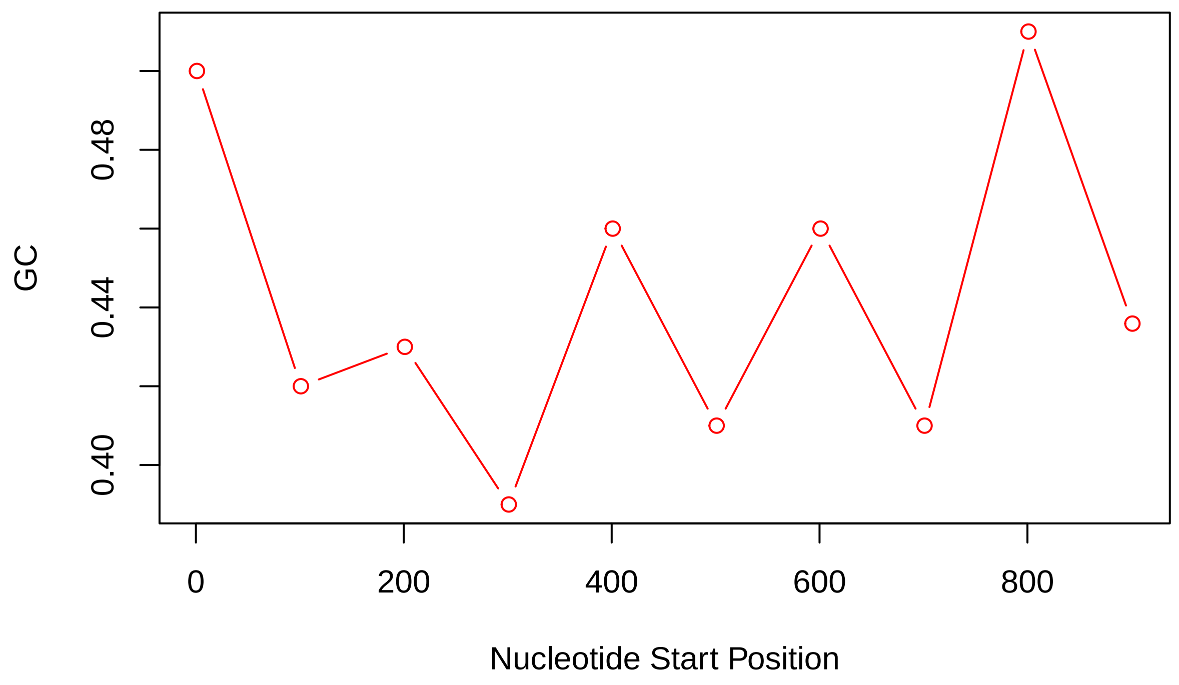
Figure 1: “fasta_output”

Task C

```
# Get the 100th sequence
seq_100 <- getSequence(all_sequences[[100]])

# i. Calculate GC content for each 500-nucleotide chunk
gc_chunks <- sapply(split(seq_100, ceiling(seq_along(seq_100)/500)),
    GC)
print(gc_chunks)
```

```
##         1         2
## 0.4400000 0.4464692
```

```
# ii. Create a sliding window scattered plot of GC content
# using red lines
plot(seq(from = 1, to = length(seq_100), by = 100), sapply(split(seq_100,
    ceiling(seq_along(seq_100)/100)), GC), type = "b", col = "red",
    xlab = "Nucleotide Start Position", ylab = "GC")
```



Task D

```
# Get the longest sequence
longest_seq <- all_sequences[length_vector == max(length_vector)]
lseq <- getSequence(longest_seq)

# i. Print the count of all the nucleotide bases in the sequence
```

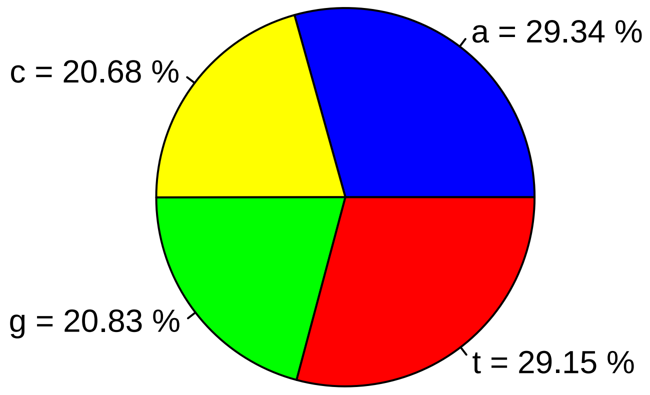
```
no_of_bases <- table(lseq)
print(no_of_bases)
```

```
## lseq
##      a      c      g      t
## 317978 224192 225744 315979
```

```
# ii. Calculate the percentage of each base and plot it using a pie chart
prop_of_bases <- proportions(no_of_bases)
print(prop_of_bases)
```

```
## lseq
##         a         c         g         t
## 0.2933666 0.2068396 0.2082715 0.2915223
```

```
colors = c("blue", "yellow", "green", "red")
labs <- names(prop_of_bases)

pie(prop_of_bases, labels = paste(labs, "=", round(prop_of_bases * 100, 2), "%"), col = colors)
```



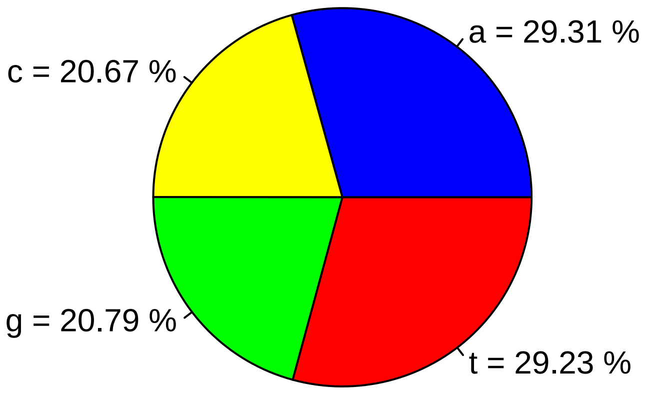
```
# iii. Generate a random sequence with the same length and base composition
random_seq <- sample(names(no_of_bases), getLength(lseq), replace = TRUE, prob = prop_of_bases)

# iv. Repeat i. and ii. for the randomly generated sequence from iii.
random_count <- table(random_seq)
print(random_count)
```

```
## random_seq
##      a      c      g      t
## 317726 224030 225333 316804
```

```
random_prop <- proportions(random_count)
print(random_prop)
```

```
## random_seq
##         a         c         g         t
## 0.2931341 0.2066901 0.2078923 0.2922835
```

```
pie(random_prop, labels = paste(names(random_prop), "=", round(random_prop * 100, 2), "%"), col = color
```



Task E

```
# Create a function that return the top 3 most frequent codons
freq_codons <- function (seq) {
  codons <- count(seq, 3)
  codons <- codons[order(codons, decreasing = TRUE)]
  return (head(codons, 3))
}

# Testing the function on 2 sequences
freq_codons(getSequence(all_sequences[[125]]))
```

```
##
## tct aaa ata
##  10   7   7
```

```
freq_codons(getSequence(all_sequences[[567]]))
```

```
##
## ttt tct tta
##  68  52  50
```

Task F

```
# Write a function that returns all under-represented DNA words with a
# specific length
under_represented <- function (seq, length) {
  rep <- rho(seq, wordsize = length)
  return (rep[rep < 1])
}

# Testing the function on two sequences of length 2 and 4 respectively
under_represented(getSequence(all_sequences[[876]]), 2)
```

```
##
##        ac        ag        ca        cc        cg        gt        ta
## 0.8060463 0.9672717 0.9778267 0.9453314 0.9167125 0.7410564 0.8020845
```

```
under_represented(getSequence(all_sequences[[1024]]), 4)
```

```
##
##      aaac      aaat      aaca      aacc      aacg      aact      aagg      aagt
## 0.9760984 0.3798533 0.9760984 0.0000000 0.7568299 0.6663000 0.5018111 0.4417859
##      aata      aatg      aatt      acaa      acat      accc      accg      acga
## 0.5697799 0.8835718 0.1944703 0.3253661 0.6663000 0.0000000 0.6637771 0.3784150
##      acgc      acgt      acta      actg      actt      agac      agct      agga
## 0.6637771 0.7749359 0.6663000 0.3874679 0.6822402 0.0000000 0.7749359 0.5018111
##      aggg      aggt      agta      agtc      agtg      agtt      ataa      atac
## 0.5836282 0.2569081 0.8835718 0.3874679 0.5138162 0.6785324 0.3798533 0.9994500
##      atag      atca      atgt      atta      attg      caac      caat      cacg
## 0.8835718 0.6663000 0.4523549 0.5834110 0.9047098 0.5707242 0.3331500 0.6637771
##      cact      cagc      cagg      cagt      cata      catg      ccaa      ccac
## 0.0000000 0.6637771 0.8802261 0.3874679 0.6663000 0.3874679 0.5707242 0.0000000
##      ccag      ccca      cccc      cccg      ccga      ccgg      ccgt      cctg
## 0.6637771 0.0000000 0.0000000 0.0000000 0.6637771 0.0000000 0.6796569 0.0000000
##      cgaa      cgac      cgag      cgat      cgcg      cgta      cgtc      cgtt
## 0.7568299 0.0000000 0.8802261 0.3874679 0.0000000 0.7749359 0.6796569 0.3967375
##      ctaa      ctag      ctat      ctca      ctcg      ctga      ctgg      cttg
## 0.9994500 0.7749359 0.3411201 0.0000000 0.6796569 0.7749359 0.4506421 0.7934750
##      gaac      gaca      gacg      gact      gagg      gcat      gcca      gccc
## 0.3784150 0.0000000 0.4401130 0.3874679 0.8754423 0.7749359 0.6637771 0.0000000
##      gccg      gcga      gcgt      gcta      gctc      ggaa      ggca      ggcc
## 0.7720016 0.0000000 0.9012841 0.7749359 0.0000000 0.2509056 0.4401130 0.0000000
##      gggc      gggg      gggt      ggtc      ggtg      ggtt      gtaa      gtac
## 0.5118706 0.3393925 0.2987953 0.9012841 0.2987953 0.5261084 0.8835718 0.3874679
##      gtag      gtca      gtcc      gtcg      gtct      gtgc      gtgg      gtgt
## 0.5138162 0.3874679 0.6796569 0.9012841 0.7934750 0.4506421 0.8963858 0.7891627
##      gtta      gttc      taaa      taac      taag      taat      taca      tacc
## 0.6785324 0.3967375 0.5697799 0.6663000 0.2208929 0.7778814 0.9994500 0.5843779
##      taga      tagg      tagt      tata      tatg      tcac      tcag      tcat
## 0.8835718 0.5138162 0.2261775 0.9723517 0.9047098 0.5843779 0.0000000 0.3411201
##      tccg      tcgt      tcta      tctg      tgac      tgag      tgct      tgga
## 0.6796569 0.7934750 0.6822402 0.7934750 0.3874679 0.7707243 0.3967375 0.7707243
##      tggc      tggg      tgta      tgtc      ttaa      ttag      ttat      ttcc
## 0.0000000 0.8963858 0.6785324 0.7934750 0.1944703 0.4523549 0.9956137 0.5983582
##      ttga      ttgg      ttta
## 0.4523549 0.5261084 0.3982455
```