Main

Wyatt Thompson, Saaya Yasuda

December 5, 2017

Figure 1:

So where ya'll from?

Every person who speaks, or has ever spoken, has an accent, a lingering effect from a linguistic history.

But how different are accents, really? With a unique dataset, recordings of 2132 people reading the same text:

"Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station."

First we have to convert these audio files to quantities we can work with, and then we will explore the relationship between one's gender, age, native language, and the various measurable quantities of the audio clip. After that, we'll build methods to classify speakers.

Come on, even the computer knows you have an accent!

You can find our dataset here: https://www.kaggle.com/rtatman/speech-accent-archive

First Words

Although people can hear the differences between accents, a statistical model is deaf without a programmer's help. So first things first, converting the .mp3 files into measurable features.

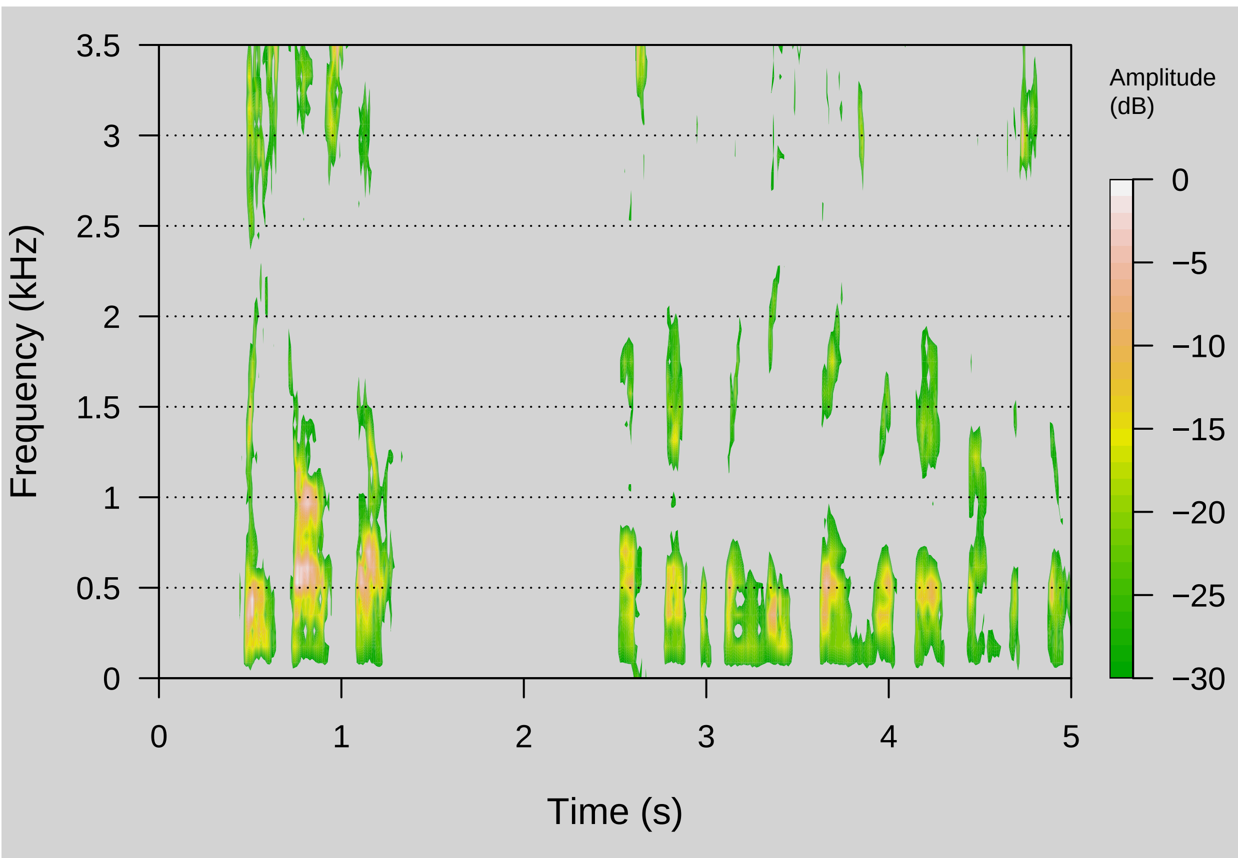
Mp3s are great for our phones and computers because they save a lot of space. Where a raw, .Wav file may take up 50 megabytes, the compressed .Mp3 may be 5mb. However, this compression makes the files more difficult to work with and extract meaningful information. So we need to turn our 2132 mp3s into .wav files.

Luckily, the package tuneR in R has a method to convert all of the files to the format we can work with. Then we're ready to start looking into the data.

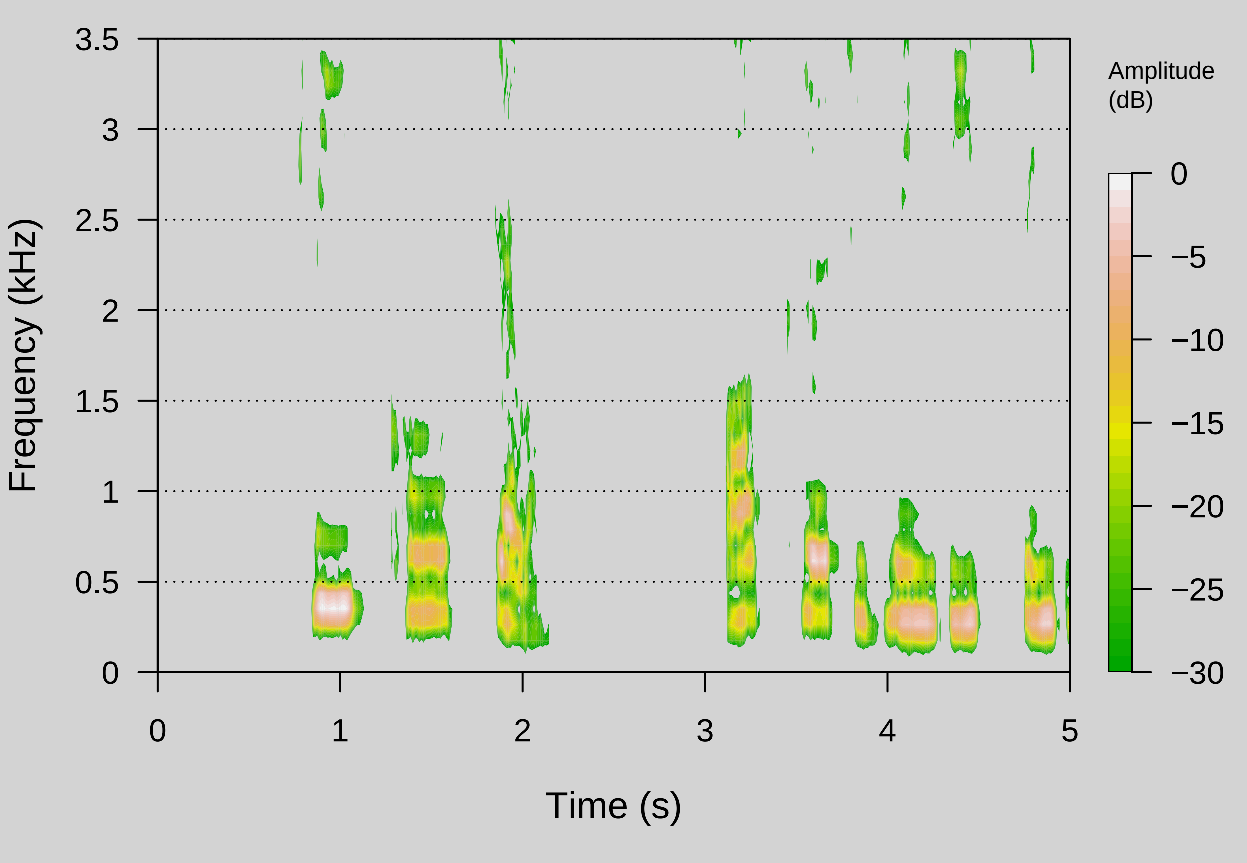
Frequency

For one, we can look at the change in frequency, which we hear as pitch, over time. It's these changes in frequency and amplitude that allow us to enunciate.

```
ex_wave<-readWave("../data/english46.wav")
spectro(ex_wave,f=44100,flim=c(0,3.5),tlim=c(0,5),colbg = "lightgray",palette = terrain.colors)
```

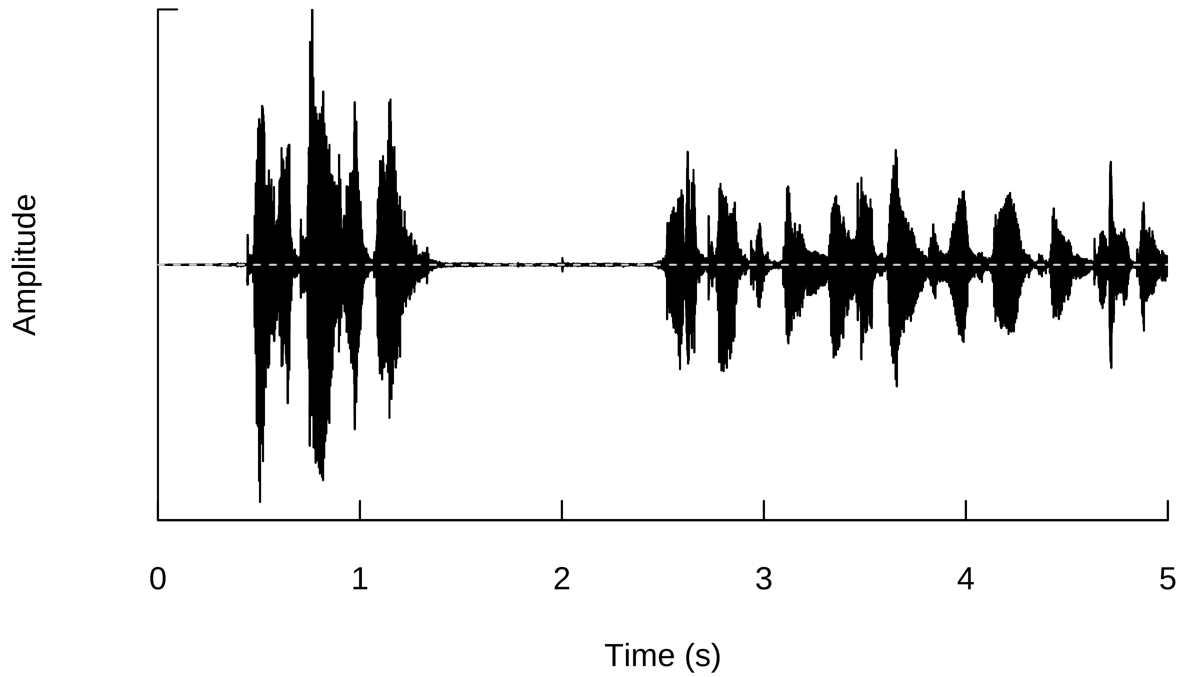



```
ex_wave2<-readWave("../data/dutch30.wav")
spectro(ex_wave2,f=44100,flim=c(0,3.5),tlim=c(0,5),colbg = "lightgray",palette = terrain.colors)
```

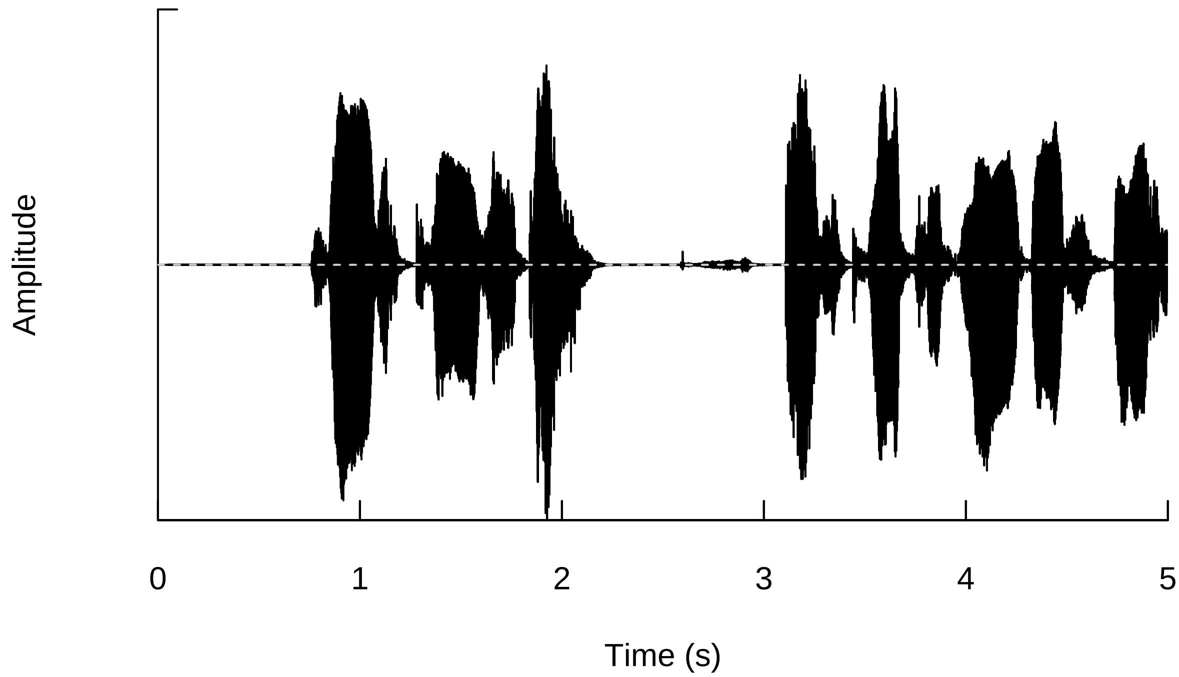


Furthermore, we can look at the change in amplitude (loudness) over time. This shows us a visual understanding of the space between words or syllables and the range of volume a speaker uses.

```
osc<-oscillo(ex_wave,from=0,to=5)
```

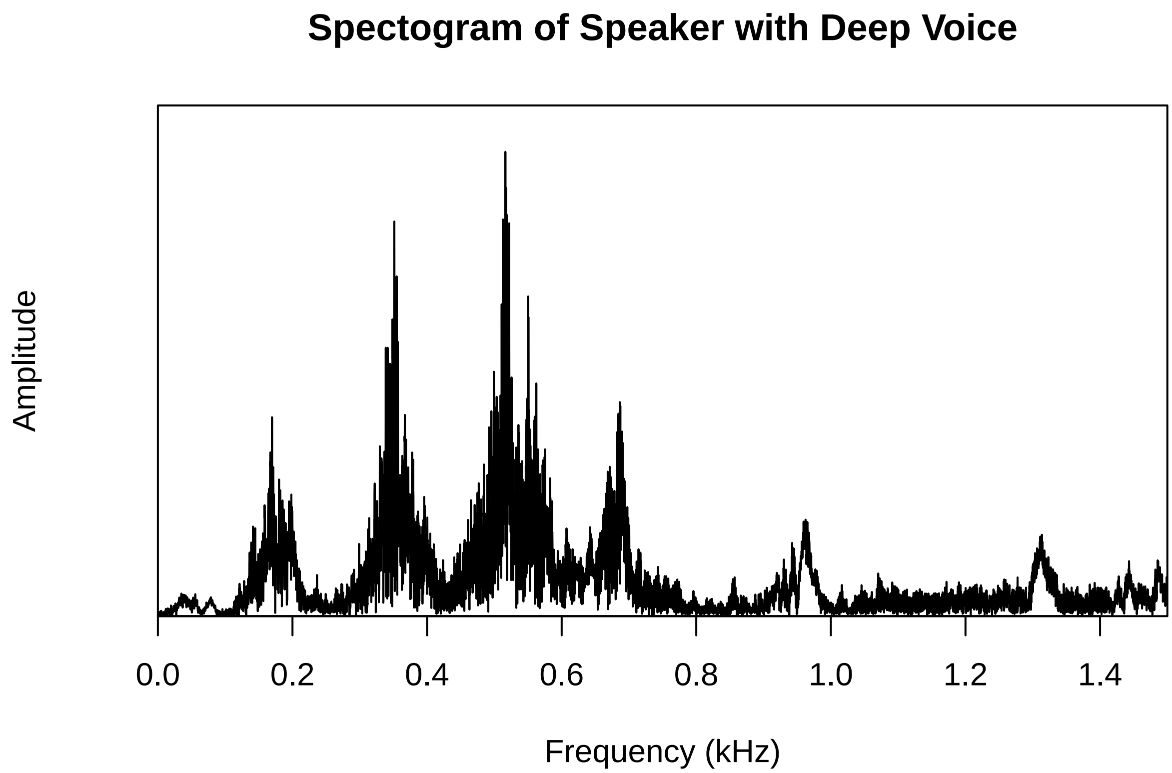



```
osc2<-oscillo(ex_wave2,from=0,to=5)
```



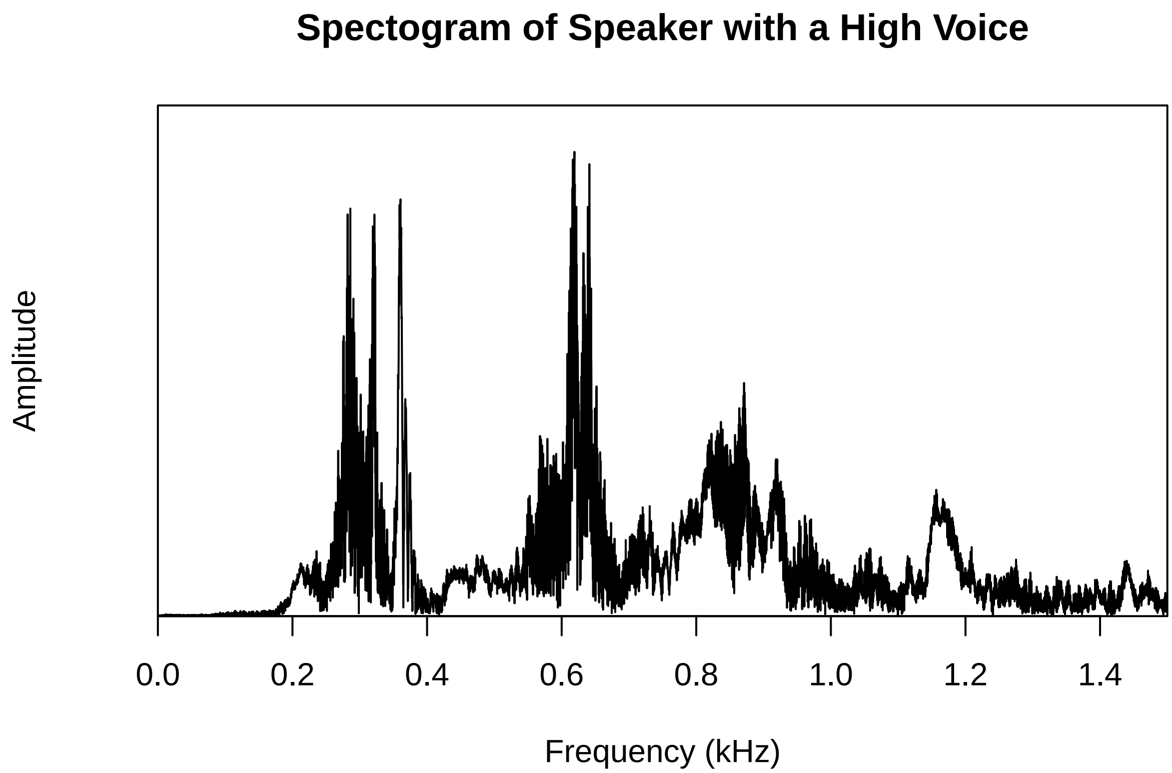
The spectographic image is even more insightful. We can get an idea of the tone of a speakers voice. In these two plots, we see the spectograms of a low voice and a high voice.

```
ex_spec<-spec(ex_wave,main="Spectogram of Speaker with Deep Voice",from=0,to=5,flim=c(0,1.5))
```



Note the bump around

```
ex_spec2<-spec(ex_wave2,main="Spectogram of Speaker with a High Voice",from=0,to=5,flim=c(0,1.5))
```



Note the way the high voice shows increased amplitudes at higher frequencies. It's differences like these that will allow us to classify accents!

Turning it into Data

While plots are great for visualization, they do little to help model differences in the audio clips quantitatively. To do that, we extract summary statistics from the audio files. We extract:

1. mean frequency (in kHz) 2.standard deviation of frequency 3.median frequency (in kHz) 4.standard error of frequency 5.mode of the frequency 6.first quantile 7.third quantile 8.interquantile range 9.centroid 10.skewness 11.kurtosis 12.spectral flatness 13.spectral entropy 14.Precision of frequency 15.Mean Fundamental Frequency (Most prominent tone) 16.Min Fundamental Frequency 17.Max Fundamental Frequency 18.Mean Fundamental Frequency 19.Differential Range 20.Modulation Index (measure of pace)

(Note if you are interested in the extraction process, check the lib folder for FeatureExtraction2.R. The process is computational and tedious so we omit it here)

We then use these observed features to classify Age, Sex, and Country.

```
data<-read.csv("../output/all_features.csv")
data<-data[(data$age>0),]
data$sex[data$sex=="famale"]<-"female"
data$sex = factor(data$sex,levels = c("female","male"))

head(data[1:6])
```

```
##         file     mean        sd   median        sem      mode
```

```
## 1 afrikaans1 208.6925 37.43829 212.9307 0.4903644 211.3933
## 2 afrikaans2 193.5731 64.80026 215.5873 0.8252968 223.9455
## 3 afrikaans3 208.6856 59.78416 232.6525 0.6884488 267.4111
## 4 afrikaans4 175.3418 69.73779 149.1152 0.8590639 129.1425
## 5 afrikaans5 169.5784 65.21669 146.2091 0.8651073 120.4365
## 6      agni1 187.8257 64.14447 168.6661 0.6467372 131.5738
```

```r
head(data[7:12])
```

```
##        Q25      Q75       IQR     cent skewness  kurtosis
## 1 196.1153 228.5450  32.42965 208.6925 2.457101  9.535111
## 2 136.7748 244.1142 107.33939 193.5731 2.074303  8.503729
## 3 142.7480 260.6525 117.90451 208.6856 2.175647  8.690548
## 4 125.1055 245.7490 120.64350 175.3418 4.070846 25.894280
## 5 120.3872 236.0436 115.65646 169.5784 1.668817  5.656690
## 6 133.7942 252.6434 118.84913 187.8257 1.959983  7.072717
```

```r
head(data[13:18])
```

```
##         sfm        sh       prec   meanfun     minfun    maxfun
## 1 0.2591110 0.8828894 0.04803168 0.1565614 0.04344828 0.2791139
## 2 0.5179635 0.9419297 0.04541074 0.1120971 0.04315068 0.2791139
## 3 0.3217430 0.9192578 0.03713021 0.1194145 0.04315068 0.2791139
## 4 0.4511409 0.9133502 0.04248751 0.1148991 0.04310850 0.2791139
## 5 0.5383520 0.9396168 0.04926802 0.1210706 0.04310850 0.2791139
## 6 0.3845247 0.9259169 0.02846190 0.1165454 0.04310850 0.2791139
```

```r
head(data[19:23])
```

```
##     meandom mindom    maxdom   dfrange    modindx
## 1 0.6658538      0 10.917334 10.917334 0.06063079
## 2 0.9092162      0 21.210205 21.210205 0.03583999
## 3 0.8221106      0 11.025000 11.025000 0.04869588
## 4 0.5365841      0 13.953516 13.953516 0.03583355
## 5 1.1564153      0 20.004346 20.004346 0.05207502
## 6 0.3973365      0  5.921631  5.921631 0.06611488
```

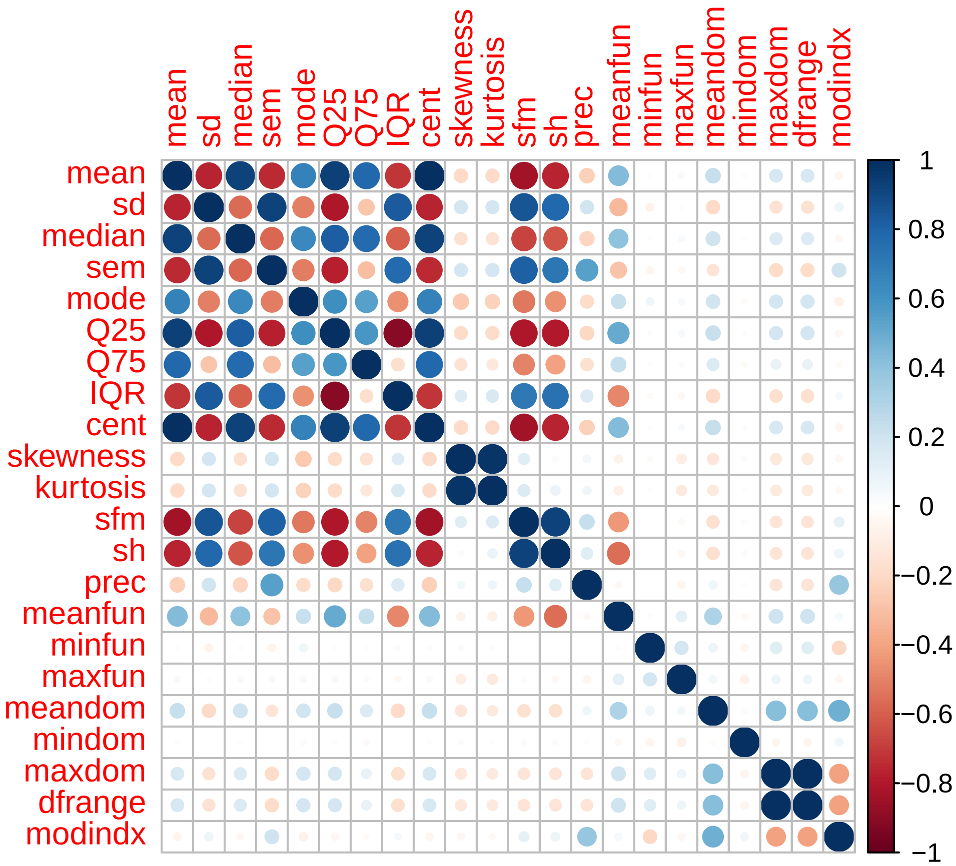
```r
head(data[24:30])
```

```
##   age age_onset                        birthplace native_language    sex
## 1  27         9            virginia, south africa       afrikaans female
## 2  40         5            pretoria, south africa       afrikaans   male
## 3  43         4 pretoria, transvaal, south africa       afrikaans   male
## 4  26         8            pretoria, south africa       afrikaans   male
## 5  19         6           cape town, south africa       afrikaans   male
## 6  25        15              diekabo, ivory coast            agni   male
##   speakerid      country
## 1         1 south africa
## 2         2 south africa
## 3       418 south africa
## 4      1159 south africa
## 5      1432 south africa
## 6         3  ivory coast
```

Exploratory Analysis

Before we start building our classifier, let's check out what's going on between our variables.

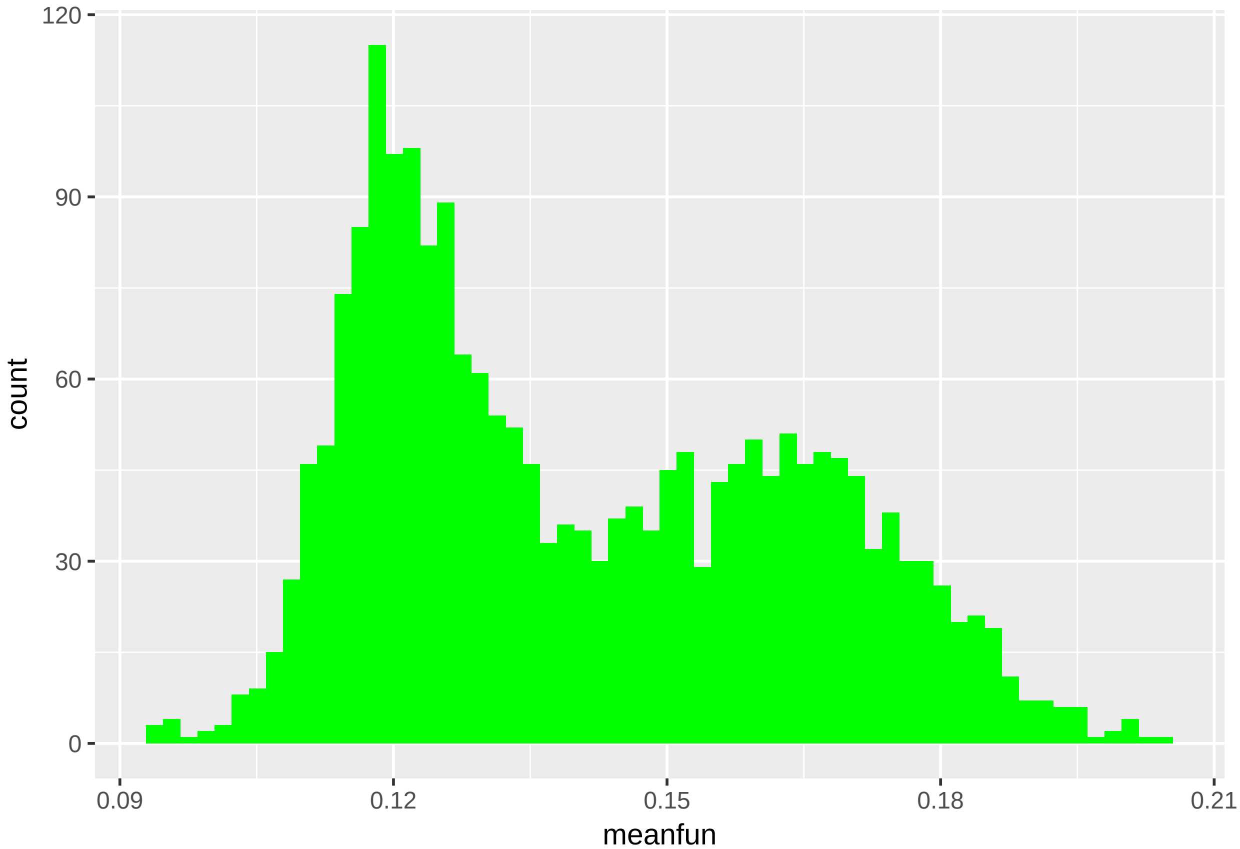
```r
c<-cor(data[,2:23])
corrplot(c)
```



Well, we see that many of the frequency summary statistics contain similar information. This is expected, but it's promising to see such low correlation between fundamental frequencies and the frequency summary statistics.

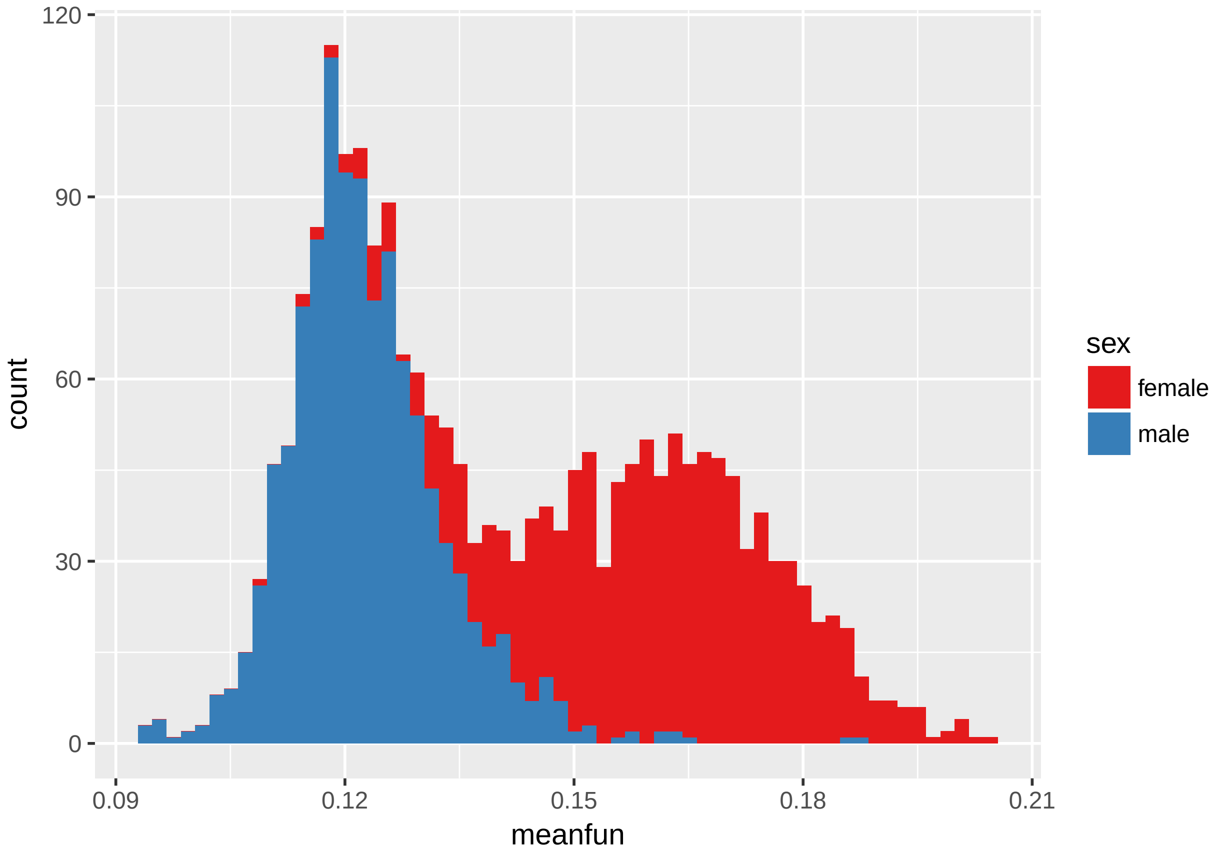
Let's take a closer look at the mean fundamental frequency. The fundamental frequency is defined as the lowest frequency observed in a waveform, so it should give us a great idea of the tone of voice.

```r
ggplot(data)+geom_histogram(aes(meanfun),bins=60,fill="green")
```



Note the two peaks in fundamental frequency. It appears there's a significantly different fundamental frequency for two groups in our population.

```
ggplot(data)+geom_histogram(aes(meanfun,fill=sex),bins=60)+scale_fill_brewer(palette="Set1")
```




```
genderdf<-data[,c(2:23,28)]
genderdf$sex<-as.factor(genderdf$sex)
train<-round(.75*nrow(genderdf))
train.ind<-sample(1:nrow(genderdf),train)
traindata<-genderdf[train.ind,]
testdata<-genderdf[-train.ind,]


genderSVM<-svm(sex~.,data=traindata,gamma=.02,cost=2)

#TrainTest
predictSvm <- predict(genderSVM, testdata)
table(predictSvm, testdata$sex)
```

```
##
## predictSvm female male
##     female    234   15
##     male       13  271
```

```
sum(predictSvm ==testdata$sex)/length(testdata$sex)
```

```
## [1] 0.9474672
```

```
#SVM.tune<-tune(svm,sex~.,data=traindata,
            #   ranges = list(gamma = c(0,.01,.02,.03,.04), cost = 2^(-1:2)))
#SVM.tune
```

Prediction! Can we guess the speaker from the voice data...?

Process data & divide into train & test

```
RUNALL = FALSE # Set this to true to run time-consuming functions

rownames(data) = data[,1] # moving file names to rownames
data = data[,-1] # removing the file name col

#######################################
# Remove uncommon countries.
# Countries with a few recording data initially caused problems in predictions.
#######################################
countries = sort(table(data$country),decreasing=T)
uncommon = countries[countries<=5] # less than 5 occurences
uncommon = names(uncommon)
common = countries[countries>5] # less than 5 occurences
common = names(common)

uncommon =data[data$country %in% uncommon,]
common =data[data$country %in% common,]
uncommon$country="other"
data = rbind(common,uncommon)

data$country = droplevels(data$country) # reduce levels

#######################################
# Divide into test and train for testing
#######################################
set.seed(123)
index = sample(1:nrow(data), size=0.7*nrow(data))
train = data[index,]
test = data[-index,]

train_age = data.frame(train["age"], train[,c(1:(ncol(data)-7))])
train_sex = data.frame(train["sex"], train[,c(1:(ncol(data)-7))])
train_country = data.frame(train["country"], train[,c(1:(ncol(data)-7))])

test_age = data.frame(test["age"], test[,c(1:(ncol(data)-7))])
test_sex = data.frame(test["sex"], test[,c(1:(ncol(data)-7))])
test_country = data.frame(test["country"], test[,c(1:(ncol(data)-7))])
```

Model 1: SVM

```
# Gender prediction - Basic Model
set.seed(123)
svmfit_sex = svm(sex ~ ., train_sex)
svmpred_sex = predict(svmfit_sex, test_sex)
table(svmpred_sex, test_sex$sex)
```

```
##
## svmpred_sex female male
##      female    297   19
##      male       20  304
```

```r
postResample(svmpred_sex, test_sex$sex)
```

```
##  Accuracy     Kappa
## 0.9390625 0.8781107
```

Not bad. Let's see if we can tune it to make it better.

```r
#It takes a while to run.
if (RUNALL){
  svmtuned_sex <- tune(svm, sex ~ .,  data = train_sex,
                     ranges = list(epsilon = seq(0,1,0.1), cost = 2^(2:9)))
  print(svmtuned_sex)
  plot(svmtuned_sex)
}

# According to the output, best parameters are:
#  epsilon cost
#       0    8
```

```r
if (TRUE){
  svmfit_sex_better = svm(sex ~ ., train_sex, epsilon=0, cost=8)
  svmpred_sex_better = predict(svmfit_sex_better, test_sex)
  table(svmpred_sex_better, test_sex$sex)
  postResample(svmpred_sex_better, test_sex$sex)

  # Unfortunately it didn't improve...
  # Accuracy     Kappa
  #0.9328125 0.8656093
}
```

```
##  Accuracy     Kappa
## 0.9328125 0.8656093
```

```r
# Age & Country prediction.
set.seed(123)
svmfit_age = svm(age ~ ., train_age)
svmpred_age = predict(svmfit_age, test_age)
MSE_svm = mean( (svmpred_age - test_age$age)^2)
MSE_svm
```

```
## [1] 185.0999
```

```r
if (RUNALL){
  svmtuned_age <- tune(svm, age ~ .,  data = train_age,
                       ranges = list(epsilon = seq(0,1,0.1), cost = 2^(2:9)))
  print(svmtuned_age)
  #- best parameters:
  #  epsilon cost
  #    0.5    4
  #plot(svmtuned_age)


}
svmfit_age_better = svm(age ~ ., train_age, epsilon=0.5, cost=4)
  svmpred_age_better = predict(svmfit_age_better, test_age)
  MSE_svm_better = mean( (svmpred_age_better - test_age$age)^2)
  MSE_svm_better # 182.7476 slight improvement
```

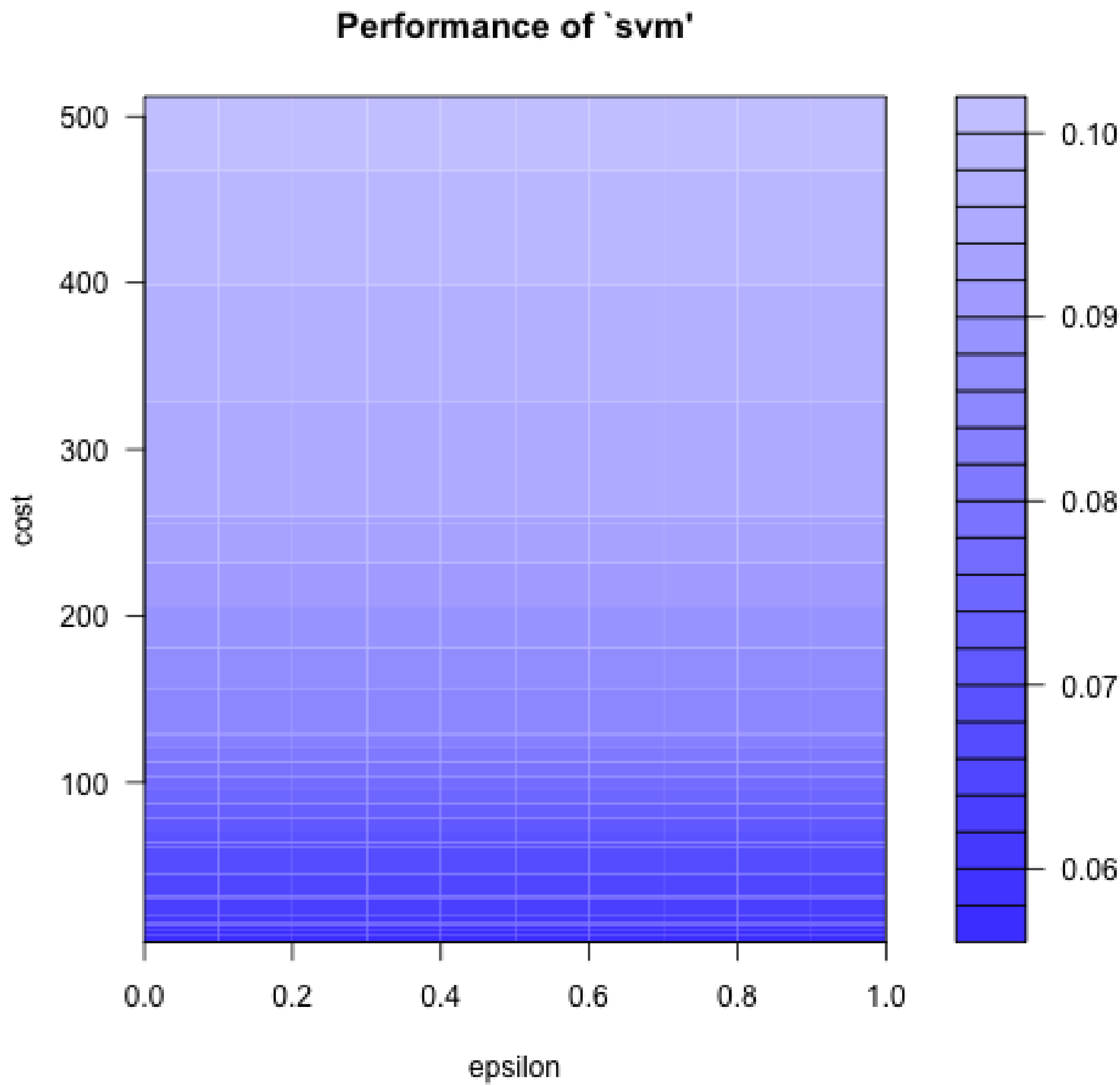

Performance of `svm'
cost
epsilon
500
400
300
200
100
0.0
0.2
0.4
0.6
0.8
1.0
0.10
0.09
0.08
0.07
0.06

Figure 2:

```
## [1] 182.7476
```

```r
# Country
svmfit_co = svm(country ~ ., train_country)
svmpred_co = predict(svmfit_co, test_country)
#table(svmpred_co,test_country$country)
postResample(svmpred_co, test_country$country)
```

```
##   Accuracy      Kappa
## 0.21562500 0.07901526
```

For country, tuning took too long. For more details, please see the (prediction.r)[../lib/prediction.r] file

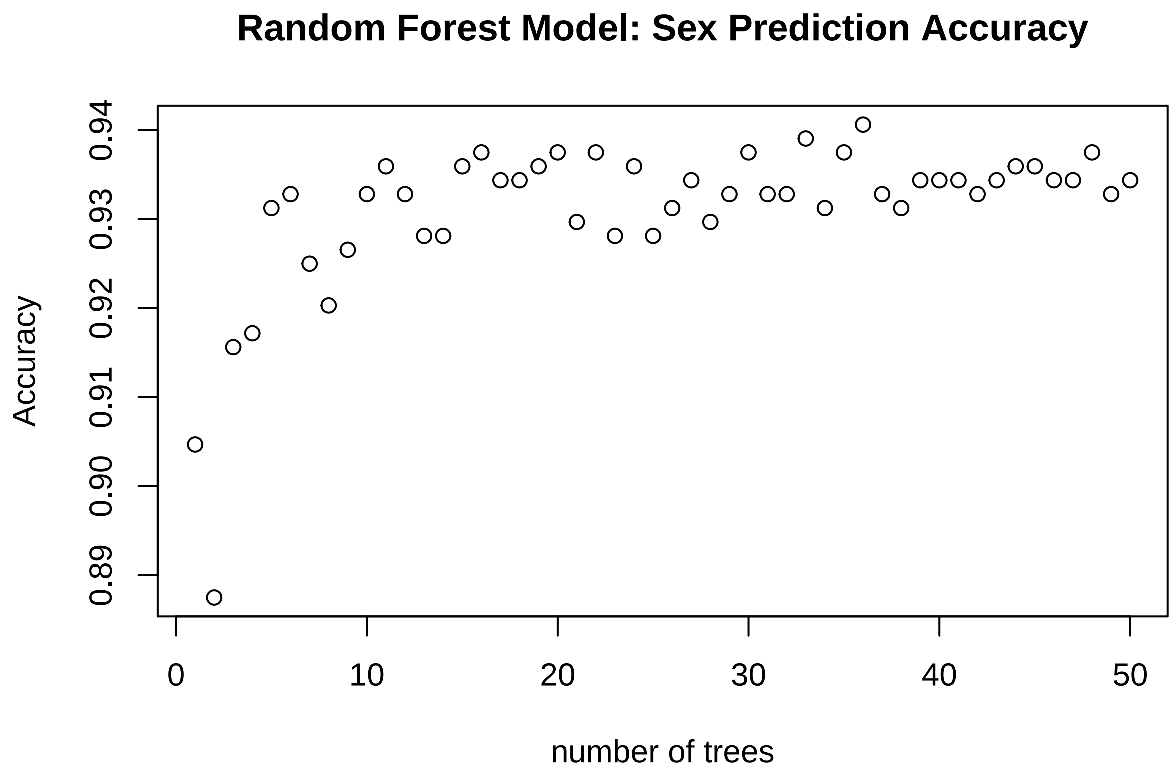
Model 2: Random Forest

```r
#Gender prediction
if(TRUE){
  set.seed(123)
  accuracy_vec_sex = c()
  for (ntree in 1:50){
    mtry_sex = tuneRF(x=subset(train_sex, select=-sex), y = train_sex$sex,
                    ntree=ntree,trace=FALSE,plot=FALSE)
    best_mtry_sex = mtry_sex[,1][which.min(mtry_sex[,2])]
    rffit_sex = randomForest(sex ~ ., data = train_sex, ntree=ntree,
                           importance=T, mtry = best_mtry_sex)
    rfpred_sex = predict(rffit_sex, test_sex)
    accuracy = postResample(rfpred_sex, test_sex$sex)
    accuracy_vec_sex = c(accuracy_vec_sex, accuracy[[1]])
}
names(accuracy_vec_sex) = 1:50

plot(accuracy_vec_sex,xlab="number of trees",ylab="Accuracy",
     main="Random Forest Model: Sex Prediction Accuracy")

which.max(accuracy_vec_sex)
}
```

```
## -0.4009091 0.05
## -0.3213645 0.05
## -0.6190117 0.05
## 0.2577747 0.05
## -0.1134849 0.05
## -0.3734977 0.05
## 0.135051 0.05
## 0.09255152 0.05
## 0.06947474 0.05
## -0.2402994 0.05
## 0.2498253 0.05
## 0.02257972 0.05
## -0.3785383 0.05
## 0.2378638 0.05
## -0.02413198 0.05
## -0.3329477 0.05
## 0.1196002 0.05
## 0.09867891 0.05
## -0.06244752 0.05
```

```
## -0.3236313 0.05
## 0.09027292 0.05
## 0.04078385 0.05
## -0.1399274 0.05
## 0.2787234 0.05
## -0.09188963 0.05
## -0.2553214 0.05
## 0.1087654 0.05
## 0.03640681 0.05
## -0.4141646 0.05
## 0.0496748 0.05
## -0.07751287 0.05
## 0.2013889 0.05
## 0.04831835 0.05
## -0.02324545 0.05
## 0.2341142 0.05
## -0.07078775 0.05
## -0.2651589 0.05
## 0.1743411 0.05
## -0.1325712 0.05
## -0.1027072 0.05
## 0.1022416 0.05
## 0.03080281 0.05
## -0.273655 0.05
## -0.01960784 0.05
## -0.4854369 0.05
## 0.09708738 0.05
## -0.0774367 0.05
## -0.03846154 0.05
## -0.00268998 0.05
## -0.2075472 0.05
## 0.1226415 0.05
## -0.1397849 0.05
## -0.3541667 0.05
## 0.04102349 0.05
## -0.3229167 0.05
## -0.07291667 0.05
## -0.1122449 0.05
## 0.08163265 0.05
## -0.1444444 0.05
## -0.1958763 0.05
## -0.04123711 0.05
## -0.1057692 0.05
## 0.08653846 0.05
## -0.03157895 0.05
## -0.1888889 0.05
## -0.2555556 0.05
## -0.1684211 0.05
## 0.030929 0.05
## -0.1269033 0.05
## 0.01130185 0.05
## -0.1136364 0.05
## -0.04545455 0.05
## -0.3103448 0.05
```

```
## -0.08045977 0.05
## -0.2626263 0.05
## 0.1212121 0.05
## -0.1034483 0.05
## -0.3863636 0.05
## -0.01136364 0.05
## -0.1263158 0.05
## 0.07368421 0.05
## -0.04545455 0.05
## -0.4805195 0.05
## -0.1298701 0.05
## -0.2197802 0.05
## -0.03296703 0.05
## -0.1521739 0.05
## 0.02173913 0.05
## -0.1978022 0.05
## 0.03296703 0.05
## -0.2298851 0.05
## -0.02298851 0.05
## -0.3536585 0.05
## -0.1097561 0.05
## -0.4545455 0.05
## -0.07792208 0.05
## -0.06521739 0.05
## 0.08695652 0.05
## -0.0952381 0.05
## -0.2111111 0.05
## 0.1222222 0.05
## -0.1265823 0.05
## -0.2289157 0.05
## -0.2289157 0.05
## -0.08139535 0.05
## -0.03488372 0.05
## -0.2875 0.05
## -0.1875 0.05
## -0.07368421 0.05
## 0.1052632 0.05
## -0.05882353 0.05
## -0.125 0.05
## 0.06818182 0.05
## -0.1463415 0.05
## -0.1647059 0.05
## 0 0.05
## -0.05952381 0.05
## 0 0.05
## -0.3292683 0.05
## -0.1097561 0.05
## -0.0952381 0.05
## -0.02380952 0.05
## -0.08888889 0.05
## 0.01111111 0.05
```




```
## 36
## 36
```

```r
#36 is the best with 0.940625
```

```r
#Age prediction
if(TRUE){
  set.seed(123)
  MSE_vec = c()
  for (ntree in 1:40){
    mtry_age = tuneRF(x=subset(train_age, select=-age), y = train_age$age,
                      ntree=ntree,trace=FALSE,plot=FALSE)
    best_mtry_age = mtry_age[,1][which.min(mtry_age[,2])]
    rffit_age = randomForest(age ~ ., data = train_age, ntree=ntree,
                             importance=T, mtry = best_mtry_age)
    rfpred_age = predict(rffit_age, test_age)
    MSE = mean( (rfpred_age - test_age$age)^2)
    MSE_vec = c(MSE_vec,MSE)
  }
  names(MSE_vec) = 1:40

plot(MSE_vec,xlab="number of trees",ylab="MSE",
     main="Random Forest Model: Age Prediction MSE")

which.min(MSE_vec)
}
```

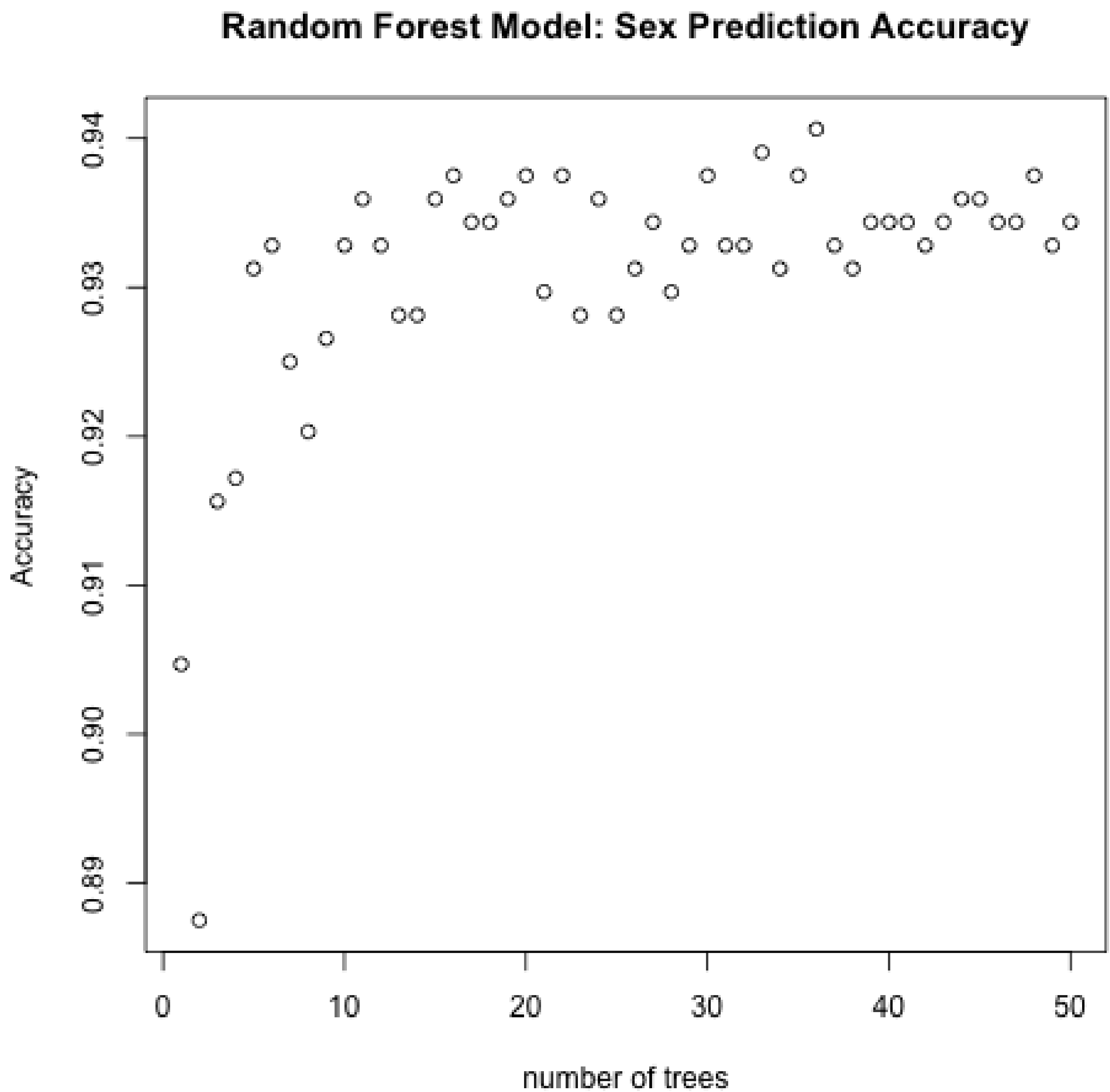
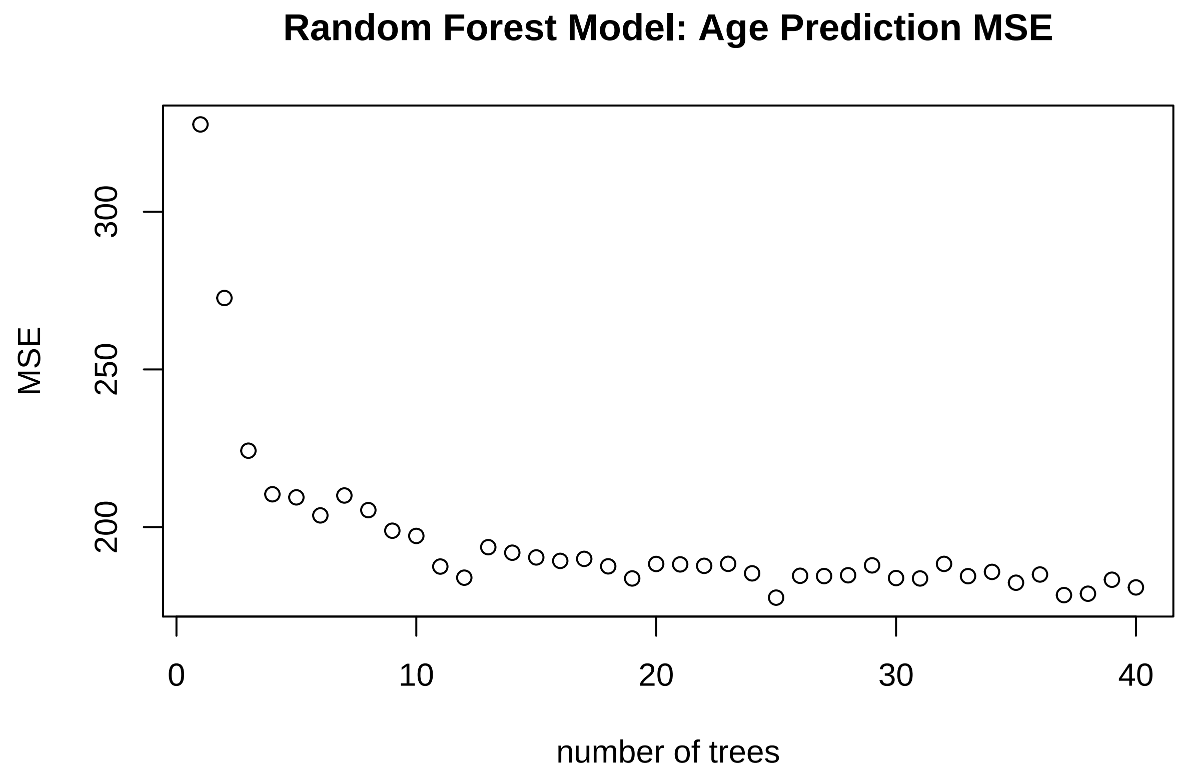

Random Forest Model: Sex Prediction Accuracy
Accuracy
0.89
0.90
0.91
0.92
0.93
0.94
0
10
20
30
40
50
number of trees

Figure 3:

```
## 0.1134478 0.05
## -0.0943922 0.05
## -0.214465 0.05
## 0.04020385 0.05
## -0.1007799 0.05
## -0.06260382 0.05
## -0.04035293 0.05
## -0.0810629 0.05
## -0.03512339 0.05
## -0.1346439 0.05
## -0.093296 0.05
## 0.02756434 0.05
## 0.01836474 0.05
## 0.02290105 0.05
## 0.02567405 0.05
## -0.009274264 0.05
## 0.01919978 0.05
## -0.01758002 0.05
## -0.07055988 0.05
## -0.02775243 0.05
## -0.03152331 0.05
## -0.001226361 0.05
## -0.008665743 0.05
## 0.07013819 0.05
## -0.0688364 0.05
## -0.07819427 0.05
## 0.03450358 0.05
## 0.02373763 0.05
## 0.02275459 0.05
## -0.02883539 0.05
## -0.005963563 0.05
## 0.01791411 0.05
## 0.0006736191 0.05
## -0.02529823 0.05
## -0.06458473 0.05
## -0.09851405 0.05
## -0.008986429 0.05
## 0.009845981 0.05
## 0.004046841 0.05
## -0.01236555 0.05
## 0.04764963 0.05
## 0.03716024 0.05
## -0.0101686 0.05
## -0.05592386 0.05
## 0.04724262 0.05
## -0.001397754 0.05
## -0.0465014 0.05
## -0.03346199 0.05
## 0.02124646 0.05
## 0.03343712 0.05
## 0.02735975 0.05
## -0.007705909 0.05
## 0.04756534 0.05
## 0.04482387 0.05
```

```
## 0.01903206 0.05
## 0.01548935 0.05
## 0.0215103 0.05
## -0.01796799 0.05
## 0.02274687 0.05
## 0.02286062 0.05
## 0.03012756 0.05
## 0.0008751451 0.05
## -0.02015274 0.05
## -0.008228953 0.05
## 0.001480469 0.05
## 0.0125013 0.05
## -0.005006792 0.05
## -0.003873472 0.05
## -0.01577598 0.05
## 0.02354879 0.05
## 0.03557664 0.05
## -2.742125e-05 0.05
## 0.03598004 0.05
## 0.01907109 0.05
## -0.004299022 0.05
## -0.005600321 0.05
## -0.005140435 0.05
## -0.02407333 0.05
## -0.00214227 0.05
## 0.001256738 0.05
## 0.02441138 0.05
## -0.01265446 0.05
```




```
## 25
## 25
```

```
#25 is the best with 177.6382
```

```
# Country prediction
if(RUNALL){
  set.seed(123)
  accuracy_vec_country = c()
  for (ntree in seq(5,200,5)){
    mtry_country = tuneRF(x=subset(train_country, select=-country),
                          y = train_country$country,
                          ntree=ntree,trace=FALSE,plot=FALSE)
    best_mtry_country = mtry_country[,1][which.min(mtry_country[,2])]
    rffit_country = randomForest(country ~ ., data = train_country, ntree=ntree,
                                 importance=T, mtry = best_mtry_country)
    rfpred_country = predict(rffit_country, test_country)
    accuracy = postResample(rfpred_country, test_country$country)
    accuracy_vec_country = c(accuracy_vec_country, accuracy[[1]])
  }
  names(accuracy_vec_country) = seq(5,200,5)

  plot(accuracy_vec_country,xlab="number of trees",ylab="Accuracy",
       main="Random Forest Model: Country Prediction Accuracy")

  which.max(accuracy_vec_country)
}
```

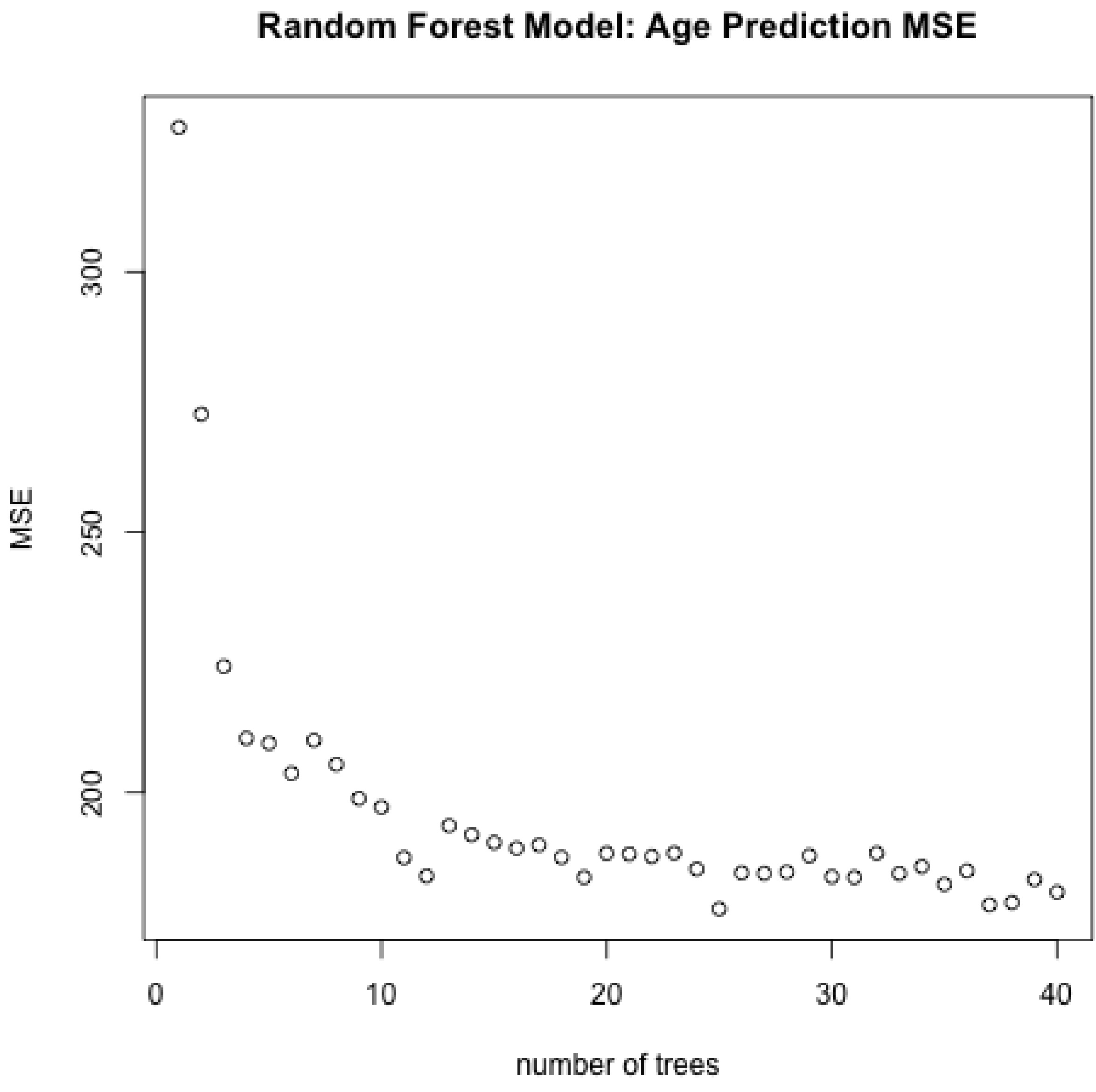

Random Forest Model: Age Prediction MSE
MSE
300
250
200
0
10
20
30
40
number of trees

Figure 4:

```
# 165 is the best with 0.2109375
```

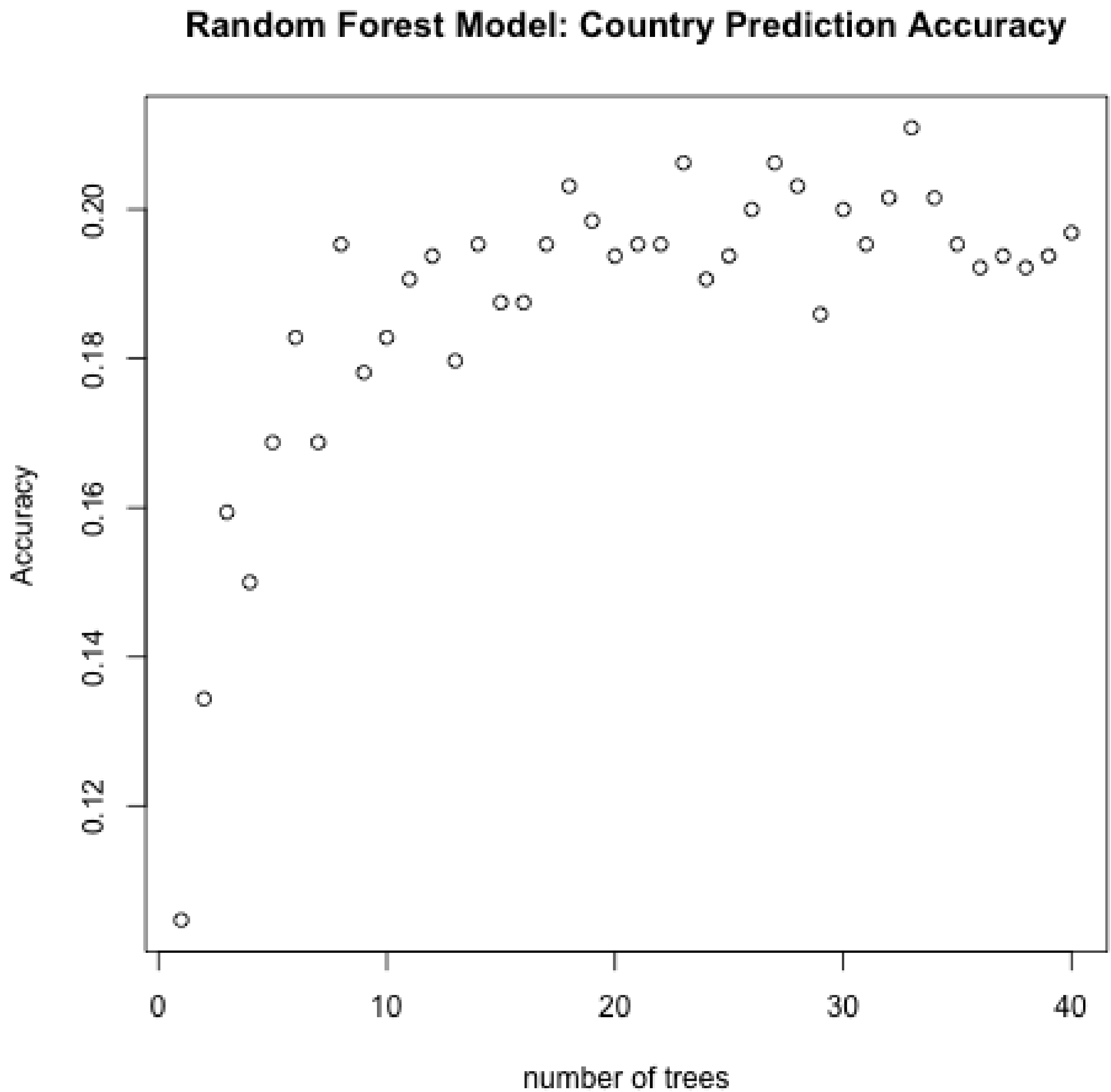


Figure 5:

Model 3: XGBoost

```
### Setting up the parameters
set.seed(123)
params_df = expand.grid(nrounds=c(100,200),
                     eta = c(0.1,0.3,0.5),
                     gamma=1,
                     max_depth = c(3,5,7,10),
```

```
                      colsample_bytree=c(0.5,0.7,0.9),
                      min_child_weight=1:2,
                      subsample = c(0.5,0.75,1))

train_control = trainControl(method = "cv", number = 5,
                              verboseIter = T, returnData = F,
                              returnResamp = "all", allowParallel = T)
```

```
# Gender prediction
if(RUNALL){
  labels_train = as.matrix(as.integer(train_sex$sex)-1)
  xgb_train_sex = train(x=subset(data.matrix(train_sex), select=-sex),
                      y=as.factor(labels_train),
                      trControl = train_control,
                      tuneGrid = params_df,
                      method = "xgbTree")

  # best param
  head(xgb_train_sex$results[with(xgb_train_sex$results,order(Accuracy, decreasing=T)),],5)

  xgb_train_sex$bestTune
  #     nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
  #26     200         3 0.1     1              0.9                1       0.5

  # run the best model
  xgbfit_sex = xgboost(data =subset(data.matrix(train_sex), select=-sex),
                        label = labels_train, objective="multi:softmax",
                        eval_metric="merror",num_class=2, verbose=F,
                        params = xgb_train_sex$bestTune,
                        nrounds=xgb_train_sex$bestTune$nrounds)

  labels_test = as.matrix(as.integer(test_sex$sex)-1)
  test_sex2 = data.frame(labels_test, subset(data.matrix(test_sex), select=-sex))

  xgpred_sex = predict(xgbfit_sex, subset(data.matrix(test_sex), select=-sex),reshape=T)
  xgpred_sex = factor(xgpred_sex,labels = c("female","male"))

  postResample(xgpred_sex, test_sex$sex)
  #Accuracy     Kappa
  #0.9421875 0.8843682

  table(xgpred_sex, test_sex$sex)
  #xgpred_sex female male
  #female    299   19
  #male       18  304
}
```

```
### For predictions with more values like country, reducing the param df & train control.
### to shorten the running time
set.seed(123)
params_df = expand.grid(nrounds=c(100,200),
                      eta = c(0.1,0.3,0.5),
                      gamma=1,
                      max_depth = c(3,5,7,10),
```

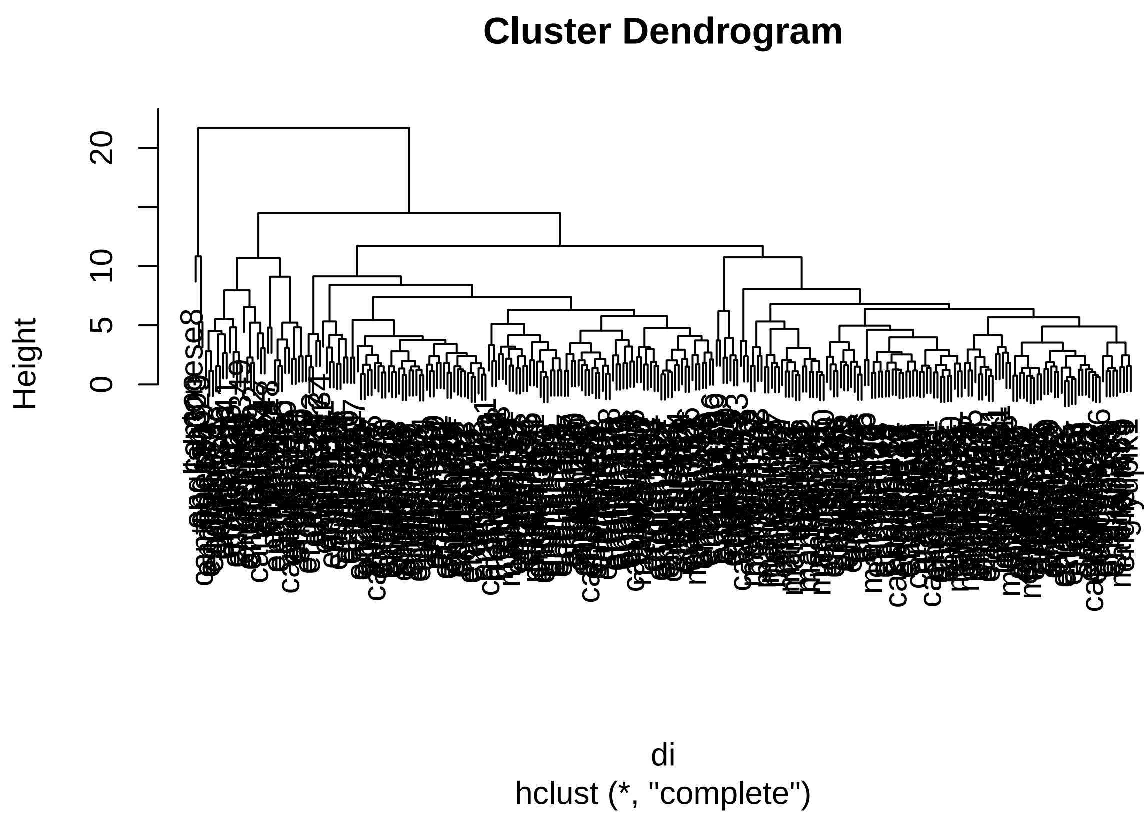
```
                      colsample_bytree=c(0.5,0.7,0.9),
                      min_child_weight=1:2,
                      subsample = c(0.5,0.75,1))

train_control = trainControl(method = "cv", number = 5,
                              verboseIter = T, returnData = F,
                              returnResamp = "all", allowParallel = T)
```

```
# Country prediction
if(RUNALL){
  set.seed(123)
  labels_train = as.matrix(as.integer(train_country$country)-1)

  # this takes REALLY long time
  xgb_train_country = train(x=subset(data.matrix(train_country), select=-country),
                          y=as.factor(labels_train),
                          trControl = train_control2,
                          tuneGrid = params_df2,
                          method = "xgbTree")
  # Best param
  head(xgb_train_country$results[with(xgb_train_country$results,order(Accuracy, decreasing=T)),],5)
  xgb_train_country$bestTune
    #     nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
    #5      100         3 0.1     1              0.7                1       0.5

  xgbfit_country = xgboost(data =subset(data.matrix(train_country), select=-country),
                         label = labels_train, objective="multi:softprob",
                         eval_metric="merror",num_class=88, verbose=F,
                         params = xgb_train_country$bestTune,
                         nrounds=xgb_train_country$bestTune$nrounds)

    xgpred_country = predict(xgbfit_country,
                         subset(data.matrix(test_country), select=-country),reshape=T)
  maxcol=apply(xgpred_country, 1, which.max)
  country = levels(test_country$country)[maxcol]
  xgpred_country = data.frame(xgpred_country, country)
}
```

Model 4 Unsupervised Clustering

```
tb<-sort(table(data$country),decreasing = T)

countrydf<-data[data$country %in% names(tb[c(1,3,4)]) & data$sex=="female",]

countrydf$country<-droplevels(countrydf$country)
#rownames(countrydf)<-countrydf$country
scaledf<-scale(countrydf[,2:19,20:23])


#kcountry<-kmeans(scaledf,2)
#table(kcountry$cluster,countrydf$country)
d<-as.matrix(cbind(scaledf))
```

```r
di<-dist(d)
hc<-hclust(di)
plot(hc)
```




```r
clustercut<-cutree(hc,3)
table(clustercut,countrydf$country)
```

```
##
## clustercut china  uk usa
##          1    50  22 168
##          2     4   2  23
##          3     1   0   2
```

```r
# hcd<-as.dendrogram(hc)
# plot(clustercut)
```

Model 5: Logistic

```r
if(TRUE){
  set.seed(123)
  # Sex
  lgfit_sex = glm(formula = as.factor(sex) ~ ., data=train_sex,
                  family=binomial(link='logit'))
  lgpred_sex = plogis(predict(lgfit_sex, test_sex))
  lgpred_sex_f <- rep('female',length(lgpred_sex))
```

```
  lgpred_sex_f[lgpred_sex>=0.5] = "male"

  table(lgpred_sex_f, test_sex$sex)
  #lgpred_sex_f female male
  #female    297   23
  #male       20  300

  postResample(lgpred_sex_f, test_sex$sex)
  #Accuracy     Kappa
  #0.9328125 0.8656250
}
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
##  Accuracy     Kappa
## 0.9328125 0.8656250
```

```
  # Age
  set.seed(123)
  train_age2 = cbind(train_age, age0_1 = train_age["age"]/100)
  lgfit_age = glm(formula = age/100 ~ ., data=train_age, family=binomial(link='logit'))
```

```
## Warning: non-integer #successes in a binomial glm!
```

```
  test_age2 = cbind(age0_1 = test_age["age"]/100, test_age[,-1])
  lgpred_age = predict(lgfit_age, test_age2, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
  MSE_lg = mean( ((lgpred_age)*100 - (test_age2$age)*100 )^2)
  MSE_lg #181.7178
```

```
## [1] 181.7178
```

```
if(RUNALL){
#Country
  set.seed(123)
  lgfit_country = multinom(formula = as.factor(country) ~ .,
                          data=train_country, MaxNWts = 140000, maxit = 1000)
  lgpred_country = predict(lgfit_country, test_country)
  #table(lgpred_country, test_country$country)
  postResample(lgpred_country, test_country$country)
  #  Accuracy      Kappa
  #0.18906250 0.09001499
}
```

More Data

```
other<-read.csv("../output/example_summary_stats.csv")
other<-other[,c(2:23,27,28,29)]
colnames(other)[24:25]<-c("sex","country")
data<-read.csv("../output/all_features.csv")
data<-data[,c(2:24,28,30)]
```

```
data$country<-as.character(data$country)
other$country<-as.character(other$country)

data$sex[data$sex=="famale"]<-"female"
bigdata<-rbind(other,data)
```

```
## Warning in `[<-.factor`(`*tmp*`, ri, value = structure(c(7L, 8L, 8L, 8L, :
## invalid factor level, NA generated
```

```
bigdata<-bigdata[,-23]
bigdata<-bigdata[bigdata$sex != "other" & !is.na(bigdata$country) & bigdata$country != "african",]
bigdata$sex<-droplevels(bigdata$sex)

bigdata$country<-gsub("kosovo","serbia",bigdata$country)
bigdata$country<-gsub("wales","uk",bigdata$country)
bigdata$country<-gsub("scotland","uk",bigdata$country)
bigdata$country<-gsub("sicily","italy",bigdata$country)
bigdata$country<-gsub("tibet","china",bigdata$country)
bigdata$country<-gsub("yugoslavia","serbia",bigdata$country)
bigdata$country<-gsub("virginia","usa",bigdata$country)
bigdata$country<-gsub("african","england",bigdata$country)
bigdata$country<-gsub("southatlandtic","usa",bigdata$country)
bigdata$country<-gsub("england","uk",bigdata$country)
bigdata$regions<-countrycode((bigdata$country),"country.name","region")



bigdata$continent<-countrycode((bigdata$country),"country.name","continent")
bigdata$continent<-as.factor(bigdata$continent)
bigdata$regions<-as.factor(bigdata$regions)


bigcont<-bigdata[,c(1:23,26)]
sel.cont<-c("Americas","Asia","Europe")
bigcont<-bigcont[bigcont$continent %in% sel.cont,]
bigcont$continent<-droplevels(bigcont$continent)
set.seed(123)
index = sample(1:nrow(bigcont), size=0.7*nrow(bigcont))
train = bigcont[index,]
test = bigcont[-index,]
tree<-rpart(continent~.,data=train)
tree.pred<-predict(tree,test,type="class")
table(tree.pred,test$continent)
```

```
##
## tree.pred  Americas Asia Europe
##   Americas      302   74    174
##   Asia           56  110     64
##   Europe         27   28     24
```

```
sum(tree.pred==test$continent)/length(test$continent)
```

```
## [1] 0.5075669
```

```
##SVM
svm.cont<-svm(continent~.,data=train)
```

```r
svm.pred<-svm(continent~.,test)
table(svm.pred$fitted,test$continent)
```

```
##
##            Americas Asia Europe
##   Americas      338   91    194
##   Asia           38  112     40
##   Europe          9    9     28
```

```r
print("Accuracy On Continent")
```

```
## [1] "Accuracy On Continent"
```

```r
mean(svm.pred$fitted==test$continent)
```

```
## [1] 0.556461
```

```r
#SVM.tune<-tune(svm,continent~.,data=train,
 #              ranges = list(gamma = c(0,.01,.02,.03,.04), cost = 2^(-1:2)))

#             bigdata$USA<-ifelse(bigdata$regions=="Northern America",1,0)
#
# logdata<-bigdata[,c(1:23,27)]
#
# model <- glm(USA~.,family=binomial(link='logit'),data=logdata[index,])
# summary(model)
# lp<-predict(model,logdata[-index,],type="response")
# lp<-ifelse(lp>.5,1,0)
# table(lp,logdata$USA[-index])

# logdata$USA<-ifelse(logdata$USA==1,"USA","NOT")
# tree2<-rpart(USA~.,data=logdata[index,])
# tree.pred<-predict(tree2,logdata[-index,],type="class")
# table(tree.pred,logdata$USA[-index])
# sum(tree.pred==test$continent)/length(test$continent)

#plot(tree)
```

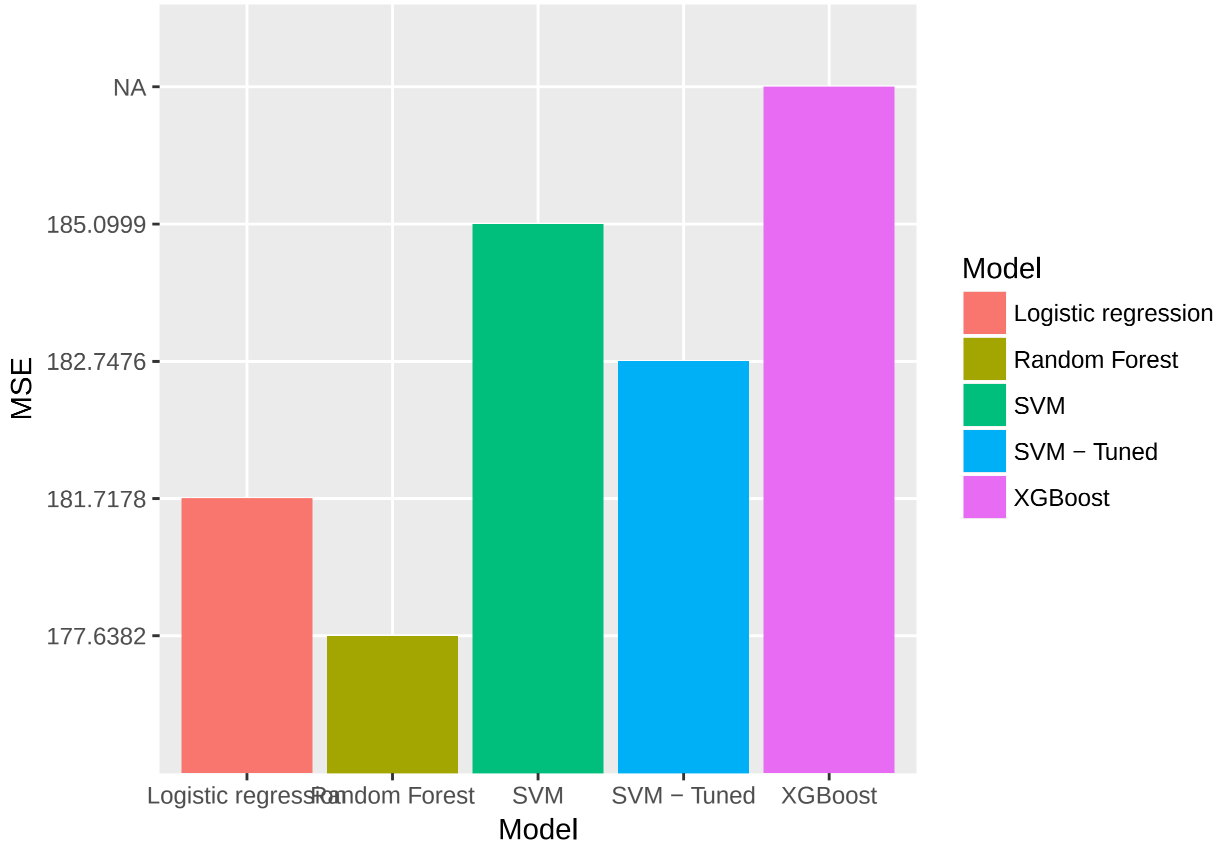
Conclusion

```r
a_row1 = c("SVM", "No parameter", 185.0999)
a_row2 = c("SVM - Tuned", "epsilon=0.5, cost=4", 182.7476)
a_row3 = c("Random Forest", "ntree = 25", 177.6382)
a_row4 = c("XGBoost", NA, NA)
a_row5 = c("Logistic regression", "No parameter", 181.7178)

age_result = data.frame(rbind(a_row1,a_row2,a_row3,a_row4,a_row5))
colnames(age_result) = c("Model", "Model Info", "MSE")
age_result
```

```
##                       Model          Model Info      MSE
## a_row1                  SVM        No parameter 185.0999
## a_row2          SVM - Tuned epsilon=0.5, cost=4 182.7476
## a_row3        Random Forest          ntree = 25 177.6382
```

```
## a_row4              XGBoost               <NA>     <NA>
## a_row5 Logistic regression        No parameter 181.7178
```

```
ggplot(age_result)+geom_bar(aes(y=MSE,x=Model,fill=Model),stat="identity")
```

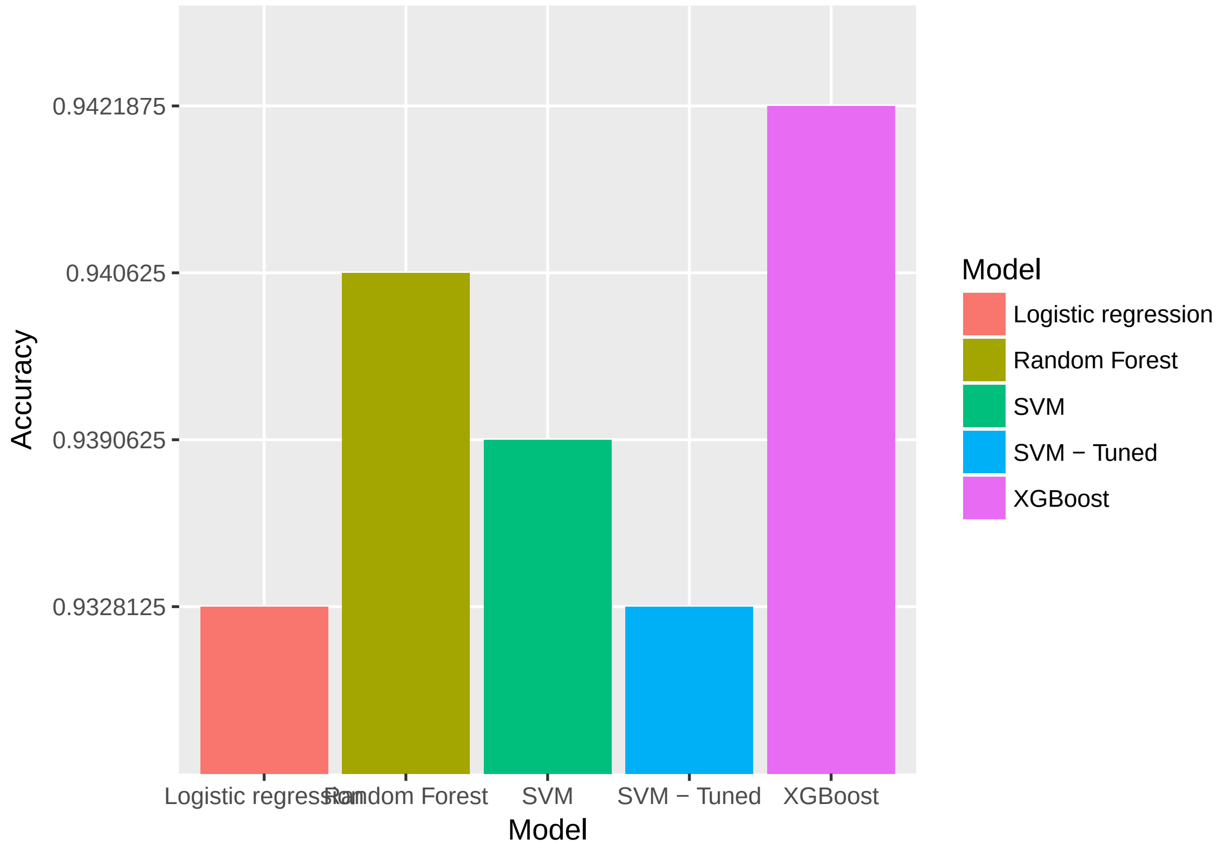



```
s_row1 = c("SVM", "No parameter", 0.9390625)
s_row2 = c("SVM - Tuned", "epsilon=0, cost=8", 0.9328125)
s_row3 = c("Random Forest", "ntree = 36", 0.940625)
s_row4 = c("XGBoost", "nrounds=200, max_depth=3, eta=0.1, gamma=1, colsample_bytree=0.9, min_child_weig
s_row5 = c("Logistic regression", "No parameter", 0.9328125)

sexpred_result = data.frame(rbind(s_row1,s_row2,s_row3,s_row4,s_row5))
colnames(sexpred_result) = c("Model", "Model Info", "Accuracy")

sexpred_result
```

```
##                      Model
## s_row1                 SVM
## s_row2         SVM - Tuned
## s_row3       Random Forest
## s_row4             XGBoost
## s_row5 Logistic regression
##                                                                                                         Mode
## s_row1                                                                                                 No para
## s_row2                                                                                            epsilon=0, 
## s_row3                                                                                                  ntree
## s_row4 nrounds=200, max_depth=3, eta=0.1, gamma=1, colsample_bytree=0.9, min_child_weight=1, subsampl
```

```
## s_row5                                                                                    No para
##           Accuracy
## s_row1 0.9390625
## s_row2 0.9328125
## s_row3  0.940625
## s_row4 0.9421875
## s_row5 0.9328125
```

```
ggplot(sexpred_result)+geom_bar(aes(y=Accuracy,x=Model,fill=Model),stat="identity")
```




```
c_row1 = c("SVM", "No parameter", 0.215625)
#c_row2 = c("SVM - Tuned", NA, NA)
c_row3 = c("Random Forest", "ntree = 165", 0.2109375)
c_row4 = c("XGBoost", "nrounds=100, max_depth=3, eta=0.1, gamma=1, colsample_bytree=0.7, min_child_weig
c_row5 = c("Multinomial logistic regression", "MaxNWts = 140000, maxit = 1000", 0.1890625)

country_result = data.frame(rbind(c_row1,c_row3,c_row4,c_row5))
colnames(country_result) = c("Model", "Model Info", "Accuracy")

country_result
```

```
##                                  Model
## c_row1                             SVM
## c_row3                   Random Forest
## c_row4                         XGBoost
## c_row5 Multinomial logistic regression
##                                                                                              Model
```

```
## c_row1                                                                                       No para
## c_row3                                                                                        ntree
## c_row4 nrounds=100, max_depth=3, eta=0.1, gamma=1, colsample_bytree=0.7, min_child_weight=1, subsampl
## c_row5                                                                      MaxNWts = 140000, maxit =
##          Accuracy
## c_row1  0.215625
## c_row3 0.2109375
## c_row4 0.1953125
## c_row5 0.1890625
```

```
ggplot(country_result)+geom_bar(aes(y=Accuracy,x=Model,fill=Model),stat="identity")
```