# Billboard Top 100

**Deeksha R Koonadi, Sonali Hornick**

**2024-08-10**

# R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(dplyr)
```

```
##
## 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(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyr' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.1
## ✔ readr     2.1.5
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(tidyr)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(dbscan)
```

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

```
##
## Attaching package: 'dbscan'
##
## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

# Wrangling the Billboard Top 100

Consider the data in billboard.csv containing every song to appear on the weekly Billboard Top 100 chart since 1958, up through the middle of 2021. Each row of this data corresponds to a single song in a single week. For our purposes, the relevant columns here are:

- *performer*: who performed the song
- *song*: the title of the song
- *year*: year (1958 to 2021)
- *week*: chart week of that year (1, 2, etc)
- *week_position*: what position that song occupied that week on the Billboard top 100 chart **Use your skills in data wrangling and plotting to answer the following three questions**

**Part A) Make a table of the top 10 most popular songs since 1958, as measured by the total number of weeks that a song spent on the Billboard Top 100. Note that these data end in week 22 of 2021, so the most popular songs of 2021 will not have up-to-the-minute data; please send our apologies to The Weeknd.**

Your table should have **10 rows and 3 columns**: *performer*, *song*, and *count*, where ***count*** represents the number of weeks that song appeared in the Billboard Top 100. Make sure the entries are sorted in **descending order of the count variable**, so that the more popular songs appear at the top of the table. Give your table a short caption describing what is shown in the table.

*(Note: you'll want to use both performer and song in any group_by operations, to account for the fact that multiple unique songs can share the same title.)*

```
billboard = read.csv("billboard.csv")

relevantColumns = c("performer", "song", "year", "week", "week_position")
billboardRelevant = billboard[relevantColumns]
billboardRelevant
```

| performer<br><chr> | song<br><chr> |
|---|---|
| Patty Duke | Don't Just Stand There |
| Patty Duke | Don't Just Stand There |
| Patty Duke | Don't Just Stand There |
| Patty Duke | Don't Just Stand There |
| Patty Duke | Don't Just Stand There |
| Patty Duke | Don't Just Stand There |
| Patty Duke | Don't Just Stand There |
| Patty Duke | Don't Just Stand There |
| Teddy Pendergrass | Don't Keep Wasting My Time |
| Teddy Pendergrass | Don't Keep Wasting My Time |

1-10 of 10,000 rows | 1-4 of 5 columns

Previous **1** 2 3 4 5 6 … 1000 Next

```
billboardTop <- billboardRelevant %>%
  filter(week_position <= 100, year >= 1958) %>%
  group_by(performer, song) %>%
  summarise(count = n(), .groups = 'drop') %>%
  arrange(desc(count))


tenthValue = unique(billboardTop$count)[10]

billboardTop10 = billboardTop %>%
  filter(count >= tenthValue)

print(billboardTop10)
```

```
## # A tibble: 17 × 3
##    performer                                song                           count
##    <chr>                                    <chr>                          <int>
##  1 Imagine Dragons                          Radioactive                       87
##  2 AWOLNATION                               Sail                              79
##  3 Jason Mraz                               I'm Yours                         76
##  4 The Weeknd                               Blinding Lights                   76
##  5 LeAnn Rimes                              How Do I Live                     69
##  6 LMFAO Featuring Lauren Bennett & GoonRock Party Rock Anthem                68
##  7 OneRepublic                              Counting Stars                    68
##  8 Adele                                    Rolling In The Deep               65
##  9 Jewel                                    Foolish Games/You Were Meant…     65
## 10 Carrie Underwood                         Before He Cheats                  64
## 11 Gabby Barrett Featuring Charlie Puth     I Hope                            62
## 12 Lifehouse                                You And Me                        62
## 13 The Lumineers                            Ho Hey                            62
## 14 Imagine Dragons                          Demons                            61
## 15 Post Malone                              Circles                           61
## 16 Lady Antebellum                          Need You Now                      60
## 17 Los Del Rio                              Macarena (Bayside Boys Mix)       60
```

***Table Caption:*** The greatest count in the table is 87 and the lowest count is 60. Other than Imagine Dragons, no other performer really repeats in the top 10 table.

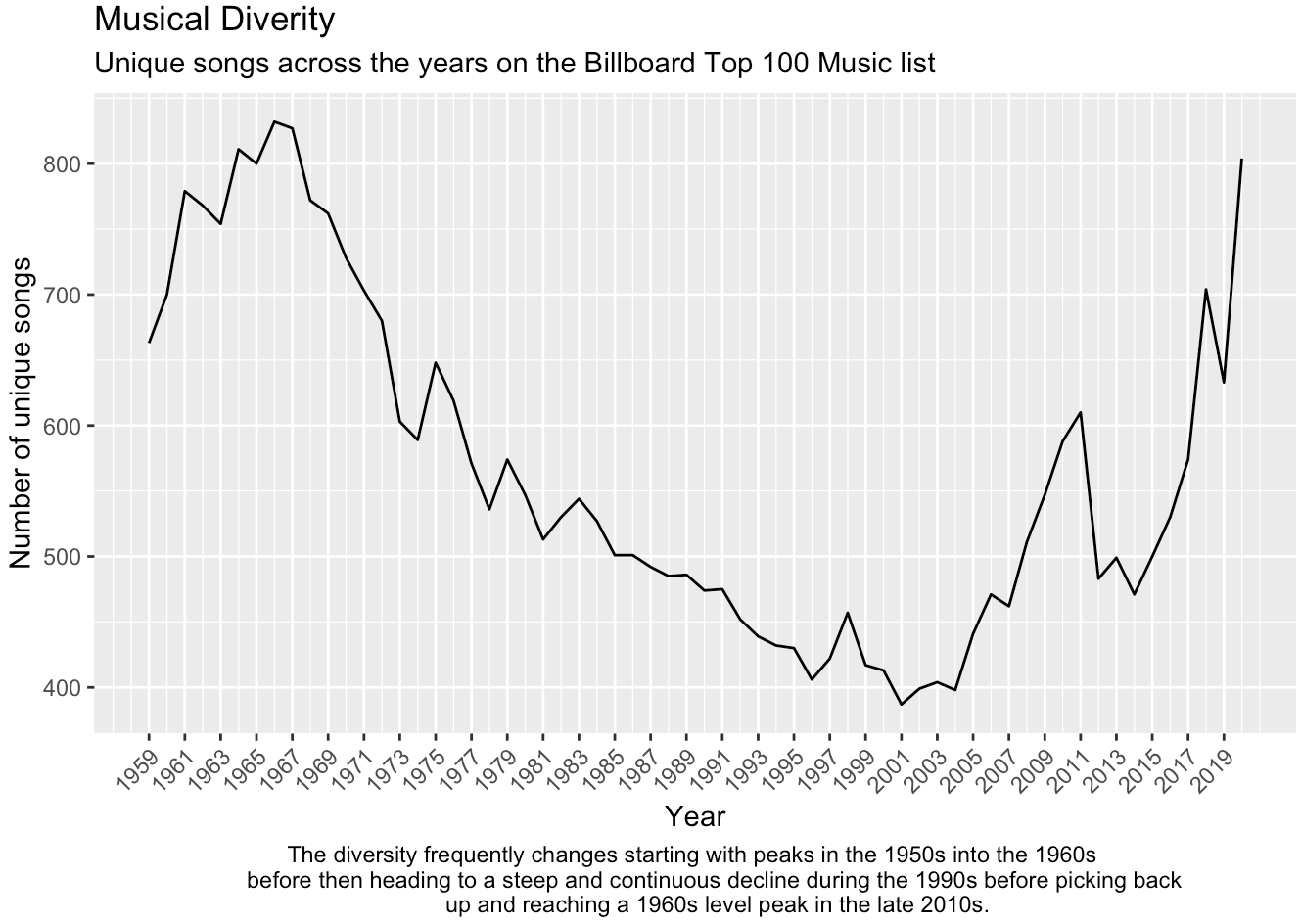
**Part B) Is the "musical diversity" of the Billboard Top 100 changing over time? Let's find out. We'll measure the musical diversity of given year as the number of unique songs that appeared in the Billboard Top 100 that year. Make a line graph that plots this measure of musical diversity over the years. The x axis should show the year, while the y axis should show the number of unique songs appearing at any position on the Billboard Top 100 chart in any week that year. For this part, please filter the data set so that it excludes the years 1958 and 2021, since we do not have complete data on either of those years. Give the figure an informative caption in which you explain what is shown in the figure and comment on any interesting trends you see.**

There are number of ways to accomplish the data wrangling here. For example, you could use two distinct sets of data-wrangling steps. The first set of steps would get you a table that counts the number of times that a given song appears on the Top 100 in a given year. The second set of steps operate on the result of the first set of steps; it would count the number of unique songs that appeared on the Top 100 in each year, irrespective of how many times it had appeared.

```
any_time_per_year = billboardRelevant %>%
  filter(week_position <= 100)

any_time_per_year = any_time_per_year %>%
  group_by(year) %>%
  summarise(countOfSongs = n_distinct(performer, song))

ggplot(data = any_time_per_year[any_time_per_year$year > 1958
                                                & any_time_per_year$year < 2021, ]) +
  geom_line(aes(x = year, y = countOfSongs)) +
  scale_x_continuous(breaks = seq(1959, 2020, 2), guide = guide_axis(angle = 45)) +
  xlab(label = "Year") +
  ylab(label = "Number of unique songs") +
  labs(title = "Musical Diverity",
       subtitle = "Unique songs across the years on the Billboard Top 100 Music list",
       caption = "The diversity frequently changes starting with peaks in the 1950s into
the 1960s
       before then heading to a steep and continuous decline during the 1990s before pic
king back
       up and reaching a 1960s level peak in the late 2010s.") +
  theme(plot.caption = element_text(hjust = 0.5))
```




***Comments:*** The first year we are looking at is 1958 at which point the graph reads a little over 650 unique songs. From there the graph sees a pretty steady overall incline until 1965-1967 where it reaches a very high peak around a little under 850 unique songs. From there the graph goes into a sharp and steady steep decline, hitting

its lowest at 2001 with less than 400 unique songs. After this year there is a turn into a increase (with a few decreases along the way), continuing up to the last year on the graph which is 2019. This also happens to be another peak year with around 800 unique songs.

**Part C) Let's define a "ten-week hit" as a single song that appeared on the Billboard Top 100 for at least ten weeks. There are 19 artists in U.S. musical history since 1958 who have had at least 30 songs that were "ten-week hits." Make a bar plot for these 19 artists, showing how many ten-week hits each one had in their musical career. Give the plot an informative caption in which you explain what is shown.**

```
tenWeekHit = billboardRelevant %>%
  filter(week_position <= 100) %>%
  group_by(performer, song) %>%
  summarise(songsWeeklyCount = n()) %>%
  filter(songsWeeklyCount >= 10) %>%
  group_by(performer) %>%
  summarise(songsCount = n()) %>%
  filter(songsCount >= 30)
```

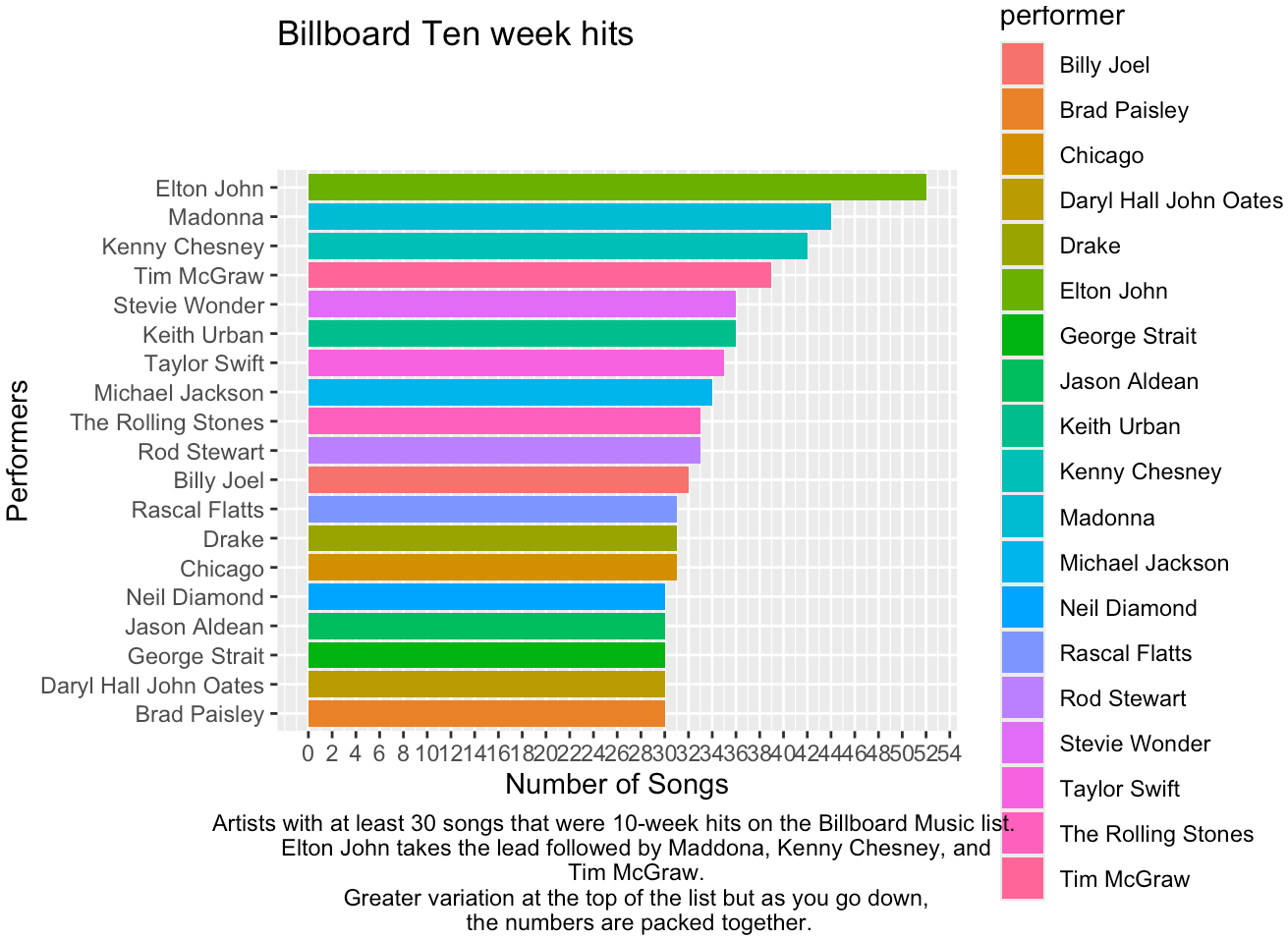
```
## `summarise()` has grouped output by 'performer'. You can override using the
## `.groups` argument.
```

```
tenWeekHit
```

| performer<br><chr> | songsCount<br><int> |
|---|---|
| Billy Joel | 32 |
| Brad Paisley | 30 |
| Chicago | 31 |
| Daryl Hall John Oates | 30 |
| Drake | 31 |
| Elton John | 52 |
| George Strait | 30 |
| Jason Aldean | 30 |
| Keith Urban | 36 |
| Kenny Chesney | 42 |

1-10 of 19 rows Previous **1** 2 Next

```
ggplot(data = tenWeekHit) +
  geom_col(aes(x = reorder(performer, songsCount), y = songsCount, fill=performer)) +
  coord_flip() +
  scale_y_continuous(breaks = seq(0, 60, 2)) +
  xlab(label = "Performers") +
  ylab(label = "Number of Songs") +
  labs(title = "Billboard Ten week hits",
       subtitle = " \n\n", # adds some space below the title so the entire legend can fi
t
       caption = "Artists with at least 30 songs that were 10-week hits on the Billboard
Music list.
       Elton John takes the lead followed by Maddona, Kenny Chesney, and
       Tim McGraw.
       Greater variation at the top of the list but as you go down,
       the numbers are packed together.") +
  theme(plot.caption = element_text(hjust = 0.5))
```




***Comments:*** *See Graph Caption* There is greater variation at the top of the list but as you move down these is more standarization in that artists start to have the same number of songs that meet this criteria.