Movie Ratings Sample Data

John Cruz

2023-02-02

Introduction

I will be collecting movie ratings through an online survey on a scale of 1 to 5.

The steps through this project will be:

1. Collect survey data
2. Create a database to store this information
3. Ability to query the data into R
4. Analyze some descriptive measures.

Required Libraries

```
library('RPostgres')
library('DBI')
library('tidyverse')
library('ggrepel')
```

Collecting Survey Data

The survey will be conducted using a Google Forms named *Spoiled Corn*. This survey will collect ratings of six recent popular movies on a scale of 1 to 5, 1 being the lowest. If a person did not watch this movie, it will be categorized with an option of N/A. Once data is collected, I will export the data via CSV, and import into a PostgreSQL database.

Online Survey

Building the Database

The database used was PostgreSQL for no particular reason but to learn how to use another database option outside of my previous ones such as SQL Server and Google Cloud Storage.

1. Using PostgreSQL within pgAdmin 4, create a database named *movie_survey*.
2. Within the *movie_survey* database, run the following code: create_tables.sql

- The SQL code is created was based off this ER Diagram.
- I created a table named *surveys* to store the data in its original format. After this initial table is made, I used the following code allowing the CSV file to be imported to the *surveys* table.

```
copy surveys from 'CSV survey filepath' delimiter ',' HEADER csv;
```

- To create the *films* table, I used the column names from the *surveys* table to name each row and also grab its release year.

```
SELECT
    SPLIT_PART(column_name, '(', 1) AS movie_name,
    SPLIT_PART(SPLIT_PART(column_name, '(', 2), ')', 1) AS release_year
FROM information_schema.columns
WHERE table_schema = 'public'
    AND table_name   = 'surveys'
```

- The *ratings* table was created in the same format as the *films* table, however, because I wanted to unpivot the columns of the *surveys* table, similar to using the *melt()* function in Pandas I was able to use two functions in PostgreSQL, *LATERAL()* and *VALUES()*, to perform the same idea.

```
SELECT s."Timestamp", s.user_id, val.*
FROM surveys s,
LATERAL(
    VALUES
        (1, s."Avengers: Endgame (2019)"),
        (2, s."The Lion King (2019)"),
        (3, s."Joker (2019)"),
        (4, s."Everything Everywhere All at Once (2022)"),
        (5, s."The Matrix Resurrections (2021)"),
        (6, s."King Richard (2021)")
    ) AS val (movie_id, full_rating)
```

- I queried the database to see if it works as intended.

Connecting to the Database

PostgreSQL Database

Using this connection, I will be able to keep the username and password from being embedded into the code and instead have RStudio pop up a dialog box to input said information.

```
cnxn <- dbConnect(RPostgres::Postgres(),
                      dbname = 'movie_survey',
                      port = 5432,
                      user = rstudioapi::askForPassword("Database Username"),
                      password = rstudioapi::askForPassword("Database Password"))
```

Query the Database

Here I am able to directly query the PostgreSQL database and pull the information from the tables I created earlier. *dbFetch()* will allow me to move the results of the query into a readable DataFrame.

```
query <- dbSendQuery(cnxn,
                     "SELECT
                         r.datetimeid,
                         r.user_id,
                         f.name,
                         r.rating,
                         f.release_year
                      FROM ratings r
                      INNER JOIN films f
                         ON r.movie_id = f.movie_id ")

df <- dbFetch(query)
```

Close Connection

Since I have gathered the data needed from the database, I will close the connection to PostgreSQL.

```
dbClearResult(query)
dbDisconnect(cnxn)
```

Analyzing the Results

Descriptive Statistics

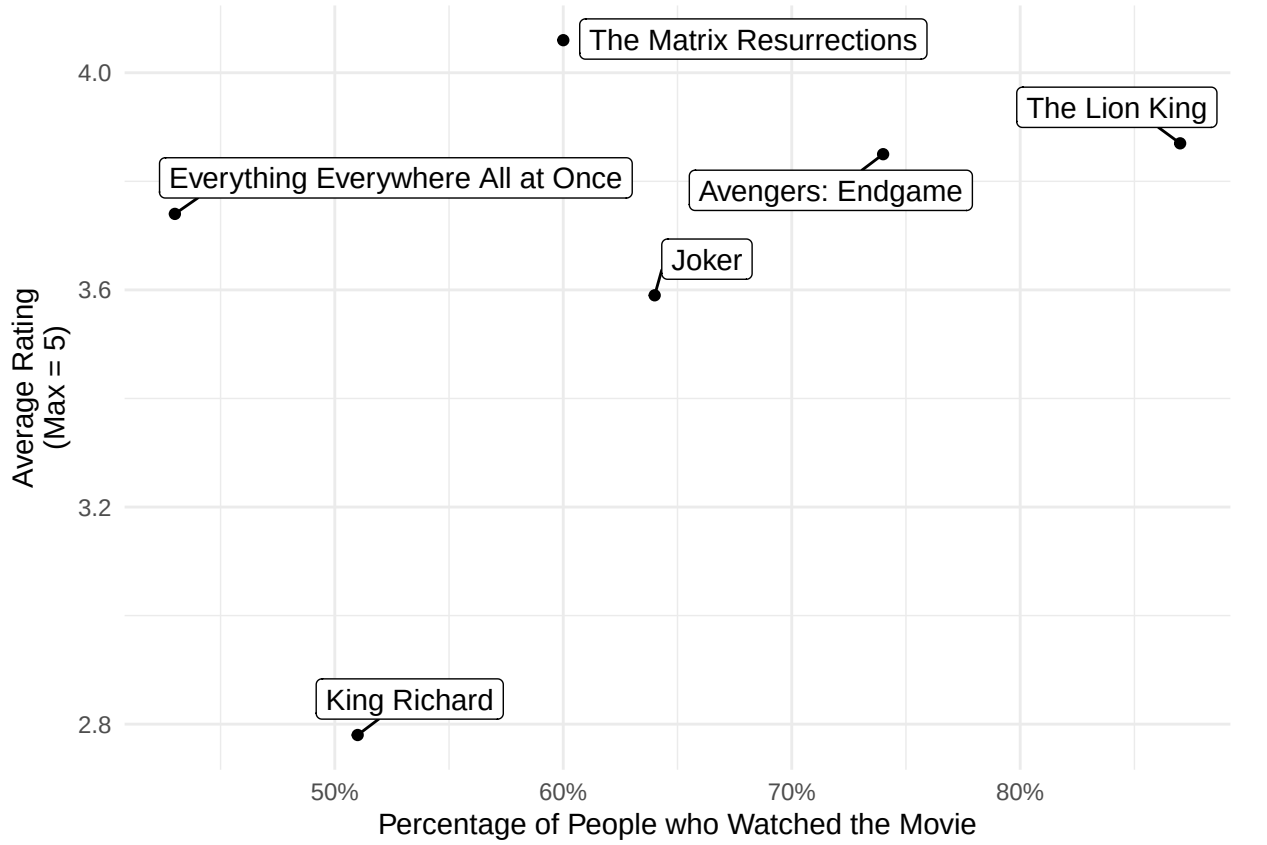
The total number of unique respondents in the survey was 53 people.

```
df$name <- trimws(df$name)

avg_rating <- df %>% filter(!is.na(rating)) %>%
  group_by(release_year, name) %>%
  summarize(avg_rating = round(mean(rating), 2), count = n(), .groups = 'drop') %>%
  select(release_year, name, avg_rating, count) %>%
  mutate(pct_viewed = round(count / num_surveys, 2)) %>%
  arrange(desc(avg_rating))

knitr::kable(avg_rating)
```

release_year	name	avg_rating	count	pct_viewed
2021	The Matrix Resurrections	4.06	32	0.60
2019	The Lion King	3.87	46	0.87
2019	Avengers: Endgame	3.85	39	0.74
2022	Everything Everywhere All at Once	3.74	23	0.43
2019	Joker	3.59	34	0.64
2021	King Richard	2.78	27	0.51

We can see in the previous table the highest rated movie was *The Matrix Resurrections* released in 2021 with an average rating of **4.06**. However, only **60 percent** of the respondents watched this movie. In the graph below, we can see the 2019 release of *The Lion King* had an average viewer rating of **3.87** while **87 percent** of the respondents had watched the movie.

```
ggplot(data = avg_rating, aes(x = pct_viewed, y = avg_rating, label = name)) +
  geom_point() +
  geom_label_repel(box.padding = 0.35) +
  theme_minimal() +
  labs(x = 'Percentage of People who Watched the Movie', y = 'Average Rating\n(Max = 5)') +
  scale_x_continuous(labels = scales::percent)
```



Future Insights

With this data, I would like to continue to investigate additional factors into how people rated their movies. Are there particular genres that ties them together such as action, adventure, or drama? Did their box office sales help determine the rating that was given? Additionally, collecting more information about the users themselves in the survey would help. Does their race, income level, education, and hobbies influence their choices?