Data Preprocessing of IMDB Data

Code

Assignment 3

Ashley Mallia s3773716 and Daniel Evans s3766654

Required packages

Hide

```
library(dplyr)
library(readr)
library(tidyr)
library(editrules)
```

Executive Summary

The 'Tab Separated Values (tsv)' files and were imported into R. The "\N" were converted to NA's and most of the variables were imported into their correct data types.

The two datasets were then joined together by the common variable type 'tconst'. Unnecessary variables were then removed and the number of observations filtered out to make the size of the dataset more manageable. The 'titleType' variable was then converted to a factor and labelled. The 'startYear' variable was converted from a character to an integer data type. All the variables in the data were then deemed to be in the correct type.

The dataset was not in a tidy format as the 'genre' variable contained multiple values in each cell. The 'genre' and 'tconst' variables were then selected and a new dataset created which holds the genre types for each title in separate cells. Both datasets were now in a tidy format. A new variable was made called 'rank' to determine the top 10 movies in the dataset. The rank variable ranked each title by the number of votes it received multiplied by its average rating.

The data was then scanned for missing values, errors, and inconsistencies. Missing values in the 'titleType' variable were deleted as there was a very small percentage while the mean was imputed for the 'runtimeMinutes' variable by 'titleType'. No missing values remained in the dataset. There were no special values or errors found.

The dataset was then scanned for outliers. Outliers were found in 'numVotes' and 'runtimeMinutes' by titleType however after research these entries were deemed to be valid.

A histogram of 'averageRating' showed the data to be left skewed. A square transformation was done which significantly reduced the skewedness and transformed the data into a normal distribution making statistical analysis much simpler.

Data

The datasets were downloaded as 'Tab Separated Values(tsv)' files from the International Movie Database (IMDB) website

https://datasets.imdbws.com/ (https://datasets.imdbws.com/).

The IMDB website is a popular website to find out any relevant information regarding thousands of movies, tv shows, documentaries etc. Users can leave reviews of titles, determine the cast, writers, directors of specific titles and much more. It is a very useful site for determining your next movie to watch. The datasets are accessible to customers for personal and non-commercial use and are refreshed daily. The two datasets we chose to join together are 'basics' and 'ratings'.

Basics contains the following information for titles:

- tconst (string): alphanumeric unique identifier of the title
- titleType (string): the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc)
- primaryTitle (string): the more popular title / the title used by the filmmakers on promotional materials at the point of release
- originalTitle (string): original title, in the original language
- isAdult (boolean): 0 non-adult title; 1 adult title
- startYear (YYYY): represents the release year of a title. In the case of TV Series, it is the series start year
- endYear (YYYY): TV Series end year. '' for all other title types
- runtimeMinutes: primary runtime of the title, in minutes
- genres (string array): includes up to three genres associated with the title

Ratings contains the IMDb rating and votes information for titles

- tconst (string): alphanumeric unique identifier of the title
- averageRating: weighted average of all the individual user ratings
- numVotes: number of votes the title has received

Hide

```
## Import data
titles <- read_delim("title.basics.tsv", "\t", escape_double = FALSE, na = "\\N", tri
m_ws = TRUE, quote='',
                     col_types = cols(
                       tconst = col_character(),
                       titleType = col_character(),
                       primaryTitle = col_character(),
                       originalTitle = col_character(),
                       isAdult = col_logical(),
                       startYear = col_character(),
                       endYear = col_integer(),
                       runtimeMinutes = col_integer(),
                       genres = col_character()))
ratings <- read_delim("title.ratings.tsv", "\t", escape_double = FALSE, na = "\\N", t
rim_ws = TRUE, quote='')
```

Hide

```
head(titles)
```

tconst <chr>	titleType <chr>	primaryTitle <chr>	originalTitle <chr>	isAdult <lgl>	startYea <chr>
tt0000001	short	Carmencita	Carmencita	FALSE	1894
tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	FALSE	1892
tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	FALSE	1892
tt0000004	short	Un bon bock	Un bon bock	FALSE	1892

tconst	titleType	primaryTitle	originalTitle	isAdult	startYea
<chr>	<chr>	<chr>	<chr>	<lgl>	<chr>
tt0000005	short	Blacksmith Scene	Blacksmith Scene	FALSE	1893
tt0000006	short	Chinese Opium Den	Chinese Opium Den	FALSE	1894

6 rows | 1-7 of 9 columns

Hide

```
head(ratings)
```

tconst	averageRating	numVotes
<chr>	<dbl>	<dbl>
tt0000001	5.8	1507
tt0000002	6.3	183
tt0000003	6.6	1154
tt0000004	6.3	112
tt0000005	6.2	1854
tt0000006	5.4	100

6 rows

Hide

```
## Join datasets
data <- titles %>% left_join(ratings, by = "tconst")
head(data)
```

tconst	titleType	primaryTitle	originalTitle	isAdult	startYea
<chr>	<chr>	<chr>	<chr>	<lgl>	<chr>
tt0000001	short	Carmencita	Carmencita	FALSE	1894
tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	FALSE	1892
tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	FALSE	1892
tt0000004	short	Un bon bock	Un bon bock	FALSE	1892
tt0000005	short	Blacksmith Scene	Blacksmith Scene	FALSE	1893
tt0000006	short	Chinese Opium Den	Chinese Opium Den	FALSE	1894

6 rows | 1-7 of 11 columns

- The datasets were imported into R using the ‘read_delim’ function.
- the “\N” were converted to NA’s and most of the variables were converted to their correct data types upon importing the data into R using the col_types argument.
- The Ratings dataset was joined to Basics by the ‘tconst’ variable using a left_join.

Understand

Hide

```
# drop unneeded columns
data <- data %>% select(-originalTitle, -endYear)
# filtered observations
data <- data %>% filter(averageRating >= 2 & numVotes >= 50)
str(data)
```

```
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':    295837 obs. of  9 variabl
es:
 $ tconst        : chr  "tt0000001" "tt0000002" "tt0000003" "tt0000004" ...
 $ titleType     : chr  "short" "short" "short" "short" ...
 $ primaryTitle  : chr  "Carmencita" "Le clown et ses chiens" "Pauvre Pierrot" "Un bo
n bock" ...
 $ isAdult       : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ startYear     : chr  "1894" "1892" "1892" "1892" ...
 $ runtimeMinutes: int  1 5 4 NA 1 1 1 1 45 1 ...
 $ genres        : chr  "Documentary,Short" "Animation,Short" "Animation,Comedy,Roman
ce" "Animation,Short" ...
 $ averageRating : num  5.8 6.3 6.6 6.3 6.2 5.4 5.5 5.6 5.5 6.9 ...
 $ numVotes      : num  1507 183 1154 112 1854 ...
```

Hide

```
# convert titlesType to factor
data$titleType <- factor(data$titleType,
                         levels = c("short", "movie", "tvSeries", "tvShort",
"tvMovie", "tvEpisode", "tvMiniSeries",                                       "tvSpec
ial", "video", "videoGame"),
                         labels = c("Short", "Movie", "TV_Series", "TV_Short",
"TV_Movie", "TV_Episode", "TV_MiniSeries",                                     "TV_Sp
ecial", "Video", "Video_Game"))
# show all the titleTypes with counts descending
data %>% group_by(titleType) %>% summarise(count = n()) %>% arrange(desc(count))
```

titleType <fctr>	count <int>
Movie	113303
TV_Episode	107583
TV_Series	22772
Short	17336
TV_Movie	14368
Video	9810
Video_Game	3806
TV_MiniSeries	3550
TV_Special	2586
TV_Short	723

1-10 of 10 rows

Hide

```
# converts startYear to an integer data type
data$startYear <- as.integer(data$startYear)
head(data)
```

tconst <chr>	titleType <fctr>	primaryTitle <chr>	isAdult <lgl>	startYear <int>	runtimeMinutes <int>
tt0000001	Short	Carmencita	FALSE	1894	1
tt0000002	Short	Le clown et ses chiens	FALSE	1892	5
tt0000003	Short	Pauvre Pierrot	FALSE	1892	4
tt0000004	Short	Un bon bock	FALSE	1892	NA
tt0000005	Short	Blacksmith Scene	FALSE	1893	1
tt0000006	Short	Chinese Opium Den	FALSE	1894	1

6 rows | 1-6 of 9 columns

- Due to the sheer size of the dataset, we decided to remove what we thought we unnecessary variables and to filter out certain observations to make the dataset more manageable.
- We removed the variables ‘originalTitle’, ‘endYear’.
    - The original title variable repeated the same information as the ‘primary title’ variable for a lot of cases.
    - A large amount of data was missing for the ‘endYear’ variable.
- We filtered out all observations where the average movie rating was less than 2, and the number of votes cast for the movie was less than 50. This significantly reduced the size of the dataset which made it much easier to work with.
- The structure of the dataset was then examined. Most of the variable datatypes were imported correctly, with the exception of ‘titleType’ which should be a factor, and ‘startYear’ which should be an integer.
- The required data conversions were then carried out.
- The final dataset consists of character, logical, numeric, factor, and integer variables.

5. Tidy & Manipulate Data I

Hide

```
# split genres column (which is a comma seperated list) into a seperate table
data_genres <- data %>% select(tconst, genres) %>% separate_rows(genres, sep=",")
# convert the genres column to a factor
data_genres$genres <- factor(data_genres$genres)
# show all the genres with counts descending
data_genres %>% group_by(genres) %>% summarise(count = n()) %>% arrange(desc(count))
```

```
Factor `genres` contains implicit NA, consider using `forcats::fct_explicit_na`
```

genres <fctr>	count <int>
Drama	131525

genres <fctr>	count <int>
Comedy	101460
Action	49398
Crime	42276
Adventure	37009
Animation	31875
Romance	30861
Mystery	23001
Documentary	22542
Family	22455

1-10 of 29 rows Previous **1** 2 3 Next

Hide

```
# drop genres from data table
data <- data %>% select(-genres)
head(data_genres)
```

tconst <chr>	genres <fctr>
tt0000001	Documentary
tt0000001	Short
tt0000002	Animation
tt0000002	Short
tt0000003	Animation
tt0000003	Comedy

6 rows

Hide

```
head(data)
```

tconst <chr>	titleType <fctr>	primaryTitle <chr>	isAdult <lgl>	startYear <int>	runtimeMinutes <int>	avera
tt0000001	Short	Carmencita	FALSE	1894	1	
tt0000002	Short	Le clown et ses chiens	FALSE	1892	5	
tt0000003	Short	Pauvre Pierrot	FALSE	1892	4	
tt0000004	Short	Un bon bock	FALSE	1892	NA	
tt0000005	Short	Blacksmith Scene	FALSE	1893	1	

tconst <chr>	titleType <fctr>	primaryTitle <chr>	isAdult <lgl>	startYear <int>	runtimeMinutes <int>	avera
tt0000006	Short	Chinese Opium Den	FALSE	1894	1	

6 rows

- In order for the dataset to conform to tidy principles;
  - Each variable must have its own column.
  - Each observation must have its own row.
  - Each value must have its own cell.
- At the moment the titles table does not conform to tidy principles since the genre column contains multiple values that are comma seperated.
- To correct this, we split out the 'genre' and 'tconst' variables into a seperate data table. Separating each comma seperated value into its own row.
- Both the data and data_genres tables are now in tidy format and have correct datatypes for all columns.

6. Tidy & Manipulate Data II

Hide

```
#6 create/mutate a variable
ranks = ratings %>% mutate(rank = dense_rank(desc(averageRating * numVotes))) %>% fil
ter(rank <= 10)
titles %>% inner_join(ranks, by="tconst") %>% arrange(rank) %>% select(primaryTitle,
 startYear, rank, averageRating, numVotes)
```

primaryTitle <chr>	startYear <chr>	r... <int>	averageRating <dbl>
The Shawshank Redemption	1994	1	9.3
The Dark Knight	2008	2	9.0
Inception	2010	3	8.8
Fight Club	1999	4	8.8
Pulp Fiction	1994	5	8.9
Game of Thrones	2011	6	9.4
Forrest Gump	1994	7	8.8
The Lord of the Rings: The Return of the King	2003	8	8.9
The Lord of the Rings: The Fellowship of the Ring	2001	9	8.8
The Godfather	1972	10	9.2

1-10 of 10 rows

Hide

```
data <- data %>% mutate(rank = dense_rank(desc(averageRating * numVotes)))
head(data)
```

tconst <chr>	titleType <fctr>	primaryTitle <chr>	isAdult <lgl>	startYear <int>	runtimeMinutes <int>	avera
tt0000001	Short	Carmencita	FALSE	1894	1	
tt0000002	Short	Le clown et ses chiens	FALSE	1892	5	
tt0000003	Short	Pauvre Pierrot	FALSE	1892	4	
tt0000004	Short	Un bon bock	FALSE	1892	NA	
tt0000005	Short	Blacksmith Scene	FALSE	1893	1	
tt0000006	Short	Chinese Opium Den	FALSE	1894	1	

6 rows | 1-7 of 9 columns

To answer the Question: **What are the top 10 movies in the database by popular vote?**

- We are trying to find the top 10 movies with the highest value for averageRating * numVotes.
- Use dense_rank function to assign order the calculation averageRating * numVotes from highest to lowest, then assign a rank.
- Use inner_join to join the titles table to the ranks table.
    - inner_join is used because we only want those titles that exist in ranks to be returned.
- The tibble prints out the top 10 movies in the dataset with “The Shawshank Redemption” being the number one movie.
- The rank varaible was then added to the entire dataset.

7. Scan I

Hide

```
colSums(is.na(data))
```

```
        tconst      titleType   primaryTitle        isAdult 
             0              0              0              0 
     startYear runtimeMinutes  averageRating       numVotes 
            15          29364              0              0 
          rank 
             0 
```

Hide

```
## removed all rows where startYear contained NA's as the amount of missing data was
 less than 5% (15/295822 < 0.05)
data <- data[!is.na(data$startYear),]
## as the amount of missing values is greater than 5% for runtimeMinutes. We imputed
 the mean by movie category
data$runtimeMinutes <-  with(data, ave(runtimeMinutes, titleType, FUN = function(x) r
eplace(x, is.na(x), mean(x, na.rm = TRUE))))
sapply(data, function(x) sum(is.na(x)))
```

```
        tconst      titleType   primaryTitle        isAdult
             0              0              0              0
     startYear runtimeMinutes  averageRating       numVotes
             0              0              0              0
          rank
             0
```

Hide

```
## checking for special values.
is.special <- function(x){
   if (is.numeric(x))(is.infinite(x) | is.nan(x))
}
sapply(data, function(x) sum(is.special(x)))
```

```
        tconst      titleType   primaryTitle        isAdult
             0              0              0              0
     startYear runtimeMinutes  averageRating       numVotes
             0              0              0              0
          rank
             0
```

Hide

```
## checking for inconsistencies or errors
(Rule1 <- editset(c("runtimeMinutes >= 0",
                    "averageRating >= 0", "averageRating <= 10",
                    "numVotes >= 0",
                    "startYear > 0", "startYear < 2020")))
```

```


Edit set:
num1 : 0 <= runtimeMinutes
num2 : 0 <= averageRating
num3 : averageRating <= 10
num4 : 0 <= numVotes
num5 : 0 < startYear
num6 : startYear < 2020
```

Hide

```
v <- violatedEdits(Rule1, data)
sum(v)
```

```
[1] 0
```

- The sum of missing values per column was calculated.
- Due to there being such a small percentage of missing values in the startYear column, it was deemed appropritate to simply remove these observations from the data
- As the amount of missing values is greater than 5% for ‘runtimeMinutes’, we imputed the mean value by ‘titleType’
- We then checked to make sure that there were no further missing values in the dataset
- We then checked for the presence of any infinite or nan values in the dataset of which there were none.

- We then checked for any inconsistencies in the dataset. Such as negative values in time varaibles, movie rating being between 0 and 10, and startYear being less than 2020.

8. Scan II

Hide

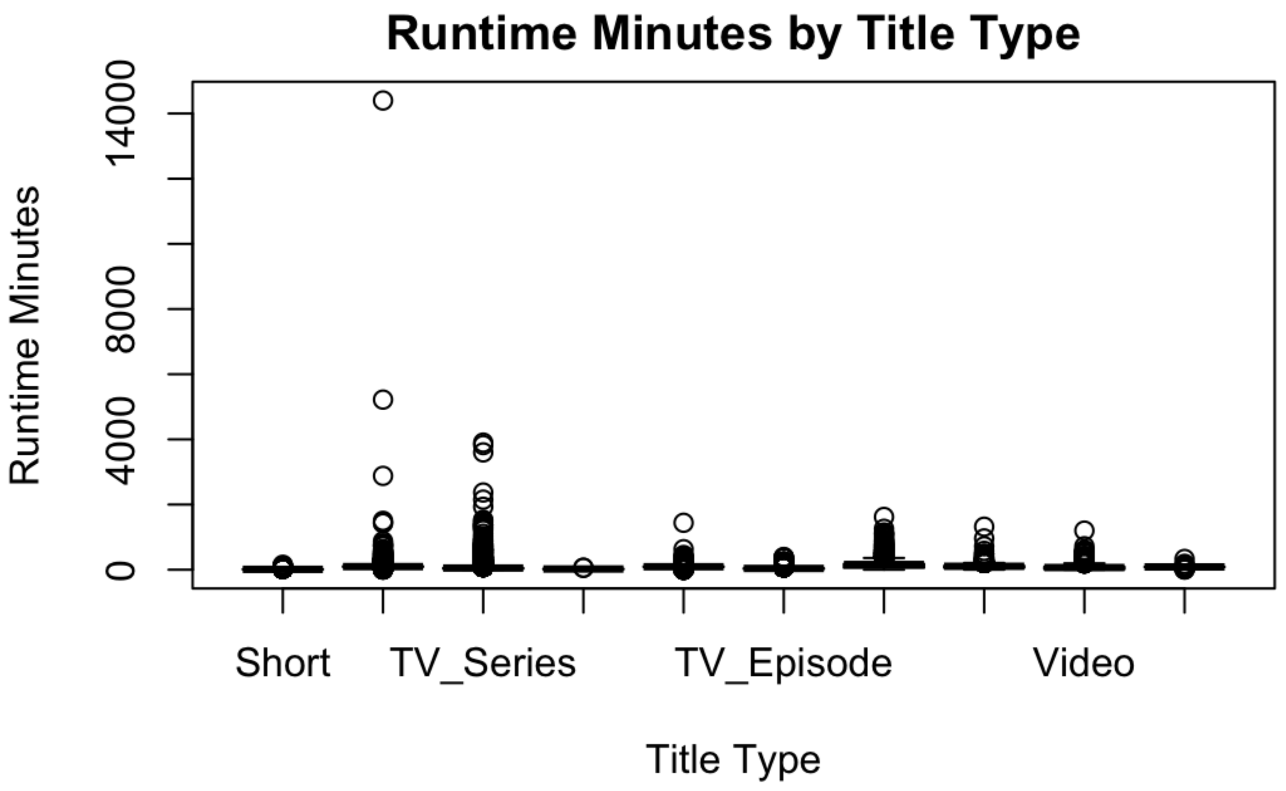
```
datas <- data %>% group_by(titleType) %>% summarise(
  N    = n(),
  MEAN = mean(runtimeMinutes, na.rm = T),
  SD   = sd(runtimeMinutes, na.rm = T),
  MIN  = min(runtimeMinutes, na.rm = T),
  Q1   = quantile(runtimeMinutes, .25, na.rm = T),
  MEDIAN = quantile(runtimeMinutes, .5, na.rm = T),
  Q3    = quantile(runtimeMinutes, .75, na.rm = T),
  MAX   = max(runtimeMinutes, na.rm = T),
  IQR   = Q3-Q1,
  LF = Q1 - 1.5*IQR,
  UF = Q3 + 1.5*IQR,
  LOUT = sum(runtimeMinutes < LF, na.rm = T),
  UOUT = sum(runtimeMinutes > UF, na.rm = T),
  PERC_OUT = round(100*(LOUT+UOUT)/N,2))
datas %>% select(titleType, N, LOUT, UOUT, PERC_OUT) %>% arrange(desc(N))
```

titleType <fctr>	N <int>	LOUT <int>	UOUT <int>	PERC_OUT <dbl>
Movie	113301	2118	7675	8.64
TV_Episode	107574	0	3923	3.65
TV_Series	22772	0	1178	5.17
Short	17334	0	805	4.64
TV_Movie	14368	1972	1853	26.62
Video	9810	0	129	1.31
Video_Game	3806	12	9	0.55
TV_MiniSeries	3548	0	199	5.61
TV_Special	2586	0	102	3.94
TV_Short	723	0	2	0.28

1-10 of 10 rows

Hide

```
boxplot(data$runtimeMinutes ~ data$titleType,
        main = "Runtime Minutes by Title Type",
        xlab = "Title Type",
        ylab = "Runtime Minutes")
```



- According to the Tukey's method of outlier detection, outliers are defined as the values in the data set that fall beyond the range of −1.5×IQR to 1.5×IQR.
- We determined the number of outliers in the 'averageRating' variable to be 5215 from the above table.
- As the percentage of outliers is 1.7%, we deemed it appropriate to simply remove these observations from the dataset.

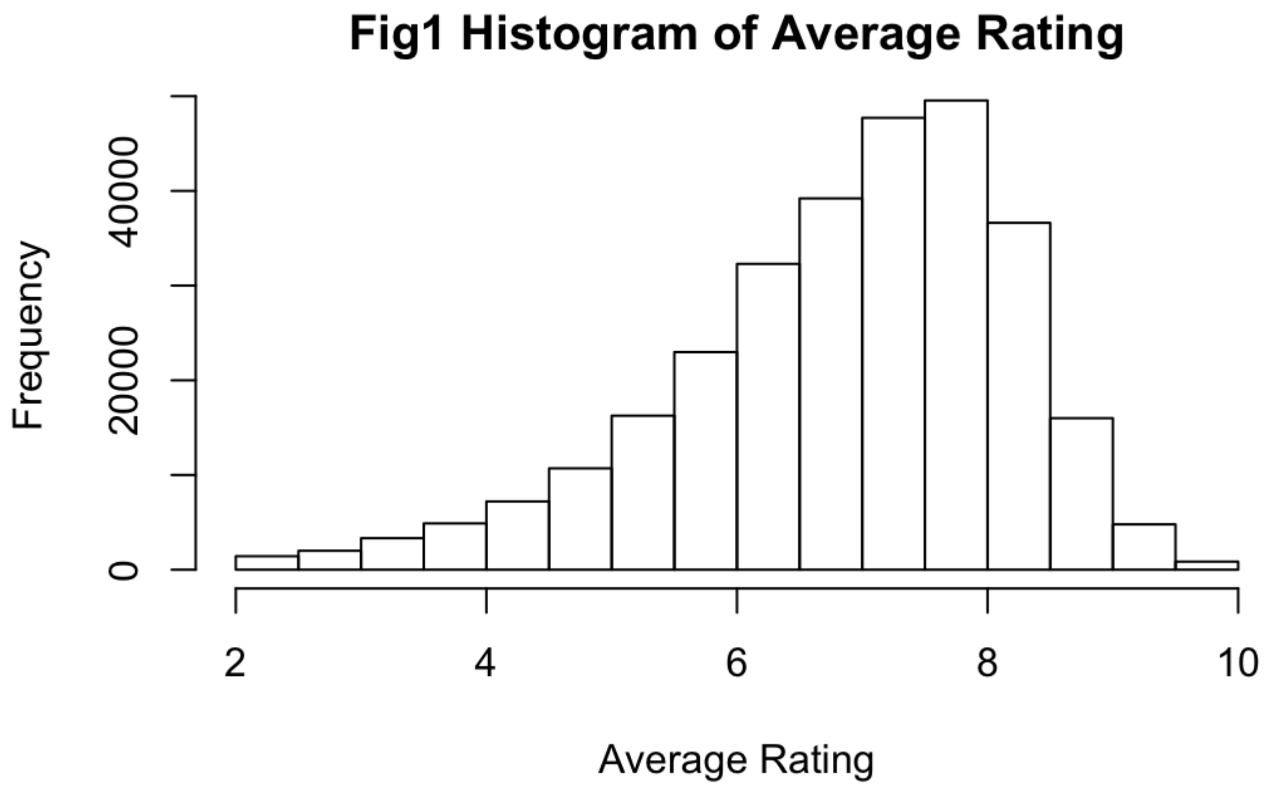
- A Multivariate boxplot was plotted to determine possible outliers in runtimeMinutes by titleType.
- Although the plot does show outliers in the variable, after conducting research the data was deemed to be valid.
- There is a movie which goes for 10 days (14400 hours!)

- The same can be said for the number of votes and average rating numeric variables. Conducting a boxplot shows the presence of many outliers however these are not in fact outliers as the most popular movies receive a huge number of votes in relation to other not so successful movies, and all ratings fell within 0 to 10 as checked above making them valid data.

Transform

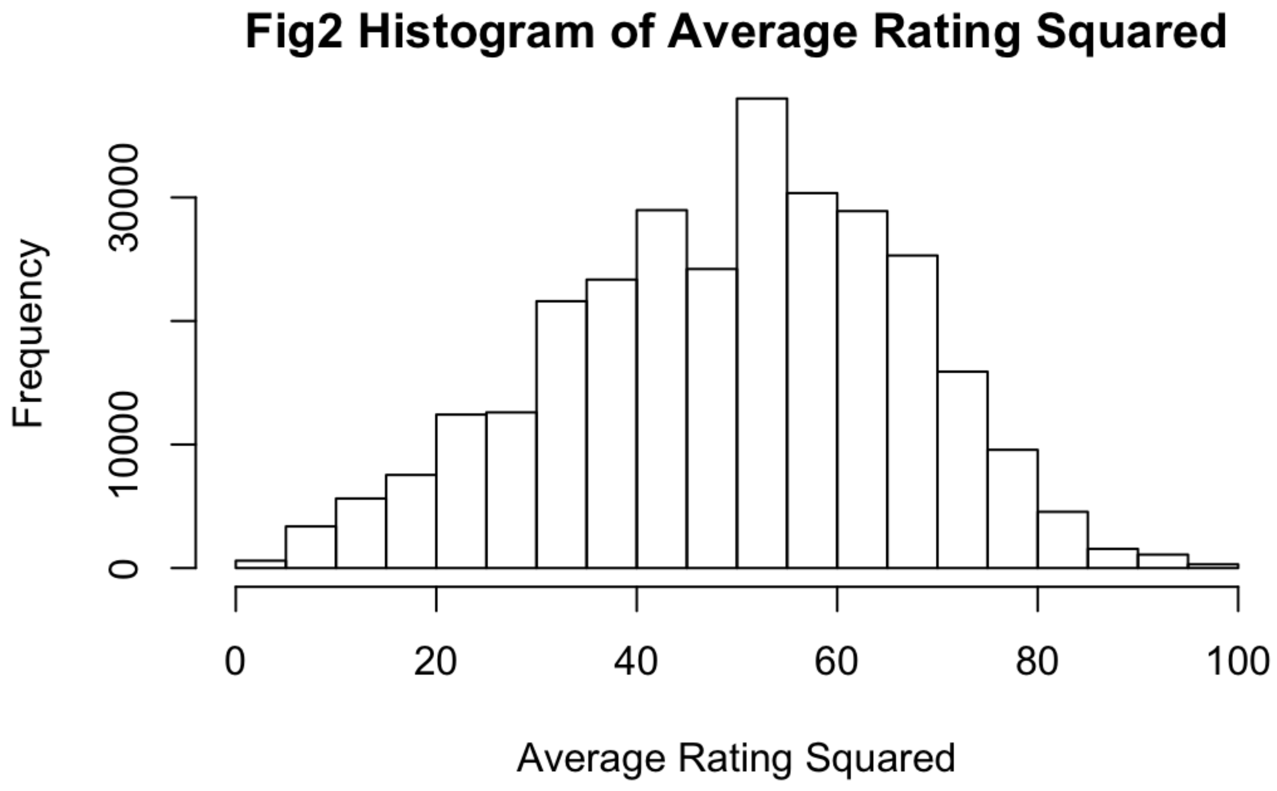
Hide

```
hist(data$averageRating  , main = "Fig1 Histogram of Average Rating", xlab = "Average
Rating")
```



Hide

```
hist(data$averageRating^2, main = "Fig2 Histogram of Average Rating Squared", xlab =
"Average Rating Squared")
```



- In Fig1 we show a histogram of average ratings from the dataset. The data is left skewed.
- Fig2 shows when a squaring transformation is applied, the histogram looks more like a normal distribution