Unsupervised_machine_learning

Olusegun Odumosu

2022-12-16

Github link: https://github.com/odumosuo/6210_assignment5

Introduction

In this paper, we continue our study of the Ursidae family of bears, which is considered vulnerable according to the International Union for Conservation of Nature (IUCN). (Darin, 2015; UCN red list of threatened species., n.d.). In the first part of the project, we explored the distribution and species richness of Ursidae. In the second part, we used supervised machine learning to develop a classifier that could identify if a species is in the Ursidae family. In this final part, we apply unsupervised machine learning techniques to group the species within the Ursidae family. To accomplish this, we used two clustering algorithms: K-means and hierarchical clustering. We also performed both hierarchical clustering by alignment and hierarchical clustering by sequence features. To visualize the results, we employed both Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE). We used the mitochondrial gene cytochrome c oxidase I (COI) for our analyses, as it has been shown to be effective in the global bio identification system for animals (Hebert et al., 2003).

Both K-means clustering and hierarchical clustering algorithms partition data into different groups with a specified number of k clusters (Gatto, 2020). We know posteriori that the number of species used in analysis is four. However, to determine the number of clusters (k) in our K-means analysis, we employed both the elbow method and the average silhouette method. The elbow method involves plotting the total within sum of squares versus k and identifying the "elbow" position (Gatto, 2020), while the average silhouette method uses the k value with the highest average silhouette width (Kumar, 2020). Both methods have been demonstrated to be effective in determining the number of clusters. We also performed hierarchical clustering using the Jukes-Cantor 69 (JC69) method and Euclidean distance for alignment and feature-based methods, respectively. JC69 assumes that all substitutions have the same probability and that this probability is the same across all sites in the DNA sequence. To visualize the results of our clustering analyses, we compared both PCA and t-SNE. While t-SNE has been found to outperform PCA in some cases (Jizheng Yi et al., 2013; Azzalini et al., 2021), PCA preserves the global structure of data while t-SNE preserves local similarities (Azzalini et al., 2021). Overall, this paper aims to investigate techniques in unsupervised machine learning to determine the number of groups, cluster groups, and reduce dimensions

Description of Data Set

The first and second part of this project used information from the Barcode of life database (BOLD) and National Center for Biotechnology Information database (NCBI) respectively. Here we combined data from the Barcode of Life database (BOLD) and the National Center for Biotechnology Information database (NCBI) for analysis (Barcode of life database (BOLD), 2022; National Center for Biotechnology Information database (NCBI), 2022). To ensure the integrity of our data, we removed any duplicate sequences and identifiers. BOLD has the Genbank accession number for each record which is the first term of the header identifier for the fasta file gotten from NCBI. We also performed preprocessing on the nucleotide data, including removing all "-" characters and any leading or trailing sequences of "N". We excluded records where the percentage of remaining "N"s exceeded 1%, a cut-off adopted from Orton et al. (2019). We also filtered out records that fell outside the first and third quartile ranges of sequence length. To analyze the data, we used a variety of sequence features, including the proportion of A, T, G, and C nucleotides, dinucleotide and trinucleotide frequencies, and k-mer frequencies of length 4. These features were used in all analyses except for hierarchical clustering using alignment. In order to reduce noise, we removed species with fewer than 10 records. Finally, we used a subset of data comprising a random sample of 10 records from the remaining species for analysis.

Code Section 1 - Data Acquisition, Exploration, Filtering, and Quality Control

Setting up

install packages

```
install.packages("Rtsne")
install.packages("BiocManager")
BiocManager::install("PCAtools")
install.packages("ggalt")
install.packages("dendextend")
install.packages("ggpubr")
```

load package(S)

```
library(tidyverse)
library(vegan)
library(Biostrings)
library(rentrez)
library(cluster)
library(Rtsne)
library(vegan)
library(ape)
library(DECIPHER)
library(PCAtools)
library(ggalt)
library(dendextend)
library(ggrepel)
library(ggpubr)
```

Get present Working Directory. Previously set working directory with setwd(). Excluded runnable function to prevent error.

```
getwd()
```

Made functions

```
###make a function that retruns the total within sum of sqaures after a k-means analysis.
↪  k is number of clusters
get_tot_withinss <- function(k, numeric_matrix){
  cl <- kmeans(numeric_matrix, k, nstart = 10)
  return(cl$tot.withinss)
}

### make a function to get the average silhouette width for k clusters
get_average_sil <- function(k, numeric_matrix){
  clust <- kmeans(numeric_matrix, k, nstart = 10)
  silh <- silhouette(clust$cluster, dist(numeric_matrix))
  return(mean(silh[, 3]))#get the average fo the silhouette width
}


### function to plot PCA biplot and choose clustering method. clustering method is column
↪  names of PCA metada
create_PCA_plot <- function(title_of_plot, clustering){
  PCAtools::biplot(pcatools_pca, lab = rownames(metadata_pca),
                   colby = {{clustering}}, pointSize = 3,
                   colkey = c("red", "blue", "green", "black"),
                   title = title_of_plot, titleLabSize = 20,
                   shape = "species_name", legendPosition = "right",
                   encircle = TRUE, encircleFill = TRUE)
}



### function to plot t-SNE analysis coloured by species name
create_tSNE_plot <- function(title_of_plot, rtsne){
  tsne_plot <- data.frame(x = {{rtsne$Y[ , 1]}}, y = {{rtsne$Y[ , 2]}}, species_name =
↪  as.factor(data_for_analysis$species_name))
  ggplot(tsne_plot, aes(x = x, y = y)) +
    geom_encircle(alpha = 0.2, aes(group = species_name, fill = species_name)) +
    geom_point(aes(color = species_name)) +
    labs(title = title_of_plot,  x = "Dimension 1", y = "Dimension 2") +
    theme(plot.title = element_text(hjust = 0.5, size = 20), legend.position = "right") +
    scale_size(range = c(0.01,0.01))
}
```

Get Bold information

```
##Get Bold file from online
Bold_ursidae <-
↪  read_tsv("http://www.boldsystems.org/index.php/API_Public/combined?taxon=Ursidae&format=tsv")
#save Bold file to current working directory
write_tsv(Bold_ursidae, "Bold_ursidae_data.txt")
head(Bold_ursidae)
Bold_ursidae <- read_tsv("Bold_ursidae_data.txt")
##Explore variables in the dataframe
names(Bold_ursidae)
##Create a subset of interested variables from Bold_ursidae to form into new tibble
↪  called Bold_ursidae2. Only take rows where markercode is COI-5P
Bold_ursidae2 <- Bold_ursidae[ , c("genbank_accession", "markercode", "genus_name",
↪  "species_name", "nucleotides")] %>%
  filter(markercode == "COI-5P")
#check to see that only COI-5p markercodes are present
unique(Bold_ursidae2$markercode)
#remove markercode column
Bold_ursidae2 <- Bold_ursidae2[, -2]
##change name of genbank_accession to identifier for simplicity
names(Bold_ursidae2) <- c("identifier", "genus_name", "species_name", "nucleotides")
names(Bold_ursidae2)
head(Bold_ursidae2)
## check for NA's in variables
sum(is.na(Bold_ursidae2))
## remove records with NA's for identifier species_name and nucleotides
Bold_ursidae2 <- Bold_ursidae2 %>%
  filter(!is.na(identifier)) %>%
  filter(!is.na(species_name)) %>%
  filter(!is.na(nucleotides))
## check new number of NA's
sum(is.na(Bold_ursidae2))
```

Get NCBI information

```
##search and fetch needed information from nuccore database
#database search of nuccore. Organism is ursidae, gene is (COI or COX1) and sequence
↪  length is between 500-1000. Web history rather than retmax because of the large
↪  amounts of sequences from NCBI. Used by setting "use_history" argument to TRUE.
dbsearch_nuccore_ursidae <- entrez_search(db = "nuccore", term = "((ursidae[Organism])
↪  AND 500:1000[Sequence Length]) AND (COI OR COX1)", use_history = T)
dbsearch_nuccore_ursidae
#get number of returned searches
length(dbsearch_nuccore_ursidae$ids)
#get count of all search results
dbsearch_nuccore_ursidae$count
#fetch fasta files of the database search using web_history to fetch
fetch_ursidae <- entrez_fetch(db = "nuccore", web_history =
↪  dbsearch_nuccore_ursidae$web_history, rettype = "fasta")
class(fetch_ursidae)
fetch_ursidae
#write fasta to hard drive and separate fields by \n (comes before identifier in string)
write(fetch_ursidae, "ursidae_fetch.fasta", sep = "\n")
```

Make dataframe with needed information

```
#read fasta file back as DNA StringSet
DNAStringset_ursidae <- readDNAStringSet("ursidae_fetch.fasta")
class(DNAStringset_ursidae)
head(names(DNAStringset_ursidae))
#convert information from DNAStringset to dataframe for further manipulation. Naming the
↪  column for the header as header_identifier and the columns for the sequences as
↪  nucleotides
```

```
NCBI_ursidae <- data.frame( header_identifier = names(DNAStringset_ursidae), nucleotides
↪  = paste(DNAStringset_ursidae))
head(NCBI_ursidae)
#make a new column called species name. The species name are the second and third terms
↪  in the header header_identifier
NCBI_ursidae$species_name <- word(NCBI_ursidae$header_identifier, 2L, 3L)
#get number of species
length(unique(NCBI_ursidae$species_name))
#make a new column called identifier for a simpler identifier that is not as long as the
↪  header. Take the first term. removing the .1 at the end of the term to compare with
↪  BOLD info.
NCBI_ursidae$identifier <- str_remove(word(NCBI_ursidae$header_identifier, 1L), ".1")
#remove longer header identifier column
NCBI_ursidae <- NCBI_ursidae[, -1]
#create new column for genus names
NCBI_ursidae$genus_name <- word(NCBI_ursidae$species_name, 1L)
#get number of genus
length(unique(NCBI_ursidae$genus_name))
#rearrange columns to match Bold data
NCBI_ursidae <- NCBI_ursidae[, c("identifier", "genus_name", "species_name",
↪  "nucleotides")]
##check for NA's
sum(is.na(NCBI_ursidae))
```

Combine information and filter nucleotides

```
## Join Bold and NCBI information
combined_NCBI_and_Bold <- rbind(Bold_ursidae2, NCBI_ursidae)
```

Remove duplicated sequences from combined NCBI and BOLD

```
combined_NCBI_and_Bold <-
↪  combined_NCBI_and_Bold[!duplicated(combined_NCBI_and_Bold$identifier), ]
```

Clean nucleotides and remove records that have N's that make up more than 1% of original sequence

```
combined_NCBI_and_Bold <- combined_NCBI_and_Bold %>%
  #create new column while keeping the old column and remove leading "-" or "N"s
  mutate(clean_nucleotides = str_remove(nucleotides, "^[-N]+")) %>%
  #remove trailing "-" or "N"s
  mutate(clean_nucleotides = str_remove(clean_nucleotides, "[-N]+$")) %>%
  #remove all "-"s
  mutate(clean_nucleotides = str_remove_all(clean_nucleotides, "-+")) %>%
  #remove records that have "N"s that make up more than 1% of original sequence
  filter(str_count(clean_nucleotides, "N") <= (0.01 * str_count(nucleotides)))
```

Filter for sequence length

Look at distribution of data

```r
## View graphically the distribution of sequence lengths
#View with histogram
hist(nchar(combined_NCBI_and_Bold$clean_nucleotides), xlab = "Sequence_Length", ylab =
↪  "Frequency", main = "Frequency Histogram of Sequence Lengths")
#View with boxplot
boxplot(nchar(combined_NCBI_and_Bold$clean_nucleotides), xlab = "Ursidae_sequence", ylab
↪  = "Sequence_length", main = "Boxplot of sequence length")
#get summary of sequence length
summary(nchar(combined_NCBI_and_Bold$clean_nucleotides))
```

```r
#Assign a vector to hold the first quartilie and third quartile of sequence lengths
q1 <- quantile(nchar(combined_NCBI_and_Bold$clean_nucleotides), probs = 0.25, na.rm =
↪  TRUE)
q1

q3 <- quantile(nchar(combined_NCBI_and_Bold$clean_nucleotides), probs = 0.75, na.rm =
↪  TRUE)
q3
#Filter records to only include COI sequences that are inbetween the interquartile range.
combined_NCBI_and_Bold <- combined_NCBI_and_Bold %>%
  filter(str_count(clean_nucleotides) >= q1 & str_count(clean_nucleotides) <= q3)
#Checks to make sure everything worked as expected
summary(str_count(combined_NCBI_and_Bold$clean_nucleotides))
```

Calculate sequence features

```r
#convert COI sequence to DNAStringset so that we can use Biostrings package.
combined_NCBI_and_Bold <- as.data.frame(combined_NCBI_and_Bold)
combined_NCBI_and_Bold$clean_nucleotides <-
↪  DNAStringSet(combined_NCBI_and_Bold$clean_nucleotides)
class(combined_NCBI_and_Bold$clean_nucleotides)
class(combined_NCBI_and_Bold)
##calculate nucleotide frequencies.
#Add column of the absolute count of A, C , G, and T in the clean_nucleotides column
↪  using letterFrequency. Use cbind to append to combined_NCBI_and_Bold
combined_NCBI_and_Bold <- cbind(combined_NCBI_and_Bold,
↪  as.data.frame(letterFrequency(combined_NCBI_and_Bold$clean_nucleotides, letters =
↪  c("A", "C","G", "T"))))
head(combined_NCBI_and_Bold)
#Add the proportional frequencies of the nucleotides in proportion to the other
↪  nucleotides. Creating new columns using "$"
combined_NCBI_and_Bold$Aprop <- (combined_NCBI_and_Bold$A) / (combined_NCBI_and_Bold$A +
↪  combined_NCBI_and_Bold$C + combined_NCBI_and_Bold$G + combined_NCBI_and_Bold$T)

combined_NCBI_and_Bold$Tprop <- (combined_NCBI_and_Bold$T) / (combined_NCBI_and_Bold$A +
↪  combined_NCBI_and_Bold$C + combined_NCBI_and_Bold$G + combined_NCBI_and_Bold$T)

combined_NCBI_and_Bold$Gprop <- (combined_NCBI_and_Bold$G) / (combined_NCBI_and_Bold$A +
↪  combined_NCBI_and_Bold$C + combined_NCBI_and_Bold$G + combined_NCBI_and_Bold$T)

combined_NCBI_and_Bold$Cprop <- (combined_NCBI_and_Bold$C) / (combined_NCBI_and_Bold$A +
↪  combined_NCBI_and_Bold$C + combined_NCBI_and_Bold$G + combined_NCBI_and_Bold$T)
```

```r
head(combined_NCBI_and_Bold)
#Add dinucleotide and trinucleotide frequencies
combined_NCBI_and_Bold <- cbind(combined_NCBI_and_Bold,
↪  as.data.frame(dinucleotideFrequency(combined_NCBI_and_Bold$clean_nucleotides, as.prob
↪  = TRUE)))

combined_NCBI_and_Bold <- cbind(combined_NCBI_and_Bold,
↪  as.data.frame(trinucleotideFrequency(combined_NCBI_and_Bold$clean_nucleotides,
↪  as.prob = TRUE)))
#Add k-mer of 4 to the oligonucleotide frequencies at combined_NCBI_and_Bold to increase
↪  the accuracy of the machine learning algorithyms.
combined_NCBI_and_Bold <- cbind(combined_NCBI_and_Bold,
↪  as.data.frame(oligonucleotideFrequency(x = combined_NCBI_and_Bold$clean_nucleotides,
↪  width = 4, as.prob = TRUE)))
#check to see everything added
names(combined_NCBI_and_Bold)
head(combined_NCBI_and_Bold)

#Change clean_nucleotides to character from Biostring
combined_NCBI_and_Bold$clean_nucleotides <-
↪  as.character(combined_NCBI_and_Bold$clean_nucleotides)
class(combined_NCBI_and_Bold$clean_nucleotides)
```

Main Software Tools Description

The Biostrings package, part of the Bioconductor suite of tools, is used for efficient manipulation of biological strings (Pagès et al., 2022). It is included in the larger BiocManager package. We used the "readDNAStringset", "DNAStringSet", and "oligonucleotidefrequency" functions from Biostrings to read a FASTA file, align the sequences using MUSCLE, and calculate oligonucleotide frequencies. We also used the PCAtools package, which is part of BiocManager (Blighe & Lun, 2022). This package was chosen for its specialized functionalities for PCA analysis, and we used its graphics tools to create visualizations for our PCA analysis.

Code Section 2 - Main Analysis

Sample dataset for analysis

```r
##figure out the representation of each species in dataset
table(combined_NCBI_and_Bold$species_name)
```

To reduce noise, we will omit species that are have sample sizes less than 10 after filter steps

```r
subset_of_combined <- combined_NCBI_and_Bold %>%
  group_by (species_name) %>%
  mutate (count = n()) %>%
  filter (count >=10) %>%
  select(-count)
## check species count again
```

```r
table(subset_of_combined$species_name)
## for each species, sample 10 records
set.seed(111)
data_for_analysis <- subset_of_combined %>%
  group_by(species_name) %>%
  sample_n(size = 10)
## check species count again
table(data_for_analysis$species_name)
### Turn data for analysis to data frame
data_for_analysis <- as.data.frame(data_for_analysis)
##add names of rows. Add the number count for each species. There are 10
rownames(data_for_analysis) <- paste(data_for_analysis$species_name, 1:10, sep = ".")
head(data_for_analysis)
```

Figure out how many clusters to use

```r
## create numberic data matrix for analysis
numeric_data_matrix <- as.matrix(data_for_analysis[ , 10:349])
#create names for samples in matrix
rownames(numeric_data_matrix) <- rownames(data_for_analysis)
head(numeric_data_matrix)
```

```r
# Get the total withing sum of squares for k's at different values (1-15). The elbow
↪  position of the plot is the K to use. However, we already know posteriori that our
↪  data has 4 species. Function get_tot_withinss defined above
set.seed(999)
ks <- 1:15
tot_within_ss <- sapply(ks, get_tot_withinss, numeric_matrix = numeric_data_matrix)
#plot a graph of tot_withinss against the value of K used
df_sum_of_squares <- data.frame(x = ks, y = tot_within_ss )

ss_plot <- ggplot(data = df_sum_of_squares, aes(x = x, y = y)) +
  geom_line(color = "blue") +
  geom_point(color = "blue") +
  labs(title ="Total sum of square analysis for optimal k", x = "Number of clusters: K",
  ↪  y = "Total within-clusters sum of squares", colour = "Genus") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure out K using sum of squares

```r
#ks will start from 2 because cannot get mean of 1 value for 1 group
set.seed(987)
ks2 <- 2:15
average_sil_values <- sapply(ks2, get_average_sil, numeric_matrix = numeric_data_matrix)
```

```r
#create dataframe for values
df_average_sil <- data.frame(x = ks2, y = average_sil_values )
#create plot
sil_plot <- ggplot(data = df_average_sil, aes(x = x, y = y)) +
  geom_line(color = "blue") +
  geom_point(color = "blue") +
  labs(title ="Average Silhouette analysis for optimal k", x = "Number of clusters: K", y
  ↪  = "Average Silhouette width", colour = "Genus") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure out K using average silhouette method

Perform unsupervised k-means clustering

Perform clustering using only numerical columns. the number of groups is 4 because there are 7 species in data. The number of times the k-means algorithm should be repeated(nstart) is 10

```r
set.seed(777)
kmeans_clustering <- kmeans(numeric_data_matrix, 4, nstart = 10)
kmeans_clustering
```

Perform unsupervised hierarchical clustering

```r
# sequences to DNAstringset
data_for_analysis$clean_nucleotides <- DNAStringSet(data_for_analysis$clean_nucleotides)
#Assign the species names as names for the sequences
names(data_for_analysis$clean_nucleotides) <- rownames(data_for_analysis)
#Align sequences
aligned_data_for_analysis <-
↪  DNAStringSet(muscle::muscle(data_for_analysis$clean_nucleotides,
↪  gapopen=-300),use.names=TRUE)
#Writing the alignment to a FASTA file
writeXStringSet(aligned_data_for_analysis, file = "Ursidae_aligned_seq.fasta")
#View alignment in browser to check for irregularities
# BrowseSeqs(aligned_data_for_analysis)
#Check the class of the alignments
class(aligned_data_for_analysis)

##Create a matrix of pairwise distances from DNA sequences
#the model that will be used is "JC69". It assumes that all substitutions have the same
↪  probability. This probability is the same for all sites along the DNA sequence.
method = "JC69"
#Converted the biostring to the DNAbin data class to use to get distance matrix. The
↪  function to calculate distance as.DNAbin is found in the ape package as well as is
↪  as.DNAbin.
dnaBin_alignment <- as.DNAbin(aligned_data_for_analysis)
class(dnaBin_alignment)
```

```r
#Create distance matrix
distance_matrix <- dist.dna(dnaBin_alignment, model = method, as.matrix = TRUE,
↪  pairwise.deletion = TRUE)
distance_matrix
#Using as.dist function to convert distance matrix for use in the hclust function
distance_matrix<-as.dist(distance_matrix)

##perform hierarchical clustering
#complete model will be used to use distance between furthest elements in cluster
model = "complete"
set.seed(456)
hierach <- hclust(distance_matrix, method = model) %>%
  as.dendrogram()
```

Alignment based hiearchical clustering

```r
#get distance matrix using sequence features
distance_matrix_features <- dist(numeric_data_matrix)
#perform hierarchical clustering
set.seed(789)
hierach2 <- hclust(distance_matrix_features, method = model) %>%
  as.dendrogram()
```

Sequence features based hierarchical clustering

Plots to reduce dimension

Use t-Distributed Stochastic Neighbour Embedding (t-SNE) to reduce dimensions

```r
set.seed(121)
rtsne <- Rtsne(numeric_data_matrix, perplexity = 1, check_duplicates = FALSE)
#perform t-SNE withe different perplexities
set.seed(131)
rtsne2 <- Rtsne(numeric_data_matrix, perplexity = 4, check_duplicates = FALSE)
set.seed(141)
rtsne3 <- Rtsne(numeric_data_matrix, perplexity = 8, check_duplicates = FALSE)
set.seed(151)
rtsne4 <- Rtsne(numeric_data_matrix, perplexity = 13, check_duplicates = FALSE)


#create t-SNE plots
sTSNE_plot1 <- create_tSNE_plot(title_of_plot = "t-SNE plot showing the clusters by
↪  species: perplexity = 1", rtsne = rtsne)
sTSNE_plot2 <- create_tSNE_plot(title_of_plot = "t-SNE plot showing the clusters by
↪  species: perplexity = 4", rtsne = rtsne2)
sTSNE_plot3 <- create_tSNE_plot(title_of_plot = "t-SNE plot showing the clusters by
↪  species: perplexity = 8", rtsne = rtsne3)
sTSNE_plot4 <- create_tSNE_plot(title_of_plot = "t-SNE plot showing the clusters by
↪  species: perplexity = 13", rtsne = rtsne4)
```

Use PCA

Transpose elements nnumerical data.For pca in PCAtools package, the variables ar expected to be in rows and samples in column

```
## PCA with PCATools package
#transpose elements nnumerical data.For pca in PCAtools package, the variables ar
↪  expected to be in rows and samples in column
transposed_numeric_data_matrix <- t(numeric_data_matrix)
#check to see new dimensions
dim(transposed_numeric_data_matrix)
dim(numeric_data_matrix)
head(transposed_numeric_data_matrix)
```

Create metadata to use in pca analysis.It is strictly enforced that rownames(metada) == colnames(mat) when using pca function. Metadata simply a dataframe with the different factors of the samples in a row and the row being nemaed the sample name. For our purposes factor would be species name, kmeans_analysis, and hierarchical results with alignment and sequence features

```
metadata_pca <- data.frame(as.factor(data_for_analysis$species_name),
↪  as.factor(kmeans_clustering$cluster), as.factor(cutree(hierach, k = 4)),
↪  as.factor(cutree(hierach2, k = 4)), row.names =
↪  colnames(transposed_numeric_data_matrix))
colnames(metadata_pca) <- c("species_name", "kmeans_cluster", "cluster_by_alignment",
↪  "cluster_by_feature")
#peform PCA analysis
set.seed(123)
pcatools_pca <- pca(mat = transposed_numeric_data_matrix, metadata = metadata_pca)

#get biplot showing loading vectors using colors from the groupin
pca_loadong_plot <- PCAtools::biplot(pcatools_pca,
                 showLoadings = TRUE, legendPosition = "right",
                 title = "PCA plot showing loading vectors", titleLabSize = 22,
                 colLoadingsArrows = "blue", colby = "species_name")


# plot PCA with different factors
PCA_plot1 <- create_PCA_plot(title_of_plot = "PCA plot showing clustering by species",
↪  clustering = "species_name")
PCA_plot2 <- create_PCA_plot(title_of_plot = "PCA plot showing clustering by k-means
↪  analysis", clustering = "kmeans_cluster")
PCA_plot3 <- create_PCA_plot(title_of_plot = "PCA plot showing clustering by alignment
↪  based Hierarchical clustering", clustering = "cluster_by_alignment")
PCA_plot4 <- create_PCA_plot(title_of_plot = "PCA plot showing clustering by feature
↪  based Hierarchical clustering", clustering = "cluster_by_feature")
```

plots used for analysis

```r
#plot 1 k-analysis
plot1 <- ggarrange(ss_plot, sil_plot,
                  labels = c("A", "B"),
                  ncol = 2, nrow = 1)
annotate_figure(plot1, top = text_grob("Plots to analyze number of K", size = 20, color =
↪  "red", face ="bold"))




#plot 2 hierarchical clustering dendograms
dendlist(hierach, hierach2) %>%
  untangle() %>%
  tanglegram(margin_inner= 15,
             main = "Figure showing hierarchical clustering results", main_left =
             ↪  "Dendogram done with alignment", main_right = "Dendogram done with
             ↪  sequence features",
             cex_main = 3, cex_main_left = 1.5, cex_main_right = 1.5)



#plot 3 t-SNE analysis for different perplexities
plot3 <- ggarrange(sTSNE_plot1, sTSNE_plot2, sTSNE_plot3, sTSNE_plot4,
                  labels = c("A", "B", "C", "D"),
                  ncol = 2, nrow = 2,
                  common.legend = TRUE, legend = "bottom")
annotate_figure(plot3, top = text_grob("t-SNE analysis for different perplexities", size
↪  = 20, color = "red", face ="bold"))



#plot 4 t-SNE vs PCA
plot4 <- ggarrange(sTSNE_plot4, PCA_plot1,
                   labels = c("A", "B"),
                   ncol = 2, nrow = 1, legend = "bottom")
annotate_figure(plot4, top = text_grob("t-SNE vs PCA", size = 20, color = "red", face
↪  ="bold"))


#plot 5 PCA showing loading vectors
pca_loadong_plot



#plot 6 Different clustering analysis on PCA
plot6 <- ggarrange(PCA_plot1, PCA_plot2, PCA_plot3, PCA_plot4,
          labels = c("A", "B", "C", "D"),
          ncol = 2, nrow = 2,
          common.legend = TRUE, legend = "bottom")
annotate_figure(plot6, top = text_grob("Different clustering analysis on PCA", size = 20,
↪  color = "red", face ="bold"))
```

```
# dev.off()
```

Figures

```
#plot 1 k-analysis
plot1 <- ggarrange(ss_plot, sil_plot,
                  labels = c("A", "B"),
                  ncol = 2, nrow = 1)
```

```
annotate_figure(plot1, top = text_grob("Plots to analyze number of K", size = 20, color =
↪  "black", face ="bold"))
```

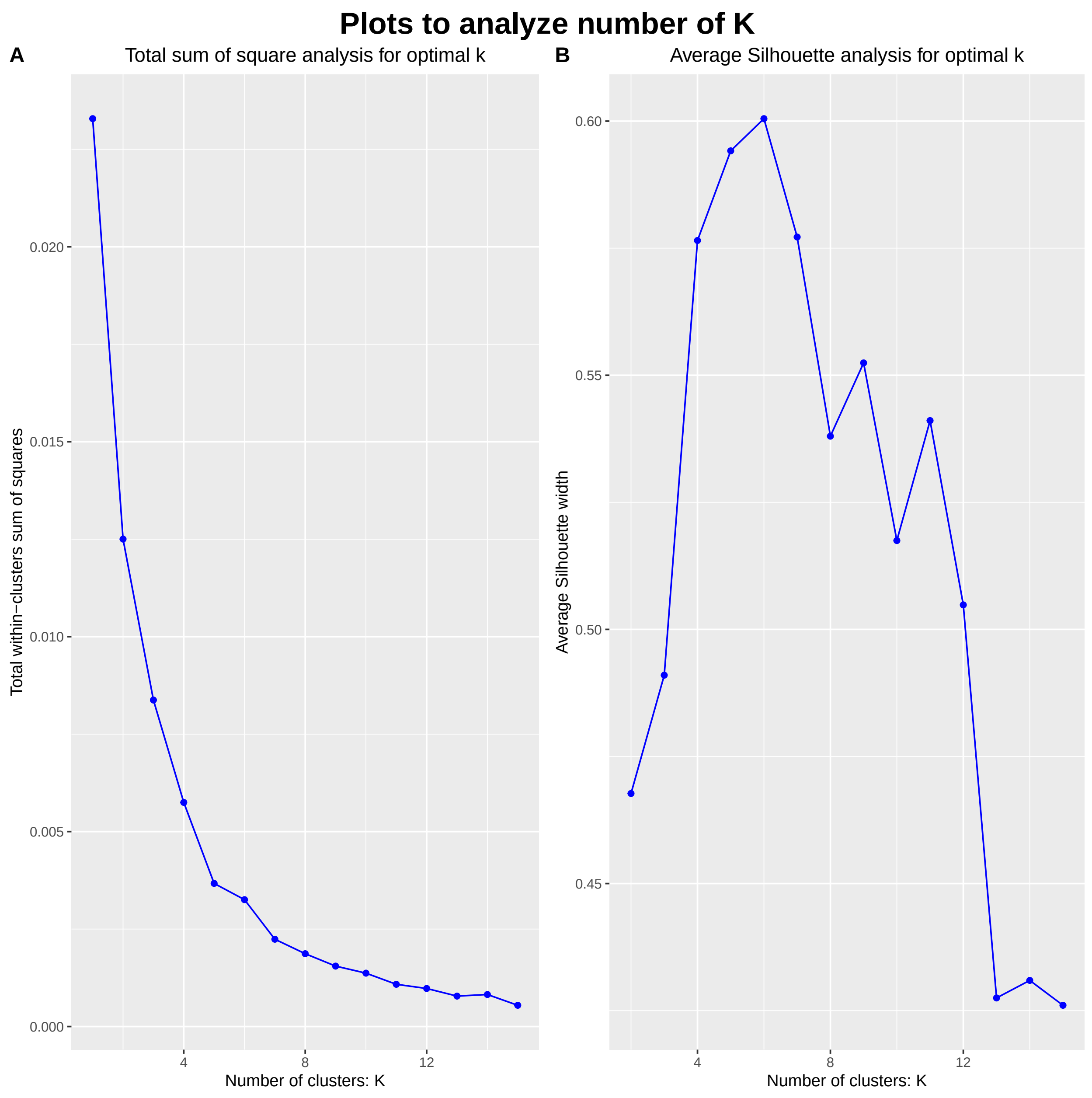

Figure 1. Plots to analyse for number of groups. Figure1.A shows the total sum of squares when k-means was performed for different values of K. Figure1.B shows the average silhouette width of observations for different values of K.

```
#alignment base hierarchical clustering result
plot(hierach)
title(main = "Alignment based hierarchical clustering result", ylab="Evolutionary
↪  Distance",line=2)
```

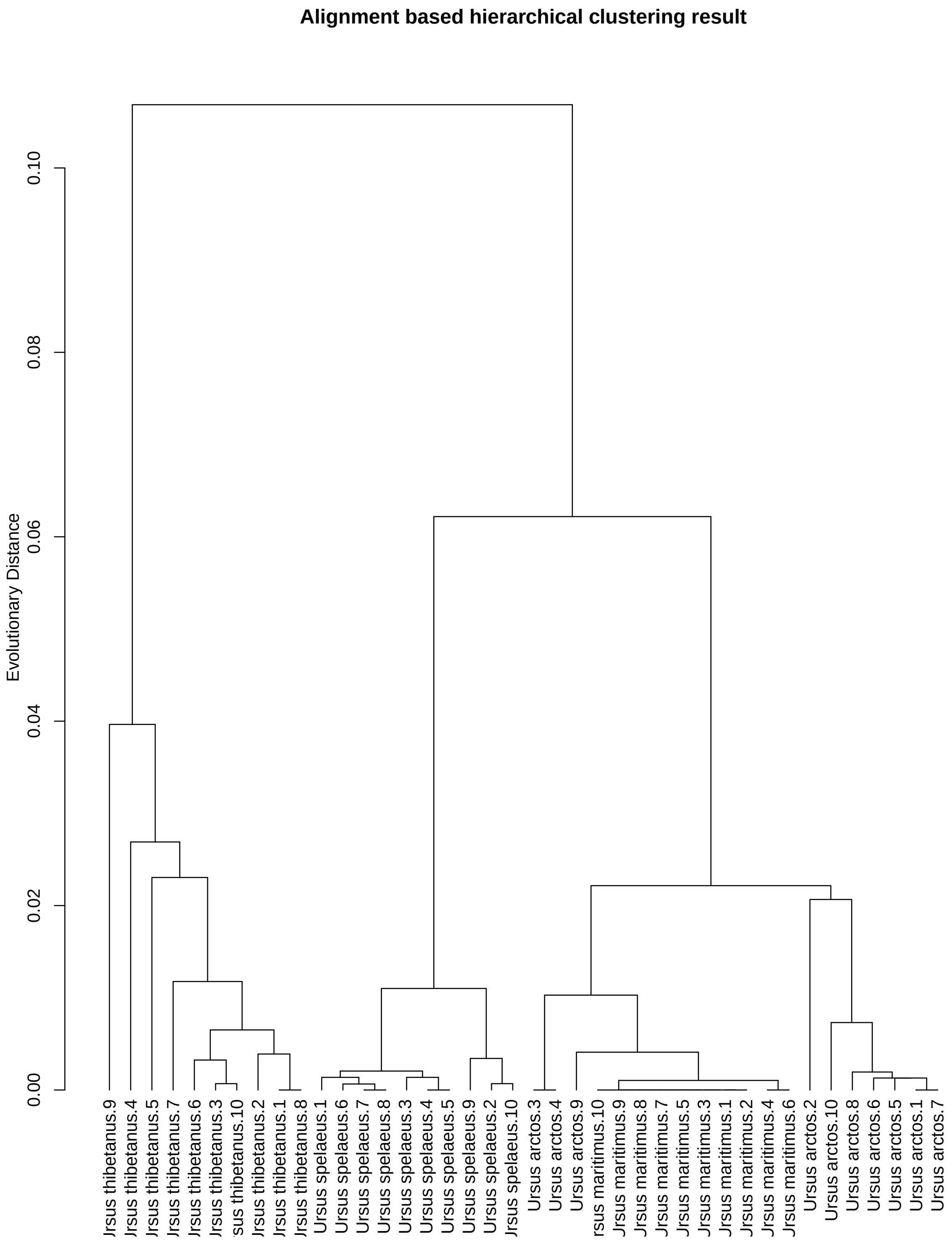

Figure 2. Dendogram of Hierarchical clustering done using alignment

```
#t-SNE plot
sTSNE_plot4
```

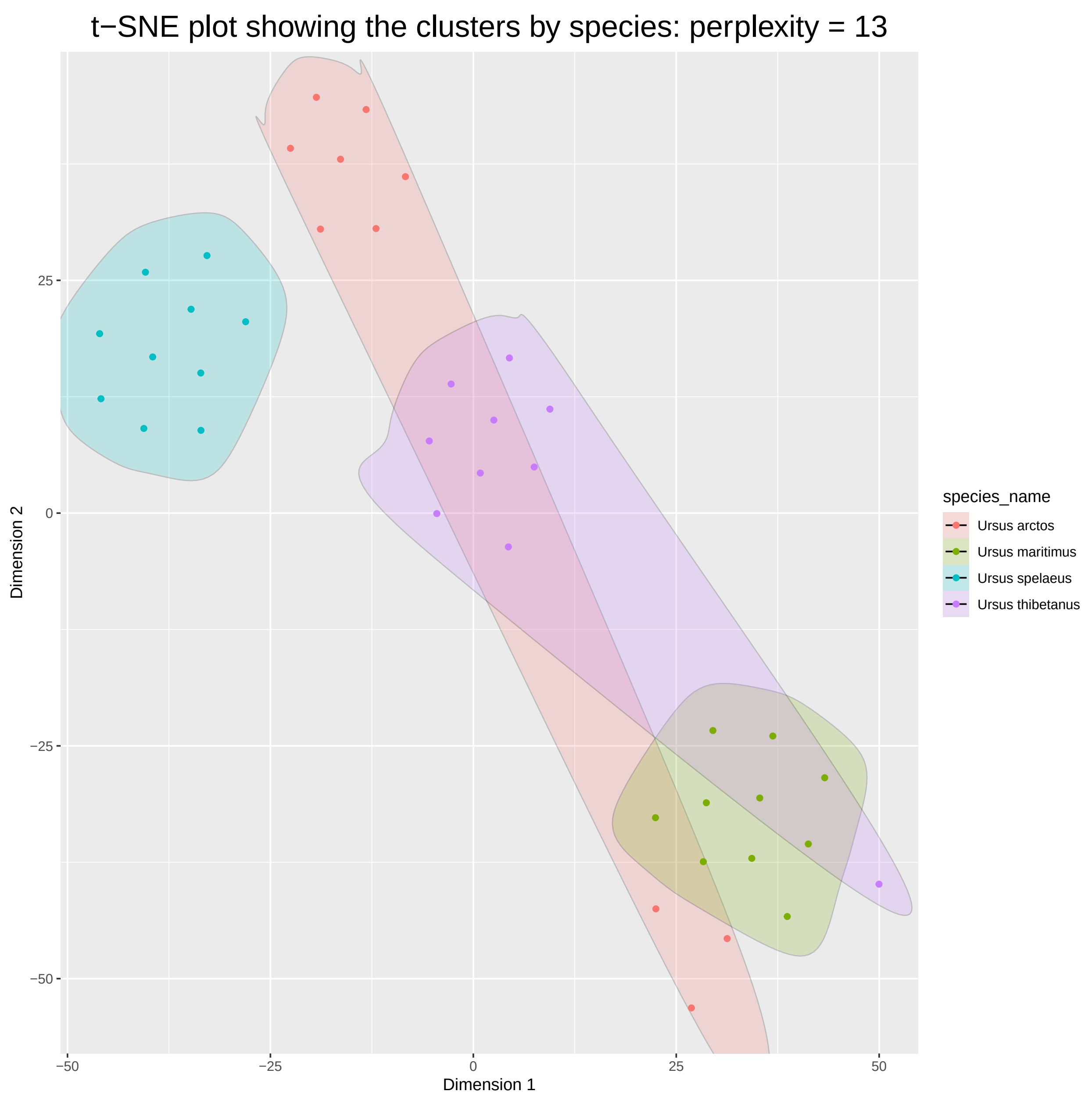

Figure 3. t-Distributed Stochastic Neighbor Embedding (t-SNE) plot to reduce dimensions. Plot shows the grouping of the species on t-SNE plot

```
#pca plot showing clustering by species_name
PCA_plot1
```

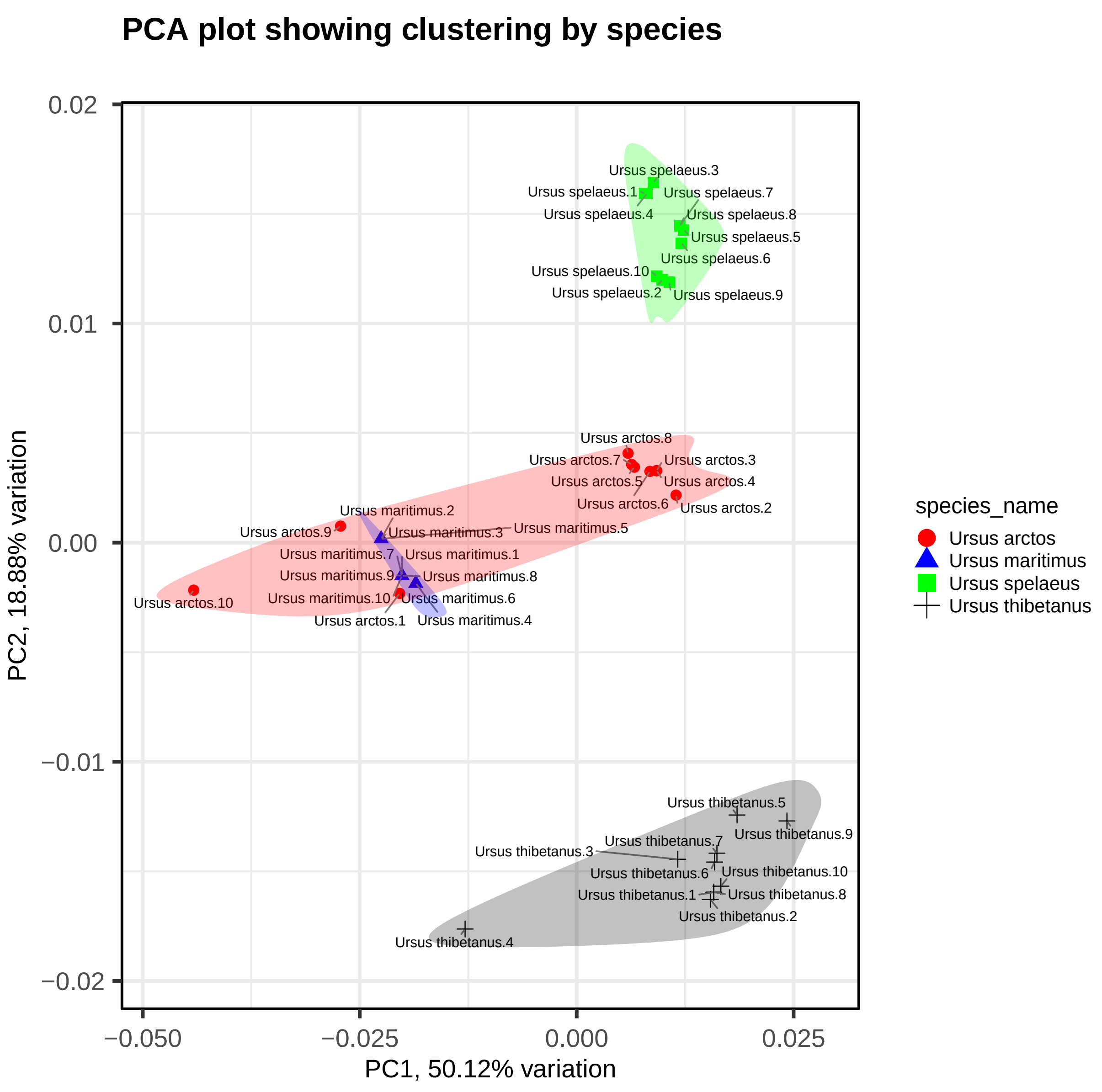

Figure 4. Principal Component Analysis (PCA) plot to reduce dimensions.Plot shows the grouping of the species on a PCA plot

```
#plot 5 PCA showing loading vectors
pca_loadong_plot
```

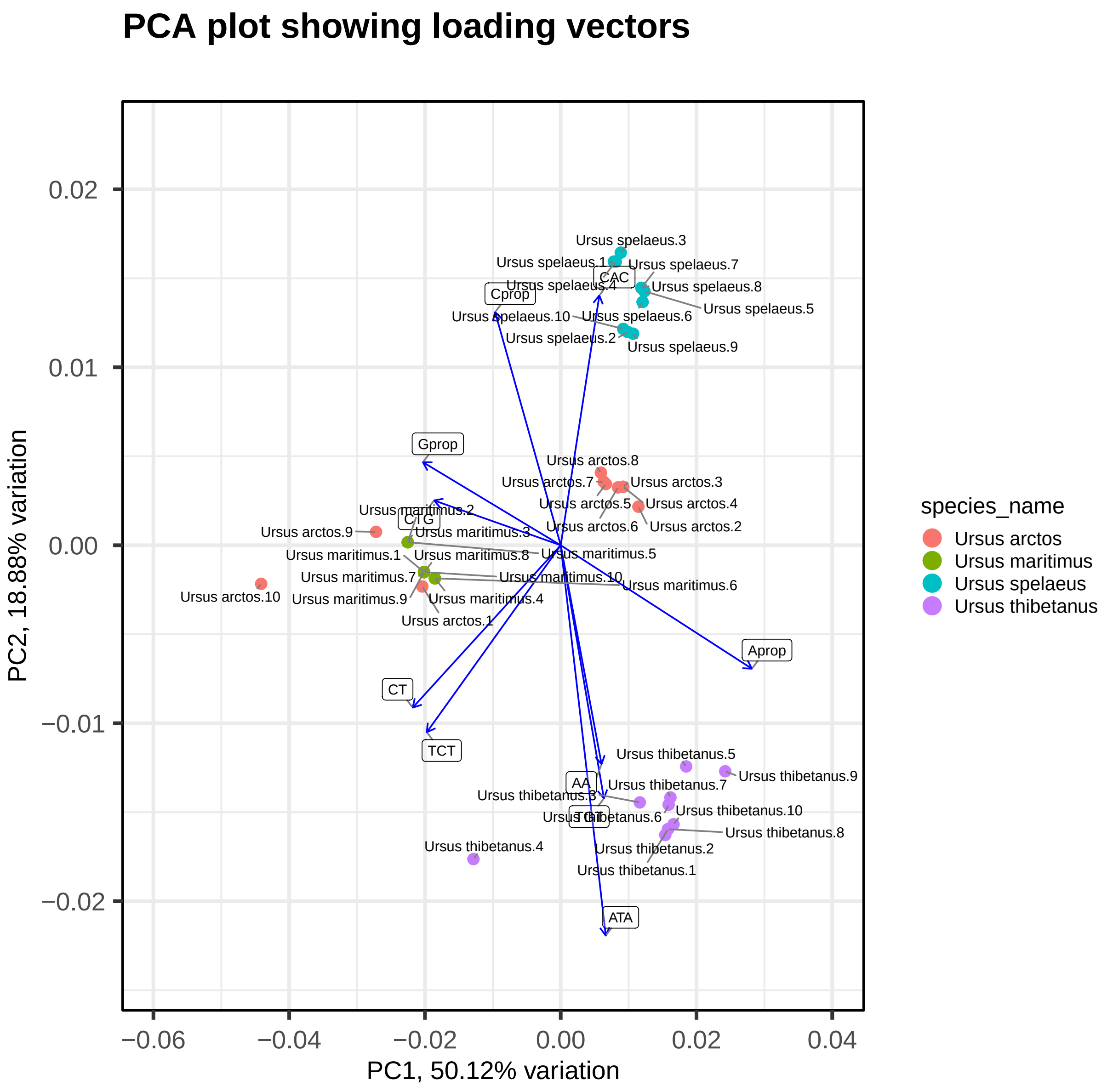

Figure 5. PCA plot showing loading vectors

```
#pca plot showing k-means clustering
PCA_plot2
```

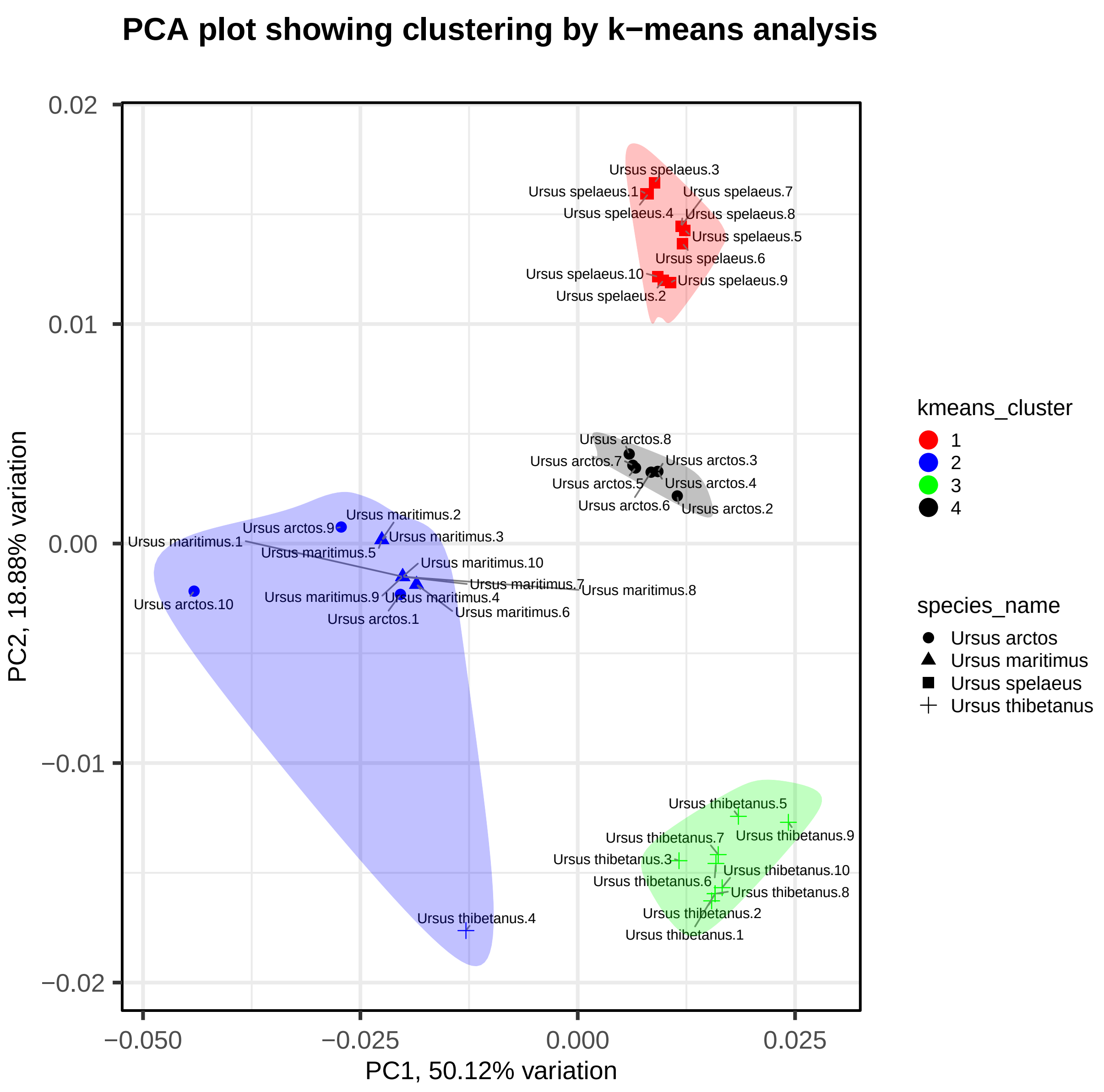

Figure 6. Plot showing the results of a k-means analysis plotted on a PCA plot to reduce dimensions

Results and Discussion

The aim of the project was to find the number of groups, cluster groups and reduce dimensions for a given dataset. There were four species of Ursidae used for analysis. The elbow method seems to have outperformed the average silhouette method for determining the number of clusters in this data set. The "elbow" position is the point on the graph where there is a significant change in the slope. It reflects the point were increasing the number of k leads to diminishing returns which can cause over-fitting (Ketchen & Shook, 1996). A k-value of four is predicted using this method as can be seen in Figure 1. A. The average silhouette method predicted six clusters: the peak average silhouette width is at k = 6 in Figure 1. B. It is possible that the average silhouette method is identifying the presence of cryptic species not recognized with the traditional taxonomic grouping. All clustering analysis were done with a k value of 4. The alignment based hierarchical

clustering grouped all species into clades (monophyletic groups) aside form 3 Ursus arctos species which were grouped together with Ursus maritimus: seen in Figure 2. The feature based hierarchical clustering analysis performed similarly to the alignment based method but grouped Ursus thibetunus.9 in the wrong group (figure is a supplementary plot). The PCA did a better job at dimensionality reduction than the t-SNE. Comparing Figure 3. And Figure 4., the PCA plot was able to cluster species into more distinct groups along the two principal dimensions. As mentioned in the introduction, the PCA preserves global structure of data which might be better when dealing with DNA. It is apt to consider what type of data is being used for analysis before choosing a method for dimensionality reduction. Subsequent plots were done using PCA. Figure 5. Shows the loading vectors (features) that are causing the most variation along the first and second principal components. The proportions of A, C and G nucleotides as well as the trinucleotide "ATA" account for the most variation along the first two principal components. Figure 6. shows the result of the *k*-means analysis on the PCA plot. The pattern of variation seen in the *k*-means analysis is similar to the PCA; all *k*-means clusters are distinct groups on the PCA plot, Figure 6.
A major limitation to the study is sample size. As already mentioned, the Ursidae family is a rare group. After filtering, the combined database (NCBI and BOLD) had nine species with sample sizes less than 10. It is worth noting that there are five extinct species of bears (Darin, 2015). Additionally, accuracy of the clustering analysis could be improved by using the frequency of longer k-mer lengths. However, this was beyond the scope of this paper. For future studies, exploring the techniques used for unbalanced data would be apt. Underrepresentation is a ubiquitous phenomenon with real-life implications. The importance of understanding how to tackle this can not be overstated.

Acknowledgements

I would like to acknowledge the following people for their contribution to the completion of this project. These people might not have helped me directly but at some point, we engaged in discussion pertaining to this project: Nishita Sharif, Daniel Amoako, and Omar Khan. I would also like to thank Jessica Labarcena for her role in my development as a coder.

References

Azzalini, F., Jin, S., Renzi, M., & Tanca, L. (2021). Blocking Techniques for Entity Linkage: A Semantics-Based Approach. Data Science and Engineering, 6(1), 20–38. https://doi.org/10.1007/s41019-020-00146-w

Barcode of life database (BOLD). (2022). Ursidae. Retrieved December 13, 2022, from http://www.boldsystems.org/index.php/API_Public/combined?taxon=Ursidae&format=tsv

Blighe K, Lun A (2022). PCAtools: PCAtools: Everything Principal Components Analysis. R package version 2.10.0, https://github.com/kevinblighe/PCAtools

Darin M. C. (2015). Chapter 50 – Ursidae. In R. E. Miller & M. E. Fowler (Eds.), Fowler's Zoo and Wild Animal Medicine, Volume 8 (pp. 498-508) W.B. Saunders. https://doi.org/10.1016/B978-1-4557-7397-8.00050-5

Gatto, L. (2020). An Introduction to Machine Learning with R. Github. https://lgatto.github.io/IntroMachineLearningWithR/unsupervised-learning.html

Hebert, P. D. N., Cywinska, A., Ball, S. L., & deWaard, J. R. (2003). Biological identifications through DNA barcodes. Proceedings of the Royal Society. B, Biological Sciences, 270(1512), 313–321. https://doi.org/10.1098/rspb.2002.2218

IUCN red list of threatened species. (n.d.). IUCN redlist. Retrieved December 13, 2022, from https://www.iucnredlist.org/

Jizheng Yi, Xia Mao, Yuli Xue, & Compare, A. (2013). Facial Expression Recognition Based on t-SNE and AdaboostM2. 2013 IEEE International Conference on Green Computing and Communications and IEEE

Internet of Things and IEEE Cyber, Physical and Social Computing, 1744–1749. https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.321

Ketchen, D. J., & Shook, C. L. (1996). The application of cluster analysis in strategic management research: an analysis and critique. Strategic Management Journal, 17(6), 441–458. https://doi.org/10.1002/(SICI)1097-0266(199606)17:6<441::AID-SMJ819>3.0.CO;2-G

Kumar, S. (2020). Silhouette Method — Better than Elbow Method to find Optimal Clusters. Towardsdatascience. https://towardsdatascience.com/silhouette-method-better-than-elbow-method-to-find-optimal-clusters-378d62ff6891

National Center for Biotechnology Information database (NCBI). (2022). Genbank. Retrieved December 13, 2022, from https://www.ncbi.nlm.nih.gov/nuccore/?term=((ursidae%5BOrganism%5D)+AND+500%3A1000%5BSequence+

Orton, M. G., May, J. A., Ly, W., Lee, D. J., & Adamowicz, S. J. (2019). Is molecular evolution faster in the tropics? Heredity, 122(5), 513–524. https://doi.org/10.1038/s41437-018-0141-7

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.