Unsupervised_machine_learning

Olusegun Odumosu

2022-12-16

Introduction

Setting up

install packages

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

load package(S)

```r
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.

```r
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))
}
```

Data exploration

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")
```

```r
##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

```r
##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

```r
#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

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

Remove duplicated sequences from combined NCBI and BOLD

```r
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

```r
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

```
#get summary of sequence length
summary(nchar(combined_NCBI_and_Bold$clean_nucleotides))
#Assign a vector to hold the first quartilie and third quartile of sequence lengths
q1 <- quantile(nchar(combined_NCBI_and_Bold$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

```
#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)
```

```
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)
```

Analysis to address questions

Sample dataset for analysis

```
##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

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

```r
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)


#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 ="Silhouette analysis for optimal k", x = "Number of clusters: K", y =
  ↪  "Average Silhouettes", 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)
#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 elementsin 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

```r
## 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

```
#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)
```

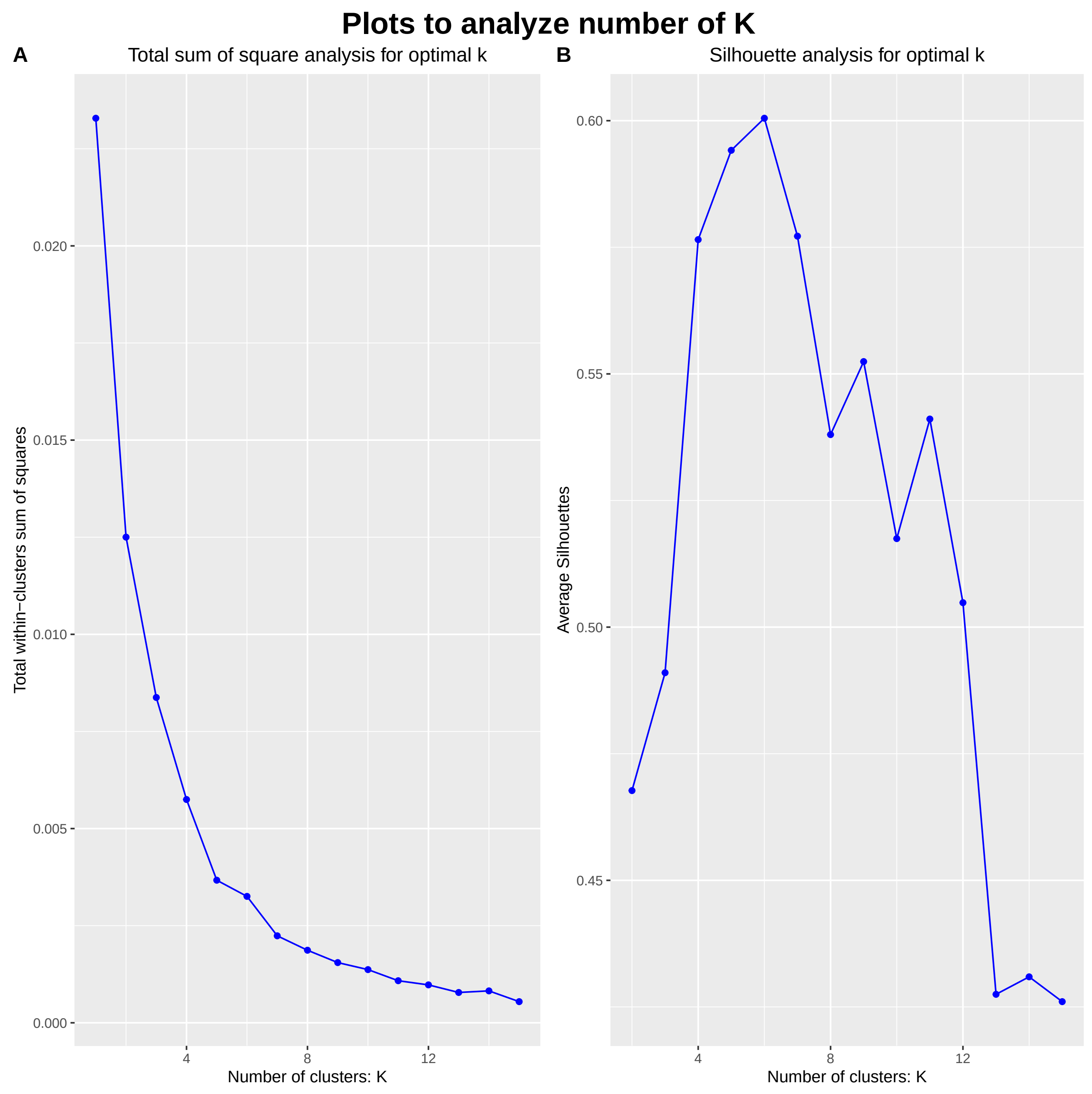
```r
#plot 3 t-SNE analysis for different perplexities
plot3 <- ggarrange(sTSNE_plot1, sTSNE_plot2, sTSNE_plot3, sTSNE_plot4,
                  labels = c("A", "B", "C", "D"),
                  ncol = 2, nrow = 2,
                  common.legend = TRUE, legend = "bottom")
annotate_figure(plot3, top = 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()
```

Conclusion

Plots

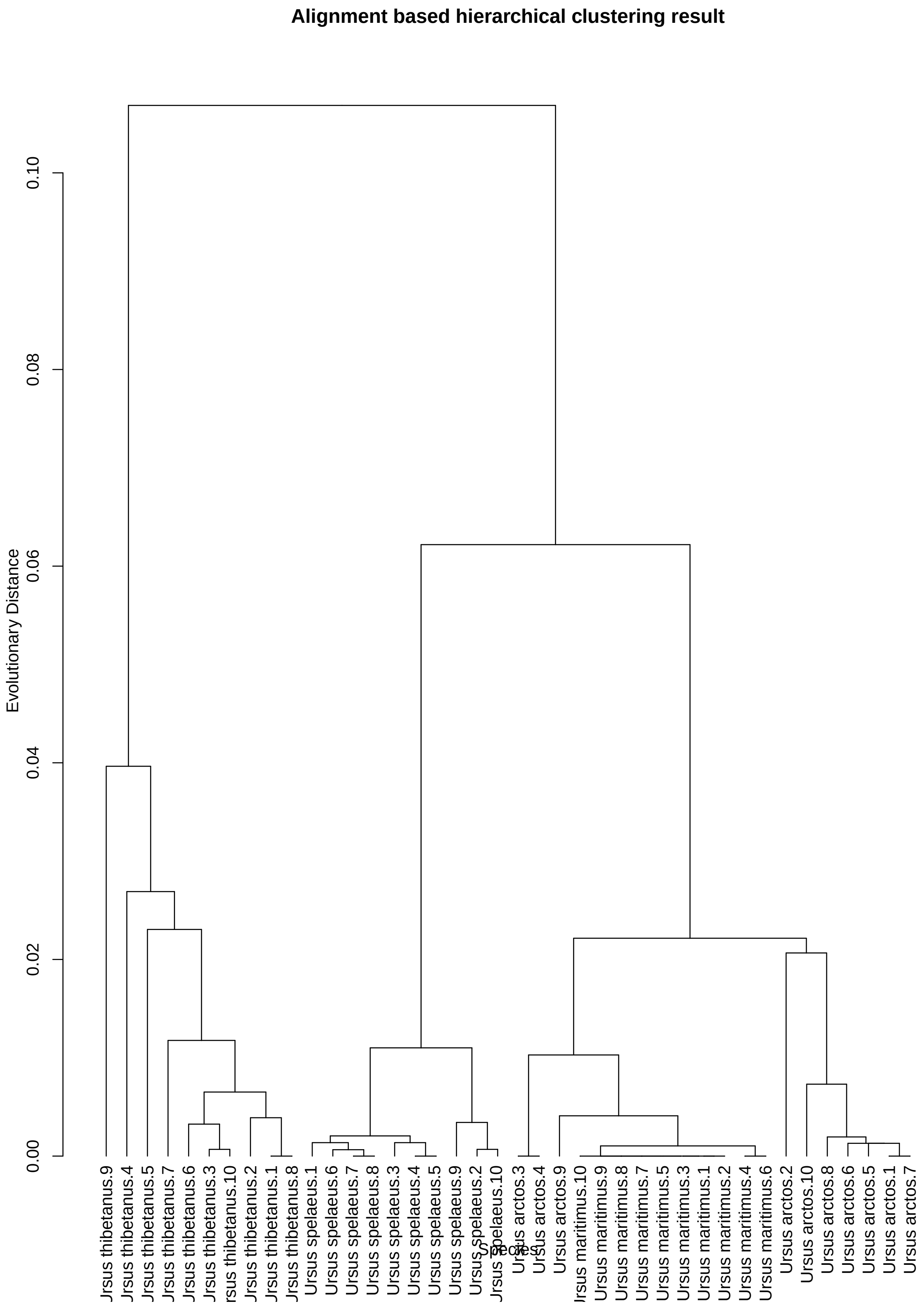
```r
#plot 1 k-analysis
plot1 <- ggarrange(ss_plot, sil_plot,
                  labels = c("A", "B"),
                  ncol = 2, nrow = 1)
```

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



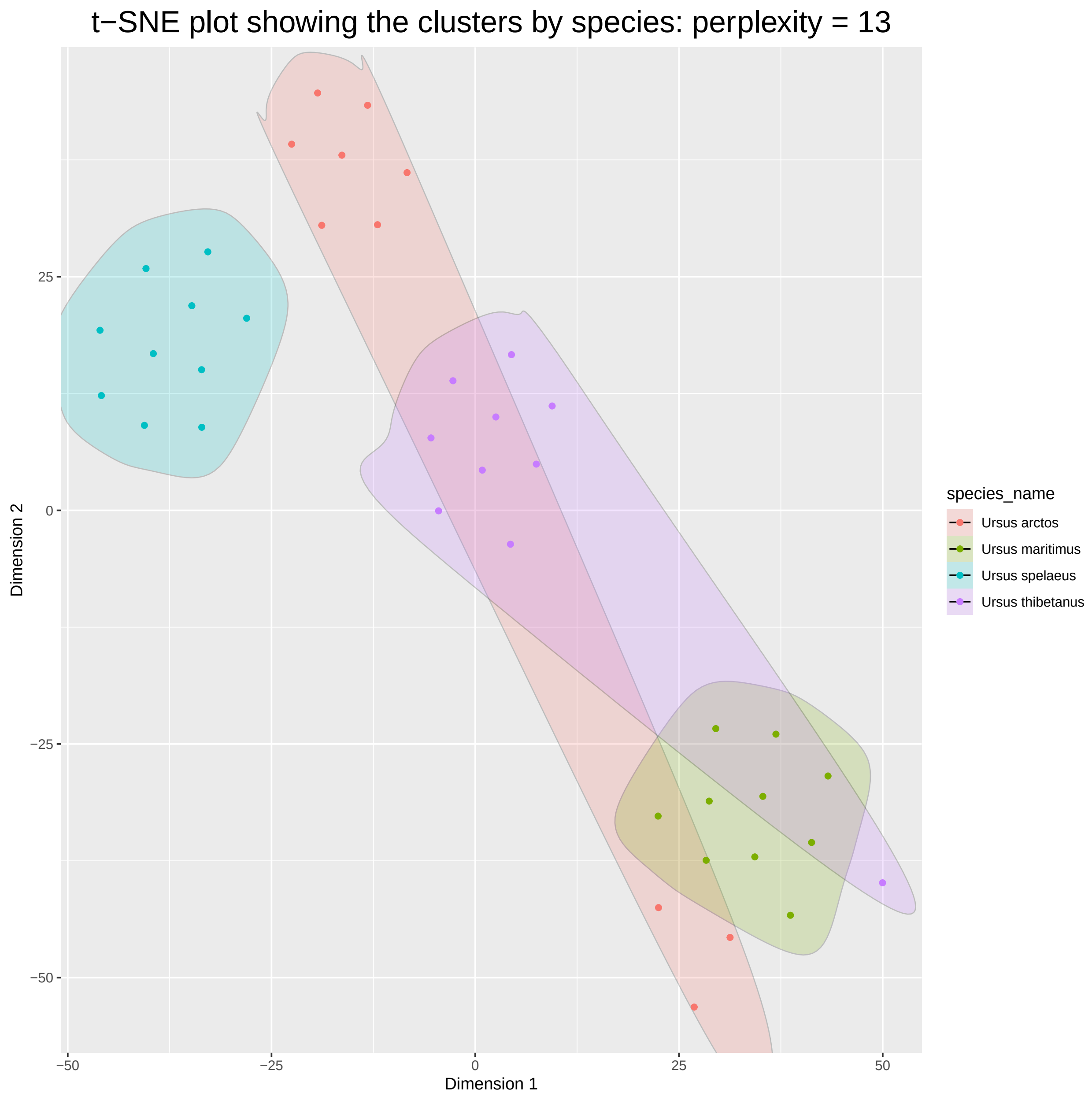
#### Figure 1. ....

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


Alignment based hierarchical clustering result
Evolutionary Distance
0.00
0.02
0.04
0.06
0.08
0.10
Species
Ursus thibetanus.9
Ursus thibetanus.4
Ursus thibetanus.5
Ursus thibetanus.7
Ursus thibetanus.6
Ursus thibetanus.3
Ursus thibetanus.10
Ursus thibetanus.2
Ursus thibetanus.1
Ursus thibetanus.8
Ursus spelaeus.1
Ursus spelaeus.6
Ursus spelaeus.7
Ursus spelaeus.8
Ursus spelaeus.3
Ursus spelaeus.4
Ursus spelaeus.5
Ursus spelaeus.9
Ursus spelaeus.2
Ursus spelaeus.10
Ursus arctos.3
Ursus arctos.4
Ursus arctos.9
Ursus maritimus.10
Ursus maritimus.9
Ursus maritimus.8
Ursus maritimus.7
Ursus maritimus.5
Ursus maritimus.3
Ursus maritimus.1
Ursus maritimus.2
Ursus maritimus.4
Ursus maritimus.6
Ursus arctos.2
Ursus arctos.10
Ursus arctos.8
Ursus arctos.6
Ursus arctos.5
Ursus arctos.1
Ursus arctos.7

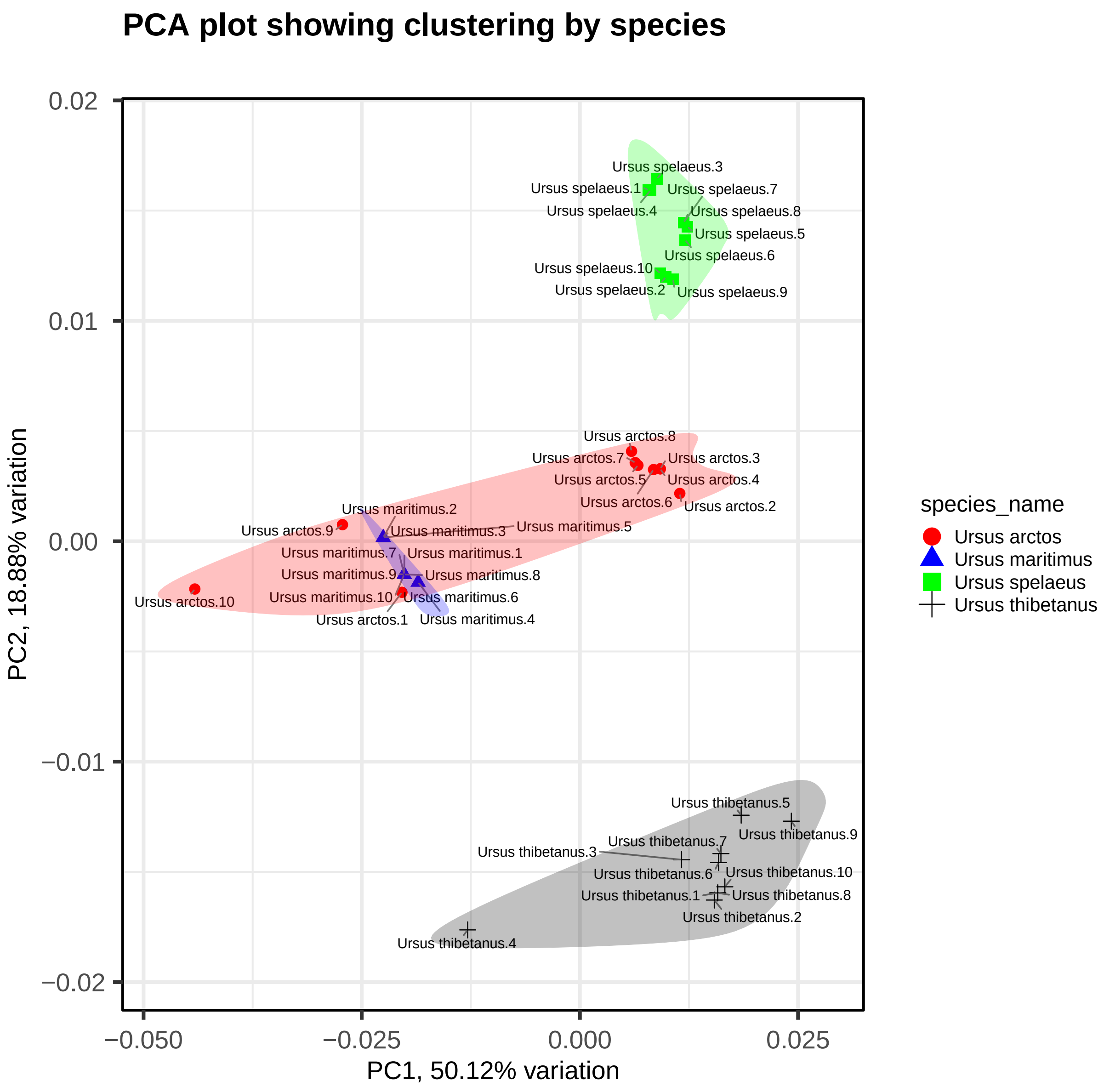
#### Figure 2. . . . .

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



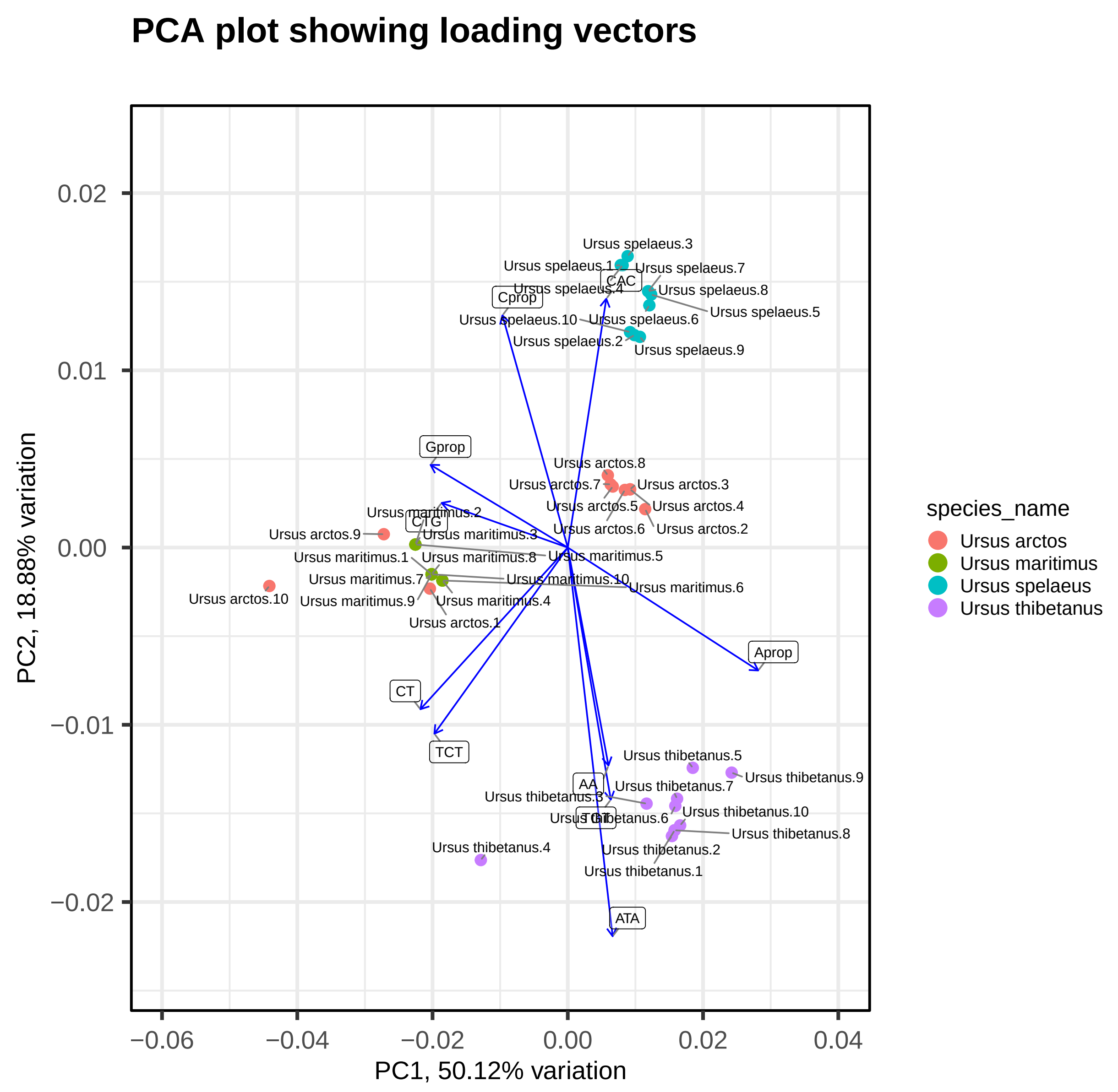
#### Figure 3. ....

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



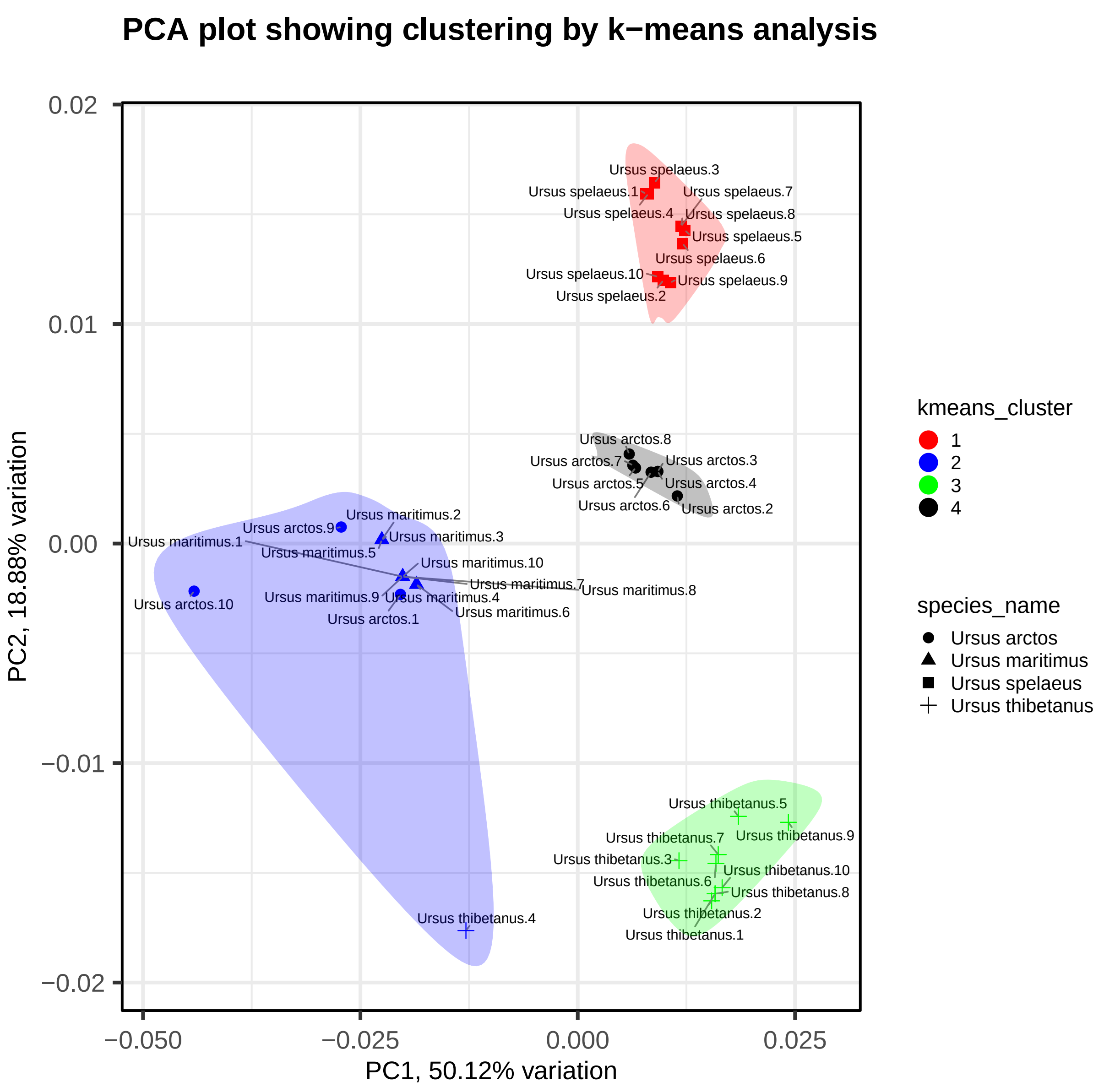
#### Figure 4. ....

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



#### Figure 5. . . . .

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


PCA plot showing clustering by k−means analysis
PC2, 18.88% variation
PC1, 50.12% variation
0.02
0.01
0.00
−0.01
−0.02
−0.050
−0.025
0.000
0.025
Ursus spelaeus.3
Ursus spelaeus.1
Ursus spelaeus.7
Ursus spelaeus.4
Ursus spelaeus.8
Ursus spelaeus.5
Ursus spelaeus.6
Ursus spelaeus.10
Ursus spelaeus.9
Ursus spelaeus.2
Ursus arctos.8
Ursus arctos.7
Ursus arctos.3
Ursus arctos.5
Ursus arctos.4
Ursus arctos.6
Ursus arctos.2
Ursus maritimus.2
Ursus arctos.9
Ursus maritimus.3
Ursus maritimus.1
Ursus maritimus.5
Ursus maritimus.10
Ursus maritimus.7
Ursus maritimus.8
Ursus arctos.10
Ursus maritimus.9
Ursus maritimus.4
Ursus maritimus.6
Ursus arctos.1
Ursus thibetanus.5
Ursus thibetanus.7
Ursus thibetanus.9
Ursus thibetanus.3
Ursus thibetanus.10
Ursus thibetanus.6
Ursus thibetanus.8
Ursus thibetanus.2
Ursus thibetanus.1
Ursus thibetanus.4
kmeans_cluster
1
2
3
4
species_name
Ursus arctos
Ursus maritimus
Ursus spelaeus
Ursus thibetanus

#### Figure 6. ....