R Project: Spotify Song Data

Tessa Mitchell

Data Cleaning

The data used is from Spotify and can be found at https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge/dataset_files. It is the first file in the Track_Features.tar.gz folder. This dataset is one of many data sets released by Spotify as part of a challenge to create models that predict whether a user will skip a song or not. The data used in this report, however, it not about skipped songs; it is simply data describing songs in the Spotify library.

```
df <- read.csv("spotifySongs1.csv")
nrow(df)
```

Each row represents an individual song in the Spotify library. There are 1,853,311 rows. The data attributes are as follows, and the meanings are described Data Description pdf found on the linked page above.

```
names(df)
```

Using this data, I will try to predict danceability - a measure of how good a songs is for dancing - using the other attributes of the song.

I discovered in my initial exploration that there are some songs where all several variables including beat_strength, danceability, bounciness, and tempo are all 0. These are outliers, since a tempo of 0 does not make logical sense unless the track is a podcast or some other form of media. Therefore, I will remove these instances from the data set.

```
df1 <- df[df$tempo != 0, ]
```

Since there are over 1.8 million rows in this csv file, my computer runs out of RAM while trying to process the data, despite having 16 gigs of RAM. This is a common issue with R that I and many others have run in to before. Therefore, I will take a subset of this data - a random sampling of 100000 rows - as shown below. I will also remove the first attribute, track_id, since it is an arbitrary unique identifier and is therefore not helpful for the following analysis.

```
set.seed(1234)
i <- sample(1:nrow(df1), 100000, replace = FALSE)
df2 <- df1[i, 2:30]
write.csv(df2, "spotifySongs2.csv")
```

The above code chunks were not run in the knitting process, since it runs out of ram before even finishing those chunks. Therefore, the above code chunks were run beforehand, and the csv file saved above is read below as part of the knitting process.

```
df2 <- read.csv("spotifySongs2.csv")
```

```
df3 <- df2[,2:30]
df3$mode <- factor(df3$mode) # changes character to factor
```

There will also be more data cleaning done throughout the project for individual models.

Data Exploration

What does the data look like?

```
str(df3)
```

```
## 'data.frame':    100000 obs. of  29 variables:
##  $ duration              : num  141 447 251 179 217 ...
##  $ release_year          : int  2016 2016 2014 2018 2014 2016 2014 2003 2017 2013 ...
##  $ us_popularity_estimate: num  91.5 99.7 98.9 99.8 99 ...
##  $ acousticness          : num  0.852 0.8394 0.1006 0.1088 0.0489 ...
##  $ beat_strength         : num  0.503 0.576 0.461 0.609 0.595 ...
##  $ bounciness            : num  0.479 0.691 0.515 0.664 0.655 ...
##  $ danceability          : num  0.596 0.623 0.66 0.739 0.798 ...
##  $ dyn_range_mean        : num  7.18 10.99 8.22 9.91 9.85 ...
##  $ energy                : num  0.349 0.699 0.688 0.69 0.451 ...
##  $ flatness              : num  1.026 0.881 1.011 1.019 1.034 ...
##  $ instrumentalness      : num  5.66e-08 3.52e-08 9.56e-10 2.27e-08 1.01e-06 ...
##  $ key                   : int  11 10 1 5 1 6 0 9 1 5 ...
##  $ liveness              : num  0.1162 0.8806 0.2424 0.0619 0.1681 ...
##  $ loudness              : num  -12.68 -11.35 -5.28 -5.3 -9.09 ...
##  $ mechanism             : num  0.376 0.146 0.464 0.383 0.588 ...
##  $ mode                  : Factor w/ 2 levels "major","minor": 2 2 2 2 1 2 2 2 1 2 ...
##  $ organism              : num  0.747 0.847 0.386 0.443 0.293 ...
##  $ speechiness           : num  0.0456 0.9465 0.0679 0.2374 0.1216 ...
##  $ tempo                 : num  84 79.8 132 96 123.9 ...
##  $ time_signature        : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ valence               : num  0.904 0.324 0.325 0.634 0.219 ...
##  $ acoustic_vector_0     : num  0.0797 -0.1906 -0.8336 -0.8283 -0.7907 ...
##  $ acoustic_vector_1     : num  0.3351 -0.0352 0.3566 0.3661 0.2353 ...
##  $ acoustic_vector_2     : num  0.115 -0.297 0.257 0.243 0.159 ...
##  $ acoustic_vector_3     : num  -0.5328 0.2483 0.034 0.0217 0.1567 ...
##  $ acoustic_vector_4     : num  -0.1278 -0.0706 -0.2849 -0.4297 -0.2289 ...
##  $ acoustic_vector_5     : num  0.2321 -0.53707 0.03852 -0.00268 -0.08487 ...
##  $ acoustic_vector_6     : num  -0.223 -0.212 -0.47 -0.447 -0.496 ...
##  $ acoustic_vector_7     : num  -0.067 -0.5041 0.2231 0.228 0.0956 ...
```

Above is a summary of the data found using the str() function. We can see that all our variables are numeric or integer with the exception of "mode", which describes whether a song is in a major or minor key.

```
summary(df3)
```

```
##     duration        release_year  us_popularity_estimate  acousticness
##  Min.   :  30.01   Min.   :1950   Min.   : 90.00         Min.   :0.00000
##  1st Qu.: 176.89   1st Qu.:2006   1st Qu.: 93.39         1st Qu.:0.02672
##  Median : 217.96   Median :2013   Median : 95.76         Median :0.21937
##  Mean   : 233.03   Mean   :2009   Mean   : 95.58         Mean   :0.34696
##  3rd Qu.: 267.33   3rd Qu.:2017   3rd Qu.: 97.93         3rd Qu.:0.65417
##  Max.   :1799.82   Max.   :2018   Max.   :100.00         Max.   :0.99580
##  beat_strength       bounciness       danceability     dyn_range_mean
##  Min.   :0.02453   Min.   :0.03136   Min.   :0.05627   Min.   : 2.348
##  1st Qu.:0.33165   1st Qu.:0.32074   1st Qu.:0.43304   1st Qu.: 5.829
##  Median :0.45869   Median :0.47896   Median :0.57228   Median : 7.515
##  Mean   :0.46204   Mean   :0.47928   Mean   :0.55687   Mean   : 7.847
##  3rd Qu.:0.58561   3rd Qu.:0.63357   3rd Qu.:0.69712   3rd Qu.: 9.462
```

```
##  Max.   :0.99782   Max.   :0.97655   Max.   :0.98843   Max.   :33.292
##      energy             flatness      instrumentalness         key
##  Min.   :0.0000193   Min.   :0.2565   Min.   :0.0000000   Min.   : 0.000
##  1st Qu.:0.4102103   1st Qu.:0.9692   1st Qu.:0.0000002   1st Qu.: 2.000
##  Median :0.6256604   Median :1.0026   Median :0.0002104   Median : 5.000
##  Mean   :0.5925617   Mean   :0.9939   Mean   :0.2067270   Mean   : 5.288
##  3rd Qu.:0.8093400   3rd Qu.:1.0291   3rd Qu.:0.3112661   3rd Qu.: 9.000
##  Max.   :0.9999821   Max.   :1.1468   Max.   :0.9998347   Max.   :11.000
##     liveness          loudness         mechanism          mode
##  Min.   :0.00729   Min.   :-58.373   Min.   :0.04215   major:65071
##  1st Qu.:0.09683   1st Qu.:-11.692   1st Qu.:0.26977   minor:34929
##  Median :0.12993   Median : -8.054   Median :0.48542
##  Mean   :0.21280   Mean   : -9.574   Mean   :0.50052
##  3rd Qu.:0.27118   3rd Qu.: -5.782   3rd Qu.:0.71795
##  Max.   :1.00000   Max.   :  4.479   Max.   :1.00000
##     organism       speechiness          tempo        time_signature
##  Min.   :0.0000   Min.   :0.02193   Min.   : 31.08   Min.   :0.000
##  1st Qu.:0.2537   1st Qu.:0.03619   1st Qu.: 96.20   1st Qu.:4.000
##  Median :0.4558   Median :0.04995   Median :119.97   Median :4.000
##  Mean   :0.4622   Mean   :0.10440   Mean   :120.27   Mean   :3.877
##  3rd Qu.:0.6480   3rd Qu.:0.10030   3rd Qu.:139.88   3rd Qu.:4.000
##  Max.   :0.9742   Max.   :0.96866   Max.   :245.95   Max.   :5.000
##     valence       acoustic_vector_0  acoustic_vector_1  acoustic_vector_2
##  Min.   :0.0000   Min.   :-1.10839   Min.   :-1.14029   Min.   :-0.82333
##  1st Qu.:0.2561   1st Qu.:-0.40534   1st Qu.:-0.17414   1st Qu.:-0.14697
##  Median :0.4755   Median :-0.03586   Median : 0.08402   Median : 0.12737
##  Mean   :0.4828   Mean   :-0.10299   Mean   : 0.01553   Mean   : 0.05889
##  3rd Qu.:0.7046   3rd Qu.: 0.19250   3rd Qu.: 0.28362   3rd Qu.: 0.27462
##  Max.   :1.0000   Max.   : 0.99251   Max.   : 0.80219   Max.   : 0.62795
##  acoustic_vector_3  acoustic_vector_4  acoustic_vector_5   acoustic_vector_6
##  Min.   :-0.85554   Min.   :-1.03662   Min.   :-1.047057   Min.   :-0.77799
##  1st Qu.:-0.23829   1st Qu.:-0.31234   1st Qu.:-0.089535   1st Qu.:-0.31612
##  Median : 0.06072   Median :-0.01359   Median : 0.043312   Median :-0.09914
##  Mean   : 0.05160   Mean   : 0.02437   Mean   : 0.001142   Mean   :-0.02559
##  3rd Qu.: 0.28299   3rd Qu.: 0.37413   3rd Qu.: 0.147826   3rd Qu.: 0.20510
##  Max.   : 1.14114   Max.   : 0.95508   Max.   : 0.388981   Max.   : 1.04095
##  acoustic_vector_7
##  Min.   :-0.991692
##  1st Qu.:-0.273700
##  Median :-0.005157
##  Mean   :-0.002709
##  3rd Qu.: 0.221621
##  Max.   : 1.164311
```

Which variables have the highest correlation? Which variable has the highest correlation with danceability?

```
df3_wo_mode <- cbind(df3[,1:15], df3[,17:29])
cor_matrix <- cor(df3_wo_mode)
max_val <- max(cor_matrix[which(cor_matrix < 1)])
      # the < 1 filters out attributes paired with themselves
which(cor_matrix == max_val, arr.ind = TRUE)
```

```
##              row col
```

```
## bounciness      6   5
## beat_strength   5   6
```

We can see that the highest correlation among any pari of attributes is between bounciness and beat strength.

```
cor_matrix_danceability <- cor_matrix["danceability", ]
max_val_danceability <-
  max(cor_matrix_danceability[which(cor_matrix_danceability < 1)])
which(cor_matrix_danceability == max_val_danceability)
```

```
## beat_strength
##             5
```

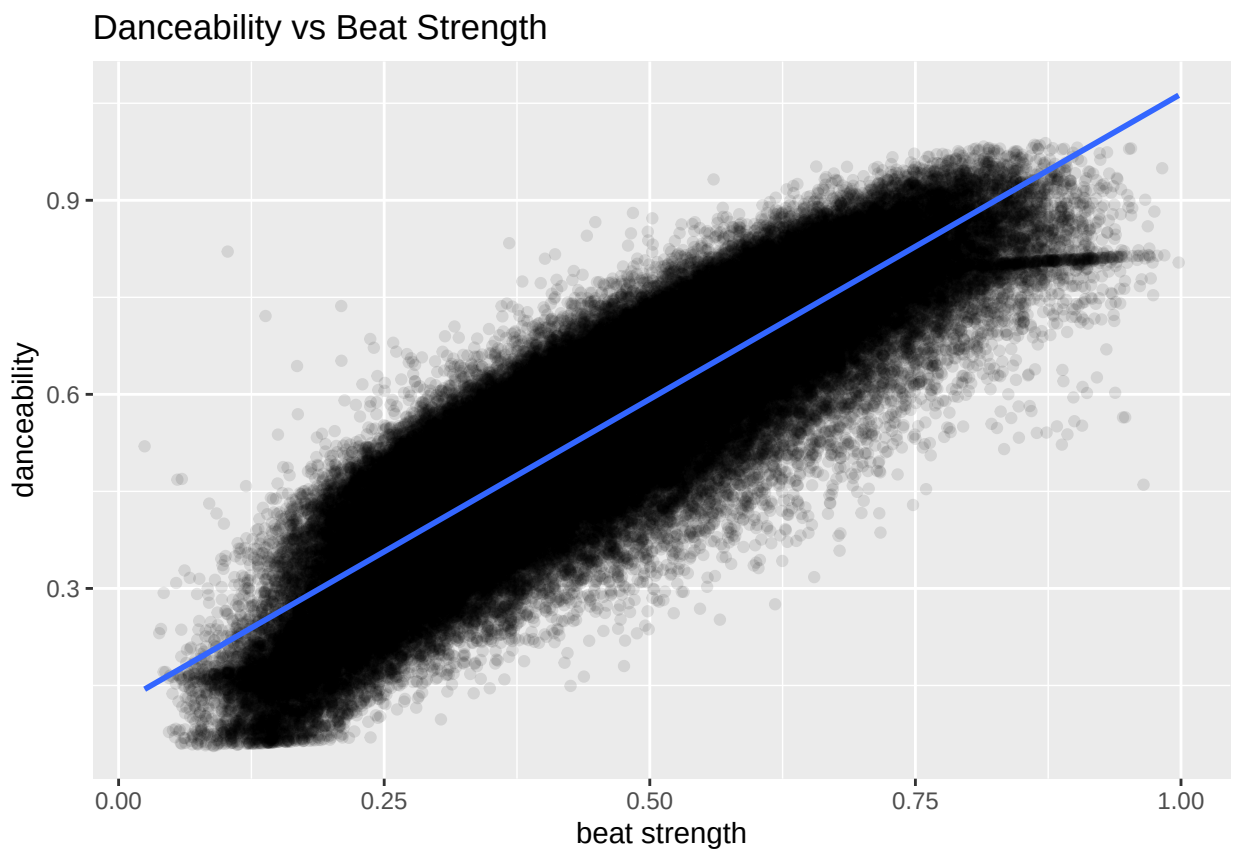
The variable with the highest correlation to danceability is beat strength. This fact will be used later when creating models.

What does the correlation between danceability and beat strength look like?

We discovered that the variable most highly correlated with danceability is beat strength. The plot below should show this correlation. (Initially, it was in this graph that I noticed the danceability values of 0, which eventually let me to discover the tempo values of 0, which were discussed and removed from the data in the data cleaning portion of this report.)

```
library(ggplot2)
ggplot(df3, aes(x=beat_strength,y=danceability)) + geom_point(alpha = 0.1) +
  geom_smooth(method ='lm') + labs(title = "Danceability vs Beat Strength") +
  xlab("beat strength") + ylab("danceability")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



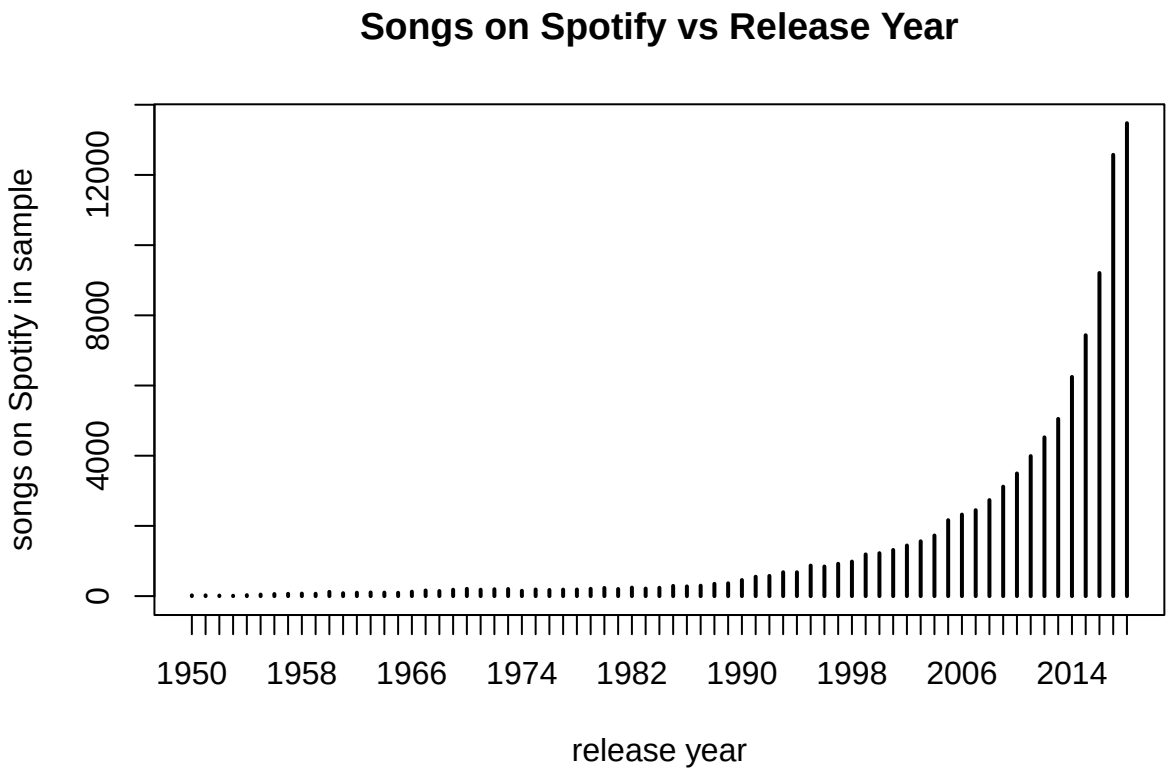
There is a positive linear correlation between beat strength and danceability.

What time period does our data range over?

```
range(df3$release_year)
```

```
## [1] 1950 2018
```

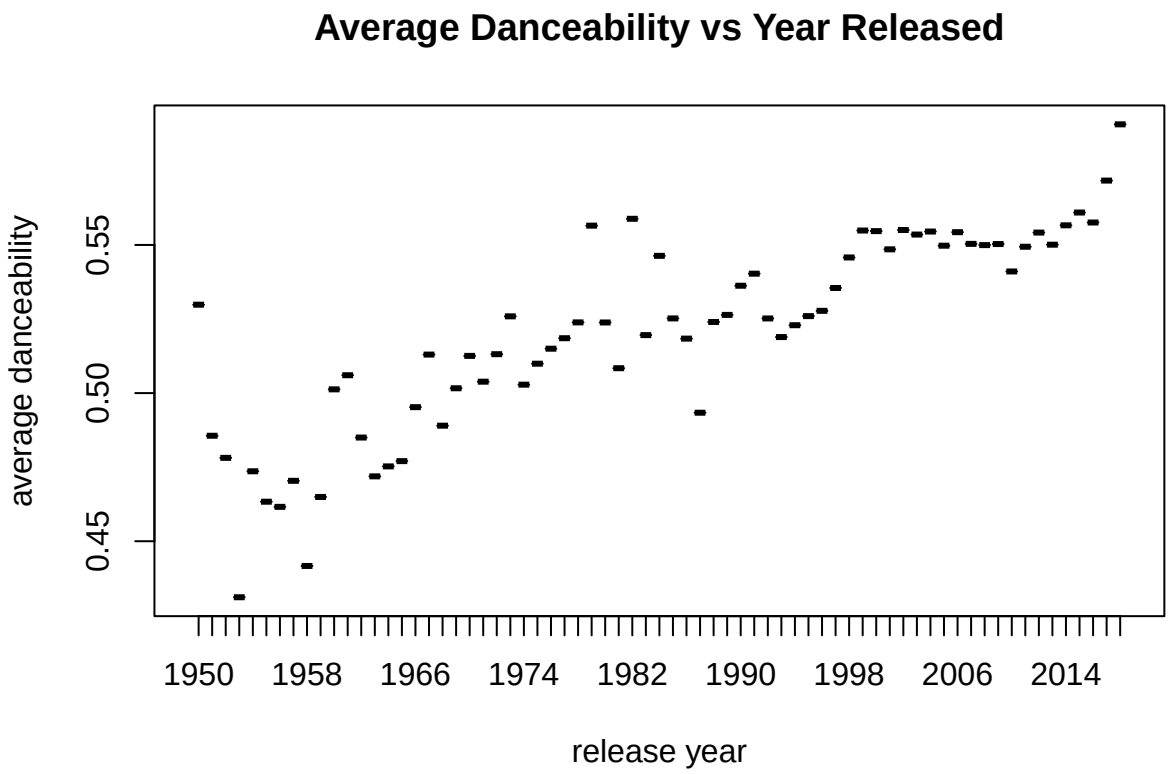
```
plot(table(df3$release_year), ylab = "songs on Spotify in sample", xlab = "release year",
     main = "Songs on Spotify vs Release Year")
```



Our data ranges from 1950 to 2018. We can see from the graph that there is an exponential increase in the number of currently on Spotify from a given year as the year increases. Although this only accounts for a small sample of the songs on spotify, it should be a good representation of the general Spotify song list since it was chosen pseudo-randomly.

How does danceability correlate with time?

```
df4 <- df3
df4$release_year <- factor(df4$release_year)
means <- aggregate(df4$danceability, list(df4$release_year), FUN=mean)
plot(means, ylab = "average danceability", xlab = "release year",
     main = "Average Danceability vs Year Released")
```



We can see from this graph that the danceability of songs has increased over time.

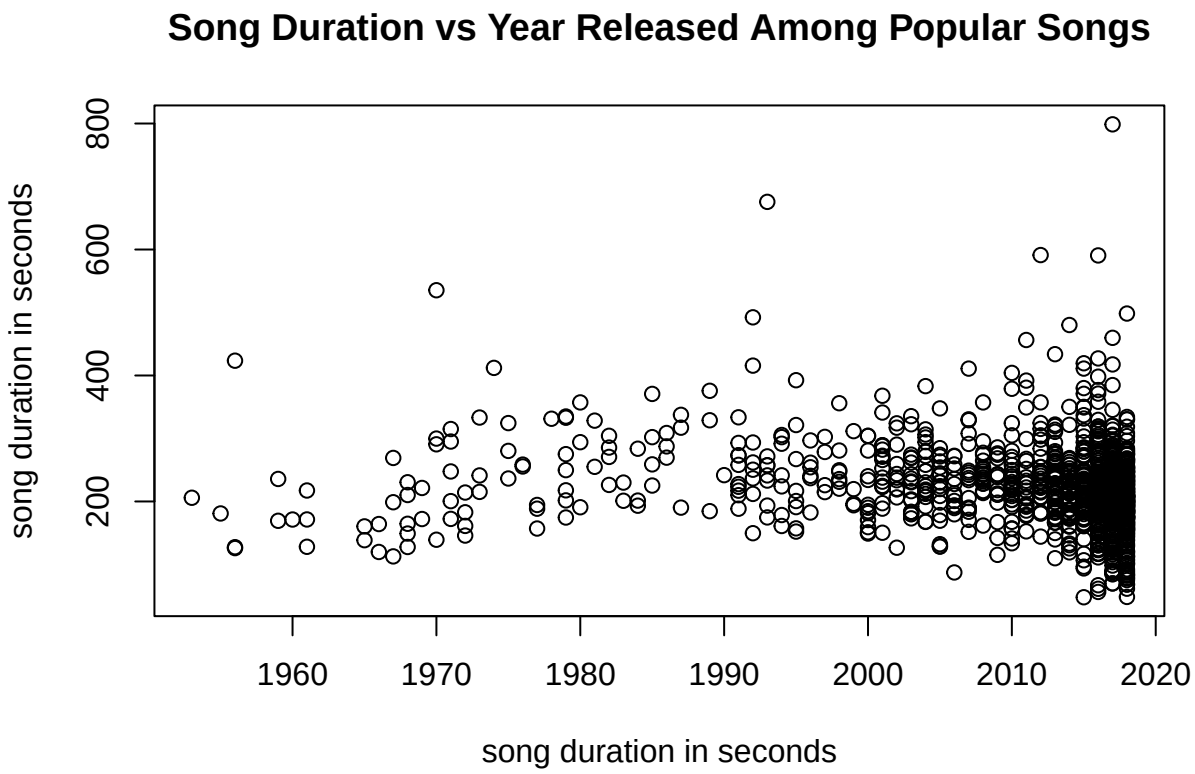
Is the duration of popular songs decreasing over time?

I have heard from several music theorists that the length of popular songs is decreasing. They claim that this is due to several causes, including the switch from CD's to online music and Spotify's financial incentive for shorter songs (since artists are paid per song stream, having multiple short songs pays more than having one long song.) Although we cannot test these possible causes from this data, we can test to see if our data supports their initial hypothesis that the length of popular songs is decreasing.

```
range(df3$us_popularity_estimate)
```

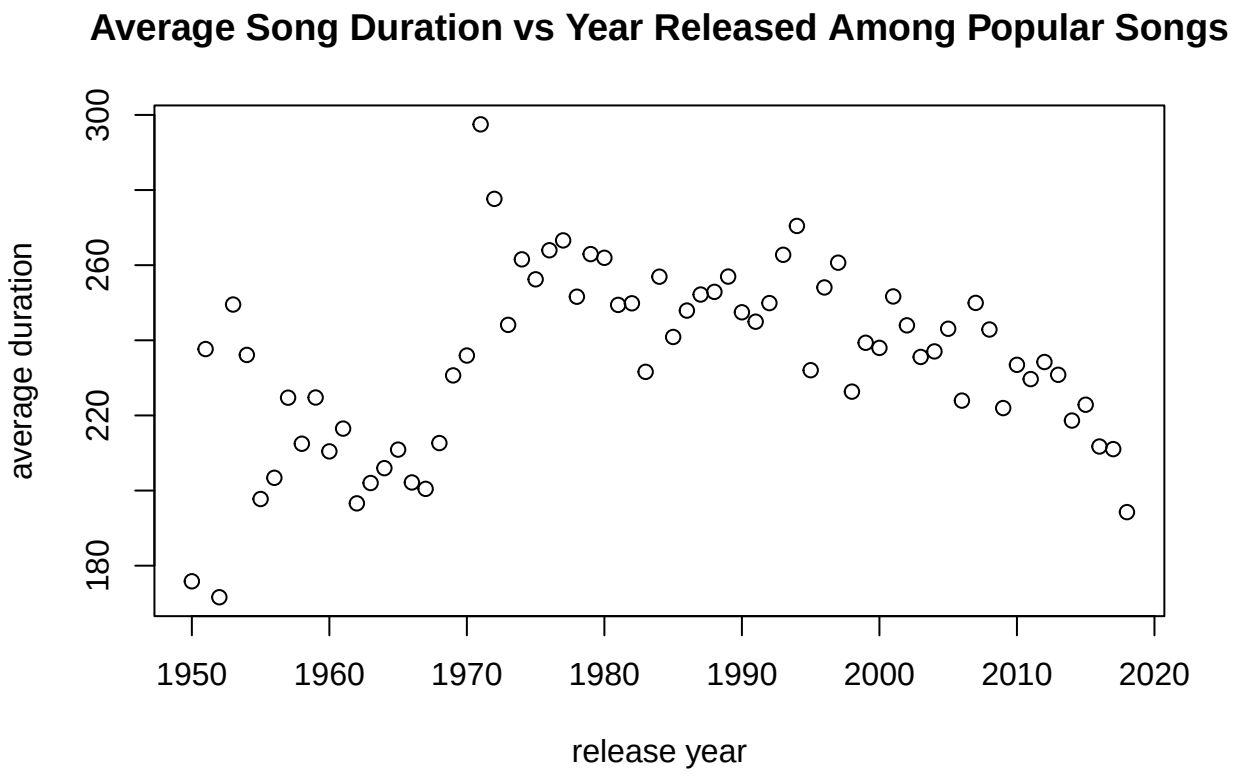
```
## [1] 90.00014 99.99999
```

```
popular_songs <- df3[which(df3$us_popularity_estimate > 99.9), ]
                    # filter to most popular songs
plot(popular_songs$release_year, popular_songs$duration,
     ylab = "song duration in seconds", xlab = "song duration in seconds",
     main = "Song Duration vs Year Released Among Popular Songs")
```



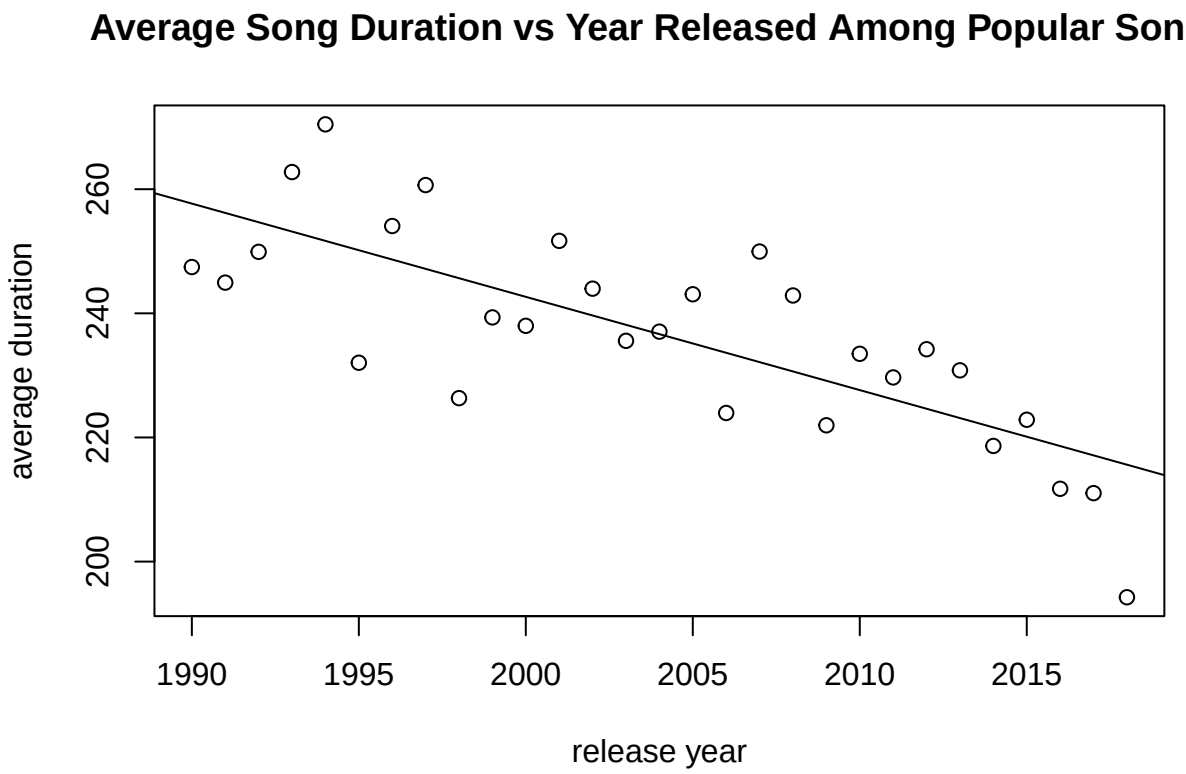
Looking at this graph, we do not see a steady negative correlation between release year and song duration. However, the graph it not very legible. To make a more legible graph, I will use the average song duration for each release year.

```
popular_songs2 <- df3[order(df3$us_popularity_estimate, decreasing = TRUE), ]
popular_songs2 <- Reduce(rbind, by(popular_songs2,
                                   popular_songs2["release_year"],
                                   head, n = 50))
                                   # filter to most popular songs for each year
means <- aggregate(popular_songs2$duration, list(popular_songs2$release_year),
                   FUN=mean)
plot(means, ylab = "average duration", xlab = "release year",
     main = "Average Song Duration vs Year Released Among Popular Songs")
```



Looking at the overall time frame of 1950 to 2018, we do not see the negative correlation that we expected. However, we do see a negative correlation at the very end of the graph, from 1990 to 2018. Below I will create another graph for those years.

```
means_after_1990 <- means[which(means$Group.1 >= 1990), ]
plot(means_after_1990, ylab = "average duration", xlab = "release year",
     main = "Average Song Duration vs Year Released Among Popular Songs") +
  abline(lm(means_after_1990$x ~ means_after_1990$Group.1))
```




```
## integer(0)
```

Looking at this graph, we can see the negative correlation that we expected. When we consider the years 1990 to 2018, we can say that the duration of popular songs has decreased on average.

Machine Learning Algorithms

Separating Testing and Training Data

First I separate my data into testing and training data. The training data contains 80% (80000 rows) of the sampled data, and the testing data contains 20% (20000 rows) of the sampled data.

```
set.seed(1234)
i <- sample(1:nrow(df3), nrow(df3)*0.8, replace=FALSE)
train <- df3[i,]
test <- df3[-i,]
```

Linear Regression

Since beat strength has the largest correlation with the target variable danceability, I would like to create a model with beat strength as the sole predictor, and a second model using all predictors.

```
lm1 <- lm(danceability~beat_strength, data = train)
summary(lm1)
```

```
##
## Call:
## lm(formula = danceability ~ beat_strength, data = train)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.57133 -0.06282  0.00819  0.06864  0.60316
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1205696  0.0009151   131.8   <2e-16 ***
## beat_strength 0.9443451  0.0018580   508.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08983 on 79998 degrees of freedom
## Multiple R-squared:  0.7636, Adjusted R-squared:  0.7636
## F-statistic: 2.583e+05 on 1 and 79998 DF,  p-value: < 2.2e-16
```

```
pred <- predict(lm1, newdata=test)
print(paste("MSE =", mean((pred - test$danceability)^2)))
```

```
## [1] "MSE = 0.00818650956407832"
```

```
print(paste("Correlation =", cor(pred, test$danceability)))
```

```
## [1] "Correlation = 0.872847071683489"
```

```
lm1 <- lm(danceability~., data = train)
summary(lm1)
```

```
##
## Call:
## lm(formula = danceability ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37271 -0.04041  0.00525  0.04669  0.22624
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             4.051e-02  4.836e-02   0.838 0.402273
## duration               -2.503e-05  2.149e-06 -11.647  < 2e-16 ***
## release_year            1.921e-04  2.343e-05   8.200 2.44e-16 ***
## us_popularity_estimate  2.000e-04  8.233e-05   2.429 0.015123 *
## acousticness           -7.586e-03  3.116e-03  -2.434 0.014915 *
## beat_strength          -2.615e-01  7.460e-03 -35.053  < 2e-16 ***
## bounciness              9.579e-01  9.131e-03 104.908  < 2e-16 ***
## dyn_range_mean         -9.836e-03  3.572e-04 -27.537  < 2e-16 ***
## energy                 -7.427e-02  2.380e-03 -31.206  < 2e-16 ***
## flatness               -1.660e-01  7.161e-03 -23.187  < 2e-16 ***
## instrumentalness       -4.275e-02  1.043e-03 -40.977  < 2e-16 ***
## key                    -1.417e-04  6.329e-05  -2.240 0.025121 *
## liveness               -4.853e-03  1.266e-03  -3.833 0.000127 ***
## loudness                2.309e-03  8.062e-05  28.646  < 2e-16 ***
## mechanism               3.115e-01  4.381e-03  71.110  < 2e-16 ***
## modeminor               3.188e-04  4.886e-04   0.653 0.514037
## organism                2.293e-02  7.047e-03   3.254 0.001138 **
## speechiness            -8.624e-03  2.491e-03  -3.462 0.000536 ***
```

```
## tempo                -8.317e-04  9.406e-06 -88.425  < 2e-16 ***
## time_signature        6.836e-03  4.808e-04  14.218  < 2e-16 ***
## valence               8.130e-03  1.253e-03   6.487 8.78e-11 ***
## acoustic_vector_0    -3.879e-02  2.459e-03 -15.773  < 2e-16 ***
## acoustic_vector_1    -2.250e-02  2.940e-03  -7.653 1.98e-14 ***
## acoustic_vector_2     2.388e-02  1.837e-03  12.996  < 2e-16 ***
## acoustic_vector_3    -2.600e-02  3.505e-03  -7.419 1.19e-13 ***
## acoustic_vector_4     2.225e-02  9.942e-04  22.384  < 2e-16 ***
## acoustic_vector_5     7.626e-03  3.010e-03   2.533 0.011304 *
## acoustic_vector_6     4.333e-03  1.669e-03   2.596 0.009431 **
## acoustic_vector_7    -2.096e-02  1.164e-03 -18.007  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0629 on 79971 degrees of freedom
## Multiple R-squared:  0.8841, Adjusted R-squared:  0.8841
## F-statistic: 2.179e+04 on 28 and 79971 DF,  p-value: < 2.2e-16
```

```
pred <- predict(lm1, newdata=test)
print(paste("MSE =", mean((pred - test$danceability)^2)))
```

```
## [1] "MSE = 0.00396507863053111"
```

```
print(paste("Correlation =", cor(pred, test$danceability)))
```

```
## [1] "Correlation = 0.940557988379894"
```

The first model, using only beat strength as a predictor, has an MSE of 0.00819 and a correlation of 0.872. This is already a good model on its own. Adding the other variables as predictors decreased the MSE by about 0.00422 (or decreased it by 55%, which is far from negligible) and increased the correlation by 0.069. Although beat strength is a good predictor on its own, adding the other variables helped to strengthen our model. However, if simplicity of a model and minimizing runtime was a priority, the beat strength model would be a good choice.

k Nearest Neighbors

Next, we will use the knn model. Since knn cannot measure the distance between levels of a factor, we will remove the variable mode and use the remaining predictors to predict danceability.

```
# Data Cleaning - variables 16 'mode' breaks knnreg() since it is a factor,
# so we remove it
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.3
```

```
## Loading required package: lattice
```

```
train_target <- train[,7]
train_predictors1 <- train[,5]
train_predictors2 <- cbind(train[,1:6], train[8:15],
                           train[,17:29]) # training data predictors minus mode
test_target <- test[,7]
train_predictors1 <- test[,5]
test_predictors2 <- cbind(test[,1:6], test[8:15],
                          test[,17:29]) # testing data predictors minus mode
```

First, we will find our ideal k value.

```
# the following code was used to choose the ideal k value
cor_k <- rep(0, 10)
mse_k <- rep(0, 10)
i <- 1
for (k in seq(1, 50, 5)){
  fit_k <- knnreg(train_predictors2, train_target, k=k)
  pred_k <- predict(fit_k, test_predictors2)
  cor_k[i] <- cor(pred_k, test_target)
  mse_k[i] <- mean((pred_k - test_target)^2)
  print(paste("k=", k, cor_k[i], mse_k[i]))
  i <- i + 1
}
```

Using the code above, our ideal k values (out of the tested values) is 11, which has the highest correlation and the lowest MSE.

```
fit <- knnreg(train_predictors2, train_target, k=11)
predictions <- predict(fit, test_predictors2)
print(paste("MSE = ", mean((predictions - test_target)^2))) # MSE
```

```
## [1] "MSE =  0.0148431460079334"
```

```
print(paste("Correlation = ", cor(predictions, test_target))) # correlation
```

```
## [1] "Correlation =  0.76464033747309"
```

Note: Since the runtime of this knn portion is already several minutes long, I decided against creating a second knn model using only beat strength as I did for the other algorithms.

Decision Tree

```
library(tree)
tree1 <- tree(danceability~beat_strength, data = train)
pred <- predict(tree1, newdata=test)
print(paste("MSE =", mean((pred - test$danceability)^2)))
```

```
## [1] "MSE = 0.00814860975429858"
```

```
print(paste("Correlation =", cor(pred, test$danceability)))
```

```
## [1] "Correlation = 0.873465643008183"
```

```
library(tree)
tree1 <- tree(danceability~., data = train)
pred <- predict(tree1, newdata=test)
print(paste("MSE =", mean((pred - test$danceability)^2)))
```

```
## [1] "MSE = 0.00681434503187663"
```

```
print(paste("Correlation =", cor(pred, test$danceability)))
```

```
## [1] "Correlation = 0.895410342100294"
```

In comparing the second model to the first, we see a 0.00134 decrease in MSE and a 0.022 increase in correlation. It is interesting to note that the two linear models had a more significant difference between them than the two decision trees.

Analysis of Algorithms

When comparing the beat strength only models, we can see very similar results between the linear model and the decision tree. The linear model has less than a 0.0004 increase in MSE compared to the decision tree. The linear model also has less than a 0.0005 decrease in correlation. Although one could technically claim than the decision tree performed slightly better, the difference is negligable, and the results could have likely been reversed given a different sampling of the original dataset. Therefore, both beat strength models performed almost identically.

When comparing the second linear model, the kNN model, and the second decision tree, all models use the same predictor variables, with the exception of kNN regression, which leaves out the "mode" variable. This may be a contributing factor for our kNN having the highest MSE value and lowest correlation value, with an MSE of 0.0148 and a correlation of 0.765. Nonetheless, this is still a very good model. Our second best model is the decision tree, which has an MSE of 0.00681 and a correlation of 0.895. The best performing model is the linear model with an MSE of a correlation of 0.941 and an MSE of 0.00396. Although the linear model could be considered the easiest to mathematically calculate (and it certainly takes my computer the least amount of runtime to calculate), it performed the best, proving that sometimes simple really is better.

In summary, here are the algorithms ranked based on performance:

1. Linear Regression
2. Decision Tree
3. k Nearest Neighbors