Data Wrangling (Data Preprocessing)

Practical Assessment 2

Snigdha Mathur and Astha Bathla

Setup

```r
# Load the necessary packages required to reproduce the report
library(kableExtra)
library(knitr)
library(magrittr)
library(dplyr)
library(readr)
library(stringr)
library(tidyr)
library(ggplot2)
```

Student names, numbers and percentage of contributions

Table 1: Group information

Student name	Student number	Percentage of contribution
Snigdha Mathur	S4017572	50
Astha Bathla	S3999096	50

Executive Summary

The preprocessing of the Movies and Box Office dataset concentrated on enhancing data quality for analyses. Our approach centered on cleansing, transforming, standardizing, and restructuring the data to address potential inaccuracies and ensure compliance with tidy data principles.

- Data Inspection and Cleaning: We started with a thorough examination of the dataset to correct discrepancies in data types and formats. Missing values in crucial columns like "Director" and "Original Language" were imputed using the mode, while numeric gaps were filled using the median to mitigate outlier impacts.
- Data Tidying: Initial assessments revealed untidy aspects, particularly in the "Genre", "Release Date" and "Cast" columns, which contained multiple data points per cell. These were separated into distinct columns, aligning each variable with a single column as per tidy data principles.
- Outlier Detection and Handling: Outliers were identified using the Interquartile Range (IQR). Significant outliers, particularly in "Box Office Collection," were noted for their extreme values. These outliers were capped at calculated thresholds, preserving data integrity while reducing skew.
- Normalization and Data Transformation: To address the skewed distribution in financial figures, a logarithmic transformation was applied to "Box Office Collection." This step normalized the distribution, making the variable normally distributed
- Visualization: Box plots for all numeric variables confirmed the effectiveness of our data transformation and outlier management strategies, showing improved uniformity across key metrics.

Through these steps, the dataset has been cleaned, is more consistent, and structured, ensuring it is well-prepared for detailed analysis.

Data

The datasets selected for detailed analysis in this report are titled 'Movies_dataset' and 'Box Office Collections'. The datasets are stored in CSV format and imported using read_csv function in R coding.

Both datasets have been sourced from Kaggle, an open source website that provides access to a wide range of datasets for various types of analysis. The datasets used are available at the following URLs:

1. IMDB Top Rated Movies Dataset: https://www.kaggle.com/datasets/shubhammaindola/tmdb-top-rated-movies-dataset

IMDb, the Internet Movie Database, is an online resource that stores and ranks the movies, television shows, and celebrities worldwide. The IMDB Top Rated Movie dataset features a dataset of over 9000 top-rated movies, which includes detailed attributes like title, release date, and genre.

2. Box Office Collections Dataset: https://www.kaggle.com/datasets/anotherbadcode/boxofficecollections

The box office collection dataset provides information on the total earnings each movie has made in theaters. Variables like IMDb rating, runtime in minutes, and Metascore are included to help describe and differentiate each movie. These metrics can influence a movie's total earnings by affecting its appeal and perceived quality to audiences and critics.

Importing the datasets :

```r
movie_dataset <- read_csv("movie_dataset.csv")
```

```
## Rows: 9335 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): title, genres, original_language, overview
## dbl  (4): id, popularity, vote_count, vote_average
## date (1): release_date
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(movie_dataset)
```

```
## # A tibble: 6 x 9
##      id title          release_date genres original_language overview popularity
##   <dbl> <chr>          <date>       <chr>  <chr>             <chr>         <dbl>
## 1   278 The Shawshank~ 1994-09-23   Drama~ en                Framed ~      125.
## 2   238 The Godfather  1972-03-14   Drama~ en                Spannin~      200.
## 3   240 The Godfather~ 1974-12-20   Drama~ en                In the ~      129.
## 4   424 Schindler's L~ 1993-12-15   Drama~ en                The tru~      108.
## 5   389 12 Angry Men   1957-04-10   Drama  en                The def~       69.9
## 6   129 Spirited Away  2001-07-20   Anima~ ja                A young~      109.
## # i 2 more variables: vote_count <dbl>, vote_average <dbl>
```

Movies dataset : The dataset comprises of 9335 rows spread across 9 columnns. The columns are defined below :

- ID : A unique identifier assigned to each movie in the dataset. Type : Numeric
- Title : The official title of the movie. Type : Character
- Release Date : The official release date of the movie. Type : Date (formatted as YYYY/MM/DD).
- Genre : The genre or genres associated with the movie, such as Action, Comedy, Thriller, etc Type : Character
- Original Language : The language in which the movie was originally produced Type : Character
- Overview :A brief description or synopsis of the movie's plot Type : Character
- Popularity : A metric that quantifies the popularity of the movie, often based on various factors like viewership, internet searches, and social media mentions. Type : Numeric
- Vote count :The total number of votes that the movie received. Type : Numeric
- Vote Average :The average rating out of 10 given to the movie by viewers. Type : Numeric

```
box_office_collections <- read_csv("BoxOfficeCollections.csv")
```

```
## Rows: 1378 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Movie, Director, Cast, Consensus, Imdb_genre
## dbl (8): Year, Score, Adjusted Score, Box Office Collection, IMDB Rating, me...
##
## 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.
```

```
head(box_office_collections)
```

```
## # A tibble: 6 x 13
##   Movie                 Year Score `Adjusted Score` Director     Cast  Consensus
##   <chr>                <dbl> <dbl>            <dbl> <chr>        <chr> <chr>
## 1 Hot Rod               2007    39             42.9 Akiva Schaf~ Andy~ For Rod ~
## 2 Game Night            2018    85             99.8 John Franci~ Jaso~ Max and ~
## 3 The First Wives Club  1996    49             53.2 Hugh Wilson  Gold~ Desponde~
## 4 Scary Movie           2000    52             55.0 Keenen Ivor~ Shaw~ Defying ~
## 5 Blockers              2018    84             96.9 Kay Cannon   Lesl~ Julie, K~
## 6 The Bank Dick         1940   100            102.  Edward F. C~ W.C.~ Egbert S~
## # i 6 more variables: `Box Office Collection` <dbl>, Imdb_genre <chr>,
## #   `IMDB Rating` <dbl>, metascore <dbl>, time_minute <dbl>, Votes <dbl>
```

Box Office Collection dataset : The dataset comprises of 1378 rows spread across 13 columnns. The columns are defined below :

- Movie : The title of the movie. Type : Character
- Year : The year the movie was released. Type : Numeric
- Adjusted Score: A score that adjusts the movie's performance or rating for inflation or other factors over time. Type : Numeric
- Director : The director of the movie. Type : Character
- Cast :The main cast involved in the movie. Type : Character
- Consensus : A general agreement or opinion about the movie Type : Character
- Box Office Collection: The total revenue a movie generated at the box office. Type : Numeric
- Imdb genre : The genre or genres of the movie as categorized by IMDb. Type : Character
- IMDB Rating : The rating given to the movie on IMDb, typically on a scale from 1 to 10. Type : Numeric
- Metascore : An aggregated score from various critics, provided by Metacritic. Type : Numeric
- Time Minute : The runtime of the movie in minutes. Type : Numeric
- Votes : The total number of votes the movie received on IMDb. Type : Numeric

```r
# Merge the datasets using an inner join
# The inner join as movies found in both datasets are included in final dataset
combined_movies <- inner_join(movie_dataset,
                              box_office_collections, by = c("title" = "Movie"))

# Show the first few rows of the joined dataset to verify successful merge
head(combined_movies)
```

```
## # A tibble: 6 x 21
##      id title          release_date genres original_language overview popularity
##   <dbl> <chr>          <date>       <chr>  <chr>             <chr>         <dbl>
## 1   389 12 Angry Men   1957-04-10   Drama  en                The def~       69.9
## 2   155 The Dark Knig~ 2008-07-16   Drama~ en                Batman ~      114.
## 3   122 The Lord of t~ 2003-12-01   Adven~ en                Aragorn~      124.
## 4   429 The Good, the~ 1966-12-22   Weste~ it                While t~       76.8
## 5 12477 Grave of the ~ 1988-04-16   Anima~ ja                In the ~        0.6
## 6   346 Seven Samurai  1954-04-26   Actio~ ja                A samur~       52.1
## # i 14 more variables: vote_count <dbl>, vote_average <dbl>, Year <dbl>,
## #   Score <dbl>, `Adjusted Score` <dbl>, Director <chr>, Cast <chr>,
## #   Consensus <chr>, `Box Office Collection` <dbl>, Imdb_genre <chr>,
## #   `IMDB Rating` <dbl>, metascore <dbl>, time_minute <dbl>, Votes <dbl>
```

```r
#
```

EXPLANATION

Data Merging:

The datasets are merged using inner_join() from the dplyr package. This function is chosen because it merges rows from both the datasets based on matching values in the specified columns. In this case, movies that appear in both datasets are retained. The by argument specifies the columns used to match rows between datasets (title from movie_dataset and Movie from box_office_collections), ensuring accurate alignment based on movie titles.

Insights The joined dataset has 1138 observations spread across 21 variables. The joined dataset comprises of a comprehensive set of variables that combine financial, critical, and demographic data about each movie. The dataset contains complex variables such as date/time fields (e.g., release_date) and string variables (e.g., overview, genres, Director, Cast).

Understand

To properly understand the structure of the merged dataset, we will perform an inspection of the joined dataset by looking at the structure for clarifying the types of data present within each column. This will help in identifying any data type mismatches. Following this identification, necessary data type conversions will be undertaken to ensure uniformity across the dataset.

```
# Detailed inspection of the data structure and variable types
glimpse(combined_movies)
```

```
## Rows: 1,138
## Columns: 21
## $ id                       <dbl> 389, 155, 122, 429, 12477, 346, 637, 539, 120,~
## $ title                    <chr> "12 Angry Men", "The Dark Knight", "The Lord o~
## $ release_date             <date> 1957-04-10, 2008-07-16, 2003-12-01, 1966-12-2~
## $ genres                   <chr> "Drama", "Drama, Action, Crime, Thriller", "Ad~
## $ original_language        <chr> "en", "en", "en", "it", "ja", "ja", "it", "en"~
## $ overview                 <chr> "The defense and the prosecution have rested a~
## $ popularity               <dbl> 69.908, 113.725, 124.372, 76.752, 0.600, 52.10~
## $ vote_count               <dbl> 8212, 31845, 23324, 8232, 5220, 3474, 12687, 9~
## $ vote_average             <dbl> 8.500, 8.500, 8.479, 8.500, 8.464, 8.453, 8.45~
## $ Year                     <dbl> 1957, 2008, 2003, 1967, 1988, 1954, 1997, 1960~
## $ Score                    <dbl> 100, 94, 93, 97, 100, 100, 80, 96, 91, 97, 95,~
## $ `Adjusted Score`         <dbl> 106.929, 108.295, 103.518, 103.720, 102.212, 1~
## $ Director                 <chr> "Sidney Lumet", "Christopher Nolan", "Peter Ja~
## $ Cast                     <chr> "Henry Fonda, Lee J. Cobb, Ed Begley, E.G. Mar~
## $ Consensus                <chr> "Following the closing arguments in a murder t~
## $ `Box Office Collection`  <dbl> NA, 998615789, NA, NA, NA, NA, NA, 37226218, N~
## $ Imdb_genre               <chr> "Drama", "Drama", "Drama", "Adventure", "Drama~
## $ `IMDB Rating`            <dbl> 9.0, 9.0, 9.0, 8.8, 8.5, 8.6, 8.6, 8.5, 8.8, 8~
## $ metascore                <dbl> 96, 84, 94, 90, 94, 98, 59, 97, 92, 87, 87, 74~
## $ time_minute              <dbl> 96, 152, 201, 178, 89, 207, 116, 109, 178, 117~
## $ Votes                    <dbl> 763942, 2556835, 1777416, 742480, 266784, 3390~
```

```
# Dropping the repetitive columns (movie genre, release date and votes)
combined_movies <- select(combined_movies, -Imdb_genre, -Year, -Votes)


# Converting 'original_language' to factor and label it
combined_movies$original_language <- factor(combined_movies$original_language,
                              levels = c("en", "it", "fr", "de", "es", "other"),
                              labels = c("English", "Italian", "French",
                                         "German", "Spanish", "Other"))


# Converting the variable Genre to a factor
combined_movies$genres <- as.factor(combined_movies$genres)
```

```r
# Showing the first few rows of the updated dataset
head(combined_movies)
```

```
## # A tibble: 6 x 18
##      id title          release_date genres original_language overview popularity
##   <dbl> <chr>          <date>       <fct>  <fct>             <chr>         <dbl>
## 1   389 12 Angry Men   1957-04-10   Drama  English           The def~       69.9
## 2   155 The Dark Knig~ 2008-07-16   Drama~ English           Batman ~      114.
## 3   122 The Lord of t~ 2003-12-01   Adven~ English           Aragorn~      124.
## 4   429 The Good, the~ 1966-12-22   Weste~ Italian           While t~       76.8
## 5 12477 Grave of the ~ 1988-04-16   Anima~ <NA>              In the ~        0.6
## 6   346 Seven Samurai  1954-04-26   Actio~ <NA>              A samur~       52.1
## # i 11 more variables: vote_count <dbl>, vote_average <dbl>, Score <dbl>,
## #   `Adjusted Score` <dbl>, Director <chr>, Cast <chr>, Consensus <chr>,
## #   `Box Office Collection` <dbl>, `IMDB Rating` <dbl>, metascore <dbl>,
## #   time_minute <dbl>
```

EXPLANATION glimpse() function was used for displaying the few intial vaules of the combined_dataset as str() function was providing a lengthy structure of the dataset making the structure output go past the page formatting.

Factorizing Genres: After merging the datasets, the genres column is converted to a factor using as.factor() function in R.

Labeling Original Language: The original_language column is first converted to a factor with factor(). The levels argument specifies the unique language codes expected in the dataset. The languages are factorised as following : * en - English * it - Italian * fr - French * de - German * es - Spanish * other - Other

Tidy & Manipulate Data I

Upon checking the structure view of the joined dataset, we proceed to evaluate the datatsets adherence to the principles of tidiness, as stipulated by the "Tidy Data Principle." This principle dictates that: 1. Each variable must have its own column. 2. Each observation must have its own row. 3. Each value must have its own cell.

Checking combined_movies dataset for tidyness

In the merged dataset obtained from "movies.csv" and "box office collection," we observe several columns and variables that do not conform to tidy data principles.

- The "Genre" column currently includes multiple genres (such as Comedy, Horror, Drama, etc.) listed together within single entries. To adhere to the principles of tidy data, which states that each variable to form one column and each observation to form a distinct row, we will separate these entries. We will decompose the "Genre" column into individual genre entries, ensuring that each genre is represented distinctly and aligns with the tidy data framework.
- The "Year" column is presently formatted as YYYY-MM-DD, capturing the full release date in a single column. To enhance data granularity and utility, we will split this column into three new columns: "Year," "Month," and "Date," each storing respective components of the release dates. T
- The "Cast" column includes names of cast members concatenated within a single row. To align with tidy data principles, we will employ the separate_rows function to split these names into separate rows. This restructuring will allow each cast member's name to occupy a unique row under the same movie, enhancing the dataset's structure for more detailed relational analyses.

```r
# Split the 'genres' column into multiple rows
movies_tidy <- combined_movies %>%
  separate_rows(genres, sep = ",")
```

```r
# Spliting the Release Date column into Year , Month and Day
movies_tidy <- movies_tidy %>% separate(release_date,
                                 into = c("Year", "Month", "Day"), sep = "-")

# Splitting the columns Cast so that each cast member name is stored in different rows
movies_tidy <- movies_tidy %>% separate_rows(Cast, sep = ",")

# Removing duplicate rows that may have occurred due to multiple entries
movies_tidy <- movies_tidy %>%
  distinct()


print("Tidy Dataset")
```

```
## [1] "Tidy Dataset"
```

```r
glimpse(movies_tidy)
```

```
## Rows: 11,996
## Columns: 20
## $ id                      <dbl> 389, 389, 389, 389, 155, 155, 155, 155, 155, 1~
## $ title                   <chr> "12 Angry Men", "12 Angry Men", "12 Angry Men"~
## $ Year                    <chr> "1957", "1957", "1957", "1957", "2008", "2008"~
## $ Month                   <chr> "04", "04", "04", "04", "07", "07", "07", "07"~
## $ Day                     <chr> "10", "10", "10", "10", "16", "16", "16", "16"~
## $ genres                  <chr> "Drama", "Drama", "Drama", "Drama", "Drama", "~
## $ original_language       <fct> English, English, English, English, English, E~
## $ overview                <chr> "The defense and the prosecution have rested a~
## $ popularity              <dbl> 69.908, 69.908, 69.908, 69.908, 113.725, 113.7~
## $ vote_count              <dbl> 8212, 8212, 8212, 8212, 31845, 31845, 31845, 3~
## $ vote_average            <dbl> 8.500, 8.500, 8.500, 8.500, 8.500, 8.500, 8.50~
## $ Score                   <dbl> 100, 100, 100, 100, 94, 94, 94, 94, 94, 94, 94~
## $ `Adjusted Score`        <dbl> 106.929, 106.929, 106.929, 106.929, 108.295, 1~
## $ Director                <chr> "Sidney Lumet", "Sidney Lumet", "Sidney Lumet"~
## $ Cast                    <chr> "Henry Fonda", " Lee J. Cobb", " Ed Begley", "~
## $ Consensus               <chr> "Following the closing arguments in a murder t~
## $ `Box Office Collection` <dbl> NA, NA, NA, NA, 998615789, 998615789, 99861578~
## $ `IMDB Rating`           <dbl> 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9~
## $ metascore               <dbl> 96, 96, 96, 96, 84, 84, 84, 84, 84, 84, 84, 84~
## $ time_minute             <dbl> 96, 96, 96, 96, 152, 152, 152, 152, 152, 152, ~
```

To adhere to tidy data principles, we have employed the separate_rows() and separate() functions. These functions have been utilized to systematically divide the "Genre," "Release Date," and "Cast" columns within the dataset. This restructuring ensures that each genre entry, release date component, and cast member is distinctly separated into individual rows and columns.

Tidy & Manipulate Data II

To enhance the tidiness of the dataset further, we will perform manipulations to create a new column from an existing one. This step is designed to improve the comprehensibility of the dataset, making it easier to understand and analyze. By creating this new column, we aim to refine the dataset's structure and enhance its utility for more detailed examinations and insights.

We aim to create 3 new variable column : 1. Popularity_category 2. Decade 3. Genre count data

```r
# Creating a popularity category for the movies based on the popularity scores
movies_tidy <- movies_tidy %>%
  mutate(popularity_category = case_when(
    popularity > 80 ~ "High",
    popularity > 40 ~ "Medium",
    TRUE ~ "Low"
  ))

# Adding a decade category for the release year
movies_tidy <- movies_tidy %>%
  mutate(decade = cut(as.numeric(Year),
                      breaks = c(1950, 1960, 1970, 1980,
                                 1990, 2000, 2010, 2020),
                      labels = c("1950s", "1960s", "1970s",
                                 "1980s", "1990s", "2000s","2010s")))

# Count the number of genres per movie and add it as a new column
# Requires regrouping data to count genres
genre_count_data <- movies_tidy %>%
  group_by(title) %>%
  summarise(genre_count = n())

#Joining the genre_count_data varible to the tidy dataset using left join
movies_tidy <- left_join(movies_tidy, genre_count_data, by = "title")

# Display the updated dataset structure and content
print("Updated Dataset with New Variables:")
```

```
## [1] "Updated Dataset with New Variables:"
```

```r
glimpse(movies_tidy)
```

```
## Rows: 11,996
## Columns: 23
## $ id                       <dbl> 389, 389, 389, 389, 155, 155, 155, 155, 155, 1~
## $ title                    <chr> "12 Angry Men", "12 Angry Men", "12 Angry Men"~
## $ Year                     <chr> "1957", "1957", "1957", "1957", "2008", "2008"~
## $ Month                    <chr> "04", "04", "04", "04", "07", "07", "07", "07"~
## $ Day                      <chr> "10", "10", "10", "10", "16", "16", "16", "16"~
## $ genres                   <chr> "Drama", "Drama", "Drama", "Drama", "Drama", "~
## $ original_language        <fct> English, English, English, English, English, E~
## $ overview                 <chr> "The defense and the prosecution have rested a~
## $ popularity               <dbl> 69.908, 69.908, 69.908, 69.908, 113.725, 113.7~
## $ vote_count               <dbl> 8212, 8212, 8212, 8212, 31845, 31845, 31845, 3~
## $ vote_average             <dbl> 8.500, 8.500, 8.500, 8.500, 8.500, 8.500, 8.50~
## $ Score                    <dbl> 100, 100, 100, 100, 94, 94, 94, 94, 94, 94, 94~
## $ `Adjusted Score`         <dbl> 106.929, 106.929, 106.929, 106.929, 108.295, 1~
## $ Director                 <chr> "Sidney Lumet", "Sidney Lumet", "Sidney Lumet"~
## $ Cast                     <chr> "Henry Fonda", " Lee J. Cobb", " Ed Begley", "~
## $ Consensus                <chr> "Following the closing arguments in a murder t~
## $ `Box Office Collection`  <dbl> NA, NA, NA, NA, 998615789, 998615789, 99861578~
## $ `IMDB Rating`            <dbl> 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9~
## $ metascore                <dbl> 96, 96, 96, 96, 84, 84, 84, 84, 84, 84, 84, 84~
## $ time_minute              <dbl> 96, 96, 96, 96, 152, 152, 152, 152, 152, 152, ~
## $ popularity_category     <chr> "Medium", "Medium", "Medium", "Medium", "High"~
## $ decade                  <fct> 1950s, 1950s, 1950s, 1950s, 2000s, 2000s, 2000~
## $ genre_count             <int> 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16~
```

EXPLANATION

1. Popularity Category Creation:

A new variable popularity_category is created using mutate() and case_when() from the dplyr library. This categorizes movies based on their popularity scores into high, medium, or low categories, which can be helpful for segmenting the data for analysis.

2. Decade Category Addition:

Another new variable decade is added, categorizing release_year into decades. This is done using the cut() function to bin release_year into decade intervals.

3. Genre Count Calculation:

The dataset is grouped by title using group_by() and the number of genres per movie is counted using summarise() and n(). This count is then joined back to the main dataset with left_join(), adding a genre_count variable. This provides insights into the diversity of genres per movie.

Scan I

Checking for missing values

The subsequent phase of our process focuses on data cleaning. This stage involves examining the dataset for any missing values and appropriately replacing them through imputation. To systematically identify and quantify missing data, we will develop a function named missing_values. This function will be designed to process the entire dataset and return a count of missing entries, represented as NA or NULL values. This method ensures a thorough and uniform approach to detecting gaps in the data, allowing for precise corrective measures to maintain the integrity and utility of our dataset.

For numeric variables, we primarily utilize the mean or median imputation methods. This approach involves replacing missing values with the mean or median of the available data points within the same variable, depending on the distribution and presence of outliers.

For categorical variables, the mode imputation method is employed. This method involves substituting missing entries with the mode, which is the most frequently occurring category within the dataset for the specific variable. This technique is particularly effective for categorical data as it maintains the integrity of the data distribution and ensures consistency in the categorical levels.

```r
# Creating a function missing_values for scanning for missing values across all variables

missing_values <- sapply(movies_tidy, function(x) sum(is.na(x)))
print("Missing Values in Each Column:")
```

```
## [1] "Missing Values in Each Column:"
```

```r
print(missing_values)
```

```
##                  id               title                Year
##                   0                   0                   0
##               Month                 Day              genres
##                   0                   0                   0
##   original_language            overview          popularity
##                 604                   0                   0
##          vote_count        vote_average               Score
##                   0                   0                   0
##      Adjusted Score            Director                Cast
##                     0                    12                     0
##             Consensus Box Office Collection           IMDB Rating
##                     0                  2174                   857
##             metascore           time_minute   popularity_category
##                  1681                   849                     0
##                decade           genre_count
##                   799                     0
```

The result shows that there are multiple missing values in the dataset.

1. 2174 missing values in Box office Collection
2. 604 missing values in Original language column
3. 12 missing values in Director column
4. 799 missing values in Decade column

Now we shall perform numerical and categorical imputation on the dataset.

```r
# Handling missing values
# Impute missing values for numeric columns using median

# Identifying numeric columns in the dataset
numeric_columns <- sapply(movies_tidy, is.numeric)

# Imputing missing values in numeric columns with the median of each column
movies_tidy[numeric_columns] <- lapply(movies_tidy[numeric_columns], function(x)
  {
  ifelse(is.na(x), median(x, na.rm = TRUE), x)
})


# Function to calculate mode (most frequent element)
calculate_mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Identify factor and character columns
categorical_columns <- sapply(movies_tidy, function(x)
  is.factor(x) || is.character(x))

# Imputing missing values in factor and character columns using the mode
movies_tidy[categorical_columns] <- lapply(movies_tidy[categorical_columns],
                                           function(x) {
  ifelse(is.na(x), calculate_mode(x), x)
})


# Checking for missing values -  ensure all missing values are imputed
missing_values <- sapply(movies_tidy, function(x) sum(is.na(x)))
print("Missing Values in Each Column:")
```

```
## [1] "Missing Values in Each Column:"
```

```r
print(missing_values)
```

```
##                    id                 title                  Year 
##                     0                     0                     0 
##                 Month                   Day                genres 
##                     0                     0                     0 
##     original_language              overview            popularity 
##                     0                     0                     0 
##            vote_count          vote_average                 Score 
##                     0                     0                     0 
##        Adjusted Score              Director                  Cast 
##                     0                     0                     0 
##             Consensus Box Office Collection           IMDB Rating 
##                     0                     0                     0 
##             metascore           time_minute   popularity_category 
##                     0                     0                     0 
##                decade           genre_count 
##                     0                     0
```

Results The results now state that there are no missing values in the dataset.

Explanation

- sapply(movies_tidy, is.numeric): This function checks each column in movies_tidy for determining and identifying the numeric column
- lapply(movies_tidy[numeric_columns], function(x) …): Applies the imputation function to each numeric column. The function replaces missing values in each column with the median of that column.
- calculate_mode: This custom function finds the most frequently occurring value (mode) in a column.
- sapply(movies_tidy, function(x) is.factor(x) || is.character(x)): Similar to the numeric columns, this function checks each column for a factor or character variable.

Checking for special values

As part of further data cleaning, we will look for presence of special values in the dataset. Special values in a dataset are data points that, while not necessarily incorrect or missing, are unusual or noteworthy due to their nature or the impact they can have on data analysis and interpretation.

Special values in R are Inf ,-Inf and NAN values. We shall be checking for the presence of these values in the numeric columns of our dataset.

```r
# Check for NaN, Inf, and -Inf in numerical columns
special_values <- movies_tidy %>%
  summarise(
    NaN_values_BoxOffice = sum(is.nan(`Box Office Collection`), na.rm = TRUE),
    Inf_values_BoxOffice = sum(is.infinite(`Box Office Collection`), na.rm = TRUE),
    NaN_values_Rating = sum(is.nan(`IMDB Rating`), na.rm = TRUE),
    Inf_values_Rating = sum(is.infinite(`IMDB Rating`), na.rm = TRUE),
    NaN_values_MScore = sum(is.nan(metascore), na.rm = TRUE),
    Inf_values_MScore = sum(is.infinite(`metascore`), na.rm = TRUE),
    NaN_values_Time = sum(is.nan(time_minute), na.rm = TRUE),
    Inf_values_Time = sum(is.infinite(time_minute), na.rm = TRUE)
  )

print("Special Values in 'box_office':")
```

```
## [1] "Special Values in 'box_office':"
```

```r
print(special_values)
```

```
## # A tibble: 1 x 8
##   NaN_values_BoxOffice Inf_values_BoxOffice NaN_values_Rating Inf_values_Rating
##                  <int>                <int>             <int>             <int>
## 1                    0                    0                 0                 0
## # i 4 more variables: NaN_values_MScore <int>, Inf_values_MScore <int>,
## #   NaN_values_Time <int>, Inf_values_Time <int>
```

The analysis confirms that none of the numeric columns within the dataset contain either NaN or Inf values.

Checking for obvious errors

This analysis includes a check for negative values in the "Box Office Collection" variable. Negative values in this variable are not plausible, as they would erroneously suggest that a film incurred losses rather than earnings, which does not align with the typical operational definition of box office collections.

```r
# Identify and print the count of negative values in 'box_office_collections'
errors <- movies_tidy %>%
  summarise(negative_box_office = sum(`Box Office Collection` < 0, na.rm = TRUE),
            negative_ratings = sum(`IMDB Rating`<0 , na.rm = TRUE),
            negative_vote= sum(vote_count < 0, na.rm = TRUE),
            negative_mscore = sum(metascore <0 , na.rm = TRUE))
print(errors)
```

```
## # A tibble: 1 x 4
##   negative_box_office negative_ratings negative_vote negative_mscore
##                 <int>            <int>         <int>           <int>
## 1                   0                0             0               0
```

The analysis of the tidy dataset confirms the absence of negative values within its numeric columns. This observation ensures the data's integrity for further statistical evaluations.

Checking for inconsistencies

- Movies with Extremely Low Vote Counts: Examing films that exhibit unusually low voting figures less than 5 votes.
- IMDB Ratings Outside the Standard Range of 0 to 10:cAny movie entries with IMDB ratings falling outside the conventional range from 0 to 10 will be flagged as special values.
- Extreme Metascores Beyond the Range of 0 to 100: Similarly, metascores that do not fall within the typical range of 0 to 100 will also be categorized as special values.

```r
# Check for zero or extremely low vote counts, which could indicate missingdata
special_values <- movies_tidy %>%
  summarise(
    Low_Vote_Count = sum(vote_count <= 5, na.rm = TRUE),

    Unlikely_IMDB_Rating = sum('IMDB Rating' > 10 | 'IMDB Rating' < 0,
                               na.rm = TRUE), # IMDB ratings should be between 0-10

    Extreme_Metascore = sum(metascore > 100 | metascore < 0,
                             na.rm = TRUE) # Metascore should be between 0 and 100
  )
print(special_values)
```

```
## # A tibble: 1 x 3
##   Low_Vote_Count Unlikely_IMDB_Rating Extreme_Metascore
##             <int>                 <int>             <int>
## 1               0                     1                 0
```

In the analysis of the dataset, the findings reveal that there is a singular instance where the "IMDB Rating" does not adhere to the expected range of 0 to 10. This outlier necessitates the implementation of correction mechanisms to ensure data accuracy and reliability.

To address this and other potential anomalies effectively, we will implement several targeted functions:

Handling Low Vote Counts: For movies with exceptionally low vote counts, which may skew analyses related to popularity or engagement, we will replace these values with the median vote count from the dataset. This approach stabilizes the data by reducing the impact of abnormally low entries without removing valuable information.

Adjusting IMDB Ratings and Metascores:To ensure that all "IMDB Rating" and "Metascore" values fall within their respective acceptable ranges, we will employ the pmin and pmax functions.

```r
# Adjusting special values

# Replace zero 'vote_count' with the median of non-zero values
median_vote_count <- median(movies_tidy$vote_count[movies_tidy$vote_count > 0],
                            na.rm = TRUE)
movies_tidy$vote_count <- ifelse(movies_tidy$vote_count == 0, median_vote_count,
                               movies_tidy$vote_count)

# Ensure IMDB Rating and Metascore are within expected ranges
movies_tidy$`IMDB Rating` <- pmin(pmax(movies_tidy$`IMDB Rating`, 0), 10)
movies_tidy$metascore <- pmin(pmax(movies_tidy$metascore, 0), 100)

# Display the corrected dataset summary
summary(movies_tidy)
```

```
##        id             title                Year               Month          
##  Min.   :    12   Length:11996       Length:11996       Length:11996      
##  1st Qu.:  2501   Class :character   Class :character   Class :character  
##  Median : 11431   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :112480                                                           
##  3rd Qu.:150540                                                           
##  Max.   :938008                                                           
##      Day               genres          original_language   overview        
##  Length:11996       Length:11996       Min.   :1.000     Length:11996      
##  Class :character   Class :character   1st Qu.:1.000     Class :character  
##  Mode  :character   Mode  :character   Median :1.000     Mode  :character  
##                                        Mean   :1.109                       
##                                        3rd Qu.:1.000                       
##                                        Max.   :5.000                       
##    popularity       vote_count     vote_average       Score    
##  Min.   :  0.60   Min.   :  306   Min.   :3.125   Min.   :  0  
##  1st Qu.: 29.68   1st Qu.:  918   1st Qu.:6.600   1st Qu.: 75  
##  Median : 46.49   Median : 2245   Median :7.043   Median : 86  
##  Mean   : 67.97   Mean   : 4247   Mean   :6.985   Mean   : 81  
##  3rd Qu.: 80.25   3rd Qu.: 5766   3rd Qu.:7.470   3rd Qu.: 93  
##  Max.   :634.21   Max.   :35633   Max.   :8.500   Max.   :100  
##  Adjusted Score      Director             Cast            Consensus        
##  Min.   :  0.775   Length:11996       Length:11996       Length:11996      
##  1st Qu.: 82.104   Class :character   Class :character   Class :character  
##  Median : 92.559   Mode  :character   Mode  :character   Mode  :character  
##  Mean   : 88.232
##  3rd Qu.:100.594
##  Max.   :129.316
##  Box Office Collection  IMDB Rating      metascore      time_minute
##  Min.   :3.809e+04     Min.   :2.100   Min.   :  9.0   Min.   : 15.0
##  1st Qu.:3.453e+07     1st Qu.:6.700   1st Qu.: 61.0   1st Qu.: 95.0
##  Median :8.241e+07     Median :7.200   Median : 69.0   Median :105.0
##  Mean   :1.765e+08     Mean   :7.089   Mean   : 68.1   Mean   :107.8
##  3rd Qu.:1.887e+08     3rd Qu.:7.600   3rd Qu.: 77.0   3rd Qu.:118.0
##  Max.   :2.798e+09     Max.   :9.000   Max.   :100.0   Max.   :386.0
##  popularity_category     decade      genre_count
##  Length:11996        Min.   :1.0   Min.   : 2.00
##  Class :character    1st Qu.:5.0   1st Qu.:12.00
##  Mode  :character    Median :6.0   Median :12.00
##                      Mean   :5.5   Mean   :14.74
##                      3rd Qu.:7.0   3rd Qu.:16.00
##                      Max.   :7.0   Max.   :48.00
```

Summary of Key Variables *1. Director, Cast, Consensus:* Categorical data values have no missing values

2. Box Office Collection: This numeric field ranges from approximately 38,090 to about 2.798 billion, indicating a wide range of movie success.

3. IMDB Rating: Ratings range from 2.1 to 9.0, with a median of 7.2 and a mean slightly lower at 7.088, indicating a somewhat left-skewed distribution (more high ratings). There's an unusually low rating observed (min. 2.1), which might be a point of interest

4. Metascore: Scores range from 9 to 100, with both mean (68.1) and median (69) suggesting a moderate level of critical reception on average. The minimum score of 9 is notably low, indicating either outlier films or potential errors in data entry.

5. Time Minute: Movie lengths vary from 15 minutes to 386 minutes. The mean (107.8 minutes) and median (105 minutes) are typical for feature films, but the max value (386 minutes) suggests inclusion of outlier content such as extended cuts

Special Values and Errors Check

1. Zero Vote Count: There are no entries with a zero vote count, which is good as it suggests that all listed movies have received at least some voter attention.

2.Unlikely IMDB Rating: There is 1 entry with an unlikely IMDB rating (outside the typical 0-10 scale). This needs to be corrected or removed from further analysis

3. Extreme Metascore: No entries with extreme metascores (outside the 0-100 range) were found, indicating that all scores are within a plausible range.

Scan II

The next step in analying the Movies dataset, we scan for outliers present in the numeric variable in the dataset.

Methodology 1. Identifying Outliers Using IQR (Interquartile Range): The IQR is a commonly used measure of identifying the statistical dispersion and is used to find outliers. The calculations of IQR are as follows : Q1 = 25% quantile Q2 = 75% quantile lower_bound <- Q1 - 1.5 * IQR upper_bound <- Q3 + 1.5 * IQR For each numeric variable, we calculate the IQR. The outliers of these numeric variables arethose points that fall below Q1 - 1.5* IQR or above Q3 + 1.5 * IQR

2. Visualization Using Boxplots: Boxplots visualise the data distribution in the dataset, highlighting the points that lie outside the whiskers as Outliers

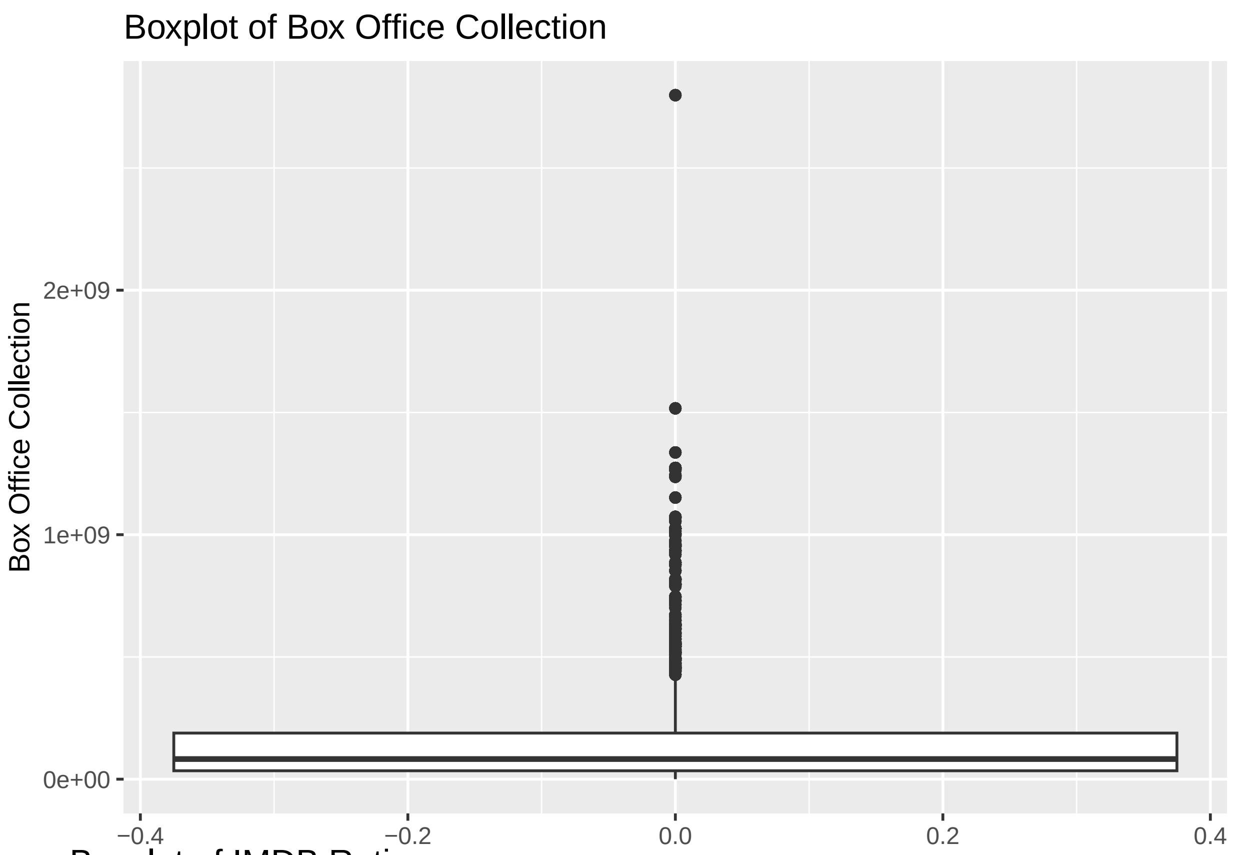
```r
# Define a function to calculate IQR and identify outliers
calculate_outliers <- function(data, column) {
  Q1 <- quantile(data[[column]], 0.25, na.rm = TRUE)
  Q3 <- quantile(data[[column]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  outliers <- data[[column]] < lower_bound | data[[column]] > upper_bound
  list(lower_bound = lower_bound, upper_bound = upper_bound,
       outliers = sum(outliers))
}

# Apply the outlier detection on numeric columns
numeric_columns <- c("Box Office Collection", "IMDB Rating",
                     "metascore", "time_minute")
outlier_summary <- lapply(numeric_columns, function(col)
                            calculate_outliers(movies_tidy, col))
names(outlier_summary) <- numeric_columns

# Convert the list to a data frame for tabular presentation
outlier_df <- do.call(rbind, lapply(names(outlier_summary), function(col) {
  cbind(Variable = col, as.data.frame(t(unlist(outlier_summary[[col]]))))
}))
outlier_df
```

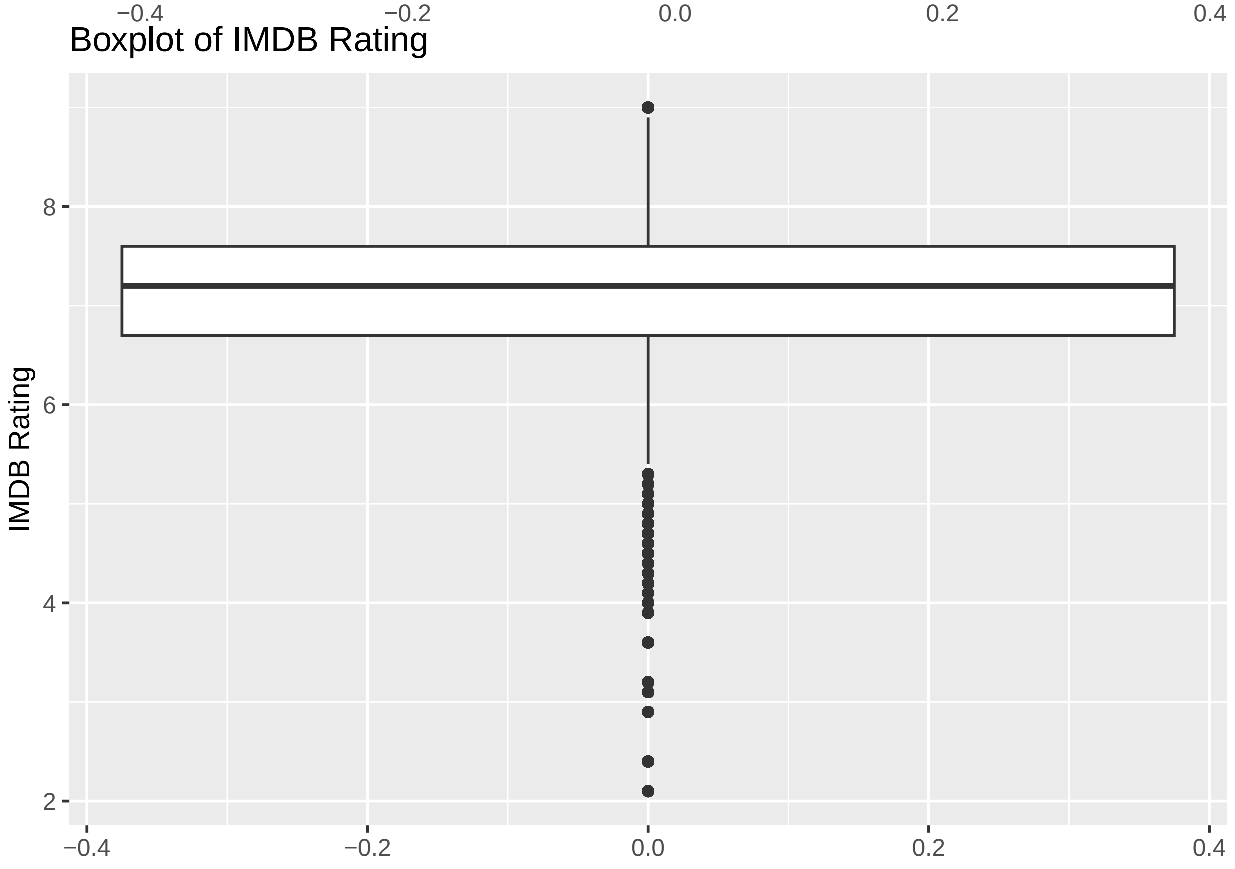
```
##                Variable lower_bound.25% upper_bound.75% outliers
## 1 Box Office Collection   -196654329.00    419847303.00     1456
## 2           IMDB Rating            5.35            8.95      564
## 3             metascore           37.00          101.00      476
## 4           time_minute           60.50          152.50      860
```

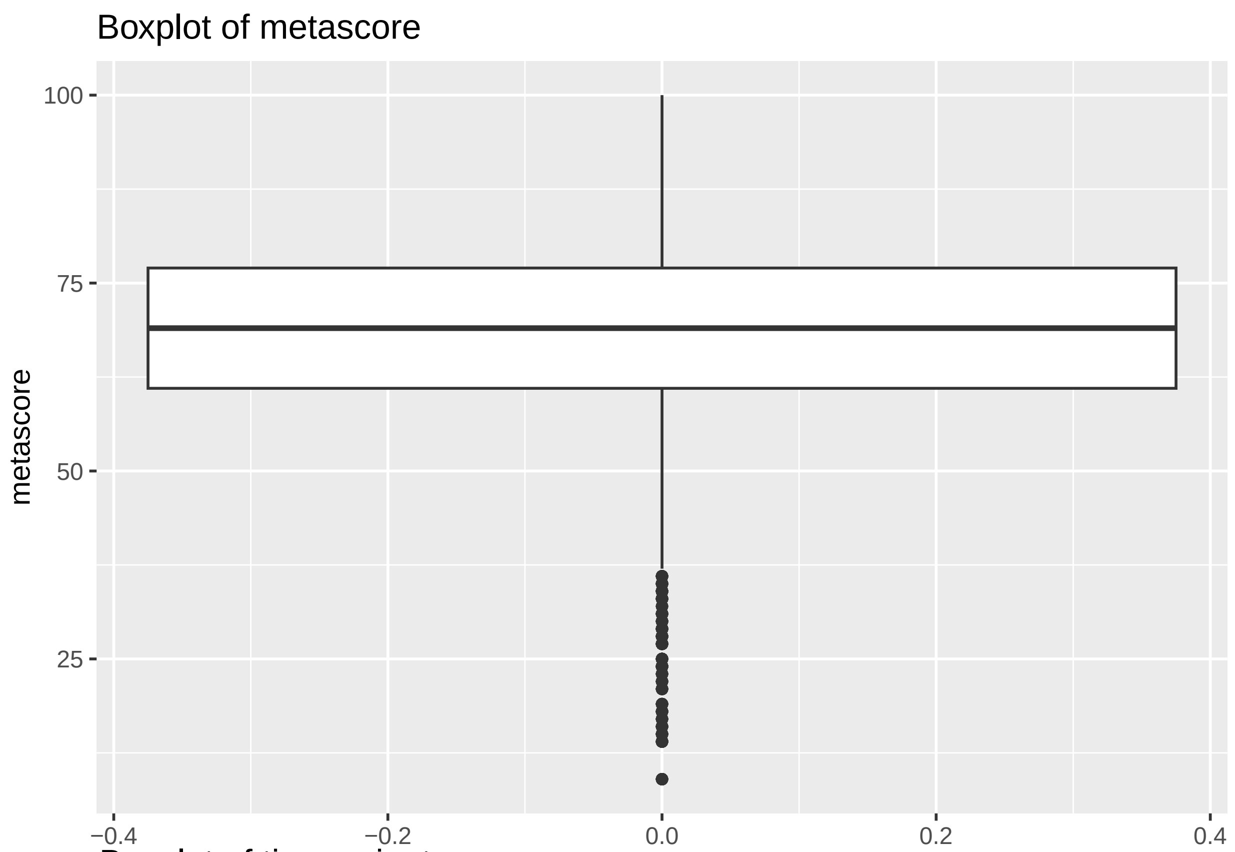
The outlier summary table provides a detailed enumeration of the number of outliers present in each numeric variable of the dataset. Notably, the “Box Office Collection” variable exhibits the highest incidence of outliers, with a total of 1,456 outliers identified. This is followed by the “Time Minute” variable, which reports approximately 860 outliers, bounded by lower and upper limits of 60.50 and 152.50, respectively. Additionally, the “IMDB Rating” and “Metascore” variables contain 564 and 476 outliers, respectively.

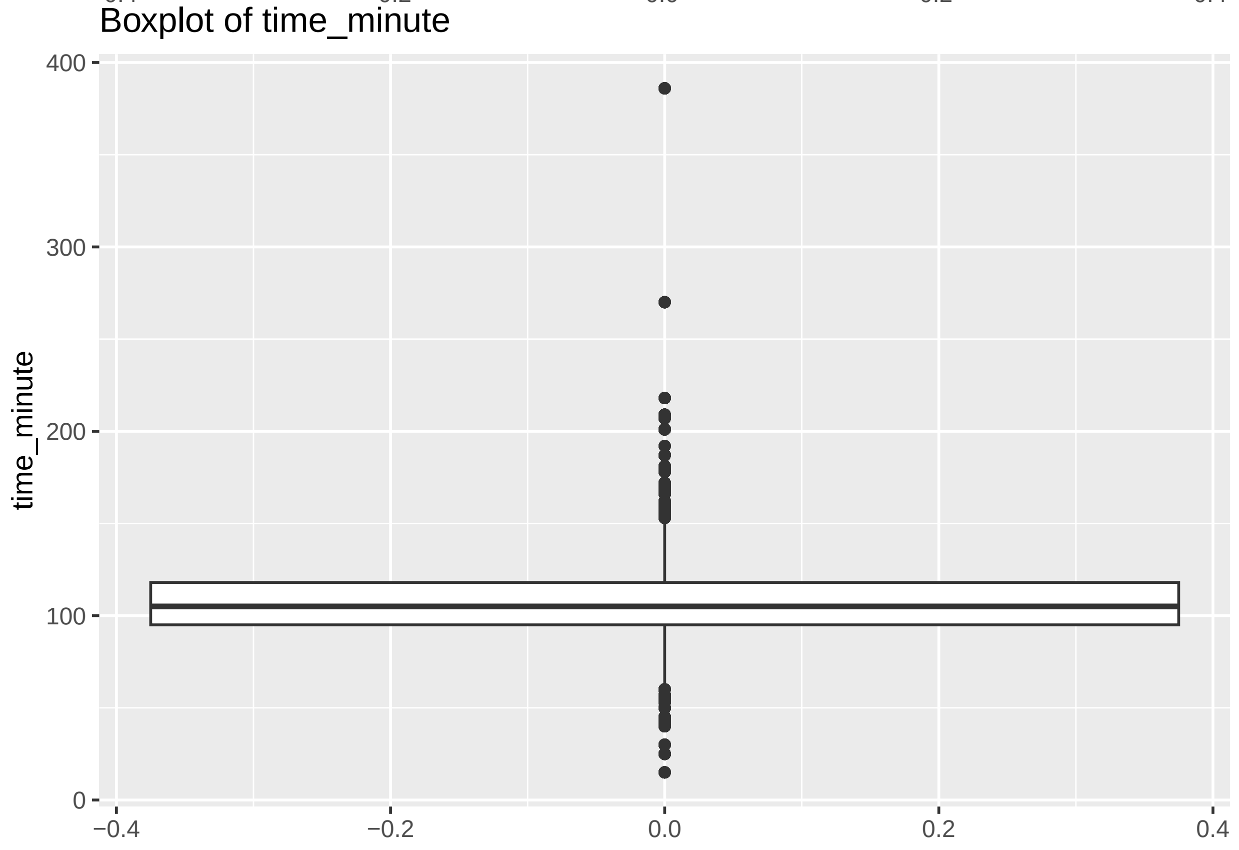
For a visual representation of the distribution of numeric variables within the dataset, we will plot box plots, which are effective tools for identifying outliers and understanding the spread of data. To accomplish this, we utilize the ggplot2 library from the ggplot2 pakcage. The box plots are generated using the geom_boxplot function.

```r
for (col in numeric_columns) {
  p <- ggplot(movies_tidy, aes(y = .data[[col]])) +
    geom_boxplot() +
    labs(title = paste("Boxplot of", col), x = NULL, y = col)
  print(p)
}
```


Boxplot of Box Office Collection
Box Office Collection
2e+09
1e+09
0e+00
−0.4
−0.2
0.0
0.2
0.4


Boxplot of IMDB Rating
IMDB Rating
8
6
4
2
−0.4
−0.2
0.0
0.2
0.4





INSIGHTS FROM THE BOX PLOTS

1. Box Office Collection This plot shows a large number of outliers above the upper whisker, indicating that several movies have earned significantly more than the general distribution. The distribution's main body is closer to the lower end, suggesting that most movies earn relatively low amounts, with a few exceptional blockbusters.

2. IMDB Rating The IMDB Ratings are concentrated between approximately 5 to 9, with a few outliers below this range. The distribution seems fairly normal but skewed slightly towards higher ratings.

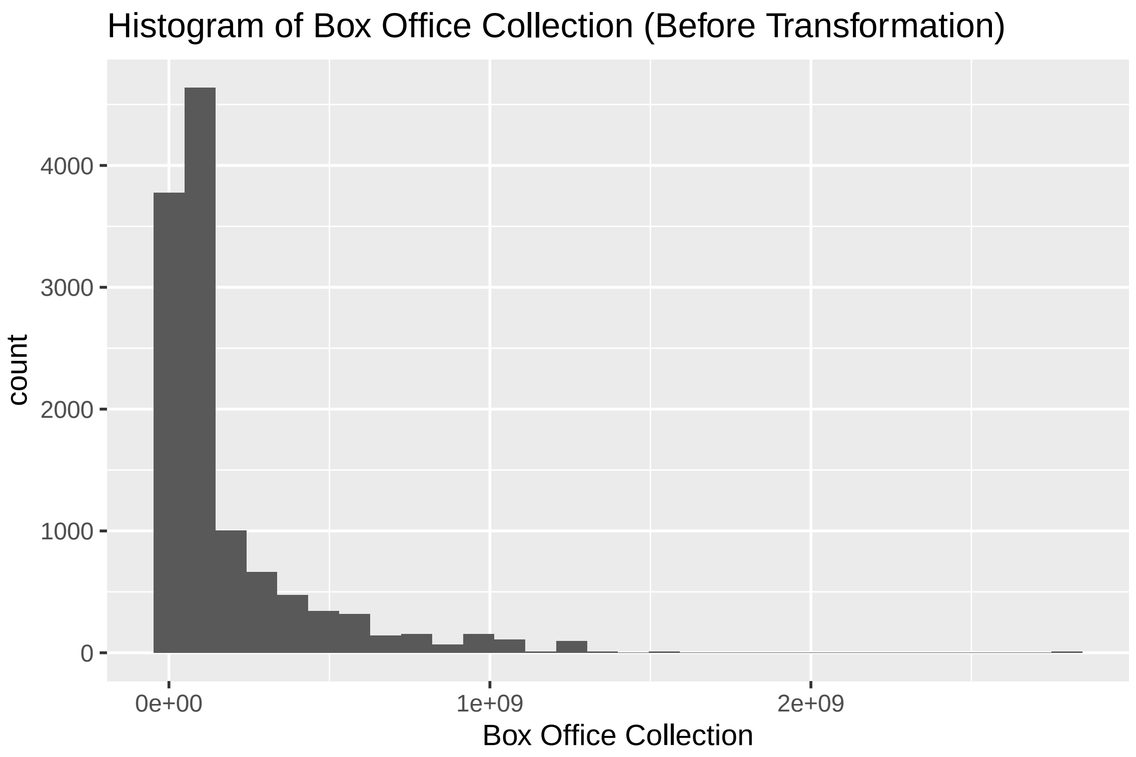
3. Metascore The Metascore data has a concentration between roughly 40 to 80, with a few outliers mainly below the lower whisker. The spread is moderate without extreme outliers on the upper end.

4. Time Minute The plot for film duration shows that most movies have a runtime clustered around 100 to 150 minutes, with several outliers indicating significantly longer movies.

The analysis of the table and the box plot clearly indicates that the "Box Office Collection" variable is significantly skewed, characterized by a wide range of numerical values, which contributes to its high outlier count. In response to this skewness and to effectively address the presence of these outliers, subsequent steps will include the implementation of transformations on the "Box Office Collection" variable. This approach aims to normalize the data distribution, thereby mitigating the impact of outliers and enhancing the robustness of our analytical outcomes.

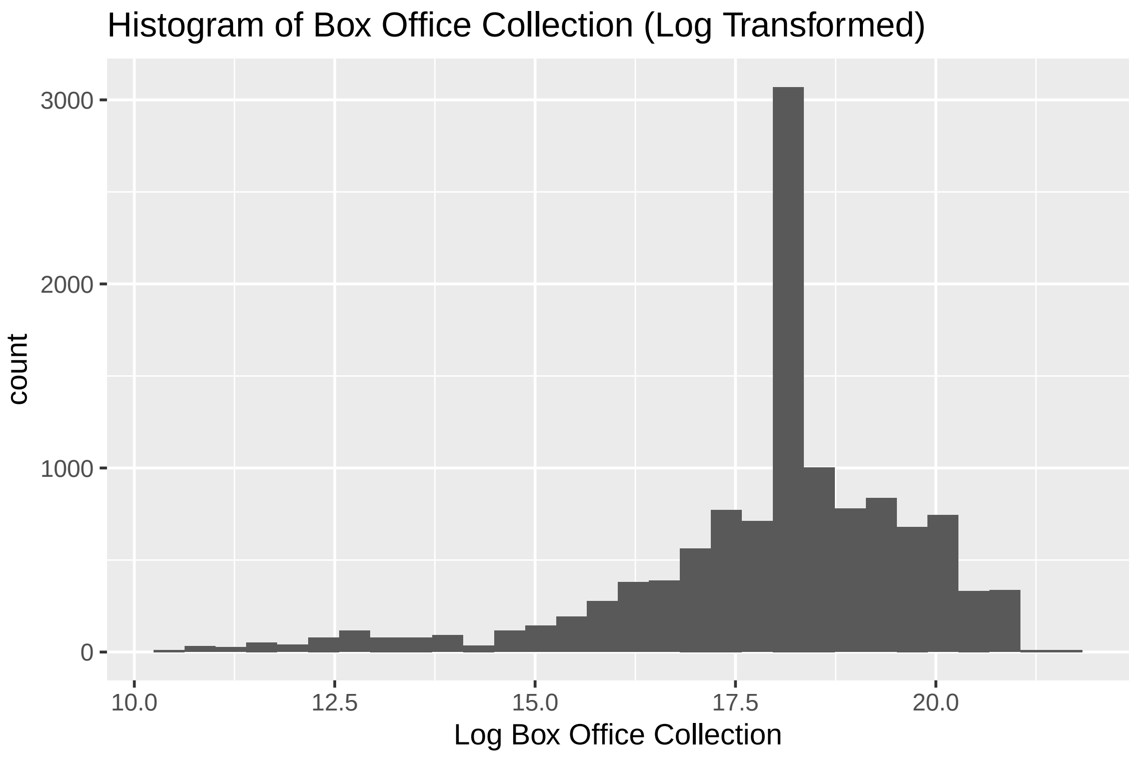
Transform

```r
# Check original distribution of 'Box Office Collection'
ggplot(movies_tidy, aes(x = `Box Office Collection`)) +
    geom_histogram(bins = 30) +
    ggtitle("Histogram of Box Office Collection (Before Transformation)")
```



```r
# Apply a logarithmic transformation to deal with skewness
movies_tidy$`Log Box Office Collection`<-log1p(movies_tidy$`Box Office Collection`)

# Plot histogram of the transformed data
```

```r
ggplot(movies_tidy, aes(x = `Log Box Office Collection`)) +
  geom_histogram(bins = 30) +
  ggtitle("Histogram of Box Office Collection (Log Transformed)")
```



Analysis of the Pre-Transformation Histogram *Skewed Distribution:* The histogram illustrates a highly right-skewed distribution where the majority of movies have relatively low box office revenues, while a few movies have exceptionally high revenues. The data concentration is significantly dense at the lower end of the scale, tapering off quickly as values increase.

Skewness of Histogram: The mean of the variable is significantly higher than the median, and the tail stretches towards the higher values (long tail). This indicates that while most movies do not achieve extremely high revenues, there are notable exceptions which are significant outliers pulling the mean upwards.

Since the data is highly skewed, we shall seleect LOG TRANSFORMATION for the variable " Box Office Collection"

Rationale for Log Transformation

Given the highly skewed nature of the "Box Office Collection" data: The logarithmic transformation aims to normalize the data distribution, making it more symmetrical and reducing the impact of extreme values. As seen in the post-transformation histogram, the log transformation significantly improved the distribution's shape, making it more bell-shaped and centered. This modification aids in stabilizing variance and makes the data more suitable for other analyses.

Analysis of the Log-Transformed Histogram

Distribution Shape: The histogram shows a more bell-shaped distribution, which is a significant improvement over the likely right-skewed distribution seen before the transformation. This suggests that the logarithmic transformation has effectively normalized the data, reducing skewness.

Centering and Scaling: The data values are now centered around a narrower range of values (approximately between 15 and 20 in log scale). This indicates that extreme values (very high box office collections) have been compressed, bringing them closer to the majority of the data. Such scaling makes the dataset more manageable and reduces the influence of outliers in statistical analyses.

Symmtery: The transformation has likely improved the symmetry and normalized the distribution, which is advantageous for many statistical modeling techniques that assume data normality

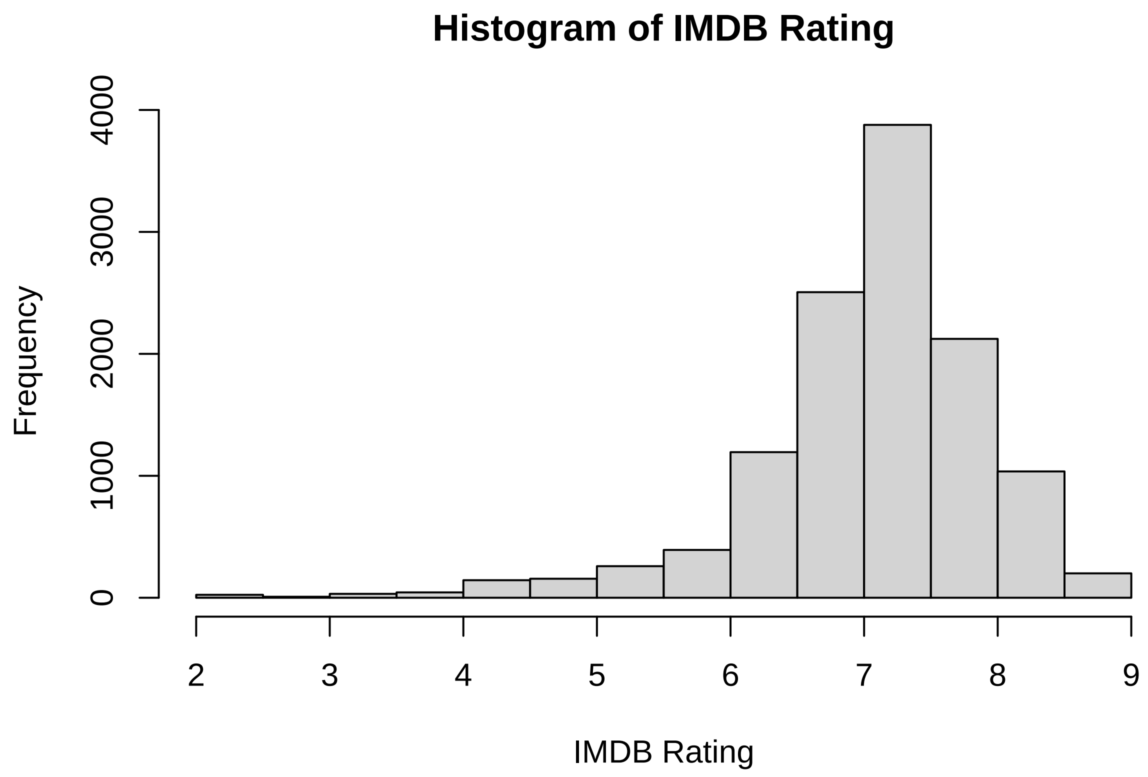
Capping the other numeric columns

Due to the observed skewness in the distribution of the 'Box Office Collection' variable within the dataset, a logarithmic transformation was applied specifically to this variable. For the rest of the numeric variable : IMDB rating, Metascore and Time Minute, we shall apply the method of Capping also known as Winsorising. This method involves defining a cutoff point, typically 1.5 times the IQR above the third quartile and below the first quartile, and then replacing any values outside these bounds with the boundary values.

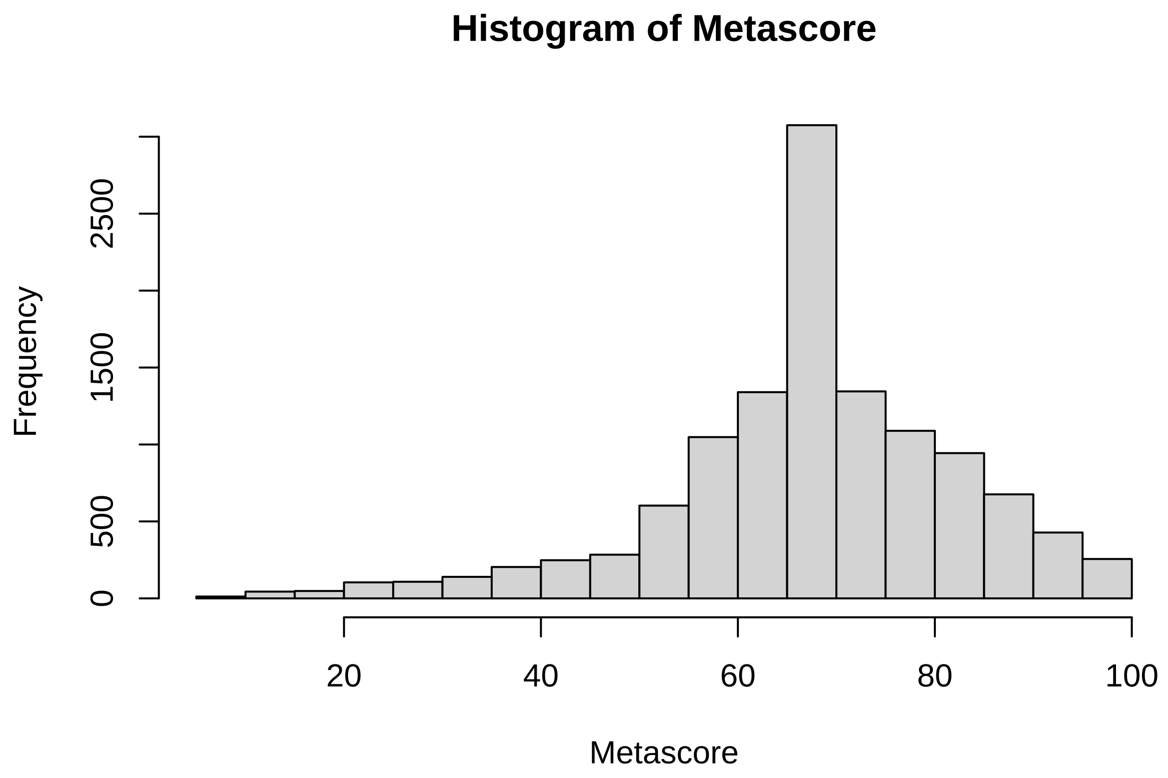
```r
# Function to cap outliers for multiple variables using the IQR method
cap_outliers <- function(data, variable) {
  for (variable in variable) {
    Q1 <- quantile(data[[variable]], 0.25, na.rm = TRUE)
    Q3 <- quantile(data[[variable]], 0.75, na.rm = TRUE)

    # Calculate the interquartile range (IQR)
    IQR <- Q3 - Q1

    # Define upper and lower limits
    upper_limit <- Q3 + 1.5 * IQR
    lower_limit <- Q1 - 1.5 * IQR

    # Cap outliers
    data <- data %>%
      mutate("{variable}" := ifelse({{variable}} > upper_limit, upper_limit,
                                  ifelse({{variable}} < lower_limit, lower_limit,
                                         {{variable}})))
  }
  return(data)
}

list_variable <- c("IMDB Rating","metascore", "time_minute")
cap_outliers_movie_tidy <- cap_outliers(movies_tidy, list_variable)
```

The capping of numeric variables was performed so that the outliers can be removed from the numeric variables. The capping method helped in reducing the number of outliers in the dataset making the dataset cleaner and efficient for analysing.

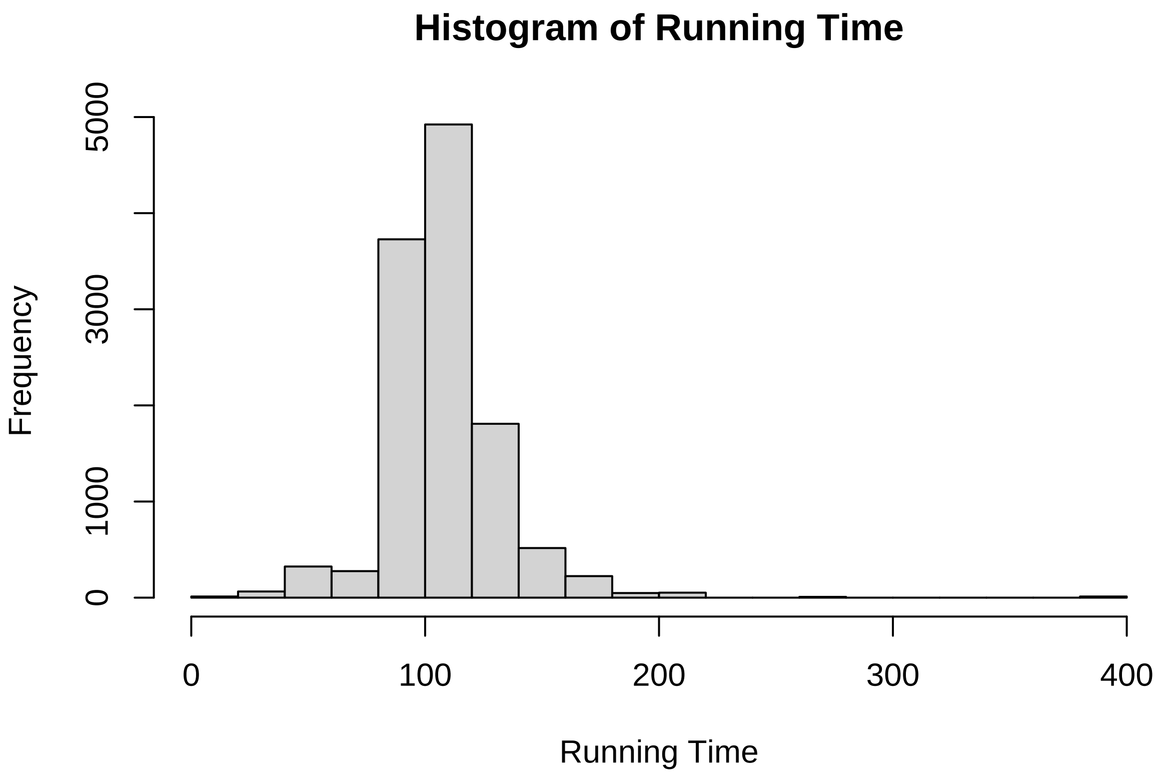
```r
# Histogrmas for the capped numeric variables
hist(movies_tidy$`IMDB Rating`,
     main = "Histogram of IMDB Rating",
     xlab = "IMDB Rating")
```



```r
hist(movies_tidy$metascore,
     main = "Histogram of Metascore",
     xlab = "Metascore")
```



```r
hist(movies_tidy$time_minute,
     main = "Histogram of Running Time",
     xlab = "Running Time")
```



References

1. Anotherbarcode (2022) *BoxOfficeCollections dataset*, Kaggle , accessed on 17th May 2024, https://www.kaggle.com/datasets/anotherbadcode/boxofficecollections
2. Shubham Maindola (2024) *IMDB Top Rated Movies Dataset* , Kaggle, accessed on 17th May 2024, https://www.kaggle.com/datasets/shubhammaindola/tmdb-top-rated-movies-dataset
3. Hadley Wickham (2022) *Data tidying* R for Data Science, accessed om 18th May 2024, https://r4ds.hadley.nz/data-tidy
4. Arimoro Olayinka (2021) *A Gentle Introduction to Tidy Data in R* Medium, accessed on 20th May 2024, https://arimoroolayinka.medium.com/a-gentle-introduction-to-tidy-data-in-r-b6673b4d304c
5. Data Preprocessing (Data Wrangling) *Module 4 -Tidy Data Principles and Manipulating Data* accessed on 21st Maay 2024, https://rare-phoenix-161610.appspot.com/secured/Module_04.html
6. Data Preprocessing (Data Wrangling) *Module 5 - Scan: Missing Values* accessed on 21st May 2024, http://rare-phoenix-161610.appspot.com/secured/Module_05.html#Overview
7. Data Preprocessing (Data Wrangling) *Module 6 - Scan: Outliers* accessed on 21st May 2024, http://rare-phoenix-161610.appspot.com/secured/Module_06.html#Overview
8. Data Preprocessing (Data Wrangling) *Module 7 - Transform: Data Transformation, Standardisation, and Reduction* accessed on 21st May 2024, http://rare-phoenix-161610.appspot.com/secured/Module_07.html#Overview
9. R Bloggers (nd) *How to Remove Outliers in R* R Bloggers, accessed on 27th May 2024, https://www.r-bloggers.com/2020/01/how-to-remove-outliers-in-r/
10. Dplyr (nd) *Create, modify, and delete columns* dplyr, accessed on 26th May 2024, https://dplyr.tidyverse.org/reference/mutate.html

```
citation("dplyr")
```

```
## To cite package 'dplyr' in publications use:
##
##   Wickham H, François R, Henry L, Müller K, Vaughan D (2023). _dplyr: A
##   Grammar of Data Manipulation_. R package version 1.1.4,
##   <https://CRAN.R-project.org/package=dplyr>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {dplyr: A Grammar of Data Manipulation},
##     author = {Hadley Wickham and Romain François and Lionel Henry and Kirill Müller and Davis Vaughan
##     year = {2023},
##     note = {R package version 1.1.4},
##     url = {https://CRAN.R-project.org/package=dplyr},
##   }
```

```
citation("ggplot2")
```

```
## To cite ggplot2 in publications, please use
##
##   H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
##   Springer-Verlag New York, 2016.
##
## A BibTeX entry for LaTeX users is
##
```

```
##   @Book{,
##     author = {Hadley Wickham},
##     title = {ggplot2: Elegant Graphics for Data Analysis},
##     publisher = {Springer-Verlag New York},
##     year = {2016},
##     isbn = {978-3-319-24277-4},
##     url = {https://ggplot2.tidyverse.org},
##   }
```

```r
citation("stringr")
```

```
## To cite package 'stringr' in publications use:
##
##   Wickham H (2023). _stringr: Simple, Consistent Wrappers for Common
##   String Operations_. R package version 1.5.1,
##   <https://CRAN.R-project.org/package=stringr>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {stringr: Simple, Consistent Wrappers for Common String Operations},
##     author = {Hadley Wickham},
##     year = {2023},
##     note = {R package version 1.5.1},
##     url = {https://CRAN.R-project.org/package=stringr},
##   }
```

```r
citation("tidyr")
```

```
## To cite package 'tidyr' in publications use:
##
##   Wickham H, Vaughan D, Girlich M (2024). _tidyr: Tidy Messy Data_. R
##   package version 1.3.1, <https://CRAN.R-project.org/package=tidyr>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {tidyr: Tidy Messy Data},
##     author = {Hadley Wickham and Davis Vaughan and Maximilian Girlich},
##     year = {2024},
##     note = {R package version 1.3.1},
##     url = {https://CRAN.R-project.org/package=tidyr},
##   }
```

```r
citation("readr")
```

```
## To cite package 'readr' in publications use:
##
##   Wickham H, Hester J, Bryan J (2024). _readr: Read Rectangular Text
##   Data_. R package version 2.1.5,
##   <https://CRAN.R-project.org/package=readr>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {readr: Read Rectangular Text Data},
##     author = {Hadley Wickham and Jim Hester and Jennifer Bryan},
##     year = {2024},
##     note = {R package version 2.1.5},
##     url = {https://CRAN.R-project.org/package=readr},
##   }
```