IST687 - Music Classification Project

Team 2 - Sebastian Castro, John Fields, Courtney Smith, Jeremy Wallner

5/13/2019

Executive Summary

The purpose of this project is to analyze the Million Song Database to predict "Hot" artists and songs based on the attributes such as familiarity, artist location, loudness, terms used, etc. The analysis was done using R software on a 10,000 track subset of the data and our model was able to predict "Hot" songs with ~80% accuracy.

Table of Contents

Introduction

Related Work

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

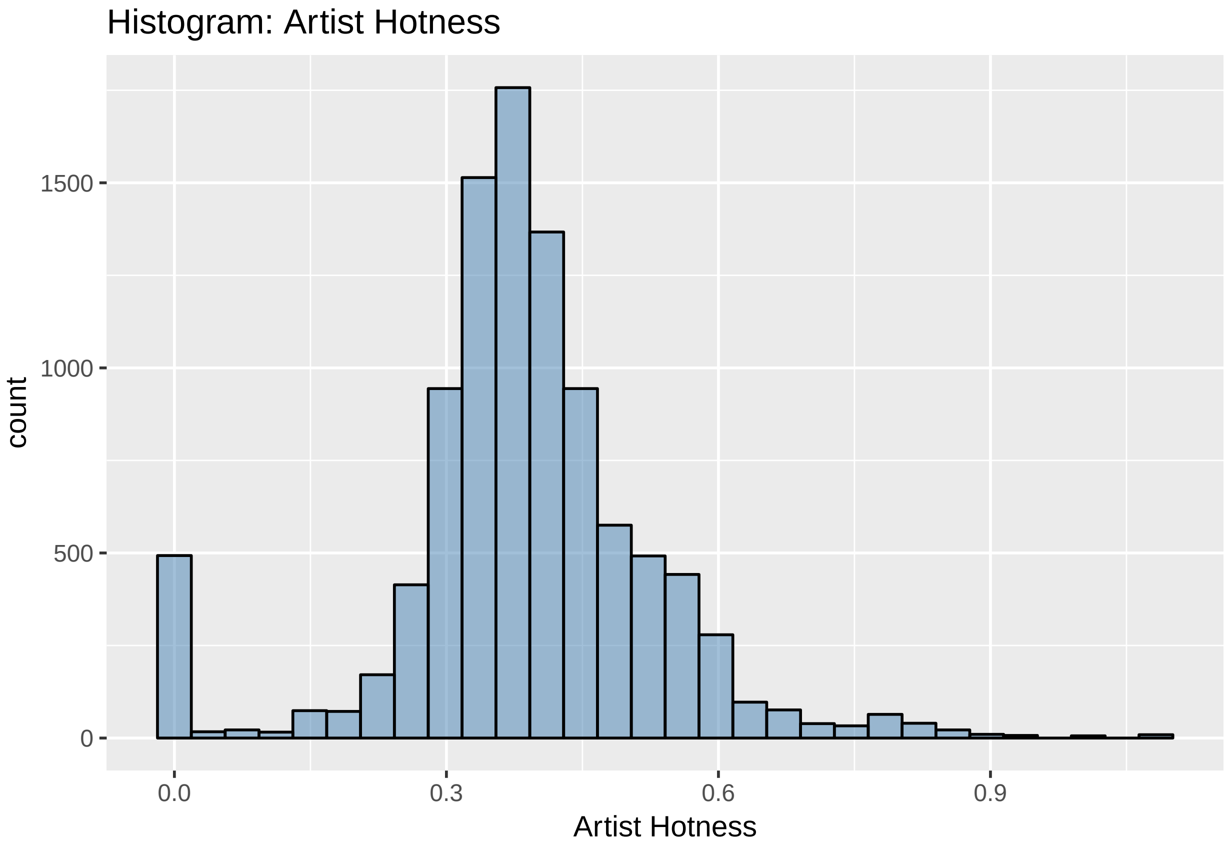
Dataset

```r
#New code from Courtney to change from 3 to 5 categories of artist hotness
music <- read.csv("/Users/johnfields/Library/Mobile Documents/com~apple~CloudDocs/Syracuse/IST687/Proje
str(music)
```

```
## 'data.frame':    9996 obs. of  36 variables:
##  $ artist.hotttnesss       : num  0.402 0.417 0.343 0.454 0.402 ...
##  $ artist.id               : Factor w/ 3885 levels "AR009211187B989185",..: 1269 2353 2168 715 3606
##  $ artist.name             : Factor w/ 4409 levels ":Blacks On :Blondes",..: 682 3796 3560 67 1569
##  $ artist_mbtags           : Factor w/ 277 levels "","0.333","60s",..: 1 52 1 262 1 1 1 1 1 1 ...
##  $ artist_mbtags_count     : num  0 1 0 1 0 0 0 0 0 0 ...
##  $ bars_confidence         : num  0.643 0.007 0.98 0.017 0.175 0.121 0.709 0.142 0.806 0.047 ...
##  $ bars_start              : num  0.585 0.711 0.732 1.306 1.064 ...
##  $ beats_confidence        : num  0.834 1 0.98 0.809 0.883 0.438 0.709 0.234 0.44 1 ...
##  $ beats_start             : num  0.585 0.206 0.732 0.81 0.136 ...
##  $ duration                : num  219 148 177 233 210 ...
##  $ end_of_fade_in          : num  0.247 0.148 0.282 0 0.066 ...
##  $ familiarity             : num  0.582 0.631 0.487 0.63 0.651 ...
##  $ key                     : num  1 6 8 0 2 5 1 4 4 7 ...
##  $ key_confidence          : num  0.736 0.169 0.643 0.751 0.092 0.635 0 0 0.717 0.053 ...
##  $ latitude                : num  37.2 35.1 37.2 37.2 37.2 ...
##  $ location                : Factor w/ 1046 levels " "," NC"," UbA!, Minas Gerais",..: 157 584 705
##  $ longitude               : num  -63.9 -90 -63.9 -63.9 -63.9 ...
##  $ loudness                : num  -11.2 -9.84 -9.69 -9.01 -4.5 ...
##  $ mode                     : int  0 0 1 1 1 1 1 0 1 0 ...
##  $ mode_confidence          : num  0.636 0.43 0.565 0.749 0.371 0.557 0 0.16 0.652 0.473 ...
##  $ release.id               : int  300848 300822 514953 287650 611336 41838 25824 8876 358182 692313
##  $ release.name             : Factor w/ 7830 levels " Lazy Afternoon En Anglais",..: 2191 1746 3535
##  $ similar                  : Factor w/ 2837 levels "AR00K8N11C8A41687B",..: 2408 2225 1145 304 2331
##  $ song.hotttnesss          : num  0.602 NA NA NA 0.605 ...
##  $ song.id                  : Factor w/ 9996 levels " Polovtsian Dances / Rimsky-Korsakov: Russian E
##  $ start_of_fade_out        : num  219 138 172 217 199 ...
##  $ tatums_confidence        : num  0.779 0.969 0.482 0.601 1 0.136 0.467 0.292 0.121 1 ...
##  $ tatums_start             : num  0.285 0.206 0.421 0.563 0.136 ...
##  $ tempo                    : num  92.2 121.3 100.1 119.3 129.7 ...
##  $ terms                    : Factor w/ 459 levels "","8-bit","acid jazz",..: 216 34 372 327 325 396
##  $ terms_freq               : num  1 1 1 0.989 0.887 ...
##  $ time_signature           : num  4 4 1 4 4 3 1 3 4 4 ...
##  $ time_signature_confidence: num  0.778 0.384 0 0 0.562 0.454 0 0.408 0.487 0.878 ...
##  $ title                    : Factor w/ 9705 levels ""," -start ID-",..: 3572 7526 481 7474 2531 828
##  $ year                     : int  0 1969 0 1982 2007 0 0 0 1984 0 ...
##  $ artist.hotttnesss.label  : Factor w/ 3 levels "Cold","Hot","Warm": 3 3 3 3 3 3 1 2 1 3 ...
```

```
##Artist Hotness Histogram
library(ggplot2)
ggplot(music, aes(x=artist.hotttnesss))  + geom_histogram(color="black", fill="steelblue", alpha=0.5) +
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
##Function to create descriptive statistics for artist hotness
descriptive_stats <- function(vector) { library(moments)
  result <- c(Mean=mean(vector),
              Median=median(vector),
              Min = min(vector),
              Max = max(vector),
              SD = sd(vector),
              Quantile = quantile(vector, probs = c(0.25,.50,0.75, 0.95)),
              Skewness = skewness(vector) )
  print(result)
}
descriptive_stats(music$artist.hotttnesss)
```

```
##         Mean       Median          Min          Max           SD
##    0.3857065    0.3807564    0.0000000    1.0825026    0.1434688
## Quantile.25% Quantile.50% Quantile.75% Quantile.95%     Skewness
##    0.3255062    0.3807564    0.4539300    0.6011861   -0.1483509
```

```
##Methodology for assigning artist hotness levels - uses quantiles from descriptitive_statistics functi
#95% Quantile: 0.6011861 - Hot
#75% Quantile: 0.453858  - Warm
#50% Quantile: 0.3807423 - Tepid
#25% Quantile: 0.3252656 - Cool
##Code for assigning labels based on above quantiles
music$artist.hotness.label <- ifelse(music$artist.hotttnesss >=0.6011861, "Hot",
                                     ifelse(music$artist.hotttnesss >=0.453858 & music$artist.hotttness
                                            ifelse(music$artist.hotttnesss >=0.3807423 & music$artist.h
                                                   ifelse(music$artist.hotttnesss >=0.3252656 & music$a
                                                          ifelse(music$artist.hotttnesss < 0.3252656, "
unique(music$artist.hotness.label)
```

```
## [1] "Tepid"  "Cool"   "Warm"   "Frigid" "Hot"
```

```
#End of new code from Courtney
#Prior to importing, a new column artist.hotttnesss.label was adding with
#Hot(>.4590), Warm(<.4590 and >.3357), Cold(<.3357).  Four rows with blanks in
#famiiarity were also deleted.
music <- na.omit(music)
#Copy original data to a new dataframe music1 and exclude unneeded data
music <- music[-c(1:5,7,16,19,21:25,30,34)]
music$artist.hotness.label <- as.factor(music$artist.hotness.label)
str(music)
```

```
## 'data.frame':    5648 obs. of  22 variables:
##  $ bars_confidence          : num  0.643 0.175 0.806 0.873 0.018 0.013 1 0.507 0.125 0.03 ...
##  $ beats_confidence         : num  0.834 0.883 0.44 0.873 1 0.699 1 0 0.768 1 ...
##  $ beats_start              : num  0.585 0.136 1.226 0.112 0.429 ...
##  $ duration                 : num  219 210 270 219 245 ...
##  $ end_of_fade_in           : num  0.247 0.066 5.3 2.125 0.357 ...
##  $ familiarity              : num  0.582 0.651 0.427 0.36 0.545 ...
##  $ key                      : num  1 2 4 5 7 9 10 7 8 7 ...
##  $ key_confidence           : num  0.736 0.092 0.717 0.354 0.07 0.205 0 1 0.041 0.725 ...
##  $ latitude                 : num  37.2 37.2 37.2 35.2 37.2 ...
##  $ longitude                : num  -63.9 -63.9 -63.9 -80 -63.9 ...
##  $ loudness                 : num  -11.2 -4.5 -13.5 -10.02 -7.54 ...
##  $ mode_confidence          : num  0.636 0.371 0.652 0.485 0.686 0.305 0.198 0.829 0.516 0.756 ...
##  $ start_of_fade_out        : num  219 199 259 207 227 ...
##  $ tatums_confidence        : num  0.779 1 0.121 0.229 0.728 1 0.774 0.377 0.767 0.238 ...
##  $ tatums_start             : num  0.285 0.136 1.226 0.112 0.173 ...
##  $ tempo                    : num  92.2 129.7 86.6 146.8 118 ...
##  $ terms_freq               : num  1 0.887 0.96 0.956 1 ...
##  $ time_signature           : num  4 4 4 1 4 4 1 4 5 4 ...
##  $ time_signature_confidence: num  0.778 0.562 0.487 0 0.835 0 0.319 0.756 0.579 0.931 ...
##  $ year                     : int  0 2007 1984 0 0 0 0 1987 0 2004 ...
##  $ artist.hotttnesss.label  : Factor w/ 3 levels "Cold","Hot","Warm": 3 3 1 1 3 3 1 3 1 2 ...
##  $ artist.hotness.label     : Factor w/ 5 levels "Cool","Frigid",..: 4 4 1 2 1 1 2 4 1 5 ...
```

Features

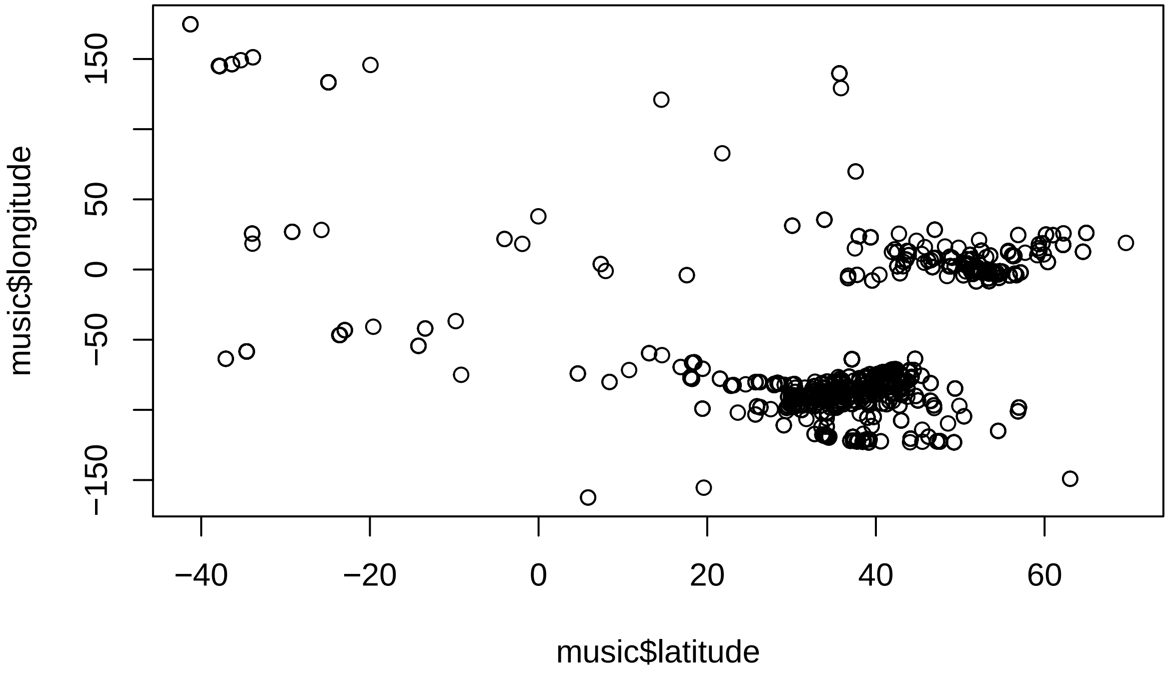
```
#View the number of Cold/Warm/Hot labels
table(music$artist.hotttnesss.label)
```

```
##
## Cold  Hot Warm
## 1180 1579 2889
```

```
#View the number of Frigid/Cool/Tepid/Warm/Hot labels
table(music$artist.hotness.label)
```

```
##
##   Cool Frigid    Hot  Tepid   Warm
##   1444    973    278   1566   1387
```

```
#Plot artists latitude and longitude
plot(music$latitude,music$longitude)
```



```
#Plot artist hotttnesss
#hist(music$artist.hotttnesss,breaks=20)
#hist(music$artist.hotness,breaks=20)
```

#THIS IS INCOMPLETE CODE FOR PLOTTING ADDITIONAL DATA… #Create a map of the world
mapWorld <- borders(“world”, colour=“gray50”, fill=“white”)

#Code from https://rpubs.com/spoonerf/global_map #Need to figure out what to put in locs locs<-read.csv("my_locations.csv") locs<- sp_dups<-data.frame(ddply(locs,.(Longitude,Latitude),nrow)) sp_dups$loc_id < -1 : length(sp_dups$Longitude) sp_dups_df<-merge(sp_dups, locs, by=c("Longitude","Latitude"))

loc<-data.frame(sp_dups_df$Longitude, sp_dups_df$Latitude,sp_dups_df$V1) loc<-unique(loc) colnames(loc)<-c("Longitude", "Latitude", "V1")

coordinates(loc)<-c("Longitude","Latitude") proj4string(loc) <- CRS("+proj=longlat")

loc_df<-data.frame(loc)

theme_opts <- list(theme(panel.grid.minor = element_blank(), panel.grid.major = element_blank(), panel.background = element_blank(), plot.background = element_rect(fill="white"), panel.border = element_blank(), axis.line = element_blank(), axis.text.x = element_blank(), axis.text.y = element_blank(), axis.ticks = element_blank(), axis.title.x = element_blank(), axis.title.y = element_blank(), plot.title = element_text(size=22)))

library(maps) library(mapdata)

ggplot(data=loc_df, aes(Longitude, Latitude, group=NULL,fill=NULL,size=V1))+#, fill=hole)) + borders(fill="light grey",colour="light grey")+ geom_point(color="black",alpha=I(7/10))+ scale_size(range=c(1,7), guide = "legend",labs(size="No. of Populations"))+ coord_equal()+ theme_opts

Methods

```
#Do analysis to determine hot/warm/cold artists based on hotttnesss
#The ramdom forest analysis is from a training video by Bharatendra Rai
#at https://www.youtube.com/watch?v=dJclNIN-TPo
#Data Partition - ind = independent samples
#The code below runs in console but not R Markdown
set.seed(123)
ind<- sample(2,nrow(music), replace=TRUE,prob=c(0.7,0.3))
train <- music[ind==1,]
test <- music[ind==2,]
#Run randomForest on 3 levels
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(222)
rf <- randomForest(music[,-21:-22],music[,21])
print(rf)
```

```
##
## Call:
##  randomForest(x = music[, -21:-22], y = music[, 21])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
```

```
##
##         OOB estimate of  error rate: 20.18%
## Confusion matrix:
##      Cold  Hot Warm class.error
## Cold  728    4  448   0.3830508
## Hot     6 1289  284   0.1836605
## Warm  200  198 2491   0.1377639
```

```r
attributes(rf)
```

```
## $names
##  [1] "call"            "type"            "predicted"
##  [4] "err.rate"        "confusion"       "votes"
##  [7] "oob.times"       "classes"         "importance"
## [10] "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"
## [16] "y"               "test"            "inbag"
##
## $class
## [1] "randomForest"
```

```r
rf$confusion
```

```
##      Cold  Hot Warm class.error
## Cold  728    4  448   0.3830508
## Hot     6 1289  284   0.1836605
## Warm  200  198 2491   0.1377639
```

```r
#Run randomForest on 5 levels
library(randomForest)
set.seed(222)
rf2 <- randomForest(music[,-21:-22],music[,22])
print(rf2)
```

```
##
## Call:
##  randomForest(x = music[, -21:-22], y = music[, 22])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 35.64%
## Confusion matrix:
##        Cool Frigid Hot Tepid Warm class.error
## Cool    845    175   0   387   37   0.4148199
## Frigid  279    620   0    73    1   0.3627955
## Hot       1      2  56    15  204   0.7985612
## Tepid   339     45   0   982  200   0.3729246
## Warm     21      6  10   218 1132   0.1838500
```

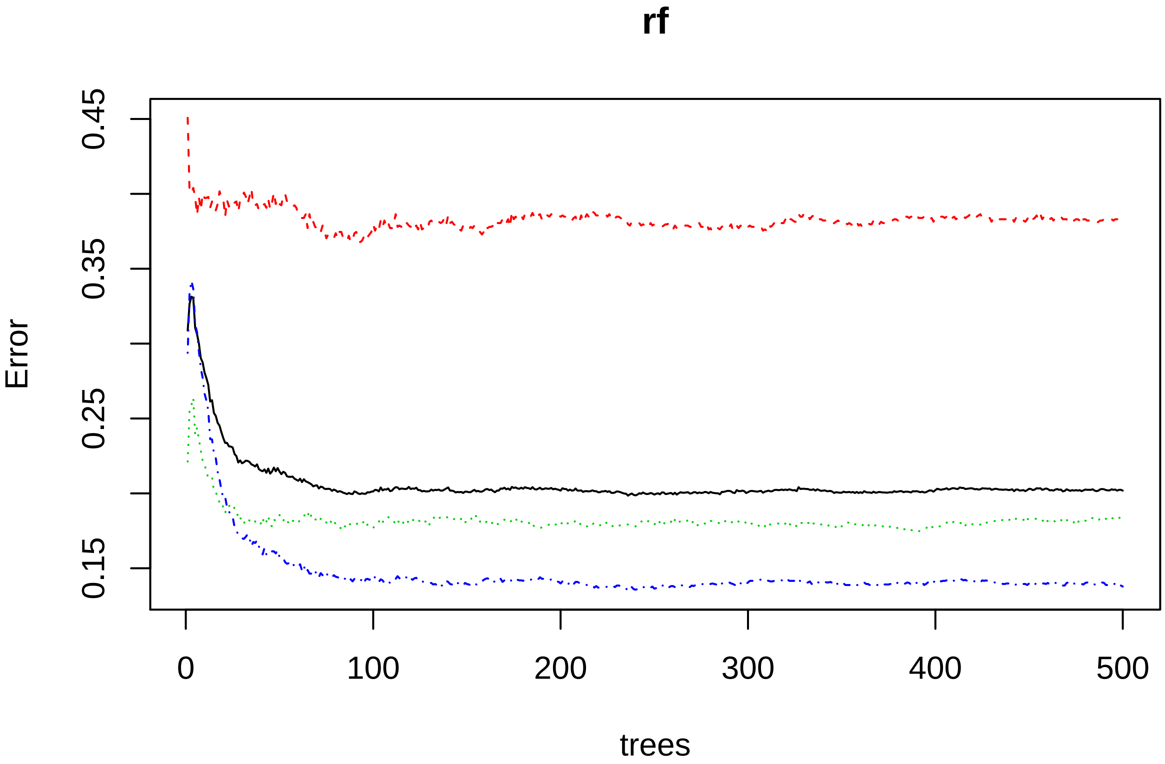
```r
attributes(rf2)
```

```
## $names
##  [1] "call"            "type"            "predicted"
##  [4] "err.rate"        "confusion"       "votes"
##  [7] "oob.times"       "classes"         "importance"
## [10] "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"
## [16] "y"               "test"            "inbag"
##
## $class
## [1] "randomForest"
```

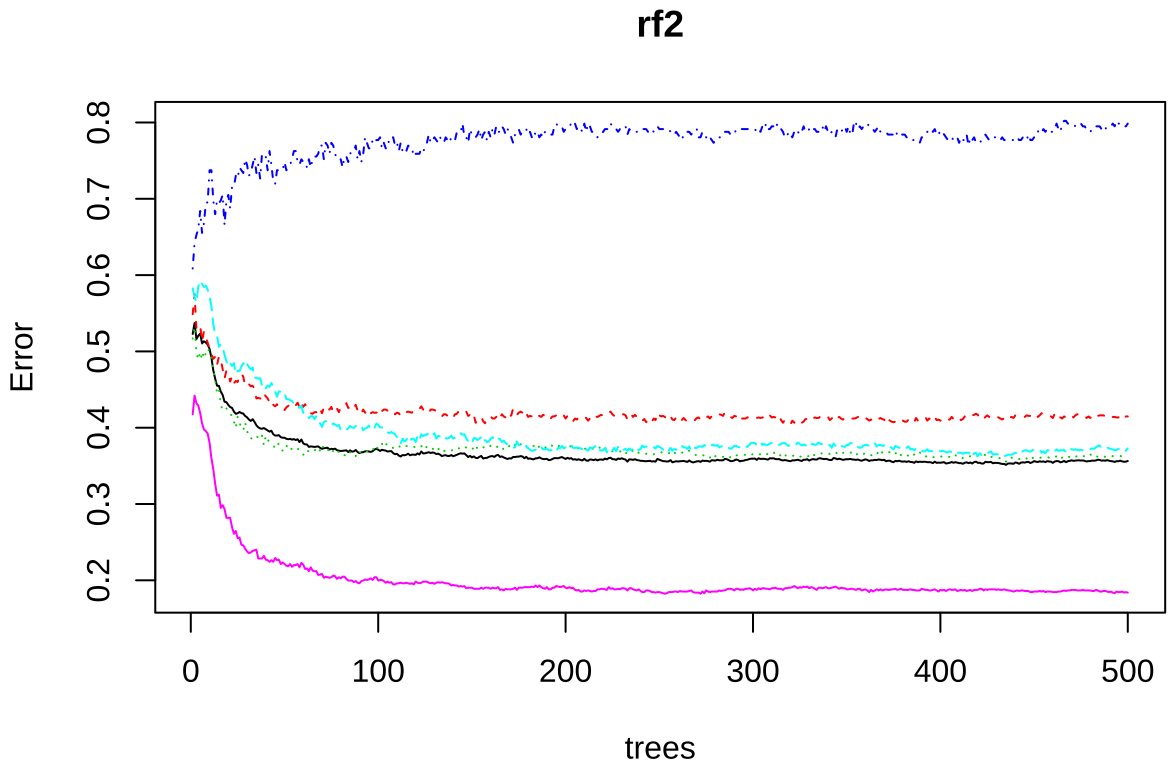
```
rf2$confusion
```

```
##        Cool Frigid Hot Tepid Warm class.error
## Cool    845    175   0   387   37   0.4148199
## Frigid  279    620   0    73    1   0.3627955
## Hot       1      2  56    15  204   0.7985612
## Tepid   339     45   0   982  200   0.3729246
## Warm     21      6  10   218 1132   0.1838500
```

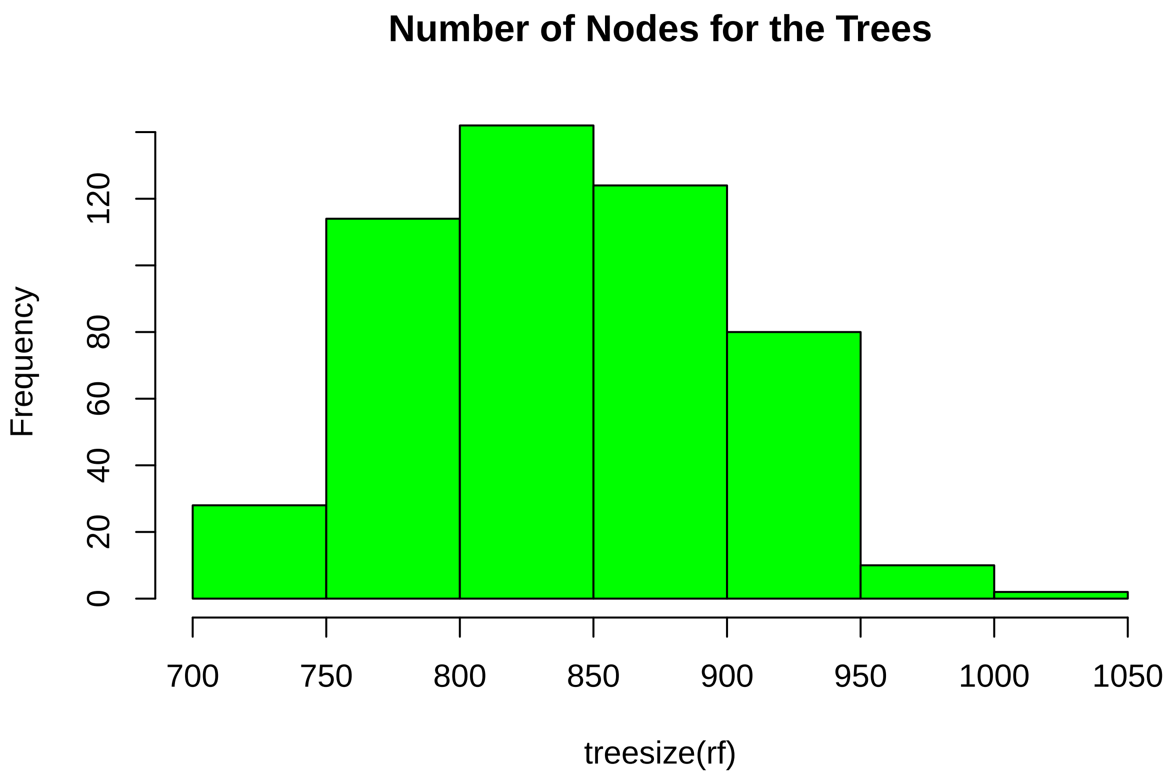
```
#Run randomForest again with tune mtry data from below
#rfx <- randomForest(artist.hotness.label ~.,data=music,ntree=200,mtry=8,importance=TRUE,proximity=TRUE
#Prediction & Confusion Matrix - train data
#library(caret)
#p1<-predict(rfx,train)
#confusionMatrix(p1,train)
#Predition & Confusion Matrix - test data
#p2<-predict(rfx,test)
#confusionMatrix(p2,test$artist.hotness.label)
#Error rate of Random Forest
plot(rf)
```



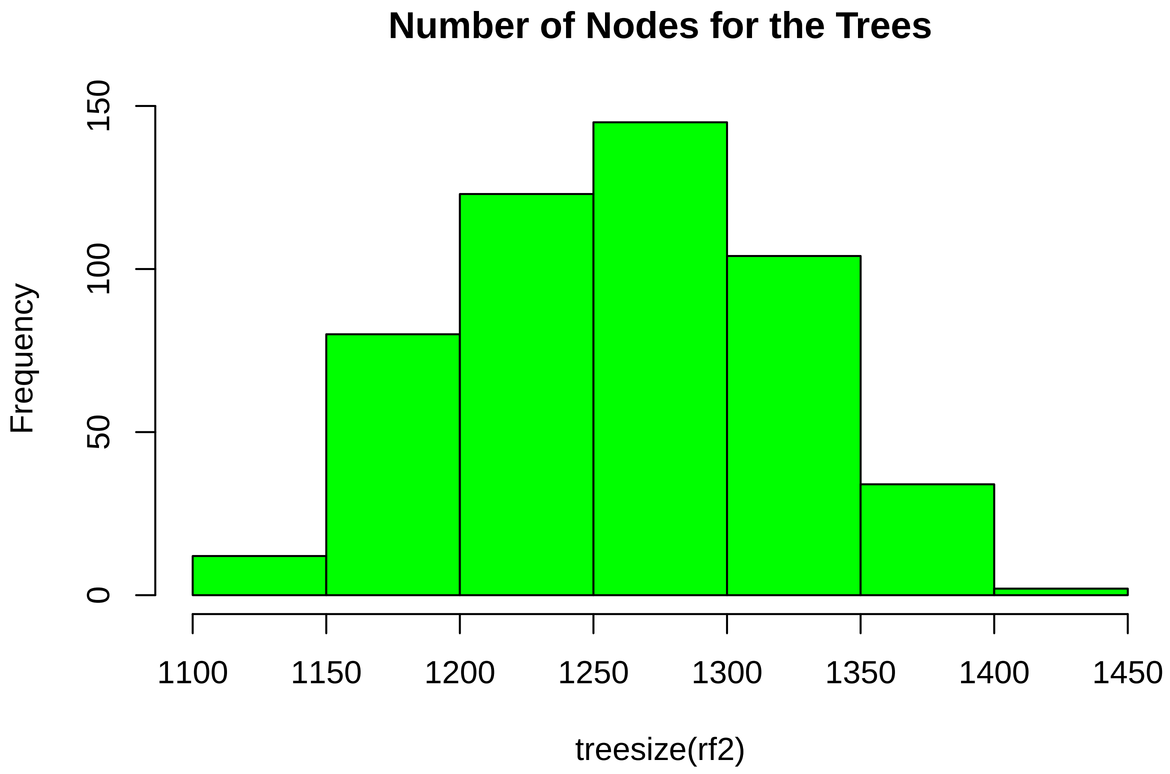
```
plot(rf2)
```



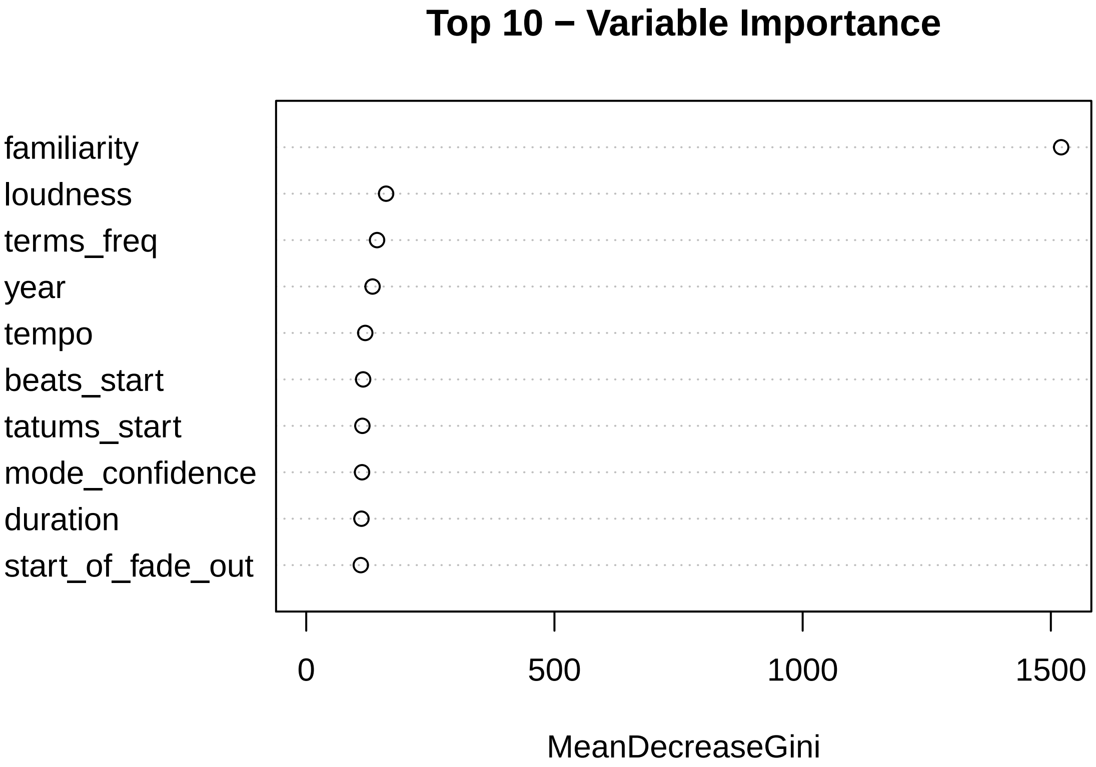
```
#The error rate is not improving after ~100 trees
#Tune mtry
#t <- tuneRF(train[,-21],train[,21],
#            stepFactor=.5,
#             plot=TRUE,
#             ntreeTry=200,
#             trace=TRUE,
#             improve=0.05)
#No. of nodes for the trees
hist(treesize(rf),
    main="Number of Nodes for the Trees",
    col="green")
```



```r
hist(treesize(rf2),
    main="Number of Nodes for the Trees",
    col="green")
```



```r
# Variable Importance
# Familiarity is much more important than the other variables.  Should it be removed and run again?
varImpPlot(rf,
           sort=T,
           n.var=10,
           main="Top 10 - Variable Importance")
```



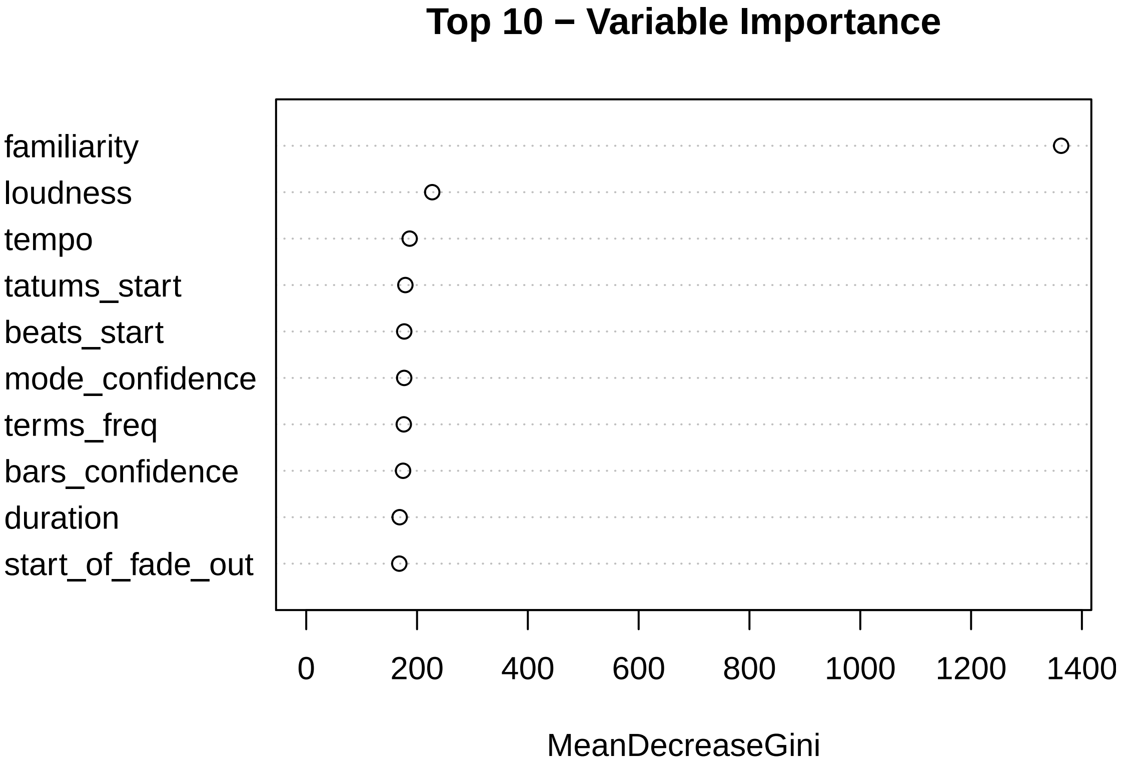
```r
importance(rf)
```

```
##                           MeanDecreaseGini
## bars_confidence                  107.58378
## beats_confidence                  94.47263
## beats_start                      114.60290
## duration                         111.20674
## end_of_fade_in                    88.18414
## familiarity                     1520.73102
## key                               70.97697
## key_confidence                   107.84287
## latitude                          80.31500
## longitude                         80.00661
## loudness                         160.85558
## mode_confidence                  112.40895
## start_of_fade_out                109.94686
## tatums_confidence                100.29904
## tatums_start                     113.12138
## tempo                            118.86929
## terms_freq                       142.67373
## time_signature                    31.82993
## time_signature_confidence         81.87304
## year                             133.56141
```

```r
varUsed(rf)
```

```
##  [1] 22981 20248 24173 23234 18362 42545 16709 23076 13678 13543 26260
## [12] 23733 23098 21580 23800 24889 17934  7870 18132 13527
```

```r
varImpPlot(rf2,
           sort=T,
           n.var=10,
           main="Top 10 - Variable Importance")
```



```r
importance(rf2)
```

```
##                           MeanDecreaseGini
## bars_confidence                  174.77925
## beats_confidence                 153.37570
## beats_start                      176.89591
## duration                         168.58389
## end_of_fade_in                   141.16350
## familiarity                     1362.55235
## key                              115.59861
## key_confidence                   167.04577
## latitude                         114.97836
## longitude                        118.58999
## loudness                         227.38098
## mode_confidence                  176.77603
## start_of_fade_out                167.99820
## tatums_confidence                160.88208
## tatums_start                     178.93330
## tempo                            186.65606
## terms_freq                       176.06618
## time_signature                    51.64041
## time_signature_confidence        134.38659
## year                             167.90690
```

```
varUsed(rf2)
```

```
##  [1] 35548 31313 36142 34492 28373 59265 26210 34634 19869 20060 39304
## [12] 36051 34736 33307 36125 37324 26336 12053 28255 20695
```

```
#Mulit-dimenstional Scaling Plot
#The code below causes R to lock up...
#MDSplot(rf,train$artist.hotttnesss.label)
```

Results

Conclusion

Appendices