Random forest taxonomic classifier of Insecta with k-mer feature selection from CytB sequences

Nathaniel Lesperance

2023-11-05

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

Set Variables and Parameters

```r
# Set variables to be used throughout script
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 parameters
num_trees <- 50
kmer <- 8
```

Code Section I:

Create a helper function to be used for efetching FASTA sequences

```r
# Query NCBI with taxon and gene of interest,
# using esearch and efetch to get FASTA files, collecting only DNA sequences and
# write them to a fasta file and read them into a DNAStringSet to be used downstream
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 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,
  # using web history from esearch result above
  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)
}
```

Import Sequences from NCBI

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

```
## [1] "1139 DNA FASTA sequences fetched matching CytB from Taxonomic Group Coleoptera"
```

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

```
## [1] "3376 DNA FASTA sequences fetched matching CytB from Taxonomic Group Lepidoptera"
```

```
# Prelimimary examination of sequences
BrowseSeqs(tax1_seq)
BrowseSeqs(tax2_seq)
# Some shorter sequences
# Name sequences and build df's from them with taxon label
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          1090            48
```

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

```
##
##           cds mitochondrial
##            11          3365
```

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

```
## [1] "236 initial unique species of Coleoptera"
```

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

```
## [1] "160 initial unique species of Lepidoptera"
```

Create a helper function to be used for Quality Control (QC)

```
# Function to filter gaps, Ns and by length
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 greater than 2% Ns
    filter(str_count(Seq2, "N") <= (prop_Ns_cutoff * str_count(Sequence))) %>%
    # Remove sequences with length >1.5 times IQR above 3rd quartile or
    # length >1.5 times IQR below 1st quartile
    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)
}
```

```
# 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    1049    1134    1057    1137    1161
## [1] "0 gaps and 88 Ns in all Coleoptera sequences were detected before QC"
## [1] "Coleoptera length distribution after QC"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     922    1122    1137    1126    1137    1161
## [1] "0 gaps and 15 Ns in all Coleoptera sequences were detected after QC"
## [1] "Coleoptera sequences before QC: 1139"
## [1] "Coleoptera sequences after QC: 886"
```

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

```
## [1] "Lepidoptera length distribution before QC"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   825.0   961.0   991.0   977.4   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   977.0   991.0   982.8   994.0  1035.0 
## [1] "0 gaps and 31 Ns in all Lepidoptera sequences were detected after QC"
## [1] "Lepidoptera sequences before QC: 3376"
## [1] "Lepidoptera sequences after QC: 2959"
```

```
# 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()
```

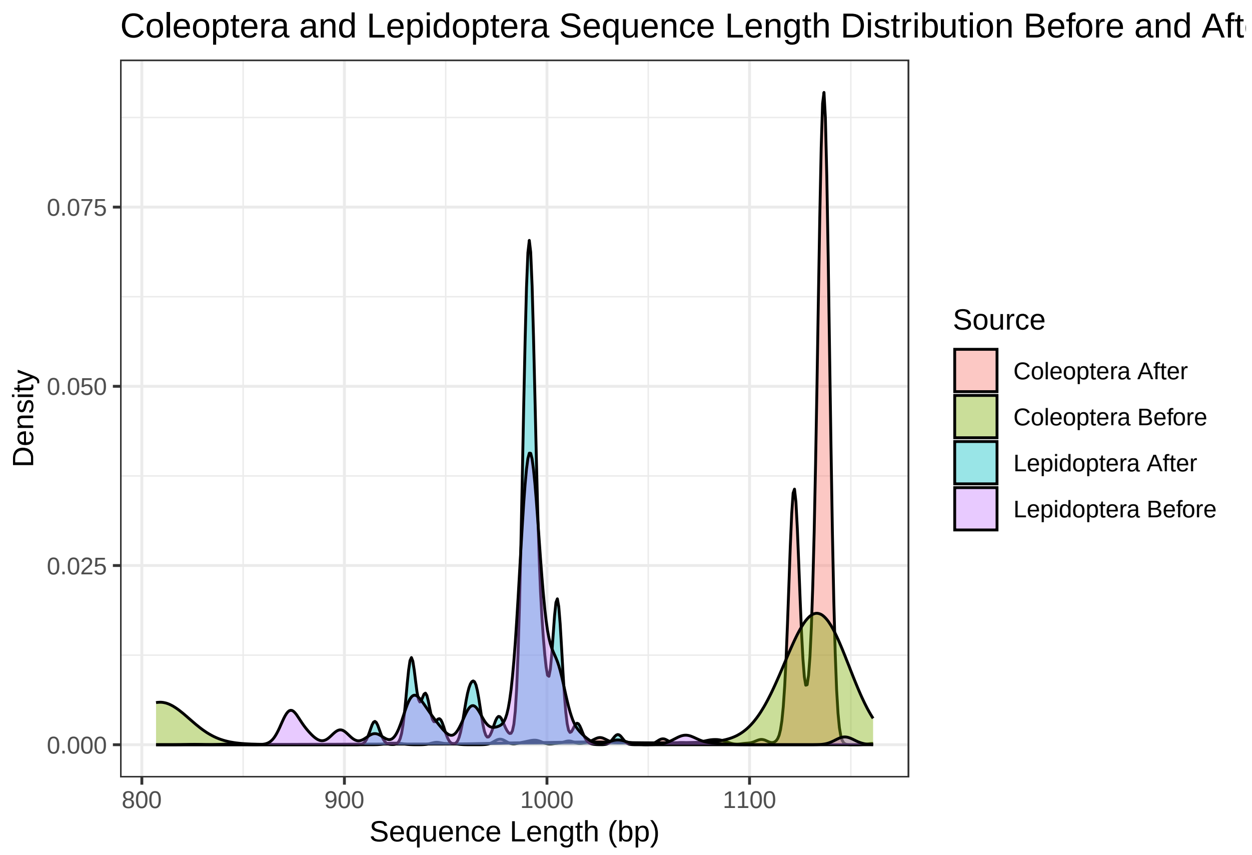


Figure 1: Sequence length distributions for each taxon before and after quality control (QC) filtering. The Coleoptera inter-quartile range of sequence length before QC was 88 bp and after it was 15 bp. The Lepidoptera inter-quartile range of sequence length before QC was 33 bp and after it was 17 bp.

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

# Ensure using same number of sequences from each
min_seq <- min(dim(tax1_df_clean)[1],dim(tax2_df_clean)[1])

set.seed(42)
tax1_df_clean <- tax1_df_clean %>%
  sample_n(min_seq)

set.seed(42)
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] "236 unique species of Coleoptera after QC"
```

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

```
## [1] "160 unique species of Lepidoptera after QC"
```

Code Section II:

Extract k-mer Features from Sequences

```r
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)

# Check results of previous steps
table(features_df$Taxonomy)
```

```
##
##  Coleoptera Lepidoptera
##         886         886
```

Build Initial RF Classifier

```r
# 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
##         177         177
```

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

```r
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 that each row has been left out of bag sufficient number of times
# (To check if effetive number of trees chosen)
summary(ranfor_classifier$oob.times)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.00   16.00   18.00   18.31   20.00   32.00
```

```
# Check initial classifier performance
ranfor_classifier$confusion
```

```
##             Coleoptera Lepidoptera class.error
## Coleoptera         709           0           0
## Lepidoptera          0         709           0
```

```
# Get predictions from training and validation set to get more detailed performance stats
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         709           0
##   Lepidoptera          0         709
##
##                Accuracy : 1
##                  95% CI : (0.9974, 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         177           0
##   Lepidoptera          0         177
##
##                Accuracy : 1
##                  95% CI : (0.9896, 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
##
```

```
# Use Mean Decrease in Accuracy to get measure of importance for each feature
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),]
```

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

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

# Create character vector of features with importance above 0
refined_features <- feat_importance$Feature[feat_importance$MeanDecreaseAccuracy > 0]
length(refined_features)  # 411 features
```

Train new models using fewer features

```r
# Set seed for consistent randomization
set.seed(42)

# 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))

# Loop through the different number of features, gradually adding features
for (i in 1:length(num_features)) {
  set.seed(42)
  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]
  err_rate2[i] <- ranfor_classifier_refined$err.rate[num_trees,3]

  # Predict classification 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)
}

# 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 from identification of each taxon during training
# along with combined error in validation
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’
```

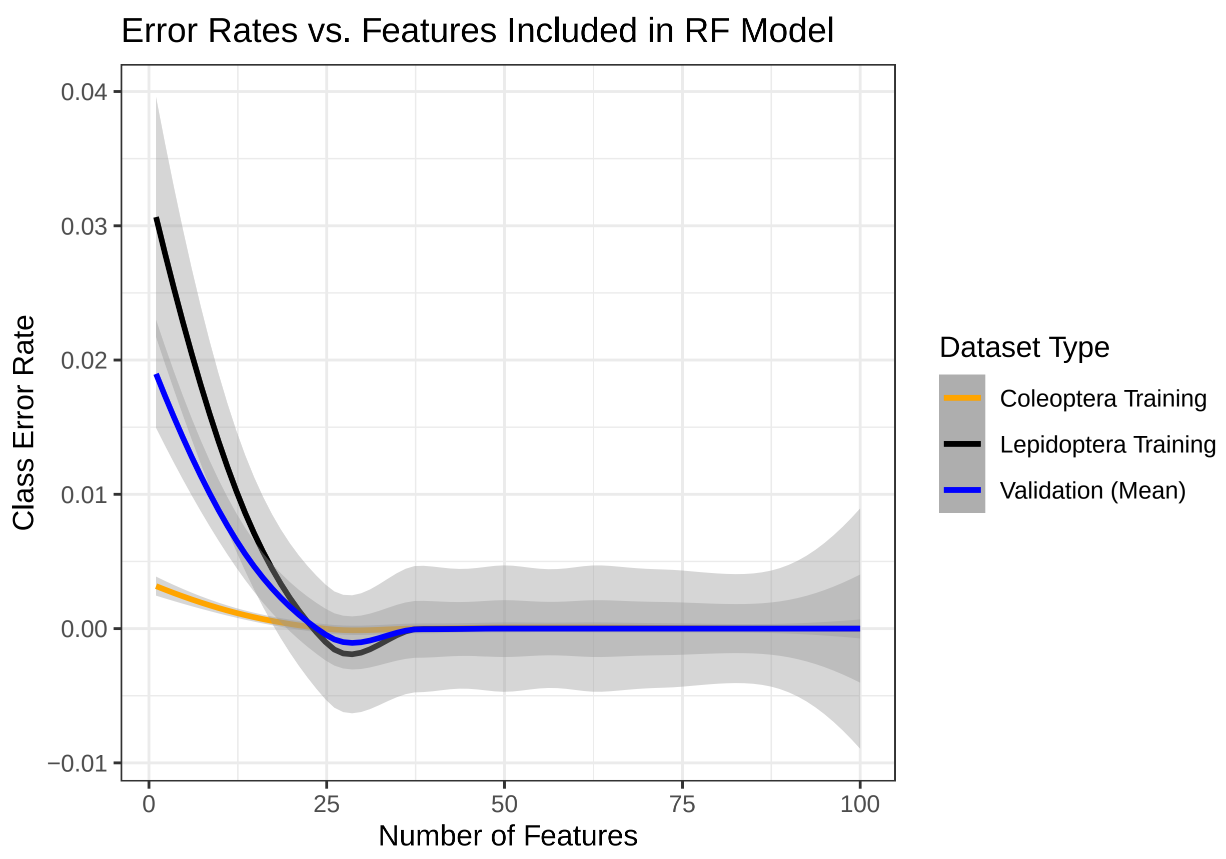
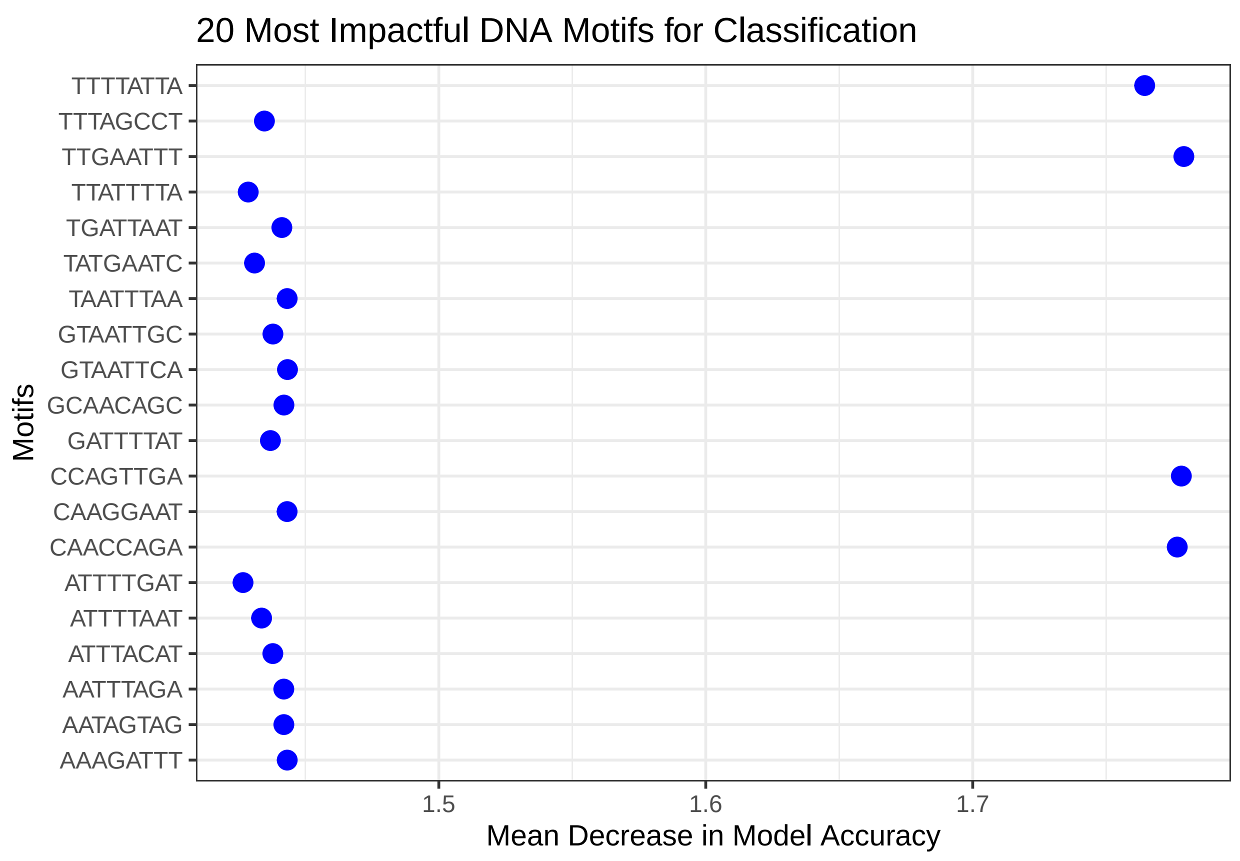


Figure 2: Class error rate from Training on each taxonomy of interest and mean class error rate from Validation, over number of features included in the model. Shading indicates 95% confidence interval of class error.

```
# Create plot showing 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
TTTTATTA
TTTAGCCT
TTGAATTT
TTATTTTA
TGATTAAT
TATGAATC
TAATTTAA
GTAATTGC
GTAATTCA
GCAACAGC
GATTTTAT
CCAGTTGA
CAAGGAAT
CAACCAGA
ATTTTGAT
ATTTTAAT
ATTTACAT
AATTTAGA
AATAGTAG
AAAGATTT
1.5
1.6
1.7
Mean Decrease in Model Accuracy