MovieMetaData_CaseStudy

Sopan

2025-06-06

```r
# Load required packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
##
## The following object is masked from 'package:purrr':
##
##     flatten
```

```r
# Read the CSV file
movies_metadata <- read_csv("MovieData/movies_metadata.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 45466 Columns: 24
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (14): belongs_to_collection, genres, homepage, imdb_id, original_langua...
## dbl   (7): budget, id, popularity, revenue, runtime, vote_average, vote_count
## lgl   (2): adult, video
## date  (1): release_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Check the structure of your data
glimpse(movies_metadata)
```

```
## Rows: 45,466
## Columns: 24
## $ adult                 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
## $ belongs_to_collection <chr> "{'id': 10194, 'name': 'Toy Story Collection', '~
## $ budget                <dbl> 30000000, 65000000, 0, 16000000, 0, 60000000, 58~
## $ genres                <chr> "[{'id': 16, 'name': 'Animation'}, {'id': 35, 'n~
## $ homepage              <chr> "http://toystory.disney.com/toy-story", NA, NA, ~
## $ id                    <dbl> 862, 8844, 15602, 31357, 11862, 949, 11860, 4532~
## $ imdb_id               <chr> "tt0114709", "tt0113497", "tt0113228", "tt011488~
## $ original_language     <chr> "en", "en", "en", "en", "en", "en", "en", "en", ~
## $ original_title        <chr> "Toy Story", "Jumanji", "Grumpier Old Men", "Wai~
## $ overview              <chr> "Led by Woody, Andy's toys live happily in his r~
## $ popularity            <dbl> 21.946943, 17.015539, 11.712900, 3.859495, 8.387~
## $ poster_path           <chr> "/rhIRbceoE9lR4veEXuwCC2wARtG.jpg", "/vzmL6fP7aP~
## $ production_companies  <chr> "[{'name': 'Pixar Animation Studios', 'id': 3}]"~
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'United States of~
## $ release_date          <date> 1995-10-30, 1995-12-15, 1995-12-22, 1995-12-22,~
## $ revenue               <dbl> 373554033, 262797249, 0, 81452156, 76578911, 187~
## $ runtime               <dbl> 81, 104, 101, 127, 106, 170, 127, 97, 106, 130, ~
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'English'}]", "[{'~
## $ status                <chr> "Released", "Released", "Released", "Released", ~
## $ tagline               <chr> NA, "Roll the dice and unleash the excitement!",~
## $ title                 <chr> "Toy Story", "Jumanji", "Grumpier Old Men", "Wai~
## $ video                 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
## $ vote_average          <dbl> 7.7, 6.9, 6.5, 6.1, 5.7, 7.7, 6.2, 5.4, 5.5, 6.6~
## $ vote_count            <dbl> 5415, 2413, 92, 34, 173, 1886, 141, 45, 174, 119~
```

```r
# Clean and expand the genres column
movies_expanded <- movies_metadata %>%
  # Convert JSON string to proper format (replace single quotes with double quotes)
  mutate(genres_clean = str_replace_all(genres, "'", '"')) %>%
  # Parse JSON strings into list columns
  mutate(genres_list = map(genres_clean, ~ fromJSON(.x)$name)) %>%
  # Unnest the list column to create multiple rows
  unnest(genres_list, keep_empty = TRUE) %>%
  # Rename the new column
  rename(genre_name = genres_list) %>%
  # Remove intermediate columns if needed
  select(-genres_clean)

# View the result
head(movies_expanded)
```

```
## # A tibble: 6 x 25
##   adult belongs_to_collection              budget genres homepage    id imdb_id
##   <lgl> <chr>                               <dbl> <chr>  <chr>    <dbl> <chr>
## 1 FALSE {'id': 10194, 'name': 'Toy Story C~  3   e7 [{'id~ http://~   862 tt0114~
## 2 FALSE {'id': 10194, 'name': 'Toy Story C~  3   e7 [{'id~ http://~   862 tt0114~
## 3 FALSE {'id': 10194, 'name': 'Toy Story C~  3   e7 [{'id~ http://~   862 tt0114~
## 4 FALSE <NA>                                6.5e7 [{'id~ <NA>      8844 tt0113~
## 5 FALSE <NA>                                6.5e7 [{'id~ <NA>      8844 tt0113~
## 6 FALSE <NA>                                6.5e7 [{'id~ <NA>      8844 tt0113~
## # i 18 more variables: original_language <chr>, original_title <chr>,
## #   overview <chr>, popularity <dbl>, poster_path <chr>,
## #   production_companies <chr>, production_countries <chr>,
## #   release_date <date>, revenue <dbl>, runtime <dbl>, spoken_languages <chr>,
## #   status <chr>, tagline <chr>, title <chr>, video <lgl>, vote_average <dbl>,
## #   vote_count <dbl>, genre_name <chr>
```

```r
# Save the expanded data if needed
write_csv(movies_expanded, "MovieData/movies_metadata_expanded.csv")
```

Clean the Data and prepare for Analysis

```r
library(tidyverse)
library(lubridate) # For timestamp conversion if needed
```

Step 1: Load and process ratings data

```r
ratings <- read_csv("MovieData/ratings_small.csv") %>%
  # Calculate average rating per movie
  group_by(movieId) %>%
  summarize(
    avg_rating = mean(rating, na.rm = TRUE),
    rating_count = n()
  ) %>%
  ungroup()
```

```
## Rows: 100004 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): userId, movieId, rating, timestamp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Step 2: Load and process movie metadata

```r
movies <- read_csv("MovieData/movies_metadata_expanded.csv") %>%
  # Convert id to numeric to match with ratings (some cleaning might be needed)
  mutate(
    movieId = as.numeric(id)  # Assuming 'id' in metadata matches 'movieId' in ratings
  ) %>%
  # Handle any rows that didn't convert properly
  filter(!is.na(movieId))
```

```
## Rows: 93548 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (15): belongs_to_collection, genres, homepage, imdb_id, original_langua...
## dbl   (7): budget, id, popularity, revenue, runtime, vote_average, vote_count
## lgl   (2): adult, video
## date  (1): release_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Step 3: Merge the datasets

```
movies_with_ratings <- movies %>%
  left_join(ratings, by = "movieId") %>%
  # You might want to arrange by rating or other criteria
  arrange(desc(avg_rating))
```

Step 4: Optional - Add rating categories

```
movies_with_ratings <- movies_with_ratings %>%
  mutate(
    rating_category = case_when(
      avg_rating >= 4.5 ~ "Excellent",
      avg_rating >= 4.0 ~ "Good",
      avg_rating >= 3.0 ~ "Average",
      avg_rating >= 2.0 ~ "Below Average",
      avg_rating >= 0 ~ "Poor",
      TRUE ~ "No Ratings"
    )
  )
```

Step 5: View and save results

```
glimpse(movies_with_ratings)
```

```
## Rows: 93,536
## Columns: 29
## $ adult                 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
## $ belongs_to_collection <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ budget                <dbl> 16500000, 16500000, 8169363, 8169363, 8169363, 4~
## $ genres                <chr> "[{'id': 18, 'name': 'Drama'}, {'id': 10749, 'na~
## $ homepage              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ id                    <dbl> 4584, 4584, 2086, 2086, 2086, 2636, 2636, 872, 8~
## $ imdb_id               <chr> "tt0114388", "tt0114388", "tt0113972", "tt011397~
## $ original_language     <chr> "en", "en", "en", "en", "en", "en", "en", "en", ~
## $ original_title        <chr> "Sense and Sensibility", "Sense and Sensibility"~
## $ overview              <chr> "Rich Mr. Dashwood dies, leaving his second wife~
## $ popularity            <dbl> 10.673167, 10.673167, 6.848591, 6.848591, 6.8485~
## $ poster_path           <chr> "/lA9HTy84Bb6ZwNeyoZKobcMdpMc.jpg", "/lA9HTy84Bb~
## $ production_companies  <chr> "[{'name': 'Columbia Pictures Corporation', 'id'~
## $ production_countries  <chr> "[{'iso_3166_1': 'GB', 'name': 'United Kingdom'}~
## $ release_date          <date> 1995-12-13, 1995-12-13, 1995-11-22, 1995-11-22,~
## $ revenue               <dbl> 135000000, 135000000, 8175346, 8175346, 8175346,~
## $ runtime               <dbl> 136, 136, 90, 90, 90, 110, 110, 103, 103, 103, 1~
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'English'}]", "[{'~
## $ status                <chr> "Released", "Released", "Released", "Released", ~
## $ tagline               <chr> "Lose your heart and come to your senses.", "Los~
## $ title                 <chr> "Sense and Sensibility", "Sense and Sensibility"~
## $ video                 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
## $ vote_average          <dbl> 7.2, 7.2, 6.1, 6.1, 6.1, 5.5, 5.5, 7.9, 7.9, 7.9~
## $ vote_count            <dbl> 364, 364, 190, 190, 190, 317, 317, 747, 747, 747~
## $ genre_name            <chr> "Drama", "Romance", "Crime", "Drama", "Thriller"~
## $ movieId               <dbl> 4584, 4584, 2086, 2086, 2086, 2636, 2636, 872, 8~
## $ avg_rating           <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ rating_count         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, ~
## $ rating_category      <chr> "Excellent", "Excellent", "Excellent", "Excellen~
```

Save the merged data

```
write_csv(movies_with_ratings, "MovieData/movies_with_ratings.csv")
```

Top rated movies

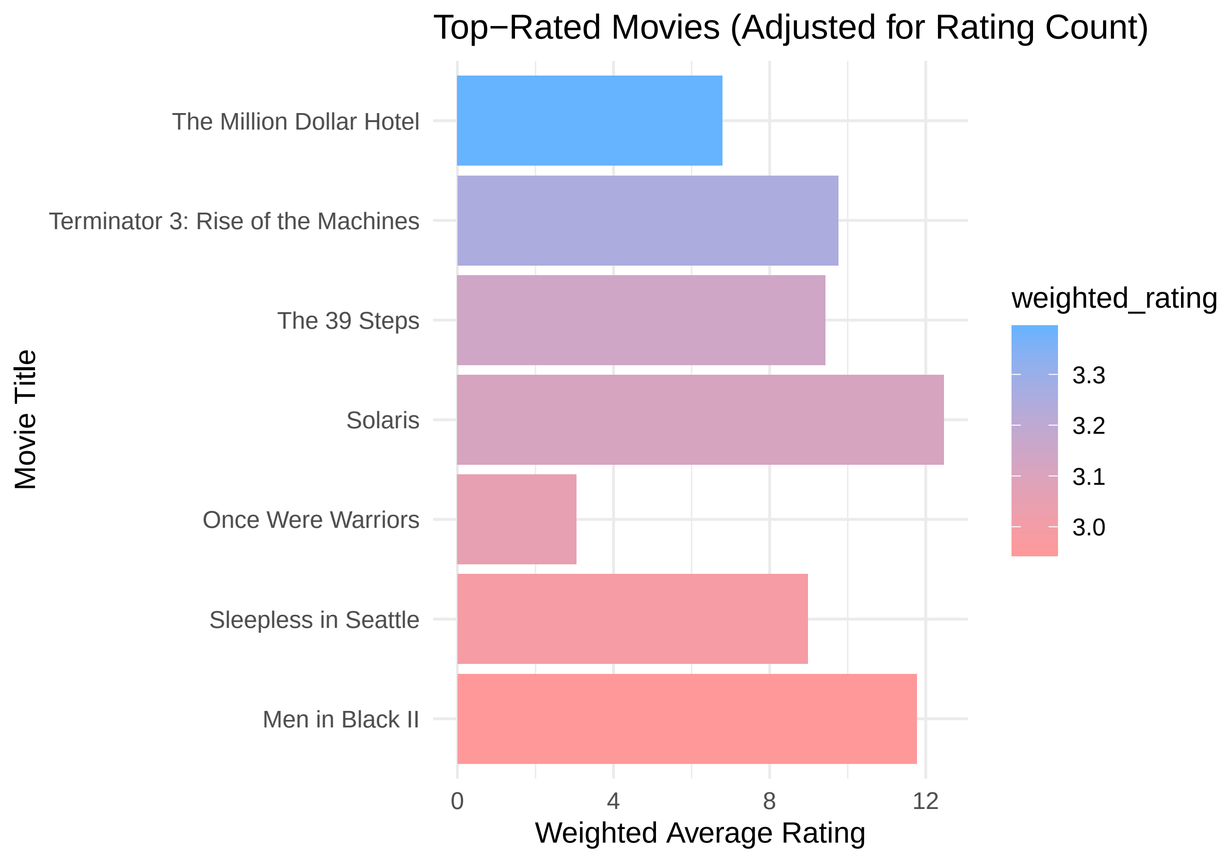
```
top_movies <- movies_with_ratings %>%
  filter(rating_count > 50) %>%  # Only movies with sufficient ratings
  arrange(desc(avg_rating)) %>%
  select(title, avg_rating, rating_count, genres)

head(top_movies, 10)
```

```
## # A tibble: 10 x 4
##    title                    avg_rating rating_count genres
##    <chr>                         <dbl>        <int> <chr>
##  1 Sleepless in Seattle           4.49          200 [{'id': 35, 'name': 'Comedy~
##  2 Sleepless in Seattle           4.49          200 [{'id': 35, 'name': 'Comedy~
##  3 Sleepless in Seattle           4.49          200 [{'id': 35, 'name': 'Comedy~
##  4 The Million Dollar Hotel       4.49          311 [{'id': 18, 'name': 'Drama'~
##  5 The Million Dollar Hotel       4.49          311 [{'id': 18, 'name': 'Drama'~
##  6 The Thomas Crown Affair        4.39           62 [{'id': 18, 'name': 'Drama'~
##  7 The Thomas Crown Affair        4.39           62 [{'id': 18, 'name': 'Drama'~
##  8 The Thomas Crown Affair        4.39           62 [{'id': 18, 'name': 'Drama'~
##  9 Lonely Hearts                  4.34           76 [{'id': 18, 'name': 'Drama'~
## 10 Lonely Hearts                  4.34           76 [{'id': 18, 'name': 'Drama'~
```

Visualizations: Movie Recommendation

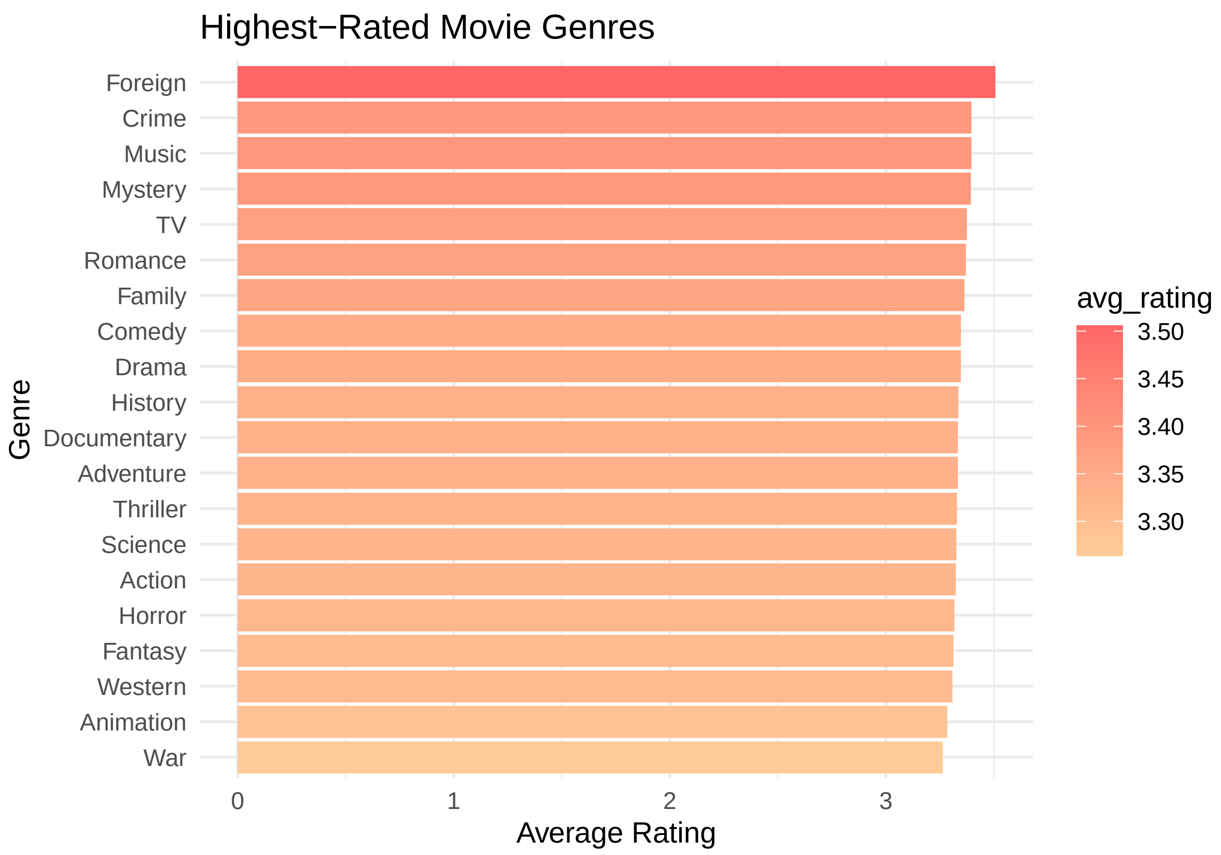
Top-Rated Movies (Weighted by Number of Ratings)

```
movies_with_ratings <- movies_with_ratings %>%
  mutate(weighted_rating = (avg_rating * rating_count) / (rating_count + 100))  # Bayesian average

top_movies <- movies_with_ratings %>%
  filter(rating_count > 30) %>%  # Minimum 30 ratings
  arrange(desc(weighted_rating)) %>%
  head(20)

ggplot(top_movies, aes(x = reorder(title, weighted_rating), y = weighted_rating, fill = weighted_rating
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Top-Rated Movies (Adjusted for Rating Count)",
       x = "Movie Title",
       y = "Weighted Average Rating") +
  scale_fill_gradient(low = "#ff9999", high = "#66b3ff") +
  theme_minimal()
```



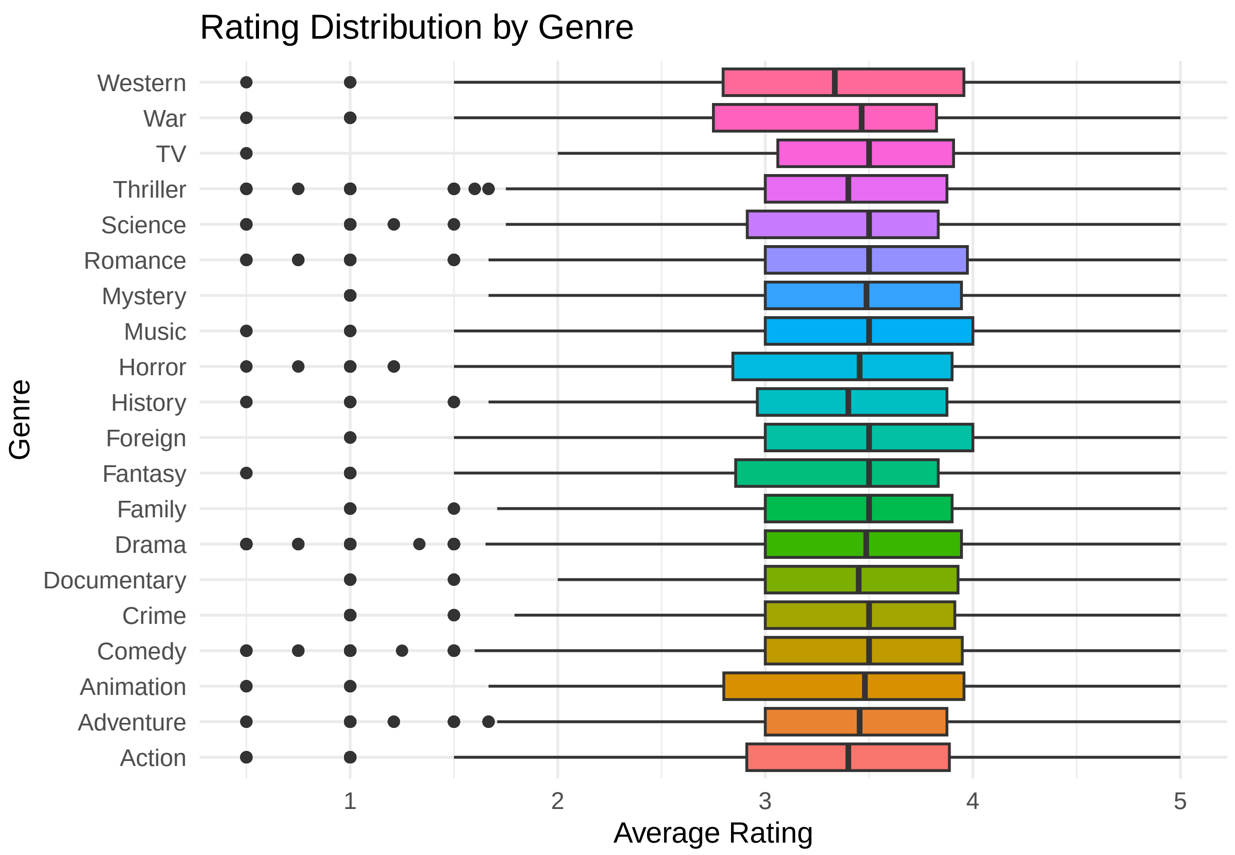
## Most Popular Genres by Average Rating

```
genre_ratings <- movies_with_ratings %>%
  separate_rows(genres, sep = "\\}, \\{") %>%  # Split genres
  mutate(genre = str_extract(genres, "'name': '[A-Za-z]+")) %>%
  mutate(genre = gsub("'name': '", "", genre)) %>%
  filter(!is.na(genre)) %>%
  group_by(genre) %>%
  summarise(
    avg_rating = mean(avg_rating, na.rm = TRUE),
    movie_count = n()
  ) %>%
  filter(movie_count > 50)  # Only genres with sufficient data

ggplot(genre_ratings, aes(x = reorder(genre, avg_rating), y = avg_rating, fill = avg_rating)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Highest-Rated Movie Genres",
       x = "Genre",
       y = "Average Rating") +
  scale_fill_gradient(low = "#ffcc99", high = "#ff6666") +
  theme_minimal()
```



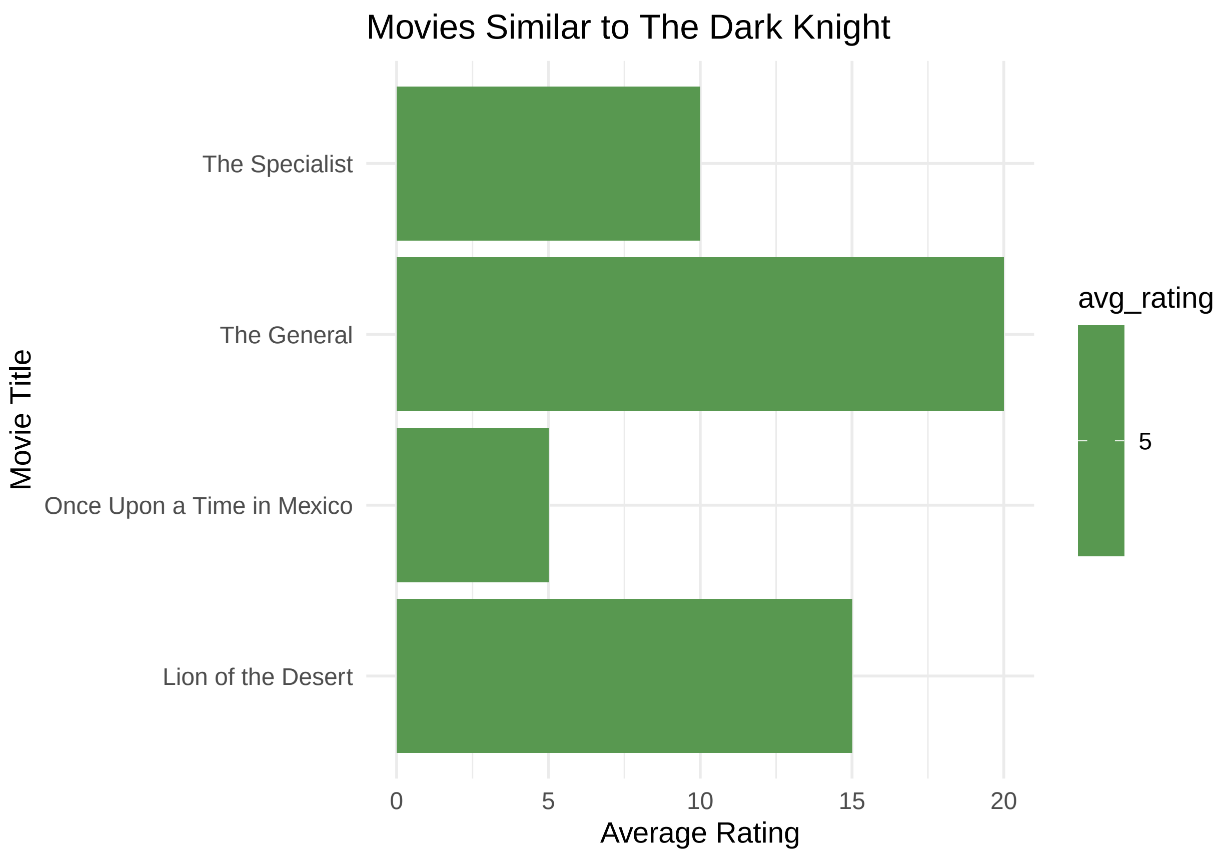
## Rating Distribution by Genre

```
movies_with_genres <- movies_with_ratings %>%
  separate_rows(genres, sep = "\\}, \\{") %>%
  mutate(genre = str_extract(genres, "'name': '[A-Za-z]+")) %>%
  mutate(genre = gsub("'name': '", "", genre)) %>%
  filter(!is.na(genre) & !is.na(avg_rating))

ggplot(movies_with_genres, aes(x = genre, y = avg_rating, fill = genre)) +
  geom_boxplot() +
  coord_flip() +
  labs(title = "Rating Distribution by Genre",
       x = "Genre",
       y = "Average Rating") +
  theme_minimal() +
  theme(legend.position = "none")
```



## Movie Recommendations Based on User Preferences

```
# Example: User likes "The Dark Knight" (assuming movieId = 155)
target_movie <- "The Dark Knight"

# Find similar movies by genre and rating
similar_movies <- movies_with_ratings %>%
  filter(str_detect(genres, "Action") &  # Same genre
         avg_rating >= 4.0 &             # Highly rated
         title != target_movie) %>%      # Exclude the target
  arrange(desc(avg_rating)) %>%
  head(10)

ggplot(similar_movies, aes(x = reorder(title, avg_rating), y = avg_rating, fill = avg_rating)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = paste("Movies Similar to", target_movie),
       x = "Movie Title",
       y = "Average Rating") +
  scale_fill_gradient(low = "#99cc99", high = "#006600") +
  theme_minimal()
```


Movies Similar to The Dark Knight
Movie Title
The Specialist
The General
Once Upon a Time in Mexico
Lion of the Desert
0
5
10
15
20
Average Rating
avg_rating
5