Kaggle Rock&Roll

JCB

2023-05-09

Rock and Roll-Data Analysis with GGplot in R

Introduction

Purpose- To exhibit proficiency in R's package ggplot2(ggplot). This package allows the users to visualize data in different and interesting ways.

Load the Packages

Load the following packages:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.
2 --
## v ggplot2 3.4.1     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.4     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts
() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

Tidyverse: The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures. ggplot2: ggplot2 is a system for creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. lubridate: Lubridate makes it easier to do the things R does with date-times and possible to do the things R does not.

Read the Data

```
Rock.df <- read.csv("history_of_rock.csv")
```

Rock.df is now a dataframe that contains the data in the “history_of_rock.csv” file downloaded from Kaggle. https://www.kaggle.com/datasets/lukaszamora/history-of-rock-19502020 (https://www.kaggle.com/datasets/lukaszamora/history-of-rock-19502020)

Filter and Order Data into New Variables

```
Pop_Order.df <- Rock.df[with(Rock.df,order(-popularity)),]
Top100.df <- Pop_Order.df[1:100,]
Fav_Bands.df <- filter(Rock.df, artist == 'Green Day' |artist== 'U2' | artist=
='Led Zeppelin' | artist=='AC/DC' | artist=='The Beatles')
```

This gives me the data frame ordered by popularity in descending order. Top100.df now contains the top 100 most popular songs in the dataframe. Fav_Bands.df now includes 5 bands. This is just to illustrate the concept of filtering and ordering dataframes.

Glimpse the data

Get an idea for what sort of data you are working with. I prefer glimpse(), but you can get a look into your data using other functions such as: head(), colnames(), summarize()

```
glimpse(Top100.df)
```

```
## Rows: 100
## Columns: 18
## $ index            <int> 16, 85, 96, 116, 1116, 2153, 2619, 5, 358, 502, 158
1,~
## $ name             <chr> "Back In Black", "In the End", "Highway to Hell",
"Ye~
## $ artist           <chr> "AC/DC", "Linkin Park", "AC/DC", "Coldplay", "Coldp
la~
## $ release_date     <int> 1980, 2000, 1979, 2000, 2002, 2011, 1992, 1976, 198
2,~
## $ length           <dbl> 4.258217, 3.614667, 3.473333, 4.446217, 5.160000,
3.9~
## $ popularity       <int> 84, 84, 84, 84, 84, 84, 84, 83, 83, 83, 83, 82, 8
2, 8~
## $ danceability     <dbl> 0.310, 0.556, 0.574, 0.429, 0.557, 0.733, 0.754, 0.
57~
## $ acousticness     <dbl> 0.011000, 0.009580, 0.061000, 0.002390, 0.731000,
0.1~
## $ danceability.1   <dbl> 0.310, 0.556, 0.574, 0.429, 0.557, 0.733, 0.754, 0.
57~
## $ energy           <dbl> 0.700, 0.864, 0.913, 0.661, 0.442, 0.710, 0.424, 0.
50~
## $ instrumentalness <dbl> 9.65e-03, 0.00e+00, 1.58e-03, 1.21e-04, 1.46e-05,
1.1~
## $ key              <int> 9, 3, 6, 11, 5, 5, 2, 2, 11, 6, 9, 7, 7, 4, 9, 10,
2,~
## $ liveness         <dbl> 0.0828, 0.2090, 0.1560, 0.2340, 0.1100, 0.0956, 0.0
65~
## $ loudness         <dbl> -5.678, -5.870, -4.793, -7.227, -7.224, -5.849, -8.
46~
## $ speechiness      <dbl> 0.0470, 0.0584, 0.1330, 0.0281, 0.0243, 0.0292, 0.0
36~
## $ tempo            <dbl> 188.386, 105.143, 115.728, 173.372, 146.277, 127.97
5,~
## $ time_signature   <int> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4,~
## $ valence          <dbl> 0.763, 0.400, 0.423, 0.285, 0.213, 0.965, 0.806, 0.
60~
```

```
glimpse(Fav_Bands.df)
```

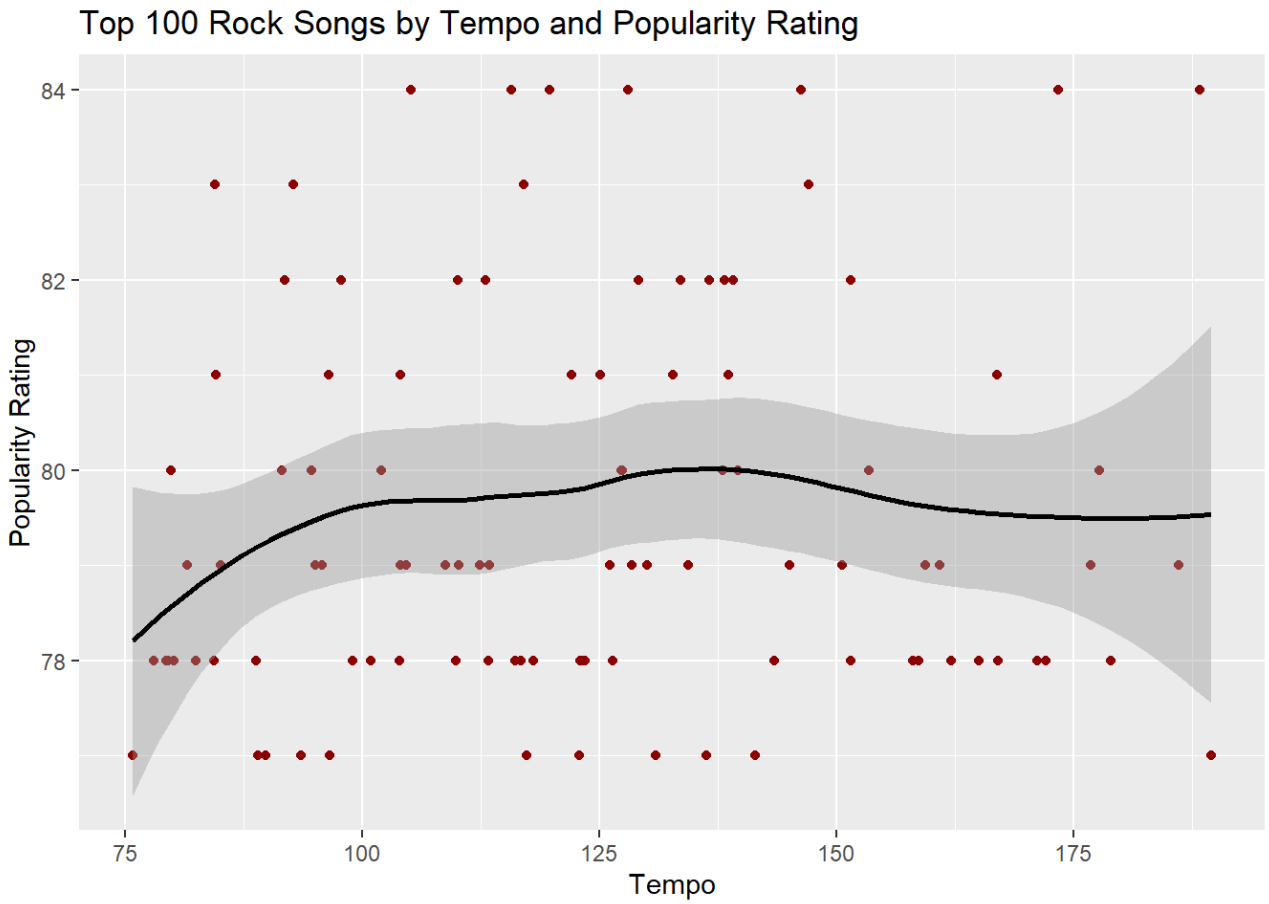
```
## Rows: 225
## Columns: 18
## $ index            <int> 1, 7, 9, 12, 16, 19, 21, 26, 33, 39, 41, 42, 43, 4
4, ~
## $ name             <chr> "Stairway to Heaven - Remaster", "Whole Lotta Love
- ~
## $ artist           <chr> "Led Zeppelin", "Led Zeppelin", "U2", "The Beatle
s", ~
## $ release_date     <int> 1971, 1969, 1991, 2000, 1980, 1987, 1987, 1975, 198
0,~
## $ length           <dbl> 8.047167, 5.564883, 4.603100, 7.094217, 4.258217,
4.9~
## $ popularity       <int> 78, 77, 76, 76, 84, 79, 78, 69, 80, 59, 71, 67, 7
3, 7~
## $ danceability     <dbl> 0.338, 0.412, 0.392, 0.386, 0.310, 0.540, 0.564, 0.
48~
## $ acousticness     <dbl> 5.80e-01, 4.84e-02, 2.45e-01, 1.12e-02, 1.10e-02,
2.0~
## $ danceability.1   <dbl> 0.338, 0.412, 0.392, 0.386, 0.310, 0.540, 0.564, 0.
48~
## $ energy           <dbl> 0.340, 0.902, 0.534, 0.607, 0.700, 0.429, 0.774, 0.
53~
## $ instrumentalness <dbl> 3.20e-03, 1.31e-01, 1.04e-03, 1.38e-05, 9.65e-03,
3.5~
## $ key              <int> 9, 9, 0, 10, 9, 2, 1, 2, 7, 4, 6, 9, 9, 0, 4, 4,
4, 3~
## $ liveness         <dbl> 0.1160, 0.4050, 0.1550, 0.0880, 0.0828, 0.1410, 0.0
86~
## $ loudness         <dbl> -12.049, -11.600, -8.793, -7.700, -5.678, -11.822,
-9~
## $ speechiness      <dbl> 0.0339, 0.4050, 0.0369, 0.0261, 0.0470, 0.0285, 0.0
36~
## $ tempo            <dbl> 82.433, 89.740, 181.305, 147.207, 188.386, 110.17
1, 1~
## $ time_signature   <int> 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4,~
## $ valence          <dbl> 0.197, 0.422, 0.325, 0.532, 0.763, 0.113, 0.657, 0.
58~
```

Plot the Data

Now it is time to explore the data using ggplot. ### Graph 1.0

```
ggplot(Top100.df, aes(x= tempo, y= popularity))+
  geom_point(color='darkred')+
  geom_smooth(color='black')+
  labs(title = "Top 100 Rock Songs by Tempo and Popularity Rating",
       x= "Tempo",
       y= "Popularity Rating")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



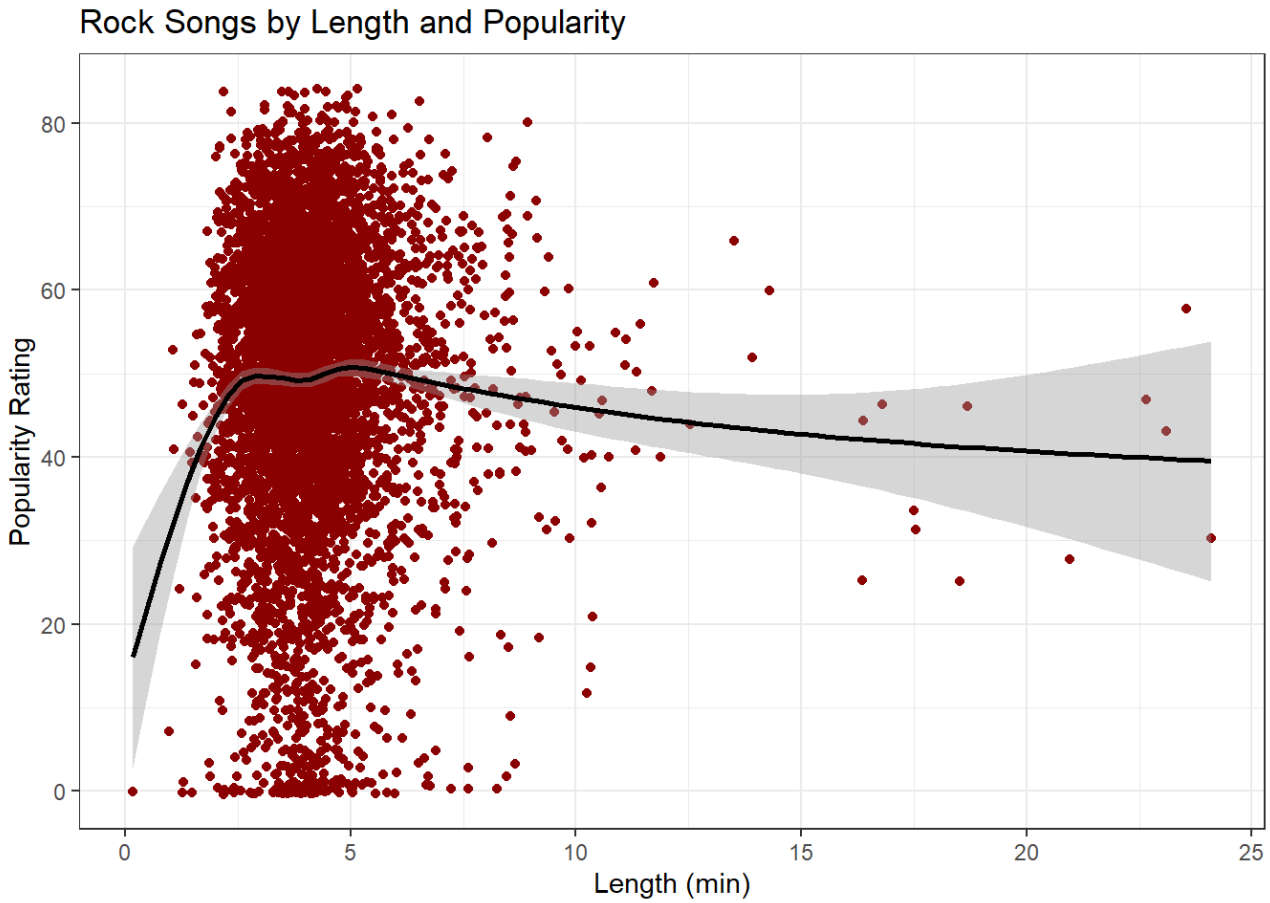
Tempo and popularity stood out as interesting variables to begin looking at. I use ggplot, select my data (Top100.df), define my graphs aesthetics (aes). Next I want to create a scatterplot colored red, and a smoothed conditional means chart colored black. Finally I want to add context through labs.

Graph 1.1

What about the length of songs? Do people hate long songs? Whats the most popular song length?

```
ggplot(Rock.df, aes( x=length, y=popularity))+
  geom_jitter(color='darkred')+
  geom_smooth(color='black')+
  theme_bw()+
  labs(title= "Rock Songs by Length and Popularity",
       x= "Length (min)",
       y= "Popularity Rating")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

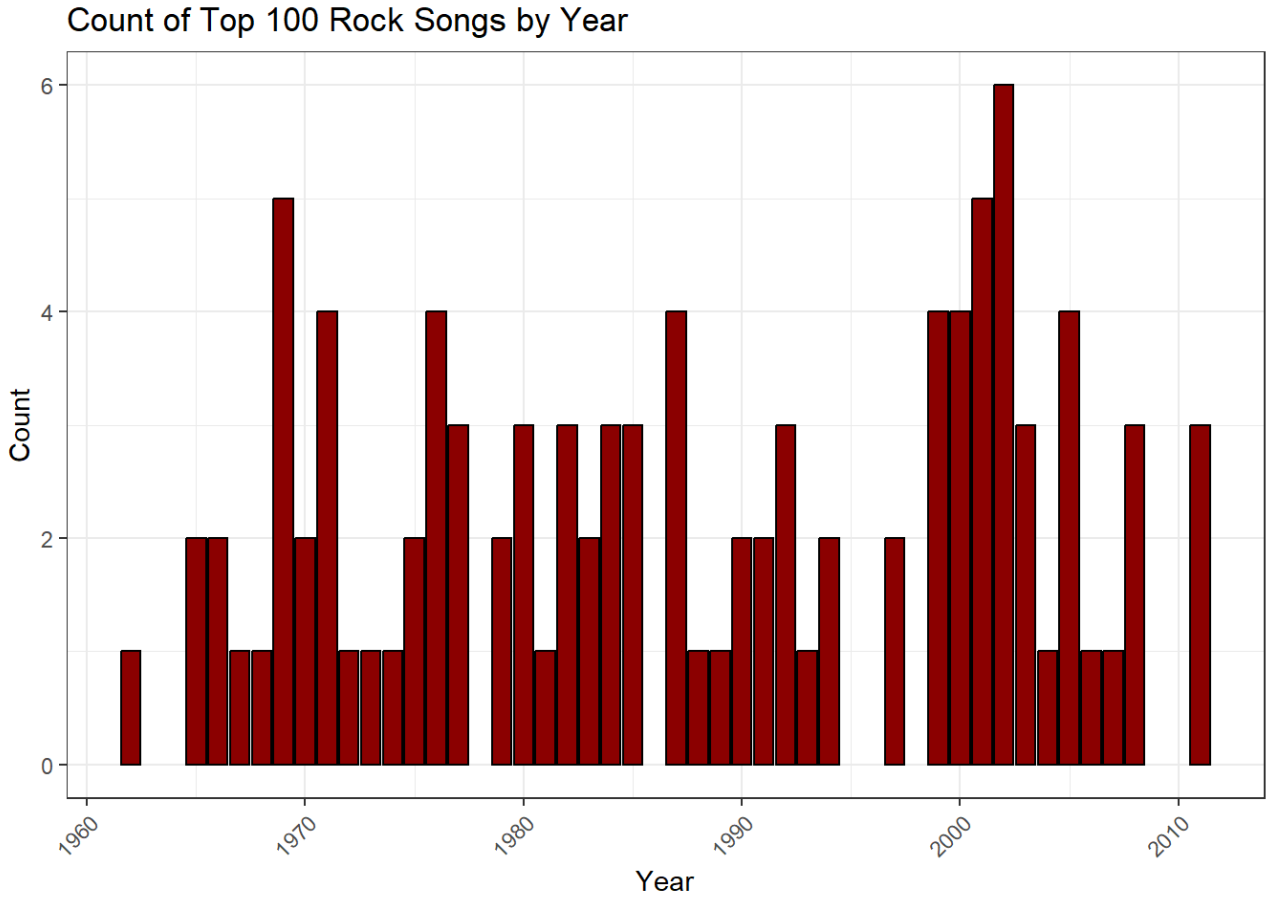


This tells us most of the songs are between 2.5 and 5 minutes. The smooth conditional means chart shows us the most popular average length is around 5 minutes.

Graph 2.0

Now I am curious about the release date of the songs in the top 100. I make a bar graph.

```
ggplot(Top100.df, aes(x=release_date))+
  geom_bar(color='black',fill='darkred')+
  theme_bw()+
  theme(axis.text.x=element_text(angle=45, hjust=1))+
  labs(title = "Count of Top 100 Rock Songs by Year",
       x= "Year",
       y= "Count")
```

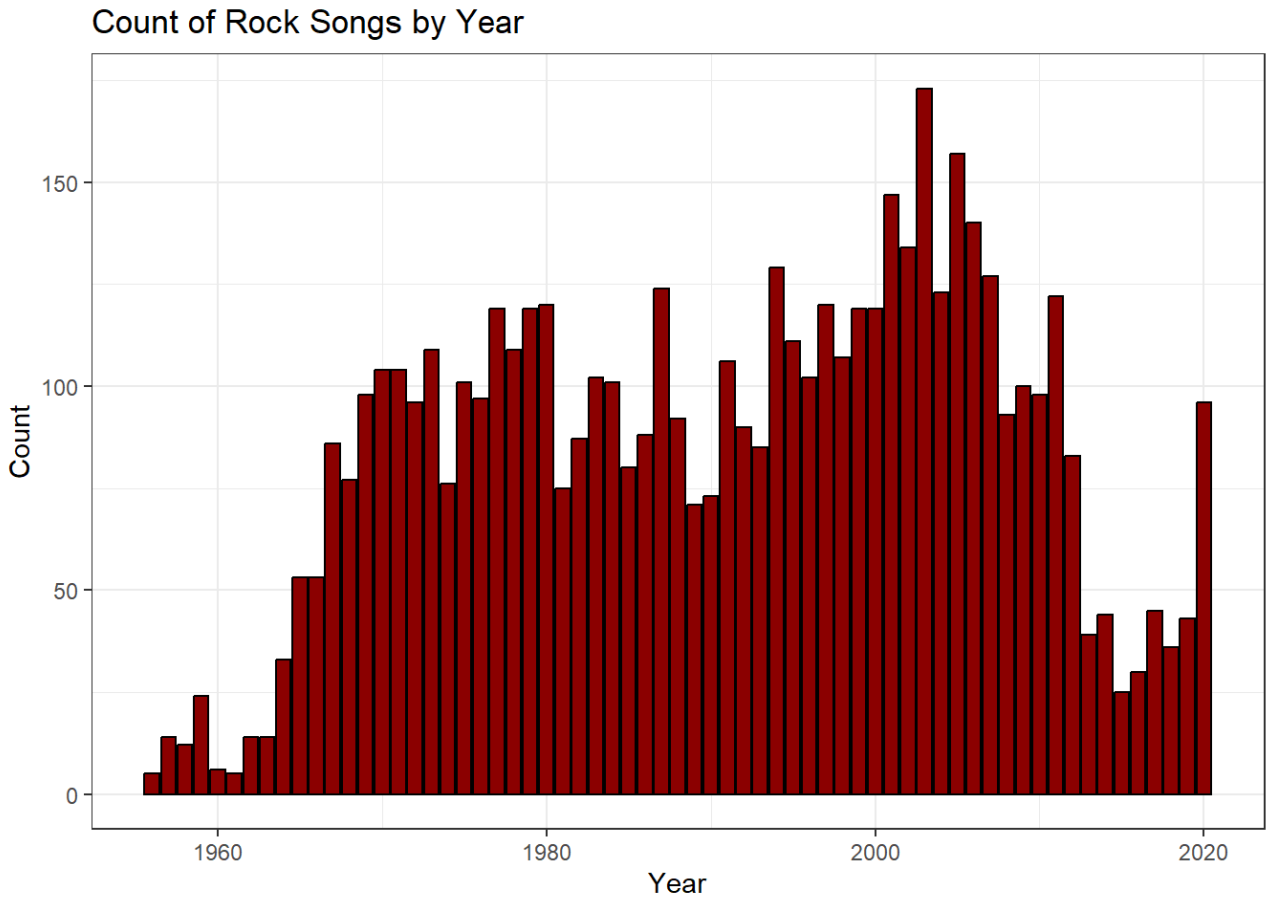


geom_bar only requires 1- defined variable (x in this case). Note the change in the x axis text angle. Looks like the best year for Rock was 2002. Who would have thought!?

Graph 3.1

Now that I know about the release date of the top songs, I am curious about release of ALL the rock songs (in the data set).

```
ggplot(Rock.df, aes(x=release_date))+
  geom_bar(color='black',fill='darkred')+
  theme_bw()+
  labs(title="Count of Rock Songs by Year",
       x="Year",
       y="Count")
```

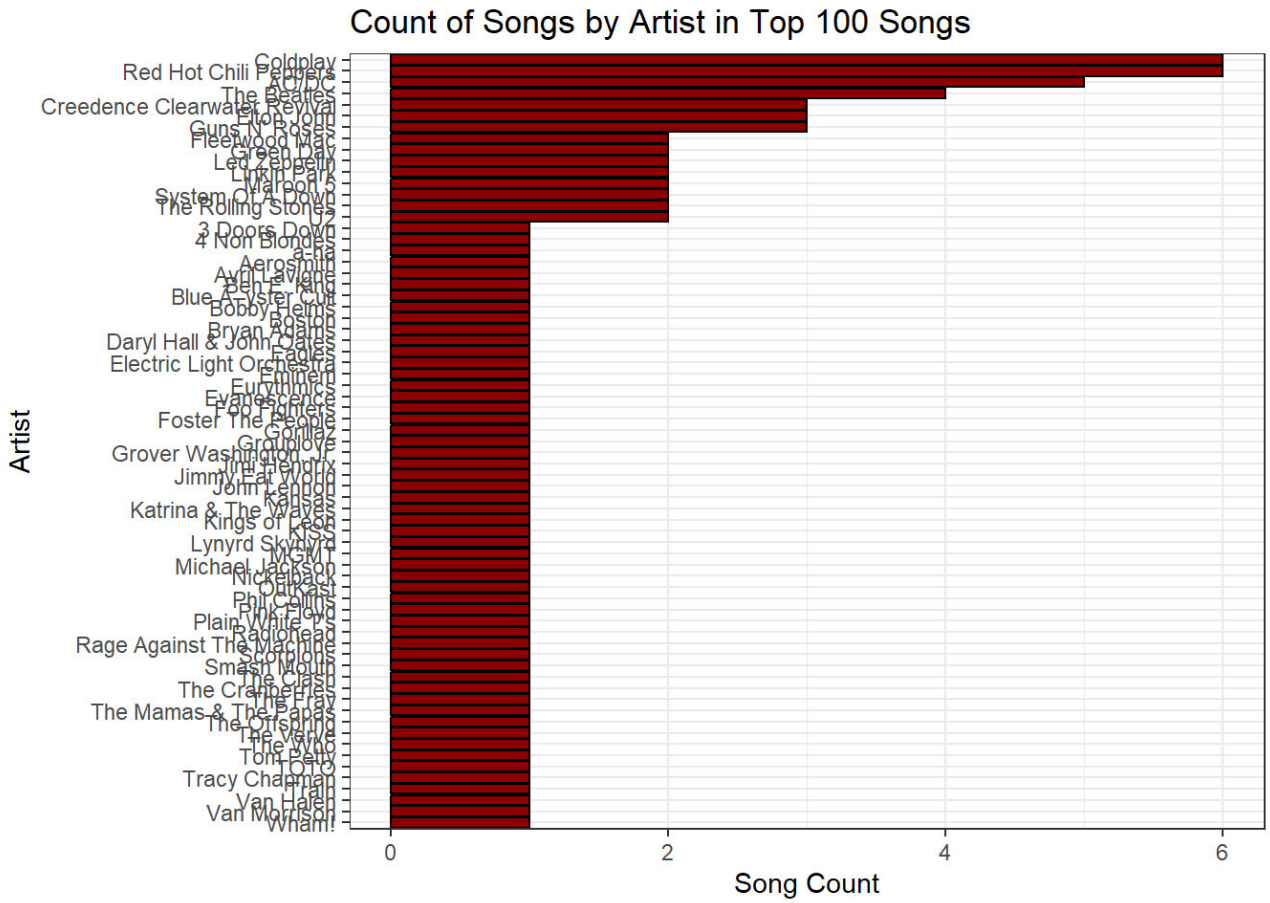


Looks like 2003 was the year with the MOST rock songs released.

Graph 4.0

The Years are intersting, but what about the artists? Who has the most hits? Who has the most popular songs?

```
ggplot(Top100.df,aes(y= fct_rev(fct_infreq(artist))))+
  geom_bar(fill='darkred',color='black')+
  theme_bw()+
  theme(legend.position = "none")+
  labs(title="Count of Songs by Artist in Top 100 Songs",
       x="Song Count",
       y="Artist")
```

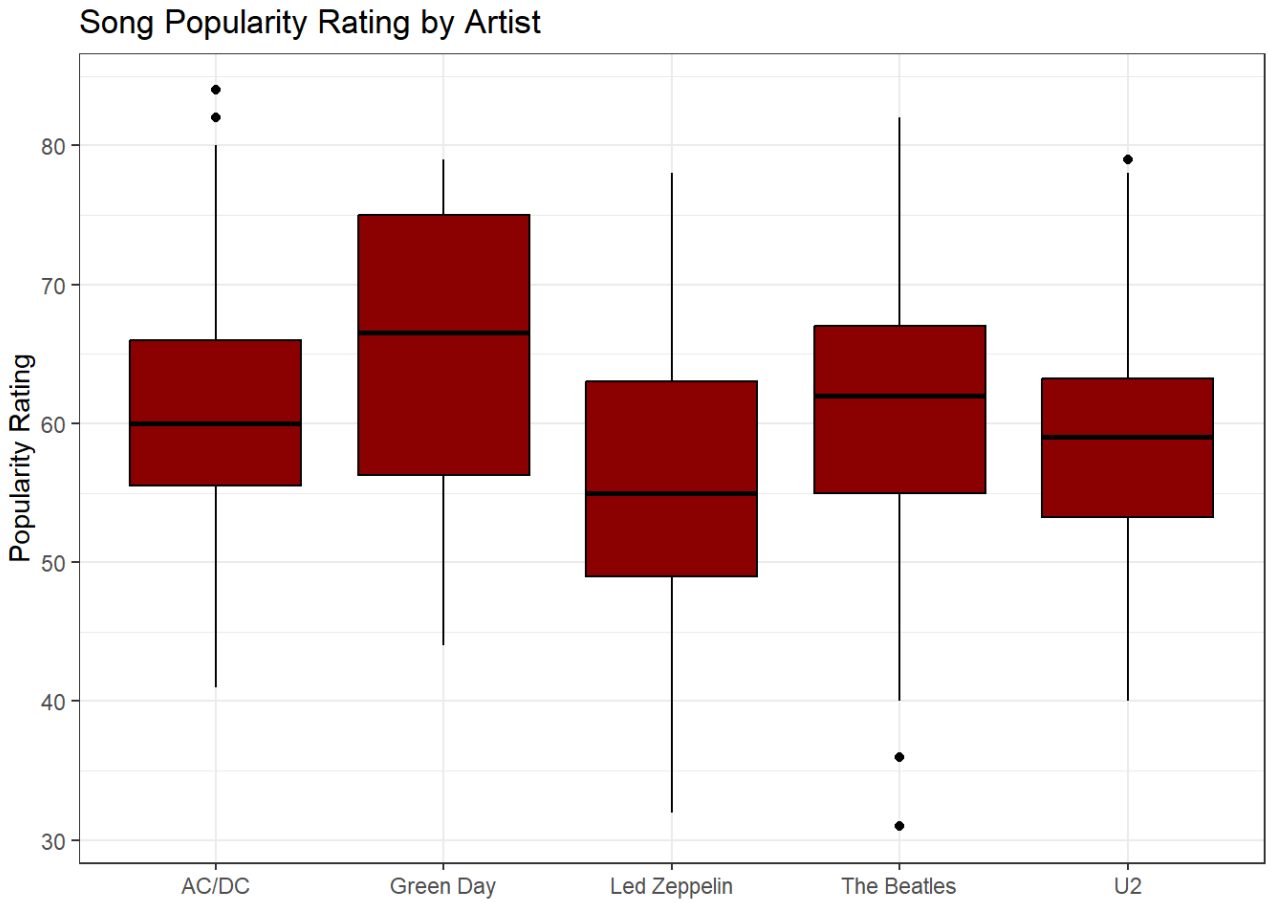


Note the defined y value includes the fct_rev and fct_infreq functions, they allow the frequency of the y value instances to be counted. Coldplay and Red Hot Chili Peppers are the best bands confirmed. Fascinating!

Graph 5.0

This is interesting but I don't know most of these songs. How about I visualize some of the data from my favorite bands?

```
ggplot(Fav_Bands.df, aes(y=popularity, x=artist))+
  geom_boxplot(fill='darkred',color='black')+
  theme_bw()+
  labs(title = "Song Popularity Rating by Artist",
       x=NULL,
       y="Popularity Rating")
```

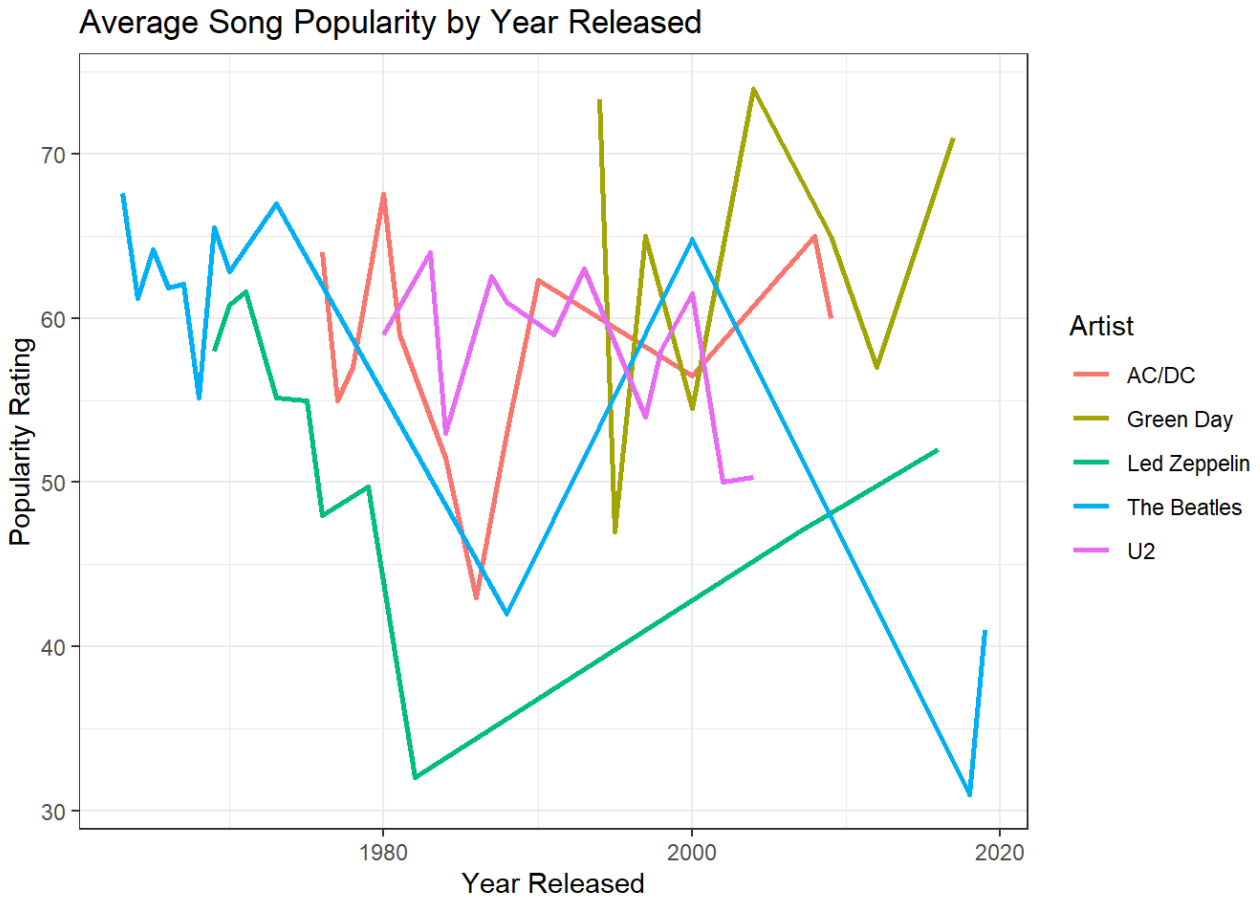


I use Fav_Bands.df as my filtered data set. Since I defined the data to only have 5 artists, I can select 'artist' for my x aesthetic. This makes single continuous variable graphs clean looking and simplified.

Graph 5.1

I like this group of artists, lets look at their average popularity by year released.

```
ggplot(Fav_Bands.df, aes(y=popularity, x=release_date,color=artist))+
  geom_line(stat = "summary",fun="mean", lwd=1)+
  theme_bw()+
  labs(title = "Average Song Popularity by Year Released",
       x= "Year Released",
       y= "Popularity Rating")+
  guides(color = guide_legend(title = "Artist"))
```



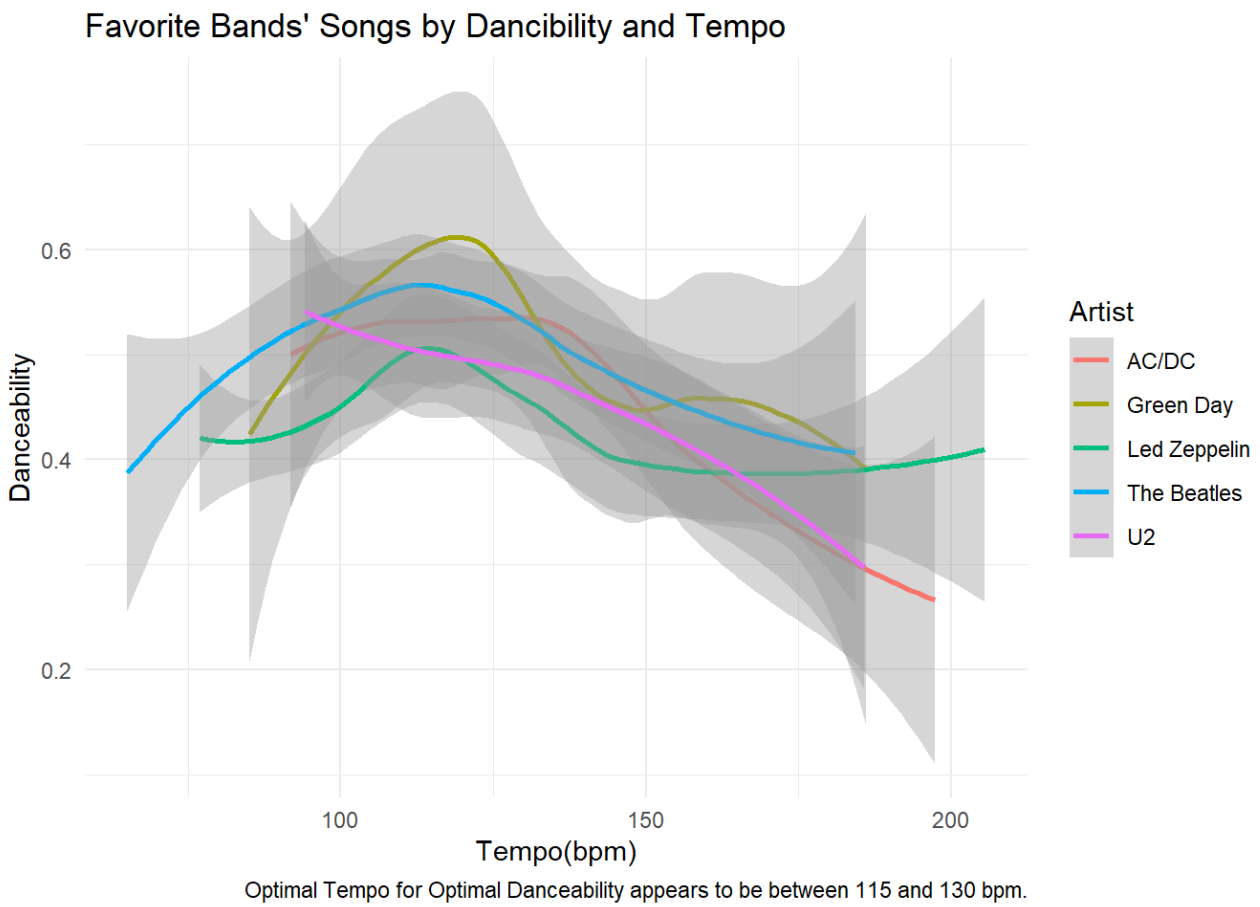
Group by artist by using color in ggplot line, stat"summary",fun="mean" will give you the average rating for that year. You can change the legend title using guides().

Graph 5.2

Let's play with the geom_smooth one again. Let's look for relationships in the data.

```
ggplot(Fav_Bands.df, aes(x=tempo, y=danceability, color=artist))+
  geom_smooth()+
  theme_minimal()+
  labs(title="Favorite Bands' Songs by Dancibility and Tempo",
       x="Tempo(bpm)",
       y= "Danceability",
       caption = "Optimal Tempo for Optimal Danceability appears to be between
115 and 130 bpm.")+
  guides(color=guide_legend(title = "Artist"))
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

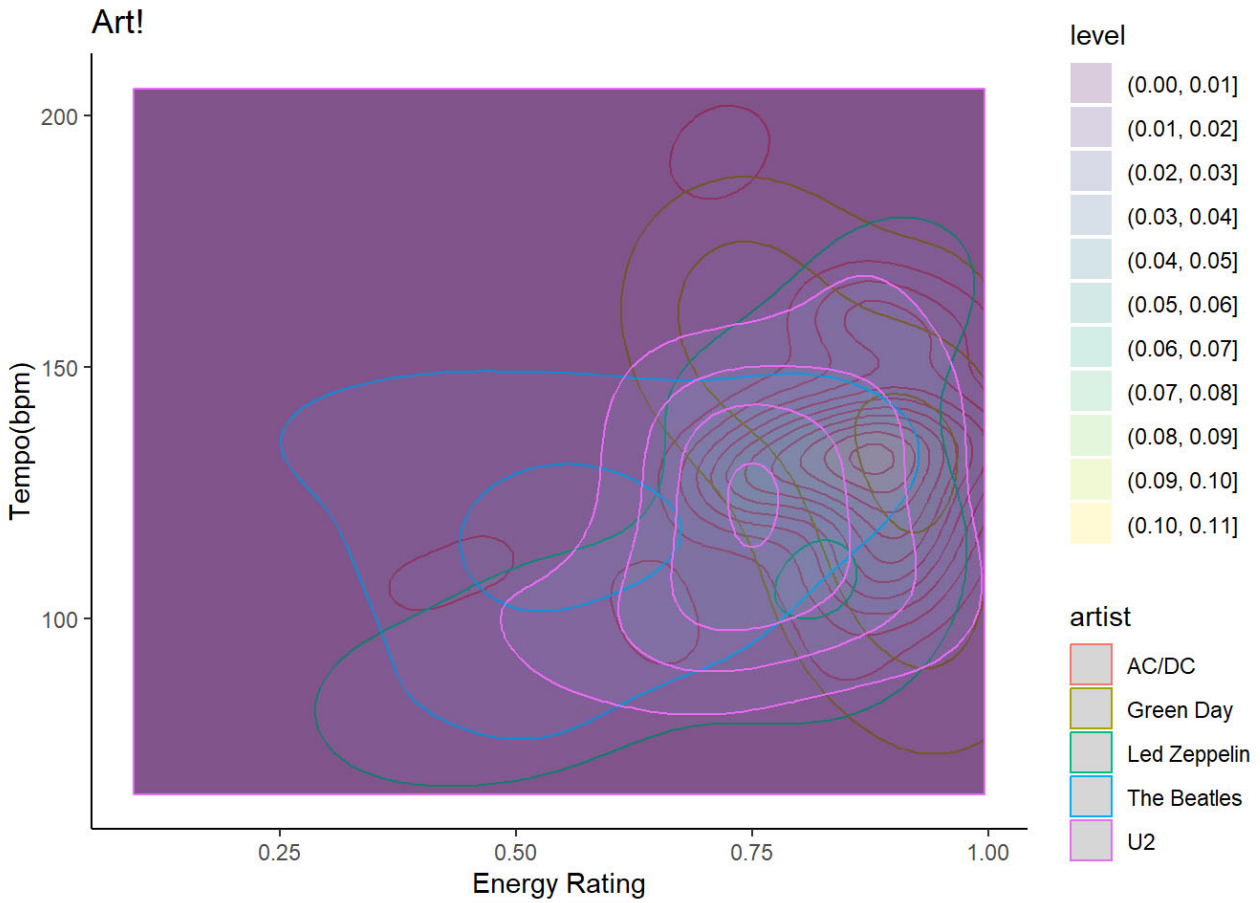


Note the added caption. It's an educated guess! You're ready to make the perfect dance song!

Graph 6.0

Sometimes you won't find any obvious relationships while exploring the ggplot graphs available, and that's OK! It shouldn't be considered wasted time, because you can learn from every graph you make. Even the error messages can help you learn the syntax.

```
ggplot(Fav_Bands.df, aes (energy,tempo,color=artist))+
  geom_density_2d_filled(alpha=.2)+
  theme_classic()+
  labs(title="Art!",
       x="Energy Rating",
       y="Tempo(bpm)")
```


Art!
Tempo(bpm)
200
150
100
0.25
0.50
0.75
1.00
Energy Rating
level
(0.00, 0.01]
(0.01, 0.02]
(0.02, 0.03]
(0.03, 0.04]
(0.04, 0.05]
(0.05, 0.06]
(0.06, 0.07]
(0.07, 0.08]
(0.08, 0.09]
(0.09, 0.10]
(0.10, 0.11]
artist
AC/DC
Green Day
Led Zeppelin
The Beatles
U2