Data Cleaning with R

Simone Vito Lopes

2023-10-17

1. Introduction:

This dataset shows data about movies. It provides information about films and their performance. With this data, we can identify trends and whether each element contributes to the excellent performance of the film or not.

Below, is an explanation of each column before analysis and data cleaning:

MOVIES: This column contains the movie names or titles.
YEAR: Represents the release year of each movie.
GENRE: Contains the category or type of each film.
RATING: This column contains the rating or score for each movie.
ONE-LINE: Consists of the descriptions of each movie.
STARS: This column contains the names of actors or actresses who appear in the movie.
VOTES: Contains the number of ratings each film has received.
RunTime: Logs the duration of each film, probably in minutes.
Gross: Represents the total revenue generated by the film.

2. Data Exploration

The tool used to analyze and clean the data is RStudio.
For the Data Exploration part, I Loaded the file Movies.csv and identified the columns, structure and type of columns.

a. Load the dataset.

```r
# Loading a csv file
options(digits=5)
movies_dataset <- read.csv("Movies.csv", header = TRUE, sep = ",")
```

b. Initial exploration of the dataset, such as checking the dimensions, previewing a few rows, and identifying the data types of each column.

```
## Movies dataset has 9999 rows and 9 columns.

## Data types:

## 'data.frame':    9999 obs. of  9 variables:
##  $ MOVIES  : chr  "Blood Red Sky" "Masters of the Universe: Revelation"
"The Walking Dead" "Rick and Morty" ...
##  $ YEAR    : chr  "-2021" "(2021– )" "(2010–2022)" "(2013– )" ...
##  $ GENRE   : chr  "\nAction, Horror, Thriller            " "\nAnimation,
Action, Adventure            " "\nDrama, Horror, Thriller            "
"\nAnimation, Adventure, Comedy            " ...
##  $ RATING  : num  6.1 5 8.2 9.2 NA 7.6 6.8 8.6 7.9 7.4 ...
##  $ ONE.LINE: chr  "\nA woman with a mysterious illness is forced into
action when a group of terrorists attempt to hijack a transa"| __truncated__
"\nThe war for Eternia begins again in what may be the final battle between
He-Man and Skeletor. A new animated "| __truncated__ "\nSheriff Deputy Rick
Grimes wakes up from a coma to learn the world is in ruins and must lead a
group of survi"| __truncated__ "\nAn animated series that follows the
exploits of a super scientist and his not-so-bright grandson." ...
##  $ STARS   : chr  "\n    Director:\nPeter Thorwarth\n| \n    Stars:\nPeri
Baumeister, \nCarl Anton Koch, \nAlexander Scheer, \nKais Setti\n" "\n
\n    Stars:\nChris Wood, \nSarah Michelle Gellar, \nLena Headey, \nMark
Hamill\n" "\n            \n    Stars:\nAndrew Lincoln, \nNorman Reedus,
\nMelissa McBride, \nLauren Cohan\n" "\n            \n    Stars:\nJustin
Roiland, \nChris Parnell, \nSpencer Grammer, \nSarah Chalke\n" ...
##  $ VOTES   : chr  "21,062" "17,870" "885,805" "414,849" ...
##  $ RunTime : int  121 25 44 23 NA 50 110 53 30 44 ...
##  $ Gross   : chr  "" "" "" "" ...

##                                 MOVIES        YEAR
## 1                        Blood Red Sky       -2021
## 2 Masters of the Universe: Revelation    (2021– )
## 3                     The Walking Dead (2010–2022)
## 4                       Rick and Morty    (2013– )
## 5                      Army of Thieves       -2021
##                                       GENRE RATING
## 1     \nAction, Horror, Thriller               6.1
## 2 \nAnimation, Action, Adventure               5.0
## 3      \nDrama, Horror, Thriller               8.2
## 4 \nAnimation, Adventure, Comedy               9.2
## 5        \nAction, Crime, Horror                NA
##
ONE.LINE
## 1
\nA woman with a mysterious illness is forced into action when a group of
terrorists attempt to hijack a transatlantic overnight flight.
## 2                                                               \nThe war
for Eternia begins again in what may be the final battle between He-Man and
Skeletor. A new animated series from writer-director Kevin Smith.
## 3
```

```
\nSheriff Deputy Rick Grimes wakes up from a coma to learn the world is in
ruins and must lead a group of survivors to stay alive.
## 4
\nAn animated series that follows the exploits of a super scientist and his
not-so-bright grandson.
## 5 \nA prequel, set before the events of Army of the Dead, which focuses on
German safecracker Ludwig Dieter leading a group of aspiring thieves on a top
secret heist during the early stages of the zombie apocalypse.
##
STARS
## 1             \n    Director:\nPeter Thorwarth\n| \n    Stars:\nPeri
Baumeister, \nCarl Anton Koch, \nAlexander Scheer, \nKais Setti\n
## 2                                    \n            \n    Stars:\nChris
Wood, \nSarah Michelle Gellar, \nLena Headey, \nMark Hamill\n
## 3                                   \n            \n    Stars:\nAndrew
Lincoln, \nNorman Reedus, \nMelissa McBride, \nLauren Cohan\n
## 4                                   \n            \n    Stars:\nJustin
Roiland, \nChris Parnell, \nSpencer Grammer, \nSarah Chalke\n
## 5 \n    Director:\nMatthias Schweighöfer\n| \n    Stars:\nMatthias
Schweighöfer, \nNathalie Emmanuel, \nRuby O. Fee, \nStuart Martin\n
##     VOTES RunTime Gross
## 1  21,062     121
## 2  17,870      25
## 3 885,805      44
## 4 414,849      23
## 5              NA
```

3. Handling Missing Values

a. Identifying columns with missing values.

```
# Clearing '\n' characters from all columns
for (col in names(movies_dataset)) {
  movies_dataset[[col]] <- gsub("\n", "", movies_dataset[[col]])
}

# Looking for missing values
movies_dataset[movies_dataset == ""] <- NA
colSums(is.na(movies_dataset))

##   MOVIES     YEAR    GENRE   RATING ONE.LINE    STARS    VOTES  RunTime
##        0      644       80     1820        0      456     1820     2958
##    Gross
##     9539
```

b. Evaluating the extent of missing values in each column and decide on an appropriate strategy for handling them.

The strategy will depend on the number of NA values, I'm thinking in follow these steps.
First, the Gross column will be deleted because have more than 95% of the missing values.
Then I'll find movies that have more than one line, and if a line has value, I can use it to fill the NA's, as follows:
YEAR - Fill the column with NA using the year found.
GENRE - Fill the column with NA using the found genre.
RATING - Fill the column with NA using the average of all lines found.
STARS - Fill the column with NA using the found stars.
VOTES - Fill the column with NA using the average.
RunTime - Fill the column with NA using the value that appeared the most.
After these searches, I will analyze again to see if the numbers have improved.
Then I can delete the rows where the RATING, VOTES and RunTime columns have NA.
I will do the analysis again to see if there are still NAs.

c. Start implement the chosen strategy.

```r
# Removing the column Gross because have more than 95% of the missing values.
movies_dataset <- movies_dataset[, -which(names(movies_dataset) == "Gross"),
drop = FALSE]

# Clear leading and trailing whitespace in character-type columns
# Some columns have space at the beginning of the value, so I clean it to
ensure that the space will not interfere with the result.
movies_dataset <- movies_dataset %>%
  mutate_if(is.character, trimws)


## I filled the columns with some reasonable data to decrease data loss.

# YEAR - Looking for values on the lines of the same movie.
# order by MOVIE and YEAR
movies_dataset <- movies_dataset[order(movies_dataset$MOVIES,
movies_dataset$YEAR), ]

movies_dataset <- movies_dataset %>%
  group_by(MOVIES) %>%
  mutate(YEAR = if (any(!is.na(YEAR))) na.locf(YEAR) else YEAR)

# GENRE - Looking for values on the lines of the same movie.
# order by MOVIE, YEAR and GENRE
movies_dataset <- movies_dataset[order(movies_dataset$MOVIES,
movies_dataset$YEAR, movies_dataset$GENRE), ]

movies_dataset <- movies_dataset %>%
  group_by(MOVIES) %>%
  mutate(GENRE = if (any(!is.na(GENRE))) na.locf(GENRE) else GENRE)
```

```r
# STARS - Looking for values on the lines of the same movie.
# order by MOVIE and STARS
movies_dataset <- movies_dataset[order(movies_dataset$MOVIES,
movies_dataset$STARS), ]

# Fills NA with the median.
movies_dataset <- movies_dataset %>%
  group_by(MOVIES) %>%
  fill(STARS)

# RATING - Looking for values on the lines of the same movie.
# order by MOVIE and RATING
movies_dataset <- movies_dataset[order(movies_dataset$MOVIES,
movies_dataset$RATING), ]
# convert to numeric
movies_dataset$RATING <- as.numeric(movies_dataset$RATING)

# Fills NA with the median in the column RATING.
movies_dataset <- movies_dataset %>%
  group_by(MOVIES) %>%
  mutate(RATING = na.aggregate(RATING))

# round the value
movies_dataset$RATING <- round(movies_dataset$RATING, 1)

# RunTime - Looking for values on the lines of the same movie.
# order by MOVIE and RunTime
movies_dataset <- movies_dataset[order(movies_dataset$MOVIES,
movies_dataset$RunTime), ]
# convert to numeric
movies_dataset$RunTime <- as.numeric(movies_dataset$RunTime)

# Function to find the mode (most frequent value)
find_mode <- function(x) {
  if (length(x) == 0) {
    return(NA) # Returns NA if the array is empty
  } else {
    tbl <- table(x, useNA = "always") # Count NA values
    mode_val <- as.numeric(names(tbl[tbl == max(tbl)]))
    if (length(mode_val) > 0) {
      return(mode_val[1])  # Select the first most frequent value in case of
ties.
    } else {
      return(NA) # Returns NA if all values are NA
    }
  }
}
```

```
# Fills NA with the most frequent value per movie.
movies_dataset <- movies_dataset %>%
  group_by(MOVIES) %>%
  mutate(RunTime = ifelse(is.na(RunTime), find_mode(RunTime), RunTime))

# VOTES - Looking for values on the lines of the same movie.
movies_dataset <- movies_dataset[order(movies_dataset$MOVIES,
movies_dataset$VOTES), ]
# remove commas
movies_dataset$VOTES <- gsub(",", "", movies_dataset$VOTES)
# convert to numeric
movies_dataset$VOTES <- as.numeric(movies_dataset$VOTES)

# Fills NA with the median.
movies_dataset <- movies_dataset %>%
  group_by(MOVIES) %>%
  mutate(VOTES = na.aggregate(VOTES))

colSums(is.na(movies_dataset))

##   MOVIES     YEAR    GENRE   RATING ONE.LINE    STARS    VOTES  RunTime
##        0      614       73     1370        0      358     1370     2873

cat("Movies dataset has", dim(movies_dataset)[1], "rows and",
dim(movies_dataset)[2],
    "columns.\n")

## Movies dataset has 9999 rows and 8 columns.

# I deleted the lines with the RATING, VOTES and RunTime columns filled with
NA.
# These cells are the basis for performing numerous analyzes on this dataset.
I understand that the line is irrelevant if there is no value in these
columns.

# The command complete.cases return all data that don't have NA.
movies_dataset <- movies_dataset[complete.cases(movies_dataset[, c("RATING",
"VOTES", "RunTime")]), ]

# Show the data information again to see if NA values still exist.
colSums(is.na(movies_dataset))

##   MOVIES     YEAR    GENRE   RATING ONE.LINE    STARS    VOTES  RunTime
##        0        0        7        0        0       24        0        0

cat("Movies dataset has", dim(movies_dataset)[1], "rows and",
dim(movies_dataset)[2],
    "columns.\n")

## Movies dataset has 6956 rows and 8 columns.
```

```
# As we had only 7 NAs on GENRE and 24 on STARS, I filled the columns with
the information "Unknown value" to avoid losing more data.
colunas_com_NA <- colnames(movies_dataset)[apply(is.na(movies_dataset), 2,
any)]
movies_dataset[colunas_com_NA] <- lapply(movies_dataset[colunas_com_NA],
function(x) ifelse(is.na(x), "Unknown value", x))
```

d. Validating the changes made and ensuring that no missing values remain.

```
# Verify the data information to see if NA values still exist.
colSums(is.na(movies_dataset))

##   MOVIES     YEAR    GENRE   RATING ONE.LINE    STARS    VOTES  RunTime
##        0        0        0        0        0        0        0        0

cat("Movies dataset has", dim(movies_dataset)[1], "rows and",
dim(movies_dataset)[2],
    "columns.\n")

## Movies dataset has 6956 rows and 8 columns.

head(movies_dataset, 5) # The analysis was about the whole dataset, but I'm
printing only 5 lines for demonstration.

## # A tibble: 5 × 8
## # Groups:   MOVIES [5]
##   MOVIES                        YEAR  GENRE   RATING ONE.LINE STARS VOTES
RunTime
##   <chr>                         <chr> <chr>   <dbl> <chr>    <chr> <dbl>
<dbl>
## 1 '76                           -2016 Drama…    7.2 "The st… Dire…    58
118
## 2 #AnneFrank - Parallel Stories -2019 Drama…    6.5 "It is … Dire…  1296
92
## 3 #BlackAF                      -2020 Comedy    6.8 "A fath… Star…  4200
36
## 4 #cats_the_mewvie              -2020 Docum…    5.3 "It's a… Dire…   439
90
## 5 #HappyBirthdaySense8          -2017 Short     8.9 "Add a … Dire…   828
15
```

Steps 1. When I ran the head command to see the data, I saw many '' characters before the values, so I cleared those characters from all columns. 2. I looked for missing values. I found missing values on columns YEAR, GENRE, RATING, STARS, VOTES, RunTime, and Gross. 3. How I handdled each column is explain in part 3b.

4. Removing Duplicates

a. Checking for duplicate rows in the dataset.

```
# To identify duplicates, I used the duplicates command.
linhas_duplicadas <- duplicated(movies_dataset)
print(movies_dataset[linhas_duplicadas, ])

## # A tibble: 28 × 8
## # Groups:   MOVIES [2]
##    MOVIES                   YEAR     GENRE   RATING ONE.LINE STARS  VOTES
RunTime
##    <chr>                    <chr>    <chr>    <dbl> <chr>    <chr>  <dbl>
<dbl>
##  1 Bridgerton               (2020– ) Drama…     7.7 Add a P… Star… 10986.
61
##  2 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
##  3 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
##  4 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
##  5 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
##  6 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
##  7 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
##  8 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
##  9 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
## 10 Power Rangers: Dino Fury -2021    Actio…     8.1 Add a P… Dire…    68
21
## # ℹ 18 more rows
```

b. Removing Duplicates.

```
# I created a new dataset. Here I'm using the same name, with unique values.
movies_dataset <- unique(movies_dataset)
```

c. Verifying if duplicate rows have been successfully removed.

```
# Verify the duplicates again to be sure.
linhas_duplicadas <- duplicated(movies_dataset)
print(movies_dataset[linhas_duplicadas, ])
```

```
## # A tibble: 0 × 8
## # Groups:   MOVIES [0]
## # ℹ 8 variables: MOVIES <chr>, YEAR <chr>, GENRE <chr>, RATING <dbl>,
## #   ONE.LINE <chr>, STARS <chr>, VOTES <dbl>, RunTime <dbl>
```

I first checked for duplicates using the duplicate() command, then removed the duplicates using the unique() command and checked again to make sure all duplicates were removed.

5. Dealing with Inconsistent Data

a. Identifying columns that contain inconsistent or erroneous data.
1. Column names are not standardized.
2. The STARS column also contains director information.
3. There are special characters in the data - column MOVIES.
4. The YEAR column contains characters, not just numbers, sometimes just one year, sometimes two.
5. Adjustment of column types and order.

b. Implementing necessary corrections to resolve inconsistencies.

```
# 1. Column names are not standardized.
# Standardizing column names. To standardize the column names I will use the
CamelCase technique.
# I changed the column name ONE.LINE to description.

colnames(movies_dataset) <- c("movies", "year", "genre", "rating",
"description", "stars", "votes", "runTime")

# 2. Separate column stars to director and stars

# stars to director and stars
split_column <- function(info) {
  parts <- strsplit(info, "\\|")
  if (length(parts[[1]]) == 2) {
    return(c(director = parts[[1]][1], stars = parts[[1]][2]))
  } else {
    return(c(director = NA, stars = parts[[1]][1]))
  }
}

movies_dataset[, c("director", "stars")] <- t(sapply(movies_dataset$stars,
split_column))

# clean the names from value
movies_dataset$director <- gsub("Directors:", "", movies_dataset$director)
```

```r
movies_dataset$director <- gsub("Director:", "", movies_dataset$director)
movies_dataset$stars <- gsub("Stars:", "", movies_dataset$stars)
movies_dataset$stars <- gsub("Star:", "", movies_dataset$stars)

# Set the columns NA after split as "Unknown value"
colunas_com_NA <- colnames(movies_dataset)[apply(is.na(movies_dataset), 2,
any)]
movies_dataset[colunas_com_NA] <- lapply(movies_dataset[colunas_com_NA],
function(x) ifelse(is.na(x), "Unknown value", x))

# 3. Clean special characters on movies column.
clean_chars <- function(x) {
  x <- gsub("[^[:alnum:] :]", "", x)  # Removes all non-alphanumeric
characters, spaces and ":"
  x <- gsub("\\s+", " ", x)          # Remove extra spaces and replace them
with a single space.
  return(x)
}

movies_dataset$movies <- clean_chars(movies_dataset$movies)
colSums(is.na(movies_dataset))

##      movies        year       genre      rating description       stars
##           0           0           0           0           0           0
##       votes     runTime    director
##           0           0           0

# 4. I will clean the charateres and create two columns initialYear, and
finalYear

# year to initialYear, and finalYear
movies_dataset[c("initialYear", "finalYear")] <- do.call(rbind,
strsplit(as.character(movies_dataset$year), "-|–"))

# Clears non-numeric characters
movies_dataset$initialYear <- as.numeric(gsub("[^0-9]", "",
movies_dataset$initialYear))
movies_dataset$finalYear <- as.numeric(gsub("[^0-9]", "",
movies_dataset$finalYear))

# If one of the columns is NA, it will be filled in with the year of the
other column.
movies_dataset$initialYear <- ifelse(is.na(movies_dataset$initialYear),
movies_dataset$finalYear, movies_dataset$initialYear)
movies_dataset$finalYear <- ifelse(is.na(movies_dataset$finalYear),
movies_dataset$initialYear, movies_dataset$finalYear)

# Check if NA exists
rows_with_na <- subset(movies_dataset, is.na(initialYear) | is.na(finalYear))
print(rows_with_na)
```

```
## # A tibble: 4 × 11
## # Groups:   movies [4]
##   movies           year  genre  rating description stars  votes runTime
director
##   <chr>            <chr> <chr>   <dbl> <chr>       <chr>  <dbl>   <dbl>
<chr>
## 1 Blackout         (XLI) Drama     6.5 Six differ… "    …   230      90
Ben Cha…
## 2 House Arrest     (III) Comedy    5.5 It is the … "Dir…  1516     104
Unknown…
## 3 Paranormal       (I)   Drama…    8.7 Depicts th… "Ahm… 15780.     45
Unknown…
## 4 Sword Art Online (I)   Actio…    7.6 The series… "Yos… 39325      24
Unknown…
## # ℹ 2 more variables: initialYear <dbl>, finalYear <dbl>
```

```
#Adjustments in NA found
movies_dataset <- movies_dataset[order(movies_dataset$movies,
movies_dataset$initialYear), ]

movies_dataset <- movies_dataset %>%
  group_by(movies) %>%
  mutate(
    initialYear = if (any(!is.na(initialYear))) na.locf(initialYear) else
initialYear,
    finalYear = if (any(!is.na(finalYear))) na.locf(finalYear) else finalYear
  )

# 5. Other adjustments, such as, order, clean spaces.

# Work with the column year.
movies_dataset <- movies_dataset[, -which(names(movies_dataset) == "year"),
drop = FALSE]
movies_dataset <- movies_dataset[, -which(names(movies_dataset) ==
"finalYear"), drop = FALSE]
names(movies_dataset)[names(movies_dataset) == "initialYear"] <- "year"

# Order columns
movies_dataset <- movies_dataset[, c("movies", "year", "genre",
"description", "director", "stars", "rating", "votes", "runTime")]

# Clean the spaces
movies_dataset <- movies_dataset %>%
  mutate_if(is.character, trimws)
```

```
## `mutate_if()` ignored the following grouping variables:
## • Column `movies`
```

```
# Type of column
movies_dataset$year <- as.character(movies_dataset$year)
```

c. Below is a quick explanation of the changes made.

1. Column names are not standardized.
To make it easier to understand the columns and have a naming pattern.
2. The STARS column also contains director information.
I separated the columns to better identify the directors and actors, a lot of people search for movies by the name of the director or actor, so I understand that the evaluations can also be influenced by these elements.
3. There are special characters in the data - column MOVIES.
I removed the special characters so as not to generate problems in future analyses.
4. The YEAR column contains characters, not just numbers, sometimes just one year, sometimes two.
The year column was a mess. I tried to clean all non-numeric characters and separate between the initial and final year, I saw that many films were without one or the other, so I decided to merge the two and gave priority to the initial year, so we were left with a year column that can contain the initial or final year of the film.
5. Adjustment of column types and order.
I put the columns in an order that makes more sense, with character types first and numbers after, for example. I cleaned the spaces that might still exist in the data. The dataset is simpler and easier to understand if it has a coherent order.
I thought of facilitating the visualization and understanding of the data in the best possible way.

6. Handling Outliers

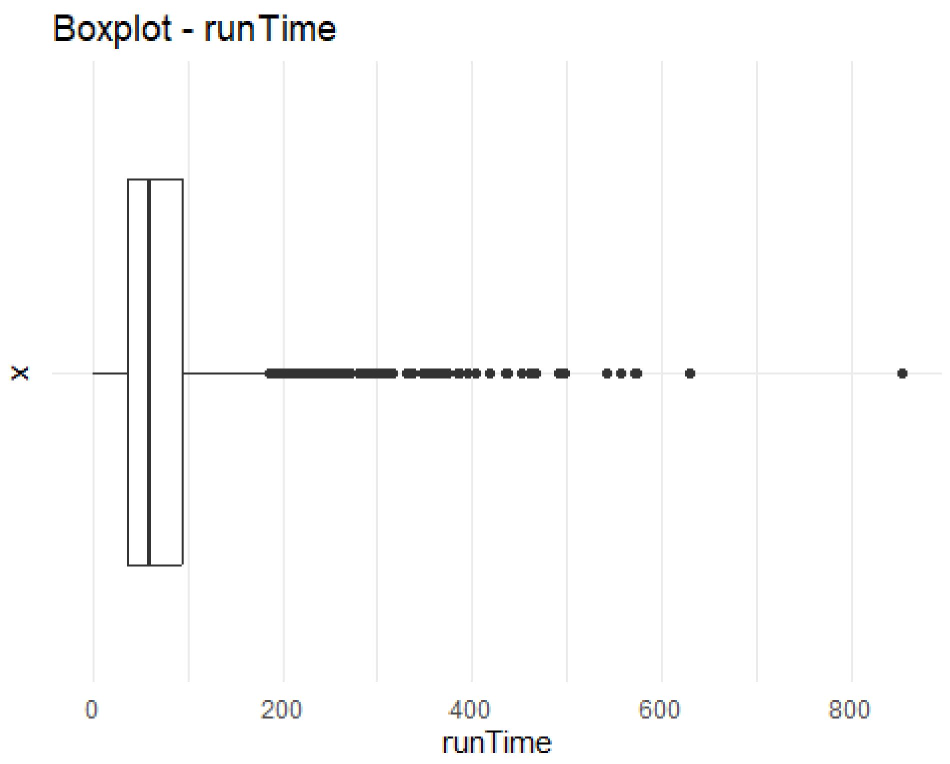
a. Identifying columns that may contain outliers.
Identify outliers in the runTime column. It’s not normal for a movie or show with less than 20 minutes.

```
summary(movies_dataset[, c("rating", "votes", "runTime")])
```

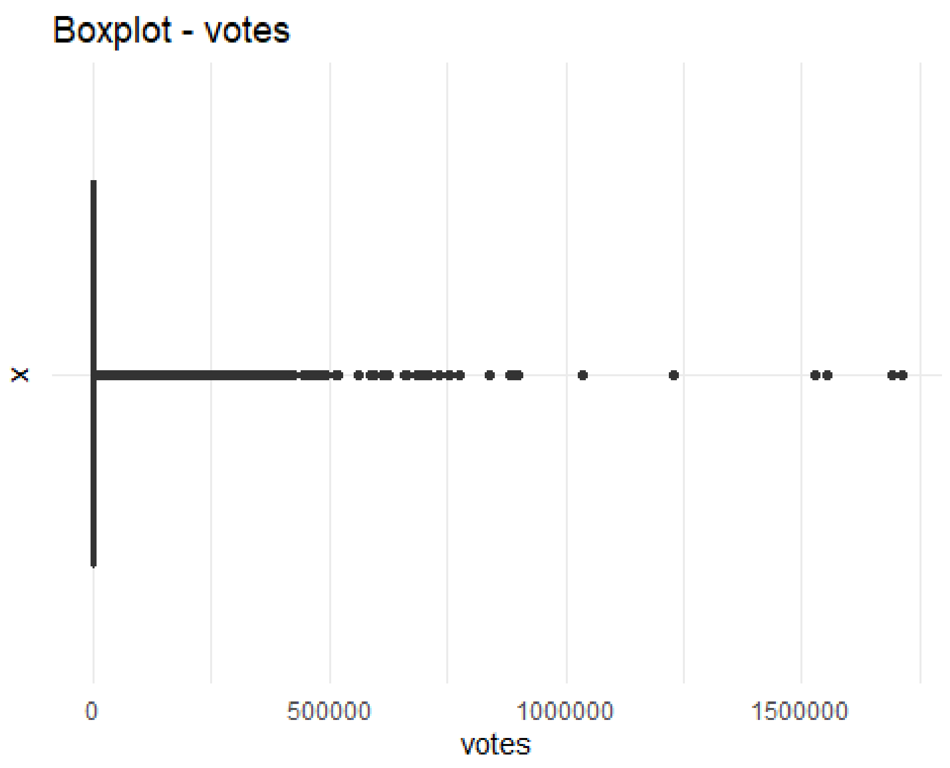
```
##      rating          votes              runTime     
##  Min.   :1.10   Min.   :      5   Min.   :  1.0  
##  1st Qu.:6.10   1st Qu.:    276   1st Qu.: 37.0  
##  Median :7.00   Median :   1198   Median : 60.0  
##  Mean   :6.87   Mean   :  17834   Mean   : 68.9  
##  3rd Qu.:7.70   3rd Qu.:   5132   3rd Qu.: 95.0  
##  Max.   :9.90   Max.   :1713028   Max.   :853.0  
```

b. Visualizing the distributions of columns using plots.

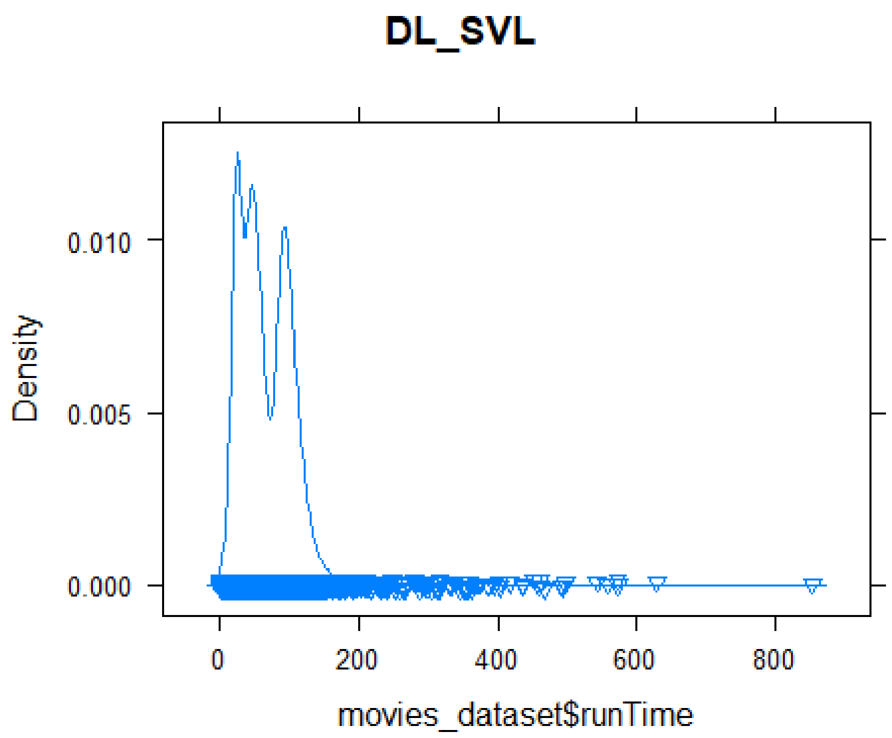
```
ggplot(data = movies_dataset, aes(x = "", y = runTime)) +
  geom_boxplot() +
  labs(title = "Boxplot - runTime", y = "runTime") +
  theme_minimal() +
  coord_flip()
```



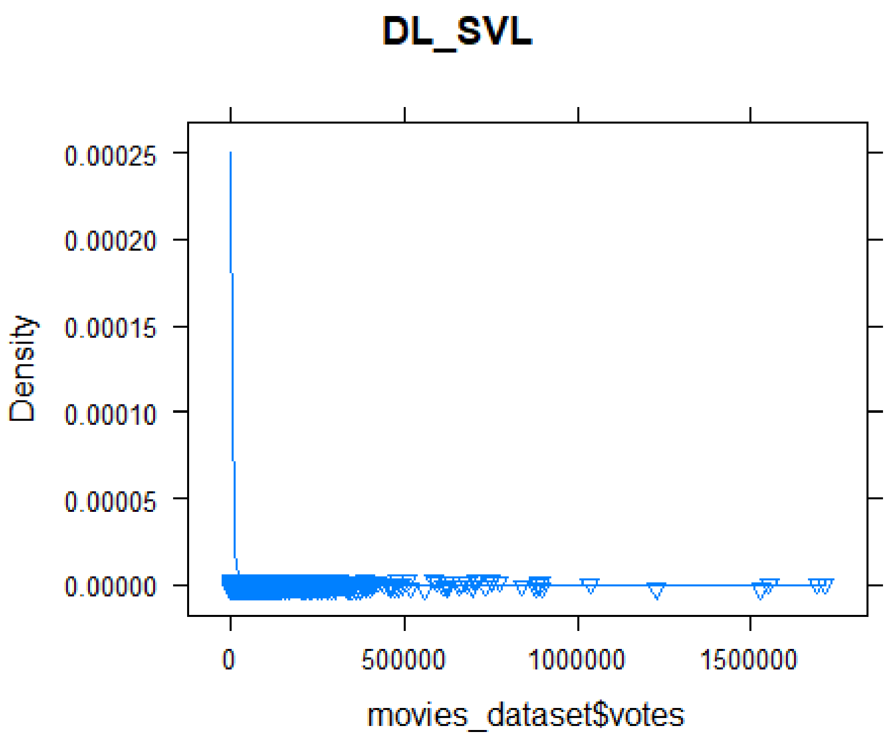
```
ggplot(data = movies_dataset, aes(x = "", y = votes)) +
  geom_boxplot() +
  labs(title = "Boxplot - votes", y = "votes") +
  theme_minimal() +
  coord_flip()
```



```r
densityplot( ~ movies_dataset$runTime, pch=6, main = 'DL_SVL')
```



```r
densityplot( ~ movies_dataset$votes, pch=6, main = 'DL_SVL')
```



c. Analyzing the outliers and deciding on an appropriate strategy for handling them.
In the runTime column, there are outliers both in the minimum and maximum values, in the graphs, the maximum values are clear, but I will eliminate all values above 400 and below 15 minutes.
In the votes column also have outliers, I will remove the values above 1000000.

d. Implementing the chosen strategy and validate the changes made.

```r
# Removing values below 15 and above 400 from column runTime.
movies_dataset <- subset(movies_dataset, runTime >= 15 & runTime <= 400)

# Removing values above 1000000 from column votes
movies_dataset <- subset(movies_dataset, votes <= 1000000)
```

I tried to understand which columns could have outliers and then generated the graphs to be more visual. Cleaning was a mix of what I analyzed in the data with my idea of what would be correct.

7. Data Validation

a. Additional checks or validations to ensure the overall quality of the dataset.

```
# Show the dimensions
cat("Movies dataset has", dim(movies_dataset)[1], "rows and",
dim(movies_dataset)[2],
    "columns.\n")

## Movies dataset has 6805 rows and 9 columns.

# Identifying the data types of each column
summary(movies_dataset)

##     movies              year               genre           description
##  Length:6805        Length:6805        Length:6805        Length:6805
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##    director             stars               rating          votes
##  Length:6805        Length:6805        Min.   :1.10   Min.   :     5
##  Class :character   Class :character   1st Qu.:6.10   1st Qu.:   279
##  Mode  :character   Mode  :character   Median :7.00   Median :  1209
##                                        Mean   :6.86   Mean   : 16724
##                                        3rd Qu.:7.70   3rd Qu.:  5116
##                                        Max.   :9.90   Max.   :897444
##     runTime
##  Min.   : 15.0
##  1st Qu.: 39.0
##  Median : 60.0
##  Mean   : 68.5
##  3rd Qu.: 95.0
##  Max.   :395.0

# Previewing a few rows
head(movies_dataset,5)

## # A tibble: 5 × 9
## # Groups:   movies [5]
##   movies             year  genre description director stars rating votes
runTime
##   <chr>              <chr> <chr> <chr>       <chr>    <chr>  <dbl> <dbl>
<dbl>
## 1 1 Chance 2 Dance   2014  Dram… When a sev… Adam De… Lexi…    4.8   544
89
## 2 10 000 timmar      2014  Come… When he wi… Joachim… Pete…    4.4   953
94
## 3 10 Days in Sun Ci… 2017  Adve… A globetro… Adze Ug… Gben…    5.1    58
87
## 4 100 Coco           2017  Drama A quirky 1… Tessa S… Nola…    4.7   838
88
## 5 100 Hotter         2016  Real… A make-und… Unknown… Dani…    4.4   864
60
```

```
# Verify NA
movies_dataset$movies <- clean_chars(movies_dataset$movies)
colSums(is.na(movies_dataset))
```

```
##      movies        year       genre description    director       stars 
##           0           0           0           0           0           0 
##      rating       votes     runTime 
##           0           0           0
```

```
# Creating a new file with the changes made.
write_excel_csv(movies_dataset, file = "movies_dataset_cleaned.csv")
```

I searched again for the dimensions of the dataset. I checked the data and if NA's still appeared.

8. Observations

It's an exciting dataset, and it's exactly like that in real life; a very big job to be done to clean the data, especially if there is no validation at the time of inclusion by users.

If you open the cleaning dataset in Excel and the letter "é" is displayed as "√©", for example. Instead of opening directly, try importing the data. –> Go to: 'Data' –> Get the data 'From Text/CSV' and then select '65001:Unicode (UTF-8)'. This will match the encoding from R.

9. Conclusion

Finally, the dataset provides a comprehensive view of diverse films, containing critical aspects that shed light on their performance and qualities. The collection comprises movie titles, release years, genres, and descriptions that provide insight into the diversity of movie topics and genres. The ratings and number of votes provide useful information regarding the reception and popularity of each film among the general public. Furthermore, including famous actors, actresses, and directors allows us to investigate the impact of casting decisions on a film's popularity.
The dataset's cleanliness and coherence and well-structured and relevant column names make it more accessible and usable for data analysis and interpretation. Filmmakers and industry experts can use this dataset to make educated decisions, establish effective film strategies, and better understand the film industry's dynamic terrain.