Will My Song Be Popular?

Final Report

Sophia Cho, Masaoud Haidar, Thu Pham, Michelle Qin

December 8, 2021

Introduction and Motivation

It's December – we know what that means. It's time for our Spotify Wrapped! As we marvel over our most frequently listened-to songs, we wonder – what makes some songs popular after all? What keeps them playing over and over on our playlists, until they break the top of our Spotify Wrapped and possibly the overall World Playlist? In this project, we seek to answer this burning question, by investigating several models to predict songs' popularity based on factors like danceability, speechiness, and more, and picking a final best model.

Data and EDA

We used the "Music Genre Classification" data set from Kaggle.com. This data set comes from the August 2021 MachineHack Hackathon, which challenged participants to predict a song's genre from other predictors using machine learning. The variables (danceability, energy, key, etc.) in this data set were defined by Spotify. In this project, we used the training data from the hackathon (since the test data did not include songs' genres) to predict popularity, which is measured on a scale of 1 to 100.

We used 14 predictors from the initial data set: danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence (defined as how "happy" a song is), tempo, duration, time signature, and genre. To investigate which predictors needed to be transformed, we examined each of their distributions via histograms. (For our categorical variables, we made sure each category contained enough songs.) From the histograms (which can be seen in our EDA submission on Canvas), we decided to log-transform `speechiness` and `liveness`, as their distributions are positive and right-skewed, and sqrt-transform `duration`, as its distribution is also positive and right-skewed, but taking the log of it overcorrected and causes it to be left-skewed. Meanwhile, the `instrumentalness` column contained some NA values, so we imputed the median (which is more robust to outliers than the mean) of the non-NA values into those rows and created a new predictor `instrumentalness_is_na`, which recorded that we did this, in case the songs that contained NAs have something in common that is useful for prediction. Moreover, we chose not to standardize our data around the sample means because we did not see significant evidence of collinearity between predictors (the only two predictors that had pairwise correlation greater than 0.5 were energy and loudness, which had correlation 0.767 – see appendix for more details). Thus, going forward, we have 15 predictors. Before beginning any modeling, we hypothesize that genre will be the most important predictor, and that the random forest model will perform the best. Thus, we anticipate that are some non-linearities and non-independence in our data, which will hinder the performance of classic regression methods.

Methods (and Intermediate Results from Tuning)

We considered several different models, then chose a final best model using 10-fold cross-validation, applying each model to the the same 10 training and validation sets. To find the best version of each model before cross-validating, we tuned our random forest model and performed bidirectional variable selection prior to tuning our LASSO and ridge regression models.

The models we considered were:

- Linear regression on our 15 predictors, the baseline model
- Random forest on our 15 predictors
- Mixed effects model with a random slope for tempo, random intercept, and fixed slope for all 15 predictors, clustered by genre
- LASSO on our 15 predictors
  - Using `lambda.min`
  - Using `lambda.1se`
- Ridge regression on our 15 predictors
  - Using `lambda.min`
  - Using `lambda.1se`
- Variable selection based on Ridge regression on our 15 predictors, followed by linear regression (selecting variables with coefficients having $p < 0.05$)
- Sequential variable selection on 44 predictors: our 15 predictors, plus each of our 15 predictors squared, plus an interaction term between `genre` and each of the other 14 predictors. Followed by:
  - Linear regression
  - LASSO
    - Using `lambda.min`
    - Using `lambda.1se`
  - Ridge regression
    - Using `lambda.min`
    - Using `lambda.1se`

.

Below is a more detailed description of each model.

The following models use all predictors (no interaction or polynomial terms):

- Linear regression, the baseline model
  - We ran this model on all of our transformed variables. Almost all predictors turn out to be significant. The only ones who aren't significant are the time signatures, the keys, the tempo, and the speechiness. While this prompts doing some variable selection, we leave that to a later step. We saw from this model that the popularity of a song increases with increased danceability and loudness, and decreases when the song is more energetic, acoustic and instrumental. The popularity also slightly decreases with increased duration of the song. In addition, different genres have very different popularity. The very big effect of almost all genres prompted us to explore more on that through using mixed effects clustered by genres or through interaction terms. Some of the later models explore that.
  - The strength of this baseline model is that it is fast and easily interpretable.
  - The weakness of this model is that it may overfit, given that we are using all of the predictors. Furthermore, it does not allow for nuances like interaction terms and non-linear relationships.
- Random forest

  - We tuned this model by calculating the MSE of several random forests on the entire data set. In particular, we used several values for `mtry` and `maxnodes` and picked the model with the best `MSE`. We found that the optimal `mtry` was 4, which is low considering we have 15 predictors. This is an indicator that we have a lot of strong predictors in our dataset, which is consistent with our variable selection in other models. Because of that, the random forest model performs better when it is forced to use all of these predictors in its different trees.
  - The strength of this model is that it does not make any distributional assumptions, such as Normality, constant variance, linearity, and independence. Furthermore, random forests naturally allow for interaction effects because any predictor can have several different effects by virtue of the branches they are included in.
  - The weakness of this model is that it does not yield insight into underlying relationships between the predictors and the outcome (popularity), since its foundation is computational rather than probabilistic.

- Mixed effects model with a random slope for tempo, clustered by genre
  - Originally, we wanted to allow more predictors to have random slopes. However, we worried that this would overfit to the training data, leading to a very high mean squared error on the validation sets. We also thought this would take too long. Thus, we thought about which predictors would be most important to allow to have different beta values across genres. We also examined the correlation matrix (included in the Appendix of this report) for an idea of which predictors might be redundant with each other, finding that energy and loudness have a high correlation of 0.767. In the end, we decided on energy, valence, and tempo. We chose energy, valence (how happy the track is), and tempo for our random effects because we believe that different genres have different expectations for how energetic, happy, and fast a song should be. For instance, popular songs in the hip hop genre are probably more energetic, happier, and higher tempo than popular songs in the blues genre.
  - Next, using 10-fold cross-validation, we evaluated the MSE of models that used all possible subsets of energy, valence, and tempo as random effects. We made sure to use a different seed than our final cross-validation, so that this tuning process would be independent from determining the best overall model. We found that tempo as the only random effect was the best mixed effects model!
  - We considered using a mixed effects model that clustered by artist. This model would have had a nice interpretation because we could take the average beta values across all the artists, to get a final model that predicts a song's popularity (based on all the other predictors) for an *average* artist. However, this approach caused an overflow of memory, due to the large amount of artists, and there are a lot of artists who have only written a few songs. In fact, we ultimately decided to exclude `Artist.Name` from all models due to this issue.
  - The strength of this mixed effects model is that it has a probabilistic underpinning, allowing us to incorporate the underlying relationships between the predictors (and how they vary with other predictors) and the outcome variable. Specifically, the mixed effects model is hierarchical.
  - The weakness of the mixed model is that is prone to overfitting, even after limiting the number of predictors with random slopes. Furthermore, the model is complex, so it took quite long to run.

- LASSO and Ridge regression on our 15 predictors
  - See the next page for discussion of tuning process, pros, and cons of LASSO and Ridge.

Next, we used ridge regression for variable selection. The selected variables with p-values less than 0.05, in increasing order of their p-values, were `danceability`, `acousticness`, `instrumentalness`, `genre1`, `genre3`, `genre4`, `genre7`, `genre9`, `genre10` (tie), `genre8`, `genre2`, `genre5`, `log_liveness`, `energy`, `loudness`, `instrumentalness_is_na`, `genre6`, `mode`, `sqrt_duration`, and `key11`. We then used these variables (except `key` since only one of its categories was flagged as significant) in the following model:

- Linear regression after variable selection by ridge regression

  - The strength of this model is that it improves upon the baseline model by predicting on only significant predictors, while still being fast and easily interpretable.
  - The weakness of this model is that, like the baseline model, it does not explore interaction terms and non-linear relationships. We also tried to include quadratic and interaction terms, as we do below in sequential variable selection, but it turned out to be too complicated. As a result, we decided to exclude using ridge to select among quadratic and interaction terms, and isolated those types of terms to sequential variable selection. This decision may hurt the performance of this model compared to the sequential variable selection model described below, which does consider all of these interaction and polynomial terms. (Indeed, as can be seen in our Results section, this pans out.)

Finally, we performed sequential variable selection so that our remaining models could include some interaction and quadratic terms (but not all interaction and quadratic terms because that would introduce a lot of collinearity and slow computation). We included quadratic terms because we suspected that for some predictors, the relationship with popularity isn't linear. For example, it might be that there is some optimal loudness of the song, and the popularity of the song decreases with more or less loudness.

We first ran a linear regression model on all the interaction and quadratic terms. Then, we looked at the significance of the coefficients for all the interaction terms and found that the interaction between genre and most other variables is significant. Therefore, we performed a bidirectional variable selection procedure starting with our baseline model, with the upper scope for the final model set to to include all the interaction terms with `genre` as well as the quadratic terms for all of the predictors. The final selected variables were `danceability`, `energy`, `loudness`, `mode`, `acousticness`, `instrumentalness`, `valence`, `time_signature`, `genre`, `instrumentalness_is_na`, `log_speechiness`, `log_liveness`, `sqrt_duration`, `genre:log_speechiness`, `energy:genre`, `valence:genre`, `instrumentalness:genre`, `danceability:genre`, `mode:genre`, `loudness:genre`, `genre:sqrt_duration`, `genre:log_liveness`, `acousticness:genre`, `log_liveness^2`, `loudness^2`, `log_speechiness^2`, and `instrumentalness^2`. We then used these variables in the following models:

- Linear regression
  - The strength of this model is that, like the baseline model, it is quick and easily interpretable. Furthermore, we addressed the overfitting from the full linear model through variable selection.
  - The weakness of this model is that the model still makes distributional assumptions like normality and independence.
- LASSO (regularization)
  - We attempted to find a well-tuned LASSO regression model via `cv.glmnet`, examining the predictors selected from our bidirectional variable selection procedure.
  - We calculated that the best λ (`lambda.min`, which has the minimal MSE) was 0.000322 and the largest lambda with MSE within 1 standard error of the minimal MSE (`lambda.1se`) was 0.0337, both of which we used in our final cross-validation to compare LASSO with the other models we considered. The advantage of using `lambda.1se` in a LASSO model is that it may be less overfit than `lambda.min`, which may be too small due to chance in the cross-validation used in the tuning process.
  - A strength of this model is that it penalizes the typical SSE in linear regression by an 'absolute value' penalty term (λ is the penalty factor). In addition, it further solves for overfitting.
  - The weakness of this model is that Ridge will outperform it if an outcome is better predicted by many weak predictors.
- Ridge (regularization)
  - We attempted to find a well-tuned Ridge regression model via `cv.glmnet`, examining the predictors selected from our bidirectional variable selection procedure.

- We calculated that the best λ (`lambda.min`, which has the minimal MSE) was 0.322 and the largest lambda with MSE within 1 standard error of the minimal MSE (`lambda.1se`) was 1.18, both of which we used in our final cross-validation to compare ridge regression with the other models we considered. The advantage of using `lambda.1se` in a ridge model is that it may be less overfit than `lambda.min`, which may be too small due to chance in the cross-validation used in the tuning process.
- A strength of this model is that it penalizes the typical SSE in linear regression by a squared penalty (λ is the penalty factor). In addition, it further solves for overfitting.
- The weakness of this model is that LASSO will outperform it if an outcome is better predicted by only a few very powerful predictors.

Key R code

```
# Set up
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-2
```

```
library(lme4)
require(randomForest)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-14
```

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

```
library(Matrix)
library(ridge)

# Read in data
songs <- read.csv("data/train_transformed.csv")
songs$key <- factor(songs$key)
songs$time_signature <- factor(songs$time_signature)
songs$genre <- factor(songs$genre)
songs$X <- NULL # remove row number from variables
songs <- songs[complete.cases(songs), ]
print(dim(songs))
```

```
## [1] 17568    16
```

```
# Baseline model
baseline <- lm(popularity ~ ., data = songs)
fullmodel <- lm(popularity ~ .^2, data = songs)
X <- model.matrix(fullmodel)[,-1]

# Function to calculate MSE
mse <- function(y, yhat){
  return(mean((y-yhat)^2))
}
```

```
# Tune random forest parameter
# Setting eval=F because this chunk takes a long time to knit
set.seed(239)
all_mtry <- c(4,9,13)
```

```
all_maxnodes <- c(1000,3000,5000)
rf_tuning <- randomForest(popularity ~ ., data = songs, mtry = all_mtry[1], maxnodes = all_maxnodes[1],
mse_tuning <- mse(songs$popularity, rf_tuning$predicted)
best_mtry <- all_mtry[1]
best_maxnodes <- all_maxnodes[1]
for (i in all_mtry){
  for (j in all_maxnodes){
    rf_temp <- randomForest(popularity ~ ., data = songs, mtry = i, maxnodes = j, ntree=200)
    mse_temp <- mse(songs$popularity, rf_temp$predicted)
    if (mse_temp < mse_tuning){
      rf_tuning <- rf_temp
      mse_tuning <- mse_temp
      best_mtry <- i
      best_maxnodes <- j
    }
  }
}
```

```
best_mtry <- 4
best_maxnodes <- 3000
```

```
# Tune mixed effects model (choose predictors to have random slopes)
set.seed(239) # use a different seed than used in cross-validation

# set up 10-fold cross-validation
n <- nrow(songs)
breaks <- seq(1, n, by = floor(n/10)) # compute left-indices of folds
songs <- songs[sample(1:n), ] # shuffle rows
mse_energy = mse_valence = mse_tempo = mse_energy_valence = mse_energy_tempo = mse_valence_tempo = mse_a

# for each fold, get training and validation set
for (i in 1:10){
  if (i == 10){
    valid <- songs[breaks[i]:n, ]
    train <- songs[-(breaks[i]:n), ]
  } else{
    valid <- songs[breaks[i]:(breaks[i+1]-1), ]
    train <- songs[-(breaks[i]:(breaks[i+1]-1)), ]
  }

  # compute models
  energy_model <- lmer(popularity ~ . + ((1 + energy) | genre), data = train)
  valence_model <- lmer(popularity ~ . + ((1 + valence) | genre), data = train)
  tempo_model <- lmer(popularity ~ . + ((1 + tempo) | genre), data = train)
  energy_valence_model <- lmer(popularity ~ . + ((1 + energy + valence) | genre), data = train)
  energy_tempo_model <- lmer(popularity ~ . + ((1 + energy + tempo) | genre), data = train)
  valence_tempo_model <- lmer(popularity ~ . + ((1 + valence + tempo) | genre), data = train)
  all_model <- lmer(popularity ~ . + ((1 + energy + valence + tempo) | genre), data = train)

  # compute MSEs
  mse_energy <- mse_energy + mse(valid$popularity, predict(energy_model, newx = valid))
  mse_valence <- mse_valence + mse(valid$popularity, predict(valence_model, newx = valid))
  mse_tempo <- mse_tempo + mse(valid$popularity, predict(tempo_model, newx = valid))
  mse_energy_valence <- mse_energy_valence + mse(valid$popularity, predict(energy_valence_model, newx =
```

```
  mse_energy_tempo <- mse_energy_tempo + mse(valid$popularity, predict(energy_tempo_model, newx = valid
  mse_valence_tempo <- mse_valence_tempo + mse(valid$popularity, predict(valence_tempo_model, newx = va
  mse_all <- mse_all + mse(valid$popularity, predict(all_model, newx = valid))
}
cat(mse_energy/10, mse_valence/10, mse_tempo/10, mse_energy_valence/10, mse_energy_tempo/10, mse_valenc
```

```
# Tune ridge and lasso (without variable selection)
set.seed(239)

ridges = cv.glmnet(X, songs$popularity, alpha = 0)
lambda_min_ridge <- ridges$lambda.min
lambda_1se_ridge <- ridges$lambda.1se
cat("Best lambda value for ridge (without variable selection):", lambda_min_ridge)
cat("Largest lambda with MSE within 1 SE of minimal MSE:", lambda_1se_ridge)

lassos = cv.glmnet(X, songs$popularity, alpha = 1)
lambda_min_lasso <- lassos$lambda.min
lambda_1se_lasso <- lassos$lambda.1se
cat("Best lambda value for LASSO (without variable selection):", lambda_min_lasso)
cat("Largest lambda with MSE within 1 SE of minimal MSE:", lambda_1se_lasso)
```

```
lambda_min_ridge <- 0.3220338
lambda_1se_ridge <- 1.42681
lambda_min_lasso <- 0.02118767
lambda_1se_lasso <- 0.05371847
```

```
# Variable selection by ridge regression (selecting variables with p < 0.05)
ridge_selection <- linearRidge(popularity ~ ., songs)
```

```
# Sequential variable selection
variable_selection_model <- step(baseline, scope = list(lower = popularity ~ 1, upper = popularity ~ (.
lm_selected_variables <- lm(formula(variable_selection_model), data = songs)
selected_X = model.matrix(lm_selected_variables)[,-1]
```

```
# This chunk accomplishes the same thing as the previous chunk but runs faster
variable_selection_model <- lm(popularity ~ danceability + energy + loudness + mode + acousticness + ins
lm_selected_variables <- lm(formula(variable_selection_model), data = songs)
selected_X = model.matrix(lm_selected_variables)[,-1]
```

```
# Tune ridge and lasso models (based on sequential variable selection)
set.seed(239)
selected_ridges = cv.glmnet(selected_X, songs$popularity, alpha = 0)
selected_lambda_min_ridge <- selected_ridges$lambda.min
selected_lambda_1se_ridge <- selected_ridges$lambda.1se
cat("Best lambda value for ridge (after variable selection):", selected_lambda_min_ridge)
cat("Largest lambda with MSE within 1 SE of minimal MSE:", selected_lambda_1se_ridge)

selected_lassos = cv.glmnet(selected_X, songs$popularity, alpha = 1)
selected_lambda_min_lasso <- selected_lassos$lambda.min
selected_lambda_1se_lasso <- selected_lassos$lambda.1se
cat("Best lambda value for LASSO (after variable selection):", selected_lambda_min_lasso)
cat("Largest lambda with MSE within 1 SE of minimal MSE:", selected_lambda_1se_lasso)
```

```
selected_lambda_min_ridge <- 0.3220338
selected_lambda_1se_ridge <- 1.184563
```

```r
selected_lambda_min_lasso <- 0.0003220338
selected_lambda_1se_lasso <- 0.03373677
```

```r
set.seed(139)

# Set up 10-fold cross-validation to choose final best model!!!
n <- nrow(songs)
breaks <- seq(1, n, by = floor(n/10)) # compute left-indices of folds
songs <- songs[sample(1:n), ] # shuffle rows
mse_baseline = mse_lasso = mse_ridge = mse_lasso_1se = mse_ridge_1se = mse_selected = mse_sequential = m

# For each fold, get training and validation set and compute models and MSEs
for (i in 1:10){
  if (i == 10){
    valid <- songs[breaks[i]:n, ]
    train <- songs[-(breaks[i]:n), ]
  } else{
    valid <- songs[breaks[i]:(breaks[i+1]-1), ]
    train <- songs[-(breaks[i]:(breaks[i+1]-1)), ]
  }

  # Baseline Model
  baseline_model <- lm(data = train, popularity ~ .)
  mse_baseline <- mse_baseline + mse(valid$popularity, predict(baseline_model, new = valid))

  # LASSO and Ridge Regression
  if (i == 10){
    X_valid <- X[breaks[i]:n, ]
    X_train <- X[-(breaks[i]:n), ]
  } else{
    X_valid <- X[breaks[i]:(breaks[i+1]-1), ]
    X_train <- X[-(breaks[i]:(breaks[i+1]-1)), ]
  }

  lasso_model <- glmnet(X_train, train$popularity, lambda = lambda_min_lasso, alpha = 1)
  mse_lasso <- mse_lasso + mse(valid$popularity, predict(lasso_model, newx = X_valid))
  ridge_model <- glmnet(X_train, train$popularity, lambda = lambda_min_ridge, alpha = 0)
  mse_ridge <- mse_ridge + mse(valid$popularity, predict(ridge_model, newx = X_valid))

  lasso_1se_model <- glmnet(X_train, train$popularity, lambda = lambda_1se_lasso, alpha = 1)
  mse_lasso_1se <- mse_lasso_1se + mse(valid$popularity, predict(lasso_1se_model, newx = X_valid))
  ridge_1se_model <- glmnet(X_train, train$popularity, lambda = lambda_1se_ridge, alpha = 0)
  mse_ridge_1se <- mse_ridge_1se + mse(valid$popularity, predict(ridge_1se_model, newx = X_valid))

  # Linear Regression after Ridge Variable Selection
  selected_model <- lm(popularity ~ danceability + acousticness + instrumentalness + genre + log_livenes
  mse_selected <- mse_selected + mse(valid$popularity, predict(selected_model, new = valid))

  # Linear Regression after Sequential Variable Selection
  sequential_model <- lm(formula(variable_selection_model), data = train)
  mse_sequential <- mse_sequential + mse(valid$popularity, predict(sequential_model, new = valid))

  # LASSO and Ridge Regression after Sequential Variable Selection
  if (i == 10){
```

```
    selected_X_valid <- selected_X[breaks[i]:n, ]
    selected_X_train <- selected_X[-(breaks[i]:n), ]
  } else{
    selected_X_valid <- selected_X[breaks[i]:(breaks[i+1]-1), ]
    selected_X_train <- selected_X[-(breaks[i]:(breaks[i+1]-1)), ]
  }

  selected_lasso_model <- glmnet(selected_X_train, train$popularity, lambda = selected_lambda_min_lasso
  mse_selected_lasso <- mse_selected_lasso + mse(valid$popularity, predict(selected_lasso_model, newx =
  selected_ridge_model <- glmnet(selected_X_train, train$popularity, lambda = selected_lambda_min_ridge
  mse_selected_ridge <- mse_ridge + mse(valid$popularity, predict(selected_ridge_model, newx = selected_

  selected_lasso_1se_model <- glmnet(X_train, train$popularity, lambda = selected_lambda_1se_lasso, alp
  mse_selected_lasso_1se <- mse_selected_lasso_1se + mse(valid$popularity, predict(selected_lasso_1se_m
  selected_ridge_1se_model <- glmnet(X_train, train$popularity, lambda = selected_lambda_1se_ridge, alp
  mse_selected_ridge_1se <- mse_ridge_1se + mse(valid$popularity, predict(selected_ridge_1se_model, new

  # Mixed Effects Model
  mixed_model <- lmer(popularity ~ . + ((1 + tempo) | genre), data = train)
  mse_mixed <- mse_mixed + mse(valid$popularity, predict(mixed_model, newx = valid))

  # Random Forest Model
  rf_model <- randomForest(popularity ~ ., data = train, mtry = best_mtry, maxnodes = best_maxnodes, nt
  mse_rf <- mse_rf + mse(valid$popularity, predict(rf_model, new = valid))
}
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

Results

Ranking of Models

```
##         MSE                              Model
## 13 226.6894                      Random Forest
## 7  247.1994     Linear selected by Sequential
## 1  259.0782                           Baseline
## 6  259.6728          Linear selected by Ridge
## 4  305.5480                         LASSO 1 SE
## 8  306.0890      LASSO selected by Sequential
## 10 306.8659 LASSO 1 SE selected by Sequential
## 2  308.1942                              LASSO
## 5  308.1953                         Ridge 1 SE
## 3  310.7160                              Ridge
## 11 338.2330 Ridge 1 SE selected by Sequential
## 9  340.1750      Ridge selected by Sequential
## 12 349.0129                      Mixed Effects
```

We found random forest to be the best model with an average MSE of 227 across 10 folds (see the above code for MSE values of the other models). A random forest model has no formal assumptions, so we did not explore those further. On the other hand, we found the mixed effects model to perform the worst, with an average MSE of of 349 across 10 folds. Several models ranked in the middle perform similarly, which is

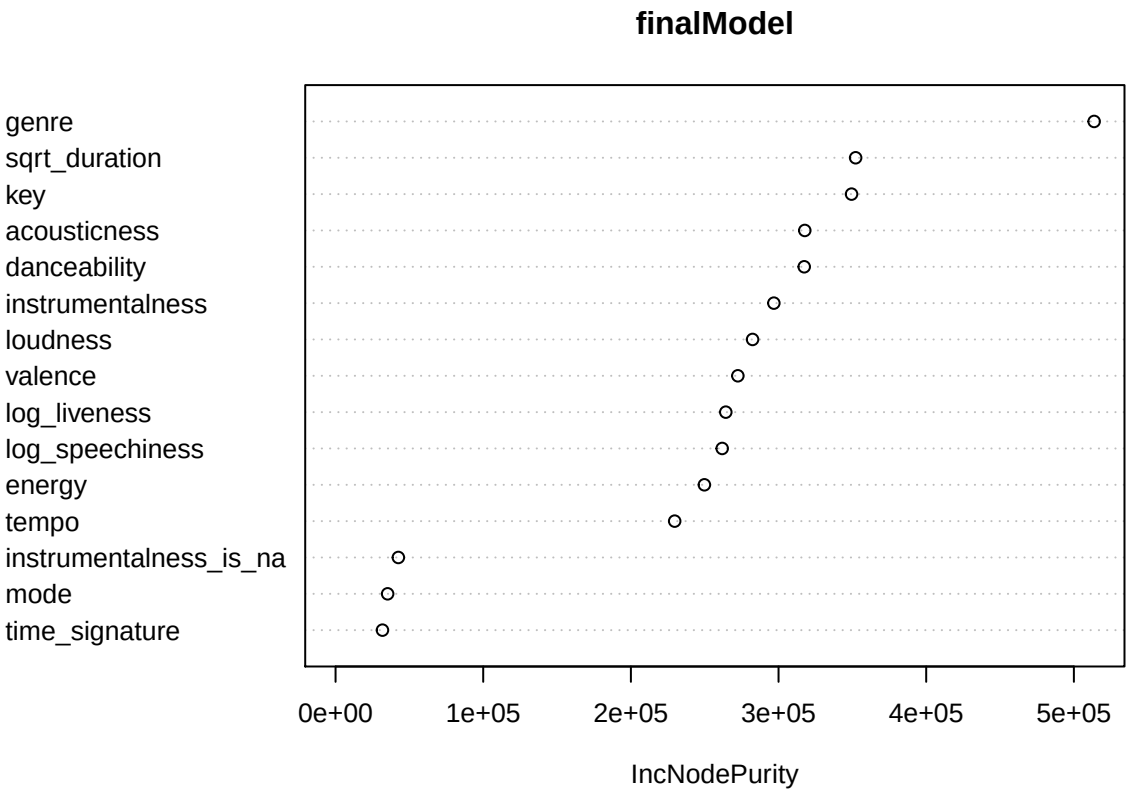
discussed further in the next section.

Our final model, fitted to the entire data set, is computed below.

```
finalModel <- randomForest(popularity ~ ., data = songs, mtry = best_mtry, maxnodes = best_maxnodes, nti
```

Variable Importance

```
varImpPlot(finalModel, cex = 0.8)
```



One useful result of our random forest model is the variable importance graph, which shows that `genre`, `key`, and `sqrt_duration` are the most important predictors. Furthermore, genre is far more important than the second most important predictor, the key of the song.

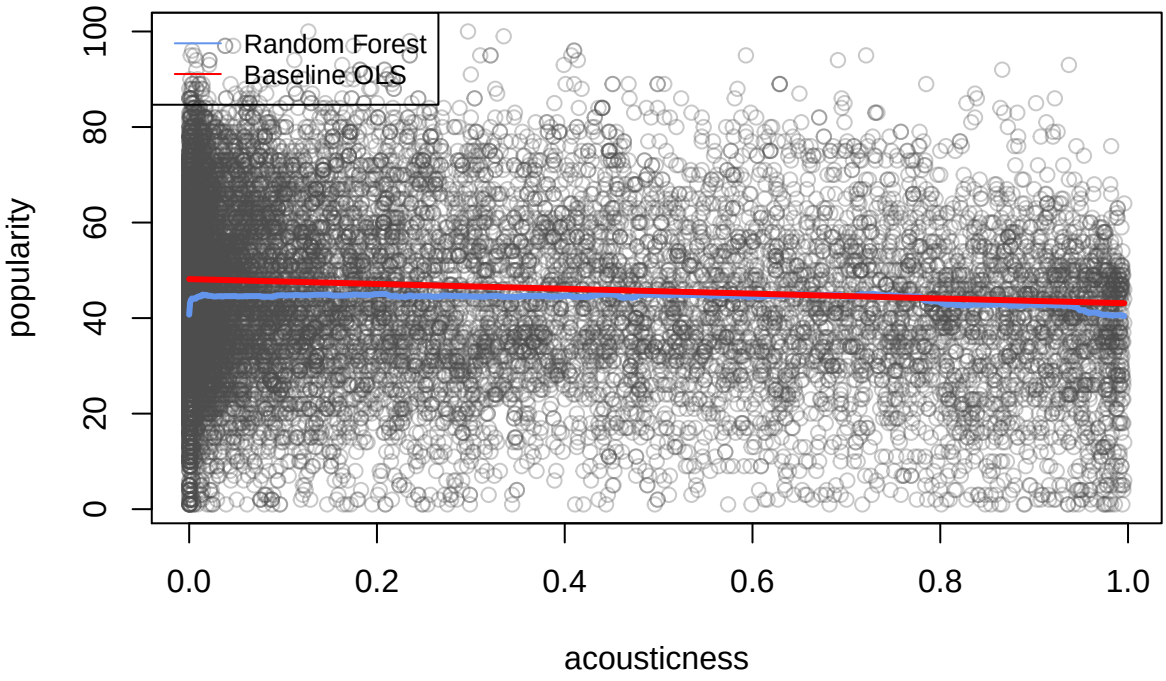
Two of our other models looked for the most important variables as well: (i) variable selection by ridge regression and (ii) bidirectional sequential variable selection. Both of these models found that `genre`, `instrumentalness`, and `danceability` are the most important predictors. For model (i), we define the most important predictors as the ones whose beta coefficient estimates have the most significant p-values. For model (ii), we define the most important predictors as the least likely predictors to be removed in the last step in the trace (i.e. they had the most impact on the AIC score). Neither of these models found `key` to be an important predictor.

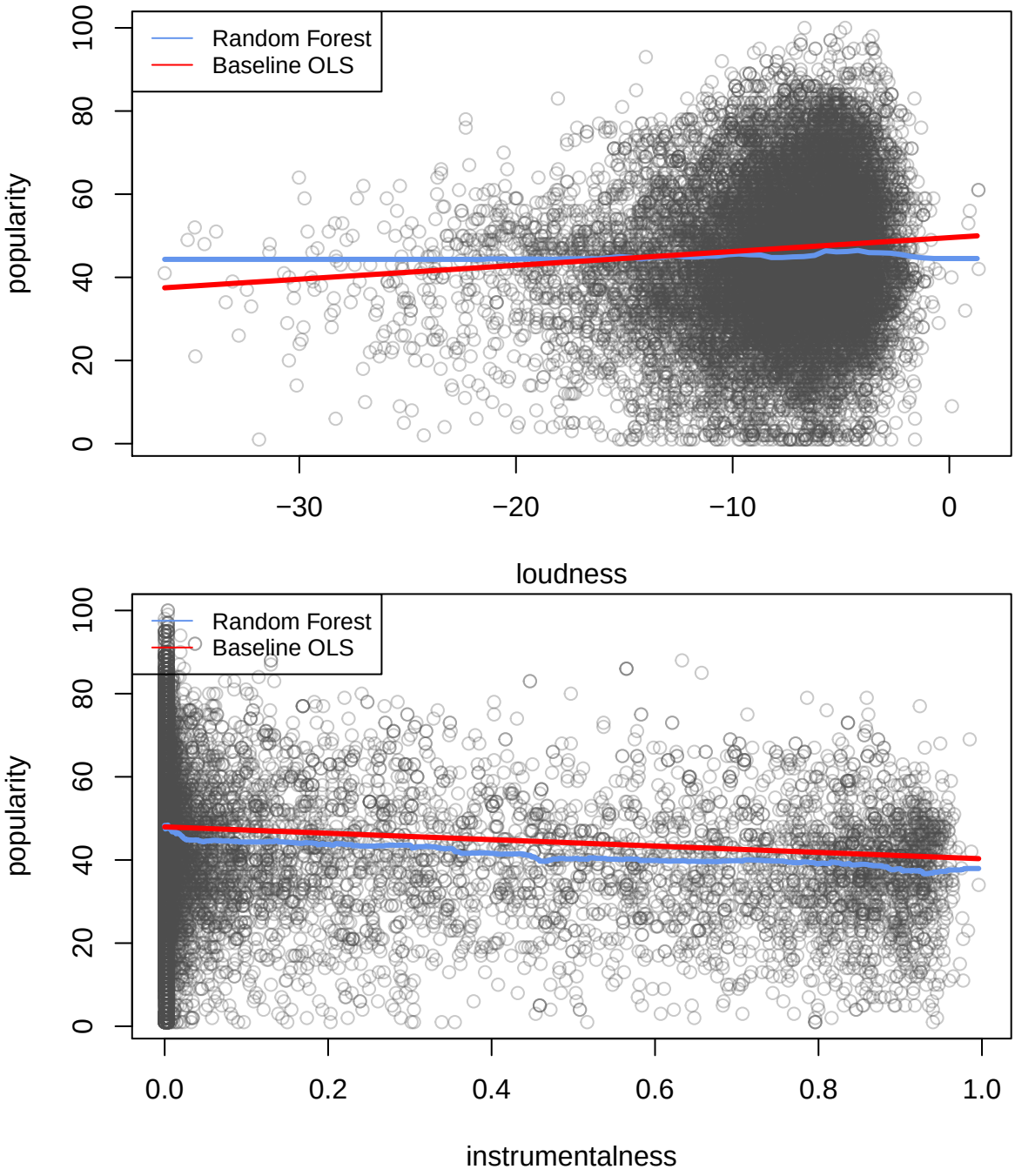
Prediction Plots

A random forest model is especially powerful because it allows for non-linear relationships. Below, we show prediction plots for three predictors that appear to have non-linear relationships with popularity: `acousticness`, `loudness`, and `instrumentalness`. (Moreover, random forest ranks these three predictors as in the top seven most important predictors and `acousticness` as the second most important quantitative predictor.)

Each plot include a regular scatter of popularity vs. that predictor, the baseline OLS predictions, and the random forest predictions. To plot the OLS and random forest predictions, we hold all other variables but the predictor in question (acousticness, loudness, or instrumentalness) constant at the mean for quantitative variables and the mode for categorical variables.

The prediction plot of popularity vs. acousticness shows that the random forest and linear models are generally the same, except for low values of acousticness, where the random forest predictions begin lower and briefly increase in a non-linear fashion, before following the same general pattern as the baseline OLS prediction. Although this deviation looks brief on the graph, it is important to notice the high density of points behind the prediction curves. Because there is so much data that is accounted for in this more noticeably non-linear part of the random forest predictions, it is possible that the random forest especially outperforms the baseline OLS model in this segment of the data. The popularity vs. loudness plot has a similar trend, where the random forest model allows for more subtlety and flexibility from loudness values of -15 and onwards, as it both increases and decreases several times to better reflect the data. Finally, the random forest model in the popularity vs. instrumentalness prediction plot looks generally linear (if we were to smooth out the curve a little more) and does not have as a noticeable difference in inflection points as the popularity vs. acousticness, for example.





Discussion

Ranking of Models

We were not surprised that random forest is the best model because the model, in general, allows for a lot of flexibility both in the relaxation of assumptions and the incorporation of non-linear relationships and interaction terms. As a result, it is able to account for more complexities in the data set, and our tuning of the forest has also accounted for overfitting and other potential wrenches in the prediction, at least to some degree.

After the random forest model, the next best models are the linear model with sequential variable selection

and baseline model, respectively. Because none of the LASSO and Ridge models do not outperform our baseline model, we conclude that the baseline model and the linear regression after variable selection did not need regularization as they were not overfitting. We note here that we only have 15 predictors and we have more than 17,000 data points, so a simple model like linear regression on the main terms will likely underfit, and that is why using regularization like LASSO or Ridge will not improve the model. Comparing between the different versions of the LASSO and Ridge models, we see that all of them perform very similarly. So, they aren't affected a lot by increasing their parameter. This is again as a result of the models not being overfit from the first place.

We were also initially surprised that the mixed model performs the worst. We had originally expected the predictor that we had chosen, tempo, would vary so much by genre that the mixed model would account for complexities that the other models would not. It is possible that the model is so complex that it overfits to the train data, as we did not address overfitting for this model as we did with previous ones. Furthermore, it is possible that tempo simply is not the right predictor to have a random slope. For example, tempo was not flagged as a significant predictor by any of our variable selection procedures. Intuitively, we can also say that perhaps tempo doesn't vary significantly enough across genre that it would warrant a different beta value for each genre (for example, slow pop songs can be as equally likely to exist as fast country songs).

Variable Importance

Within our final random forest model, we found that genre was the most important variable by far. This is not surprising, as pop music tends to populate Top-Ten charts these days. In second, we had key, which makes sense to a certain extent as many pop songs are either in C or G (they have minimal sharps and flats, making them easy to play on the piano and guitar). Last but not least, in third, we had duration, which also makes sense as most pop songs are 3 to 4 minutes long.

It makes sense that the significance and the AIC score will result in the same variable importance, as they are both based on linear models. On the other hand, it is interesting that the random forest finds important patterns in the keys of the songs while the other two don't. This is likely because the key variable has some interactions with some other variables, possible including some non-linear patterns with other predictors, which the random forest can recognize but the linear regression can't.

Prediction Plots

As discussed in the previous section, the random forest model seems to be able to account for some non-linear nuances of the data. Overall, however, the random forest and baseline OLS predictions don't show any drastic differences on this dataset – which makes sense given that the MSE of random forest (227, which corresponds to a RMSE of 15.1 popularity points) and the MSE of the baseline OLS (259, which corresponds to a RMSE of 16.1 popularity points) are similar.

Conclusion

One might say our project is a testament to the capabilities of computational methods like random forest over probabilistic methods like mixed effects and ordinary least squares (our baseline model).

However, even though random forest was the best model, the majority of this project was dedicated to experimenting with different ways to improve linear regression. We considered multiple types of interaction terms (including quadratic predictors and interaction terms between `genre` and other predictors), multiple ways to select variables, and combinations of models.

Appendix

The following are some outputs that we cited but did not include explicitly in the Methods and Results sections above.

```r
# Correlation between quantitative variables
quant_var <- c("danceability", "energy", "loudness", "acousticness", "instrumentalness", "valence", "ten
cor(subset(songs, select = quant_var))
```

```
##                  danceability     energy    loudness acousticness
## danceability       1.00000000 -0.1016630  0.05158621   0.01411778
## energy            -0.10166301  1.0000000  0.76682827  -0.74359347
## loudness           0.05158621  0.7668283  1.00000000  -0.61033529
## acousticness       0.01411778 -0.7435935 -0.61033529   1.00000000
## instrumentalness  -0.21885873 -0.1507426 -0.34630358   0.14530198
## valence            0.44304454  0.2209319  0.17797841  -0.12167458
## tempo             -0.18664312  0.2094784  0.16169468  -0.16730079
## log_speechiness    0.15666785  0.2484125  0.17003994  -0.15536198
## log_liveness      -0.12333104  0.1829465  0.11680769  -0.10027005
## sqrt_duration     -0.05311034  0.3073248  0.26074032  -0.38737237
##                  instrumentalness      valence       tempo log_speechiness
## danceability          -0.21885873  0.443044543 -0.18664312      0.15666785
## energy                -0.15074260  0.220931930  0.20947845      0.24841252
## loudness              -0.34630358  0.177978414  0.16169468      0.17003994
## acousticness           0.14530198 -0.121674585 -0.16730079     -0.15536198
## instrumentalness       1.00000000 -0.225516735 -0.02590325     -0.09209886
## valence               -0.22551674  1.000000000  0.04883102      0.04620928
## tempo                 -0.02590325  0.048831018  1.00000000      0.09889176
## log_speechiness       -0.09209886  0.046209284  0.09889176      1.00000000
## log_liveness          -0.03873007 -0.009559366  0.03179805      0.09124072
## sqrt_duration         -0.03360184 -0.040598164  0.06698918      0.08411349
##                  log_liveness sqrt_duration
## danceability     -0.123331036   -0.05311034
## energy            0.182946485    0.30732481
## loudness          0.116807685    0.26074032
## acousticness     -0.100270047   -0.38737237
## instrumentalness -0.038730072   -0.03360184
## valence          -0.009559366   -0.04059816
## tempo             0.031798050    0.06698918
## log_speechiness   0.091240724    0.08411349
## log_liveness      1.000000000    0.04319249
## sqrt_duration     0.043192492    1.00000000
```

```r
# Baseline model
summary(baseline)
```

```
##
## Call:
## lm(formula = popularity ~ ., data = songs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.874 -10.742  -0.135  10.816  49.723
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               36.670550   2.422136  15.140  < 2e-16 ***
## danceability              11.306222   0.991998  11.397  < 2e-16 ***
## energy                    -7.248907   1.150321  -6.302 3.02e-10 ***
## key1                       0.651076   0.544692   1.195 0.231982
```

```
## key2                         -0.121076   0.516451  -0.234 0.814648
## key3                         -0.974983   0.814883  -1.196 0.231529
## key4                          0.304591   0.556092   0.548 0.583881
## key5                         -0.295307   0.571367  -0.517 0.605272
## key6                          0.654959   0.602918   1.086 0.277355
## key7                         -0.172880   0.508586  -0.340 0.733918
## key8                          0.295284   0.609118   0.485 0.627844
## key9                          0.761311   0.519205   1.466 0.142584
## key10                         0.308438   0.636450   0.485 0.627950
## key11                         1.261000   0.568570   2.218 0.026578 *
## loudness                      0.333079   0.054605   6.100 1.08e-09 ***
## mode                          1.020751   0.266798   3.826 0.000131 ***
## acousticness                 -5.071175   0.639462  -7.930 2.32e-15 ***
## instrumentalness             -7.672435   0.534788 -14.347  < 2e-16 ***
## valence                      -0.597872   0.650882  -0.919 0.358341
## tempo                         0.002478   0.004346   0.570 0.568527
## time_signature3              -0.073744   1.593178  -0.046 0.963082
## time_signature4               1.650544   1.532198   1.077 0.281388
## time_signature5              -1.321733   1.906241  -0.693 0.488086
## genre1                       10.830216   1.021319  10.604  < 2e-16 ***
## genre2                       -1.964424   1.040067  -1.889 0.058942 .
## genre3                       -9.240863   1.091839  -8.464  < 2e-16 ***
## genre4                       16.511361   1.104167  14.954  < 2e-16 ***
## genre5                        9.227066   1.080275   8.541  < 2e-16 ***
## genre6                        6.838012   0.973683   7.023 2.25e-12 ***
## genre7                       12.669804   1.063153  11.917  < 2e-16 ***
## genre8                        9.396037   1.055295   8.904  < 2e-16 ***
## genre9                       11.794423   0.929370  12.691  < 2e-16 ***
## genre10                      11.443511   0.955891  11.972  < 2e-16 ***
## instrumentalness_is_naTRUE    1.525073   0.322325   4.731 2.25e-06 ***
## log_speechiness              -0.001682   0.221114  -0.008 0.993929
## log_liveness                 -1.369174   0.187424  -7.305 2.89e-13 ***
## sqrt_duration                -0.006503   0.001234  -5.272 1.37e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.08 on 17531 degrees of freedom
## Multiple R-squared:  0.1505, Adjusted R-squared:  0.1487
## F-statistic: 86.25 on 36 and 17531 DF,  p-value: < 2.2e-16
```

```r
# Variable selection by ridge regression
summary(ridge_selection)
```

```
##
## Call:
## linearRidge(formula = popularity ~ ., data = songs)
##
##
## Coefficients:
##                               Estimate Scaled estimate Std. Error (scaled)
## (Intercept)                  3.836e+01              NA                   NA
## danceability                 1.135e+01       2.505e+02            2.109e+01
## energy                      -6.405e+00      -1.989e+02            3.240e+01
## key1                         6.059e-01       2.337e+01            1.977e+01
## key2                        -1.826e-01      -7.577e+00            2.011e+01
```

```
## key3                        -1.033e+00      -2.260e+01          1.722e+01
## key4                         2.438e-01       9.070e+00          1.950e+01
## key5                        -3.629e-01      -1.287e+01          1.915e+01
## key6                         5.929e-01       1.956e+01          1.886e+01
## key7                        -2.495e-01      -1.064e+01          2.031e+01
## key8                         2.526e-01       8.090e+00          1.855e+01
## key9                         7.112e-01       2.938e+01          2.011e+01
## key10                        2.371e-01       7.231e+00          1.847e+01
## key11                        1.172e+00       4.267e+01          1.951e+01
## loudness                     3.107e-01       1.652e+02          2.703e+01
## mode                         1.002e+00       6.392e+01          1.672e+01
## acousticness                -5.052e+00      -2.072e+02          2.483e+01
## instrumentalness            -7.564e+00      -2.752e+02          1.895e+01
## valence                     -4.778e-01      -1.519e+01          1.991e+01
## tempo                        2.244e-03       8.792e+00          1.673e+01
## time_signature3             -2.111e-01      -7.020e+00          3.994e+01
## time_signature4              1.541e+00       5.699e+01          4.254e+01
## time_signature5             -1.483e+00      -2.075e+01          2.215e+01
## genre1                       7.220e+00       2.544e+02          2.461e+01
## genre2                      -5.407e+00      -1.847e+02          2.463e+01
## genre3                      -1.122e+01      -2.186e+02          1.913e+01
## genre4                       1.416e+01       2.684e+02          1.874e+01
## genre5                       5.629e+00       2.037e+02          2.734e+01
## genre6                       3.293e+00       1.536e+02          2.954e+01
## genre7                       1.053e+01       2.425e+02          2.217e+01
## genre8                       5.582e+00       2.256e+02          2.862e+01
## genre9                       8.453e+00       3.889e+02          2.844e+01
## genre10                      7.831e+00       4.644e+02          3.570e+01
## instrumentalness_is_naTRUE   1.643e+00       9.333e+01          1.789e+01
## log_speechiness              4.196e-02       3.849e+00          1.959e+01
## log_liveness                -1.342e+00      -1.189e+02          1.634e+01
## sqrt_duration               -3.670e-03      -8.627e+01          2.428e+01
##                            t value (scaled) Pr(>|t|)
## (Intercept)                              NA       NA
## danceability                         11.882  < 2e-16 ***
## energy                                6.139 8.30e-10 ***
## key1                                  1.182 0.237042
## key2                                  0.377 0.706314
## key3                                  1.312 0.189367
## key4                                  0.465 0.641772
## key5                                  0.672 0.501598
## key6                                  1.037 0.299631
## key7                                  0.524 0.600230
## key8                                  0.436 0.662757
## key9                                  1.461 0.143992
## key10                                 0.391 0.695461
## key11                                 2.187 0.028771 *
## loudness                              6.112 9.87e-10 ***
## mode                                  3.824 0.000132 ***
## acousticness                          8.343  < 2e-16 ***
## instrumentalness                     14.521  < 2e-16 ***
## valence                               0.763 0.445470
## tempo                                 0.525 0.599266
## time_signature3                       0.176 0.860488
```

```
## time_signature4                        1.340 0.180325
## time_signature5                        0.937 0.348866
## genre1                                10.333  < 2e-16 ***
## genre2                                 7.501 6.33e-14 ***
## genre3                                11.428  < 2e-16 ***
## genre4                                14.323  < 2e-16 ***
## genre5                                 7.452 9.19e-14 ***
## genre6                                 5.200 2.00e-07 ***
## genre7                                10.936  < 2e-16 ***
## genre8                                 7.881 3.33e-15 ***
## genre9                                13.672  < 2e-16 ***
## genre10                               13.008  < 2e-16 ***
## instrumentalness_is_naTRUE             5.216 1.83e-07 ***
## log_speechiness                        0.196 0.844253
## log_liveness                           7.275 3.46e-13 ***
## sqrt_duration                          3.554 0.000380 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Ridge parameter: 0.01410054, chosen automatically, computed using 25 PCs
##
## Degrees of freedom: model 34.62 , variance 33.5 , residual 35.74
```

```
errors_df <- data.frame(MSE = c(mse_baseline, mse_lasso, mse_ridge, mse_lasso_1se, mse_ridge_1se, mse_s
errors_df$Model <- c("Baseline", "LASSO", "Ridge", "LASSO 1 SE", "Ridge 1 SE", "Linear selected by Ridg
errors_df <- errors_df[order(errors_df$MSE), ]
errors_df
```

```
# Prediction plots for three predictors: acousticness, loudness, and instrumentalness
baselineModel <- lm(popularity ~ ., data = songs)

getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

dummy_vals <- seq(min(songs$acousticness), max(songs$acousticness), by = 0.001)
dummy_loudness <- seq(min(songs$loudness), max(songs$loudness), by = 0.5)

dummy_df = list("key" = getmode(songs$key), "acousticness" = mean(songs$acousticness),
                "tempo" = mean(songs$tempo), "instrumentalness_is_na" =
                  getmode(songs$instrumentalness_is_na), "sqrt_duration" =
                  mean(songs$sqrt_duration), "danceability" = mean(songs$danceability),
                "loudness" = mean(songs$loudness), "instrumentalness" =
                  mean(songs$instrumentalness), "time_signature" = getmode(songs$time_signature),
                "log_speechiness" = mean(songs$log_speechiness), "energy" = mean(songs$energy),
                "mode" = getmode(songs$mode), "valence" = mean(songs$valence),
                "genre" = getmode(songs$genre), "log_liveness" = mean(songs$log_liveness))

dummy_df = data.frame(dummy_df)

dummy_df_acousticness = dummy_df[rep(seq_len(nrow(dummy_df)), each = length(dummy_vals)), ]
dummy_df_loudness = dummy_df[rep(seq_len(nrow(dummy_df)), each = length(dummy_loudness)), ]
dummy_df_instrumentalness = dummy_df[rep(seq_len(nrow(dummy_df)), each = length(dummy_vals)), ]
```

```r
dummy_df_acousticness$acousticness = dummy_vals
dummy_df_loudness$loudness = dummy_loudness
dummy_df_instrumentalness$instrumentalness = dummy_vals

plot(popularity ~ acousticness, data = songs, col = rgb(0.3, 0.3, 0.3, 0.3))
lines(predict(finalModel, new = dummy_df_acousticness) ~ dummy_vals,
      col = "cornflower blue", lwd=3)
lines(predict(baselineModel, newdata = dummy_df_acousticness) ~ dummy_vals,
      col = "red", lwd=3)
legend("topleft", legend=c("Random Forest", "Baseline OLS"), col=c("cornflower blue", "red"), lty = 1:1

plot(popularity ~ loudness, data = songs, col = rgb(0.3, 0.3, 0.3, 0.3))
lines(predict(finalModel, new = dummy_df_loudness) ~ dummy_loudness,
      col = "cornflower blue", lwd=3)
lines(predict(baselineModel, newdata = dummy_df_loudness) ~ dummy_loudness,
      col = "red", lwd=3)
legend("topleft", legend=c("Random Forest", "Baseline OLS"), col=c("cornflower blue", "red"), lty = 1:1

plot(popularity ~ instrumentalness, data = songs, col = rgb(0.3, 0.3, 0.3, 0.3))
lines(predict(finalModel, new = dummy_df_instrumentalness) ~ dummy_vals,
      col = "cornflower blue", lwd=3)
lines(predict(baselineModel, newdata = dummy_df_instrumentalness) ~ dummy_vals,
      col = "red", lwd=3)
legend("topleft", legend=c("Random Forest", "Baseline OLS"), col=c("cornflower blue", "red"), lty = 1:1
```