



R project Data Analysis on Spotify Top Hits

CIS-5250: Visual Analytics

Yadav, Pratiksha V

Wong, Philip A

A.Introduction:

Spotify is a Swedish-based audio streaming and media service provider. It is now one of the most prominent digital music, podcast, and video streaming services, giving access to millions of songs from artists worldwide. Spotify is one of the largest music streaming service providers worldwide. As a freemium service, it has free features with advertisements and limited control and additional features such as offline listening and commercial-free listening, which are offered via paid subscriptions ^[1]. Subscribing to Spotify Premium is a great way to get rid of pesky ads, but there is more to the paid version of the service such as no advertisement anywhere, better audio quality, and downloading songs ^[2]. Users can search for music based on artist, genre, and popularity and create playlists. Not only does Spotify give us access to good songs on multiple platforms, but it has also exposed everyone to trending and upcoming artists from various genres that we had never experienced.

The dataset from kaggle.com provides several variables for exploratory analysis where we can visualize results and perform statistical analysis in RStudio using R language. Through the process, we expect to find valuable insights and answers to what makes some music more popular than others. This analysis will help better understand the unique insights and enable Spotify to create a more optimized content distribution that would be helpful for their technical teams, allowing further investigation, such as trends. Furthermore, the study should provide insight into how to better support users, thus increasing profits and improving the overall user experience.

B. Data Set URL:

The dataset for this project has been taken from the Kaggle website and below is the link:

<https://www.kaggle.com/code/lusfernandotorres/spotify-top-hits-2000-2019-eda/data>

Data Format: CSV

Total Rows: 2000

Total Columns: 15

Below is the dataset overview in RStudio, which displays the dataset's first few rows and the total number of columns.

artist	song	explicit	year	popularity	danceability	energy	loudness	speechiness	acousticness	liveness	valence	tempo	genre	duration_min
Britney Spears	Ooops...I Did It Again	FALSE	2000	77	0.751	0.834	-5.444	0.0437	3.00e-01	0.3550	0.8940	95.053	pop	3.519333
blink-182	All The Small Things	FALSE	1999	79	0.434	0.897	-4.918	0.0488	1.03e-02	0.6120	0.6840	146.726	rock, pop	2.784433
Faith Hill	Breathe	FALSE	1999	66	0.529	0.496	-9.007	0.0290	1.73e-01	0.2510	0.2780	136.859	pop, country	4.175767
Bon Jovi	It's My Life	FALSE	2000	78	0.551	0.913	-4.063	0.0466	2.63e-02	0.3470	0.5440	119.992	rock, metal	3.741550
Eric Prydz	Proper Education - Radio Edit	FALSE	2007	66	0.537	0.937	-4.543	0.0523	1.02e-03	0.0917	0.3240	124.936	pop, Dance/Electronic	3.309450
Fedde Le Grand	Put Your Hands Up For Detroit - Radio Edit	FALSE	2015	66	0.827	0.931	-4.474	0.2020	1.53e-02	0.0892	0.4910	127.995	pop, Dance/Electronic	2.508883
*NSYNC	Bye Bye Bye	FALSE	2000	65	0.614	0.928	-4.806	0.0516	4.08e-02	0.0845	0.6790	172.656	pop	3.342667
Sisqo	Thong Song	TRUE	1999	69	0.706	0.888	-6.959	0.0654	1.19e-01	0.0700	0.7140	121.549	hip hop, pop, R&B	4.228883

C. Data Description:

Data is an important source of information in any kind of project as it allows the analyst to build relationships between what is happening in different measures and categories.

Field Name	Data Description
Artist	Name of the Artist.
Song	Name of the Track.
duration_min	Duration of the track in minutes.
Explicit	The lyrics or content of a song or a music video contain one or more criteria that could be considered offensive or unsuitable for children.
Year	Release Year of the track.
Popularity	The higher the value the more popular the song is.
Danceability	Danceability describes a track's suitability for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

energy	It represents the perceptual measure of intensity and activity going from 0.0 to 1.0
loudness	Represents the overall loudness of a track in decibels (dB) and it's the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Loudness values are averaged across the entire track, with values typically between the range of -60 and 0 dB.
speechiness	It detects the presence of spoken words in a track. The more exclusively speech-like the recording, for instance, a talk show or an audiobook, the closer to 1.0 will be the attributed value. Values above 0.66 describe tracks that are mostly made entirely of spoken words, while values between 0.33 and 0.66 describe tracks that may contain both music and speech. Values below 0.33 most likely represent music and other non-speech tracks.

acousticness	A confidence measure from 0.0 to 1.0 indicates whether a track is acoustic or not, whereas 1.0 represents high confidence, the track is acoustic.
liveness	Detects the presence of an audience in the recording. A value above 0.8 indicates a higher probability that the track was performed live.
valence	A measure from 0.0 to 1.0 that describes the positiveness conveyed by the track. Tracks with higher valence may trigger positive emotions, such as happiness, cheerfulness, and euphoria, while tracks with lower valence may trigger negative emotions, such as sadness, depression, and anger.
tempo	Tempo in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

genre	Genre of the track.
-------	---------------------

D. Data Cleaning:

The data we download from the official sites is usually raw, and some house cleaning is generally required. This data needs to be cleaned to make it easier to analyze data. We need to remove incorrect, duplicate, or incomplete data. Various cleaning methods can be used for data cleaning, varying from dataset to dataset. Below are some of the data-cleaning methods used to clean the Spotify Dataset. The data has been cleaned in RStudio using R language.

1. Replacing Na values with median:

Using the summary function, we noticed 94 NA values in the dataset for the popularity column. We then decided to replace the NA values with the median value.

The R code below performs the following steps. Firstly, it sets up the working directory using the setwd function. The R script and the CSV file containing the data are stored in this working directory. Further, the read.csv this function is utilized to load the data contained in the CSV file into a data frame titled “spotify_songs.” Lastly, using the colSums() function, we aggregated 94 NA values by column. We have replaced Na values with median values using na.rm = TRUE.

Code Screenshot:

```
> setwd("C:/R/5250_R_Project_Script")
> spotify_songs<-read.csv("spotify_top_hits.csv")
> View(spotify_songs)
> dim(spotify_songs)
[1] 2000 18
> head(spotify_songs)
  artist song duration_ms explicit year
1 Britney Spears Oops!...I Did It Again 211160 FALSE 2000
2 blink-182 All The Small Things 167066 FALSE 1999
3 Faith Hill Breathe 250546 FALSE 1999
4 Bon Jovi It's My Life 224493 FALSE 2000
5 Eric Prydz Proper Education - Radio Edit 198567 FALSE 2007
6 Fedde Le Grand Put Your Hands Up For Detroit - Radio Edit 150533 FALSE 2015
  popularity danceability energy key loudness mode speechiness acousticness
1 77 0.751 0.834 1 -5.444 0 0.0437 0.30000
2 79 0.434 0.897 0 -4.918 1 0.0488 0.01030
3 66 0.529 0.496 7 -9.007 1 0.0290 0.17300
4 78 0.551 0.913 0 -4.063 0 0.0466 0.02630
5 NA 0.537 0.937 0 -4.543 1 0.0523 0.00102
6 NA 0.827 0.931 7 -4.474 1 0.2020 0.01530
  instrumentalness liveness valence tempo genre
1 1.77e-05 0.3550 0.894 95.053 pop
2 0.00e+00 0.6120 0.684 148.726 rock, pop
3 0.00e+00 0.2510 0.278 136.859 pop, country
4 1.35e-05 0.3470 0.544 119.992 rock, metal
5 3.50e-02 0.0917 0.324 124.938 pop, Dance/Electronic
6 2.10e-01 0.0992 0.491 127.995 pop, Dance/Electronic
> colSums(is.na(spotify_songs))
  artist song duration_ms explicit year
0 0 0 0 0 0
  popularity danceability energy key loudness
94 0 0 0 0 0
  mode speechiness acousticness instrumentalness liveness
0 0 0 0 0
  valence tempo genre
0 0 0
> spotify_songs[is.na(spotify_songs)]<-median(spotify_songs$popularity,na.rm=TRUE)
> colSums(is.na(spotify_songs))
  artist song duration_ms explicit year
0 0 0 0 0 0
  popularity danceability energy key loudness
0 0 0 0 0
  mode speechiness acousticness instrumentalness liveness
0 0 0 0 0
  valence tempo genre
0 0 0
```

Code Text:

```
> setwd("C:/R/5250_R_Project_Script")

> spotify_songs<-read.csv("spotify_top_hits.csv")

> View(spotify_songs)

> dim(spotify_songs)

[1] 2000 18

> head(spotify_songs)

> colSums(is.na(spotify_songs))

>spotify_songs[is.na(spotify_songs)]<-median(spotify_songs$popularity,na.rm=TRUE)
```



```
> colSums(is.na(spotify_songs))
```

Pre-cleaning screenshot:

song	duration_ms	explicit	year	popularity
Oops!...I Did It Again	211160	FALSE	2000	77
All The Small Things	167066	FALSE	1999	79
Breathe	250546	FALSE	1999	66
It's My Life	224493	FALSE	2000	78
Proper Education - Radio Edit	198567	FALSE	2007	NA
Put Your Hands Up For Detroit - Radio Edit	150533	FALSE	2015	NA
Bye Bye Bye	200560	FALSE	2000	65
Thong Song	253733	TRUE	1999	69
The Real Slim Shady	284200	TRUE	2000	86
Rock DJ	258560	FALSE	2000	68
Say My Name	271333	FALSE	1999	75
Lady - Hear Me Tonight	307153	FALSE	2001	77
L'Amour Toujours	238759	FALSE	2011	1
Move Your Body - Gabry Ponte Original Radio Edit	268863	FALSE	1999	56

Post-cleaning screenshot:

song	duration_ms	explicit	year	popularity
Oops!...I Did It Again	211160	FALSE	2000	77
All The Small Things	167066	FALSE	1999	79
Breathe	250546	FALSE	1999	66
It's My Life	224493	FALSE	2000	78
Proper Education - Radio Edit	198567	FALSE	2007	66
Put Your Hands Up For Detroit - Radio Edit	150533	FALSE	2015	66
Bye Bye Bye	200560	FALSE	2000	65
Thong Song	253733	TRUE	1999	69
The Real Slim Shady	284200	TRUE	2000	86
Rock DJ	258560	FALSE	2000	68
Say My Name	271333	FALSE	1999	75
Lady - Hear Me Tonight	307153	FALSE	2001	77
L'Amour Toujours	238759	FALSE	2011	1

2. Deleting duplicate records:

Duplicate records can make results inaccurate. It also causes needlessly increased memory usage; duplicate rows and columns should be removed from the data set. It is necessary to check for duplicates in the excel sheet. Accordingly, we performed the “unique” function in R studio. Before, we had 2000 rows in the column; after removing the duplicate records, we see a total of 1941. As shown below is the result of removing the duplicate records.

Code Screenshot:

```
> View(spotify_songs)
> dim(spotify_songs)
[1] 2000  18

> spotify_songs<-unique(spotify_songs)
> dim(spotify_songs)
[1] 1941  18
> View(spotify_songs)
```

Code Text:

```
> View(spotify_songs)

> dim(spotify_songs)

[1] 2000  18

> spotify_songs<-unique(spotify_songs)

> dim(spotify_songs)

[1] 1941  18

> View(spotify_songs)
```

Pre-cleaning screenshot:

	artist	song
1915	Kodak Black	ZEZE (feat. Travis Scott & Offset)
138	P.O.D.	Youth of the Nation
1080	Kesha	Your Love Is My Drug
793	Manic Street Preachers	Your Love Alone Is Not Enough (feat. Nina Persson)
562	Pretty Ricky	Your Body
1850	5 Seconds of Summer	Youngblood
1263	Snoop Dogg	Young, Wild & Free (feat. Bruno Mars)
1085	JAY-Z	Young Forever
91	Marc Anthony	You Sang To Me
127	Michael Jackson	You Rock My World
576	Westlife	You Raise Me Up
1131	Cobra Starship	You Make Me Feel... (feat. Sabi)
1416	DJ Snake	You Know You Like It
516	DJ Snake	You Know You Like It
672	Amy Winehouse	You Know I'm No Good
692	James Morrison	You Give Me Something

Post-cleaning screenshot:

	artist	song
562	Pretty Ricky	Your Body
1850	5 Seconds of Summer	Youngblood
1263	Snoop Dogg	Young, Wild & Free (feat. Bruno Mars)
1085	JAY-Z	Young Forever
91	Marc Anthony	You Sang To Me
127	Michael Jackson	You Rock My World
576	Westlife	You Raise Me Up
1131	Cobra Starship	You Make Me Feel... (feat. Sabi)
1416	DJ Snake	You Know You Like It
672	Amy Winehouse	You Know I'm No Good
692	James Morrison	You Give Me Something
1000	The Fray	You Found Me

3. Data type conversion from milliseconds to minutes:

Data conversion aims to provide significant support for transforming data to match specific requirements. This helps to remove irrelevant data, allowing for more accurate data analysis and insightful findings. In this example, most users wouldn't be able to measure in milliseconds but are familiar with minutes and hours. We have converted the milliseconds to minutes to make the results more meaningful to the audience. For the field named "duration," we used the mutate function and applied the formula: $(\text{duration_ms}/1000)/60$ to convert from milliseconds to minutes.

Code Screenshot:

```
> spotify_songs <- spotify_songs %>% mutate(duration_min = (duration_ms / 1000)/ 60,  
year)  
> View(spotify_songs)
```

Code Text:

```
> spotify_songs <- spotify_songs %>% mutate(duration_min = (duration_ms / 1000)/  
60, year)  
> View(spotify_songs)
```

Pre-cleaning screenshot:

duration_ms
211160
167066
250546
224493
198567
150533
200560
253733
284200
258560
271333
307153
238759
268863
306333

Post-cleaning screenshot:

duration_min
3.519333
2.784433
4.175767
3.741550
3.309450
2.508883
3.342667
4.228883
4.736667
4.309333
4.522217
5.119217
3.979317
4.481050
5.105550

4. Removing irrelevant columns:

The Spotify dataset contains a total of 19 columns. For this project, 15 of the 19 columns were needed. As a result, one of the steps to clean the data was removing unnecessary fields from the dataset. We have used the below code to drop out-of-scope data such as duration_ms, key, mode, and instrumentals that do not add value.

Code Screenshot:

```
> View(spotify_songs)
> View(spotify_songs)
> dim(spotify_songs)
[1] 1941  19
> spotify_songs<-spotify_songs[,-c(3,9,11,14)]
> View(spotify_songs)
> dim(spotify_songs)
[1] 1941  15
```

Code Text:

```
> View(spotify_songs)

> View(spotify_songs)

> dim(spotify_songs)

[1] 1941  19

> spotify_songs<-spotify_songs[,-c(3,9,11,14)]

> View(spotify_songs)

> dim(spotify_songs)

[1] 1941  15
```

Pre-cleaning screenshot:

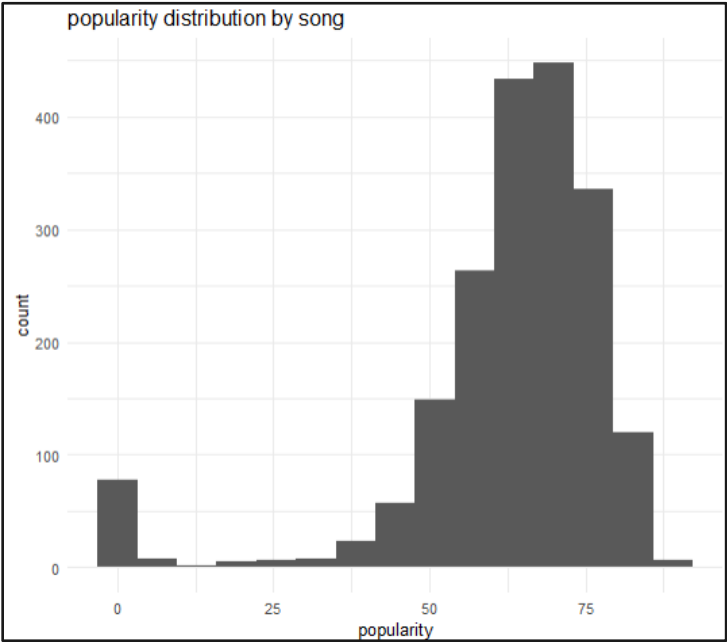
duration_ms	explicit	year	popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	genre	duration_min
113000	FALSE	2019	76	0.907	0.5300	1	-6.112	1	0.1270	5.78e-02	2.23e-06	0.1010	0.5070	135.998	hip hop, pop	1.883333
114893	FALSE	2019	71	0.703	0.5940	5	-6.146	0	0.0752	3.42e-01	0.00e+00	0.1230	0.4750	153.848	hip hop, pop	1.914883
119133	TRUE	2017	83	0.872	0.3910	0	-9.144	0	0.2420	4.69e-01	4.13e-06	0.2970	0.4370	134.021	hip hop	1.985550
121886	FALSE	2018	79	0.669	0.3080	11	-10.068	1	0.0290	8.83e-01	0.00e+00	0.0894	0.5200	64.934	hip hop	2.031433
124055	TRUE	2017	64	0.936	0.5230	5	-6.710	1	0.0597	2.39e-01	0.00e+00	0.1170	0.6990	119.689	hip hop	2.067563
126446	TRUE	2019	71	0.899	0.6110	2	-6.294	1	0.2580	6.77e-04	0.00e+00	0.0672	0.5870	105.038	hip hop, pop, R&B	2.107433
127920	FALSE	2009	2	0.617	0.7780	9	-8.671	0	0.0270	4.59e-01	9.25e-01	0.1280	0.1520	100.363	rock	2.132000
129264	TRUE	2018	1	0.906	0.3620	10	-12.890	0	0.2690	1.80e-01	0.00e+00	0.1130	0.3910	104.025	hip hop	2.154400
131064	TRUE	2018	75	0.710	0.7890	4	-3.874	1	0.0722	1.61e-02	2.77e-06	0.4510	0.7170	142.929	pop, Dance/Electronic	2.164400
131213	FALSE	2019	72	0.630	0.6300	9	-6.211	1	0.0395	1.31e-02	0.00e+00	0.1420	0.1630	80.512	hip hop, pop	2.166883
131240	TRUE	2019	78	0.745	0.6420	7	-6.237	0	0.2870	2.04e-02	0.00e+00	0.0658	0.2260	179.974	hip hop	2.187333
135090	TRUE	2018	82	0.921	0.5370	9	-5.723	0	0.0804	5.56e-01	4.04e-03	0.1020	0.7110	128.009	hip hop	2.251500
136568	TRUE	2019	69	0.677	0.7140	11	-5.637	1	0.0287	1.62e-01	0.00e+00	0.0717	0.3550	94.956	hip hop, pop	2.276133
138842	FALSE	2014	33	0.776	0.5740	5	-9.882	0	0.0317	4.66e-01	7.83e-05	0.1310	0.4120	121.030	pop, Dance/Electronic	2.314033
142273	TRUE	2018	79	0.963	0.3460	5	-9.309	0	0.5300	3.55e-02	0.00e+00	0.1080	0.5620	119.957	hip hop	2.371217
144244	FALSE	2016	57	0.807	0.8870	1	-3.892	1	0.2750	3.81e-03	0.00e+00	0.3910	0.7800	160.517	set()	2.404067
148186	FALSE	2008	63	0.639	0.9760	1	-5.503	1	0.3540	2.13e-02	0.00e+00	0.0856	0.3540	147.990	pop	2.469767
149546	TRUE	2019	79	0.829	0.5390	11	-7.339	0	0.2080	1.36e-01	1.78e-06	0.1030	0.3880	99.960	hip hop	2.492433

Post-cleaning screenshot:

explicit	year	popularity	danceability	energy	loudness	speechiness	acousticness	liveness	valence	tempo	genre	duration_min
FALSE	2000	77	0.751	0.834	-5.444	0.0437	3.00e-01	0.3550	0.8940	95.053	pop	3.519333
FALSE	1999	79	0.434	0.897	-4.918	0.0488	1.03e-02	0.6120	0.6840	148.726	rock, pop	2.784433
FALSE	1999	66	0.529	0.496	-9.007	0.0290	1.73e-01	0.2510	0.2780	136.859	pop, country	4.175767
FALSE	2000	78	0.551	0.913	-4.063	0.0466	2.63e-02	0.3470	0.5440	119.992	rock, metal	3.741550
FALSE	2007	66	0.537	0.937	-4.543	0.0523	1.02e-03	0.0917	0.3240	124.938	pop, Dance/Electronic	3.309450
FALSE	2015	66	0.827	0.931	-4.474	0.2020	1.53e-02	0.0992	0.4910	127.995	pop, Dance/Electronic	2.508883
FALSE	2000	65	0.614	0.928	-4.806	0.0516	4.08e-02	0.0845	0.8790	172.656	pop	3.342667
TRUE	1999	69	0.706	0.888	-6.959	0.0654	1.19e-01	0.0700	0.7140	121.549	hip hop, pop, R&B	4.228883
TRUE	2000	86	0.949	0.661	-4.244	0.0572	3.02e-02	0.0454	0.7600	104.504	hip hop	4.736667
FALSE	2000	68	0.708	0.772	-4.264	0.0322	2.67e-02	0.4670	0.8610	103.035	pop, rock	4.309333
FALSE	1999	75	0.713	0.678	-3.525	0.1020	2.73e-01	0.1490	0.7340	138.009	pop, R&B	4.522217
FALSE	2001	77	0.720	0.808	-5.627	0.0379	7.93e-03	0.0634	0.8690	126.041	Dance/Electronic	5.119217
FALSE	2011	1	0.617	0.728	-7.932	0.0292	3.28e-02	0.3600	0.8080	139.066	pop	3.979317
FALSE	1999	56	0.745	0.958	-9.664	0.0287	8.13e-02	0.5330	0.9600	129.962	pop	4.481050

E. Analysis & Visualizations:

1. Exploratory Analysis: What is the distribution of popularity by song? What are the top 10 songs?



Code screenshot:

```
> library(ggplot2)
> library(tidyverse)
>
> ggplot(spotify_songs, aes( x = popularity)) +
+ geom_histogram(bins=15) +
+ labs(title="popularity distribution by song") + theme_minimal()
```

Code Text:

```
> library(ggplot2)

> library(tidyverse)

>ggplot(spotify_songs, aes( x = popularity)) +

+ geom_histogram(bins=15) +

+ labs(title="popularity distribution by song") + theme_minimal()
```

R features:

- Plot Type: Histogram
- Functions: aes, geom_histogram, labs, theme_minimal
- Libraries: ggplot2, tidyverse

Insights:

We can see that the popularity distribution shows a normal distribution which in the figure above shows a bell-shaped curve. We can also see that there are songs that are of 0 value. We can interpret this as the value “0” as the actual value, meaning those songs are very unpopular or can be seen as possible outliers. It is also possible that the 0 value could represent a rating with a default

value of “0”. For this example, we assumed that the zero value meant a valid value as we have no evidence to the contrary to lead us to believe otherwise.

What are the top 10 songs by popularity?

```
# A tibble: 1,941 × 3
# Groups:   song [1,879]
  song                popularity genre
  <chr>              <dbl> <chr>
1 Sweater Weather      89 rock, pop
2 Another Love          88 pop
3 Without Me            87 hip hop
4 The Real Slim Shady   86 hip hop
5 Wait a Minute!        86 pop, R&B, Dance/Electronic
6 Lovely (with Khalid)  86 pop, Dance/Electronic
7 'Till I Collapse      85 hip hop
8 Locked out of Heaven  85 pop
9 Daddy Issues          85 rock, pop
10 The Nights           85 pop, Dance/Electronic
# ... with 1,931 more rows
# Use `print(n = ...)` to see more rows
```

Code Screenshot:

```
> spotify_songs %>%
+ select(song, popularity, genre) %>%
+ group_by(song) %>%
+ arrange(desc(popularity))
```

Code text:

```
> spotify_songs %>%

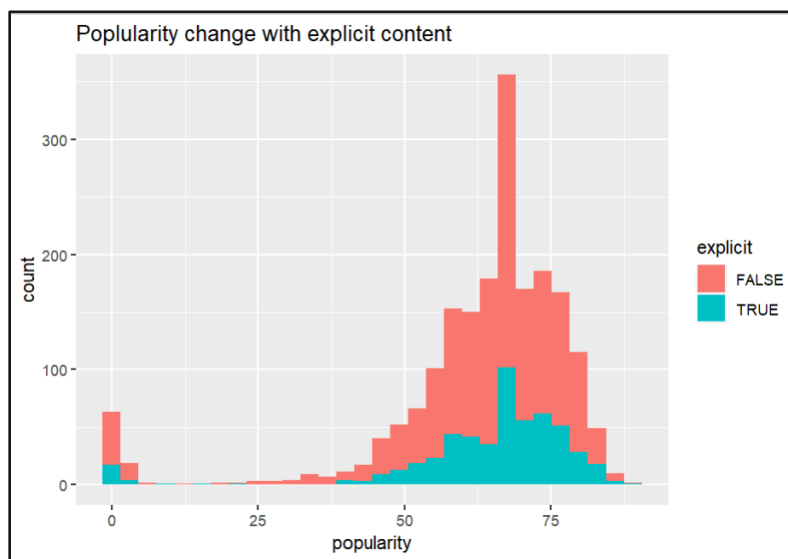
+ select(song, popularity, genre) %>%

+ group_by(song) %>%

+ arrange(desc(popularity))
```

Based on the initial observation, we can see that, by no coincidence, the most popular songs are of the genre “pop.”

2. How does the popularity change with the explicit content? Does explicit content positively, negatively, or neutral impact on popularity?



Code Screenshot:

```
> popularity_explicit_content<- spotify_songs %>%  
+ ggplot(aes(x=popularity, fill=explicit))+  
+ geom_histogram()+  
+ ggtitle("Poplularity change with explicit content")  
> popularity_explicit_content
```

Code Text:

```
> popularity_explicit_content<- spotify_songs %>%  
  
+ ggplot(aes(x=popularity, fill=explicit))+  
  
+ geom_histogram()+  
  
+ ggtitle("Poplularity change with explicit content")  
  
> popularity_explicit_content
```

R features:

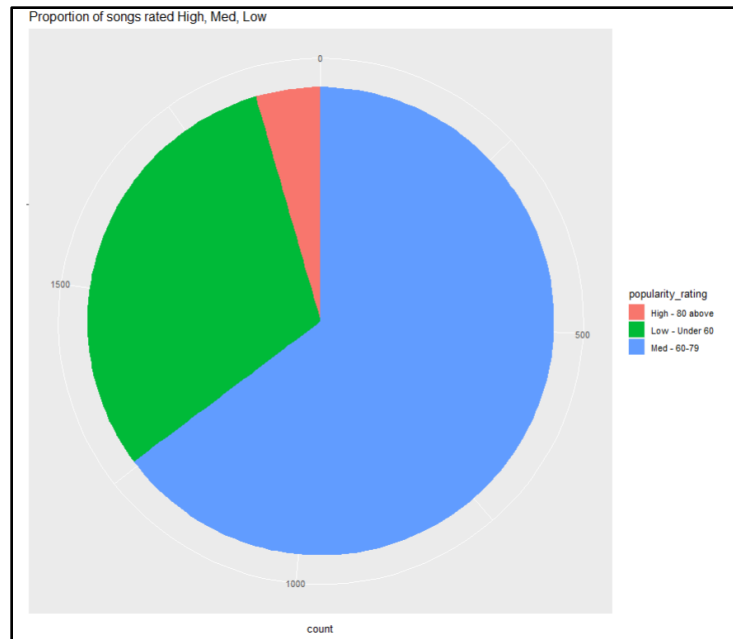
- Plot Type: histogram
- Functions: aes, geom_histogram, labs, theme_minimal
- Libraries: ggplot2, tidyverse

Insights:

Explicit content is ‘TRUE’ when the lyrics or content of a song contain one or more criteria that could be considered offensive or unsuitable for children. In this visualization, we try to identify whether the Explicit nature of the songs has any bearing on the song's popularity. As you can see, both explicit and non-explicit songs resemble a normal distribution. We can conclude that explicit songs (where explicit = “TRUE”) have little to no impact on the popularity of songs compared to non-explicit songs.

3. What is the proportion of high, medium, or low-rated songs?

In this example, we had to create the categorical variables high, medium, and low from popularity. A script created a new column named ‘popularity_rating’ and assigned each popularity value into one of the three categories using a nested if statement. The as.factor was used to convert the numerical value into a categorical data type for the histograms and pie charts below.



Code Screenshot:

```
> summary(spotify_songs$popularity)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.00  58.00   66.00  62.83  73.00   89.00
> spotify_songs$popularity
 [1] 77 79 66 78 66 66 65 69 86 68 75 77 1 56 55 62 61 53 64 73 64
 [22] 82 83 65 62 54 54 72 69 53 54 47 71 49 58 52 58 59 55 69 61 55
 [43] 54 66 36 79 49 68 43 59 64 52 47 60 55 60 59 59 65 63 65 66 58
 [64] 52 0 60 61 58 78 55 70 66 55 55 58 59 56 70 64 78 52 57 49 57
 [85] 54 68 62 65 60 54 56 54 60 70 64 68 65 71 64 69 65 68 76 70 83
 [106] 73 68 1 74 78 77 62 71 69 62 57 76 53 73 65 54 54 57 63 43 71
 [127] 64 73 66 50 78 76 82 50 42 73 57 69 57 68 52 60 50 74 57 48 61
 [148] 67 65 55 63 47 54 75 57 68 26 53 58 58 67 68 63 1 61 46 56 60
 [169] 69 60 58 64 53 56 68 46 66 65 66 64 54 66 0 51 48 0 54 75 59
 [190] 68 63 69 56 30 64 56 65 45 61 77 87 78 75 72 64 27 58 68 85 69
 [211] 60 53 53 40 66 0 54 59 60 43 66 61 46 49 42 52 60 55 59 47 58
 [232] 59 59 64 75 53 68 49 45 48 73 43 26 2 80 58 59 53 47 55 58 35
 [253] 63 45 24 55 62 62 32 53 71 61 54 68 52 56 79 70 60 64 69 73 70
 [274] 75 63 58 67 73 60 65 50 64 57 59 47 56 70 60 40 40 60 58 56 64
 [295] 72 51 60 81 74 77 76 50 68 70 56 39 66 61 59 71 76 69 72 70 79

> spotify_songs$popularity_rating <- as.factor(
+   ifelse(spotify_songs$popularity<=60, "Low - Under 60",
+         ifelse(spotify_songs$popularity<=80, "Med - 60-79",
+               ifelse(spotify_songs$popularity>80, "High - 80 above", "High - 80 above")
+         ))
+ )
> spotify_songs$popularity_rating
 [1] Med - 60-79 Med - 60-79 Med - 60-79 Med - 60-79
 [5] Med - 60-79 Med - 60-79 Med - 60-79 Med - 60-79
 [9] High - 80 above Med - 60-79 Med - 60-79 Med - 60-79
[13] Low - Under 60 Low - Under 60 Low - Under 60 Med - 60-79
[17] Med - 60-79 Low - Under 60 Med - 60-79 Med - 60-79
[21] Med - 60-79 High - 80 above High - 80 above Med - 60-79
[25] Med - 60-79 Low - Under 60 Low - Under 60 Med - 60-79
[29] Med - 60-79 Low - Under 60 Low - Under 60 Low - Under 60
...

Levels: High - 80 above Low - Under 60 Med - 60-79
> ggplot(spotify_songs, aes(x = popularity_rating))+
+   geom_bar() + labs(title="Popularity of songs rated High, Med, Low") + theme_minimal()
>
> ggplot(spotify_songs, aes(x=factor(""), fill=popularity_rating))+geom_bar()+coord_polar(
+   (theta = "y") +scale_x_discrete("") + labs(title="Proportion of songs rated High, Med, Lo
+   w")
+ )
```

Code Text:

```
'create popular_rating column for analysis'

summary(spotify_songs$popularity)

spotify_songs$popularity

spotify_songs$popularity_rating <- as.factor(
  ifelse(spotify_songs$popularity<=60, "Low - Under 60",
    ifelse(spotify_songs$popularity<=80, "Med - 60-79",
      ifelse(spotify_songs$popularity>80,'High - 80 above',"High - 80 above")
    ))
)

spotify_songs$popularity_rating

'plot bar and pie graph'

ggplot(spotify_songs, aes(x = popularity_rating))+
  geom_bar() + labs(title="Popularity of songs rated High, Med, Low") + theme_minimal()

ggplot(spotify_songs,
  aes(x=factor(""),fill=popularity_rating))+geom_bar()+coord_polar(theta = "y")
+scale_x_discrete("") + labs(title= "Proportion of songs rated High, Med, Low")
```

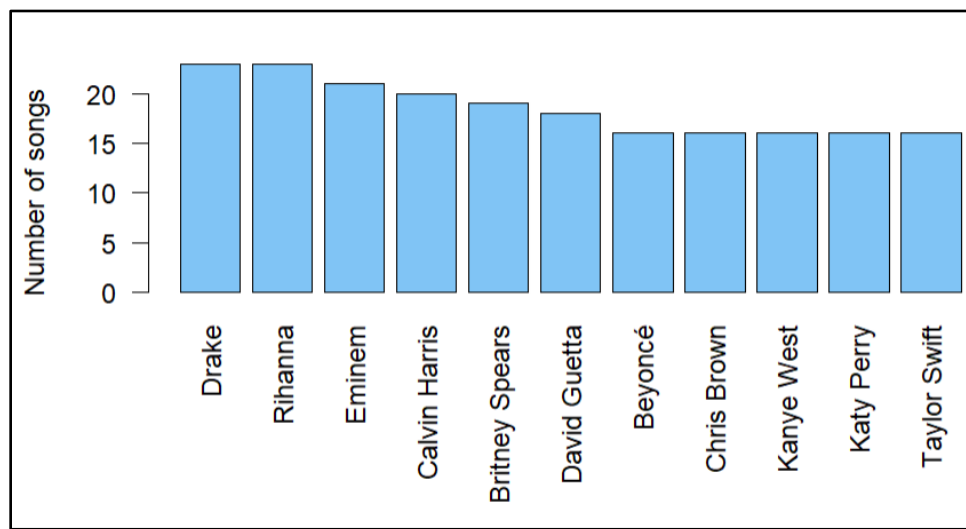
R features:

- Plot Type: Pie graph
- Functions: summary, as.factor, ifelse, aes, ggplot, geom_bar, labs, theme_minimal(), fill, coord_polar, scale_x_discrete
- Libraries: ggplot2, tidyverse, scales

Insights:

We can see from the pie graph(above) that there are significantly fewer songs rated 80 and above or the ‘high’ category compared to the others. As expected, we have a substantial number of Spotify songs in the medium category. Also, to note here is that since we did not perform any data cleaning on popularity values (that are equal to zero) we can see that we have less of a normal distribution of the dataset than we previously assumed, as visualized in question 1.

4. Top 15 Artists with the most releases of the songs from the year 1998 – 2020?



Code screenshot:

```
> Artist_Fil <- Artist_Popular %>% filter(Count >= 15)
> par(mar = c(12, 5, 4, 2)+ 0.1)
> barplot(Artist_Fil$Count,
+         ylab = "Number of songs",
+         col = "#80C4F5",
+         names.arg= Artist_Fil$artist,
+         width= 0.01,
+         ylim = c(0,20),
+         las = 2
+ )
```

Code text:

```
> Artist_Popular <- spotify_songs %>% count(artist, sort = TRUE, name =
"Count")

> Artist_Fil <- Artist_Popular %>% filter(Count >= 15)

> par(mar = c(12, 5, 4, 2)+ 0.1)

> barplot(Artist_Fil$Count,
+         ylab = "Number of songs",
+         col = "#80C4F5",
+         names.arg= Artist_Fil$artist,
+         width= 0.01,
+         ylim = c(0,20),
+         las = 2
+ )
```

R features:

- Plot Type: Bar plot(Horizontal)
- Functions: par, ylab, las, mar, ylim, filter, width, barplot
- Libraries: ggplot2

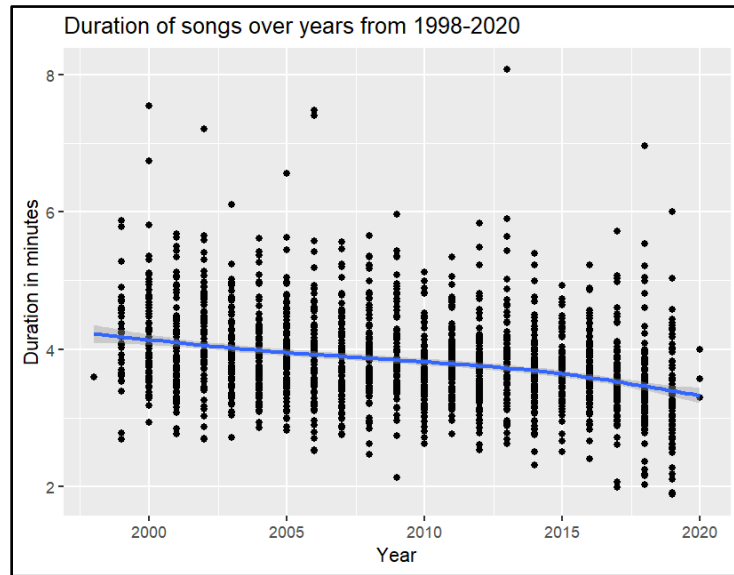
Insights:

The bar plot shows the most popular artists with more than 20 or more songs in the dataset over the period between 2000-2020. The artists with the most popular song are Drake and Rihanna, with more than 20 song count. This implies that they are the most popular artist on Spotify. They were followed by Eminem, having a count of 20 with Calvin Harris. This graphical information provides artists with a ton of useful information that can help them to increase their fanbase and number of listens to their songs ^[3].

As we had the large group names for the Artists category. In that case, we have used a few functions below to increase the margin size:

The **par ()** function sets the parameters, and the **mar** parameter increases the bottom margin. Four values are provided: bottom, left, top, and right respectively. The **las** function rotates the name of the artists to avoid overlapping in the visual ^[4].

5. Has the length of songs changed through the years?



Code screenshot:

```
> song_duration<- transmute(spotify_songs, duration_min = (duration_ms /  
1000)/60 , year)  
> ggplot(song_duration, aes(x=year, y=duration_min)) +  
+ labs(title = "Duration of songs over years from 2000-2020") +  
+ labs(x="Year") + labs(y= "Duration in minutes") +  
+ geom_smooth() +  
+ geom_point()
```

Code text:

```
> song_duration<- transmute(spotify_songs, duration_min = (duration_ms / 1000)/60 ,  
year)  
  
> ggplot(song_duration, aes(x=year, y=duration_min)) +  
  
+ labs(title = "Duration of songs over years from 2000-2020") +  
  
+ labs(x="Year") + labs(y= "Duration in minutes") +  
  
+ geom_smooth() +  
  
+ geom_point()
```

R features:

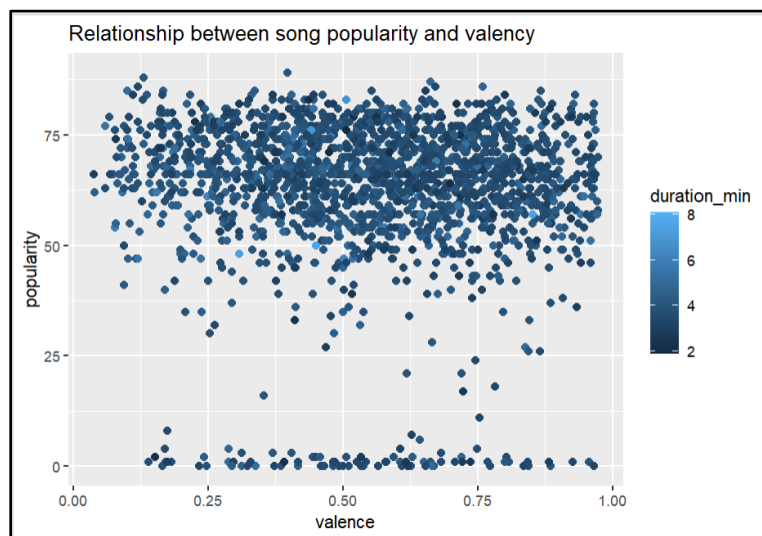
- Plot Type: Scatter Trend Line graph
- Functions: transmute, aesthetics(aes), ggplot, labs, geom_point, geom_smooth(), title
- Libraries: ggplot, tidyverse

Insights:

The graph shows the change in the duration of songs over the years. According to the above scatter trend line graph, songs' average duration has decreased over time, clearly portraying a downward trend. The blue trend line shows a downward regression of song duration; Before 2005, songs' duration was more than 4 minutes, whereas, after 2005, there seems to be a gradual drop in the duration. It is falling from an average of 4 minutes in the early 2000s to an average

closer to 3.5 minutes in 2019. This is because the younger generation may be inclined to listen to songs of shorter duration and hop on to the next music which coincides with the increasing importance of AI metrics and personalized recommendations. This analysis could be helpful for the artist or the song's writer to know the Spotify users' preferences.

6. How do the song's popularity and valency correlate with the duration?



Code Screenshot:

```
> ggplot(spotify_songs, aes(x=valence, y=popularity, color=duration_min)) +  
+ geom_point(size=2) +  
+ ggtitle("Relationship between song popularity and valency")
```

Code text:

```
>ggplot(spotify_songs, aes(x=valence, y=popularity, color=duration_min)) +  
  
+ geom_point(size=2) +  
  
+ ggtitle("Relationship between song popularity and valency")
```

R features:

- Plot Type: Scatter Plot
- Functions: aesthetics(aes), ggplot, ggtitle, geom_point, color
- Libraries: ggplot2, tidyverse

Insights:

The above visual scatter plot shows the relationship between song popularity and their valency. The Valency is the positivism of the song - the greater the valency number, the more positive the song is. As shown, the most positive songs are not necessarily the most popular ones. The majority is between 50-85 on the popularity scale regardless of the valency. However, the duration is a valid criterion in popularity since the longest (light blue) songs are lower on both the popularity scale and valency.

F. Statistical Summary, Script, Functions:

Statistical Summary and Summary function:

We can compute the minimum, 1st quartile, median, mean, 3rd quartile, and maximum for all the numeric variables of a dataset at once using the **summary()** function. The following screenshot displays the screenshot of the entire dataset for the summary function.

```
> summary(spotify_songs)
```

artist	song	explicit	year	popularity	danceability	energy	loudness
Length:1941	Length:1941	Mode :logical	Min. :1998	Min. : 0.00	Min. :0.1290	Min. :0.0549	Min. : -20.514
Class :character	Class :character	FALSE:1404	1st Qu.:2004	1st Qu.:58.00	1st Qu.:0.5810	1st Qu.:0.6240	1st Qu.: -6.490
Mode :character	Mode :character	TRUE :537	Median :2010	Median :66.00	Median :0.6760	Median :0.7390	Median : -5.285
			Mean :2010	Mean :62.83	Mean :0.6678	Mean :0.7215	Mean : -5.514
			3rd Qu.:2015	3rd Qu.:73.00	3rd Qu.:0.7650	3rd Qu.:0.8400	3rd Qu.: -4.168
			Max. :2020	Max. :89.00	Max. :0.9750	Max. :0.9990	Max. : -0.276

speechiness	acousticness	liveness	valence	tempo	genre	duration_min
Min. :0.0232	Min. :0.0000192	Min. :0.0215	Min. :0.0381	Min. : 60.02	Length:1941	Min. :1.883
1st Qu.:0.0397	1st Qu.:0.0135000	1st Qu.:0.0884	1st Qu.:0.3900	1st Qu.: 98.99	Class :character	1st Qu.:3.392
Median :0.0610	Median :0.0558000	Median :0.1240	Median :0.5600	Median :120.03	Mode :character	Median :3.720
Mean :0.1038	Mean :0.1281726	Mean :0.1817	Mean :0.5530	Mean :120.16		Mean :3.810
3rd Qu.:0.1290	3rd Qu.:0.1760000	3rd Qu.:0.2420	3rd Qu.:0.7310	3rd Qu.:134.20		3rd Qu.:4.132
Max. :0.5760	Max. :0.9760000	Max. :0.8530	Max. :0.9730	Max. :210.85		Max. :8.069

The summary function has been applied to the valence, duration_min, and tempo categories in the dataset.

1. Statistical summary of valence:

The summary function has been applied to the duration_min field. Below is the screenshot that displays the output provided by the summary function and with individual functions for the duration_min field.

Code Screenshot:

```
> summary(spotify_songs$valence)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0381 0.3900 0.5600 0.5530 0.7310 0.9730
> min(spotify_songs$valence)
[1] 0.0381
> max(spotify_songs$valence)
[1] 0.973
> mean(spotify_songs$valence)
[1] 0.5529662
> median(spotify_songs$valence)
[1] 0.56
> sd(spotify_songs$valence)
[1] 0.2208454
```

Code Text:

```
> summary(spotify_songs$valence)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0381 0.3900 0.5600 0.5530 0.7310 0.9730

> min(spotify_songs$valence)

[1] 0.0381

> max(spotify_songs$valence)

[1] 0.973

> mean(spotify_songs$valence)

[1] 0.5529662
```

```
> median(spotify_songs$valence)
```

```
[1] 0.56
```

```
> sd(spotify_songs$valence)
```

```
[1] 0.2208454
```

Valency is the positivism of the song - the higher the valency number, the more positive the song is. Based on the statistical summary, the **minimum** “valence” value is **0.0381**. Based on the dataset, lower valence triggers negative emotions, such as sadness, depression, and anger. On the other hand, the **maximum** valency number **0.973** indicates the positivity conveyed by the song. We can infer that the range is from 0 to 1.0. The **mean** valence value is **0.553**, which is comparable to the **median of .56**. This means a balance of both positive and negative songs from the dataset with slight left-skewness.

Lastly, the **standard deviation** of **.221** tells us that the data points are spread out evenly, where 68% of the data points lie approximately between .21 and .77.

2. Statistical summary of duration_min:

The summary function has been applied to the duration_min field. Below is the screenshot that displays the output provided by the summary function and with individual statistical functions duration_min field.

Code Screenshot:

```
> summary(spotify_songs$duration_min)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.883   3.392   3.720   3.810   4.132   8.069
> min(spotify_songs$duration_min)
[1] 1.883333
> max(spotify_songs$duration_min)
[1] 8.0691
> median(spotify_songs$duration_min)
[1] 3.719767
> mean(spotify_songs$duration_min)
[1] 3.809916
> sd(spotify_songs$duration_min)
[1] 0.6541633
```

Code Text:

```
> summary(spotify_songs$duration_min)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.883   3.392   3.720   3.810   4.132   8.069

> min(spotify_songs$duration_min)

[1] 1.883333

> max(spotify_songs$duration_min)

[1] 8.0691

> median(spotify_songs$duration_min)

[1] 3.719767

> mean(spotify_songs$duration_min)

[1] 3.809916

> sd(spotify_songs$duration_min)

[1] 0.6541633
```

The “duration_min” is the duration of any given song in minutes in the Spotify dataset. The statistical summary shows that the **minimum** duration of the song is **1.88 minutes**. The **max** song length is **8.069 minutes**. We can state that the song’s duration range is approximately 2 to 8 minutes. The **mean** duration of the songs is **3.80 minutes** which is comparable to the **median** of **3.71**. Since there isn’t a significant difference in the duration of songs between the mean and median values, we can conclude that there aren’t many outliers. There is a slight right-skewness.

Lastly, the **standard deviation** tells us that the duration of songs varies by **0.65**. This means that 65% of songs are within .65 minutes (or 39 seconds) of the mean of 3.8 (or 3 minutes 48 seconds). This shows the song's duration is not widely spread-out and tends to center on the mean.

3. Statistical summary of tempo:

The summary function has been applied to the tempo field. Below is the screenshot that displays the results provided by the summary function and with individual functions for the tempo field.

Code Screenshot:

```
> summary(spotify_songs$tempo)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 60.02  98.99  120.03  120.16  134.20  210.85
> min(spotify_songs$tempo)
[1] 60.019
> max(spotify_songs$tempo)
[1] 210.851
> median(spotify_songs$tempo)
[1] 120.028
> mean(spotify_songs$tempo)
[1] 120.1584
> sd(spotify_songs$tempo)
[1] 26.99047
```


Code text:

```
> summary(spotify_songs$tempo)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 60.02  98.99 120.03 120.16 134.20 210.85

> min(spotify_songs$tempo)

[1] 60.019

> max(spotify_songs$tempo)

[1] 210.851

> median(spotify_songs$tempo)

[1] 120.028

> mean(spotify_songs$tempo)

[1] 120.1584

> sd(spotify_songs$tempo)

[1] 26.99047
```

“Tempo” in beats per minute (BPM) indicates the number of beats in one minute. In musical terminology, the tempo is the speed or pace of a song. Based on the statistical summary, the **minimum** tempo is **60.01**, which means that beats that beat sound exactly once per second. Based on the dataset lower tempo triggers the song to be at a slow pace and indicates the calmness in the song with its minimum value. On the other hand, the **maximum** tempo number is **210.85** indicating the fastest tempo based on the dataset. We can infer that range is from 60.01 to 210.85. The **mean** tempo of the song is **120.15** which is twice as fast, with two beats per second. The

mean value is quite comparable with the **median** value of **120.02** this means that there is a balance between both indicating fast and slow tempo. This statistic is used in the application where were musical duration, such as in the music industry, must be completely precise.

Lastly, the **standard duration** of **26.99** tells us that the data points are spread out evenly and are not closely clustered around the mean.

Script 1:

This script is used to create a new column with popularity from numerical to classification to be used to create a visualization of the proportion of high, medium, and low-rated songs.

Code Screenshot:

```
> View(spotify_songs)
> View(spotify_songs$popularity_rating)
> f1 <- function()
+ {
+   spotify_songs$popularity_rating <- as.factor(
+     ifelse(spotify_songs$popularity<=60, "Low - Under 60",
+           ifelse(spotify_songs$popularity<=80, "Med - 60-79",
+                 ifelse(spotify_songs$popularity>80, 'High - 80 above', "High - 80 above")
+           ))
+   )
+   print("script executed")
+ }
> f1()
[1] "script executed"
> spotify_songs$popularity_rating
 [1] Med - 60-79    Med - 60-79    Med - 60-79    Med - 60-79    Med - 60-79    Med - 60-79
 [7] Med - 60-79    Med - 60-79    High - 80 above Med - 60-79    Med - 60-79    Med - 60-79
[13] Low - Under 60 Low - Under 60 Low - Under 60 Med - 60-79    Med - 60-79    Low - Under 60
[19] Med - 60-79    Med - 60-79    Med - 60-79    High - 80 above High - 80 above Med - 60-79
[25] Med - 60-79    Low - Under 60 Low - Under 60 Low - Under 60 Med - 60-79    Med - 60-79    Low - Under 60
[31] Low - Under 60 Low - Under 60 Med - 60-79    Low - Under 60 Low - Under 60 Low - Under 60
[37] Low - Under 60 Low - Under 60 Low - Under 60 Med - 60-79    Med - 60-79    Low - Under 60
[43] Low - Under 60 Med - 60-79    Low - Under 60 Med - 60-79    Low - Under 60 Med - 60-79
[49] Low - Under 60 Low - Under 60 Med - 60-79    Low - Under 60 Low - Under 60 Low - Under 60
[55] Low - Under 60 Low - Under 60 Low - Under 60 Low - Under 60 Med - 60-79    Med - 60-79
[61] Med - 60-79    Med - 60-79    Low - Under 60 Low - Under 60 Low - Under 60 Low - Under 60
[67] Med - 60-79    Low - Under 60 Med - 60-79    Low - Under 60 Med - 60-79    Med - 60-79
[73] Low - Under 60 Low - Under 60 Low - Under 60 Low - Under 60 Low - Under 60 Med - 60-79
```

Code Text:

```
'script for categorization of popularity'

> f1 <- function()

+ {

+   spotify_songs$popularity_rating <- as.factor(

+     ifelse(spotify_songs$popularity<=60, "Low - Under 60",

+       ifelse(spotify_songs$popularity<=80, "Med - 60-79",

+         ifelse(spotify_songs$popularity>80,'High - 80 above',"High - 80

above")

+       ))

+   )

+   print("script executed")

+ }

> f1()

[1] "script executed"

> spotify_songs$popularity_rating
```

Below is the result of the above code:

popularity_rating
Med - 60-79
Med - 60-79
Med - 60-79
Med - 60-79
Med - 60-79
Med - 60-79
Med - 60-79
Med - 60-79
High - 80 above
Med - 60-79
Med - 60-79
Med - 60-79
Low - Under 60
Low - Under 60
Low - Under 60
Med - 60-79
Med - 60-79

Script 2:

This script is used to create a new column from the duration_ms. We used the transmute function to create the new column duration_min and apply a function to convert from milliseconds to minutes from the original spotify_songs dataset. Finally, we filtered the new duration_min column with the year column into a new dataset called song_duration. This will be used for visualization purposes to make the units more recognizable and understandable to the average viewer.

Code Screenshot:

```
> song_duration<- transmute(spotify_songs, duration_min = (duration_ms / 1000)/60 , year)
> View(song_duration)
> head(song_duration)
  duration_min year
1    3.519333 2000
2    2.784433 1999
3    4.175767 1999
4    3.741550 2000
5    3.309450 2007
6    2.508883 2015
```

Code text:

```
> song_duration<- transmute(spotify_songs, duration_min =  
(duration_ms / 1000)/60 , year)  
> View(song_duration)
```

Below is the result of the above code:

	duration_min	year
1	3.519333	2000
2	2.784433	1999
3	4.175767	1999
4	3.741550	2000
5	3.309450	2007
6	2.508883	2015
7	3.342667	2000
8	4.228883	1999
9	4.736667	2000
10	4.309333	2000
11	4.522217	1999
12	5.119217	2001
13	3.979317	2011
14	4.481050	1999
15	5.105550	2000

User-Defined Function:

The user-defined function is specific to what the user requires, and once created, it can be used like the built-in function. The user-defined function is used to separate comma-delimited values in a column into its own column. The value passed to the function is the column name. In this case, the value that is passed to the f2 function is 'genre'.

Code Screenshot:

```
> f2 <- function(column){  
+   separate_genre <- separate(spotify_songs,column,c("G1","G2","G3"),sep=',')  
+   View(separate_genre)  
+ }  
> f2('genre')  
Warning messages:  
1: Expected 3 pieces. Additional pieces discarded in 7 rows [321, 336, 513, 777, 862, 1101, 1107].  
2: Expected 3 pieces. Missing pieces filled with 'NA' in 1538 rows [1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, ...].
```

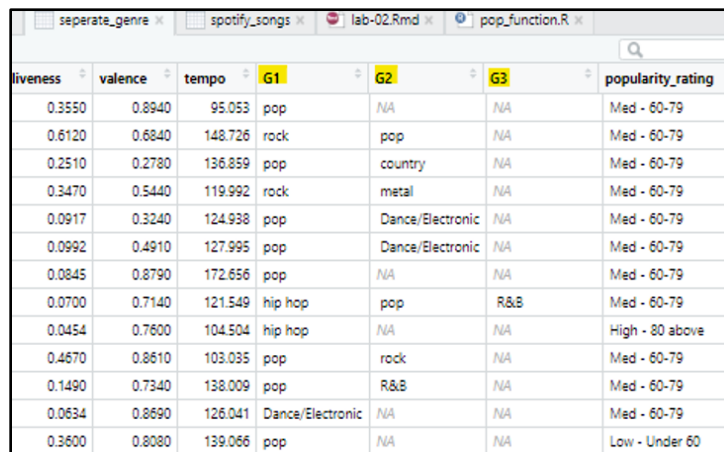
Code Text:

'f2 function is used separate comma delimited values into its column.'

```
f2 <- function(column){  
  separate_genre <- separate(spotify_songs,column,c("G1","G2","G3"),sep=',')  
  View(separate_genre)  
}  
f2('genre')
```

Note: Warning messages are expected as the columns have NA roles

Below is the result of the above code:



liveness	valence	tempo	G1	G2	G3	popularity_rating
0.3550	0.8940	95.053	pop	NA	NA	Med - 60-79
0.6120	0.6840	148.726	rock	pop	NA	Med - 60-79
0.2510	0.2780	136.859	pop	country	NA	Med - 60-79
0.3470	0.5440	119.992	rock	metal	NA	Med - 60-79
0.0917	0.3240	124.938	pop	Dance/Electronic	NA	Med - 60-79
0.0992	0.4910	127.995	pop	Dance/Electronic	NA	Med - 60-79
0.0845	0.8790	172.656	pop	NA	NA	Med - 60-79
0.0700	0.7140	121.549	hip hop	pop	R&B	Med - 60-79
0.0454	0.7600	104.504	hip hop	NA	NA	High - 80 above
0.4670	0.8610	103.035	pop	rock	NA	Med - 60-79
0.1490	0.7340	138.009	pop	R&B	NA	Med - 60-79
0.0634	0.8690	126.041	Dance/Electronic	NA	NA	Med - 60-79
0.3600	0.8080	139.066	pop	NA	NA	Low - Under 60

G. References:

1. Spotify. (2022, November 21). In *Wikipedia*. <https://en.wikipedia.org/wiki/Spotify>.
2. Fernandez, Nick. “What Is Spotify? Here's Everything You Need to Know in 2022.” *Android Authority*, 6 Dec. 2022, <https://www.androidauthority.com/what-is-spotify-1129032/>.
3. Fruci, Chris. “Spotify Helps Artists Grow Their Global Fan Bases through Customer-Facing Metrics.” *Keen*, 22 Jan. 2020, <https://keen.io/blog/spotify-helps-artists-grow-their-global-fan-bases-through-customer-facing-metrics/>.
4. Holtz, Yan. “Advanced R Barplot Customization.” – *The R Graph Gallery*, <https://r-graph-gallery.com/210-custom-barplot-layout.html>.