FINAL PROJECT REPORT

MOVIE DATA ANALYSIS

Submitted By:

Thasneem Therodam Kandi
Amal Alappat
Lingeshwaran C
Jayalakshmi

Under the guidance of:

Shalini Kumari

Table of Contents

1.1 Purpose

This document describes the details of our final group project "Movie Data Analysis".

1.2 Background

A commercial movie is not only for entertaining the people but also for financial gain. The success of movies are also helps to thrive the industry which eventually helps to the people work in the movie industry and their dependents. A lot of factors such as good directors, actors, stories are considerable for creating good movies. However, these factors often bring an expected box office income, but cannot guarantee good movie rating.

1.3 Data Description

The dataset is from Kaggle website. It contains

Variable name	**Description**

name	Name of the movie
rating	Content rating of the movie like PG, PG13, R etc
genre	Film categorization like ‘Animation’, ‘Comedy’, ‘Romance’, ‘Horror’, ‘Sci-Fi’, ‘Action’, ‘Family’
year	The year in which the movie is released
released	Date and country of release
score	IMDB score of the movie
votes	Number of people who voted for the movie
director	Name of the Director of the Movie
writer	Name of the writer of the movie
star	Name of the main actor/actress of the movie
country	Country where the movie is produced
budget	Budget of the movie in Dollars
gross	Gross earnings of the movie in Dollars
company	The producer of the movie
runtime	Duration of the movie

1.4 Problem Statement

Based on the given information, what kind of movies are more successful than others, in other words, we would like to analyze what are the important factors that make a movie to get a high IMDB score. We also want to visualize the results.

In this project, we take score as response variable and focus on analyzing the rest of the variables in movie data.

We will be doing the data analysis and visualization in 5 various platforms, that are Python, R, SAS, Tableau and Excel. We will also use machine learning algorithm to predict the score.

1.5 Software Requirement

Language/Technology	**Tools/IDE**	**Remarks**
Python	Jupyter Notebook	Python IDE
R	R Studio	R IDE
SAS	SAS Studio	SAS IDE online
Tableau	Tableau	
Excel	Microsoft Excel	
GitHub	GitHub	Code/File/Document Repository

2 DATA ANALYSIS

2.1 Loading Data

First we import all relevant packages which are required to analyse our data.

Python:

```
In [76]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         import warnings
         warnings.filterwarnings("ignore")
         from matplotlib.font_manager import FontProperties
```

R:

```
15  ###' #### Import required libraries
16  library(scales)
17  library(ggplot2)
18  library(gridExtra)
19  library(dplyr)
20  library(rpart)
21  #install.packages("rpart.plot")
22  library(rpart.plot)
23  library(corrplot)
```

Then we read the data

Python:

```
In [77]: data= pd.read_csv("DataFiles/movies.csv")
         data.head()
```

Out[77]:

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	c
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	W B
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	C P
2	Star Wars: Episode V - The Empire Strikes Back	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000.0	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000.0	538375067.0	Lu
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000.0	83453539.0	P P
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000.0	39846344.0	O P

R:

```
###' #### Import data
movieData<-read.csv("/Users/aliayyally/Documents/Data Analytics Training/Python/DataFiles/movies.csv")
head(movieData)
dim(movieData)
str(movieData)
colnames(movieData)
summary(movieData)
```

SAS:

```
1
2  data movies;
3  infile "/home/u59211922/sasuser.v94/movies.csv" dlm="," firstobs=2 dsd;
4  input name$ rating$ genre$ year$ releaseddate$ score votes director$ writer$ star$
5  country$ budget gross company$ runtime;
6  run;
7  proc print data=movies;
8  run;
9
```

2.2 Data Cleaning

Our next step is to do the cleaning of data. We have reviewed the data and cleaned up unnecessary data. We have also removed the empty data, if that are impacting our whole data analysis. In some cases, we filled the empty data using the average values.

Python:

Check size of the data

```
In [78]: data.shape
Out[78]: (7668, 15)
```

The data has 7668 movies and related details. (15 columns)

List down the columns of the dataset to check all columns are relevant with proper names

```
In [79]: data.columns
Out[79]: Index(['name', 'rating', 'genre', 'year', 'released', 'score', 'votes',
                'director', 'writer', 'star', 'country', 'budget', 'gross', 'company',
                'runtime'],
               dtype='object')
```

all column names are perfect and no unncessary columns found

Checking null values

```
In [80]: data.isna().sum()
Out[80]: name          0
         rating       77
         genre         0
         year          0
         released      2
         score         3
         votes         3
         director      0
         writer        3
         star          1
         country       3
         budget     2171
         gross       189
```

Drop rows that has null values

```
In [81]: data.dropna(subset=['score', 'votes','star','country'], inplace=True)
         data
```

Out[81]:

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	ru
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	Warner Bros.	14
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	Columbia Pictures	10
2	Star Wars: Episode V - The Empire	PG	Action	1980	June 20, 1980 (United	8.7	1200000.0	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000.0	538375067.0	Lucasfilm	12

R:

```
###' #### Lets check column wise null values
colSums(is.na(movieData))

###' #### Proportion of missing values
###' #### Checking the missing values
sum(is.na(movieData))
####' ##### There is 2370 null values

mean(is.na(movieData))

####' ##### Since the  the proportion of  null values is very small we can simp
movieData <- na.omit(movieData)
sum(is.na(movieData))
```

SAS:

```
10
11 /* To check is there any missing values present in table*/
12 proc means data=movies nmiss;
13 run;
14
15 /* from the above, score and votes have very few null values, so lets delete those
16 data movies;
17   set movies;
18   if score = . then delete;
19   if votes =  . then delete;
20 run;
21
22
23 /* To check the summary  of the data*/
24 proc summary data=movies print n mean median  mode stddev min max;
25 var score votes budget gross runtime ;
26 run;
```

2.3 Data Exploration

In this step, we explore the data and deep dive and understand and analyze

Python:

Calculate the average movie duration

In [88]:
```
data['runtime'].mean()
```

Out[88]:
```
107.27118422770597
```

Average duration of the movies is 107.27

Sort the movies by duration to find the shortest and longest movies

In [89]:
```
data.sort_values('runtime')
```

Out[89]:

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company
5985	Winnie the	G	Animation	2011	July 15, 2011	7.2	23000.0	Stephen J.	Stephen J.	Jim	United	30000000.0	49871429.0	Walt Disney

Calculate the average score for movies 2 hours or longer, and compare that with the average score for movies shorter than 2 hours

In [95]:
```
print('Avg. score for movies 2 hours or longer: ', data[data['runtime'] >= 120]['score'].mean(),
 '\nAvg. score for movies shorter than 2 hours: ', data[data['runtime'] < 120]['score'].mean())
```

```
Avg. score for movies 2 hours or longer:  7.02940789473684
Avg. score for movies shorter than 2 hours:  6.23248086007497
```

movies with longer duration has higher rarting

Calculate the average duration for each genre

In [96]:
```
data_genrewise  = data[['runtime','genre']].groupby('genre').mean()
data_genrewise
```

Out[96]:

genre	runtime
Action	110.223136
Adventure	107.978923
Animation	92.204142
Biography	119.880361
Comedy	101.491759
Crime	111.752727
Drama	112.964993
Family	99.909091
Fantasy	99.363636
Horror	96.364486
Music	117.000000
Musical	145.000000
Mystery	115.750000
Romance	106.600000
Sci-Fi	100.300000
Sport	94.000000
Thriller	98.625000
Western	97.333333

The Lord of the Rings: The Return of the King	8.9
...	...
Toy Story 3	8.2
Bhaag Milkha Bhaag	8.2
Raging Bull	8.2
The Color of Paradise	8.2
Zindagi Na Milegi Dobara	8.2

100 rows × 1 columns

Calculate the average score for each genre, but only include genres with at least 10 movies

In [100]:
```
genres = data['genre'].value_counts()[data['genre'].value_counts() > 10].index
data[data['genre'].isin(genres)].groupby('genre')['score'].mean()
```

Out[100]:
```
genre
Action       6.202817
Adventure    6.291569
Animation    6.769231
Biography    7.030926
Comedy       6.193987
Crime        6.673455
Drama        6.694257
Family       6.363636
Fantasy      6.006818
Horror       5.750156
Mystery      6.665000
Thriller     5.912500
Name: score, dtype: float64
```

Top 5 countries with highest budget on movies

In [101]:
```
data_countrywise = data[["country","budget","gross","votes"]]
data_countrywise1 = data_countrywise.groupby(['country']).sum().sort_values("budget",ascending=False).head()
data_countrywise1
```

Out[101]:

	budget	gross	votes
country			
United States	1.616655e+11	4.834588e+11	520647110.0
United Kingdom	1.653452e+10	4.895020e+10	68535229.0
Germany	3.344750e+09	6.218591e+09	11025300.0
France	3.304500e+09	8.499480e+09	15613358.0
Canada	2.544677e+09	6.075292e+09	8742104.0

R:

```
best_director<- movieData %>% group_by(director) %>%
  summarise(mean=mean(score))  %>%
  top_n(10) %>%
  arrange(desc(mean))
```

```
## Selecting by mean
```

```
best_director
```

```
## # A tibble: 11 × 2
##    director                    mean
##    <chr>                      <dbl>
##  1 "Roberto Benigni"            8.6
##  2 "Tony Kaye"                  8.5
##  3 "Bob Persichetti"            8.4
##  4 "Nadine Labaki"              8.4
##  5 "Sergio Leone"               8.4
##  6 "Giuseppe Tornatore"         8.3
##  7 "Lee Unkrich"                8.3
##  8 "Majid Majidi"               8.3
##  9 "Juan Jos\xe9 Campanella"    8.2
## 10 "Sriram Raghavan"            8.2
## 11 "Thomas Vinterberg"          8.2
```

Listing the top 10 actors based on the average movie score

```
best_actor<-movieData %>%
  group_by(star)%>%
  summarise(mean= mean(score))%>%
  top_n(10)%>%
  arrange(desc(mean))
```

```
## Selecting by mean
```

```
best_actor
```

Top movie country with average score

```
best_moviecountry<-movieData %>%
  group_by(country)%>%
  summarise(mean= mean(score))%>%
  top_n(10)%>%
  arrange(desc(mean))
```

```
## Selecting by mean
```

```
best_moviecountry
```

SAS:

```
/* To check the correlation between columns */
proc corr data=movies;
run;
/* Result :
   The movie score  mainly correlates with the values of votes and duration of mo
```

The SUMMARY Procedure

Variable	N	Mean	Median	Mode	Std Dev	Minimum	Maximum
score	7529	6.3896401	6.5000000	6.6000000	0.9701586	1.9000000	9.3000000
votes	7529	87909.52	32000.00	11000.00	162449.61	7.0000000	2400000.00
budget	5418	36049422.02	20000000.00	20000000.00	44375340.22	3000.00	797361618
gross	7258	78445888.50	20015467.00	97.0000000	166174716	85.0000000	2847246203
runtime	7401	107.2013241	104.0000000	97.0000000	18.5976460	55.0000000	366.0000000

The CORR Procedure

5 Variables:	score votes budget gross runtime

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
score	7529	6.38964	0.97016	48108	1.90000	9.30000
votes	7529	87910	162450	661870810	7.00000	2400000
budget	5418	36049422	44375340	1.95316E11	3000	797361618
gross	7258	78445889	166174716	5.6936E11	85.00000	2847246203
runtime	7401	107.20132	18.59765	793397	55.00000	366.00000

Pearson Correlation Coefficients
Prob > |r| under H0: Rho=0
Number of Observations

	score	votes	budget	gross	runtime
score	1.00000 7529	0.41030 <.0001 7529	0.08181 <.0001 5418	0.18661 <.0001 7258	0.39809 <.0001 7401
votes	0.41030 <.0001 7529	1.00000 7529	0.41999 <.0001 5418	0.62851 <.0001 7258	0.31029 <.0001 7401
budget	0.08181 <.0001 5418	0.41999 <.0001 5418	1.00000 5418	0.73960 <.0001 5273	0.32027 <.0001 5320
gross	0.18661 <.0001 7258	0.62851 <.0001 7258	0.73960 <.0001 5273	1.00000 7258	0.24550 <.0001 7226
runtime	0.39809 <.0001 7401	0.31029 <.0001 7401	0.32027 <.0001 5320	0.24550 <.0001 7226	1.00000 7401

```
/* To find information of data */
proc contents data=movies ;
run;
```

```
/* To create Table with Diabtes patients only*/
proc sql;
create table MOVIEDATA as
select name, rating, genre, year, releaseddate, score, votes, director, writer, sta
country, budget, gross, company, runtime from movies;
quit;
proc print data=MOVIEDATA;
run;

/* get top 10 directors with highest score who has at lest 5 movies*/
proc sql outobs=10;
   title 'Top 10 directors with highest score';
   select director, avg(score) as score
      from MOVIEDATA
      group by director
      order by 2 desc;

```

Top 10 directors with highest score

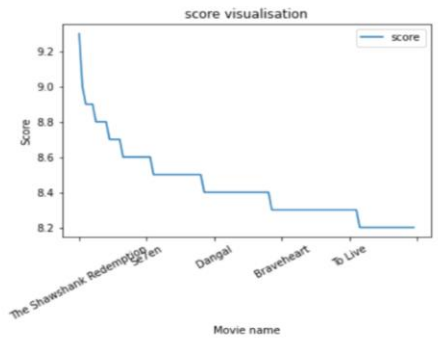
director	score
Anurag K	8.5
Marco Tu	8.5
Bob Pers	8.4
Sergio L	8.4
Aamir Kh	8.4
Nitesh T	8.35
Tengiz A	8.3
Moustaph	8.3
Lee Unkr	8.3
Tom Loga	8.3

2.4 Data Visualization

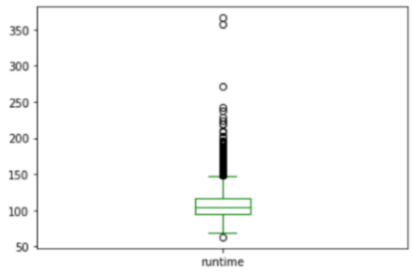
Here, we visualise our assumptions and finding using various graph such as histogram, line graph, box plot, bar graph, regression plots, etc

Python:

```
In [87]: data_scorewise.plot(kind='line', title='score visualisation')
         plt.xlabel('Movie name')
         plt.ylabel('Score')
         plt.xticks(rotation=30)
         plt.show()
```

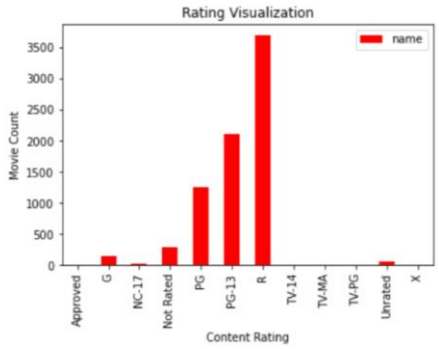


In [91]:
```
data['runtime'].plot(kind='box',color="green")
plt.show()
```

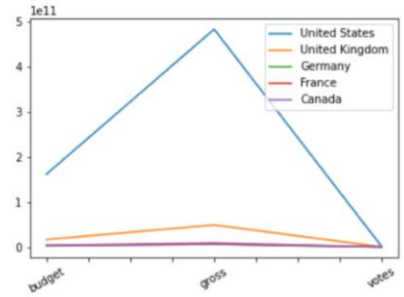


In [93]:
```
data[['rating','name']].groupby('rating').count().plot(kind='bar', title='Rating Visualization',color="Red")
plt.xlabel('Content Rating')
plt.ylabel('Movie Count')
```

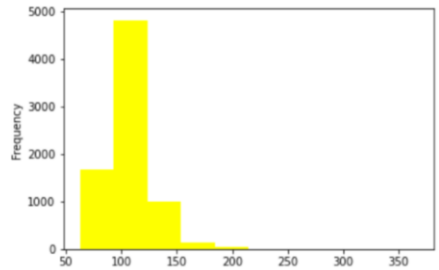
Out[93]: Text(0, 0.5, 'Movie Count')



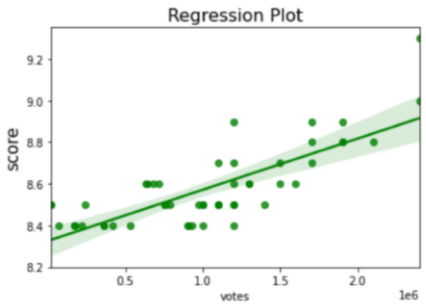
In [102]:
```
data_countrywise1.loc["United States"].plot()
data_countrywise1.loc["United Kingdom"].plot()
data_countrywise1.loc["Germany"].plot()
data_countrywise1.loc["France"].plot()
data_countrywise1.loc["Canada"].plot()
plt.xticks(rotation=30)
plt.legend()
plt.show()
```



In [90]:
```
data['runtime'].plot(kind='hist', bins=10,color="yellow")
plt.show()
```

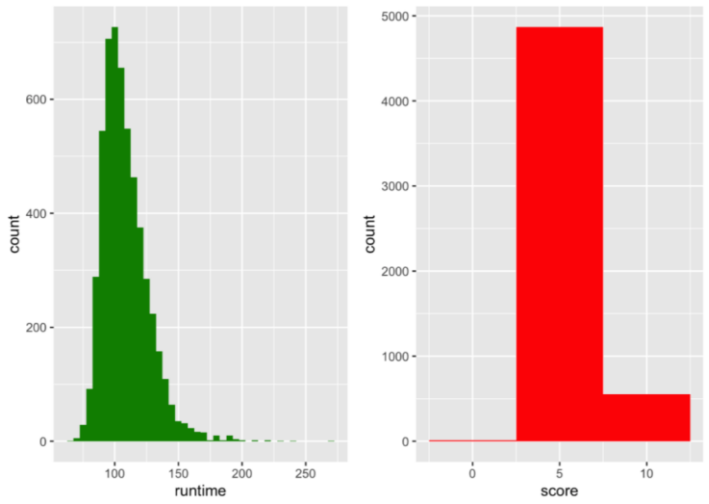


In [116]:
```
sns.regplot(y=data_top['score'],x=data_top['votes'],color="green")
plt.title('Regression Plot',size=16)
plt.ylabel('votes',size=14)
plt.ylabel('score',size=15)
plt.show()
```

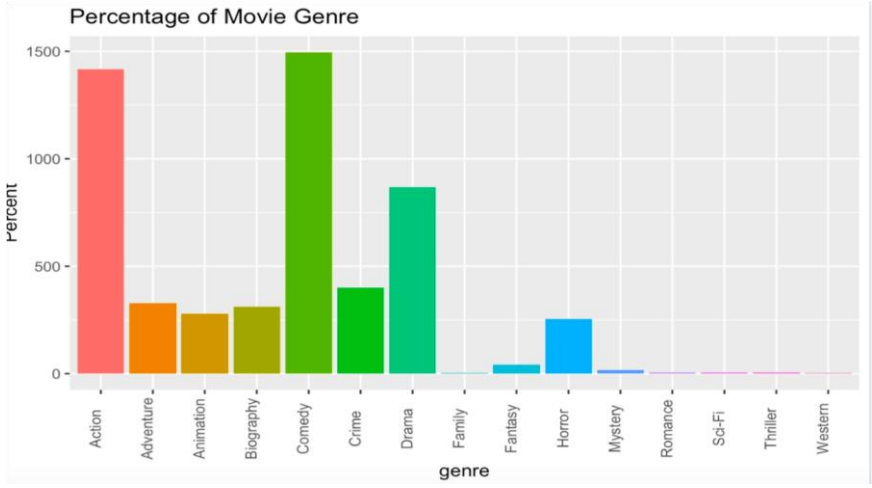


R:

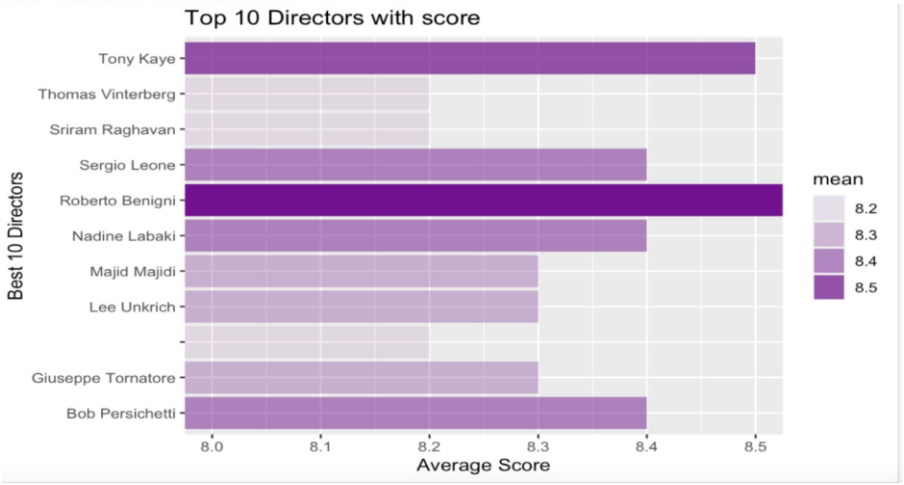
```
options(repr.plot.width=6,repr.plot.height = 4)
g1 <- ggplot(movieData,aes(x=runtime))+geom_histogram(binwidth = 5,fill="green4")
g2 <- ggplot(movieData,aes(x=score))+geom_histogram(binwidth = 5,fill="red")
grid.arrange(g1,g2,nrow=1,ncol=2)
```



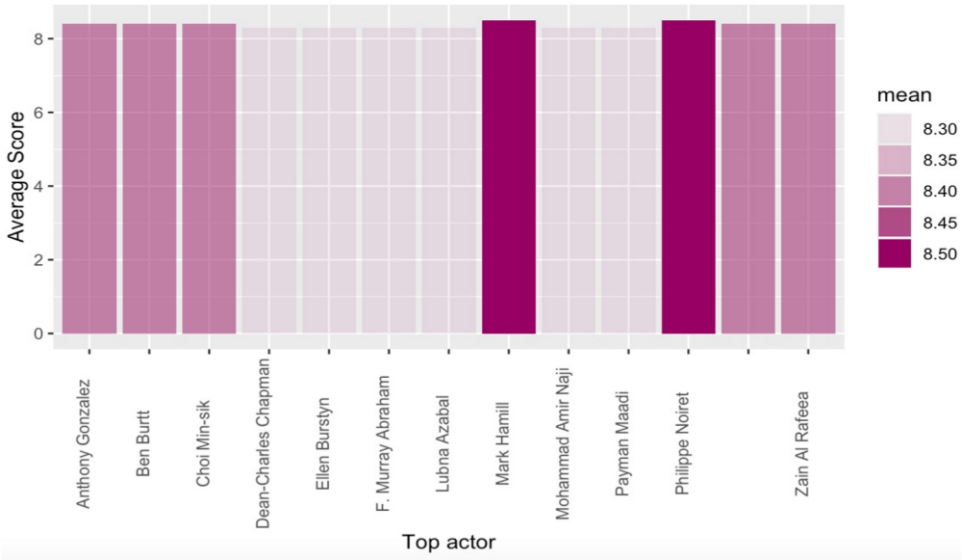

```
###' #### Percentage of Movie Genre
ggplot(movieData,aes(x=genre,fill=genre))+geom_histogram(stat="count",binwidth =
  theme(axis.text.x = element_text(angle = 90,hjust = .5,vjust = 0),legend.posi
  labs(y="Percent",title="Percentage of Movie Genre")
```



```r
###' ####  Top ten directors in terms of Score
best_director<- movieData %>% group_by(director) %>%
  summarise(mean=mean(score))  %>%
  top_n(10) %>%
  arrange(desc(mean))
best_director
```

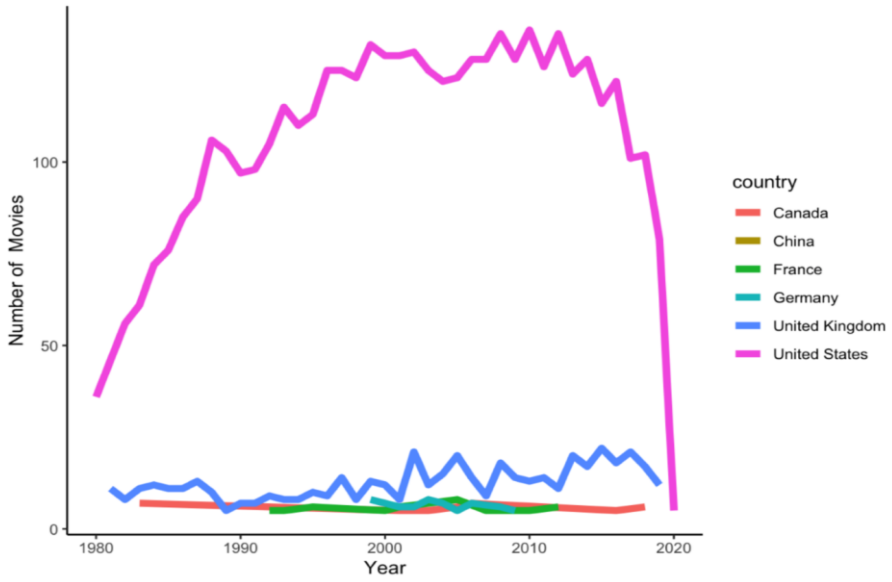



```r
ggtitle("Top 10 actor with average score")+coord_flip(ylim=c(8.0,9.0))
ggplot(best_actor, aes(x = star, y = mean, alpha = mean))+
  theme(axis.text.x = element_text(angle = 90,hjust = .5,vjust = 0))+
  geom_bar(stat = "identity",fill="maroon4") + labs(x = "Top actor", y = "Averag
```



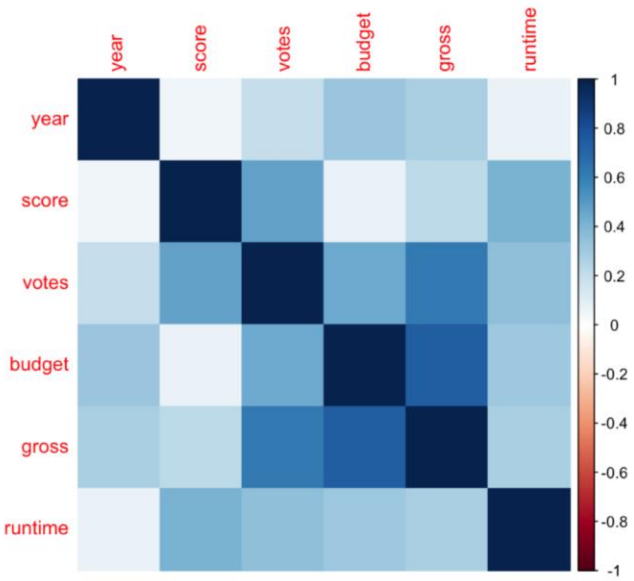
Plot the movie number through time and color them with different country

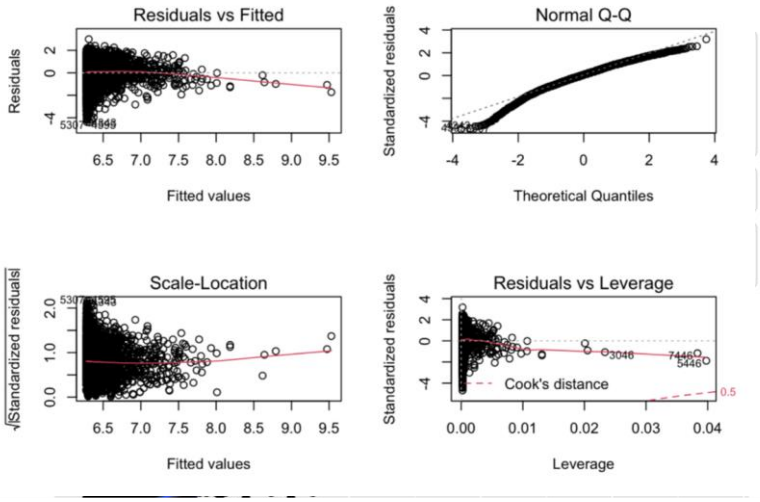
```
options(repr.plot.width=4,repr.plot.height = 4)
ggplot(year_movies,aes(x=year,y=movie_count,colour=country))+
  geom_line(size= 2)+xlab("Year")+ylab("Number of  Movies")+theme_classic()
```



create correlation matrix

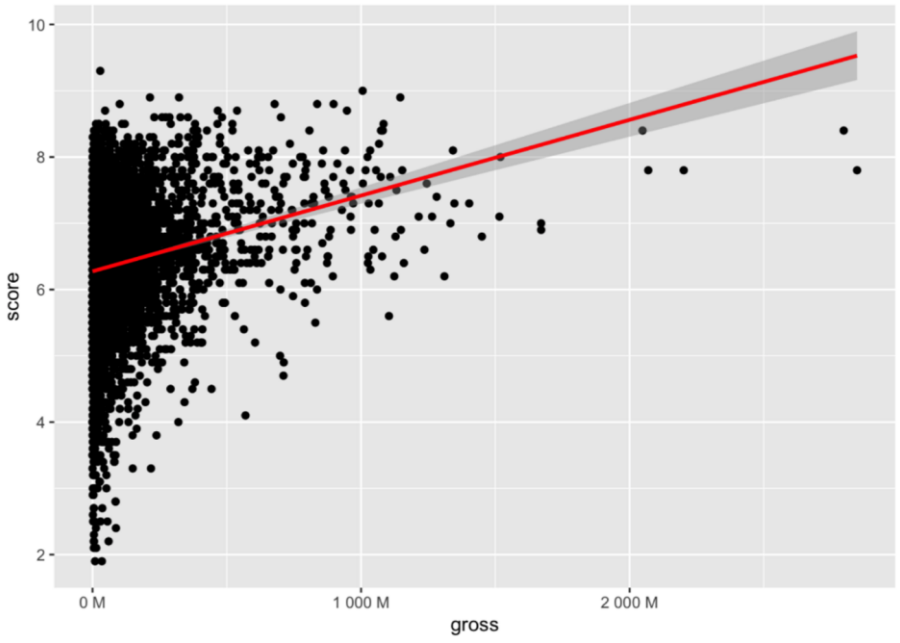
```
Correlation<-cor(movie_numeric)
corrplot(Correlation, method = "color")
```






```
i<-ggplot(movies_important_variables,aes(x=gross,y=score))+geom_point()
i<-i+geom_smooth(method='lm',col="red")
i<- i + scale_x_continuous(labels = unit_format(unit = "M", scale = 1e-6))
print(i)
```

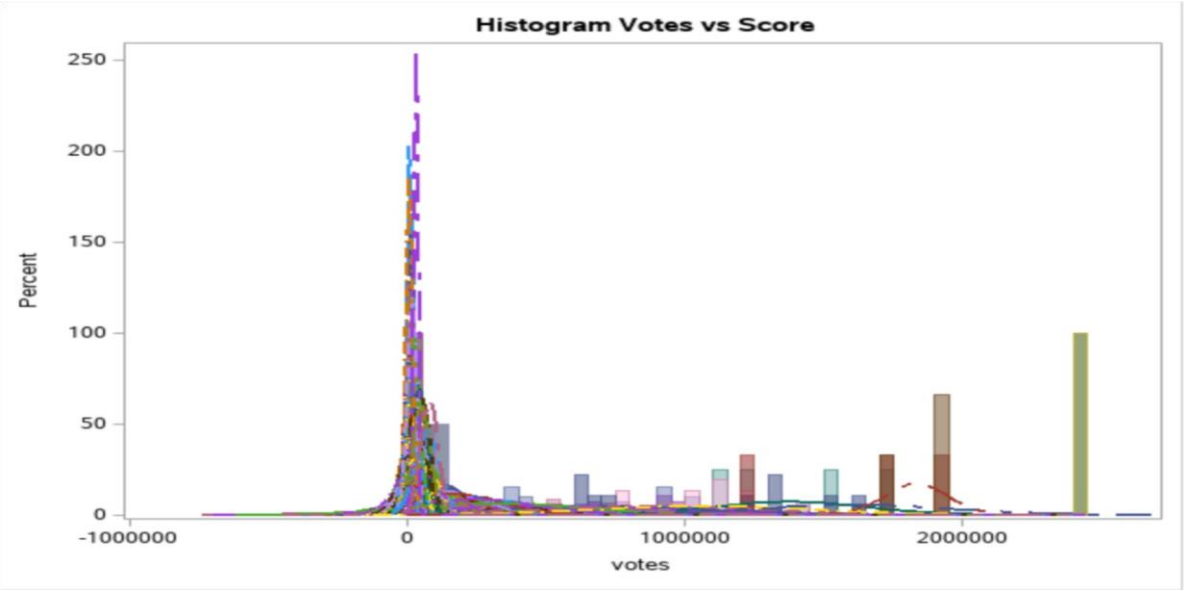
```
## `geom_smooth()` using formula 'y ~ x'
```



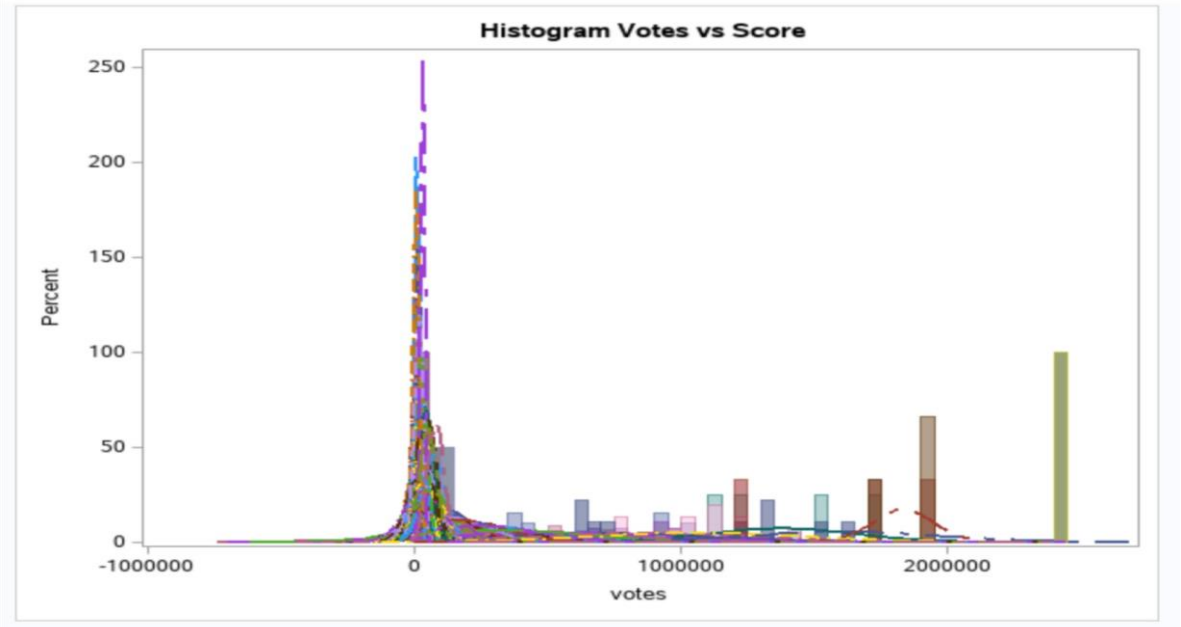

```
  geom_point(size=2,aes(colour=votes))+
  scale_x_continuous(labels = unit_format(unit = "M", scale = 1e-6))+
  labs(title = "Gross Vs Score and votes",
       x = "Gross", y = "Score")
```

SAS:

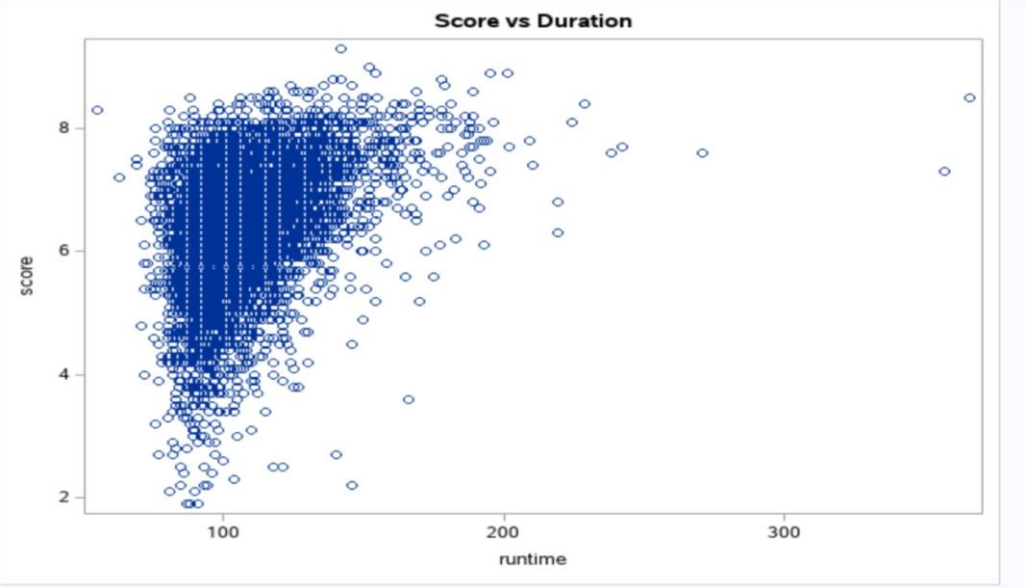
```
/* Histogram */
title"Histogram Duration vs Score";
proc sgplot  data=movies;
histogram runtime/group=score transparency=0.5 fillattrs=(color=olive);
density runtime /type=normal group=score;
keylegend /location=inside position=topright across=1;
run;
```



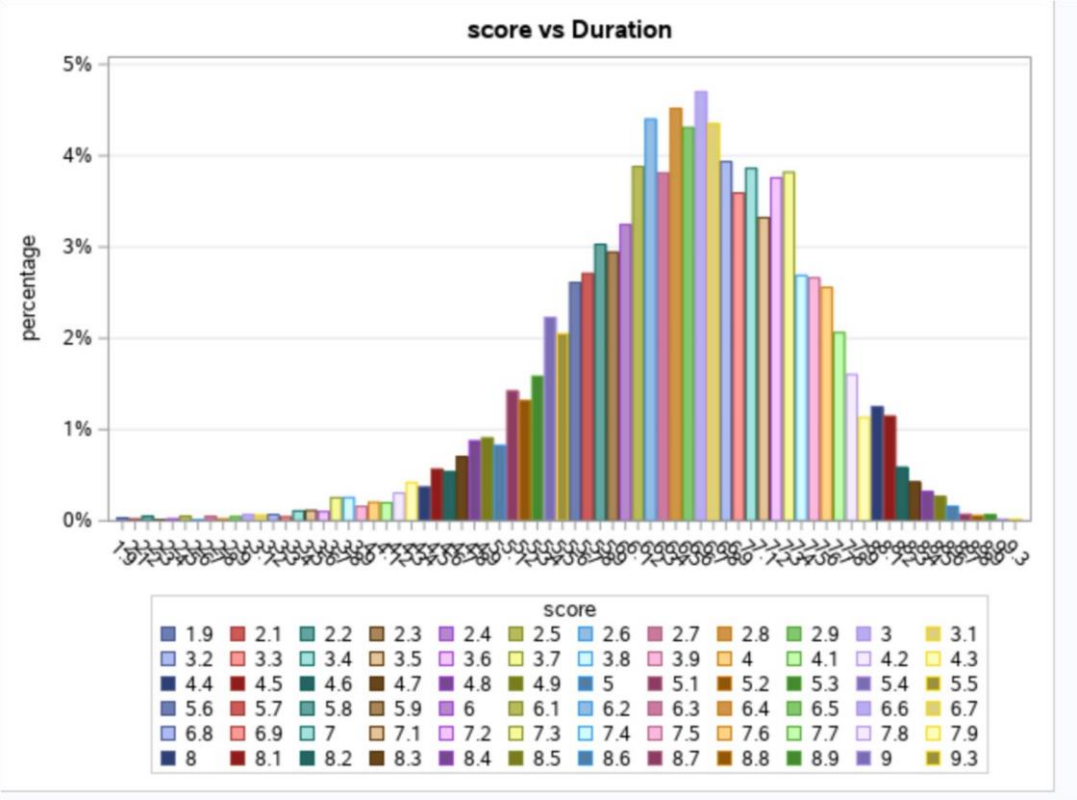

```
/* Histogram */
title"Histogram Votes vs Score";
proc sgplot  data=movies;
histogram votes/group=score transparency=0.5;
density votes /type=normal group=score;
keylegend /location=inside position=topright across=1;
run;
```



```
86 /* SCATTER PLOT OF Score VS Duration */
87 proc sgplot data=movies;
88 scatter x=runtime y= score;
89 title 'Score vs Duration';
90 run;
```




```
100 proc  sgplot data= movies;
101 title "score vs Duration";
102 vbar score/response=runtime group=score stat=percent datalabel;
103 xaxis display=(nolabel);
104 yaxis grid label='percentage';
105 run;
```



2.5 Machine Learning

Since the score data is regressive, we use Supervised Learning. We used two models, which are linear regression and Logistic regression

Creating independent (X) and dependent (Y) variable

To apply a regression model, we first need to seggregate Independent variable (X) and dependent variable (Y)

In [119]:
```
X = data[['rating', 'genre', 'year', 'released',  'votes',
       'director', 'star', 'country', 'gross',
       'runtime']]
X.head()
```

Out[119]:

	rating	genre	year	released	votes	director	star	country	gross	runtime
2443	R	Drama	1994	October 14, 1994 (United States)	2400000.0	Frank Darabont	Tim Robbins	United States	2.881729e+07	142.0
5243	PG-13	Action	2008	July 18, 2008 (United States)	2400000.0	Christopher Nolan	Christian Bale	United States	1.005974e+09	152.0
2247	R	Biography	1993	February 4, 1994 (United States)	1200000.0	Steven Spielberg	Liam Neeson	United States	3.221612e+08	195.0
2444	R	Crime	1994	October 14, 1994 (United States)	1900000.0	Quentin Tarantino	John Travolta	United States	2.139288e+08	154.0
4245	PG-13	Action	2003	December 17, 2003 (United States)	1700000.0	Peter Jackson	Elijah Wood	New Zealand	1.146031e+09	201.0

In [120]:
```
Y=data.iloc[:,5]
Y
```

Out[120]:
```
2443    9.3
5243    9.0
2247    8.9
2444    8.9
4245    8.9
       ...
4342    2.1
4412    2.1
5354    1.9
5306    1.9
4594    1.9
Name: score, Length: 7477, dtype: float64
```

Splitting X and Y to train and test data

Splitting the dataset

In [135]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

Applying Linear regression model

Using Linear Regression Algorithm to train

In [140]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
```

Out[140]:
```
LinearRegression()
```

In [141]:
```
accuracy = regressor.score(X_train,Y_train)
"Accuracy: {}%".format(int(round(accuracy*100)))
```

Out[141]:
```
'Accuracy: 100%'
```

In [142]:
```
Y_pred = regressor.predict(X_test)
```

In [143]:
```
Y_pred
```

Out[143]:
```
array([64.92054872, 43.74782365, 64.83762656, ..., 34.72168365,
       44.21904319, 51.69913051])
```

In [144]:
```
Y_test
```

Out[144]:
```
array([47, 51, 64, ..., 37, 58, 39])
```

Applying Logic regression model

In [151]:
```
#Train the model using train data
from sklearn.linear_model import LogisticRegression
regressor = LogisticRegression(random_state = 0)
regressor.fit(X_train, Y_train)
```

Out[151]:
```
LogisticRegression(random_state=0)
```

In [149]:
```
accuracy = regressor.score(X_train,Y_train)
"Accuracy: {}%".format(int(round(accuracy*100)))
```

Out[149]:
```
'Accuracy: 100%'
```

In [152]:
```
Y_pred = regressor.predict(X_test)
Y_pred
```

Out[152]:
```
array([56, 43, 49, ..., 36, 53, 53])
```

3 REPOSITORY

3.1 GITHUB Reference

Item	Link
CSV Data File	https://github.com/AmalAlappat/edubridge/blob/main/Group%20projects/movies.csv
Python	https://github.com/AmalAlappat/edubridge/blob/main/Group%20projects/movie_data_analysis.ipynb
R	https://github.com/AmalAlappat/edubridge/blob/main/Group%20projects/Movie%20Data%20Analysis-172.pdf
SAS	https://github.com/AmalAlappat/edubridge/blob/main/Group%20projects/Movie_Project.sas
Tableau	https://github.com/AmalAlappat/edubridge/blob/main/Group%20projects/Dashboard%201%20(1).pdf
Excel	https://github.com/AmalAlappat/edubridge/blob/main/Group%20projects/movies.xlsb

4 CONCLUSION

1. Comedy, Action and Drama are mostly produced movie genre
2. United states hold the highest stake in movie production
3. Lebanon top the list of average movie score, surprisingly US not featuring in the top 10 list.
4. **Finally**, duration, number of votes and gross of the movie impact the score of a movie.