BINF_6210_A5

Leon Edmiidz

2023-12-08

Introduction

Freshwater ecosystems are vital for society globally as they provide clean water and food, support livelihoods, and contribute significantly to the economy (Darwall *et al.*, 2018). These ecosystems are highly stressed due to various factors, one major cause being the harmful impacts of invasive species (Darwall *et al.*, 2018). Research on invasive fish in freshwater ecosystems has found that many species belong to the *Carassius* genus (Wouters *et al.*, 2012). DNA barcoding has emerged as an essential and effective method for species identification, which is crucial for detecting invasive species and analyzing the harm caused by their presence (Trivedi *et al.*, 2016). DNA barcoding relies on specific marker genes that vary in sequence from species to species. These marker genes cannot be too similar or different across species, therefore, there are only a small number of reliable marker genes. One gene identified as a reliable marker gene for barcoding animals is the mitochondrial *COI* gene (Trivedi *et al.*, 2016).

This study aims to compare the *COI* gene and the *Cytochrome b (CYTB)* gene to analyze whether *COI* is genuinely ideal as a DNA barcoding marker gene compared to other mitochondrial genes. Using sequence data of *COI* and *CYTB* from *Carassius* species and unsupervised machine learning to cluster the sequences. I hypothesize that each cluster of *COI* sequences will represent a particular species of *Carassius* and that the clusters for the *CYTB* sequences will not be representative of the species. If true, *COI* is a better genetic marker for DNA barcoding than *CYTB*. Dendrograms will be generated to visualize how the sequences clustered, and Silhouette plots will be used to analyze the validity of the consistency within the clusters for both *COI* and *CYTB*. As a secondary objective, I will compare Kimura's 2-parameters distance model to the Tamura-Nei model and the single-linkage clustering method to the neighbour-joining clustering method.

Data Set Description

This analysis will be conducted on data gathered from NCBI's Nucleotide database. This database consists of data from various sources, including submissions from individual researchers, sequencing projects, scientific literature, and other public databases. I used functions from the R package "rentrez" to gather the data on December 7th, 2023. Initially, the *COI* dataset consisted of 1094 samples, and the *CYTB* dataset had 2344 samples; however, much of the data was filtered out during the filtering steps, so not all of these entries were used in the analysis. The key variables of interest are the genus, species, process ID, and gene sequences. The genus, species, and process ID had to be extracted from the information within the entries. The gene sequences varied greatly in length and quality, therefore, the filtering steps were necessary for the analysis. Since the goal is to compare the genes, the *COI* and *CYTB* data were gathered into different datasets. The *CYTB* dataset contained many more entries than the *COI* dataset; therefore, the filtering steps were more stringent to reduce the number of entries closer to that of the *COI* dataset. The median values for the *COI* and *CYTB* sequence lengths were highly different, yet this is not an issue for the analysis since the datasets were not combined. **Citation** U.S. National Library of Medicine. (n.d.). Home - nucleotide - NCBI. National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/nucleotide/

Code Section 1

Part 1: Loading Packages

If the packages have not yet been installed, please uncomment the lines with "install.packages()".

```
# install.packages('tidyverse')
library(tidyverse)
# BiocManager::install('Biostrings')
library(Biostrings)
# install.packages('ggplot2')
library(ggplot2)
# install.packages('rentrez')
library(rentrez)
# install.packages('stringi')
library(stringi)
# install.packages('ape')
library(ape)
# install.packages('RSQLite')
library(RSQLite)
# install.packages('BiocManager')
library(BiocManager)
# BiocManager::install(c('Biostrings', 'muscle', 'DECIPHER'))
library(Biostrings)
library(muscle)
library(DECIPHER)
# install.packages('clValid')
library(clValid)
# install.packages('cluster')
library(cluster)
```

Part 2: Data Acquisition

We will gather our data from the Nucleotides database (nuccore) of NCBI. Let's take a look at the summary information about the database and check the searchable fields for gathering the data that we are looking for.

```
entrez_db_summary("nuccore")
entrez_db_searchable("nuccore")
```

We cannot download all the data in one go by using the "entrez_fetch" function, therefore we will use a loop to download 50 sequences at a time and write the data to a text file.

COI and *COX1* are two different names for the same gene. NCBI has data under both names, so we will gather data using both names.

First we need to check the number of hits for this search.

```
# First we need to check the number of hits for this search.
entrez_search(db = "nuccore", term = "(Carassius[ORGN] AND COI[GENE]) OR (Carassius[ORGN] AND COX1[GENE
    retmax = 548, use_history = TRUE)
# Next, we will run the same code (ensure retmax is set to the max number of
# hits) but give the output a name and create an object containing the
```

```r
# 'history' information that we need for the loop.
COI_nuccore <- entrez_search(db = "nuccore", term = "(Carassius[ORGN] AND COI[GENE]) OR (Carassius[ORGN
    retmax = 548, use_history = TRUE)

for (seq_start in seq(1, 548, 50)) {
    nuccore_fetch <- entrez_fetch(db = "nuccore", web_history = COI_nuccore$web_history,
        rettype = "fasta", retmax = 50, retstart = seq_start)
    write(nuccore_fetch, file = "COI_nuccore_fetch.fasta", append = TRUE, sep = "\n")
    writeLines(paste(seq_start + 51, "sequences downloaded"), "download_log.txt")
}

# The text file containing all of our COI data has been created. Let's read it
# back as a DNAStringSet.
COI_nuccore_string <- readDNAStringSet("COI_nuccore_fetch.fasta")
```

Let’s convert the data into a data frame and run some preparation steps.

```r
# The first column will be contain information regarding the data entries and
# the second column will contain the nucleotide sequences.
dfCOI <- data.frame(Information = names(COI_nuccore_string), nucleotides = paste(COI_nuccore_string))

# We will extract specific words from the information column to create new
# columns; Genus, Species_Name, and processid. Processid will be the species,
# followed by the processid. This will aid in our dendrogram visualization
# later.
dfCOI$Genus <- word(dfCOI$Information, 2)
dfCOI$Species_Name <- word(dfCOI$Information, 2, 3)
dfCOI$processid <- paste(word(dfCOI$Information, 3), word(dfCOI$Information, 1),
    sep = "_")

# Let's rearrange the columns.
dfCOI <- dfCOI[, c("Information", "processid", "Genus", "Species_Name", "nucleotides")]

# Take a look at the data frame to ensure that everything looks good.
view(dfCOI)

# Let's explore the data and see what we are working with.
dim(dfCOI)
unique(dfCOI$Species_Name)
unique(dfCOI$Genus)
summary(dfCOI)
summary(str_count(dfCOI$nucleotides))
```

We will repeat all the steps above to gather the data for the *CYTB* Gene.

```r
entrez_search(db = "nuccore", term = "Carassius[ORGN] AND CYTB[GENE]", retmax = 1173,
    use_history = TRUE)
CYTB_nuccore <- entrez_search(db = "nuccore", term = "Carassius[ORGN] AND CYTB[GENE]",
    retmax = 1173, use_history = TRUE)

for (seq_start in seq(1, 1173, 50)) {
    nuccore_fetch <- entrez_fetch(db = "nuccore", web_history = CYTB_nuccore$web_history,
        rettype = "fasta", retmax = 50, retstart = seq_start)
    write(nuccore_fetch, file = "CYTB_nuccore_fetch.fasta", append = TRUE, sep = "\n")
    writeLines(paste(seq_start + 51, "sequences downloaded"), "download_log.txt")
}

CYTB_nuccore_string <- readDNAStringSet("CYTB_nuccore_fetch.fasta")

dfCYTB <- data.frame(Information = names(CYTB_nuccore_string), nucleotides = paste(CYTB_nuccore_string)

dfCYTB$Genus <- word(dfCYTB$Information, 2)
dfCYTB$Species_Name <- word(dfCYTB$Information, 2, 3)
dfCYTB$processid <- paste(word(dfCYTB$Information, 3), word(dfCYTB$Information, 1),
    sep = "_")

dfCYTB <- dfCYTB[, c("Information", "processid", "Genus", "Species_Name", "nucleotides")]

# Once again, let's take a look at the new data frame and explore the data to
# make sure everything looks good.
view(dfCYTB)

# Explore data
dim(dfCYTB)
unique(dfCYTB$Species_Name)
unique(dfCYTB$Genus)
summary(dfCYTB)
summary(str_count(dfCYTB$nucleotides))
```

Part 3: Script Variables

Let's set some variables. If we need to make certain changes to our analysis, all we need to do is change these variables.

```
# We will define the acceptable amount of missing data (Ns) for our analysis.
missing.data <- 0.01

# We will define the acceptable range of sequence length variability for our
# analysis. Sequences with lengths that deviate further than this number from
# the median sequence length will be removed from the analysis.
length.var <- 50

# We will specify the model of molecular evolution we will use for our distance
# matrix.
chosen.model <- "K80"

# We will define the clustering threshold for our analysis.
clustering.threshold <- 0.03

# We will define the method of clustering for our analysis.
clustering.method <- "single"
```

###Part 4: Data Filtration Let's start filtering the data.

```r
# We need to ensure that all the COI sequences in our analysis are roughly the
# same length (close to the median) to ensure that they align properly later
# on. We are also ensuring that they are high quality data and that duplicate
# sequences are removed from our analysis.
dfCOI_filtered <- dfCOI %>%
    filter(str_count(nucleotides, "N") <= (missing.data * str_count(nucleotides))) %>%
    filter(str_count(nucleotides) >= median(str_count(nucleotides)) - length.var &
        str_count(nucleotides) <= median(str_count(nucleotides)) + length.var) %>%
    distinct(nucleotides, .keep_all = TRUE)

# We will apply a stricter filtering criteria to the CYTB data to reduce the
# number of sequences closer to that of the COI dataset. First, we will filter
# out sequences with any 'N's'. I then played around with the numbers for the
# length variability to include roughly the same number of sequences as the COI
# dataset. Once again, we removed duplicate sequences.
dfCYTB_filtered <- dfCYTB %>%
    filter(!grepl("N", nucleotides)) %>%
    filter(str_count(nucleotides) >= median(str_count(nucleotides)) - 20 & str_count(nucleotides) <=
        median(str_count(nucleotides)) + 15) %>%
    distinct(nucleotides, .keep_all = TRUE)
```

Let’s explore the filtered data.

```r
# Let's check for the data for COI
dim(dfCOI_filtered)
unique(dfCOI_filtered$Species_Name)
unique(dfCOI_filtered$Genus)
sum(is.na(dfCOI_filtered$nucleotides))
sum(str_count(dfCOI_filtered$nucleotides, "-"))
sum(str_count(dfCOI_filtered$nucleotides, "N"))
summary(str_count(dfCOI_filtered$nucleotides))
view(dfCOI_filtered)

# Let's check for the data for CYTB
dim(dfCYTB_filtered)
unique(dfCYTB_filtered$Species_Name)
unique(dfCYTB_filtered$Genus)
sum(is.na(dfCYTB_filtered$nucleotides))
sum(str_count(dfCYTB_filtered$nucleotides, "-"))
sum(str_count(dfCYTB_filtered$nucleotides, "N"))
summary(str_count(dfCYTB_filtered$nucleotides))
```

We need to remove the rows with unknown species and those that claim “UNVERIFIED:” in the information column from the *CYTB* data frame. “UNVERIFIED:” does not relate to the genus, however it was extracted into the Genus column, so we will use this to filter out those rows.

```r
dfCYTB_filtered <- dfCYTB_filtered %>%
    filter(Genus != "UNVERIFIED:") %>%
    filter(Species_Name != "Carassius sp.")

# Let's make sure that the unwanted rows were properly filtered out.
dim(dfCYTB_filtered)
unique(dfCYTB_filtered$Species_Name)
unique(dfCYTB_filtered$Genus)
view(dfCYTB_filtered)
```

These histograms help to visualize that the filtering steps filtered the data properly.

```r
# Let's create histograms to analyze the frequency distributions of the
# sequence lengths.
COI_median_length <- median(nchar(dfCOI_filtered$nucleotides))

ggplot(dfCOI_filtered, aes(x = nchar(nucleotides))) + geom_histogram(binwidth = 10,
    fill = "blue", color = "blue") + geom_vline(xintercept = COI_median_length, color = "black",
    linetype = "dashed", size = 0.5) + labs(title = "Distribution of Sequence Lengths for the COI Datas
    x = "Sequence Length", y = "Frequency") + theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
CYTB_median_length <- median(nchar(dfCYTB_filtered$nucleotides))

ggplot(dfCYTB_filtered, aes(x = nchar(nucleotides))) + geom_histogram(binwidth = 10,
    fill = "blue", color = "blue") + geom_vline(xintercept = CYTB_median_length,
    color = "black", linetype = "dashed", size = 0.5) + labs(title = "Distribution of Sequence Lengths 
    x = "Sequence Length", y = "Frequency") + theme(plot.title = element_text(hjust = 0.5))
```

Main Software Tools Description

The primary software tool used for this analysis was the “Treeline” function from the DECIPHER package. This function was used to cluster the sequences based on the distance matrices. Although I considered using the FactoMineR package to run a PCA as an alternative analysis, I chose this function as it is a very user-friendly tool. One of the tool’s strengths is its easy customizability regarding the clustering method and threshold. The dendrogram created from this tool was also easy to customize, allowing for adequate visualization of the clusters. A challenge I encountered with this tool was that the output needed to be reformatted to generate the Dunn and Silhouette indices. Although the output could not be directly used to generate the indices, several alterations allowed the output to be applied to functions to create the indices and the Silhouette plot. The other software tools that greatly contributed to the analysis include the Tidyverse, Biostring, ape, and muscle packages. To build upon the outputs from the “Treeline” tool, I used the “dunn” function from the clValid package and the “silhouette” function from the cluster package. The Dunn index allowed me to test the quality of the clusters and compare the models used for the distance matrices and the clustering methods. **Citation** Growing phylogenetic trees with treeline - bioconductor. (n.d.). https://bioconductor.org/packages/release/bioc/vignettes/DECIPHER/inst/doc/GrowingTrees.pdf

Code Section 2

###Part 5: Alignment Now that our data has been filtered, let’s convert the format to prepare for alignment.

```r
# We created the 'processid' column earlier for this part. It will be used as
# the unique identifier for the data. First for COI
dfCOI_filtered <- as.data.frame(dfCOI_filtered)
dfCOI_filtered$nucleotides <- DNAStringSet(dfCOI_filtered$nucleotides)
names(dfCOI_filtered$nucleotides) <- dfCOI_filtered$processid

# Next for CYTB
dfCYTB_filtered <- as.data.frame(dfCYTB_filtered)
dfCYTB_filtered$nucleotides <- DNAStringSet(dfCYTB_filtered$nucleotides)
names(dfCYTB_filtered$nucleotides) <- dfCYTB_filtered$processid
```

We will now align the sequences by using tools from the muscle package.

```r
# Let's align the sequences for COI (default settings).
dfCOI.alignment <- DNAStringSet(muscle::muscle(dfCOI_filtered$nucleotides))

# Let's align the sequences for CYTB (default settings).
dfCYTB.alignment <- DNAStringSet(muscle::muscle(dfCYTB_filtered$nucleotides))
```

###Part 6: Clustering We will start by converting the format of the data once again to prepare for clustering.

```r
# We will convert the data to a DNAbin.
dnaBin.COI <- as.DNAbin(dfCOI.alignment)
dnaBin.CYTB <- as.DNAbin(dfCYTB.alignment)
```

Model settings are very important and can change the results of the analysis. We can easily explore different model settings by changing the “chosen.model” variable that we set earlier.

```r
# Let's create the distance matrices.
COI_distanceMatrix <- dist.dna(dnaBin.COI, model = chosen.model, as.matrix = TRUE,
    pairwise.deletion = TRUE)

CYTB_distanceMatrix <- dist.dna(dnaBin.CYTB, model = chosen.model, as.matrix = TRUE,
    pairwise.deletion = TRUE)
```

Let’s explore the distance matrices a little to get an idea of what they look like.

```r
head(COI_distanceMatrix)
head(CYTB_distanceMatrix)
```

Using the clustering parameters that we set earlier, we will now cluster.

```r
# We are clustering by using our settings defined earlier for clustering method
# and clustering cutoff.
clusters.COI <- DECIPHER::TreeLine(myDistMatrix = COI_distanceMatrix, method = clustering.method,
    cutoff = clustering.threshold, showPlot = TRUE, type = "both", verbose = TRUE)
```

```r
clusters.CYTB <- DECIPHER::TreeLine(myDistMatrix = CYTB_distanceMatrix, method = clustering.method,
    cutoff = clustering.threshold, showPlot = TRUE, type = "both", verbose = TRUE)
```

Let’s explore these objects

```
clusters.COI
clusters.CYTB
```

Let’s count the number of unique clusters that are assigned to the specimens.

```
length(unique(unlist(clusters.COI[[1]][1])))
length(unique(unlist(clusters.CYTB[[1]][1])))
```

We will now create dendrograms to visualize the clusters. Since we included the species name in the processid, we can analyze how sequences from different species clustered.

```
# Plot the dendrogram for COI data
plot(as.hclust(clusters.COI[[2]]), main = "Dendrogram for COI Data", cex = 0.35,
    sub = NA, xlab = "Species and ProcessID", ylab = "Height (Distance)")
abline(h = 0.015, col = "red", lty = 2)
```

```
# Plot the dendrogram for CYTB data
plot(as.hclust(clusters.CYTB[[2]]), main = "Dendrogram for CYTB Data", cex = 0.35,
    sub = NA, xlab = "Species and ProcessID", ylab = "Height (Distance)")
abline(h = 0.015, col = "red", lty = 2)
```

###Part 7: Cluster Strength Indices We will analyze the quality of our clusters via the Dunn Index

```
# Let's format the clusters properly for the 'dunn' function. We need named
# vectors.
COI_cluster_vector <- unlist(clusters.COI[[1]])
names(COI_cluster_vector) <- rownames(clusters.COI[[1]])
dunn(COI_distanceMatrix, COI_cluster_vector)

CYTB_cluster_vector <- unlist(clusters.CYTB[[1]])
names(CYTB_cluster_vector) <- rownames(clusters.CYTB[[1]])
dunn(CYTB_distanceMatrix, CYTB_cluster_vector)
```

Now we will analyze the consistency within the clusters via a Silhouette index.

```
# Let's format the clusters properly for the 'silhouette' function by
# extracting cluster assignments as integers for the data.
cluster_COI <- as.integer(clusters.COI[[1]][[1]])
cluster_CYTB <- as.integer(clusters.CYTB[[1]][[1]])

# Let's calculate Silhouette Indices for the data
silhouette_COI <- silhouette(cluster_COI, COI_distanceMatrix)
silhouette_CYTB <- silhouette(cluster_CYTB, CYTB_distanceMatrix)
```

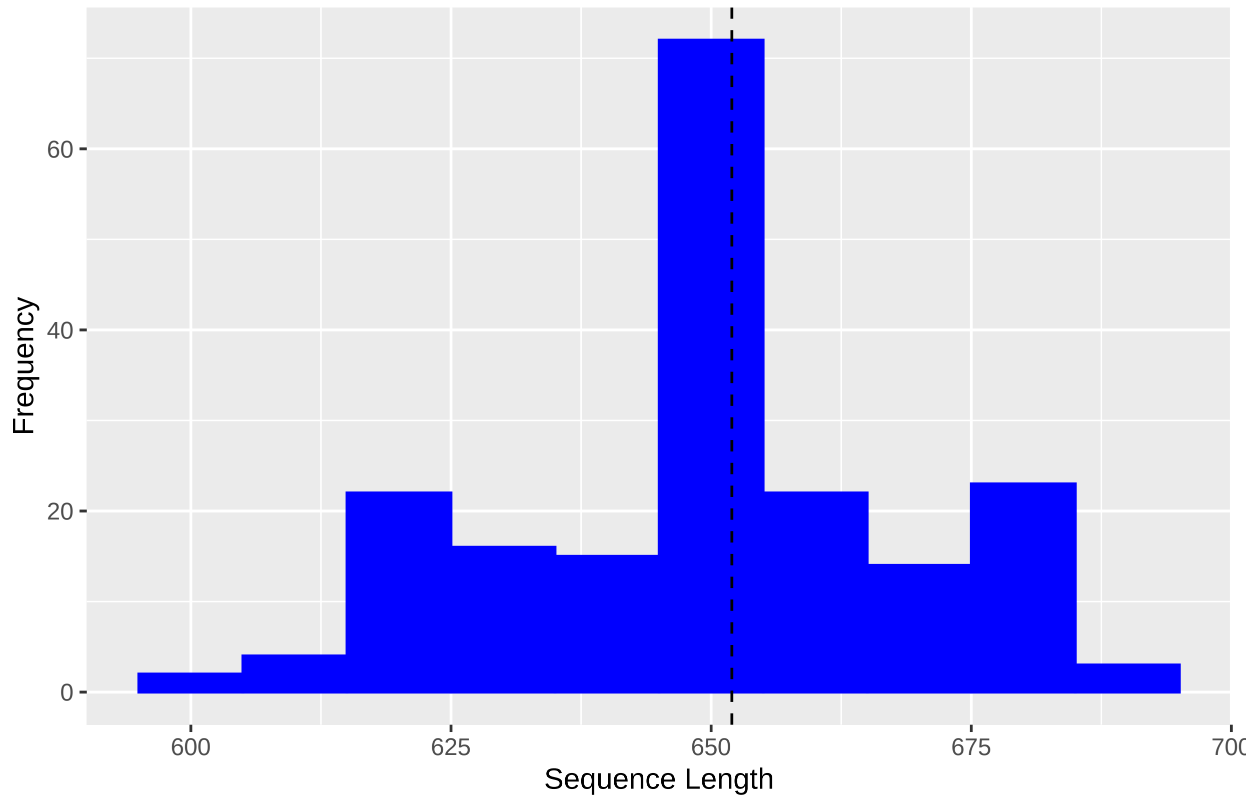
We will create Silhouette plots to visualize the validity of the consistency of the clusters.

```
# Silhouette plot for COI data
my_colors_COI <- c("red", "green", "blue")
plot(silhouette_COI, col = my_colors_COI[cluster_COI], main = "Silhouette Plot for COI Data")
abline(h = mean(silhouette_COI[, "sil_width"]), col = "red", lty = 2)
```

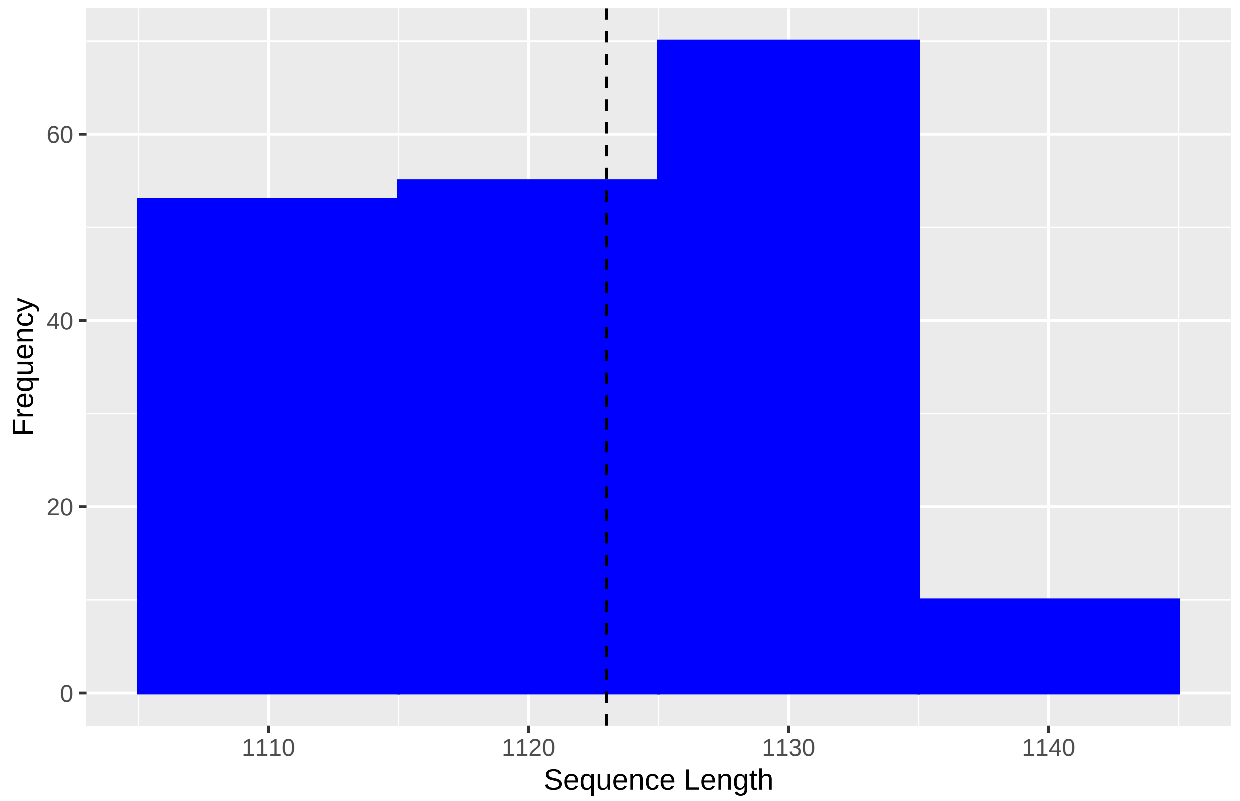
```r
# Silhouette plot for CYTB data
my_colors_CYTB <- c("magenta", "green", "red", "orange", "black", "blue")
plot(silhouette_CYTB, col = my_colors_CYTB[cluster_CYTB], main = "Silhouette Plot for CYTB Data")
abline(h = mean(silhouette_CYTB[, "sil_width"]), col = "red", lty = 2)
```

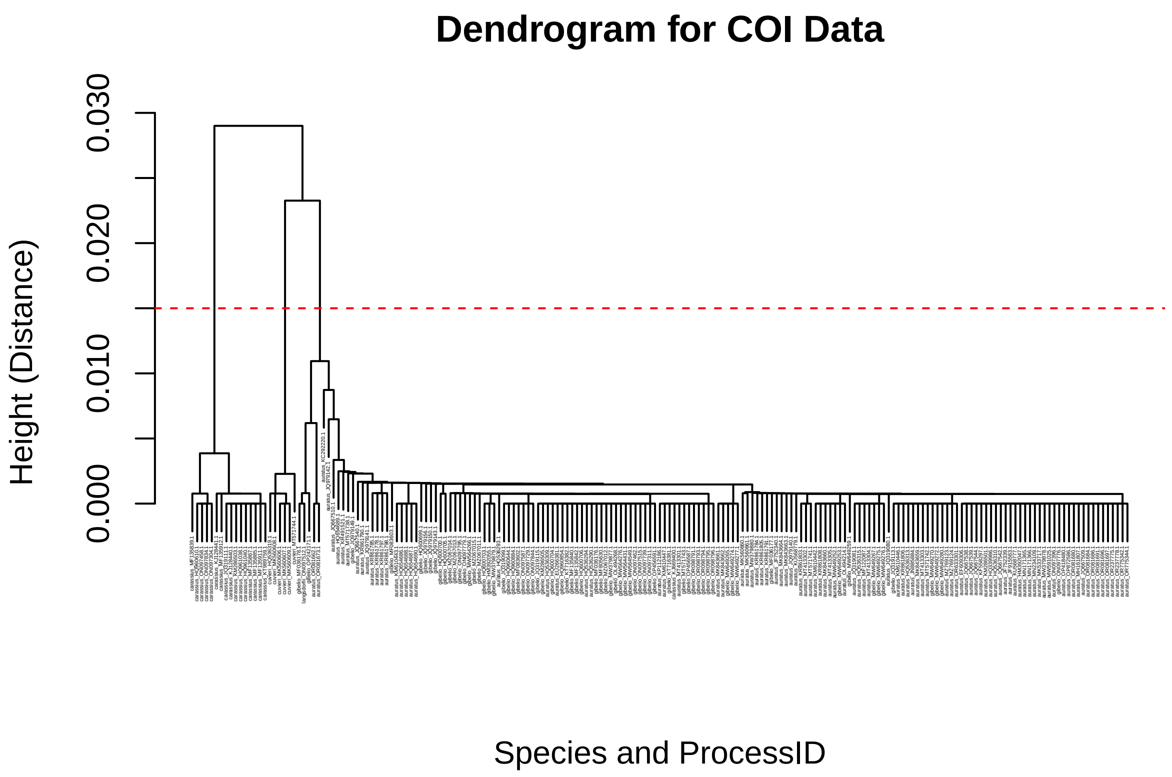
Visualizations

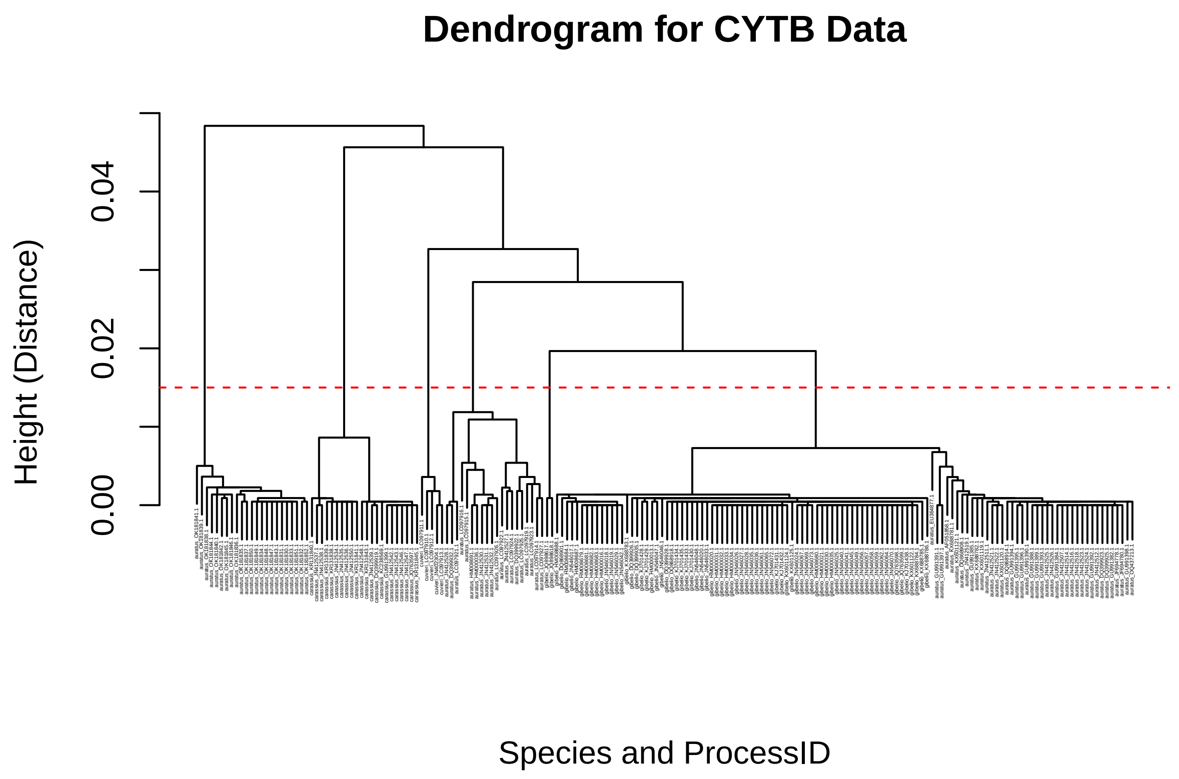
Distribution of Sequence Lengths for the COI Dataset After Filtering

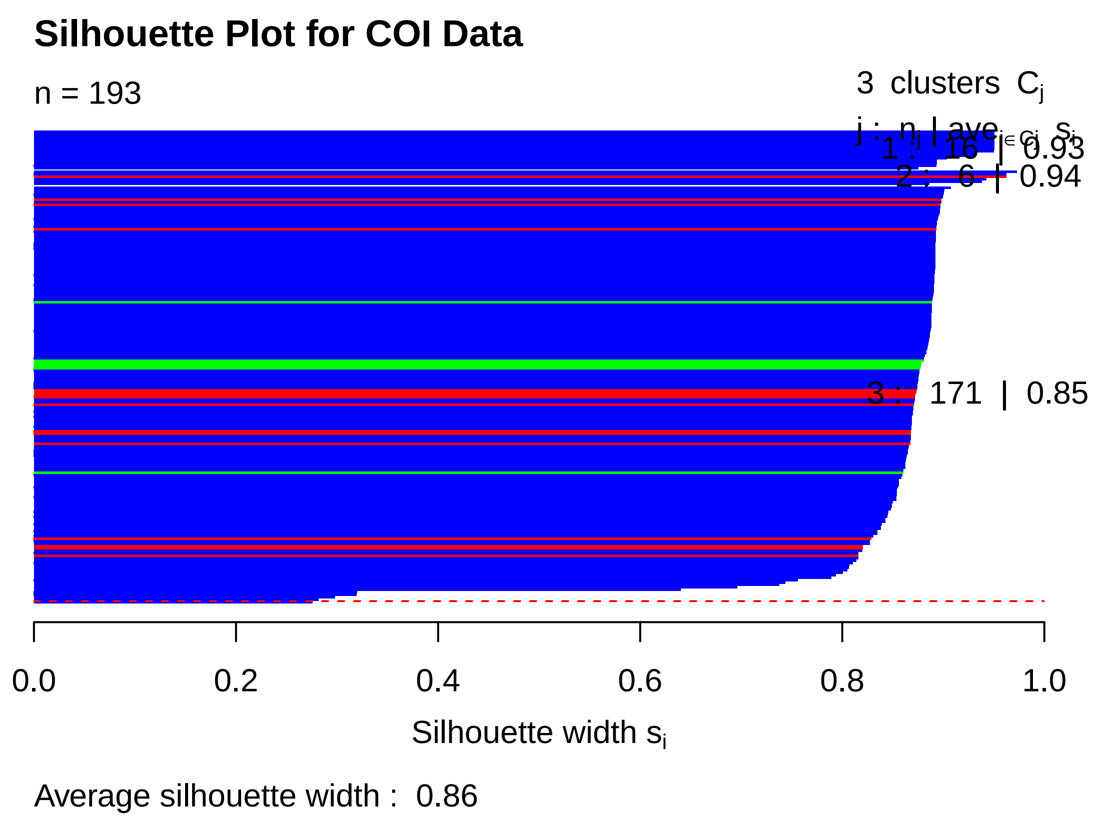


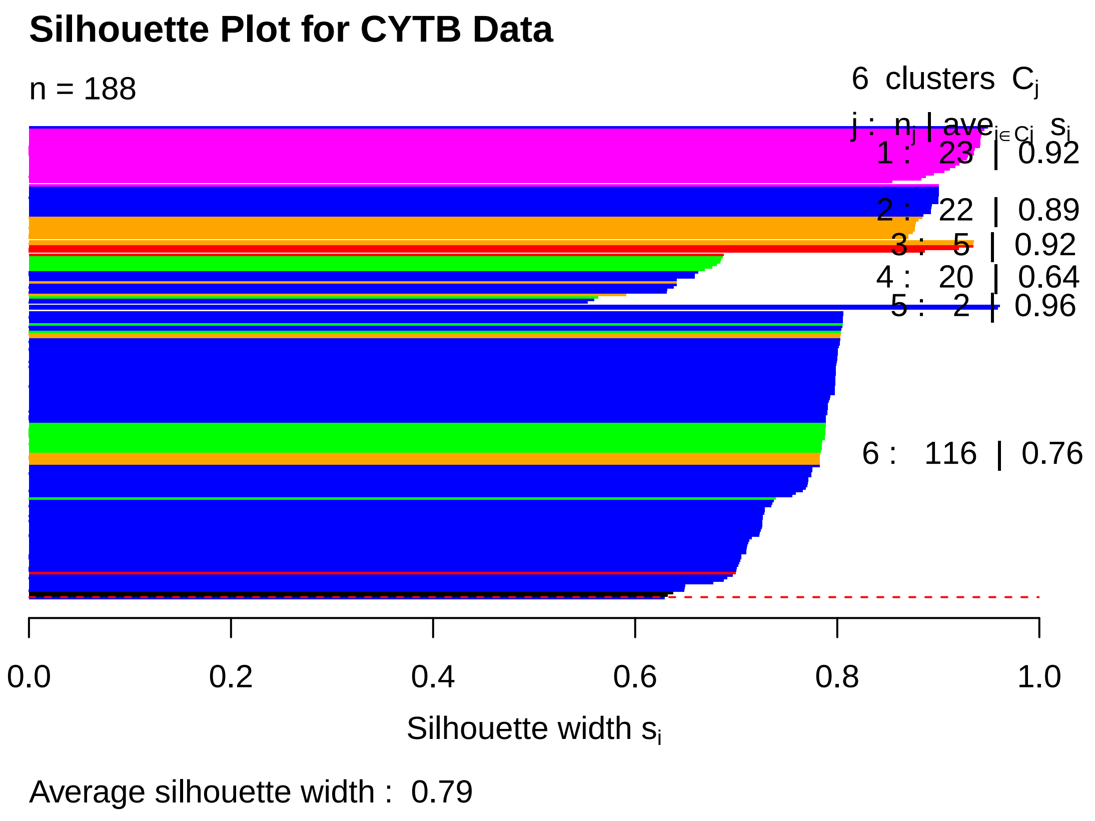
Distribution of Sequence Lengths for the CYTB Dataset After Filtering




Dendrogram for COI Data
Height (Distance)
0.000
0.010
0.020
0.030
Species and ProcessID


Dendrogram for CYTB Data
Height (Distance)
0.00
0.02
0.04
Species and ProcessID


Silhouette Plot for COI Data
n = 193
3 clusters C_j
$j : n_j \mid ave_{i \in Cj}\ s_i$
1 : 16 | 0.93
2 : 6 | 0.94
3 : 171 | 0.85
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.86



Results and Discussion

Investigating the distinct clustering patterns exhibited by the *COI* and *CYTB* genes within the *Carassius* fish species unveils divergences in evolutionary signals encoded by these genes, assessing their suitability as molecular markers for taxonomic classification and deciphering nuanced evolutionary pressures and serves as an exploration of molecular taxonomy. First, I should specify that I used Kimura's 2-parameters distance model and single-linkage clustering as these parameters fit better than their counterparts. The dendrograms indicate that *COI* is a better DNA barcoding marker than *CYTB*. The clusters of *COI* overall represented the different species of *Carassius*, meanwhile, the cluster patterns of *CYTB* were far more random and contained more clusters than there were species. The analysis results were expected, yet a key caveat must be addressed.

The filter *COI* dataset consists of five species, yet only three major clusters were identified. The first cluster consists exclusively of *Carassius carassius*, and the second consists exclusively of *Carassius cuvieri*. The caveat lies in the third cluster, consisting mainly of *Carassius auratus* and *Carassius gibelio*, one of *Carassius langsdorfii* and one of *Carassius carassius*. This could indicate that *COI* is an insufficient marker for identifying *auratus*, *gibelio*, and *langsdorfi*. However, *gibelio* and *langsdorfi* are often classified as intraspecies of *Carassius auratus* (Wouters *et al.*, 2012). This means that the three may, in some cases, be considered as the same species, *Carassius auratus*. This explains the highly similar sequences of *COI* among these three intraspecies and reflects why they were also clustered together. Only one sequence of *langsdorfi* was included in the analysis due to sample size limitation, and the one *Carassius carassius* sequence clustered in the third cluster was likely an outlier.

This research could be expanded to analyze more genes, potentially identifying more genes that could be used as genetic markers for DNA barcoding. The sample size for this analysis was relatively small and only analyzed five different species; therefore, running the same analysis with a larger dataset and more species would be very interesting. Although this analysis showed that *CYTB* is not a sufficient marker gene for the *Carassius* genus, it may serve as a sufficient marker gene for species of other genera. This project could also be expanded to study marker genes in other kingdoms, such as plants. The *COI* gene is considered the DNA barcoding marker gene only for animals. There are several candidates for DNA barcode genes for plants, including *MatK*, *RbcL*, *TrnH-psbA*, and *ITS* (Li *et al.*, 2014). This project can potentially discover new candidate DNA barcode genes for plants. Overall, I would love to expand on this project to analyze more genera and genes while using a larger dataset and analyze clustering patterns to understand DNA barcoding marker genes better.

References

Darwall, W., Bremerich, V., De Wever, A., Dell, A. I., Freyhof, J., Gessner, M. O., Grossart, H., Harrison, I., Irvine, K., Jähnig, S. C., Jeschke, J. M., Lee, J. J., Lu, C., Lewandowska, A. M., Monaghan, M. T., Nejstgaard, J. C., Patricio, H., Schmidt-Kloiber, A., Stuart, S. N., … Weyl, O. (2018). The Alliance for Freshwater Life A global call to unite efforts for Freshwater Biodiversity Science and Conservation. Aquatic Conservation: Marine and Freshwater Ecosystems, 28(4), 1015–1022. https://doi.org/10.1002/aqc.2958

Dunn Index. R. (n.d.). https://search.r-project.org/CRAN/refmans/clValid/html/dunn.html#:~:text=The%20Dunn%20Ind (Accessed on December 7th, 2023). Growing phylogenetic trees with treeline - bioconductor. (n.d.). https://bioconductor.org/packages/release/bioc/vignettes/DECIPHER/inst/doc/GrowingTrees.pdf (Accessed on December 7th, 2023).

Li, X., Yang, Y., Henry, R. J., Rossetto, M., Wang, Y., & Chen, S. (2014). Plant DNA barcoding: From gene to genome. Biological Reviews, 90(1), 157–166. https://doi.org/10.1111/brv.12104 Silhouette: Compute or extract silhouette information from clustering. RDocumentation. (n.d.). https://www.rdocumentation.org/packages/cluster/versions/2.1.4/topics/silhouette (Accessed on December 7th, 2023).

Trivedi, S., Aloufi, A. A., Ansari, A. A., & Ghosh, S. K. (2016). Role of DNA barcoding in Marine Biodiversity Assessment and Conservation: An Update. Saudi Journal of Biological Sciences, 23(2), 161–171. https://doi.org/10.1016/j.sjbs.2015.01.001

U.S. National Library of Medicine. (n.d.). Home - nucleotide - NCBI. National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/nucleotide/ (Accessed on December 7th, 2023). Wouters, J., Janson, S., Lusková, V., & Olsén, K. H. (2012). Molecular identification of hybrids of the invasive gibel carp Carassius auratus gibelio and crucian carp Carassius carassius in Swedish waters. Journal of Fish Biology, 80(7), 2595–2604. https://doi.org/10.1111/j.1095-8649.2012.03312.x