Assignment E

Overview

1. **World Development Bank Data (WDBD)** - economic indicators with years stored as columns
2. **MovieLens Data** - movie information with multiple values in single cells

The analysis follows tidy data principles where:

- Each variable forms a column
- Each observation forms a row
- Each value has its own cell

```
# Load required libraries
library(tidyverse)
library(lubridate)
library(scales)
library(knitr)

# Create output directory for cleaned data
dir.create("clean_data", showWarnings = FALSE, recursive = TRUE)

# Set global options
knitr::opts_chunk$set(
  echo = TRUE,
  message = FALSE,
  warning = FALSE,
  fig.width = 10,
  fig.height = 6
)
```

Part 1: World Development Bank Data

1.1 Identifying Messy Aspects

The WDBD.csv dataset has several issues that violate tidy data principles:

1. **Wide format**: Years (2018-2024) are stored as separate columns instead of as values in a single Year column
2. **Trailing metadata**: The file contains documentation and unit information at the bottom, mixed with the data
3. **Multiple variables per row**: Each row contains multiple year observations for a single indicator
4. **Unit inconsistencies**: Different indicators use different units (%, US$, counts) not explicitly marked in the data
5. **Missing value notation**: Empty strings and ".." both represent missing values

1.2 Importing and Cleaning

```
# Import the dataset
wdbd_raw <- read_csv("../WDBD.csv", show_col_types = FALSE)

# Preview the structure
cat("Raw dataset dimensions:", dim(wdbd_raw), "\n")
```

```
Raw dataset dimensions: 3031 11
```

```
cat("Column names:\n")
```

```
Column names:
```

```
head(names(wdbd_raw), 15)
```

```
 [1] "Country Name"  "Country Code"  "Series Name"   "Series Code"
 [5] "2018 [YR2018]" "2019 [YR2019]" "2020 [YR2020]" "2021 [YR2021]"
 [9] "2022 [YR2022]" "2023 [YR2023]" "2024 [YR2024]"
```

```
# Remove metadata rows at the bottom
# These are rows without valid 3-letter country codes
wdbd_clean <- wdbd_raw %>%
  filter(str_detect(`Country Code`, "^[A-Z]{3}$"))

cat("After removing metadata rows:", dim(wdbd_clean), "\n")
```

```
After removing metadata rows: 2976 11
```

```
cat("Countries:", n_distinct(wdbd_clean$`Country Code`), "\n")
```

```
Countries: 62
```

```
cat("Series (indicators):", n_distinct(wdbd_clean$`Series Code`), "\n")
```

```
Series (indicators): 48
```

1.3 Reshaping to Tidy Format

```
# Identify year columns (they follow pattern: YYYY [YRYYYY])
year_cols <- names(wdbd_clean) %>%
  str_subset("^\\d{4} \\[YR\\d{4}\\]$")

cat("Year columns found:", length(year_cols), "\n")
```

```
Year columns found: 7
```

```
cat("Years:", year_cols[1:7])
```

```
Years: 2018 [YR2018] 2019 [YR2019] 2020 [YR2020] 2021 [YR2021] 2022 [YR2022] 2023 [YR2023]
2024 [YR2024]
```

```
# Pivot longer: gather year columns into a single Year column
wdbd_long <- wdbd_clean %>%
  pivot_longer(
    cols = all_of(year_cols),
    names_to = "Year",
    values_to = "Value"
  ) %>%
  # Extract 4-digit year from column name
  mutate(
    Year = as.integer(str_extract(Year, "\\d{4}")),
    # Convert Value to numeric (handles ".." and empty strings as NA)
    Value = as.numeric(Value)
  )

# Display summary
cat("\nTidy dataset dimensions:", dim(wdbd_long), "\n")
```

```
Tidy dataset dimensions: 20832 6
```

```
print(summary(wdbd_long))
```

```
 Country Name       Country Code       Series Name        Series Code
 Length:20832       Length:20832       Length:20832       Length:20832
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character



      Year          Value
 Min.   :2018   Min.   :-1.325e+10
 1st Qu.:2019   1st Qu.: 8.000e+00
 Median :2021   Median : 5.800e+01
 Mean   :2021   Mean   : 4.973e+09
 3rd Qu.:2023   3rd Qu.: 1.000e+02
 Max.   :2024   Max.   : 2.870e+12
                NA's   :12652
```

```
# Preview the tidy dataset
wdbd_long %>%
  select(`Country Name`, `Series Name`, Year, Value) %>%
  head(10) %>%
  kable(caption = "First 10 rows of tidy WDBD data")
```

First 10 rows of tidy WDBD data

Country Name	Series Name	Year	Value
Afghanistan	Access to clean fuels and technologies for cooking, rural (% of rural population)	2018	14.5
Afghanistan	Access to clean fuels and technologies for cooking, rural (% of rural population)	2019	15.6
Afghanistan	Access to clean fuels and technologies for cooking, rural (% of rural population)	2020	16.4
Afghanistan	Access to clean fuels and technologies for cooking, rural (% of rural population)	2021	17.4
Afghanistan	Access to clean fuels and technologies for cooking, rural (% of rural population)	2022	18.5
Afghanistan	Access to clean fuels and technologies for cooking, rural (% of rural population)	2023	NA
Afghanistan	Access to clean fuels and technologies for cooking, rural (% of rural population)	2024	NA
Afghanistan	Access to clean fuels and technologies for cooking (% of population)	2018	31.4
Afghanistan	Access to clean fuels and technologies for cooking (% of population)	2019	32.6
Afghanistan	Access to clean fuels and technologies for cooking (% of population)	2020	33.8

1.4 Handling Units of Measure

Different indicators use different units. We can classify units based on the Series Name and Series Code:

```
# Create a unit classification function
classify_unit <- function(series_name, series_code) {
  case_when(
    str_detect(series_name, "%") | str_detect(series_code, "\\.ZS$") ~ "Percentage",
    str_detect(series_name, "US\\$|current") | str_detect(series_code, "\\.CD$") ~ "Currency (
    str_detect(series_name, "per 1,?000") ~ "Rate per 1,000",
    str_detect(series_name, "per 100") ~ "Rate per 100",
    TRUE ~ "Other/Count"
  )
}

# Add unit classification
wdbd_long <- wdbd_long %>%
  mutate(Unit = classify_unit(`Series Name`, `Series Code`))

# Save the cleaned and tidy WDBD dataset
write_csv(wdbd_long, "clean_data/wdbd_tidy.csv")
cat("Saved tidy WDBD dataset to: clean_data/wdbd_tidy.csv\n")
```

```
Saved tidy WDBD dataset to: clean_data/wdbd_tidy.csv
```

```
# Show unit distribution
wdbd_long %>%
  count(Unit) %>%
  arrange(desc(n)) %>%
  kable(caption = "Distribution of measurement units")
```

Distribution of measurement units

Unit	n
Percentage	14322
Other/Count	5208
Currency (US$)	868
Rate per 1,000	434

1.5 Data Quality Checks

Missing Values Analysis

```
# Calculate missingness by series
missingness <- wdbd_long %>%
  group_by(`Series Code`, `Series Name`, Unit) %>%
  summarise(
    Total_Obs = n(),
    Missing = sum(is.na(Value)),
    Missing_Pct = round(100 * Missing / Total_Obs, 1),
    .groups = "drop"
  ) %>%
  arrange(desc(Missing_Pct))

# Save missingness analysis
write_csv(missingness, "clean_data/wdbd_missingness.csv")

# Top 10 series with highest missingness
missingness %>%
  head(10) %>%
  select(`Series Name`, Total_Obs, Missing, Missing_Pct) %>%
  kable(caption = "Top 10 indicators with highest missing data percentage")
```

Top 10 indicators with highest missing data percentage

Series Name	Total_Obs	Missing	Missing_Pct
Disaster risk reduction progress score (1-5 scale; 5=best)	434	434	100.0
Average working hours of children, study and work, female, ages 7-14 (hours per week)	434	434	100.0
Average working hours of children, study and work, male, ages 7-14 (hours per week)	434	434	100.0
Average working hours of children, study and work, ages 7-14 (hours per week)	434	434	100.0
Average working hours of children, working only, female, ages 7-14 (hours per week)	434	434	100.0
Average working hours of children, working only, male, ages 7-14 (hours per week)	434	434	100.0

Series Name	Total_Obs	Missing	Missing_Pct
Average working hours of children, working only, ages 7-14 (hours per week)	434	434	100.0
Adjusted net enrollment rate, primary, female (% of primary school age children)	434	424	97.7
Adjusted net enrollment rate, primary, male (% of primary school age children)	434	424	97.7
Adequacy of unemployment benefits and ALMP (% of total welfare of beneficiary households)	434	424	97.7

Out-of-Range Values for Percentages

```
# Check percentage values outside [0, 100]
out_of_range <- wdbd_long %>%
  filter(Unit == "Percentage", !is.na(Value)) %>%
  filter(Value < 0 | Value > 100)

if (nrow(out_of_range) > 0) {
  # Save out-of-range values
  write_csv(out_of_range, "clean_data/wdbd_out_of_range.csv")

  cat("Found", nrow(out_of_range), "percentage values outside [0, 100]\n")
  out_of_range %>%
    select(`Country Name`, `Series Name`, Year, Value) %>%
    head(10) %>%
    kable(caption = "Sample of out-of-range percentage values")
} else {
  cat("All percentage values are within valid [0, 100] range\n")
}
```

```
Found 63 percentage values outside [0, 100]
```

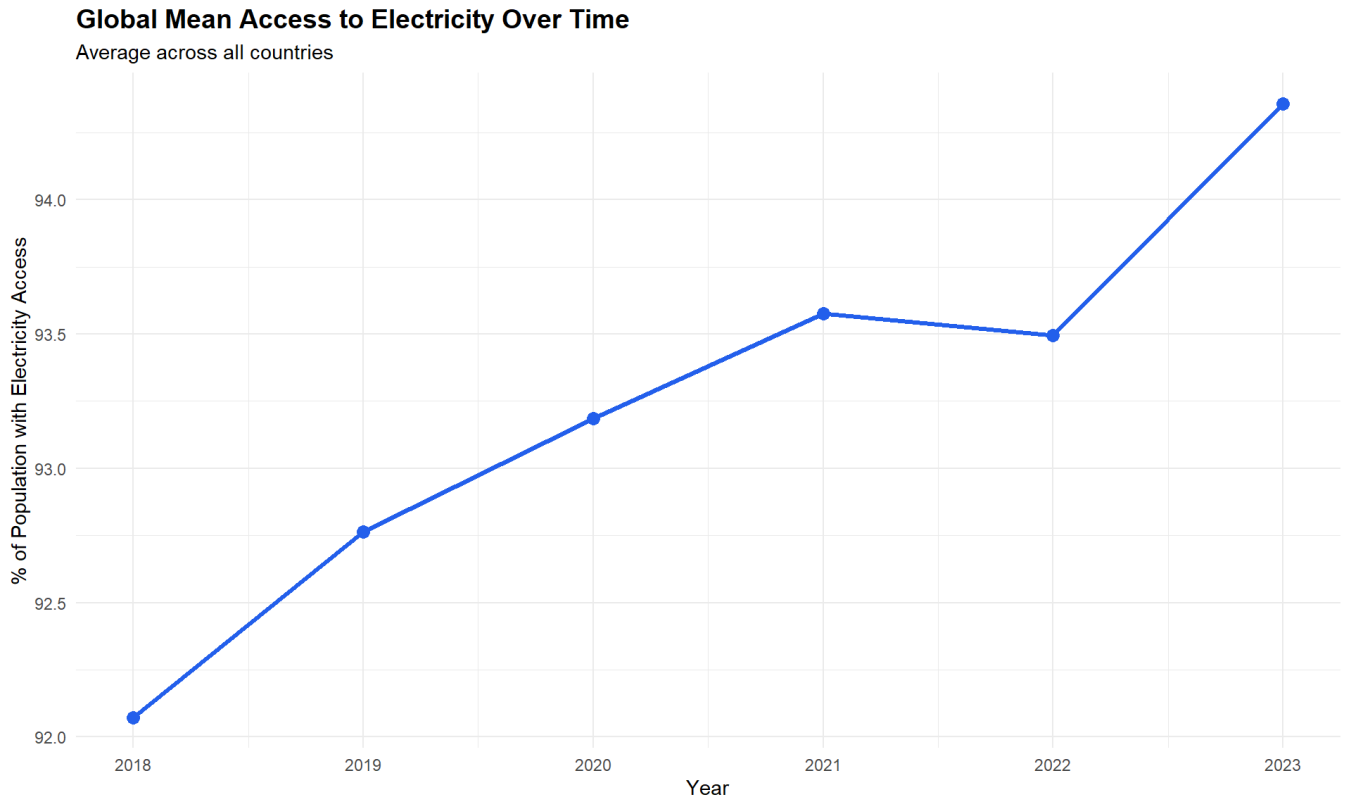
Sample of out-of-range percentage values

Country Name	Series Name	Year	Value
Afghanistan	Adjusted net national income (annual % growth)	2021	-19.505734
Argentina	Adjusted net national income (annual % growth)	2018	-5.606847
Argentina	Adjusted net national income (annual % growth)	2019	-5.094532
Argentina	Adjusted net national income (annual % growth)	2020	-7.836383
Bhutan	Adjusted net national income (annual % growth)	2020	-4.579493
Canada	Adjusted net national income (annual % growth)	2020	-7.425972
Costa Rica	Adjusted net national income (annual % growth)	2020	-4.693479
Ecuador	Adjusted net national income (annual % growth)	2018	-2.978351
Ecuador	Adjusted net national income (annual % growth)	2020	-5.041996
El Salvador	Adjusted net national income (annual % growth)	2020	-9.942253

1.6 Exploratory Visualizations

Access to Electricity Over Time

```
# Filter for electricity access indicator
electricity <- wdbd_long %>%
  filter(`Series Code` == "EG.ELC.ACCS.ZS", !is.na(Value))

# Global trend
electricity_trend <- electricity %>%
  group_by(Year) %>%
  summarise(Mean_Access = mean(Value, na.rm = TRUE))

ggplot(electricity_trend, aes(x = Year, y = Mean_Access)) +
  geom_line(color = "#2563eb", size = 1.2) +
  geom_point(color = "#2563eb", size = 3) +
  labs(
    title = "Global Mean Access to Electricity Over Time",
    subtitle = "Average across all countries",
    x = "Year",
    y = "% of Population with Electricity Access"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14))
```



Summary Statistics

```
# Overall summary
wdbd_summary <- wdbd_long %>%
```

```
  summarise(
    Total_Rows = n(),
    Countries = n_distinct(`Country Code`),
    Indicators = n_distinct(`Series Code`),
    Years = n_distinct(Year),
    Year_Range = paste(min(Year), "-", max(Year)),
    Non_Missing = sum(!is.na(Value)),
    Missing = sum(is.na(Value)),
    Missing_Pct = round(100 * Missing / (Non_Missing + Missing), 1)
  )

t(wdbd_summary) %>%
  as.data.frame() %>%
  rownames_to_column("Metric") %>%
  kable(col.names = c("Metric", "Value"),
        caption = "WDBD Dataset Summary Statistics")
```

WDBD Dataset Summary Statistics

Metric	Value
Total_Rows	20832
Countries	62
Indicators	48
Years	7
Year_Range	2018 - 2024
Non_Missing	8180
Missing	12652
Missing_Pct	60.7

Part 2: Movies Data (MovieLens)

2.1 Identifying Messy Aspects

The movies.csv file has several tidy data violations:

1. **Multiple values in one cell**: The `genres` column contains multiple pipe-separated values (e.g., "Action|Adventure|Sci-Fi")
2. **Year embedded in title**: Release year is stored within the title string instead of a separate column
3. **Inconsistent genre notation**: Some movies have "(no genres listed)" instead of proper NA values

2.2 Importing Movies Data

```r
# Import movies dataset
movies_raw <- read_csv("../AssignE/movies.csv", show_col_types = FALSE)

cat("Movies dataset dimensions:", dim(movies_raw), "\n")
```

```
Movies dataset dimensions: 4000 3
```

```r
head(movies_raw, 5) %>%
  kable(caption = "Original movies data structure")
```

Original movies data structure

movieId	title	genres
182337	Cinétracts (1968)	(no genres listed)
195495	Familia (2005)	Drama
3078	Liberty Heights (1999)	Drama
134704	Comedy Central Roast of Charlie Sheen (2011)	Comedy
219976	47 Hours to Live (2019)	Horror|Thriller

2.3 Tidying Movies Data

```r
# Extract year from title and create clean title
movies_tidy <- movies_raw %>%
  # Extract year (4 digits in parentheses at end)
  mutate(
    year = as.integer(str_extract(title, "\\((\\d{4})\\)$") %>% str_extract("\\d{4}")),
    clean_title = str_remove(title, "\\s*\\(\\d{4}\\)\\s*$")
  ) %>%
  # Handle missing genres
  mutate(genres = ifelse(genres == "(no genres listed)", NA_character_, genres)) %>%
  # Split genres and create one row per genre
  separate_rows(genres, sep = "\\|") %>%
  # Rename for clarity
  rename(genre = genres)

cat("\nTidy movies dimensions:", dim(movies_tidy), "\n")
```

```
Tidy movies dimensions: 7083 5
```

```r
cat("Unique movies:", n_distinct(movies_tidy$movieId), "\n")
```

```
Unique movies: 4000
```

```r
cat("Unique genres:", n_distinct(movies_tidy$genre, na.rm = TRUE), "\n")
```

```
Unique genres: 19
```

```
# Save the tidy movies dataset
write_csv(movies_tidy, "clean_data/movies_tidy.csv")
cat("Saved tidy movies dataset to: clean_data/movies_tidy.csv\n")
```

```
Saved tidy movies dataset to: clean_data/movies_tidy.csv
```

```
head(movies_tidy)
```

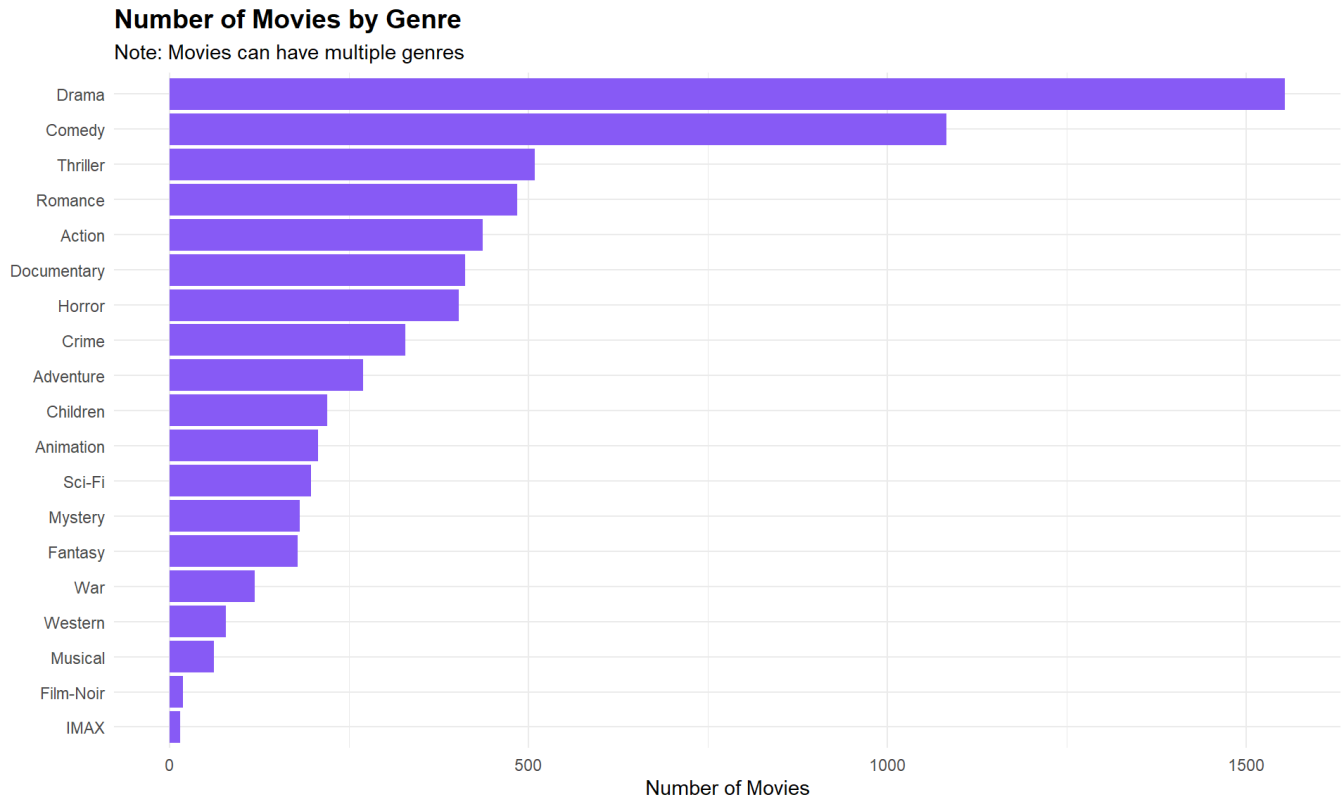
```
# A tibble: 6 × 5
  movieId title                                        genre    year clean_title
    <dbl> <chr>                                        <chr>   <int> <chr>
1  182337 Cinétracts (1968)                            <NA>     1968 Cinétracts
2  195495 Familia (2005)                               Drama    2005 Familia
3    3078 Liberty Heights (1999)                       Drama    1999 Liberty He…
4  134704 Comedy Central Roast of Charlie Sheen (2011) Comedy   2011 Comedy Cen…
5  219976 47 Hours to Live (2019)                      Horror   2019 47 Hours t…
6  219976 47 Hours to Live (2019)                      Thrill…  2019 47 Hours t…
```

```
# Movies without genres
no_genre <- movies_raw %>%
  filter(genres == "(no genres listed)") %>%
  select(movieId, title)

# Save movies with no genres
write_csv(no_genre, "clean_data/movies_no_genre.csv")

cat("Movies with no genres listed:", nrow(no_genre), "\n")
```

```
Movies with no genres listed: 335
```

2.4 Genre Distribution

```
# Count movies per genre
genre_counts <- movies_tidy %>%
  filter(!is.na(genre)) %>%
  count(genre, sort = TRUE)

ggplot(genre_counts, aes(x = reorder(genre, n), y = n)) +
  geom_col(fill = "#8b5cf6") +
  coord_flip() +
  labs(
    title = "Number of Movies by Genre",
    subtitle = "Note: Movies can have multiple genres",
    x = NULL,
    y = "Number of Movies"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14))
```



Part 3: Extended Analysis with MovieLens Data

3.1 Ratings Analysis

```
# Import ratings
ratings_raw <- read_csv("../AssignE/ratings.csv", show_col_types = FALSE)

cat("Ratings dataset dimensions:", dim(ratings_raw), "\n")
```

```
Ratings dataset dimensions: 32000204 4
```

```
cat("Unique users:", n_distinct(ratings_raw$userId), "\n")
```

```
Unique users: 200948
```

```
cat("Unique movies rated:", n_distinct(ratings_raw$movieId), "\n")
```

```
Unique movies rated: 84432
```

Average Rating and Count per Movie

```
# Compute per-movie statistics
movie_ratings <- ratings_raw %>%
  group_by(movieId) %>%
  summarise(
```

```
    rating_count = n(),
    rating_mean = mean(rating, na.rm = TRUE),
    rating_sd = sd(rating, na.rm = TRUE)
  ) %>%
  ungroup()

# Save movie ratings summary
write_csv(movie_ratings, "clean_data/movie_ratings_summary.csv")
cat("Saved movie ratings summary to: clean_data/movie_ratings_summary.csv\n")
```

```
Saved movie ratings summary to: clean_data/movie_ratings_summary.csv
```

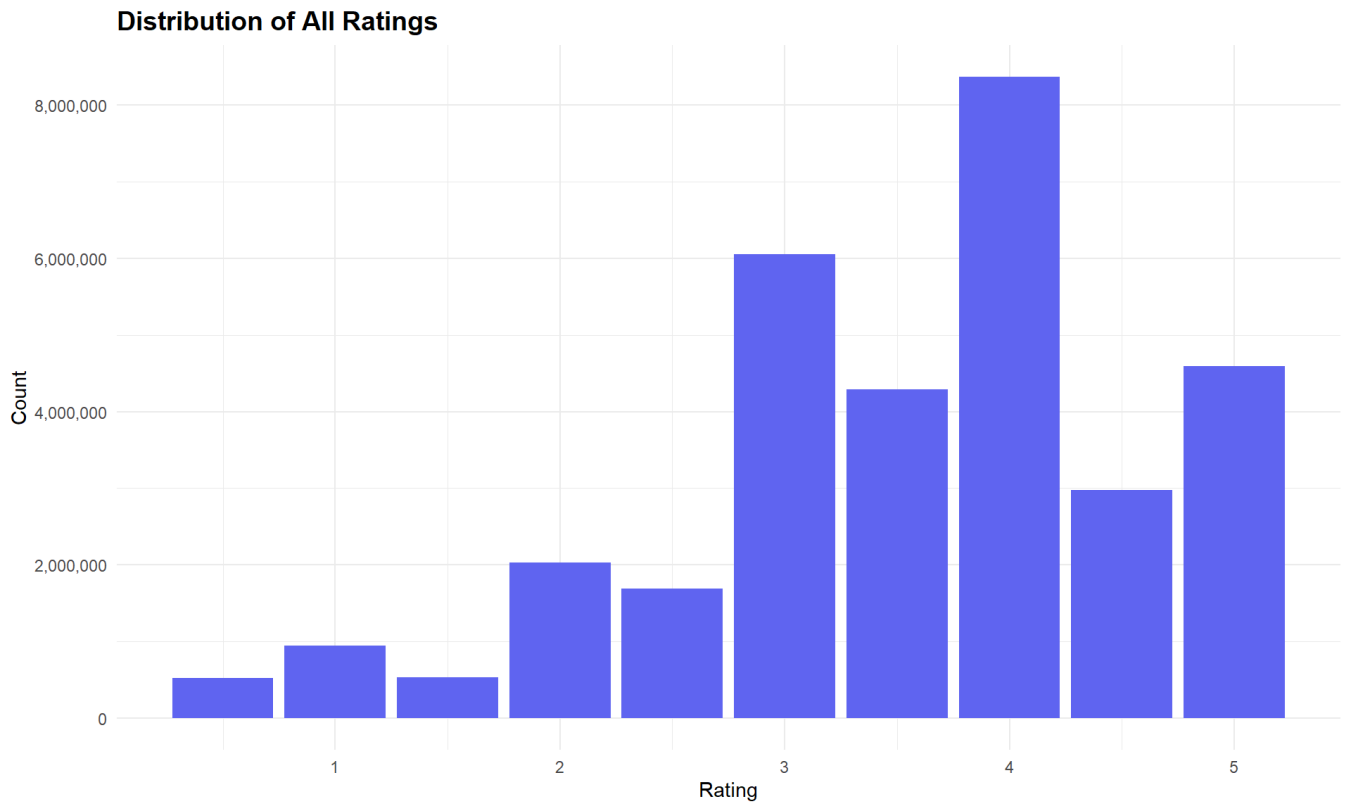
```
# Summary statistics
summary(movie_ratings) %>%
  kable(caption = "Summary of movie rating statistics")
```

Summary of movie rating statistics

movieId	rating_count	rating_mean	rating_sd
Min. : 1	Min. : 1	Min. :0.500	Min. :0.000
1st Qu.:109374	1st Qu.: 2	1st Qu.:2.543	1st Qu.:0.741
Median :166596	Median : 5	Median :3.071	Median :0.968
Mean :157209	Mean : 379	Mean :3.005	Mean :0.964
3rd Qu.:213536	3rd Qu.: 25	3rd Qu.:3.500	3rd Qu.:1.157
Max. :292757	Max. :102929	Max. :5.000	Max. :3.182
NA	NA	NA	NA's :18607

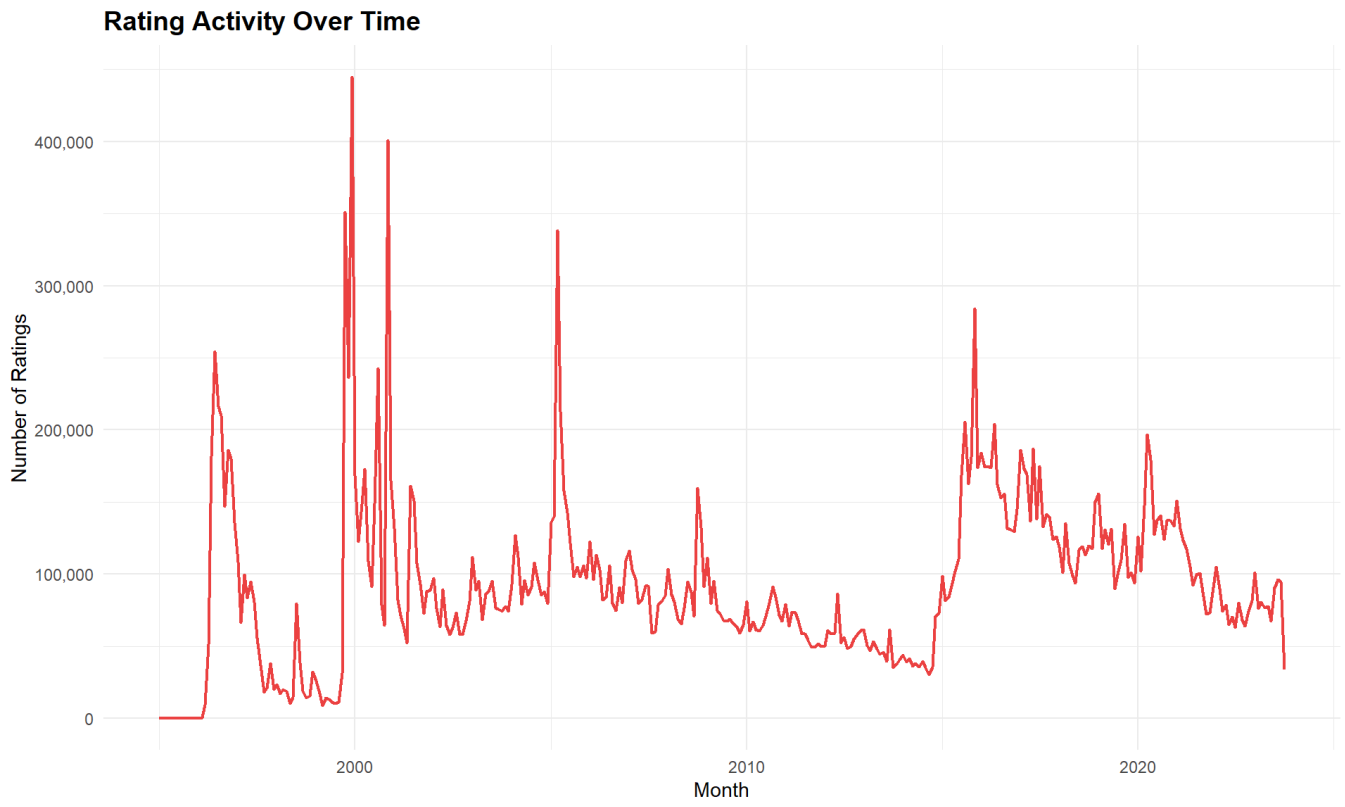
Distribution of Ratings

```
# Overall rating distribution
rating_dist <- ratings_raw %>%
  count(rating)

ggplot(rating_dist, aes(x = rating, y = n)) +
  geom_col(fill = "#6366f1") +
  scale_y_continuous(labels = comma) +
  labs(
    title = "Distribution of All Ratings",
    x = "Rating",
    y = "Count"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14))
```



Ratings Over Time

```
# Convert timestamp to date and aggregate by month
ratings_timeline <- ratings_raw %>%
  mutate(
    date = as_datetime(timestamp),
    month = floor_date(date, "month")
  ) %>%
  group_by(month) %>%
  summarise(
    rating_count = n(),
    avg_rating = mean(rating, na.rm = TRUE)
  )

ggplot(ratings_timeline, aes(x = month, y = rating_count)) +
  geom_line(color = "#ef4444", size = 0.8) +
  scale_y_continuous(labels = comma) +
  labs(
    title = "Rating Activity Over Time",
    x = "Month",
    y = "Number of Ratings"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14))
```



3.2 Tags Analysis

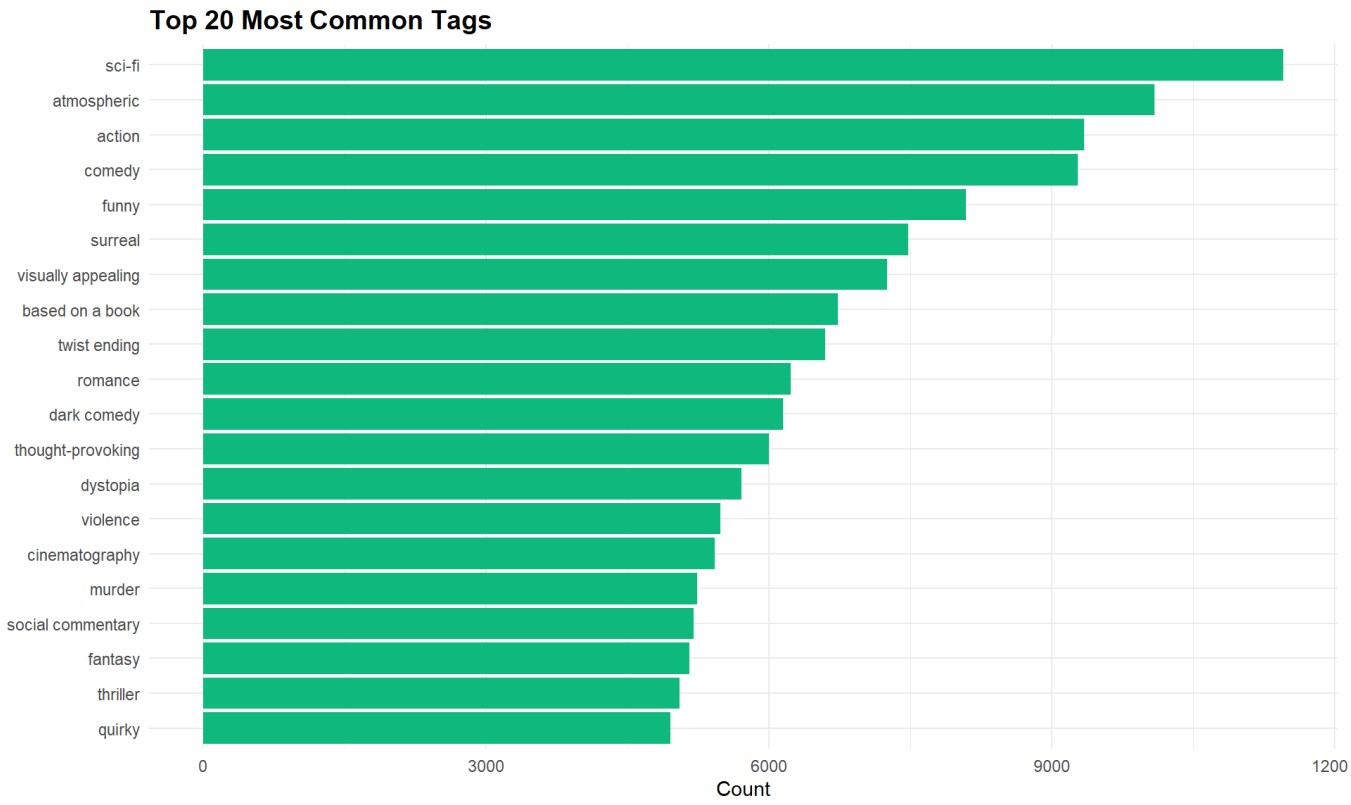
```
# Import and analyze tags
tags_raw <- read_csv("../AssignE/tags.csv", show_col_types = FALSE)

cat("Tags dataset dimensions:", dim(tags_raw), "\n")
```

```
Tags dataset dimensions: 2000072 4
```

Most Common Tags

```
# Clean and count tags
top_tags <- tags_raw %>%
  mutate(tag = str_to_lower(str_trim(tag))) %>%
  count(tag, sort = TRUE) %>%
  head(20)

# Save top tags
write_csv(top_tags, "clean_data/top_tags.csv")

ggplot(top_tags, aes(x = reorder(tag, n), y = n)) +
  geom_col(fill = "#10b981") +
  coord_flip() +
  labs(
    title = "Top 20 Most Common Tags",
    x = NULL,
    y = "Count"
  ) +
```

```
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14))
```



3.3 Links Analysis

```
# Import links dataset
links_raw <- read_csv("../AssignE/links.csv", show_col_types = FALSE)

# Check for missing identifiers
missing_links_summary <- links_raw %>%
  summarise(
    Total = n(),
    Missing_imdbId = sum(is.na(imdbId)),
    Missing_tmdbId = sum(is.na(tmdbId)),
    Missing_Either = sum(is.na(imdbId) | is.na(tmdbId))
  )

# Save detailed missing links data
missing_links_detail <- links_raw %>%
  filter(is.na(imdbId) | is.na(tmdbId))

write_csv(missing_links_detail, "clean_data/missing_links.csv")

missing_links_summary %>%
  kable(caption = "Missing external identifiers in links.csv")
```

Missing external identifiers in links.csv

Total	Missing_imdbId	Missing_tmdbId	Missing_Either
87585	0	124	124

Note on links.csv: This file provides external identifiers (IMDb and TMDb IDs) that could be used to enrich the dataset by joining with external movie databases for additional metadata like budgets, director information, or detailed cast lists.

Part 4: Cross-Dataset Analysis

4.1 Genre-Rating Patterns

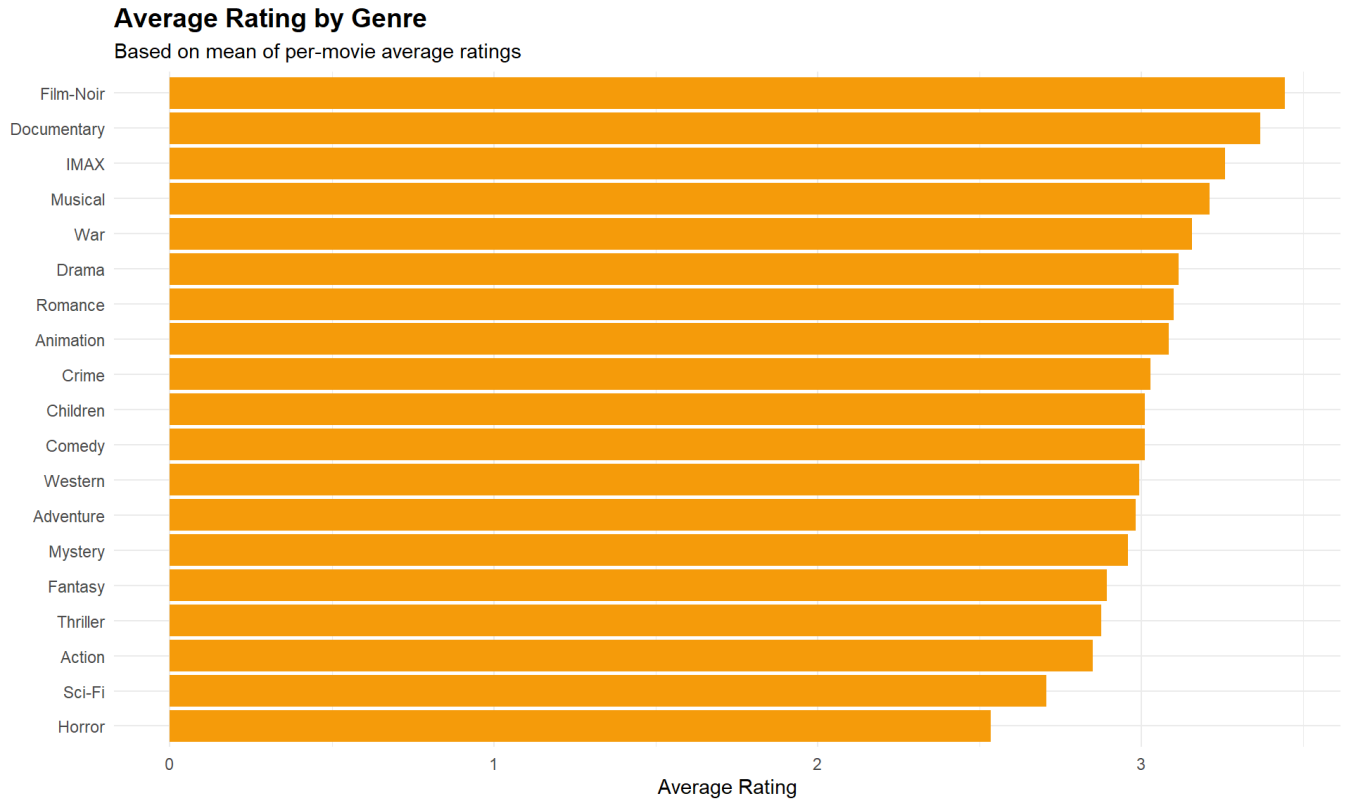
```
# Join movies and ratings to explore genre patterns
genre_ratings <- movies_tidy %>%
  filter(!is.na(genre)) %>%
  inner_join(movie_ratings, by = "movieId") %>%
  group_by(genre) %>%
  summarise(
    movie_count = n_distinct(movieId),
    total_ratings = sum(rating_count),
    avg_rating = mean(rating_mean, na.rm = TRUE)
  ) %>%
  arrange(desc(avg_rating))

# Save genre ratings analysis
write_csv(genre_ratings, "clean_data/genre_ratings.csv")

genre_ratings %>%
  kable(digits = 2, caption = "Genre statistics with average ratings")
```

Genre statistics with average ratings

genre	movie_count	total_ratings	avg_rating
Film-Noir	18	11427	3.44
Documentary	400	18771	3.37
IMAX	14	116933	3.26
Musical	57	111366	3.21
War	107	165632	3.16
Drama	1502	839375	3.12
Romance	467	406189	3.10
Animation	205	192533	3.08
Crime	310	219993	3.03
Children	211	193553	3.01
Comedy	1041	657782	3.01
Western	65	36547	2.99

genre	movie_count	total_ratings	avg_rating
Adventure	258	449312	2.98
Mystery	172	164205	2.96
Fantasy	176	140252	2.89
Thriller	498	466343	2.88
Action	418	455204	2.85
Sci-Fi	192	283387	2.71
Horror	391	105992	2.53

```
ggplot(genre_ratings, aes(x = reorder(genre, avg_rating), y = avg_rating)) +
  geom_col(fill = "#f59e0b") +
  coord_flip() +
  labs(
    title = "Average Rating by Genre",
    subtitle = "Based on mean of per-movie average ratings",
    x = NULL,
    y = "Average Rating"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14))
```



4.2 Top-Rated Movies

```
# Identify top-rated movies (with minimum rating threshold)
min_ratings <- 500
```

```r
# Ensure consistent types and one row per movie for joining
movies_reference <- movies_raw %>%
  transmute(
    movieId = as.integer(movieId),
    clean_title = str_remove(title, "\\s*\\(\\d{4}\\)\\s*$"),
    year = as.integer(str_extract(title, "\\((\\d{4})\\)") %>% str_extract("\\d{4}"))
  ) %>%
  distinct(movieId, .keep_all = TRUE)

movie_ratings_int <- movie_ratings %>% mutate(movieId = as.integer(movieId))

top_movies <- movie_ratings_int %>%
  filter(rating_count >= min_ratings) %>%
  inner_join(movies_reference, by = "movieId") %>%
  slice_max(order_by = rating_mean, n = 20, with_ties = FALSE) %>%
  transmute(
    Title = clean_title,
    Year = year,
    `Avg Rating` = round(rating_mean, 2),
    `# Ratings` = rating_count
  )

# Save top rated movies
write_csv(top_movies, "clean_data/top_rated_movies.csv")

kable(
  top_movies,
  digits = 2,
  col.names = c("Title", "Year", "Avg Rating", "# Ratings"),
  caption = paste0("Top 20 Rated Movies (minimum ", min_ratings, " ratings)")
)
```

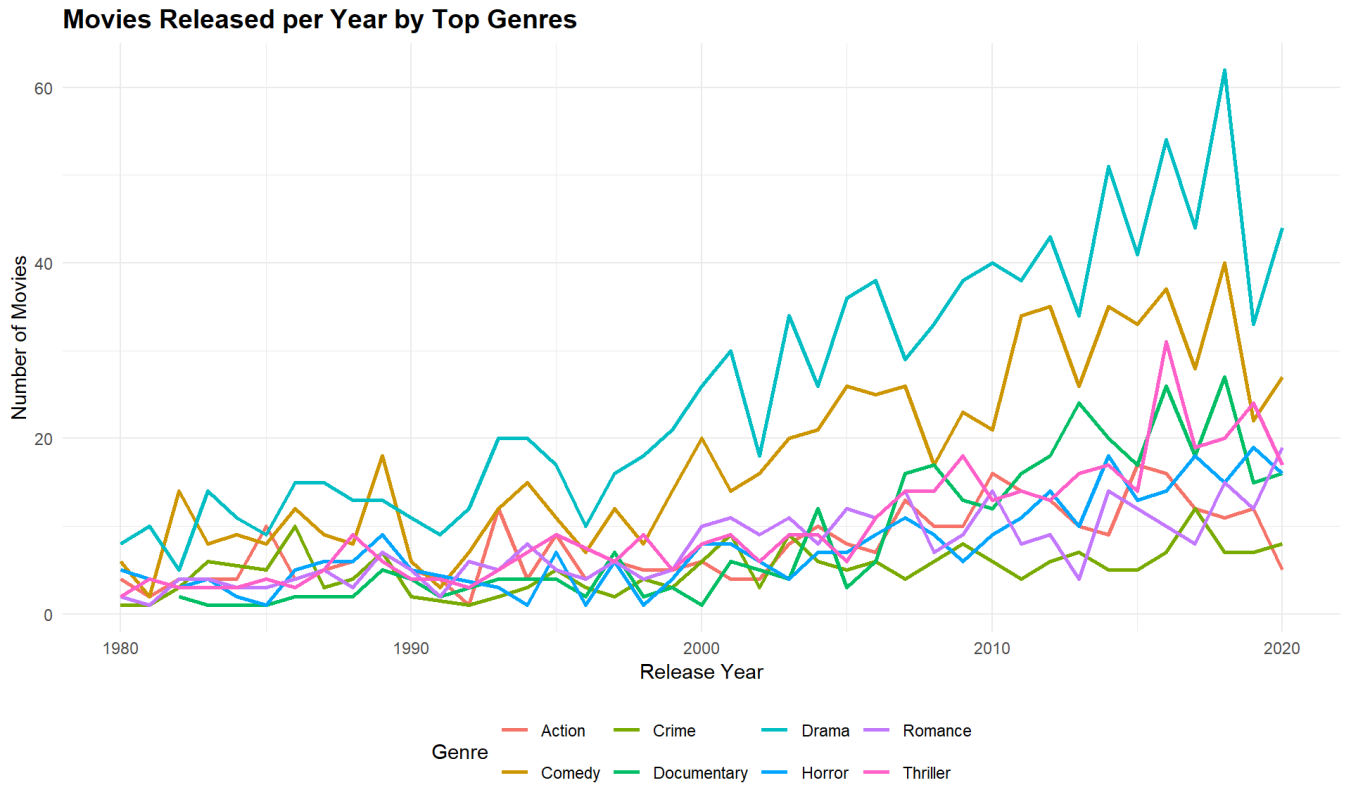
Top 20 Rated Movies (minimum 500 ratings)

Title	Year	Avg Rating	# Ratings
Godfather: Part II, The	1974	4.26	43111
Paths of Glory	1957	4.18	5490
Boot, Das (Boat, The)	1981	4.13	15804
Prestige, The	2006	4.12	32965
Vertigo	1958	4.12	19529
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)	1981	4.11	67408
Oppenheimer	2023	4.08	1152
Red Beard (Akahige)	1965	4.07	678
Forrest Gump	1994	4.05	100296
Inherit the Wind	1960	4.05	2230
Memories of Murder (Salinui chueok)	2003	4.05	2860

Title	Year	Avg Rating	# Ratings
Grand Day Out with Wallace and Gromit, A	1989	4.04	8854
Mister Roberts	1955	4.03	2644
Face in the Crowd, A	1957	4.03	777
Kind Hearts and Coronets	1949	4.02	1216
Coco	2017	4.02	9638
Moon	2009	4.01	15744
Crimes and Misdemeanors	1989	4.00	4760
Wolf Children (Okami kodomo no ame to yuki)	2012	4.00	1192
Strada, La	1954	3.99	1502

4.3 Genre Popularity Over Time

```
# Analyze genre trends by release year
genre_timeline <- movies_tidy %>%
  filter(!is.na(genre), !is.na(year), year >= 1980, year <= 2020) %>%
  count(year, genre)

# Top genres by total volume
top_genres <- movies_tidy %>%
  filter(!is.na(genre)) %>%
  count(genre, sort = TRUE) %>%
  head(8) %>%
  pull(genre)

# Plot trends for top genres
genre_timeline %>%
  filter(genre %in% top_genres) %>%
  ggplot(aes(x = year, y = n, color = genre)) +
  geom_line(size = 1) +
  labs(
    title = "Movies Released per Year by Top Genres",
    x = "Release Year",
    y = "Number of Movies",
    color = "Genre"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    legend.position = "bottom"
  )
```



4.4 Tag-Rating Correlation

```
# Join tags with ratings to see if certain tags correlate with ratings
# Focus on popular tags
popular_tags <- tags_raw %>%
  mutate(tag = str_to_lower(str_trim(tag))) %>%
  count(tag, sort = TRUE) %>% head(15) %>% pull(tag)

tag_ratings <- tags_raw %>%
  mutate(tag = str_to_lower(str_trim(tag))) %>%
  filter(tag %in% popular_tags) %>%
  inner_join(movie_ratings, by = "movieId") %>%
  group_by(tag) %>% summarise(
    movies_tagged = n_distinct(movieId),
    avg_rating = mean(rating_mean, na.rm = TRUE) ) %>%
  arrange(desc(avg_rating))

# Save tag-rating correlation
write_csv(tag_ratings, "clean_data/tag_ratings.csv")

tag_ratings %>% kable( digits = 2,
    col.names = c("Tag", "# Movies Tagged", "Avg Rating"),
    caption = "Average ratings for movies with popular tags"
  )
```

Average ratings for movies with popular tags

Tag	# Movies Tagged	Avg Rating

Tag	# Movies Tagged	Avg Rating
thought-provoking	360	3.91
twist ending	386	3.86
atmospheric	817	3.84
dark comedy	735	3.80
surreal	687	3.78
cinematography	757	3.77
visually appealing	481	3.76
sci-fi	789	3.65
dystopia	441	3.63
based on a book	1670	3.60
funny	1486	3.56
action	1154	3.55
romance	1676	3.55
violence	2231	3.53
comedy	2009	3.50

Insight: Tags like “atmospheric”, or “thought-provoking” tend to be associated with higher-rated movies, while more generic tags show average ratings closer to the overall mean.

Summary: Cleaned Datasets

All cleaned datasets have been saved to the `clean_data/` folder:

Part 1: WDBD Data

- **wdbd_tidy.csv**: Main tidy dataset with all indicators in long format
- **wdbd_missingness.csv**: Missingness analysis by series
- **wdbd_out_of_range.csv**: Percentage values outside valid range (if any)

Part 2: Movies Data

- **movies_tidy.csv**: Tidy movies dataset with one row per movie-genre combination
- **movies_no_genre.csv**: Movies without genre information
- **movie_ratings_summary.csv**: Per-movie rating statistics
- **top_rated_movies.csv**: Top 20 rated movies (minimum 500 ratings)
- **top_tags.csv**: Top 20 most common tags
- **missing_links.csv**: Movies with missing IMDb or TMDb IDs
- **genre_ratings.csv**: Average ratings by genre
- **tag_ratings.csv**: Tag-rating correlation analysis