ECON 3170 – Term paper

Intro

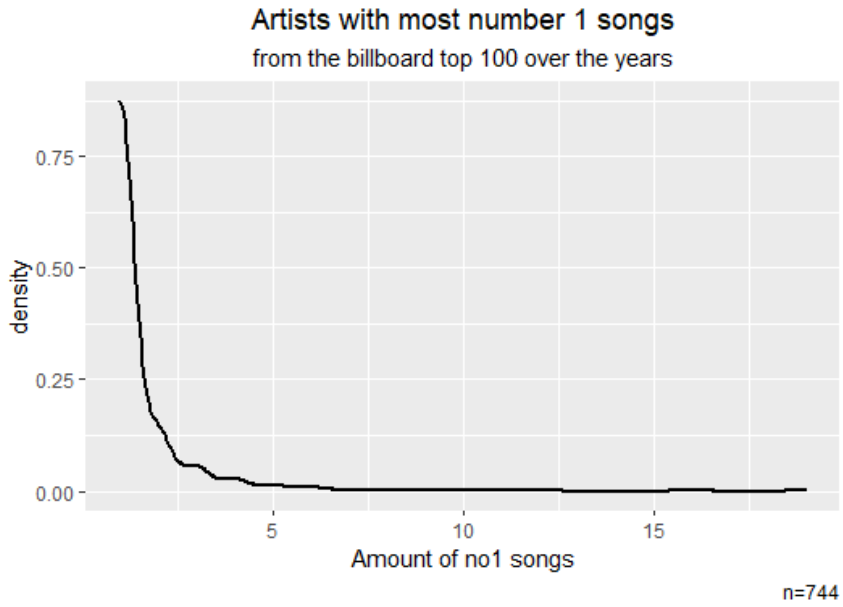
In intro, in the code, I defined everything so I can easy access it later. After I downloaded the *billboard* and *audio_features* data. I merged them together, and merged the common columns, and called that variable "*songs*". I also changed the *week_id* to *year*, because its more relevant later on when I look at yearly progress. This variable will be useful when I look at weekly positioning on the list, so it will have song duplicates. Therefore I made a second variable called *unique_songs,* containing all the variables like in "songs", just without song duplicates. I made it because it's easier later, if I want to analyze the songs itself.

In all the plots, I have n = sample size

Section 1

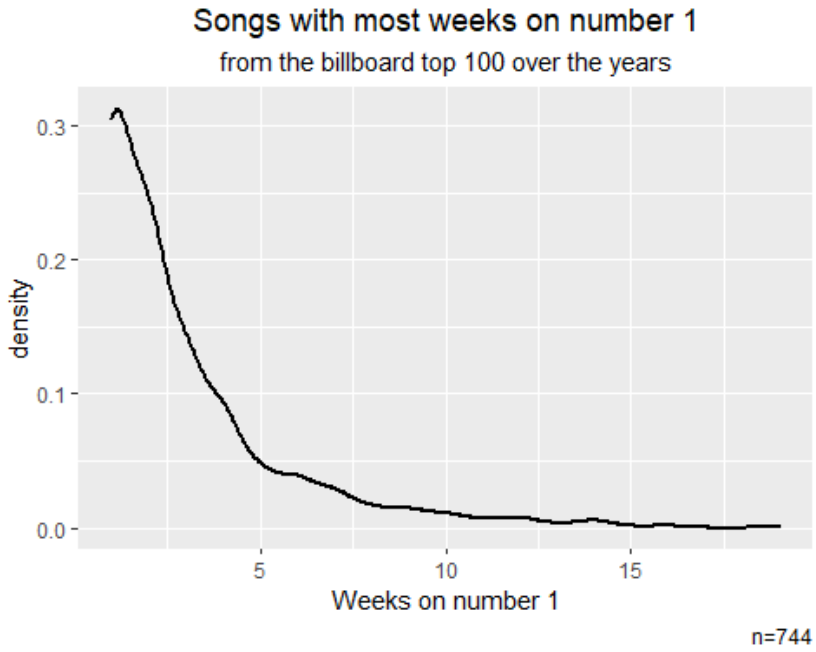
In this section, I started relatively easy. I wanted to make a density curve for artists that have had the most songs on number 1. Just to visualize how rare it is for an artist to get multiple number 1 songs. I made a variable called *no1.songs,* where I selected *peak_position, song* and *performer* from *songs.* Then I filtered it, such that I only get the rows that contains *peak_position* = 1, and distinct *performer* and *song*. Now I have a data frame containing all the song peaking at number 1, with the artist. All I had left now was to count the performers and arrange them.

For the plotting part, I tried to keep it clean and simple. I made a density graph. Where I edited the text and adjusted the position. And it looked like this:



Section 2

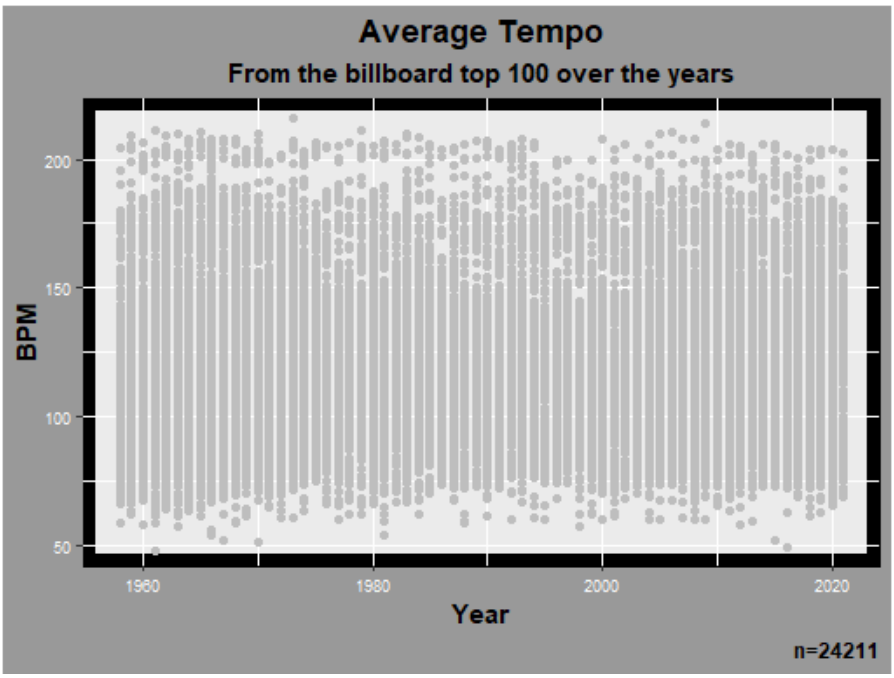
This section is very similar to section 1, so there is not so much to say. The only difference is that it is a density graph showing songs with most weeks on number 1. Instead of artists with most number 1 songs. The graph is also clean and simple.



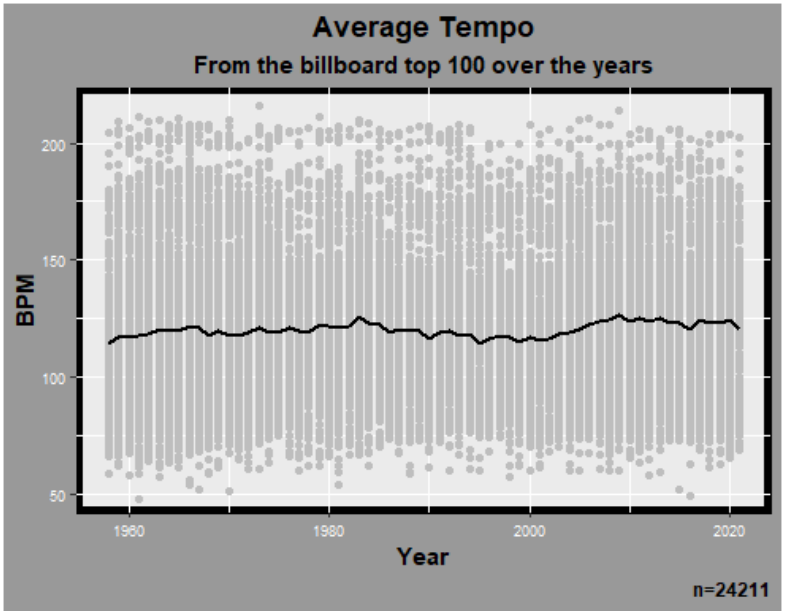
Section 3

This is the section when it's starting to really get interesting. I wanted to visualize how the tempo of songs have changed over the years. Firstly, I selected *song_id*, *year* and *tempo* from *unique_songs.* There is obviously not data for every song on this list from 1958 to 2021, so I removed the rows that did not have data, with *drop_na()*. I also made a variable called *tempo_average* that contained the average tempo for every year.

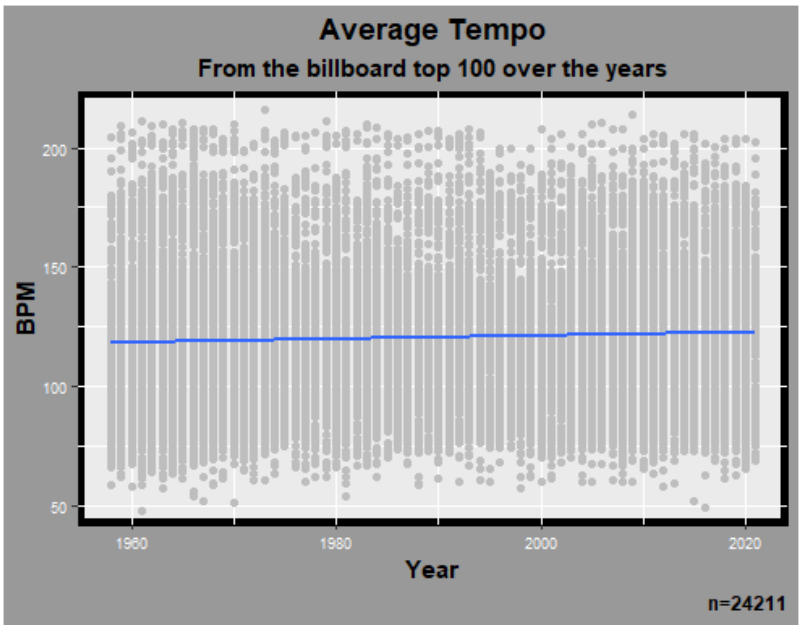
For the plotting part, I made a variable called *tempo_plot*. In this variable, I made points for every year, which were distributed according to the tempo. Now there were a lot of emptiness in the plot, so I used *coord_cartesian(ylim=c(50,215))*, to zoom in on the plot to remove the empty spaces, and make it look better. Now the customizing is what's left. I did the same as I did in section 2 and section 3. Added text and adjusted its position. But here I also customized the colors, on both the text and the background. I tried to make it pleasing to look at. Here is the result:



When I looked at this, it didn't quite look right. The reason why I put everything in a variable, was because I wanted to add two different lines on top of this plot separately. Firstly, I added a line showing the average for every year, like this:



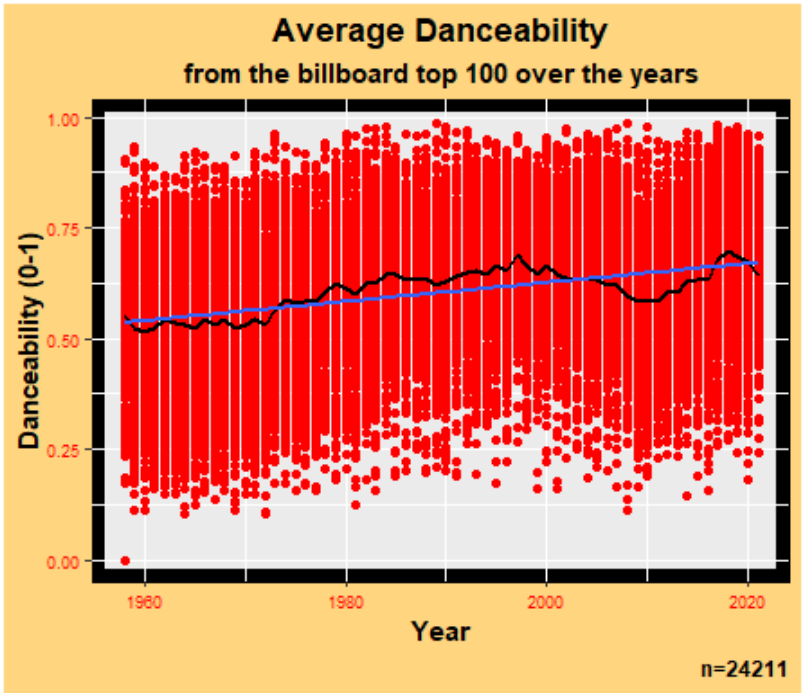
Now I think that the graph looks better, but its still hard to see if the tempo has decreased or increased from 1958 to 2021. We can see that the tempo was high in the 80s, then it fell down, and it rose again during the 2000s. But is the tempo in general higher or lower than before? I therefor plotted a regression line showing the matter:



Now we can see that the regression line in slightly tilted upwards, hence the tempo has in general increased from 1958 to 2021.

Section 4

This section is very similar to section 3. But here I looked at danceability. I tried to make the plot more colorful to match the dance theme. This is how I it turned out:



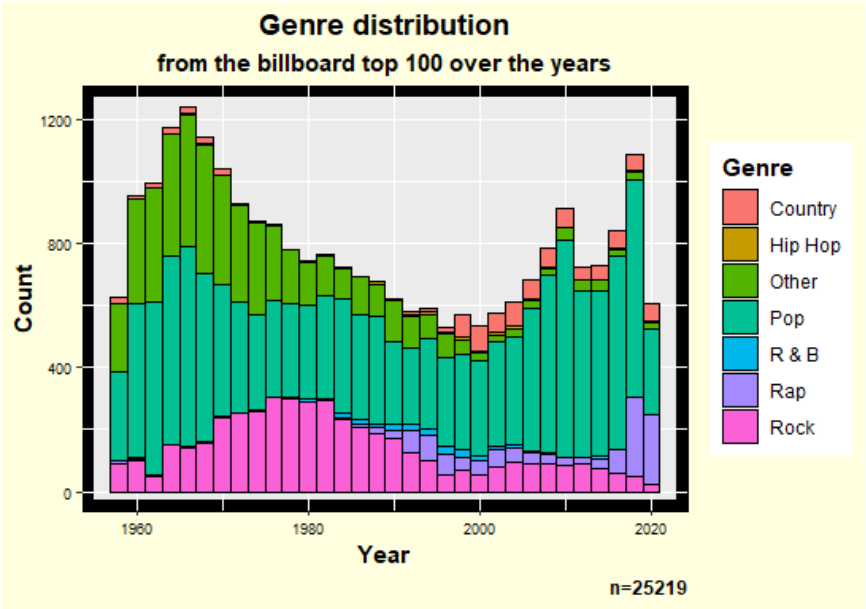
Here I decided to plot the average line and the line regression in the same plot. The reason for that is because since the average line is varying a lot more than the average tempo, another line on top will still look good. We can see on the line regression that danceability also has increased since 1958, but it can change a lot from decade to decade

Section 5

In this section I wanted to show the distribution of genres from 1958 to 2021. I started by define a variable called *songs_genre*. Where I selected columns from *unique_songs*, and filtered out the songs that didn't have any genre assigned. (This *songs_genre* variable will also be useful for section 7). As I stand now, songs can have multiple genres. I defined a function that I can extract the first genre from the songs. I did that with main genres, like pop, rap and rock. And not subgenres. One problem I faced was that some songs had a list of assign genres, and the function extracted the first genre in the list every time. So for example can a list of genres look like this : [____rock, ___pop, pop___, pop___]. This should then be considered a pop song, but since the first genre was rock, the function would extract this

genre instead. Since the sample is so big, I decided to ignore this problem. I defined a list called *main_genre*, which contained all the main genres for the song, and I added that as a column in *songs_genre*. I made another variable called *songs_genre0*, to make this variable I selected *year* and *Genre* from *songs_genre*.

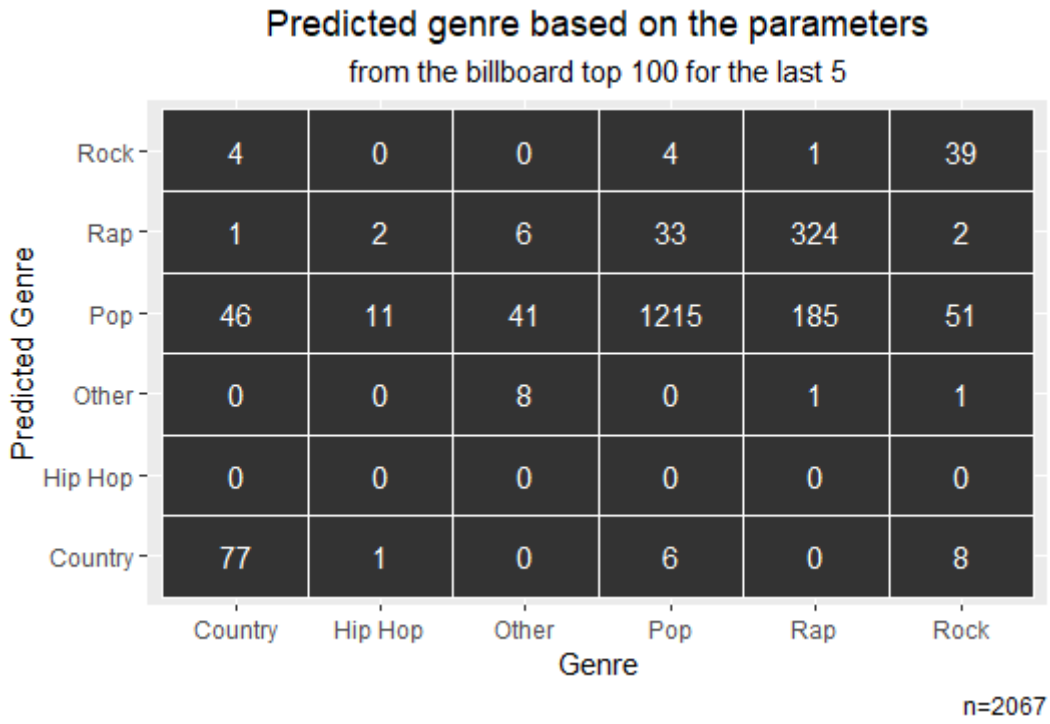
I made a histogram, and added *fill = Genre*, so that I can see the different genres on the plot, and a black color to see the outline. Then I customized it like always, added text and color etc.. It turned out like this, which I'm very happy with:



This plot gives me a lot of information. We can for instance see that rock peaked in the 70-80s. And that rap started in the 90s and got bigger and bigger.

Section 6

This was the section I struggled the most with. I tried to predict the genre of song based on all the metrics/columns in *songs.last5*. I did it for the last 5 years, for consistency, because the genres evolved over time. For *songs.last5* variable, I used *songs_genre*, and filtered it for the last 5 years, where I selected all the elements I that I was going to use to predict. I did the normal procedure, where I made the recipe, sat the model and workflow. The model I used was from *nearest_neighbor()* from the *tidymodels* pack. I used that to define a model that uses similar point to predict new samples. I made a new variable called *result*, where I fitted the *songs.last5* to the workflow. For the data I needed to plot, *result.plot*, I finally predicted new data from *songs.last5* based on the result I got. This is how the plot turned out:



We can see what got predicted based on the actual genre. For instance, if we look at country. Based on 128 country songs, 4 predictions thought it was rock,1 rap, 46 pop and 77 country. Which is a decent result, the highest prediction was country. I tried to make percentages instead of number, because some genres has more songs than other, but I did not know how to do it. Like always, I also customized it to make it look good.

CODE:

```
library(tidytuesdayR)

library(tidyverse)

library(tidymodels)


# intro----

tuesdata <- tidytuesdayR::tt_load(2021, week = 38)


billboard <- tuesdata$billboard

audio_features <- tuesdata$audio_features
```

```
#Merged billboard and audio_features
songs <- tibble(merge(audio_features,billboard,
                      c("song_id","song","performer"))) %>%
  rename(year=week_id)

#Converting week_id to year by remove the first 4 signs
songs$year <- as.numeric(str_extract_all(songs$year, "\\d{4}$"))

#Removing songs dupclicate
unique_songs <- songs %>% distinct(song_id, .keep_all = TRUE)



# Section 1 ----

# Defining a variable with artists with
# the most number 1 songs, and sorted them
no1.songs <- songs %>%
  select(peak_position,song, performer) %>%
  filter(peak_position==1) %>%
  distinct(performer,song) %>%
  count(performer) %>%
  arrange(desc(n))
```

```
#plotted no1.songs, and customized plot
ggplot(data=no1.songs, aes(x=n))+
  geom_density(size=1)+
  labs(title="Artists with most number 1 songs",
       x="Amount of no1 songs",
       caption="n=744",
       subtitle="from the billboard top 100 over the years")+
  theme(plot.title = element_text(hjust=0.5),
        plot.subtitle = element_text(hjust=0.5))



# Section 2  ----

# Defining a variable with songs with
# most instances on number 1, and sorted them
songs.most.weeks.no1 <- billboard %>%
  select(week_position, song) %>%
  filter(week_position==1) %>%
  count(song) %>%
  arrange(desc(n))

#plotted songs.most.weeks.no1, and customized plot
ggplot(data=songs.most.weeks.no1, aes(x=n))+
  geom_density(size=1)+
```

```
  labs(title="Songs with most weeks on number 1",
       x="Weeks on number 1",
       caption="n=744",
       subtitle="from the billboard top 100 over the years")+
  theme(plot.title = element_text(hjust=0.5),
        plot.subtitle = element_text(hjust=0.5))




# Section 3 ----


# Selected song_id, year and tempo
# to a new variable
tempo <- unique_songs %>%
  select(song_id,year,tempo) %>%
  drop_na()


# Made a variable constisting of
# the average tempo for each year
tempo_average <- tempo %>% select(year,tempo) %>%
  group_by(year) %>% summarise(
  tempo = mean(tempo))


# Made a variable containing a
```

```
# plot of tempo, and customized
# the plot.
tempo_plot <- ggplot(data=tempo,aes(x=year,y=tempo)) +
  geom_point(data=tempo, color="grey") +
  coord_cartesian(ylim=c(50,215)) +
  labs(title="Average Tempo",
       subtitle="From the billboard top 100 over the years",
       x="Year",
       y="BPM",
       caption="n=24211") +
  theme(title = element_text(color="Black",face ="bold"),
       axis.text.y = element_text(color="White",size=7),
       axis.text.x = element_text(color="White",size=7),
       plot.title = element_text(hjust = 0.5),
       plot.subtitle = element_text(hjust = 0.5),
       plot.background = element_rect(fill="#999999"),
       panel.background = element_rect(color="Black",size=3))

# Added line to the plot which consisted
# of average tempo for each year
tempo_plot + geom_line(data=tempo_average,size=1)

# Added fitting linear regression line
# for the tempo plot
```

```
tempo_plot + geom_smooth(method="lm")
```

```
# Section 4----

# Selected song_id, year and danceability
# from unique songs, and put it in a variable
danceability <- unique_songs %>%
  select(song_id,year,danceability) %>%
  drop_na()

# Made a variable constisting of
# the average danceability for each year
danceability_average <- danceability %>%
  select(year,danceability) %>%
  group_by(year) %>%
  summarise(danceability = mean(danceability))

# plotted "danceability", and added both
# the average line, and the linear regression line.
# Also customized the plot
ggplot(data=danceability,aes(x=year,y=danceability))+
```

```
  geom_point(data=danceability,color="red")+
  labs(title="Average Danceability",
       subtitle="from the billboard top 100 over the years",
       x="Year",
       y="Danceability (0-1)",
       caption="n=24211")+
  theme(title = element_text(color="Black",face ="bold"),
        axis.text.y = element_text(color="red",size=7),
        axis.text.x = element_text(color="red",size=7),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        plot.background = element_rect(fill="#FFD580"),
        panel.background = element_rect(color="Black",size=5))+
        geom_line(data=danceability_average,size=1)+
        geom_smooth(method="lm")
```

```
# Section 5----

# Making a variable ready for genre-based manipulation.
# Selected useful parameters from "unique_songs"
```

```
songs_genre <- unique_songs %>%
  select(spotify_genre,year,danceability, energy, key, loudness,
         mode, speechiness, acousticness, instrumentalness, liveness, valence,
         tempo, spotify_track_popularity) %>%
  filter(spotify_genre!="[]")


# Made a function that finds genres
# and return main genres
find_genre <- function(word){
  if (is.na(word)){
    return(NA)
  }
  if (grepl("pop", word, fixed = TRUE)){
    return("Pop")
  }
  if (grepl("rap", word, fixed = TRUE)){
    return("Rap")
  }
  if (grepl("rock", word, fixed = TRUE)){
    return("Rock")
  }
  if (grepl("country", word, fixed = TRUE)){
    return("Country")
  }
```

```
  if (grepl("r&b", word, fixed = TRUE)){
    return("R & B")
  }
  if (grepl("hip hop", word, fixed = TRUE)){
    return("Hip Hop")
  }
  else {
    return("Other")
  }
}

# Used the function to grab the first genres
# from "songs_genre$spotify_genre
main_genre <- unname(sapply(songs_genre$spotify_genre, function(x) find_genre(x)))

# Added main_genre as a column in "songs_genre"
songs_genre$Genre <- main_genre

# Made a new variable that selects
# year and Genre from "song_genre"
songs_genre0 <- songs_genre %>%
  select(year,Genre)

# plotted "songs_genre0", and customized it
```

```
ggplot(songs_genre0, aes(x=year))+
  geom_histogram(binwidth=2, aes(fill=Genre), colour="Black")+
  labs(title="Genre distribution",
       subtitle="from the billboard top 100 over the years",
       x="Year",
       y="Count",
       caption="n=25219")+
  theme(title = element_text(color="Black",face ="bold"),
        axis.text.y = element_text(color="Black",size=7),
        axis.text.x = element_text(color="Black",size=7),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        plot.background = element_rect(fill="#FFFFE0"),
        panel.background = element_rect(color="Black",size=5))
```

```
# Section 6----

# Made a variable for the last 5 years
```

```
# and selected important parameters
songs.last5 <- songs_genre %>%
  filter(year==2021|year==2020|year==2019|year==2018|year==2017) %>%
  select(danceability, energy, key, loudness, mode,
         speechiness, acousticness, instrumentalness, liveness, valence,
         tempo, spotify_track_popularity, Genre) %>%
  drop_na()

# Made recipe
recipe <- recipe(Genre~., data=songs.last5)

# Set a K-nearest neighbor model
set_model <- nearest_neighbor(
  neighbors = 7,
  weight_func = "gaussian") %>%
  set_mode("classification") %>%
  set_engine("kknn")

# Added a workflow
work_flow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(set_model)

# fit "songs.last5" to the workflow
```

```
# and added it to a variable
result <- work_flow %>% fit(data=songs.last5)


# Made a variable for the plot,
# where I predict new data based on "songs.last 5".
result.plot <-  result %>% predict(new_data = songs.last5) %>%
  cbind(songs.last5) %>%
  select(.pred_class, Genre) %>% table()
result.plot <- data.frame(result.plot)


# Plotted "result.plot", and customized the plot
ggplot(data=result.plot, mapping = aes(y = .pred_class,
                    x = Genre)) +
  geom_tile(color="white") +
  geom_text(aes(label = sprintf("%1.0f", Freq)), colour="white")+
  labs(title="Predicted genre based on the parameters",
       subtitle="from the billboard top 100 for the last 5 years",
       y="Predicted Genre",
       caption="n=2067")+
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```