Random forest classifier for Insecta taxonomy with k-mer feature selection from CytB sequences

Nathaniel Lesperance

2023-11-05

Start by loading the libraries we'll use for this analysis.

```
library(rentrez)
library(tidyverse)
library(Biostrings)
library(randomForest)
library(DECIPHER)
library(RSQLite)
library(ggplot2)
library(caret)
library(pROC)
```

Set Variables and Parameters

Start by setting some variables to be used throughout the script including taxons of interest, max proportional of ambiguous bases (Ns) allowed, name and approximate length of the gene of interest and some hyperparameters used downstream with the Random Forest algorithm.

```
TAXON1 <- "Coleoptera" # 1st Taxonomy of interest
TAXON2 <- "Lepidoptera" # 2nd Taxonomy of interest

# Max proportion of Ns allowed in sequences
prop_Ns_cutoff <- 0.02

# Gene of interest (name, approx length)
GENE_OI <- c("CytB","800:1200")

# Random forest downstream hyperparameters
num_trees <- 50
kmer <- 8
```

`efetch_FASTAs()` is a helper function to be used for fetching FASTA sequences from NCBI

We'll use `esearch` and `efetch` to get FASTA files of a particular gene and taxon. We'll collect only DNA sequences, write them to a FASTA file and read them into a `DNAStringSet object` for use downstream. The `efetch_FASTAs()` function returns this object.

```
efetch_FASTAs <- function(geneOI, taxon) {
  # Preliminary nucleotide db esearch
  goi_search <- entrez_search(db = "nuccore",
                              term = paste(taxon, "[ORGN] AND",
                                           geneOI[1], "[Gene] AND",
                                           geneOI[2], "[SLEN] AND biomol_genomic[PROP]"))

  # Repeat esearch with retmax as count from esearch result above
  # and use history = T for use with downstream efetch
  goi_search <- entrez_search(db = "nuccore",
                              term = paste(taxon, "[ORGN] AND",
                                           geneOI[1], "[Gene] AND",
                                           geneOI[2], "[SLEN] AND biomol_genomic[PROP]"),
                              retmax = goi_search$count,
                              use_history = T)

  Sys.sleep(5) # Wait 5s so don't query NCBI too frequently

  # Get FASTA sequences from NCBI using efetch
  goi_fastas <- entrez_fetch(db = "nuccore",
                                 rettype = "FASTA",
                                 web_history = goi_search$web_history)
  # Report results
  print(paste(goi_search$count,
              "DNA FASTA sequences fetched matching",
              geneOI[1],
              "from Taxonomic Group",
              taxon))

  # Write fetched sequences to fasta file then read into DNAStringSet
  write(goi_fastas, paste(taxon, "_", geneOI[1],".fasta"), sep = "\n")
  taxon1_seq <- readDNAStringSet(paste(taxon, "_", geneOI[1],".fasta"))

  return(taxon1_seq)
}
```

Get sequences from NCBI

Let's now use the `efetch_FASTAs()` function to get our DNA sequences from NCBI for gene and taxons of interest. With the help of `BrowseSeqs()` from the `DECIPHER` package, we can view the `DNAStringSet` in a web browser as an HTML file.

```
# eFetch FASTA sequences that match taxonomic groups
tax1_seq <- efetch_FASTAs(geneOI = GENE_OI, taxon = TAXON1)
```

```
## [1] "1276 DNA FASTA sequences fetched matching CytB from Taxonomic Group Coleoptera"
```

```
tax2_seq <- efetch_FASTAs(geneOI = GENE_OI, taxon = TAXON2)
```

```
## [1] "3985 DNA FASTA sequences fetched matching CytB from Taxonomic Group Lepidoptera"
```

```
# Prelimimary examination of sequences
BrowseSeqs(tax1_seq)
BrowseSeqs(tax2_seq)
```

It appears there are some sequences that are shorter than most of the others.

Let's build a dataframe from the sequences and examine them a little closer.

```
tax1_df <- data.frame(Taxonomy = TAXON1,
                      Title = names(tax1_seq),
                      Sequence = paste(tax1_seq))
tax2_df <- data.frame(Taxonomy = TAXON2,
                      Title = names(tax2_seq),
                      Sequence = paste(tax2_seq))

# Parse species name from FASTA label
tax1_df$Species <- word(tax1_df$Title, start = 2L, end = 3L)
tax2_df$Species <- word(tax2_df$Title, start = 2L, end = 3L)

# Build quick table to check descriptors for sequences
table(word(tax1_df$Title, start = -1, end = -1))
```

```
##
##       Contig5 mitochondrial       product
##             1          1227            48
```

```
table(word(tax2_df$Title, start = -1, end = -1))
```

```
##
##           cds mitochondrial
##            11          3974
```

```
# Report number of unique species
sprintf("%d initial unique species of %s", length(unique(tax1_df$Species)), TAXON1)
```

```
## [1] "310 initial unique species of Coleoptera"
```

```
sprintf("%d initial unique species of %s", length(unique(tax2_df$Species)), TAXON2)
```

```
## [1] "172 initial unique species of Lepidoptera"
```

Create a helper function (`quality_and_length_filt()`) to be used for quality control

We'll filter gaps as well as Ns at the start and end of the sequence, remove sequences with more Ns than acceptable (using N proportion cutoff at top of script) and apply a length filter to remove sequences with length >1.5*IQR above the 75th percentile or length >1.5*IQR below the 25th percentile. The function will also report some starting, intermediate and final stats.

```r
quality_and_length_filt <- function(tax_df, taxon) {

  # Report and save initial length distribution stats
  summ_stats <- summary(str_count(tax_df$Sequence))
  print(paste(taxon, "length distribution before QC"))
  print(summ_stats)

  print(paste(sum(str_count(tax_df$Sequence, "-")),
              "gaps and", sum(str_count(tax_df$Sequence, "N")),
              "Ns in all", taxon,
              "sequences were detected before QC"))

  # Save quartiles for downstream length filtering
  q1 <- summ_stats[2]
  q3 <- summ_stats[5]
  IQR <- q3 - q1

  # Save number of sequences before QC
  before_length <- nrow(tax_df)


  tax_df <- tax_df %>%
    # Remove starting and ending gaps or Ns
    mutate(Seq2 = str_remove_all(Sequence, "^N+|N+$|-")) %>%
    # Remove sequences with too many Ns
    filter(str_count(Seq2, "N") <= (prop_Ns_cutoff * str_count(Sequence))) %>%
    # Remove sequences too short or too long
    filter(str_count(Seq2) >= (q1 - 1.5 * IQR) &
           str_count(Seq2) <= (q3 + 1.5 * IQR))

  # Report length distribution stats after QC
  print(paste(taxon, "length distribution after QC"))
  print(summary(str_count(tax_df$Sequence)))

  print(paste(sum(str_count(tax_df$Seq2, "-")),
              "gaps and", sum(str_count(tax_df$Seq2, "N")),
              "Ns in all", taxon,
              "sequences were detected after QC"))

  # Report number of sequences before and after QC
  print(paste(taxon, "sequences before QC:", before_length))
  print(paste(taxon, "sequences after QC:", nrow(tax_df)))

  return(tax_df)
}
```

Let's now use this custom function to clean the sequences in the dataframes built previously.

```r
# QC on both sets of sequences
tax1_df_clean <- quality_and_length_filt(tax1_df, TAXON1)
```

```
## [1] "Coleoptera length distribution before QC"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##     807    1096    1134    1063    1137    1161
## [1] "0 gaps and 223 Ns in all Coleoptera sequences were detected before QC"
## [1] "Coleoptera length distribution after QC"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1035    1127    1134    1131    1137    1161
## [1] "0 gaps and 6 Ns in all Coleoptera sequences were detected after QC"
## [1] "Coleoptera sequences before QC: 1276"
## [1] "Coleoptera sequences after QC: 982"
```

```
tax2_df_clean <- quality_and_length_filt(tax2_df, TAXON2)
```

```
## [1] "Lepidoptera length distribution before QC"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   813.0   961.0   991.0   973.1   994.0  1152.0
## [1] "0 gaps and 31 Ns in all Lepidoptera sequences were detected before QC"
## [1] "Lepidoptera length distribution after QC"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   913.0   989.0   991.0   984.8   998.0  1035.0
## [1] "0 gaps and 31 Ns in all Lepidoptera sequences were detected after QC"
## [1] "Lepidoptera sequences before QC: 3985"
## [1] "Lepidoptera sequences after QC: 3408"
```

Let's now create some temporary dataframes for plotting the results of QC and plot the length distributions before QC and after QC.

```
# Create dfs for QC plotting purposes
len1df_B <- tax1_df %>%
  mutate(Source = paste(TAXON1, "Before")) %>%
  mutate(Seq2 = Sequence)

len2df_B <- tax2_df %>%
  mutate(Source = paste(TAXON2, "Before")) %>%
  mutate(Seq2 = Sequence)

len1df_A <- tax1_df_clean %>%
  mutate(Source = paste(TAXON1, "After"))

len2df_A <- tax2_df_clean %>%
  mutate(Source = paste(TAXON2, "After"))

length_df <- rbind(len1df_B, len2df_B, len1df_A, len2df_A)

# Plot before and after sequence length distributions
ggplot(data = length_df, mapping = aes(x = as.vector(str_count(Seq2)), fill = Source)) +
  geom_density(alpha = 0.4, n = 512) +
  labs(x = "Sequence Length (bp)",
       y = "Density",
       title = sprintf("%s and %s Sequence Length Distribution Before and After QC",
                       TAXON1, TAXON2)) +
  theme_bw()
```

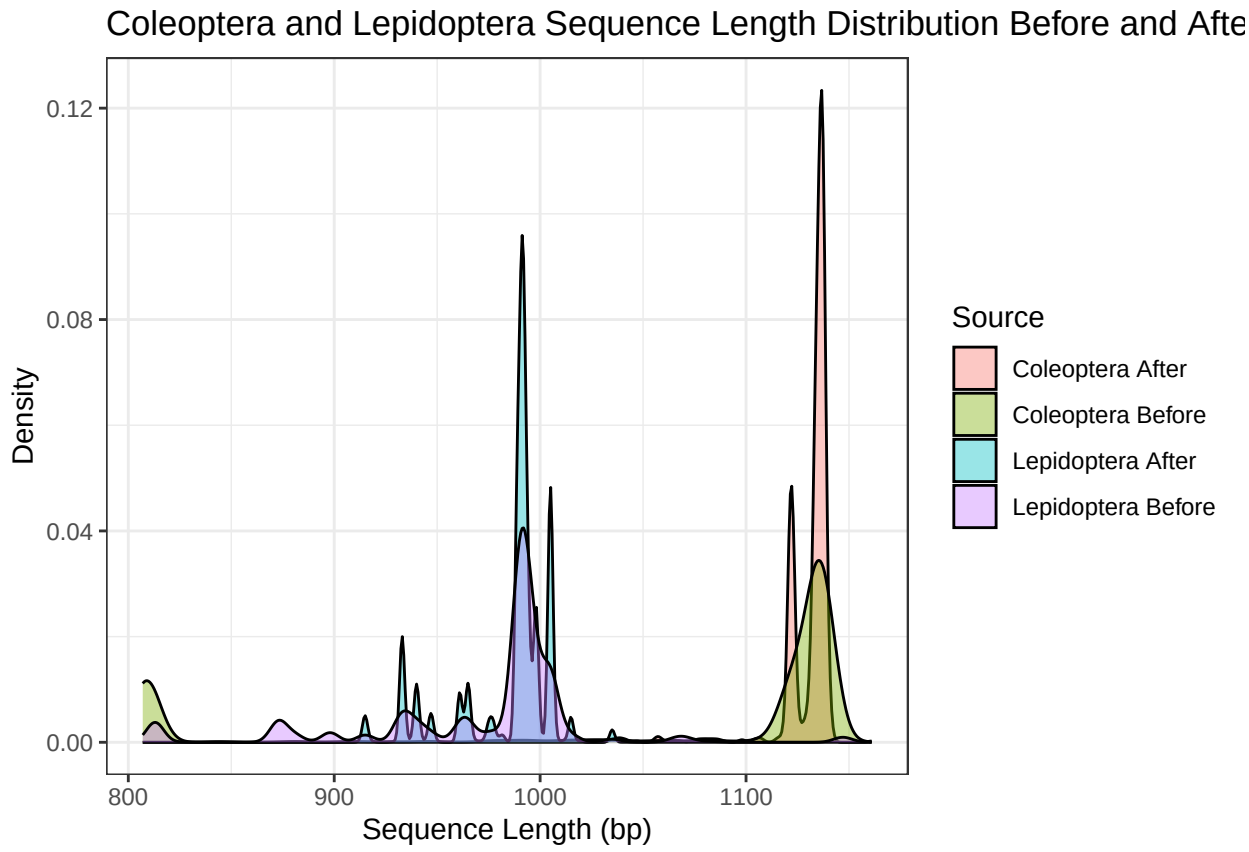


Figure 1: Sequence length distributions for each taxon before and after quality control (QC) filtering. The Coleoptera inter-quartile range of sequence length before QC was 88 bp and after it was 15 bp. The Lepidoptera inter-quartile range of sequence length before QC was 33 bp and after it was 17 bp.

Let's now examine the cleaned sequences in-browser using `BrowseSeqs()`.

```
# Examine Sequences after QC
BrowseSeqs(DNAStringSet(tax1_df_clean$Seq2))
BrowseSeqs(DNAStringSet(tax2_df_clean$Seq2))
```

Let's ensure we are using the same number of sequences from each taxon by taking the minimum number of both. We'll then sample that number of sequences from both, being careful to set the seed to 42 (the answer to the universe) for reproducibility.

```
min_seq <- min(dim(tax1_df_clean)[1],dim(tax2_df_clean)[1])

set.seed(42) # Set seed for reproducible results
tax1_df_clean <- tax1_df_clean %>%
  sample_n(min_seq)

set.seed(42) # Set seed for reproducible results
tax2_df_clean <- tax2_df_clean %>%
  sample_n(min_seq)

# Report number of unique species
sprintf("%d unique species of %s after QC", length(unique(tax1_df$Species)), TAXON1)
```

```
## [1] "310 unique species of Coleoptera after QC"
```

```
sprintf("%d unique species of %s after QC", length(unique(tax2_df$Species)), TAXON2)
```

```
## [1] "172 unique species of Lepidoptera after QC"
```

Extracting k-mer features from sequences

We're going to combine the two separate dataframes from each taxon into a single dataframe and then extract k-mers (k-mer size set at top of script) as features from each sequence.

```
all_taxa <- rbind(tax1_df_clean, tax2_df_clean)

features_df <- data.frame(Taxonomy = all_taxa$Taxonomy,
                          Name = all_taxa$Title,
                          oligonucleotideFrequency(DNAStringSet(all_taxa$Seq2),
                                                   as.prob = T, width = kmer),
                          verbose = TRUE)
```

Let's double check we got the results we expect from the previous step.

```
table(features_df$Taxonomy)
```

```
##
##  Coleoptera Lepidoptera
##         982         982
```

We're almost ready to build an initial Random Forest classifier model

We'll prepare a training and validation set with equal representation of each taxon, being careful to set the seed to 42 (the answer to the universe) for reproducibility.

```
# Create Validation set
set.seed(42)
valid_df <- features_df %>%
  group_by(Taxonomy) %>%
  sample_n(floor(0.2 * nrow(features_df)/2))

# Check equal representation
table(valid_df$Taxonomy)
```

```
##
##  Coleoptera Lepidoptera
##         196         196
```

```
# Create Training set
set.seed(42)
train_df <- features_df %>%
  filter(!Name %in% valid_df$Name)
```

Let's build an initial RF model now!

We'll check each row (sample) has been left out of bag (OOB) sufficient number of times to ensure we have chosen an effective number of trees (set at top of script) and to assist in overfitting prevention. We'll also keep the feature importance matrix to investigate.

```
set.seed(42)
ranfor_classifier <- randomForest(x = train_df[, 3:ncol(train_df)],
                                  y = as.factor(train_df$Taxonomy),
                                  ntree = num_trees,
                                  importance = T)

# Check the OOB
summary(ranfor_classifier$oob.times)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     9.0    16.0    18.0    18.4    21.0    28.0
```

Let's check the performance of the initial classifier we've built by checking predictions on the training and validation sets and show the confusion matrices.

```
# Check performance
ranfor_classifier$confusion
```

```
##             Coleoptera Lepidoptera class.error
## Coleoptera         785           1 0.001272265
## Lepidoptera          0         786 0.000000000
```

```
# Get predictions from training and validation set
predict_val_initial <- predict(ranfor_classifier,
                               valid_df[, 3:ncol(train_df)])
predict_train_initial <- predict(ranfor_classifier,
                                 train_df[, 3:ncol(train_df)])

# Get detailed performance stats to report
confusionMatrix(predict_train_initial,
                as.factor(train_df$Taxonomy))
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Coleoptera Lepidoptera
##   Coleoptera         786           0
##   Lepidoptera          0         786
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
```

```
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0
##             Specificity : 1.0
##          Pos Pred Value : 1.0
##          Neg Pred Value : 1.0
##              Prevalence : 0.5
##          Detection Rate : 0.5
##    Detection Prevalence : 0.5
##       Balanced Accuracy : 1.0
##
##        'Positive' Class : Coleoptera
##
```

```
confusionMatrix(predict_val_initial,
                as.factor(valid_df$Taxonomy))
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Coleoptera Lepidoptera
##   Coleoptera         196           0
##   Lepidoptera          0         196
##
##                Accuracy : 1
##                  95% CI : (0.9906, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0
##             Specificity : 1.0
##          Pos Pred Value : 1.0
##          Neg Pred Value : 1.0
##              Prevalence : 0.5
##          Detection Rate : 0.5
##    Detection Prevalence : 0.5
##       Balanced Accuracy : 1.0
##
##        'Positive' Class : Coleoptera
##
```

Let's look at feature importance by seeing how much the accuracy decreases when we leave out that feature in the model (called mean decrease in accuracy). We'll reorder the features based on this value to determine the most important features.

```
feat_importance <- importance(ranfor_classifier, type = 1)

# Create df of features and their importance to classification
feat_importance <- data.frame(Feature = rownames(feat_importance),
```

```
                                Importance = feat_importance)

# Order by descending importance
feat_importance <- feat_importance[order(-feat_importance$MeanDecreaseAccuracy),]
```

Let's look at the results from the previous steps.

```
# Examine 30 most important
head(feat_importance, n = 30)

# Examine 30 least important
tail(feat_importance, n = 30)
```

How many features have an importance above zero?

```
# Create character vector of features with importance above 0
refined_features <- feat_importance$Feature[feat_importance$MeanDecreaseAccuracy > 0]
length(refined_features)
```

```
## [1] 375
```

411 features have an importance above zero.

Train new models based on feature importance

Let's train a model using fewer features based on the importance examination above. We'll start by setting up some vectors for reporting error rate results.

```
set.seed(42) # So long, and thanks for the fish!

# Create vectors for varying number of features included in the models
num_features <- c(seq(1, 100, 1))
err_rate1 <- vector("numeric", length(num_features))
err_rate2 <- vector("numeric", length(num_features))
val_rate <- vector("numeric", length(num_features))
```

We'll loop through the `num_features` sequence to build several models, starting with a few features and then based on gradually more and more features. Let's keep track of performance (error rate) of each model for each class during training and on the validation set.

```
# Loop through the different number of features, gradually adding more
for (i in 1:length(num_features)) {
  set.seed(42)
  # Build model
  ranfor_classifier_refined <- randomForest(x = train_df[, refined_features[1:(i+1)]],
                                    y = as.factor(train_df$Taxonomy),
                                    ntree = num_trees,
                                    importance = F)
  # Save error rate of last iteration of RF algorithm
  err_rate1[i] <- ranfor_classifier_refined$err.rate[num_trees,2] # Taxon1
```

```
  err_rate2[i] <- ranfor_classifier_refined$err.rate[num_trees,3] # Taxon2

  # Predict on validation set
  predict_val <- predict(ranfor_classifier_refined,
                         valid_df[, refined_features[1:(i+1)]])
  # Save validation error rate
  val_rate[i] <- mean(predict_val == valid_df$Taxonomy)
}
```

Let's prepare some temporary dataframes for plotting the results above and create a plot of error rate against number of features included in the model. We'll incorporate each of the three error rates mentioned above - each taxon during training and combined error rate on the validation set.

```
# Create df for downstream graphing error rate over number of features included in model
graph_err_df <- data.frame(Features = num_features,
                           error_taxon1 = err_rate1,
                           error_taxon2 = err_rate2,
                           val_err_rate = (1-val_rate))

# Plot error rate during training (each taxon) and validation (combined)
ggplot(graph_err_df, aes(x = Features)) +
  geom_smooth(aes(y = error_taxon1, color = paste(TAXON1, "Training"))) +
  geom_smooth(aes(y = error_taxon2, color = paste(TAXON2, "Training"))) +
  geom_smooth(aes(y = val_err_rate, color = "Validation (Mean)")) +
  scale_color_manual(values = c("orange", "black", "blue")) +
  labs(
    title = "Error Rates vs. Features Included in RF Model",
    x = "Number of Features",
    y = "Class Error Rate",
    color = "Dataset Type") +
  theme_bw()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

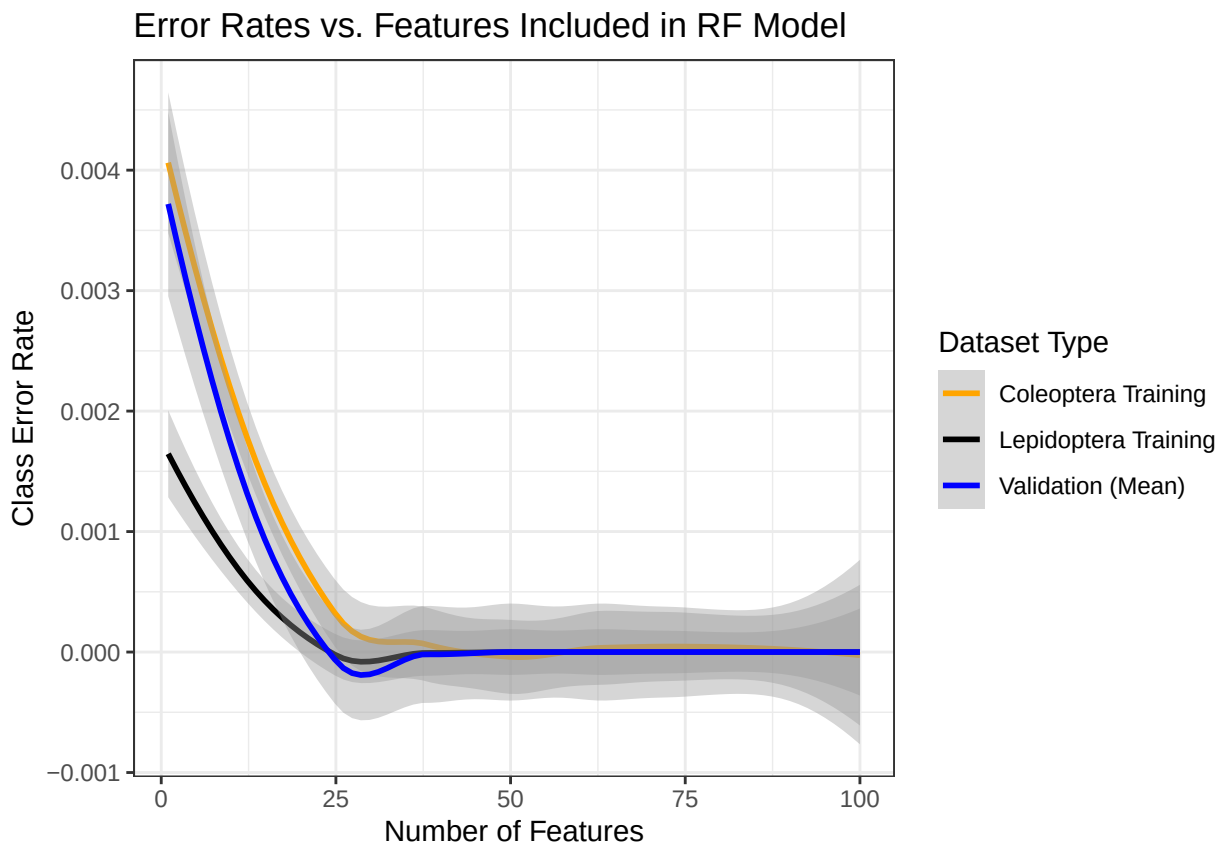
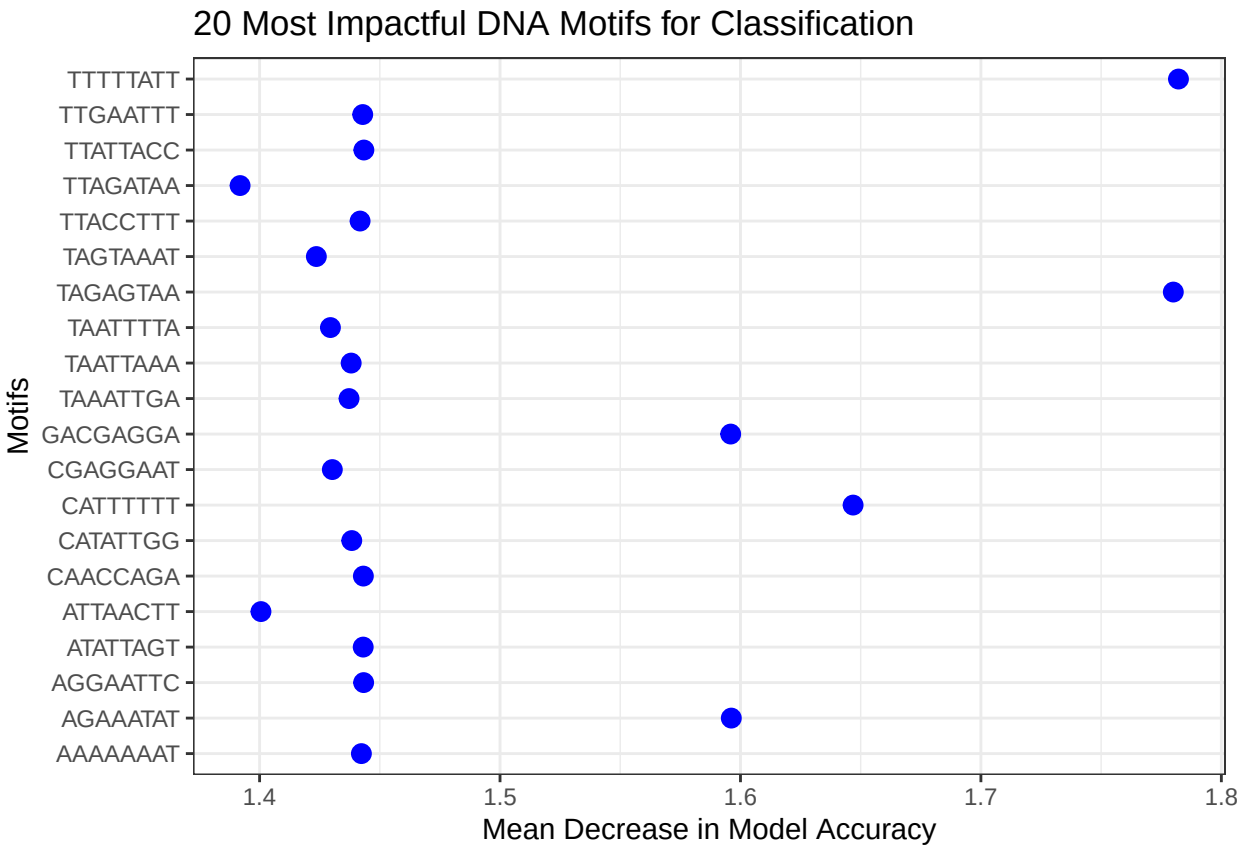


Figure 2: Class error rate from training on each taxonomy of interest and mean class error rate from validation, over number of features included in the model. Shading indicates 95% confidence interval of class error.

Let's finish by showing the actual DNA motifs of each of the top-20 most important features (k-mers) and their mean decrease in accuracy.

```
# Plot 20 most important features (motifs)
ggplot(feat_importance[order(-feat_importance$MeanDecreaseAccuracy)[1:20],],
       aes(x = MeanDecreaseAccuracy, y = Feature)) +
  geom_point(size = 3, color = "blue") +
  labs(title = "20 Most Impactful DNA Motifs for Classification",
       x = "Mean Decrease in Model Accuracy", y = "Motifs") +
  theme_bw()
```


20 Most Impactful DNA Motifs for Classification
Motifs
TTTTTATT
TTGAATTT
TTATTACC
TTAGATAA
TTACCTTT
TAGTAAAT
TAGAGTAA
TAATTTTA
TAATTAAA
TAAATTGA
GACGAGGA
CGAGGAAT
CATTTTTT
CATATTGG
CAACCAGA
ATTAACTT
ATATTAGT
AGGAATTC
AGAAATAT
AAAAAAAT
1.4
1.5
1.6
1.7
1.8
Mean Decrease in Model Accuracy