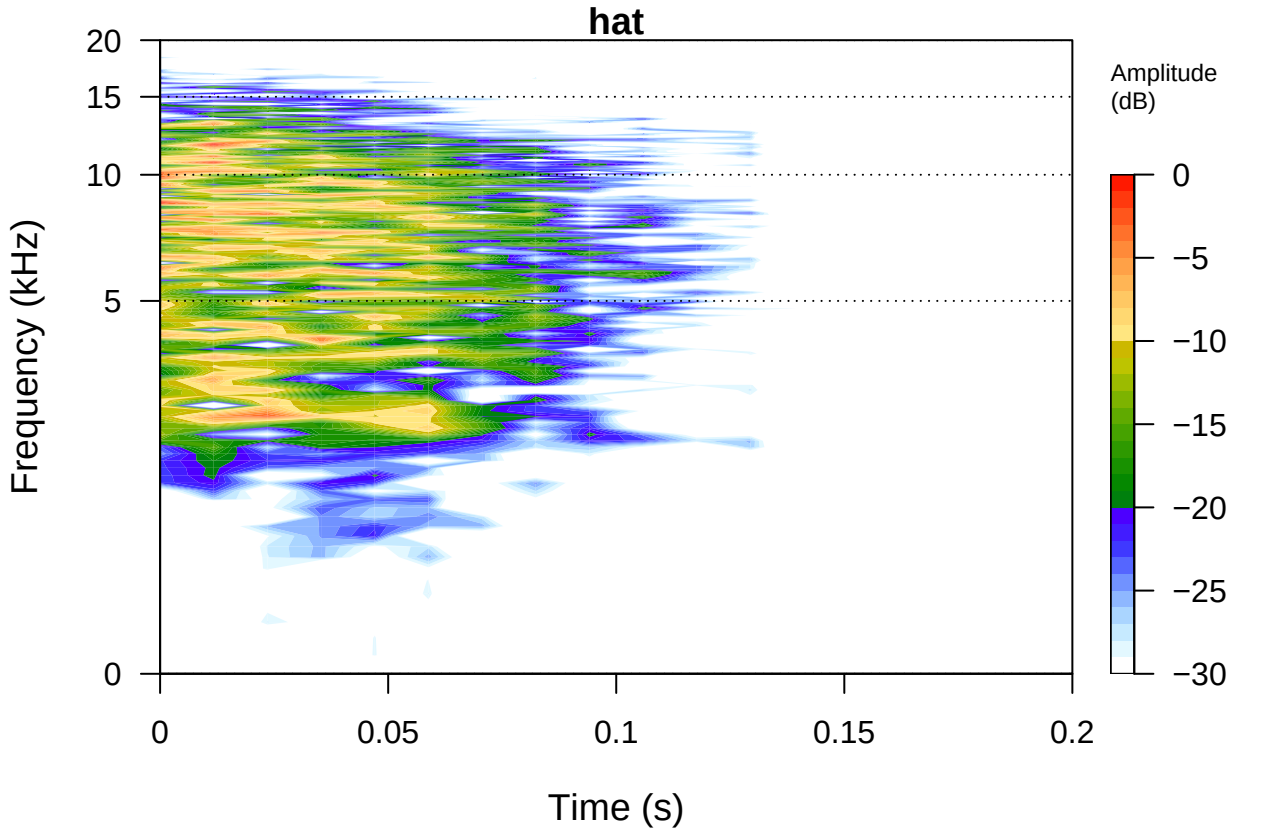
Waveform Analysis

Read in files and convert to Wave class type. This is a custom dataset of 150 percussion sounds, 50 of each class: snare drum, closed hi-hat, and kick drum.

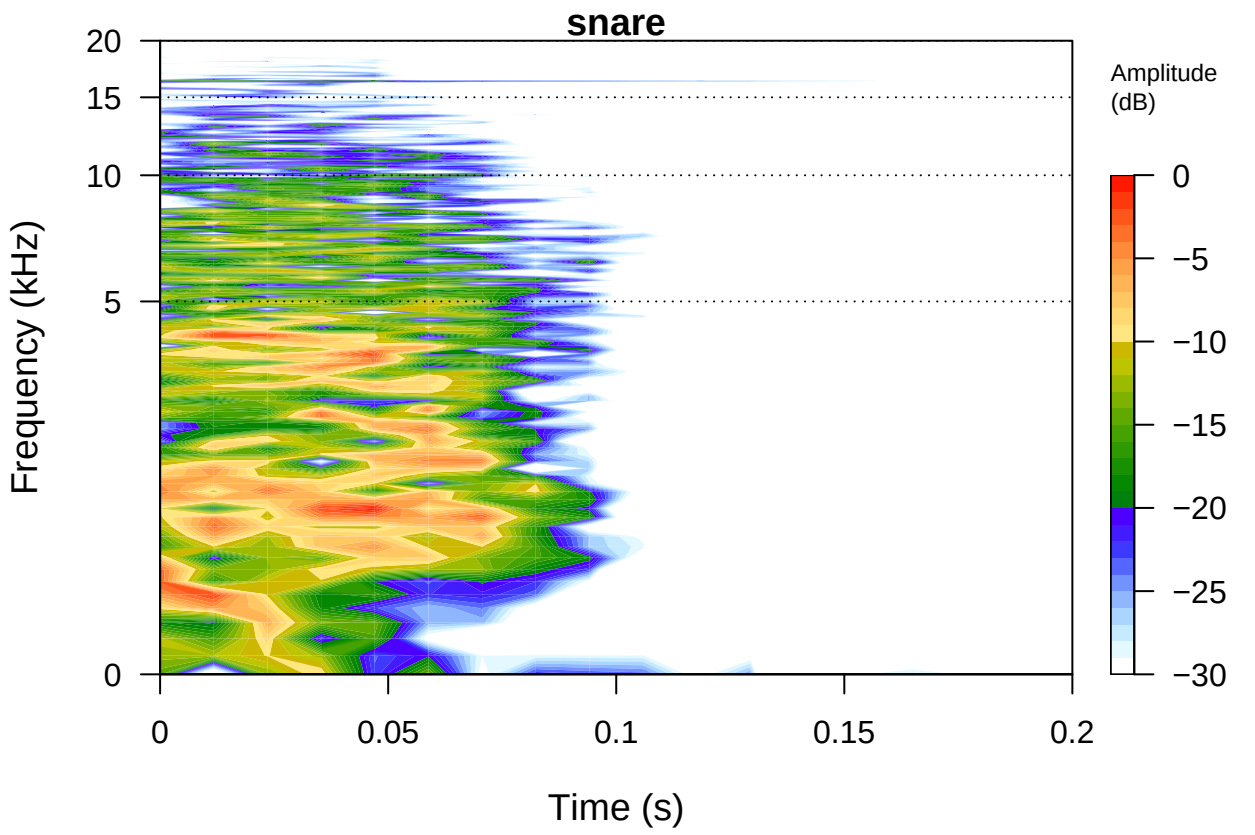
```
filenames <- list.files("150data", pattern="*.wav", full.names=TRUE)
ldf <- lapply(filenames, tuneR::readWave)
```

Plot spectrograms:

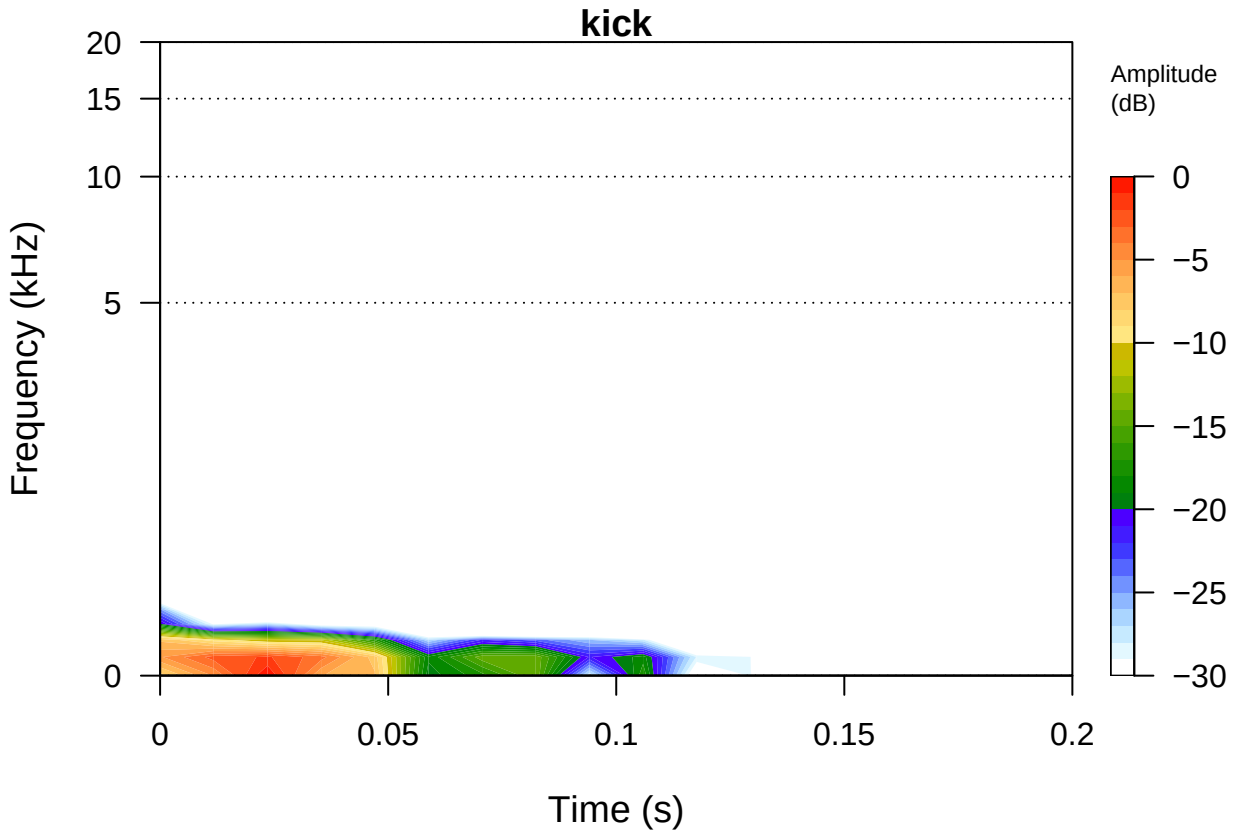
```
hat <- ldf[[3]]
seewave::spectro(hat, f=48000, flog=TRUE, flim=c(0,20), tlim=c(0,0.2), main='hat')
```




```
snare <- ldf[[115]]
seewave::spectro(snare, f=48000, flog=TRUE, flim=c(0,20), tlim=c(0,0.2), main='snare')
```

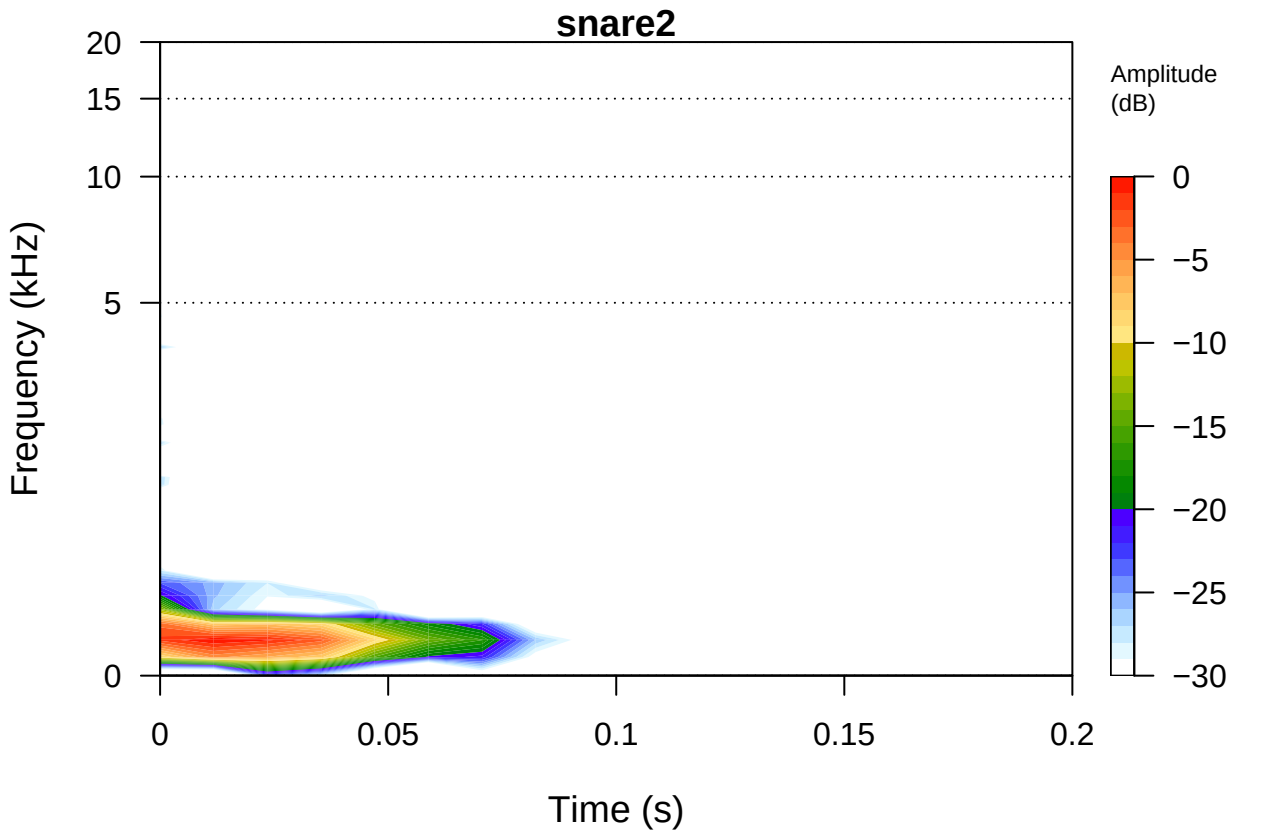



```
kick <- ldf[[51]]
seewave::spectro(kick, f=48000, flog=TRUE, flim=c(0,20), tlim=c(0,0.2), main='kick')
```

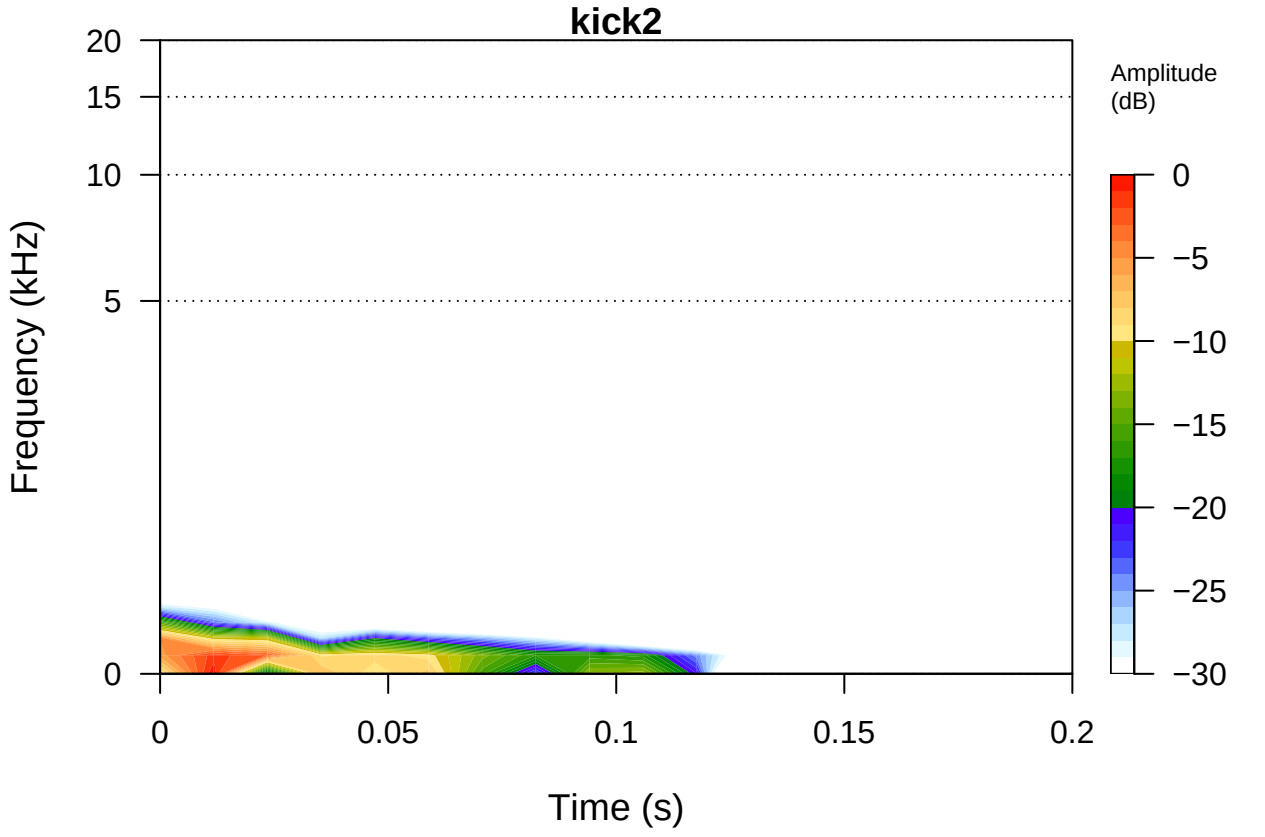


These examples are easy to tell apart, as each instrument occupies a distinct frequency range. This indicates that an audio feature such as dominant frequency may be effective in distinguishing each instrument. However, in the training data this separation is not always the case:

```
snare2 <- ldf[[101]]
kick2 <- ldf[[52]]
seewave::spectro(snare2, f=48000, flog=TRUE, flim=c(0,20), tlim=c(0,0.2), main='snare2')
```




```
seewave::spectro(kick2, f=48000, flog=TRUE, flim=c(0,20), tlim=c(0,0.2), main='kick2')
```

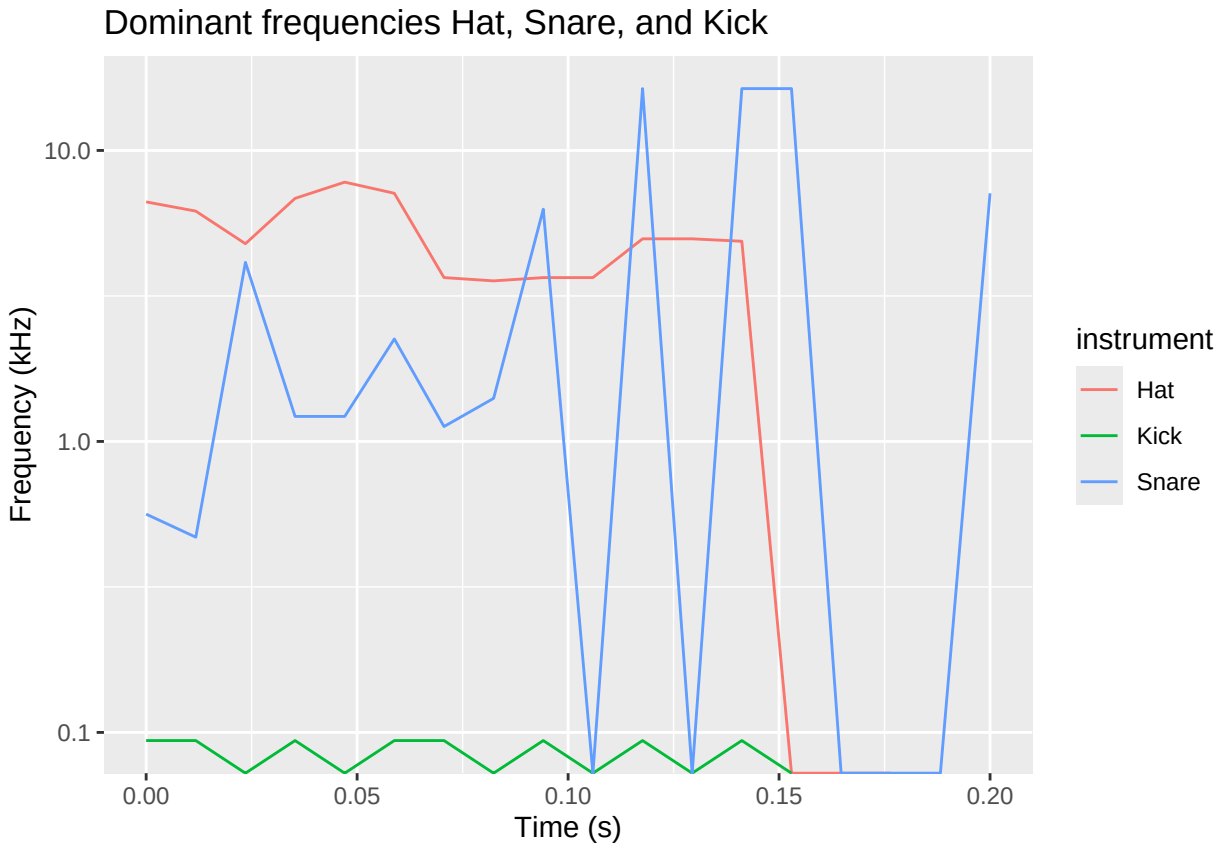


Here, both the snare and kick drum share similar spectrogram data, which suggest a simple audio analysis of dominant frequency may not be sufficient.

```
hatDfreq <- data.frame(seewave::dfreq(ldf[[2]], plot=FALSE, tlim=c(0,0.2), flim=c(0,20)))
snareDfreq <- data.frame(seewave::dfreq(ldf[[115]], plot=FALSE, tlim=c(0,0.2), flim=c(0,20)))
kickDfreq <- data.frame(seewave::dfreq(ldf[[51]], plot=FALSE, tlim=c(0,0.2), flim=c(0,20)))

hatDfreq$instrument <- "Hat"
snareDfreq$instrument <- "Snare"
kickDfreq$instrument <- "Kick"
combined_df <- rbind(hatDfreq, snareDfreq, kickDfreq)

ggplot(data = combined_df, aes(x = x, y = y, color = instrument)) +
  geom_line() +
  scale_y_log10() +
  labs(title = "Dominant frequencies Hat, Snare, and Kick") +
  xlab("Time (s)") +
  ylab("Frequency (kHz)")
```



Because of this, I will experiment with other audio features in addition to dominant frequency.

```
#medians of amplitude envelopes
medians <- sapply(ldf, seewave::M, simplify = TRUE)

#dominant frequencies each wave
Dfreq <- lapply(ldf, seewave::dfreq, plot=FALSE)
maxes <- as.data.frame(Dfreq) %>% select(-starts_with("x"))
maxes <-sapply(maxes, function(x) max(x, na.rm = TRUE), simplify = TRUE)
```

```r
#spectral flatnesses
specs <- lapply(ldf, spec, plot=FALSE)
sfms <- sapply(specs, sfm, simplify = TRUE)

#mfccs
M1 <- extract_features(filenames,
  features = c("mfcc"),
  check.mono=FALSE,
  windowShift = 20,
  numcep = 4)
mfccs <- subset(M1, section_seq_file == 1)
mfccdf <- mfccs[, -c(1:3)]
```

Train Random Forest Classifier

Go through training/testing data and add each file to a class based on its filename.

```
# Function to trim strings based on substrings
trim_to_instrument <- function(string) {
  if (grepl("snare", string, ignore.case = TRUE)) {
    return("snare")
  } else if (grepl("hihat", string, ignore.case = TRUE)) {
    return("hat")
  } else if (grepl("kick", string, ignore.case = TRUE)) {
    return("kick")
  } else {
    stop("Error: No match found for instrument in string '", string, "'")
  }
}

class_list <- sapply(filenames, trim_to_instrument)
class_list <- factor(class_list)
```

Combine into single dataframe.

```
data <- data.frame(medians, maxes, sfms, class_list)
rownames(data) <- NULL
df <- cbind(data, mfccdf)
head(df)
```

```
##        medians    maxes      sfms class_list    mfcc1      mfcc2      mfcc3
## 1 2.689808e-05  4.68750 0.2046096        hat 179.1318  -8.418625 -22.226650
## 2 1.146886e-05  7.78125 0.3320101        hat 183.1877 -11.421466 -10.217383
## 3 5.199873e-07 16.78125 0.3741705        hat 172.1669 -23.335483 -17.090043
## 4 1.891459e-06  9.37500 0.4017943        hat 178.1639 -14.888476  -2.462827
## 5 1.624056e-06 13.59375 0.3160590        hat 168.3710 -21.557909  -5.593358
## 6 1.441962e-06 11.25000 0.3508222        hat 179.1779 -16.524507  -9.372708
##       mfcc4
## 1  5.498373
## 2 10.919913
## 3  8.895906
## 4 10.691643
## 5 -3.346226
## 6  7.209761
```

Convert class list into factor variables and separate into test and training data.

```
set.seed(222)
ind <- sample(2, nrow(df), replace = TRUE, prob = c(0.7, 0.3))
train <- df[ind==1,]
test <- df[ind==2,]

df$class_list <- as.factor(df$class_list)
table(df$class_list)
```

```
##
##   hat  kick snare
##    50    50    50
```

\textbf{Create random forest classifier using R Random Forest package and check work by classifying training data. Accuracy should be 100%.}

```
rf <- randomForest(class_list~., data=train, proximity=TRUE, ntree=150)
print(rf)
```

```
##
## Call:
##  randomForest(formula = class_list ~ ., data = train, proximity = TRUE,      ntree = 150)
##                Type of random forest: classification
##                      Number of trees: 150
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 12.87%
## Confusion matrix:
##       hat kick snare class.error
## hat    30    0     2   0.0625000
## kick    0   30     5   0.1428571
## snare   2    4    28   0.1764706
```

```
p1 <- predict(rf, train)
confusionMatrix(p1, train$ class_list)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction hat kick snare
##      hat    32    0     0
##      kick    0   35     0
##      snare   0    0    34
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9641, 1)
##     No Information Rate : 0.3465
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: hat Class: kick Class: snare
## Sensitivity              1.0000      1.0000       1.0000
## Specificity              1.0000      1.0000       1.0000
## Pos Pred Value           1.0000      1.0000       1.0000
## Neg Pred Value           1.0000      1.0000       1.0000
## Prevalence               0.3168      0.3465       0.3366
## Detection Rate           0.3168      0.3465       0.3366
## Detection Prevalence     0.3168      0.3465       0.3366
## Balanced Accuracy        1.0000      1.0000       1.0000
```

Classify test data and output confusion matrix.

```
p2 <- predict(rf, test)
confusionMatrix(p2, test$ class_list)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction hat kick snare
##      hat    18    0     3
##      kick    0   14     0
##      snare   0    1    13
##
## Overall Statistics
##
##                Accuracy : 0.9184
##                  95% CI : (0.804, 0.9773)
##     No Information Rate : 0.3673
##     P-Value [Acc > NIR] : 9.58e-16
##
##                   Kappa : 0.8767
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: hat Class: kick Class: snare
## Sensitivity              1.0000      0.9333       0.8125
## Specificity              0.9032      1.0000       0.9697
## Pos Pred Value           0.8571      1.0000       0.9286
## Neg Pred Value           1.0000      0.9714       0.9143
## Prevalence               0.3673      0.3061       0.3265
## Detection Rate           0.3673      0.2857       0.2653
## Detection Prevalence     0.4286      0.2857       0.2857
## Balanced Accuracy        0.9516      0.9667       0.8911
```

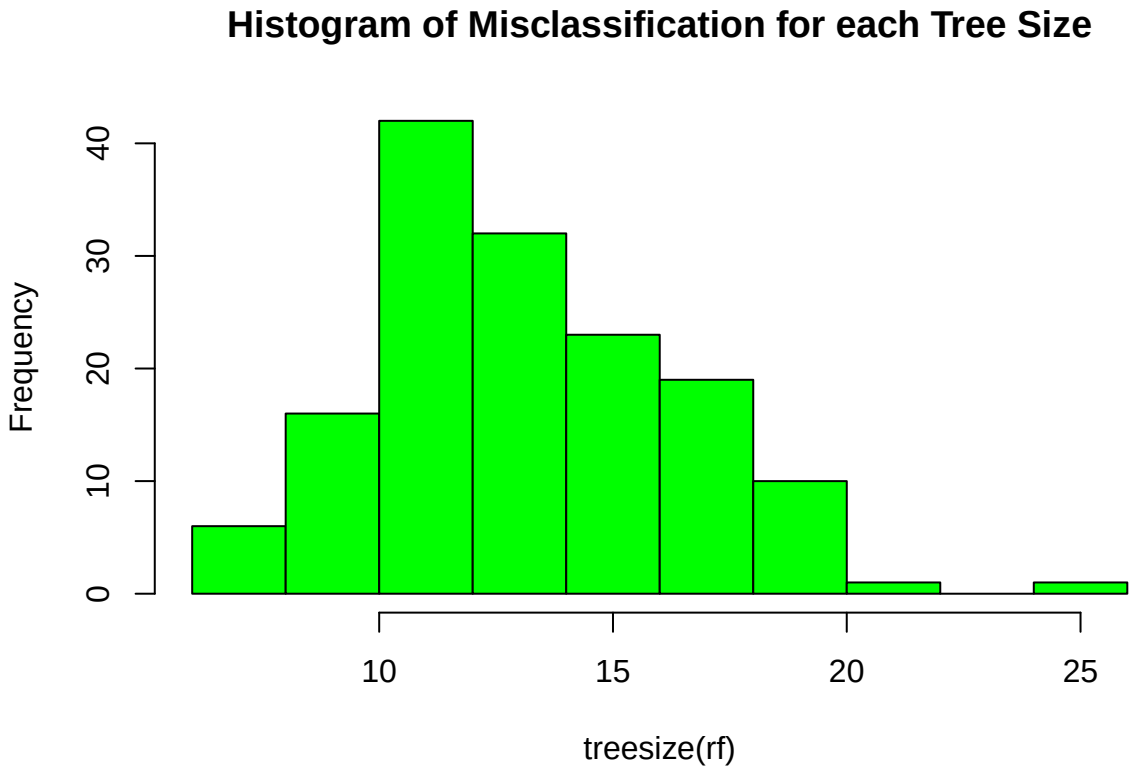
Experiment with RandomForest's tuneRF function to explore different ntree values. This didn't seem to be very helpful given that the tuning method has no patience for anything besides a decrease in error.

```
t <- tuneRF(train[,-4], train[,4],
       stepFactor = 5,
       plot = FALSE,
       ntreeTry = 50,
       trace = TRUE,
       doBest=TRUE,
       improve = 0.01)
```

```
## mtry = 2  OOB error = 13.86%
## Searching left ...
## mtry = 1     OOB error = 18.81%
## -0.3571429 0.01
## Searching right ...
## mtry = 7     OOB error = 14.85%
## -0.07142857 0.01
```

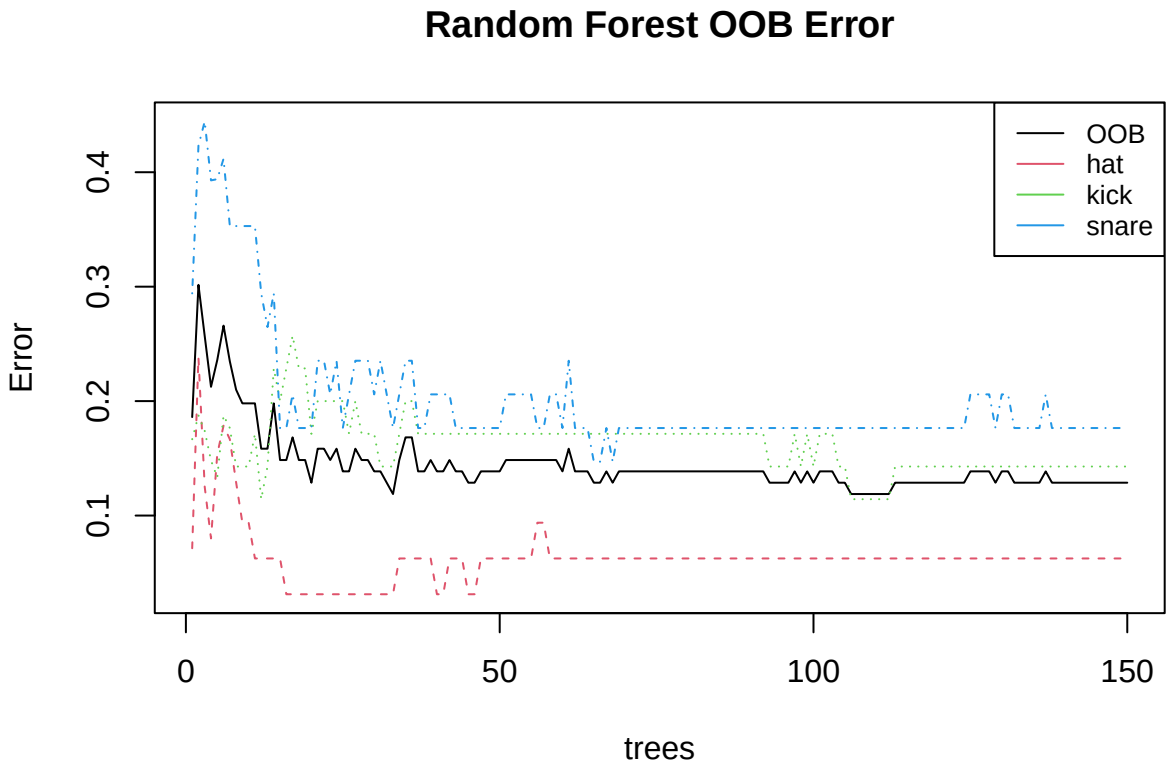
Classifier Analysis

```
hist(treesize(rf),
     main = "Histogram of Misclassification for each Tree Size",
     col = "green")
```



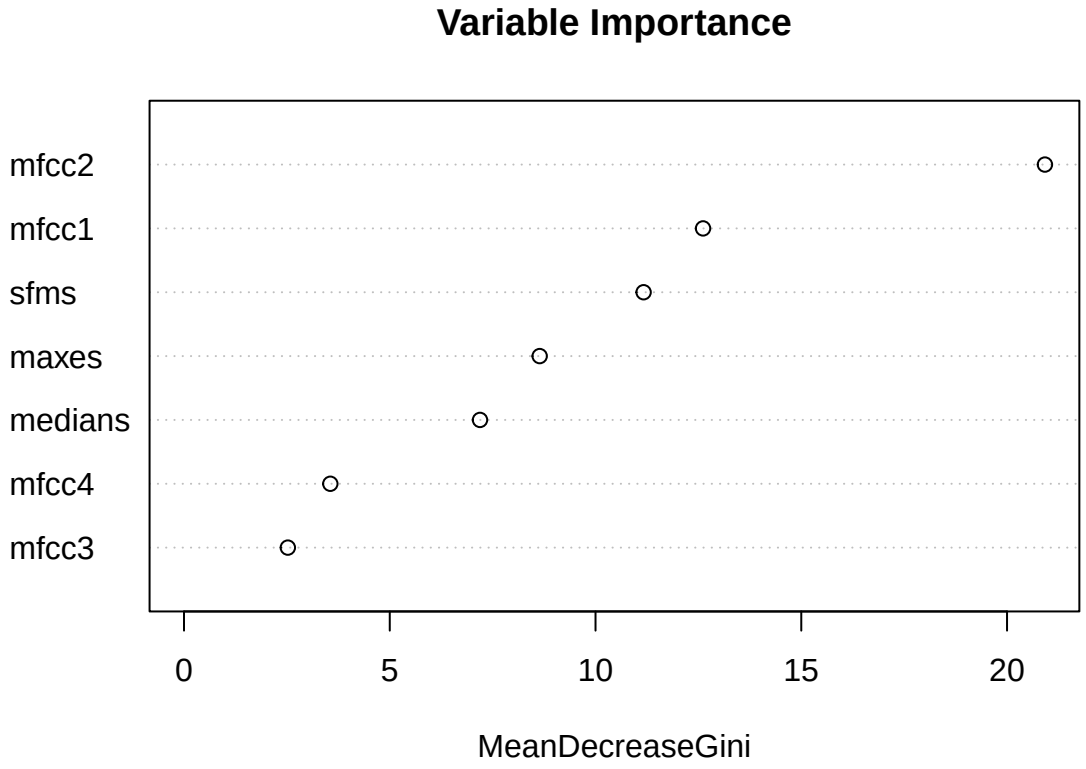

```
plot(rf, main = "Random Forest OOB Error")

legend("topright",
       legend = colnames(rf$err.rate),
       col    = 1:ncol(rf$err.rate),
       lty    = 1,
       cex    = 0.8
)
```



Out-of-bag error plot. Black line represents overall classification error rate, with the 'kick' and 'snare' classes sitting consistently above it. 'Snare' class exhibits the highest error and the 'hat' class remains consistently the lowest. This indicates that the model struggles most with snares, performs moderately on kick drums, and finds closed hi-hats easiest to classify.

```
varImpPlot(rf, n.var = 7,main = "Variable Importance")
```

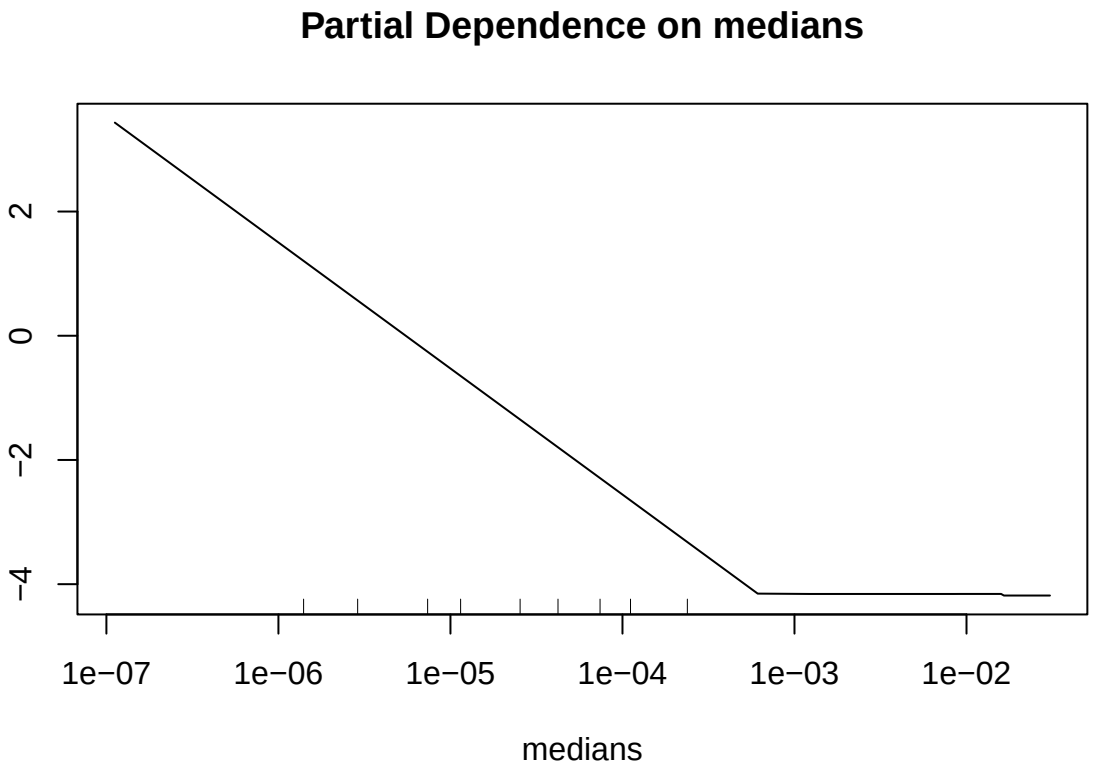


```r
importance(rf)
```

```
##         MeanDecreaseGini
## medians         7.192921
## maxes           8.643237
## sfms           11.168097
## mfcc1          12.612912
## mfcc2          20.920772
## mfcc3           2.523114
## mfcc4           3.555317
```

Dotchart of variable importance as measured by a Random Forest. This indicates that the second-window mfcc value is the most useful variable in determining instrument class.

```r
partialPlot(rf, train, medians, "hat", log = "x")
```

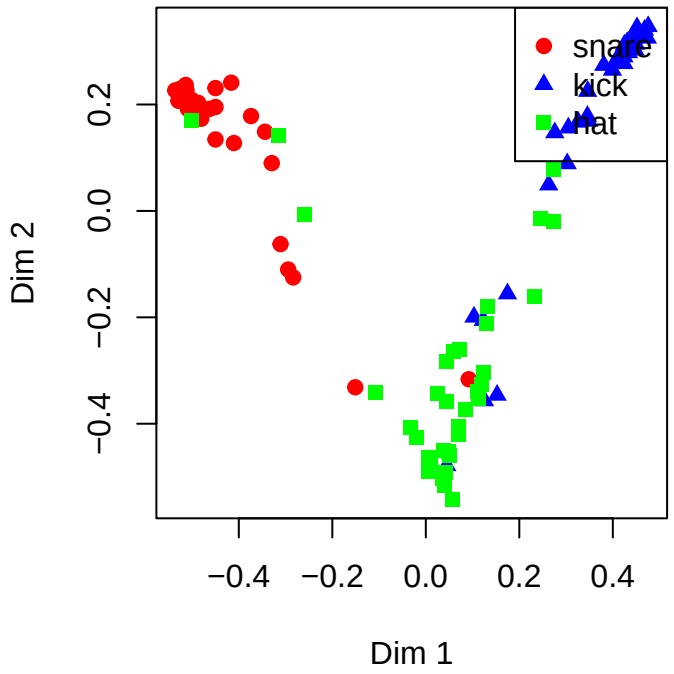


Partial dependence plot gives a graphical depiction of the marginal effect of a variable on the class probability (classification). X-axis represents variable value, Y-axis represents partial dependence (on the logit scale) of predicting the given class. Here we can see the partial dependence on median frequncy for classifying a hi-hat.

```r
MDSplot(rf,
        train$class_list,
        palette = c("red", "blue", "green"),
        pch = c(19, 17, 15)[as.numeric(train$class_list)]
)

legend("topright",
       legend = c("snare", "kick", "hat"),
       col    = c("red", "blue", "green"),
       pch    = c(19, 17, 15)
```

```
)
```



Proximity measure between pairs of observations (Reduced to two dimentions using multidimensional scaling).