Spotify Data

Eddie Coda

2023-09-06

Sounds & Statistics: My Spotify Study. A Data-Driven Exploration

> Have you ever stopped to think about your relationship with music? How it changes over time, and what it says about you? This guide is for beginners, made by a beginner.

In this analysis, I'll take you on a journey through my musical world. Using my Spotify streaming history, I'll show what I listen to and project some insight into my music world. Additionally I'll show you how easy it is to get your data from Spotify and create your own vibrant visuals. We'll learn essential data preprocessing and transformation techniques, and create stunning visualizations.

Here's a quick roadmap of everything I will go through:

1. Document Metadata: Download your data
2. Import Data into Data Frame & Mutate Data: Walkthrough of data import and transformation.
3. Insights into Songs Played: Analysis and insights into song play counts and patterns.
4. Creating a Weekly Listening Activity Heatmap
5. Visualizing Listening Activity by Artist
6. Conclusion & Further Exploration

Getting Started:

Before we hit the play button, there are a few things you'll need to set up. Here's what you'll need to embark on this musical data adventure:

- R and RStudio: If you're new to R, don't worry! You can find installation guides and beginner tutorials here.
- Spotify Data: You can use your personal Spotify data or explore public datasets. Learn how to access your Spotify data here.
- Required Libraries: This project uses libraries like `ggplot2`, `lubridate`, `jsonlite`, and more. You'll find installation instructions within the guide.
- A Sense of Curiosity: Bring your passion for music and an eagerness to explore data. There's a symphony of insights waiting for you!

Downloading Your Spotify Data

Before we can begin our musical journey, we first need our map; our Spotify data. This data is a record of every step we've taken in our musical exploration, and it's surprisingly easy to obtain.

To download your Spotify data, simply log into your Spotify account and navigate to the settings page. There, under the "Your account" tab, you'll find an option called "Privacy settings". Scroll down, and you'll see a tool labeled "Download your data". Follow the steps listed there, and Spotify will prepare a zip file containing your data in JSON format.

Add Gif

Spotify says to allow 30 days for them to prepare and send your file, but in my case, I received my data after just 5 days. The zip file contained several files, but for this analysis, I focused on the streaming history data, which is titled "StreamingHistory0.json" and so on. The number of streaming history files you have will depend on your Spotify usage.

Add image of data

Import Data into Data Frame & Mutate Data: Walkthrough of data import and transformation.

With our Spotify data in hand, it's time to set the stage for our analysis. This involves importing our data into R and setting up our data frame.

First, we need to install some packages that will be used in the analysis. These packages include `jsonlite` for handling JSON data, `lubridate` for dealing with dates and times, `gghighlight` for creating beautiful visualizations, `spotifyr` for accessing Spotify's API, and several others.

```r
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
# Import packages
#install.packages('gghighlight')
#install.packages('rjson')
library(purrr)
library(rjson)
library(jsonlite)
library(lubridate)
library(ggplot2)
library(gghighlight)
library(spotifyr)
library(tidyverse)
library(knitr)
library(ggplot2)
#library(plotly)
```

Import data into dataframe && mutate data.

Once the packages are installed, we can import our Spotify data into an R data frame. We do this using the `fromJSON()` function from the jsonlite library, which allows us to read our JSON files into R. If we have multiple streaming history files, we can use the `rbind()` function to concatenate them into one data frame.

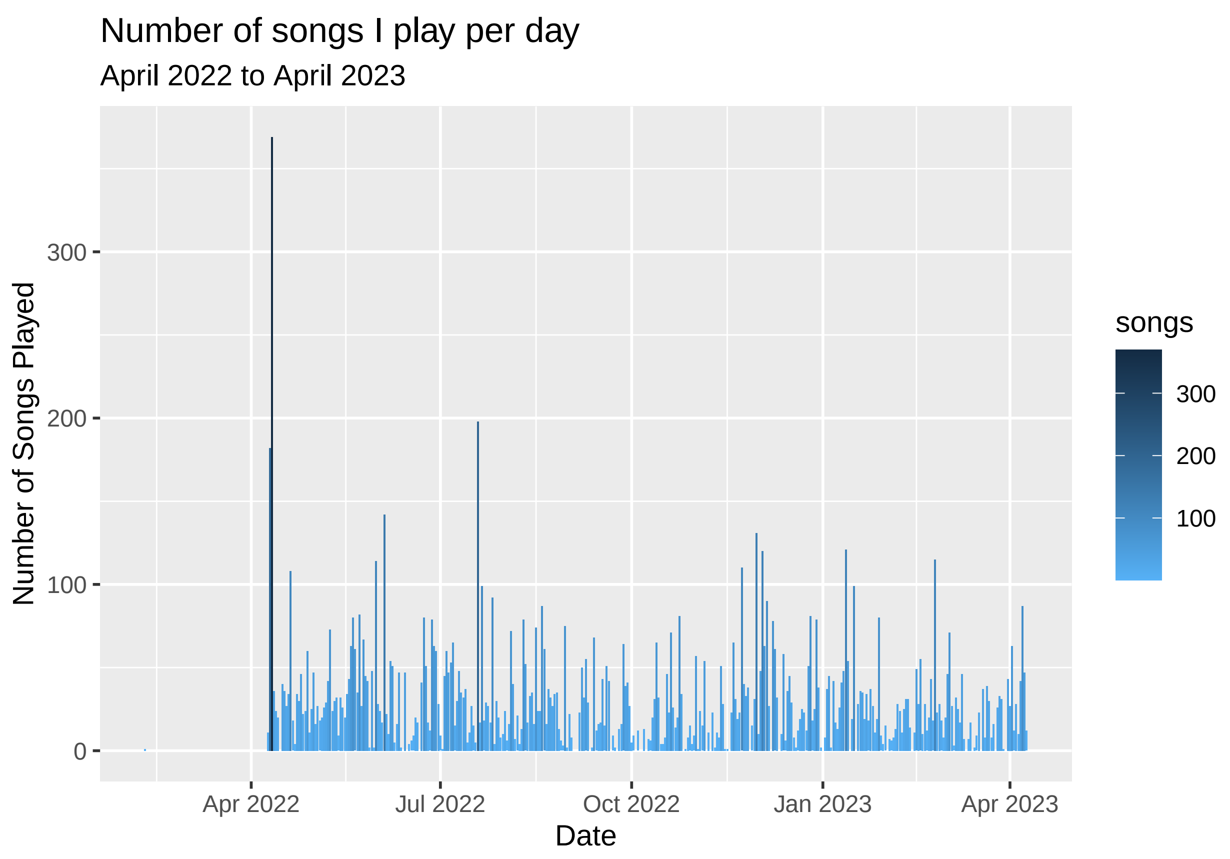
Finally, we reformat the date-time data using the `mutate()` function to create date and minute variables. This will make our analysis much easier down the line.

```r
# Export my Spotify data to an R dataframe

streamHistory1 <-
  jsonlite::read_json(
    "C:/Users/eddie/OneDrive/Desktop/my_spotify_data/MyData/StreamingHistory0.json",
    flatten = TRUE
  )
streamHistory2 <-
  jsonlite::read_json(
    "C:/Users/eddie/OneDrive/Desktop/my_spotify_data/MyData/StreamingHistory1.json",
    flatten = TRUE
  )

streamHistory1 <- do.call(rbind, lapply(streamHistory1, data.frame))
streamHistory2 <- do.call(rbind, lapply(streamHistory2, data.frame))

# str(streamHistory1)
# str(streamHistory2)

streamingData <- rbind(streamHistory1, streamHistory2)
streamingData <- streamingData %>%
  as_tibble() %>%
  mutate_at("endTime", ymd_hm) %>%
  mutate(endTime = endTime - hours(6)) %>%
  mutate(date = floor_date(endTime, "day") %>% as_date, minutes = msPlayed / 60000)
```

Insights into Songs Played: Analysis and insights into song play counts and patterns.

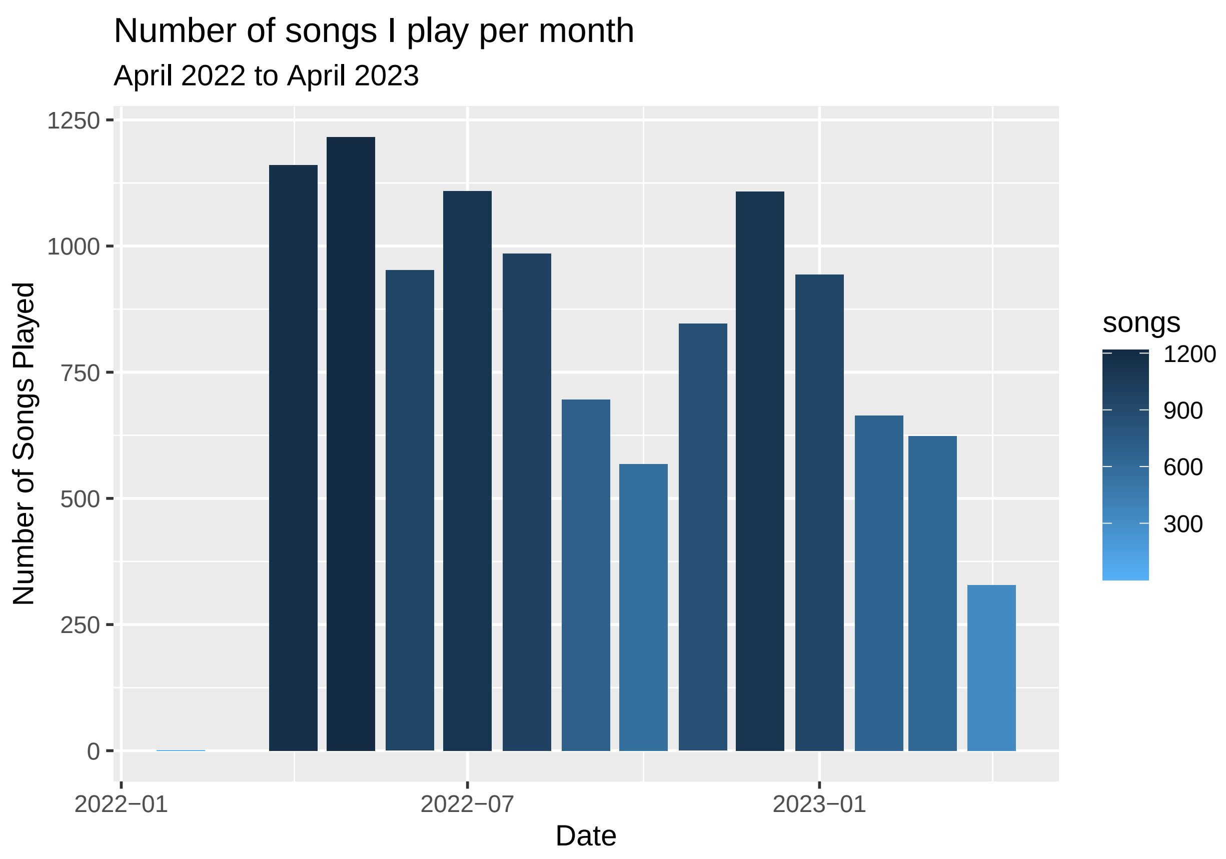
Music is my motivation. But how often do I actually listen to music? To find out, I created a line chart showing the number of songs I played each day.

```r
# Number of songs I play per day : bar chart
songsByDay <- streamingData %>%
  filter(msPlayed >= 1000) %>%
  group_by(date) %>%
  group_by(date = floor_date(date, "day")) %>%
  summarize(songs = n()) %>%
  arrange(date) %>%
  ggplot(aes(x = date, y = songs)) +
  geom_col(aes(fill = songs)) +
  scale_fill_gradient(high = "#132B43", low = "#56B1F7") +
  labs(x= "Date", y= "Number of Songs Played") +
  ggtitle("Number of songs I play per day", "April 2022 to April 2023")
songsByDay
```



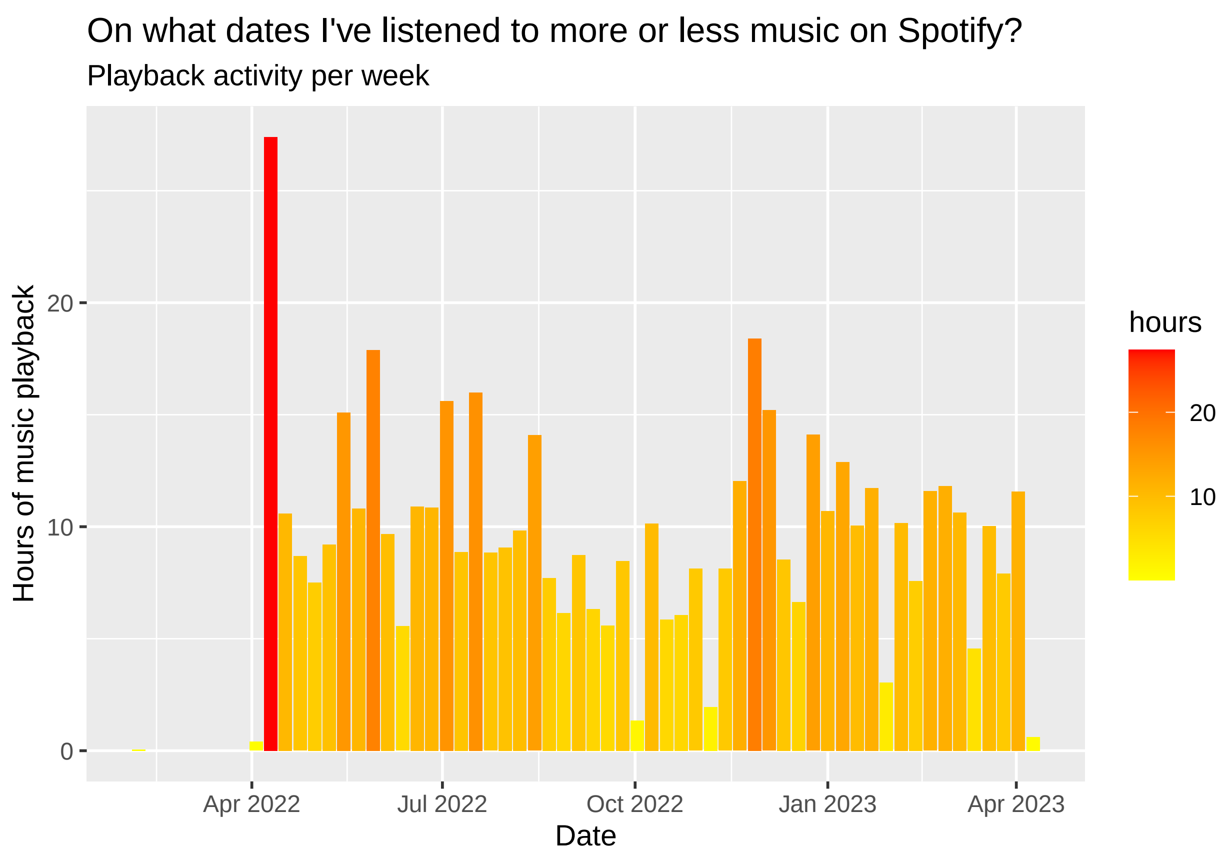
This chart is a great way to get an overall sense of what my listening was like. However to examine a frequency trend over the past year, I have changed one grouping aspect from `day` to `month` in the code above: `group_by(date = floor_date(date, "month")) %>%` Hence:

```r
# Number of songs I play per month : bar chart
songsByDay <- streamingData %>%
  filter(msPlayed >= 1000) %>%
  group_by(date) %>%
  group_by(date = floor_date(date, "month")) %>%
  summarize(songs = n()) %>%
  arrange(date) %>%
  ggplot(aes(x = date, y = songs)) +
  geom_col(aes(fill = songs)) +
  scale_fill_gradient(high = "#132B43", low = "#56B1F7") +
  labs(x= "Date", y= "Number of Songs Played") +
  ggtitle("Number of songs I play per month", "April 2022 to April 2023")
songsByDay
```



Instead of songs played, let's try hours listened to:

```
# PLAYBACK ACTIVITY PER WEEK AND HOURS
streamingHours <- streamingData %>%
  filter(date >= "2021-01-01") %>%
  group_by(date) %>%
  group_by(date = floor_date(date, "week")) %>%
  summarize(hours = sum(minutes) / 60) %>%
  arrange(date) %>%
  ggplot(aes(x = date, y = hours)) +
  geom_col(aes(fill = hours)) +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x= "Date", y= "Hours of music playback") +
  ggtitle("On what dates I've listened to more or less music on Spotify?", "Playback activity per week"
streamingHours
```

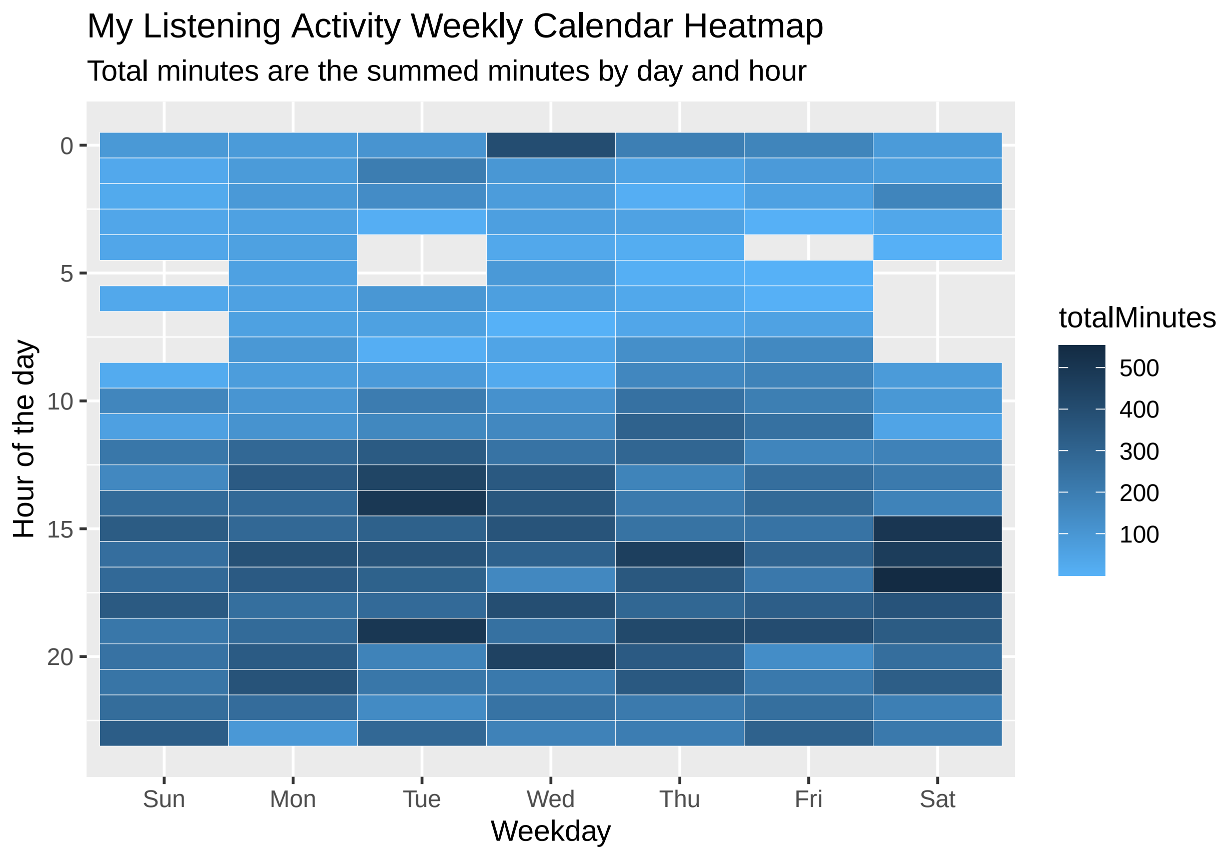


Creating my listening activity weekly calendar heatmap

Heatmaps can be a great way to provide a visual overview of how frequently items appear in a dataset over time.

To make the listening heatmap, I used the `group_by()` and `summarise()` functions to create the total minutes value each tile is filled with and then used the `geom_tile()` function to create the heatmap. The specifications after define the gradient used, which is the reverse of the default gradient the function uses because I wanted the higher values to be darker, and make the `y - axis` reversed to look more like a calendar.

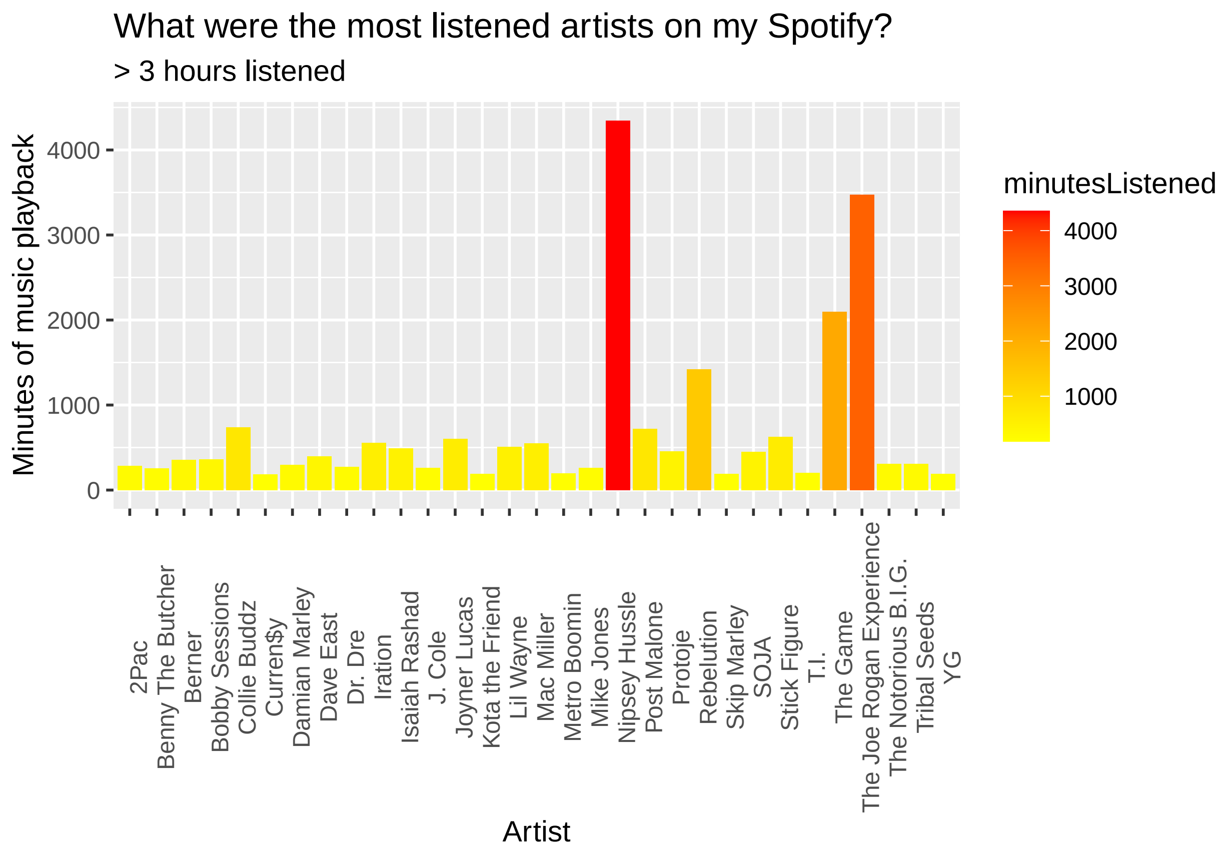
```
# Listening Weekly Calendar Heatmap
weekdaysHoursData <- streamingData %>%
  group_by(date, hour = hour(endTime), weekday = wday(date, label = TRUE))%>%
  summarize(minutesListened = sum(minutes))
listeningHeatMap <-
  weekdaysHoursData %>%
  group_by(hour, weekday) %>%
  summarize(totalMinutes = sum(minutesListened)) %>%
  ggplot(aes(weekday, hour, fill = totalMinutes)) +
  geom_tile(color = "white", size = 0.1) +
  scale_fill_gradient(high = "#132B43", low = "#56B1F7") +
  scale_y_continuous(trans = "reverse") +
  labs(x= "Weekday", y= "Hour of the day") +
  ggtitle(
    "My Listening Activity Weekly Calendar Heatmap",
    "Total minutes are the summed minutes by day and hour"
  )
listeningHeatMap
```



Visualizing Listening Activity by Artist

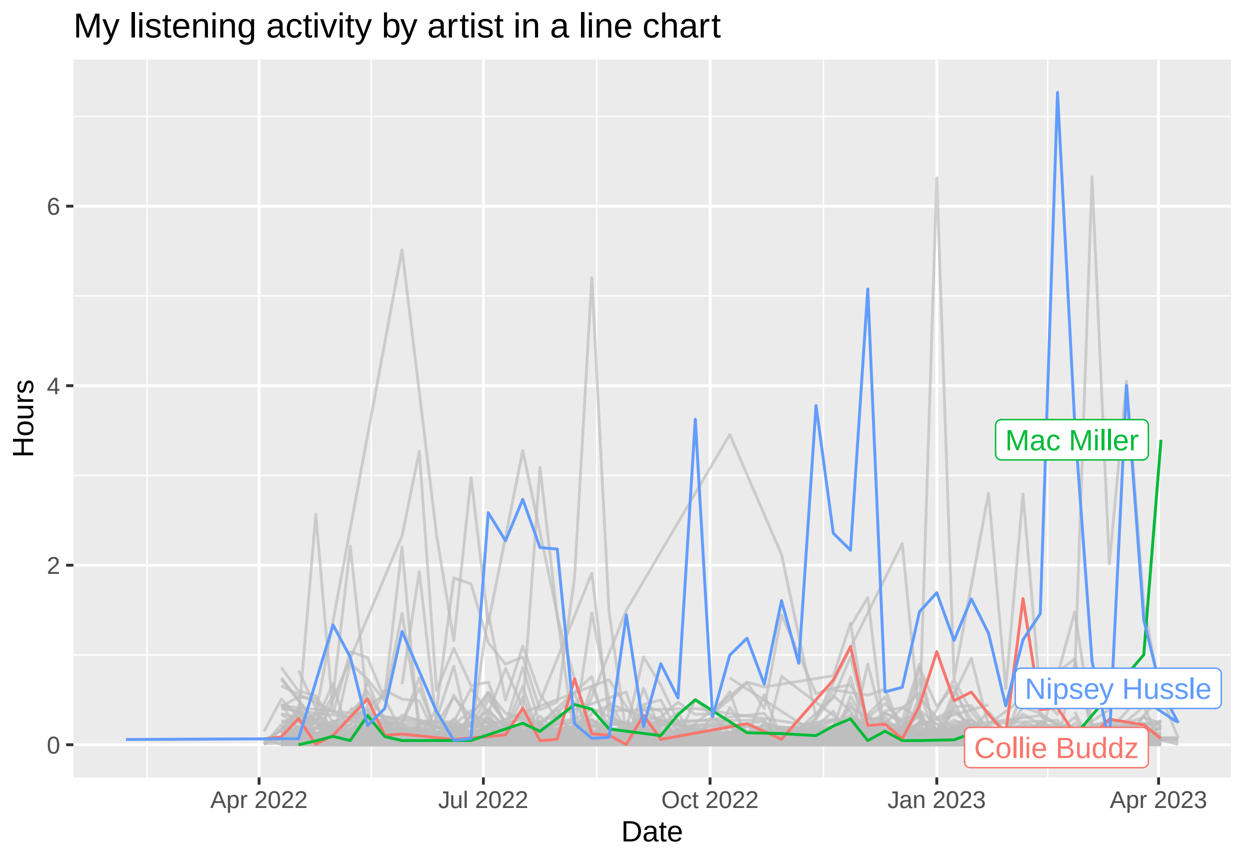
Music is a journey, and every artist I listen to is a companion on that journey. But who are my most frequent companions? To find out, I created a line chart showing my listening activity by artist over time.

```
# Most listened to artists
minutesMostListened <- streamingData %>%
  filter(date >= "2021-01-01") %>%
  group_by(artistName) %>%
  summarize(minutesListened = sum(minutes)) %>%
  filter(minutesListened >= 180) %>%
  ggplot(aes(x = artistName, y = minutesListened)) +
  geom_col(aes(fill = minutesListened)) +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x= "Artist", y= "Minutes of music playback") +
  ggtitle("What were the most listened artists on my Spotify?", "> 3 hours listened") +
  theme(axis.text.x = element_text(angle = 90))
minutesMostListened
```



This information can be used to create a "listening history" line chart of the minutes listened to each artist over time. To do this, I used the `group_by()` and `summarise()` functions in a similar way to previous plots to isolate the streaming data by artist and create data points by week. Then I plotted the data over a line chart using `geom_line()`. Finally, `gghighlight()` was used to color artists of interest on the graph which were hard-coded into the plot.

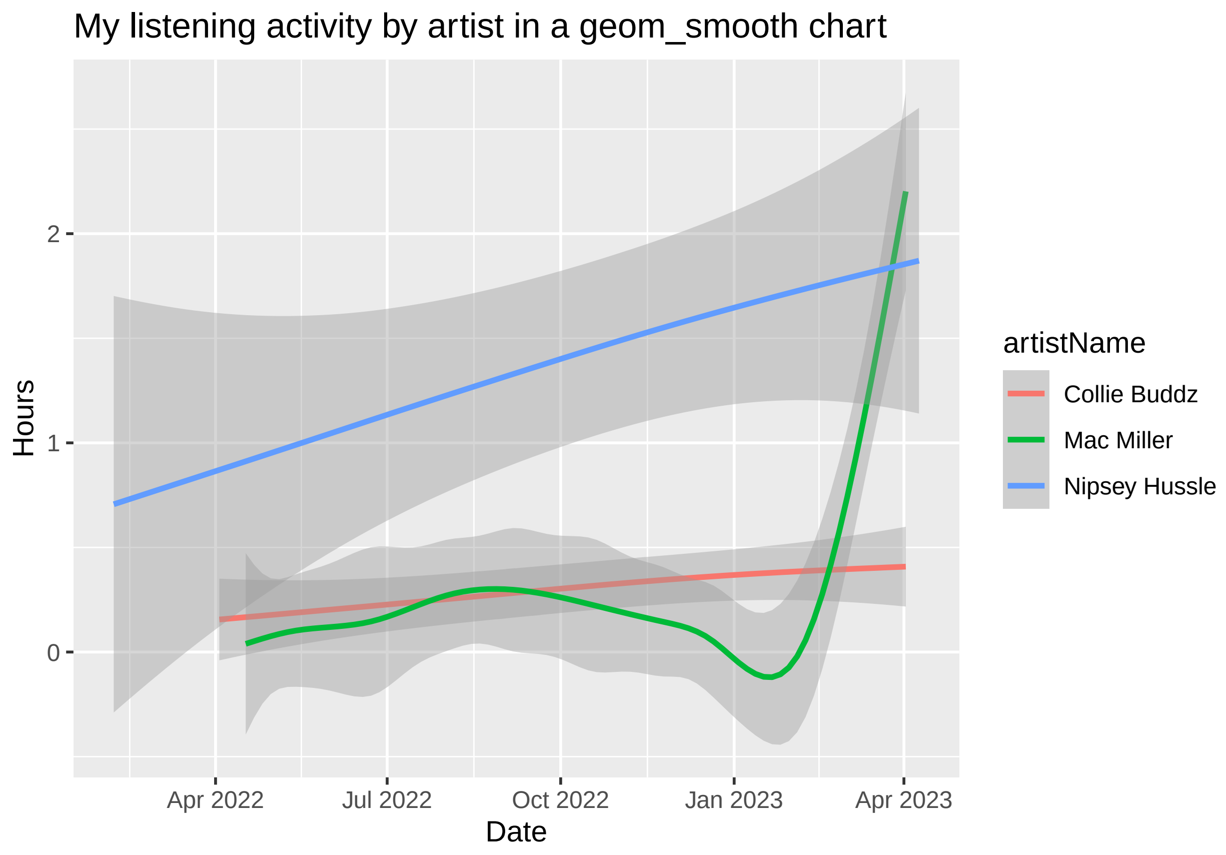
```r
# My listening activity by artist in a line chart
artistLineChart <- streamingData %>%
  group_by(artistName, date = floor_date(date, "week")) %>%
  summarize(hours = sum(minutes) / 60) %>%
  ggplot(aes(x = date, y = hours, group = artistName, color = artistName)) +
  geom_line() +
  gghighlight(artistName == "Nipsey Hussle" ||
              artistName == "Collie Buddz" || artistName == "Mac Miller") +
  labs(x= "Date", y= "Hours") +
  ggtitle("My listening activity by artist in a line chart")
artistLineChart
```



An interesting takeaway: many artists have more than one spike in activity, meaning that I was really interested in them, forgot about them, and then got really into them again. You can clearly see this pattern with Mac Miller, there seems to be a lot of spikes and then huge absences.

I converted the line chart above to a `geom_smooth()` chart by replacing `geom_line()` with `geom_smooth()`.

```
# My listening activity by artist in a geom_smooth chart
artistLineChart2 <- streamingData %>%
  group_by(artistName, date = floor_date(date, "week")) %>%
  summarize(hours = sum(minutes) / 60) %>%
  ggplot(aes(x = date, y = hours, group = artistName, color = artistName)) +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs")) +
  gghighlight(artistName == "Nipsey Hussle" ||
                artistName == "Collie Buddz" || artistName == "Mac Miller") +
  labs(x= "Date", y= "Hours") +
  ggtitle("My listening activity by artist in a geom_smooth chart")
artistLineChart2
```



Conclusion & Further Exploration:

For me, the results have a trend that I listen to music a lot more later in the day and a lot at night, compared to almost none in the mornings. # Visualizing listening activity by artist in a line chart The personal streaming history data from Spotify also includes the track name and artist name of each song you listened to. This information can be used to create a "listening history" line chart of the minutes listened to each artist over time.

Through this analysis, I've gained a deeper understanding of my relationship with music. I've seen how my musical tastes have evolved, how my listening habits reflect my routines, and how the characteristics of songs influence how long I listen to them. But this is just the beginning. There's a whole world of musical data out there, waiting to be explored. So stay tuned for the next chapter of my musical journey!