RMarkdown

2025-10-06

RMarkdown

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.

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:

INTRODUCTION

This document provides a detailed report of our data exploration where we focus on exploring the raw data before any cleaning or modeling occurs, including: - How the datasets were loaded - Data quality observations: missing values and outliers - An overview of the structure and the relevant variables

RESEARCH QUESTION

How does runtime influence audience ratings for movies compared to TV episodes, controlling for the release year?

RESEARCH METHOD

By conducting a multiple linear regression, we will estimate the independent effect of runtime on ratings while controlling for release year and content type (movie/TV episode).

DEPENDENCIES

For the data exploration step, we used the following dependencies: - R - Make - Installed packages in R:

```r
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   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- 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 errors
```

```r
library(ggplot2)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

LOADING THE DATASETS

Two separate datasets were acquired from IMDb (https://datasets.imdbws.com/): - The first dataset contains information on the contents' duration and release year, saved as "title_basics_raw" - The second dataset contains information on the contents' ratings, saved as "ratings_raw"

The following chunk of code will first create the folder "raw" to store the raw datasets, and second, load the datasets into R:

```r
# Ensure the "raw" folder exists
dir.create("raw", recursive = TRUE, showWarnings = FALSE)

# Retrieve dataset for Movie and TV Episode Duration and Release Year
title_url <- "https://datasets.imdbws.com/title.basics.tsv.gz"
title_basics_raw <- read_tsv(title_url, na = c("\\N", ""))
```

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

```
## Rows: 11951516 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (5): tconst, titleType, primaryTitle, originalTitle, genres
## dbl (4): isAdult, startYear, endYear, runtimeMinutes
##
## 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.
```

```
# Retrieve dataset for rating
rating_url <- "https://datasets.imdbws.com/title.ratings.tsv.gz"
ratings_raw <- read_tsv(rating_url, na = c("\\N", ""))
```

```
## Rows: 1622212 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (1): tconst
## dbl (2): averageRating, numVotes
##
## 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.
```

```
# Save the downloaded datasets to folder "raw"
write_tsv(title_basics_raw, "raw/title_basics_raw.tsv")
write_tsv(ratings_raw, "raw/ratings_raw.tsv")
```

DATA QUALITY OBSERVATIONS

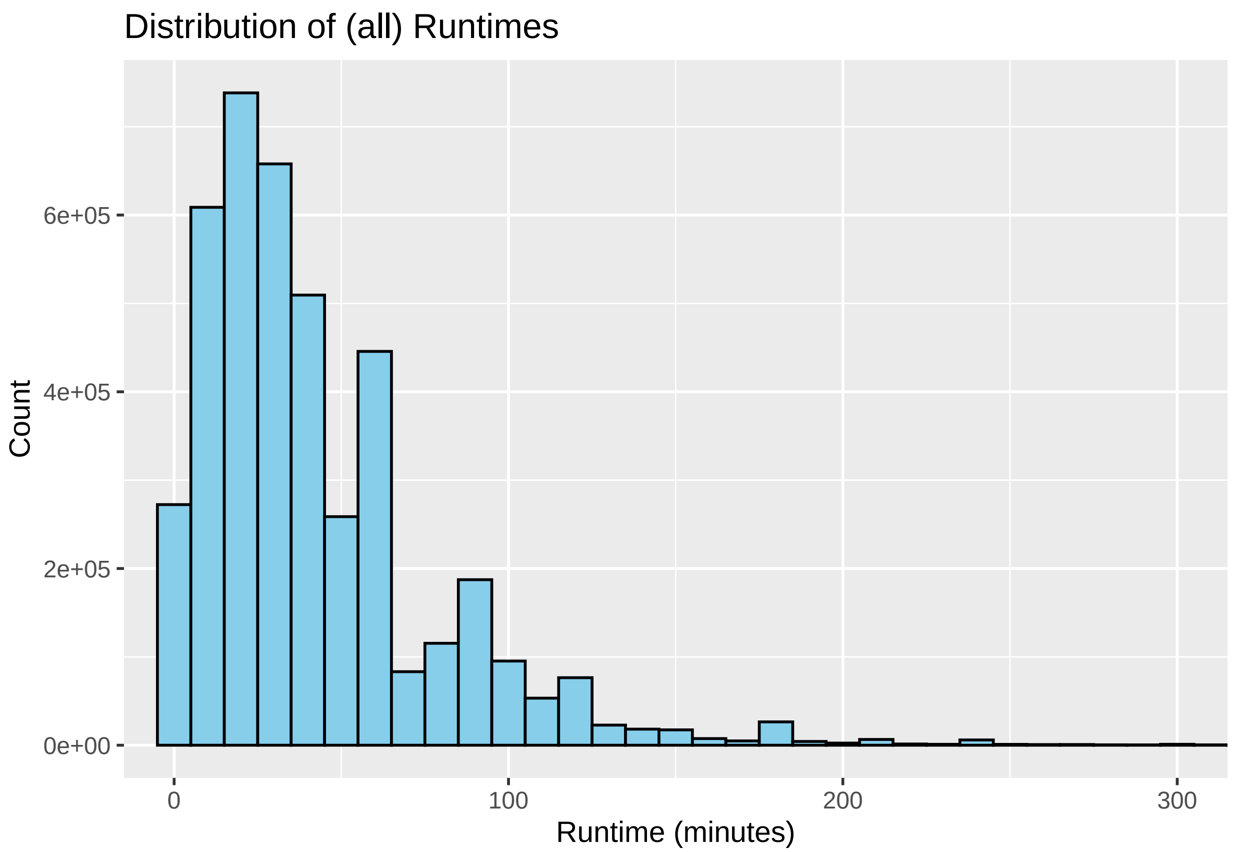
The following chunk of code provides a data quality check to identify missing values and outliers:

```
# Data quality check for title_basics_raw
  # Missing values
colSums(is.na(title_basics_raw[c("tconst", "primaryTitle", "startYear", "runtimeMinutes")]))
```

```
##         tconst   primaryTitle      startYear runtimeMinutes
##              0              0        1441256        7719868
```

```
  # Outliers
setDT(title_basics_raw)
ggplot(title_basics_raw, aes(x = runtimeMinutes)) +
  geom_histogram(binwidth = 10, fill = "skyblue", color = "black") +
  coord_cartesian(xlim = c(0, 300)) +  # focus on typical runtimes
  labs(x = "Runtime (minutes)", y = "Count", title = "Distribution of (all) Runtimes")
```

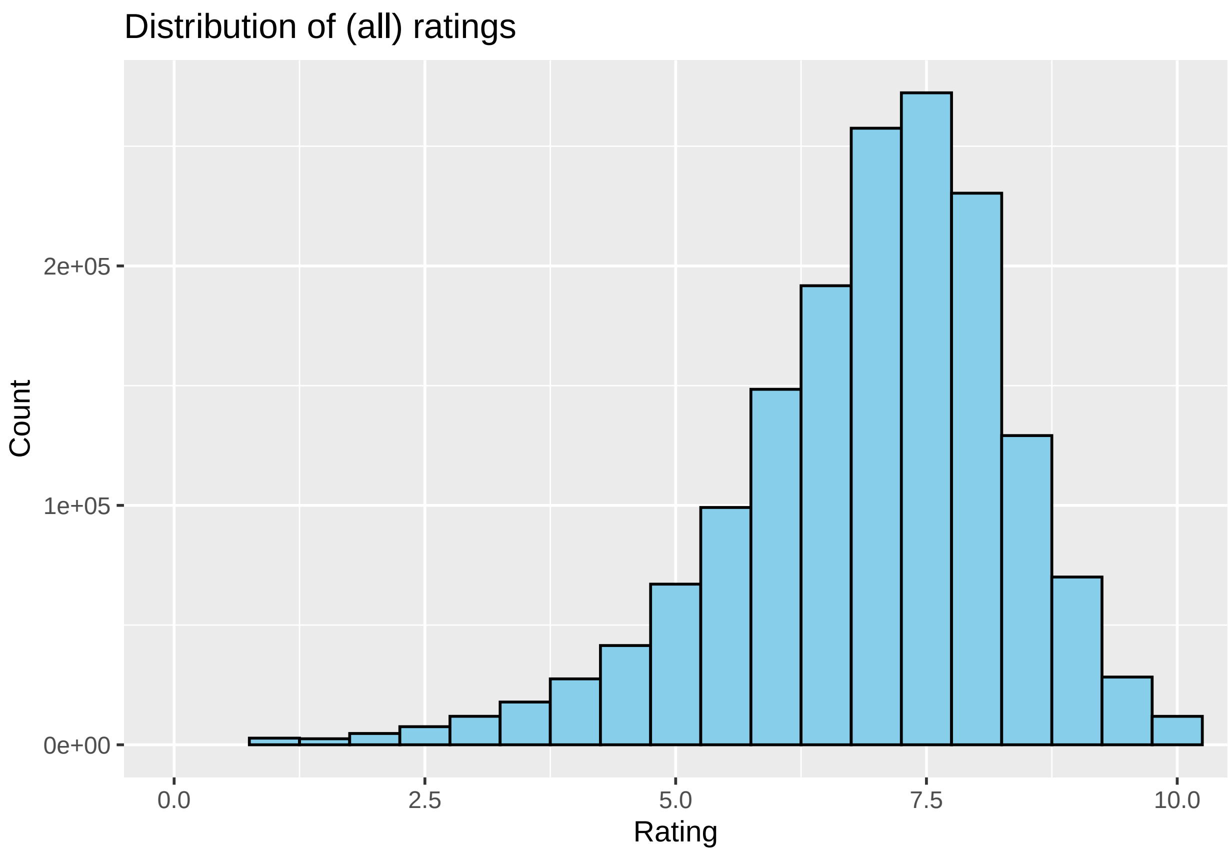
```
## Warning: Removed 7719868 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



```
# Data quality check for title_ratings
  # Missing values
colSums(is.na(ratings_raw[c("tconst","averageRating")]))
```

```
##        tconst averageRating
##             0             0
```

```
  # Outliers
setDT(ratings_raw)
ggplot(ratings_raw, aes(x = averageRating)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black") +
  coord_cartesian(xlim = c(0, 10)) +
  labs(x = "Rating", y = "Count", title = "Distribution of (all) ratings")
```



OVERVIEW OF THE STRUCTURE AND RELEVANT VARIABLES

Below you will find the variable names and descriptions that are relevant to this study. 1. tconst(string): alphanumeric unique identifier of the title 2. primaryTitle (string):the more popular title/the title used by the filmmakers on promotional materials at the point of release 3. startYear: represents the release year of a title 4. runtimeMinutes: primary runtime of the title, in minutes 5. averageRating: weighted average of all the individual user ratings 6. is_tvepisode (boolean): 0: content type is Movie, 1: content type is TV Episode

DATA CLEANING

The following chunk of code will first create the folder "tmp" to store the tmp datasets, and second, filter only on movies and TVepisodes.

```r
# Ensure the "tmp" folder exists
dir.create("tmp", recursive = TRUE, showWarnings = FALSE)

# Both \N and empty strings to be treated as NAs
title_basics_raw <- read_tsv("raw/title_basics_raw.tsv", na = c("\\N", ""), show_col_types = FALSE)
```

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

```
# Filter only movies and TVepisodes > save in tmp
filtered_movies_TVepisodes <- title_basics_raw %>%
  filter(titleType %in% c("movie", "tvEpisode"))

# Output cleanded data
write_tsv(filtered_movies_TVepisodes, "tmp/filtered_movies_TVepisodes.tsv")
```

DATA MERGING

The following chunk of code will merge the datasets and make a dummy variable.

```
# Merge with ratings
merged_data <- filtered_movies_TVepisodes %>%
  left_join(ratings_raw, by = "tconst")

cat("Number of unmatched rows:", sum(is.na(merged_data$averageRating)), "\n")
```

```
## Number of unmatched rows: 8764181
```

```
#  Create dummy variable for TVepisodes > save in tmp
merged_data <- merged_data %>%
  mutate(is_tvepisode = ifelse(titleType == "tvEpisode", 1,0))

# Output merged_data
write_tsv(merged_data, "tmp/merged_data.tsv")
```

DATA FINAL CLEANING

The following chunk of code will provide the final cleaning. This includes selecting revelant columns, removing missing values, creating the gen folder and removing the extreme outliers.

```
# Select relevant columns, remove missing values, de-duplicate > save in gen
analysis_data <- fread("tmp/merged_data.tsv")
```

```
## Warning in fread("tmp/merged_data.tsv"): Stopped early on line 777000. Expected
## 12 fields but found 13. Consider fill=TRUE and comment.char=. First discarded
## non-empty line: <<tt10233364 tvEpisode Rolling in the Deep Dish Rolling in the
## Deep Dish 0 2019 NA NA NA NA NA NA 1>>
```

```
analysis_data[, c("originalTitle", "titleType", "isAdult", "endYear", "genres", "numVotes") := NULL]
analysis_data <- analysis_data[!is.na(runtimeMinutes) & !is.na(averageRating)& !is.na(startYear)] %>% di

write_tsv(analysis_data, "tmp/analysis_data.tsv")
```

```
# Create gen folder for analysis_data_clean
dir.create("gen", recursive = TRUE, showWarnings = FALSE)

# Remove extreme outliers for runtime
analysis_data_clean <- analysis_data %>%
  group_by(is_tvepisode) %>%
  filter({
    Q1 <- quantile(runtimeMinutes, 0.25)
    Q3 <- quantile(runtimeMinutes, 0.75)
    IQR_val <- Q3 - Q1
    runtimeMinutes >= (Q1 - 1.5 * IQR_val) & runtimeMinutes <= (Q3 + 1.5 * IQR_val)
  }) %>%
  ungroup()

# Remove extreme outliers for runtime TVepisodes
analysis_data_clean %>%
  filter(runtimeMinutes < 15, is_tvepisode==1) %>%
  summarise(count = n())
```

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1  1710
```

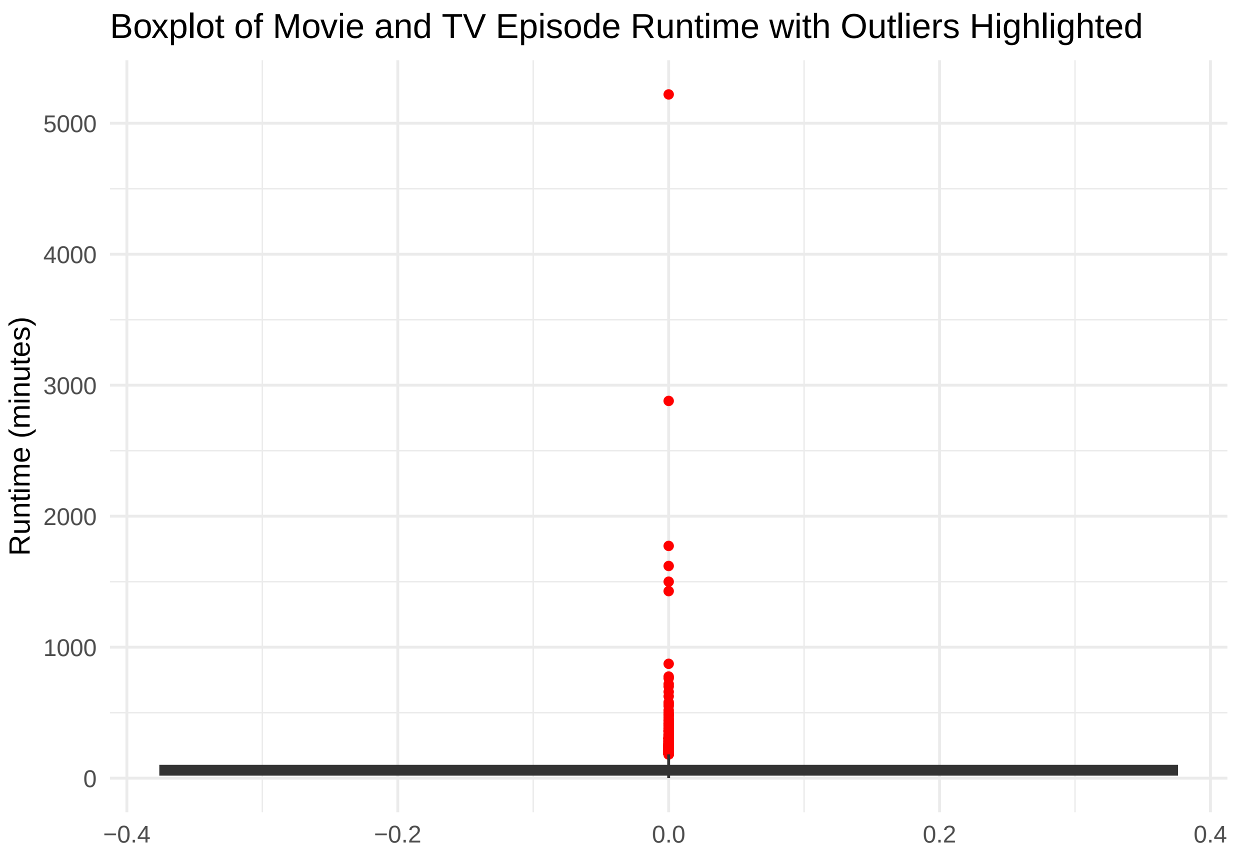
```
## Removing TV episodes that are shorter than 15 minutes or longer than 65 minutes
analysis_data_clean <- analysis_data_clean %>%
  filter(is_tvepisode != 1 | (runtimeMinutes >= 15 & runtimeMinutes <= 65))

# Output final cleaned data
write_tsv(analysis_data_clean, "gen/analysis_data_clean.tsv")
```

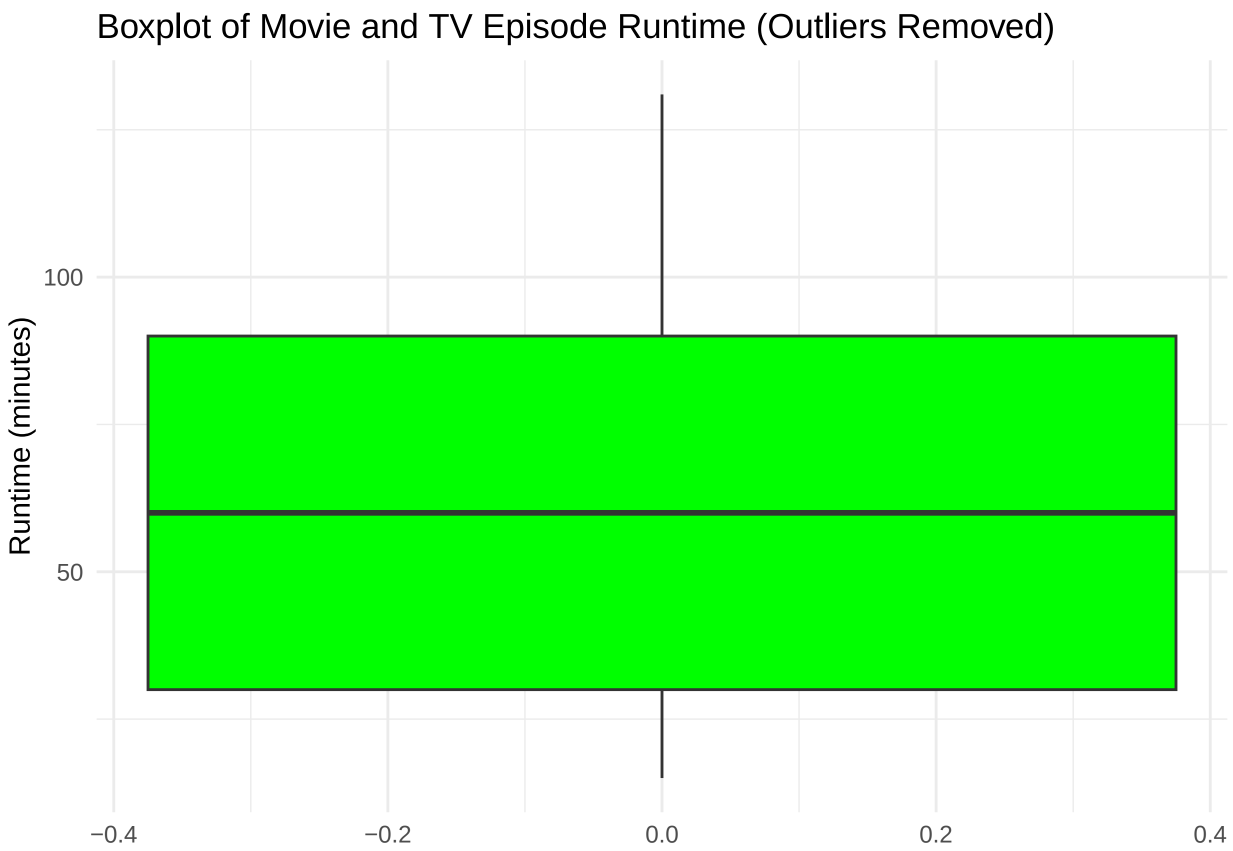
INCLUDING PLOTS

The following chunk of code will provide the boxplots that are used for the final cleaning of the data.

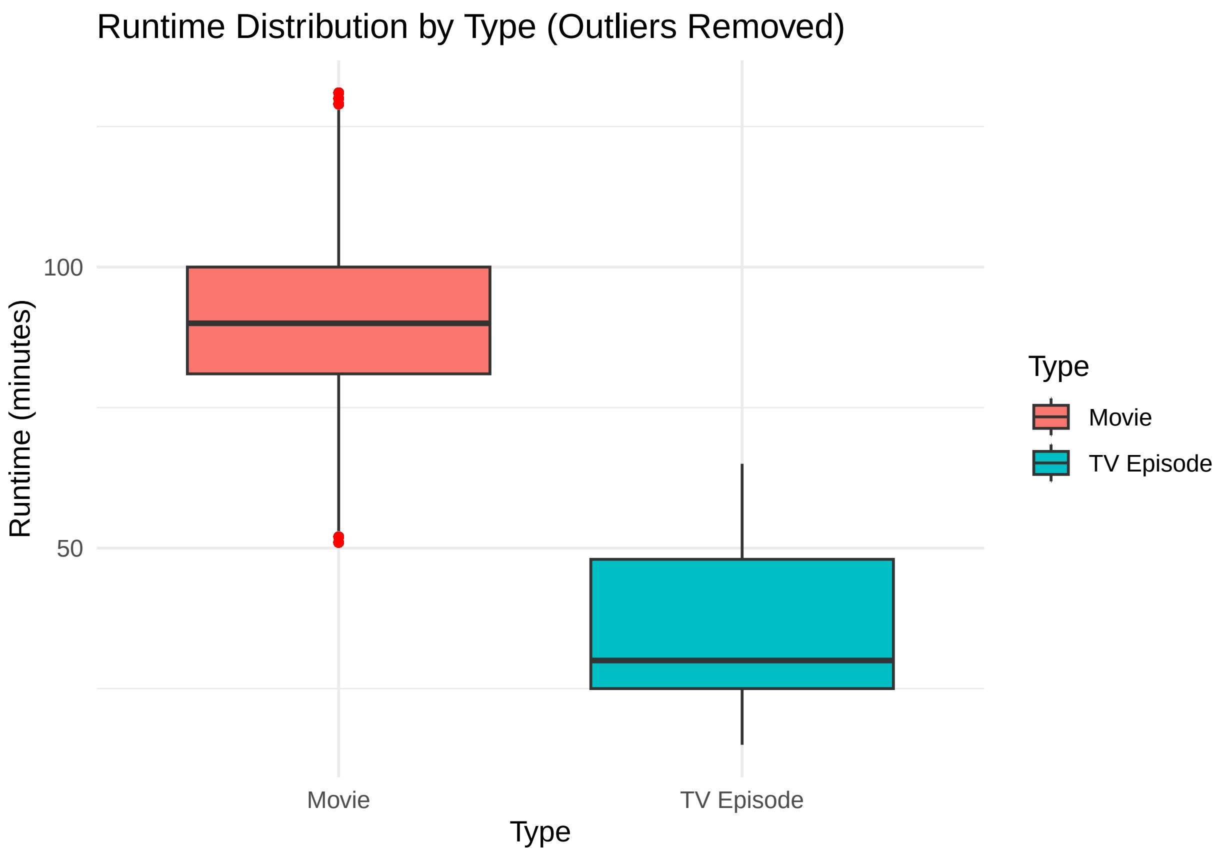
```
# Analyzing the date through a plot
ggplot(analysis_data, aes(y = runtimeMinutes)) +
  geom_boxplot(fill = "blue", outlier.color = "red", outlier.shape = 16) +
  labs(
    y = "Runtime (minutes)",
    title = "Boxplot of Movie and TV Episode Runtime with Outliers Highlighted"
  ) +
  theme_minimal()
```



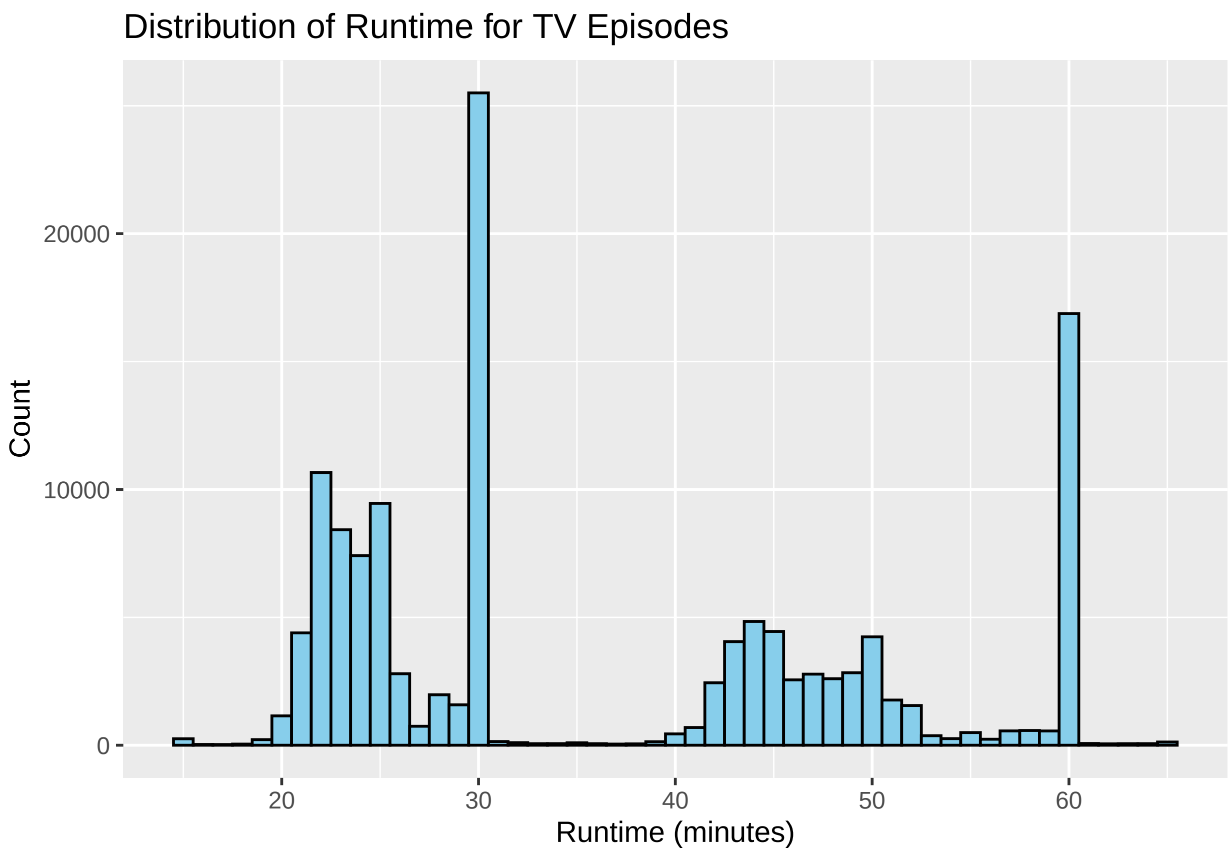
```r
# Boxplot with outliers Removed
ggplot(analysis_data_clean, aes(y = runtimeMinutes)) +
  geom_boxplot(fill = "green") +
  labs(
    y = "Runtime (minutes)",
    title = "Boxplot of Movie and TV Episode Runtime (Outliers Removed)"
  ) +
  theme_minimal()
```



```
# Plot after removing outliers per group
ggplot(analysis_data_clean, aes(x = factor(is_tvepisode, labels = c("Movie", "TV Episode")),
                                y = runtimeMinutes,
                                fill = factor(is_tvepisode, labels = c("Movie", "TV Episode")))) +
  geom_boxplot(outlier.color = "red", outlier.shape = 16) +
  labs(
    x = "Type",
    y = "Runtime (minutes)",
    fill = "Type",
    title = "Runtime Distribution by Type (Outliers Removed)"
  ) +
  theme_minimal()
```



```r
# Analysing the data through a plot
analysis_data_clean %>%
  filter(is_tvepisode == 1) %>%
  ggplot(aes(x = runtimeMinutes)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Runtime for TV Episodes",
       x = "Runtime (minutes)",
       y = "Count")
```



REGRESSION ANALYSIS

The following chunk of code will provide the regression analysis.

```
# Change runtimeMinutes to mean
analysis_data_clean <- analysis_data_clean %>%
  mutate(mruntimeMinutes = runtimeMinutes - mean(runtimeMinutes, na.rm = TRUE))

# Change baseline to 1896
analysis_data_clean <- analysis_data_clean %>%
  mutate(startYearCentered = startYear - 1896)

# Regression with interaction effect controlling for release year
model_1 <- lm(averageRating ~ mruntimeMinutes * is_tvepisode + startYearCentered, data = analysis_data_
```