INFO523 Decision Trees

Sebastian Deimen & Noah Giebink

21 März 2020

Preprocessing

At first, we are going to make two sets of our spot-data: one only related to the music vaiables and one also including the socio- variables.

```
# preprocessing the data set

spot_music <- spot %>% select(track.popularity, track.explicit, danceability, key,
                              loudness, mode, speechiness, acousticness, instrumentalness,
                              liveness, valence, tempo, country)

# trying another set of varibales due to horrible error rates for the first set
# I tried without track.popularity, makes it worse, so another try, including socio-variables

spot_music_socio <- spot %>% select(track.popularity, track.explicit, danceability,
                                    key, loudness, mode, speechiness, acousticness,
                                    instrumentalness,
                                    liveness, valence, tempo, happiness, median_age,
                                    percent_urban, percent_internet_users, density_sqkm,
                                    freedom, gdp, country)
```

Overview

Step 1: build a decision tree to classify countries using social variables.

Step 1B: Interesting rules for distinguishing countries

Step 2: use most important variable from Stage 1 to cluster countries (the tree in Step 3 performed better with fewer classes this way)

Step 3: build a decision tree to classify clustered countries by music variables (dimensions of music taste)

Step 3B: Interesting rules

Step 4: Compare performance of decision tree in Step 3 to Random Forest

Step 1. Decision tree

Split Train/Test

We split the spot_music_SOCIO data into training and test data, not using a validation set.

```
split_index <- createDataPartition(spot_music_socio$country, p= 0.8, list = F)
```

```r
spot_music_socio_train <- spot_music_socio[split_index,]
spot_music_socio_features_test <-
  spot_music_socio[-split_index, !(colnames(spot_music_socio) %in% c("country"))]
spot_music_socio_target_test <- spot_music_socio[-split_index, "country"]
```

```r
# tree building

ct <- rpartXse(country ~ ., spot_music_socio_train, se=0.5)

# prediction using the trees

pred <- predict(ct, spot_music_socio_features_test, type = "class")

# have a look at the variable.importance
ct$variable.importance
```

```
##              happiness           density_sqkm percent_internet_users
##            774.8000000            681.4666667            632.8000000
##          percent_urban             median_age                freedom
##            618.1333333            574.1333333            418.0301075
##                    gdp       track.popularity           danceability
##            330.9918280             34.2359857             16.0000000
##            speechiness                  tempo         track.explicit
##             12.0000000             10.0000000             10.0000000
##           acousticness               loudness               liveness
##              4.0000000              3.0000000              1.0000000
##       instrumentalness
##              0.9932616
```

```r
# confusion matrix
cm <- table(pred, spot_music_socio_target_test)

# error rate
error <- (1-sum(diag(cm))/sum(cm))
cat(" error rate: ",error)
```

```
##  error rate:  0
```

```r
prp(ct, type = 1, extra = 103, roundint = FALSE)
```

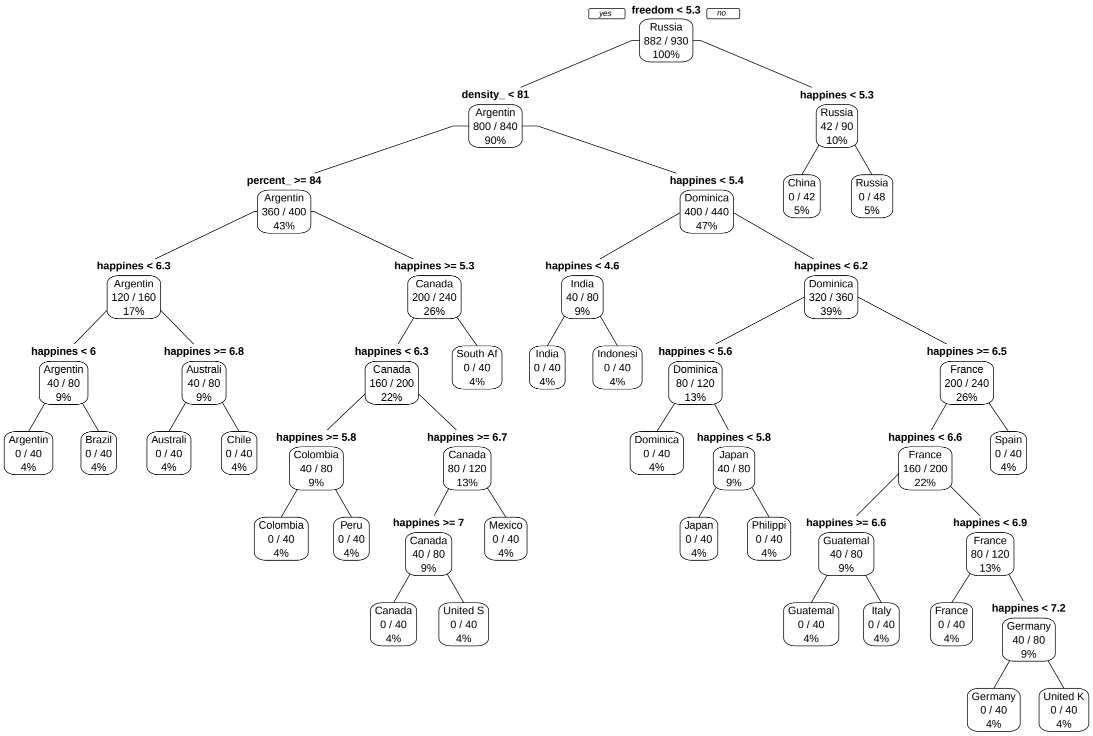


Figure 1. First decision tree: classifying countries by social variables.

Why is the error rate 0?

Seems to good to be true…Let's examine the happiness variable.

```
ggplot(spot_music_socio, aes(country, happiness))+
  geom_boxplot()+
  theme(axis.text.x = element_text(angle = 90))
```

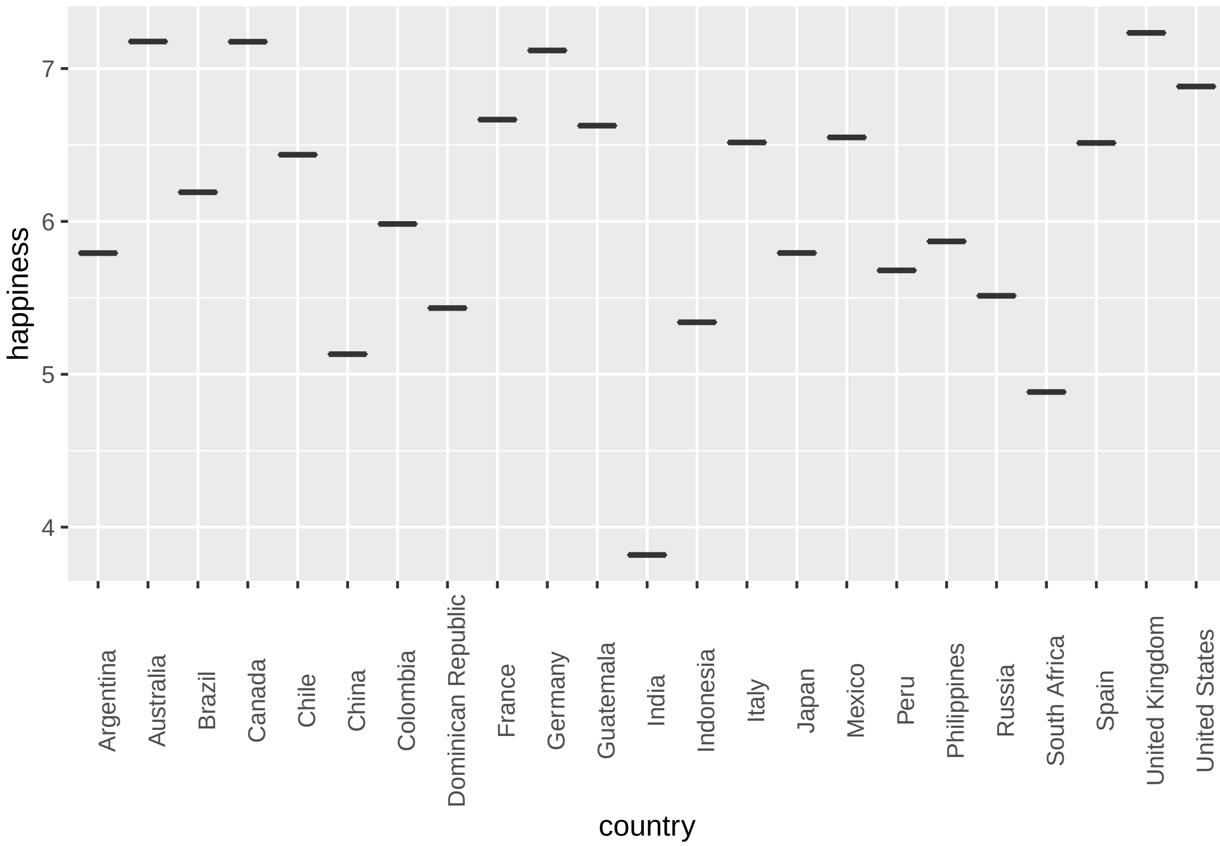


Figure 2. Each country has a single happiness value (boxplot lacks quantiles, etc) spread over each tuple for that country (by virtue of the sociopolitical data source's methods). Therefore, if at least one tuple from each country made it into both the training and test data, this could lead to a perfect error rate.

Solution: Discretize variables and re-run decision tree

```
disc <- function(x){
  cut(x, breaks = 4,
      labels = c(1:4))}

  # apply disc fun to all dbl vars except track popularity
soc_disc <- mutate_if(spot_music_socio, is.numeric, funs(disc))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

Examine distribution of levels

```r
soc2 <- select(soc_disc, -track.explicit, -country)
soc_long <- pivot_longer(soc2, cols = colnames(soc2),
                         names_to = 'variable', values_to = 'level')
ggplot(soc_long, aes(level))+
  geom_bar()+
  facet_wrap(~variable)+
  theme(axis.text.x = element_text(angle = 90))
```

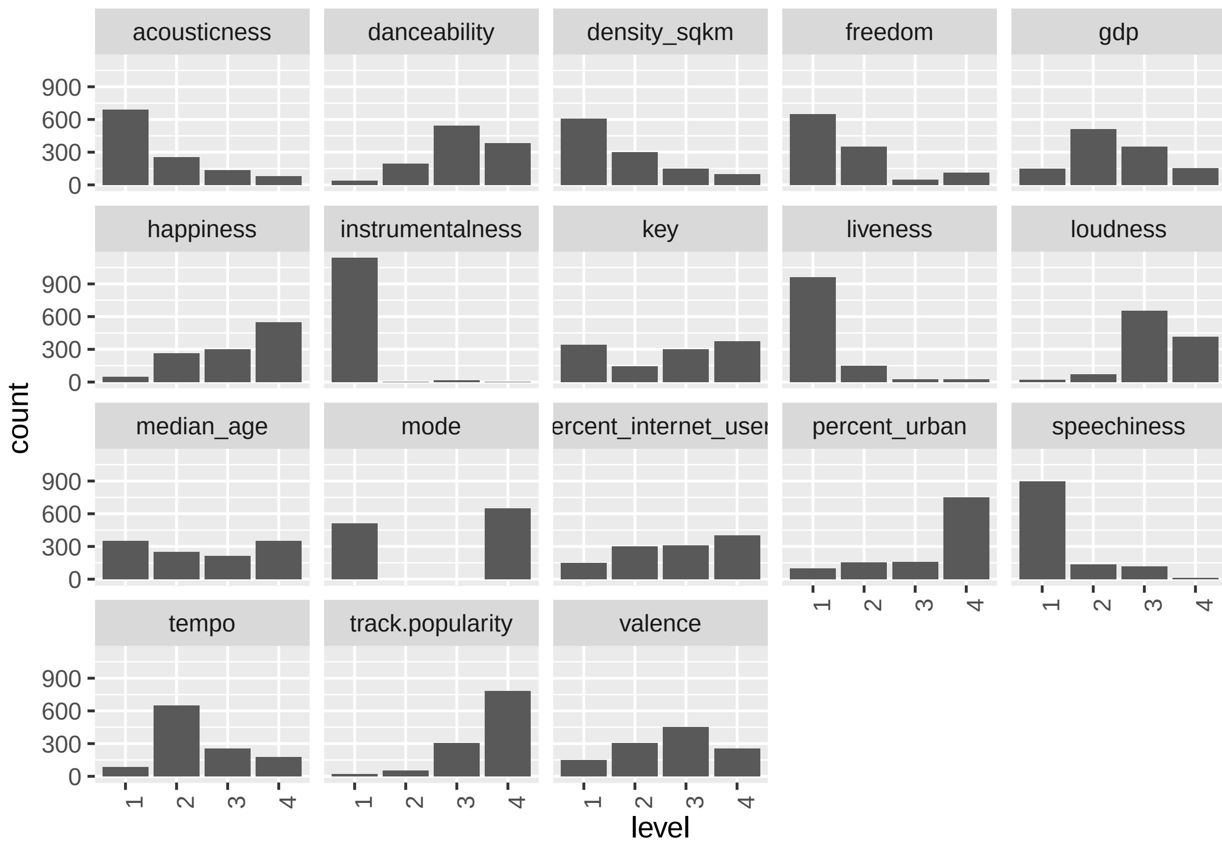


Figure 3. Distribution of discretized levels.

Socio-political tree with discretized variables

```r
# splitting the data

split_index <- createDataPartition(soc_disc$country, p= 0.8, list = F)

soc_train <- soc_disc[split_index,]
soc_test <- soc_disc[-split_index, !(colnames(soc_disc) %in% c("country"))]
soc_target <- soc_disc[-split_index, "country"]

# build the tree

ct2 <- rpartXse(country ~ ., soc_train, se=0.5)

# prediction using the trees

pred2 <- predict(ct2, soc_test, type = "class")
```

```r
# have a look at the variable.importance
ct2$variable.importance
```

```
##            median_age                    gdp             happiness
##            280.966308             240.000000            180.000000
##         percent_urban percent_internet_users               freedom
##            122.800000             120.000000             96.696774
##          density_sqkm       track.popularity          danceability
##             92.000000              84.182616             67.216667
##                 tempo         track.explicit              loudness
##             61.800000              59.000000             54.500000
##               valence            speechiness          acousticness
##             50.666667              49.000000             36.633333
##                  mode               liveness                   key
##             27.133333              16.000000              8.000000
##      instrumentalness
##              3.493262
```

```r
# confusion matrix
cm2 <- table(pred, soc_target)

# error rate
error2 <- (1-sum(diag(cm2))/sum(cm2))
cat("error rate (categorical features): ",error2)
```

```
## error rate (categorical features):  0
```

```r
prp(ct2, type = 1, extra = 103, roundint = FALSE)
```

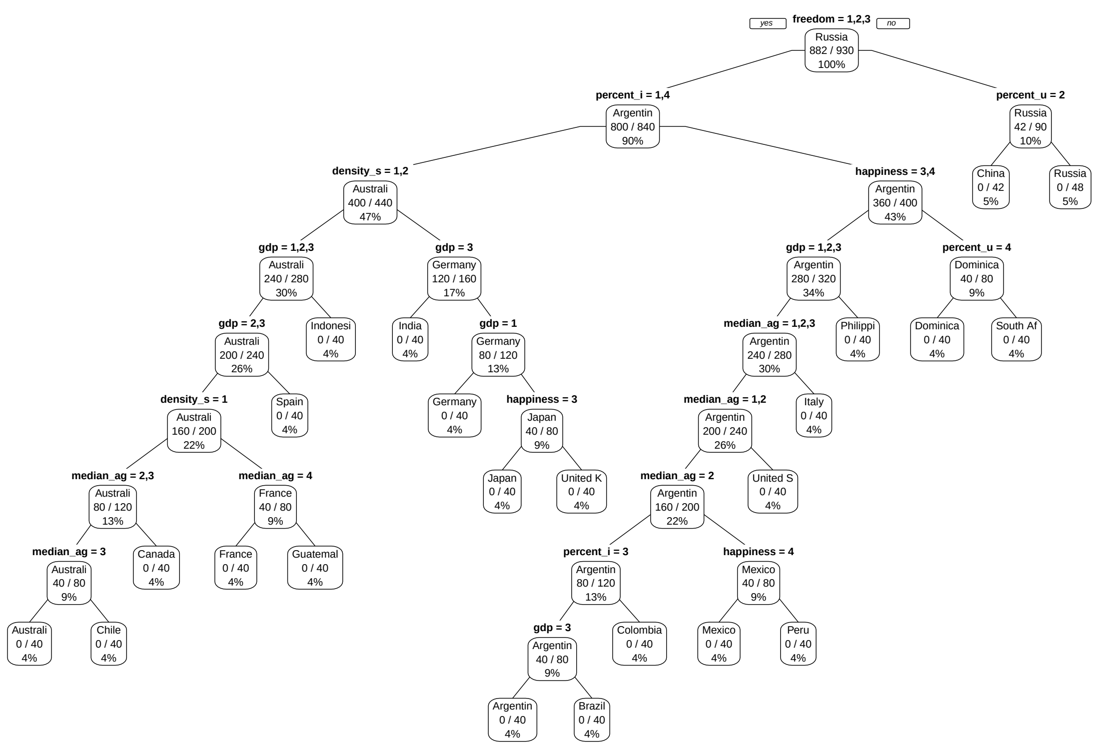


Figure 4. Classification of countries using discretized social variables. We chose not to prune the tree because it already has impeccable performance on the test data. The error rate is still 0.

Step 1B: interesting rules

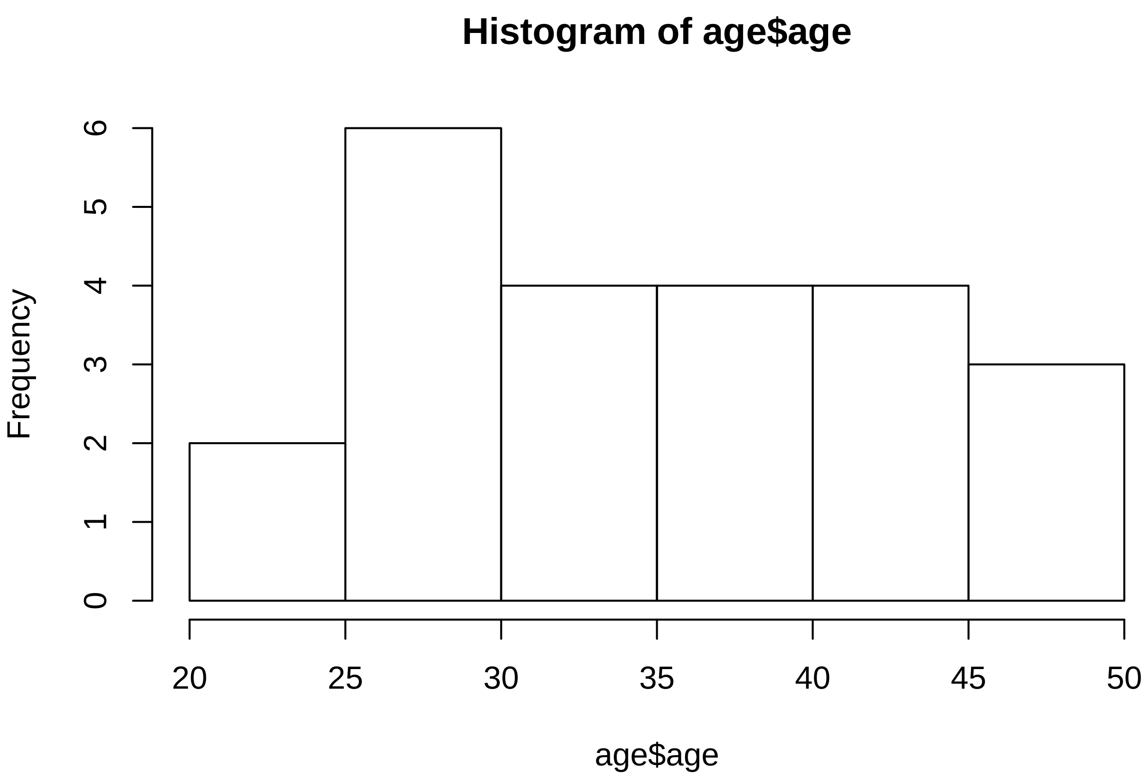
1. If freedom != 1,2,3 (1 is highest) and percent urban = 2, then country = China
2. If freedom != 1,2,3 (1 is highest) and percent urban != 2, then country = Russia (note: Russia's percent urban is 74.3 (> level 2))
3. If freedom = 1,2,3 (all but lowest), percent internet users != 1,4 (moderate), happiness != 3,4 (below 50th percentile), and percent urban = 4 (highest), then country = Dominican Republic

Step 2. Use important variable from tree in Step 1 to cluster countries

Our goal is to classify countries by music tastes. To make results more interpretable, we clustered countries by the most important variable in the decision tree shown in Fig. 4, *median_age*, for classification (this also improved performance over a previous tree, not shown). We decided to use two k = 2 to get "old" and "young" countries. We then bound the clusters to our solely music-variable data and used this to grow the tree.

In essence, our question is: what are the most important music variables that distinguish 'old' countries' music taste from 'young' countries?

```
set.seed(42)
age <- spot_music_socio %>% group_by(country) %>%
  summarise(age = mean(median_age))
age_hist <- hist(age$age)
```




```
# cluster countries by happiness, 3 clusters
a <- kmeans(age$age, 2)
a$cluster
```

```
##  [1] 1 2 1 2 1 2 1 1 2 2 1 1 1 2 2 1 1 1 2 1 2 2 2
```

```r
age_clust <- cbind(age, a$cluster)
age_clust <- rename(age_clust, cluster = 'a$cluster')
arrange(age_clust, cluster)
```

```
##                country  age cluster
## 1            Argentina 30.8       1
## 2               Brazil 31.3       1
## 3                Chile 33.7       1
## 4             Colombia 30.1       1
## 5   Dominican Republic 26.1       1
## 6            Guatemala 21.3       1
## 7                India 26.7       1
## 8            Indonesia 28.0       1
## 9               Mexico 27.5       1
## 10                Peru 27.5       1
## 11         Philippines 24.1       1
## 12        South Africa 26.1       1
## 13           Australia 37.4       2
## 14              Canada 40.5       2
## 15               China 37.0       2
## 16              France 41.2       2
## 17             Germany 45.9       2
## 18               Italy 45.9       2
## 19               Japan 46.3       2
## 20              Russia 38.7       2
## 21               Spain 43.2       2
## 22      United Kingdom 40.2       2
## 23       United States 37.6       2
```

```r
young <- filter(age_clust, cluster == 1) %>%
  select(country)
old <- filter(age_clust, cluster == 2) %>%
  select(country)

age_music <- spot_music %>% mutate(cluster =
                                      ifelse(country %in% young$country,
                                             'young', 'old'))
```

```r
# get rid of country
age_music2 <- select(age_music, -country)
# splitting the data
index_age <- sample(1:nrow(age_music2),0.8*nrow(age_music2))
train_age <- age_music2[index_age,]
test_age <- age_music2[-index_age,]
```

```r
# making a tree
set.seed(42)
ct_age <- rpartXse(cluster ~ ., train_age, se=0.1)

# prediction using the trees

pred_age <- predict(ct_age, test_age, type = "class")

# have a look at the variable.importance
ct_age$variable.importance
```

```
## track.popularity          loudness      danceability          liveness
##        56.298012         46.181401         40.188362         39.264202
##          valence       speechiness             tempo      acousticness
##        36.702339         31.893695         31.614387         27.126734
##   track.explicit               key              mode  instrumentalness
##        21.803740         13.448384          8.021733          6.615569
```

```r
# contingency tables
cm_age <- table(pred_age,test_age$cluster)

# error rate
error_age <- (1-sum(diag(cm_age))/sum(cm_age))
cat("DT on age clusters error rate: ",error_age)
```

```
## DT on age clusters error rate:  0.2618026
```

```r
prp(ct_age, type = 1, extra = 103, roundint = FALSE)
```

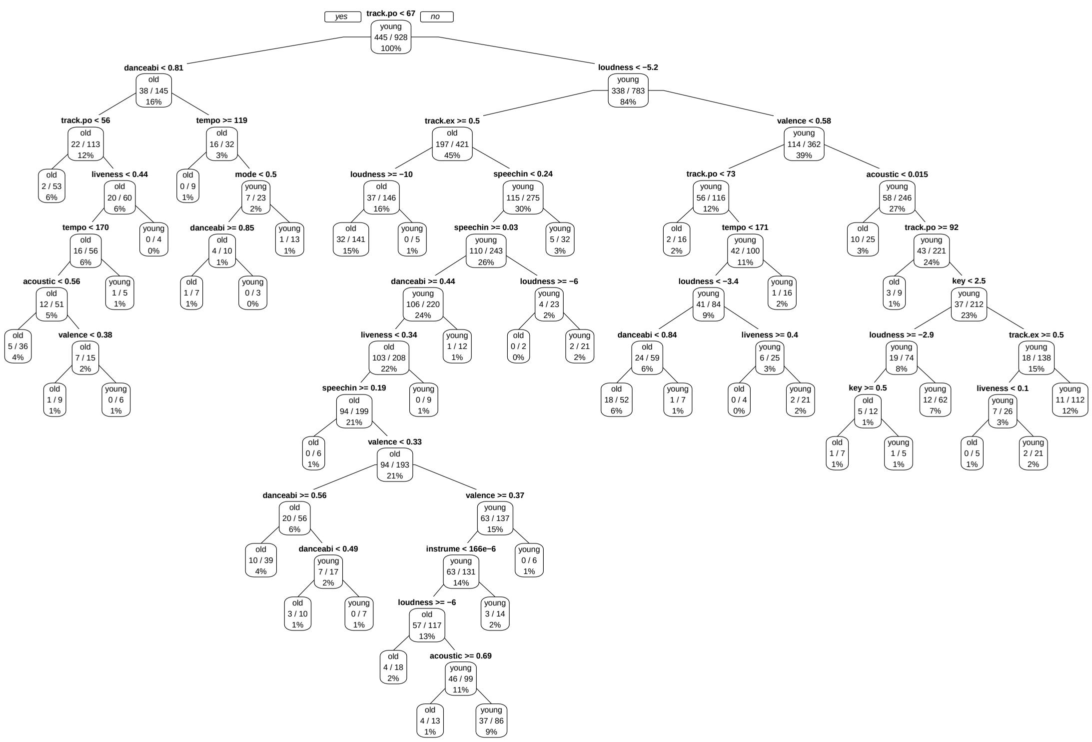


Figure 6. Classification of young and old countries.

Step 3B: interesting rules

1. If track popularity >= 70 (scale 0-100) and speechiness >= 0.046 (range 0.02-0.56 in our data), then median age is 'old.'
2. If track popularity <70 then median age is 'young.'
3. IF track popularity >70 and speechiness < 0.046 (extremely low), then median age is 'young.'

Step 4. Compare performance with Random Forest

```r
set.seed(42)
# make sure all variables are factors
soc_disc$track.explicit <- as.factor(soc_disc$track.explicit)
# remove non-musical variables
mus <- soc_disc %>% select(-happiness, -median_age, -percent_urban,
                           -percent_internet_users, -density_sqkm,
                           -freedom, -gdp) %>%
  mutate(cluster = ifelse(country %in% young$country,
                          'young', 'old')) %>%
  select(-country)
mus$cluster <- as.factor(mus$cluster)

# new train/test split for mus
mus_index <- createDataPartition(mus$cluster, p= 0.8, list = F)
mus_train <- mus[mus_index,]
mus_test <- mus[-mus_index,]

# grow the random forest
rf_tree <- randomForest(cluster ~ ., mus, ntree=500, importance=TRUE, na.action = na.omit )


# View(rf_tree$predicted)
# View(rf_tree$votes) #get the probablity of the prediction

# predict
rf_pred <- predict(rf_tree, mus_test, type = "class")
#cm_music <- table(pred_music,test_music$cluster)
# confusion matrix
rf_cm <- rf_tree$confusion

# error rate
rf_error <- (1-sum(diag(rf_cm))/sum(rf_cm))
cat("error rate of random forest: ", rf_error)
```

```
## error rate of random forest:  0.3061371
```

```r
# variable importance
importance(rf_tree)
```

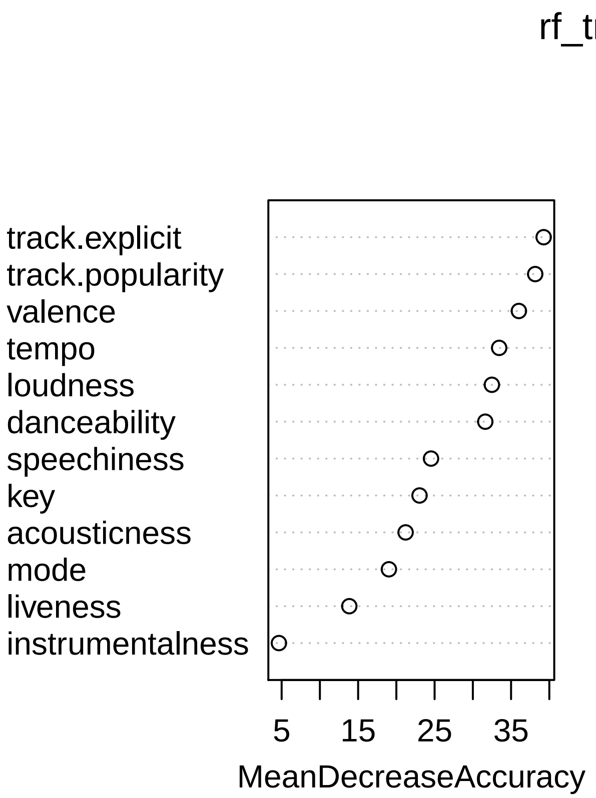
```
##                          old     young MeanDecreaseAccuracy
## track.popularity 14.8610114 38.536835            38.156838
## track.explicit   26.1189837 28.218747            39.260859
## danceability      6.6209921 31.329210            31.627768
## key               0.3814949 25.155949            23.031012
## loudness         20.0696730 24.866302            32.488752
## mode              3.4348690 18.752703            19.025058
## speechiness       8.8587460 22.417634            24.533138
## acousticness      3.3322197 21.872132            21.210333
## instrumentalness  3.7132997  1.161682             4.643728
## liveness          7.5477464 10.266746            13.846465
## valence           7.3661636 33.532441            36.020983
## tempo             8.4027534 32.876324            33.443795
##                  MeanDecreaseGini
## track.popularity        41.219844
```

```
## track.explicit          21.723402
## danceability            34.020828
## key                     37.256145
## loudness                29.532470
## mode                    15.334032
## speechiness             23.142324
## acousticness            31.761173
## instrumentalness         2.750786
## liveness                19.798520
## valence                 41.143108
## tempo                   38.462655
```

```
rf_tree$importance
```

```
##                           old        young MeanDecreaseAccuracy
## track.popularity 0.0205116475 0.0527427290         0.0371596379
## track.explicit   0.0403204628 0.0347164965         0.0373467459
## danceability     0.0095001035 0.0501888096         0.0305905640
## key              0.0005122247 0.0401616204         0.0210420559
## loudness         0.0308169769 0.0365875559         0.0337248980
## mode             0.0040735301 0.0235115733         0.0141087717
## speechiness      0.0079679666 0.0227406534         0.0155535272
## acousticness     0.0042688935 0.0271475209         0.0161161694
## instrumentalness 0.0011389665 0.0002532277         0.0006699091
## liveness         0.0061082189 0.0082612429         0.0072204467
## valence          0.0109830882 0.0678681232         0.0404535990
## tempo            0.0110016668 0.0502082969         0.0312828891
##                  MeanDecreaseGini
## track.popularity        41.219844
## track.explicit          21.723402
## danceability            34.020828
## key                     37.256145
## loudness                29.532470
## mode                    15.334032
## speechiness             23.142324
## acousticness            31.761173
## instrumentalness         2.750786
## liveness                19.798520
## valence                 41.143108
## tempo                   38.462655
```

```
# plot variable importance
varImpPlot(rf_tree)
```



How did Random Forest stack up against the decision tree?

With an error rate of ~.41, random forest performed *slightly* worse than our single decision tree, which yielded an error rate of ~0.38. Surprisingly, variable importance changed: track.popularity is still most important, but speechiness is relatively unimportant in the random forest model; instead, the second most important variable here is whether music is explicit.