BINF6210 - Assignment 5

Thomas Papp-Simon

2022-12-16

Introduction

Unsupervised machine learning is a technique commonly used to discover clustering patterns in large data sets (Gentleman & Carey, 2008). It is commonly compared to supervised machine learning, where a classifier uses training data in order to perform accurate predictions on a given set of data. There are many algorithms and methods to choose from when it comes to unsupervised learning, and choosing the best one can become a cumbersome task, especially through trial and error. In the context of biology, there are many ways to apply unsupervised learning to achieve different goals. For instance, one could look at the clustering patterns of certain genes involved in a certain biological function to understand how their expression levels might vary in the context of that process.

As mentioned before, there are several options to choose from when it comes to choosing which clustering method to apply to an unsupervised machine learning analysis (Gentleman & Carey, 2008). Different clustering methods such as hierarchical clustering as well as k-means clustering can be considered when conducting this type of analysis. Hierarchical clustering involves the generation of a hierarchy of clusters, where each cluster becomes a subset of a larger cluster (Kaisers et al., 2018). K-means clustering is known as a partitioning clustering method that uses centroids, which means it will partition data points into a number of groups until it creates the pre-determined number of clusters based on distances (Arora et al., 2016). There are also variations of these types of clustering methods, such as the partitioning around medoids (PAM) algorithm which is a type of k-means clustering (meaning it requires a set number of clusters) that clusters around medoids instead of centroids and is known to be more resistant to noise than the k-means clustering method (Gentleman & Carey, 2008).

The specific objective of this study is to compare the clustering patterns of the COI and ND1 genes in the family Microchiroptera using unsupervised clustering. In particular, I will be clustering the genes using k-mer frequencies as the primary sequence feature for this analysis. I will also be comparing the degree of clustering between two k-mer lengths for each gene (k-mers of 4 and 6). Do COI and ND1 exhibit different clustering patterns within the Microchiroptera family, and does clustering strength between these genes increase with increasing k-mer length?

Description of Data Set

For my analysis, I've chosen to analyze two genes from the Microchiroptera family: COI and ND1. The main reason I chose these genes is because I knew that there was going to be a lot of data present for these particular genes, since they are commonly used in these types of analyses. I needed to pick genes with a good amount of data since I wanted to use k-mer frequencies as my sequence feature of choice to use for clustering, which requires more data than if I were to use alignments. This sequence data was obtained from Nucleotide database from NCBI using the entrez package in R. Since there was a large amount of data to fetch from the database, I used the functions from the EntrezFunctions script provided by our instructors to obtain and import the data into RStudio. I separated the data from each gene into their own data frames, which contained 18202 COI sequences and 1973 ND1 sequences respectively. My first step was to filter for any duplicate sequences, which removed nearly 50% of the COI sequences. My goal was to equalize the number

of sequences before starting my analysis, which I did by randomly sampling from the 9685 remaining COI sequences. Finally I also removed sequences that I thought were low-quality such as sequences with very long lengths or with a high percentage of N's within them. The final number of sequences was 1411 COI sequences and 1256 ND1 sequences.

Source of dataset citation: Nucleotide. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; 2004 – 2022-12-15. Available from: https://www.ncbi.nlm.nih.gov/nucleotide/

```
#entrez_db_searchable(db = "nuccore")

# Searching the NCBI database for sequence data for COI and ND1 genes.
COI_search <- entrez_search(db = "nuccore", term = "Microchiroptera[ORGN] AND COI[Gene] AND 500:700[SLE
COI_search
```

```
## Entrez search result with 18202 hits (object contains 5000 IDs and no web_history object)
##  Search term (as translated):  "Microchiroptera"[Organism] AND COI[Gene] AND 0000 ...
```

```
ND1_search <- entrez_search(db = "nuccore", term = "Microchiroptera[ORGN] AND ND1[Gene] AND 600:1400[SL
ND1_search
```

```
## Entrez search result with 1835 hits (object contains 1835 IDs and no web_history object)
##  Search term (as translated):  "Microchiroptera"[Organism] AND ND1[Gene] AND 0000 ...
```

```
# Sourced the FetchFastaFiles and MergeFastaFiles functions from the Entrez_Functions.R script to obtai
source("Entrez_Functions.R")
#FetchFastaFiles(searchTerm = "Microchiroptera[ORGN] AND COI[Gene] AND 500:700[SLEN]", seqsPerFile = 50
#FetchFastaFiles(searchTerm = "Microchiroptera[ORGN] AND ND1[Gene] AND 600:1400[SLEN]", seqsPerFile = 5

dfCOI <- MergeFastaFiles(filePattern = "COI*")
dfND1 <- MergeFastaFiles(filePattern = "ND1*")

#view(dfCOI)
#view(dfND1)

# Let's check how many sequences we have for each gene.
nrow(dfCOI) # 18202 sequences for COI.
```

```
## [1] 18202
```

```
nrow(dfND1) # 1973 sequences for ND1.
```

```
## [1] 2081
```

```
# Obviously there's a huge difference in the number of samples between each gene so I need adjust that.


# Setting the seed to get reproducible results.
set.seed(999)


dfCOI$Species_Name <- word(dfCOI$Title, 2L, 3L)
length(unique(dfCOI$Species_Name)) # data from 582 unique species
```

```
## [1] 582
```

```
dfND1$Species_Name <- word(dfND1$Title, 2L, 3L)
length(unique(dfND1$Species_Name)) # data from 240 unique species
```

```
## [1] 240
```

```
# Removal of identical sequences before my analysis.
# For this analysis, I've decided to remove identical sequences from my data. I did this because I want

length(unique(dfCOI$Sequence)) # 9685 unique sequences
```

```
## [1] 9685
```

```
length(unique(dfND1$Sequence)) # 1469 unique sequences
```

```
## [1] 1499
```

```
dfCOI <- dfCOI %>%
  distinct(Sequence, .keep_all = TRUE)

dfND1 <- dfND1 %>%
  distinct(Sequence, .keep_all = TRUE)

# Confirming that it worked
identical(length(dfCOI$Sequence), length(unique(dfCOI$Sequence))) # TRUE
```

```
## [1] TRUE
```

```
identical(length(dfND1$Sequence), length(unique(dfND1$Sequence))) # TRUE
```

```
## [1] TRUE
```

```
# Cleaning up the sequence data for both genes.
# In terms of sample size, I wanted a similar sample size for both COI and ND1 data sets. Therefore, I
# Also filtering out any missing sequences, and removing all N's from the beginning and ends of our seq
dfCOI_cleaned <- dfCOI %>%
  sample_n(length(dfND1$Sequence)) %>%
  filter(!is.na(Sequence)) %>%
  mutate(Nucleotides = str_remove_all(Sequence, "^N+|N+$|-")) %>%
  filter(str_count(Nucleotides, "N") <= (0.001* str_count(Nucleotides))) %>%
  mutate(Seq_Len = str_count(Nucleotides)) %>%
  mutate(gene = "COI")

dfND1_cleaned <- dfND1 %>%
  filter(!is.na(Sequence)) %>%
  mutate(Nucleotides = str_remove_all(Sequence, "^N+|N+$|-")) %>%
  filter(str_count(Nucleotides, "N") <= (0.001* str_count(Nucleotides))) %>%
  mutate(Seq_Len = str_count(Nucleotides)) %>%
  mutate(gene = "ND1")

# Summary of filtered data
nrow(dfCOI_cleaned) # 1408 sequences after filtering
```

```
## [1] 1438
```

```
nrow(dfND1_cleaned) # 1267 sequences after filtering
```

```
## [1] 1297
```

```
# Information on sequence length for both genes
summary(str_count(dfCOI_cleaned$Nucleotides)) # 639.6 is the mean length for COI
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   501.0   643.0   657.0   639.7   657.0   697.0
```

```
summary(str_count(dfND1_cleaned$Nucleotides)) # 818.9 is the mean length for ND1
```

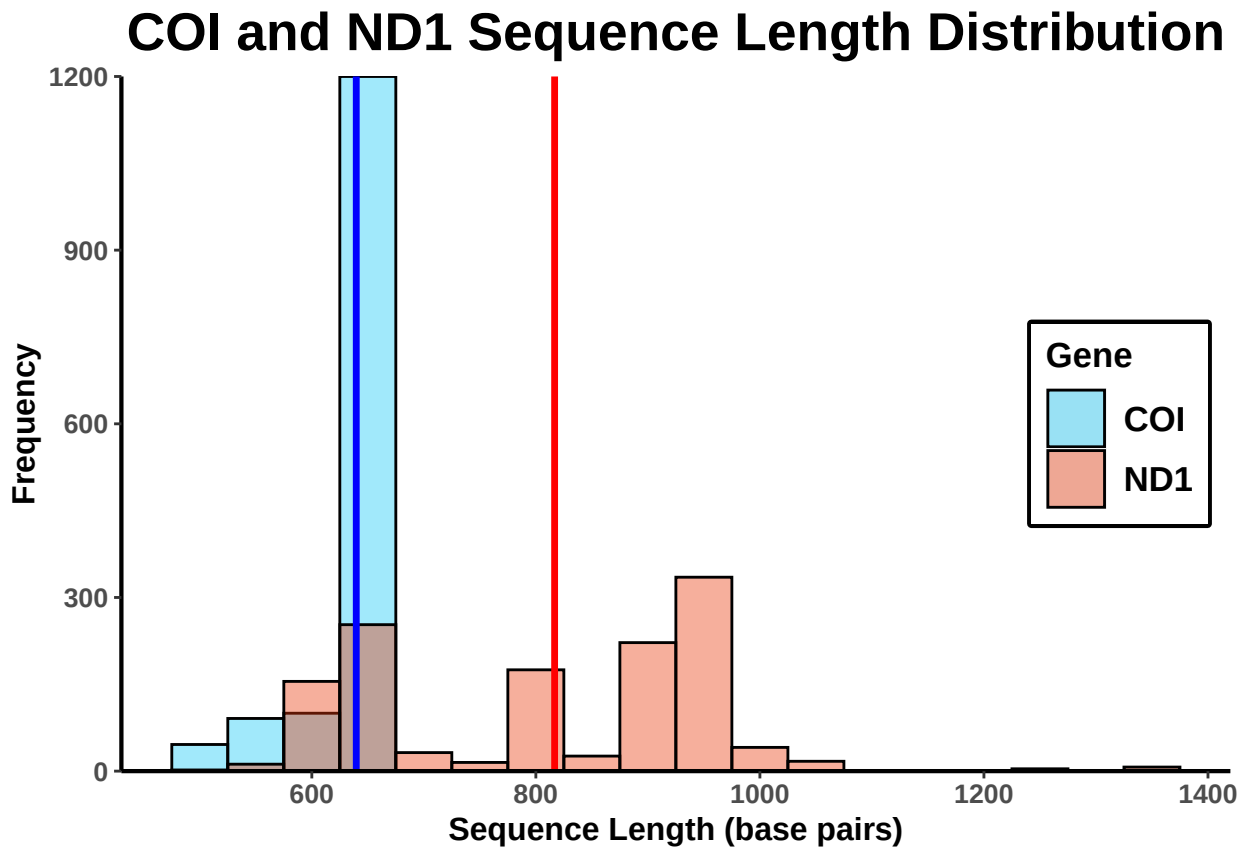
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   503.0   662.0   832.0   816.7   957.0  1369.0
```

```
dfCombine <- rbind(dfCOI_cleaned, dfND1_cleaned)
dfhisto <- dfCombine %>%
  select(Seq_Len, gene)
# From these summary stats, we can see that the average ND1 sequence is longer than the average COI seq


# Figure 1: Histogram
histogram <- ggplot(dfhisto, aes(Seq_Len, fill=gene)) +
  scale_fill_manual(name = "Gene", values=c("#15c8f7", "#ea3708")) +
  geom_histogram(position="identity", binwidth = 50, alpha = 0.4, color = "black") +
  scale_y_continuous(expand = c(0,0)) +
  ggtitle("COI and ND1 Sequence Length Distribution") +
  labs(x = "Sequence Length (base pairs)", y = "Frequency") +
  geom_vline(xintercept=mean(str_count(dfCOI_cleaned$Nucleotides)), lwd=1.2, linetype=1, color="blue") +
  geom_vline(xintercept=mean(str_count(dfND1_cleaned$Nucleotides)), lwd=1.2, linetype=1, color="red") +
  theme(
    legend.box.background = element_rect("black", linewidth = 1.5),
    legend.position = c(0.9, 0.5),
    legend.key.size = unit(0.8, 'cm'),
    legend.title = element_text(face="bold", size=13),
    legend.text = element_text(face="bold", size=13),
    panel.background = element_blank(),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 10),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 20, hjust = 0.5) # Formatting tit
  )
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
```

```
## Warning: The ‘size‘ argument of ‘element_line()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
```

```
histogram
```




```
# After looking at the distribution of the ND1 gene length, I realize that it would probably be better
dfND1_cleaned <- dfND1_cleaned %>%
  filter(Seq_Len < 1200)

nrow(dfCOI_cleaned)
```

Figure 1: Overlapping histograms comparing distribution of sequence lengths for both COI (pink) and ND1 (green). Both genes have a similar sample size (COI: n = 1412, ND1: n = 1267). A wider distribution of sequence length is clearly demonstrated for the ND1 gene, ranging from 600 to over 1000 bpm whereas COI sequences show little variation in their sequence length (mostly around ~650 bp).

```
## [1] 1438
```

```
nrow(dfND1_cleaned) # removed 11 sequences that were over 1200 base pairs long.
```

```
## [1] 1286
```

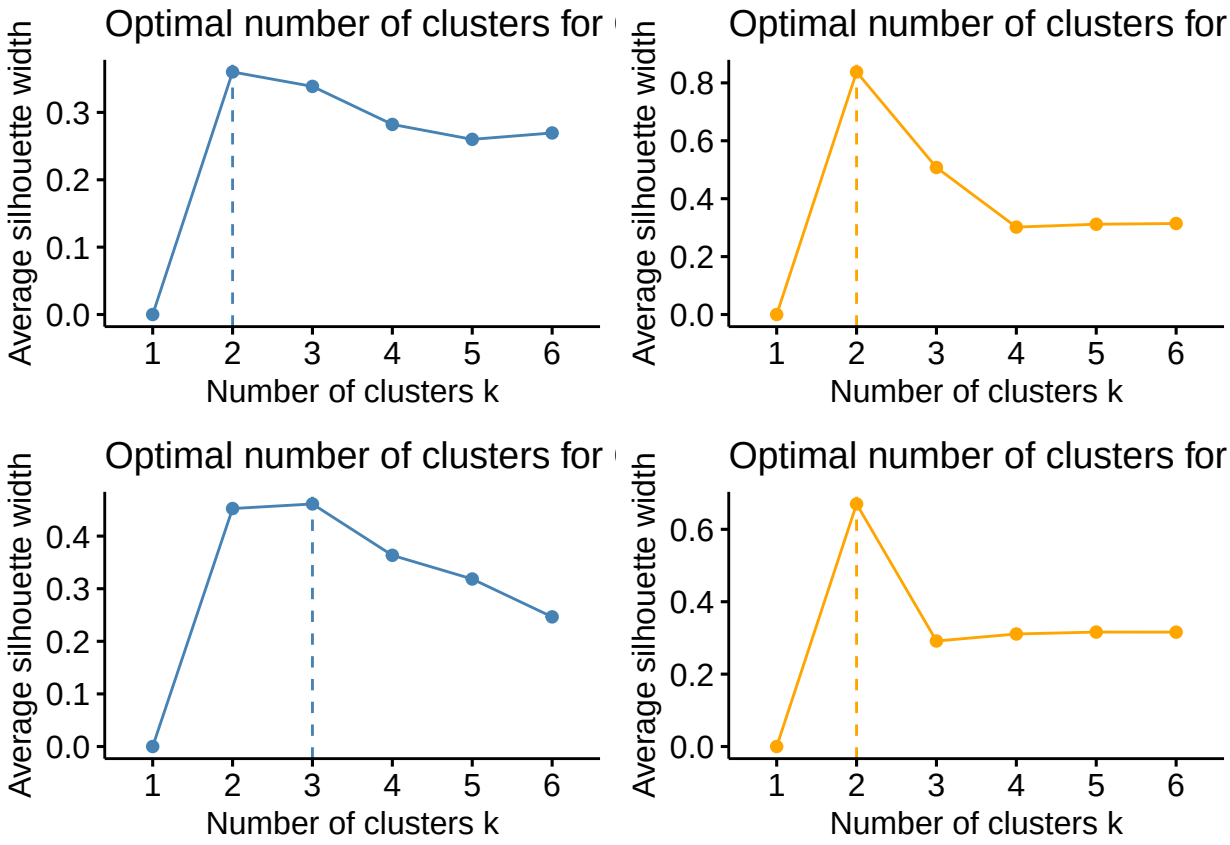
Main Software Tools Description

For the unsupervised clutsering of the COI and ND1 sequences, I chose to cluster them based on k-mer frequencies using the PAM clustering method. I chose to use k-mer frequencies as the sequence feature because I was interested in analyzing a larger data set for this analysis. If I were to use an alignment-based approach, it would not be feasible to use such a larger data set. Then, I chose a partitioning method (PAM) rather than the hierarchical clustering method for a few reasons. First, clusters made by partitioning are computed faster than those created by hierarchical clustering. Also, they are less sensitive to any potential outliers in the data. In particular, the PAM algorithm is a more robust version of the k-means algorithm. (Kaufman, 1990). During my analysis, I used the pam() function from the cluster package (Maechler et al., 2019). I was able to pick between which metric to use for cluster. I chose to use the 'Manhattan' metric for the PAM clustering because it also less sensitive to outliers compared to using Euclidean distances since it uses absolute values and is easier to interpret compared to Euclidean distances.

```
## Code Section 2 - Main Analysis

# Obtaining k-mer frequencies and then calculating pairwise distance matrix among sequences for each ge
# This step can be done using the kdistance() from the kmer package.

# To use the kdistance() function, I first need to convert my sequence data from the dataframe into a D

#
dfCOI_cleaned$Nucleotides <- DNAStringSet(dfCOI_cleaned$Nucleotides) # Convert sequence data into a DNA
dnabin_COI <- as.DNAbin(dfCOI_cleaned$Nucleotides) # Convert to DNAbin object before generating distanc


dfND1_cleaned$Nucleotides <- DNAStringSet(dfND1_cleaned$Nucleotides) # Convert sequence data into a DNA
dnabin_ND1 <- as.DNAbin(dfND1_cleaned$Nucleotides) # Convert to DNAbin object before generating distanc

# Left these here just incase I needed to reconvert the sequence list back to a character class
# dfCOI_cleaned$Nucleotides <- as.character(dfCOI_cleaned$Nucleotides)
# dfND1_cleaned$Nucleotides <- as.character(dfND1_cleaned$Nucleotides)

# Generate pairwise distance matrix using k-mer counts with the kdistance() function from the kmer pack
# This step can take a few minutes depending on how fast your computer is because the resulting matrix

# Distance matrices for k-mers of length 4
distmatrix_COI_k4 <- kmer::kdistance(dnabin_COI, k = 4, method = "manhattan")
distmatrix_ND1_k4 <- kmer::kdistance(dnabin_ND1, k = 4, method = "manhattan")

# Distances matrices for k-mers of length 6
distmatrix_COI_k6 <- kmer::kdistance(dnabin_COI, k = 6, method = "manhattan")
distmatrix_ND1_k6 <- kmer::kdistance(dnabin_ND1, k = 6, method = "manhattan")


# Removal of outliers and performing imputation to replace those values with the mean of the respective
# I've decided to forgo the imputation of outliers with the mean value of the distance matrix, as it wa

# COI_4_dist_mean <- mean(distmatrix_COI_k4)
# COI_4_outlier <- boxplot(distmatrix_COI_k4,plot=FALSE)$out
# distmatrix_COI_k4[which(distmatrix_COI_k4 %in% COI_4_outlier)] <- COI_4_dist_mean
#
#
# ND1_4_dist_mean <- mean(distmatrix_ND1_k4)
```

```
# ND1_4_outlier <- boxplot(distmatrix_ND1_k4,plot=FALSE)$out
# distmatrix_COI_k4[which(distmatrix_COI_k4 %in% COI_4_outlier)] <- ND1_4_dist_mean
#
# COI_6_dist_mean <- mean(distmatrix_COI_k6)
# COI_6_outlier <- boxplot(distmatrix_COI_k6,plot=FALSE)$out
# distmatrix_COI_k6[which(distmatrix_COI_k6 %in% COI_6_outlier)] <- COI_6_dist_mean
#
# ND1_6_dist_mean <- mean(distmatrix_ND1_k6)
# ND1_6_outlier <- boxplot(distmatrix_ND1_k6)$out
# distmatrix_ND1_k6[which(distmatrix_ND1_k6 %in% ND1_6_outlier)] <- ND1_6_dist_mean



# Choosing the optimal number of clusters
# In order to use the PAM algorithm for clustering, there needs to be a pre-determined number of cluste

# Generating plots for different methods (average silhouette method, elbow method (within cluster sums
# The generation of these plots can take some time due to the large distance matrix generated from my s
# I've decided to forgo the use of the elbow method plots for this part, since they were not very infor

# Silhouette method and elbow method plots for COI and ND1 with a k-mer length of 4.
# COI
silhouette_method_plot_COI_4 <- fviz_nbclust(as.matrix(distmatrix_COI_k4), cluster::pam, k.max = 6, meth
silhouette_method_plot_COI_4 <- silhouette_method_plot_COI_4 +
  ggtitle("Optimal number of clusters for COI (k-mer = 4) (Silhouette)")

# elbow_method_plot_COI_4 <- fviz_nbclust(as.matrix(distmatrix_COI_k4), cluster::pam, k.max = 6, method
# elbow_method_plot_COI_4 <- elbow_method_plot_COI_4 +
#   ggtitle("Optimal number of clusters for COI (k-mer = 4) (WSS)")

# ND1
silhouette_method_plot_ND1_4 <- fviz_nbclust(as.matrix(distmatrix_ND1_k4), cluster::pam, k.max = 6, meth
silhouette_method_plot_ND1_4 <- silhouette_method_plot_ND1_4 +
  ggtitle("Optimal number of clusters for ND1 (k-mer = 4) (Silhouette)")

# elbow_method_plot_ND1_4 <- fviz_nbclust(as.matrix(distmatrix_ND1_k4), cluster::pam, k.max = 6, method
# elbow_method_plot_ND1_4 <- elbow_method_plot_ND1_4 +
#   ggtitle("Optimal number of clusters for ND1 (k-mer = 4) (WSS)")


# Silhouette method and elbow method plots for COI and ND1 with a k-mer length of 6.
# COI
silhouette_method_plot_COI_6 <- fviz_nbclust(as.matrix(distmatrix_COI_k6), cluster::pam, k.max = 6, meth
silhouette_method_plot_COI_6 <- silhouette_method_plot_COI_6 +
  ggtitle("Optimal number of clusters for COI (k-mer = 6) (Silhouette)")

# elbow_method_plot_COI_6 <- fviz_nbclust(as.matrix(distmatrix_COI_k6), cluster::pam, k.max = 6, method
# elbow_method_plot_COI_6 <- elbow_method_plot_COI_6 +
#   ggtitle("Optimal number of clusters for COI (k-mer = 6) (WSS)")

# ND1
silhouette_method_plot_ND1_6 <- fviz_nbclust(as.matrix(distmatrix_ND1_k6), cluster::pam, k.max = 6, meth
silhouette_method_plot_ND1_6 <- silhouette_method_plot_ND1_6 +
```

```
  ggtitle("Optimal number of clusters for ND1(k-mer = 6) (Silhouette)")

# elbow_method_plot_ND1_6 <- fviz_nbclust(as.matrix(distmatrix_ND1_k6), cluster::pam, k.max = 6, method
# elbow_method_plot_ND1_6 <- elbow_method_plot_ND1_6 +
#   ggtitle("Optimal number of clusters for ND1 (k-mer = 6) (WSS)")

# Generating the figures containing the graphs for each gene in one plot.
grid.arrange(silhouette_method_plot_COI_4, silhouette_method_plot_ND1_4, silhouette_method_plot_COI_6, s
```

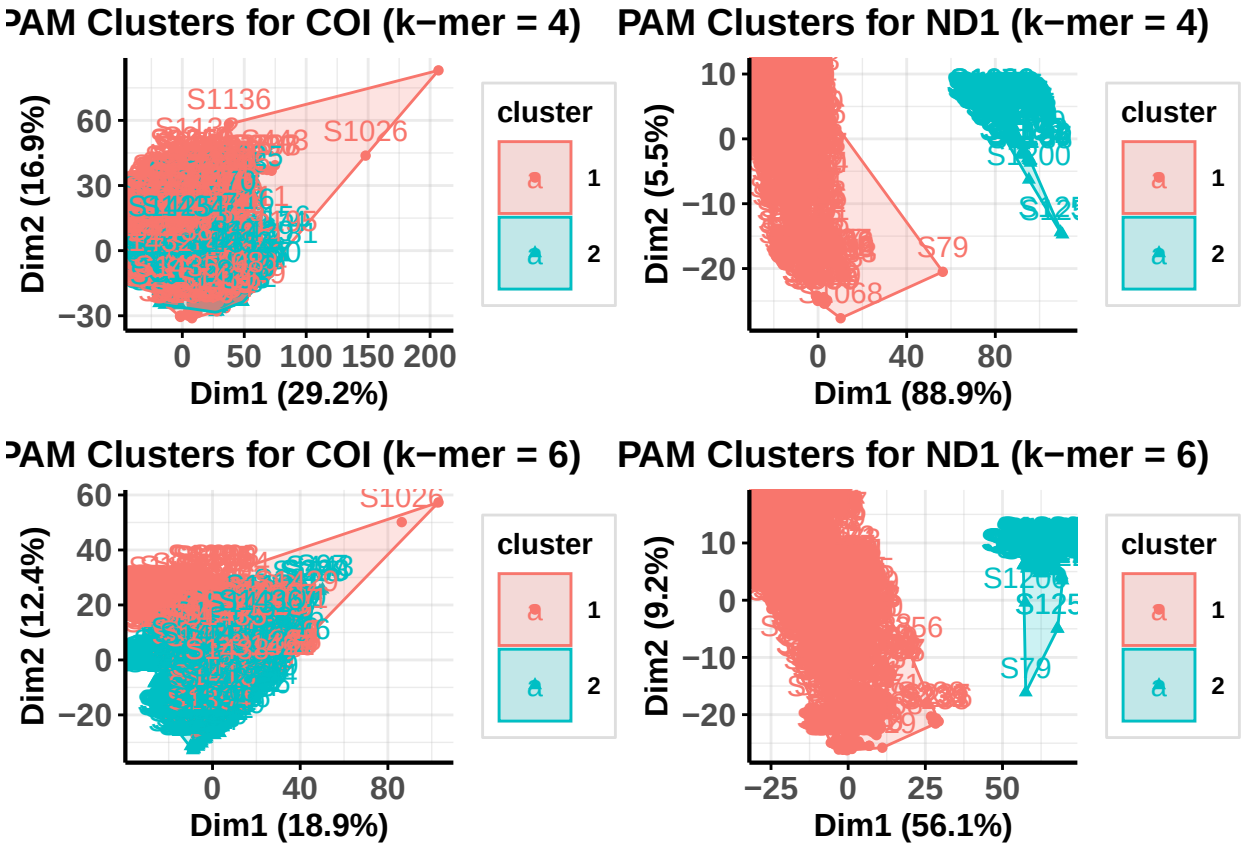



```
# There are a few functions that can be used to cluster sequences based on k-mer frequencies, which dep

# The pam() function from the cluster package uses the distance matrix generated from the kdistance() f

# Generating clusters
pam_COI_4 <- cluster::pam(distmatrix_COI_k4, k = 2, metric = "manhattan")
pam_ND1_4 <- cluster::pam(distmatrix_ND1_k4, k = 2, metric = "manhattan")
pam_COI_6 <- cluster::pam(distmatrix_COI_k6, k = 2, metric = "manhattan")
pam_ND1_6 <- cluster::pam(distmatrix_ND1_k6, k = 2, metric = "manhattan")

# Plotting the clusters using the fviz_cluster() function from the factoextra package. Before using the
# COI clusters with k-mer length of 4
pam_COI_4$data = distmatrix_COI_k4
```

```
pamplot_COI_4 <- fviz_cluster(pam_COI_4, main = "PAM Clusters for COI (k-mer = 4)") +
  theme(
    panel.background = element_blank(),
    panel.grid = (element_line(alpha("gray", 0.3))),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 12),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 14, hjust = 0.5),
    legend.title = element_text(face="bold"),
    legend.key.size = unit(1, 'cm'),
    legend.text = element_text(face="bold"),
    legend.background = element_rect(colour = alpha("gray", 0.5))
    )


# ND1 clusters with k-mer length of 4
pam_ND1_4$data = distmatrix_ND1_k4
pamplot_ND1_4 <- fviz_cluster(pam_ND1_4, main = "PAM Clusters for ND1 (k-mer = 4)")+
  theme(
    panel.background = element_blank(),
    panel.grid = (element_line(alpha("gray", 0.3))),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 12),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 14, hjust = 0.5),
    legend.title = element_text(face="bold"),
    legend.key.size = unit(1, 'cm'),
    legend.text = element_text(face="bold"),
    legend.background = element_rect(colour = alpha("gray", 0.5))
  )

# COI clusters with k-mer length of 6
pam_COI_6$data = distmatrix_COI_k6
pamplot_COI_6 <- fviz_cluster(pam_COI_6, main = "PAM Clusters for COI (k-mer = 6)")+
  theme(
    panel.background = element_blank(),
    panel.grid = (element_line(alpha("gray", 0.3))),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 12),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 14, hjust = 0.5),
    legend.title = element_text(face="bold"),
    legend.key.size = unit(1, 'cm'),
    legend.text = element_text(face="bold"),
    legend.background = element_rect(colour = alpha("gray", 0.5))
  )
```

```
# ND1 clusters with k-mer length of 6
pam_ND1_6$data = distmatrix_ND1_k6
pamplot_ND1_6 <- fviz_cluster(pam_ND1_6, main = "PAM Clusters for ND1 (k-mer = 6)") +
  theme(
    panel.background = element_blank(),
    panel.grid = (element_line(alpha("gray", 0.3))),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 12),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 14, hjust = 0.5),
    legend.title = element_text(face="bold"),
    legend.key.size = unit(1, 'cm'),
    legend.text = element_text(face="bold"),
    legend.background = element_rect(colour = alpha("gray", 0.5))
  )


grid.arrange(pamplot_COI_4, pamplot_ND1_4, pamplot_COI_6, pamplot_ND1_6)
```

Figure 2: Using average silhouette and WSS methods to obtain the optimal number of clusters for k-means clustering of COI (dark blue) and ND1 (orange) for both k-mer lengths (4 and 6). The optimal number of clusters was found to be 2 for both genes and respective k-mer lengths.



```
# Silhouette plots to determine cluster strength.

COI_silh_4 <- fviz_silhouette(pam_COI_4, label = FALSE)
```

Figure 3: Cluster plots for both COI and ND1 genes for k-mer lengths of 4 and 6 using the PAM clustering algorithm.

```
##   cluster size ave.sil.width
## 1       1 1029          0.04
## 2       2  409          0.10
```

```
COI_4_width <- paste0("Average Silhouette Width: ", round(mean(COI_silh_4$data$sil_width), 2))
COI_silh_4 <- COI_silh_4 +
  labs(title = "COI Silhouette Plot (k-mer = 4)", tag = COI_4_width)+
  theme(
    plot.tag = element_text(size = 12),
    plot.tag.position = c(0.5, 0.9),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 12),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 14, hjust = 0.5),
    legend.title = element_text(face="bold"),
    legend.text = element_text(face="bold"),
    legend.background = element_rect(colour = alpha("gray", 0.5))
  )

ND1_silh_4 <- fviz_silhouette(pam_ND1_4, label = FALSE)
```

```
##   cluster size ave.sil.width
## 1       1 1113          0.53
## 2       2  173          0.84
```

```
ND1_4_width <- paste0("Average Silhouette Width: ", round(mean(ND1_silh_4$data$sil_width), 2))
ND1_silh_4 <- ND1_silh_4 +
  labs(title = "ND1 Silhouette Plot (k-mer = 4)", tag = ND1_4_width)+
  theme(
    plot.tag = element_text(size = 12),
    plot.tag.position = c(0.5, 0.9),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 12),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 14, hjust = 0.5),
    legend.title = element_text(face="bold"),
    legend.text = element_text(face="bold"),
    legend.background = element_rect(colour = alpha("gray", 0.5))
  )
```

```
COI_silh_6 <- fviz_silhouette(pam_COI_6, label = FALSE)
```

```
##   cluster size ave.sil.width
## 1       1  684          0.05
## 2       2  754          0.04
```

```
COI_6_width <- paste0("Average Silhouette Width: ", round(mean(COI_silh_6$data$sil_width), 2))
COI_silh_6 <- COI_silh_6 +
  labs(title = "COI Silhouette Plot (k-mer = 6)", tag = COI_6_width)+
  theme(
    plot.tag = element_text(size = 12),
    plot.tag.position = c(0.5, 0.9),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 12),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 14, hjust = 0.5),
    legend.title = element_text(face="bold"),
    legend.text = element_text(face="bold"),
    legend.background = element_rect(colour = alpha("gray", 0.5))
  )


ND1_silh_6 <- fviz_silhouette(pam_ND1_6, label = FALSE)
```

```
##   cluster size ave.sil.width
## 1       1 1112          0.23
## 2       2  174          0.78
```

```
ND1_6_width <- paste0("Average Silhouette Width: ", round(mean(ND1_silh_6$data$sil_width), 2))
ND1_silh_6 <- ND1_silh_6 +
  labs(title = "ND1 Silhouette Plot (k-mer = 6)", tag = ND1_6_width)+
  theme(
    plot.tag = element_text(size = 12),
    plot.tag.position = c(0.5, 0.9),
    axis.line.y = element_line(size = 0.75),
    axis.line.x = element_line(size = 0.75),
    axis.text = element_text(face = "bold", size = 12),
    axis.title.x = element_text(face = "bold", size = 12),
    axis.title.y = element_text(face = "bold", size = 12),
    plot.title = element_text(face = "bold", colour = "black", size = 14, hjust = 0.5),
    legend.title = element_text(face="bold"),
    legend.text = element_text(face="bold"),
    legend.background = element_rect(colour = alpha("gray", 0.5))
  )

grid.arrange(COI_silh_4, ND1_silh_4, COI_silh_6, ND1_silh_6)
```

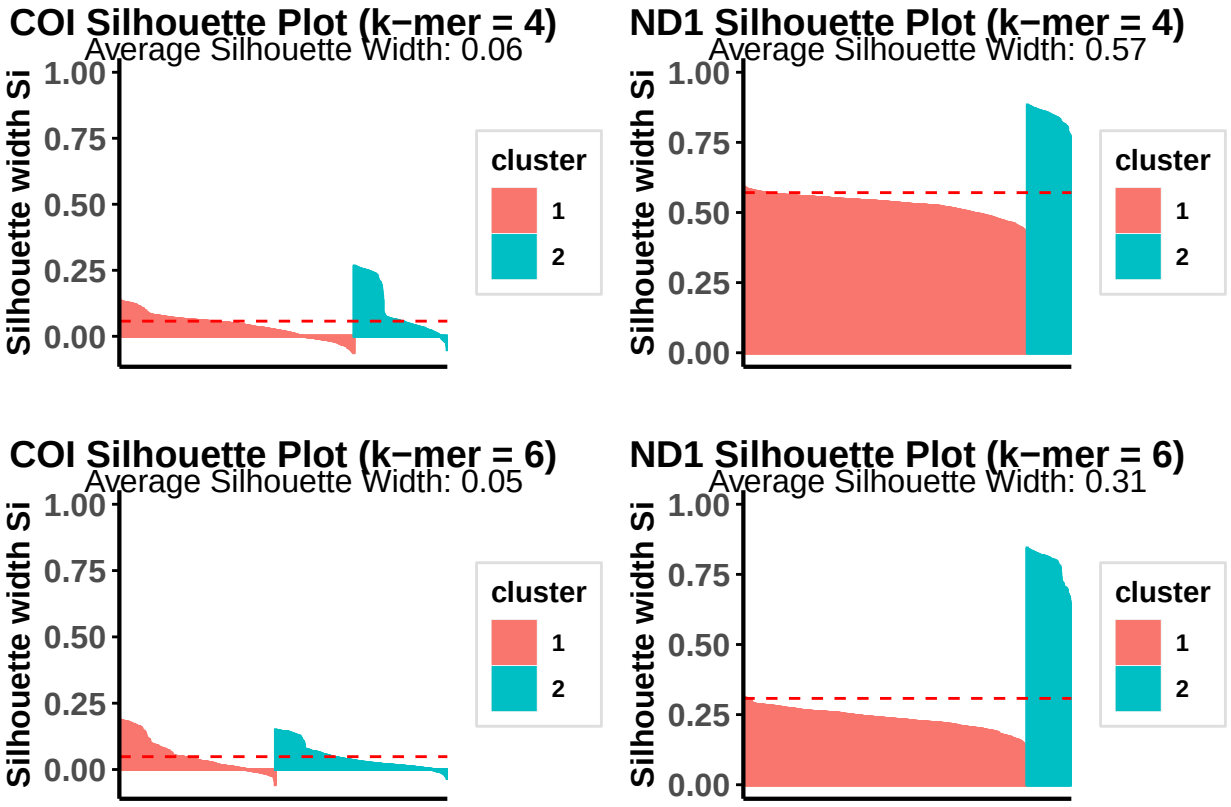


Figure 4: Silhouette plots for all each gene and their respective k-mer lengths (4 and 6). COI silhouette plots are shown on the left, and ND1 silhouette plots are shown on the right. Each silhouette plot also shows the average silhouette width for all the clusters in a particular subset. By using the average silhouette widths, it is easy to see that ND1 clusters exhibited much stronger clustering relative to COI clusters.

Results & Discussion

By looking at the results of this analysis, it is clear to see that the ND1 gene exhibits far better clustering patterns than the COI gene within the family Microchiroptera. The cluster plots demonstrates this obvious difference between the clustering patterns of both genes (Figure 3). This is further supported by the Silhouette plots that showcase the difference in cluster strength between both genes (Figure 4). ND1 sequences had a much larger average silhouette width compared to COI sequences. Silhouette indices range from -1 to 1, where 1 is indicative of optimal clustering (Gentleman & Carey, 2008). These results were not expected, as COI is the hallmark gene used for DNA barcoding. Therefore, I estimated that the clustering patterns for COI would be more distinct than what my analysis has shown. I did not expect to see such a big difference in the degree of clustering between these two genes, as they are both mitochondrial genes from the same taxonomic group. In terms of comparing k-mer lengths (4 and 6), I expected to see stronger clustering as k-mer length increased.

The conclusions of this analysis are not completely definitive, as there are many parameters and methods that could be altered to achieve potentially more accurate results. For instance, a larger sample size could have influenced the clustering patterns seen between both genes. As mentioned before, I had to randomly sample from the abundance of COI sequences to equalize the number of sequences to that of the ND1 gene. Of course, the clustering method, metrics, and parameters chosen throughout this analysis could also all

be altered in order to give potentially different results. In the context of choosing the optimal number of clusters, I simply used a heuristic method of determining that number by using the silhouette method plots. This served as a guideline for my analysis, instead of just arbitrarily choosing a number of clusters to work with. In terms of choosing the optimal k-mer length for my analysis, I had to find a 'sweetspot' between using a length that would give me the best results, and a length that is computationally feasible for me to work with. As a result, I found that a k-mer length of 6 was the optimal choice for my analysis, since I found that any larger k-mer values took far too long to compute. There exists methods to computing the optimal k-mer length using and comparing abundance histograms for different lengths, but these methods take a lot of time to perform (Chikhi et al., 2014).

The next steps in this research would involve using alternative clustering methods that for these specific genes (such as hierarchical clusters and k-means clustering) and seeing how the results might change. For instance, the application of hierarchical clustering of DNA k-mers for RNA-seq data was already done in a study by Kaisers et al. (2018). If I were to conduct a larger project on this topic, I would investigate and compare different clustering methods to see which ones would reveal the most accurate clustering patterns in a given set of sequence data. I would also look to investigate the possibility of clustering for larger k-mer lengths, to see if the results seen in my analysis align with the results of this future research. The results of my analysis showcase difference between the clustering patterns of the COI and ND1 genes, which could be worth further investigating to better understand which genes are more suitable for taxonomic classification in a given taxonomic group.

Acknowledgements

I would like to acknowledge Jesse Wolf and Nishita Sharif for providing me with valuable insight and knowledge to help me during this project.

Published References

Arora, P. et al. (2016) Analysis of K-Means and K-Medoids Algorithm For Big Data, Procedia Computer Science. https://doi.org/10.1016/j.procs.2016.02.095

Chikhi, R., & Medvedev, P. (2014). Informed and automated k-mer size selection for genome assembly. Bioinformatics, 30(1), 31–37. https://doi.org/10.1093/bioinformatics/btt310

Kaisers, W., Schwender, H., & Schaal, H. (2018). Hierarchical Clustering of DNA k-mer Counts in RNAseq Fastq Files Identifies Sample Heterogeneities. International Journal of Molecular Sciences, 19(11), 3687. https://doi.org/10.3390/ijms19113687

Kaufman, L. & Rousseeuw, P. J. (1990) Partitioning Around Medoids (Program PAM). In Finding Groups in Data (pp. 68–125). John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470316801.ch2

Kislyuk, A., Bhatnagar, S., Dushoff, J., & Weitz, J. S. (2009). Unsupervised statistical clustering of environmental shotgun sequences. BMC Bioinformatics, 10(1), 316. https://doi.org/10.1186/1471-2105-10-316

References for R packages

Auguie, B. (2017. gridExtra: Miscellaneous Functions for "Grid" Graphics.

Brock G, Pihur V, Datta S, Datta S (2008). "clValid: An R Package for Cluster Validation." Journal of Statistical Software, 25(4), 1–22. https://www.jstatsoft.org/v25/i04/.

Gagolewski M., stringi: Fast and portable character string processing in R, Journal of Statistical Software 103(2), 2022, 1–59, doi:10.18637/jss. v103. i02.

Kassambara, A. and Mundt, F. (2020) Factoextra: Extract and Visualize the Results of Multivariate Data Analyses. R Package Version 1.0.7.

Pagès H, Aboyoun P, Gentleman R, DebRoy S (2022). Biostrings: Efficient manipulation of biological strings. R package version 2.66.0, https://bioconductor.org/packages/Biostrings.

Wickham H, Girlich M (2022). tidyr: Tidy Messy Data. https://tidyr.tidyverse.org, https://github.com/tidyverse/tidyr.

Wilkinson S (2018). kmer: an R package for fast alignment-free clustering of biological sequences. doi: 10.5281/zenodo.1227690, https://cran.r-project.org/package=kmer.

Winter, D. (2017). rentrez: an R package for the NCBI eUtils API. The R Journal, 9, 520–526.

Other sources that were consulted:

https://www.r-bloggers.com/2017/02/finding-optimal-number-of-clusters/

https://www.statology.org/elbow-method-in-r/

https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html

https://uc-r.github.io/kmeans_clustering#kmeans

https://stats.stackexchange.com/questions/145192/what-is-the-benefit-of-using-manhattan-distance-for-k-medoid-than-using-euclidea

https://stackoverflow.com/questions/61501842/overlaying-two-histograms-with-different-rows-using-ggplot2

https://stackoverflow.com/questions/25571547/select-unique-values-with-select-function-in-dplyr-library

https://lgatto.github.io/IntroMachineLearningWithR/unsupervised-learning.html#how-does-k-means-work

https://datascience.stackexchange.com/questions/48883/what-could-this-mean-if-your-elbow-curve-looks-like-this

https://stackoverflow.com/questions/28890627/r-how-do-i-concatenate-decimal-number-value-and-string-in-print-statement

https://www.datanovia.com/en/lessons/k-medoids-in-r-algorithm-and-practical-examples/

http://www.sthda.com/english/articles/29-cluster-validation-essentials/96-determiningthe-optimal-number-of-clusters-3-must-know-methods/#computing-the-number-of-clusters-using-r

http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/81-ggplot2-easy-way-to-mix-multiple-graphs-on-the-same-page/

https://www.r-bloggers.com/2020/01/how-to-remove-outliers-in-r/

https://stackoverflow.com/questions/61445767/plot-pam-cluster-results-with-fviz-clust