Exploring mRNA Gene Identification in Salmoninae through K-Mer Features and Random Forest Classifiers

BINF6210 Assignment #2

Yuqi Li

2023-10-27

Contents

Introduction

The Cytochrome Oxidase I (COI) and Cytochrome b (CYTB) mitochondrial RNA (mRNA) genes are commonly used as molecular markers for species identification [1,2]. Both being protein-coding genes within the mitochondria, they share many biological and functional properties. These similarities necessitate precise classification techniques to accurately differentiate their nucleotide sequences. Having well-classified sequences may potentially lead to improved alignment and genome assembly outcomes in subsequent analyses. In the context of this study, the primary objective is to develop a robust classifier utilizing Random Forest algorithms to distinguish between the COI and CYTB gene groups. A central focus of this investigation revolves around K-Mer frequencies, defined as the frequency DNA sequences of length K. These features are renowned for their ability to capture intricate sequence patterns, making them invaluable in biological analyses. Previous research has demonstrated an inverse relationship between misclassification rates and K-Mer lengths, albeit with increased computational cost [3]. This report delved deeper into this correlation by training classifiers on K-Mer lengths of 1, 2, and 3, to observe corresponding changes in prediction accuracy.

In this study, we continue to examine the taxonomic group Salmoninae due to its ecological significance in both terrestrial and aquatic ecosystems. Nucleotide sequence data was sourced from the NCBI website (https://www.ncbi.nlm.nih.gov/) and analyzed in RStudio. This research commenced with detailed exploratory data analyses, followed by the development of Random Forest classifiers, and concluded with model validations. Through this report, our objectives are to 1) construct Random Forest classifiers to classify nucleotide sequences into COI or CYTB genes, and 2) investigate the relationship between prediction accuracy and K-Mer lengths.

Code Section 1: Exploratory Data Analysis (EDA)

1) Loading Library

```
# Loading library
library(Biostrings)
library(cvms)
library(ggimage)
library(ggnewscale)
library(ggplot2)
library(gridExtra)
library(grid)
library(randomForest)
library(rentrez)
library(rsvg)
library(tidyverse)
```

2) Data Acquisition

```
# Data acquisition from NCBI
search_result <- entrez_search(db = "nuccore",
  term = "Salmoninae[ORGN] AND (COI[Gene] OR CytB[Gene])", use_history = T)

# Fetch data from NCBI using helper function "Entrez_Functions.R"
source("Entrez_Functions.R")
FetchFastaFiles(searchTerm = "Salmoninae[ORGN] AND (COI[Gene] OR CytB[Gene])",
```

```
                seqsPerFile = 100, fastaFileName = "Salmon_fetch.fasta")

# Merge all fasta files and store it as a data frame
dfNCBI <- MergeFastaFiles(filePattern = "Salmon_fetch.fasta*")
```

3) Data Filtering & Exploration

```
dfSalmon <- dfNCBI %>%
  # filter out samples with missing sequence
  filter(!is.na(Sequence)) %>%
  # filter out sequences with more than 5% N
  filter(str_count(Sequence, "N") <= (0.05 * str_count(Sequence))) %>%
  # count the number of characters in each sequence, store in Sequence_Length variable
  mutate(Sequence_Length = nchar(Sequence)) %>%

  # extract words inside the parentheses, store in Gene_Name variable
  mutate(Gene_Name = word(str_extract(Title, "\\(\\w+\\)"), 1)) %>%
  # replace parentheses with nothing
  mutate(Gene_Name = str_replace_all(Gene_Name, "\\(|\\)", "")) %>%
  # convert all letter to upper case
  mutate(Gene_Name = str_to_upper(Gene_Name)) %>%
  # filter out samples of COI or CYTB genes
  filter(Gene_Name == "COI" | Gene_Name == "CYTB") %>%
  # extract column 2 within in the Title column, store in Genus_Name variable

  mutate(Genus_Name = word(Title, 2L)) %>%
  # filter selected genera
  filter(Genus_Name == "Brachymystax" |Genus_Name == "Hucho" |
         Genus_Name == "Oncorhynchus" | Genus_Name == "Parahucho" |
         Genus_Name == "Salmo" | Genus_Name == "Salvelinus" |
         Genus_Name == "Salvethymus") %>%

  # create a new variable called id
  mutate(id = row_number()) %>%
  # rearrange columns
  select("id", "Title", "Genus_Name", "Gene_Name", "Sequence", "Sequence_Length")
```

```
# Summary statistics
dfSalmon_COI <- dfSalmon %>%
  filter(Gene_Name == "COI")
summary(dfSalmon_COI$Sequence_Length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    51.0   648.2   652.0   655.8   652.0  1581.0
```

```
dfSalmon_CYTB <- dfSalmon %>%
  filter(Gene_Name == "CYTB")
summary(dfSalmon_CYTB$Sequence_Length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    53.0   866.0  1018.0   892.7  1141.0  1599.0
```

4) Data Exploration & Figures

```
dfSalmon_COI %>% ggplot(aes(x = Sequence_Length)) +
  geom_histogram(bins = 10, color = "#000000", fill = "mediumpurple") +
  ggtitle("Distribution of COI Gene Sequence Length in Salmoninae Samples") +
  xlab('Sequence Length (Nucleotides)') +
  ylab('Frequency') +
  # solid vertical lines represent the median
  geom_vline(aes(xintercept = median(Sequence_Length)),
             color = "#000000", linewidth = 1) +
    # dashed lines represent the 1st & 3rd quartiles
  geom_vline(aes(xintercept = quantile(Sequence_Length, probs = c(0.25))),
             color = "orange", linewidth = 1, linetype = "dashed") +
  geom_vline(aes(xintercept = quantile(Sequence_Length, probs = c(0.75))),
             color = "orange", linewidth = 1, linetype = "dashed") +
  theme(plot.title=element_text(size = 14))
```

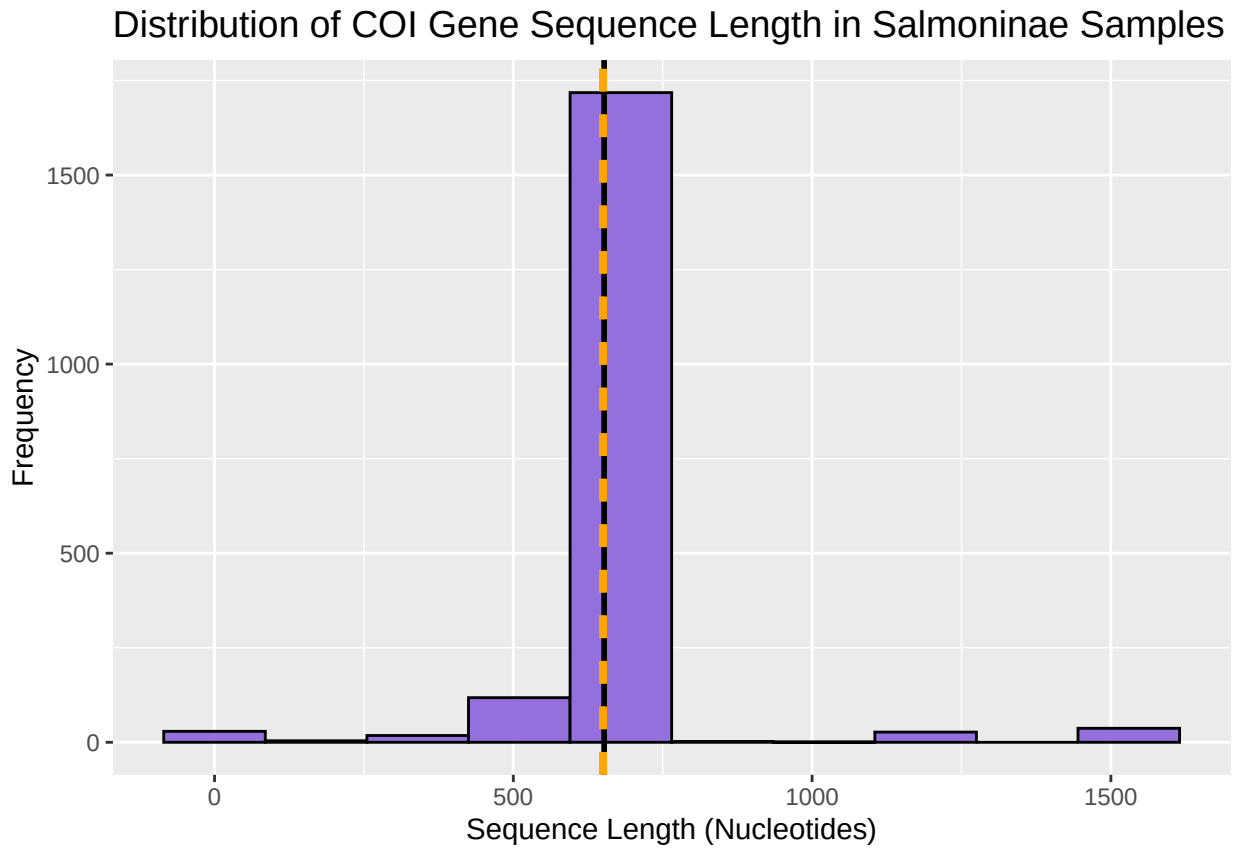

Figure 1: **Histogram Illustrating the Distribution of COI Gene Sequence Length in Salmoninae Samples**. The x-axis divides mRNA sequence lengths into 10 equal bins, ranging approximately from 0 to 1500 nucleotides. The y-axis illustrates frequency. Each bar represents the number of samples falling within a specific bin. The solid black vertical line denotes the median value of 652 nucleotides, and the two orange dashed lines signify the 1st (q1) and 3rd (q3) quartiles. The clustering of the three lines results from the proximity of their values. Notably, the majority of COI gene sequences span 648 to 652 nucleotides in length. (n = 1954)

```r
dfSalmon_CYTB %>% ggplot(aes(x = Sequence_Length)) +
  geom_histogram(bins = 10, color = "#000000", fill = "mediumpurple") +
  ggtitle("Distribution of CYTB Gene Sequence Length in Salmoninae Samples") +
  xlab('Sequence Length (Nucleotides)') +
  ylab('Frequency') +
  # solid lines represent the median
  geom_vline(aes(xintercept = median(Sequence_Length)),
             color = "#000000", linewidth = 1) +
  # dashed lines represent the 1st & 3rd quartiles
  geom_vline(aes(xintercept = quantile(Sequence_Length, probs = c(0.25))),
             color = "orange", linewidth = 1, linetype = "dashed") +
  geom_vline(aes(xintercept = quantile(Sequence_Length, probs = c(0.75))),
             color = "orange", linewidth = 1, linetype = "dashed") +
  theme(plot.title=element_text(size = 14))
```

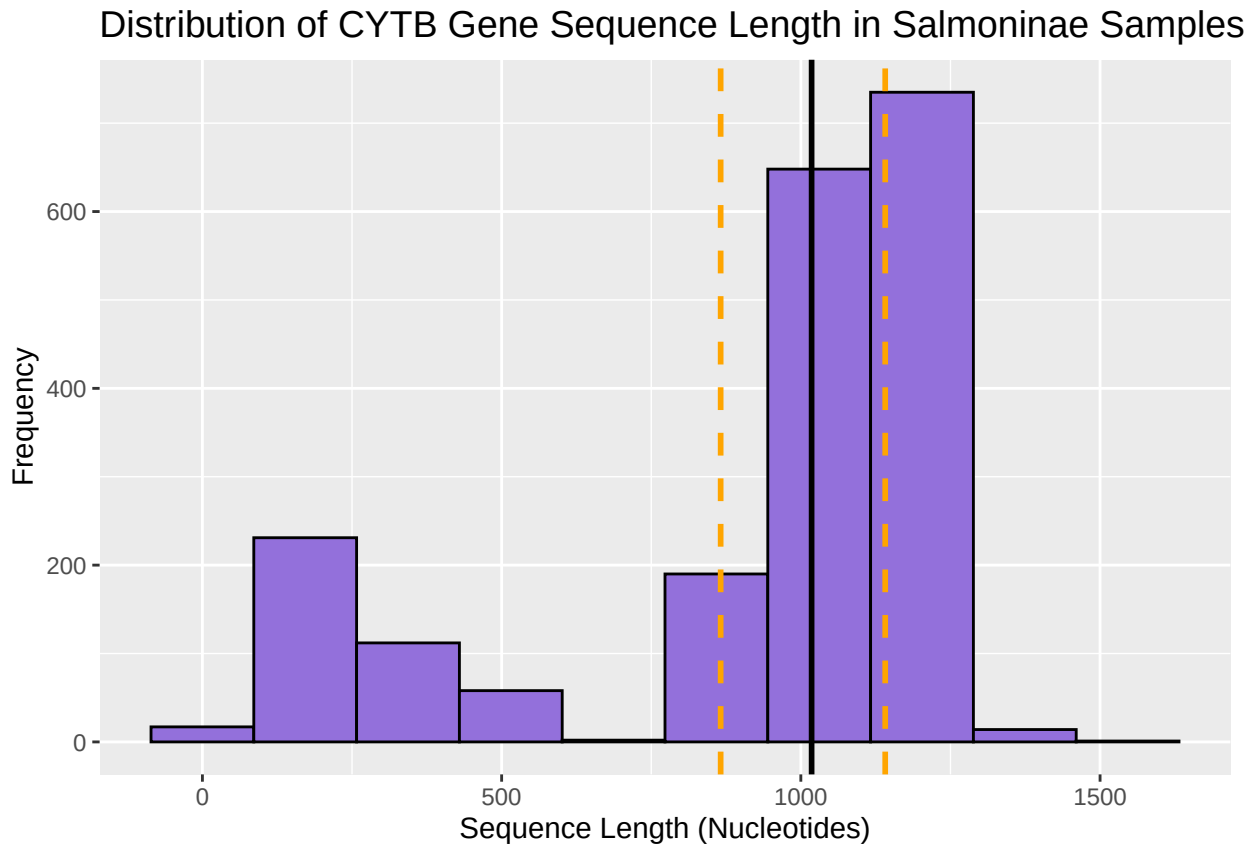

Figure 2: **Histogram Illustrating the Distribution of CYTB Gene Sequence Length in Salmoninae Samples**. The x-axis divides mRNA sequence lengths into 10 equal bins, ranging approximately from 0 to 1500 nucleotides. The y-axis illustrates frequency. Each bar represents the number of samples falling within a specific bin. The solid black vertical line denotes the median value of 1018 nucleotides, and the two orange dashed lines signify the 1st (q1) and 3rd (q3) quartiles. Notably, the majority of CYTB gene sequences span 866 to 1141 nucleotides in length. However, a substantial portion of shorter sequences spanning approximately 50 to 550 nucleotides also contribute to the overall distribution. (n = 2008)

5) Quality Control

```r
# Calculate the 1st and 3rd quartiles of COI sequences
q1_COI <- quantile(nchar(dfSalmon$Sequence[dfSalmon$Gene_Name == "COI"]),
                   probs = 0.25, na.rm = TRUE)
q3_COI <- quantile(nchar(dfSalmon$Sequence[dfSalmon$Gene_Name == "COI"]),
                   probs = 0.75, na.rm = TRUE)

# Calculate the 1st and 3rd quartiles of CYTB sequences
q1_CYTB <- quantile(nchar(dfSalmon$Sequence[dfSalmon$Gene_Name == "CYTB"]),
                   probs = 0.25, na.rm = TRUE)
q3_CYTB <- quantile(nchar(dfSalmon$Sequence[dfSalmon$Gene_Name == "CYTB"]),
                   probs = 0.75, na.rm = TRUE)

dfSalmon <- dfSalmon %>%
  # filter out sequences outside the interquartile range
  filter(str_count(Sequence) >= q1_COI & str_count(Sequence) <= q3_COI &
           Gene_Name == "COI" | str_count(Sequence) >= q1_CYTB &
           str_count(Sequence) <= q3_CYTB & Gene_Name == "CYTB")
```

Code Section 2: Random Forest Classifiers

1) Calculate Sequence Features

```
# Change Sequence in dfSalmon to DNAStringSet class
dfSalmon$Sequence <- DNAStringSet(dfSalmon$Sequence)

# Calculate the frequency of A, C, G, & T
dfSalmon <- cbind(dfSalmon, as.data.frame(
  letterFrequency(dfSalmon$Sequence, letters = c("A", "C", "G", "T"))))

# Calculate the proportions of A, C, G, & T
dfSalmon$Aprop <- (dfSalmon$A) / (dfSalmon$A + dfSalmon$T + dfSalmon$C + dfSalmon$G)
dfSalmon$Tprop <- (dfSalmon$T) / (dfSalmon$A + dfSalmon$T + dfSalmon$C + dfSalmon$G)
dfSalmon$Gprop <- (dfSalmon$G) / (dfSalmon$A + dfSalmon$T + dfSalmon$C + dfSalmon$G)
dfSalmon$Cprop <- (dfSalmon$C) / (dfSalmon$A + dfSalmon$T + dfSalmon$C + dfSalmon$G)

# Calculate the dinucleotide frequency
dfSalmon <- cbind(dfSalmon, as.data.frame(dinucleotideFrequency(dfSalmon$Sequence,
                                                                as.prob = TRUE)))

# Calculate the trinucleotide frequency
dfSalmon <- cbind(dfSalmon, as.data.frame(trinucleotideFrequency(dfSalmon$Sequence,
                                                                 as.prob = TRUE)))
```

2) Dataset Spliting

```
# Change Sequence in dfSalmon back to character class
dfSalmon$Sequence <- as.character(dfSalmon$Sequence)

# Set seed
set.seed(000)

# Sample 350 observations in each gene group into the validation dataset
dfValidation <- dfSalmon %>%
  group_by(Gene_Name) %>%
  sample_n(350)

# From the remaining observations, sample 789 observations in each gene group into
# the training dataset
dfTraining <- dfSalmon %>%
  filter(!id %in% dfValidation$id) %>%
  group_by(Gene_Name) %>%
  sample_n(789)
```

3) Random Forest Classifiers

```
# Building classifier using only mRNA sequence
classifier_1 <- randomForest::randomForest(x = dfTraining[, 5],
  y = as.factor(dfTraining$Gene_Name), ntree = 50, importance = TRUE)
classifier_1
```

```
##
## Call:
##  randomForest(x = dfTraining[, 5], y = as.factor(dfTraining$Gene_Name),      ntree = 50, importance =
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 0.82%
## Confusion matrix:
##      COI CYTB class.error
## COI  783    6 0.007604563
## CYTB   7  782 0.008871990
```

```
# Building classifier using A, C, T, and G frequency (K = 1)
classifier_2 <- randomForest::randomForest(x = dfTraining[, 11:14],
  y = as.factor(dfTraining$Gene_Name), ntree = 50, importance = TRUE)
classifier_2
```

```
##
## Call:
##  randomForest(x = dfTraining[, 11:14], y = as.factor(dfTraining$Gene_Name),      ntree = 50, importa
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.06%
## Confusion matrix:
##      COI CYTB class.error
## COI  789    0 0.000000000
## CYTB   1  788 0.001267427
```

```
# Building classifier using dinucleotide frequency (K = 2)
classifier_3 <- randomForest::randomForest(x = dfTraining[, 15:30],
  y = as.factor(dfTraining$Gene_Name), ntree = 50, importance = TRUE)
classifier_3
```

```
##
## Call:
##  randomForest(x = dfTraining[, 15:30], y = as.factor(dfTraining$Gene_Name),      ntree = 50, importa
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.06%
```

```
## Confusion matrix:
##      COI CYTB class.error
## COI  789    0 0.000000000
## CYTB   1  788 0.001267427
```

```
# Building classifier using trinucleotide frequency (K = 3)
classifier_4 <- randomForest::randomForest(x = dfTraining[, 31:94],
  y = as.factor(dfTraining$Gene_Name), ntree = 50, importance = TRUE)
classifier_4
```

```
##
## Call:
##  randomForest(x = dfTraining[, 31:94], y = as.factor(dfTraining$Gene_Name),      ntree = 50, importa
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 0.06%
## Confusion matrix:
##      COI CYTB class.error
## COI  789    0 0.000000000
## CYTB   1  788 0.001267427
```

4) Classifier Validation

```
# Using classifier 1, predict the classification of samples in the validation dataset
predictValidation <- predict(classifier_1, dfValidation[, 5])
# Create a confusion matrix in the table format
matrix_table <- table(observed = dfValidation$Gene_Name,
                      predicted = predictValidation)
# Store the table as a tibble
matrix_tibble <- as_tibble(matrix_table)
# Plot the tibble as a confusion matrix plot
plot_1 <- plot_confusion_matrix(matrix_tibble, target_col = "observed",
                                prediction_col = "predicted", counts_col = "n",
                                add_sums = TRUE, add_normalized = FALSE,
                                add_col_percentages = FALSE,
                                add_row_percentages = FALSE, palette = "Purples",
                                sums_settings = sum_tile_settings(palette = "Oranges",
                                                                  label = "Total"))
```

```
# Using classifier 2, predict the classification of samples in the validation dataset
predictValidation_2 <- predict(classifier_2, dfValidation[, 11:14])
# Create a confusion matrix in the table format
matrix_table_2 <- table(observed = dfValidation$Gene_Name,
                        predicted = predictValidation_2)
# Store the table as a tibble
matrix_tibble_2 <- as_tibble(matrix_table_2)
# Plot the tibble as a confusion matrix plot
plot_2 <- plot_confusion_matrix(matrix_tibble_2, target_col = "observed",
                                prediction_col = "predicted", counts_col = "n",
```

```
                                add_sums = TRUE, add_col_percentages = FALSE,
                                add_row_percentages = FALSE,
                                add_normalized = FALSE, palette = "Purples",
                                sums_settings = sum_tile_settings(palette = "Oranges",
                                                                  label = "Total"))
```

```
# Using classifier 3, predict the classification of samples in the validation dataset
predictValidation_3 <- predict(classifier_3, dfValidation[, 15:30])
# Create a confusion matrix in the table format
matrix_table_3 <- table(observed = dfValidation$Gene_Name,
                        predicted = predictValidation_3)
# Store the table as a tibble
matrix_tibble_3 <- as_tibble(matrix_table_3)
# Plot the tibble as a confusion matrix plot
plot_3 <- plot_confusion_matrix(matrix_tibble_3, target_col = "observed",
                                prediction_col = "predicted", counts_col = "n",
                                add_sums = TRUE, add_col_percentages = FALSE,
                                add_row_percentages = FALSE,
                                add_normalized = FALSE, palette = "Purples",
                                sums_settings = sum_tile_settings(palette = "Oranges",
                                                                  label = "Total"))
```

```
# Using classifier 4, predict the classification of samples in the validation dataset
predictValidation_4 <- predict(classifier_4, dfValidation[, 31:94])
# Create a confusion matrix in the table format
matrix_table_4 <- table(observed = dfValidation$Gene_Name,
                        predicted = predictValidation_4)
# Store the table as a tibble
matrix_tibble_4 <- as_tibble(matrix_table_4)
# Plot the tibble as a confusion matrix plot
plot_4 <- plot_confusion_matrix(matrix_tibble_4, target_col = "observed",
                                prediction_col = "predicted", counts_col = "n",
                                add_sums = TRUE, add_col_percentages = FALSE,
                                add_row_percentages = FALSE,
                                add_normalized = FALSE, palette = "Purples",
                                sums_settings = sum_tile_settings(palette = "Oranges",
                                                                  label = "Total"))
```

```r
# Adding title to confusion matrix plot
title_1 <- textGrob("Classifier 1: Sequence", gp = gpar(fontsize = 12, fontface = "bold"))
title_2 <- textGrob("Classifier 2: K = 1", gp = gpar(fontsize = 12, fontface = "bold"))
title_3 <- textGrob("Classifier 3: K = 2", gp = gpar(fontsize = 12, fontface = "bold"))
title_4 <- textGrob("Classifier 4: K = 3", gp = gpar(fontsize = 12, fontface = "bold"))

# Arrange confusion matrix plots and titles in a 2 x 2 grid
grid.arrange(arrangeGrob(plot_1, top = title_1),
             arrangeGrob(plot_2, top = title_2),
             arrangeGrob(plot_3, top = title_3),
             arrangeGrob(plot_4, top = title_4), ncol = 2)
```

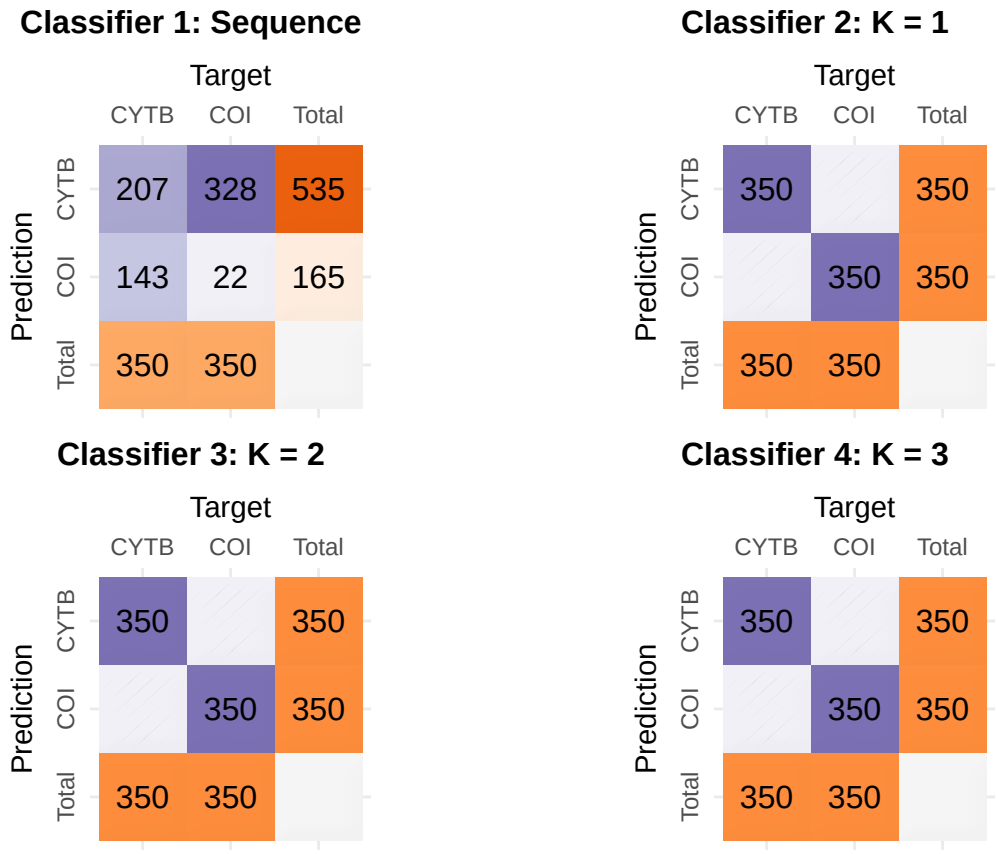

Figure 3: **Confusion Matrix Plots Illustrating the Performance of COI and CYTB Classifiers on Validation Dataset in Salmoninae Samples.** The x-axis represents the expected counts of COI and CYTB genes, while the y-axis indicates the predicted counts. Each plot illustrates the performance of a specific classifier on the validation dataset. True positive, false positive, true negative, and false negative counts are visualized, and coloured based on values. Darker shades of colour correspond to larger numerical values. Notably, classifiers using K-Mer features achieved 100% accuracy in gene classification, while the classifier utilizing solely mRNA sequences displayed a higher error rate. (n = 700)

Results and Discussion

In Code Section 2 Part 3), four classifiers were constructed using the training dataset. Classifier #1, built solely on mRNA sequences displayed the highest out-of-bag (OOB) error rate of 0.82. In contrast, classifiers #2, #3, and #4 built on K-Mer lengths of 1, 2, and 3 respectively, all exhibited a low OOB error rate of 0.06. In Part 4) of this section, the confusion matrix plots (Figure 3) provided a comprehensive view of the classifiers' performance against unseen data. Similarly, classifier #1 also exhibited the highest error rate. It only successfully classified 22 out of 350 COI sequences and 207 out of 350 CYTB sequences. On the contrary, classifiers #2 - 4 provided perfect separation of the mRNA sequences. Surprisingly, our results did not align with the existing knowledge that the length of K-Mer directly correlates with prediction accuracy. This could be due to the distinctiveness of COI and CYTB gene sequences in Salmoninae species. Despite their biological and functional similarities, having unique K-Mer frequencies may have led to a perfect split, defying the anticipated trend.

One key caveat of this study design was the limited sample size, with only 2464 sequence samples after extensive data filtration. This limited sample size may have contributed to the robustness of the classifiers. To enhance the credibility of our findings, future research could involve testing these classifiers on well-established reference databases or diverse taxonomic groups. In conclusion, our study demonstrated that Random Forest classifiers using K-Mer features showcased 100% classification accuracy regardless of the K-Mer lengths.

References

Scientific Literature:

1. Hebert, P. D., Cywinska, A., Ball, S. L. & deWaard, J. R. Biological identifications through DNA barcodes. Proceedings of the Royal Society of London. Series B: Biological Sciences 270, 313–321 (2003).
2. Yacoub, H. A., Fathi, M. M. & Mahmoud, W. M. DNA barcode through cytochrome b gene information of mtDNA in native chicken strains. Mitochondrial DNA 24, 528–537 (2013).
3. Meher, P. K., Sahu, T. K. & Rao, A. R. Identification of species based on DNA barcode using K-mer feature vector and random forest classifier. Gene 592, 316–324 (2016).

Websites:

1. https://www.datanovia.com/en/blog/awesome-list-of-657-r-color-names/
2. https://cran.r-project.org/web/packages/cvms/vignettes/Creating_a_confusion_matrix.html
3. ggplot2 barplots : Quick start guide - R software and data visualization http://www.sthda.com/english/wiki/ggplot2-violin-plot-quick-start-guide-r-software-and-data-visualization

R Packages:

1. A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.
2. Auguie B (2017). *gridExtra: Miscellaneous Functions for "Grid" Graphics*. R package version 2.3, https://CRAN.R-project.org/package=gridExtra.
3. Campitelli E (2023). *ggnewscale: Multiple Fill and Colour Scales in 'ggplot2'*. R package version 0.4.9, https://CRAN.R-project.org/package=ggnewscale.
4. H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
5. Ooms J (2023). *rsvg: Render SVG Images into PDF, PNG, (Encapsulated) PostScript, or Bitmap Arrays*. R package version 2.6.0, https://CRAN.R-project.org/package=rsvg.
6. Olsen L, Zachariae H (2023). *cvms: Cross-Validation for Model Selection*. R package version 1.6.0, https://CRAN.R-project.org/package=cvms.
7. Pagès H, Aboyoun P, Gentleman R, DebRoy S (2023). *Biostrings: Efficient manipulation of biological strings*. doi:10.18129/B9.bioc.Biostrings https://doi.org/10.18129/B9.bioc.Biostrings, R package version 2.68.1, https://bioconductor.org/packages/Biostrings.
8. R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.
9. Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the tidyverse." *Journal of Open Source Software*, *4*(43), 1686. doi:10.21105/joss.01686 https://doi.org/10.21105/joss.01686.
10. Winter, D. J. (2017) rentrez: an R package for the NCBI eUtils API The R Journal 9(2):520-526
11. Yu G (2023). *ggimage: Use Image in 'ggplot2'*. R package version 0.3.3, https://CRAN.R-project.org/package=ggimage.