AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

DEPARTMENT OF CSE

Introduction to Data Science

Fall 2022-2023

Section: C

FINAL TERM PROJECT ON

Web scraping, Data Preprocessing, Data manipulation and Data Visualization (Highest Budget Hollywood movies)

Supervised By

Dr. Akinul Islam Jony

Submitted By:

Name	ID	Contribution
1. Muhammad Shahriar Zaman	20-41840-1	All

Date of Submission: **December 11th, 2022**

TABLE OF CONTENTS

Project Overview:

For this project we have been assigned to scrap data from webpages, perform preprocessing techniques on them, describe them in the light of descriptive statistics and visualizing them using R language.

For this group we have chosen to work with data regarding movies. We will scrap data about the highest grossing and highest earning movies of all time, store them in datasets and them perform our assigned project objectives on them.

Project Solution Design:

For this project we will scrape data from webpages using R libraries and other useful tools, then we shall prepare a clean preprocessed dataset which would be free of any discrepancies. After that we shall describe our data by methods of descriptive statistics and finally we shall use various tools of R language to visualize and explain our data. To summarize, we can look at the following points.

- Web scraping data and storing them
- Performing preprocessing on them employing data manipulation techniques
- Expressing descriptive statistical parameters for our data
- Visualization of our data using R libraries.

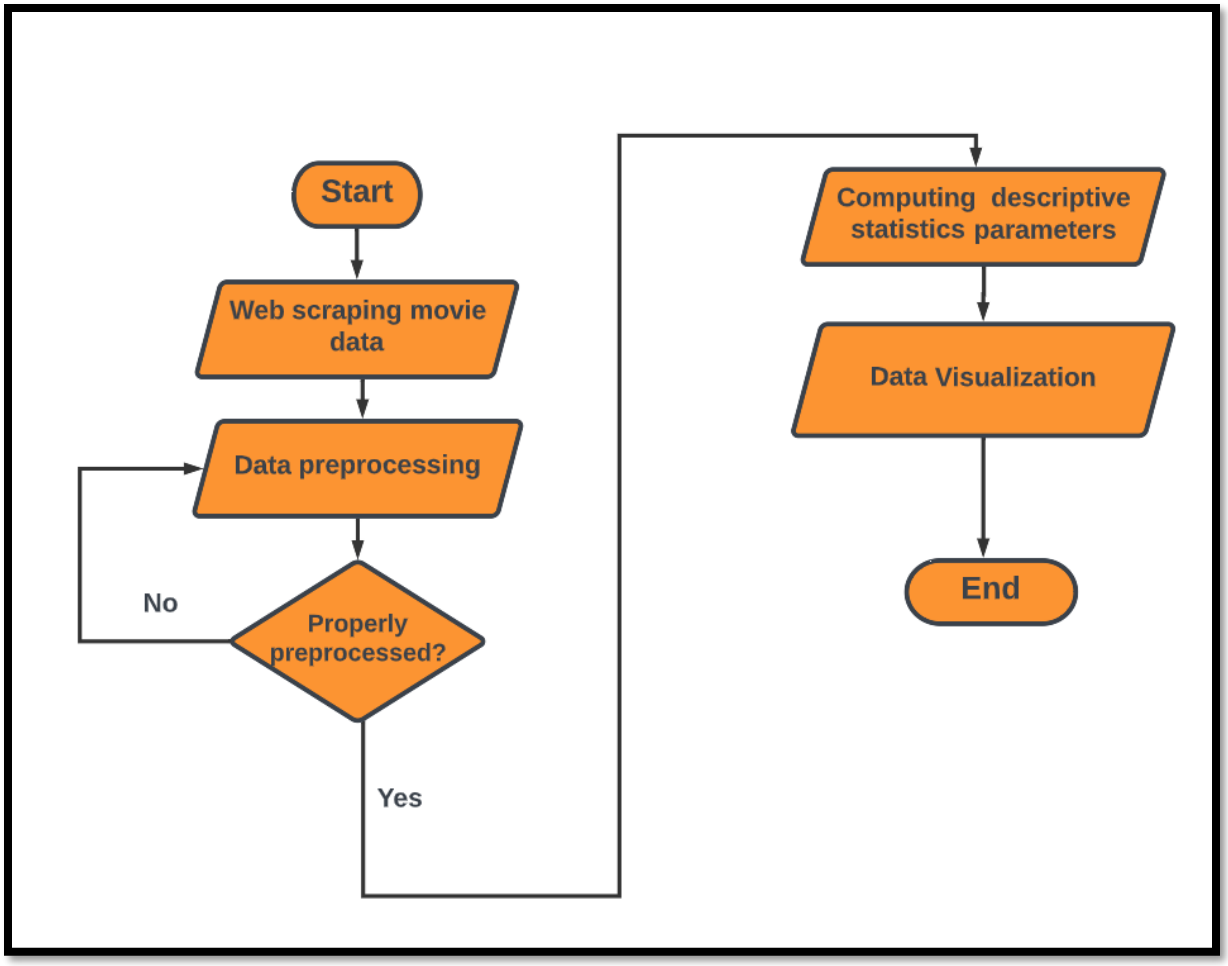


Fig. 1: Flowchart of our project Solution Design

Part 1: Web scraping

For this project we shall scrape movie data from two separate web pages. They are the following:

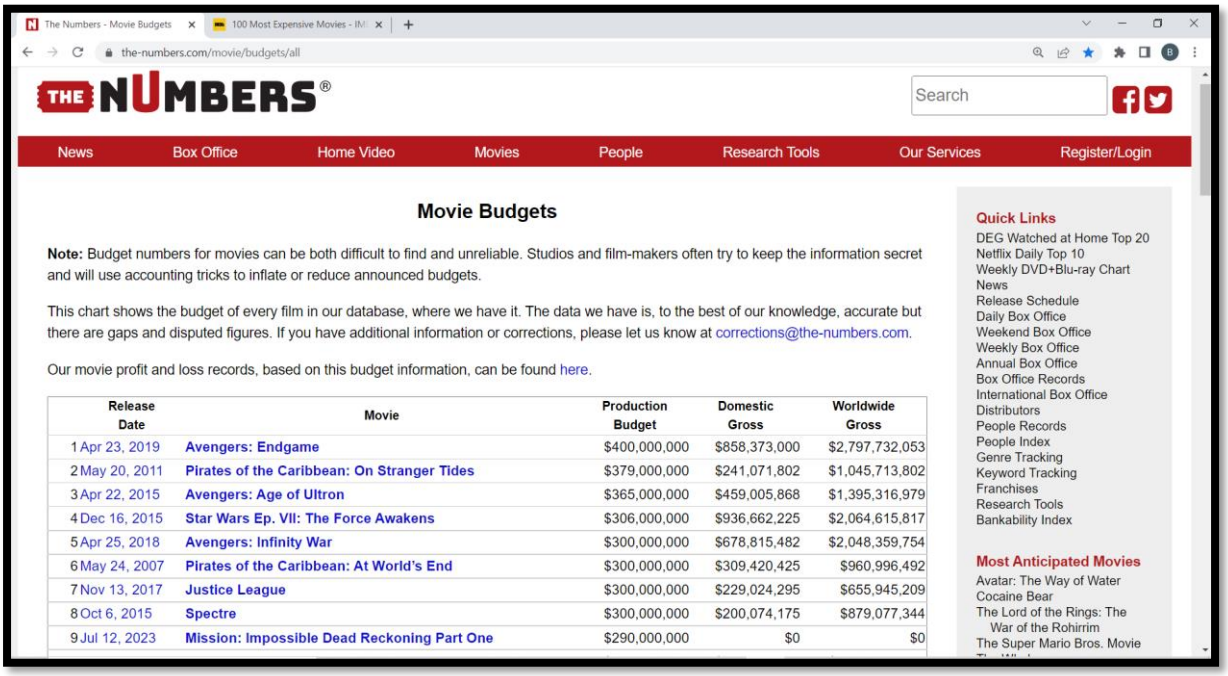


Fig. 2: First webpage to be scraped (https://www.the-numbers.com/movie/budgets/all)

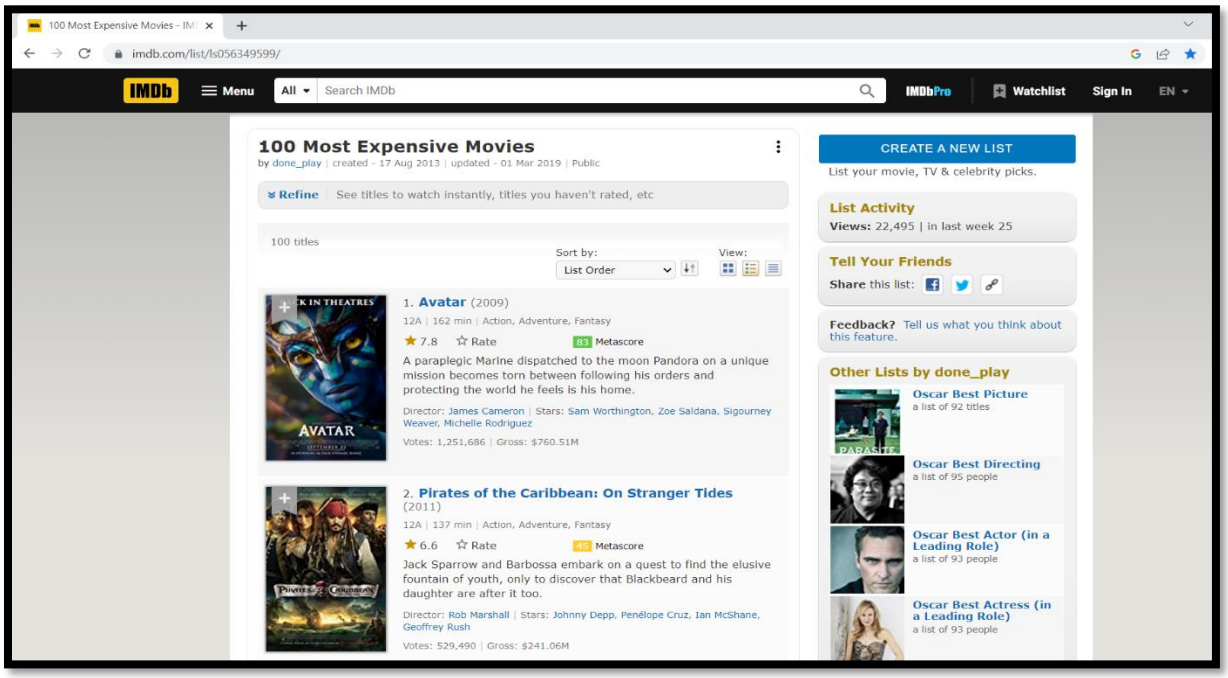


Fig. 3: Second webpage to be scraped (https://www.imdb.com/list/ls056349599/)

Selectorgadget:

For ease of scraping we shall use an extension known as selector gadget, using it we can simply select data on a webpage and it will determine its HTML/CSS tags, ids and classes.

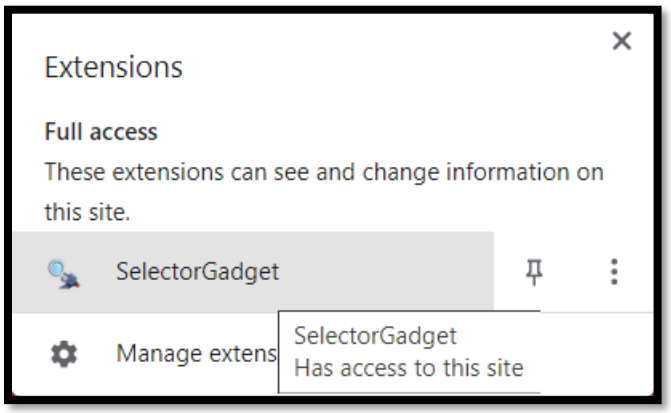


Fig. 4: Selectorgadget browser extension

Choosing HTML/CSS selectors to scrape(First Webpage):

From the first webpage we will scrap the names of the movies, their release dates, budgets and gross(earning) figures.

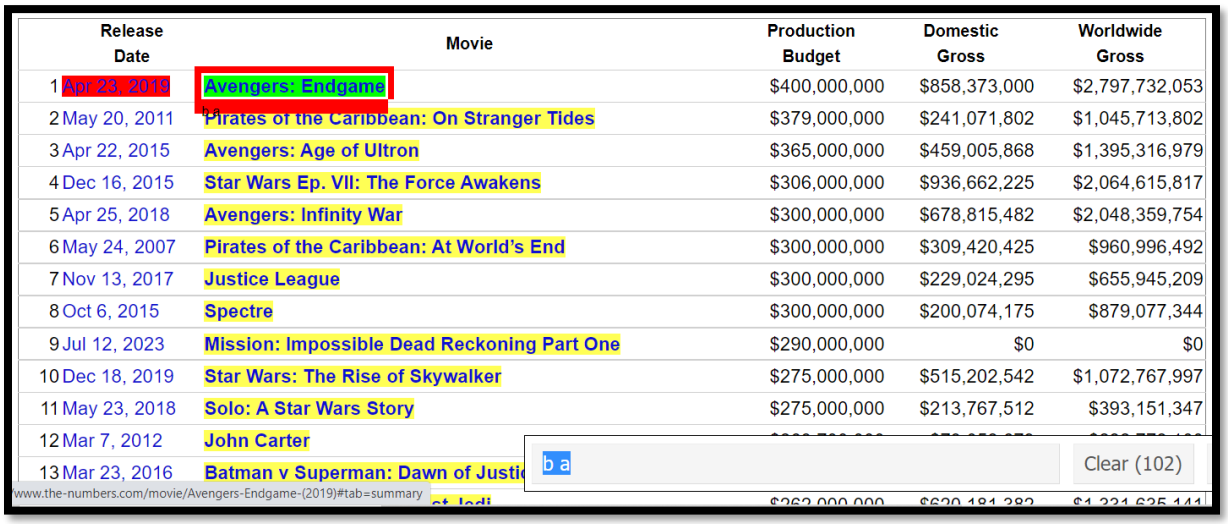
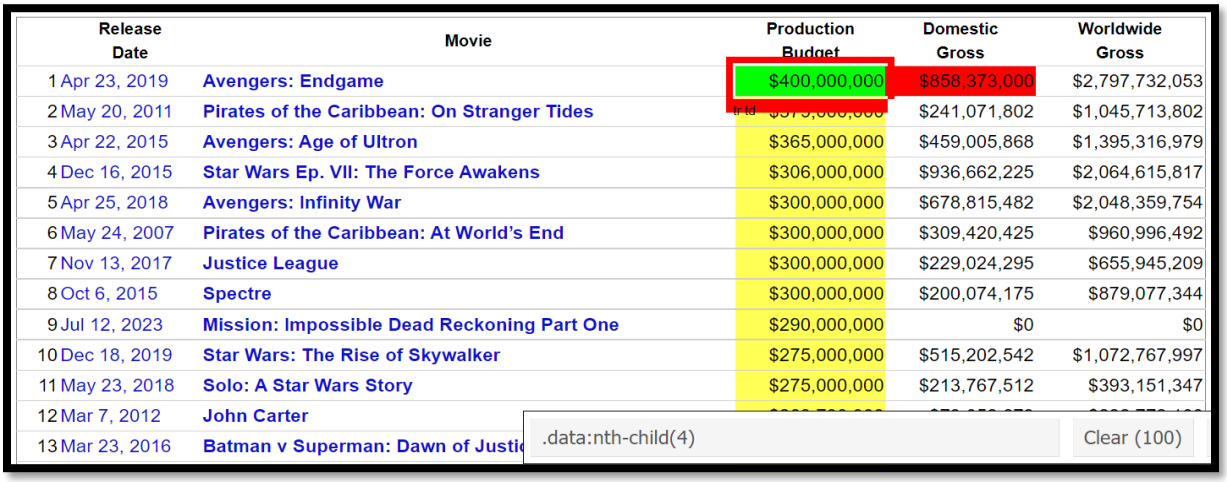


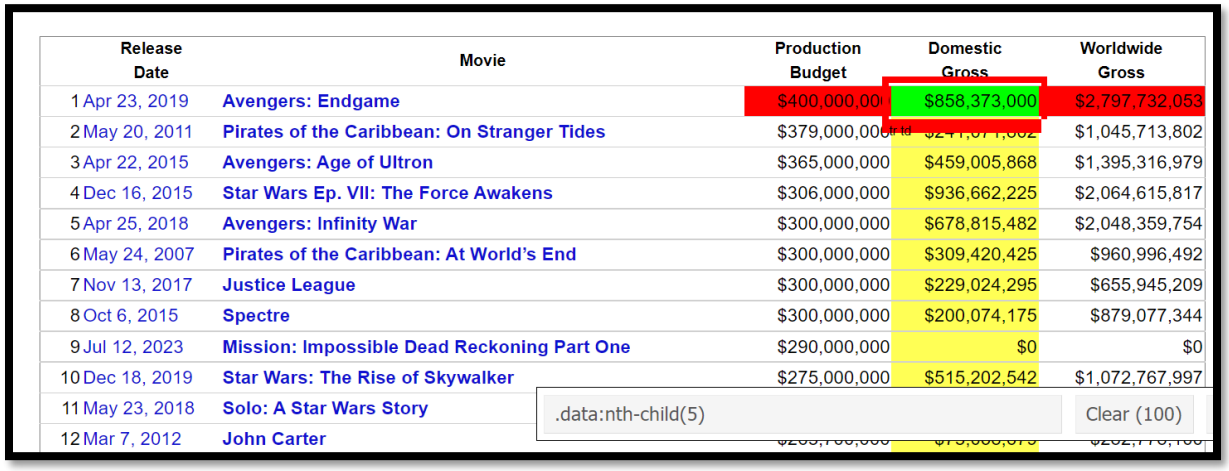
Fig. 5: Selecting movie names



	Release Date	Movie	Production Budget	Domestic Gross	Worldwide Gross
1	Apr 23, 2019	Avengers: Endgame	$400,000,000	$858,373,000	$2,797,732,053
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides		$241,071,802	$1,045,713,802
3	Apr 22, 2015	Avengers: Age of Ultron	$365,000,000	$459,005,868	$1,395,316,979
4	Dec 16, 2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	$936,662,225	$2,064,615,817
5	Apr 25, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,359,754
6	May 24, 2007	Pirates of the Caribbean: At World's End	$300,000,000	$309,420,425	$960,996,492
7	Nov 13, 2017	Justice League	$300,000,000	$229,024,295	$655,945,209
8	Oct 6, 2015	Spectre	$300,000,000	$200,074,175	$879,077,344
9	Jul 12, 2023	Mission: Impossible Dead Reckoning Part One	$290,000,000	$0	$0
10	Dec 18, 2019	Star Wars: The Rise of Skywalker	$275,000,000	$515,202,542	$1,072,767,997
11	May 23, 2018	Solo: A Star Wars Story	$275,000,000	$213,767,512	$393,151,347
12	Mar 7, 2012	John Carter			
13	Mar 23, 2016	Batman v Superman: Dawn of Justi			

.data:nth-child(4) Clear (100)

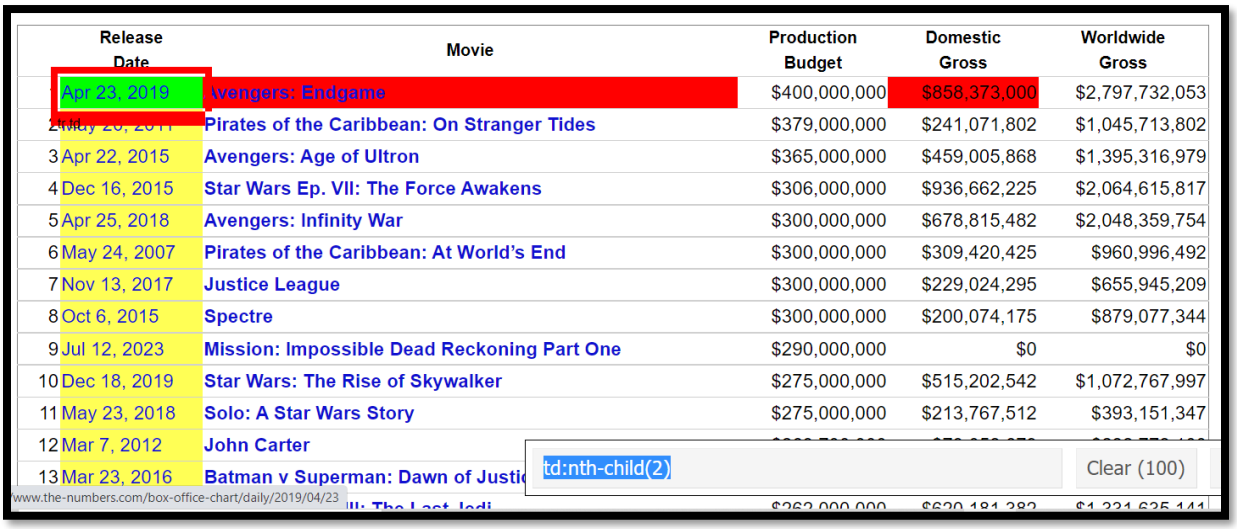
Fig. 6: Selecting movie budgets



	Release Date	Movie	Production Budget	Domestic Gross	Worldwide Gross
1	Apr 23, 2019	Avengers: Endgame	$400,000,000	$858,373,000	$2,797,732,053
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$379,000,000		$1,045,713,802
3	Apr 22, 2015	Avengers: Age of Ultron	$365,000,000	$459,005,868	$1,395,316,979
4	Dec 16, 2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	$936,662,225	$2,064,615,817
5	Apr 25, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,359,754
6	May 24, 2007	Pirates of the Caribbean: At World's End	$300,000,000	$309,420,425	$960,996,492
7	Nov 13, 2017	Justice League	$300,000,000	$229,024,295	$655,945,209
8	Oct 6, 2015	Spectre	$300,000,000	$200,074,175	$879,077,344
9	Jul 12, 2023	Mission: Impossible Dead Reckoning Part One	$290,000,000	$0	$0
10	Dec 18, 2019	Star Wars: The Rise of Skywalker	$275,000,000	$515,202,542	$1,072,767,997
11	May 23, 2018	Solo: A Star Wars Story			
12	Mar 7, 2012	John Carter			

.data:nth-child(5) Clear (100)

Fig. 7: Selecting movie gross figures(domestic)



	Release Date	Movie	Production Budget	Domestic Gross	Worldwide Gross
	Apr 23, 2019	Avengers: Endgame	$400,000,000	$858,373,000	$2,797,732,053
2		Pirates of the Caribbean: On Stranger Tides	$379,000,000	$241,071,802	$1,045,713,802
3	Apr 22, 2015	Avengers: Age of Ultron	$365,000,000	$459,005,868	$1,395,316,979
4	Dec 16, 2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	$936,662,225	$2,064,615,817
5	Apr 25, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,359,754
6	May 24, 2007	Pirates of the Caribbean: At World's End	$300,000,000	$309,420,425	$960,996,492
7	Nov 13, 2017	Justice League	$300,000,000	$229,024,295	$655,945,209
8	Oct 6, 2015	Spectre	$300,000,000	$200,074,175	$879,077,344
9	Jul 12, 2023	Mission: Impossible Dead Reckoning Part One	$290,000,000	$0	$0
10	Dec 18, 2019	Star Wars: The Rise of Skywalker	$275,000,000	$515,202,542	$1,072,767,997
11	May 23, 2018	Solo: A Star Wars Story	$275,000,000	$213,767,512	$393,151,347
12	Mar 7, 2012	John Carter			
13	Mar 23, 2016	Batman v Superman: Dawn of Justi			

td:nth-child(2) Clear (100)

www.the-numbers.com/box-office-chart/daily/2019/04/23

Fig. 8: Selecting movie release dates

Retrieving all these data and storing them using R language:

Now we will run R language codes to store these data into suitable data structures. We will run the following code for this purpose.

Code:

```
#A : We will use the rvest library for scraping or harvesting web data

#B : read_html() function will read the entire webpage and store it in a variable

#C : html_nodes() function will return the specific nodes/tags requested from the webpage

#D : html_text() will return the specific text in those returned tags


library(rvest) #A

web_page <- read_html("https://www.the-numbers.com/movie/budgets/all")  #B

movie_name<- html_nodes(web_page, "b a") #C

name<-html_text(movie_name) #D


name<-name[1:100] # cancelling out some unwanted scraped data

#library(rvest) #A

web_page <- read_html("https://www.the-numbers.com/movie/budgets/all") #B

movie_release_date<- html_nodes(web_page, "td:nth-child(2)") #C

release_date<-html_text(movie_release_date) #D


#library(rvest) #A

web_page <- read_html("https://www.the-numbers.com/movie/budgets/all") #B

movie_budget<- html_nodes(web_page, ".data:nth-child(4)") #C

budget<-html_text(movie_budget) #D
```

```
#library(rvest) #A

web_page <- read_html("https://www.the-numbers.com/movie/budgets/all") #B

movie_gross<- html_nodes(web_page, ".data:nth-child(5)") #C

gross<-html_text(movie_gross) #D


dataset1<-data.frame(name,release_date,budget,gross, stringsAsFactors = FALSE)

# storing all the vectors(name,release_date,budget and gross) into a single dataframe

# as of now we don't want numeric values of our string so we set stringsAsFactors = FALSE

View(dataset1)
```

dataset1 | dataset2 | Final Project.R* | friday.R | saturday.R | script 2.R

Filter

	name	release_date	budget	gross
1	Avengers: Endgame	Apr 23, 2019	$400,000,000	$858,373,000
2	Pirates of the Caribbean: On Stranger Tides	May 20, 2011	$379,000,000	$241,071,802
3	Avengers: Age of Ultron	Apr 22, 2015	$365,000,000	$459,005,868
4	Star Wars Ep. VII: The Force Awakens	Dec 16, 2015	$306,000,000	$936,662,225
5	Avengers: Infinity War	Apr 25, 2018	$300,000,000	$678,815,482
6	Pirates of the Caribbean: At World's End	May 24, 2007	$300,000,000	$309,420,425
7	Justice League	Nov 13, 2017	$300,000,000	$229,024,295
8	Spectre	Oct 6, 2015	$300,000,000	$200,074,175
9	Mission: Impossible Dead Reckoning Part One	Jul 12, 2023	$290,000,000	$0
10	Star Wars: The Rise of Skywalker	Dec 18, 2019	$275,000,000	$515,202,542
11	Solo: A Star Wars Story	May 23, 2018	$275,000,000	$213,767,512
12	John Carter	Mar 7, 2012	$263,700,000	$73,058,679

Showing 1 to 13 of 100 entries, 4 total columns

Fig. 9: dataset1 for first webpage

Choosing HTML/CSS selectors to scrape(Second Webpage):

Fig. 10: Selecting movie names

Fig. 11: Selecting movie genres

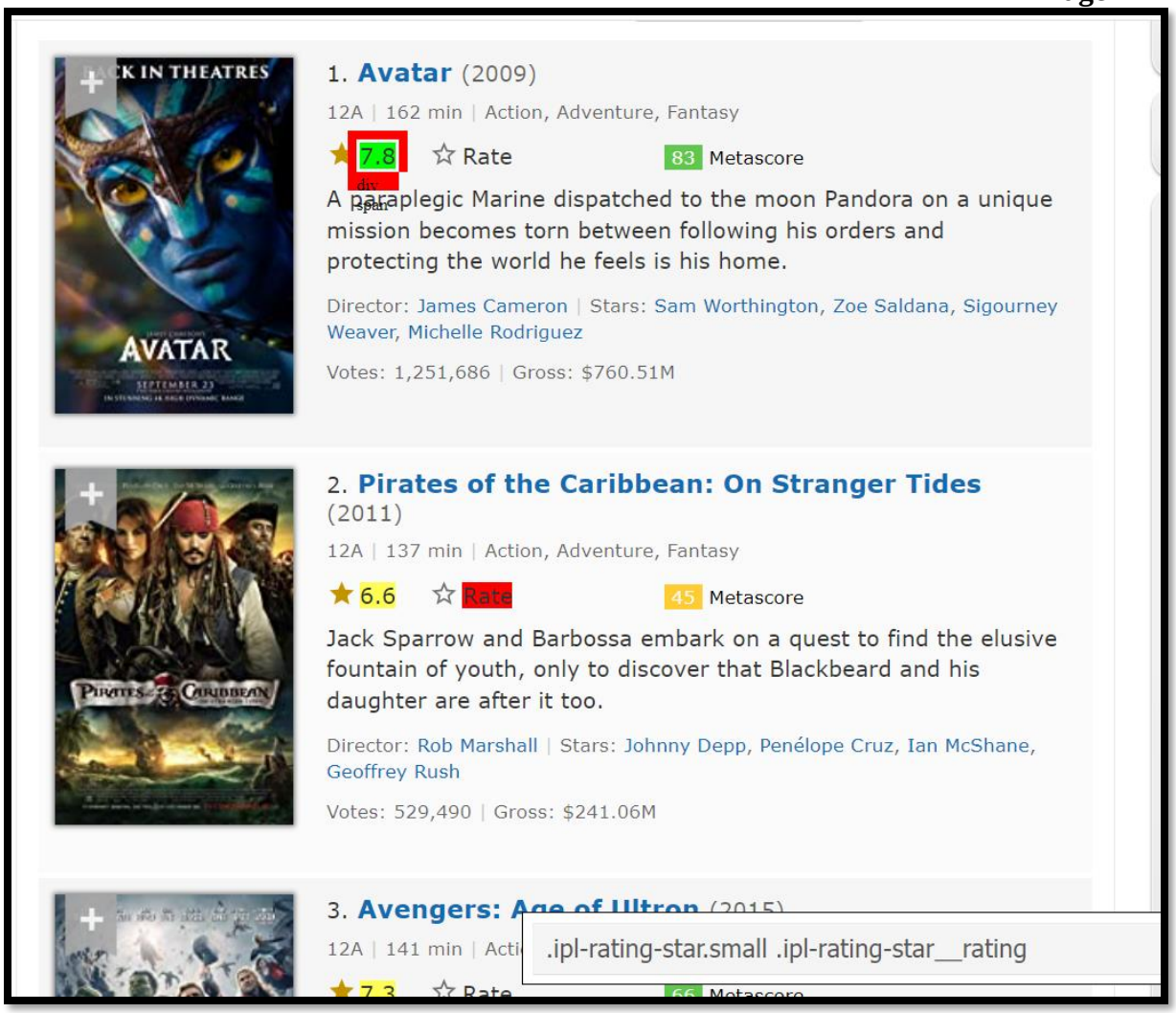


Fig. 12: Selecting movie ratings

Retrieving all these data and storing them using R language:

Like the previous case we will run R codes to store these data into suitable data structures(vectors) and then data frames. We will run the following code for this purpose.

Code:

```
#A : We will use the rvest library for scraping or harvesting web data
#B : read_html() function will read the entire webpage and store it in a variable
#C : html_nodes() function will return the specific nodes/tags requested from the webpage
#D : html_text() will return the specific text in those returned tags

library(rvest) #A
web_page <- read_html("https://www.imdb.com/list/ls056349599/") #B
movie_name<- html_nodes(web_page, ".lister-item-header a") #C
name<-html_text(movie_name) #D
```

```
#library(rvest) #A
web_page <- read_html("https://www.imdb.com/list/ls056349599/") #B
movie_genre<- html_nodes(web_page, ".genre") #C
genre<-html_text(movie_genre) #D
```

```
#library(rvest) #A
web_page <- read_html("https://www.imdb.com/list/ls056349599/") #B
movie_rating<- html_nodes(web_page, ".ipl-rating-star.small .ipl-rating-star__rating") #C
rating<-html_text(movie_rating) #D

dataset2<-data.frame(name,genre,rating, stringsAsFactors = FALSE)
# storing all the vectors(name,release_date,budget and gross) into a single dataframe
# as of now we don't want numeric values of our string so we set stringsAsFactors = FALSE

View(dataset2)
```

dataset1 | dataset2 | Final Project.R* | friday.R | saturday.R | script 2.R | final

Filter

	name	genre	rating
1	Avatar	Action, Adventure, Fantasy	7.8
2	Pirates of the Caribbean: On Stranger Tides	Action, Adventure, Fantasy	6.6
3	Avengers: Age of Ultron	Action, Adventure, Sci-Fi	7.3
4	Star Wars: Episode VIII - The Last Jedi	Action, Adventure, Fantasy	6.9
5	Star Wars: Episode VII - The Force Awakens	Action, Adventure, Sci-Fi	7.8
6	Avengers: Infinity War	Action, Adventure, Sci-Fi	8.4
7	Pirates of the Caribbean: At World's End	Action, Adventure, Fantasy	7.1
8	Spectre	Action, Adventure, Thriller	6.8
9	Justice League	Action, Adventure, Fantasy	6.1
10	The Dark Knight Rises	Action, Drama	8.4
11	John Carter	Action, Adventure, Sci-Fi	6.6
12	The Lone Ranger	Action, Adventure, Western	6.4

Showing 1 to 13 of 100 entries, 3 total columns

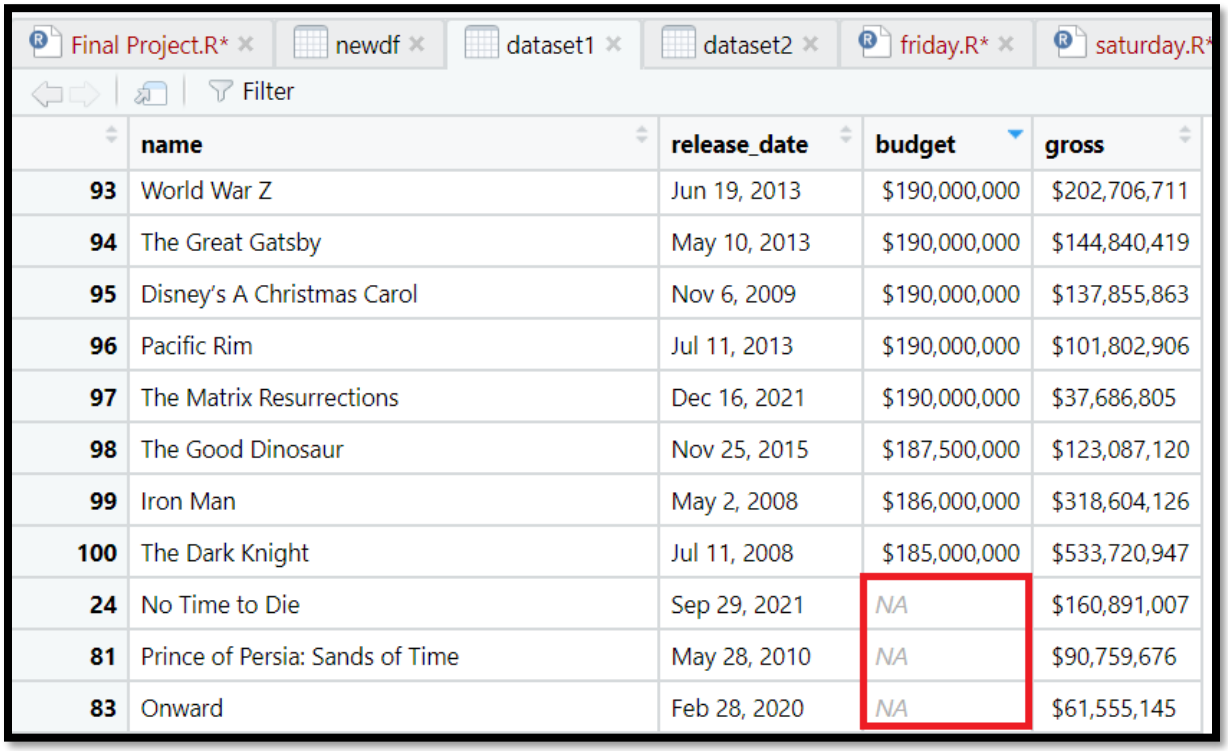
Fig. 13: dataset2 for second webpage

Part 2: Data preprocessing:

Now we shall perform preprocessing techniques on these two datasets so as to have a prepared dataset for statistical analysis and visualization.

Data cleaning(Handling Missing Data)(dataset1):

From our first dataset we can see some records having null budgets. We shall omit these from the dataset entirely.



	name	release_date	budget	gross
93	World War Z	Jun 19, 2013	$190,000,000	$202,706,711
94	The Great Gatsby	May 10, 2013	$190,000,000	$144,840,419
95	Disney's A Christmas Carol	Nov 6, 2009	$190,000,000	$137,855,863
96	Pacific Rim	Jul 11, 2013	$190,000,000	$101,802,906
97	The Matrix Resurrections	Dec 16, 2021	$190,000,000	$37,686,805
98	The Good Dinosaur	Nov 25, 2015	$187,500,000	$123,087,120
99	Iron Man	May 2, 2008	$186,000,000	$318,604,126
100	The Dark Knight	Jul 11, 2008	$185,000,000	$533,720,947
24	No Time to Die	Sep 29, 2021	*NA*	$160,891,007
81	Prince of Persia: Sands of Time	May 28, 2010	*NA*	$90,759,676
83	Onward	Feb 28, 2020	*NA*	$61,555,145

Fig. 14: NA values for budget column in dataset1

Code(Omitting NA rows)(dataset1):

```
library(sqldf) #here we are using the sqldf library,
dataset1<-sqldf("select * from dataset1 where budget is not null")
#using this library we can manipulate our data like database tables
#We have passed a query which will only keep the rows where budget is not null/NA
 View(dataset1)
```

dataset1 | Final Project.R* | newdf | dataset2 | friday.R* | saturday.R*

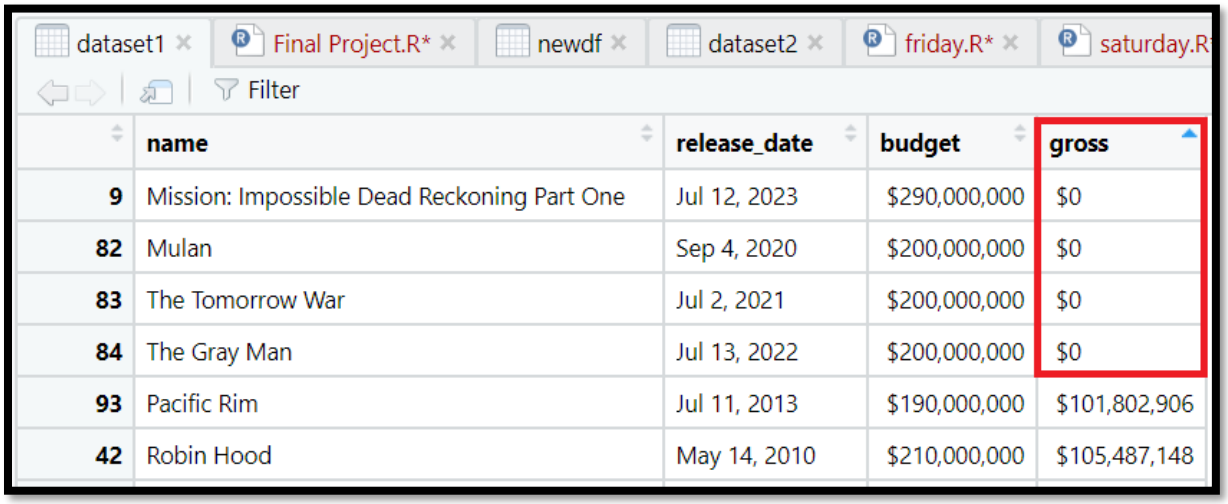
Filter

	name	release_date	budget	gross
10	Star Wars: The Rise of Skywalker	Dec 18, 2019	$275,000,000	$515,202,542
11	Solo: A Star Wars Story	May 23, 2018	$275,000,000	$213,767,512
9	Mission: Impossible Dead Reckoning Part One	Jul 12, 2023	$290,000,000	$0
5	Avengers: Infinity War	Apr 25, 2018	$300,000,000	$678,815,482
6	Pirates of the Caribbean: At World's End	May 24, 2007	$300,000,000	$309,420,425
7	Justice League	Nov 13, 2017	$300,000,000	$229,024,295
8	Spectre	Oct 6, 2015	$300,000,000	$200,074,175
4	Star Wars Ep. VII: The Force Awakens	Dec 16, 2015	$306,000,000	$936,662,225
3	Avengers: Age of Ultron	Apr 22, 2015	$365,000,000	$459,005,868
2	Pirates of the Caribbean: On Stranger Tides	May 20, 2011	$379,000,000	$241,071,802
1	Avengers: Endgame	Apr 23, 2019	$400,000,000	$858,373,000

Fig. 15: NA values removed from dataset1

Data cleaning(Smooth Noisy Data)(dataset1):

For many of the gross values of our dataset1 we have 0 which is unusually low, so we shall omit these observations as well.


dataset1 | Final Project.R* | newdf | dataset2 | friday.R* | saturday.R

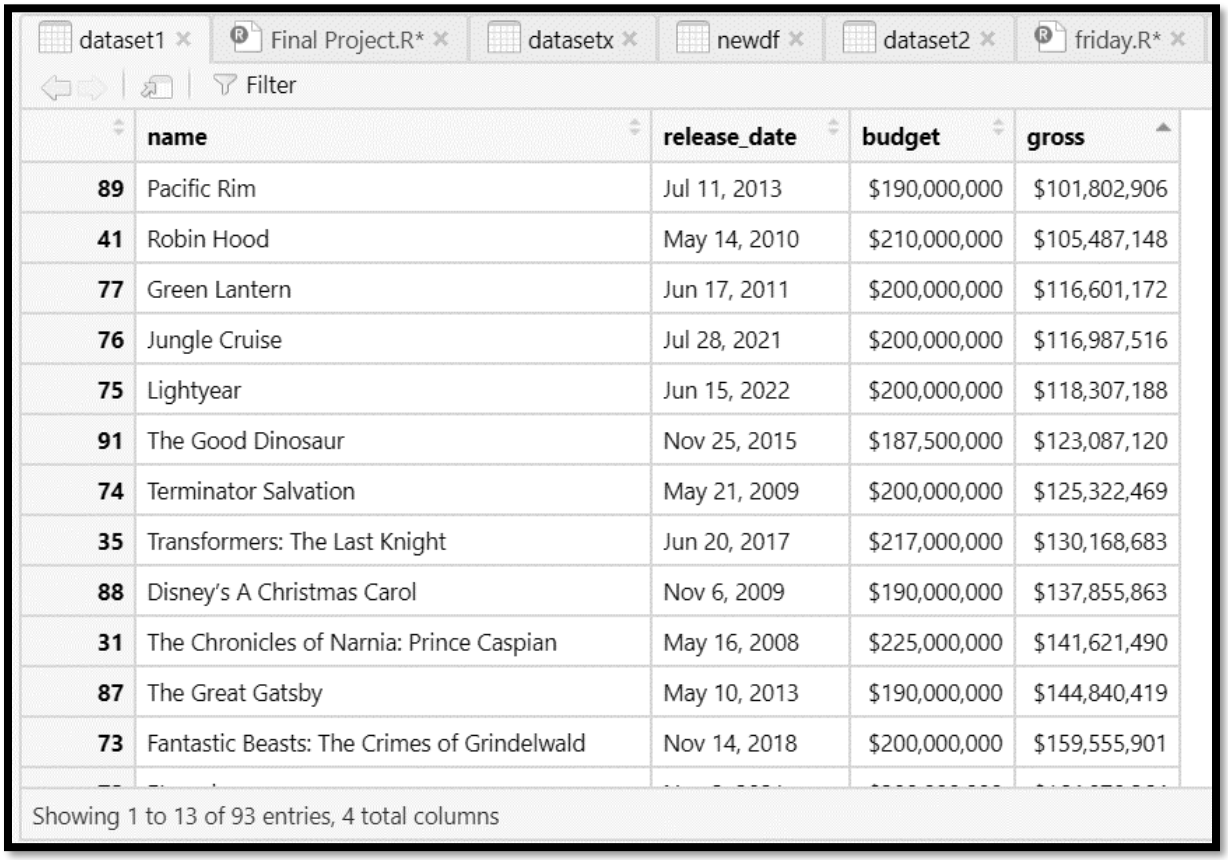
Filter

	name	release_date	budget	gross
9	Mission: Impossible Dead Reckoning Part One	Jul 12, 2023	$290,000,000	$0
82	Mulan	Sep 4, 2020	$200,000,000	$0
83	The Tomorrow War	Jul 2, 2021	$200,000,000	$0
84	The Gray Man	Jul 13, 2022	$200,000,000	$0
93	Pacific Rim	Jul 11, 2013	$190,000,000	$101,802,906
42	Robin Hood	May 14, 2010	$210,000,000	$105,487,148

Fig. 16: 0 (noisy) values for gross figures of released/unreleased movies

Code(Handling noisy data)(dataset1):

```
library(sqldf)

dataset1<-sqldf("select * from dataset1 where name!='$0'")

#here we are using the sqldf library, using this we can manipulate our data like database #tables

#Here we have passed a query which will keep the rows from dataset1 where budget is not #$0 and return it to dataset1

View(dataset1)
```



dataset1 | Final Project.R* | datasetx | newdf | dataset2 | friday.R*

Filter

	name	release_date	budget	gross
89	Pacific Rim	Jul 11, 2013	$190,000,000	$101,802,906
41	Robin Hood	May 14, 2010	$210,000,000	$105,487,148
77	Green Lantern	Jun 17, 2011	$200,000,000	$116,601,172
76	Jungle Cruise	Jul 28, 2021	$200,000,000	$116,987,516
75	Lightyear	Jun 15, 2022	$200,000,000	$118,307,188
91	The Good Dinosaur	Nov 25, 2015	$187,500,000	$123,087,120
74	Terminator Salvation	May 21, 2009	$200,000,000	$125,322,469
35	Transformers: The Last Knight	Jun 20, 2017	$217,000,000	$130,168,683
88	Disney's A Christmas Carol	Nov 6, 2009	$190,000,000	$137,855,863
31	The Chronicles of Narnia: Prince Caspian	May 16, 2008	$225,000,000	$141,621,490
87	The Great Gatsby	May 10, 2013	$190,000,000	$144,840,419
73	Fantastic Beasts: The Crimes of Grindelwald	Nov 14, 2018	$200,000,000	$159,555,901

Showing 1 to 13 of 93 entries, 4 total columns

Fig. 16: Noisy values removed from dataset1

Data Transformation(dataset1):

In dataset1 the variables budget and gross are in character format, having alphanumeric values with '$' and ','. We will convert these into pure numeric forms using Data transformation techniques.

Code(Data Transformation)(dataset1):

```
library(stringr)
library(dplyr)
dataset1 <- within(dataset1,{
  budget<- str_remove_all(budget,"\\s+") #removing whitespace
  budget<- str_remove_all(budget,",")    #removing comma
  budget<- str_remove_all(budget,"\\$")  #removing $ sign

  gross<- str_remove_all(gross,"\\s+") #removing whitespace
  gross<- str_remove_all(gross,",")    #removing comma
  gross<- str_remove_all(gross,"\\$")  #removing $ sign

   num_budget<-strtoi(budget) #converting budgets from char to int
  num_gross<-strtoi(gross)   #converting gross values from char to int
  release_date<-as.Date(release_date,format="%b %d, %Y")
  #formatting the dates from %b %d, %Y to default format
  })

columns <- names(dataset1) %in% c("budget", "gross")
dataset1 <- dataset1[!columns]
#first we are selecting the variable names to be deleted then were deleting those columns
#entirely
dataset1 <- rename(dataset1, budget = "num_budget", gross = "num_gross")
#renaming the new columns
```

dataset1 | Final Project.R* | datasetx | newdf | dataset2 | friday

Filter

	name	release_date	budget	gross
1	Avengers: Endgame	2019-04-23	400000000	858373000
2	Pirates of the Caribbean: On Stranger Tides	2011-05-20	379000000	241071802
3	Avengers: Age of Ultron	2015-04-22	365000000	459005868
4	Star Wars Ep. VII: The Force Awakens	2015-12-16	306000000	936662225
5	Avengers: Infinity War	2018-04-25	300000000	678815482
6	Pirates of the Caribbean: At World's End	2007-05-24	300000000	309420425
7	Justice League	2017-11-13	300000000	229024295
8	Spectre	2015-10-06	300000000	200074175
9	Star Wars: The Rise of Skywalker	2019-12-18	275000000	515202542
10	Solo: A Star Wars Story	2018-05-23	275000000	213767512
11	John Carter	2012-03-07	263700000	73058679
12	Batman v Superman: Dawn of Justice	2016-03-23	263000000	330360194

Showing 1 to 13 of 93 entries, 4 total columns

Fig. 17: dataset1 after transformation

Data Transformation(dataset2):

For dataset2 we will simplify genre by decomposing them into a single genre for each row.

Code:

```
library(stringr)
library(dplyr)
  #Using the within function to recode variables of dataset2
dataset2 <- within(dataset2,{

  genre<- str_remove_all(genre,"\\s+") #removing whitespace
  genre[genre=='Action,Adventure,Comedy']<-"Action"
  genre[genre=='Action,Adventure,Sci-Fi']<-"Action"
  genre[genre=='Action,Comedy,Crime']<-"Action"
  genre[genre=='Action,Drama']<-"Action"
  genre[genre=='Animation,Adventure,Comedy']<-"Animation"
  genre[genre=='Animation,Action,Adventure']<-"Animation"
  genre[genre=='Animation,Adventure,Family']<-"Animation"
  genre[genre=='Adventure,Drama,Family']<-"Adventure"
  genre[genre=='Action,Adventure,Fantasy']<-"Adventure"
  genre[genre=='Action,Adventure']<-"Adventure"
  genre[genre=='Action,Adventure,Horror']<-"Adventure"
  genre[genre=='Adventure,Family,Fantasy']<-"Fantasy"
  genre[genre=='Adventure,Fantasy']<-"Fantasy"
  genre[genre=='Action,Adventure,Family']<-"Fantasy"
  genre[genre=='Action,Adventure,Thriller']<-"Thriller"
  genre[genre=='Drama,Romance']<-"Drama"})
#genres that were misplaced were fixed by data munging
```

Code:

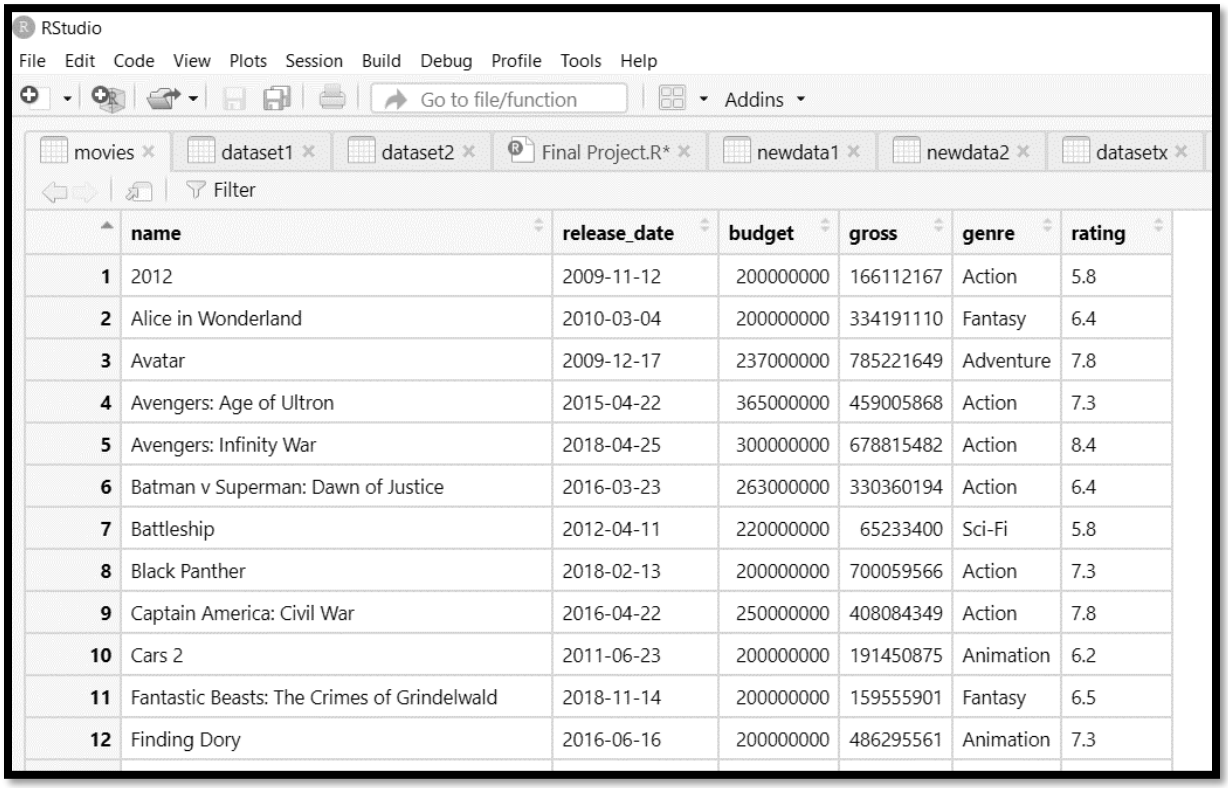
```
library(dplyr)
  #Transforming,replacing and renaming the ratings variable

columns <- names(dataset2) %in% c("rating")
dataset2 <- dataset2[!columns]
dataset2$num_rating<-as.double(dataset2$rating)
columns <- names(dataset2) %in% c("rating")
dataset2 <- dataset2[!columns]
dataset2 <- rename(dataset2, rating = "num_rating")
```

Data Integration(dataset1 and dataset2):

Now we will merge these two datasets where the movie names match.

Code(Data Integration):

```
newdata1 <- subset(dataset1, name %in% dataset2$name,)
newdata2 <- subset(dataset2, name %in% dataset1$name,)
#we are taking two subsets from the datasets where the movie names match
movies<-merge(newdata1, newdata2, by="name")
#we are merging the two datasets by their common names
View(movies)
```



	name	release_date	budget	gross	genre	rating
1	2012	2009-11-12	200000000	166112167	Action	5.8
2	Alice in Wonderland	2010-03-04	200000000	334191110	Fantasy	6.4
3	Avatar	2009-12-17	237000000	785221649	Adventure	7.8
4	Avengers: Age of Ultron	2015-04-22	365000000	459005868	Action	7.3
5	Avengers: Infinity War	2018-04-25	300000000	678815482	Action	8.4
6	Batman v Superman: Dawn of Justice	2016-03-23	263000000	330360194	Action	6.4
7	Battleship	2012-04-11	220000000	65233400	Sci-Fi	5.8
8	Black Panther	2018-02-13	200000000	700059566	Action	7.3
9	Captain America: Civil War	2016-04-22	250000000	408084349	Action	7.8
10	Cars 2	2011-06-23	200000000	191450875	Animation	6.2
11	Fantastic Beasts: The Crimes of Grindelwald	2018-11-14	200000000	159555901	Fantasy	6.5
12	Finding Dory	2016-06-16	200000000	486295561	Animation	7.3

Fig. 18: dataset1 and dataset2 merged into movies dataset

Data Reduction(movies dataset):

From our movies dataset we can see that the budget and gross figures are in millions and occupy 9-10 digits each. We can reduce them by decimal scaling. We will divide all these figures so as to show the figures in millions. Such as, a gross value of 700 million instead of 700000000.

Code(data reduction):

```
library(dplyr)
movies <- within(movies,{

 budget<-round(budget/1000000) #rounding/reducing the budgets

 gross<-round(gross/1000000) #rounding/reducing the gross values
})

movies <- rename(movies, budget_millions = "budget", gross_millions = "gross")
#renaming the variables
```

movies | dataset1 | dataset2 | Final Project.R* | newdata1 | newdata2 | datasetx | friday.R

Filter

	name	release_date	budget_millions	gross_millions	genre	rating
1	2012	2009-11-12	200	166	Action	5.8
2	Alice in Wonderland	2010-03-04	200	334	Fantasy	6.4
3	Avatar	2009-12-17	237	785	Adventure	7.8
4	Avengers: Age of Ultron	2015-04-22	365	459	Action	7.3
5	Avengers: Infinity War	2018-04-25	300	679	Action	8.4
6	Batman v Superman: Dawn of Justice	2016-03-23	263	330	Action	6.4
7	Battleship	2012-04-11	220	65	Sci-Fi	5.8
8	Black Panther	2018-02-13	200	700	Action	7.3
9	Captain America: Civil War	2016-04-22	250	408	Action	7.8
10	Cars 2	2011-06-23	200	191	Animation	6.2

Fig. 19: movies dataset after reduction of budget and gross variables

Our preprocessing stage is over and now we have a clean preprocessed movies dataset. It has 6 variables and 63 records which we shall use for further analysis.

movies | dataset1 | dataset2 | Final Project.R* | newdata1 | newdata2 | datasetx | friday.R

Filter

	name	release_date	budget_millions	gross_millions	genre	rating
1	2012	2009-11-12	200	166	Action	5.8
2	Alice in Wonderland	2010-03-04	200	334	Fantasy	6.4
3	Avatar	2009-12-17	237	785	Adventure	7.8
4	Avengers: Age of Ultron	2015-04-22	365	459	Action	7.3
5	Avengers: Infinity War	2018-04-25	300	679	Action	8.4
6	Batman v Superman: Dawn of Justice	2016-03-23	263	330	Action	6.4
7	Battleship	2012-04-11	220	65	Sci-Fi	5.8
8	Black Panther	2018-02-13	200	700	Action	7.3
9	Captain America: Civil War	2016-04-22	250	408	Action	7.8
10	Cars 2	2011-06-23	200	191	Animation	6.2
11	Fantastic Beasts: The Crimes of Grindelwald	2018-11-14	200	160	Fantasy	6.5
12	Finding Dory	2016-06-16	200	486	Animation	7.3
13	Green Lantern	2011-06-17	200	117	Action	5.5
14	Harry Potter and the Half-Blood Prince	2009-07-15	250	302	Fantasy	7.6
15	Incredibles 2	2018-06-15	200	609	Animation	7.6
16	Iron Man	2008-05-02	186	319	Action	7.9
17	Jack the Giant Slayer	2013-02-27	195	65	Adventure	6.3
18	John Carter	2012-03-07	264	73	Action	6.6
19	Jurassic World	2015-06-09	215	652	Action	6.9
20	Justice League	2017-11-13	300	229	Adventure	6.1

Fig. 20: Completely preprocessed movies dataset

Now we shall save this preprocessed datset into a .csv file and save it.

Code:

```
write.csv(movies,"C:\\Users\\Asus\\Desktop\\Semester 9\\Introduction to Data Science\\movies.csv" )

#This dataset can be downloaded from the following link:
# https://drive.google.com/drive/u/0/folders/1h2zPQPwaBRlKXHZYc5N1Vlcme2Cl6esa
```

Applying descriptive statistics:

Now we shall compute various descriptive statistics parameters for our dataset.

First let's try to inspect the central tendency for the various variables of our dataset.

Code:

```
######## making the mode function

mode <- function(x) {
   unique_values <- unique(x)
   table <- tabulate(match(x, unique_values))
   unique_values[table ==max(table)]
 }
######## mean

 mean(movies$rating)
 mean(movies$gross_millions)
 mean(movies$budget_millions)

 ######## median
 median(movies$rating)
 median(movies$gross_millions)
 median(movies$budget_millions)

######## mode
 mode(movies$rating)
 mode(movies$gross_millions)
 mode(movies$budget_millions)
```

```
>  mean(movies$rating)
[1] 6.919048
>   mean(movies$gross_millions)
[1] 284.5873
>   mean(movies$budget_millions)
[1] 224.0635
>
>   ######## median
>   median(movies$rating)
[1] 6.8
>   median(movies$gross_millions)
[1] 234
>   median(movies$budget_millions)
[1] 210
>
>   ######## mode
>   mode(movies$rating)
[1] 6.6
>   mode(movies$gross_millions)
[1]  65 229 200 203 234
>   mode(movies$budget_millions)
[1] 200
>
```

Fig. 21: central tendency parameters

So by calculating the above parameters it is apparent that in general, most movies are profitable since their average gross is higher that their budget. Also, movies of our dataset have been rated 6.6-6.9 on average. Another thing is that the mean values are greater than the median values, so we can say this dataset is **right skewed.**

Variance and standard deviation:

Code:

```
var(movies$rating)

sd(movies$gross_millions)

sd(movies$budget_millions)

######## built-in functions
```

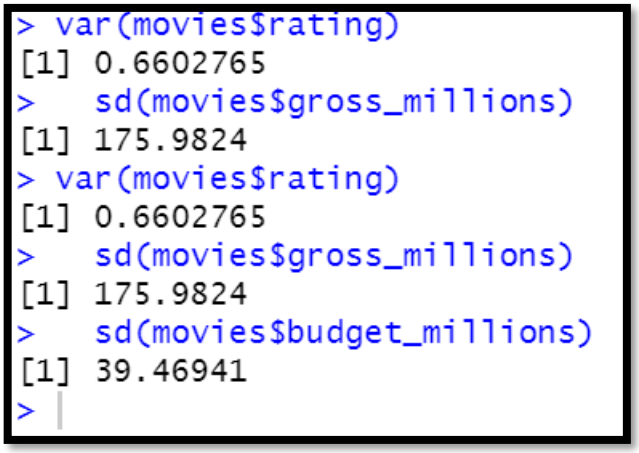


Fig. 22: Computing dispersion parameters

Here by computing dispersion we can say that the values for the ratings are closely clustered around the mean. In case of budget and gross the budgets are much less dispersed than the gross figures. This could be explained by the idea that movie budgets generally lie in a specific range while the earnings are much more dispersed due to the box office performances.

Quartiles:

If we divided our dataset variables into 4 divisions after sorting they would look something like the following:

Code:

```
quantile(movies$rating)

quantile(movies$gross_millions)

quantile(movies$budget_millions)

######## built-in functions
```

```
> quantile(movies$rating)
  0%  25%  50%  75% 100%
 5.2  6.4  6.8  7.6  9.0
>   quantile(movies$gross_millions)
    0%    25%    50%    75%   100%
 65.0 167.5 234.0 344.5 785.0
>   quantile(movies$budget_millions)
    0%    25%    50%    75%   100%
185.0 200.0 210.0 234.5 379.0
```

Fig. 23: Computing quartiles

Now we shall visualize the mean values with standard errors for all genres of movies of our dataset.

Code:

```
library(dplyr)

plotdata <- movies %>%

   group_by(genre) %>%

   summarize(n=n(), mean = mean(gross_millions), se = sd(gross_millions)/sqrt(n))

 #creating a plotdata where the mean earnings of the movies will be grouoped by genre

ggplot(plotdata, aes(x=reorder(genre, mean), y=mean)) +

 geom_bar(stat="identity", fill="green") +

 geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=0.2) +

  #Plotting the barcharts along with error bars

 labs(x="Genre", y="Earning", title = "Mean Domestic Earninga of Hollywood movies",

       subtitle = "with standard error bars")

#Setting up the labels
```

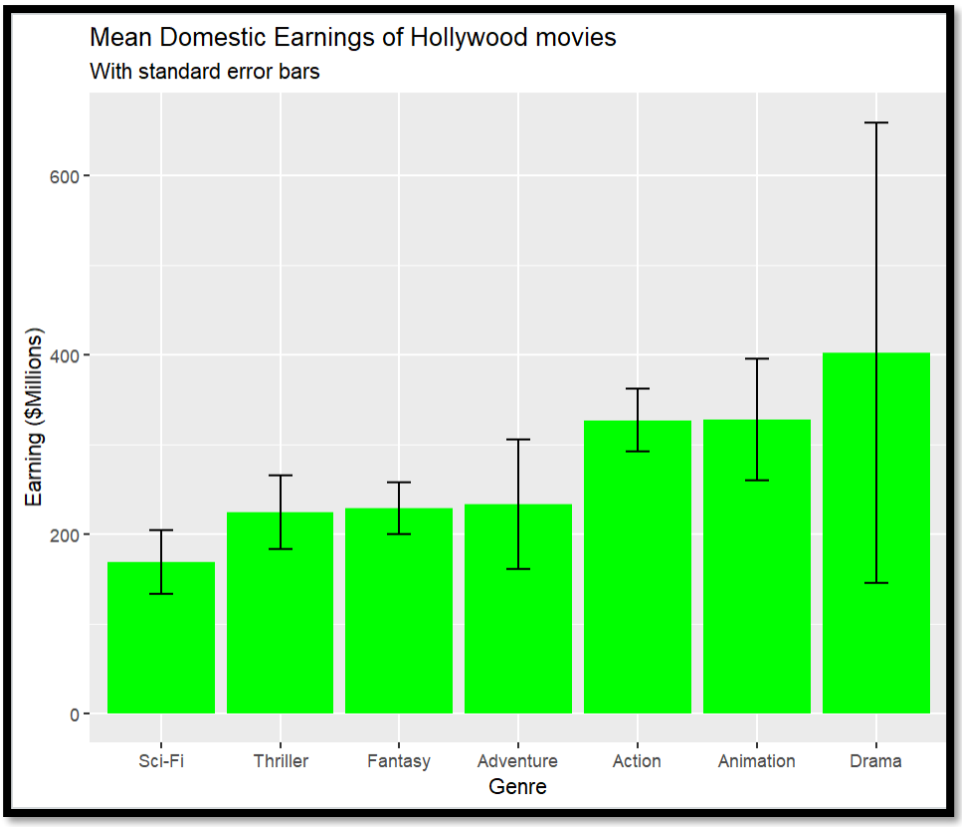


Fig. 24: Representing mean gross figures for different genres of movies along with error rates

Explanation: Here we can see mean earnings with standard errors for each movie genre. Here Drama genre has the highest error rate because our dataset has only a few drama movies and some of them like Titanic (1997) have very high domestic earnings while the others have low earnings, thus they are quite dispersed from the mean.

Data Visualization(Scatterplot):

First we shall compute a scatterplot of movie gross figures vs budgets and try to fit in a linear regression.

Code:

```
library(ggplot2) #Plotting tools belong to this library
ggplot(data = movies, mapping = aes(x = budget_millions, y = gross_millions)) +  #Specifying dataset and its variables to be plotted
geom_point(color = "purple", alpha = .8, size = 3)+ #Specifying color, opacity and size
geom_smooth( method = "lm")+ #Specifying a linear model to be fitted
ggtitle("Plot of gross vs budget ($millions) with best fit line")  + #Title/heading of the plot
xlab("Budget in millions") + #Label of x axis
ylab("Grosss in millions") #Label of y axis
```

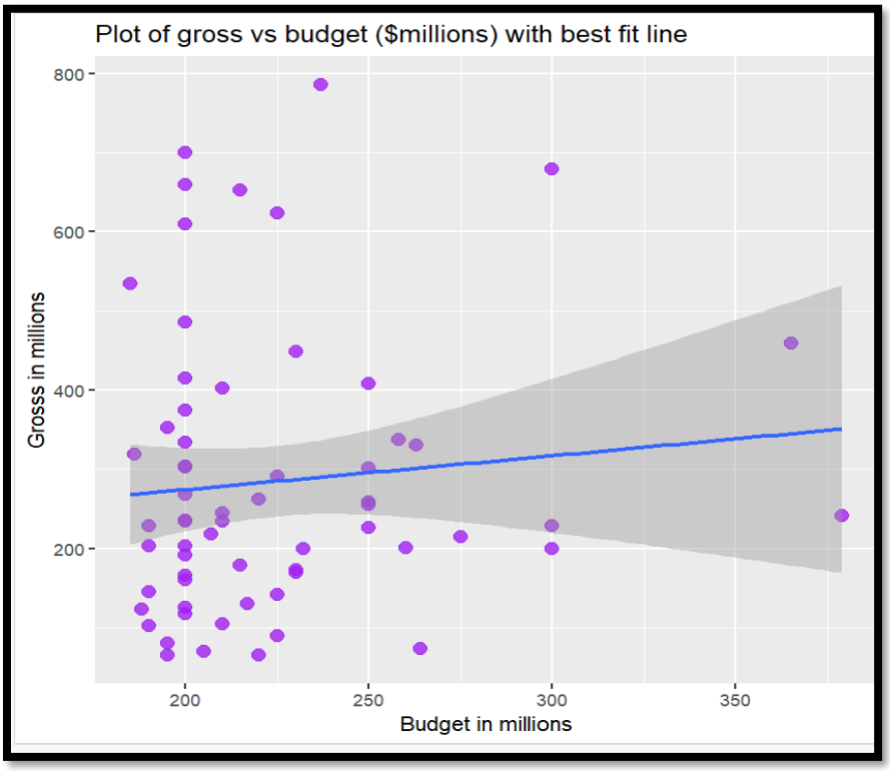


Fig. 25: Gross vs budget scatterplot with best fitted line(linear regression)

Explanation: From the above graph we can state that there is a positive relation in terms of movie budget and its performance at the box office, which means at generally high budget is among the key factors for a film to succeed commercially.

Again, let's see if we can establish a similar relationship for groups. We will divide our movie records into two simple categories of successful or flop based on the difference between budget and gross and then do the plotting.

Code:

```
grader<-function(budget,gross)
 {if(budget>=gross){return("flop")}
   else if(budget<gross){return("success")}}  #Making a function to grade movies
 for(i in 1:length(movies$name)) #Using the function to make a new variable
 {movies$verdict[i]<-grader( movies$budget_millions[i], movies$gross_millions[i])}
```

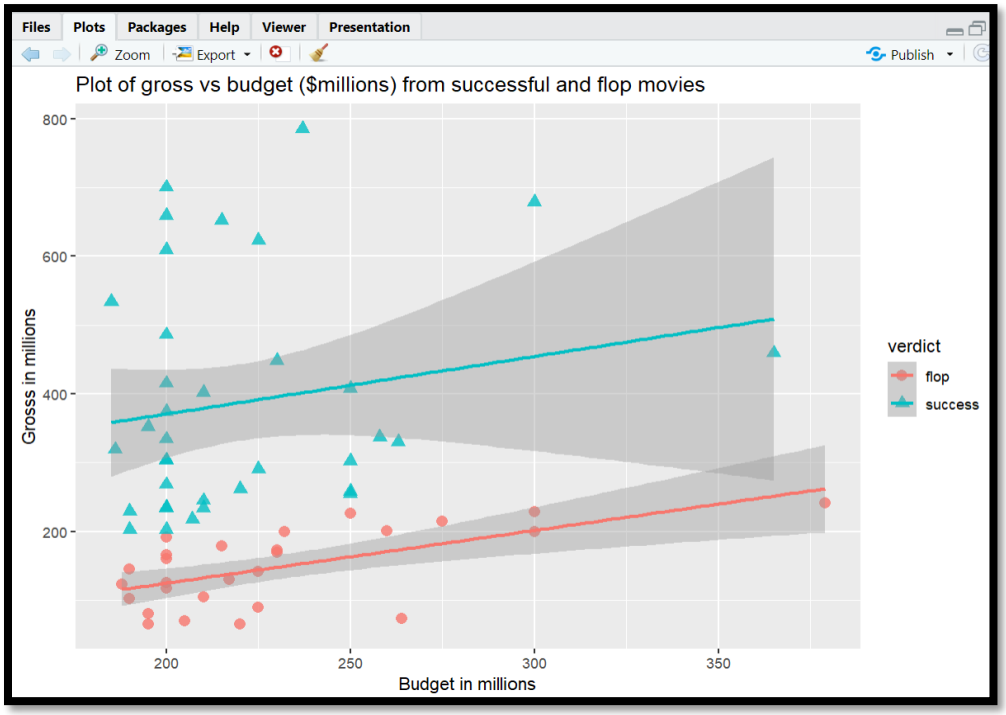


Fig. 26: Gross vs budget scatterplot for successful and flop movies

Explanation: From the above graph we can state that there is a positive relation in terms of movie budget and its performance at the box office regardless of box office results.

Now lets us divide our scatterplot based on movie genres.

Code:

```
library(ggplot2) #Plotting tools belong to this library
 ggplot(data = movies, mapping = aes(x = budget_millions, y = gross_millions)) + #Specifying dataset and its variables to be plotted
   geom_point(color = "red", alpha = .7, size = 2)+ #Specifying color, opacity and size
   geom_smooth( method = "lm")+ #Specifying a linear model to be fitted
   ggtitle("Plot of gross vs budget ($millions) with best fit line")  + #Title/heading of the plot
   xlab("Budget in millions") + #Label of x axis
  ylab("Budget in millions") +  #Label of y axis
   facet_wrap(~genre) #Separating the plots based on genre
```

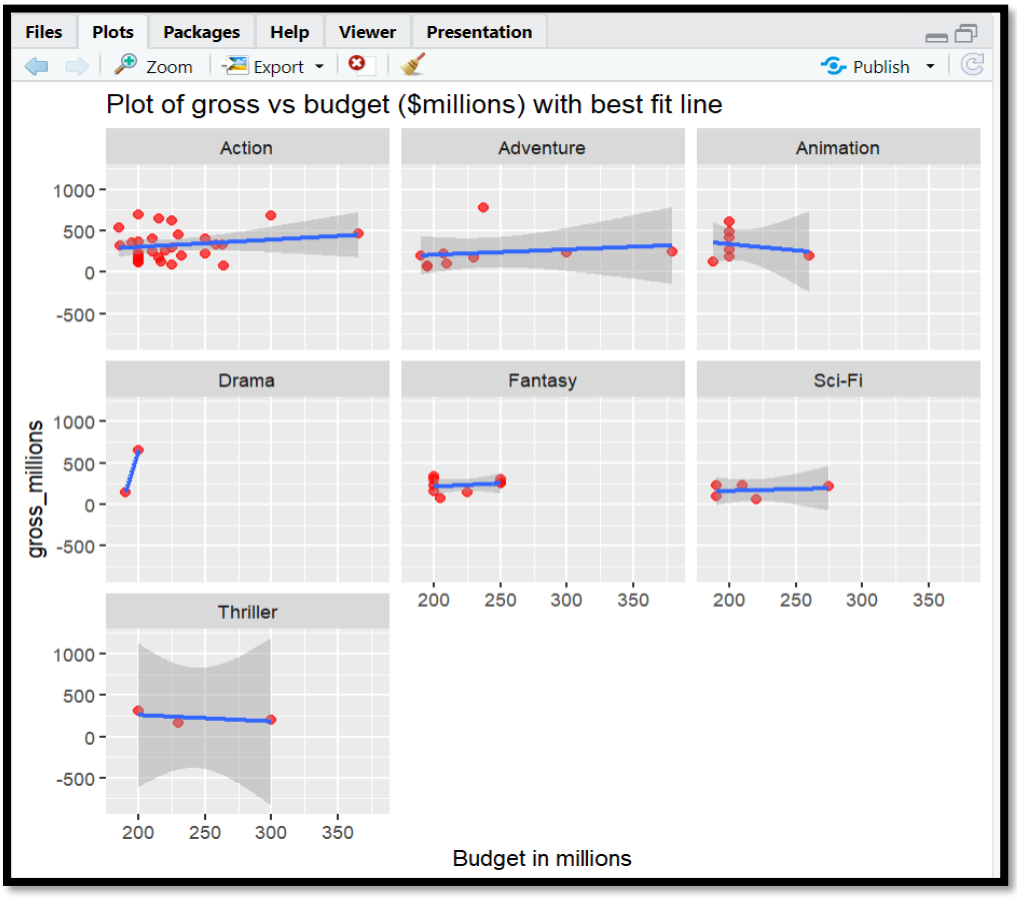


Fig. 27: Gross vs budget scatterplot for each movie genre

Explanation: From the above plot we can see both positive and negative relationships between the variables for different genres. Our project would have yielded better results in this regard if we had a larger dataset encompassing more genres. But since we have based our work on highest budget/grossing movies the genre distribution is not properly spread out because most of the expensive movies belong to specific genres.

Bar charts: Now let's plot bar charts, we shall plot a bar chart showing frequency of movies released per month of the year

Code:

```
library(ggplot2) #Plotting tools belong to this library
movies$release_month<-format(movies$release_date,"%B")  #New variable release_month

format(movies$release_month <- factor(movies$release_month,levels = c("January",
"February", "March", "April", "May", "June", "July", "August", "September",
"October","November","December"))) #Factoring the release_months as to maintain order

ggplot(movies, aes(x=release_month)) + geom_bar(fill="purple") + # setting x axis and bar
#color
labs(title="Simple Bar chart regarding movie release months", #Title
x="Months",  #Label of x axis
y="Movies Released") #Label of y axis
```

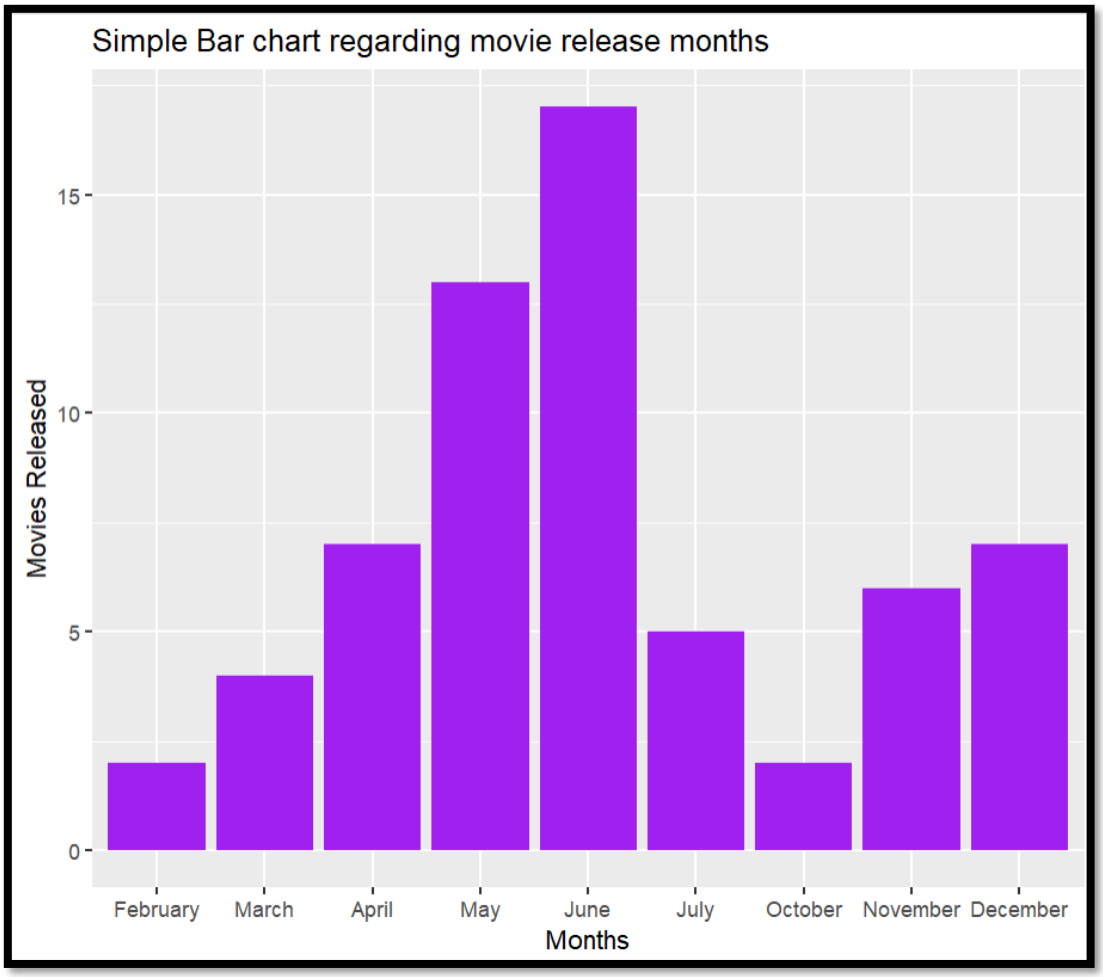


Fig. 28: Bar chart showing frequency of movies released per month

Explanation: From the above plot we can come to the conclusion that most high-budget Hollywood movies are released in June/May i.e. summer vacation time and comparatively much fewer movies are released in October/February/March i.e. the fall season.

Other types of bar charts: Now we shall plot stacked, grouped and filled bar charts.

Code:

```
library(ggplot2) #Plotting tools belong to this library
 ggplot(movies, aes(x=release_month, fill=genre)) + #setting up axes and variables to fill
   geom_bar(position = "stack") + #position=”stack” for stacked bar chart
   labs(title="Stacked Bar chart", #Plot title
        x="Months", #X axis label
        y="Movies Released") #Y axis label

 ggplot(movies, aes(x=release_month, fill=genre)) + #setting up axes and variables to fill
   geom_bar(position = "dodge") + #position=”dodge” will produce grouped bar chart
   labs(title="Grouped Bar chart", #Plot title
        x="Months", #X axis label
        y="Movies Released") #Y axis label

 ggplot(movies, aes(x=release_month, fill=genre)) + #setting up axes and variables to fill
   geom_bar(position = "fill") + #position=”fill” for  percent stacked bar chart
   labs(title="Percent Stacked Bar chart", #Plot title
        x="Months", #X axis label
        y="Movies Released") #Y axis label
```

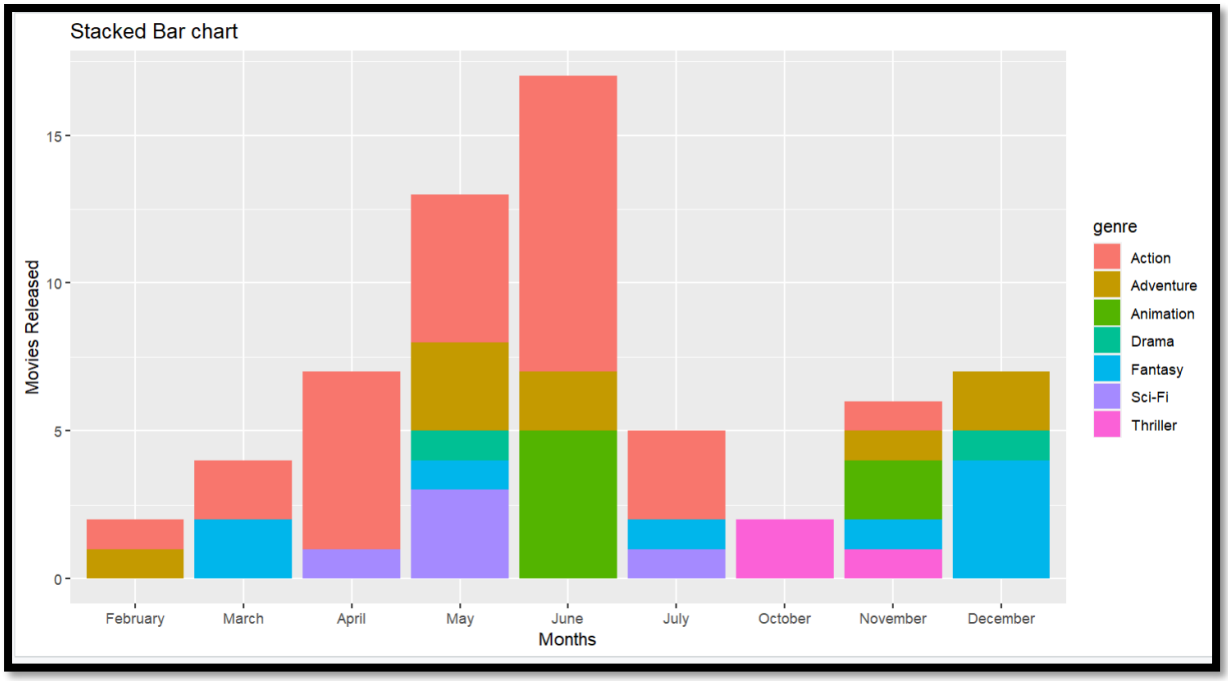


Fig. 29: Stacked Bar chart showing movies released per month and their genres

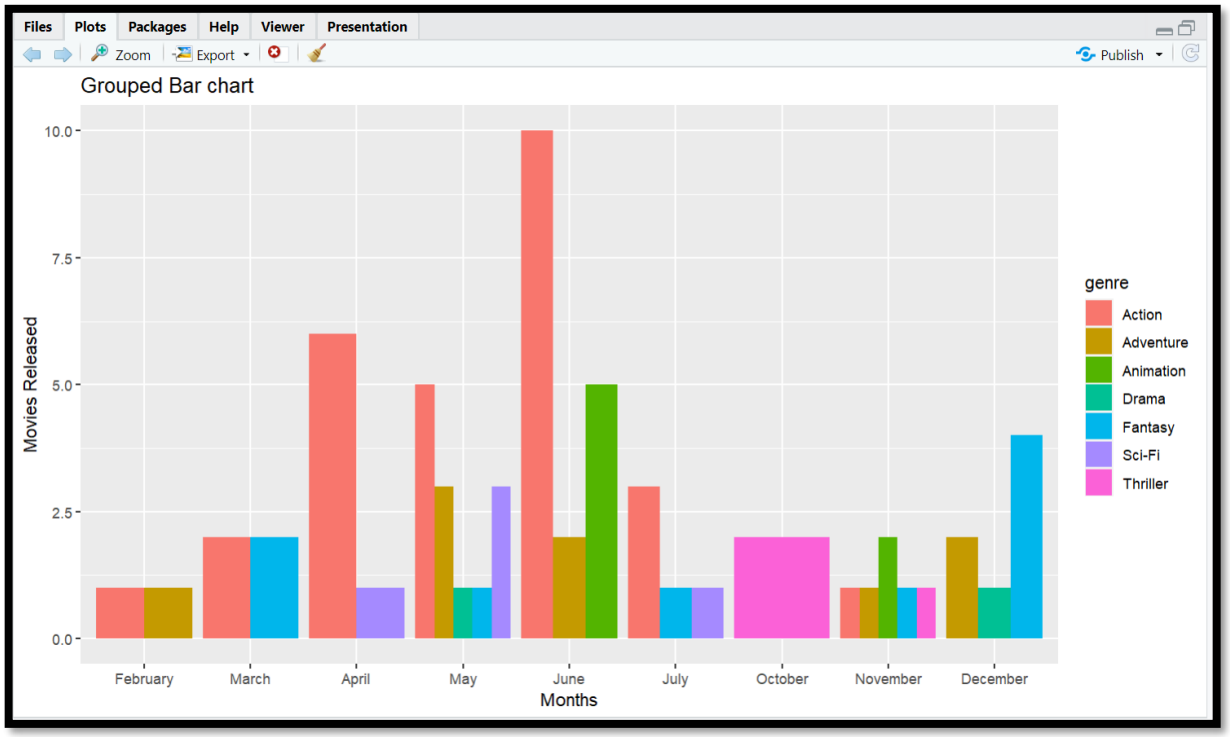


Fig. 30: Grouped Bar chart showing movies released per month and their genres side-by-side

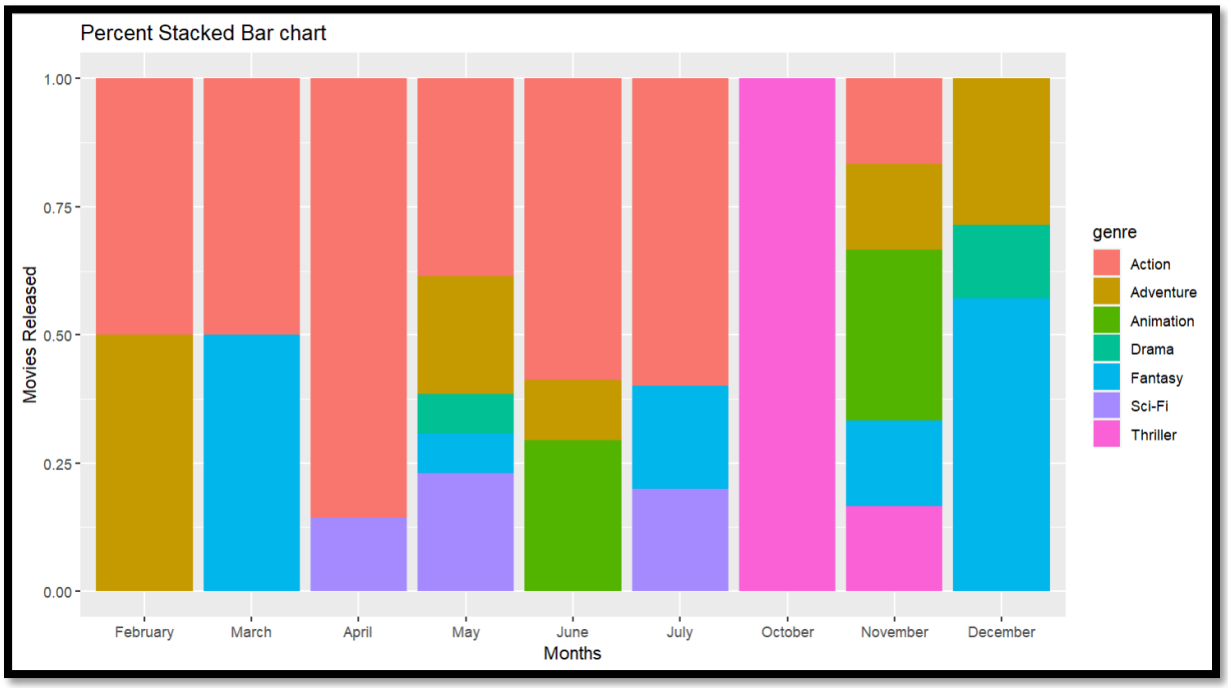


Fig. 31: Percent stacked bar chart showing movies released per month and their genres

Explanation: The stacked and grouped bar charts above show the number of movies released every month from our dataset and their genre. The percent stacked bar chart shows the percent or portion of all the genres released every month. From the above charts we can conclude that action movies occupy most theatres throughout the year, followed by adventure and fantasy genres.

Tree map: Lets create a tree map for our dataset

Code:

```
library(ggplot2) #Plotting tools belong to this library
 library(dplyr)
 library(treemapify) #For tree maps
plotdata <- movies %>% count(release_month, genre) #Plotdata of genres per month
 ggplot(plotdata,aes(fill = release_month, area = n, label = release_month,
subgroup=genre))  +  #Groups will be months and they'll join in groups of genres
geom_treemap() +
geom_treemap_subgroup_border(color="white") + #White borders between genres
geom_treemap_subgroup_text( place = "middle", colour = "black", alpha = 0.9, grow =
TRUE) + #Genre text will be in middle, colored black, with opacity 0.9 and enlarged
geom_treemap_text(colour = "white", place = "centre", grow=FALSE) + #Months in white
theme(legend.position = "bottom")+ #Months in white
labs(title="Tree map showing frequency of movie genres by their release months")
#Title
```

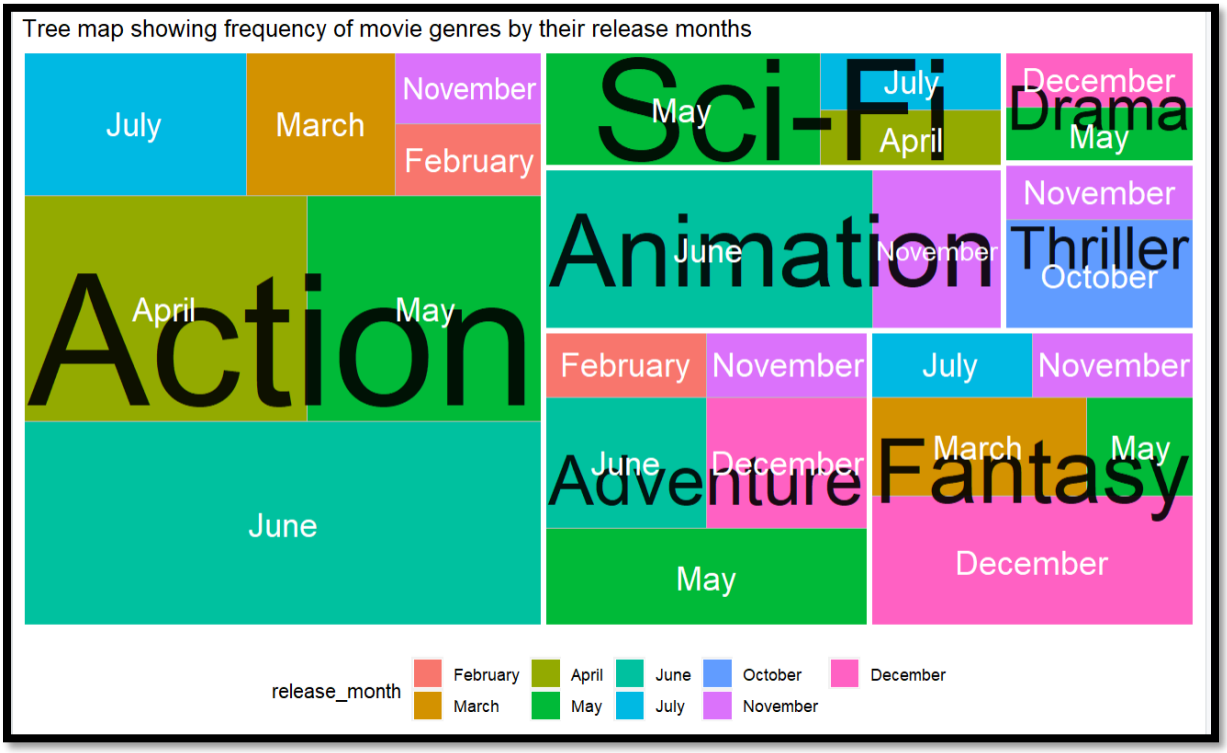


Fig. 32: Tree map showing movies released per month and their genres

Explanation: From the above tree map we can deduce similar summaries as the bar charts, as in which genres of movies are released in which months. From the tree map we can conclude that action movies occupy most theatres through the year, followed by adventure and fantasy genres.

Bar chart with standard error:

Finally let's plot a barchart with standard error bars for the mean ratings of different movie genres.

Code:

```
library(dplyr) #for using the pipe operator
 library(ggplot2) #for using the ggplot function and others

 plotdata <- movies %>% #1  # First we are grouping the genres
  group_by(genre) %>%
  summarize(n=n(),
            mean = mean(rating),
            se = sd(rating)/sqrt(n))



 ggplot(plotdata, aes(x=reorder(genre, mean), y=mean)) +
  geom_bar(stat="identity", fill="orange") +
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=0.2) +
  labs(x="Genre",
       y="Rating (Out of 10)",
       title = "Mean Ratings of Highest Budget Hollywood movie genres",
       subtitle = "With standard error bars")
```

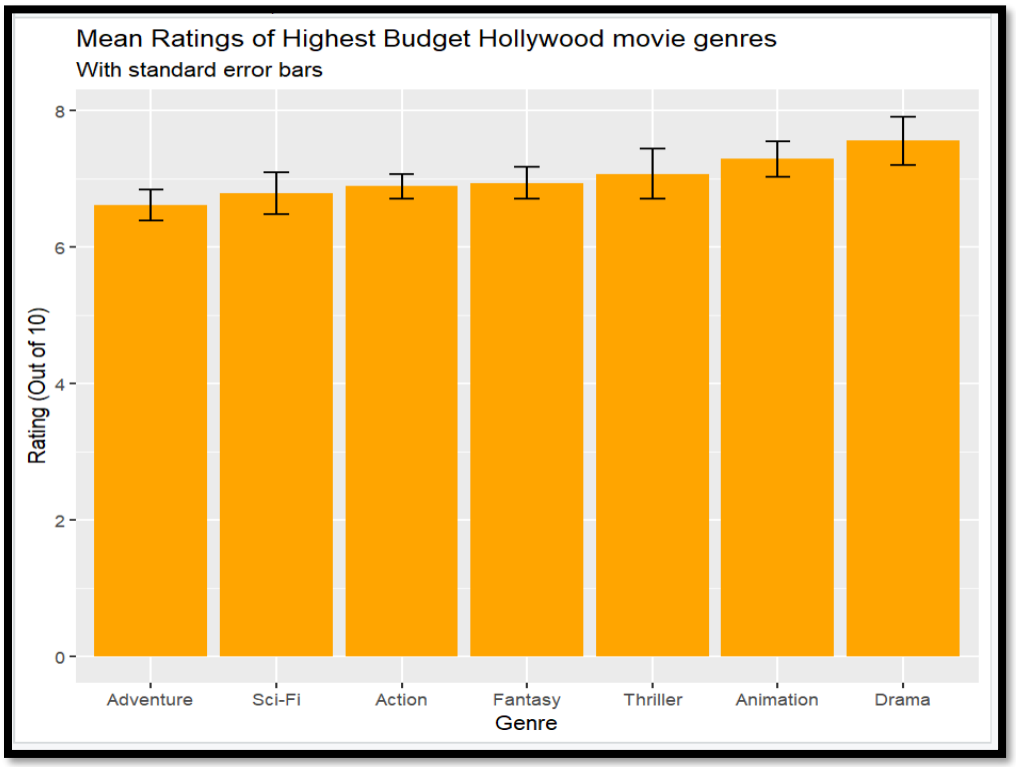


Fig. 32: Bar chart with standard error bars showing mean ratings for genres

Explanation: Here we have plotted the mean values in the Y axis and have divided it through the X axis through genres. From the above plot we can deduce that Animation and Drama have the highest mean ratings and adventures having the lowest. Standard error for the entirety of this plot is pretty low all things considered. Here drama and fantasy genres have the highest standard error which means their values are much more dispersed than the other ones.

Discussion and Conclusion: For this project we have scraped multiple webpages to store our data, then we have performed rigorous preprocessing such as handling missing data, handling noisy data, data transformation, integration and reduction on them to prepare them for analysis. After this we computed the descriptive statistical parameters such as mean, median, mode, variance, standard deviation for our dataset. Finally we visualized our data using various libraries and functions of the R language into scatter plots, bar charts, stacked bar charts, percent stacked bar charts, tree maps etc.Various impediments during project work were resolved using course materials and by consulting the course teacher. Meticulous care has been taken to reach the conclusions of this project and we believe this project has been a success.