DATA 606 Data Project

Alice Ding

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Part 1 - Introduction

Music is an integral part of society and is present in so many of our lives – it is hard to go an entire day without hearing at least part of a song, whether from our own devices (computer, phone, etc.) or just from walking down the street or going into a coffee shop. We are constantly surrounded by music, melodies, and songs.

We all know what music we personally enjoy, but is there a technical or specific reason as to why certain songs top the charts? Are there specific factors of each track that constantly contribute to why we like certain songs more than others? Using certain factors and information about each song, is there then a way to even engineer a track into being "popular" or predict what songs will top the charts?

Using Spotify's API data for the top 2,000 songs from 1956 to 2019, this project will try to find whether there is a relationship between the various variables that Spotify has documented for each of their songs and whether or not the song is popular.

Part 2 - Data

The data pulled from Spotify's API holds the following fields:

- Index: ID
- Title: Name of the Track
- Artist: Name of the Artist*
- Top Genre: Genre of the Track
- Year: Release Year of the track
- Beats per Minute(BPM): The tempo of the song*
- Energy: The energy of a song - the higher the value, the more energetic the song*
- Danceability: The higher the value, the easier it is to dance to this song*
- Loudness: The higher the value, the louder the song*
- Valence: The higher the value, the more positive mood for the song*
- Length: The duration of the song*
- Acousticness: The higher the value the more acoustic the song is*

- Speechiness: The higher the value the more spoken words the song contains*
- Popularity: The higher the value the more popular the song is*

The fields with stars are all ones that will used for this project and the last field, `popularity`, is the value we're trying to predict.

Part 3 - Exploratory data analysis

We'll begin with a bit of clean-up.

```
df <- data |> select(
  Artist,
  Beats.Per.Minute..BPM.,
  Energy,
  Danceability,
  Loudness..dB.,
  Valence,
  Length..Duration.,
  Acousticness,
  Speechiness,
  Popularity
  )

df <- df |>
  rename("artist" = "Artist"
         , "bpm" = "Beats.Per.Minute..BPM."
         , "energy" = "Energy"
         , "danceability" = "Danceability"
         , "loudness" = "Loudness..dB."
         , "valence" = "Valence"
         , "duration" = "Length..Duration."
         , "acousticness" = "Acousticness"
         , "speechiness" = "Speechiness"
         , "popularity" = "Popularity"
         )

df$duration<- gsub(",", '', df$duration)

df$duration <- as.numeric(df$duration)

summary(df)
```

```
##     artist                bpm            energy        danceability
##  Length:1994        Min.   : 37.0   Min.   :  3.00   Min.   :10.00
##  Class :character   1st Qu.: 99.0   1st Qu.: 42.00   1st Qu.:43.00
##  Mode  :character   Median :119.0   Median : 61.00   Median :53.00
##                     Mean   :120.2   Mean   : 59.68   Mean   :53.24
##                     3rd Qu.:136.0   3rd Qu.: 78.00   3rd Qu.:64.00
##                     Max.   :206.0   Max.   :100.00   Max.   :96.00
##     loudness          valence         duration       acousticness
##  Min.   :-27.000   Min.   : 3.00   Min.   :  93.0   Min.   : 0.00
##  1st Qu.:-11.000   1st Qu.:29.00   1st Qu.: 212.0   1st Qu.: 3.00
##  Median : -8.000   Median :47.00   Median : 245.0   Median :18.00
##  Mean   : -9.009   Mean   :49.41   Mean   : 262.4   Mean   :28.86  
##  3rd Qu.: -6.000   3rd Qu.:69.75   3rd Qu.: 289.0   3rd Qu.:50.00  
##  Max.   : -2.000   Max.   :99.00   Max.   :1412.0   Max.   :99.00  
##   speechiness       popularity    
##  Min.   : 2.000   Min.   : 11.00  
##  1st Qu.: 3.000   1st Qu.: 49.25  
##  Median : 4.000   Median : 62.00  
##  Mean   : 4.995   Mean   : 59.53  
##  3rd Qu.: 5.000   3rd Qu.: 71.00  
##  Max.   :55.000   Max.   :100.00
```

Distributions Let's start by looking at the top artists.

```r
artist_counts <- df |>
  group_by(artist) |>
  summarise(count = n(),
            .groups = 'drop') |>
  arrange(desc(count))

head(artist_counts, 10)
```

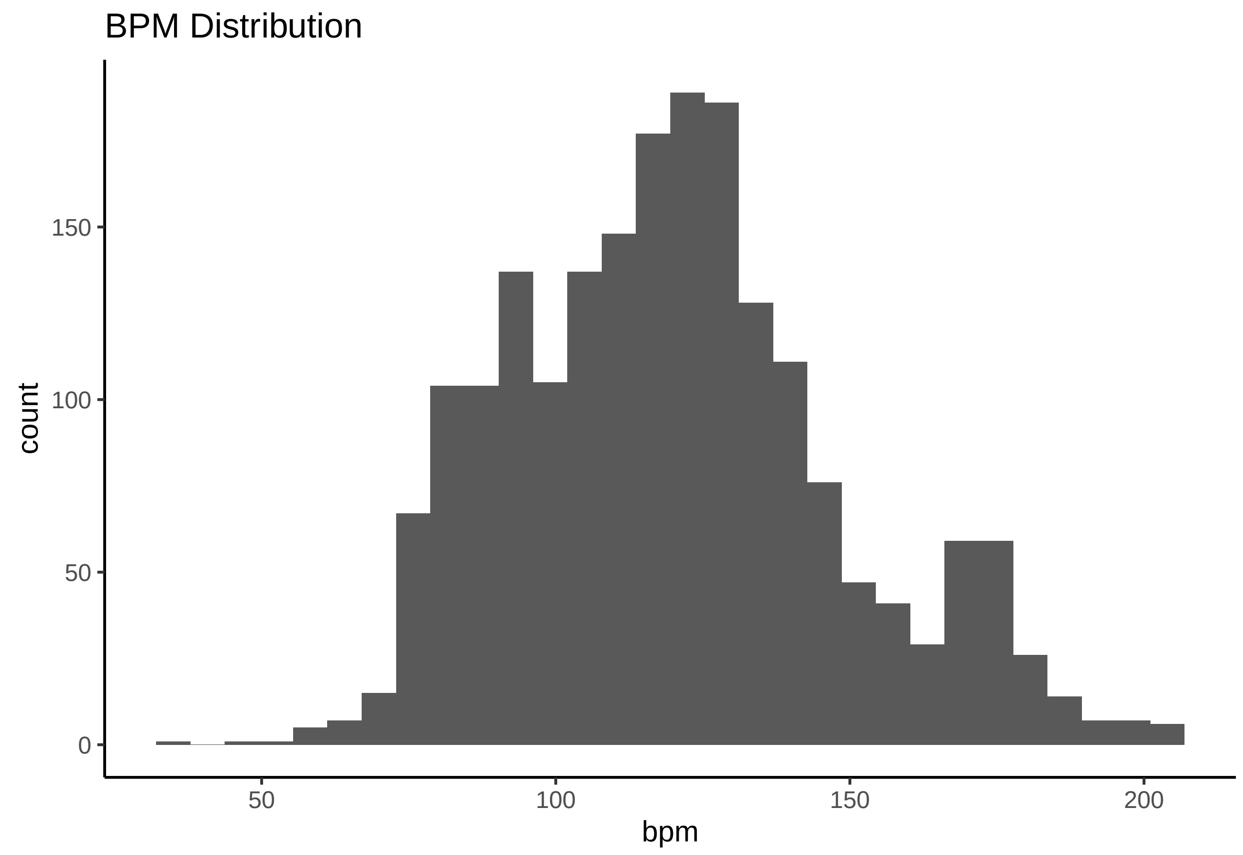
```
## # A tibble: 10 x 2
##    artist             count
##    <chr>              <int>
##  1 Queen                 37
##  2 The Beatles           36
##  3 Coldplay              27
##  4 U2                    26
##  5 The Rolling Stones    24
##  6 Bruce Springsteen     23
##  7 Michael Jackson       23
##  8 ABBA                  22
##  9 David Bowie           21
## 10 Fleetwood Mac         18
```

Not surprising that Queen and The Beatles are top of this list – these artists dominated the late 1900s.

Now, we'll take a look at the numerical columns and view the distribution of each.

```r
ggplot(df, aes(x = bpm)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("BPM Distribution")
```

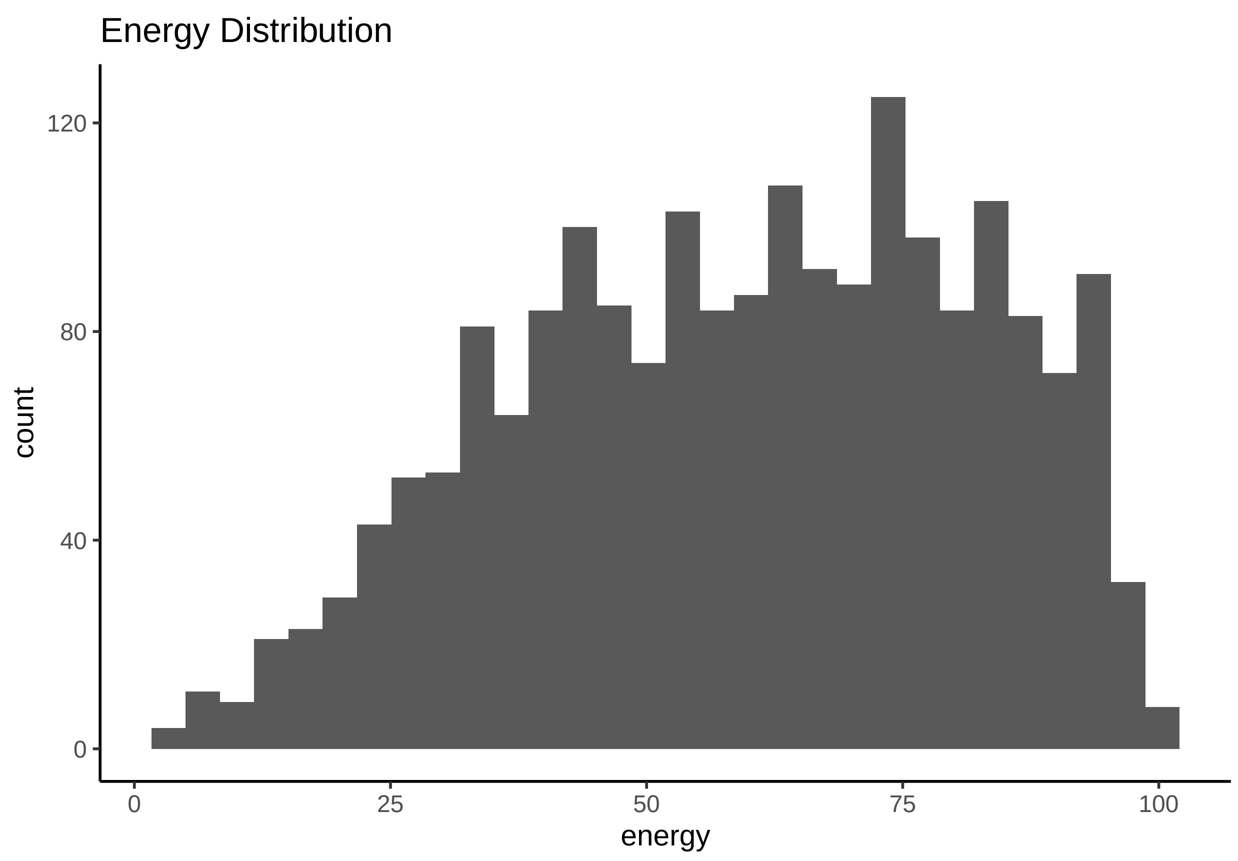
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



BPM seems relatively normal – there's a clear peak at around 120 and a few more outliners at around 170 with a small tail, but other than that, it's more normal than I'd expect actually.

```
ggplot(df, aes(x = energy)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("Energy Distribution")
```

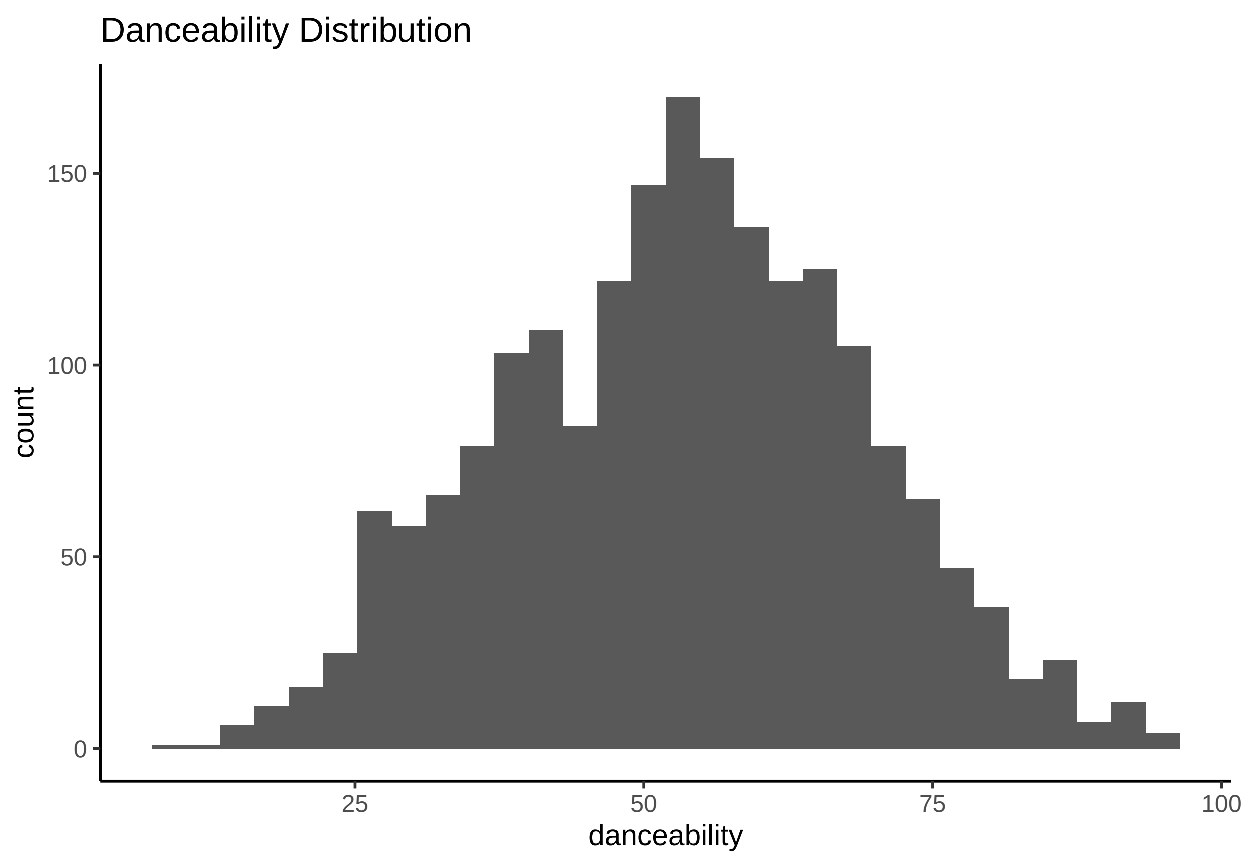
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



Energy seems to be skewed slightly towards the left, indicating the songs here are higher on the energy scale rather than not.

```
ggplot(df, aes(x = danceability)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("Danceability Distribution")
```

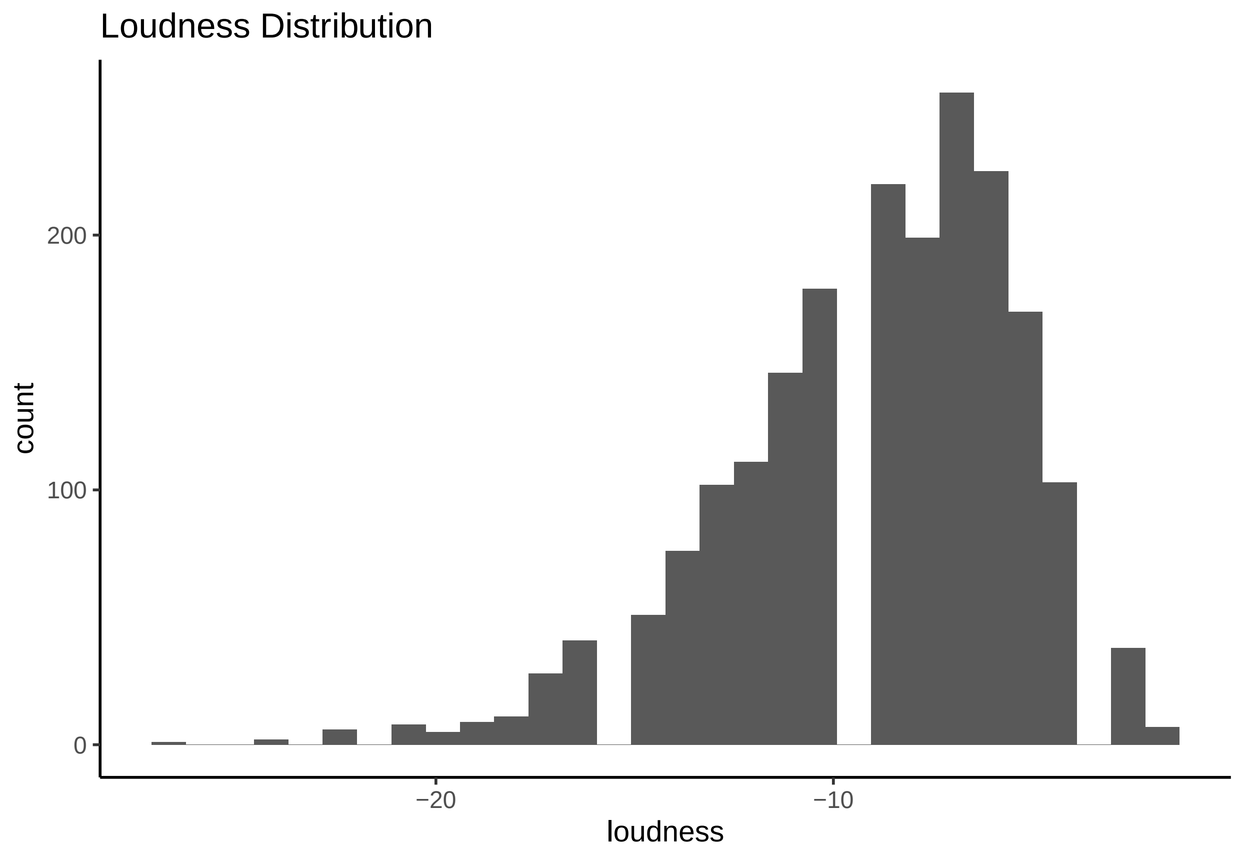
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



This distribution is surprisingly very normal as well – not too skewed and has a very clear peak at around 55.

```
ggplot(df, aes(x = loudness)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("Loudness Distribution")
```

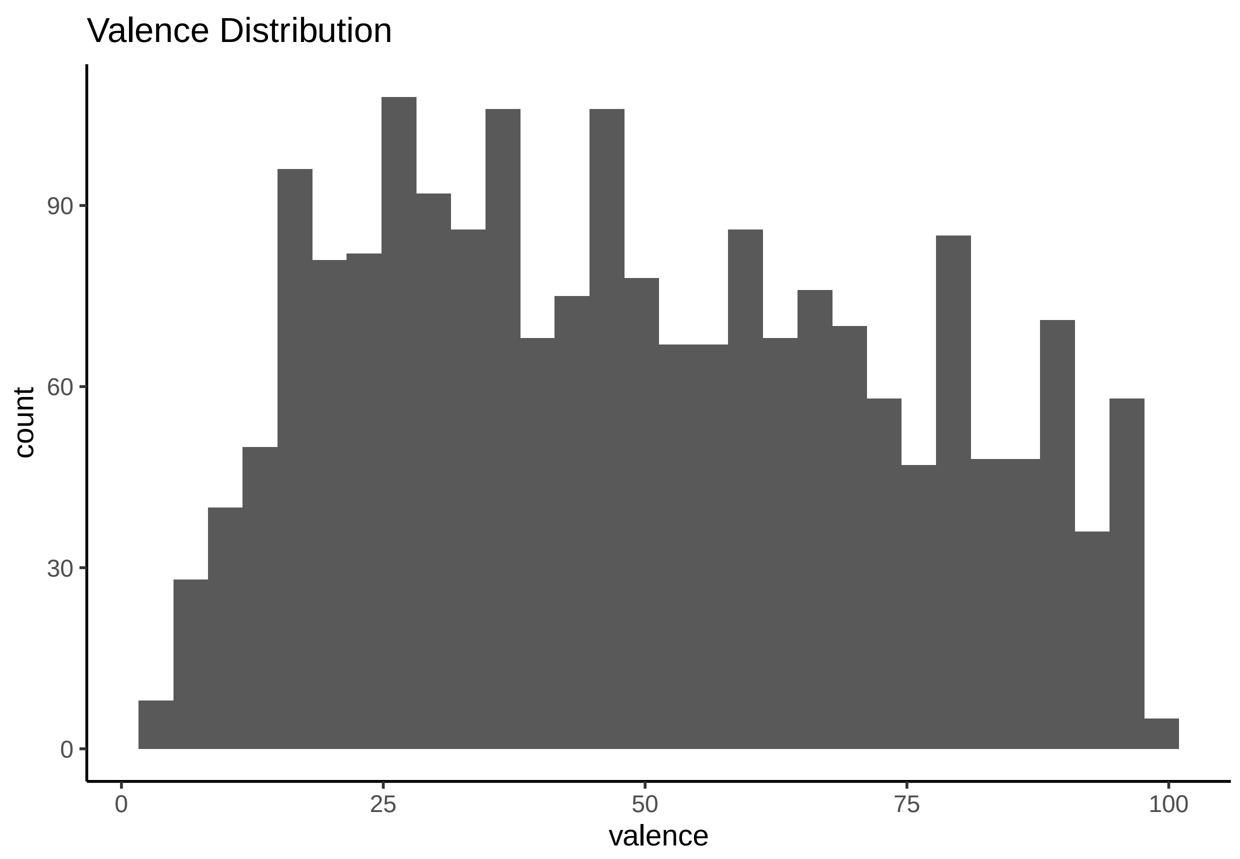
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There seems to be a left skewness associated with `loudness` here, signalling perhaps that higher dB songs are more prominent here..

```
ggplot(df, aes(x = valence)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("Valence Distribution")
```

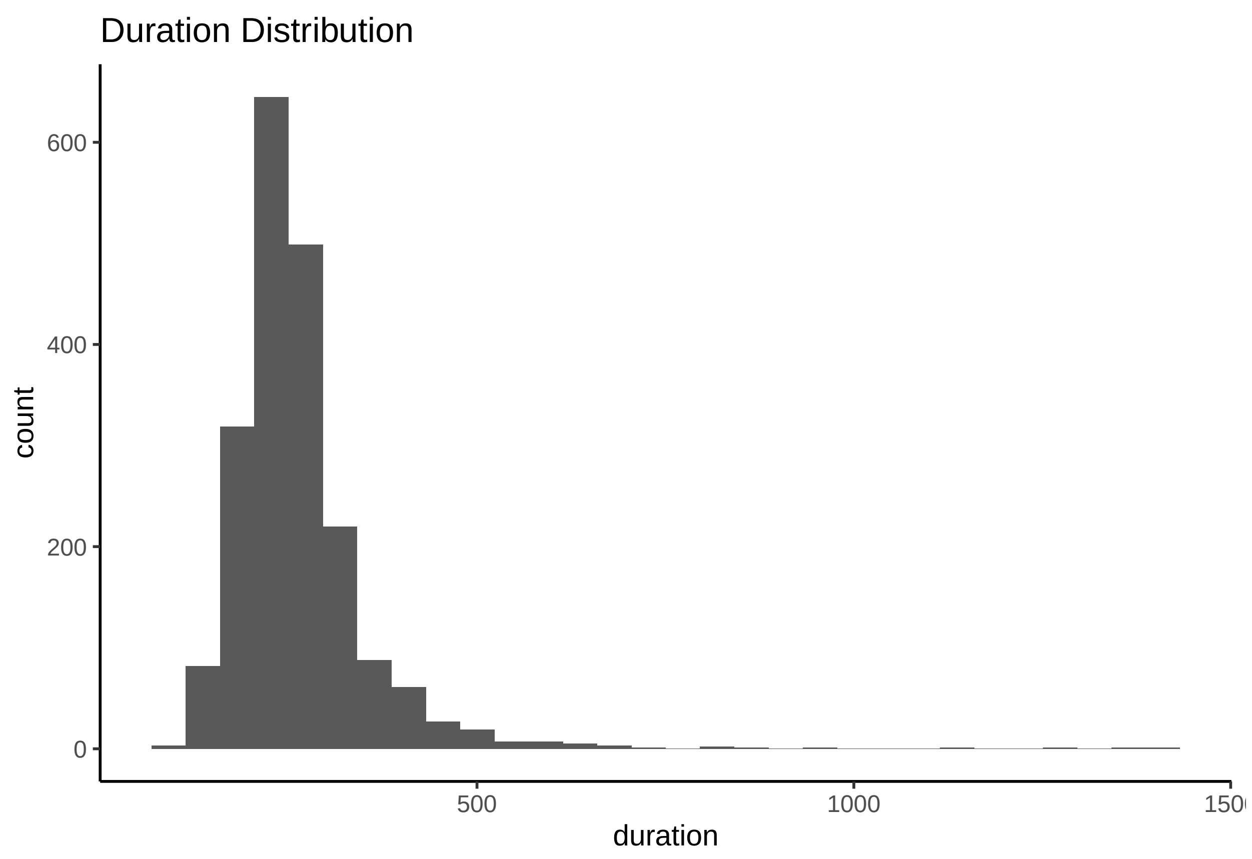
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



The distribution here is consistent here and isn't much of a bell-shape.

```
ggplot(df, aes(x = duration)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("Duration Distribution")
```

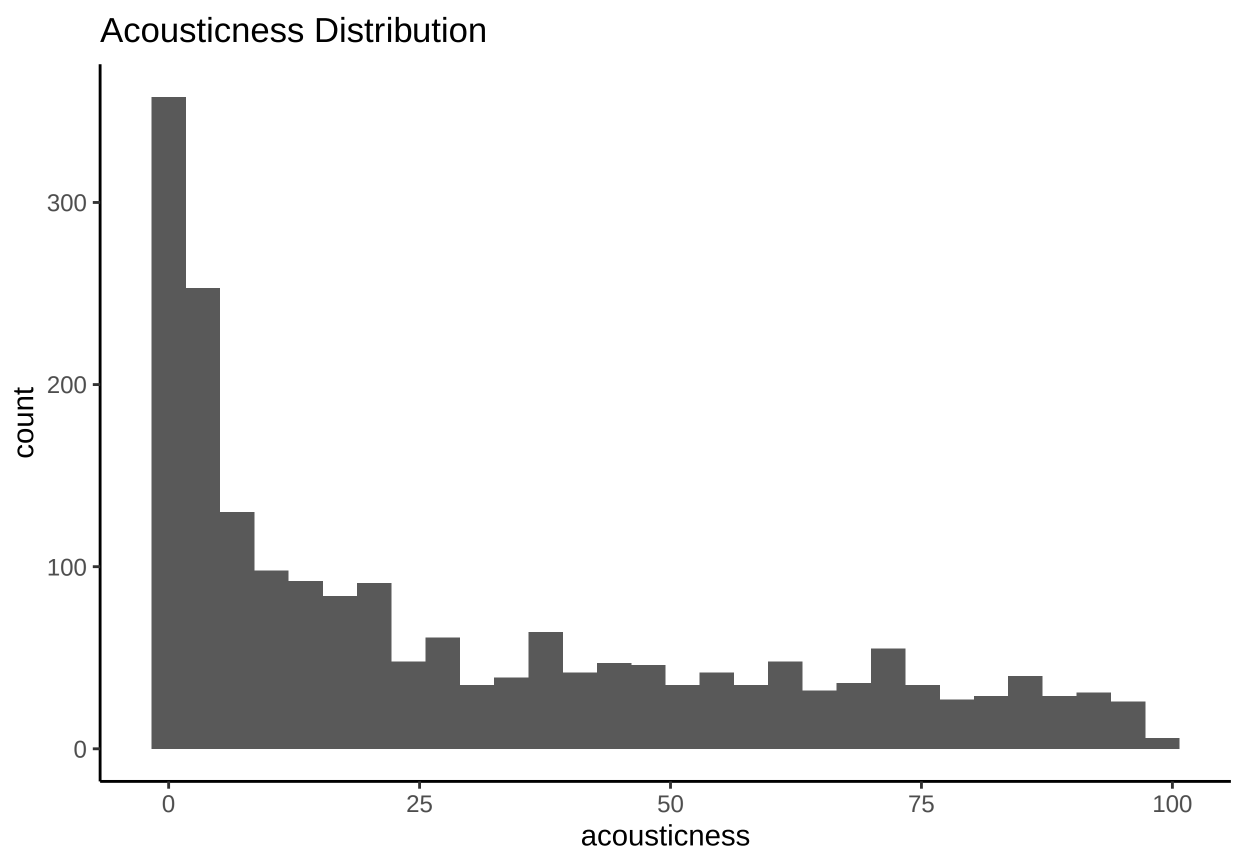
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



This seems to be right skewed which makes sense – a song can be abnormally long, but it can't be short in the same way as 0 is the lowest value for duration.

```r
ggplot(df, aes(x = acousticness)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("Acousticness Distribution")
```

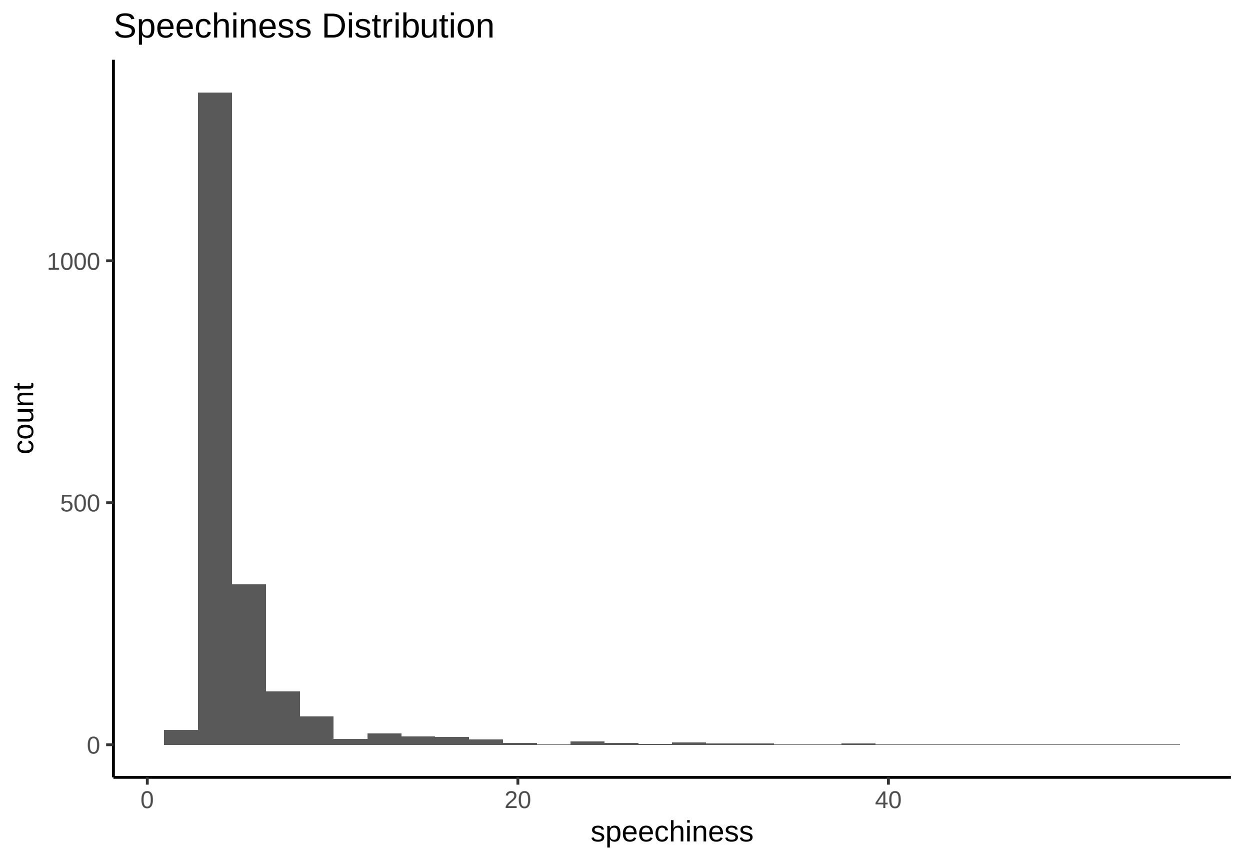
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



This isn't normal at all in distribution.

```r
ggplot(df, aes(x = speechiness)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("Speechiness Distribution")
```

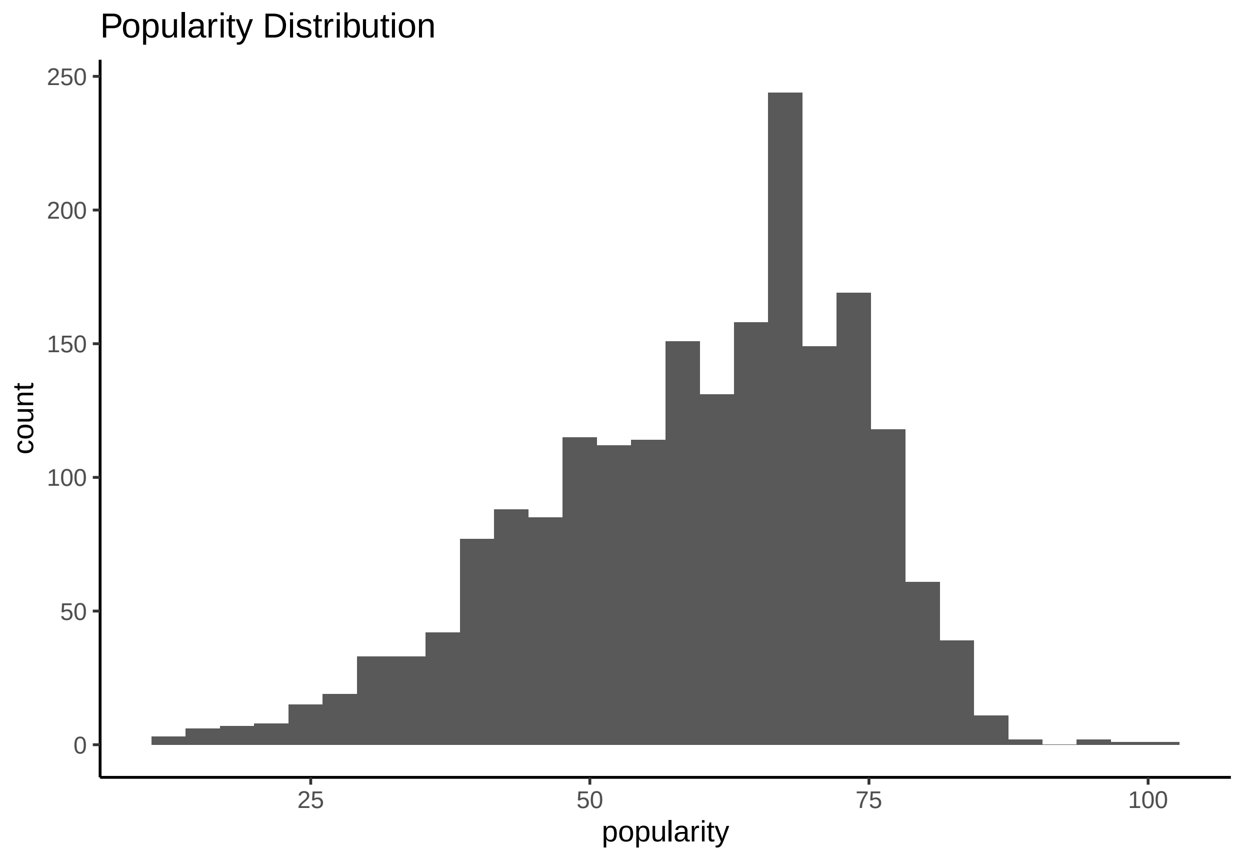
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Similar to `acousticness`, this isn't normal at all in distribution – you could argue it's very much right skewed though more than the chart above.

```r
ggplot(df, aes(x = popularity)) +
  geom_histogram() +
  theme_classic() +
  ggtitle("Popularity Distribution")
```

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



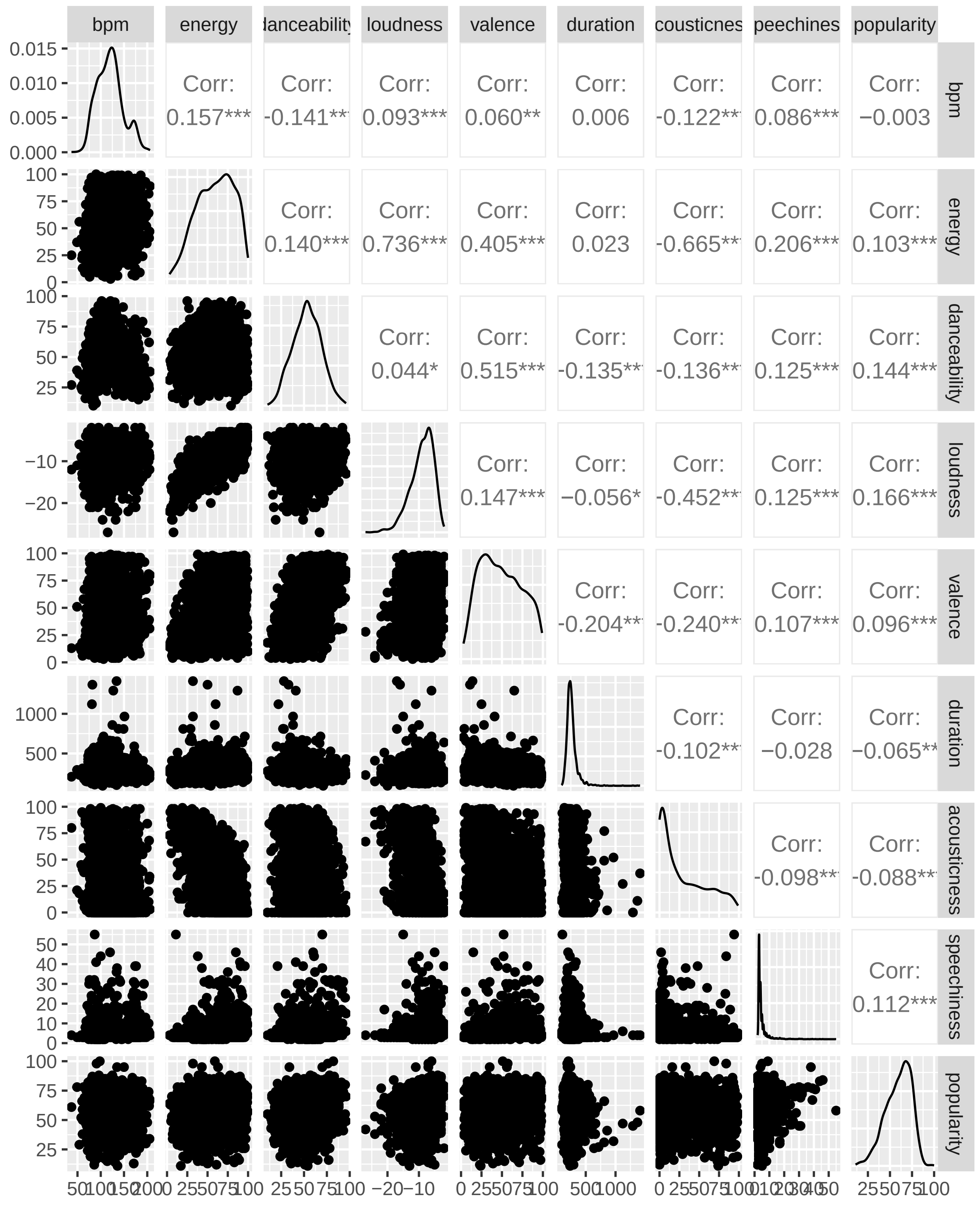
There is a slight left skew here and given this dataset is the top tracks over the years, it makes sense that they would be higher on the popularity scale.

```r
library(GGally)
```

Relationships

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
ggpairs(df[, c(2:10)])
```



Looks like every field but `bpm` correlates with `popularity` and a lot of these fields seem correlated with each other; this makes sense as for example, I'd imagine a song high in energy is also relatively easy to dance to.

Part 4 - Inference

Binomial Regression

Defining the Target When deciding on a model, I think it actually makes more sense to take the `popularity` field and translate it into a binary – the songs greater than 50 are popular (1) and the ones lower than or equal to 50 are not (0). Let's see what happens if we do that.

```
df <- df |>
  mutate(
    popularity_b = ifelse(popularity > 50, 1, 0)
    )

popularity_counts <- df |>
  group_by(popularity_b) |>
  summarise(count = n(),
            pct = 100 * n() / nrow(df),
            .groups = 'drop') |>
  arrange(desc(count))

popularity_counts
```

```
## # A tibble: 2 x 3
##   popularity_b count   pct
##          <dbl> <int> <dbl>
## 1            1  1463  73.4
## 2            0   531  26.6
```

It looks like using this methodology, 73% of the dataset is popular and 27% aren't.

Modeling Let's begin the modeling process by splitting our data into a training and testing set; we'll be doing a 70/30 split.

```
set.seed(1234)
train.rows <- sample(nrow(df), nrow(df) * .7)
df_train <- df[train.rows,]
df_test <- df[-train.rows,]

popularity_prediction <- table(df_train$popularity_b) %>% prop.table
popularity_prediction
```

```
##
##         0         1
## 0.2645161 0.7354839
```

Note that with this split, if we were just to predict that if a song was popular, we would be correct 74% of the time.

Now, let's create a binomial model using all variables and look at the coefficient values and statistical significance for everything other than `artist`.

```r
binomial_regression <- glm(popularity_b ~ bpm + energy + danceability
                           + loudness + valence + duration
                           + acousticness + speechiness
                           + artist, data = df_train
                           , family = binomial(link = 'logit'))

summary(binomial_regression)$coefficients[c("bpm"
                                           , "energy"
                                           , "danceability"
                                           , "loudness"
                                           , "valence"
                                           , "duration"
                                           , "acousticness"
                                           , "speechiness")
                                         ,]
```

```
##                  Estimate  Std. Error    z value   Pr(>|z|)
## bpm           0.001231570 0.004526663  0.2720701 0.78556814
## energy       -0.002529935 0.011873702 -0.2130705 0.83127201
## danceability  0.013382915 0.010676686  1.2534710 0.21003436
## loudness      0.036919474 0.060963261  0.6056020 0.54477906
## valence       0.010745389 0.007677022  1.3996820 0.16160858
## duration     -0.002715659 0.001358491 -1.9990254 0.04560560
## acousticness  0.009460143 0.006126279  1.5441908 0.12254212
## speechiness  -0.066865509 0.039385687 -1.6977109 0.08956234
```

Only two of these look like statistically significant predictors as `duration` and `speechiness` are both below 0.05 and 0.10 respectively. Nonetheless, we can see that `danceability`, `loudness`, `valence`, and `acousticness` contribute to popularity while `energy`, `duration`, and `speechiness` have a negative impact (all things held constant).

Would this model look better if we removed the fields that are more than 0.50 in z-score? This means running a model without `bpm`, `energy`, and `loudness`.

```r
binomial_regression_2 <- glm(popularity_b ~ danceability + valence
                             + duration + acousticness
                             + speechiness + artist, data = df_train
                             , family = binomial(link = 'logit'))

summary(binomial_regression_2)$coefficients[c("danceability"
                                           , "valence"
                                           , "duration"
                                           , "acousticness"
                                           , "speechiness")
                                         ,]
```

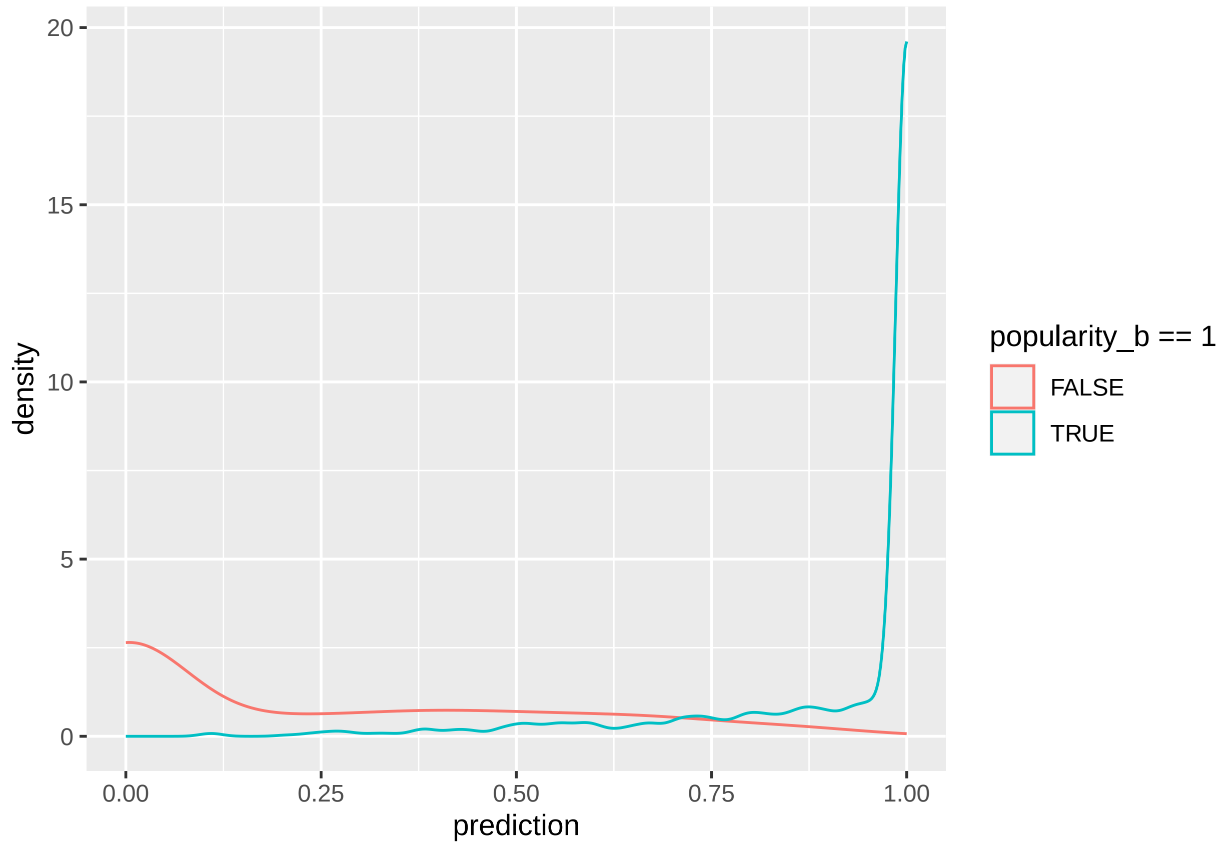
```
##                  Estimate  Std. Error   z value   Pr(>|z|)
## danceability  0.012403689 0.010362034  1.197032 0.23129399
## valence       0.011341082 0.006818430  1.663298 0.09625276
## duration     -0.002798483 0.001348606 -2.075093 0.03797795
## acousticness  0.008722360 0.005299068  1.646018 0.09976006
## speechiness  -0.068973551 0.037616440 -1.833601 0.06671321
```

This has left our model with a lot more significant predictors – almost all of them are statistically significant ($< .10$) now!

Verification

Train Dataset Let's see how this model performs for the training set.

```r
df_train$prediction <- predict(binomial_regression, type = 'response', newdata = df_train)
ggplot(df_train, aes(x = prediction, color = popularity_b == 1)) + geom_density()
```



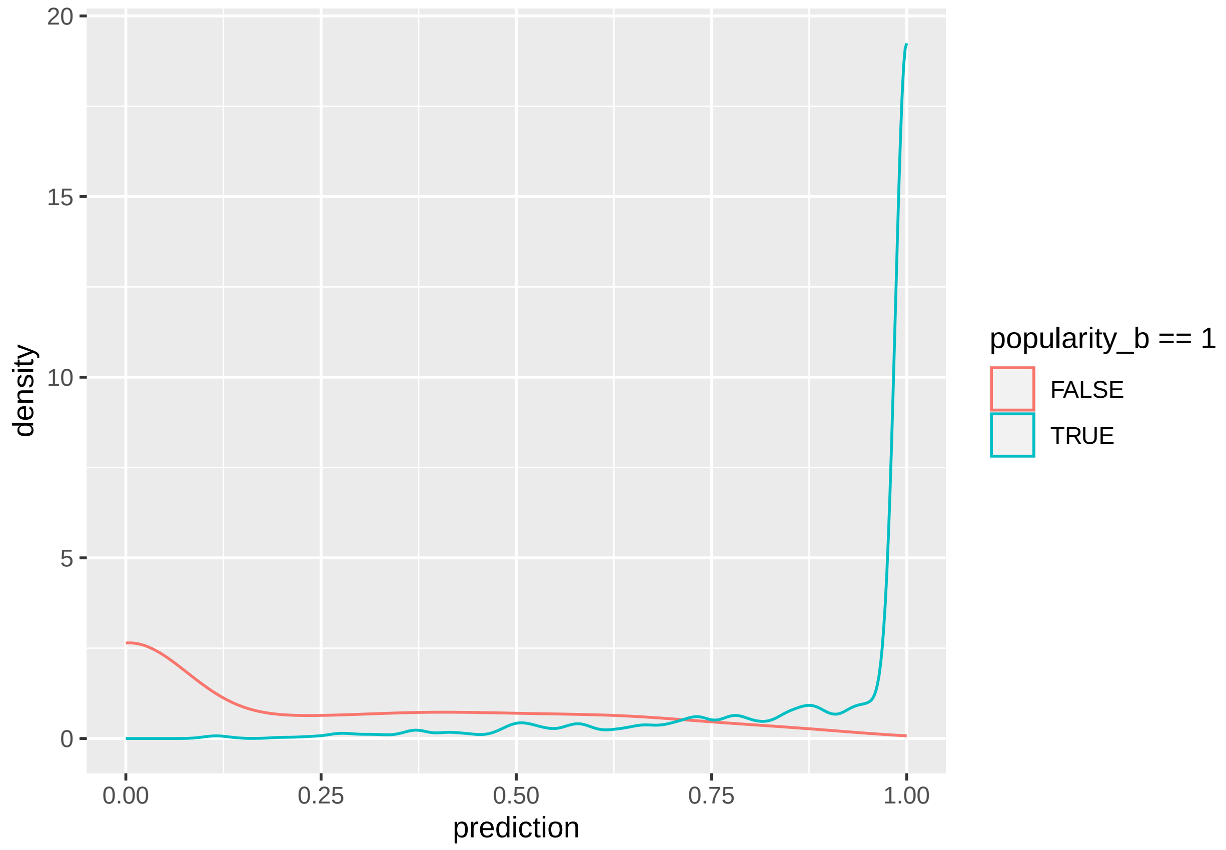
```r
df_train$prediction_class <- df_train$prediction > 0.5
tab <- table(df_train$prediction_class,
             df_train$popularity_b) |> prop.table() |> print()
```

```
##
##                  0          1
##   FALSE 0.20430108 0.03369176
##   TRUE  0.06021505 0.70179211
```

The accuracy for the training set is 20.43% + 70.18% = 90.61%. Given the simplest prediction which is just to guess every song is popular was at a 74% accuracy, this model is much better.

What about for the second model we have?

```r
df_train$prediction <- predict(binomial_regression_2, type = 'response', newdata = df_train)
ggplot(df_train, aes(x = prediction, color = popularity_b == 1)) + geom_density()
```



```r
df_train$prediction_class <- df_train$prediction > 0.5
tab <- table(df_train$prediction_class,
             df_train$popularity_b) |> prop.table() |> print()
```

```
##
##                    0          1
##   FALSE 0.20573477 0.03369176
##   TRUE  0.05878136 0.70179211
```

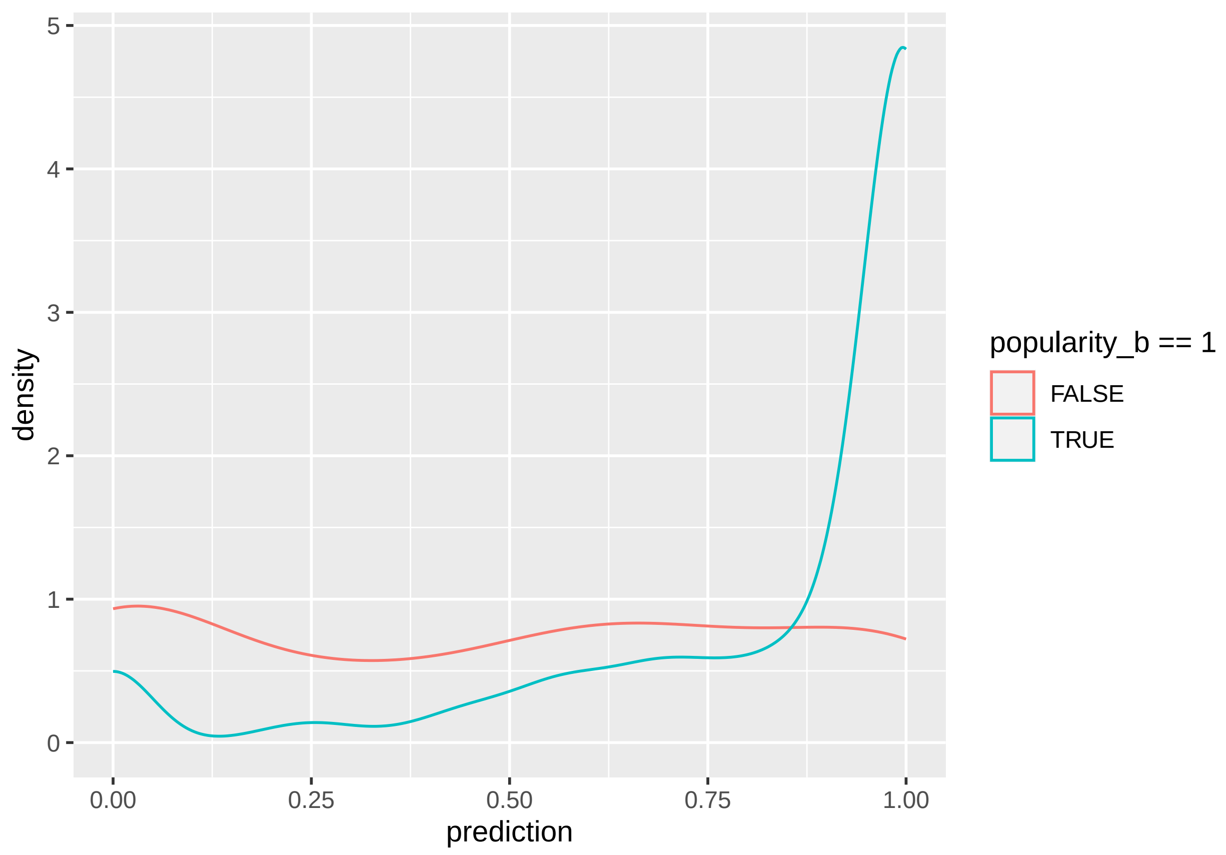
The accuracy here is 20.57% + 70.18% = 90.75% – slightly better than the first model! This seems to do a better job at predicting those that aren't popular.

Test Dataset Now let's see how this model does with the test dataset.

```r
df_test_new <- df_test
df_test_new$artist[which(!(df_test_new$artist %in% unique(df_train$artist)))] <- NA  # for cases where

df_test$prediction <- predict(binomial_regression, newdata = df_test_new, type = 'response')
ggplot(df_test, aes(x = prediction, color = popularity_b == 1)) + geom_density()
```

```
## Warning: Removed 157 rows containing non-finite values (`stat_density()`).
```



This chart looks a little less accurate than the training one did upon first glance.

```r
popularity_prediction_test <- table(df_test$popularity_b) |> prop.table()
popularity_prediction_test
```

```
##
##         0         1
## 0.2704508 0.7295492
```

Here, we can see that we would be correct 72.95% of the time if we just guessed a song was popular.

```r
df_test$prediction_class <- df_test$prediction > 0.5
tab_test <- table(df_test$prediction_class, df_test$popularity_b) %>%
    prop.table() %>% print()
```

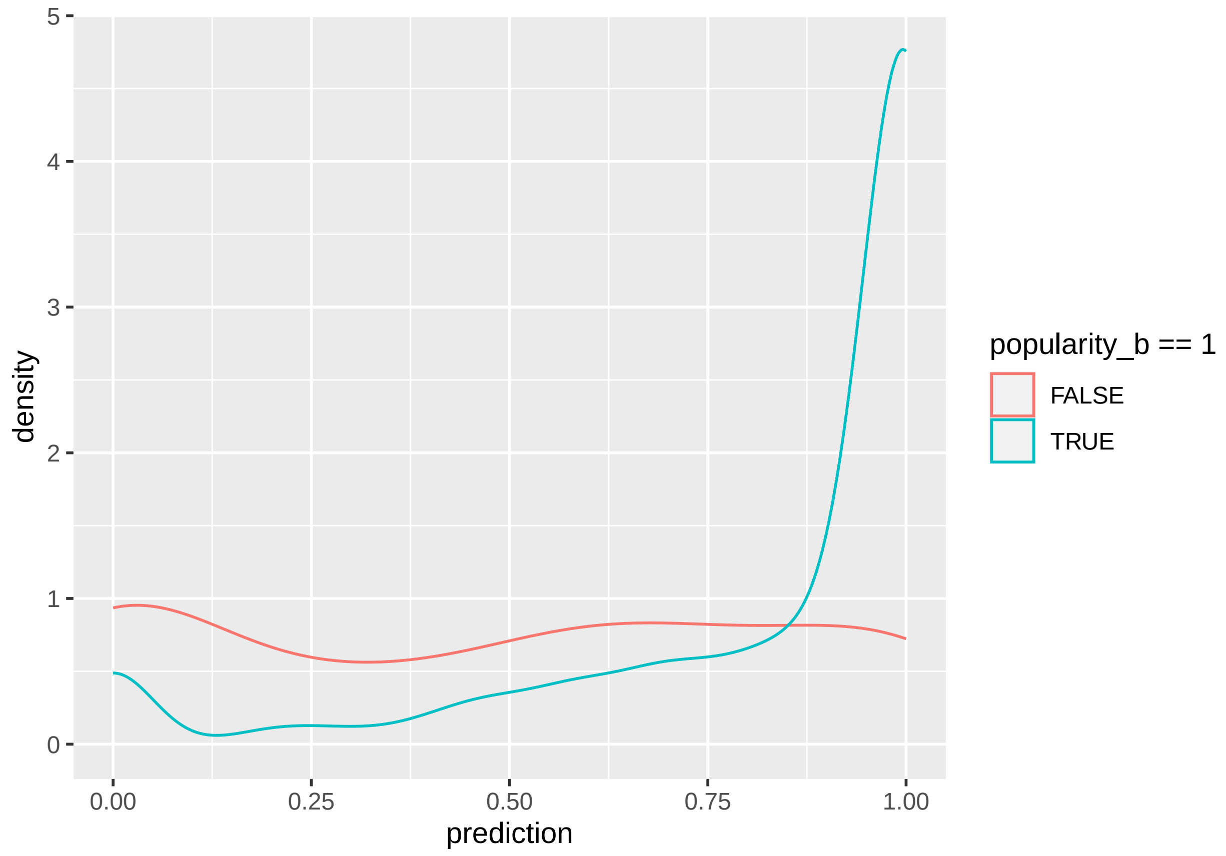
```
##
##                  0          1
##   FALSE 0.12217195 0.08823529
##   TRUE  0.12669683 0.66289593
```

12.22% + 66.29% = 78.51% – this model is ~5.5% more accurate than just guessing. Given there are NAs for this situation, it makes sense that the accuracy is lower than when compared to the training set's performance.

How does the second model we created look?

```r
df_test$prediction <- predict(binomial_regression_2, newdata = df_test_new, type = 'response')
ggplot(df_test, aes(x = prediction, color = popularity_b == 1)) + geom_density()
```

```
## Warning: Removed 157 rows containing non-finite values (‘stat_density()‘).
```



```r
df_test$prediction_class <- df_test$prediction > 0.5
tab_test <- table(df_test$prediction_class, df_test$popularity_b) %>%
    prop.table() %>% print()
```

```
##
##                  0          1
##   FALSE 0.11764706 0.09954751
##   TRUE  0.13122172 0.65158371
```

11.76% + 65.16% = 76.92% – this model does a little worse for our test dataset it seems, but still better than just guessing "yes" for everything.

Part 5 - Conclusion

Overall, the process to find whether there is a relationship between the various variables and popularity was made possible with a binomial regression. While only two fields were statistically significant in the first model (there may have been a few `artist` values that were also statistically significant, however looking at each one would take much more time) and the second model with removing of the less statistically significant variables did a little worse than the first for performance, but both models were still better than just blind guessing.

While there seems to be correlation between popularity and several of these variables, the only slight improvement the model gives seems to indicate that trying to engineer a "popular" song may not be possible with these metrics; I would only really rely on this to predict whether a song is popular, and even with that, it's not *that* much of an improvement overall.

Future iterations of this project could be to:

- Test different model types
- Try to predict `popularity` as a score rather than a boolean (attempted in Appendix A)
- Try to incorporate more fields (genre) or remove more
- Adjust some of the fields (log) to get them into a more normal distribution before adding them to the model

References

https://www.kaggle.com/datasets/iamsumat/spotify-top-2000s-mega-dataset

Appendix

Predicting Popularity as a Number Instead Without Categorical Variables Let's see what the model looks like with all of the variables.

```
regression <- lm(popularity ~ bpm + energy +
               danceability + loudness + valence +
                 duration + acousticness + speechiness,
             data = df_train)

summary(regression)
```

```
##
## Call:
## lm(formula = popularity ~ bpm + energy + danceability + loudness +
##     valence + duration + acousticness + speechiness, data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.987  -8.834   2.035  10.729  36.697
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.310322   3.963982  17.485  < 2e-16 ***
## bpm          -0.005929   0.013830  -0.429 0.668218
## energy       -0.126652   0.034497  -3.671 0.000250 ***
## danceability  0.114093   0.029639   3.849 0.000124 ***
## loudness      1.008937   0.162894   6.194 7.72e-10 ***
## valence       0.047301   0.020359   2.323 0.020305 *
## duration     -0.005238   0.004167  -1.257 0.208945
## acousticness -0.033681   0.017665  -1.907 0.056769 .
## speechiness   0.297578   0.093916   3.169 0.001565 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.04 on 1386 degrees of freedom
## Multiple R-squared:  0.07019,    Adjusted R-squared:  0.06483
## F-statistic: 13.08 on 8 and 1386 DF,  p-value: < 2.2e-16
```

Interestingly, it seems like a lot of these variables are statistically significant predictors (`energy`, `dancebaility`, `loudness`, `valence`, `acousticness`, `speechiness`).

The Adjusted r-squared is only 0.06483 though, signalling that this isn't a very strong model.