Arvind Srinivas
1301477

Assignment 2: Building a Taxonomic Classifier for the Cytochrome B gene among *Artiodactyla* *Carnivora*, and *Squamata*

Introduction

In recent years, the exploration of genetic diversity within specific taxonomic groups has shown great strides of advancement. With supervised machine learning algorithms, researchers are now able to make predictions on biological categories based on predictor variables.

This project looks to create a classifier using supervised machine learning, to discern between different taxonomic groups based on a common/conserved marker code. The marker code in question is the Cytochrome B (CytB) gene, which is widely used as a marker gene for vertebrates due to it being highly conserved (Parson et al., 2000). In general, this mitochondrial gene produces a protein that plays a role in energy metabolism and works in tandem with the electron transport chain (Pal et al., 2019). While the gene is typically found in a similar form across different taxa, variations in its nucleotide composition are still expected. Therefore, developing a classifier can be particularly valuable, especially when dealing with well-conserved genes. This project focuses on building a classifier to discern between the CytB gene of *Artiodactyla* (hoofed animals), the CytB gene of *Carnivora* (carnivorous animals), and the CytB gene of *Squamata* (reptiles) by looking at nucleotide proportions and 4-mers. The more sequence data available, the better the classifier would be when dealing with unseen data.

Code Part 1:

```
library(tidyverse)
library(Biostrings)
library(BiocManager)
library(seqinr)
library(rentrez)

####----PART 1: DATA EXPLORATION, FILTERING, AND QUALITY CHECKING----
```

#Searching for hits. This function takes in the order_name and returns the search results for carnivora, artiodactyla, and squamata. The terms are already defined in search_terms, where we only want the CytB gene and for it to be 800 to 1200 base pairs in length (CytB gene is on average 1140bp). Also using web_history to gain a large dataset, meaning that it might take time to process. Search_results performs the search based on search_term in the nuccore database.

```
perform_cytb_search <- function(order_name, retmax = NULL) {
  search_term <- paste(order_name, "[ORGN] AND CytB[Gene] AND 800:1200[SLEN]")
  search_results <- entrez_search(db = "nuccore", term = search_term, use_history = TRUE)

  #Data exploration.
  cat("Search results for", order_name, ":\n")
  cat("Class:", class(search_results), "\n")
  cat("Types of IDs:", search_results$ids, "\n")
  cat("Count of total number of hits:", search_results$count, "\n")
  return(search_results)
}

#Searching through carnivora.
search_results_carnivora <- perform_cytb_search("Carnivora")

#Searching through artiodactyla.
search_results_artiodactyla <- perform_cytb_search("Artiodactyla")

#Searching through squamata.
search_results_squamata <- perform_cytb_search("Squamata")
```

#Fetching sequences. Using entrez_fetch function to retrieve carnivora, artiodactyla, and squamata squences based on the web_history. Fetching them in a fasta file format. This may take some time.

```
fetch_sequences_and_explore <- function(search_results) {
  sequences_fetch <- entrez_fetch(db = "nuccore", web_history = search_results$web_history, rettype = "fasta")

  #Data exploration: checking the class of fetched sequences.
  cat("Class of fetched sequences:", class(sequences_fetch), "\n")
  return(sequences_fetch)
}

#Carnivora fetched sequences.
carnivora_sequences_fetch <- fetch_sequences_and_explore(search_results_carnivora)

#Artiodactyla fetched sequences.
artiodactyla_sequences_fetch <- fetch_sequences_and_explore(search_results_artiodactyla)

#Squamata fetched sequences.
squamata_sequences_fetch <- fetch_sequences_and_explore(search_results_squamata)
```

#Writing all sets of fasta files to a text editor to observe next quality checking steps. Also checking for any unusual sequence lengths, but there shouldn't be too much to worry as SLEN was already set for 800:1200.

```
write(carnivora_sequences_fetch, "cytB_carnivora_fetch.fasta", sep = "\n")

write(artiodactyla_sequences_fetch, "cytB_artiodactyla_fetch.fasta", sep = "\n")

write(squamata_sequences_fetch, "cytB_squamata_fetch.fasta", sep = "\n")
```

Arvind Srinivas
1301477

```
#Read all sets of character data into DNAStringSet.
carnivora_stringset <- readDNAStringSet("cytB_carnivora_fetch.fasta")

artiodactyla_stringset <- readDNAStringSet("cytB_artiodactyla_fetch.fasta")

squamata_stringset <- readDNAStringSet("cytB_squamata_fetch.fasta")

#Function that changes all stringsets from previous lines into individual dataframes. cytB_title
contains the sequence names and cytB_sequence is the actual sequence of each name.
stringset_to_dataframe <- function(stringset, df_name) {
  df <- data.frame(cytB_title = names(stringset), cytB_sequence = paste(stringset))

  #Data exploration: checking class.
  cat("Class of", df_name, "data frame:", class(df), "\n")
  return(df)
}

#Carnivora dataframe.
carnivora_df <- stringset_to_dataframe(carnivora_stringset, "carnivora")

#Artiodactyla dataframe.
artiodactyla_df <- stringset_to_dataframe(artiodactyla_stringset, "artiodactyla")

#Squamata dataframe.
squamata_df <- stringset_to_dataframe(squamata_stringset, "squamata")
```

```
#Adding a column called NucleotideCount to each individual dataframe. It counts the number of characters in each sequence and puts the each count as observations in the new NucleotideCount column.
add_nucleotide_count <- function(df, stringset) {
  df$NucleotideCount <- sapply(stringset, function(seq) nchar(as.character(seq)))
  return(df)
}

#Adding it to carnivora_df.
carnivora_df <- add_nucleotide_count(carnivora_df, carnivora_stringset)

#Adding it to artiodactyla_df.
artiodactyla_df <- add_nucleotide_count(artiodactyla_df, artiodactyla_stringset)

#Adding it to squamata_df.
squamata_df <- add_nucleotide_count(squamata_df, squamata_stringset)

#Adding columns to all dataframes called "Order" and placing the order names with respect to their sequences.
carnivora_df$Order <- "Carnivora"
artiodactyla_df$Order <- "Artiodactyla"
squamata_df$Order <- "Squamata"

#Create new combined dataframe by combining the carnivora, artiodactyla, and squamata data frames by rows.
combined_df <- rbind(carnivora_df, artiodactyla_df, squamata_df)

#Filtering steps in the combined_df to create combined_df1. New dataframe combines individual dataframes, as well as removing N's and other symbols from the sequences. The cap
```

was set low (0.0001) to ensure little to no sequences containing Ns. This may take some time to load.

```
combined_df1 <- combined_df %>%
  mutate(cytB_sequence = str_remove(cytB_sequence, "^[-N]+")) %>%
  mutate(cytB_sequence = str_remove(cytB_sequence, "[-N]+$")) %>%
  mutate(cytB_sequence = str_remove_all(cytB_sequence, "-+")) %>%
  filter(str_count(cytB_sequence, "N") <= (0.0001 * str_count(cytB_sequence)))
view(combined_df1)

class(combined_df1) #Checking class.
dim(combined_df1) #Checking dimenstions. There should be 4 columns but many rows.
table(combined_df1$Order) #Observing counts of order data. Confirming that sequences with N's were removed.
summary(nchar(combined_df1$cytB_sequence)) #Getting statistics for number of nucleotides in sequence for the CytB gene for each order.

#Create a faceted histogram for sequence length frequency. Inputting combined_df1 as the dataframe.
fill_colors <- c("Artiodactyla" = "purple", "Carnivora" = "blue", "Squamata" = "orange")

ggplot(combined_df1, aes(x = NucleotideCount, fill = Order)) +
  geom_histogram(binwidth = 20, color = "black") +
  scale_fill_manual(values = fill_colors) +
  labs(title = "Distribution of Sequence Lengths Between Artiodactyla, Carnivora, and Squamata", x = "Sequence Length (bp)", y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5),  strip.text = element_text(size = 12)) + #Centers title.
  facet_wrap(~Order) #Combines all histograms into one.
```

Arvind Srinivas
1301477

Code Part 2:

```
####----MAIN ANALYSIS (PART 2): BUILDING THE CLASSIFIER----

#Calculate sequence features. Making a new dataframe called cytB_df. Changing cytB_df to
DNAStringset.
cytB_df <- as.data.frame(combined_df1)
cytB_df$cytB_sequence <- DNAStringSet(cytB_df$cytB_sequence)
view(cytB_df)

#Looking at nucleotide frequencies from the cytB_df
cytB_df <- cbind(cytB_df, as.data.frame(letterFrequency(cytB_df$cytB_sequence, letters =
c("A", "C", "G", "T"))))

#Proportions of A, C, T, and G into new columns of cytB_df.
cytB_df$Aproportion <- (cytB_df$A) / (cytB_df$A + cytB_df$T + cytB_df$C + cytB_df$G)

cytB_df$Tproportion <- (cytB_df$T) / (cytB_df$A + cytB_df$T + cytB_df$C + cytB_df$G)

cytB_df$Gproportion <- (cytB_df$G) / (cytB_df$A + cytB_df$T + cytB_df$C + cytB_df$G)

cytB_df$Cproportion <- (cytB_df$C) / (cytB_df$A + cytB_df$T + cytB_df$C + cytB_df$G)

#Use k-mer of length 4 to get tetranucleotide frequency and add these frequencies as a new
column in cytB_df.
cytB_df <- cbind(cytB_df, as.data.frame
        (oligonucleotideFrequency(x = cytB_df$cytB_sequence, width = 4, as.prob = TRUE)))

cytB_df$cytB_sequence <- as.character(cytB_df$cytB_sequence)
```

```
#Take sample from "Order" column of dataframe.
sample <- min(table(cytB_df$Order))
sample #Lowest number of sequence counts of all the orders. Comes from carnivora.

validation_size <- floor(0.3 * sample) #Validation size is 30% of sample size.
validation_size

#Reproducibility.
set.seed(7845)

#Create a validation data set by sampling from each category of "Order".
cytB_dfValidation <- cytB_df %>%
  group_by(Order) %>%
  sample_n(validation_size)

#Reproducibility.
set.seed(759385)

#Create a training dataset by filtering out validation samples and sampling the remaining
amount of data.
cytB_dfTraining <- cytB_df %>%
  filter(!cytB_title %in% cytB_dfValidation$cytB_title) %>% #Samples not from Validation data.
  group_by(Order) %>%
  sample_n(ceiling(0.7 * sample))

#Checking the distribution of orders in the training dataset.
table(cytB_dfTraining$Order)
ncol(cytB_df) #Finding number of columns.
```

```
#Making a randomForest classifier with training data.
order_classifier <- randomForest(
  x = cytB_dfTraining[, 9:268], #Takes proportions and 4-mer possibilities.
  y = as.factor(cytB_dfTraining$Order),
  ntree = 150,
  importance = TRUE
)

#Viewing results of classifier. This take some time.
order_classifier

#Plot the error rate proximity plot with custom line colors.
custom_colors <- c("black", "purple", "blue", "orange")

plot(order_classifier, main = "Error Rate Proximity Plot Based on Order Classification for Training
Data", col = custom_colors, lty = c(1,2,2,2), lwd = c(1,1.5,1.5,1.5))
legend("topright", legend = c("Out-of-Bag Samples", "Artiodactyla", "Carnivora", "Squamata"),
col = custom_colors, lty = c(1,2,2,2), cex = 0.7)

#Check on validation data.
predictCytBValidation <- predict(order_classifier, cytB_dfValidation[,9:268])

predictCytBValidation #The squamata entries were omitted because it went beyond the max
amount of entries to display.
```

Arvind Srinivas
1301477

Results:

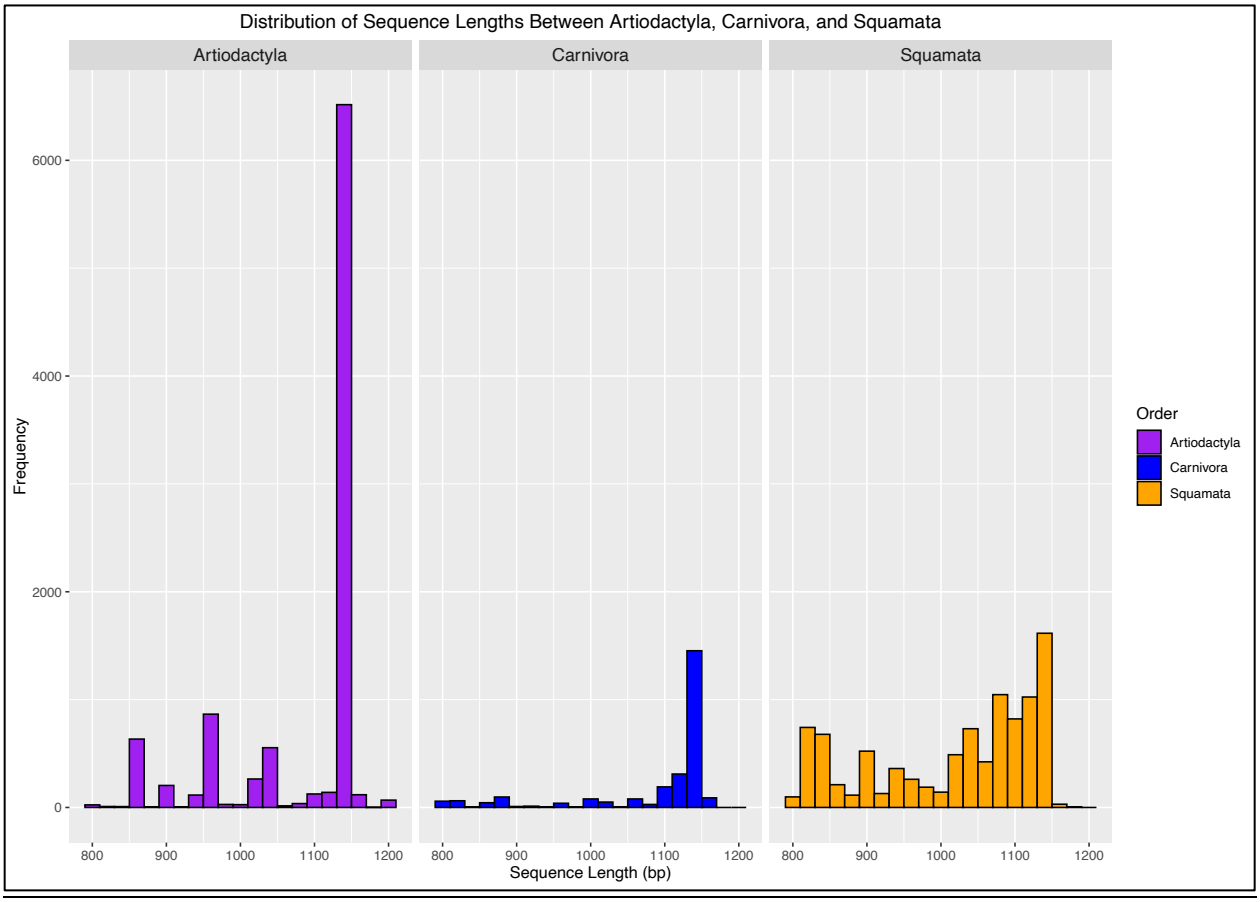

Figure 1: Faceted histogram displaying the distribution of sequence lengths between *Artiodactyla, Carnivora*, and *Squamata*. The sequences for each order have been filtered to only include sequences between 800bp to 1200bp.

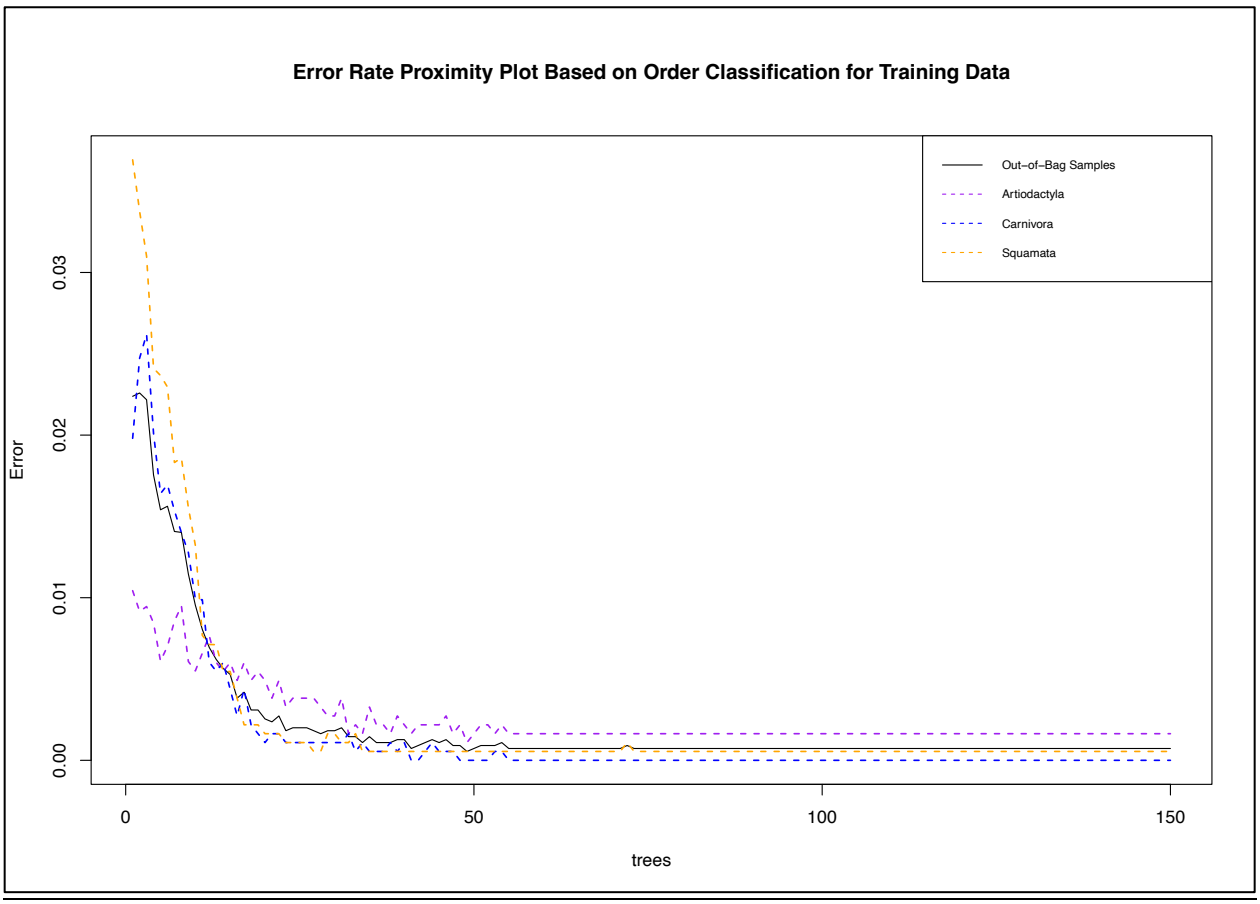

Figure 2: Error rate proximity plot showcasing the order classifier error rate from the randomForest function. It's a valuable tool for understanding the machine learning model's performance, especially in the context of taking training data to classify taxonomy. From this plot, it appears the model had slight error when classifying *Artiodactyla* and *Squamata*. Eventually, all lines leveled off.

Comparing the sequence distribution among all the three taxonomic orders displays the consistency of sequence lengths. Maintaining a consistent range for sequence length is important to ensure compatibility of data with the machine learning model, as outliers can affect classification. Figure 1 displays a faceted bar graph with frequencies of sequence lengths of CytB for all three orders. The highest frequencies of sequence lengths are 900bp, 1140bp, and 1140bp for *Artiodactyla*, *Carnivora*, and *Squamata* respectively. Figure 2 shows the error rates of the classifier. The classifier seemed to have 0% error when classifying *Carnivora*, while

there was a 0.0016% and 0.00054% error rate for *Artiodactyla* and *Squamata* sequences respectively. The Out-of-Bag samples are points that are not included in the decision trees and shows how effective the model is.

Discussion:

In general, the classifier performed as anticipated. Despite the CytB gene's high degree of conservation, there were nucleotide composition discrepancies among these taxonomic orders. The use of a k-mer length of 4 likely allowed the model to capture local sequence distinctions, particularly when dealing with a highly conserved gene. Along with tetranucleotide patterns, nucleotide proportions were used for order classification, resulting in low error rate for the model. The 0.0016% error rate and 0.00054% error rate for *Artiodactyla* and *Squamata* respectively is possibly due to these orders containing more partial coding sequences compared to *Carnivora*. Originally, 9999 sequences were fetched for both *Artiodactyla* and *Squamata*, while *Carnivora* fetched around 2000. Figure 1 shows that 900bp was the most common sequence length for *Artiodactyla* CytB genes. Moreover, there was almost an even distribution in terms of frequency for the sequence length of the fetched *Squamata* data (despite the greatest frequency of the data being 1140bp in length). The average CytB length in vertebrates in 1140bp, meaning that partial sequences of the CytB gene may have led to a few misclassifications (Linacre, 2012). Nevertheless, these error rates are so low that they can be deemed negligible. The validation data also proved that the classifier works for unseen data. A future incentive for this classification model is multi-taxonomic classification and multi-omics integration for classification (proteomic data, metabolomic data, etc.). Looking at multiple variables for taxonomic classification could prove to be useful when inputting different kinds of data.

References:

Geekoverdosegeekoverdose. (2016, July 4). Random Forest graph interpretation in R. Cross Validated. https://stats.stackexchange.com/questions/222039/random-forest-graph-interpretation-in-r

Kosourova, E. (2022). How to Write Functions in R. DataQuest.

https://www.dataquest.io/blog/write-functions-in-r/

Kosourova, E. (2023, March 6). Apply functions in R with examples [apply(), sapply(),lapply (), tapply()]. Dataquest. https://www.dataquest.io/blog/apply-functions-in-r-sapply-lapply-tapply/#:~:text=The%20sapply()%20function%20is,the%20most%20simplified%20data%20structure.

Linacre, A. (2012). Capillary electrophoresis of mtDNA cytochrome b gene sequences for animal species identification. Methods in molecular biology (Clifton, N.J.), 830, 321–329. https://doi.org/10.1007/978-1-61779-461-2_22

Pal, A., Banerjee, S., Batabyal, S., & Chatterjee, P. N. (2019). Mutation in Cytochrome B gene causes debility and adverse effects on health of sheep. Mitochondrion, 46, 393–404. https://doi.org/10.1016/j.mito.2018.10.003

Parson, W., Pegoraro, K., Niederstätter, H. et al. (2000). Species identification by means of the cytochrome b gene. Int J Leg Med **114**, 23–28. https://doi.org/10.1007/s004140000134

SEQ: Sequence generation. RDocumentation. (n.d.). https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/seq

Wrap a 1D ribbon of panels into 2D - facet_wrap. - facet_wrap • ggplot2. (n.d.). https://ggplot2.tidyverse.org/reference/facet_wrap.html

Wright, E.S. (2023, October 15). Classify Sequences - Bioconductor. https://www.bioconductor.org/packages/devel/bioc/vignettes/DECIPHER/inst/doc/ClassifySequences.pdf