Lab Report 3

Yvone Yang, Natalie Tam, Fares Alhatabeh

1. Histogram & Density

```
## Read data here
spotify <- read.csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/202

# ...

## call tidyverse and any other library you'll need ( you can call later if you want )
library('tidyverse')
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## Set theme to something else (optional)
theme_set(theme_minimal())
```

Q1.

- By calling str(spotify), we can see that the dataset has 32833 observations.
  - There are 23 variables.
  - 13 are numerical (denoted by datatype num or int) and 10 are categorical (denoted by datatype char) variables.
  - There is not any categorical variable treated as a numerical variable in this dataset.
- Mode and Key columns are converted into factors because they are categorical variables treated as numeric variables.
- Numerical Variable: spotify$speechiness
- Categorical Variable: spotify$playlist_genre

```
str(spotify)
```

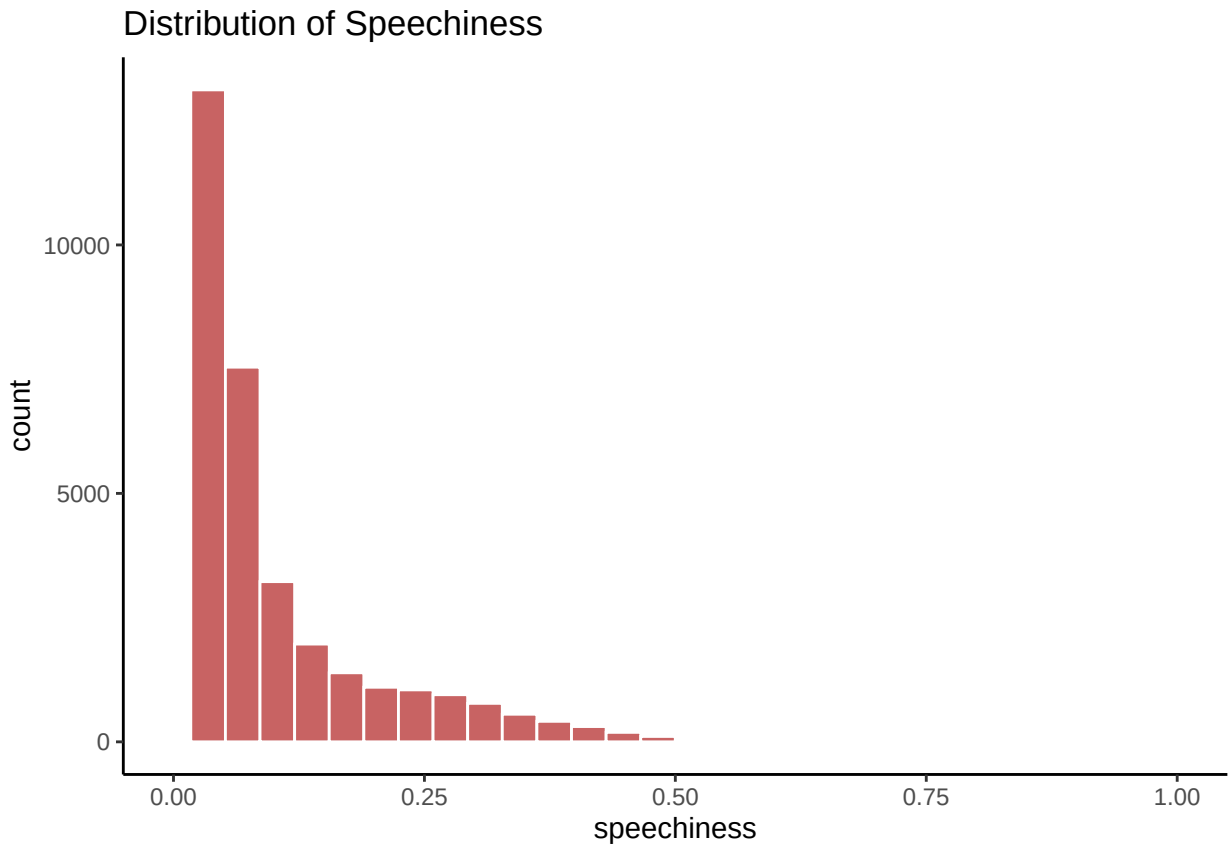
```
## 'data.frame':    32833 obs. of  23 variables:
##  $ track_id                : chr  "6f807x0ima9a1j3VPbc7VN" "0r7CVbZTWZgbTCYdfa2P31" "1z1Hg7Vb0A
##  $ track_name              : chr  "I Don't Care (with Justin Bieber) - Loud Luxury Remix" "Memo
##  $ track_artist            : chr  "Ed Sheeran" "Maroon 5" "Zara Larsson" "The Chainsmokers" ...
##  $ track_popularity        : int  66 67 70 60 69 67 62 69 68 67 ...
##  $ track_album_id          : chr  "2oCs0DGTsRO98Gh5ZSl2Cx" "63rPSO264uRjW1X5E6cWv6" "1HoSmj2eLc
##  $ track_album_name        : chr  "I Don't Care (with Justin Bieber) [Loud Luxury Remix]" "Memo
##  $ track_album_release_date: chr  "2019-06-14" "2019-12-13" "2019-07-05" "2019-07-19" ...
##  $ playlist_name           : chr  "Pop Remix" "Pop Remix" "Pop Remix" "Pop Remix" ...
##  $ playlist_id             : chr  "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1D
##  $ playlist_genre          : chr  "pop" "pop" "pop" "pop" ...
##  $ playlist_subgenre       : chr  "dance pop" "dance pop" "dance pop" "dance pop" ...
##  $ danceability            : num  0.748 0.726 0.675 0.718 0.65 0.675 0.449 0.542 0.594 0.642 ..
##  $ energy                  : num  0.916 0.815 0.931 0.93 0.833 0.919 0.856 0.903 0.935 0.818 ..
##  $ key                     : int  6 11 1 7 1 8 5 4 8 2 ...
##  $ loudness                : num  -2.63 -4.97 -3.43 -3.78 -4.67 ...
##  $ mode                    : int  1 1 0 1 1 1 0 0 1 1 ...
##  $ speechiness             : num  0.0583 0.0373 0.0742 0.102 0.0359 0.127 0.0623 0.0434 0.0565
##  $ acousticness            : num  0.102 0.0724 0.0794 0.0287 0.0803 0.0799 0.187 0.0335 0.0249
##  $ instrumentalness        : num  0.00 4.21e-03 2.33e-05 9.43e-06 0.00 0.00 0.00 4.83e-06 3.97e
##  $ liveness                : num  0.0653 0.357 0.11 0.204 0.0833 0.143 0.176 0.111 0.637 0.091
##  $ valence                 : num  0.518 0.693 0.613 0.277 0.725 0.585 0.152 0.367 0.366 0.59 ..
##  $ tempo                   : num  122 100 124 122 124 ...
##  $ duration_ms             : int  194754 162600 176616 169093 189052 163049 187675 207619 19318
```

```r
spotify$key <- as.factor(spotify$key)
spotify$mode <- as.factor(spotify$mode)

# Q1-1
ggplot(spotify, aes(x = speechiness)) +
  geom_histogram(fill=adjustcolor('firebrick',.7), color='white') +
  theme_classic() +
  xlim(c(0, 1)) +
  ggtitle('Distribution of Speechiness')
```

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

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



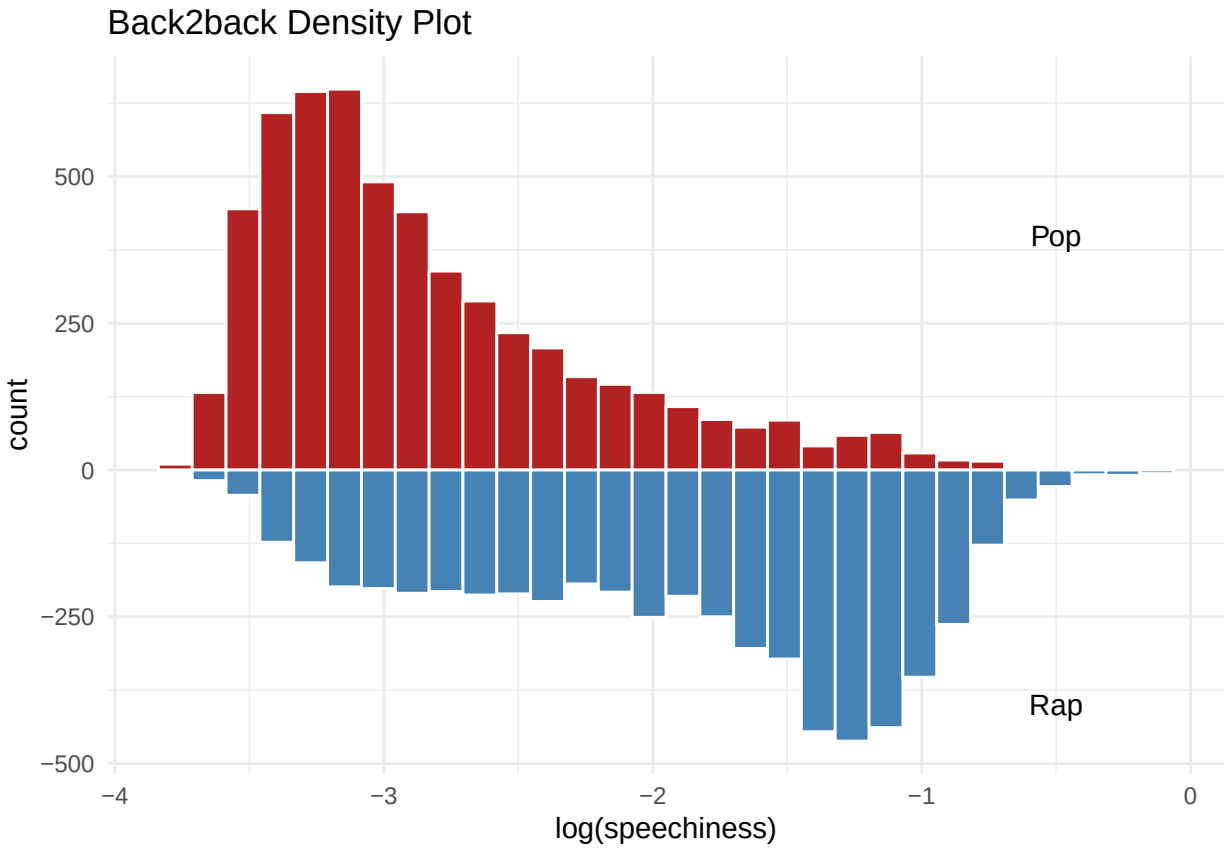
Your comment about the histogram here

By observing the speechiness histogram, the distribution is right skewed meaning most songs are not very speechy. There does not seem to be many songs that have higher speechiness score than 0.5. The density of songs exponentially decreases as speechiness increases.

```r
#Q1-2
ggplot(spotify) +
  geom_histogram(data= subset(spotify, playlist_genre == 'pop'),
```

```
                aes(x = log(speechiness), y = ..count..),
                fill="firebrick", colour='white') +
  geom_histogram(data= subset(spotify, playlist_genre == 'rap'),
                aes(x = log(speechiness), y = -..count..),
                fill= "steelblue", colour='white') +
  ggtitle('Back2back Density Plot') +
  annotate('text', x = -0.5, y = 400, label='Pop') +
  annotate('text', x = -0.5, y = -400, label='Rap')
```

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



Your comment about moments here

Q1-2 The values speechiness is on a logarithmic scale for better readability.

By comparing the density of speechiness between songs from the pop and rap genre, we see that pop songs tend to have lower speechiness score while rap songs have higher speechiness scores.

The density of Pop songs are right skewed and the density of Rap songs are left skewed.

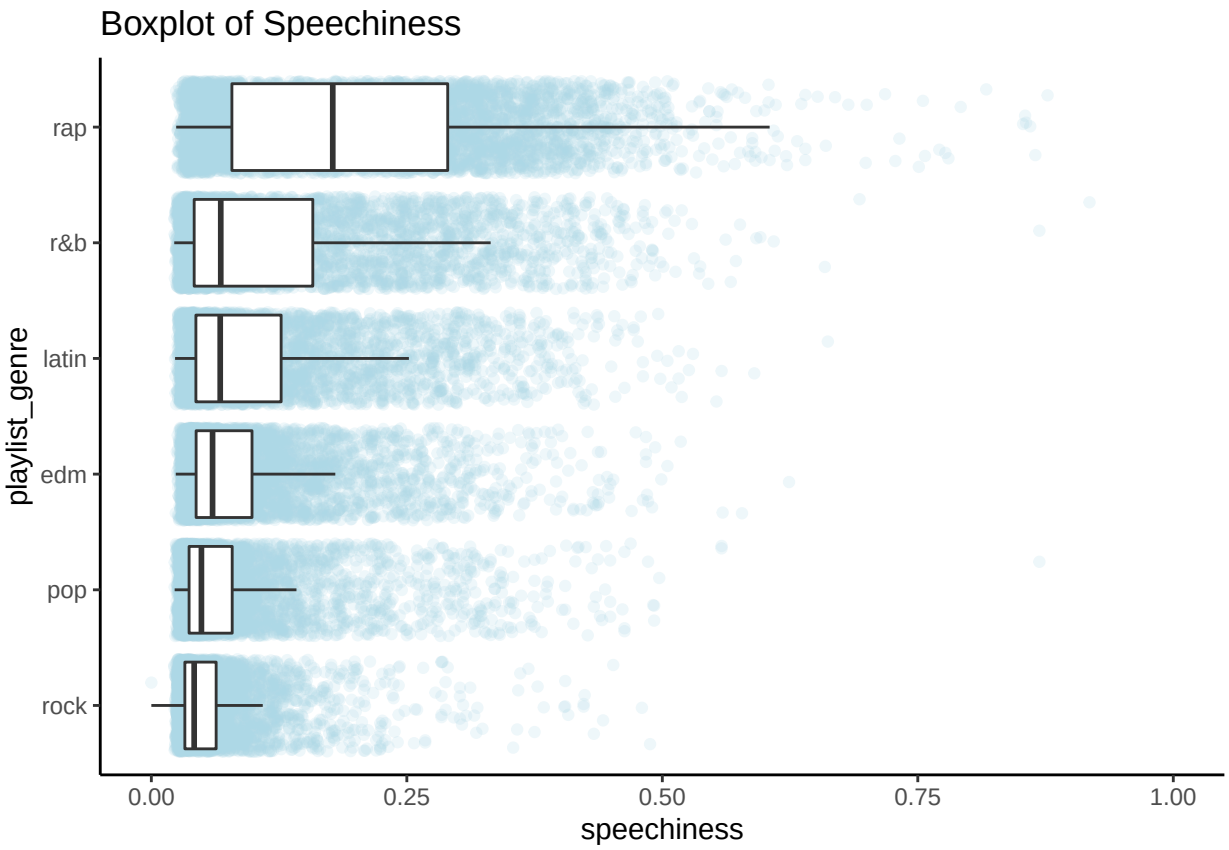
This makes sense because rappers tend to speak a lot of words very quickly in comparision to pop artists.

2. Boxplot

```
#Q2-1
categories <- unique(spotify$playlist_genre)
medians <- sapply(categories, function(z)
  median(spotify$speechiness[spotify$playlist_genre == z]))
ord <- order(medians)

spotify$playlist_genre <- factor(spotify$playlist_genre, levels = categories[ord])

ggplot(spotify, aes(x = speechiness, y = playlist_genre)) +
  geom_jitter(alpha=0.2, color = "light blue") +
  geom_boxplot(outlier.alpha = 0) +
  theme_classic() + xlim(c(0, 1)) +
  ggtitle('Boxplot of Speechiness')
```



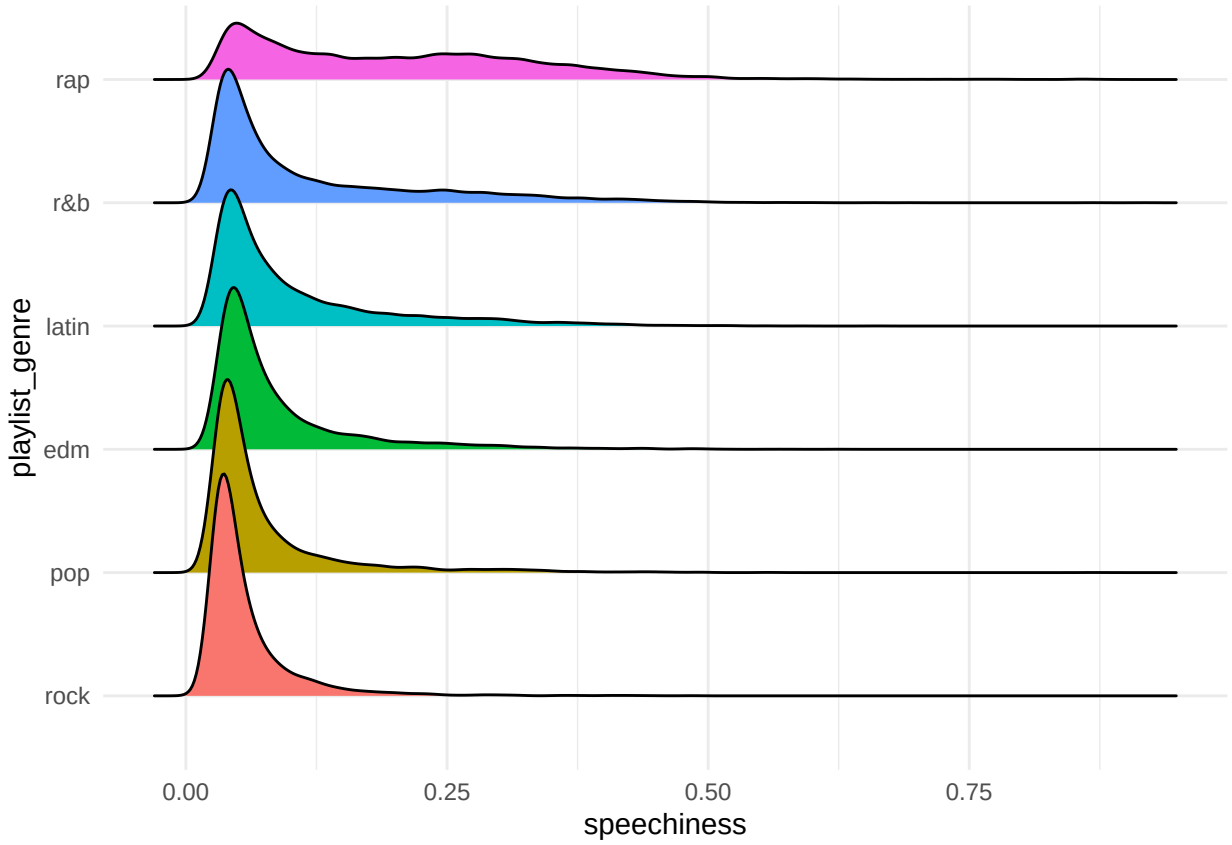

```
#Q2-2
library('ggridges')
```

```
## Warning: package 'ggridges' was built under R version 4.0.5
```

```
ggplot(spotify, aes(y=playlist_genre, x=speechiness, fill=playlist_genre)) +
  geom_density_ridges() +
```

```
  theme_minimal() +
  theme(legend.position = 'none')
```

```
## Picking joint bandwidth of 0.0101
```



Your comment here

Songs in the rap playlist genre tend to have a wide spread of speechiness and the highest average speechiness.

All other categories (everything but rap) have relatively similar/close in terms of speechiness, with most hovering around a score of 0.05.

Sorting by the speechiness medium, we see that Rap caegory has the highest speechiness medium value, followed by r&b, then latin, then edm, then pop, then rock.

Rock song speechiness has low variability and rap song speechiness has very high variability (as observed by the length of the boxplot box).

3. Aggregating Data

```
spotify %>%

  select(playlist_genre, key)%>%
  group_by(playlist_genre, key)%>%
  arrange(key)%>%
  summarize(Number_of_Songs = n())
```

```
## `summarise()` has grouped output by 'playlist_genre'. You can override using the `.groups` argu
```

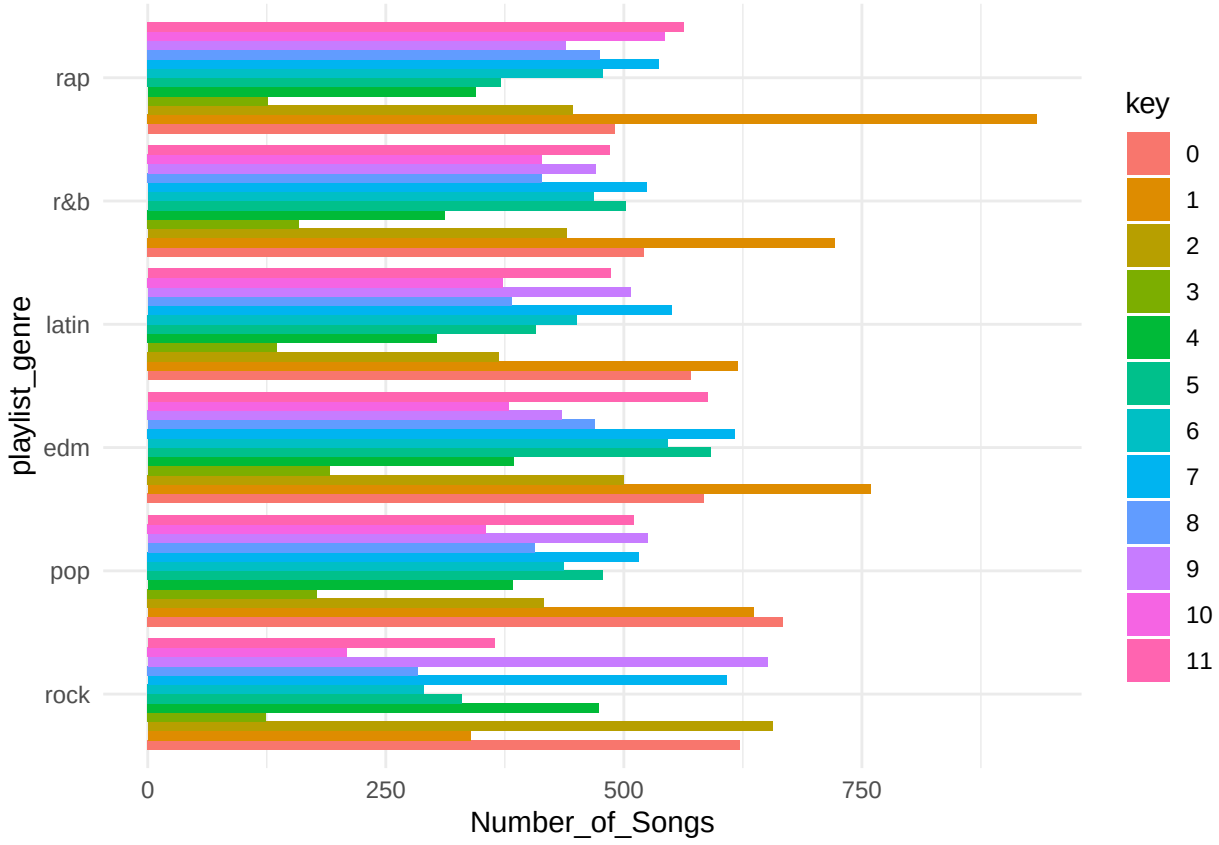
```
## # A tibble: 72 x 3
## # Groups:   playlist_genre [6]
##    playlist_genre key   Number_of_Songs
##    <fct>          <fct>           <int>
##  1 rock           0                 622
##  2 rock           1                 339
##  3 rock           2                 656
##  4 rock           3                 124
##  5 rock           4                 474
##  6 rock           5                 330
##  7 rock           6                 290
##  8 rock           7                 608
##  9 rock           8                 284
## 10 rock           9                 651
## # ... with 62 more rows
```

4. Visualizing Categorical Variables

```
count = spotify%>%
  select(playlist_genre, key)%>%
  group_by(playlist_genre, key)%>%
  summarize(Number_of_Songs = n())%>%
  arrange(key)
```

```
## `summarise()` has grouped output by 'playlist_genre'. You can override using the `.groups` argu
```

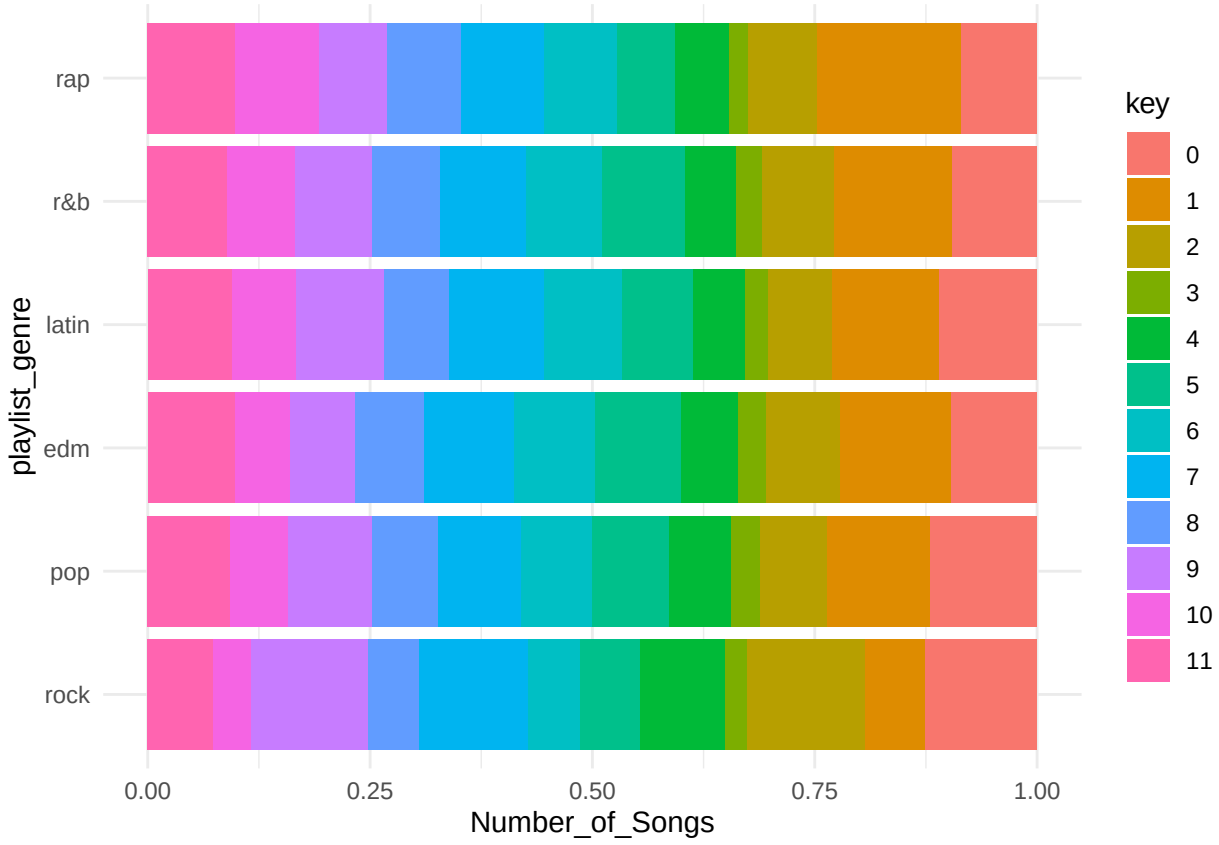
```
ggplot(count, aes(y = playlist_genre, x = Number_of_Songs, fill = key))+
  geom_bar(stat = "identity", position='dodge')
```




```
count = spotify%>%
  select(playlist_genre, key)%>%
  group_by(playlist_genre, key)%>%
  summarize(Number_of_Songs = n())%>%
  arrange(key)
```

```
## `summarise()` has grouped output by 'playlist_genre'. You can override using the `.groups` argu
```

```
ggplot(count, aes(y = playlist_genre, x = Number_of_Songs, fill = key))+
  geom_bar(stat = "identity", position ='fill')
```



5. Other Visuals

```
playlistGenres <- group_by(spotify, playlist_genre, mode) %>% summarise(num = n())
```

```
## `summarise()` has grouped output by 'playlist_genre'. You can override using the `.groups` argu
```

```
ggplot(playlistGenres) +
      geom_col(aes(x = 1, y = num, fill = mode), position = "fill") +
      facet_wrap( ~ playlist_genre) +
      coord_polar(theta = "y") +
      scale_fill_manual(name="", values = c("firebrick","#007ea1"))
```