NAme & Gatech ID: Ahmad Aldabbagh (aaldabbagh3)

Email: aaldabbagh3@gatech.edu (mailto:aaldabbagh3@gatech.edu)

Code ▾

Project 1: Explore and Prepare Data

CSE6242 - Data and Visual Analytics - Spring 2017 Due: Sunday, March 5, 2017 at 11:59 PM UTC-12:00 on T-Square

Note: This project involves getting data ready for analysis and doing some preliminary investigations. Project 2 will involve modeling and predictions, and will be released at a later date. Both projects will have equal weightage towards your grade.

Data

In this project, you will explore a dataset that contains information about movies, including ratings, budget, gross revenue and other attributes. It was prepared by Dr. Guy Lebanon, and here is his description of the dataset:

> The file `movies_merged` (https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6242/project/movies_merged) contains a dataframe with the same name that has 40K rows and 39 columns. Each row represents a movie title and each column represents a descriptor such as `Title`, `Actors`, and `Budget`. I collected the data by querying IMDbâ™s API (see www.omdbapi.com (http://www.omdbapi.com/)) and joining it with a separate dataset of movie budgets and gross earnings (unknown to you). The join key was the movie title. This data is available for personal use, but IMDbâ™s terms of service do not allow it to be used for commercial purposes or for creating a competing repository.

Objective

Your goal is to investigate the relationship between the movie descriptors and the box office success of movies, as represented by the variable `Gross`. This task is extremely important as it can help a studio decide which titles to fund for production, how much to bid on produced movies, when to release a title, how much to invest in marketing and PR, etc. This information is most useful before a title is released, but it is still very valuable after the movie is already released to the public (for example it can affect additional marketing spend or how much a studio should negotiate with on-demand streaming companies for âœsecond windowâ streaming rights).

Instructions

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. Open this file in RStudio to get started.

When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Hide

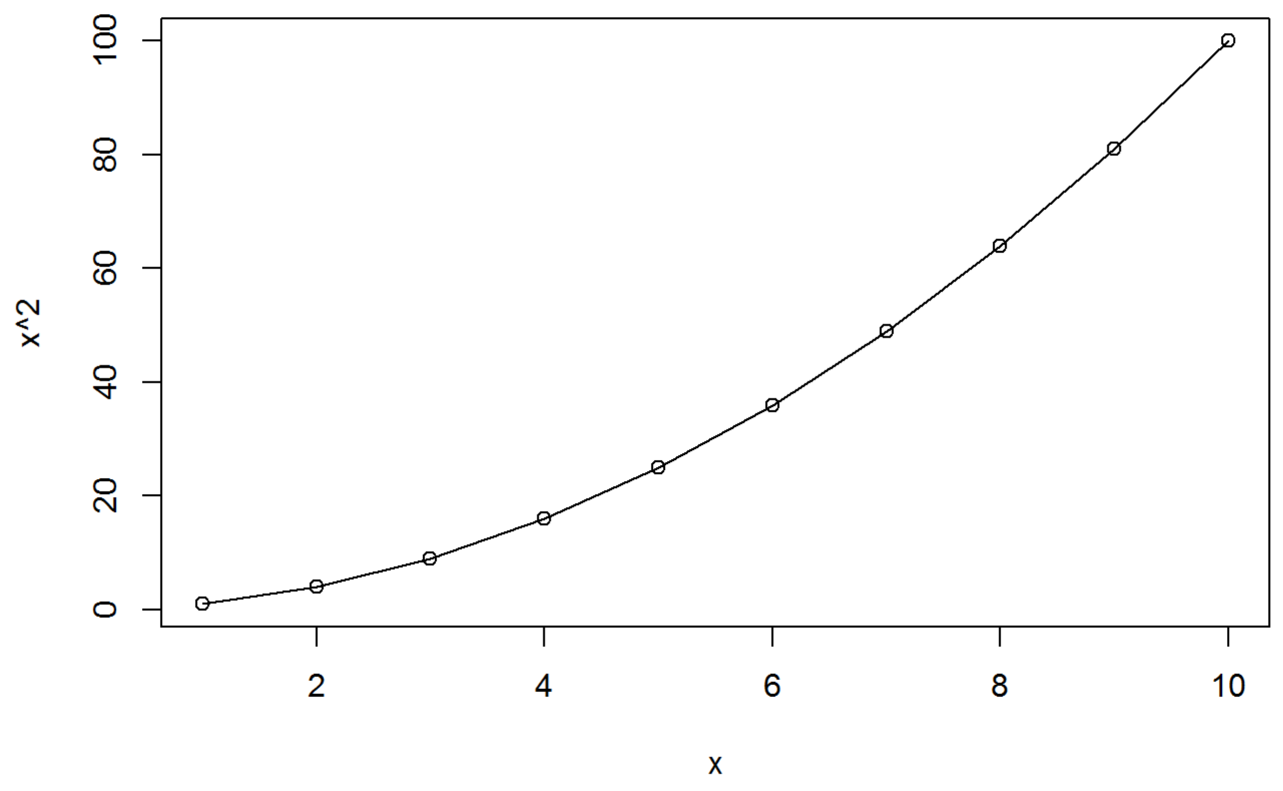
```
x = 1:10
print(x^2)
```

```
 [1]   1   4   9  16  25  36  49  64  81 100
```

Plots appear inline too:

Hide

```
plot(x, x^2, 'o')
```



Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

Please complete the tasks below and submit this R Markdown file (as **pr1.Rmd**) as well as a PDF export of it (as **pr1.pdf**). Both should contain all the code, output, plots and written responses for each task.

Setup

Load data

Make sure you've downloaded the `movies_merged` (https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6242/project/movies_merged) file and it is in the current working directory. Now load it into memory:

Hide

```
setwd("C:/Users/AMD/Desktop/Classes/GaTech/Data Visualization/P1")
load('movies_merged')
```

This creates an object of the same name (`movies_merged`). For convenience, you can copy it to `df` and start using it:

Hide

```
df = movies_merged
cat("Dataset has", dim(df)[1], "rows and", dim(df)[2], "columns", end="\n", file="")
```

```
Dataset has 40789 rows and 39 columns
```

Hide

```
colnames(df)
```

```
 [1] "Title"             "Year"              "Rated"            
 [4] "Released"          "Runtime"           "Genre"            
 [7] "Director"          "Writer"            "Actors"           
[10] "Plot"              "Language"          "Country"          
[13] "Awards"            "Poster"            "Metascore"        
[16] "imdbRating"        "imdbVotes"         "imdbID"           
[19] "Type"              "tomatoMeter"       "tomatoImage"      
[22] "tomatoRating"      "tomatoReviews"     "tomatoFresh"      
[25] "tomatoRotten"      "tomatoConsensus"   "tomatoUserMeter"  
[28] "tomatoUserRating"  "tomatoUserReviews" "tomatoURL"        
[31] "DVD"               "BoxOffice"         "Production"       
[34] "Website"           "Response"          "Budget"           
[37] "Domestic_Gross"    "Gross"             "Date"             
```

```
Title

Year

Rated

Released

Runtime

Genre

Director

Writer

Actors

Plot

Language

Country

Awards

Poster

Metascore

imdbRating

imdbVotes

imdbID

Type

tomatoMeter

tomatoImage

tomatoRating

tomatoReviews

tomatoFresh

tomatoRotten

tomatoConsensus

tomatoUserMeter
```

```
tomatoUserRating

tomatoUserReviews

tomatoURL

DVD

BoxOffice

Production

Website

Response

Budget

Domestic_Gross

Gross

Date
```

Load R packages

Load any R packages that you will need to use. You can come back to this chunk, edit it and re-run to load any additional packages later.

Hide

```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 3.3.2
```

Hide

```
library(GGally)
```

```
Warning: package 'GGally' was built under R version 3.3.2
```

If you are loading any non-standard packages (ones that have not been discussed in class or explicitly allowed for this project), please mention them below. Include any special instructions if they cannot be installed using the regular `install.packages('<pkg name>')` command.

Non-standard packages used: None

Tasks

Each task below is worth **10** points, and is meant to be performed sequentially, i.e. do step 2 after you have processed the data as described in step 1. Total points: **100**

Complete each task by implementing code chunks as described by `TODO` comments, and by responding to questions (“**Q**:”) with written answers (“**A**:”). If you are unable to find a meaningful or strong relationship in any of the cases when requested, explain why not by referring to appropriate plots/statistics.

It is OK to handle missing values below by omission, but please omit as little as possible. It is worthwhile to invest in reusable and clear code as you may need to use it or modify it in project 2.

1. Remove non-movie rows

The variable `Type` captures whether the row is a movie, a TV series, or a game. Remove all rows from `df` that do not correspond to movies.

Hide

```
# TODO: Remove all rows from df that do not correspond to movies
df = subset(df, Type=="movie")
```

Q: How many rows are left after removal? *Enter your response below.*

Hide

```
rows_left = nrow(df)
rows_left
```

```
[1] 40000
```

A:40000

2. Process `Runtime` column

The variable `Runtime` represents the length of the title as a string. Write R code to convert it to a numeric value (in minutes) and replace `df$Runtime` with the new numeric column.

Hide

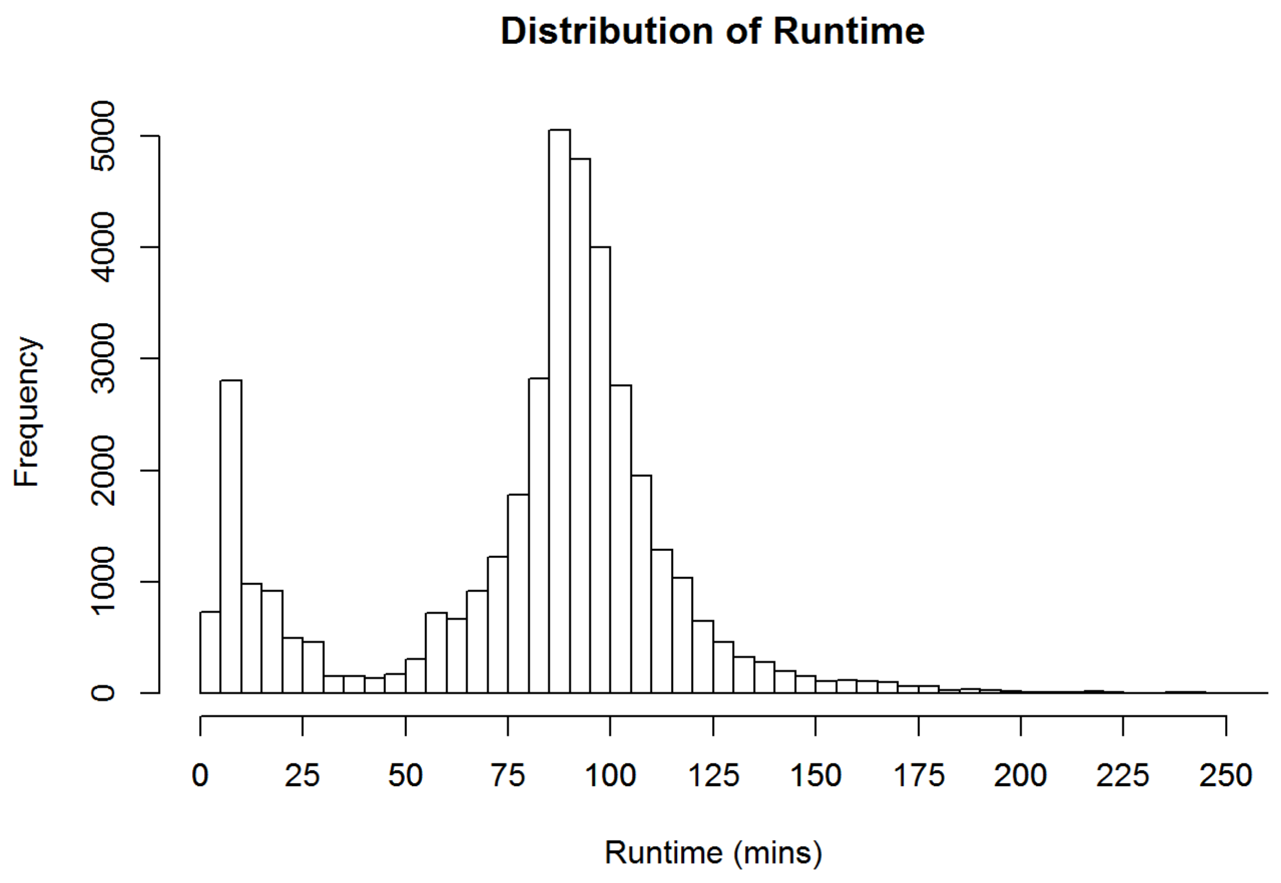
```
runtime_Categories = unique(gsub('[0-9]+', '', df$Runtime))
changeToInt = function(number){
  category = unique(gsub('[0-9]+', '', number))
  if(category==" min"){
    numberList = na.omit(as.numeric(unlist(strsplit(number, "[^0-9]+"))))
    return (numberList[1])
  }
  if(category==" h  min"){
    numberList = na.omit(as.numeric(unlist(strsplit(number, "[^0-9]+"))))
    return (numberList[1]*60+numberList[2])
  }
  if(category==", min"){
    return (1618)
  }
  else{
    return (NaN)
  }
}
runtime = df$Runtime
# TODO: Replace df$Runtime with a numeric column containing the runtime in minutes
temp_runtime = vector(mode="numeric", length=length(runtime))

i = 1
for(tm in runtime){
  temp_runtime[i] = changeToInt(tm)
  i = i+1
}

df$Runtime=temp_runtime

#Maximium Runtime
max(temp_runtime)
```

```
[1] NaN
```

Now investigate the distribution of `Runtime` values and how it changes over years (variable `Year`, which you can bucket into decades) and in relation to the budget (variable `Budget`). Include any plots that illustrate.

Hide

```
# TODO: Investigate the distribution of Runtime values and how it varies by Year and Budget
hist(df$Runtime,breaks=140, xlim=c(0,250),main="Distribution of Runtime", xlab="Runtime (mins)")
axis(side=1, at=seq(0,250, 25) )
```



Hide

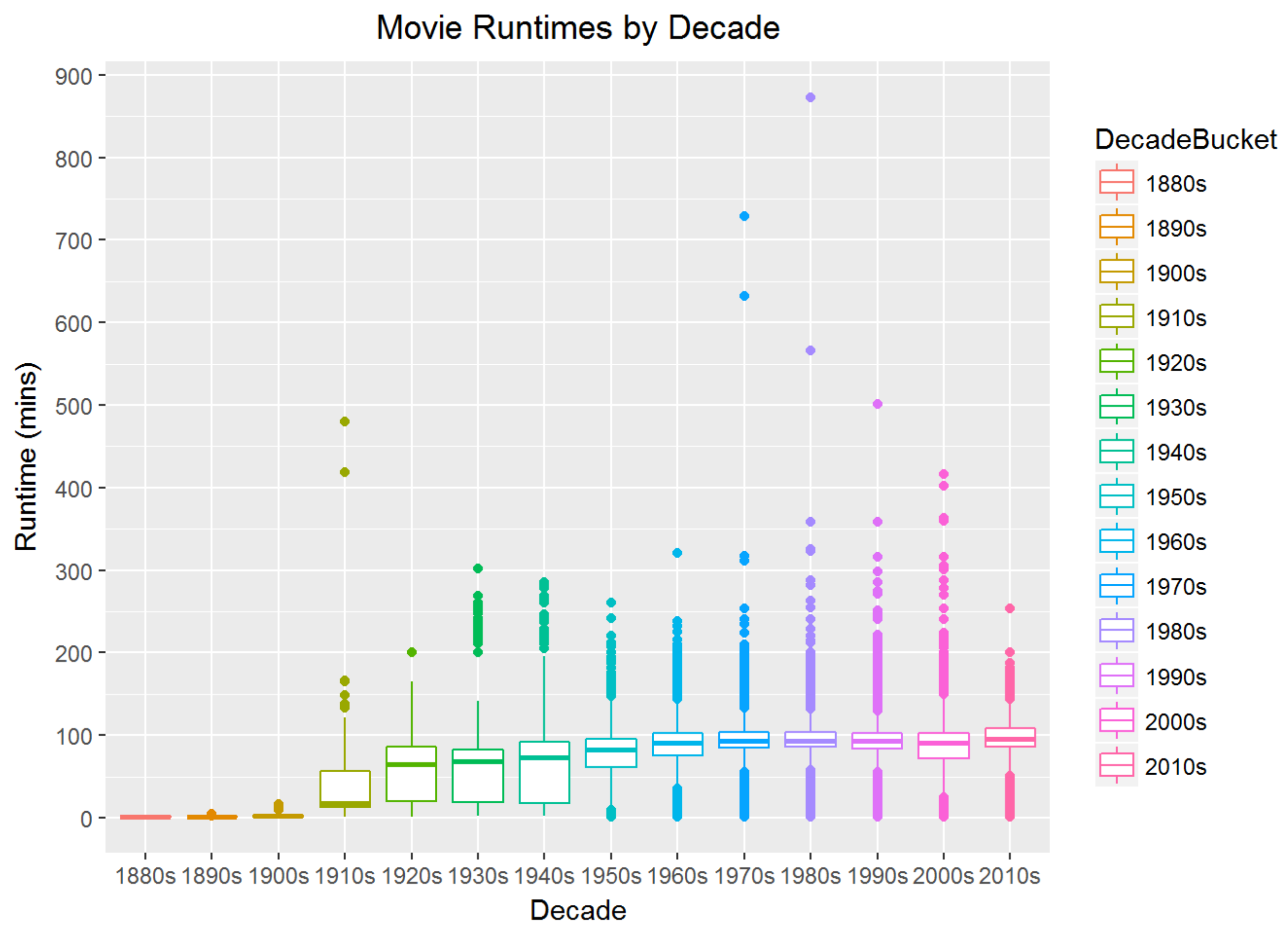
```
#oldest year
min(df$Year)
```

```
[1] 1888
```

Hide

```
decade_bucket = vector(mode="numeric", length=length(df$Year))
decade_bucket[df$Year>=2010] = "2010s"
decade_bucket[df$Year>=2000 & df$Year<2010] = "2000s"
decade_bucket[df$Year>=1990 & df$Year<2000] = "1990s"
decade_bucket[df$Year>=1980 & df$Year<1990] = "1980s"
decade_bucket[df$Year>=1970 & df$Year<1980] = "1970s"
decade_bucket[df$Year>=1960 & df$Year<1970] = "1960s"
decade_bucket[df$Year>=1950 & df$Year<1960] = "1950s"
decade_bucket[df$Year>=1940 & df$Year<1950] = "1940s"
decade_bucket[df$Year>=1930 & df$Year<1940] = "1930s"
decade_bucket[df$Year>=1920 & df$Year<1930] = "1920s"
decade_bucket[df$Year>=1910 & df$Year<1920] = "1910s"
decade_bucket[df$Year>=1900 & df$Year<1910] = "1900s"
decade_bucket[df$Year>=1890 & df$Year<1900] = "1890s"
decade_bucket[df$Year<1890] = "1880s"

df$DecadeBucket = decade_bucket

ggplot(data=df,mapping=aes(x=DecadeBucket,y=Runtime,color=DecadeBucket))+
  geom_boxplot()+
  scale_y_continuous(breaks = round(seq(0, 900, by = 100),1))+
  xlab("Decade")+
  ylab("Runtime (mins)")+
  ggtitle("Movie Runtimes by Decade")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 759 rows containing non-finite values (stat_boxplot).
```

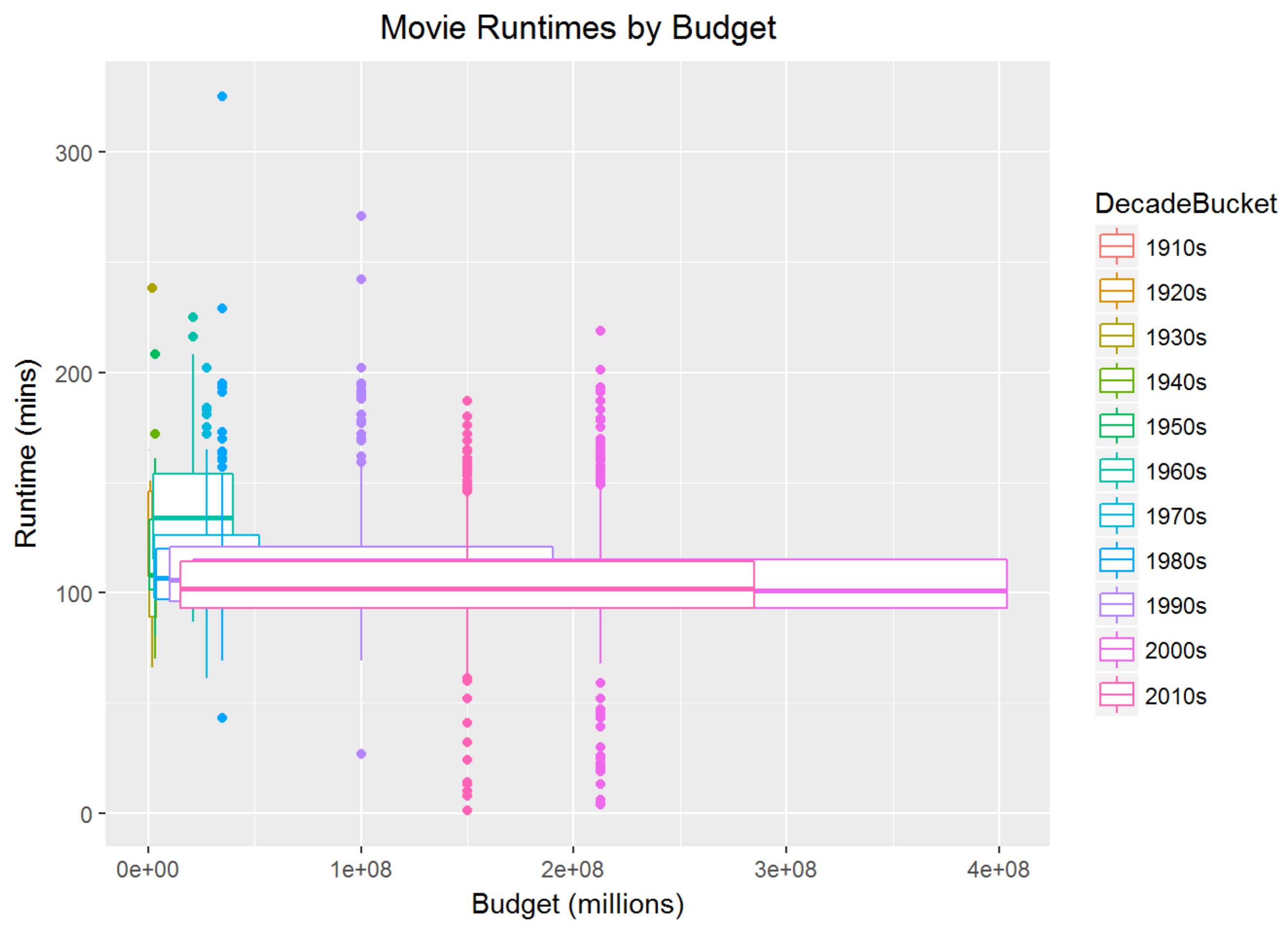


Hide

```
ggplot(data=df,mapping=aes(x=Budget,y=Runtime,color=DecadeBucket))+
  geom_boxplot()+
  xlab("Budget (millions)")+
  ylab("Runtime (mins)")+
  ggtitle("Movie Runtimes by Budget")+
  theme(plot.title = element_text(hjust = 0.5))
```

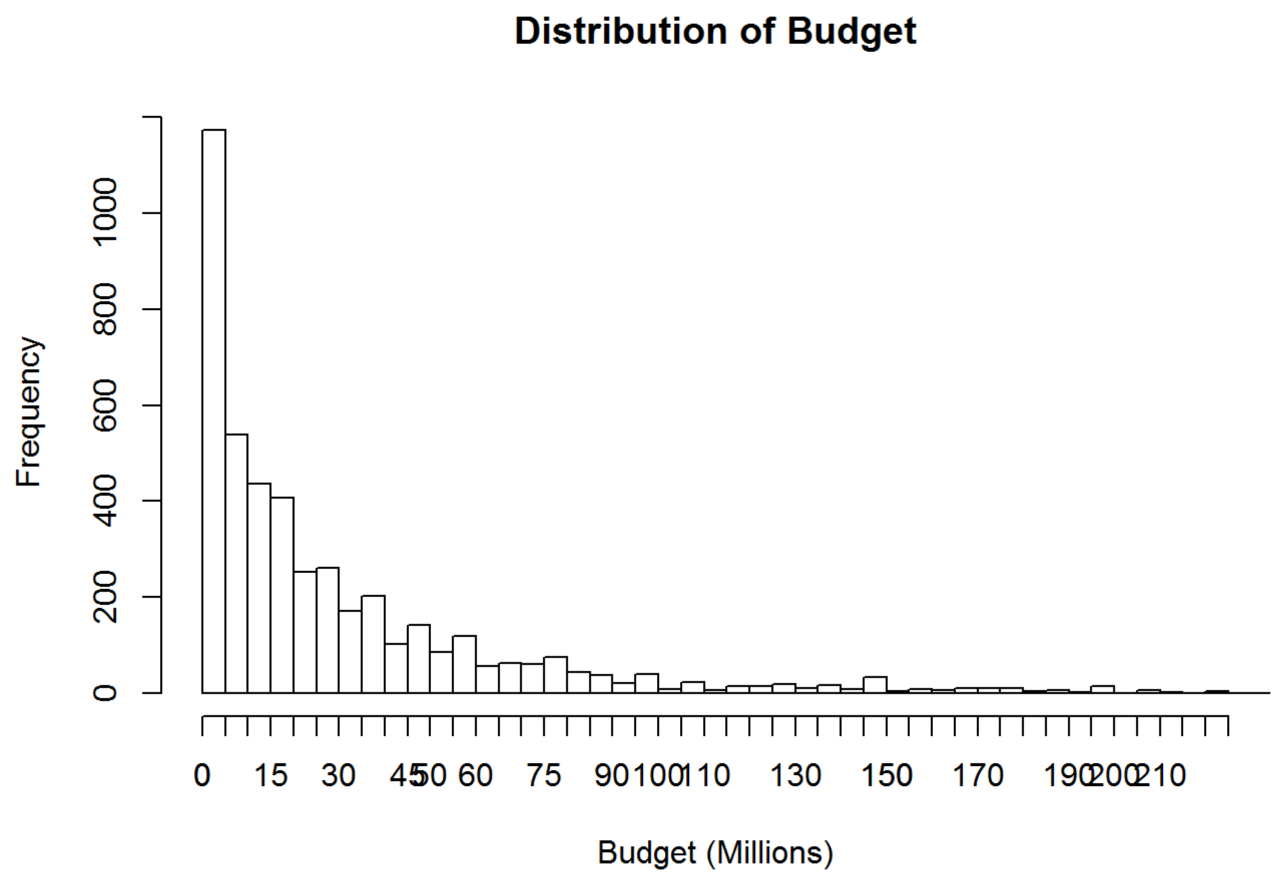
```
Warning: Removed 35480 rows containing non-finite values (stat_boxplot).
```

```
Warning: position_dodge requires non-overlapping x intervals
```



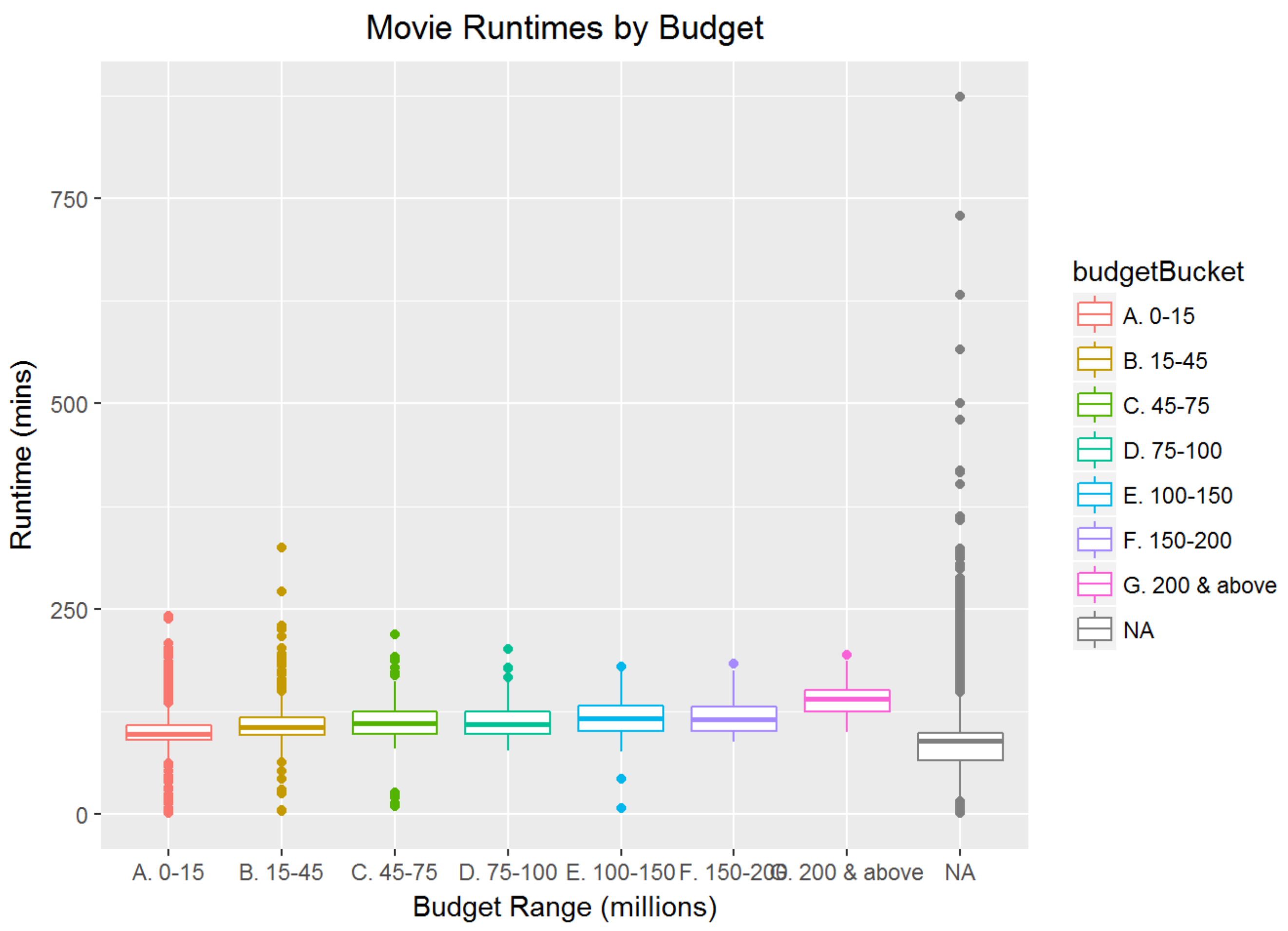
Hide

```
#This is done to choose the buckets for budget
hist(df$Budget/1000000,breaks=65,xlim=c(0,225),main="Distribution of Budget", xlab="Budget (Mill
ions)")
axis(side=1, at=seq(0,225, 5))
```



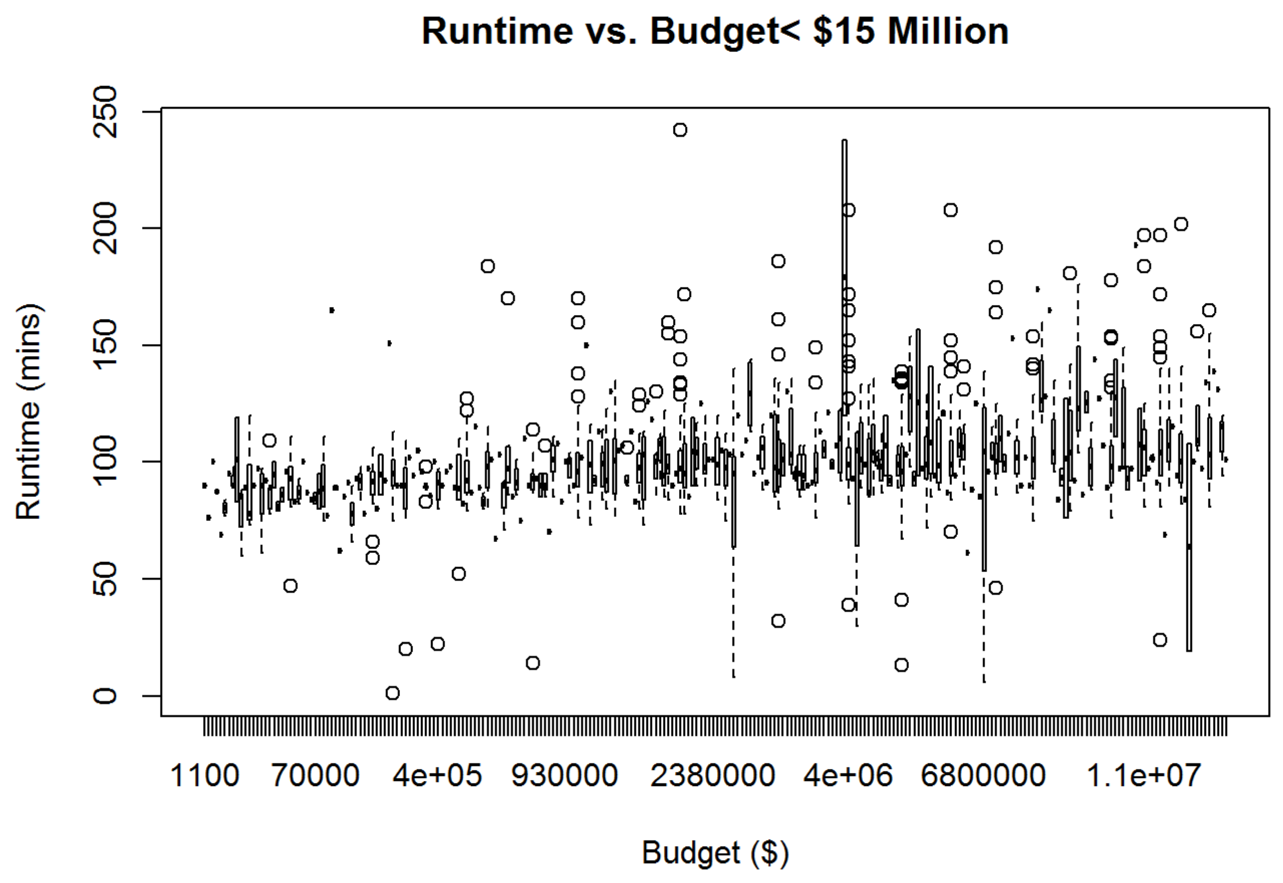
```
tempbucket = (df$Budget)/1000000
budgetBucket = tempbucket
budgetBucket[tempbucket>=200] = "G. 200 & above"
budgetBucket[tempbucket>=150 & tempbucket<200] = "F. 150-200"
budgetBucket[tempbucket>=100 & tempbucket<150] = "E. 100-150"
budgetBucket[tempbucket>=75 & tempbucket<100] = "D. 75-100"
budgetBucket[tempbucket>=45 & tempbucket<75] = "C. 45-75"
budgetBucket[tempbucket>=15 & tempbucket<45] = "B. 15-45"
budgetBucket[tempbucket>=0 & tempbucket<15] = "A. 0-15"

df$budgetBucket = budgetBucket

ggplot(data=df,mapping=aes(x=budgetBucket,y=Runtime,color=budgetBucket))+
  geom_boxplot()+
  xlab("Budget Range (millions)")+
  ylab("Runtime (mins)")+
  ggtitle("Movie Runtimes by Budget")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 759 rows containing non-finite values (stat_boxplot).
```



Hide

```
boxplot(df$Runtime[df$budgetBucket=="A. 0-15"]~df$Budget[df$budgetBucket=="A. 0-15"],
        main="Runtime vs. Budget< $15 Million",
        ylab="Runtime (mins)",
        xlab="Budget ($)")
```



Feel free to insert additional code chunks as necessary.

Q: Comment on the distribution as well as relationships. Are there any patterns or trends that you can observe?

A: It is noticed that the distribution is has two predominant peaks. These most likely corrospond to the main movie categories: “Shorts” and “Feature Films”. Further more, there is an increasing trend in movie lengths till the 1970s then it seems to plateu between 90-100 mins.

As for the relationhip between budget and runtime, it seems that longer movies tend to have a larger cost which makes sense since on average longer movies tend to cost more than shorter movies since they require more resources and a higher budget over a longer period of time.

For the budget distribution, we notice that there are many low budget films <= 15 million. To investigate it further, I looked at there relation to runtime and it still seems to follow the previous linear trend where the runtime increases as the budget increases.

3. Encode `Genre` column

The column `Genre` represents a list of genres associated with the movie in a string format. Write code to parse each text string into a binary vector with 1s representing the presence of a genre and 0s the absence, and add it to the dataframe as additional columns. Then remove the original `Genre` column.

For example, if there are a total of 3 genres: Drama, Comedy, and Action, a movie that is both Action and Comedy should be represented by a binary vector <0, 1, 1>. Note that you need to first compile a dictionary of all possible genres and then figure out which movie has which genres (you can use the R `tm` package to create the dictionary).

Hide

```
# TODO: Replace Genre with a collection of binary columns
library('tm')
```

```
Warning: package 'tm' was built under R version 3.3.2
```

```
Loading required package: NLP
```

```
Warning: package 'NLP' was built under R version 3.3.2
```

```


Attaching package: 'NLP'
```

```
The following object is masked from 'package:ggplot2':

    annotate
```

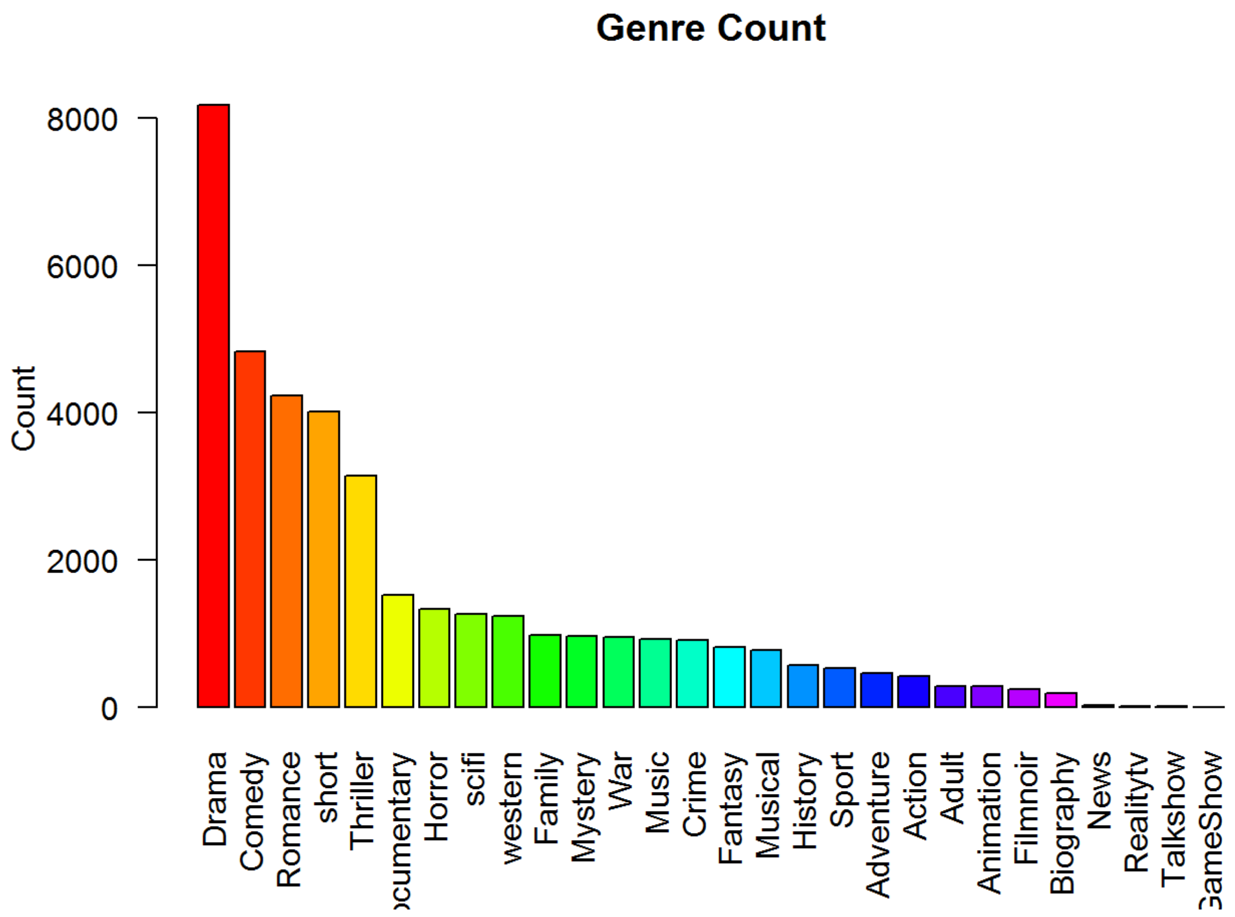
Hide

```
splitGenresToVector = function(genreString){
  stringVector = strsplit(genreString,", ")
  return (stringVector)
}

genreColumn = vector(length=length(df$Genre))

i = 1
for (entry in df$Genre){
  genreColumn[i] = splitGenresToVector(entry)
  i=i+1
}

allGenres = c()
for(i in seq(1,length(df$Genre))){
  temp = genreColumn[[i]]
  for(genre in temp){
    allGenres = c(allGenres,genre)
  }
}
uniqueGenres = unique(allGenres)

df$Documentary = vector(mode="numeric", length=length(df$Genre))
df$Biography = vector(mode="numeric", length=length(df$Genre))
df$Romance = vector(mode="numeric", length=length(df$Genre))
df$short = vector(mode="numeric", length=length(df$Genre))
df$Thriller = vector(mode="numeric", length=length(df$Genre))
df$Drama = vector(mode="numeric", length=length(df$Genre))
df$Documentary = vector(mode="numeric", length=length(df$Genre))
df$War = vector(mode="numeric", length=length(df$Genre))
df$Comedy = vector(mode="numeric", length=length(df$Genre))
df$Horror = vector(mode="numeric", length=length(df$Genre))
df$scifi = vector(mode="numeric", length=length(df$Genre))
df$Adventure = vector(mode="numeric", length=length(df$Genre))
df$Family = vector(mode="numeric", length=length(df$Genre))
df$History = vector(mode="numeric", length=length(df$Genre))
df$Crime = vector(mode="numeric", length=length(df$Genre))
df$Action = vector(mode="numeric", length=length(df$Genre))
df$Music = vector(mode="numeric", length=length(df$Genre))
df$Mystery = vector(mode="numeric", length=length(df$Genre))
df$Fantasy = vector(mode="numeric", length=length(df$Genre))
df$Sport = vector(mode="numeric", length=length(df$Genre))
df$Animation = vector(mode="numeric", length=length(df$Genre))
df$Musical = vector(mode="numeric", length=length(df$Genre))
df$Talkshow = vector(mode="numeric", length=length(df$Genre))
df$Adult = vector(mode="numeric", length=length(df$Genre))
df$western = vector(mode="numeric", length=length(df$Genre))
df$Filmnoir = vector(mode="numeric", length=length(df$Genre))
df$Realitytv = vector(mode="numeric", length=length(df$Genre))
df$News = vector(mode="numeric", length=length(df$Genre))
df$GameShow = vector(mode="numeric", length=length(df$Genre))

for(i in seq(1,length(df$Genre))){
```

```
temp = genreColumn[[i]]
for(genre in temp){
  if (genre =="Documentary"){
    df$Documentary[i] = 1
  } else{
    df$Documentary[i] = 0
  }
  if (genre=="Biography"){
      df$Biography[i] = 1
  }else{
      df$Biography[i] = 0
  }
  if (genre=="Romance"){
    df$Romance[i] = 1
  }else{
    df$Romance[i] = 0
  }
  if(genre=="Short"){
    df$short[i] = 1
  }else{
    df$short[i] = 0
  }
if(genre== "Thriller"){
  df$Thriller[i] = 1
}else{
  df$Thriller[i] = 0}

  if(genre=="Drama" ){
   df$Drama[i] = 1
  }else{df$Drama[i] = 0}
if(genre=="War"){
  df$War[i] = 1}
  else{df$War[i] = 0}

  if(genre=="Comedy"){
    df$Comedy[i]=1}
  else{df$Comedy[i]=0
  }
  if(genre=="Horror"){
    df$Horror[i] = 1
  }else{df$Horror[i] = 0}

  if(genre=="Sci-Fi"){
    df$scifi[i] = 1
  }else{df$scifi[i] = 0}

  if(genre=="Adventure"){
    df$Adventure[i]=1}
  else{df$Adventure[i]=0
  }

  if(genre=="Family"){
    df$Family[i]= 1}
  else{df$Family[i]= 0
```

```
    }
    if(genre=="History"){df$History[i] = 1
    }else{df$History[i] = 0
    }
    if(genre=="Crime"){df$Crime[i]=1
    }else{df$Crime[i]=0
    }
    if(genre=="Action"){df$Action[i] = 1
    }else{df$Action[i] = 0
    }
    if(genre=="Music"){
      df$Music[i] = 1
    }else{df$Music[i] = 0}

    if(genre=="Mystery"){df$Mystery[i] = 1
    }else{df$Mystery[i] = 0}

    if(genre=="Fantasy"){
      df$Fantasy[i] = 1
    }else{df$Fantasy[i] = 0}

    if(genre=="Sport"){
      df$Sport[i] = 1
    }else{df$Sport[i] = 0}

    if(genre=="Animation"){
      df$Animation[i] = 1
    }else{df$Animation[i] = 0}
    if(genre=="Musical"){df$Musical[i] = 1
    }else{
      df$Musical[i] = 0}
    if(genre=="Talk-Show"){df$Talkshow[i] = 1
    }else{df$Talkshow[i] = 0}

    if(genre=="Adult"){df$Adult[i] = 1
    }else{df$Adult[i] = 0}

    if(genre=="Western"){df$western[i] = 1
    }else{df$western[i] = 0}

    if(genre=="Film-Noir"){df$Filmnoir[i] = 1
    }else{df$Filmnoir[i] = 0}

    if(genre=="Reality-TV"){df$Realitytv[i] = 1
    }else{df$Realitytv[i] = 0}

    if(genre=="News"){df$News[i] = 1
    }else{df$News[i] = 0}

    if(genre=="Game-Show"){
      df$GameShow[i] = 1
    }else{df$GameShow[i] = 0}

  }
```

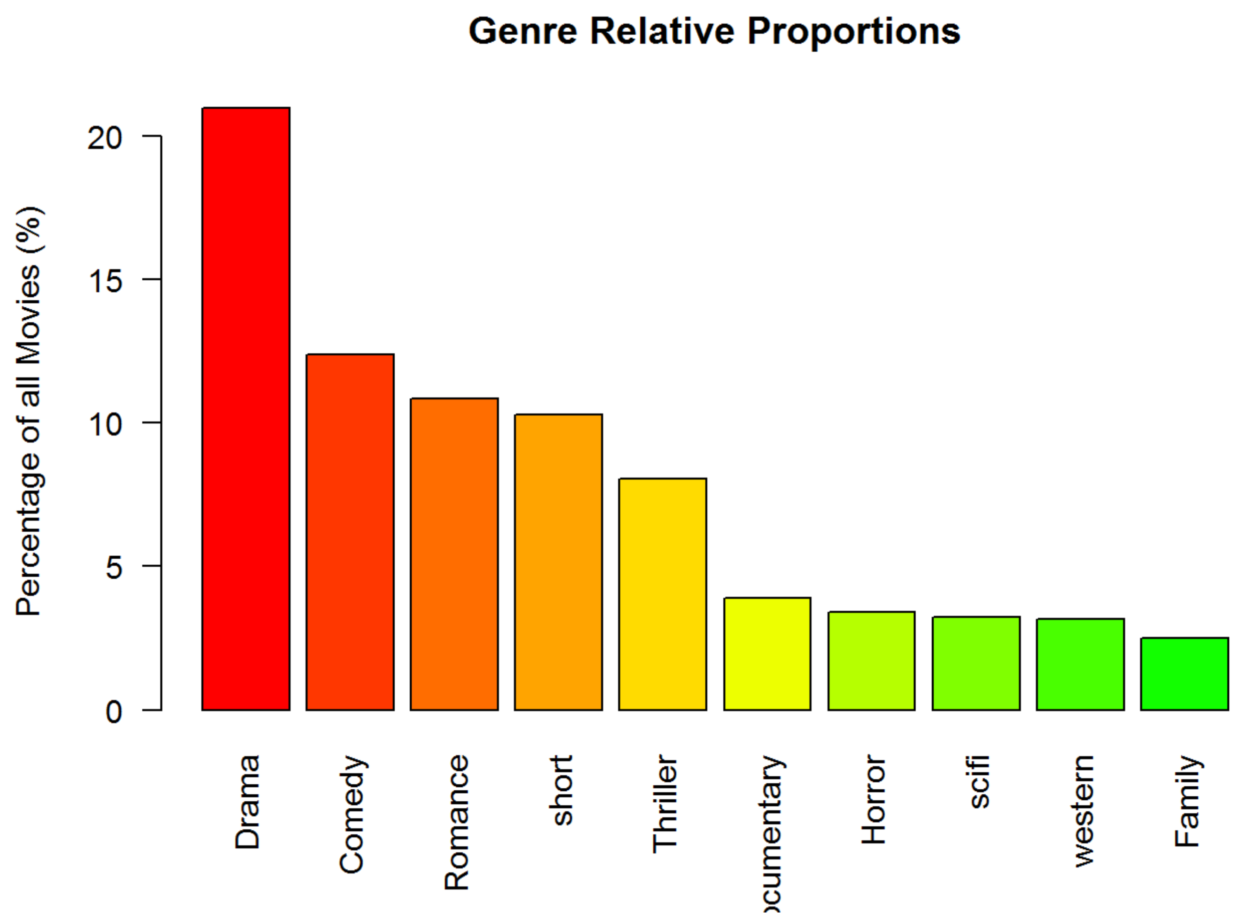
```
}

GenreSummation = sort(colSums(df[,42:69]),decreasing = TRUE)

GenreSummationProportion = GenreSummation/sum(GenreSummation)*100.0


barplot(GenreSummation, main="Genre Count", ylab="Count",col=rainbow(28),las=2)
```



Plot the relative proportions of movies having the top 10 most common genres.

Hide

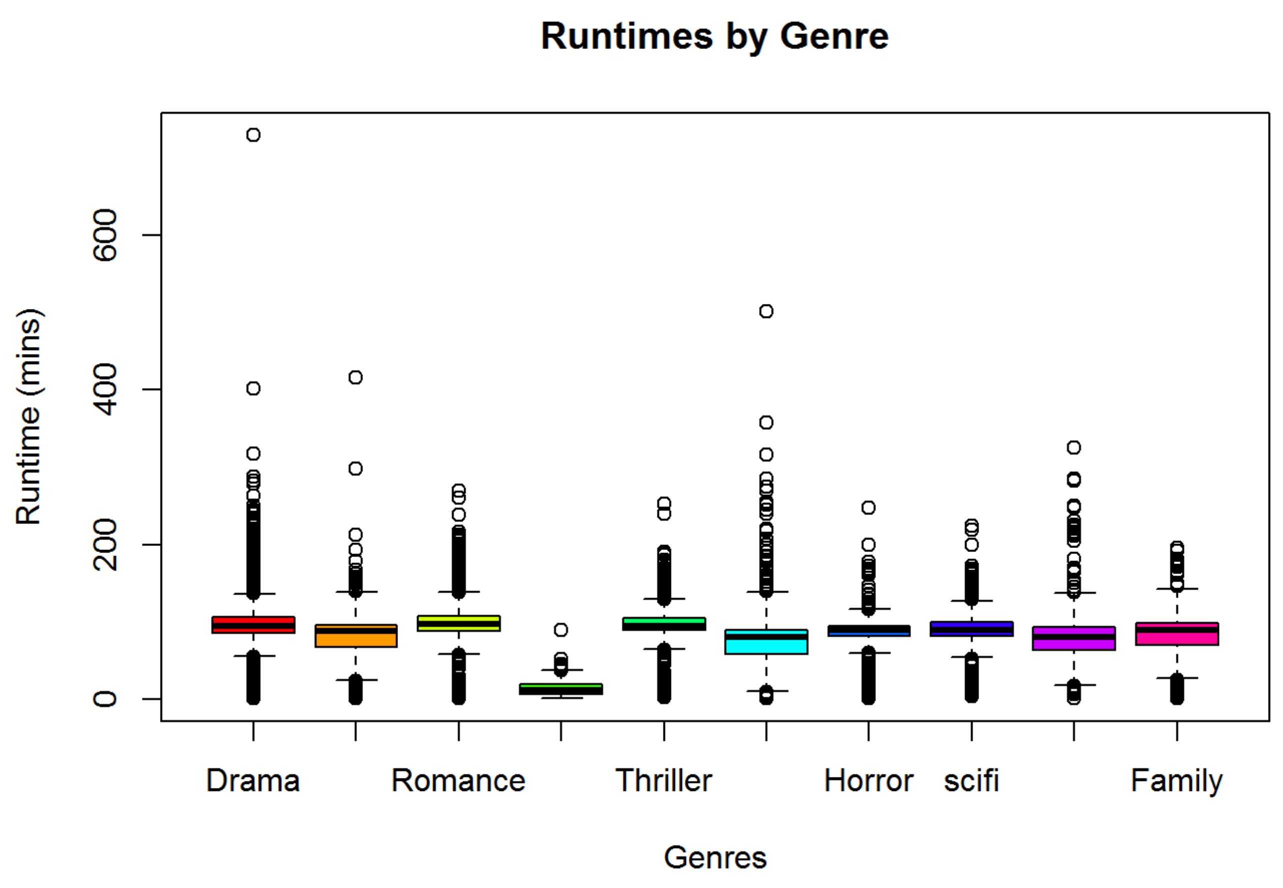
```
# TODO: Select movies from top 10 most common genres and plot their relative proportions
barplot(GenreSummationProportion[1:10], main="Genre Relative Proportions", ylab="Percentage of a
ll Movies (%)",col=rainbow(28),las=2)
```



Examine how the distribution of `Runtime` changes across genres for the top 10 most common genres.

Hide

```
# TODO: Plot Runtime distribution for top 10 most common genres
RuntimeByGenres = df[,names(GenreSummationProportion[1:10])]
RuntimeByGenres$Drama[RuntimeByGenres$Drama==1] = df$Runtime[df$Drama==1]
RuntimeByGenres$Drama[RuntimeByGenres$Drama==0] = NaN

RuntimeByGenres$Comedy[RuntimeByGenres$Comedy==1] = df$Runtime[df$Comedy==1]
RuntimeByGenres$Comedy[RuntimeByGenres$Comedy==0] = NaN


RuntimeByGenres$Romance[RuntimeByGenres$Romance==1] = df$Runtime[df$Romance==1]
RuntimeByGenres$Romance[RuntimeByGenres$Romance==0] = NaN

RuntimeByGenres$short[RuntimeByGenres$short==1] = df$Runtime[df$short==1]
RuntimeByGenres$short[RuntimeByGenres$short==0] = NaN

RuntimeByGenres$Thriller[RuntimeByGenres$Thriller==1] = df$Runtime[df$Thriller==1]
RuntimeByGenres$Thriller[RuntimeByGenres$Thriller==0] = NaN

RuntimeByGenres$Documentary[RuntimeByGenres$Documentary==1] = df$Runtime[df$Documentary==1]
RuntimeByGenres$Documentary[RuntimeByGenres$Documentary==0] = NaN

RuntimeByGenres$Horror[RuntimeByGenres$Horror==1] = df$Runtime[df$Horror==1]
RuntimeByGenres$Horror[RuntimeByGenres$Horror==0] = NaN

RuntimeByGenres$scifi[RuntimeByGenres$scifi==1] = df$Runtime[df$scifi==1]
RuntimeByGenres$scifi[RuntimeByGenres$scifi==0] = NaN

RuntimeByGenres$western[RuntimeByGenres$western==1] = df$Runtime[df$western==1]
RuntimeByGenres$western[RuntimeByGenres$western==0] = NaN

RuntimeByGenres$Family[RuntimeByGenres$Family==1] = df$Runtime[df$Family==1]
RuntimeByGenres$Family[RuntimeByGenres$Family==0] = NaN

library(ggplot2)
boxplot(RuntimeByGenres,xlab="Genres",ylab="Runtime (mins)",main="Runtimes by
Genre",col=rainbow(10))
```



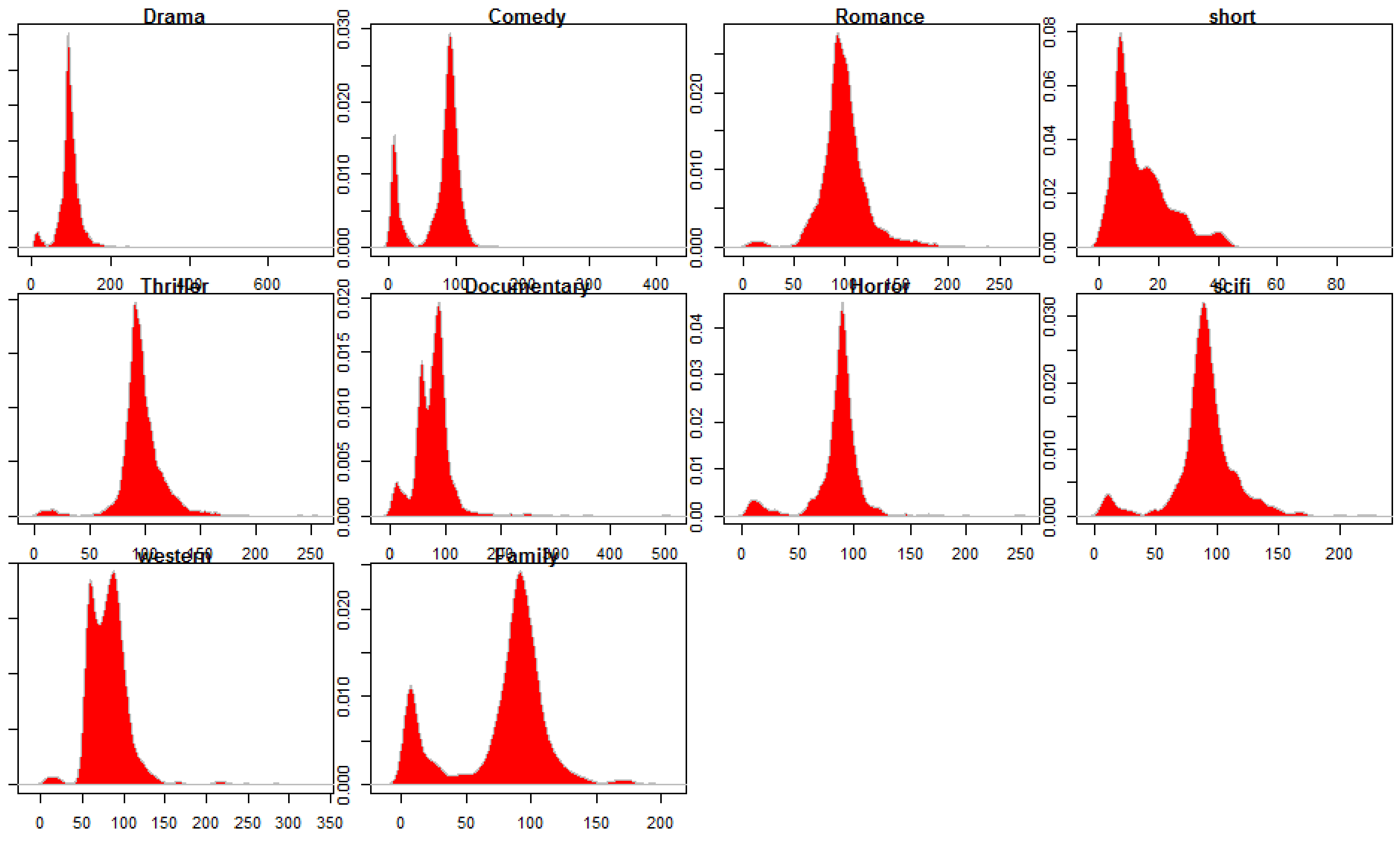
Hide

```
graphics.off()
par("mar")
```

```
[1] 5.1 4.1 4.1 2.1
```

Hide

```
par(mar=c(1,1,1,1))

par(mfrow=c(4, 4))
colnames <- dimnames(RuntimeByGenres)[[2]]
for (i in 1:10) {
    d <- density(na.omit(RuntimeByGenres[,i]))
    plot(d, type="n", main=colnames[i])
    polygon(d, col="red", border="gray")
}
```



Q: Describe the interesting relationship(s) you observe. Are there any expected or unexpected trends that are evident?

A: As Expected, shorts have the shortest runtimes on average. Most other movie categories have almost the same average runtimes except for comedy, documentary and western are slightly shorter on average. It was expected that comdedy films will be shorter as they usually they don't tell an engaging story like other categories which usually requires a longer runtime. I did not expect that documentary movies are shorter on average as I thought they usually run longer.

When looking at distributions, interesting observations were made. Specifically, most movies seemed to have single peaks for runtime except for comedy and family movies where both were multimodal with two main peaks. Comedy & Kids seem to have two types of movies: long (~100 mins) and short (~10 mins). These two far away peaks in the comedy category might explain why on average it had a lower runtime than other movies. In contrast, drama films seemed to have little variance (narrow peak at around 100).

4. Eliminate mismatched rows

The dataframe was put together by merging two different sources of data and it is possible that the merging process was inaccurate in some cases (the merge was done based on movie title, but there are cases of different movies with the same title). The first sourceâ™s release time was represented by the column `Year` (numeric representation of the year) and the second by the column `Released` (string representation of release date).

Find and remove all rows where you suspect a merge error occurred based on a mismatch between these two variables. To make sure subsequent analysis and modeling work well, avoid removing more than 10% of the rows that have a `Gross` value present.

Hide

```
# TODO: Remove rows with Released-Year mismatch
secondYearColumn = na.omit(df$Released)
secondYearColumnLength = length(na.omit(secondYearColumn))
newYear = vector(mode="numeric", length=secondYearColumnLength)

i=1
for(releaseDate in secondYearColumn){
    newYear[i] = as.numeric(substr(secondYearColumn[i],1,4))
    i=i+1
}
noMismatch = subset(df, (!is.na(df[,4])) )
#noMismatch$Year2 = newYear
noMismatch = noMismatch[noMismatch$Year==newYear,] #This is the dataframe where Released year an
d Year Match

ReleasedMissing = subset(df, (is.na(df[,4])) )#This is the dataframe where released is missing

newdf = rbind(noMismatch[,1:69],ReleasedMissing) #This is the joined data frame

remainingMoviePercentage = nrow(newdf)/nrow(df)*100.0
remainingMoviePercentage #This is the remaining movies dataframe
```

```
[1] 85.6825
```

Q: What is your precise removal logic and how many rows did you end up removing?

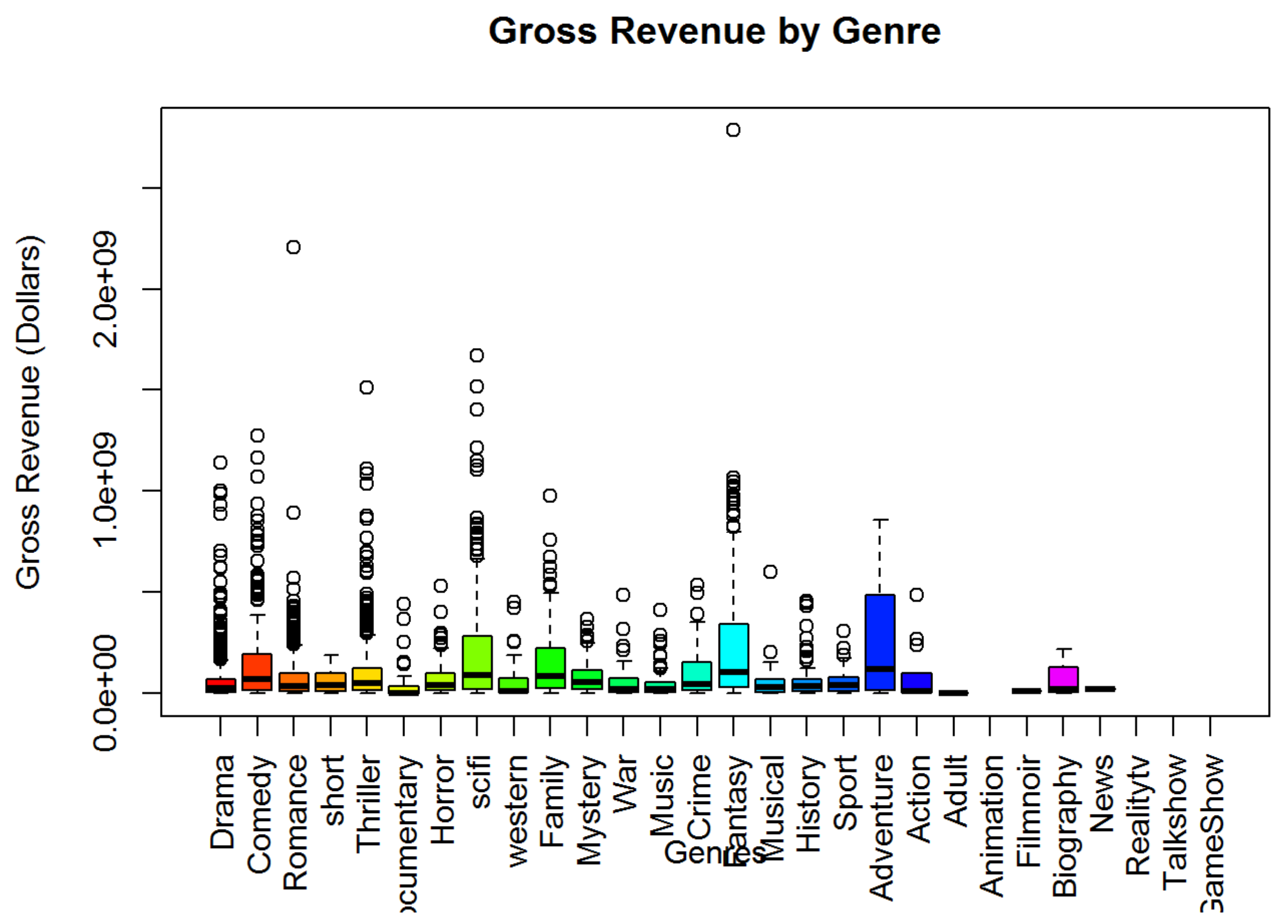
A: I first extracted the year from the “Released” attribute. Then, converted to numeric value and compared to the “year” attribute. In case of a mismatch between them, I removed that row. I decided to keep the movies where “Released” has a N/A value since I cannot really tell if they are matching or not. 8 ## 5. Explore `Gross` revenue

For the commercial success of a movie, production houses want to maximize Gross revenue. Investigate if Gross revenue is related to Budget, Runtime or Genre in any way.

Note: To get a meaningful relationship, you may have to partition the movies into subsets such as short vs. long duration, or by genre, etc.

Hide

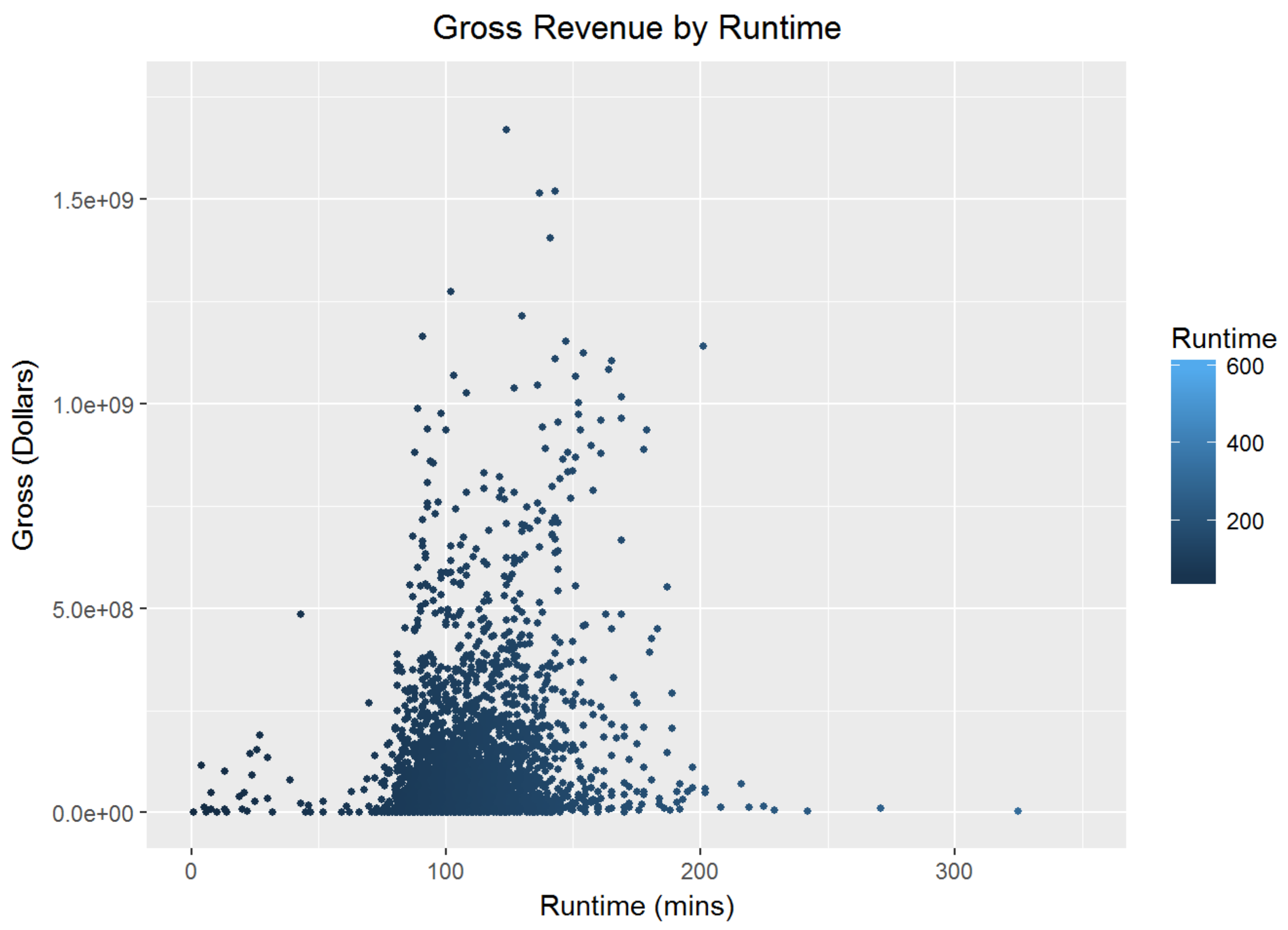
```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre

#Genre & Gross Revenue
GrossRevenueByGenres = newdf[,names(GenreSummationProportion)]*as.vector(newdf$Gross)
GrossRevenueByGenres[GrossRevenueByGenres == 0] <- NaN
boxplot(GrossRevenueByGenres,xlab="Genres",ylab="Gross Revenue (Dollars)",main="Gross Revenue by
 Genre",col=rainbow(28),las=3)
```



Hide

```
#Runtime and Gross Revenue
ggplot(data=newdf,mapping=aes(x=Runtime,y=Gross,color=Runtime))+
  geom_point(size=1)+
  xlab("Runtime (mins)")+
  ylab("Gross (Dollars)")+
  xlim(0,350)+
  ylim(0,1.75e09)+
  ggtitle("Gross Revenue by Runtime")+
  theme(plot.title = element_text(hjust = 0.5))
```

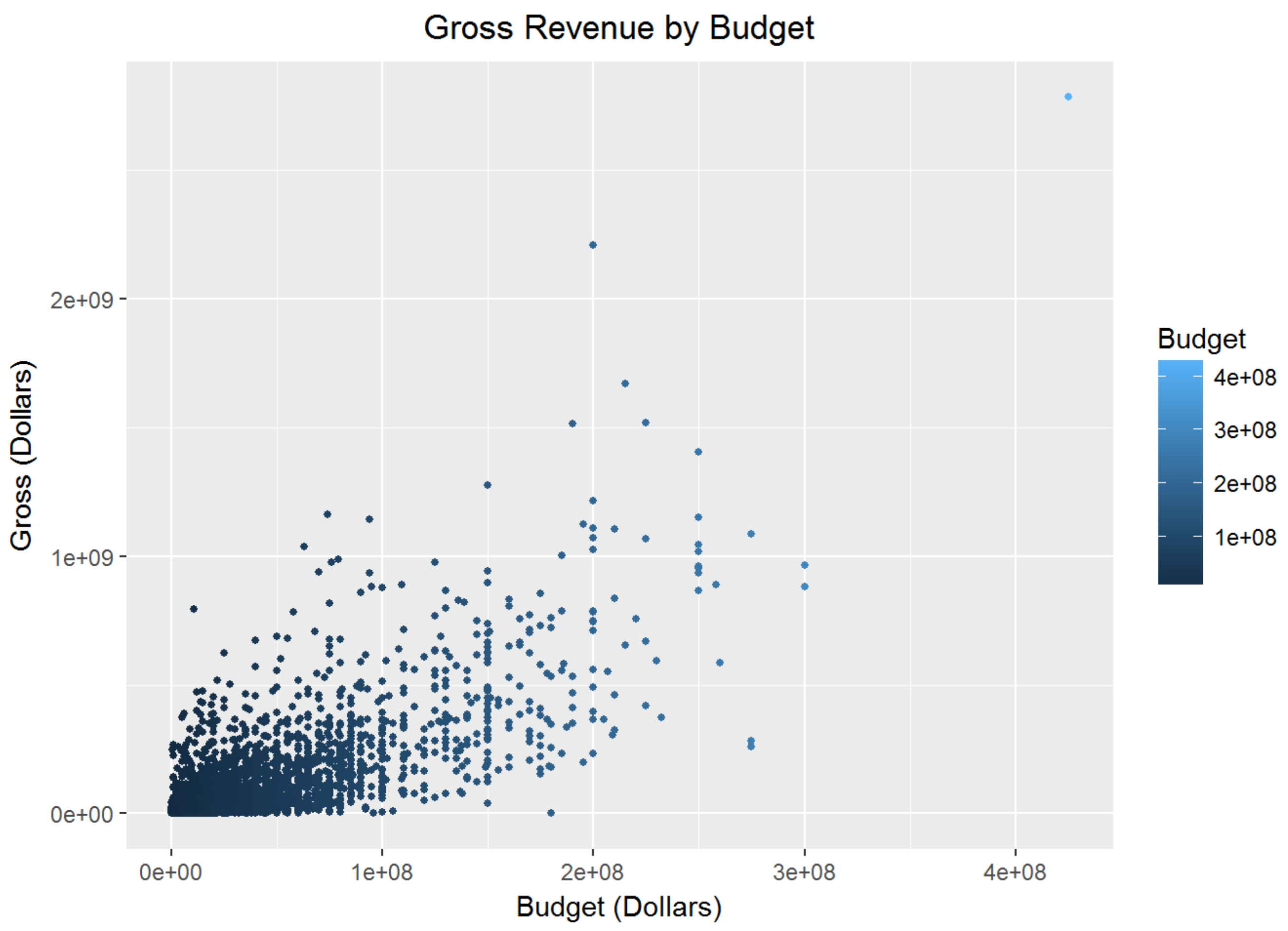
```
Warning: Removed 30525 rows containing missing values (geom_point).
```



Hide

```
#Budget and Gross Revenue
ggplot(data=newdf,mapping=aes(x=Budget,y=Gross,color=Budget))+
  geom_point(size=1)+
  xlab("Budget (Dollars)")+
  ylab("Gross (Dollars)")+
  ggtitle("Gross Revenue by Budget")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 30485 rows containing missing values (geom_point).
```



Hide

```
cor(newdf[,c(5,36,38)], use="complete.obs", method="pearson") #Runtime, Budget and Gross Correla
tions
```

```
          Runtime    Budget     Gross
Runtime 1.0000000 0.2913913 0.2631453
Budget  0.2913913 1.0000000 0.7390704
Gross   0.2631453 0.7390704 1.0000000
```

Q: Did you find any observable relationships or combinations of Budget/Runtime/Genre that result in high Gross revenue? If you divided the movies into different subsets, you may get different answers for them - point out interesting ones.

A: In the case of Genre, Adventure, Fantasy and scifi seemed to have the highest median gross revenue which does not follow the most popular movies. I expected the most popular movies to be the top grossing movies but it was not the case.

For the case of the budget, it was clear that there was correlation between the two. As budget increases, the Revenue increases. Runtime on the other hand, did not seem to have this relationship. To verify this, I calculated the Pearson correlation coefficients for both variables against Gross and it was observed that Gross and Budget had a value of 0.739 whereas Gross and Runtime had a value of 0.26314.

When looking at release months versus gross, we notice that theyre are two periods in the year with higher gross values. These are during summer break (May - July) and during Thanksgiving(November) and December. It is interesting to see that these vacation periods have the highest gross revenue values. It could be attributed that many movie makers tend to release movies during those times to gain popularity and increase the number of viewers.

Hide

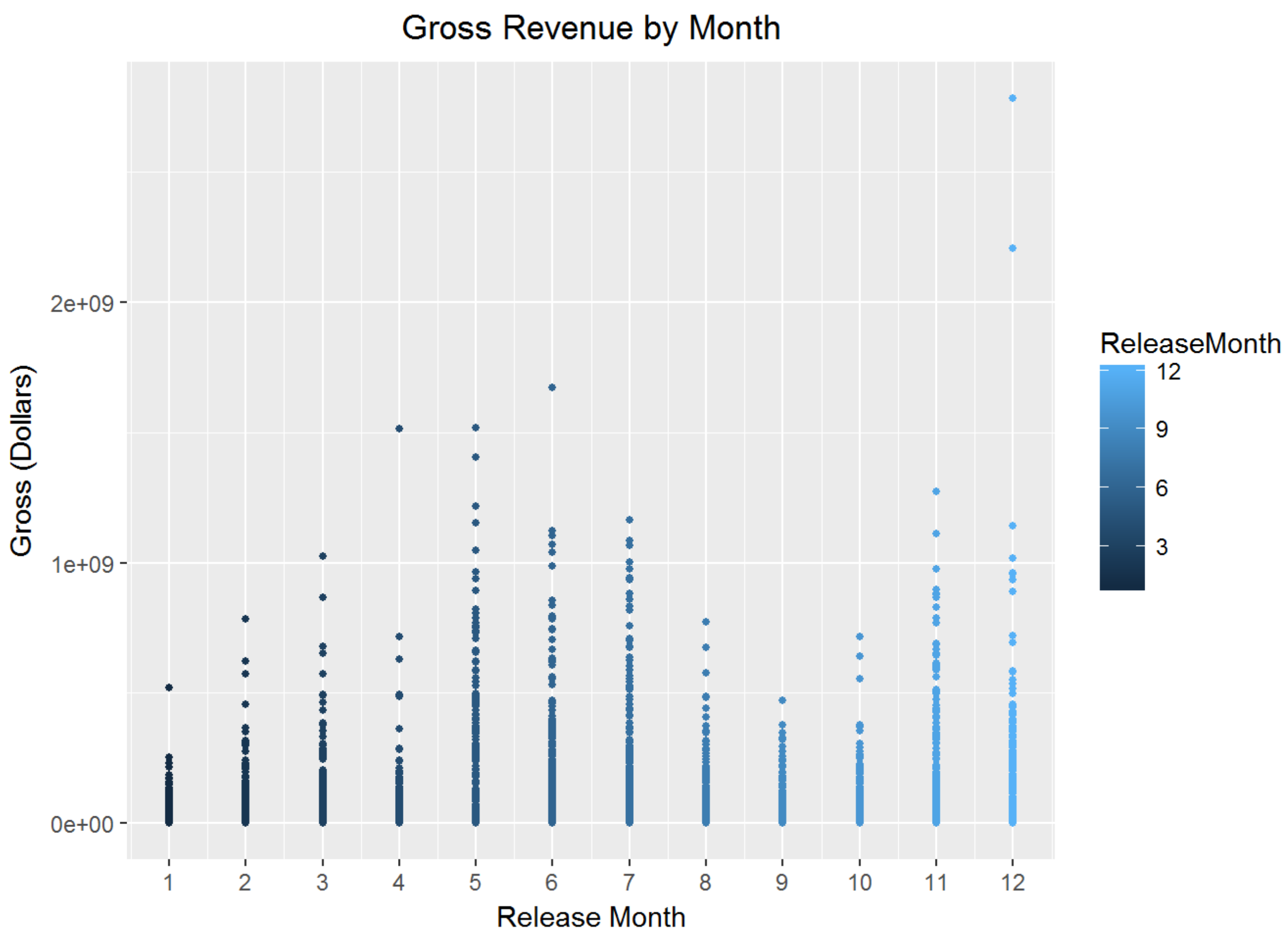
```
# TODO: Investigate if Gross Revenue is related to Release Month
newMonth = vector(mode="numeric", length=length(newdf$Released))
newdf$Released[2]
```

```
[1] "2005-01-25"
```

Hide

```
i=1
for(releaseDate in newdf$Released){
    newMonth[i] = as.numeric(substr(newdf$Released[i],6,7))
    i=i+1
}

newdf$ReleaseMonth = newMonth

ggplot(data=newdf,mapping=aes(x=ReleaseMonth,y=Gross,color=ReleaseMonth))+
  geom_point(size=1)+
  xlab("Release Month")+
  ylab("Gross (Dollars)")+
  ggtitle("Gross Revenue by Month")+
  theme(plot.title = element_text(hjust = 0.5))+
  scale_x_continuous(breaks=seq(1, 12, 1))
```

```
Warning: Removed 30530 rows containing missing values (geom_point).
```



6. Process `Awards` column

The variable `Awards` describes nominations and awards in text format. Convert it to 2 numeric columns, the first capturing the number of wins, and the second capturing nominations. Replace the `Awards` column with these new columns, and then study the relationship of `Gross` revenue with respect to them.

Note that the format of the `Awards` column is not standard; you may have to use regular expressions to find the relevant values. Try your best to process them, and you may leave the ones that don't have enough information as NAs or set them to 0s.

Hide

```
# TODO: Convert Awards to 2 numeric columns: wins and nominations
newAwards = gsub('s', '', newdf$Awards)
numAward= vector(mode="numeric", length=length(newAwards))
numNomination = vector(mode="numeric", length=length(newAwards))

for(i in seq(1,length(newAwards))){
    current = newAwards[i]
    noNumbers = gsub('[0-9]+', '', newAwards[i])
    numberList = na.omit(as.numeric(unlist(strsplit(newAwards[i], "[^0-9]+"))))

    if (noNumbers==" win."){
      numAward[i] = numberList[1]
      numNomination[i]= 0
    }
    if (noNumbers=="N/A"){
      numAward[i] =0
      numNomination[i]=0
    }

if (noNumbers== " win &  nomination."){
      numAward[i] = numberList[1]
      numNomination[i]=numberList[2]
    }

if (noNumbers== " nomination."){
      numAward[i] = 0
      numNomination[i]= numberList[1]
    }

if (noNumbers== "Nominated for  Ocar. Another  win &  nomination." ){
      numAward[i] = numberList[2]
      numNomination[i]= numberList[1]+numberList[3]
    }
if (noNumbers=="Won  Ocar. Another  win &  nomination."){
      numAward[i] = numberList[1]+numberList[2]
      numNomination[i]= numberList[3]
    }
if (noNumbers=="Won  Golden Globe. Another  win &  nomination."){
      numAward[i] = numberList[1]+numberList[2]
      numNomination[i]=numberList[3]
    }
 if (noNumbers=="Nominated for  Primetime Emmy. Another  win &  nomination."){
      numAward[i] = numberList[2]
      numNomination[i]= numberList[1]+numberList[3]
    }
 if (noNumbers=="Nominated for  Golden Globe. Another  nomination."){
      numAward[i] = 0
      numNomination[i]=numberList[1]+numberList[2]
    }

 if (noNumbers== "Nominated for  Golden Globe. Another  win &  nomination."){
      numAward[i] = numberList[2]
      numNomination[i]= numberList[1]+numberList[3]
```

```
     }
  if (noNumbers=="Nominated for  BAFTA Film Award. Another  win &  nomination."){
      numAward[i] = numberList[2]
       numNomination[i]= numberList[1]+numberList[3]
     }
 if (noNumbers=="Won  Primetime Emmy. Another  win &  nomination."){
      numAward[i] = numberList[1]+numberList[2]
       numNomination[i]= numberList[3]
     }

 if (noNumbers== "Nominated for  Ocar. Another  win."){
       numAward[i] = numberList[2]
       numNomination[i]= numberList[1]
     }
  if (noNumbers=="Nominated for  Ocar. Another  nomination."){
       numAward[i] = 0
       numNomination[i]=numberList[1]+numberList[2]
     }
  if (noNumbers== "Won  Primetime Emmy. Another  nomination."){
       numAward[i] =numberList[1]
       numNomination[i]=numberList[2]
     }
  if (noNumbers== "Nominated for  BAFTA Film Award. Another  nomination."){
       numAward[i] = 0
       numNomination[i]= numberList[1]+numberList[2]
     }
  if (noNumbers== "Nominated for  BAFTA Film Award. Another  win."){
       numAward[i] = numberList[2]
       numNomination[i]=numberList[1]
     }
  if (noNumbers=="Nominated for  Primetime Emmy. Another  nomination."){
       numAward[i] = 0
       numNomination[i]= numberList[1]+numberList[2]
     }
  if (noNumbers=="Won  Ocar. Another  win."){
       numAward[i] = numberList[1]+numberList[2]
       numNomination[i]=0
     }
 if (noNumbers=="Won  BAFTA Film Award. Another  win."){
       numAward[i] = numberList[1]+numberList[2]
       numNomination[i]= 0
     }
 if (noNumbers== "Nominated for  Primetime Emmy. Another  win."){
       numAward[i] = numberList[2]
       numNomination[i]=numberList[1]
     }
  if (noNumbers== "Won  Golden Globe. Another  nomination."){
       numAward[i] = numberList[1]
       numNomination[i]=numberList[2]
     }
  if (noNumbers=="Won  BAFTA Film Award. Another  win &  nomination."){
       numAward[i] = numberList[1]+numberList[2]
       numNomination[i]=numberList[3]
     }
```

```
if (noNumbers=="Won  Ocar. Another  nomination."){
      numAward[i] = numberList[1]
      numNomination[i]=numberList[2]
    }
if (noNumbers=="Won  BAFTA Film Award. Another  nomination."){
      numAward[i] = numberList[1]
      numNomination[i]= numberList[2]
    }
if (noNumbers=="Nominated for  Golden Globe. Another  win."){
      numAward[i] = numberList[2]
      numNomination[i]=numberList[1]
    }
}

newdf$numAwards = numAward
newdf$numNomination = numNomination

colSums(newdf[,71:72] != 0)
```

```
    numAwards numNomination
         8552          8970
```

Q: How did you construct your conversion mechanism? How many rows had valid/non-zero wins or nominations?

A: I ommited the “s” and all digits from the awards column. I then looked at the different unique string formats which turned out to be 27 different formats. I used the strsplit function to extract the numbers in sequence. As I looped through the different values of “Awards” I checked which format out of the 27 templates matches the value and assigned the numbers that I got from the “strsplit” function according to the template’s order. (Example: if my value was “Won 1 BAFTA Film Award. Another 3 win & 5 nomination.”, strsplit will return A[1, 3, 5]. This will be assigned as follows: awards = A[1]+A[2]; nominations = A[3]).

As for the number of rows that had valid values: 8552 Awards & 8970 Nominations.

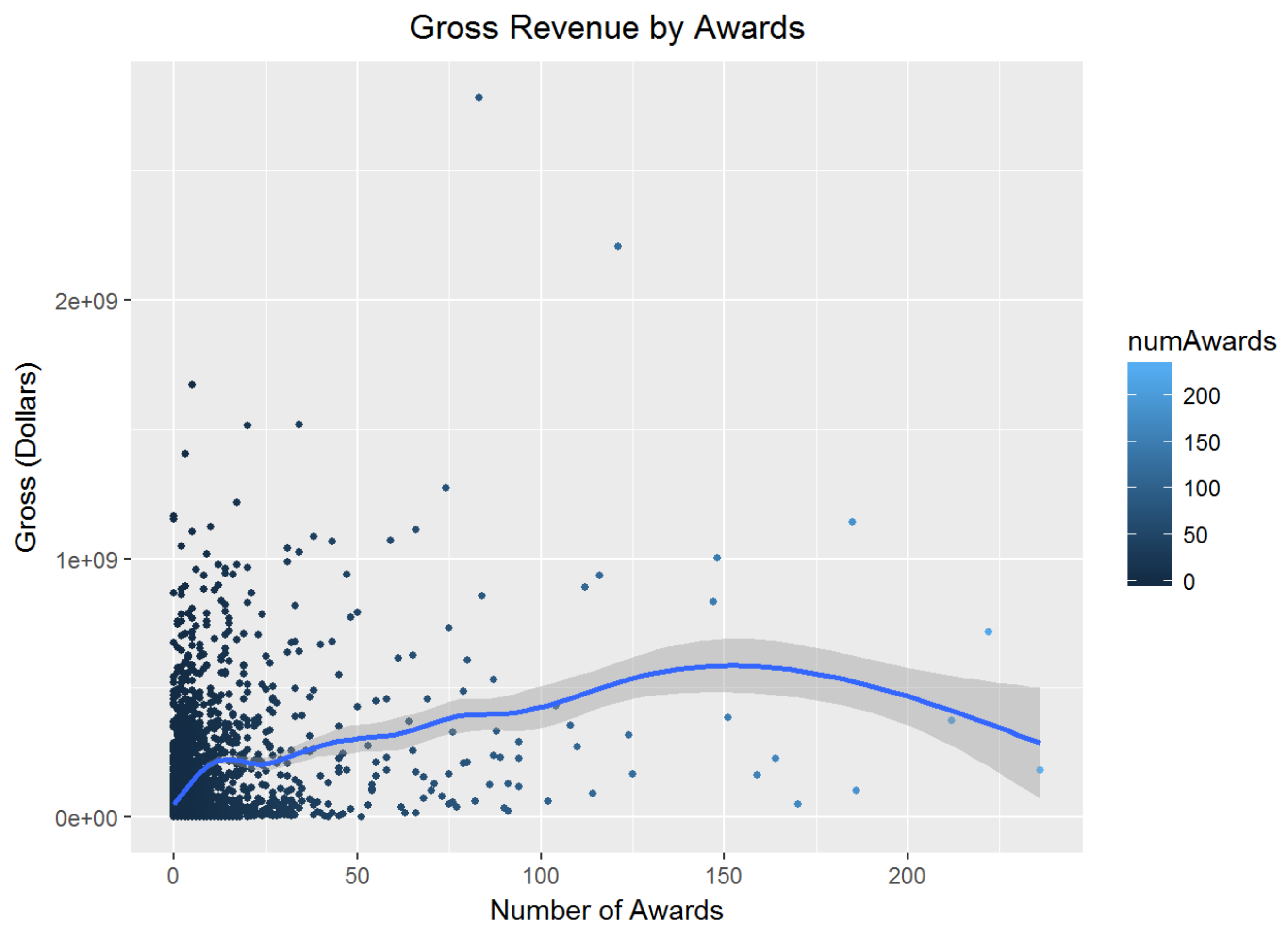
Hide

```
# TODO: Plot Gross revenue against wins and nominations
library(ggplot2)
ggplot(data=newdf,mapping=aes(x=numAwards,y=Gross,color=numAwards))+
  geom_point(size=1)+
  geom_smooth()+
  xlab("Number of Awards")+
  ylab("Gross (Dollars)")+
  ggtitle("Gross Revenue by Awards")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
`geom_smooth()` using method = 'gam'
```

```
Warning: Removed 30485 rows containing non-finite values (stat_smooth).
```

```
Warning: Removed 30485 rows containing missing values (geom_point).
```

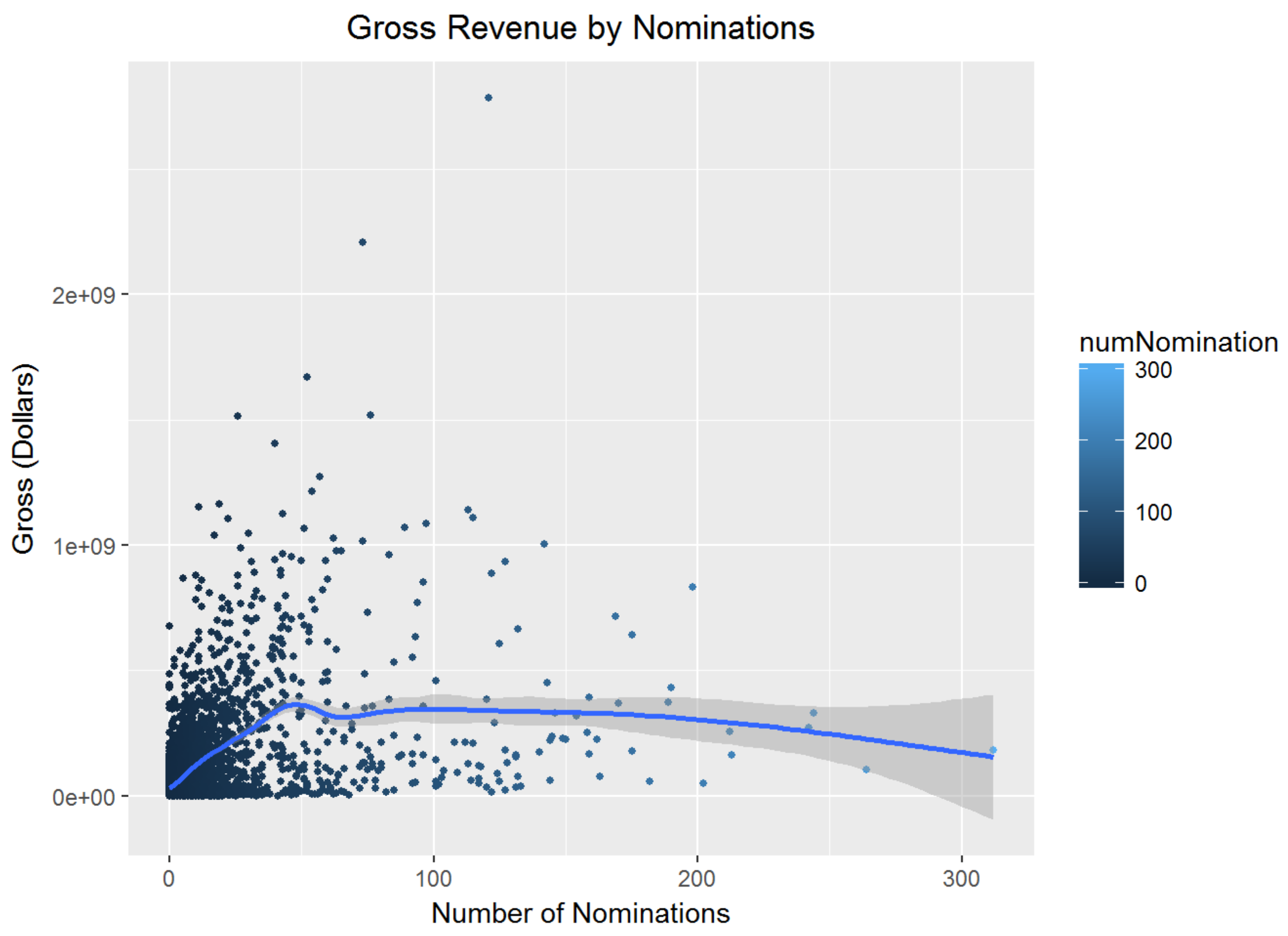


Hide

```
ggplot(data=newdf,mapping=aes(x=numNomination,y=Gross,color=numNomination))+
  geom_point(size=1)+
  geom_smooth()+
  xlab("Number of Nominations")+
  ylab("Gross (Dollars)")+
  ggtitle("Gross Revenue by Nominations")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
`geom_smooth()` using method = 'gam'
```

```
Warning: Removed 30485 rows containing non-finite values (stat_smooth).

Warning: Removed 30485 rows containing missing values (geom_point).
```



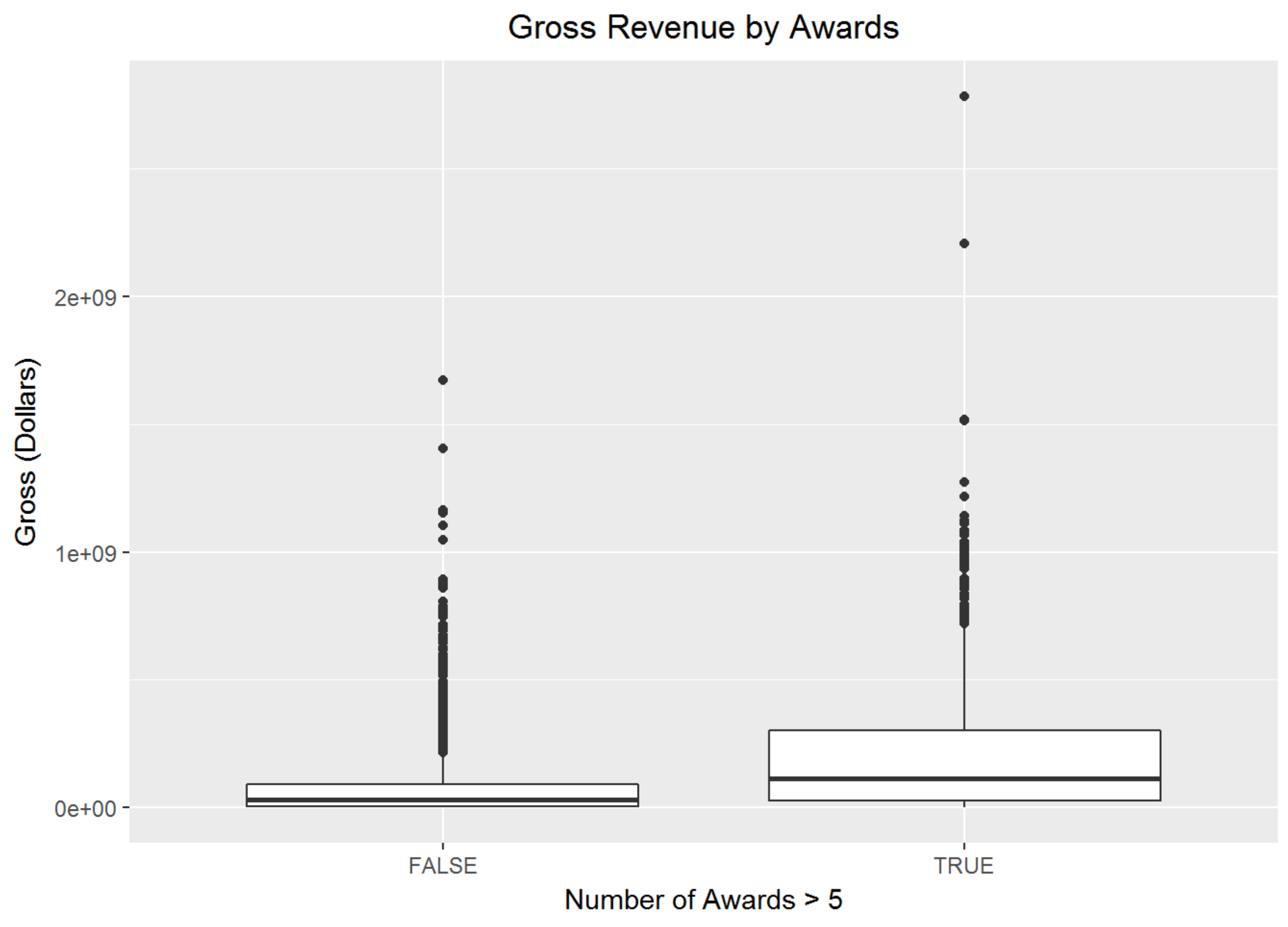
Hide

```
ggplot(data=newdf,mapping=aes(x=numAwards>5,y=Gross,color=numAwards))+
  geom_boxplot()+
  geom_smooth()+
  xlab("Number of Awards > 5")+
  ylab("Gross (Dollars)")+
  ggtitle("Gross Revenue by Awards")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 30485 rows containing non-finite values (stat_boxplot).
```

```
`geom_smooth()` using method = 'gam'
```

```
Warning: Removed 30485 rows containing non-finite values (stat_smooth).
```



Hide

```
cor(newdf[,c(71,72,38)], use="complete.obs", method="pearson")
```

```
              numAwards numNomination     Gross
numAwards     1.0000000     0.8394153 0.3407654
numNomination 0.8394153     1.0000000 0.3942925
Gross         0.3407654     0.3942925 1.0000000
```

Q: How does the gross revenue vary by number of awards won and nominations received?

A: There does not seem to be a very high correlation as we can see many high grossing movies with little awards and nominations as well as many low grossing movies with many awards and nominations. To understand this numerically I looked at the correlation between these gross and awards & nominations which had the following correlation table:

```
            numAwards     numNomination    Gross
```

numAwards 1.0000000 0.8394153 0.3407654 numNomination 0.8394153 1.0000000 0.3942925 Gross 0.3407654 0.3942925 1.0000000

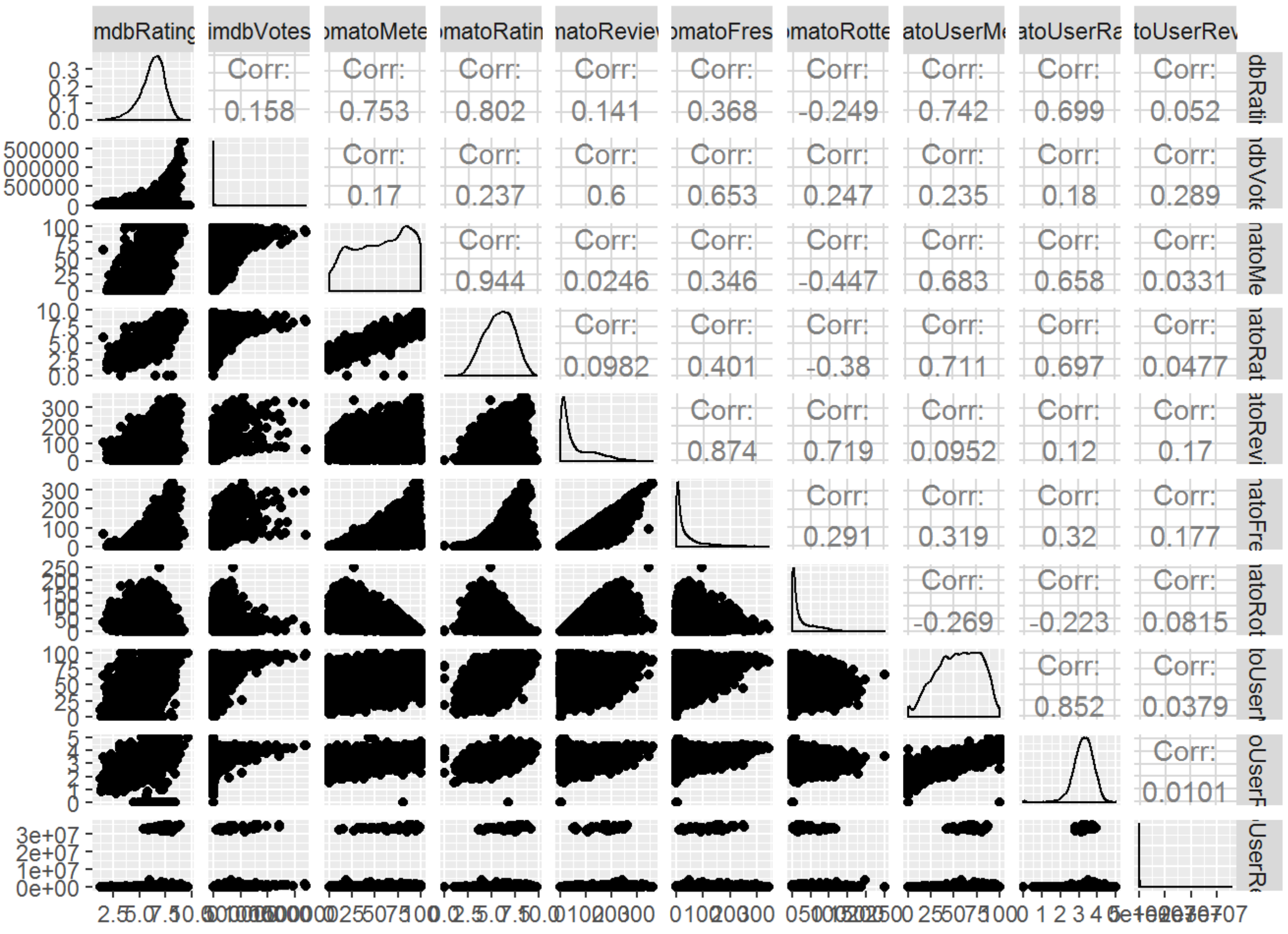
7. Movie ratings from IMDb and Rotten Tomatoes

There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by several variables pre-fixed by `tomato`). Read up on such ratings on the web (for example rottentomatoes.com/about (https://www.rottentomatoes.com/about) and www.imdb.com/help/show_leaf?votestopfaq (http://%20www.imdb.com/help/show_leaf?votestopfaq)).

Investigate the pairwise relationships between these different descriptors using graphs.

Hide

```
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
library(GGally)
ggpairs(newdf, columns=c(16,17,20,22,23,24,25,27,28,29))
```



Q: Comment on the similarities and differences between the user ratings of IMDb and the critics ratings of Rotten Tomatoes.

A: there seems to be a high correlation between the imdbRating and the tomatoRating (0.802). Also, there seems to be correlation between the tomatoRating with the critic and user ratings with fairly similar distributions.

8. Ratings and awards

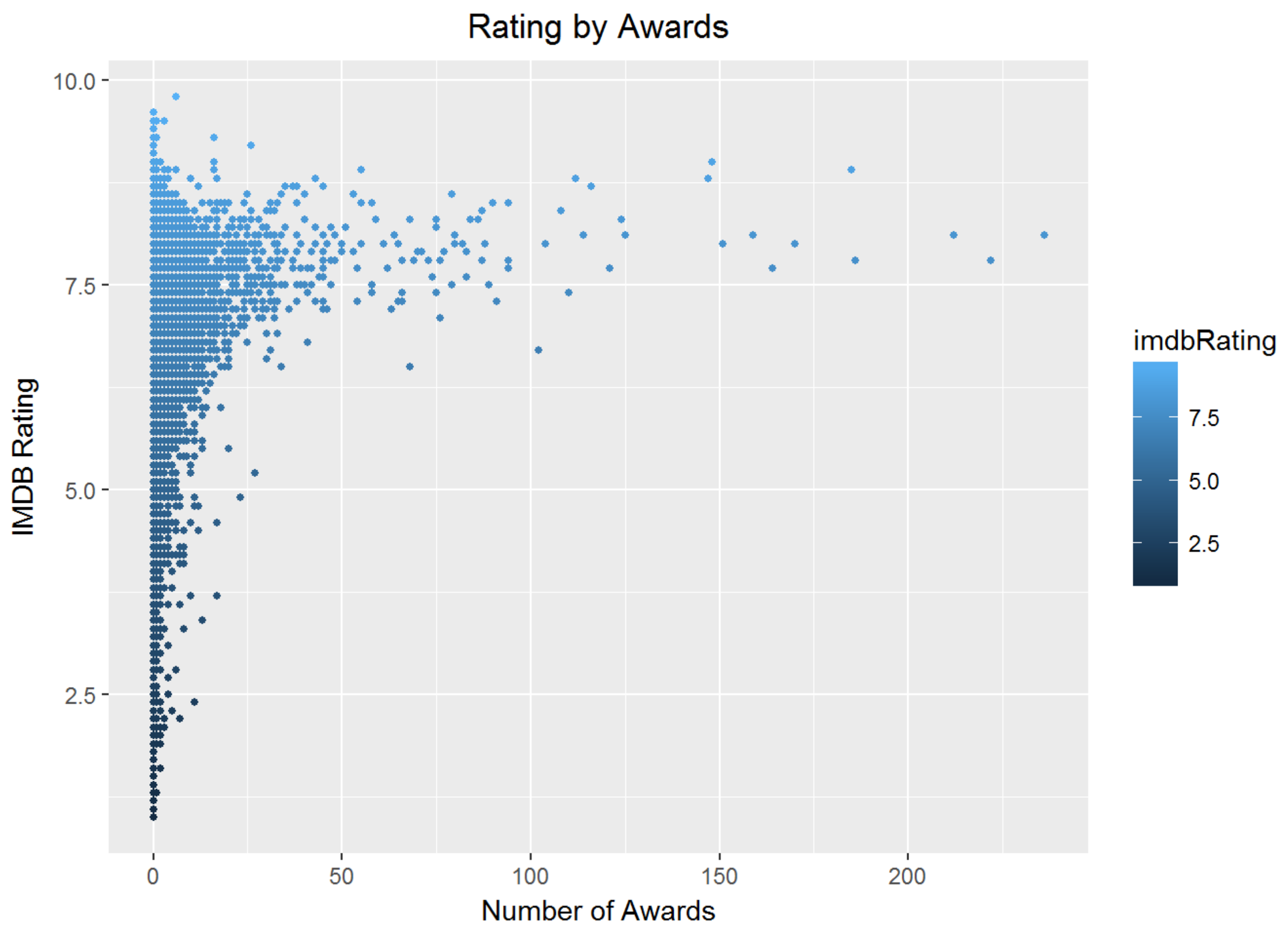
These ratings typically reflect the general appeal of the movie to the public or gather opinions from a larger body of critics. Whereas awards are given by professional societies that may evaluate a movie on specific attributes, such as artistic performance, screenplay, sound design, etc.

Study the relationship between ratings and awards using graphs (awards here refers to wins and/or nominations).

Hide

```
# TODO: Show how ratings and awards are related
ggplot(data=newdf  ,mapping=aes(x=numAwards,y=imdbRating,color=imdbRating))+
  geom_point(size=1)+
  xlab("Number of Awards")+
  ylab("IMDB Rating")+
  ggtitle("Rating by Awards")+
  theme(plot.title = element_text(hjust = 0.5))
```

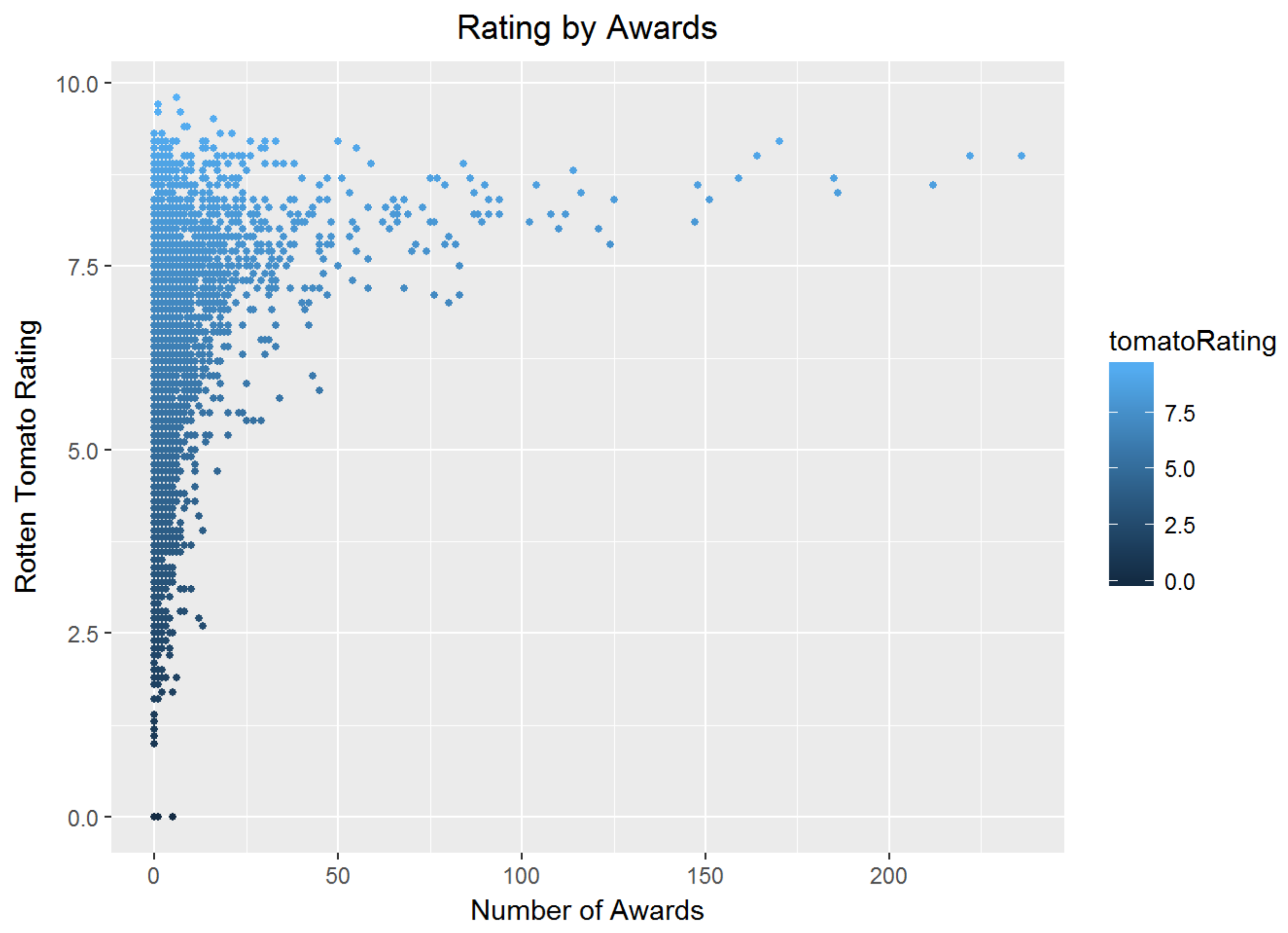
```
Warning: Removed 1123 rows containing missing values (geom_point).
```



Hide

```
ggplot(data=newdf ,mapping=aes(x=numAwards,y=tomatoRating,color=tomatoRating))+
  geom_point(size=1)+
  xlab("Number of Awards")+
  ylab("Rotten Tomato Rating")+
  ggtitle("Rating by Awards")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 26774 rows containing missing values (geom_point).
```



Hide

```
cor(newdf[,c(22,16,71)], use="complete.obs", method="pearson")
```

```
             tomatoRating imdbRating numAwards
tomatoRating    1.0000000  0.8023038 0.3251795
imdbRating      0.8023038  1.0000000 0.3042582
numAwards       0.3251795  0.3042582 1.0000000
```

Q: How good are these ratings in terms of predicting the success of a movie in winning awards or nominations? Is there a high correlation between two variables?

A: It can be seen that as the rating increases beyond 7, there are more movies that earn a large number of awards (larger than 50). However, in general the correlation is not that high and no clear trend is observed except for the aforementioned realtion. The following is the correlation map

```
         tomatoRating imdbRating numAwards
```

tomatoRating 1.0000000 0.8023038 0.3251795 imdbRating 0.8023038 1.0000000 0.3042582 numAwards 0.3251795 0.3042582 1.0000000

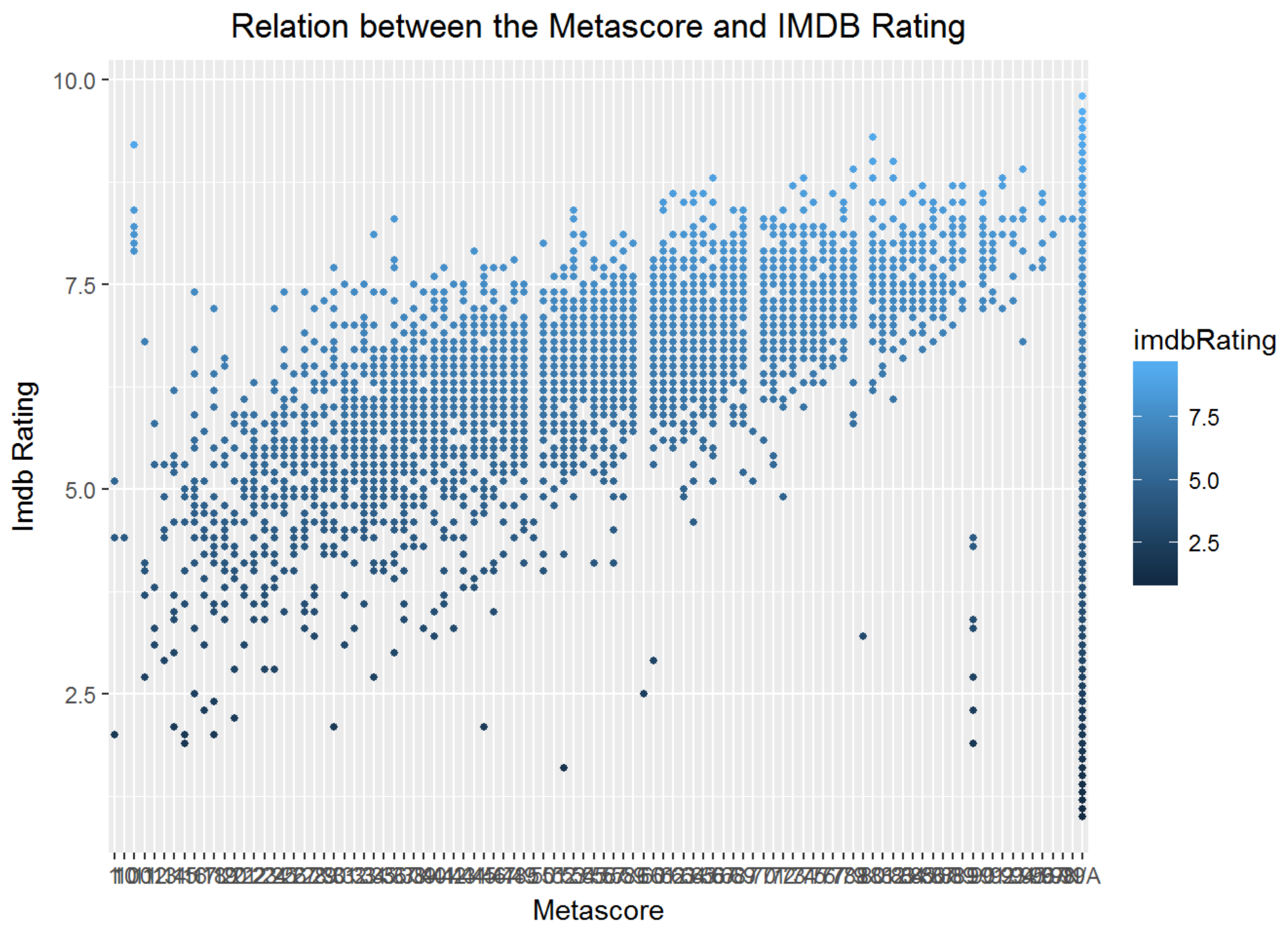
9. Expected insights

Come up with two new insights (backed up by data and graphs) that is expected. Here âœnewâ means insights that are not an immediate consequence of one of the above tasks. You may use any of the columns already explored above or a different one in the dataset, such as `Title` , `Actors` , etc.

Hide

```
# TODO: Find and illustrate two expected insights
ggplot(data=newdf ,mapping=aes(x=Metascore,y=imdbRating,color=imdbRating))+
  geom_point(size=1)+
  xlab("Metascore")+
  ylab("Imdb Rating")+
  ggtitle("Relation between the Metascore and IMDB Rating")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 1123 rows containing missing values (geom_point).
```



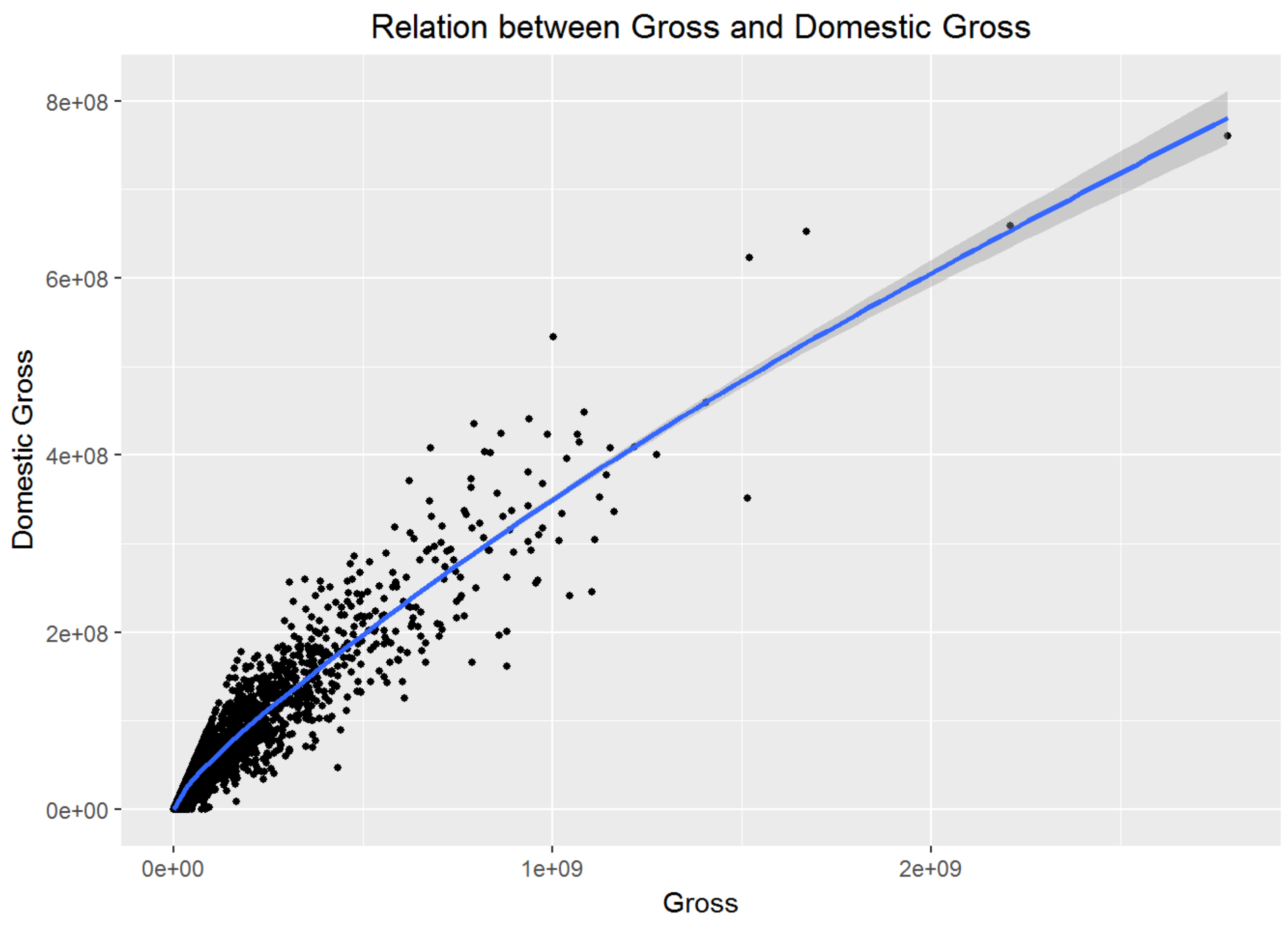
Hide

```
ggplot(data=newdf ,mapping=aes(x=Gross,y=Domestic_Gross))+
  geom_point(size=1)+
  geom_smooth()+
  xlab("Gross")+
  ylab("Domestic Gross")+
  ggtitle("Relation between Gross and Domestic Gross")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
`geom_smooth()` using method = 'gam'
```

```
Warning: Removed 30485 rows containing non-finite values (stat_smooth).
```

```
Warning: Removed 30485 rows containing missing values (geom_point).
```



Q: Expected insight #1.

A: We can see a linear relationship between the metascore and imdbRating as metascore increases, imdb increases as well. This is quite expected because the metascore already incorporates the imdbRating. Since metascore is a collection of ratings, we can also conclude that the imdbRating is a fair representation of the other scores or at least correlated with them.

Q: Expected insight #2.

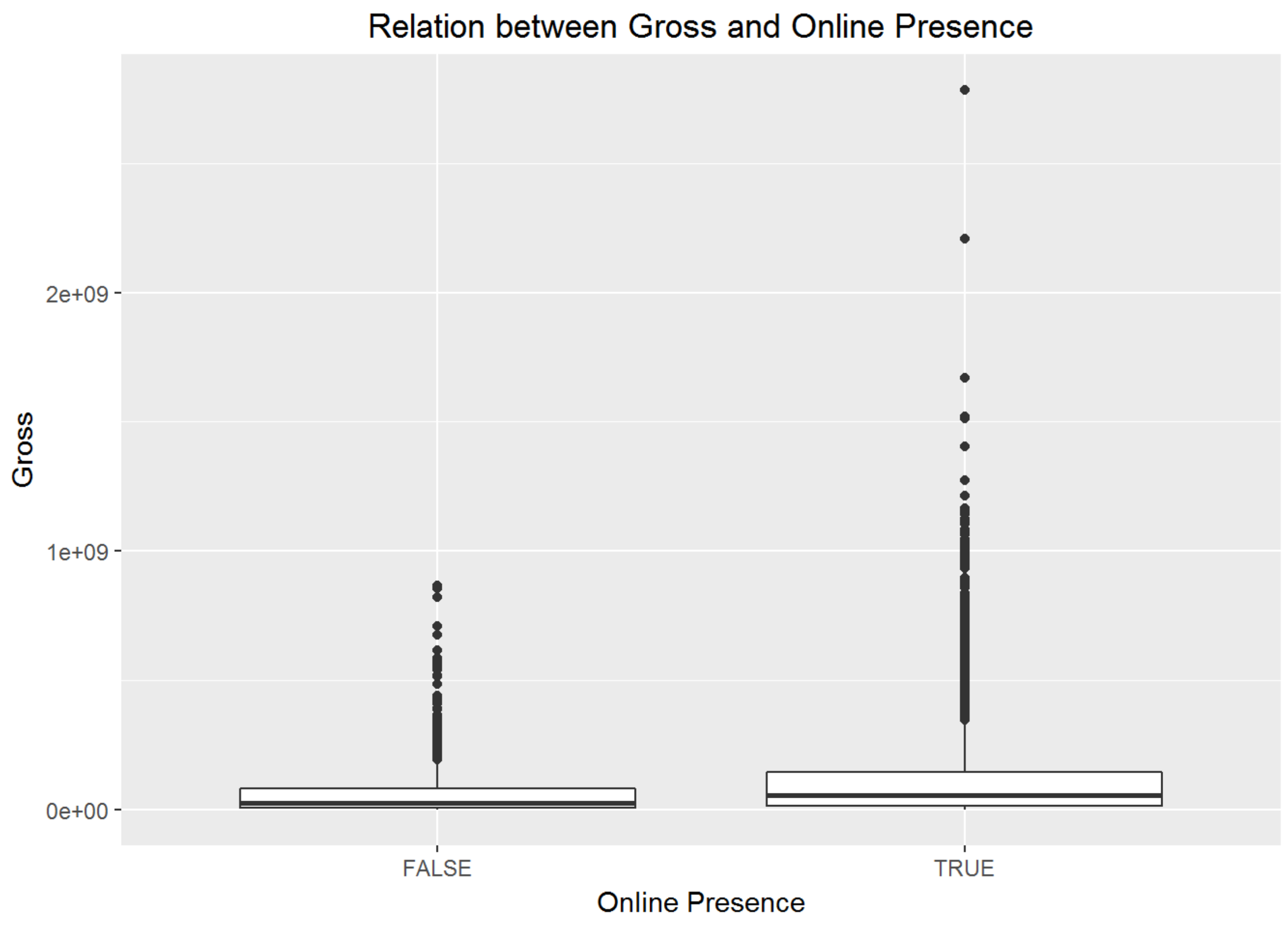
A: Another obvious insight was the straightforward relationship between the gross and domestic gross variables where they exhibit a linear relationship. Also, domestic gross makes a big portion of gross which is around 30% of Gross based on the best line fit.

10. Unexpected insight

Come up with one new insight (backed up by data and graphs) that is unexpected at first glance and do your best to motivate it. Same instructions apply as the previous task.

Hide

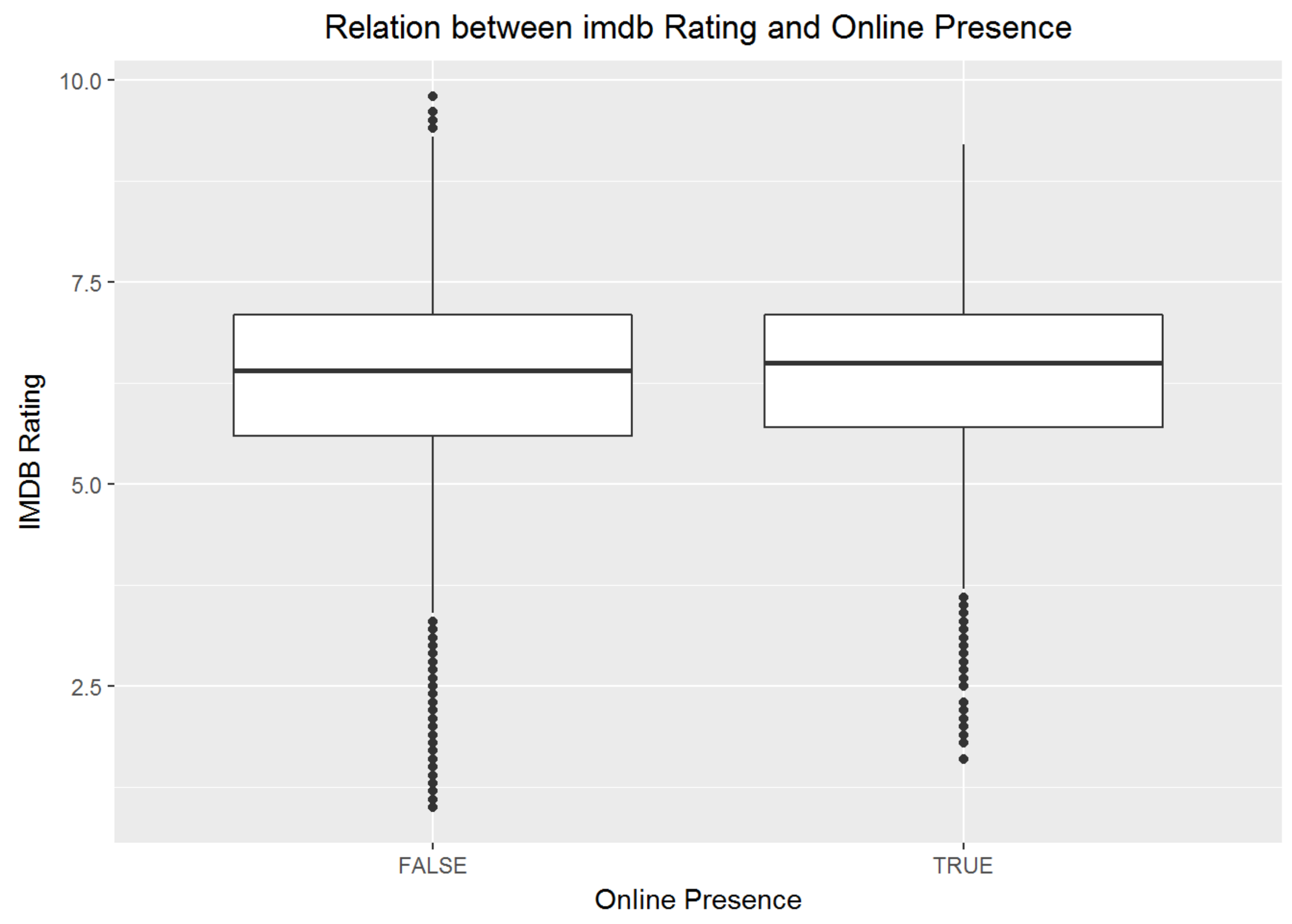
```
# TODO: Find and illustrate one unexpected insight
tempDF = newdf[newdf$Gross>0,]
tempDF = tempDF[!is.na(tempDF$Gross>0),]
ggplot(data=tempDF ,mapping=aes(x=Website!="N/A",y=Gross))+
  geom_boxplot()+
  xlab("Online Presence")+
  ylab("Gross")+
  ggtitle("Relation between Gross and Online Presence")+
  theme(plot.title = element_text(hjust = 0.5))
```



Hide

```
ggplot(data=newdf ,mapping=aes(x=Website!="N/A",y=imdbRating))+
  geom_boxplot()+
  xlab("Online Presence")+
  ylab("IMDB Rating")+
  ggtitle("Relation between imdb Rating and Online Presence")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 1123 rows containing non-finite values (stat_boxplot).
```



Q: Unexpected insight.

A: One interesting variable that was not explored in the assignment was the website field. It caught my attention since many notable movies use a website to promote their film beyond the conventional use of theatrical trailers. By comparing films that had a website with those that don't, it was found that those films with a website had a higher Gross median. Similarly, movies with online presence (website) had a slightly higher median IMDB rating.