R Notebook - Billboard 100 Data Cleaning & Visualization

Maya Reese Farmer

October 16, 2021

Introduction

I will be demonstrating some data cleaning and visualization techniques using the dplyr and tidyr packages in R. I will also be using the Billboard Hot 100 dataset from kaggle.com

This dataset contains the following variables:

- rank

- song
- artist
- last-week (rank in previous week)

- peak-week (top rank achieved by song)
- weeks-on-board
- date

```
## importing dataset into r
charts <- read.csv("~/Downloads/charts.csv")
attach(charts)
head(charts)
```

```
##   rank                            song                    artist last.week
## 1    1                          Butter                       BTS         1
## 2    2                        Good 4 U            Olivia Rodrigo         2
## 3    3                      Levitating Dua Lipa Featuring DaBaby         4
## 4    4                    Kiss Me More    Doja Cat Featuring SZA         3
## 5    5 Montero (Call Me By Your Name)                Lil Nas X         8
## 6    6                      Bad Habits                Ed Sheeran         5
##   peak.rank weeks.on.board       date
## 1         1              7 2021-07-17
## 2         1              8 2021-07-17
## 3         2             40 2021-07-17
## 4         3             13 2021-07-17
## 5         1             15 2021-07-17
## 6         5              2 2021-07-17
```

Data Cleaning

```
##
library(dplyr)
```

task 1: convert date column to date format

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```
library(stringr)

charts %>% select(date, song, artist) %>% mutate(date = ymd(date)) %>% head(10)
```

```
##          date                          song
## 1  2021-07-17                        Butter
## 2  2021-07-17                      Good 4 U
## 3  2021-07-17                    Levitating
## 4  2021-07-17                  Kiss Me More
## 5  2021-07-17 Montero (Call Me By Your Name)
## 6  2021-07-17                    Bad Habits
## 7  2021-07-17           Leave The Door Open
## 8  2021-07-17                       Peaches
## 9  2021-07-17               Save Your Tears
## 10 2021-07-17                       Deja Vu
##                                         artist
## 1                                          BTS
## 2                               Olivia Rodrigo
## 3                    Dua Lipa Featuring DaBaby
## 4                       Doja Cat Featuring SZA
## 5                                    Lil Nas X
## 6                                   Ed Sheeran
## 7        Silk Sonic (Bruno Mars & Anderson .Paak)
## 8  Justin Bieber Featuring Daniel Caesar & Giveon
## 9                      The Weeknd & Ariana Grande
## 10                              Olivia Rodrigo
```

```r
charts %>% mutate(date = ymd(date)) %>%
  distinct(date) %>%
  mutate(month = floor_date(date, "month"), year = floor_date(date, "year")) %>% head(10)
```

task 2: create 2 new columns containing month and year

```
##          date      month       year
## 1  2021-07-17 2021-07-01 2021-01-01
## 2  2021-07-10 2021-07-01 2021-01-01
## 3  2021-07-03 2021-07-01 2021-01-01
## 4  2021-06-26 2021-06-01 2021-01-01
## 5  2021-06-19 2021-06-01 2021-01-01
## 6  2021-06-12 2021-06-01 2021-01-01
## 7  2021-06-05 2021-06-01 2021-01-01
## 8  2021-05-29 2021-05-01 2021-01-01
## 9  2021-05-22 2021-05-01 2021-01-01
## 10 2021-05-15 2021-05-01 2021-01-01
```

task 3: separate the artist column Here, I want to break up the artist column so that each artist is designated as a primary artist (the first artist) or a featured artist (the 2nd, 3rd, or 4th artist) in different columns. To do that, I'm going to use the separate function in the tidyr package. With this I'm able to designate what characters or values I want to use as separators between columns and I can also name columns. Then, I'll create a new data frame called clean.charts that contains the original data along with the updated date columns and artist columns.

```r
library(tidyr)

clean.charts = charts %>%
    mutate(date = ymd(date)) %>%
    mutate(month = floor_date(date, "month"), year = floor_date(date,
        "year")) %>%
    separate(artist, c("primary.artist", "featured.artist", "featured.artist2",
        "featured.artist3"), "\\sFeaturing\\s|\\s&\\s|,\\s|\\sX\\s|\\sx\\s|\\sWith\\s",
        remove = FALSE)
```

```
## Warning: Expected 4 pieces. Additional pieces discarded in 665 rows [888, 943,
## 3392, 3490, 3591, 3679, 3773, 3795, 3875, 3897, 3992, 4076, 4100, 4176, 4198,
## 4275, 4299, 4364, 4398, 4442, ...].
```

```
## Warning: Expected 4 pieces. Missing pieces filled with `NA` in 326334 rows [1,
## 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```r
clean.charts %>%
    select(song, primary.artist, featured.artist, featured.artist2,
        featured.artist3) %>%
    head(10)
```

```
##                              song         primary.artist featured.artist
## 1                          Butter                    BTS            <NA>
## 2                        Good 4 U         Olivia Rodrigo            <NA>
## 3                      Levitating               Dua Lipa          DaBaby
## 4                    Kiss Me More               Doja Cat             SZA
## 5  Montero (Call Me By Your Name)              Lil Nas X            <NA>
## 6                      Bad Habits             Ed Sheeran            <NA>
## 7            Leave The Door Open Silk Sonic (Bruno Mars Anderson .Paak)
## 8                         Peaches          Justin Bieber   Daniel Caesar
## 9                 Save Your Tears             The Weeknd   Ariana Grande
## 10                        Deja Vu         Olivia Rodrigo            <NA>
##    featured.artist2 featured.artist3
## 1              <NA>             <NA>
## 2              <NA>             <NA>
## 3              <NA>             <NA>
## 4              <NA>             <NA>
## 5              <NA>             <NA>
## 6              <NA>             <NA>
## 7              <NA>             <NA>
## 8            Giveon             <NA>
## 9              <NA>             <NA>
## 10             <NA>             <NA>
```

Overall, this is a pretty good way to separate these columns, but it isn't perfect. The original artist column is very messy and artists aren't separated the same way. For example, one of the separators I had to use was a comma. For artists that have a comma in their name, like Tyler, The Creator, our code will break up his name into Tyler in the primary artist column and The Creator in the featured artist column. This is something to remember if you're trying to isolate a single artist for further analysis later. But for the most part, the separate function did a pretty good job of breaking up the data into appropriate columns.

task 4: pivot table Now that the data are fairly clean, I want to collapse the primary artist and featured artist columns (total 4 columns) into just 2 columns using the pivot_longer function in tidyr. This format will be more conducive for analyses in the future.

```
pivot.artist = clean.charts %>%
    pivot_longer(primary.artist:featured.artist3, names_to = "artist.type",
        values_to = "artist.name")
pivot.artist %>%
    select(song, artist.type, artist.name) %>%
    head(15)
```

```
## # A tibble: 15 x 3
##    song         artist.type      artist.name
##    <chr>        <chr>            <chr>
##  1 Butter       primary.artist   BTS
##  2 Butter       featured.artist  <NA>
##  3 Butter       featured.artist2 <NA>
##  4 Butter       featured.artist3 <NA>
##  5 Good 4 U     primary.artist   Olivia Rodrigo
##  6 Good 4 U     featured.artist  <NA>
##  7 Good 4 U     featured.artist2 <NA>
##  8 Good 4 U     featured.artist3 <NA>
##  9 Levitating   primary.artist   Dua Lipa
## 10 Levitating   featured.artist  DaBaby
## 11 Levitating   featured.artist2 <NA>
## 12 Levitating   featured.artist3 <NA>
## 13 Kiss Me More primary.artist   Doja Cat
## 14 Kiss Me More featured.artist  SZA
## 15 Kiss Me More featured.artist2 <NA>
```

Now that we have a new table where the artist name and type have been pivoted/collapsed into 2 columns, we can easily query artists and determine the number of songs where they were features or the primary artist.

```
## example using SZA
pivot.artist %>% distinct(song, artist.type,artist.name) %>%
  filter(artist.name == "SZA")
```

```
## # A tibble: 14 x 3
##    song                artist.type      artist.name
##    <chr>               <chr>            <chr>
##  1 Kiss Me More        featured.artist  SZA
##  2 Good Days           primary.artist   SZA
##  3 Hit Different       primary.artist   SZA
##  4 The Other Side      primary.artist   SZA
##  5 Staring At The Sun  featured.artist  SZA
##  6 Just Us             featured.artist  SZA
##  7 Power Is Power      primary.artist   SZA
##  8 All The Stars       featured.artist  SZA
##  9 I Do                featured.artist  SZA
## 10 Broken Clocks       primary.artist   SZA
## 11 What Lovers Do      featured.artist  SZA
## 12 The Weekend         primary.artist   SZA
## 13 Love Galore         primary.artist   SZA
## 14 Homemade Dynamite   featured.artist3 SZA
```

task 5: join genre Finally, I want to join another data frame, the mtv10000 data, to my clean.charts data frame. The mtv10000 dataset will provide the genre of music for each primary artist. Because I want to retain all the rows in my 'clean.charts' data frame, I'm going to perform a left join. A left join keeps all data in the left table (clean.charts) and only the rows that match the condition in the second table will be added (NAs will be added in rows where there is no match in the second table). Essentially, if an artist in the clean.charts table is also in the mtv10000 table, then a genre will be returned. If an artist in the clean.charts table is not in the mtv10000 table, then a NA will be returned.

```
mtv10000 <- read.csv("~/Downloads/mtv10000.csv")
attach(mtv10000)

## it's VERY important to trim the white spaces from the
## data before performing any joins
clean.charts$primary.artist = trimws(clean.charts$primary.artist)
mtv10000$name = trimws(mtv10000$name)
attach(clean.charts)
```

```
## The following objects are masked from charts:
##
##     artist, date, last.week, peak.rank, rank, song, weeks.on.board
```

```r
attach(mtv10000)
```

```
## The following objects are masked from mtv10000 (pos = 4):
##
##     facebook, genre, mtv, name, twitter, website
```

```r
clean.charts %>%
    left_join(mtv10000, by = c(primary.artist = "name")) %>%
    select(primary.artist, genre) %>%
    head(10)
```

```
##            primary.artist             genre
## 1                     BTS              <NA>
## 2          Olivia Rodrigo              <NA>
## 3                Dua Lipa              <NA>
## 4                Doja Cat              <NA>
## 5               Lil Nas X              <NA>
## 6              Ed Sheeran Singer/Songwriter
## 7  Silk Sonic (Bruno Mars              <NA>
## 8           Justin Bieber               Pop
## 9              The Weeknd          R&B/Soul
## 10         Olivia Rodrigo              <NA>
```

So, the left join was successful, but as you can see there are a lot of missing artists in the mtv10000 dataset. This means that there are a lot of missing data points in the genre column of our joined data. If this were for a real project, the mtv10000 data would not be the best joining genre because the lack of data might skew our analyses. But it's fine for this informal project.
Now, we can use our cleaned and joined data to create a few visualizations!

Data Visualization

First, let's look at how the popularity of the top 5 genres have changed over time.

```r
## create new dataframe from join
genre.join = clean.charts %>% left_join(mtv10000, by = c("primary.artist" = "name"))

## find the top 5 genres in df
top5.genre = genre.join %>% select(date, primary.artist, genre) %>%
  filter(!is.na(genre)) %>%
  distinct() %>%
  count(genre) %>%
  top_n(5) %>%
  pull(genre)
```

```
## Selecting by n
```

```r
top5.genre
```

```
## [1] "Country"     "Hip-Hop/Rap" "Pop"         "R&B/Soul"    "Rock"
```

```
## create new dataframe only including yearly counts of top
## 5 genres
genre.year = genre.join %>%
    select(year, song, genre) %>%
    filter(genre %in% top5.genre) %>%
    count(year, genre)

## load ggplot2 and cowplot for graphs
library(ggplot2)
library(cowplot)
```

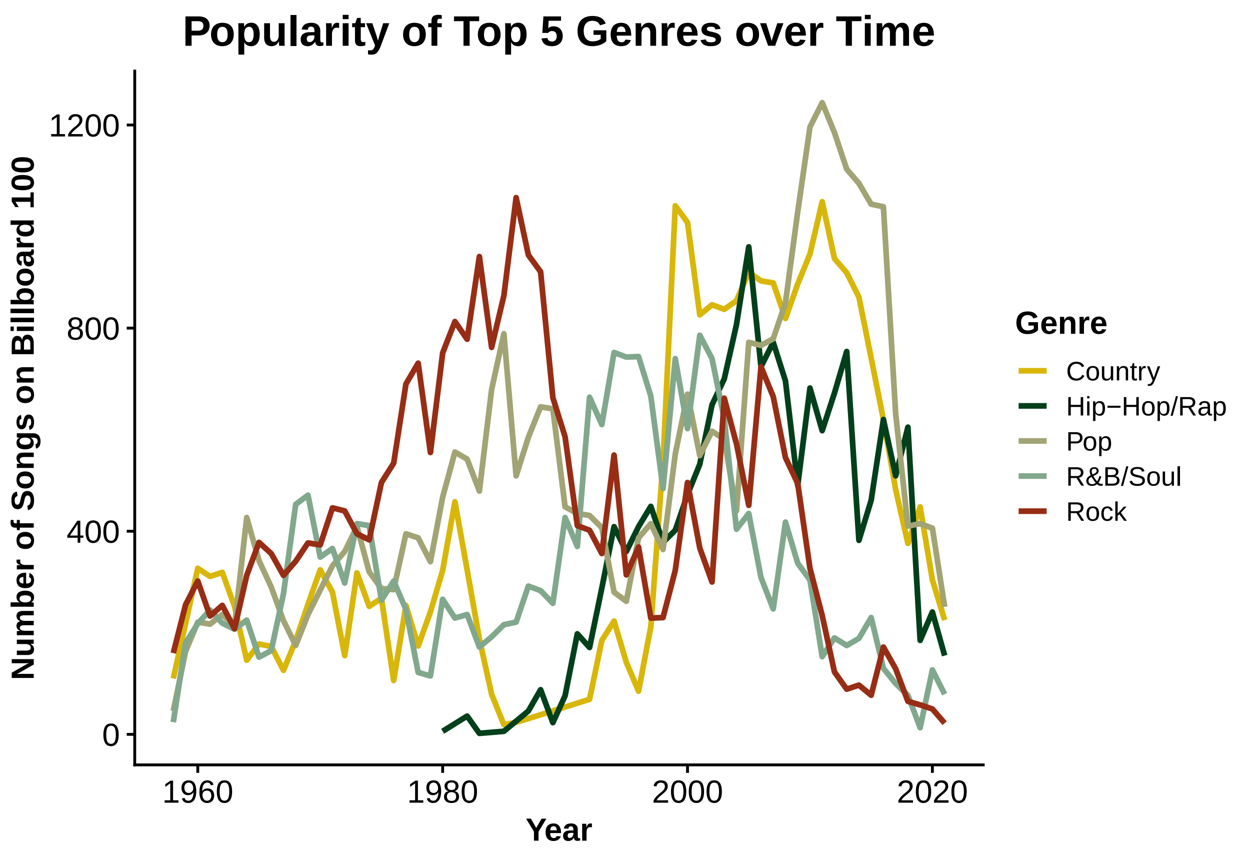
```
##
## Attaching package: 'cowplot'

## The following object is masked from 'package:lubridate':
##
##     stamp
```

```
library(wesanderson)

cavalcanti = wes_palette("Cavalcanti1")

ggplot(genre.year, aes(year, n, color = genre)) + geom_line(size = 1) +
    theme_cowplot(12) + scale_color_manual(values = cavalcanti,
    name = "Genre") + labs(x = "Year", y = "Number of Songs on Billboard 100",
    title = "Popularity of Top 5 Genres over Time") + theme(plot.title = element_text(size = 16,
    face = "bold", hjust = 0.5), axis.text.x = element_text(size = 12),
    axis.title = element_text(face = "bold"), axis.text.y = element_text(size = 12),
    legend.title = element_text(face = "bold"))
```



It's very important to note that I cannot draw any concrete conclusions based on this figure because of the missing genre data from our join. It's clear that the mtv10000 data was missing new artists from the past 10 years because there is an overall decline in all genres starting around 2010 to the present (based on the figure). However, it's still pretty interesting to see how the popularity of the top 5 genres have shanged over time. For example, the popularity of country music increased significantly in the early 2000s. Pop music had the highest popularity overall around 2010. Also, Hip-Hop and Rap was a new genre on the Billboard 100 around 1980, but it is still in the top 5 genres in the dataset (very popular).

Next, let's look at the top 5 songs that were repeated ranked in the top 10 in 2019.

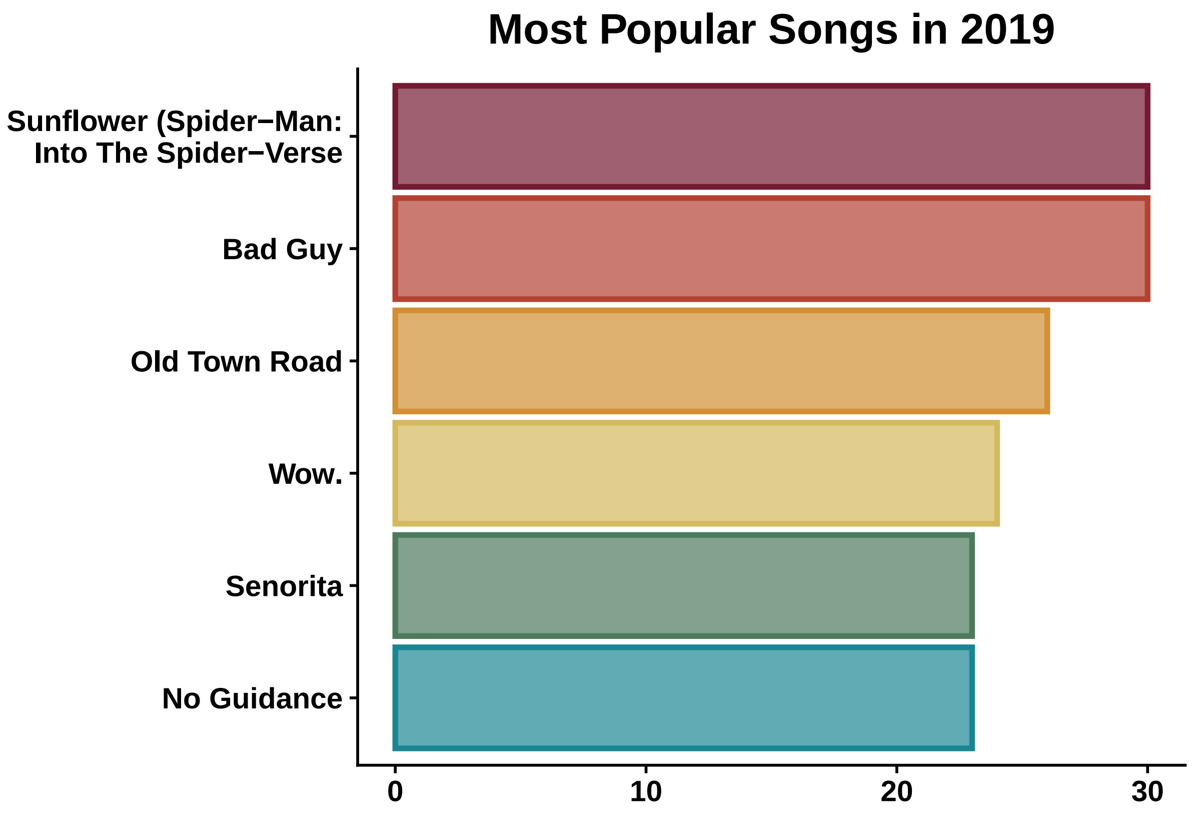
```r
top5.rank = clean.charts %>%
  select(rank, song, year) %>% filter(year == "2019-01-01", rank < 11) %>%
  count(song) %>% arrange(desc(n)) %>% top_n(5)
```

```
## Selecting by n
```

```r
top5.rank
```

```
##                                        song  n
## 1                                   Bad Guy 30
## 2 Sunflower (Spider-Man: Into The Spider-Verse) 30
## 3                            Old Town Road 26
## 4                                     Wow. 24
## 5                              No Guidance 23
## 6                                 Senorita 23
```

```r
ggplot(top5.rank, aes(x = reorder(song, n), y = n, fill = reorder(song,
    n), color = reorder(song, n))) + geom_bar(stat = "identity",
    size = 1) + coord_flip() + theme_cowplot(12) + labs(x = "",
    y = "", title = "Most Popular Songs in 2019") + scale_fill_manual(values = alpha(c("#1A8693",
    "#4D7A5C", "#D4B95E", "#D28F33", "#B34233", "#751A33"), 0.7)) +
    scale_color_manual(values = c("#1A8693", "#4D7A5C", "#D4B95E",
        "#D28F33", "#B34233", "#751A33")) + theme(plot.title = element_text(size = 16,
    face = "bold", hjust = 0.5), legend.position = "none", axis.text = element_text(face = "bold",
    size = 11)) + scale_x_discrete(labels = c("No Guidance",
    "Senorita", "Wow.", "Old Town Road", "Bad Guy", "Sunflower (Spider-Man:\n Into The Spider-Verse"))
```



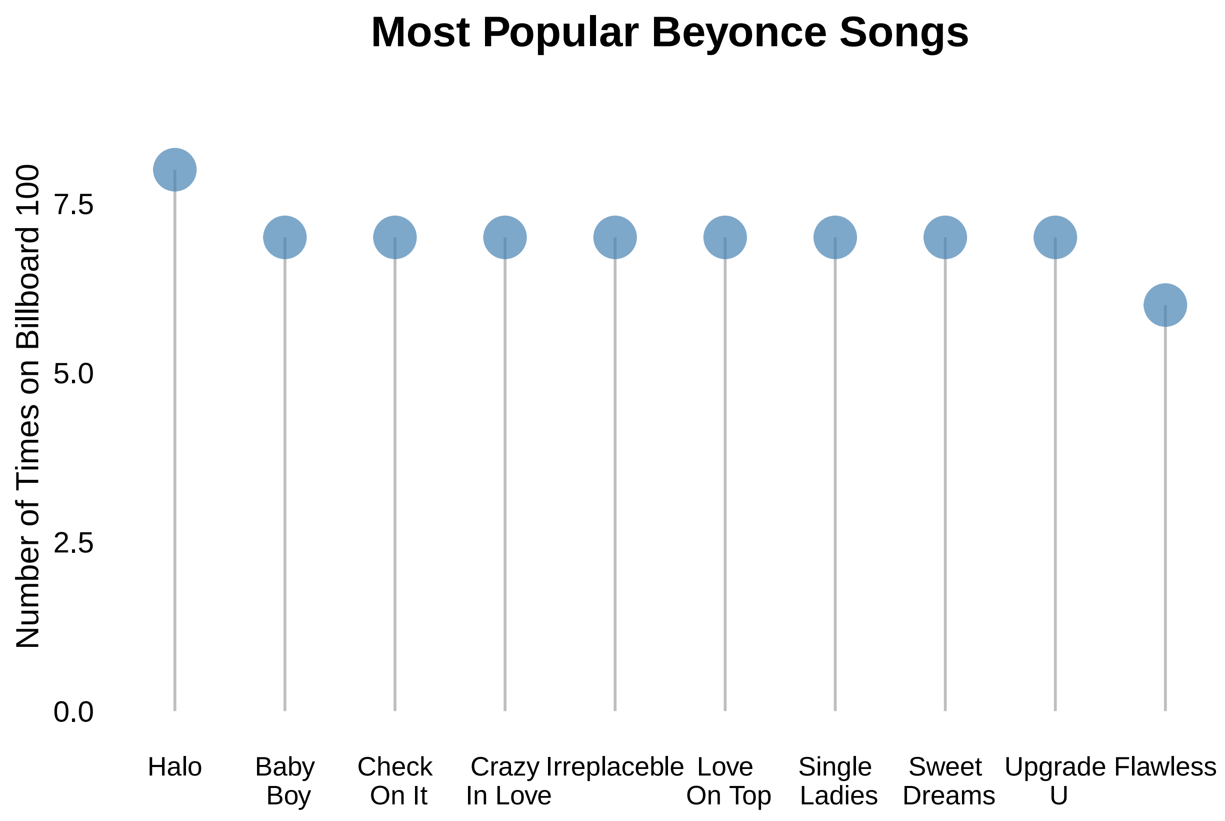
This shows the songs that repeatedly ranked in the top 10 (most popular) on the Billboard 100 in 2019. For example, Old Town Road ranked in the top 10 for 26 weeks in 2019.

Finally, I'm a Beyonce fan so I want to explore her top 10 most popular songs on Billboard 100.

```r
beyonce10 = clean.charts %>%
    select(primary.artist, song, month) %>%
    filter(primary.artist == "Beyonce") %>%
    distinct() %>%
    count(song) %>%
    arrange(desc(n)) %>%
    head(10)
beyonce10
```

```
##                                  song n
## 1                                Halo 8
## 2                            Baby Boy 7
## 3                        Check On It 7
## 4                      Crazy In Love 7
## 5                      Irreplaceable 7
## 6                        Love On Top 7
## 7  Single Ladies (Put A Ring On It) 7
## 8                       Sweet Dreams 7
## 9                          Upgrade U 7
## 10                        ***Flawless 6
```

```
ggplot(beyonce10, aes(reorder(song, -n), n)) + geom_segment(aes(x = reorder(song,
    -n), xend = reorder(song, n), y = 0, yend = n), color = "grey") +
    geom_point(size = 5, color = "steelblue", fill = alpha("steelblue",
        0.3), alpha = 0.7, shape = 21, stroke = 2) + theme_cowplot(12) +
    labs(x = "", y = "Number of Times on Billboard 100", title = "Most Popular Beyonce Songs") +
    scale_x_discrete(labels = c("Halo", "Baby\n Boy", "Check\n On It",
        "Crazy\n In Love", "Irreplaceble", "Love\n On Top", "Single\n Ladies",
        "Sweet\n Dreams", "Upgrade\n U", "Flawless")) + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), panel.background = element_blank(),
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
    axis.ticks = element_blank(), axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 11), axis.line = element_blank()) +
    ylim(0, 9)
```


Most Popular Beyonce Songs
Number of Times on Billboard 100
Halo
Baby Boy
Check On It
Crazy In Love
Irreplaceble
Love On Top
Single Ladies
Sweet Dreams
Upgrade U
Flawless