Bioinformatics Lab 1. Introduction to R, Online bioinformatics resources, nucleotide frequency statistics. Paste your color-coded responses and output in this Word document, re-saved with your name in the file name. For example, Brett_McKinney_lab1.docx. You should also save your commands in an R script with a similar file name with R extension. Submit your report and any scripts on Harvey.

Noah L. Schrick - 1492657

A. Use the R function **`seq`** to create the following vectors. Paste your commands and output.

a. Create a vector where the first element is 1, the last element is 33, with an increment of 2 between elements.

```
AAvec <- seq(from = 1, to = 33, by = 2)
[1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33
```

---

b. Create a vector with 15 equally spaced elements in which the first element is 7 and the last element is 40. Hint: use ?seq for help and the option length.out option.

```
ABvec <- seq(from = 7, to = 40, length.out = 15)
 [1]  7.000000  9.357143 11.714286 14.071429 16.428571 18.785714
        21.142857 23.500000 25.857143 28.214286 30.571429
        32.928571 35.285714
[14] 37.642857 40.000000
```

---

c. Use the **`sample`** function to create a vector with variable name `my.dna` that consists of 20 uniformly-random letters “A”, “C”, “G”, and “T”.

```
my.dna <- sample(c("A", "C", "G", "T"), size = 20, replace = T)
[1] "G" "C" "T" "G" "T" "C" "C" "T" "A" "C" "A" "A" "C" "A" "T" "G" "A" "A"
        "A" "G"
```

---

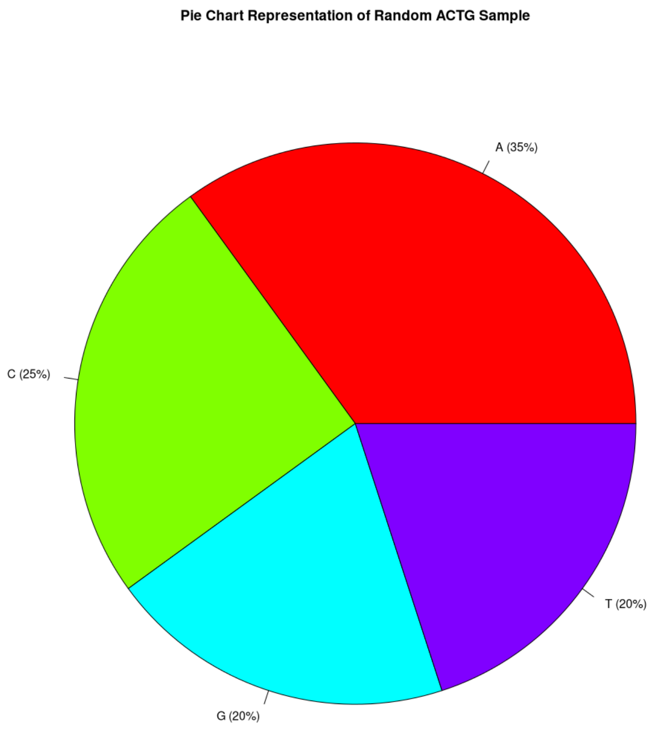
d. Use the == logic operator and other R functions on your `my.dna` variable to determine how many of the letters are “A”. Hint: you can use `sum` on a TRUE/FALSE vector or you can use the functions **`which`** and **`length`**.

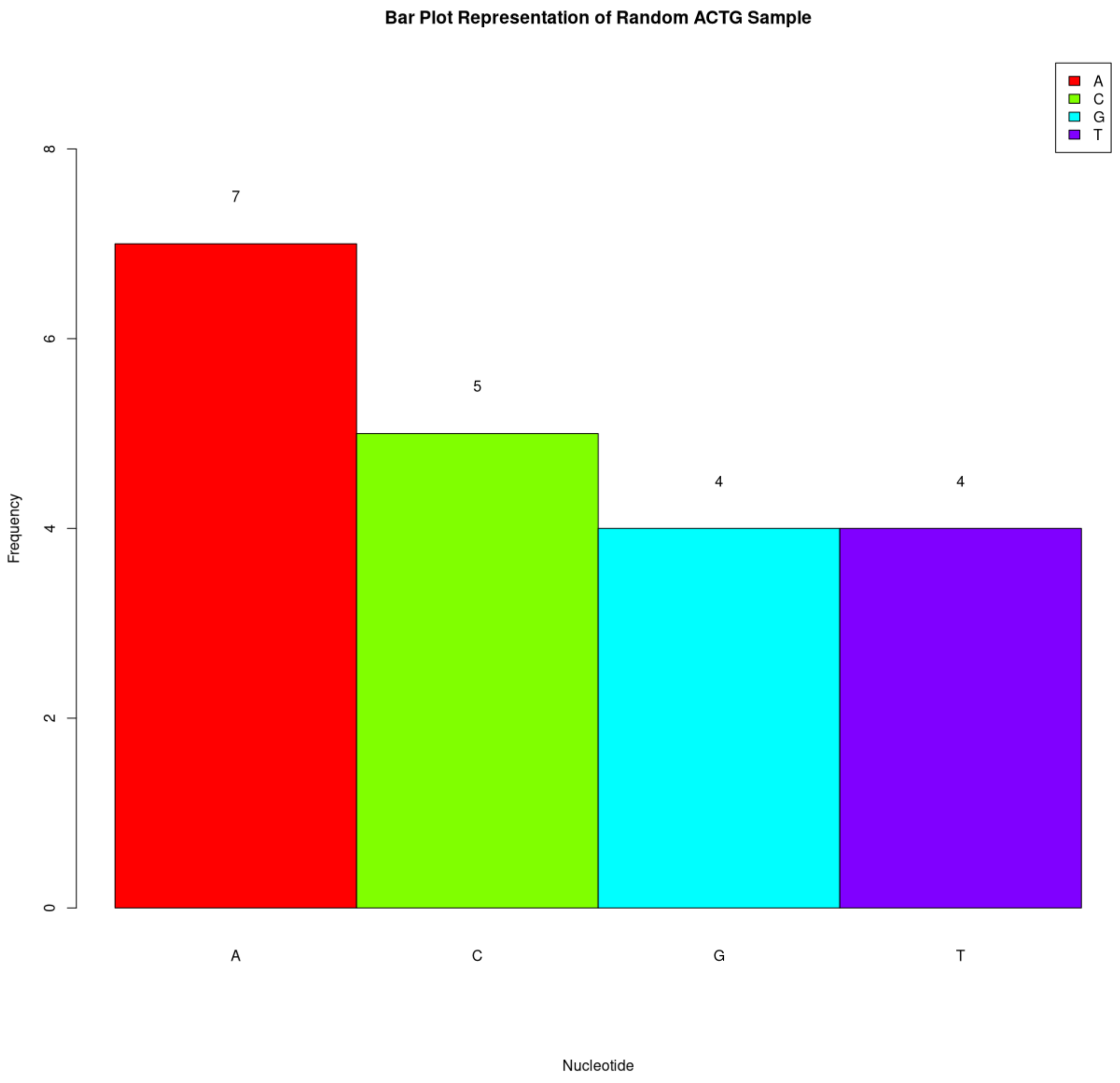
```
my.dna.A <- length(which(my.dna == "A"))
[1] 7
```

---

e. Confirm your answer in d with the **`table`**`(my.dna)`. From the output of table, create a pie chart and barplot. Add x and y labels to your barplot.

```
my.dna.table <- table(my.dna)

my.dna
A C G T
7 5 4 4

my.dna.table.df <- as.data.frame(my.dna.table)
cols <- rainbow(nrow(my.dna.table))
my.dna.table.df$percent <-
             round(100*my.dna.table.df$Freq/sum(my.dna.table.df$Freq),
                        digits = 1)
my.dna.table.df$label <- paste(my.dna.table.df$my.dna,
              " (", my.dna.table.df$percent, "%)", sep = "")

pie(my.dna.table.df$Freq, labels = my.dna.table.df$label, col = cols,
             main = "Pie Chart Representation of Random ACTG Sample")


bp <- barplot(as.matrix(my.dna.table), beside = TRUE, xlab =
                                                                              "Nucleotide",
             ylab = "Frequency", ylim = c(-1, max(as.numeric(my.dna.table))
                                                         +2),
            main = "Bar Plot Representation of Random ACTG Sample", col =
                        cols, legend = TRUE)

text(x = bp, y = my.dna.table + 0.5, labels = as.numeric(my.dna.table))
text(x = bp, y = -0.5, labels = names(my.dna.table))
```




Bar Plot Representation of Random ACTG Sample
A
C
G
T
8
6
4
2
0
Frequency
7
5
4
4
A
C
G
T
Nucleotide

---

f. Use the **`sample`** function with the option `prob=c(.1,.4,.4,.1)`to create a vector with variable name `my.dna2` that consists of 20 non-uniformly random letters “A”, “C”, “G”, and “T”. Use **table** to show the nucleotide counts.

```
my.dna2 <- sample(c("A", "C", "G", "T"), size = 20, replace = T,
          prob = c(0.1, 0.4, 0.4, 0.1))
my.dna2.table <- table(my.dna2)
my.dna2
 A  C  G
 2 11  7
```

B. Basic NCBI search. Search NCBI (http://www.ncbi.nlm.nih.gov/) for “Alzheimer human.” This will take you to Entrez gene, which shows you the hits in the NCBI databases. Choose the top hit for Alzheimer under “Gene” information. 1. What is the name of the gene? 2. What chromosome is the gene on? 3. What species has the most similar gene to the human version (on the right column, click on Homologene)? Go back to the gene page, scroll down to genomic regions, select fasta and the “send” (download) to a fasta file of the gene’s sequence. Put the file in a directory where your R session can find it. Name the file something like genename.fa.

1.) APOE
2.) Chromosome 19
3.) Chimpanzee

C. Reading fasta files, nucleotide and dinucleotide frequencies. Add the following to your R script to install and load the `seqinr` library. The install line only needs to be done once then can be commented out. The library command must be run each time you start a new R session.

```
#install.packages("seqinr") # only have to do this if not installed
library(seqinr) # do this once during the R session
```

Next ensure your R script and fasta file are in the same directory, and ensure that this directory is set as your working directory (use `setwd`, or on the lower right panel click the “File” tab, navigate to the directory, click the “More...” button, and set as working directory). Add the following to your R script to read your fasta file into an R variable.

```
my.fasta <- read.fasta(file="apoe.fa", as.string=TRUE)
```

1. Using the **`class`** function, what data type is `my.fasta`?

```
class(myfasta)
[1] "list"
```

Use the following commands with **`strsplit`** and **unlist** to convert to a vector of characters:

```
my.fasta.string <- my.fasta[[1]][1]
my.fasta.list <- strsplit(my.fasta.string,"")
my.fasta.vec <- unlist(my.fasta.list)
```

2. Convert the above commands into a function and add to your script. Fill in the braces below.

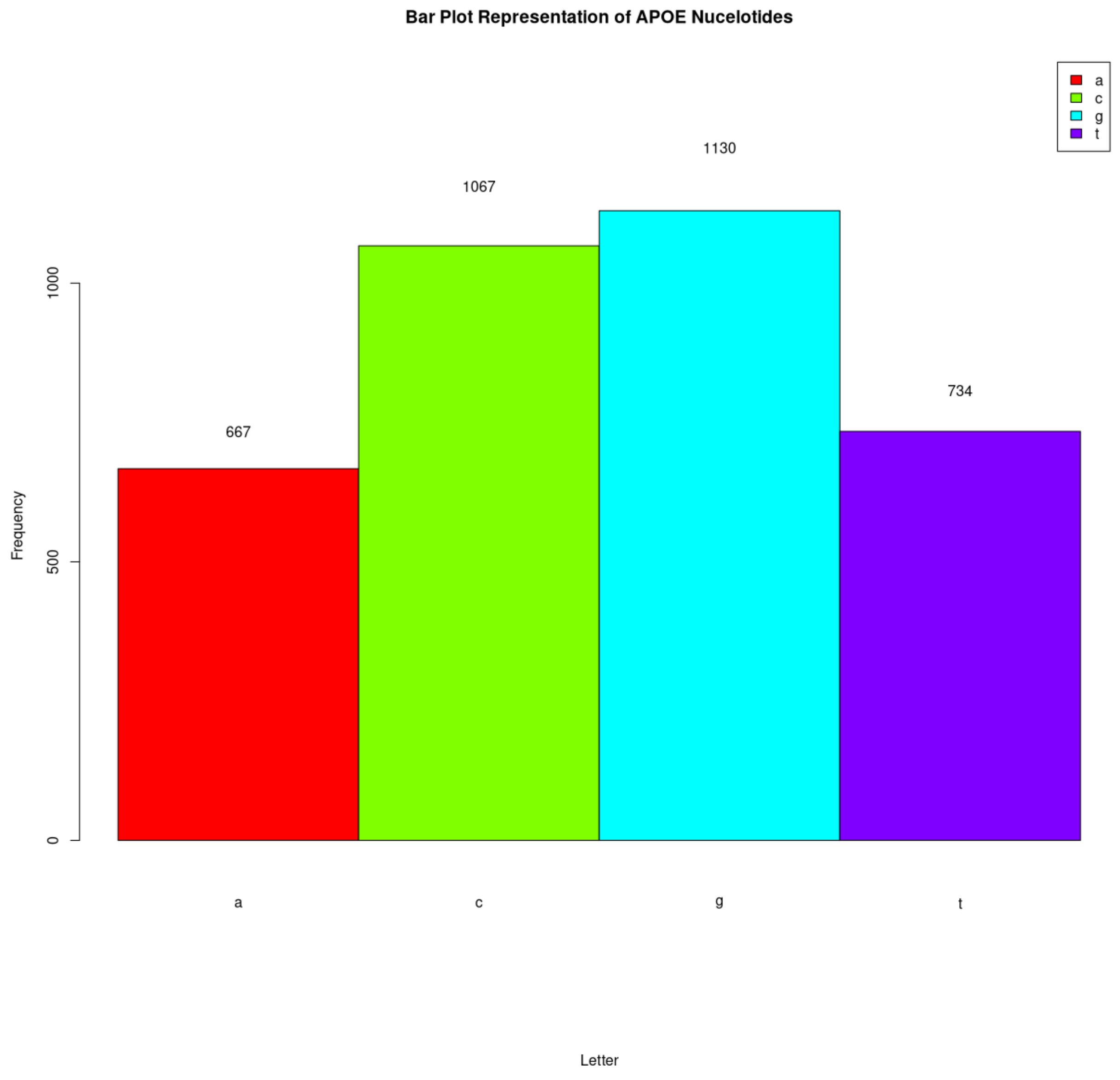
```
fasta2vec <- function(fasta.file){
  if (!require("seqinr")) install.packages("seqinr")
  library(seqinr)
  fasta <- read.fasta(file=fasta.file, as.string= TRUE)
  fasta.string <- fasta[[1]][1]
  fasta.list <- strsplit(fasta.string,"")
  fasta.vec <- unlist(fasta.list)
  }
```

Execute the function and test it like this

```
apoe.vec <- fasta2vec("apoe.fa")
```

3. How long is your sequence? 4. Show the first 20 nucleotides of your sequence? Hint: use the my.fasta.vec[ ] with : for array slicing (or apoe.vec). 5. How many of each nucleotide are there in the full sequence? 6. Create a barplot of the counts, including axes labels. 7. Calculate the probability of each nucleotide (divide by length).

```
3.) fasta.len <- length(fasta.vec)
        3598
4.) fasta.vec[1:20]
         [1] "c" "t" "a" "c" "t" "c" "a" "g" "c" "c" "c" "c" "a" "g" "c" "g" "g" "a" "g" "g"
5.) fasta.table <- table(fasta.vec)
          a   c      g     t
         667 1067 1130  734
6.) fasta.cols <- rainbow(nrow(fasta.table))

fasta.bp <- barplot(as.matrix(fasta.table), beside = TRUE, xlab = "Letter",
          ylab = "Frequency", ylim = c(-0.25*max(as.numeric(fasta.table)),
                              1.25*max(as.numeric(fasta.table))),
          main = "Bar Plot Representation of APOE Nucelotides",
          col = fasta.cols, legend = TRUE)

text(x = fasta.bp, y = 1.1*fasta.table, labels = as.numeric(fasta.table))
text(x = fasta.bp, y = -0.10*max(as.numeric(fasta.table)), labels = names(fasta.table))
```



```
7.) fasta.nuc_prob <- fasta.table/fasta.len
                a             c                    g            t
          0.1853808 0.2965536 0.3140634 0.2040022
```

D. GC Content. The GC content or G+C content of a sequence is the sum of G's and C's, and it is usually expressed as a percentage of the total number of nucleotides. Within eukaryotic species, GC content is often higher in coding regions. GC content also varies between species of bacteria (prokaryotes).

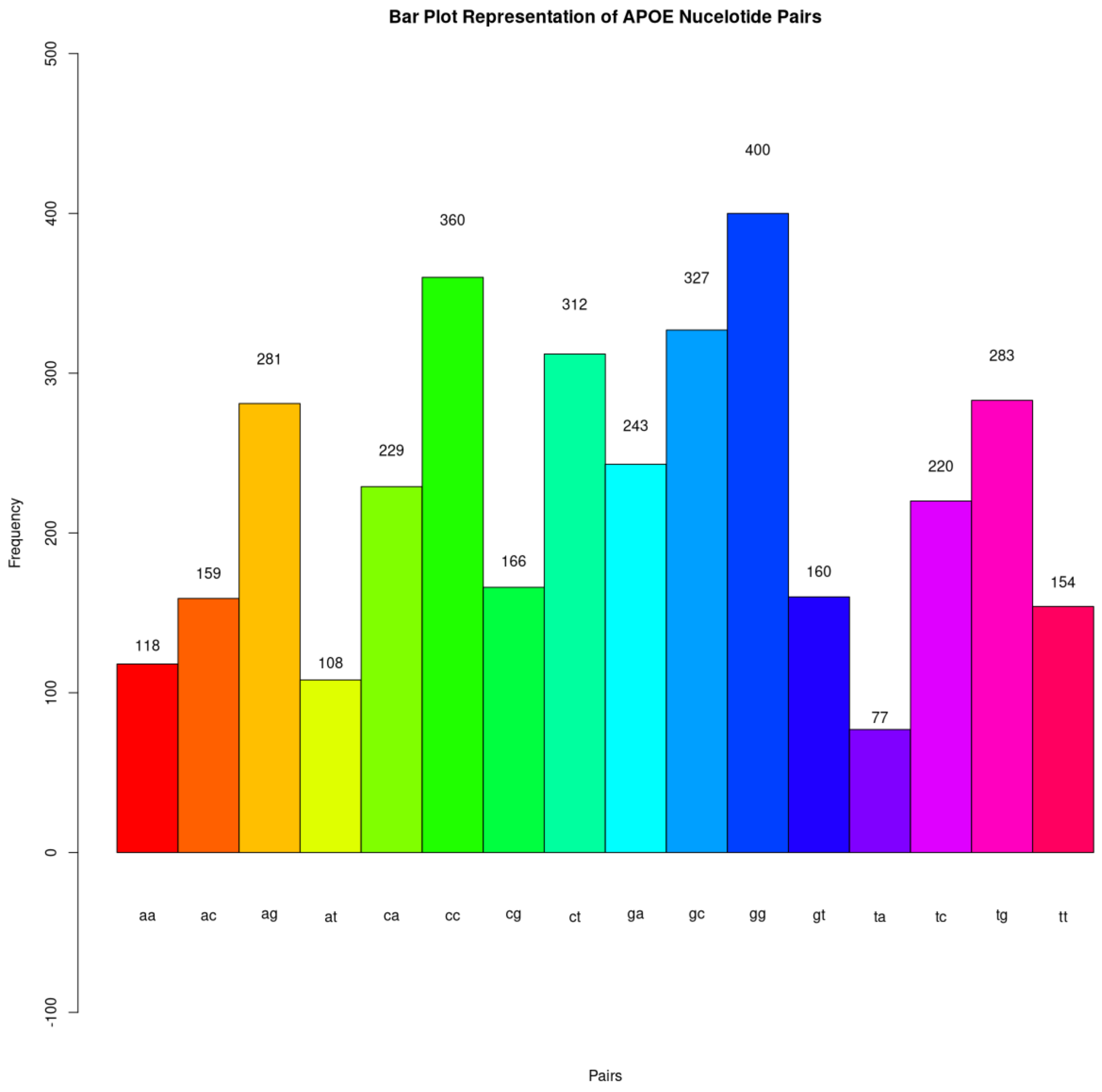
1. Add code to your R script to calculate the G+C content of my.fasta.vec (or apoe.vec). Hint: one way to get the number of g's is to use `ng <- sum(my.fasta.vec=="g")`. Verify your answer with seqinr's function `GC(my.fasta.vec)`.

```
fasta.gc <- (sum(fasta.vec=="g") + sum(fasta.vec=="c"))/fasta.len
# Verify:
fasta.gc == GC(fasta.vec)
TRUE

[1] 0.610617
```

As shown in the command below, use seqinr's `count` function with second argument set to 2 to count pairs of adjacent nucleotides (dinucleotides). It uses a sliding window to count the number of pairs, so there are n-1 pairs for a length-n sequence. 3. How many gc pairs are there? 4. Show a barplot of all dinucleotide counts (or probabilities).

```
pair.counts←seqinr::count(my.fasta.vec,2)
```

```
3.)   fasta.pairs <- seqinr::count(fasta.vec,2)
     fasta.pairs.gc <- fasta.pairs['gc']
      gc
      327

4.)fasta.pairs.cols <- rainbow(nrow(fasta.pairs))

fasta.pairs.bp <- barplot(as.matrix(fasta.pairs), beside = TRUE, xlab
                                                            = "Pairs",
                    ylab = "Frequency",
                    ylim = c(-0.25*max(as.numeric(fasta.pairs)),
                              1.25*max(as.numeric(fasta.pairs))),
                    main = "Bar Plot Representation of APOE Nucelotide
                           Pairs",
                    col = fasta.pairs.cols)

text(x = fasta.pairs.bp, y = 1.1*fasta.pairs, labels =
     as.numeric(fasta.pairs))
text(x = fasta.pairs.bp, y = -0.10*max(as.numeric(fasta.pairs)),
     labels = names(fasta.pairs))
```



E. Here are some links for one of the first sequences of the coronavirus.

A new coronavirus associated with human respiratory disease in China (paper for reference): https://www.ncbi.nlm.nih.gov/pubmed/32015508

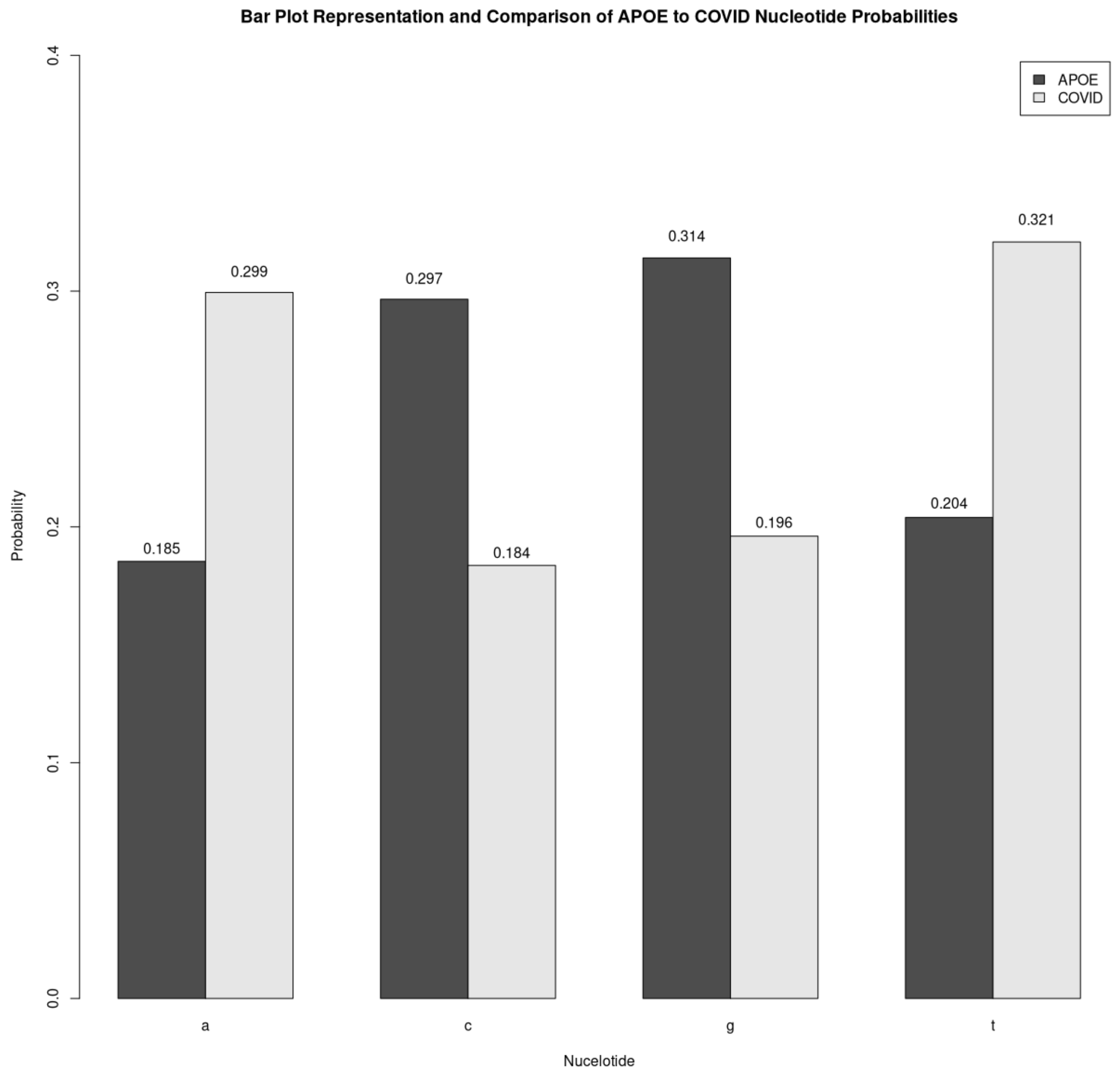
DNA/RNA:
https://www.ncbi.nlm.nih.gov/nuccore/MN908947.3?report=fasta

protein (**not used in lab**):
https://www.ncbi.nlm.nih.gov/protein/QHD43415.1?report=fasta

1. Download the DNA/RNA fasta file and determine the nucleotide frequencies. Comment on how the frequencies compare with the human APOE gene. 2. Determine the G+C content of this gene.

Note sometimes the **overall** GC content in a gene is important, but more often we are interested in **local enrichment** of GC in DNA called CpG islands **in different regions of the gene**.

```
1.) covid.vec  <- fasta2vec("covid.fasta")
     covid.table <- table(covid.vec)
    covid.nuc_prob <- covid.table / length(covid.vec)

        a                   c                g               t
0.2994348 0.1836605 0.1960673 0.3208374

compare.bind <- rbind(fasta.nuc_prob, covid.nuc_prob)

compare.bp <- barplot(as.matrix(compare.bind), beside = TRUE,
                          xlab = "Nucelotide",
                          ylab = "Probability",
                          ylim = c(-0.0,
                                   1.25*max(c(fasta.nuc_prob, covid.nuc_prob))),
                          main = "Bar Plot Representation and Comparison of APOE to COVID
                                   Nucleotide Probabilities",
                          legend = TRUE, legend.text = c("APOE", "COVID"))

ycords = 1.03*c(rbind(as.vector(fasta.nuc_prob), as.vector(covid.nuc_prob)))
tlabs = c(rbind(as.vector(fasta.nuc_prob), as.vector(covid.nuc_prob)))
text(x = compare.bp, y = ycords,
     labels = round(as.numeric(tlabs), digits = 3))
```



```
2.) covid.gc <- GC(covid.vec)
        [1] 0.3797278
```