# EDA of Most Streamed Songs on Spotify 2023

Jason A. Esquivel

2024-04-17

## Data Analysis on the most streamed songs on spotify as of 2023

We will be getting our hands dirty with a ton of data from the "Most Streamed Spotify Songs 2023" data set I retrieved from the kaggle website. We are especially going to focus on the energy quality in the music. The following are going to be answered and explored in our analysis:

- In which months did the most streamed songs as of 2023 release? Any possible reason why?
- Does a higher energy level in the song make it more popular ? (i.e. have more streams)?
- Any trends in the energy level of songs for the last 10 years?
- Do songs released in the Summer have higher energy levels than those released in the Winter?

### Setting up my R environment to begin the analysis

Notes: Installing and loading up the 'tidyverse', 'readr', and 'dplyr' packages

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(readr)
```

```r
library(dplyr)
```

## Using the 'readr' package to load in the data set

Notes: Using the 'read.csv' function, I load in the .csv file containing the data. After, I will be assigning this now loaded data set to a data frame for R.

```r
top_songs_2023 <- read.csv('spotify-2023.csv')
```

## Getting familiarized with the data

Notes: Using the 'View' and 'head' functions to see the data

```r
View(top_songs_2023)
head(top_songs_2023)
```

```
##                             track_name    artist.s._name artist_count
## 1 Seven (feat. Latto) (Explicit Ver.)  Latto, Jung Kook            2
## 2                                 LALA       Myke Towers            1
## 3                              vampire    Olivia Rodrigo            1
## 4                         Cruel Summer      Taylor Swift            1
## 5                       WHERE SHE GOES         Bad Bunny            1
## 6                             Sprinter Dave, Central Cee            2
##   released_year released_month released_day in_spotify_playlists
## 1          2023              7           14                  553
## 2          2023              3           23                 1474
## 3          2023              6           30                 1397
## 4          2019              8           23                 7858
## 5          2023              5           18                 3133
## 6          2023              6            1                 2186
##   in_spotify_charts   streams in_apple_playlists in_apple_charts
## 1               147 141381703                 43             263
## 2                48 133716286                 48             126
## 3               113 140003974                 94             207
## 4               100 800840817                116             207
## 5                50 303236322                 84             133
## 6                91 183706234                 67             213
##   in_deezer_playlists in_deezer_charts in_shazam_charts bpm key  mode
## 1                  45               10              826 125   B Major
## 2                  58               14              382  92  C# Major
## 3                  91               14              949 138   F Major
## 4                 125               12              548 170   A Major
## 5                  87               15              425 144   A Minor
## 6                  88               17              946 141  C# Major
##   danceability_. valence_. energy_. acousticness_. instrumentalness_.
## 1             80        89       83             31                  0
## 2             71        61       74              7                  0
## 3             51        32       53             17                  0
## 4             55         58        72             11                 0
## 5             65         23        80             14                63
## 6             92         66        58             19                 0
##   liveness_. speechiness_.
## 1          8             4
## 2         10             4
## 3         31             6
## 4         11            15
## 5         11             6
## 6          8            24
```

## Filtering the main data set

Notes: With the 'dplyr' package, I use the 'rename' function to rename some of the columns we need into simple names instead of those currently containing characters.

```
top_songs_2023 <- top_songs_2023 %>%
  rename(energy = energy_.)
```

Notes: One of the rows in the data set had a corrupt cell in the 'stream' column. Unable to figure out the number for the streams, I have to remove this row.

```
top_songs_2023 <- top_songs_2023[-c(575),]
```

## Creating the Summer and Winter data frames.

Notes: Creating the data frame for songs released in the Summer, represented by the months of June(6), July(7), and August(8). Using the 'select' function to keep necessary columns and 'filter' function to return only the months I want.

```
summer_release_songs <- top_songs_2023 %>%
  select(released_month, streams, bpm, energy) %>%
  filter(released_month %in% c(6, 7, 8))
```

```
View(summer_release_songs)
head(summer_release_songs)
```

```
##   released_month   streams bpm energy
## 1              7 141381703 125     83
## 2              6 140003974 138     53
## 3              8 800840817 170     72
## 4              6 183706234 141     58
## 5              7  58149378 100     71
## 6              7  58255150 150     82
```

Notes: Creating the data frame for songs released in the Winter, represented by the months of December(12), January(1), and February(2).

```r
winter_release_songs <- top_songs_2023 %>%
  select(released_month, streams, bpm, energy) %>%
  filter(released_month %in% c(12, 1, 2))
```

```r
View(winter_release_songs)
head(winter_release_songs)
```

```
##   released_month    streams bpm energy
## 1              1 1316855716 118     68
## 2             12 1163093654  89     73
## 3              2  496795686 120     59
## 4              1 1297026226 135     42
## 5              1  429829812 204     68
## 6             12  843957510  98     61
```

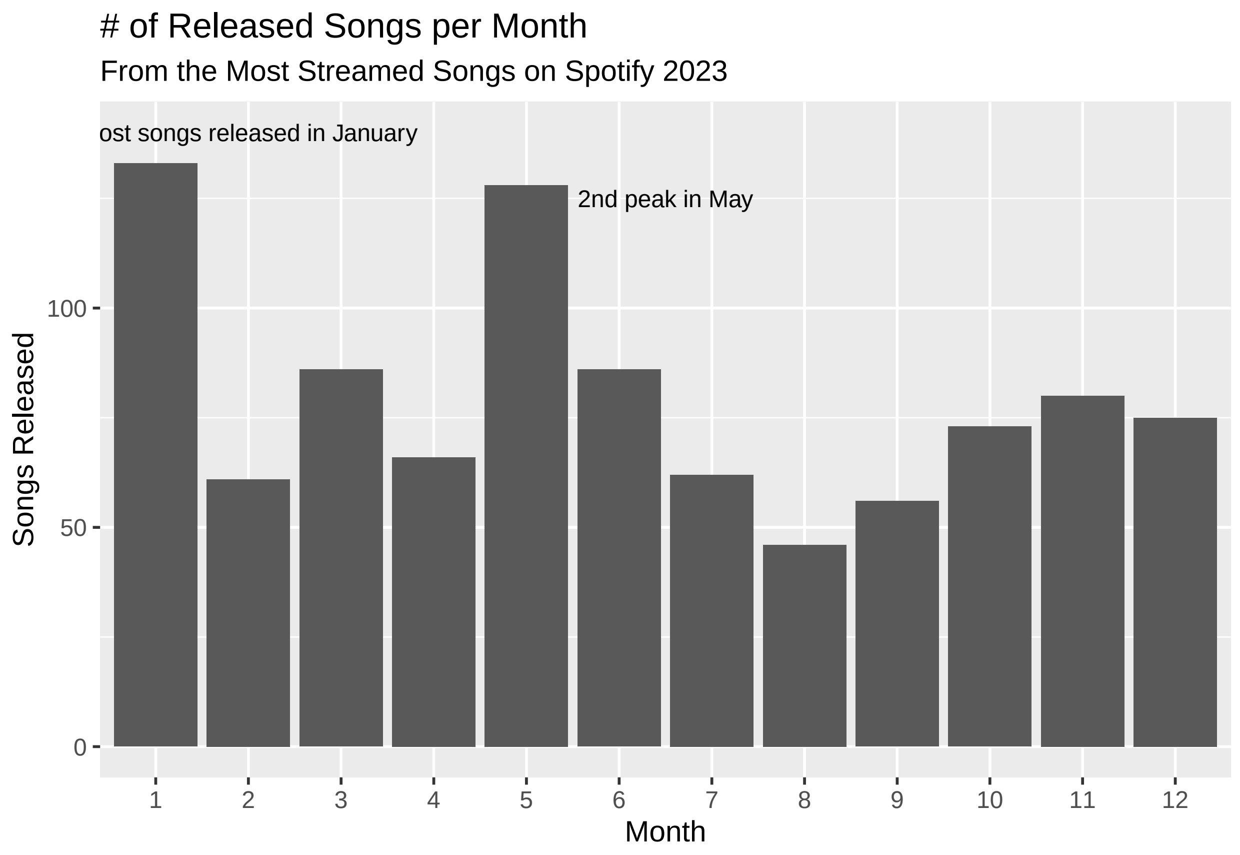
### Begin the visualization process

Notes: First I will be Installing and loading in the ‘ggplot2’ package to provide the tools necessary for visualization.

```r
library(ggplot2)
```

## Which months did the most streamed songs as of July 2023 release? What do we think?

```r
ggplot(data = top_songs_2023) + geom_bar(mapping = aes(x = factor(released_month))) +
labs(title = "# of Released Songs per Month", subtitle =
"From the Most Streamed Songs on Spotify 2023") +
scale_x_discrete(breaks = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)) +
annotate("text", x = 2, y = 140, label = "Most songs released in January", size = 3) +
annotate("text", x = 6.5, y = 125, label = "2nd peak in May", size = 3) +
ylab("Songs Released") + xlab("Month")
```



Conclusion: January is the 1st month of the new year, a time where many music festivals are active, usually carried over by the holidays, as well as an opportunity for artist to release new music for the new year. This music could have been promoted and gathered hype from the previous year(s) so much that it presents the highest peak in this graph. Next we have the month of May with the 2nd most song releases. This shows me that artists are likely releasing their music just before summer as there are also many festivals during the summer. Not to mention a larger demographic will be available to listen to more music because of a summer break.

### Does a higher energy level in the song make it more popular ? (i.e. have more streams)?

Notes: First I need to take a look at each column for its' data type.

```
str(top_songs_2023)
```

```
## 'data.frame':    952 obs. of  24 variables:
##  $ track_name          : chr  "Seven (feat. Latto) (Explicit Ver.)" "LALA" "vampire" "Cruel Summer"
##  $ artist.s._name      : chr  "Latto, Jung Kook" "Myke Towers" "Olivia Rodrigo" "Taylor Swift" ...
##  $ artist_count        : int  2 1 1 1 1 2 2 1 1 2 ...
##  $ released_year       : int  2023 2023 2023 2019 2023 2023 2023 2023 2023 2023 ...
##  $ released_month      : int  7 3 6 8 5 6 3 7 5 3 ...
##  $ released_day        : int  14 23 30 23 18 1 16 7 15 17 ...
##  $ in_spotify_playlists: int  553 1474 1397 7858 3133 2186 3090 714 1096 2953 ...
##  $ in_spotify_charts   : int  147 48 113 100 50 91 50 43 83 44 ...
##  $ streams             : chr  "141381703" "133716286" "140003974" "800840817" ...
##  $ in_apple_playlists  : int  43 48 94 116 84 67 34 25 60 49 ...
##  $ in_apple_charts     : int  263 126 207 207 133 213 222 89 210 110 ...
##  $ in_deezer_playlists : chr  "45" "58" "91" "125" ...
##  $ in_deezer_charts    : int  10 14 14 12 15 17 13 13 11 13 ...
##  $ in_shazam_charts    : chr  "826" "382" "949" "548" ...
##  $ bpm                 : int  125 92 138 170 144 141 148 100 130 170 ...
##  $ key                 : chr  "B" "C#" "F" "A" ...
##  $ mode                : chr  "Major" "Major" "Major" "Major" ...
##  $ danceability_.      : int  80 71 51 55 65 92 67 67 85 81 ...
##  $ valence_.           : int  89 61 32 58 23 66 83 26 22 56 ...
##  $ energy              : int  83 74 53 72 80 58 76 71 62 48 ...
##  $ acousticness_.      : int  31 7 17 11 14 19 48 37 12 21 ...
##  $ instrumentalness_.  : int  0 0 0 0 63 0 0 0 0 0 ...
##  $ liveness_.          : int  8 10 31 11 11 8 8 11 28 8 ...
##  $ speechiness_.       : int  4 4 6 15 6 24 3 4 9 33 ...
```

Notes: The 'streams' column, having numbers, is found to be a character data type. I will need to convert this to a numerical type in order to move forward with the sum functions.

```
top_songs_2023$streams <- as.numeric(top_songs_2023$streams)
```

Notes: Sum all of the streams for songs that have an energy level <= 25 percent.

```
sum(top_songs_2023[which(top_songs_2023[,20]<= 25 ), 9])
```

```
## [1] 5798289306
```

Notes: Sum all of the streams for songs that have an energy level between 26 and 50 percent.

```
sum(top_songs_2023[which(top_songs_2023$energy %in% c(26:50)), 9])
```

```
## [1] 94893422821
```

Notes: An extra step I took is checking if my above method of adding the streams per energy level was actually accurate. So, I created a data frame which will only include songs with energy levels between 26 and 50. Finally, I use a simple sum function of the streams from this data set and find that the answer is the same. I can move forward.

```
second_quarter_energy <- top_songs_2023 %>% filter(energy %in% c(26:50))
sum(second_quarter_energy$streams)
```

```
## [1] 94893422821
```

Notes: Sum all of the streams for songs that have an energy level between 51 and 75 percent.

```
sum(top_songs_2023[which(top_songs_2023$energy %in% c(51:75)), 9])
```

```
## [1] 253829859864
```

Notes: Sum all of the streams for songs that have an energy level between 76 and 100 percent.

```r
sum(top_songs_2023[which(top_songs_2023$energy %in% c(76:100)), 9])
```

```
## [1] 134937256551
```

Notes: Plotting the 4 percentiles of energy levels vs streams. First I will be creating a new data frame out of the data we gathered in the previous steps. Grouping each sum of streams to their respective energy percentile. I will plot this after.

```r
energy_percentile <- c("0-25%","26-50%","51-75%","76-100%")
number_of_streams <- c(5798289306, 94893422821, 253829859864, 134937256551)
energy_streams <- data.frame(energy_percentile, number_of_streams)
```

Notes: Now I take a look at the new data frame.

```r
View(energy_streams)
head(energy_streams)
```

```
##   energy_percentile number_of_streams
## 1             0-25%        5798289306
## 2            26-50%       94893422821
## 3            51-75%      253829859864
## 4           76-100%      134937256551
```

Notes: load in the ‘scales’ package to give the y axis values in our plot commas. Since the sum of streams added to the billion, I will need to make the values clear.

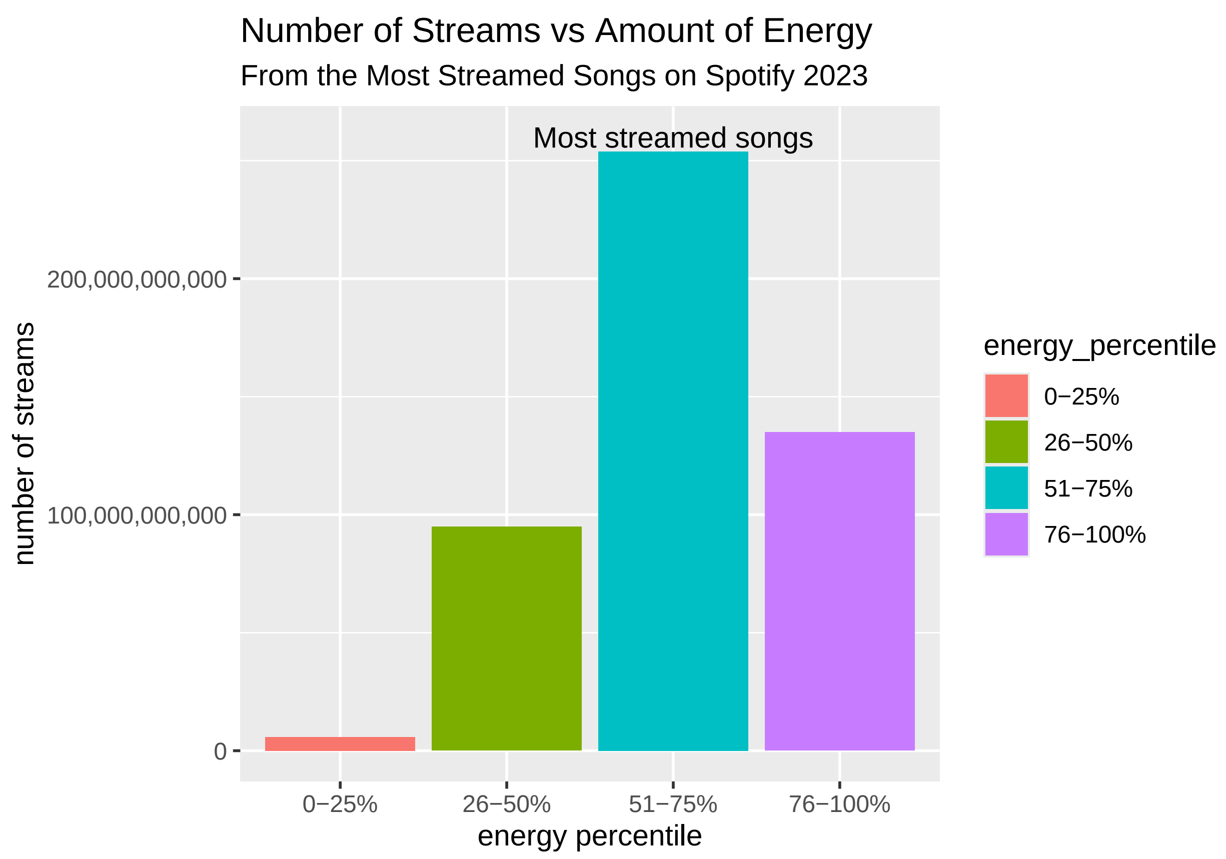
```r
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
ggplot(data = energy_streams) +  geom_col(mapping = aes(x =    energy_percentile, y = number_of_streams
labs(title = "Number of Streams vs Amount of Energy", subtitle =
"From the Most Streamed Songs on Spotify 2023") +
annotate("text", x = "51-75%", y = 260000000000, label = "Most streamed songs") +
scale_y_continuous(labels = scales::comma, name = "number of streams") +
scale_x_discrete(name = "energy percentile")
```



Conclusion: It is clear from this plot that songs with energy levels between 50 - 75% have the most streams. Specifically, the sum of the streams at this level reach 258.3 Billion. This is a stark contrast to songs with energy levels between 0 and 25%, which only add up to 5.8 Billion streams. There seems to be a trend where the less energy level means less streams, as the 2nd largest amount of streams falls between 76 - 100%, which is then followed by 26 - 50% and finally 0 - 25%.

### Any trends in the energy level of songs for the last 10 years?

Notes: Here we will be taking a look at the energy levels in the most streamed spotify songs of 2023 by their release date for a 10 year span (2013 - 2023). Time to create a new data frame with data of only these years.

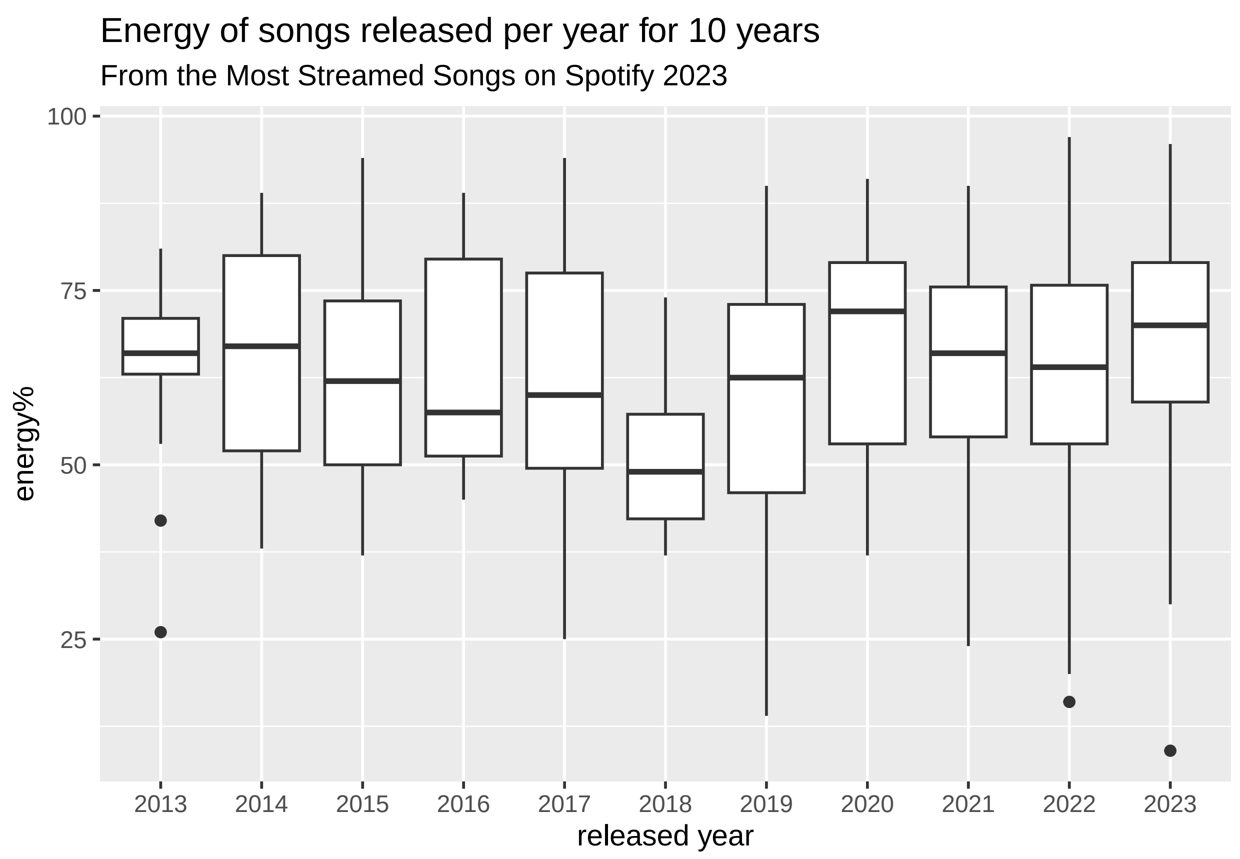
```
songs_2013_to_2023 <- top_songs_2023 %>% filter(released_year %in% c(2013:2023))
```

Notes: Taking a look at this data frame.

```
View(songs_2013_to_2023)
```

Notes: Time to plot the data.

```
ggplot(data = songs_2013_to_2023) + geom_boxplot(mapping = aes(x = factor(released_year), y = energy))
scale_x_discrete(breaks = c(2012, 2013, 2014, 2015,
2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023), name = "released year") +
scale_y_continuous(name = "energy%") +
labs(title = "Energy of songs released per year for 10 years",subtitle = "From the Most Streamed Songs
```

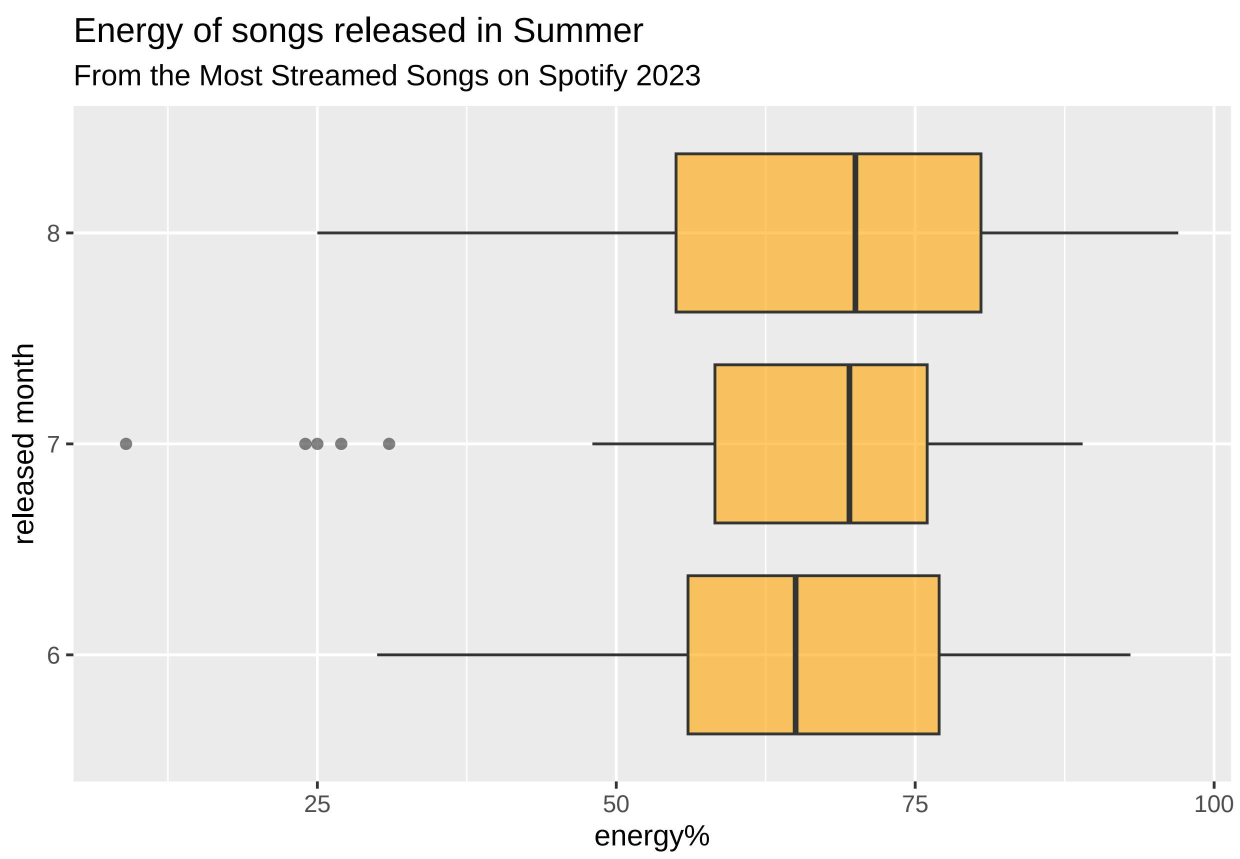


Conclusion: My observations conclude that roughly a 50 - 75% energy level in songs keep the people listening, even from songs released up to 10 years ago. We can see this from the boxes(which account for 50% of all the data points per year) falling within the 50 - 75% range, only ever minimally spilling over or under. The median, represented as the horizontal line in the boxes, also all fit between 50 - 75% with just one exception in the year 2018.

## Do songs released in the Summer have higher energy levels than those released in the Winter?

Notes: Our next objective is to find out if songs released during the Summer months have more energy levels than those in the Winter. Let's plot these variables.

```
ggplot(data = summer_release_songs) + geom_boxplot(mapping = aes(x = factor(released_month),
y = energy), fill = 'orange', alpha = 0.6) +
scale_x_discrete(breaks = c(6, 7, 8), name = "released month") +
scale_y_continuous(name = "energy%") +
coord_flip() +
labs(title = "Energy of songs released in Summer",
subtitle = "From the Most Streamed Songs on Spotify 2023")
```

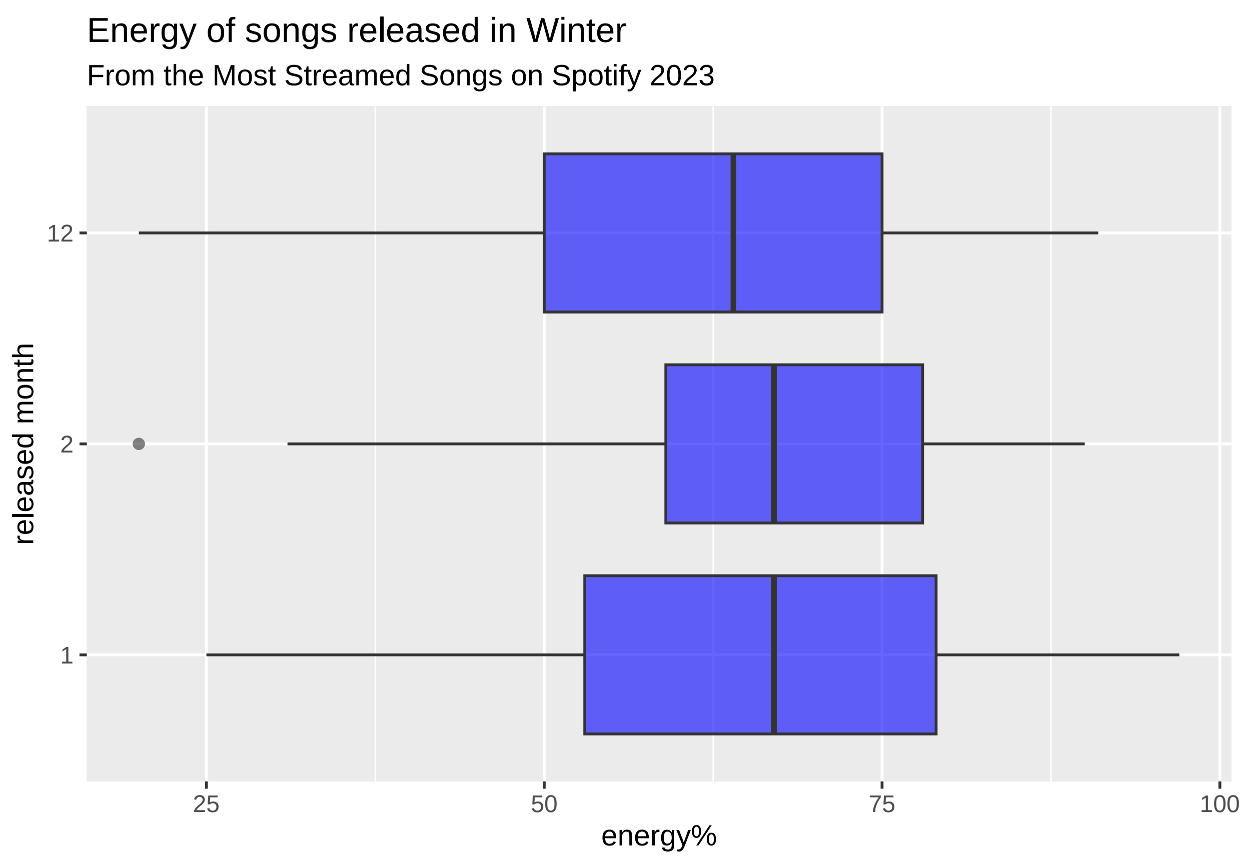


Notes: Take a look at the average energy level for Summer.

```
summer_release_songs %>% summarise(mean(energy))
```

```
##   mean(energy)
## 1     65.80928
```

```
ggplot(data = winter_release_songs) + geom_boxplot(mapping = aes(x = factor(released_month),
y = energy), fill = 'blue', alpha = 0.6) +
scale_x_discrete(breaks = c(12, 1, 2), name = "released month") +
scale_y_continuous(breaks = c(25, 50, 75, 100), name = "energy%") +
coord_flip() +
labs(title = "Energy of songs released in Winter",
subtitle = "From the Most Streamed Songs on Spotify 2023")
```



Notes: Take a look at the average energy level for winter.

```
winter_release_songs %>% summarise(mean(energy))
```

```
##   mean(energy)
## 1     64.73234
```

Conclusion: I have observed that more songs with a low energy level are streamed during Winter, whereas Summer contains songs that are more kept within the higher energy levels, with not as much range aside from the outliers in the month of July. However, this is simply an observation of the range. The averages of the two seasons' energy levels are pretty much the same. Summers' being 65.8 and Winters' 64.7. The median and the boxes can also be seen falling roughly between 50 and 75% energy, further enforcing the previous questions of this analysis. Summer is just barely higher, but still the # is insignificant and I can conclude that these seasons of the year do not change the energy level of the songs listened to, only that Winter allows for a greater range towards lower energy level.

## Conclusion

First, most songs from the most streamed spotify songs of 2023 are released in January, a great way for artists to start the new year. This is followed closely by the month of May which is the perfect way to begin the summer. Next, there is a sweet spot of 51 - 75% energy level in songs that will sum the streams to be the highest number. To further back the answer to this hypothesis, we find that having a 50 - 75% energy level in a song will keep people listening to it, even if those songs were released 10 years prior to 2023. Finally, the energy levels are barely higher in songs released during the summer than in the winter, so I accept that there is no significant difference. However, a final observation is that both of the seasons' months have an average energy level between 50 - 75 %.