Code ▾

Data Preprocessing

Vu Kieu Chinh

Setup

Hide

```
library(readr)
library(kableExtra)
library(magrittr)
library(readxl)
library(dplyr)
library(tidyr)
library(tidyverse)
library(lubridate)
library(Hmisc)
library(mice)
library(MVN)
library(forecast)
```

Executive Summary

In this report, there are several steps to perform effective data preprocessing to clean, transform, and prepare the data set for analysis. First, this study set up the packages to run the data set and provide a brief overview of the data set descriptions. Second, import all the data sets and merge these data into R markdown. Third, provide an understanding of the data set by examining the original data structure and converting variables to match the data type. Next, the tidy and manipulating data step will be performed. Specifically, the steps to perform include: tidying the column, cleaning data before separating and splitting the column into the separator character on commas, replacing missing values in a column with corresponding values from the other column, and retrieving unique values from a column for detecting incorrect values. Fifth, calculating the consistency between the two columns, “my_score” and “mean_score,” and delivering a distribution of consistency in the “Score” column. Then, handling the missing values and delivering the visualization of the outliers of all numeric variables function will be used to detect outliers and visualize the distribution of transformed values. Finally, the “score” variable was selected for transformation steps to stabilized variance and making “score” variable more suitable for further parametric analyses. Applying various mathematical operations that to try decrease the skewness and convert the distribution into a normal distribution and the distributions of each transformed score variable are visualized by “hist” function to assess the effects of the transformations.

Data

The data used for this study is based on the Anime Dataset 2023, which sourced from the original data set “final_animedataset.csv” on Kaggle website which is located at https://www.kaggle.com/datasets/dbdmobile/myanimelist-dataset (https://www.kaggle.com/datasets/dbdmobile/myanimelist-dataset).
Due to the large size of the original data set, the data set used for this assignment comprises 999 observations and 13 variables, taken from various rows of the original dataset “final_animedataset.csv”. The subset of the data set named “rate” is created by selecting 15 specific users from the original data, including Aries_Yuki, Beibei, Beowulf-, cmoonbeam1, DanaDreams, Evorsor, GothicRainbow, kupoartist, marko8819, Miro112, Peod6, starspawn16, steffy0taku, syanez, and yumenights. The “rate” data set contains the following variables:

- username: [Character] : The username of the user.
- anime_id [numeric] : Unique ID for each anime.
- my_score [numeric]: The rating score given by the user to the anime.
- user_id [numeric]: Unique ID for each user.
- gender [factor]: The gender of the user.
- title [Character]: The title of the anime,
- type[factor]: The type of the anime (e.g., TV series, movie, OVA, etc.).
- source [factor]: The source material of the anime (e.g., manga, light novel, original).
- score [numeric]: The overall score of the anime.
- scored By [numeric]: The number of users who scored the anime.
- rank [numeric]: The rank of the anime based on popularity or other criteria.
- popularity [numeric]: The popularity rank of the anime.
- genre [character] : The genre of the anime

Hide

```
#Import data from excel file "filtered_anime_rate.xlsx"
rate <- read_excel("filtered_anime_rate.xlsx",sheet = "Sheet1")
head(rate,5) #Display the first 5 rows
```

username <chr>	anime_id <dbl>	my_sc… <dbl>	user_id <dbl>	gen… <chr>	title <chr>	t… <chr>	sou… <chr>	sc… <dbl>	scored_by <dbl>
yumenights	1535	0	333839	Female	Death Note	TV	Manga	8.67	1009477
Aries_Yuki	1535	9	120456	Female	Death Note	TV	Manga	8.67	1009477
cmoonbeam1	1535	10	267082	Female	Death Note	TV	Manga	8.67	1009477
steffy0taku	1535	10	258118	Female	Death Note	TV	Manga	8.67	1009477
marko8819	1535	10	378666	Male	Death Note	TV	Manga	8.67	1009477

5 rows | 1-10 of 13 columns

The “read_excel” function serves the purpose of importing data from the “filtered_anime_rate.xlsx” file and then assigns this data set to the “rate” variable. The “head” function is utilized to show the first 5 rows for the “rate” data set.

The second data set is a subset derived from the “user_detail_2023.csv” dataset, consisting of 731,291 observations, which was sourced from the same website. Additionally, this data set includes 5 out of the 16 variables present in the original data set. These 5 variables are:

- Mal ID [numeric] : Unique ID for each user.
- Birthday [date]: The birthday of the user (in ISO format).
- Location [character]: The location or country of the user.
- Joined [date]: The date when the user joined the platform (in ISO format).

- Mean Score [numeric]: The average score given by the user to the anime they have watched.

Hide

```
#Import data from csv file "users-details-2023.csv"
user <- read_csv("users-details-2023.csv")
```

```
Rows: 731290 Columns: 16
── Column specification ────────────────────────────────────────────────────────────
Delimiter: ","
chr   (3): Username, Gender, Location
dbl  (11): Mal ID, Days Watched, Mean Score, Watching, Completed, On Hold, Droppe...
dttm  (2): Birthday, Joined

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.
```

Hide

```
#Refine "user" data set
user <- user %>% select("Mal ID":"Mean Score") %>% select(-Username,-Gender,-`Days Watched`)
head(user,5) #Display the first 5 rows
```

Mal ID <dbl>	Birthday <S3: POSIXct>	Location <chr>	Joined <S3: POSIXct>	Mean Score <dbl>
1	1985-03-04	California	2004-11-05	7.37
3	<NA>	Oslo, Norway	2004-11-11	7.34
4	<NA>	Melbourne, Australia	2004-11-13	6.68
9	<NA>	NA	2004-12-05	7.71
18	<NA>	NA	2005-01-03	6.27

5 rows

The “read_csv” function serves the purpose of importing data from the “users-details-2023.csv” file and then assigns this data set to the “user” variable. The “select” function is used to subset the data set appropriately. It selects columns from “Mal ID” to “Mean Score” and then removes the “Username,” “Gender,” and “Days Watched” columns from the previously selected range. The “head” function is utilized to show the first 5 rows for the “user” data set.

Hide

```
#Merge two data sets
anime <- rate %>% left_join(user, c("user_id" = "Mal ID"))
head(anime,5) #Display the first 5 rows
```

username <chr>	anime_id <dbl>	my_sc... <dbl>	user_id <dbl>	gen... <chr>	title <chr>	t... <chr>	sou... <chr>	sc... <dbl>	scored_by <dbl>
yumenights	1535	0	333839	Female	Death Note	TV	Manga	8.67	1009477
Aries_Yuki	1535	9	120456	Female	Death Note	TV	Manga	8.67	1009477

username	anime_id	my_sc...	user_id	gen...	title	t...	sou...	sc...	scored_by
<chr>	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>
cmoonbeam1	1535	10	267082	Female	Death Note	TV	Manga	8.67	1009477
steffy0taku	1535	10	258118	Female	Death Note	TV	Manga	8.67	1009477
marko8819	1535	10	378666	Male	Death Note	TV	Manga	8.67	1009477

In particular, the merging will be done by joining two data sets which are “rate” and “user” and the matching columns are taken from the “user_id” from “rate” and the “Mal ID” from “user” data set. By using left_join, the code above ensure that all rows from the “rate” data set are retained, and user’s detail are added where available. This approach will ensure that the integrity of the ratings data is maintained.

Understand

Hide

```
#Check the structure of all columns in data set
str(anime)
```

```
tibble [998 × 17] (S3: tbl_df/tbl/data.frame)
 $ username  : chr [1:998] "yumenights" "Aries_Yuki" "cmoonbeam1" "steffy0taku" ...
 $ anime_id  : num [1:998] 1535 1535 1535 1535 1535 ...
 $ my_score  : num [1:998] 0 9 10 10 10 10 8 10 0 9 ...
 $ user_id   : num [1:998] 333839 120456 267082 258118 378666 ...
 $ gender    : chr [1:998] "Female" "Female" "Female" "Female" ...
 $ title     : chr [1:998] "Death Note" "Death Note" "Death Note" "Death Note" ...
 $ type      : chr [1:998] "TV" "TV" "TV" "TV" ...
 $ source    : chr [1:998] "Manga" "Manga" "Manga" "Manga" ...
 $ score     : num [1:998] 8.67 8.67 8.67 8.67 8.67 8.67 8.67 8.67 8.49 8.49 ...
 $ scored_by : num [1:998] 1009477 1009477 1009477 1009477 1009477 ...
 $ rank      : num [1:998] 51 51 51 51 51 51 51 51 110 110 ...
 $ popularity: num [1:998] 1 1 1 1 1 1 1 1 2 2 ...
 $ genre     : chr [1:998] "Mystery,Police,Psychological,Supernatural,Thriller,Shounen" "Myst
ery,Police,Psychological,Supernatural,Thriller,Shounen" "Mystery,Police,Psychological,Superna
tural,Thriller,Shounen" "Mystery,Police,Psychological,Supernatural,Thriller,Shounen" ...
 $ Birthday  : POSIXct[1:998], format: "1995-05-02" "1993-02-03" ...
 $ Location  : chr [1:998] "Rio de Janeiro, Brazil" "Forks, Washington" "New Brunswick, Canad
a" "san mateo,rizal" ...
 $ Joined    : POSIXct[1:998], format: "2010-05-27" "2008-12-04" ...
 $ Mean Score: num [1:998] 8.09 8.17 8.59 7.48 7.67 7.81 7.06 7.33 8.09 7.48 ...
```

The data types are detected by using the “str” function. There are five variables from the output above that are not in the correct data type.

Hide

```
#Convert data type to factor (with labelled and ordered)
anime$gender <- factor(anime$gender)
anime$type <- factor(anime$type,
                    levels=c("Movie","Music","ONA","OVA","Special", "TV"),
                    labels = c("Movie","Music","ONA","OVA","TVSpecial", "TV"))
anime$source <- factor(anime$source,
                      levels=c("4-koma manga","Book","Game", "Light novel","Manga", "Novel",
                               "Original","Other","Unknown", "Visual novel","Web manga"),
                      labels=c("Manga","Book","Game", "Novel","Manga","Novel","Original",
                               "Other","Unknown","Novel","Manga"))
#Convert data type to date
anime$Birthday <- as.Date(anime$Birthday)
anime$Joined <- as.Date(anime$Joined)
#Recheck the structure
str(anime[ , c(5, 7, 8, 14, 16)])
```

```
tibble [998 × 5] (S3: tbl_df/tbl/data.frame)
 $ gender  : Factor w/ 2 levels "Female","Male": 1 1 1 1 2 2 2 2 1 1 ...
 $ type    : Factor w/ 6 levels "Movie","Music",..: 6 6 6 6 6 6 6 6 6 6 ...
 $ source  : Factor w/ 7 levels "Manga","Book",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Birthday: Date[1:998], format: "1995-05-02" "1993-02-03" ...
 $ Joined  : Date[1:998], format: "2010-05-27" "2008-12-04" ...
```

Hide

```
head(anime,5) #Display the first 5 rows
```

username <chr>	anime_id <dbl>	my_sc... <dbl>	user_id <dbl>	gen... <fctr>	title <chr>	t... <fctr>	sou... <fctr>	sc... <dbl>	scored_by <dbl>
yumenights	1535	0	333839	Female	Death Note	TV	Manga	8.67	1009477
Aries_Yuki	1535	9	120456	Female	Death Note	TV	Manga	8.67	1009477
cmoonbeam1	1535	10	267082	Female	Death Note	TV	Manga	8.67	1009477
steffy0taku	1535	10	258118	Female	Death Note	TV	Manga	8.67	1009477
marko8819	1535	10	378666	Male	Death Note	TV	Manga	8.67	1009477

5 rows | 1-10 of 17 columns

Therefore, the " factor" function is utilized to convert "gender", "type" and "source" to factor data type. Moreover, "levels" and "labels" composition are also applied to establish the level and labeling of these factors appropriately. According to "type" variable, "Special" is relabeled to "TVSpecial" for clarity, while the others remain unchanged. According to "source" variable, "4-koma manga," "Manga," and "Web manga" are all relabeled as "Manga," while "Light novel," "Novel," and "Visual novel" are all relabeled as "Novel". The"as.date" function is utilized to convert "birthday" and "joined" to date data type. The "str" function is used once again to double check all the variables that were converted (columns 5, 7, 8, 14 and 16). The "head" function is used to display first 5 rows of the "anime" data set.

Tidy & Manipulate Data I

Hide

```
#Tidy genre column
head(anime$genre,5)
```

```
[1] "Mystery,Police,Psychological,Supernatural,Thriller,Shounen"
[2] "Mystery,Police,Psychological,Supernatural,Thriller,Shounen"
[3] "Mystery,Police,Psychological,Supernatural,Thriller,Shounen"
[4] "Mystery,Police,Psychological,Supernatural,Thriller,Shounen"
[5] "Mystery,Police,Psychological,Supernatural,Thriller,Shounen"
```

The “genre” variable is considered as an untidy variable because values in this column contains multiple values separated by commas. In this case, every individual genre should ideally be treated as an observation related to a particular item.

Hide

```
#Clean data before separating
anime <- anime %>% mutate(genre = str_trim(genre))%>% #trim white space
          mutate(genre = str_replace_all(genre, "\\s*,\\s*", ",")) # Remove spaces around com
mas
#Split genre into the separator character on commas
anime_tidy <- anime %>%
  separate(genre, into = c("genre1","genre2","genre3","genre4","genre5",
                           "genre6","genre7","genre8","genre9","genre10"), sep= ",")
```

Therefore, the “separate” function is used for splitting the column into multiple rows from “genre1” to “genre10” based on the comma delimiter. However, before separating, the data need to be cleaned. Firstly, to remove any leading or trailing white space from each genre entry, “str_trim(genre)” is utilized to remove leading and trailing whitespace from each entry in the genre column. Then, the “mutate” function creates a new genre column with the trimmed values, replacing the old genre column.

Hide

```
#Tidy Location column
head(anime_tidy$Location,5)
```

```
[1] "Rio de Janeiro, Brazil"         "Forks, Washington"
[3] "New Brunswick, Canada"          "san mateo,rizal"
[5] "Mostar, Bosnia and Herzegovina"
```

The “head” function is used to display 5 rows of the “Location”. Similar to the “genre” variable, the “Location” variable is also considered untidy because it contains multiple values.

Hide

```
#Clean data before tidying
anime_tidy <- anime_tidy %>% mutate(Location = str_trim(Location), # Trim white space
    Location = str_replace_all(Location,"\\s*,\\s*", ","), #Remove spaces around commas
    Location = toupper(Location)) #Convert to upper case for consistency
#Split Location into the separator character "City" & "Country" on commas
anime_tidy <- anime_tidy %>% separate(Location,into = c("City","Country"), sep = ",")
```

A similar approach with "genre" is applied to clean "Location" column. In addition, "toupper" function is employed to convert values to uppercase for consistency. Then, "separate" function splits the Location column into two new columns, City and Country, based on the comma separator.

Hide

```
#Replace missing values in Country column with corresponding values from the City column
anime_tidy$Country <- ifelse(is.na(anime_tidy$Country), anime_tidy$City, anime_tidy$Country)
#Keep only Country column for data set
anime_tidy <- anime_tidy %>% select(-City)
#Retrieve unique values from Country column for detecting incorrect values
anime_tidy %>% distinct(Country, .keep_all = FALSE)%>% pull(Country) %>% cat(.,sep = ", ")
```

```
BRAZIL, WASHINGTON, CANADA, RIZAL, BOSNIA AND HERZEGOVINA, SWEDEN, LOUISIANA, QUEBEC, THAILAN
D, QUEENSLAND, GA, O_O WATCHING YOUR EVERY MOVEMENT.....HEHEHE, BRASIL, UK, KENTUCKY
```

However, due to the incomplete data in the "Location" column, where not all rows specify the "City," this study will only focus on "Country". Therefore, the "ifelse" function is used as a conditional element condition to replace missing values in the "Country" column with corresponding values from the "City" column. After that, the "City" column is removed by using "select" function. The "distinct" function is used to list the unique data contained in the "Country" column to detect incorrect or non-standard values. Finally, the result is extracted as a vector using the "pull" function and then printed, separated by commas.

Hide

```
#Replace non-standard entries
anime_tidy$Country <- ifelse(anime_tidy$Country ==
        "o_o watching your every movement.....hehehe", "Unknown", anime_tidy$Country)
#Country Adjustment
anime_tidy$Country <- ifelse(anime_tidy$Country == "WASHINGTON"| anime_tidy$Country == "GA"|
                          anime_tidy$Country == "KENTUCKY"|anime_tidy$Country == "LOUISIANA",
"US",
                      ifelse(anime_tidy$Country == "BRASIL", "BRAZIL",
                      ifelse(anime_tidy$Country== "QUEENSLAND", "AUS",
                      ifelse(anime_tidy$Country== "RIZAL", "PHILIPPINES",
                      ifelse(anime_tidy$Country== "QUEBEC", "CANADA", anime_tidy$Country)))))
```

It can be seen from the result above, there are some non-standard and incorrect values in "Country" column. For non-standard values, the "ifelse" function is used to replace them with a more appropriate value, "Unknown". For incorrect values, the "ifelse" function adjusts them to standardized country names.

Hide

```
head(anime_tidy[ , c(13:22,24)]) #Display the first 5 rows of tidied columns
```

genre1 <chr>	gen… <chr>	genre3 <chr>	genre4 <chr>	genre5 <chr>	genre6 <chr>	gen… <chr>	gen… <chr>	gen… <chr>	genre <chr>
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	NA	NA	NA	NA
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	NA	NA	NA	NA
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	NA	NA	NA	NA
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	NA	NA	NA	NA

genre1 <chr>	gen… <chr>	genre3 <chr>	genre4 <chr>	genre5 <chr>	genre6 <chr>	gen… <chr>	gen… <chr>	gen… <chr>	genre <chr>
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	NA	NA	NA	NA
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	NA	NA	NA	NA

The "head" function is used to display first 5 rows of specific tidied columns from the anime_tidy data set.

Tidy & Manipulate Data II

In this section, this study will focus on analyzing user behavior by introducing a derived variable, "Consistency_Score". This variable is calculated as the difference between each user's rating "my_score" and their average rating "Mean Score". The objective is to quantify the consistency of user ratings, which might be useful for refining recommendation systems. By mutating the data set ("mutate" function) to include this new metric, the below code then display the first five entries through "head" function to verify the integration.

Hide

```
#Calculate the consistency of a user's ratings against their average rating
anime_tidy <- anime_tidy %>% mutate(Consistency_Score = my_score - `Mean Score`)
head(anime_tidy,5) #Display the first 5 rows
```

username <chr>	anime_id <dbl>	my_sc… <dbl>	user_id <dbl>	gen… <fctr>	title <chr>	t… <fctr>	sou… <fctr>	sc… <dbl>	scored_by <dbl>
yumenights	1535	0	333839	Female	Death Note	TV	Manga	8.67	1009477
Aries_Yuki	1535	9	120456	Female	Death Note	TV	Manga	8.67	1009477
cmoonbeam1	1535	10	267082	Female	Death Note	TV	Manga	8.67	1009477
steffy0taku	1535	10	258118	Female	Death Note	TV	Manga	8.67	1009477
marko8819	1535	10	378666	Male	Death Note	TV	Manga	8.67	1009477

5 rows | 1-10 of 27 columns

Hide

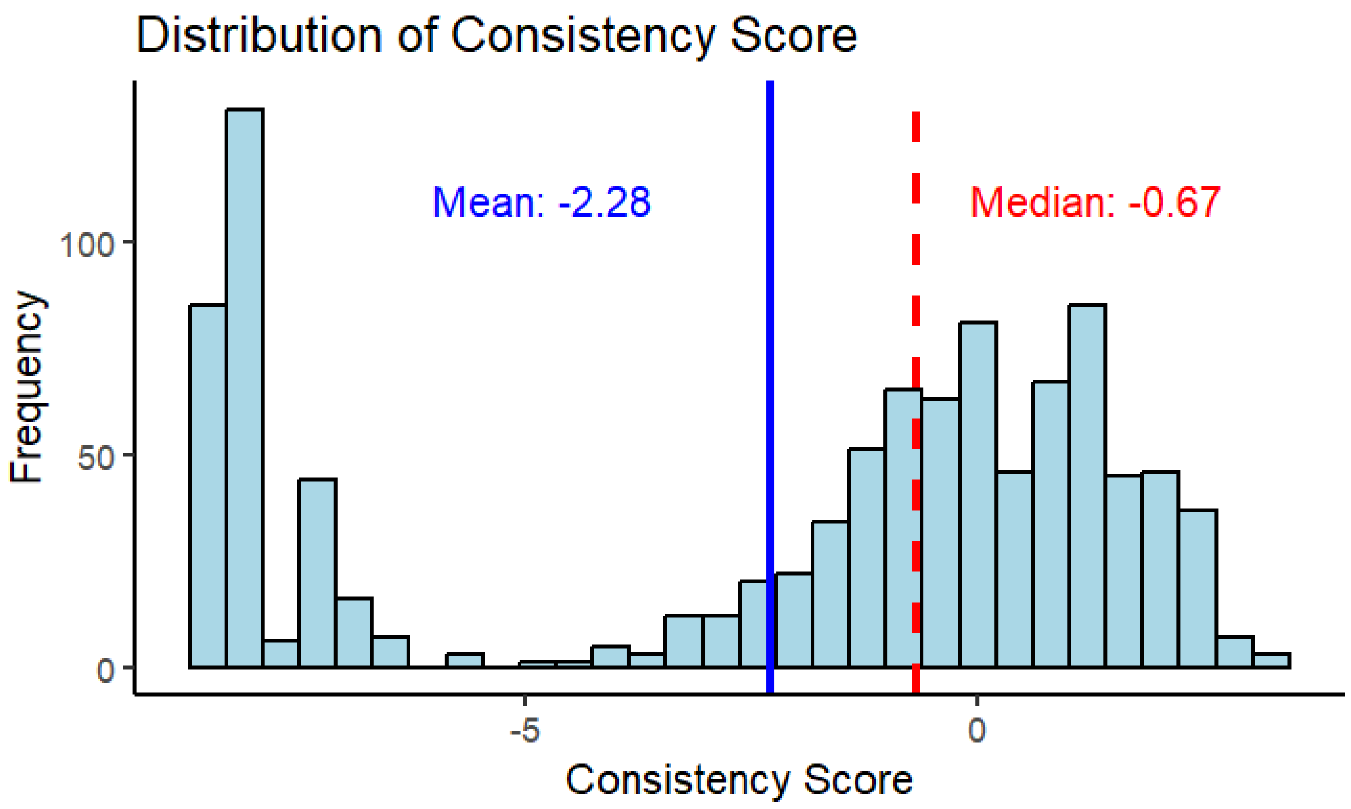
```
#Summary Statistics of Consistency Score
summary(anime_tidy$Consistency_Score)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 -8.630  -7.330  -0.670  -2.276   0.870   3.110 
```

The "summary" function is used to provide a descriptive statistic for the distribution of "Consistency_score".

Hide

```
#Distribution of Consistency Score
ggplot(anime_tidy, aes(x = Consistency_Score)) +
  geom_histogram(bins = 30, fill = "lightblue", color = "black") +
  labs(title = "Distribution of Consistency Score",x="Consistency Score", y="Frequency")+
  theme_classic()+
geom_vline(aes(xintercept=mean(anime_tidy$Consistency_Score,na.rm=TRUE)),#Add mean line to th
e plot
color="blue", linetype="solid", size=1) + annotate("text", x = -4.8, y = 110,
label = paste("Mean:", round(mean(anime_tidy$Consistency_Score,na.rm=TRUE), 2)),color = "blu
e")+
geom_vline(aes(xintercept=median(anime_tidy$Consistency_Score,na.rm=TRUE)),
           #Add median line to the plot
color="red", linetype="dashed", size=1)+ annotate("text", x = 1.3, y = 110,
  label = paste("Median:", round(median(anime_tidy$Consistency_Score,na.rm=TRUE), 2)),color =
"red")
```



The histogram is distributed by “ggplot” function to give a clear view of distribution for the user rating behaviors and find any hidden exceptions or discrepancies in the data. Moreover, “geom_vline” and “annonate” functions are used to add line and description at median and mean value on the plot. According to the distribution, the absence of a bell-shaped curve on the histogram suggests that the data does not follow a normal distribution. A bimodal distribution is indicated with peaks around -7 and 0, suggesting two distinct clusters of user rating behavior. This result mean that users with highly negative consistency scores may have a tendency to be more critical or have a specific bias in rating animes. The summary statistics reveal a mean “Consistency_score” of -2.28 and a median of -0.67, with the mean being lower, indicating a left-skewed distribution. This suggests that there are more users with negative deviations, which pulls the mean down.

Scan I

Hide

```
#Detect & handle missing value
colSums(is.na(anime_tidy))
```

```
        username         anime_id         my_score          user_id 
               0                0                0                0 
          gender            title             type           source 
               0                0                0                0 
           score        scored_by             rank       popularity 
               0                0                1                0 
          genre1           genre2           genre3           genre4 
               0               10               55              194 
          genre5           genre6           genre7           genre8 
             420              622              844              944 
          genre9          genre10         Birthday          Country 
             980              988              136                0 
          Joined       Mean Score Consistency_Score 
               0                0                0 
```

The “colSums()” function is utilized to compute the total missing values in each column of the anime_tidy data set. As can be seen in the table, missing values are detected in “rank”, “genre2” to “genre10” and “Birthday” columns.

Hide

```
#Impute missing values in rank column by median
shapiro.test(anime_tidy$rank) #Test for normal distribution
```

```

	Shapiro-Wilk normality test

data:  anime_tidy$rank
W = 0.78406, p-value < 2.2e-16
```

Hide

```
anime_tidy$rank <-impute (anime_tidy$rank, fun = median)
head(anime_tidy[c("anime_id", "rank")],5)
```

anime_id <dbl>	rank <dbl>
1535	51
1535	51
1535	51
1535	51
1535	51

5 rows

According to “rank” column, this study apply median to replace the missing values by using “impute” function from “Hmisc” package. This technique is applied because “rank” column is numerical value and and its distribution is not in normal form (based on p_value from “shapiro.test”). Using the median will provide a robust measure that is not affected by outliers, is less likely to be skewed by extremely high or low values, and is more representative of the majority of the data. The “head” function is used to recheck the imputation.

Hide

```
#Impute missing values in genre column by a constant
anime_tidy <- anime_tidy %>% mutate(across(starts_with("genre"),
                                          ~ifelse(is.na(.), "No Additional", .)))
head(anime_tidy[c("genre1","genre2","genre3","genre4","genre5",
                  "genre6","genre7","genre8","genre9","genre10")],5)
```

genre1 <chr>	gen… <chr>	genre3 <chr>	genre4 <chr>	genre5 <chr>	genre6 <chr>	genre7 <chr>	genre8 <chr>
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	No Additional	No Additiona
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	No Additional	No Additiona
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	No Additional	No Additiona
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	No Additional	No Additiona
Mystery	Police	Psychological	Supernatural	Thriller	Shounen	No Additional	No Additiona

5 rows | 1-8 of 10 columns

The “impute” function is used to impute missing values in “genre” column by a constant “No Additional” and the “ifelse” function is to identify whether the value in each cell is “NA” or not, if not , it will keep the original data value. This technique is applied because “genre” contain categorical data and missing values in this likely indicate the absence of additional genres for an anime. Therefore, it will be more informative to label missing values in this column explicitly. The “head” function is used to recheck the imputation.

Hide

```
#Impute missing values in Birthday column by predictive models (mice)
anime_tidy <- anime_tidy %>%
  mutate(Birthday_numeric = as.numeric(Birthday),
         Joined_numeric = as.numeric(Joined)) #Convert dates to numeric
temp_data <- mice(anime_tidy[, c("Birthday_numeric", "Joined_numeric")],
                  m=5, method='pmm', seed=681) #Multiple imputation to fill in missing values
```

```

 iter imp variable
  1   1  Birthday_numeric
  1   2  Birthday_numeric
  1   3  Birthday_numeric
  1   4  Birthday_numeric
  1   5  Birthday_numeric
  2   1  Birthday_numeric
  2   2  Birthday_numeric
  2   3  Birthday_numeric
  2   4  Birthday_numeric
  2   5  Birthday_numeric
  3   1  Birthday_numeric
  3   2  Birthday_numeric
  3   3  Birthday_numeric
  3   4  Birthday_numeric
  3   5  Birthday_numeric
  4   1  Birthday_numeric
  4   2  Birthday_numeric
  4   3  Birthday_numeric
  4   4  Birthday_numeric
  4   5  Birthday_numeric
  5   1  Birthday_numeric
  5   2  Birthday_numeric
  5   3  Birthday_numeric
  5   4  Birthday_numeric
  5   5  Birthday_numeric
```

Hide

```
completed_data <- complete(temp_data, 1) #Extract a single complete data set from a mice obje
ct
anime_tidy$Birthday <- as.Date(completed_data$Birthday_numeric)
#Convert back the imputed numeric values to date
anime_tidy <- anime_tidy %>% select(-(Birthday_numeric:Joined_numeric))
#Remove temporary numeric column
head(anime_tidy[c("user_id", "Birthday")],5)
```

user_id <dbl>	Birthday <date>
333839	1995-05-02
120456	1993-02-03
267082	1987-08-04
258118	1998-05-22
378666	1988-10-21

5 rows

According to “Birthday”, this column contains date values and missing values can be imputed using predictive modeling. The “mice” (Multiple Imputation by Chained Equations) package is used to perform multiple imputations and predict the missing dates based on other related columns (for this study, it is Joined column).

This technique is chosen to be applied because it will provide more accurate imputation for dates by predicting with the consideration of the relationship between Birthday and Joined columns. To employ “mice” method, first, “Birthday” and “Joined” date columns are converted to numeric format by “as.numeric” function. This conversion is necessary because the “mice” package operates on numeric data. Then, “mice” function is applied to perform multiple imputations on the numeric columns “Birthday_numeric” and “Joined_numeric”. The method=‘pmm’ specifies Predictive Mean Matching as the imputation method, which is suitable for numeric data. The “m=5” argument indicates that five imputed data sets are to be created. The “seed=681” ensures reproducibility of the results. From the result generated, a single data set is extracted by using the “complete” function. “1” specifies that the first of the five imputed data sets should be used. After that, “Birthday_numeric” is converted back to date format using the “as.Date” function and imputed into the “anime_tidy” data set. Finally, The temporary columns “Birthday_numeric” and “Joined_numeric” are removed. The “head” function is used to recheck the imputation.

Hide

```
#Recheck
colSums(is.na(anime_tidy[ , c(11, 14:23)]))
```

```
    rank   genre2   genre3   genre4   genre5   genre6   genre7   genre8   genre9 
       0        0        0        0        0        0        0        0        0 
 genre10 Birthday 
       0        0 
```

Finally, the “colSums()” function is used once again to double check for any remaining missing values in the data set.

Hide

```
#Detect & handle special Value
sapply(anime_tidy,function(x) sum(is.infinite(x))) #Check for infinite values
```

```
         username          anime_id          my_score           user_id 
                0                 0                 0                 0 
           gender             title              type            source 
                0                 0                 0                 0 
            score         scored_by              rank        popularity 
                0                 0                 0                 0 
           genre1            genre2            genre3            genre4 
                0                 0                 0                 0 
           genre5            genre6            genre7            genre8 
                0                 0                 0                 0 
           genre9           genre10          Birthday           Country 
                0                 0                 0                 0 
           Joined        Mean Score Consistency_Score 
                0                 0                 0 
```

Hide

```
sapply(anime_tidy,function(x) sum(is.nan(x))) #Check for NaN values
```

```
        username         anime_id         my_score          user_id 
               0                0                0                0 
          gender            title             type           source 
               0                0                0                0 
           score        scored_by             rank       popularity 
               0                0                0                0 
          genre1           genre2           genre3           genre4 
               0                0                0                0 
          genre5           genre6           genre7           genre8 
               0                0                0                0 
          genre9          genre10         Birthday          Country 
               0                0                0                0 
          Joined       Mean Score Consistency_Score 
               0                0                0 
```

To detect & handle special Value, the “sapply” function is utilized along with the “is.infinite” function to detect infinite values and utilized along with the “is.nan” function to check for NaN values in the data set. As can be seen from the result, there are no infinite or NaN values in “anime_tidy”.

Hide

```
#Detect & handle obvious inconsistencies or errors
#Check limit of numeric & ordinal columns
summary(anime_tidy[, c(3, 9:12,26:27)])
```

```

 1 values imputed to 732

    my_score          score          scored_by            rank        popularity    
 Min.   : 0.000   Min.   :5.530   Min.   :    263   Min.   :   1   Min.   :   1.0  
 1st Qu.: 0.000   1st Qu.:7.433   1st Qu.:  29504   1st Qu.: 225   1st Qu.: 126.8  
 Median : 7.000   Median :7.880   Median :  76502   Median : 732   Median : 560.0  
 Mean   : 5.547   Mean   :7.851   Mean   : 155044   Mean   :1299   Mean   :1017.6  
 3rd Qu.: 9.000   3rd Qu.:8.320   3rd Qu.: 205925   3rd Qu.:1852   3rd Qu.:1379.5  
 Max.   :10.000   Max.   :9.250   Max.   :1009477   Max.   :8321   Max.   :9191.0  
   Mean Score    Consistency_Score
 Min.   :6.890   Min.   :-8.630   
 1st Qu.:7.330   1st Qu.:-7.330   
 Median :7.810   Median :-0.670   
 Mean   :7.823   Mean   :-2.276   
 3rd Qu.:8.250   3rd Qu.: 0.870   
 Max.   :8.670   Max.   : 3.110   
```

Hide

```
#Identify potential data entry errors in date columns
anime_tidy %>%
  summarise(Future_Birthdays=sum(Birthday>Sys.Date(),na.rm = TRUE),
            #Is Birthday in the future?
        Joined_Birthdays=sum(Birthday>Joined,na.rm = TRUE))
```

Future_Birthdays <int>	Joined_Birthdays <int>
0	0

1 row

Hide

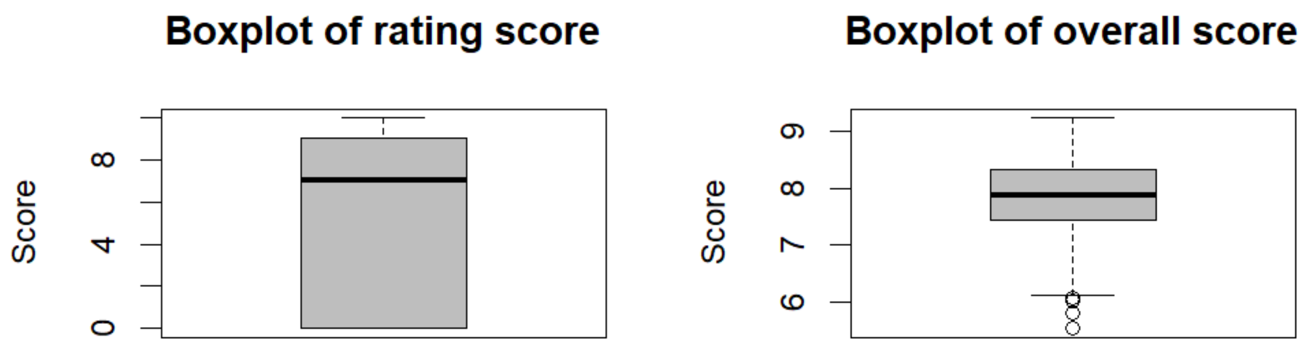
```
#Is Birthday occurring after the joined date?
```

To detect & handle obvious inconsistencies or errors, the “summary” function is used to detect limit of numeric & ordinal variables. In particular, “my_score”, “score” and “mean score” were checked to ensure that their range from 0 to 10, while “scored_by”, “rank” and “popularity” need to be greater than 0. Additionally, the above code checked the date columns for logical inconsistencies, identifying any future Birthday values and whether the Birthday occurred after the Joined date. The detection was performed using the “summarise” function to check these errors by conditional replacement with the “ifelse” function. From the result, the data used for analysis is both accurate and reliable.

Scan II

Hide

```
#Visualize the outliers of all numeric variables
par(mfrow = c(1, 2))
anime_tidy$my_score%>%boxplot(main="Boxplot of rating score",ylab="Score",col="grey")
anime_tidy$score%>%boxplot(main="Boxplot of overall score",ylab="Score",col="grey")
```

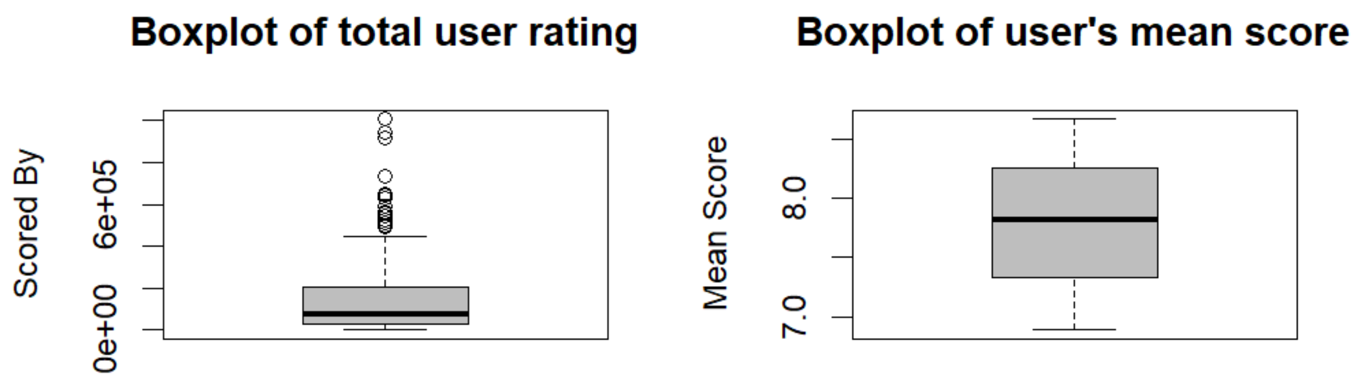


Hide

```
anime_tidy$scored_by%>%boxplot(main="Boxplot of total user rating",ylab="Scored By",col="gre
y")
anime_tidy$`Mean Score`%>%boxplot(main="Boxplot of user's mean score",ylab="Mean Score",col
="grey")
```

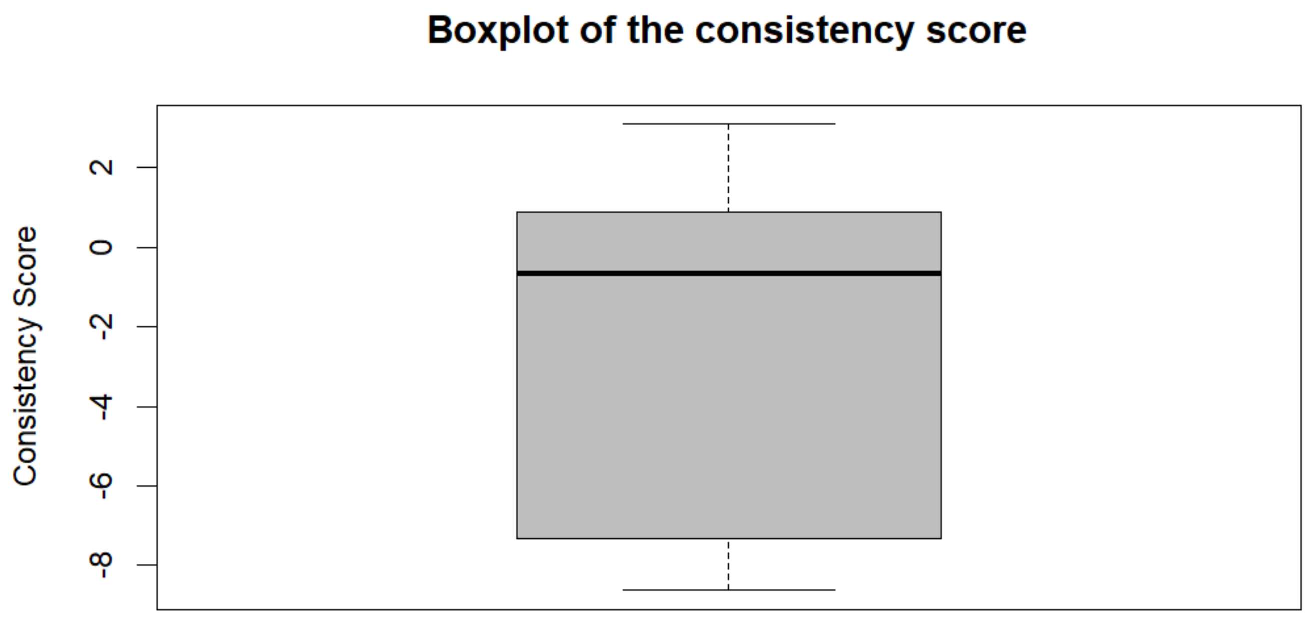
Hide

```
par(mfrow = c(1, 1))
```



Hide

```
anime_tidy$Consistency_Score%>%boxplot(main="Boxplot of the consistency score",
                                       ylab="Consistency Score",col="grey")
```



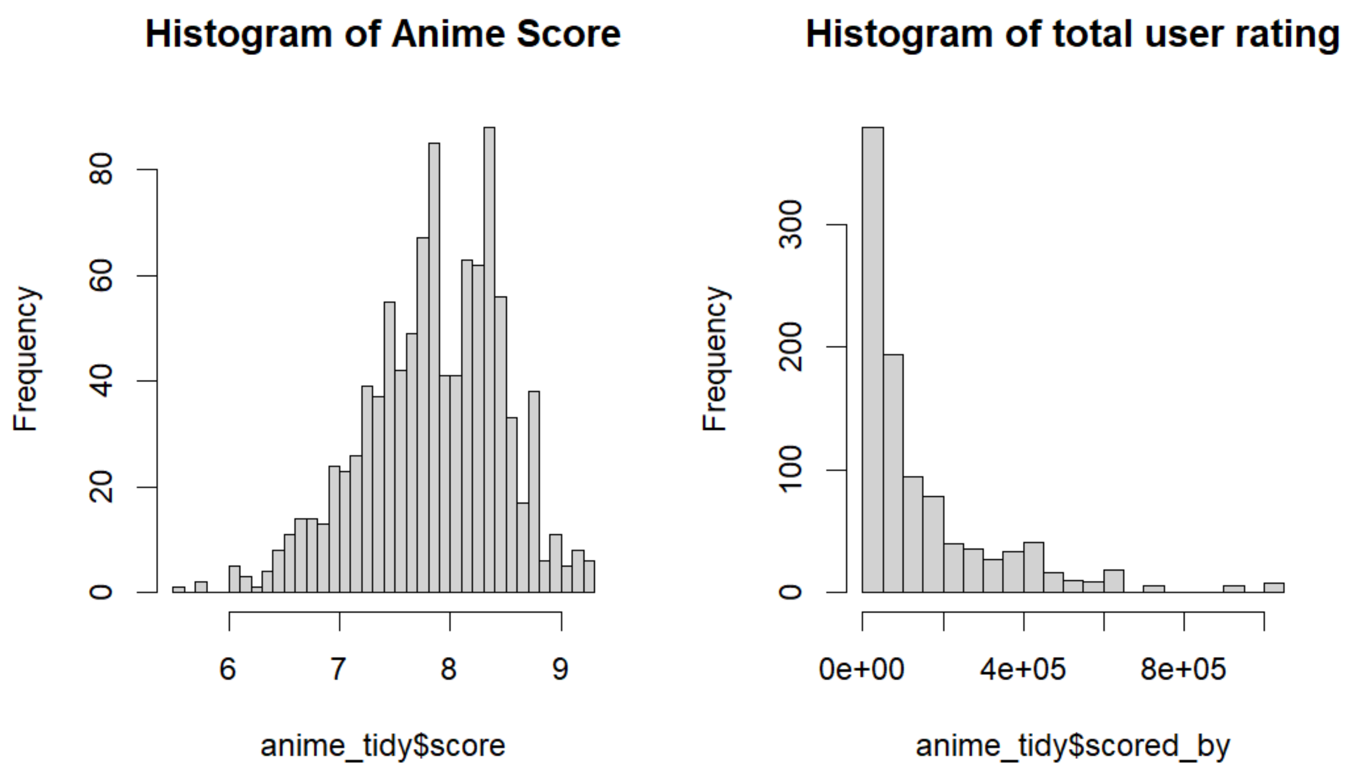
Boxplots by “boxplot” fuction are used to detect outliers in all the numeric variables in data set based on the “Tukey’s method of outlier detection”. The variables analyzed include “my_score”, “score”, “scored_by”, “Mean Score” and “Consistency_Score” (“rank” and “popularity” are ordinal). Tukey’s method was chosen because it is a nonparametric technique that effectively identifies outliers in non-symmetric or non-normal data distributions, which is characteristic of all analyzed data distributions in this step. From the output, it is clear that “score” and “scored_by” variables seem to have many outliers.

Hide

```
#Histograms and normality tests of two variables that have outliers
par(mfrow = c(1, 2))
hist(anime_tidy$score,main = "Histogram of Anime Score", breaks = 30)
hist(anime_tidy$scored_by,main ="Histogram of total user rating", breaks = 30)
```

Hide

```
par(mfrow = c(1, 1))
```



Histograms by “hist” function are used to provide a visual representation of the distribution of two variables with outliers. From the two histograms, it is evident that while “score” left skew, “scored_by” right skew and further supports the use of Tukey’s method for outlier detection.

Moving to handling outliers, with “score” variable represents the overall score of an anime. Intentional errors can occur in this case when some users rate an anime 0 out of spite or bias. Humans have a psychological tendency known as negativity bias, where negative experiences and impressions have a more significant impact than positive ones. When users dislike an anime, they might rate it as low as possible. These extremely low ratings might lead to intentional outliers that significantly skew the dataset and misrepresent the true average ratings. Additionally, sampling errors may arise because the data is a subset of a much larger dataset, and the sample size is considerably smaller. These sampling errors can result in outliers that do not accurately reflect the broader population’s opinions. Therefore, it is essential to handle these outliers for “score” variable.

However, according to “scored_by” variable, extremely high values can provide valuable insights about real user engagement. High “scored_by” values might indicate highly popular animes that attract a large number of ratings or indicate a recent promotion. Moreover, in real world, user engagement metrics often exhibit right-skewed where a few items have significantly higher engagement than the rest. Therefore, this study retains outliers of “scored_by” column to preserve the relistic patterns and behaviors.

Hide

```
#Define a function to cap the values outside the limits
cap <- function(x){
  quantiles <- quantile( x, c(.05, 0.25, 0.75, .95 ) )
  x[ x < quantiles[2] - 1.5*IQR(x) ] <- quantiles[1]
  x[ x > quantiles[3] + 1.5*IQR(x) ] <- quantiles[4]
  x}
#Apply "cap" function to a data frame
anime_score <- anime_tidy %>% select("score")
score_capped <- as.data.frame(sapply(anime_score, FUN=cap))
```

To deal with outliers of “score” variable, Capping or winsorising method is applied. This technique replaces outliers with the nearest non-outliers keeping all the information intact and preserving the data sets completeness. Limiting adjustments are made to the extreme values lessening their impact without completely removing them. This approach is also particularly beneficial, for data sets that do not follow a distribution as outliers have the potential to greatly distort the findings.

In order to cap the outliers this study uses a user-defined function as above (taken from: Stackoverflow). This study use the “sapply” function to apply the “cap” function to each column in the “anime_score” data frame, which is converted using the “as.data.frame” function from a subset data set of “anime_tidy” data set.

Hide

```
#See descriptive statistics
summary(anime_score)
```

```
     score
 Min.   :5.530
 1st Qu.:7.433
 Median :7.880
 Mean   :7.851
 3rd Qu.:8.320
 Max.   :9.250
```

Hide

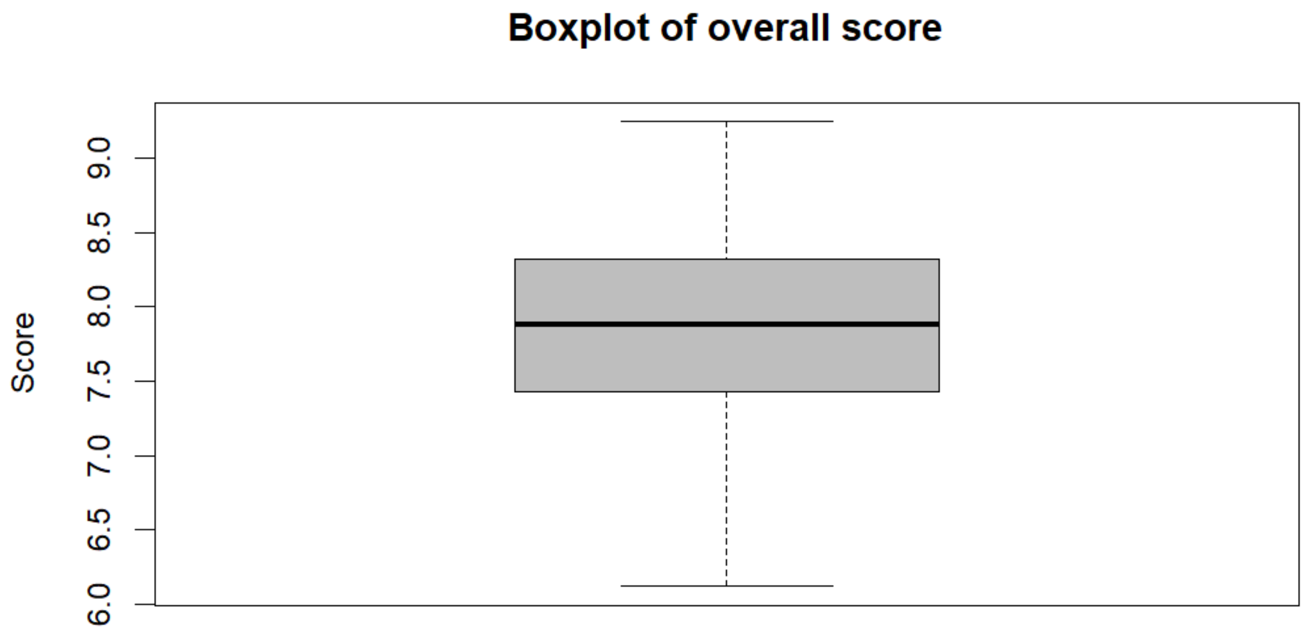
```
#Check summary statistics after apply "cap" function
summary(score_capped)
```

```
     score
 Min.   :6.120
 1st Qu.:7.433
 Median :7.880
 Mean   :7.858
 3rd Qu.:8.320
 Max.   :9.250
```

Then, the “summary” function is used to provide a comparision between to descriptive statistics for the data frame before and after capping. From the result, the minimum value increased from 5.530 to 6.120 after capping, which means the lowest outliers were adjusted closer to the lower bound. The first quartile, median, and third quartile values remained the same, which suggest that the central tendency were not affected. The mean value increased slightly from 7.851 to 7.858, showing a minor adjustment after capping. The maximum value remained the same.

Hide

```
#Check boxplots after apply"cap" function
score_capped$score %>%  boxplot(main="Boxplot of overall score", ylab="Score", col = "grey")
```



A boxplot is used to recheck the outliers of the score variable after capping. The visualization confirmed that no outliers remained, indicating that the capping was highly effective.

Transform

To fulfill the requirement of this step, the “score” variable was selected for transformation due to its critical role in anime quality, representing overall user ratings. Ensuring its distribution is appropriate for analysis is essential for further analysis.

Hide

```
sqr_score <- anime_tidy$score^2 #Square
cube_score <- anime_tidy$score^3 #Cube
boxcox_score<- BoxCox(anime_tidy$score,lambda = "auto") #Box-cox
log_score <- log10(anime_tidy$score) #Log
ln_score <- log(anime_tidy$score) #Ln
sqrt_score <- sqrt(anime_tidy$score) #Sqrt
cubrt_score <- anime_tidy$score^(1/3) #Cuberoot
rec_score <- 1/anime_tidy$score #Recipricol
rec_sq_score <- 1/(anime_tidy$score)^2 #Recipricol square
```

Various mathematical operations are applied to try decrease the skewness and convert the distribution into a normal distribution. These includes Square by “^2”, Cube by “^3”, Box-Cox by “BoxCox” and setting lambda = “auto”, the function automatically determines the optimal value of lambda that maximizes the log-likelihood function, Logarithmic by “log10” and “log”, Square Root by “sqrt”, Cube Root by “^(1/3)”, Reciprocal by “1/”, Reciprocal Square by “1/()^2”.

Hide

```
#Visualize the p_value for each Shapiro-Wilk test to check for assess the normality
results_shapiro <-anime_tidy %>% summarise(
        square = shapiro.test(sqr_score)$p.value,
        cube = shapiro.test(cube_score)$p.value,
        boxcox = shapiro.test(boxcox_score)$p.value,
        log = shapiro.test(log_score)$p.value,
        ln = shapiro.test(ln_score)$p.value,
        sqrt = shapiro.test(sqrt_score)$p.value,
        cuberoot = shapiro.test(cubrt_score)$p.value,
        reciprocal=shapiro.test(rec_score)$p.value,
        reciprocal_square=shapiro.test(rec_sq_score)$p.value)
results_shapiro
```

square <dbl>	cube <dbl>	boxcox <dbl>	log <dbl>	ln <dbl>	sqrt <dbl>	c
0.0001993256	0.01448617	0.0001992119	2.238552e-13	2.238552e-13	5.510484e-11	8.766

1 row | 1-7 of 9 columns

Then, the Shapiro-Wilk test assesses the normality of each transformed variable by “shapiro.test” function. Then “summarise” function is used to gather all the results in a table.

Hide

```
#Visualize distribution of transformed values
par(mfrow = c(3, 3))
hist(sqr_score, main = "Histogram of Cube-Transformed Anime Score",breaks=30)
hist(cube_score, main = "Histogram of Square-Transformed Anime Score",breaks=30)
```

Hide

```
hist(boxcox_score, main = "Histogram of Box-Cox Transformation /nof Anime Score", breaks = 3
0)
hist(log_score, main = "Histogram of Log Transformation /nof Anime Score",breaks=30)
```

Hide

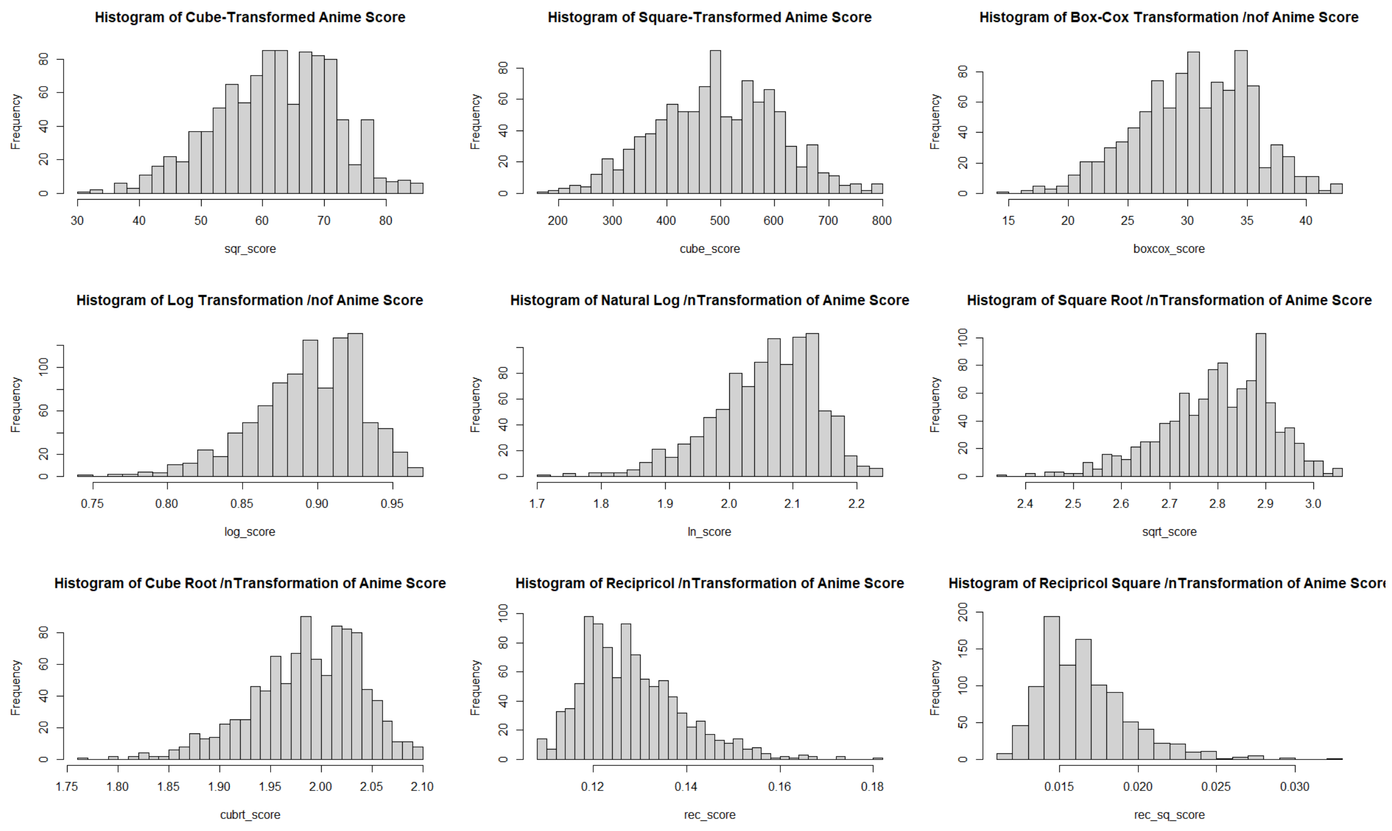
```
hist(ln_score, main = "Histogram of Natural Log /nTransformation of Anime Score",breaks=30)
hist(sqrt_score, main = "Histogram of Square Root /nTransformation of Anime Score",breaks=30)
```

Hide

```
hist(cubrt_score, main = "Histogram of Cube Root /nTransformation of Anime Score",breaks=30)
hist(rec_score, main = "Histogram of Recipricol /nTransformation of Anime Score",breaks=30)
```

Hide

```
hist(rec_sq_score, main = "Histogram of Recipricol Square /nTransformation of Anime Score",br
eaks=30)
par(mfrow = c(1, 1))
```



Finally, the distributions of each transformed score variable are visualized by “hist” function to assess the effects of the transformations. Based on the Shapiro-Wilk test results and the visual inspection, it is evident that the cube mathematical operation (p_value = 0.0145) is the most effective to stabilize variance.

Hide

```
#Compare distribution after transformating
par(mfrow = c(2, 1))
hist(anime_tidy$score, main = "Histogram of Anime Score", breaks = 30)
hist(cubrt_score, main = "Histogram of Cube Root /nTransformation of Anime Score",breaks=30)
par(mfrow = c(1, 1))
```

The code above compares the distribution of the score variable before and after applying the cube root transformation. As in the result, the transformed distribution appears more symmetric and closer to a normal distribution. This transformation helped in stabilizing variance and making “score” variable more suitable for further parametric analyses.

References

Kaggle (n.d) Anime Dataset 2023, Kaggle website, accessed 31 May 2024. https://www.kaggle.com/datasets/dbdmobile/myanimelist-dataset (https://www.kaggle.com/datasets/dbdmobile/myanimelist-dataset)

Vu C and Su N (2024) Data Wrangling (Data Preprocessing) Practical assessment 1 [unpublished paper for MATH2349], RMIT University, Melbourne.

Wickham H, Hester and Bryan J (2024) readr: Read Rectangular Text Data , CRAN.R-Project website, accessed 31 May 2024. https://cran.r-project.org/web/packages/readr/index.html (https://cran.r-project.org/web/packages/readr/index.html)

Wickham H, François R, Henry L, Müller K and Vaughan D (2023) dplyr: A Grammar of Data Manipulation, CRAN.R-Project website, accessed 31 May 2024.https://cran.r-project.org/web/packages/dplyr/index.html

Wickham H, Vaughan D, Girlich M, Ushey K and Software P (2024) tidyr: Tidy Messy Data, CRAN.R-Project website, accessed 31 May 2024. https://cran.r-project.org/web/packages/tidyr/index.html (https://cran.r-project.org/web/packages/tidyr/index.html)

Bache S and Wickham H (2022) magrittr: A Forward-Pipe Operator for R, CRAN.R-Project website, accessed 31 May 2024. https://cran.r-project.org/web/packages/magrittr/index.html (https://cran.r-project.org/web/packages/magrittr/index.html)

Zhu H (2024) kableExtra: Construct Complex Table with ‘kable’ and Pipe Syntax, CRAN.R-Project website, accessed 31 May 2024. https://cran.r-project.org/web/packages/kableExtra/index.html (https://cran.r-project.org/web/packages/kableExtra/index.html)

Wickham H and Jennifer B (2023) readxl : Read Excel Files, CRAN.R-Project website, accessed 31 May 2024. https://cran.r-project.org/web/packages/readxl/index.html (https://cran.r-project.org/web/packages/readxl/index.html)

Wickham H (2023) tidyverse: Easily Install and Load the ‘Tidyverse’, CRAN.R-Project website, accessed 31 May 2024.https://cran.r-project.org/web/packages/tidyverse/index.html

Wickham H, Spinu V and Grolemund G (2023) lubridate: Make Dealing with Dates a Little Easier, CRAN.R-Project website, accessed 31 May 2024. https://cran.r-project.org/web/packages/lubridate/index.html (https://cran.r-project.org/web/packages/lubridate/index.html)

Buuren S and Groothuis K (2023) mice: Multivariate Imputation by Chained Equations, CRAN.R-Project website, accessed 31 May 2024.https://cran.r-project.org/web/packages/mice/index.html

Korkmaz S, Goksuluk D and Zararsiz G (2021) MVN: An R Package for Assessing Multivariate Normality, CRAN.R-Project website, accessed 31 May 2024.https://cran.r-project.org/web/packages/MVN/vignettes/MVN.html

Hyndman R, Athanasopoulos G (2024) forecast: Forecasting Functions for Time Series and Linear Models, CRAN.R-Project website, accessed 31 May 2024.https://cran.r-project.org/web/packages/forecast/index.html

Harrell F (2024) Hmisc: Harrell Miscellaneous, CRAN.R-Project website, accessed 31 May 2024.https://cran.r-project.org/web/packages/Hmisc/index.html

Stackoverflow (2024) How to replace the outliers with the 5th and 95th percentile values in R , Stackoverflow website , accessed 31 May 2024 . https://stackoverflow.com/questions/13339685/how-to-replace-outliers-with-the-5th-and-95th-percentile-values-in-r?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa (https://stackoverflow.com/questions/13339685/how-to-replace-outliers-with-the-5th-and-95th-percentile-values-in-r?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa)