Investigating the potential immunological of COI in Bat reservoir protection

Thomas Tekle

2024-10-25

```
##******************************
## Thomas Tekle - 1079669
##
## University of Guelph
##
## October 18th, 2024
##
## 6210 Assignment 2
##
##******************************
```

```
# Introduction ----
# Bats are one of the major reservoirs for the some the most dangerous viruses in world.
↪  Biological explanations behind this commensalistic relationship between bats and
↪  viruses have been attributed to physiological traits such as flight, internal
↪  temperature, immunological traits and microRNA expression profiles. However,
↪  molecular features aiding in the bats carrier ability have yet been elucidated
↪  (Mackenzie et al., 2016).

# The objective of this projects is investigate whether the gene COI, a gene that has
↪  been elucidated to play a critical role the regulation of T Cell fates (Tarasenko et
↪  al., 2017),  demonstrates distinct clustering patterns between two genera of bats
↪  that would suggest a critical function or contribution to the different pathogenesis
↪  between these taxonomies and the viruses they carry. The analysis of clustering
↪  profiles between the bats is preceded by the comparison of SVM and randomForest to
↪  identify which classifier would be more appropriate for downstream analysis, as both
↪  models are commonly used for clustering of biological classes (Solis-Reyes et al.,
↪  2018). This analysis aims to demonstrate that the COI gene exhibits distinct
↪  clustering patterns between the two bat genera, indicating potential molecular
↪  adaptations that may contribute to differential viral pathogenesis or immune
↪  responses between genera.
```

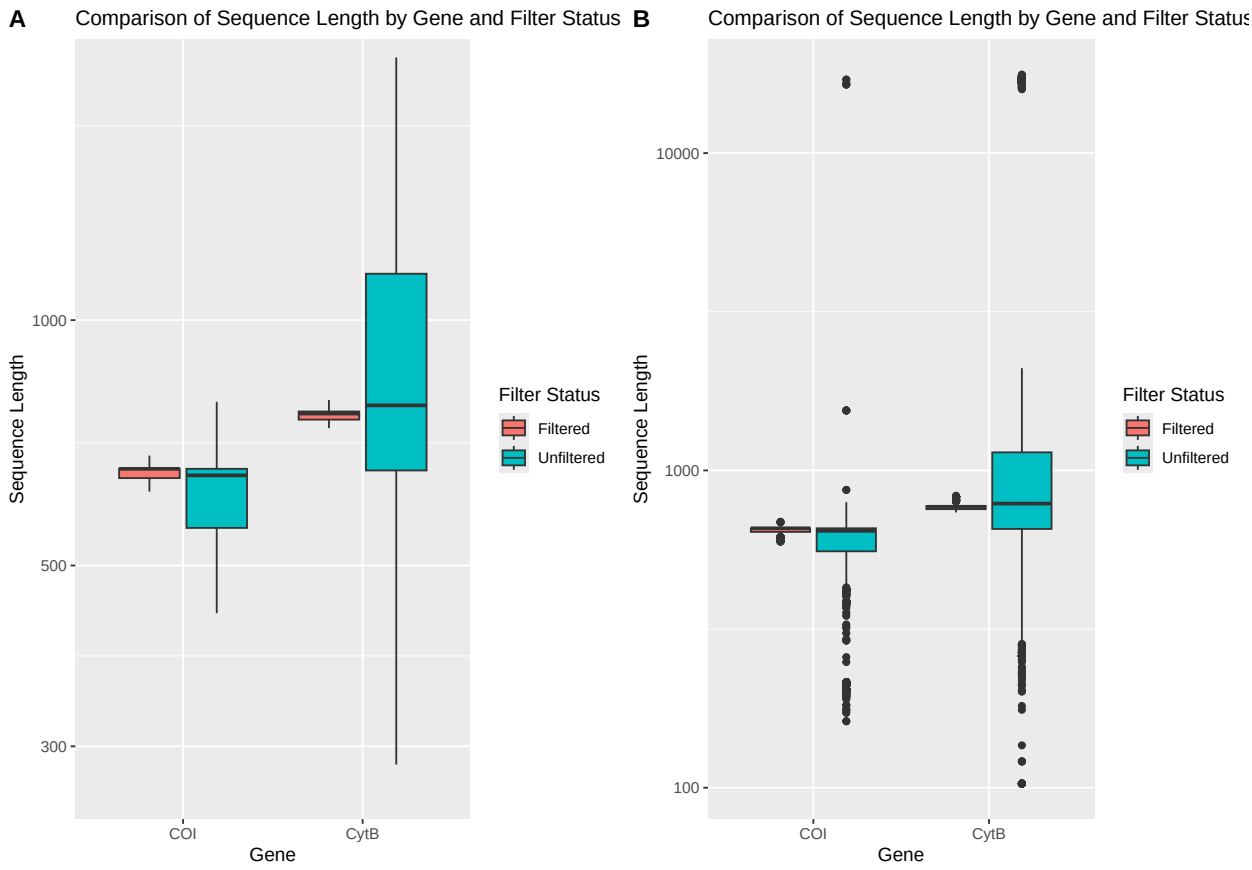
```
# Initializing relevant packages ----

package.list <- c("tidyverse", "ggplot2", "rentrez", "seqinr", "Biostrings",
↪  "randomForest", "ggpubr", "caret", "kernlab", "e1071", "ggrepel")

for (i in 1:length(package.list)) {
  lapply(package.list[i], library, character = T)
```

```
}


# Data Acquisition and Filtering ----


# Defining my variables to be used in the acquisition of data using the rentrez package
↪  for ease of readability and accessibility. This allows me to quickly edit my search
↪  parameters. The searches were conducted on OCT 19th, 2024. Sequences were retrieved
↪  from the NCBI nucleotide (nuccore) database.

database <- "nuccore"
VespertilionidaeCOI.searchterm <- "Vespertilionidae[ORGN] AND COI[Gene]"
VespertilionidaeCytB.searchterm <- "Vespertilionidae[ORGN] AND CytB[Gene]"
length.var <- 50
N.composition.limit <- 0.01


# Retrieving e_search results with default retmax parameter (20), since there isn't an
↪  object containing the maximum number of hits available at this point. Once the
↪  initial esearch is ran, the number of hits relayed by the NCBI servers is placed into
↪  separate objects serving as the parameter for the second iteration of esearch for the
↪  same terms but instead retrieving all available hits.

#VCOI.search.results <- entrez_search(db = database, term =
↪  VespertilionidaeCOI.searchterm)
#VCOI.maxHits <- VCOI.search.results$count

#VCytB.search.results <- entrez_search(db = database, term =
↪  VespertilionidaeCytB.searchterm)
#VCytB.maxHits <- VCytB.search.results$count

 #retrieving the e_search results data with the maximum number of hits (retmax = Max
 ↪  Hits)

#VCOI.search.results <- entrez_search(db = database, term =
↪  VespertilionidaeCOI.searchterm, retmax = VCOI.maxHits, use_history = T)

#VCytB.search.results <- entrez_search(db = database, term =
↪  VespertilionidaeCytB.searchterm, retmax = VCytB.maxHits, use_history = T)

# As mentioned above, Obtaining NCBI gene sequences using entrez fetch function in fasta
↪  format, done via a function which seperates the list of hits denoted on the web
↪  history of the NCBI server into batchs of 500 and concatenating them into a single
↪  sequence fasta folder. This eleviates the server load as well as bypasses the
↪  computational restriction set in place by NCBI when retriving large abundances of
↪  sequence data.

#fetch.by.batch <- function(search_results) {
  #total.size <- search_results$count #placing total hits from esearch into a total size
  ↪  object
  #totalSeqs <- c() #initializing empty vector for which fasta sequences will be placed
  ↪  in iteratively
```

```
  #for (batch in seq(0, total.size, by = 500)) { #efetch for loop that retreives
  ↪  sequences in batches of 500
    #NewSeq <- entrez_fetch(db = database, web_history = search_results$web_history,
    ↪  retmax = 500, rettype = "fasta", retstart = batch)

    #totalSeqs <- c(totalSeqs, NewSeq) #Appending next batch of sequences into total
    ↪  sequences vector
  #}
  #return(totalSeqs)
#}

#Calling the fetch.by.batch function for both sequence data sets, then subsequently
↪  writing the fasta data onto harddrive.

 #dir.create("data")
#VCOI.sequences <- fetch.by.batch(VCOI.search.results)
#rite(VCOI.sequences, "../data/VCOI.fasta", sep = "\n")

#VCytB.sequences <- fetch.by.batch(VCytB.search.results)
#write(VCytB.sequences, "../data/VCytB.fasta", sep = "\n")


# Loading the fasta files back onto the R workspace as a DNAStringSet object, then
↪  placing the Sequences into two data frames (for each gene [CytB and COI]), extracting
↪  and separating key information [sequences, NCBI ID, seq length, species_name, gene,
↪  filtering status and seq length] as distinct variables for downstream filtering and
↪  analysis.

VCOI.stringSet <- readDNAStringSet("../data/VCOI.fasta")

dfVCOI <- data.frame(NCBI_ID = word(names(VCOI.stringSet), 1L), Species_Name =
↪  word(names(VCOI.stringSet), 2L, 3L), Sequence = paste(VCOI.stringSet), Gene = "COI",
↪  Filter_Status = "Unfiltered")

dfVCOI <- dfVCOI %>%
  mutate(Sequence_length = nchar(dfVCOI$Sequence))

#Another data frame was created for filtering out sequences with high N composition and
↪  sequence lengths that are suspected to be outliers, and would otherwise induce loss
↪  of accuracy for the classifier models.

dfVCOI_filtered <- dfVCOI %>%
  mutate(Sequence_filtered = str_remove_all(Sequence, "^N+|N+$|-")) %>% #removing
  ↪  terminal Ns and all gaps from sequences
  filter(str_count(Sequence_filtered, "N") <= (N.composition.limit *
  ↪  str_count(Sequence))) %>%
  select(!"Sequence") %>% #removing unfiltered sequence variable from new df.
  filter(str_count(Sequence_filtered) >= median(str_count(Sequence_filtered)) -
  ↪  length.var & str_count(Sequence_filtered) <= median(str_count(Sequence_filtered)) +
  ↪  length.var) %>%
  mutate(Filter_Status = "Filtered") #changing filter status values to filtered for
  ↪  Boxplot distribution chart color labelling.
```

```r
VCytB.stringSet <- readDNAStringSet("../data/VCytB.fasta")

dfVCytB <- data.frame(NCBI_ID = word(names(VCytB.stringSet), 1L), Species_Name =
↪  word(names(VCytB.stringSet), 2L, 3L), Sequence = paste(VCytB.stringSet), Gene =
↪  "CytB", Filter_Status = "Unfiltered")

dfVCytB <- dfVCytB %>%
  mutate(Sequence_length = nchar(dfVCytB$Sequence))


dfVCytB_filtered <- dfVCytB %>%
  mutate(Sequence_filtered = str_remove_all(Sequence, "^N+|N+$|-")) %>%
  filter(str_count(Sequence_filtered, "N") <= (N.composition.limit *
  ↪  str_count(dfVCytB$Sequence))) %>%
  select(!"Sequence") %>%
  filter(str_count(Sequence_filtered) >= median(str_count(Sequence_filtered)) -
  ↪  length.var & str_count(Sequence_filtered) <= median(str_count(Sequence_filtered)) +
  ↪  length.var) %>%
  mutate(Filter_Status = "Filtered")

#All dfs are combined to compare the distribution of sequence lengths between filtered
↪  and unfiltered dfs to determine if there was significant change in sequence length
↪  variability.

dfV_combined <- bind_rows(dfVCOI, dfVCOI_filtered, dfVCytB, dfVCytB_filtered)

# A Boxplot figure was generated to compare the distribution of sequence lengths, as
↪  aforementioned, using the gglot2 package. 2 plot were made, one showing the outliers
↪  and the other without. This is to show that the filtering step has greatly reduced
↪  the outliers of the graph, and the the remaining sequences are very close in length,
↪  therefore reducing the risk of inaccuracy.
```

```r
SeqLenBoxplot.withOutliers <- ggplot(dfV_combined, aes(x = Gene, y = Sequence_length,
↪  fill = Filter_Status)) +
  labs(title = "Comparison of Sequence Length by Gene and Filter Status - Outliers
  ↪  Included", x = "Gene", y = "Sequence Length") +
  theme(plot.title = element_text(size = 12)) +
  geom_boxplot() +
  scale_y_log10() +
  guides(fill = guide_legend(title = "Filter Status"))

SeqLenBoxplot.withoutOutliers <- ggplot(dfV_combined, aes(x = Gene, y = Sequence_length,
↪  fill = Filter_Status)) +
  geom_boxplot(outlier.shape = NA) +
  labs(title = "Comparison of Sequence Length by Gene and Filter Status - Outliers
  ↪  Removed", x = "Gene", y = "Sequence Length") +
  theme(plot.title = element_text(size = 12)) +
  coord_cartesian(ylim = c(270, 2000)) +
  scale_y_log10() +
  guides(fill = guide_legend(title = "Filter Status"))

ggpubr::ggarrange(SeqLenBoxplot.withoutOutliers, SeqLenBoxplot.withOutliers, labels =
↪  c("A", "B"))
```

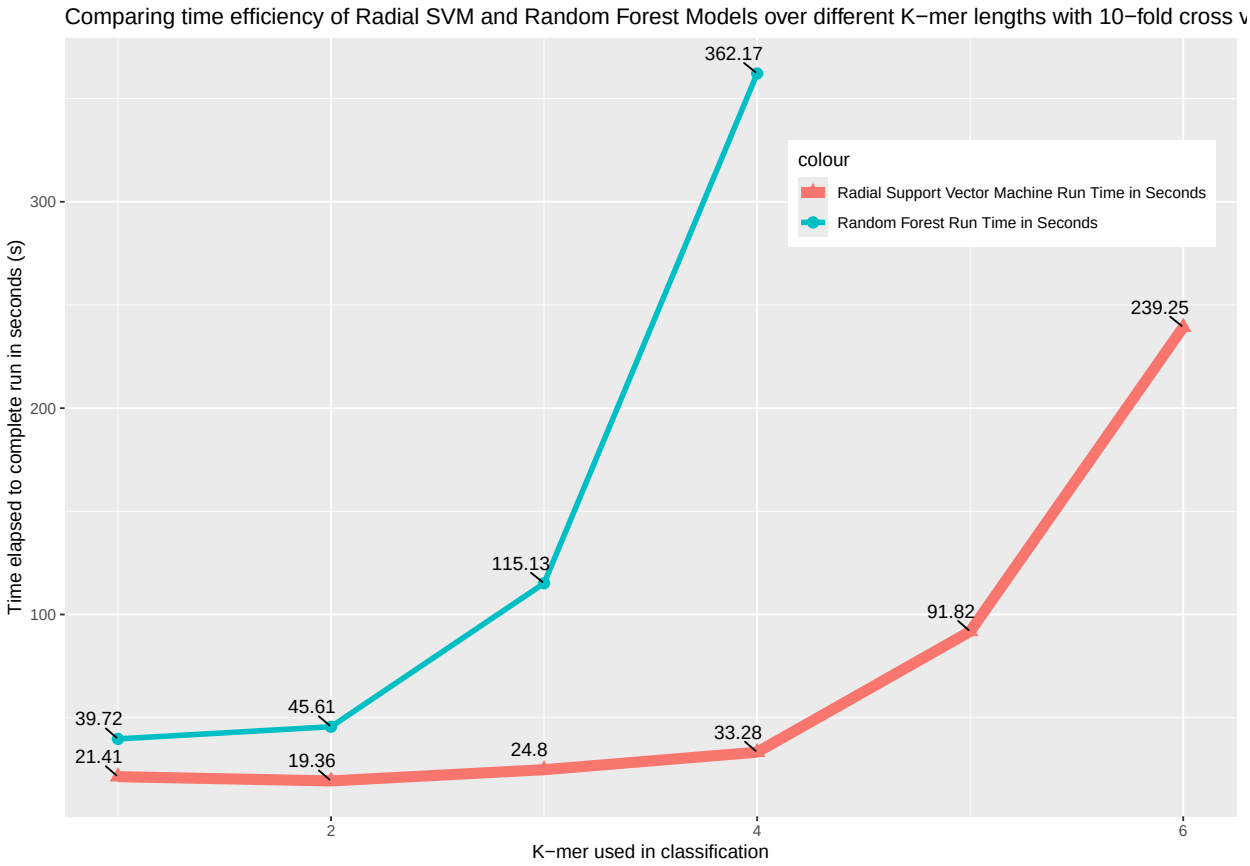



```
#Figure 1: Boxplot demonstrating a significant change in sequence length distribution
↪  between filtered and unfiltered data frames. A. shows the boxplot without the
↪  outliers included, while B. includes the outliers. Red = Filtered datasets, Blue =
↪  unfiltered datasets.

# Main Analysis ----

#The objectives for the main analysis is to: a) compare two popular classifiers in
↪  academia for both accuracy and time efficiency over varying lengths of k-mers, and b)
↪  investigate if COI, a gene described to be associated with immunological capacity in
↪  humans, can be used as a marker to discriminatly cluster bat genera using the faster
↪  model.

# The Caret package is used for the first part of the analysis rather then two distinct
↪  packages pertaining to the two classifiers (Random Forest and Radial Support Vector
↪  Machine). This is done to improve intelligibility of the code, and computational
↪  tractability of the k-fold cross validation tests made available through the Caret
↪  package, when comparing the classifiers. This is not as easily available through the
↪  randomForest package since it uses out-of-bag (OOB) error rates to describe the
↪  accuracy of the model internally.

#Both Random Forest (RF) and Support Vector Machine (SVM) are both highly regarded
↪  classifying models in scientific research. However, both function quite differently
↪  from each other, with RF using many decision trees and optimizing with each tree made
↪  and random samples, while SVM imposes a hyperplane (or if linear, a border) to
↪  develop a classifier boundary. Radial SVM was employed for this analysis since
↪  classification of the data isn't binary.
```

```r
#1000 records from both gene data frames were combined to reduce the risk of bias.

set.seed(290)
dfVGenes.filtered.combined <- rbind(sample_n(dfVCOI_filtered, 1000),
↪  sample_n(dfVCytB_filtered, 1000))

#A function for producing k-mer features to easily implent the Biostrings oligonucleotide
↪  Frequency function  into a for loop.

generate_kmers <- function(sequence, k) {
  oligonucleotideFrequency(DNAString(sequence), width = k, as.prob = T)
}

#Initializing empty data frames to input time and accuarcy for the models downstream.

results.randomForest <- data.frame(k = integer(), accuracy = double())
results.radialSVM <- data.frame(k = integer(), accuracy = double())

#A for loop that iterates over multiple values of K (in this case 1-4) to test the model
↪  accuracy and time elapsed for the Random Forest model. This uses the caret
↪  trainControl and train function from the caret package to use 10-fold cross
↪  validation. This would indicate accuracy of the model by separating the dataset into
↪  k folds and randomly sampling them with unused folds acting as test datasets.

for (k in seq(1, 4)) {
  kmer_features <- t(sapply(dfVGenes.filtered.combined$Sequence_filtered, generate_kmers,
↪  k = k)) #apply the generate_kmers function on all sequences and transpose the data so
↪  the features are variables and not rows. k = k from the for loop sequence.

  labels <- dfVGenes.filtered.combined$Gene #response variables

  cross.validation <- trainControl(method = "cv", number = 10) #specify parameters of
↪  cross validation (10-fold)

  time.taken.seconds <- system.time({
    random.forest.model.compare <- train(x = kmer_features, y = labels, method = "rf",
↪  trControl = cross.validation) #while the random forest model is training, system.time
↪  function records the elapsed time from start to finish
  })["elapsed"]
  results.randomForest <- rbind(results.randomForest, data.frame(k = k, accuracy =
↪  max(random.forest.model.compare$results$Accuracy), model = "Random Forest",
↪  time.taken.seconds = time.taken.seconds))
}

#6 kmers are tested for the radial SVM loop instead of 4. Random forest requires greater
↪  lengths of time to complete training for k lengths > 4. Hence, for the sake of time,
↪  fewer K lengths were used for the RF model.

for (k in seq(1, 6)) {
  kmer_features <- t(sapply(dfVGenes.filtered.combined$Sequence_filtered, generate_kmers,
↪  k = k))
```

```
  labels <- dfVGenes.filtered.combined$Gene

  cross.validation <- trainControl(method = "cv", number = 10)

  time.taken.seconds <- system.time({
    radialSVM.model.compare <- train(x = kmer_features, y = labels, method = "svmRadial",
↪   cost = 1, trControl = cross.validation)
  })["elapsed"]
  results.radialSVM <- rbind(results.radialSVM, data.frame(k = k, accuracy =
↪   max(radialSVM.model.compare$results$Accuracy), model = "svmRadial",
↪   time.taken.seconds = time.taken.seconds))
}

# Graphing comparison of models in line chart, with time being the dependent variable and
↪   k-mer value as the independent variable.
```

```
ggplot() +
  geom_line(data = results.randomForest, aes(x = k, y = time.taken.seconds, color =
  ↪   "Random Forest Run Time in Seconds"), size = 1.5) +
  geom_point(data = results.randomForest, aes(x = k, y = time.taken.seconds, color =
  ↪   "Random Forest Run Time in Seconds"), size = 3, shape = 16) +
  geom_text_repel(data = results.randomForest, aes(x = k, y = time.taken.seconds, label =
  ↪   round(time.taken.seconds, 2)), hjust = 0.9, vjust = -1) +
  geom_line(data = results.radialSVM, aes(x = k, y = time.taken.seconds, color = "Radial
  ↪   Support Vector Machine Run Time in Seconds"), size = 3) +
  geom_point(data = results.radialSVM, aes(x = k, y = time.taken.seconds, color = "Radial
  ↪   Support Vector Machine Run Time in Seconds"), size = 3, shape = 17) +
  geom_text_repel(data = results.radialSVM, aes(x = k, y = time.taken.seconds, label =
  ↪   round(time.taken.seconds, 2)), hjust = 0.9, vjust = -1) +
  theme(legend.position = c(.80, .80)) +
  labs(x = "K-mer used in classification", y = "Time elapsed to complete run in seconds
  ↪   (s)", title = "Comparing time efficiency of Radial SVM and Random Forest Models
  ↪   over different K-mer lengths with 10-fold cross validation")
```

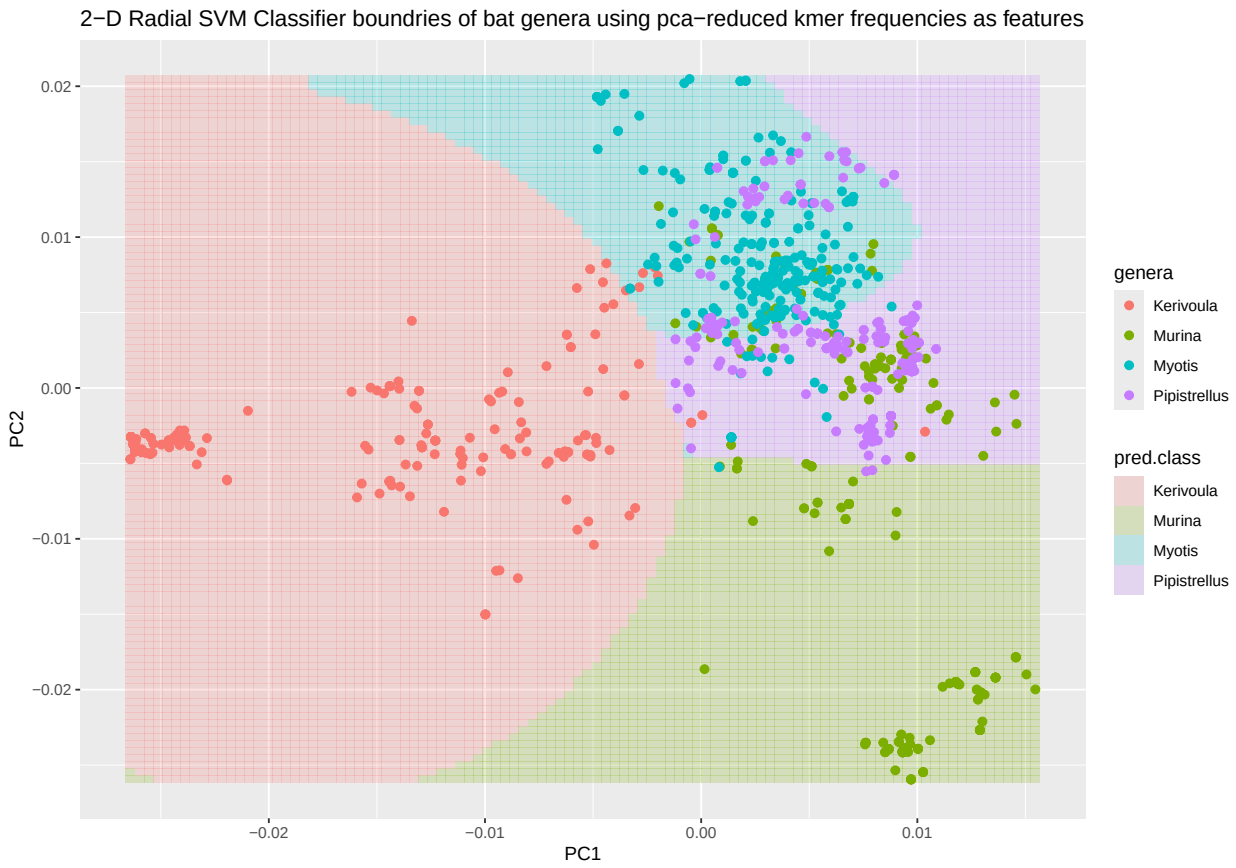



```
#Figure 2. Line plot depicting time needed for each model to complete classification
↪  training with increasing K values. Blue = Random Forest time (s), Red = Radial SVM
↪  time (s).



# Although both models show 100% accuracy over tested k-mers (Random Forest [k = 1 to 4]
↪  and Radial SVM [k = 1 to 6]), Radial SVM is much faster at classification, especially
↪  at greater lengths of K. Therefore, the remainder of the analysis will use Radial
↪  SVM.

# For the second half of the analysis, the radial SVM model is employed to classify
↪  genera of bats that have been described in literature to demonstrate varying profiles
↪  of pathogenesis, leading to different bat taxa serving as reservoirs to distinct
↪  viral families.



dfVCOI_filtered.withGenera <- dfVCOI_filtered %>% #Using the top 4 abundant genera of
↪  bats for the classification analysis
  mutate(genera = word(Species_Name, 1L)) %>%
  filter(genera == "Myotis" | genera == "Murina" | genera == "Pipistrellus" | genera ==
   ↪  "Kerivoula") %>%
  filter(!is.na(Sequence_filtered))
```

```r
#table(dfVCOI_filtered.withGenera)

Kmer.five.feature.VCOI <-
↪  t(sapply(dfVCOI_filtered.withGenera$Sequence_filtered,generate_kmers,k=4)) #Inputing
↪  4-mer sequence features into a designated data frame

dfVCOI_filtered.withGenera <- cbind(dfVCOI_filtered.withGenera,Kmer.five.feature.VCOI)
↪  #Concatenating sequence features with the data frame containing all bat genera of
↪  interest.

set.seed(217)
dfValidation <- dfVCOI_filtered.withGenera %>%
  group_by(genera) %>%
  sample_n(50) #Randomly sampling 50 of each genera of bat for the test set.

set.seed(13)
dfTraining <- dfVCOI_filtered.withGenera %>%
  filter(!NCBI_ID %in% dfValidation$NCBI_ID) %>%
  group_by(genera) %>%
  sample_n(250) #Randomly sampling 250 of each genera of bat for the training set



VCOI.radial.SVM <-  svm(factor(dfTraining[[7]]) ~ ., data = dfTraining[,8:263], scale =
↪  FALSE, kernel = "radial", cost = 1, cross = 10) #Training a radial SVM classification
↪  model using the svm function from the e1071 package, with 10-fold cross validation. A
↪  cost value of 1 was used based on other also studies also conducting taxonomic
↪  clustering based on genomic features and to avoid overfitting(Solis-Reyes et al.,
↪  2018).

#Testing the trained model on the test set.
predictionValidation <- predict(VCOI.radial.SVM, dfValidation[,8:263])

#The confusion matrix indicated an accuracy of 92.5% for classifying the data.
#confusionMatrix(predictionValidation, as.factor(dfValidation$genera))


#To visualize the radial classification boundaries, the dimensionality of the predictor
↪  k-mer features were reduced using Principle componenet analysis (PCA), and a new
↪  model was trained.

pca_features <- prcomp(dfTraining[,8:263]) #Reducing the features into PCs and converting
↪  data into pca object.

df.pca.svm.training <- as.data.frame(pca_features$x[,1:2]) %>% #Extracting only the first
↪  two PCs two use in the 2D classification plot.
  mutate(genera = dfTraining$genera)

VCOI.svm.model.pca <- svm(as.factor(genera)~., data = df.pca.svm.training, kernal =
↪  "radial", cost = 1) #Training a new model to predict the genera using the pca data.
↪  Once again a cost 1 was used.
```

```r
#Defining 100 points of the minimum to maximum values for both PC1 and PC2, adjusting the
↪  plot grid to accommodate points and boundaries of model in the PCA
grid.x <- seq(min(df.pca.svm.training$PC1), max(df.pca.svm.training$PC1), length.out =
↪  100)
grid.y <- seq(min(df.pca.svm.training$PC2), max(df.pca.svm.training$PC2), length.out =
↪  100)
gridx <- expand.grid(PC1 = grid.x, PC2 = grid.y) #Generating grid space in pca graph


ygrid <- predict(VCOI.svm.model.pca,gridx) #predicting the labels of the points in the
↪  PCA
gridx$pred.class <- ygrid #Including the labels back into grid space data to  identify
↪  where each class is located on the pca grid.

#graphing both the area of class occupations (along with the boundaries of
↪  classification) and the points of at which the true data sits on the PCA.
```

```r
ggplot() +
  geom_tile(data = gridx, aes(x = PC1, y = PC2, fill = pred.class), alpha = 0.2) +

  geom_point(data = df.pca.svm.training, aes(x = PC1, y = PC2, color = genera), size = 2)
  ↪  +
  labs(title = "2-D Radial SVM Classifier boundries of bat genera using pca-reduced kmer
  ↪  frequencies as features")
```




```
# Figure 3. 2D Radial SVM classification PCA plot. The although the true points reside
↪  mostly in their respective boundaries, there is still some significant overlap. This
↪  might indicate COI is not informative in the pathogenesis of bats, and specific
↪  immunological genes that play a direct role , such as ARM1, may need to be to be
↪  investigated in future directions.
```

```
# Discussion ----

#Although this analysis was successful in distinctly classifying the different genera of
↪  bats, however, the apparent overlapping of labels between classes may indicate that
↪  COI does not have a significant role in the reservoir profiles of bats. This aligns
↪  well with the review published by Mackenzie et al., (2016), which has suggested
↪  physiological features playing large roles in the protection of bats from virulence,
↪  such as through high body temperature generated by frequent flight. It has also been
↪  suggested the bats contain specialized miRNA's that behave as a defense against
↪  intruding virions (Cowled et al., 2014). This overlap may also be consequence of low
↪  number of sequences for each genera tested.

#Interestingly, radial SVM demonstrated faster run times at all lengths of K , perhaps
↪  due to its specialization in handling non-linear classification tasks, when compared
↪  to Random forest which would run for significantly longer as K lengths increased.
↪  Future studies should investigate miRNA classification of bat genera using radial SVM
↪  as a proxy for elucidating pathogenic profiles of various taxa of bats.

# References ----
```

```
#Literature references:

#1. Solis-Reyes, S., Avino, M., Poon, A., & Kari, L. (2018). An open-source k-mer based
↪  machine learning tool for fast and accurate subtyping of HIV-1 genomes. PLoS ONE,
↪  13(11), e0206409. https://doi.org/10.1371/journal.pone.0206409

#2. Cowled, C., Stewart, C. R., Likic, V. A., Friedländer, M. R., Tachedjian, M.,
↪  Jenkins, K. A., Tizard, M. L., Cottee, P., Marsh, G. A., Zhou, P., Baker, M. L.,
↪  Bean, A. G., & Wang, L. (2014). Characterisation of novel microRNAs in the Black
↪  flying fox (Pteropus alecto) by deep sequencing. BMC Genomics, 15(1), 682.
↪  https://doi.org/10.1186/1471-2164-15-682

#3. Tarasenko, T. N., Pacheco, S. E., Koenig, M. K., Gomez-Rodriguez, J., Kapnick, S. M.,
↪  Diaz, F., Zerfas, P. M., Barca, E., Sudderth, J., DeBerardinis, R. J., Covian, R.,
↪  Balaban, R. S., DiMauro, S., & McGuire, P. J. (2017). Cytochrome c Oxidase Activity
↪  Is a Metabolic Checkpoint that Regulates Cell Fate Decisions During T Cell Activation
↪  and Differentiation. Cell Metabolism, 25(6), 1254-1268.e7.
↪  https://doi.org/10.1016/j.cmet.2017.05.007

#Helpful Tutorials Used:

#1. R Box Plot (With examples). (n.d.). https://www.datamentor.io/r-programming/box-plot

#2. ggplot2: Add points to geom_line. (n.d.). Stack Overflow.
↪  https://stackoverflow.com/questions/12332298/ggplot2-add-points-to-geom-line

#3. Label points in geom_point. (n.d.). Stack Overflow.
↪  https://stackoverflow.com/questions/15624656/label-points-in-geom-point

#4. trainControl function - RDocumentation. (n.d.).
↪  https://www.rdocumentation.org/packages/caret/versions/6.0-92/topics/trainControl

#5. How to change legend title in ggplot. (n.d.). Stack Overflow.
↪  https://stackoverflow.com/questions/14622421/how-to-change-legend-title-in-ggplot

#6. Applying k-fold Cross Validation model using caret package. (n.d.). Stack Overflow.
↪  https://stackoverflow.com/questions/33470373/applying-k-fold-cross-validation-model-using-caret-pac

#7. Kuhn, M. (2019, March 27). 7 train Models By Tag | The caret Package.
↪  https://topepo.github.io/caret/train-models-by-tag.html#support-vector-machines

#8. Le, J. (2018, August 21). Support vector machines in R.
↪  https://www.datacamp.com/tutorial/support-vector-machines-r

#9. Winter, D. (2020, November 11). Rentrez tutorial.
↪  https://cran.r-project.org/web/packages/rentrez/vignettes/rentrez_tutorial.html#using-ncbis-web-his

#10. Archowdhury. (n.d.). Principal-Component-Analysis-PCA-and-SVM-using-R/PCA.R at
↪  master · archowdhury/Principal-Component-Analysis-PCA-and-SVM-using-R. GitHub.
↪  https://github.com/archowdhury/Principal-Component-Analysis-PCA-and-SVM-using-R/blob/master/PCA.R

#11. Plot boxplots over time using multiple categories. (n.d.). Stack Overflow.
↪  https://stackoverflow.com/questions/65207141/plot-boxplots-over-time-using-multiple-categories
```

```
#Package References:

#1. Wickham H, Averick M, Bryan J, Chang W, McGowan L, 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 D, 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.
↪  https://doi.org/10.21105/joss.01686.

#2. Wickham H (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New
↪  York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

#3. Winter DJ (2017). "rentrez: An R package for the NCBI eUtils API." The R Journal,
↪  9(2), 520-526. https://doi.org/10.32614/RJ-2017-058.

#4. Charif D, Lobry JR (2007). "seqinr 1.0-2: a contributed package to the R project for
↪  statistical computing devoted to biological sequences retrieval and analysis." In
↪  Structural Approaches to Sequence Evolution: Molecules, Networks, Populations,
↪  Bastolla U, Porto M, Roman HE, Vendruscolo M (eds.), Springer Verlag, New York. ISBN
↪  978-3-540-35305-8.

#5. Pagès H, Aboyoun P, Gentleman R, DebRoy S (2023). Biostrings: Efficient manipulation
↪  of biological strings. R package version 2.68.1,
↪  https://bioconductor.org/packages/Biostrings.

#6. Kassambara A (2020). ggpubr: 'ggplot2' Based Publication Ready Plots. R package
↪  version 0.4.0, https://CRAN.R-project.org/package=ggpubr.

#7. Kuhn M (2008). "Building Predictive Models in R Using the caret Package." Journal of
↪  Statistical Software, 28(5), 1-26. https://doi.org/10.18637/jss.v028.i05.

#8. Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). "kernlab - An S4 Package for
↪  Kernel Methods in R." Journal of Statistical Software, 11(9), 1-20.
↪  https://doi.org/10.18637/jss.v011.i09.

#9. Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2023). e1071: Misc
↪  Functions of the Department of Statistics, Probability Theory Group (Formerly:
↪  E1071), TU Wien. R package version 1.7-13, https://CRAN.R-project.org/package=e1071.

#Acknowledgements ----

#I would like to thank Mirza Ahmadi for assisting me in developing my question and how to
↪  download my data. Thank you Moiz Ali guiding to tutorials that allow me to generate
↪  these amazing PCA plots. I'd like to give a final thank you to Brittny McIntyre and
↪  Karl Cottnie for answering my questions when I need assistance.
```