Statistical Test and Power of Movie Rating Data

Erna Kuginyte 220013309

14/11/2022

I confirm that the following report and associated code is my own work, except where clearly indicated.

Abstract

This report summarises the statistical modelling and analysis results associated with movie ratings of Adventure, Comedy and Drama genres. The goal of this report is to check if the rating distributions across the genres are statistically similar. The data "movielens" analysed has been taken from dslabs package in R. The study uses Anova and Kruskal-Wallis Rank Sum Test, together with power and test size computation by simulation. The report results in having statistically different rating distributions in the similar to "movielens" simulated data among these movie genres. The simulation suggests to use 1500 samples with the original effect size, for the null hypotheses to be correctly rejected 90% of the time.

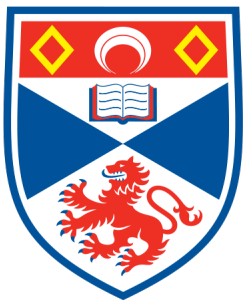
Introduction

The analysis is divided into two parts: Part 1, provides exploratory data analysis, data normality check; Part 2, simulated data distributions of the rankings across Adventure, Comedy and Drama genres, power and test size results from each scenario.

The data used in this study is imported from the R library "dslabs" titled "movielens". The data frame includes: movieId - unique ID for the movie, title - movie title (not unique), year - year the movie was released, genres - genres associated with the movie, userId - unique ID for the user, rating - a rating between 0 and 5 for the movie, timestamp - date and time the rating was given. Data assumed to be independent, there are a few missing values.

The purpose of the study is to determine whether there is statistical difference between rating distributions in Adventure, Comedy and Drama movie genres, also to compute a number of samples and effect size that should used taken to reach a 90% power when null hypothesis is known to be (analytically) false, and reach a 5% test size value when null hypothesis is known to be correct.

The data is simulated multiple times using the Monte Carlo data simulation method to increase the study's reliability. Simulation is a common tool for gaining insight into how variables (number of data samples, effect size, power, test size) impact both the variables themselves and the performance of statistical tests (i.e. how reliable they are). The study uses Anova and Kruskal-Wallis Rank Sum statistical tests.



Methods

Analysis is done using R Studio IDE version 2022.02 Libraries used: dslabs - to extract the data of movie ratings, tidyverse.

Part 1

The "movielens" data frame has eighteen rows of missing values in title, year and genres columns which are omitted for the purpose of this study. The timestamp column is converted to date. Exploratory analysis is completed using visual exploration, plots include: rating average from release year, user ID, genre; rating count from genre; rating frequency from rating value of adventure, comedy and drama genres; rating probabilities from rating value of adventure, comedy and drama genres. See Table 1 for head of wrangled data. Wilk-Shapiro test is used to check the normality of the data distribution.

Part 2

Refer to Appendices for the code diagram. First comes in the Monte-Carlo simulation method to generate a new data set based on probabilities for each rating.[1] The study uses 2 statistical tests to check whether there is any statistical difference between the means, and since the data displayed non-normality, it is a good idea to run a more robust non-parametric test: parametric Anova and non-parametric Kruskal-Wallis Rank Sum are used. Second scenario takes a greater sample size to get a desirable 90% power value. Third scenario computes test size under Null Hypothesis being true. Fourth scenario computes power under a small effect size. If the hypothesis should be accepted, study computes test size - probability of committing a Type I error, that is, of incorrectly rejecting the null hypothesis when it is true. If hypothesis should be rejected, study computes power - probability of correctly rejecting the null hypothesis when it is false. The test size and power are computed for both statistical tests.

Results

Part 1. Exploratory Analysis

Table 1: First few rows from the data frame after wrangling

movieId	title	year	genres	userId	rating	date
31	Dangerous Minds	2009	Drama	1	2.5	2009-12-14 02:52:24
1029	Dumbo	2009	Animation|Children|Drama|Musical	1	3.0	2009-12-14 02:52:59
1061	Sleepers	2009	Thriller	1	3.0	2009-12-14 02:53:02

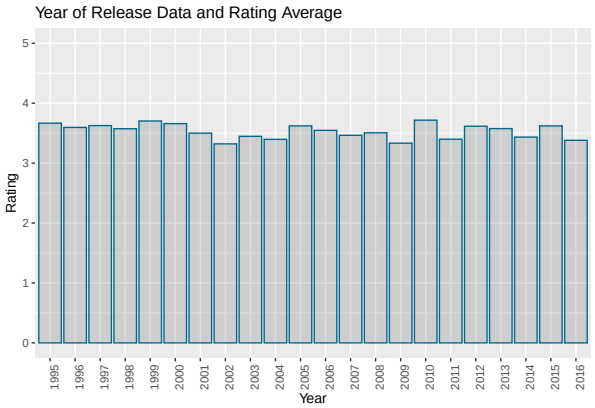
The plots below indicate: the average ratings across release years variate very slightly, users tend to give, in average, a similar rating, the most popular movies to give a rating to are of action, comedy and drama genre, the average rating across genres variate slightly.

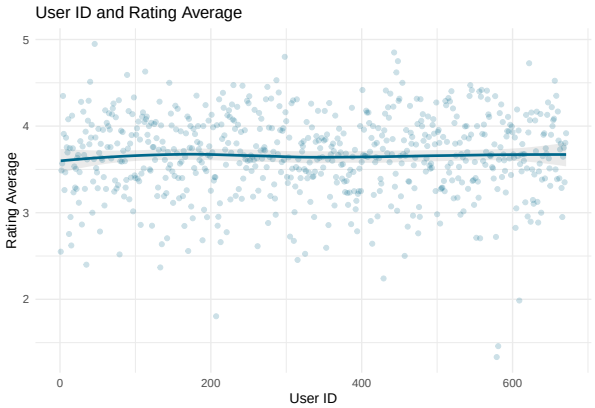
The rating values that can be given are displayed in Table 2. As seen from the "Rating Probabilities" plot, the rating values for the three selected genres are not normally distributed, it seems users tend to give ratings in normal numbers rather than halves, refer to Table 3 for p-values. This plot also indicates some difference between the means of rating values across genres.

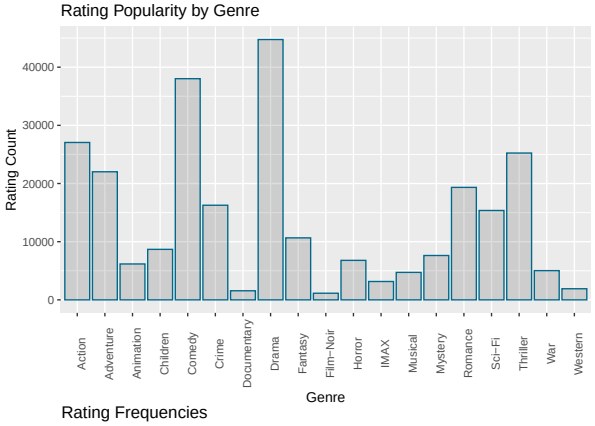
[1](Sigal and Chalmers, 2016)

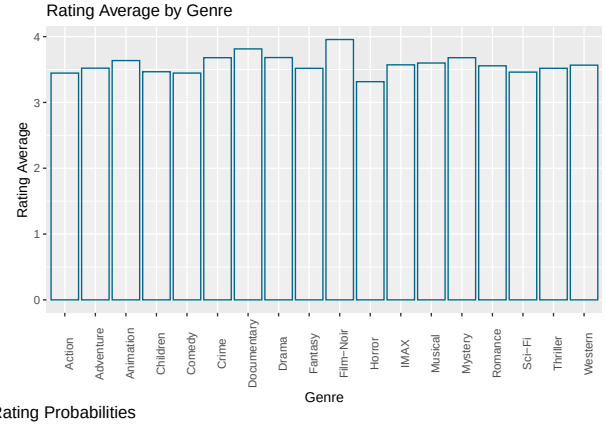
Table 2: Possible rating values

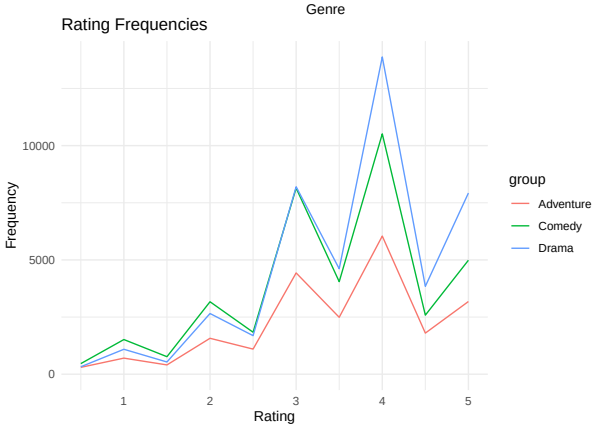
val1	val2	val3	val4	val5	val6	val7	val8	val9	val10
0.5	1	1.5	2	2.5	3	3.5	4	4.5	5











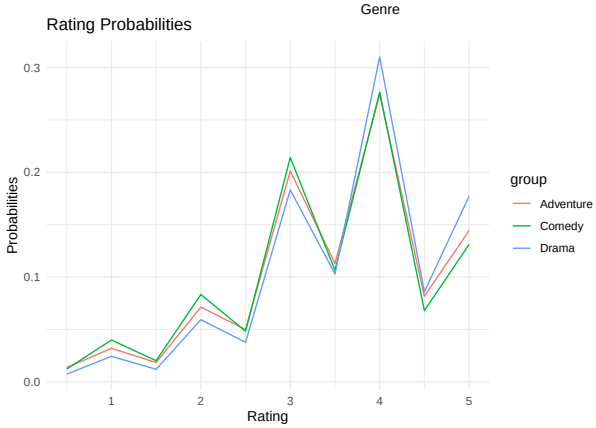


Table 3: Normality results

Adventure	Comedy	Drama
0.000000	0.000000	0.000000

Part 2. Data Simulation

Scenario 1

Simulating data for the first scenario: the probabilities of ratings for each genre are used to simulate a new dataset, see Table 4 for results. Since the Null Hypothesis should be rejected, here, the table displays the power results for both the parametric Anova and non-parametric Kruskal-Wallis statistical tests. Power comes out to be very small, increasing the sample size might help. The effect size is also rather large considering the possible rating values are from 0.5 to 5. Number of simulations is left standard.

Table 4: Scenario 1 results

Power_Anova	Power_Kruskal.Wallis	Mean_Effect_Size	Sample_Size	Simulations	Null_Hypothesis
0.295	0.287	0.265	300	1000	False

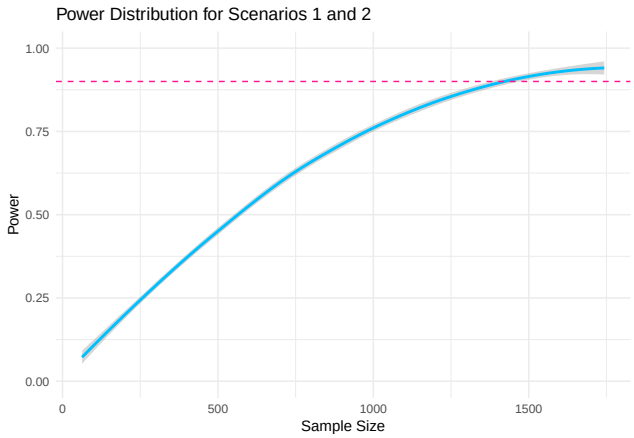
Scenario 2

As expected, for the results to have at least 90% power, sample size needs to increase, refer to Table 5.

Since the data distribution shows non-normality, the Power and Test size distribution plots are based on the non-parametric Kruskal-Wallis test. The Power of 90% is reached at around 1400 samples, see Plot "Power Distribution for Scenarios 1 and 2".

Table 5: Scenario 2 results

Power_Anova	Power_Kruskal.Wallis	Mean_Effect_Size	Sample_Size	Simulations	Null_Hypothesis
0.928	0.921	0.154	1500	1000	False

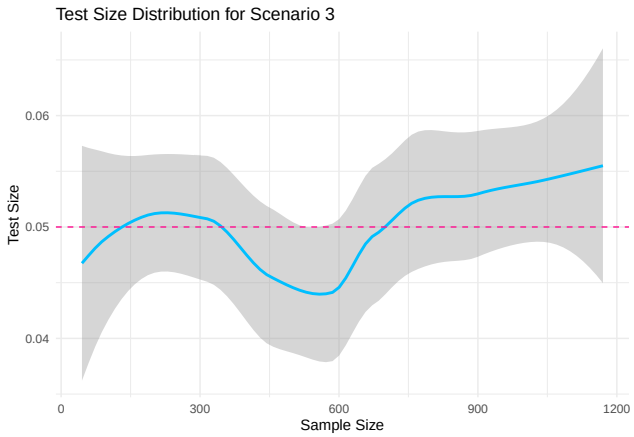


Scenario 3

Simulating data with all the rating values having 0.1 probability across all three genres, same sample size, the Null Hypothesis should be accepted, for results refer to Table 6. The results give good and expected standard test size values. Mean effect size across each simulation has decreased. The standard Test Size of 4-6% is reached, refer to Plot "Test Size Distribution for Scenario 3".

Table 6: Scenario 3 results

Test_Size_Anova	Test_Size_Kruskal.Wallis	Mean_Effect_Size	Sample_Size	Simulations	Null_Hypothesis
0.057	0.059	0.112	1500	1000	True

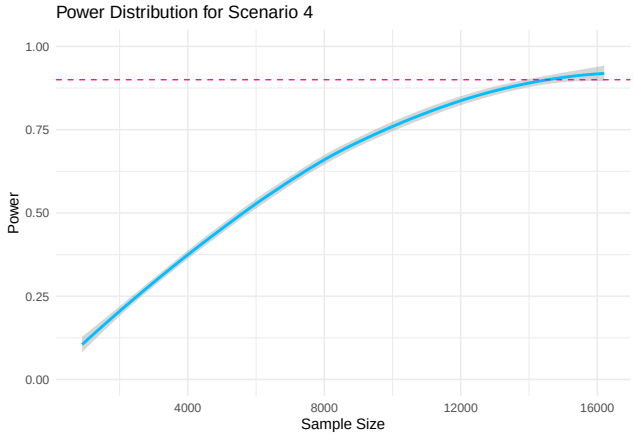


Scenario 4

Simulating data with only one of the genres to have a slightly different distribution, the Null Hypothesis should be rejected, see results in Table 7. Since the effect size has an influence on power or test size, here, the sample size has been deliberately increased. It would take a lot more samples to correctly reject the Null Hypothesis with at least 90% power, refer to Plot "Power Distribution for Scenario 4".

Table 7: Scenario 4 results

Power_Anova	Power_Kruskal.Wallis	Mean_Effect_Size	Sample_Size	Simulations	Null_Hypothesis
0.913	0.913	0.0701	15000	1000	False



Conclusions

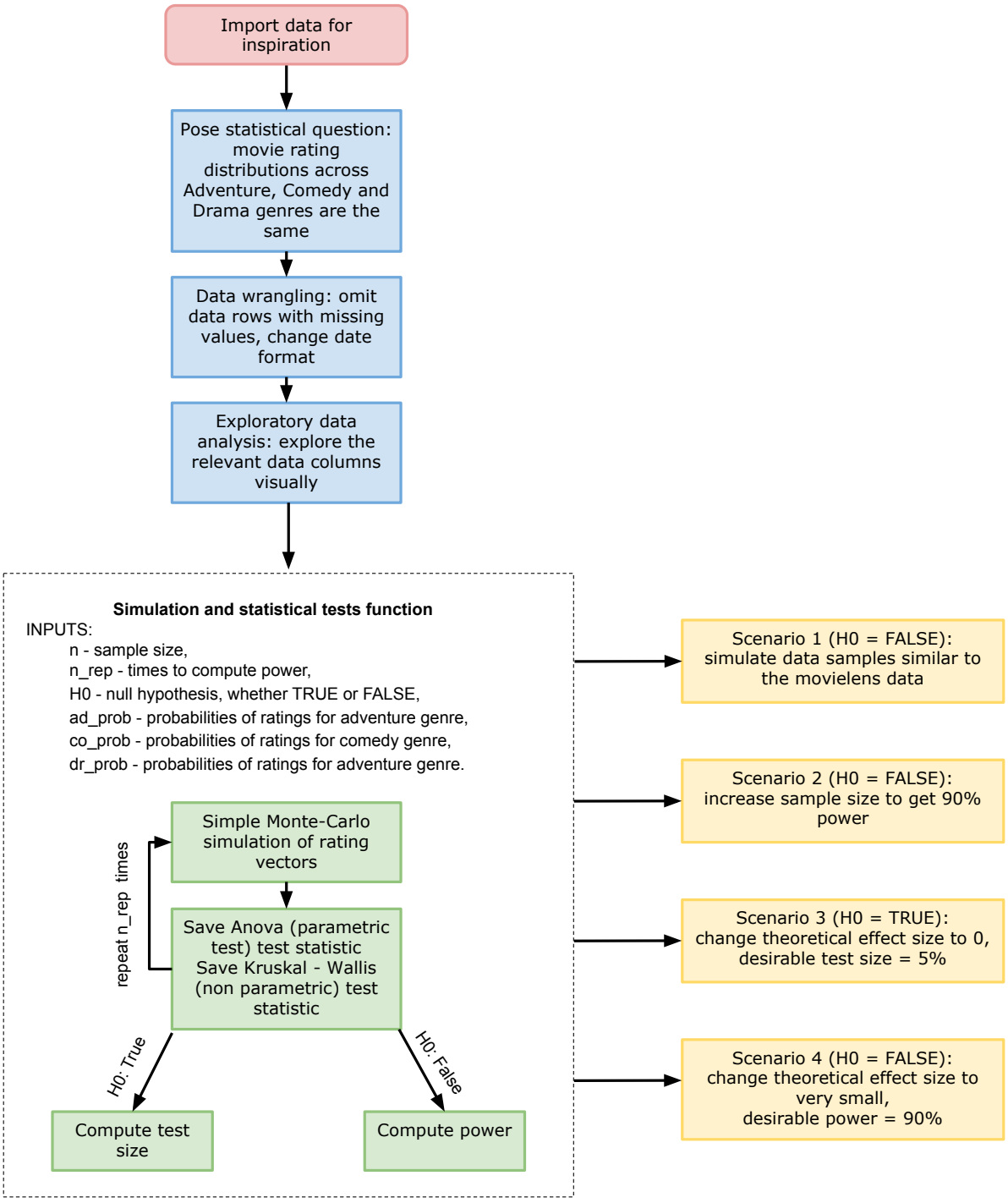
Clearly, the rating means from the original data set "movielens" across Adventure, Comedy and Drama movie genres are statistically different. Simulating new data sets has proven to be beneficial and a good way of determining the sample size needed to detect a particular effect size. To detect a small effect size, a large sample size is needed. The results are as expected: effect size, power or test size and sample size all have an affect on one another.
Anova test might not be the best choice for non normally distributed data, however, the Kruskal-Wallis test accounts for that.

References

1. Matthew J. Sigal & R. Philip Chalmers (2016) Play It Again: Teaching Statistics With Monte Carlo Simulation, Journal of Statistics Education, 24:3, 136-156, DOI: 10.1080/10691898.2016.1246953

Appendices

Code Diagram



Code

Data wrangling and exploratory analysis.

```
# Load libraries
library(dslabs)
library(tidyverse)

# Load data
data("movielens")
data <- movielens

# Check head, present in markdown
head(data)
# Check the structure of the data frame
str(data)
n <- nrow(data)
missing_data <- data[rowSums(is.na(data)) > 0, ]

# Data wrangling
data <- na.omit(data)
# Data with no genres listed deleted
data <- data[!(data$genres == "(no genres listed)"), ]

# Convert timestamp to a date format and create a data column
data <- data %>%
  mutate(date = as.POSIXct(timestamp, origin = "1970-01-01", tz = "UTC"),
         year = format(date, "%Y"))
# Remove timestamp as date column has been created
data <- subset(data, select = -timestamp)

# Exploratory Data Analysis
# Year of movie and rating average
rating_year <- aggregate(data$rating, list(data$year), FUN = mean)
ggplot() +
  geom_col(aes(x = rating_year[, 1], y = rating_year[, 2]),
           alpha = 0.2, colour = "deepskyblue4") +
  ylim(c(0, 5)) +
  labs(title = "Year of Release Data and Rating Average",
       y = "Rating", x = "Year") +
  theme(axis.text.x = element_text(angle = 90))

# User ID and rating average
rating_userID <- aggregate(data$rating, list(data$userId), FUN = mean)
ggplot() +
  geom_smooth(aes(x = rating_userID[, 1], y = rating_userID[, 2]),
              alpha = 0.2, colour = "deepskyblue4") +
  geom_point(aes(x = rating_userID[, 1], y = rating_userID[, 2]),
             alpha = 0.2, colour = "deepskyblue4") +
  labs(title = "User ID and Rating Average",
       y = "Rating Average", x = "User ID") +
  theme_minimal()

# Extract each genre listed in the data frame.
# Regroup data by genres
```

```r
data_genres <- data %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(number = n()) %>%
  arrange(desc(number)) %>%
  as.data.frame()

# Plot most popular movies by genre
ggplot() +
  geom_col(aes(x = data_genres[1:19, 1], y = data_genres[1:19, 2]),
           alpha = 0.2, colour = "deepskyblue4") +
  labs(title = "Rating Popularity by Genre",
       y = "Rating Count", x = "Genre") +
  theme(axis.text.x = element_text(angle = 90))

# Plot average rating by genre
# Create data frame with average rating for each genre
rating_by_genre <- data %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(number = mean(rating)) %>%
  arrange(desc(number)) %>%
  as.data.frame()
# Plot
ggplot() +
  geom_col(aes(x = rating_by_genre[c(1, 3:19), 1], y = rating_by_genre[c(1, 3:19), 2]),
           colour = "deepskyblue4", fill = "white", alpha = 0.2) +
  labs(title = "Rating Average by Genre",
       y = "Rating Average", x = "Genre") +
  theme(axis.text.x = element_text(angle = 90))


# Rating probabilities for Adventure, Comedy and Drama genres
# Adventure ratings
adventure_ratings <- data$rating[str_detect(data$genres, "Adventure")]
# Comedy ratings
comedy_ratings <- data$rating[str_detect(data$genres, "Comedy")]
# Drama ratings
drama_ratings <- data$rating[str_detect(data$genres, "Drama")]
# Store vector of rating values
rating_values <- unique(data$rating) %>%
  sort(decreasing = FALSE)

# Create data frame with all the rating values
acd_ratings <- data.frame(values = c(adventure_ratings,
                                     comedy_ratings,
                                     drama_ratings),
                          group = c(rep("Adventure", length(adventure_ratings)),
                                    rep("Comedy", length(comedy_ratings)),
                                    rep("Drama", length(drama_ratings))))
# Plot overlaid frequencies
ggplot(acd_ratings, aes(x = values, colour = group)) +
  geom_freqpoly(position = "identity", bins = 10) +
```

```
  xlim(c(0.5, 5)) +
  labs(title = "Rating Frequencies",
       y = "Frequency", x = "Rating") +
  theme_minimal()

# Compute initial probabilities for each value
# INPUT:
#   genre_ratings - ratings taken from the data frame.
# OUTPUT:
#   prob_values - probability values.
probabilities <- function(genre_ratings) {
  prob_values <- rep(NA, length(rating_values))
  for (i in rating_values) {
    prob_values <- c(prob_values, sum(genre_ratings == i) / length(genre_ratings))
  }
  return(prob_values[11:20])
}

# Store initial probabilities for the three genres
ad_prob_data <- probabilities(genre_ratings = adventure_ratings)
co_prob_data <- probabilities(genre_ratings = comedy_ratings)
dr_prob_data <- probabilities(genre_ratings = drama_ratings)

# Create data frame with all the rating values and probabilities
acd_ratings_prob <- data.frame(values = c(rating_values,
                                          rating_values,
                                          rating_values),
                               probabilities = c(ad_prob_data,
                                                 co_prob_data,
                                                 dr_prob_data),
                               group = c(rep("Adventure", length(ad_prob_data)),
                                         rep("Comedy", length(ad_prob_data)),
                                         rep("Drama", length(ad_prob_data))))

# Plot overlaid distributions
ggplot(acd_ratings_prob, aes(x = values, y = probabilities, colour = group)) +
  geom_path(position = "identity") +
  labs(title = "Rating Probabilities",
       y = "Probabilities", x = "Rating") +
  theme_minimal()
```

Normality check.

```
# Check normality of rating distributions
# For reproducibility
set.seed(222)
norm_test <- data.frame(Adventure = shapiro.test(sample(adventure_ratings, 5000,
                                                        replace = FALSE))$p.value,
                        Comedy = shapiro.test(sample(comedy_ratings, 5000,
                                                     replace = FALSE))$p.value,
                        Drama = shapiro.test(sample(drama_ratings, 5000,
                                                    replace = FALSE))$p.value)
```

Simulation functions.

```r
# Monte Carlo simulation of new data sets, compute power and
# test size.
# INPUTS:
#  n - sample size,
#  n_rep - times to compute power or test size,
#  H0 - null hypothesis, whether TRUE or FALSE,
#  ad_prob - probabilities of ratings for adventure genre,
#  co_prob - probabilities of ratings for comedy genre,
#  dr_prob - probabilities of ratings for adventure genre.
# OUTPUT:
#  results - data frame containing:
#     Power of Anova Statistic,
#     Test Size from Anova Statistic,
#     Power of Kruskal Wallis Statistic,
#     Test Size from Kruskal Wallis Statistic,
#     Effect Size,
#     Sample Size,
#     Number of Simulations.
my_simulation <- function(n = 1000, n_rep = 1000, H0,
                          ad_prob, co_prob, dr_prob) {
  # Check if the number of samples n and number of simulations n_rep are
  # a single natural number
  if (n <= 0 || n %% 1 != 0 || !is.numeric(n) || length(n) > 1 ||
     !is.vector(n)) {
    stop("Invalid sample size!")
  }
  if (n_rep <= 0 || n_rep %% 1 != 0 || is.na(n_rep) ||
      is.numeric(n_rep) == FALSE || length(n_rep) > 1 || !is.vector(n_rep)) {
    stop("Invalid simulation size!")
  }
  # Check if the null hypothesis value is either TRUE or FALSE
  if (!H0 == TRUE && !H0 == FALSE) {
    stop("Invalid null hypothesis value!")
  }
  # Check if the probability vectors for each genre are of length 10
  if (!is.numeric(ad_prob) || any(is.na(ad_prob)) || !is.vector(ad_prob) ||
      length(ad_prob) != 10 || any(ad_prob >= 1) || any(ad_prob <= 0)) {
    stop("Invalid probabilities for adventure genre!")
  }
  if (!is.numeric(co_prob) || any(is.na(co_prob)) || !is.vector(co_prob) ||
      length(co_prob) != 10 || any(co_prob >= 1) || any(co_prob <= 0)) {
    stop("Invalid probabilities for comedy genre!")
  }
  if (!is.numeric(dr_prob) || any(is.na(dr_prob)) || !is.vector(dr_prob) ||
      length(dr_prob) != 10 || any(dr_prob >= 1) || any(dr_prob <= 0)) {
    stop("Invalid probabilities for drama genre!")
  }

  # To store power, effect size, test size
  power_anova <- power_kruskal <- eff_size <- size_anova <- size_kruskal <-
    aov_statistic <- kr_statistic <- mean_eff_size <- NA
  # To store results
  results <- data.frame(rep(NA, 7))
```

```r
for (i in 1:n_rep) {
# To store samples
samp_ad <- samp_co <- samp_dr <- NA
# T store simulated data
simulated_data <- rep(NA, 3)
# Generate samples from non-normal distribution with discrete values
# Adventure genre
samp_ad <- sample(x = rating_values, n, replace = TRUE, prob = ad_prob)
# Comedy genre
samp_co <- sample(x = rating_values, n, replace = TRUE, prob = co_prob)
# Drama genre
samp_dr <- sample(x = rating_values, n, replace = TRUE, prob = dr_prob)
# Combined vectors to a single data frame
simulated_data <- stack(data.frame(cbind(samp_ad, samp_co, samp_dr)))

# Return the probability of making a correct decision when the null
# hypothesis is false
# Run One-way Anova test on each rating by genre distributions
aov_statistic[i] <- unlist(summary(aov(values ~ ind,
                                       data = simulated_data)))["Pr(>F)1"]
# Run non-parametric Kruskal-Wallis test statistic
kr_statistic[i] <- kruskal.test(values ~ ind, data = simulated_data)["p.value"]

# Effect size
eff_size <- max(mean(samp_ad), mean(samp_co), mean(samp_dr)) -
  min(mean(samp_ad), mean(samp_co), mean(samp_dr))
}

# Compute test size - probability of committing a Type I error, that is, of
# incorrectly rejecting the null hypothesis when it is true
if (H0 == TRUE) {
  # Test size for Anova statistic
  size_anova <- sum(aov_statistic <= 0.05) / length(aov_statistic)
  # Test size for Kruskal-Wallis statistic
  size_kruskal <- sum(kr_statistic <= 0.05) / length(kr_statistic)
  # Mean effect size
  mean_eff_size <- mean(eff_size)
  # Create data frame to return
  results <- data.frame("Test_Size_Anova" = size_anova,
                        "Test_Size_Kruskal-Wallis" = size_kruskal,
                        "Mean_Effect_Size" = mean_eff_size,
                        "Sample_Size" = n * 3,
                        "Simulations" = n_rep,
                        "Null_Hypothesis" = "True")
# Compute power - probability of correctly rejecting the null hypothesis
# when it is false
}
else {
  # Compute power for Anova statistic
  power_anova <- sum(aov_statistic <= 0.05) / length(aov_statistic)
  # Compute power for Kruskal-Wallis statistic
  power_kruskal <- sum(kr_statistic <= 0.05) / length(kr_statistic)
  # Mean effect size
```

```
    mean_eff_size <- mean(eff_size)
    # Create data frame to return
    results <- data.frame("Power_Anova" = power_anova,
                          "Power_Kruskal-Wallis" = power_kruskal,
                          "Mean_Effect_Size" = mean_eff_size,
                          "Sample_Size" = n * 3,
                          "Simulations" = n_rep,
                          "Null_Hypothesis" = "False")
  }
  # I wouldn't use return command normally in this place, since the Tidyverse
  # Style Guide advice against it, unless I wanted the return earlier in the
  # function. However, I have heard this sometimes gets penalised at the
  # university, hence, the return command at the end of the function.
  return(results)
}



# Monte Carlo simulation of new data sets, compute power and
# test size to display distributions.
# Selected power is from Kruskal-Wallis statistic, since it is more
# robuts against non-normal distributions.
# INPUTS:
#  n - sample size,
#  n_rep - times to compute power or test size,
#  H0 - null hypothesis, whether TRUE or FALSE,
#  ad_prob - probabilities of ratings for adventure genre,
#  co_prob - probabilities of ratings for comedy genre,
#  dr_prob - probabilities of ratings for adventure genre.
# OUTPUT:
#  results - containing either:
#     Power of Kruskal Wallis Statistic,
#     Test Size from Kruskal Wallis Statistic.
my_simulation_distribution <- function(n = 1000, n_rep = 1000, H0,
                                       ad_prob, co_prob, dr_prob) {
  # Check if the number of samples n and number of simulations n_rep are
  # a single natural number
  if (n <= 0 || n %% 1 != 0 || !is.numeric(n) || length(n) > 1 ||
      !is.vector(n)) {
    stop("Invalid sample size!")
  }
  if (n_rep <= 0 || n_rep %% 1 != 0 || is.na(n_rep) ||
      is.numeric(n_rep) == FALSE || length(n_rep) > 1 || !is.vector(n_rep)) {
    stop("Invalid simulation size!")
  }
  # Check if the null hypothesis value is either TRUE or FALSE
  if (!H0 == TRUE && !H0 == FALSE) {
    stop("Invalid null hypothesis value!")
  }
  # Check if the probability vectors for each genre are of length 10
  if (!is.numeric(ad_prob) || any(is.na(ad_prob)) || !is.vector(ad_prob) ||
      length(ad_prob) != 10 || any(ad_prob >= 1) || any(ad_prob <= 0)) {
    stop("Invalid probabilities for adventure genre!")
```

```
  }
  if (!is.numeric(co_prob) || any(is.na(co_prob)) || !is.vector(co_prob) ||
      length(co_prob) != 10 || any(co_prob >= 1) || any(co_prob <= 0)) {
    stop("Invalid probabilities for comedy genre!")
  }
  if (!is.numeric(dr_prob) || any(is.na(dr_prob)) || !is.vector(dr_prob) ||
      length(dr_prob) != 10 || any(dr_prob >= 1) || any(dr_prob <= 0)) {
    stop("Invalid probabilities for drama genre!")
  }

  # To store power, effect size, test size
  power_kruskal <- size_kruskal <- kr_statistic <- NA
  # To store results
  results <- NA

  for (i in 1:n_rep) {
    # To store samples
    samp_ad <- samp_co <- samp_dr <- NA
    # T store simulated data
    simulated_data <- rep(NA, 3)
    # Generate samples from non-normal distribution with discrete values
    # Adventure genre
    samp_ad <- sample(x = rating_values, n, replace = TRUE, prob = ad_prob)
    # Comedy genre
    samp_co <- sample(x = rating_values, n, replace = TRUE, prob = co_prob)
    # Drama genre
    samp_dr <- sample(x = rating_values, n, replace = TRUE, prob = dr_prob)
    # Combined vectors to a single data frame
    simulated_data <- stack(data.frame(cbind(samp_ad, samp_co, samp_dr)))

    # Return the probability of making a correct decision when the null
    # hypothesis is false
    # Run non-parametric Kruskal-Wallis test statistic
    kr_statistic[i] <- kruskal.test(values ~ ind, data = simulated_data)["p.value"]
  }

  # Compute test size - probability of committing a Type I error, that is, of
  # incorrectly rejecting the null hypothesis when it is true
  if (H0 == TRUE) {
    # Test size for Kruskal-Wallis statistic
    size_kruskal <- sum(kr_statistic <= 0.05) / length(kr_statistic)
    results <- size_kruskal
  }
  else {
    # Compute power for Kruskal-Wallis statistic
    power_kruskal <- sum(kr_statistic <= 0.05) / length(kr_statistic)
    results <- power_kruskal
  }
  return(results)
}
```

Compute scenarios, plot power and test size distributions.

```r
# Function to produce multiple power or test size values to then display
# their distribution in accordance with sample size
# INPUT:
#   samp_seq - vector with values to use as sample size for each power or
#              test size computation,
#   n_rep - times to compute power or test size,
#   H0 - null hypothesis, whether TRUE or FALSE,
#   ad_prob - probabilities of ratings for adventure genre,
#   co_prob - probabilities of ratings for comedy genre,
#   dr_prob - probabilities of ratings for adventure genre.
# OUTPUT:
#   outcome_values - either power or test size values from sample size.
outcome_distribution <- function(samp_seq, n_rep = 1000, H0,
                                 ad_prob, co_prob, dr_prob) {
  # To store power values
  outcome_values <- NA
  # Swap samp_seq to n
  n <- samp_seq
  # Loop with sequence values
  for (i in samp_seq) {
    outcome_values <- c(outcome_values,
                        my_simulation_distribution(n = i, n_rep, H0, ad_prob,
                                                   co_prob, dr_prob))
  }
  return(outcome_values[-1])
}




## Hypothesis:
#     Rating average across Adventure, Comedy and Drama genres are the same.
#     H0: mu1 = mu2 = ... = mun
#     H1: mu1 ! = mu2 ! = ... ! = mun
# Hypothesis should be rejected.
# Scenario 1. samples with same probabilities as from the original data     ###
# For reproducibility.
set.seed(999)
sim1 <- my_simulation(n = 100, H0 = FALSE,
              ad_prob = ad_prob_data,
              co_prob = co_prob_data,
              dr_prob = dr_prob_data)



# Scenario 2.                                                               ###
# Increasing the sample size of ratings for each genre, increases power
# For reproducibility.
set.seed(777)
sim2 <- my_simulation(n = 500, H0 = FALSE,
              ad_prob = ad_prob_data,
              co_prob = co_prob_data,
              dr_prob = dr_prob_data)
```

```r
# For reproducibility
set.seed(111)
# Store values of power from sample size to then plot
scenario1_2 <- outcome_distribution(samp_seq = seq(1, 600, 20), H0 = FALSE,
                                    ad_prob = ad_prob_data,
                                    co_prob = co_prob_data,
                                    dr_prob = dr_prob_data)
ggplot() +
  geom_smooth(aes(x = seq(1, 600, 20) * 3, y = scenario1_2), colour = "deepskyblue") +
  geom_hline(yintercept = 0.9, linetype = "dashed",
             color = "deeppink", size = 0.5) +
  ylim(c(0, 1)) +
  labs(title = "Power Distribution for Scenarios 1 and 2",
       y = "Power", x = "Sample Size") +
  theme_minimal()




# Scenario 3.                                                              ###
# The effect size will be changed when the means of each genre distribution are
# more or less different. In this case, probabilities of each rating value
# are made more similar.
# For reproducibility.
set.seed(555)
sim3 <- my_simulation(n = 500, H0 = TRUE,
              ad_prob = rep(0.1, 10),
              co_prob = rep(0.1, 10),
              dr_prob = rep(0.1, 10))

# For reproducibility
set.seed(111)
scenario3 <- outcome_distribution(samp_seq = seq(15, 400, 15), H0 = TRUE,
                                  ad_prob = rep(0.1, 10),
                                  co_prob = rep(0.1, 10),
                                  dr_prob = rep(0.1, 10))

ggplot() +
  geom_smooth(aes(x = seq(15, 400, 15) * 3, y = scenario3), colour = "deepskyblue") +
  geom_hline(yintercept = 0.9, linetype = "dashed",
             color = "deeppink", size = 0.5) +
  ylim(c(0, 1)) +
  labs(title = "Test Size Distribution for Scenario 3",
       y = "Test Size", x = "Sample Size") +
  theme_minimal()




# Scenario 4.                                                              ###
# Probabilities are slightly different.
set.seed(333)
sim4 <- my_simulation(n = 5000, H0 = FALSE,
```

```
              ad_prob = rep(0.1, 10),
              co_prob = c(0.08, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.12),
              dr_prob = rep(0.1, 10))

# For reproducibility
set.seed(111)
scenario4 <- outcome_distribution(samp_seq = seq(1, 5500, 300), H0 = FALSE,
                                  ad_prob = rep(0.1, 10),
                                  co_prob = c(0.08, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
                                              0.1, 0.1, 0.12),
                                  dr_prob = rep(0.1, 10))

ggplot() +
  geom_smooth(aes(x = seq(1, 5500, 300) * 3, y = scenario4), colour = "deepskyblue") +
  geom_hline(yintercept = 0.9, linetype = "dashed",
             color = "deeppink", size = 0.5) +
  ylim(c(0, 1)) +
  labs(title = "Power Distribution for Scenario 4",
       y = "Power", x = "Sample Size") +
  theme_minimal()
```