```r
################ Assignment2 - MOHAMMAD HOSSEIN KAZAZI (ID:1257245) ----

############## Exploring ureA and vacA Genes Clustering Patterns in Helicobacter pylori

##### 1. INTRODUCTION ----

# Unsupervised Machine Learning, particularly in the context of clustering sequences, is a
powerful technique that has found extensive applications in genetics. It enables the discovery
of hidden patterns and structures within large genetic datasets, such as DNA sequences and
gene expressions. In this approach, the algorithm does not rely on predefined labels or
supervision but instead groups genetic sequences based on their inherent similarities.
Unsupervised techniques involve the examination of genetic data sets that lack predefined
labels with the aim of uncovering and leveraging valuable inherent data patterns without
necessarily providing interpretations. Typical unsupervised methods encompass clustering
approaches, dimensionality reduction algorithms, auto encoders, and generative adversarial
networks (GANs), making them invaluable tools for genetic research analysis and pathology.
(McAlpine et al., 2021)

# The evolutionary history and genomic diversity of microorganisms have become essential
fields of study in the era of genomics. Understanding the genetic variation and functional
diversity within a bacterial species can provide crucial insights into its adaptation, evolution,
and potential applications. In this context, Helicobacter pylori, which is known for its
intricate relationship with the human stomach and is associated with various gastrointestinal
diseases, offers a fascinating subject for investigation. (FitzGerald & Smith, 2021; Camilo et
al., 2017) vacA, an essential gene for vacuolating cyotoxin family protein and is considered a
major virulence factor of H. pylori. It plays several roles in the bacterium's ability to cause
disease. Moreover, ureA, an essential gene for surviving the acidic gastric environment, is the
gene we selected for this analysis. (Teimoori et al., 2021; Jeyamani et al., 2018)

# Objective: In this work, we assess whether two particular genes, vacA and ureA, are
appropriate markers for Helicobacter pylori within its strains, and we will find out which one
is a better marker in this case. We investigate their potential utility for classification.


##### 2. SETTING UP THE DIRECTORY AND LOADING THE NECESSARY
PACKAGES ----

# First, we should set up our directory to ensure effective file management and minimize
errors. This allows for streamlined access to files and simplifies the execution of technical
tasks that are integral to the overall workflow:

setwd("C:/Users/Home/Desktop/MBS/Fall
2023/BINF6210/ASSIGNMENTS/Assignment.2")
getwd()

# Secondly, we will start to install and run the packages that we need:

install.packages("rentrez")
library(rentrez)
install.packages("seqinr")
```

```
library(seqinr)
install.packages("ggplot2")
library(ggplot2)
BiocManager::install("Biostrings")
library(Biostrings)
install.packages("tidyverse")
library(tidyverse)
install.packages("stringr")
library(stringr)
install.packages("stringi")
library(stringi)
install.packages("ape")
library(ape)
install.packages("RSQLite")
install.packages("parallel")
library(RSQLite)
library(muscle)
library(DECIPHER)
install.packages("kmer")
library(kmer)
install.packages("cluster")
library(cluster)

##### 3: MAIN CODING SCRIPT (PART 1) ----

# Let's see a list of available Entrez databases by using the entrez_dbs() function:
entrez_dbs()

# So, as we want to focus on the genes, now we have to figure out what fields are available
through the "gene" database. In order to achieve it, we use the entrez_db_searchable()
function:
entrez_db_searchable(db = "gene")

# Therefore, we can see that there are 36 searchable fields available under the "gene"
database two of which are ORGN, stands for the organism, and Gene, stands for symbol or
symbols for the gene.

# Next, here we are attempting to search a given NCBI database with a particular query. In
this way, we use genes as our database for records that match the "Helicobacter pylori,"
which is our investigated specimen.
Hp_search <- entrez_search(db = "gene" , term = "Helicobacter pylori [ORGN]")

# checking the class of the result:
class(Hp_search)

Hp_search

# Here, we are going through some details about the IDs of our entrez_search result:
Hp_search$ids
class(Hp_search$ids)
```

```
length(Hp_search$ids)

# Checking the hits in our result:
Hp_search$count

# So, as you can see, our result has only 20 IDs in 91172 results; however, IDs should be
unique and equal to the number of hits, which is 91172. This is happening due to the default
setting of the entrez_search, in which the retmax is equal to 20.

# Therefore, we will define a new list that is the same as the previous one, but the retmax is
equal to the all amount of hits:
Hp_search <- entrez_search(db = "gene" , term = "Helicobacter pylori[ORGN]" , retmax =
Hp_search$count)

Hp_search
length(unique(Hp_search$ids))

# Now, we have 91172 hits with 91172 IDs from NCBI on the term "Helicobacter
pylori."Thus, the list is now as complete as possible.

# The next step of our project is to find the sample size of our interested genes, vacA and
ureA, and check whether they have similar amounts of hits or if one gene has a lot more data.


# First, the sample size of vacA:
Hp_vacA.gene <- entrez_search(db = "gene" , term = "Helicobacter pylori[ORGN] AND
vacA[Gene]"  , retmax = Hp_search$count)

Hp_vacA.gene

# Secondly, the sample size of ureA:
Hp_ureA.gene <- entrez_search(db = "gene" , term = "Helicobacter pylori[ORGN] AND
ureA[Gene]"  , retmax = Hp_search$count)

Hp_ureA.gene

# As the result of ureA is 22 hits and vacA is 18 hits, we can conclude that there is not such a
big difference between these two sample sizes.

# Now we want to check out the fields that are available through the "nucleotide" database, in
order to know how we can narrow our data:
entrez_db_searchable(db = "nucleotide")

#We would also check the result of these two genes based on the "nucleotide" database:

Hp_vacA.nucleotide <- entrez_search(db = "nucleotide" , term = "Helicobacter
pylori[ORGN] AND vacA[Gene]"  , retmax = Hp_search$count)

Hp_ureA.nucleotide <- entrez_search(db = "nucleotide" , term = "Helicobacter
pylori[ORGN] AND ureA[Gene]"  , retmax = Hp_search$count)
```

```
Hp_vacA.nucleotide
Hp_ureA.nucleotide

# Therefore, we can see that under the "nucleotide" database, these two genes have approximately the same number of hits, as vacA has 3667 and ureA has 3817.

# Also, as the hits are too big, we narrow our data set with retmax = 300 in order to have a more certain result.

# According to the NCBI, the sequence length of the ureA is 717nt and for vacA is 3867nt; therefore, we are going to obtain data for each gene from NCBI with a sequence length of 150-750 for ureA and 1000-3000 for vacA:
Hp_vacA.nucleotide2 <- entrez_search(db = "nucleotide" , term = "Helicobacter pylori[ORGN] AND vacA[Gene] AND 1000:3000[SLEN]"  , retmax = 300)

Hp_ureA.nucleotide2 <- entrez_search(db = "nucleotide" , term = "Helicobacter pylori[ORGN] AND ureA[Gene] AND 150:750[SLEN]"  , retmax = 300)


# Let's get a summary of our data in order to check everything is associated with the genes and Helicobacter pylori.
  #ureA:
  summary.ureA <- entrez_summary(db = "nucleotide", id = Hp_ureA.nucleotide2$ids)
  summary.ureA
    # Going through the particular ID:
    summary.ureA$`2592186061`
    summary.ureA$`2592186061`$organism
    summary.ureA$`2592186061`$statistics
    # Extracting the sequence lengths:
    extract_from_esummary(summary.ureA, "slen")
    sort(table(extract_from_esummary(summary.ureA, "slen")), decreasing = TRUE)

    # Let's visualize the distribution of the sequence lengths in our data set through a plot:
        # Create a data frame for the histogram
        ureA_hist_data <- data.frame(x = extract_from_esummary(summary.ureA, "slen"))

        # Create a histogram with custom units and multiples of 20
        ureA.custom_breaks <- seq(0, max(ureA_hist_data$x), by = 20)
        ureA.custom_ylim <- c(0, 75)
        ggplot(ureA_hist_data, aes(x)) +
          geom_histogram(binwidth = 10, fill = "blue") +
          scale_x_continuous(breaks = ureA.custom_breaks) +
          scale_y_continuous(limits = ureA.custom_ylim) +
          xlab("Sequence length (nt)") +
          ylab("Frequency") +
          ggtitle("Frequency of ureA Sequence Lengths") +
          theme_minimal() +
          theme(plot.background = element_rect(fill = "black")) +
          theme(plot.title = element_text(color = "red"), axis.title = element_text(color =
"red"),axis.text = element_text(color = "white"))


  #vacA:
  summary.vacA <- entrez_summary(db = "nucleotide", id = Hp_vacA.nucleotide2$ids)
  summary.vacA
    # Going through the particular ID:
    summary.vacA$`1778483824`
    summary.vacA$`1778483824`$organism
    summary.vacA$`1778483824`$statistics
    # Extracting the sequence lengths:
    extract_from_esummary(summary.vacA, "slen")
    sort(table(extract_from_esummary(summary.vacA, "slen")), decreasing = TRUE)

    # A plot for frequency of vacA sequence length:
    # Create a data frame for the histogram
    vacA_hist_data <- data.frame(x = extract_from_esummary(summary.vacA, "slen"))

    # Create a histogram with custom units and multiples of 200
    vacA.custom_breaks <- seq(1000, max(vacA_hist_data$x), by = 200)
    vacA.custom_ylim <- c(0, 30)
    ggplot(vacA_hist_data, aes(x)) +
      geom_histogram(binwidth = 25, fill = "green") +
      scale_x_continuous(breaks = vacA.custom_breaks) +
      scale_y_continuous(limits = vacA.custom_ylim) +
      xlab("Sequence length(nt)") +
      ylab("Frequency") +
      ggtitle("Frequency of vacA Sequence Lengths") +
      theme_minimal() +
      theme(plot.background = element_rect(fill = "black")) +
      theme(plot.title = element_text(color = "red"), axis.title = element_text(color = "red"),
axis.text = element_text(color = "white"))
```

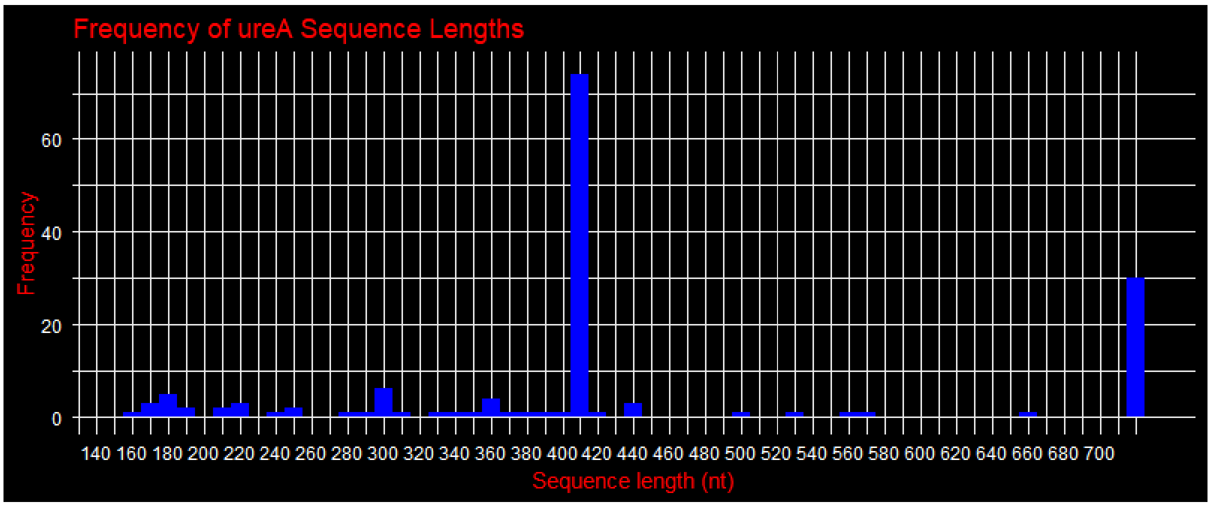


Figure 1: Frequency of ureA Sequence Lengths

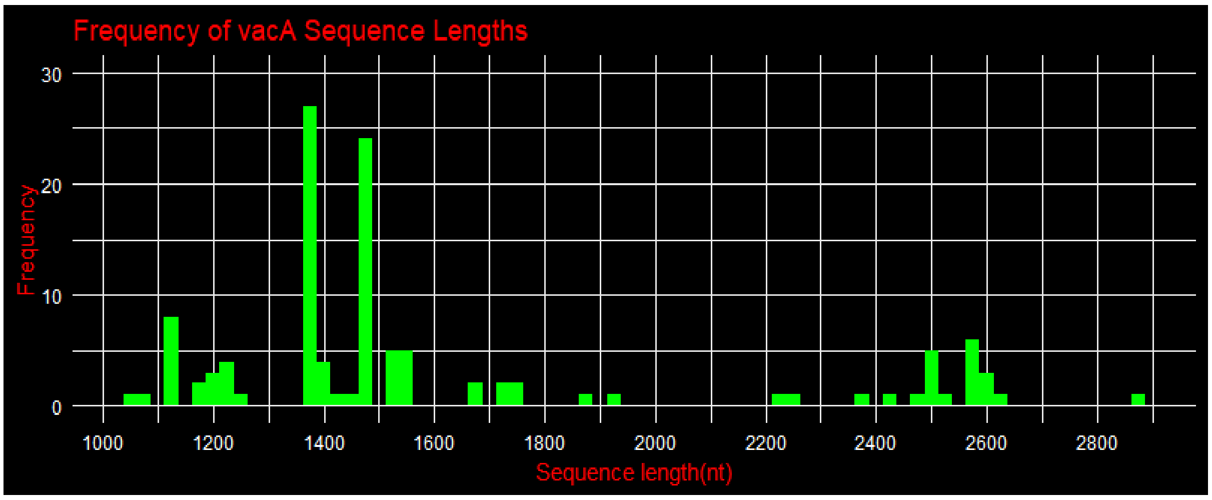


Figure 2: Frequency of vacA Sequence Lengths

# We can assume from the plots that most of the data from NCBI data sets for ureA in *Helicobacter pylori* have the sequence length of 412 and 717, and for vacA have the sequence length of 1381 and 1477.

# Now, we are going to obtain sequence data in FASTA format:

```
vacA.FASTA <- entrez_fetch(db = "nucleotide", id = Hp_vacA.nucleotide2$ids, rettype =
"fasta")
vacA.FASTA

ureA.FASTA <- entrez_fetch(db = "nucleotide", id = Hp_ureA.nucleotide2$ids, rettype =
"fasta")
ureA.FASTA


# writing them in the directory in order to have a backup of our data:
write(vacA.FASTA, "vacA.FASTA", sep = "\n")

write(ureA.FASTA, "ureA.FASTA", sep = "\n")

# setting them as a DNA String:
vacA.StringSet <- readDNAStringSet("vacA.FASTA")
vacA.StringSet
BrowseSeqs(vacA.StringSet)

ureA.StringSet <- readDNAStringSet("ureA.FASTA")
ureA.StringSet
BrowseSeqs(ureA.StringSet)

# Converting them to a data frame:
dfvacA <- data.frame(vacA_Title = names(vacA.StringSet), vacA_Sequence =
paste(vacA.StringSet))
class(dfvacA)
```

```
View(dfvacA)

dfureA <- data.frame(ureA_Title = names(ureA.StringSet), ureA_Sequence =
paste(ureA.StringSet))
class(dfureA)
View(dfureA)

# Let's organize our data frame, as the names do not look very good.
   # Making a new column:
   dfvacA$Bacterium_Name <- word(dfvacA$vacA_Title, 2L, 3L)
   dfureA$Bacterium_Name <- word(dfureA$ureA_Title, 2L, 3L)
   # Reorganize the data frame:
   dfvacA <- dfvacA[, c("vacA_Title", "Bacterium_Name", "vacA_Sequence")]
   dfureA <- dfureA[, c("ureA_Title", "Bacterium_Name", "ureA_Sequence")]

   class(dfvacA)
   class(dfureA)
   dim(dfvacA)
   dim(dfureA)

   View(dfvacA)
   View(dfureA)
# Checking a summary of the sequence length of the both genes:
summary(str_count(dfvacA$vacA_Sequence))
summary(str_count(dfureA$ureA_Sequence))

# So, the next step is filtering the data frame of each gene; so, we set the missing data to 1
percent and the length variability to 100.
  #vacA
  dfvacA2 <- dfvacA %>%
   select(vacA_Title, Bacterium_Name, vacA_Sequence) %>%
   filter(!is.na(vacA_Sequence)) %>%
   mutate(vacA_Sequence2 = str_remove_all(vacA_Sequence, "^N+|N+$|-")) %>%
   filter(str_count(vacA_Sequence2, "N") <= (0.01 * str_count(vacA_Sequence))) %>%
   filter(str_count(vacA_Sequence2) >= median(str_count(vacA_Sequence2)) - 100 &
str_count(vacA_Sequence2) <= median(str_count(vacA_Sequence2)) + 100)

  view(dfvacA2)
  dim(dfvacA2)
  sum(is.na(dfvacA2$vacA_Sequence2))
  sum(str_count(dfvacA2$vacA_Sequence2, "-"))
  summary(str_count(dfvacA2$vacA_Sequence2))

  #ureA:
  dfureA2 <- dfureA %>%
   select(ureA_Title, Bacterium_Name, ureA_Sequence) %>%
   filter(!is.na(ureA_Sequence)) %>%
   mutate(ureA_Sequence2 = str_remove_all(ureA_Sequence, "^N+|N+$|-")) %>%
   filter(str_count(ureA_Sequence2, "N") <= (0.01 * str_count(ureA_Sequence))) %>%
    filter(str_count(ureA_Sequence2) >= median(str_count(ureA_Sequence2)) - 100 &
str_count(ureA_Sequence2) <= median(str_count(ureA_Sequence2)) + 100)

  view(dfureA2)
  dim(dfureA2)
  sum(is.na(dfureA2$ureA_Sequence2))
  sum(str_count(dfureA2$ureA_Sequence2, "-"))
  summary(str_count(dfureA2$ureA_Sequence2))


##### 4: MAIN CODING SCRIPT (PART 2) ----

# What we have done so far: we have obtained our data from NCBI for two genes, ureA and
vacA, of Helicobacter pylori, and narrowed our data by their sequence length. Then we get
the FASTA format of our data sequence and convert them to a data frame, and we have
filtered the data frame of each gene in order to remove the gaps of the sequences.

# After we filtered the data frames, the next step is ALIGNMENT.
  #vacA:
  dfvacA2 <- as.data.frame(dfvacA2)
  dfvacA2$vacA_Sequence2 <- DNAStringSet(dfvacA2$vacA_Sequence2)
  names(dfvacA2$vacA_Sequence2) <- dfvacA2$vacA_Title

  dfvacA.alignment <- DNAStringSet(muscle::muscle(dfvacA2$vacA_Sequence2))

  BrowseSeqs(dfvacA.alignment)

  #ureA:
  dfureA2 <- as.data.frame(dfureA2)
  dfureA2$ureA_Sequence2 <- DNAStringSet(dfureA2$ureA_Sequence2)
  names(dfureA2$ureA_Sequence2) <- dfureA2$ureA_Title

  dfureA.alignment <- DNAStringSet(muscle::muscle(dfureA2$ureA_Sequence2))

  BrowseSeqs(dfureA.alignment)

# The dashes "-" indicate gaps in the alignment. Gaps are introduced to account for insertions
or deletions in the sequences being aligned. Each dash represents a position where there is no
corresponding nucleotide in the sequence.

# For calculating the distance matrix for each gene, we use "TN93" as a model. The reason
for choosing these models is that we have different sequences in our data for each gene, and
also this model is appropriate for distinguishing between transversions and the two possible
types of transitions. The option pairwise.deletion = TRUE means that if a pairwise sequence
comparison involves gaps or missing data, those positions are excluded from the calculation.
If there are too many gaps or missing data in a particular pairwise comparison, it can lead to
an undefined or "NaN" result.
  #vacA:
  vacA.DNAbin <- as.DNAbin(dfvacA.alignment)
  class(vacA.DNAbin)
  vacA.TN93.distanceMatrix <- dist.dna(vacA.DNAbin, model = "TN93", as.matrix = TRUE,
pairwise.deletion = TRUE)
  view(vacA.TN93.distanceMatrix)
  # Checking the "NA" values:
  table(is.na(vacA.TN93.distanceMatrix))

  #ureA:
  ureA.DNAbin <- as.DNAbin(dfureA.alignment)
  class(ureA.DNAbin)
  ureA.TN93.distanceMatrix <- dist.dna(ureA.DNAbin, model = "TN93", as.matrix = TRUE,
pairwise.deletion = TRUE)
  view(ureA.TN93.distanceMatrix)
  # Checking the "NA" values:
  table(is.na(ureA.TN93.distanceMatrix))
  # Identify rows and columns without NA values
  complete_cases <- complete.cases(ureA.TN93.distanceMatrix)
  # Create a new distance matrix without NA values
  ureA.TN93.distanceMatrix <- ureA.TN93.distanceMatrix[complete_cases, complete_cases]
  table(is.na(ureA.TN93.distanceMatrix))
```

# K-means clustering is a powerful technique for categorizing gene sequences based on shared characteristics. By iteratively assigning gene sequences to clusters and redefining cluster centroids based on sequence similarity, K-means helps uncover hidden structures within the data. It's advantageous in genetic research, revealing patterns and relationships in the data. We can apply K-means to group H. pylori strains by their ureA and vacA sequences, aiding in the identification of distinct strain clusters.
# So, for doing a K-means cluster we should first figure out how many centers(clusters) are suitable for our data. In order to achieve this goal, we use the "Elbow Method" which is based on calculating the "Sum of Squared Distances (SSD)" for each cluster. By creating a plot based on SSD for number of clusters. we can see an elbow which shows where the reduction in SSD starts to slow down significantly and the K value at the elbow is a good choice.

```
  ssd <- vector("numeric", length = 7)
  #vacA:
  # Making a loop through different values of K (1 to 7)
  for (k in 1:7) {
    vacA.kmeans <- kmeans(vacA.TN93.distanceMatrix, centers = k, nstart = 25)
    ssd[k] <- vacA.kmeans$tot.withinss}
  plot(1:7, ssd, type = "b", xlab = "Number of Clusters (K)", ylab = "Sum of Squared
Distances (SSD)", main = "Elbow Method for vacA Number of Clusters")

  #ureA:
  for (k in 1:7) {
    ureA.kmeans <- kmeans(ureA.TN93.distanceMatrix, centers = k, nstart = 25)
    ssd[k] <- ureA.kmeans$tot.withinss}
  plot(1:7, ssd, type = "b", xlab = "Number of Clusters (K)", ylab = "Sum of Squared
Distances (SSD)", main = "Elbow Method for ureA Number of Clusters")
```

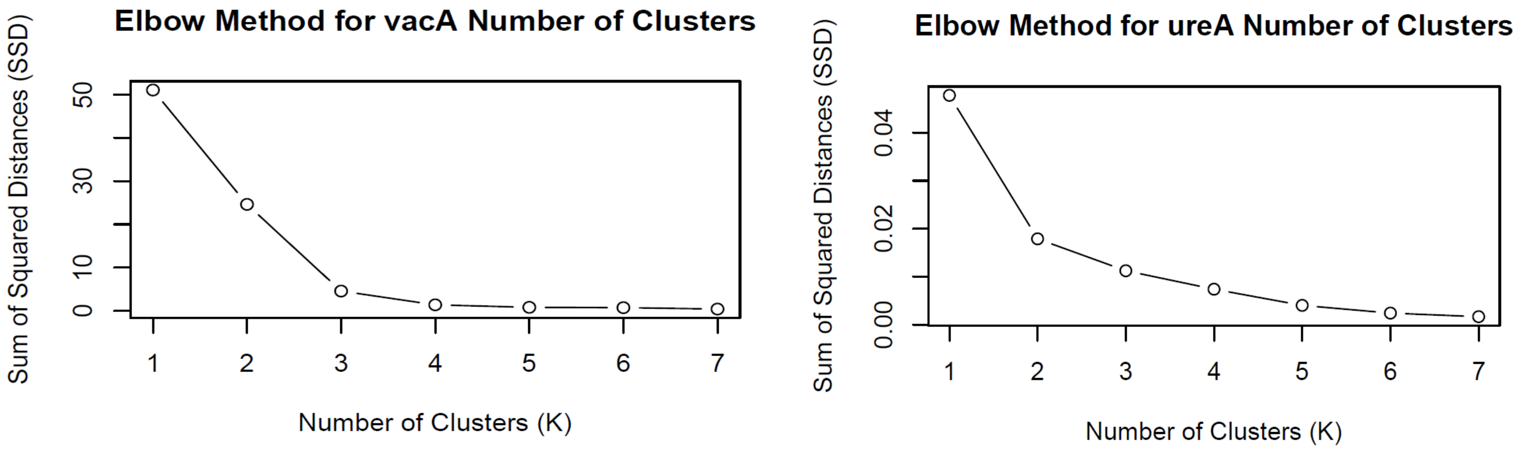


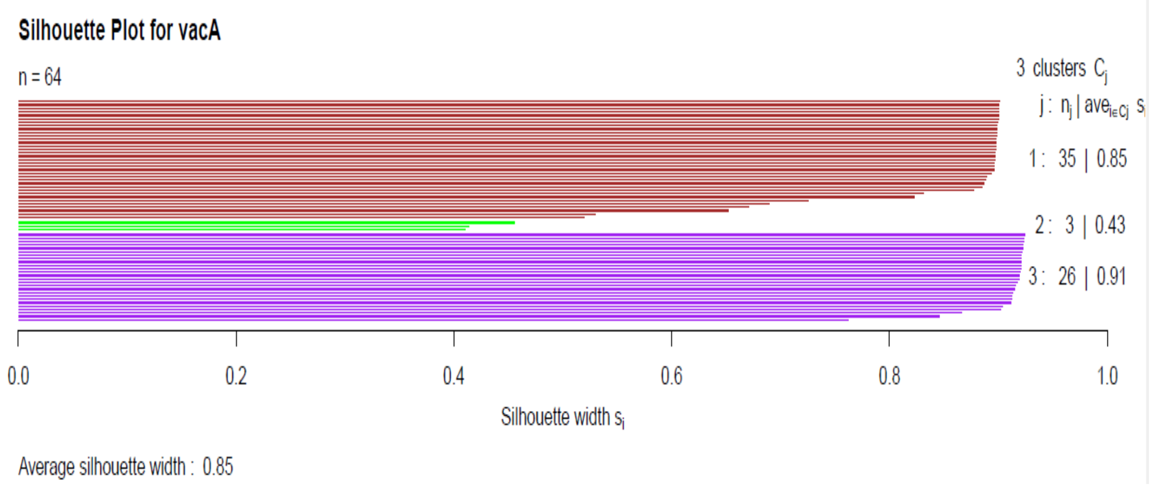
Figure 3: Elbow Method for vacA and ureA in order to find the number of clusters.

```
# Therefore, as we can see based on the "Elbow Method", the best number of clusters for
vacA is 3, while for ureA, this number is 2.
# Now, it is time to do our K-means clustering for each of the vacA and ureA genes.
  #vacA:
  vacA.KmeansClustering <- kmeans(vacA.TN93.distanceMatrix, centers = 3 )

  #ureA:
  ureA.KmeansClustering <- kmeans(ureA.TN93.distanceMatrix, centers = 2 )

# In order to evaluate our clustering, we are calculating the Silhouette Score for each gene.
The Silhouette Score is used metric in the field of cluster analysis and unsupervised machine
learning. Assessing the effectiveness of clustering algorithms can be done using the
Silhouette Score metric. This metric takes into account both the intra-cluster distance, which
is the closeness of data points within individual clusters, and the inter-cluster distance, which
is the distance between different clusters. The Silhouette Score calculates an overall score
that reflects the performance of the clustering algorithm. This score ranges from −1 to +1. A
high value indicates that an object is well-matched to its own cluster but poorly matched to
neighbouring clusters.
  #vacA:
  vacA.SilhouetteScore <- silhouette(vacA.KmeansClustering$cluster,
dist(vacA.TN93.distanceMatrix))
  view(vacA.SilhouetteScore)
  summary(vacA.SilhouetteScore)
  vacA.cluster_colors <- c("brown", "green","purple")
  plot(vacA.SilhouetteScore, main = "Silhouette Plot for vacA", col = vacA.cluster_colors)

  #ureA:
  ureA.SilhouetteScore <- silhouette(ureA.KmeansClustering$cluster,
dist(ureA.TN93.distanceMatrix))
  view(ureA.SilhouetteScore)
  summary(ureA.SilhouetteScore)
  ureA.cluster_colors <- c("red", "blue")
  plot(ureA.SilhouetteScore, main = "Silhouette Plot for ureA", col = ureA.cluster_colors)
```



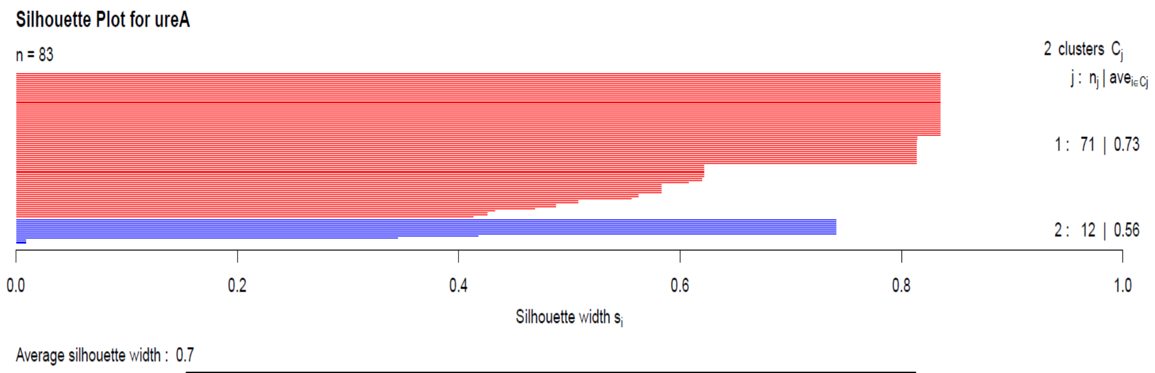


Figure 4: The Silhouette Score plot for the clustering of vacA and ureA

# Thus, we can see that based on the silhouette scores, vacA appears to have more well-defined clusters with higher silhouette scores. However, ureA has lower silhouette scores, suggesting that the clusters are less distinct and might have some degree of overlap.

##### 5. CONCLUSION AND DISCUSSION ----

# All in all, in this effort, the genetic sequences from two genes—vacA and ureA— have been analyzed in relation to *Helicobacter pylori*. We involved a number of processes, including K-means clustering, distance matrix computation, sequence alignment, data preprocessing, and data retrieval. We tried an unsupervised analysis of *Helicobacter pylori* genetic sequences from the ureA and vacA genes. The main conclusions show that vacA seems to have separate clusters that are well-defined, as indicated by a higher Silhouette Score. Conversely, ureA has lower Silhouette Scores, indicating that there may be considerable overlap between its clusters, making it more difficult to classify. Therefore, the aforementioned results can provide valuable insights into the genetic variation and diversity within *Helicobacter pylori* strains.

# In the case of future steps, it is necessary to involve experts in the field to provide a biological perspective on the identified clusters, particularly concerning their implications for *Helicobacter pylori*'s virulence and survival strategies. Additionally, we should utilize statistical tests to assess the importance of these clusters in relation to specific traits or features of *Helicobacter pylori* strains. Furthermore, we can also explore alternative machine learning methods, including hierarchical clustering and dimensionality reduction techniques, in order to achieve a more in-depth comprehension of the genetic diversity within the dataset.

##### 6. ACKNOWLEDGEMENT ----

#To complete this project, I relied on R scripts authored by Dr. Sarah Adamowicz and Jessica Castellanos-Labarcena, the swirl tutorial R package and did extensive research on Google. Also, I used the information from the following links:
#1. https://lgatto.github.io/IntroMachineLearningWithR/unsupervised-learning.html
#2. https://cran.r-project.org/web/packages/kmer/kmer.pdf
#3. https://cran.r-project.org/web/packages/cluster/cluster.pdf

##### 7. REFERENCES ----

# 1. McAlpine, E. D., Michelow, P., &amp; Celik, T. (2021). The utility of unsupervised machine learning in anatomic pathology. American Journal of Clinical Pathology, 157(1), 5–14. https://doi.org/10.1093/ajcp/aqab085

# 2. FitzGerald, R., &amp; Smith, S. M. (2021). An overview of helicobacter pylori infection. Methods in Molecular Biology, 1–14. https://doi.org/10.1007/978-1-0716-1302-3_1

# 3. Camilo, V., Sugiyama, T., &amp; Touati, E. (2017). Pathogenesis of Helicobacter pylori infection. Helicobacter, 22(S1). https://doi.org/10.1111/hel.12405

# 4. Teimoori, F., Hajilooei, M., Abdolsamadi, H., Eslami, K., Moghimbeigi, A., &amp; Ahmadi- Motamayel, F. (2021). Evaluation of salivary helicobacter pylori, calcium, urea, ph and flow rate in hemodialysis patients. Infectious Disorders - Drug Targets, 21(7). https://doi.org/10.2174/1871526520666201218154530

# 5. Jeyamani, L., Jayarajan, J., Leelakrishnan, V., &amp; Swaminathan, M. (2018). Caga and Vaca Genes of helicobacter pylori and their clinical relevance. Indian Journal of Pathology and Microbiology, 61(1), 66. https://doi.org/10.4103/ijpm.ijpm_234_17