Data Wrangling (Data Preprocessing)

Code

Practical assessment 2

Akshay Kumar

31st May,2024

Setup

Hide

```
# Load the necessary packages required to reproduce the report. For example:

library(kableExtra)
library(magrittr)
library(dplyr)
library(tidyr)
library(stringr)
library(ggplot2)
```

Student names, numbers and percentage of contributions

Group information

Student name	Student number	Percentage of contribution
Akshay Kumar	S4026599	100%

Executive Summary

Based on your project and the provided content, here is an executive summary in points:

1. **Project Overview:**
   - Focus on data wrangling and preprocessing of an anime dataset from two sources.
   - Aim to merge datasets and enhance data analysis.
2. **Data Import and Merging:**
   - Merged two datasets on common columns `'title'` and `'Anime_Name'`.
3. **Initial Data Cleaning:**
   - Removed unnecessary columns like `'img_url'` and `'link'`.
   - Standardized the `'aired'` date and converted `'episodes'`, `'score'`, `'members'`, and `'popularity'` to numeric types.
   - Converted `'genre'` to a factor.
4. **Data Structuring and Transformation:**
   - Split `'Anime_Episodes'` into `'Type'` and `'Ep_Count'`.
   - Divided `'Anime_Air_Years'` into `'Airing_Start'` and `'Airing_End'`.
5. **Feature Engineering:**
   - Created `'Episode_Category'` to classify episodes.
   - Developed `'Genre_Count'` to tally genres.

6. **Data Validation:**
    - Checked for missing values, inconsistencies, and outliers using the IQR method.
7. **Normalization and Log Transformation:**
    - Normalized numeric columns to a 0-1 range.
    - Applied log transformation to address skewed data.
    - Developed `'Score_Popularity_Index'` to integrate `'Score'` and `'Popularity'`.
8. **Scan I (Data Checking):**
    - Identified missing values, inconsistencies, and outliers in the dataset.
    - Removed rows with missing values and validated the cleaned data.
9. **Scan II (Outlier Detection):**
    - Detected outliers using the IQR method.
    - Created improved boxplots for visualizing outliers and distributions.
10. **Transform (Data Transformation):**
    - Applied Min-Max normalization and log transformation to relevant numeric columns.
    - Reorganized the dataset for better accessibility and analysis.
    - Visualized the effects of transformations using histograms and boxplots.
11. **Conclusion:**
    - Min-Max normalization ensured uniform scaling.
    - Log transformation reduced skewness.

Data

Data Sources
The datasets used for this project are sourced from Kaggle:

Dataset1 - MyAnimeList Dataset (March 2020) - **URL:** MyAnimeList Dataset (https://www.kaggle.com/datasets/marlesson/myanimelist-dataset-animes-profiles-reviews?select=animes.csv) - **Variables:** - `'uid'` : Unique identifier for each anime. - `'title'` : The title of the anime. - `'synopsis'` : A brief summary of the anime's plot. - `'genre'` : The genre(s) of the anime. - `'aired'` : Date range during which the anime aired. - `'episodes'` : The number of episodes in the anime. - `'members'` : Number of members who have added the anime to their list. - `'popularity'` : Popularity rank of the anime. - `'ranked'` : Rank of the anime based on its score. - `'score'` : Average score of the anime. - `'img_url'` : URL to an image of the anime. - `'link'` : URL to the anime's MyAnimeList page.

Dataset2 - Top 10,000 Anime Movies, OVAs, and TV Shows (April 2021) - **URL:** Top 10,000 Anime Movies, OVAs, and TV Shows (https://www.kaggle.com/datasets/thomaskonstantin/top-10000-anime-movies-ovas-and-tvshows) - **Variables:** - `'Anime_Name'` : The name of the anime. - `'Anime_Episodes'` : Information about the type and number of episodes. - `'Anime_Air_Years'` : The years during which the anime aired. - `'Anime_Rating'` : The rating of the anime (e.g., PG-13, R). - `'Synopsis'` : A brief synopsis of the anime.

Import and Merge Process
The datasets were imported from CSV files and merged to form a comprehensive dataset. The merge was performed using an inner join on the `'title'` column from Dataset1 and the `'Anime_Name'` column from Dataset2, ensuring only the common entries from both datasets were included.

Hide

```
# Import the data, provide your R codes here.
Dataset1<- read.csv("C://Users//Lenovo//Desktop//animes.csv")
#head(Dataset1)
#colnames(Dataset1)

Dataset2<- read.csv("C://Users//Lenovo//Desktop//Anime_Top10000.csv")
```

```
Warning: embedded nul(s) found in input
```

Hide

```
#head(Dataset2)
#colnames(Dataset2)


# Perform an inner merge on the columns 'title' and 'Anime_Name'
merged_dataset <- merge(Dataset1, Dataset2, by.x = "title", by.y = "Anime_Name", all = FALSE)
head(merged_dataset)
```

	title <chr>	uid <int>
1	"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	7669
2	"Bungaku Shoujo" Memoire	8481
3	"Bungaku Shoujo" Movie	6408
4	"Eiji"	6076
5	"Eiyuu" Kaitai	33363
6	"Eiyuu" Kaitai	33363

6 rows | 1-3 of 16 columns

Hide

```
#sapply(merged_dataset, class)
```

Understand

Hide

```
# This is the R chunk for the Understand Section

colnames(merged_dataset)
```

```
 [1] "title"          "uid"            "synopsis"       "genre"
 [5] "aired"          "episodes"       "members"        "popularity"
 [9] "ranked"         "score"          "img_url"        "link"
[13] "Anime_Episodes" "Anime_Air_Years" "Anime_Rating"   "Synopsis"
```

Hide

```
# Remove unnecessary columns
merged_dataset <- merged_dataset %>%
  select(-img_url, -link)

# Convert 'aired' to Date type (handle cases with ranges appropriately)
merged_dataset <- merged_dataset %>%
  mutate(aired = if_else(str_detect(aired, "to"),
                         as.Date(str_extract(aired, "[A-Za-z]{3} \\d{1,2}, \\d{4}"), format
="%b %d, %Y"),
                         as.Date(aired, format="%b %d, %Y")))

# Convert 'episodes' to numeric after removing non-numeric characters
merged_dataset$episodes <- as.numeric(gsub("[^0-9]", "", merged_dataset$episodes))

# Convert 'genre' to factor
merged_dataset$genre <- as.factor(merged_dataset$genre)

# Ensure 'score', 'members', and 'popularity' are numeric
merged_dataset$score <- as.numeric(merged_dataset$score)
merged_dataset$members <- as.numeric(merged_dataset$members)
merged_dataset$popularity <- as.numeric(merged_dataset$popularity)

# Check the structure of the merged dataset
# str(merged_dataset)

# Summary of the merged dataset to understand the data distribution
summary(merged_dataset)
```

```
   title               uid          synopsis
Length:10897       Min.   :    1   Length:10897
Class :character   1st Qu.: 3035   Class :character
Mode  :character   Median :11083   Mode  :character
                   Mean   :16607
                   3rd Qu.:32005
                   Max.   :40956

                        genre          aired               episodes
['Music']                  : 375   Min.   :1917-06-30   Min.   :   1.00
['Comedy']                 : 280   1st Qu.:2002-01-07   1st Qu.:   1.00
['Slice of Life', 'Comedy']:  99   Median :2010-09-20   Median :   3.00
['Fantasy']                :  86   Mean   :2007-02-12   Mean   :  14.25
['Drama']                  :  80   3rd Qu.:2015-10-03   3rd Qu.:  13.00
['Slice of Life']          :  71   Max.   :2020-08-07   Max.   :1818.00
(Other)                    :9906   NA's   :547          NA's   :271
   members          popularity       ranked          score       Anime_Episodes
Min.   :     49   Min.   :    1   Min.   :    1   Min.   :3.170   Length:10897
1st Qu.:   1688   1st Qu.: 2016   1st Qu.: 1596   1st Qu.:6.270   Class :character
Median :   8304   Median : 4692   Median : 4336   Median :6.860   Mode  :character
Mean   :  59279   Mean   : 5321   Mean   : 4516   Mean   :6.921
3rd Qu.:  45836   3rd Qu.: 8374   3rd Qu.: 7226   3rd Qu.:7.540
Max.   :1871043   Max.   :16338   Max.   :14658   Max.   :9.230
                                  NA's   :282     NA's   :276
Anime_Air_Years     Anime_Rating     Synopsis
Length:10897       Min.   :5.430   Length:10897
Class :character   1st Qu.:6.180   Class :character
Mode  :character   Median :6.770   Mode  :character
                   Mean   :6.847
                   3rd Qu.:7.470
                   Max.   :9.180
```

Hide

```
sapply(merged_dataset, class) # Provides list of variables with their current data types
```

```
         title            uid       synopsis          genre          aired
   "character"      "integer"    "character"       "factor"         "Date"
      episodes         members     popularity         ranked          score
     "numeric"      "numeric"      "numeric"      "numeric"      "numeric"
Anime_Episodes Anime_Air_Years   Anime_Rating       Synopsis
   "character"    "character"      "numeric"    "character"
```

Initial Exploration

The script first retrieves and displays the names of all columns to understand the dataset's structure and plan data processing steps effectively.

Column Removal

Unnecessary columns (`img_url` and `link`) are removed to streamline the dataset and focus on relevant features.

Data Type Conversion

Several columns are converted to appropriate data types:

- **Date Conversion:** The `aired` column is standardized by extracting start dates and converting them to proper Date types, ensuring consistency.
- **Numeric Conversion:** Columns such as `episodes`, `score`, `members`, and `popularity` are converted to numeric types, including cleaning `episodes` by removing non-numeric characters. This ensures accurate computations and analyses.
- **Factor Conversion:** The `genre` column is converted to a factor type to handle categorical data effectively.

Data Validation

The script performs several checks to validate the processed data:

- **Structure Check:** Displays the dataset structure to verify data types and preview values, ensuring correct formatting.
- **Summary Statistics:** Generates descriptive statistics for the dataset, providing insights into the distribution and central tendencies.
- **Data Type Verification:** Lists the current data types of all variables, ensuring they are correctly converted to the desired types.

These steps confirm the dataset is clean, consistent, and ready for further analysis.

Tidy & Manipulate Data I

Hide

```
# This is the R chunk for the Tidy & Manipulate Data I

# Step 1: Separate 'Anime_Episodes' into 'Type' and 'Ep_Count'
merged_dataset <- merged_dataset %>%
  separate(Anime_Episodes, into = c("Type", "Ep_Count"), sep = " \\(", fill = "right") %>%
  mutate(Ep_Count = str_remove(Ep_Count, " eps\\)"))

# Step 2: Separate 'Ep_Count' to remove any non-numeric characters
merged_dataset <- merged_dataset %>%
  separate(Ep_Count, into = c("Ep_Count", "Delete"), sep = "e", fill = "right") %>%
  select(-Delete) %>%
  mutate(Ep_Count = as.numeric(Ep_Count))
```

```
Warning: There was 1 warning in `mutate()`.
ℹ In argument: `Ep_Count = as.numeric(Ep_Count)`.
Caused by warning:
! NAs introduced by coercion
```

Hide

```
# Suppressing warnings for separating 'Anime_Air_Years'
merged_dataset <- suppressWarnings(
  merged_dataset %>%
    separate(Anime_Air_Years, into = c("Airing_Start", "Airing_End"), sep = " - ", fill = "ri
ght")
)

# Suppressing warnings for mutating `Ep_Count`
merged_dataset <- suppressWarnings(
  merged_dataset %>%
    mutate(Ep_Count = as.numeric(Ep_Count))
)

# Step 4: Trim whitespace from 'Type' and convert to factor
merged_dataset$Type <- str_trim(merged_dataset$Type)
merged_dataset$Type <- factor(merged_dataset$Type, levels = c("TV", "Movie", "OVA", "ONA", "S
pecial", "Music"))


#Step 5: Renaming the merged dataset variables
colnames(merged_dataset) <- c("Title", "UID", "SynopsisX", "Genre", "Airing_Startx", "Ep_Coun
tX", "Members", "Popularity", "Ranked", "Score", "Type", "Ep_countY", "Airing_Start", "Airing
_End", "Anime_Rating", "SynopsisY")

#Step 6: Removing Repeating and Un-important Variables
merged_dataset <- merged_dataset %>% select(-c("SynopsisY","Ep_countY"))

# Step 7: Display the cleaned data
head(merged_dataset)
```

	Title <chr>	UID <int>
1	"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	7669
2	"Bungaku Shoujo" Memoire	8481
3	"Bungaku Shoujo" Movie	6408
4	"Eiji"	6076
5	"Eiyuu" Kaitai	33363
6	"Eiyuu" Kaitai	33363

6 rows | 1-3 of 14 columns

Show

Detailed Steps and Justification

1. **Separate ‘Anime_Episodes’ into ‘Type’ and ’Ep_Count:**
    - Splits the ‘Anime_Episodes’ column into ‘Type’ and ‘Ep_Count’ to isolate the type of anime and the number of episodes. This makes the data more structured and easier to analyze.
2. **Remove Non-numeric Characters from ’Ep_Count:**

- Ensures that the ‘Ep_Count’ column contains only numeric values by removing any extraneous text. This allows for accurate numerical operations and analysis on the episode count.

3. **Suppress Warnings for Separating ’Anime_Air_Years:**
   - Splits the ‘Anime_Air_Years’ column into ‘Airing_Start’ and ‘Airing_End’, handling cases where the end year might be missing. Suppressing warnings ensures that the code runs smoothly even if some rows don’t have both start and end dates.
4. **Suppress Warnings for Mutating ’Ep_Count:**
   - Converts the ‘Ep_Count’ column to numeric type, ensuring consistent data type for episode count. Suppressing warnings helps avoid interruptions due to conversion issues.
5. **Trim Whitespace from ‘Type’ and Convert to Factor:**
   - Cleans the ‘Type’ column by trimming any leading or trailing whitespace and converts it to a factor. This standardizes the data and prepares it for categorical analysis.
6. **Renaming Columns for Clarity:**
   - Renames columns in the dataset to more descriptive names, improving readability and understanding of the data.
7. **Removing Unnecessary Columns:**
   - Deletes redundant or irrelevant columns, simplifying the dataset and focusing on important variables. This step streamlines the data, making it easier to work with.
8. **Displaying Cleaned Data:**
   - Shows a preview of the cleaned dataset to verify that all transformations were applied correctly and the data is in the desired tidy format.

These steps collectively ensure the dataset is reshaped and cleaned according to tidy data principles, making it suitable for further analysis.

Tidy & Manipulate Data II

Hide

```
# Create 'Episode_Category' based on 'Ep_Count'
merged_dataset <- merged_dataset %>%
  mutate(Episode_Category = case_when(
    Ep_CountX <= 12 ~ "Short",
    Ep_CountX <= 24 ~ "Medium",
    TRUE ~ "Long"
  ))

# Count the number of genres listed for each anime
merged_dataset <- merged_dataset %>%
  mutate(Genre_Count = str_count(Genre, ",") + 1)



# Display the first few rows of the cleaned dataset
head(merged_dataset)
```

	Title <chr>	UID <int>
1	"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	7669
2	"Bungaku Shoujo" Memoire	8481
3	"Bungaku Shoujo" Movie	6408

	Title <chr>	UID <int>
4	"Eiji"	6076
5	"Eiyuu" Kaitai	33363
6	"Eiyuu" Kaitai	33363

6 rows | 1-3 of 16 columns

Explanation of the Code Above

1. **Creating 'Episode_Category' Variable**
   - Categorizes the anime based on the number of episodes (`Ep_CountX`).
   - Labels categories as "Short", "Medium", and "Long" based on episode count.
2. **Creating 'Genre_Count' Variable**
   - Counts the number of genres listed for each anime in the `Genre` column.
   - Utilizes comma counting to determine the number of genres.

Scan I

Hide

```
# Check for missing values in each column
missing_values <- sapply(merged_dataset, function(x) sum(is.na(x)))
missing_values_df <- data.frame(Column = names(missing_values), Missing_Values = missing_valu
es)
print(missing_values_df)
```

	Column <chr>	Missing_Values <int>
Title	Title	0
UID	UID	0
SynopsisX	SynopsisX	0
Genre	Genre	0
Airing_Startx	Airing_Startx	547
Ep_CountX	Ep_CountX	271
Members	Members	0
Popularity	Popularity	0
Ranked	Ranked	282
Score	Score	276

1-10 of 16 rows Previous **1** 2 Next

Hide

```
# Check for inconsistencies and obvious errors
# 1. Scores should be within a valid range (0 to 10)
score_outliers <- merged_dataset %>% filter(Score < 0 | Score > 10)
if (nrow(score_outliers) > 0) {
  print("Score outliers:")
  print(score_outliers)
} else {
  print("No score outliers found.")
}
```

```
[1] "No score outliers found."
```

Hide

```
# 2. Members, Popularity, and Ranked should be non-negative
members_outliers <- merged_dataset %>% filter(Members < 0)
if (nrow(members_outliers) > 0) {
  print("Members outliers:")
  print(members_outliers)
} else {
  print("No members outliers found.")
}
```

```
[1] "No members outliers found."
```

Hide

```
popularity_outliers <- merged_dataset %>% filter(Popularity < 0)
if (nrow(popularity_outliers) > 0) {
  print("Popularity outliers:")
  print(popularity_outliers)
} else {
  print("No popularity outliers found.")
}
```

```
[1] "No popularity outliers found."
```

Hide

```
ranked_outliers <- merged_dataset %>% filter(Ranked < 0)
if (nrow(ranked_outliers) > 0) {
  print("Ranked outliers:")
  print(ranked_outliers)
} else {
  print("No ranked outliers found.")
}
```

```
[1] "No ranked outliers found."
```

Hide

```
# 3. Check for inconsistencies in date fields (Airing_Start and Airing_End)
date_inconsistencies <- merged_dataset %>%
  filter(!is.na(Airing_Start) & !is.na(Airing_End) & Airing_Start > Airing_End)

if (nrow(date_inconsistencies) > 0) {
  print("Date inconsistencies (Airing_Start later than Airing_End):")
  print(date_inconsistencies)
} else {
  print("No date inconsistencies found.")
}
```

```
[1] "Date inconsistencies (Airing_Start later than Airing_End):"
```

Title <chr>	UID <int>
"Bungaku Shoujo" Memoire	8481
.hack//Liminality	299
.hack//Quantum	9332
.hack//Quantum: Sore ike! Bokura no Chimuchimu-chan!!	10390
_Summer	1692
009-1	1583
100% Pascal-sensei	35447
11eyes	6682
12-sai. 2nd Season	31319
12-sai.: Chicchana Mune no Tokimeki 2nd Season	33419

1-10 of 2,265 rows | 1-2 of 16 columns Previous **1** 2 3 4 5 6 … 100 Next

Hide

```
# Remove rows with missing values
cleaned_dataset <- merged_dataset %>% drop_na()

# Check for missing values after removing rows with missing values
missing_values_after_removal <- sapply(cleaned_dataset, function(x) sum(is.na(x)))
missing_values_after_removal_df <- data.frame(Column = names(missing_values_after_removal), M
issing_Values = missing_values_after_removal)
print(missing_values_after_removal_df)
```

	Column <chr>	Missing_Values <int>
Title	Title	0
UID	UID	0
SynopsisX	SynopsisX	0

	Column <chr>	Missing_Values <int>
Genre	Genre	0
Airing_Startx	Airing_Startx	0
Ep_CountX	Ep_CountX	0
Members	Members	0
Popularity	Popularity	0
Ranked	Ranked	0
Score	Score	0

1-10 of 16 rows Previous **1** 2 Next

Hide

```
# Display the first few rows of the cleaned dataset
print(head(cleaned_dataset))
```

	Title <chr>	UID <int>
1	"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	7669
2	"Bungaku Shoujo" Memoire	8481
3	"Bungaku Shoujo" Movie	6408
4	"Eiji"	6076
5	"Eiyuu" Kaitai	33363
6	"Eiyuu" Kaitai	33363

6 rows | 1-3 of 16 columns

Show

Explanation of the Code

Check for Missing Values in Each Column

- **Purpose**: Identify columns with missing values.
- **Method**:
  - Uses the `sapply` function to apply the `is.na` function across all columns, summing the missing values.
  - Results are stored in a data frame (`missing_values_df`) and printed to provide a clear overview of missing values in the dataset.

Check for Inconsistencies and Obvious Errors

- **Scores**:
  - **Purpose**: Ensure scores are within the valid range (0 to 10).
  - **Method**: Filters rows where `Score` values are less than 0 or greater than 10.

    - **Output**: Prints these outliers or confirms if none are found, ensuring all scores are within the expected range.
- **Members, Popularity, and Ranked**:
    - **Purpose**: Ensure these values are non-negative.
    - **Method**: Filters rows where `Members`, `Popularity`, or `Ranked` values are negative.
    - **Output**: Prints any outliers found or confirms if all values are valid.

Check for Inconsistencies in Date Fields

- **Purpose**: Ensure `Airing_Start` dates are not later than `Airing_End` dates.
- **Method**: Filters rows where `Airing_Start` is greater than `Airing_End` and both are not `NA`.
- **Output**: Prints these inconsistencies or confirms if none are found, ensuring logical consistency in date values.

Remove Rows with Missing Values

- **Purpose**: Clean the dataset by removing rows containing any missing values.
- **Method**: Uses the `drop_na` function to remove such rows.
- **Output**: Results in a cleaned dataset (`cleaned_dataset`).

Check for Missing Values After Removal

- **Purpose**: Verify that no missing values remain in the cleaned dataset.
- **Output**: Results are stored in a new data frame (`missing_values_after_removal_df`) and printed to confirm that the cleaned dataset has no missing values.

Scan II

Hide

```
# Function to detect outliers using IQR
detect_outliers <- function(x) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  outliers <- x < lower_bound | x > upper_bound
  return(outliers)
}

# Detect outliers in numeric columns
numeric_columns <- cleaned_dataset %>% select(where(is.numeric))
outliers <- sapply(numeric_columns, detect_outliers)

# Ensure outliers list is in the correct format
outliers_df <- as.data.frame(outliers)

# Summary of outliers
outlier_summary <- data.frame(Column = colnames(outliers_df),
                              Outliers = colSums(outliers_df))
print(outlier_summary)
```

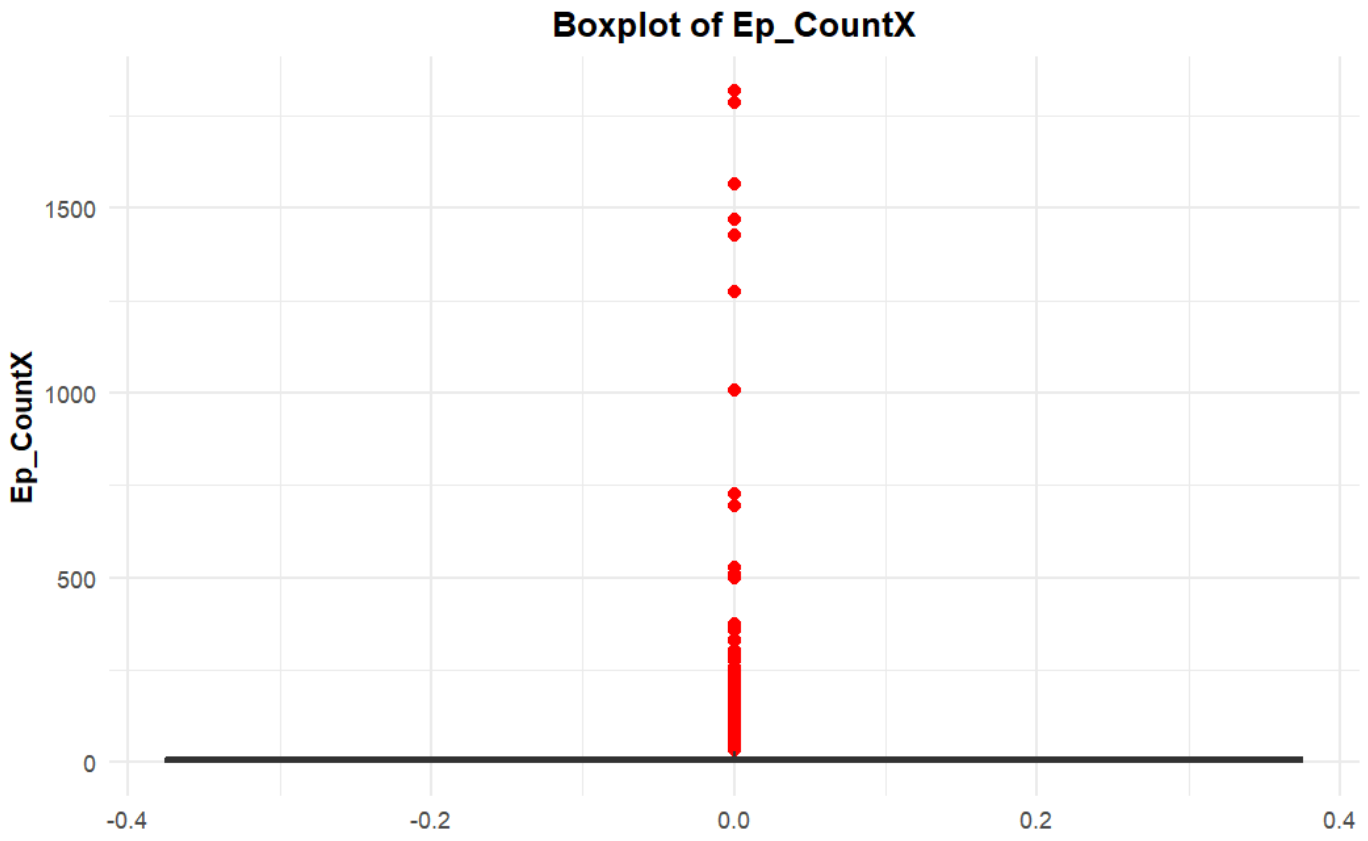
	Column <chr>	Outliers <dbl>
UID	UID	0
Ep_CountX	Ep_CountX	1039
Members	Members	1389
Popularity	Popularity	0
Ranked	Ranked	0
Score	Score	1
Anime_Rating	Anime_Rating	0
Genre_Count	Genre_Count	24

8 rows

Hide

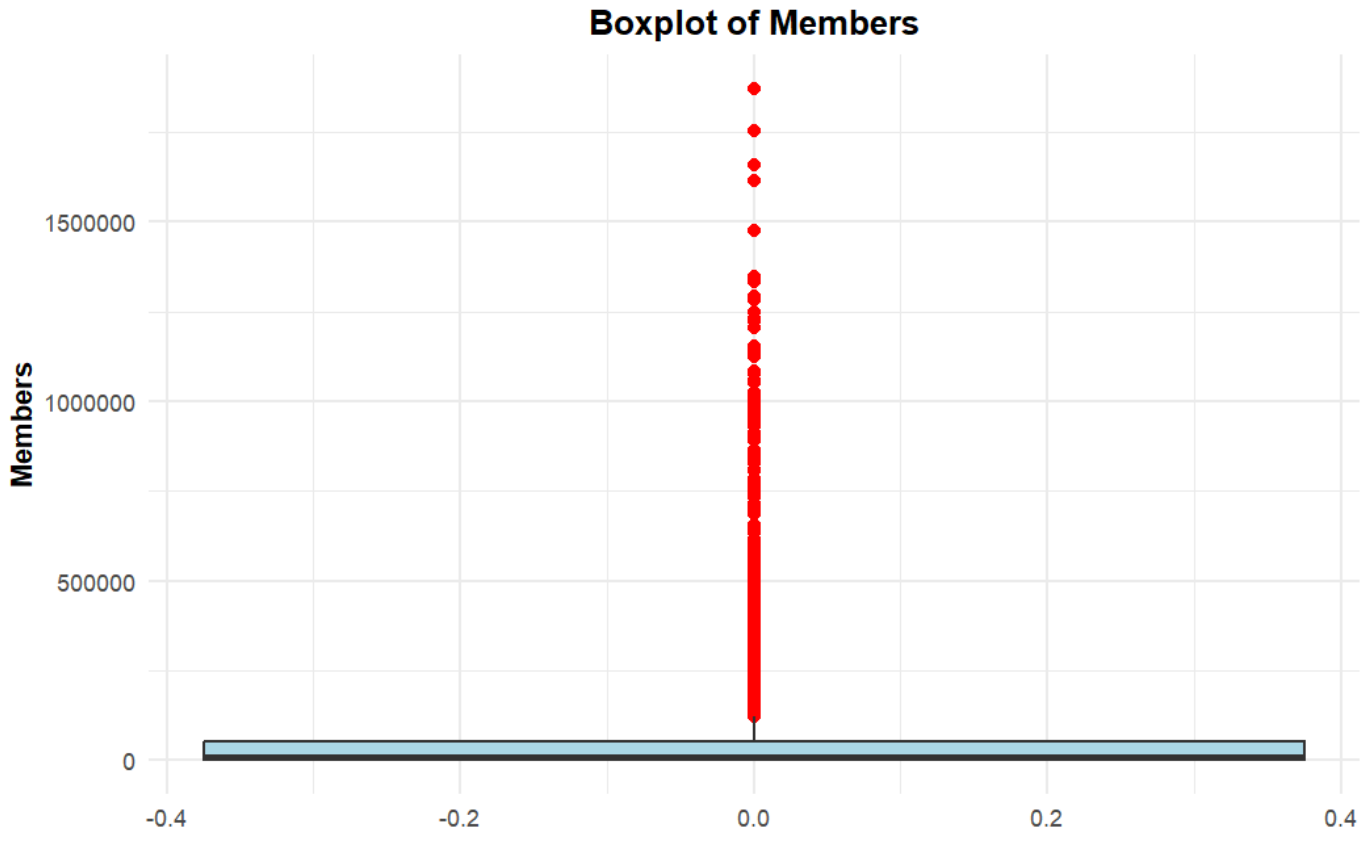
```
# Function to create and display improved boxplots
create_improved_boxplot <- function(df, column_name) {
  p <- ggplot(df, aes_string(y = column_name)) +
    geom_boxplot(outlier.colour = "red", outlier.shape = 16, outlier.size = 2, fill = "lightb
lue") +
    labs(title = paste("Boxplot of", column_name),
         y = column_name) +
    theme_minimal() +
    theme(
      plot.title = element_text(hjust = 0.5, face = "bold"),
      axis.title.y = element_text(face = "bold"),
      legend.position = "none"
    )
  print(p)
}

# Assuming cleaned_dataset is your dataframe
# Creating boxplots for specific columns
create_improved_boxplot(cleaned_dataset, "Ep_CountX")
```

```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
Please use tidy evaluation idioms with `aes()`.
See also `vignette("ggplot2-in-packages")` for more information.
```



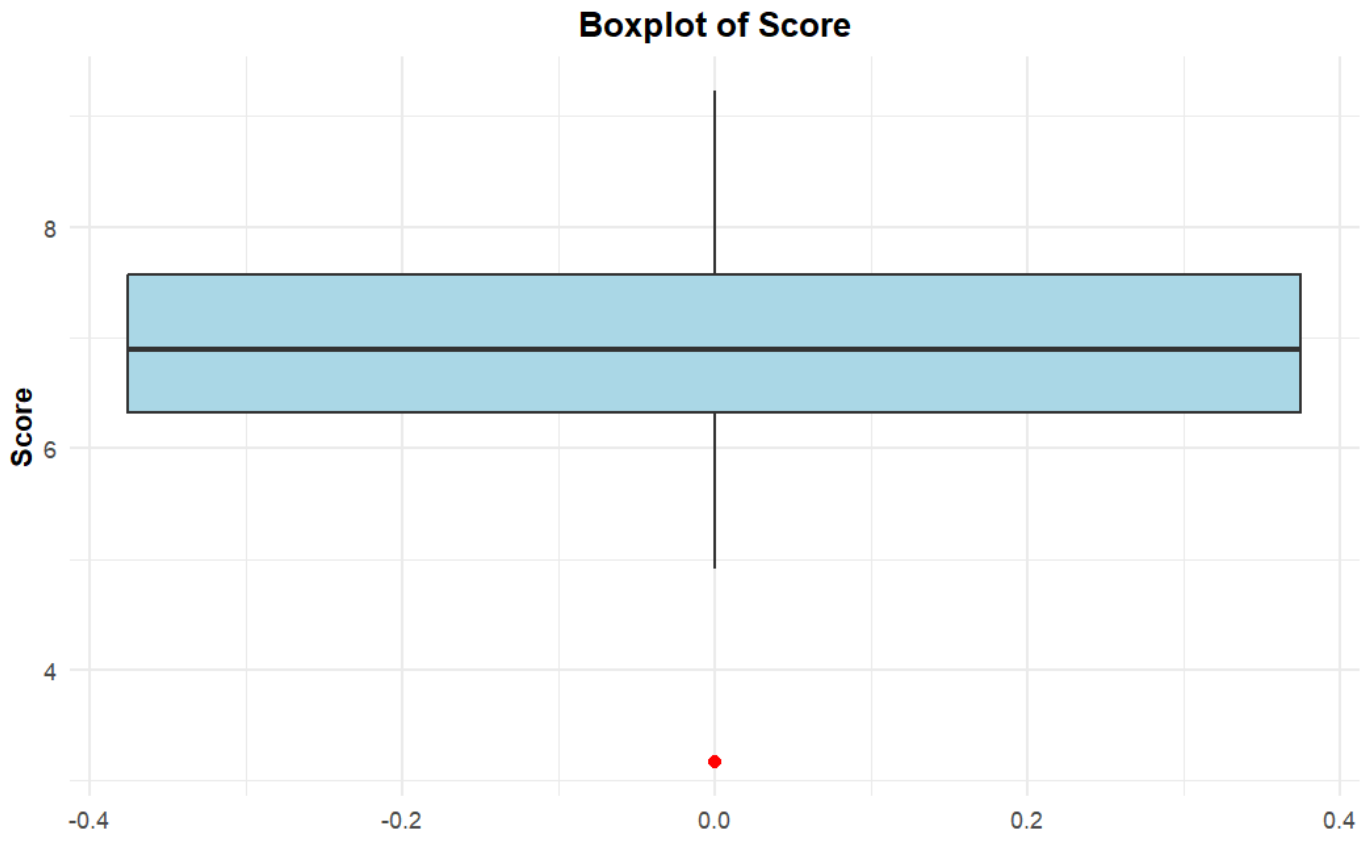
Hide

```
create_improved_boxplot(cleaned_dataset, "Members")
```



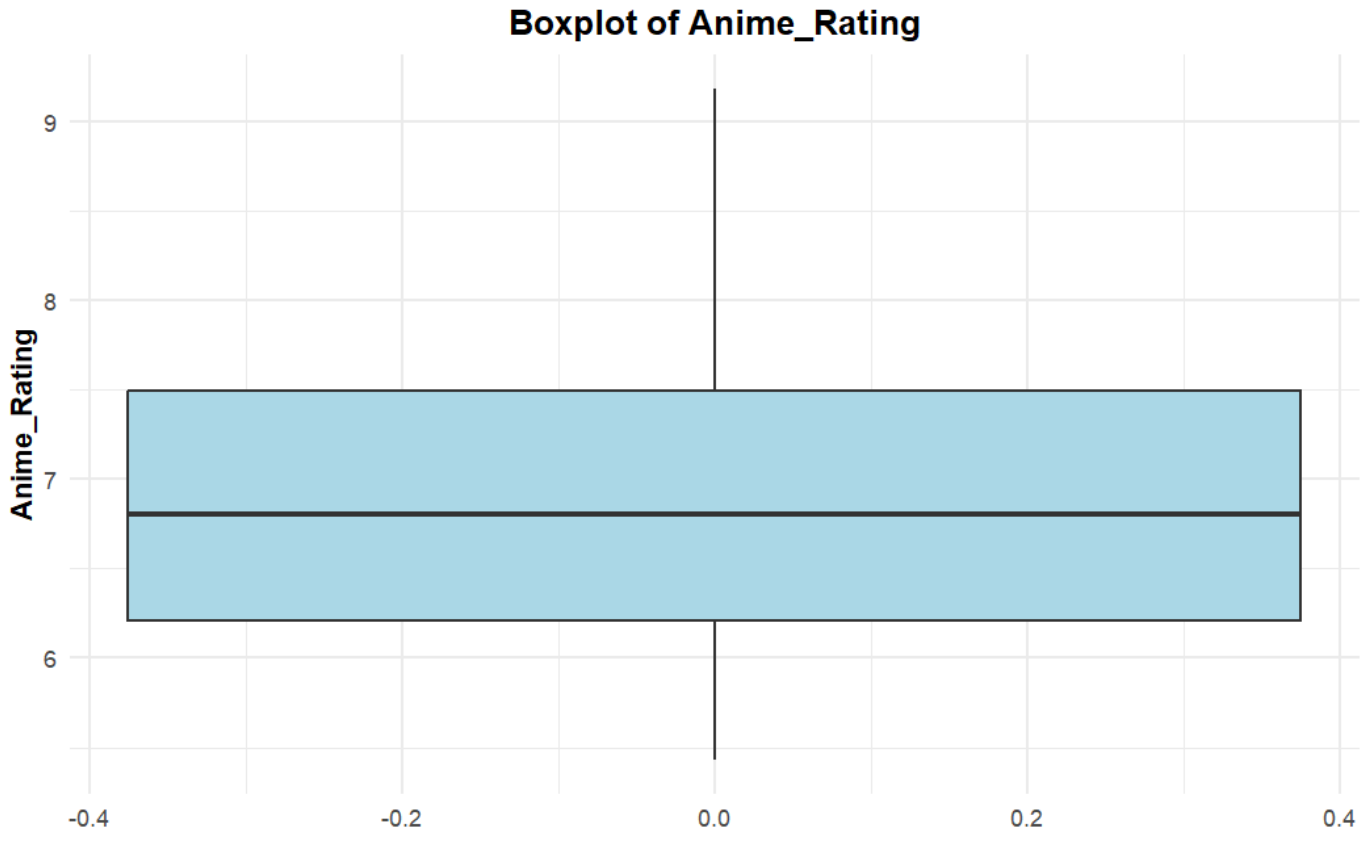
Hide

```
create_improved_boxplot(cleaned_dataset, "Score")
```



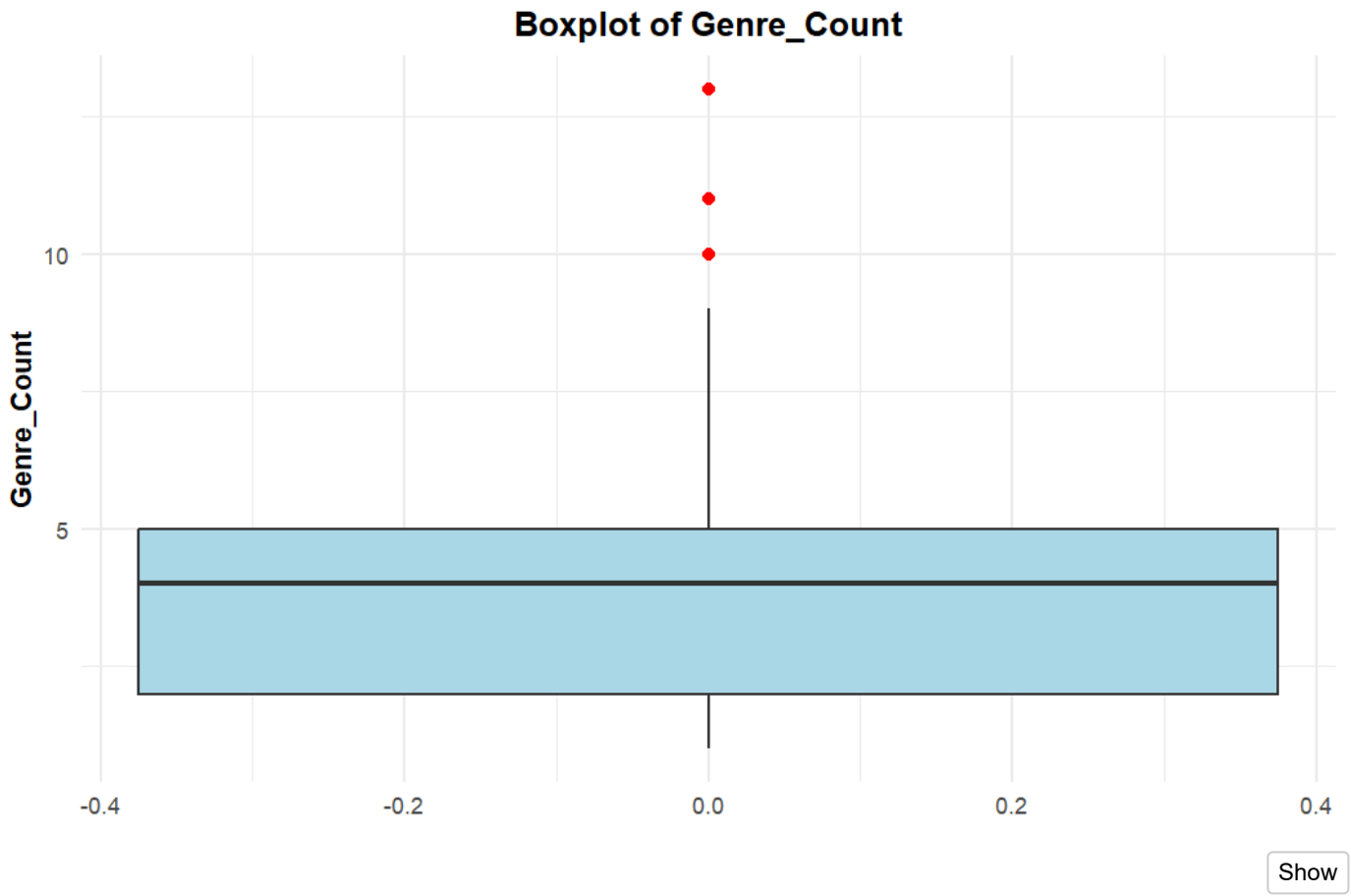
Hide

```
create_improved_boxplot(cleaned_dataset, "Anime_Rating")
```



Hide

```
create_improved_boxplot(cleaned_dataset, "Genre_Count")
```



Insights and Implications from Boxplot Analysis

1. **Ep_CountX Boxplot**
   - **Insight:** There is considerable variability in episode counts, with many outliers representing series with an unusually high number of episodes.
   - **Implication:** When analyzing average episode counts, it's crucial to consider these outliers, which often represent long-running series.
2. **Members Boxplot**
   - **Insight:** A few anime shows have exceptionally high member counts, standing out as outliers.
   - **Implication:** Such anomalies could skew analyses related to popularity metrics, suggesting the need for careful interpretation or different handling of these data points.
3. **Genre_Count Boxplot**
   - **Insight:** A small number of anime are associated with a notably higher number of genres, appearing as outliers.
   - **Implication:** These outliers may suggest genres-rich anime that offer diverse themes, potentially indicating either genre complexity or tagging inconsistencies.
4. **Anime_Rating Boxplot**
   - **Insight:** Anime ratings predominantly cluster around a median value, but there are notable outliers on both ends.
   - **Implication:** Outliers may represent either critically acclaimed masterpieces or poorly received titles, which could be highlighted in analyses focusing on rating extremes.
5. **Score Boxplot**
   - **Insight:** Scores primarily cluster around a central tendency but include outliers, similar to ratings.
   - **Implication:** While analyzing average scores, consider how outliers, representative of exceptionally high or low-quality anime, might influence the overall metrics.

Transform

Hide

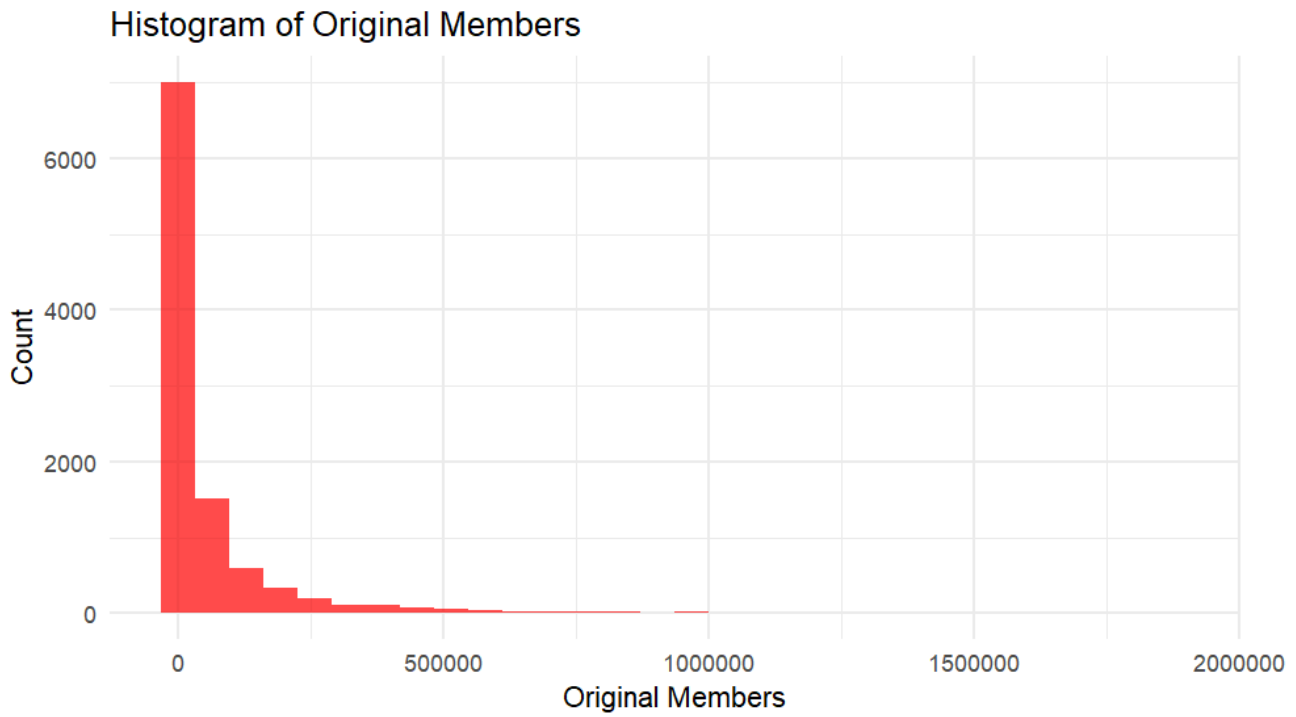
```
# Duplicating the data frame for further wrangling
clean_mm <- cleaned_dataset

# Min-Max Normalisation function
minmaxnormalise <- function(x) {
  return((x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE)))
}

# Log transformation function
log_transform <- function(x) {
  return(log1p(x)) # log1p handles zero values (log(1 + x))
}

# Apply Min-Max Normalisation to relevant numeric columns
clean_mm <- clean_mm %>%
  mutate(
    Normalized_Score = minmaxnormalise(Score),
    Normalized_Members = minmaxnormalise(Members),
    Normalized_Popularity = minmaxnormalise(Popularity)
  )

# Apply log transformation to relevant numeric columns
clean_mm <- clean_mm %>%
  mutate(
    Log_Members = log_transform(Members),
    Log_Popularity = log_transform(Popularity)
  )

# Rounding the normalized columns for easier comprehension
clean_mm <- clean_mm %>%
  mutate(
    Normalized_Score = round(Normalized_Score, digits = 5),
    Normalized_Members = round(Normalized_Members, digits = 5),
    Normalized_Popularity = round(Normalized_Popularity, digits = 5)
  )

# Reordering data frame into a desired order
clean_mm <- clean_mm %>%
  select(Title, Genre, Type, Airing_Start, Airing_End, Members, Normalized_Members, Log_Membe
rs, Popularity, Normalized_Popularity, Log_Popularity, Score, Normalized_Score, Ranked, Anime
_Rating, Episode_Category, Genre_Count)

# Display the first few rows of the transformed dataset
print(head(clean_mm))
```

	Title <chr>	Genre <fctr>
1	"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	['Comedy', 'Fantasy', 'School']
2	"Bungaku Shoujo" Memoire	['Drama', 'Romance', 'School']
3	"Bungaku Shoujo" Movie	['Mystery', 'Drama', 'Romance', 'School']

	Title <chr>	Genre <fctr>
4	"Eiji"	['Comedy', 'Drama', 'Sports']
5	"Eiyuu" Kaitai	['Comedy', 'Drama']
6	"Eiyuu" Kaitai	['Comedy', 'Drama']

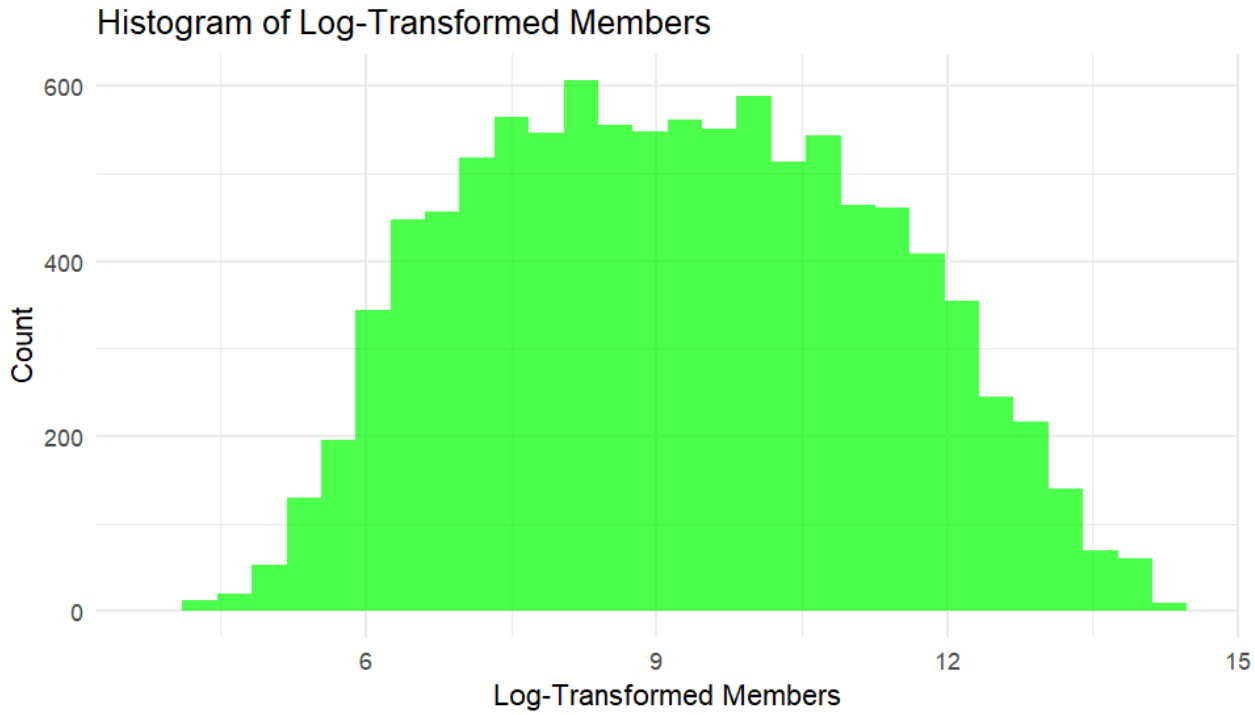
Hide

```
# Visualization of Transformations: Before and After

# Histogram for original members
ggplot(cleaned_dataset, aes(x = Members)) +
  geom_histogram(bins = 30, fill = "red", alpha = 0.7) +
  ggtitle("Histogram of Original Members") +
  xlab("Original Members") +
  ylab("Count") +
  theme_minimal() +
  theme(plot.margin = unit(c(1, 1, 1, 1), "cm"))
```



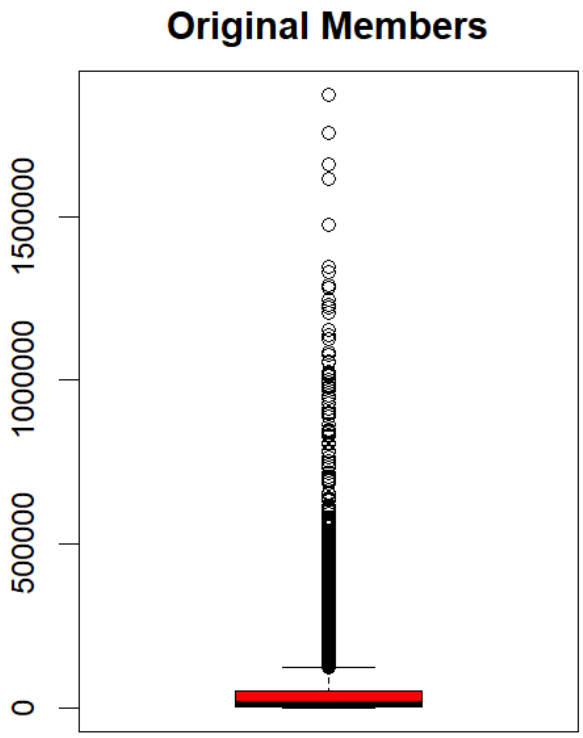
Hide

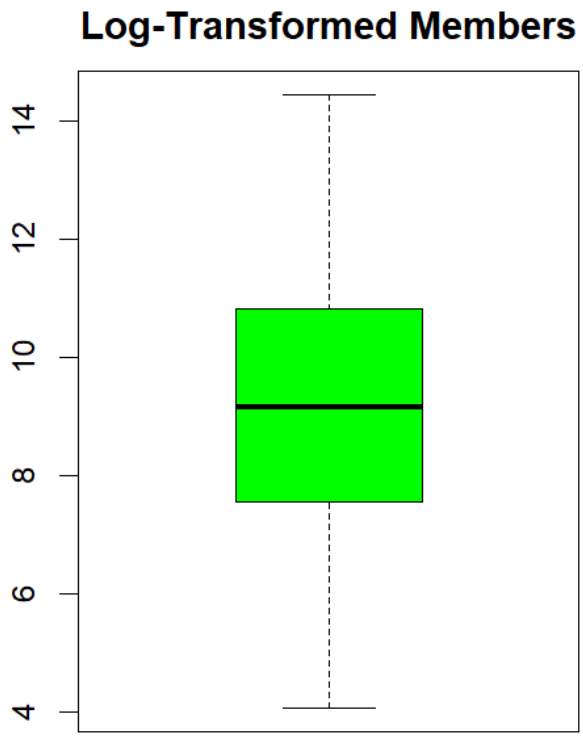
```
# Histogram for log-transformed members
ggplot(clean_mm, aes(x = Log_Members)) +
  geom_histogram(bins = 30, fill = "green", alpha = 0.7) +
  ggtitle("Histogram of Log-Transformed Members") +
  xlab("Log-Transformed Members") +
  ylab("Count") +
  theme_minimal() +
  theme(plot.margin = unit(c(1, 1, 1, 1), "cm"))

# Boxplot for original members and log-transformed members
par(mfrow = c(1, 2), mai = c(0.5, 0.5, 0.5, 0.5))
```



Hide

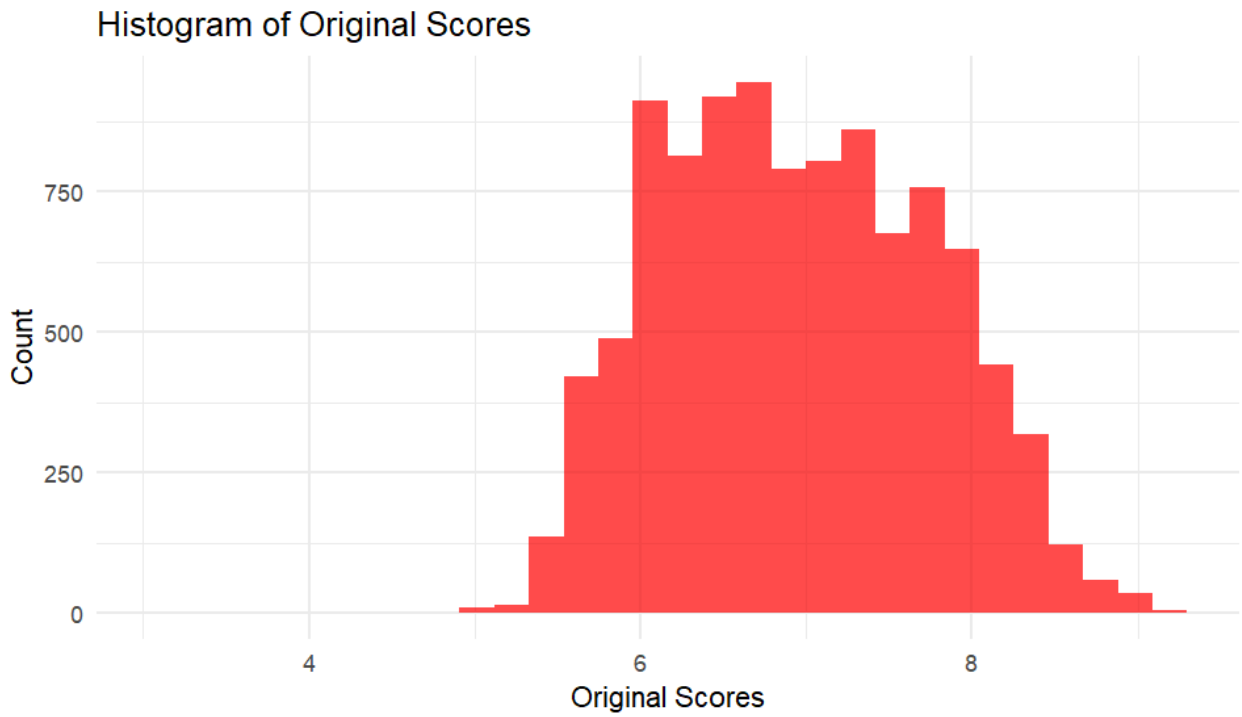
```
boxplot(cleaned_dataset$Members, main = "Original Members", col = "red")
boxplot(clean_mm$Log_Members, main = "Log-Transformed Members", col = "green")
```





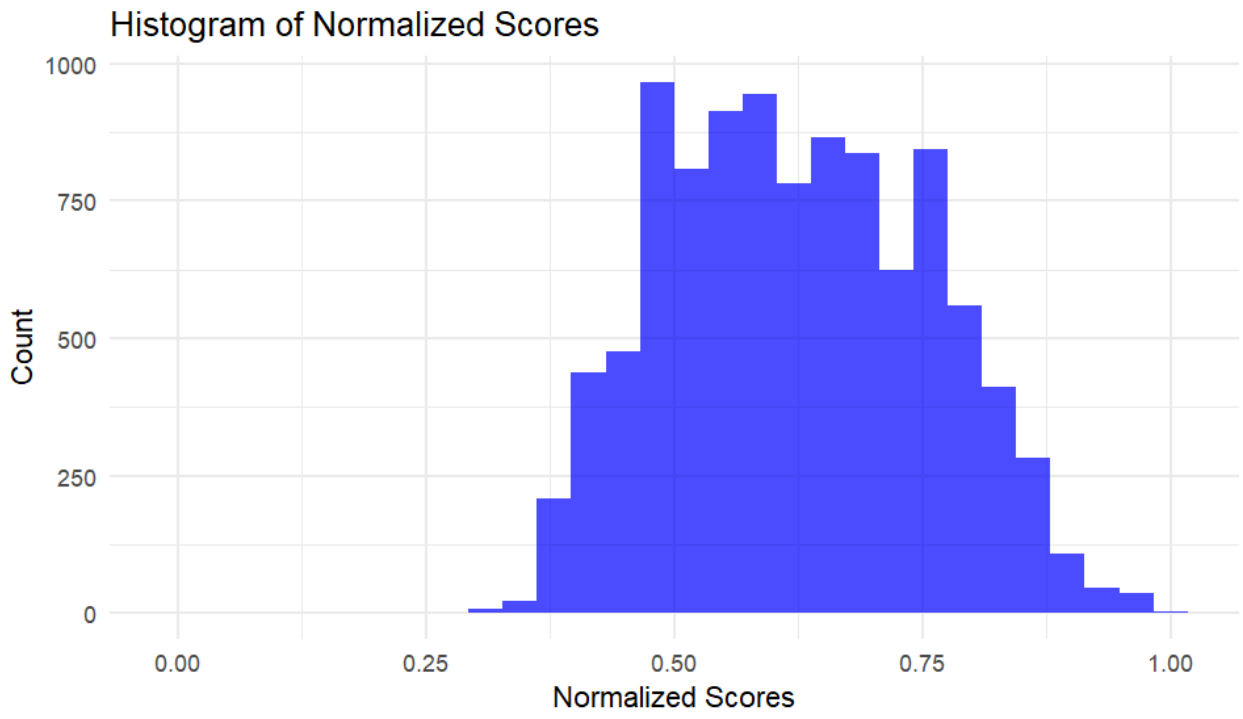
Hide

```
# Histogram for original scores
ggplot(cleaned_dataset, aes(x = Score)) +
  geom_histogram(bins = 30, fill = "red", alpha = 0.7) +
  ggtitle("Histogram of Original Scores") +
  xlab("Original Scores") +
  ylab("Count") +
  theme_minimal() +
  theme(plot.margin = unit(c(1, 1, 1, 1), "cm"))
```



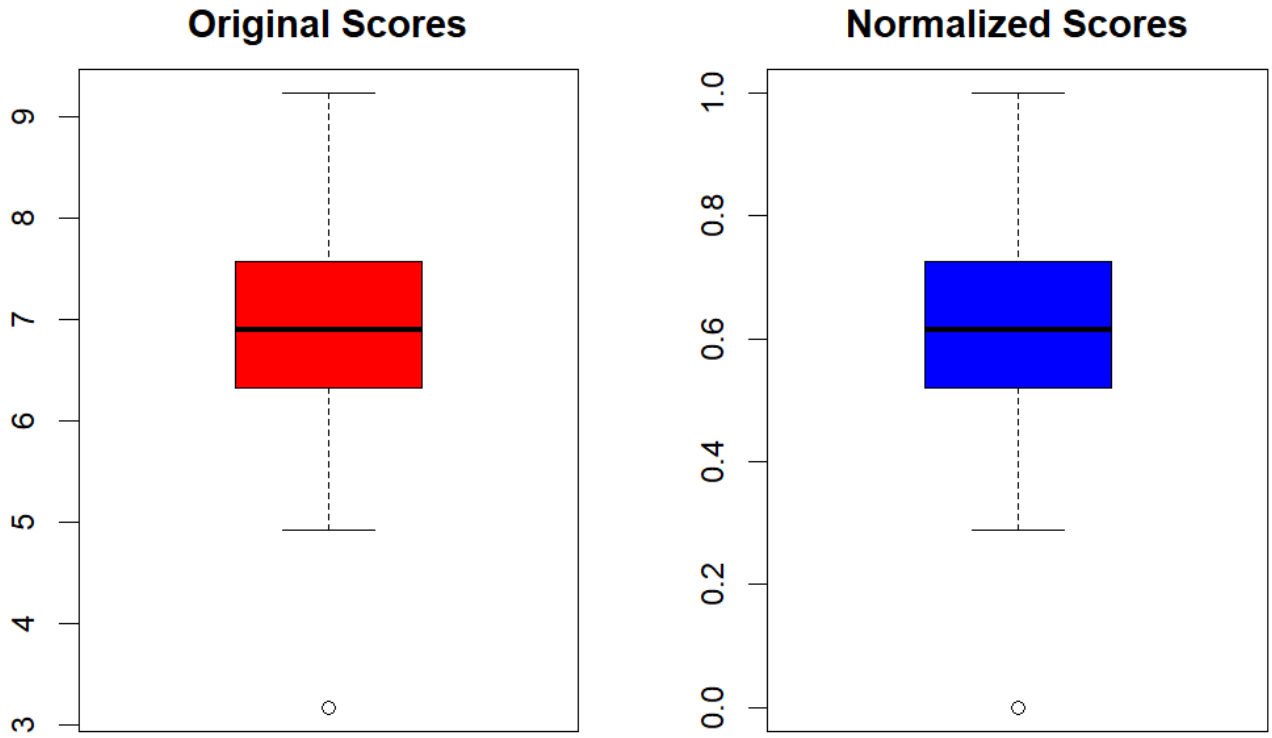
Hide

```
# Histogram for normalized scores
ggplot(clean_mm, aes(x = Normalized_Score)) +
  geom_histogram(bins = 30, fill = "blue", alpha = 0.7) +
  ggtitle("Histogram of Normalized Scores") +
  xlab("Normalized Scores") +
  ylab("Count") +
  theme_minimal() +
  theme(plot.margin = unit(c(1, 1, 1, 1), "cm"))

# Boxplot for original scores and normalized scores
par(mfrow = c(1, 2), mai = c(0.5, 0.5, 0.5, 0.5))
```



Hide

```
boxplot(cleaned_dataset$Score, main = "Original Scores", col = "red")
boxplot(clean_mm$Normalized_Score, main = "Normalized Scores", col = "blue")
```



Hide

```
# Saving the transformed dataset
write.csv(clean_mm, "cleaned_transformed_dataset.csv", row.names = FALSE)

# Conclusions on the transformations
cat("Min-Max normalization applied: Data range is now [0, 1] for normalized columns, minimizi
ng scale disparities.\n")
```

```
Min-Max normalization applied: Data range is now [0, 1] for normalized columns, minimizing sc
ale disparities.
```

Hide

```
cat("Log transformation applied: Handled positive skewness and scaled down large values, pres
erving zero values with log1p.\n")
```

```
Log transformation applied: Handled positive skewness and scaled down large values, preservin
g zero values with log1p.
```

Insights Gained from the Code

1. **Effectiveness of Min-Max Normalisation**:
   - **Normalized Columns**: `Score` , `Members` , and `Popularity` have been normalized to the [0, 1] range.
   - **Insight**: This normalization ensures that these columns are scaled uniformly, making it easier to compare their relative magnitudes and ensuring that they contribute equally in analyses that are sensitive to the scale of input features.
2. **Impact of Log Transformation**:
   - **Log-Transformed Columns**: `Members` and `Popularity` have undergone log transformation.
   - **Insight**: The log transformation effectively reduces the skewness of these distributions. Initially highly skewed data (with many extreme values) is now more evenly distributed, which is beneficial for statistical analyses and machine learning algorithms that assume normality.
3. **Distribution Visualizations**:
   - **Histograms and Boxplots**: Visual comparisons before and after transformations for `Members` and `Score` .
   - **Insight**:
     - **Original Members**: Highly skewed with many extreme outliers.
     - **Log-Transformed Members**: More symmetric and centered, indicating reduced skewness and better suitability for further analysis.
     - **Original Scores**: Shows the distribution before any transformation.
     - **Normalized Scores**: Distribution remains the same in shape but scaled to a [0, 1] range, making it easier to compare across different datasets or features.
4. **Validation of Transformations**:
   - **Visual Confirmation**: Histograms and boxplots provide visual evidence of the transformations' effects.
   - **Insight**: Visual tools confirm that the transformations have successfully normalized the data and reduced skewness, validating the correctness and effectiveness of the preprocessing steps.

Overall, these insights demonstrate that the transformations applied in the code are effective in preparing the dataset for more robust and accurate analysis by normalizing scales, reducing skewness, and organizing the data in a more accessible format.