Problem Set 3

Noah Estrada-Rand

9/19/2019

1) Data Exploration

A)

The movies dataset has 5032 rows or instances and 28 columns, or variables.

```
dim(movies)
```

```
## [1] 5043   28
```

B)

The names of the variables in the dataset are as follows:

```
colnames(movies)
```

```
##  [1] "color"                     "director_name"
##  [3] "num_critic_for_reviews"    "duration"
##  [5] "director_facebook_likes"   "actor_3_facebook_likes"
##  [7] "actor_2_name"              "actor_1_facebook_likes"
##  [9] "gross"                     "genres"
## [11] "actor_1_name"              "movie_title"
## [13] "num_voted_users"           "cast_total_facebook_likes"
## [15] "actor_3_name"              "facenumber_in_poster"
## [17] "plot_keywords"             "movie_imdb_link"
## [19] "num_user_for_reviews"      "language"
## [21] "country"                   "content_rating"
## [23] "budget"                    "title_year"
## [25] "actor_2_facebook_likes"    "imdb_score"
## [27] "aspect_ratio"              "movie_facebook_likes"
```

C)

There are 492 empty values in the initial movies dataset.

```
sum(is.na(movies$budget))
```

```
## [1] 492
```

After removing the missing values, there are still 28 columns with 4551 rows left.

```
movies <- movies[is.na(movies$budget) == FALSE,]
dim(movies)
```

```
## [1] 4551   28
```

D)

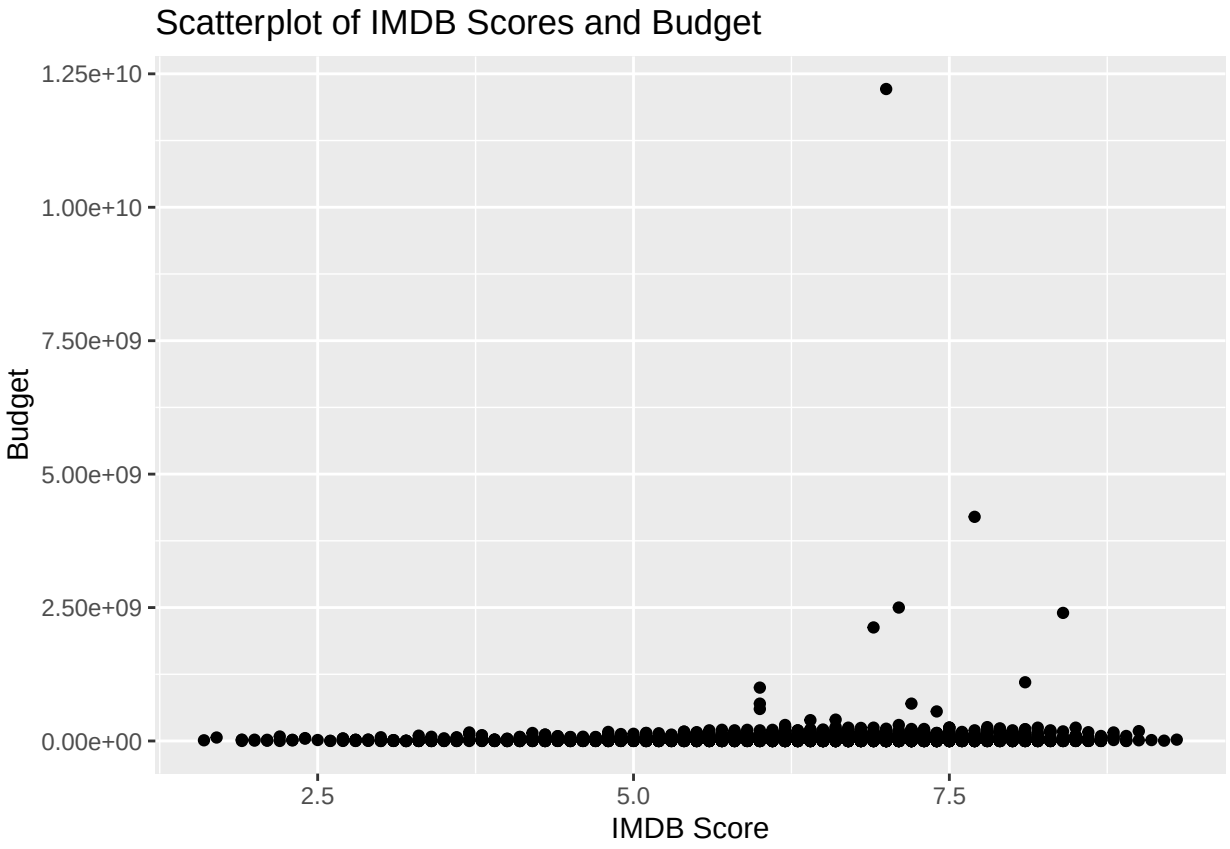
From the code below there are 2175 unique directors.

```
length(unique(movies$director_name))
```

```
## [1] 2175
```

E)

```
ggplot(movies,aes(imdb_score,budget)) + geom_point(aes(imdb_score,budget)) +
  labs(title = "Scatterplot of IMDB Scores and Budget") +
  xlab("IMDB Score") + ylab ("Budget")
```



F)

After removing movies with a budget higher than $400,000,000 the dataset now has only 4539 movies.

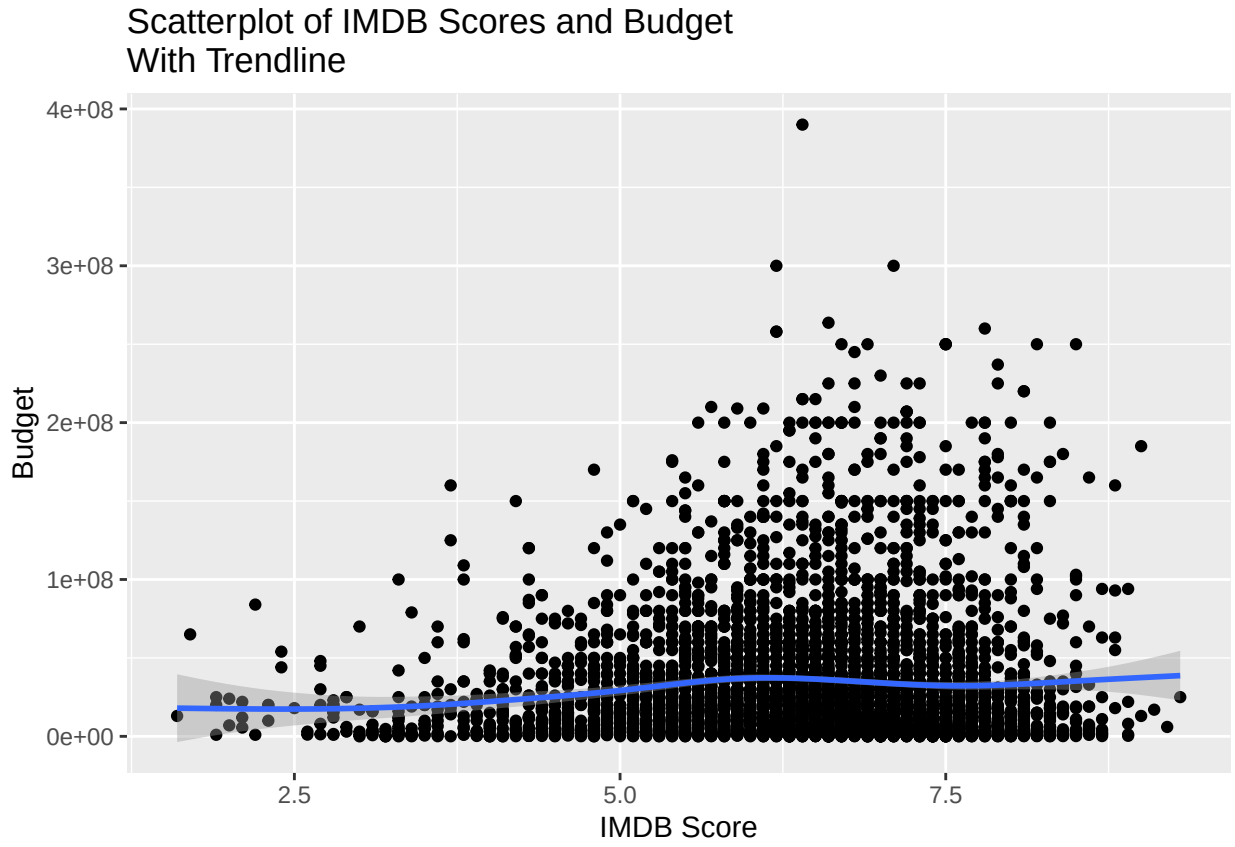
```
movies <- movies[movies$budget<400000000,]
dim(movies)
```

```
## [1] 4539   28
```

G)

```
ggplot(movies,aes(imdb_score,budget)) + geom_point(aes(imdb_score,budget)) +
  labs(title = "Scatterplot of IMDB Scores and Budget\nWith Trendline") +
  xlab("IMDB Score") + ylab ("Budget") + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



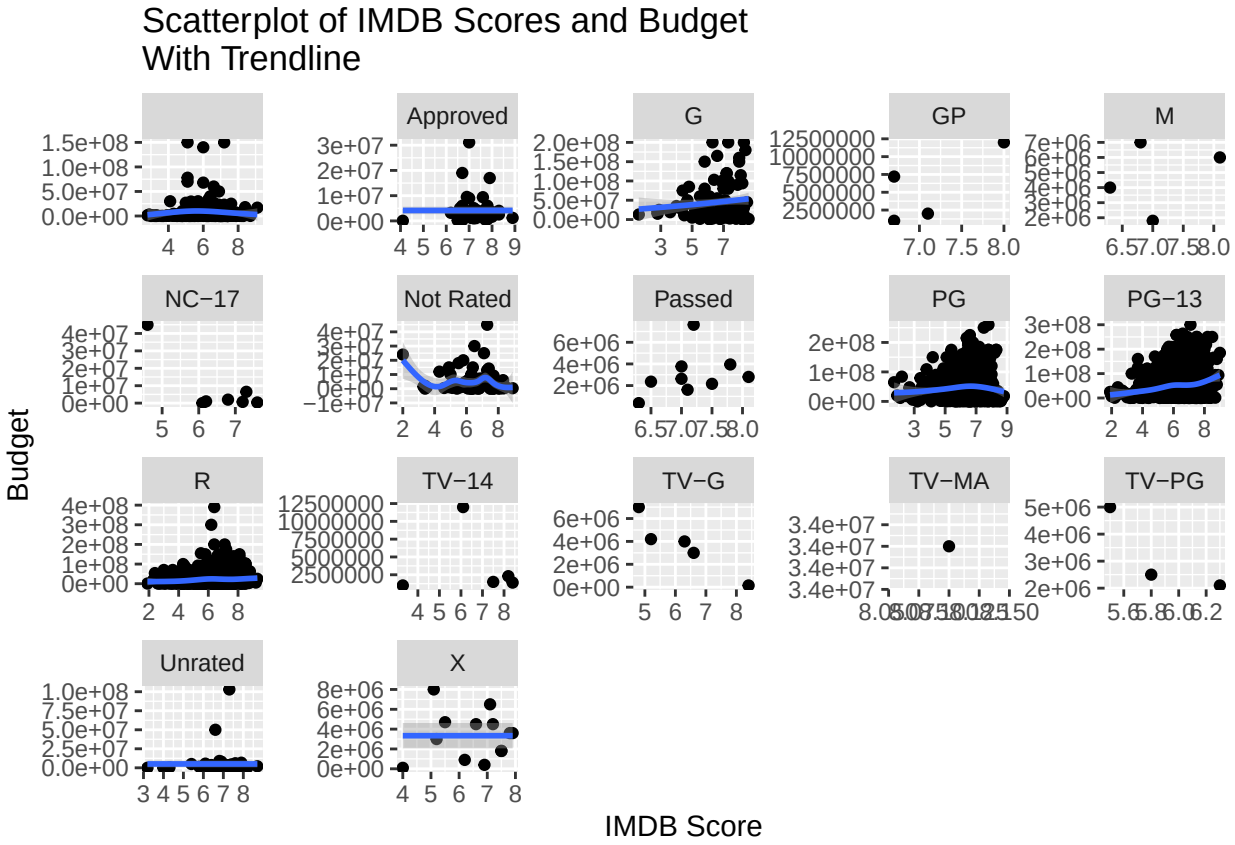
H

```
ggplot(movies,aes(imdb_score,budget)) + geom_point(aes(imdb_score,budget)) +
  labs(title = "Scatterplot of IMDB Scores and Budget\nWith Trendline") +
  xlab("IMDB Score") + ylab ("Budget") + geom_smooth() +
  facet_wrap(~content_rating,scales = "free")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.

## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.

## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.

## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.

## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.

## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```



Based on the above graphs, the strongest relationships between IMDB scores and Budget appear for G, PG, and PG-13 movies.

2) Data Manipulation

A) Creating New Variables

```
movies$grossM <- movies$gross/1e+6
movies$budgetM <- movies$budget/1e+6
movies$genre_main <- do.call('rbind',strsplit(as.character(movies$genres),
                                              '|', fixed=TRUE))[,1] ###doesnt work
```

```
## Warning in rbind(c("Action", "Adventure", "Fantasy", "Sci-Fi"),
## c("Action", : number of columns of result is not a multiple of vector
## length (arg 2)
```

```
movies$profitM <- movies$gross-movies$budget
movies$ROI <- movies$profit/movies$budget
```

B)

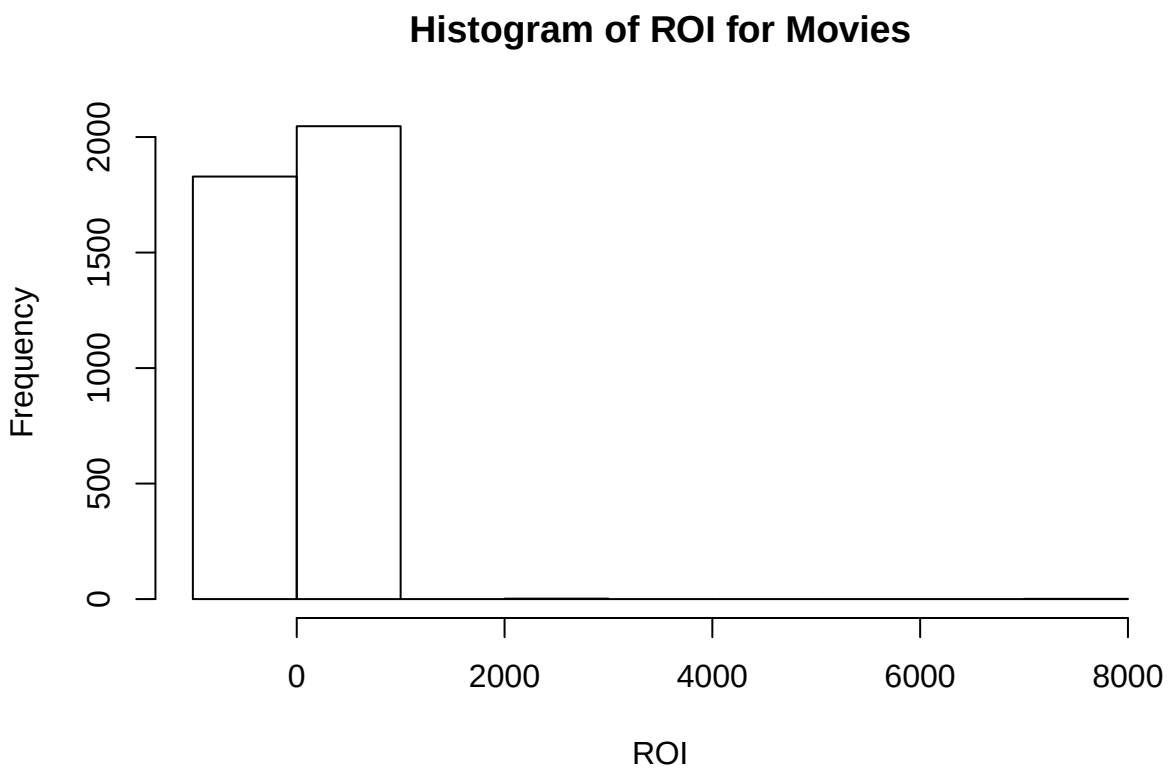
The average ROI for films in the dataset is as follows:

```r
mean(movies$ROI, na.rm = TRUE)
```

```
## [1] 5.273088
```

C)

```r
hist(movies$ROI, breaks = 10,
     main = "Histogram of ROI for Movies",
     ylab = "Frequency",
     xlab = "ROI")
```



D)

```r
length(movies[movies$ROI >10,])
```

```
## [1] 33
```

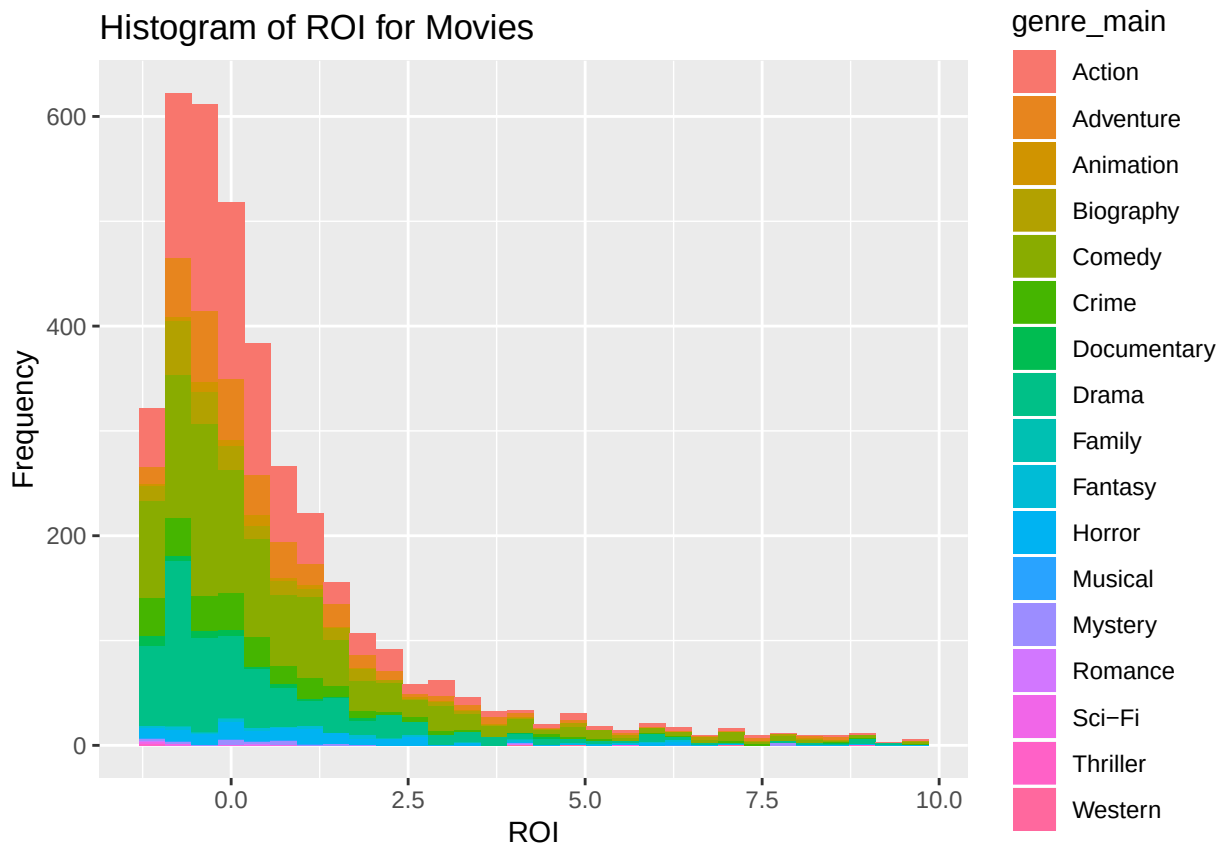
```r
movies <- movies[movies$ROI <10,]
```

E)

```r
ggplot(movies,aes(ROI)) + geom_histogram(aes(fill = genre_main)) +
  labs(title = "Histogram of ROI for Movies") + ylab("Frequency") +
  xlab("ROI")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 660 rows containing non-finite values (stat_bin).
```



F)

Based on the following code, the films with the highest ROI are Musical movies.

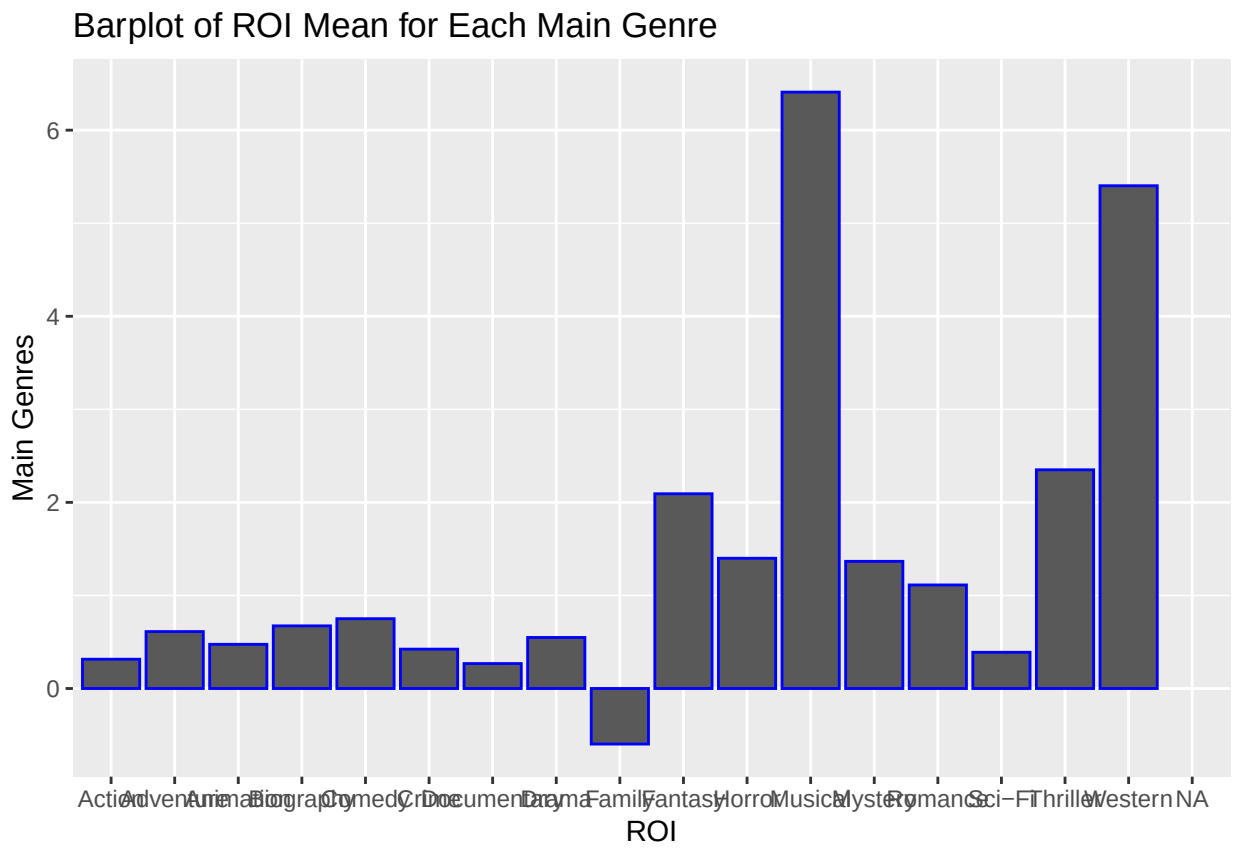
```
summaryBy(ROI~genre_main,data = movies)
```

```
##      genre_main   ROI.mean
## 1        Action  0.3146972
## 2     Adventure  0.6117778
## 3     Animation  0.4749139
## 4     Biography  0.6730581
## 5        Comedy  0.7502510
## 6         Crime  0.4230916
## 7   Documentary  0.2681136
## 8         Drama  0.5484959
## 9        Family -0.5971447
## 10      Fantasy  2.0929081
## 11       Horror  1.3994674
## 12      Musical  6.4089710
## 13      Mystery  1.3665859
## 14      Romance  1.1126902
## 15       Sci-Fi  0.3892234
## 16     Thriller  2.3503454
## 17      Western  5.4029778
## 18         <NA>         NA
```

G

```
summarys_by_genre <- summaryBy(ROI~genre_main,data = movies)
ggplot(summarys_by_genre,aes(genre_main,ROI.mean)) + geom_bar(stat = "identity",col = "blue") +
  labs(title = "Barplot of ROI Mean for Each Main Genre") +
  xlab("ROI") + ylab("Main Genres")
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```



3) Simple Linear Regression

A)

The following script splits the movies data into training and test data. 80% of the data is reserved for training while the remaining 20% is for testing.

```
set.seed(42)
train_index <- sample(1:nrow(movies),.8*nrow(movies),replace = FALSE) ##returns indices
train_data <- movies[train_index,]
test_data <- movies[-train_index,]
```

B)

The new training dataset now has 3515 rows and 33 columns or variables.

```
dim(train_data)
```

```
## [1] 3515   33
```

Next, the new testing dataset now has 879 rows and 33 columns or variables.

```
dim(test_data)
```

```
## [1] 879  33
```

C)

The summary of the regression model built between profitM and IMDB_score is displayed below:

```
mod1 <- lm(profitM~imdb_score,data = train_data)
summary(mod1)
```

```
##
## Call:
## lm(formula = profitM ~ imdb_score, data = train_data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -308524875  -25077928   -9094722   14259930  494107863
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -67126325    5768315  -11.64   <2e-16 ***
## imdb_score   12218267     884038   13.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50610000 on 2981 degrees of freedom
##   (532 observations deleted due to missingness)
## Multiple R-squared:  0.06022,    Adjusted R-squared:  0.0599
## F-statistic:   191 on 1 and 2981 DF,  p-value: < 2.2e-16
```

D)

The parameters of the model are as follows:

```
coef(mod1)
```

```
## (Intercept)  imdb_score
##   -67126324    12218267
```