Project 1: Explore and Prepare Data

CSE6242 - Data and Visual Analytics - Spring 2017 - Ebeid ElSayed - Ebeid@gatech.edu

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` 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 IMDb's API (see 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 IMDb'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 a second window streaming rights).

Instructions

This is an R Markdown Notebook. Open this file in RStudio to get started.

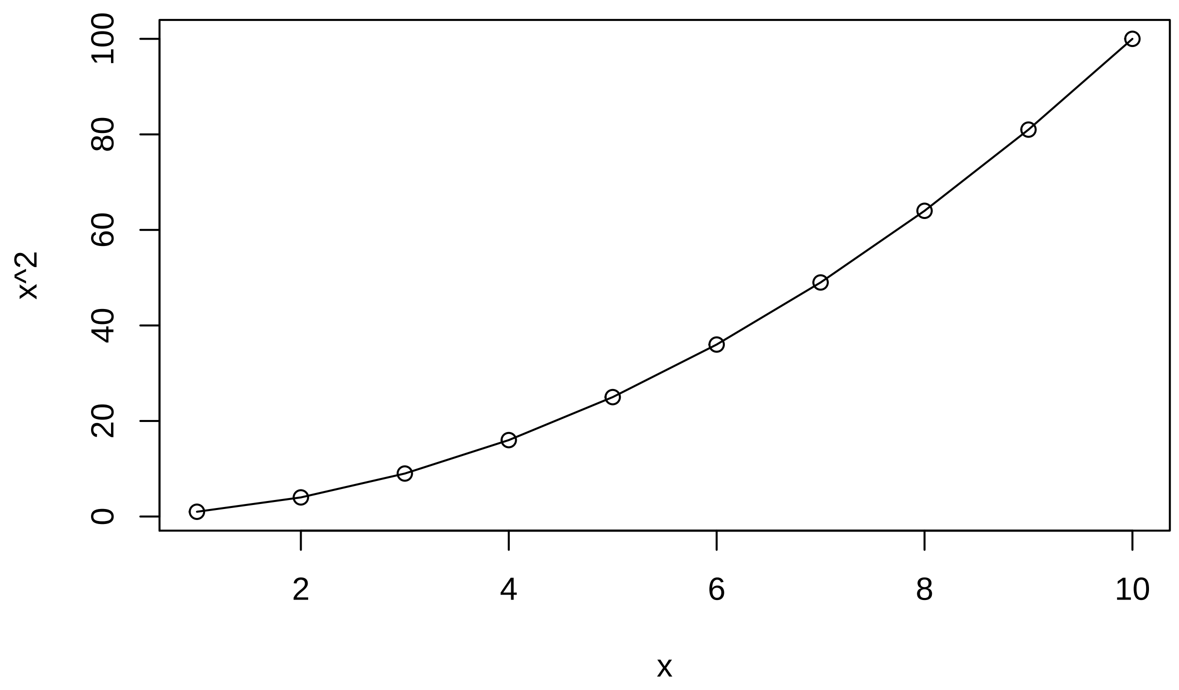
When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

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

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

Plots appear inline too:

```r
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` file and it is in the current working directory. Now load it into memory:

```
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:

```
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
```

```
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"
```

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.

```
library(ggplot2)
```

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

```
library(GGally)
```

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

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

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

```
library(SnowballC)
library(stringi)
library("tm")
library(stringr)
library(reshape2)
library(GGally)
```

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.

```r
# TODO: Remove all rows from df that do not correspond to movies
df = df[df$Type=="movie",]
nrow(df)
```

```
## [1] 40000
```

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

A: 4000

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.

```r
# TODO: Replace df$Runtime with a numeric column containing the runtime in minutes
RuntimeToMinutes = function(str){
  v = strsplit(str, " ")[[1]]
  res = 0
  if(length(v) < 2){
    print(v)
    return(0)
  }
  for (i in seq(2, length(v)+1, 2)) {
    if (v[[i]] == "min") res = res + as.integer(v[i-1])
    if (v[[i]] == "h")   res = res + as.integer(v[i-1]) * 60
  }
  return(res)
}
# Let's remove NAs
df = df[df$Runtime != "N/A" & !is.na(df$Runtime),]
# Convert Runtime variable to numeric
df$Runtime = sapply(df$Runtime, RuntimeToMinutes)
```

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.

Answer Part 1 - Runtime distribution

```r
# TODO: Investigate the distribution of Runtime values and how it varies by Year and Budget
summary(df$Runtime)
```

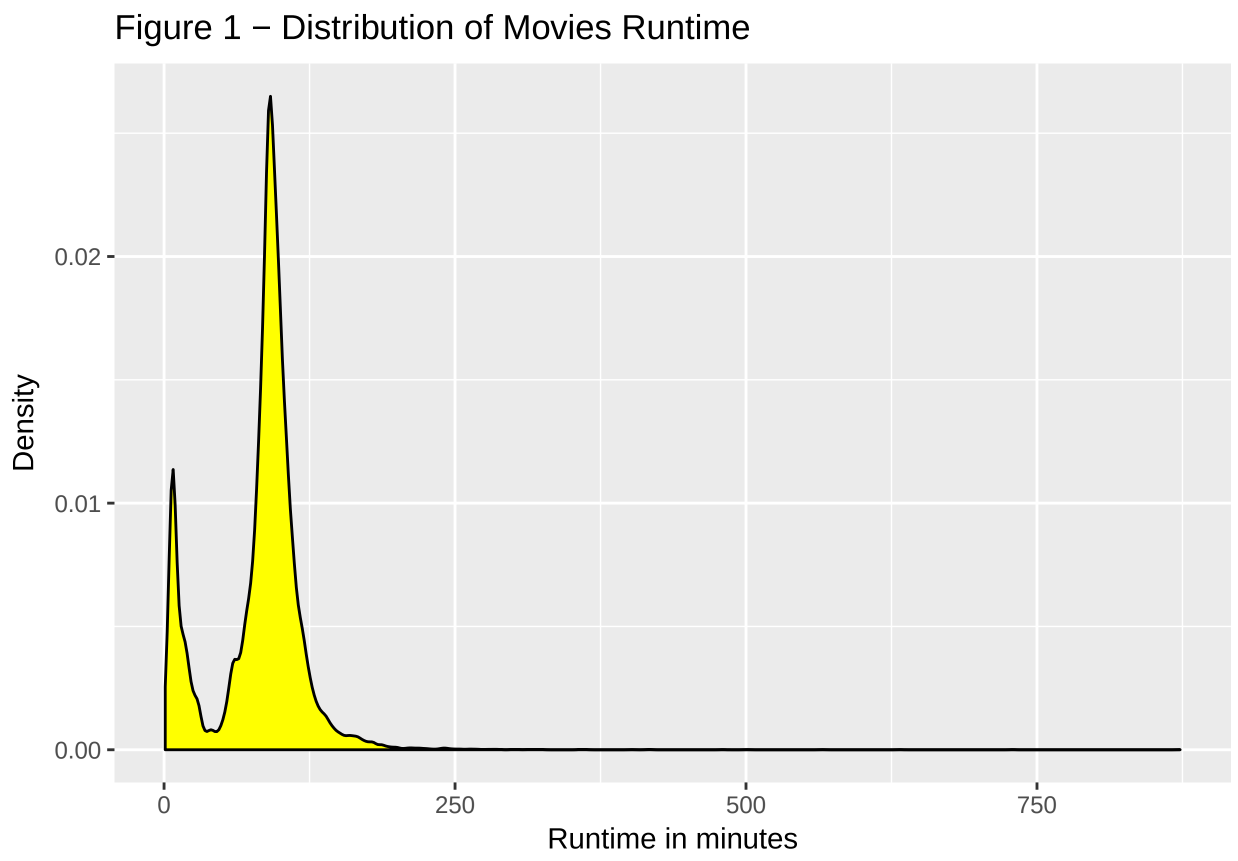
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   72.00   90.00   81.79  101.00  873.00
```

```
sd(df$Runtime)
```

```
## [1] 38.29093
```

Let's see the movies Runtime distribution

```
chart <- ggplot(df) + geom_density(aes(x=Runtime), fill='yellow') + xlab("Runtime in minutes") + ylab("
  ggtitle("Figure 1 - Distribution of Movies Runtime")
print(chart)
```



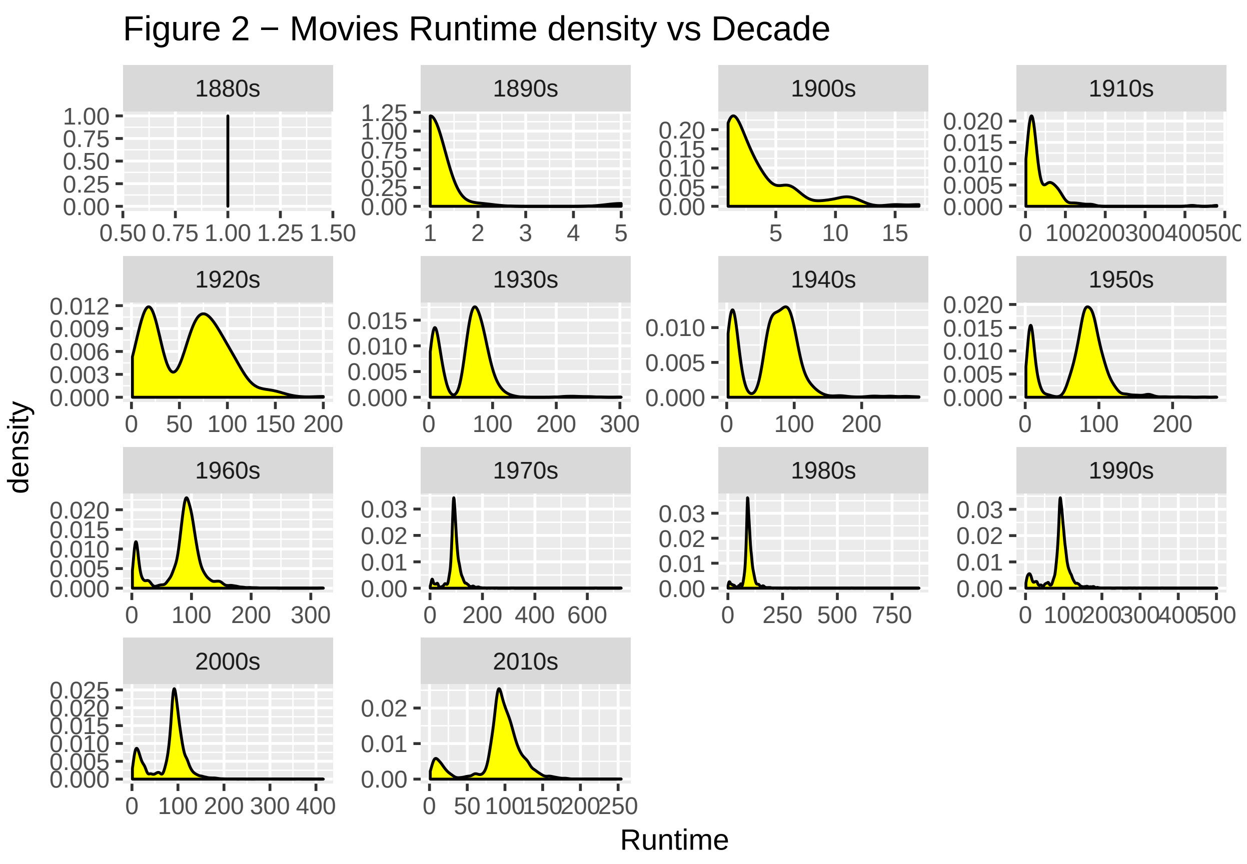
Answer Part 2 - Relationship between Runtime and Year

Let's first bucket movies into decades.

```
YearToDecade = function(year){
  if(year >= 1880 && year < 1890) return("1880s")
  if(year >= 1890 && year < 1900) return("1890s")
  if(year >= 1900 && year < 1910) return("1900s")
  if(year >= 1910 && year < 1920) return("1910s")
  if(year >= 1920 && year < 1930) return("1920s")
  if(year >= 1930 && year < 1940) return("1930s")
  if(year >= 1940 && year < 1950) return("1940s")
  if(year >= 1950 && year < 1960) return("1950s")
  if(year >= 1960 && year < 1970) return("1960s")
  if(year >= 1970 && year < 1980) return("1970s")
  if(year >= 1980 && year < 1990) return("1980s")
  if(year >= 1990 && year < 2000) return("1990s")
  if(year >= 2000 && year < 2010) return("2000s")
  if(year >= 2010) return("2010s")
  return("Other")
}

df$Decade = factor(sapply(df$Year, YearToDecade),                              levels=c("1880s","1890s"
```

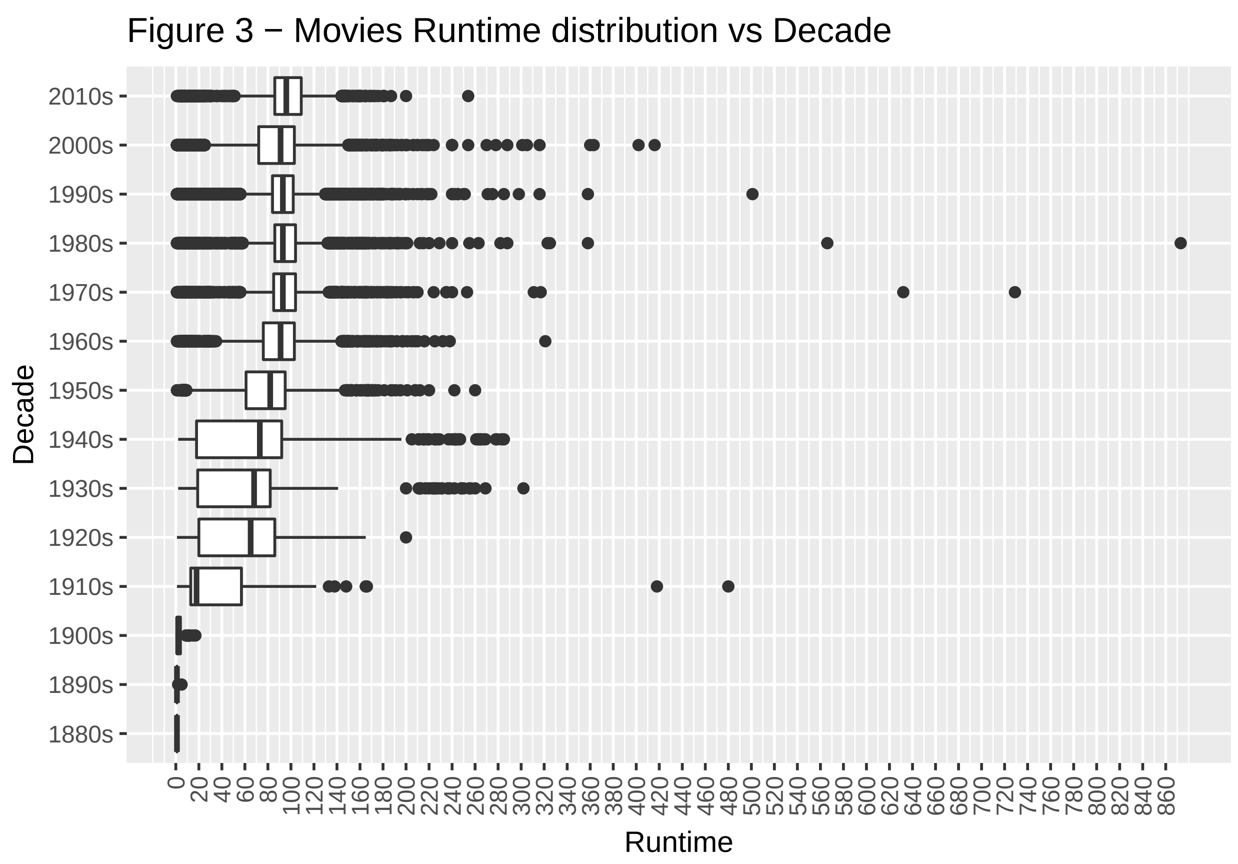
Now let's see how Runtime density look like over the decades.

```
chart <- ggplot(df) + geom_density(aes(x=Runtime), fill='yellow') + facet_wrap(~Decade,scales = "free")
print(chart)
```



Now let's see how Runtime distribution look like over the decades.

```
chart <- ggplot(df, aes(Decade,Runtime )) + geom_boxplot() + coord_flip() + scale_y_continuous(breaks =
print(chart)
```

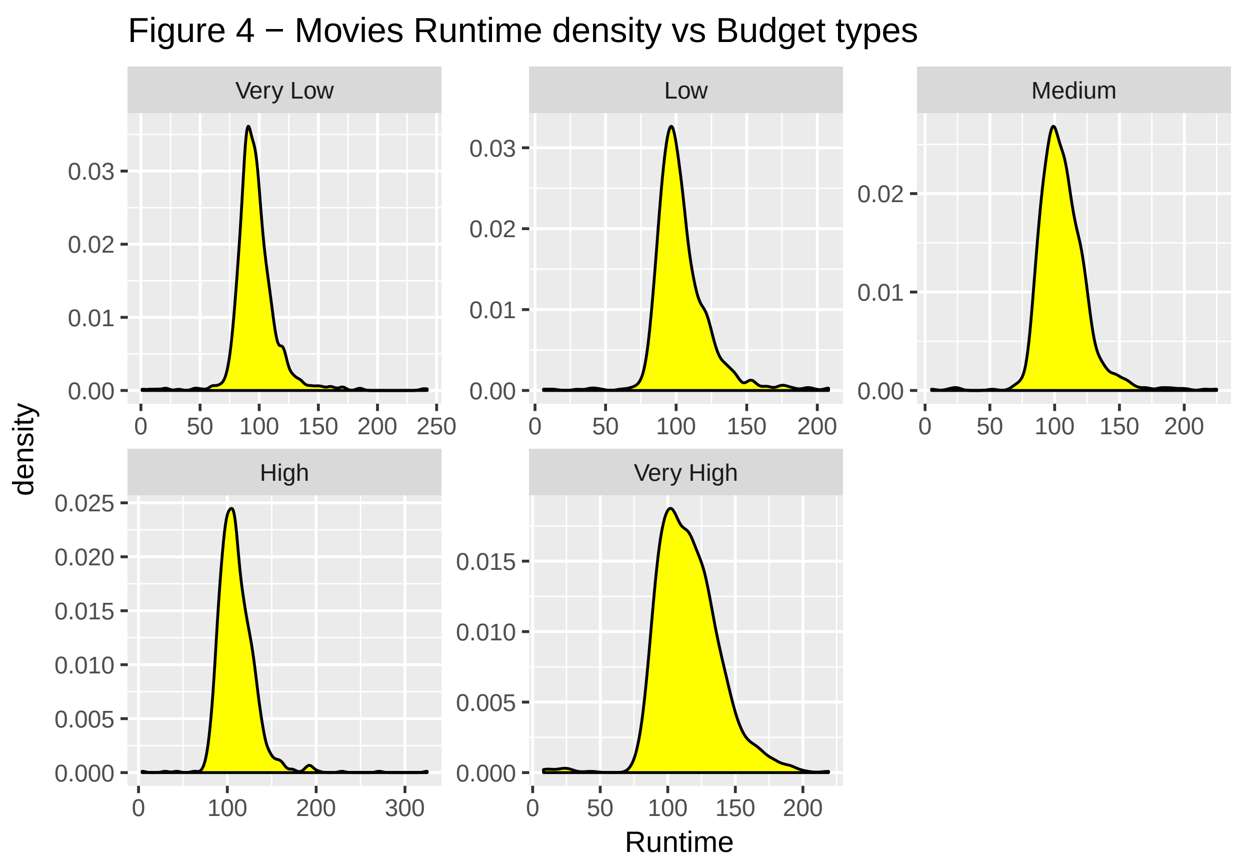


### Answer Part 3 - Relationship between Runtime and Budget Let's fisrt bucket Budget into buckets.

```
df = df[!is.na(df$Budget),]
ratios = quantile(df$Budget, c(.20, .40, .60, .80))
BudgetToType = function(budget){
  if(budget<ratios[[1]]) return("Very Low")
  if(budget<ratios[[2]]) return("Low")
  if(budget<ratios[[3]]) return("Medium")
  if(budget<ratios[[4]]) return("High")
  return("Very High")
}

df$BudgetType = factor(sapply(as.numeric(df$Budget), BudgetToType), levels = c("Very Low", "Low", "Medi
```

Now let's see how Runtime density look like over the Budget types.

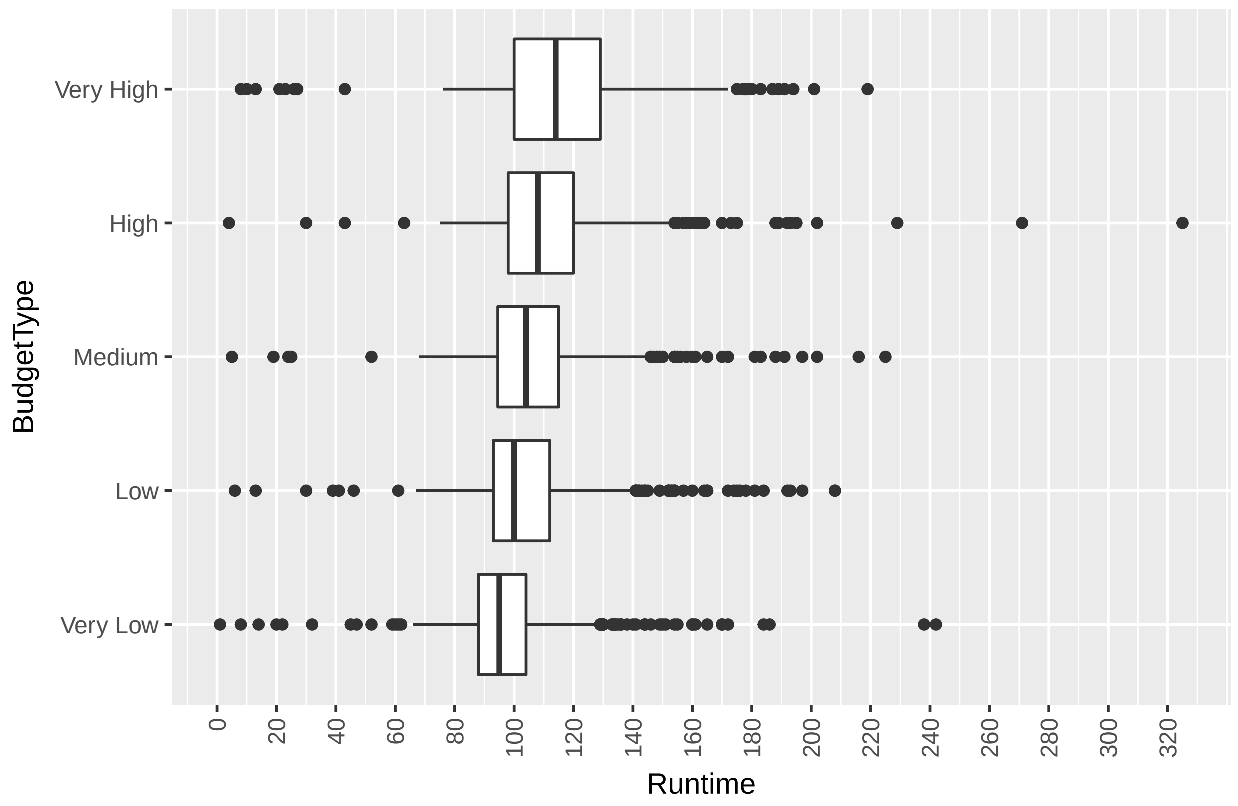
```
chart <- ggplot(df) + geom_density(aes(x=Runtime), fill='yellow') + facet_wrap(~BudgetType,scales = "fre
  ggtitle("Figure 4 - Movies Runtime density vs Budget types")
print(chart)
```



Now let's see how Runtime distribution look like over the Budget types

```
chart <- ggplot(df, aes(BudgetType,Runtime )) + geom_boxplot() + coord_flip() + scale_y_continuous(breal
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5)) + ggtitle("Figure 5 - Movies Runtime dist
print(chart)
```

Figure 5 − Movies Runtime distribution vs Budget types



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

A: We can see that the mean Runtime is 81.44 minutes with a standard deviation of 38.29 minutes. The median is 90 minutes, the minimum is 1 minute, and the maximum is 873 minutes. There are two main peaks in movies Runtime distribution: 15 minutes and 90 minutes, please refer to *Figure 1 - Distribution of Movies Runtime*. Most probably these peaks, 15 and 90 minutes , represent the most common length for short films and feature films respectively.

Regarding relationship between Runtime and Year: we grouped the movies into Decade buckets and then plotted movies Runtime density, *Figure 2 - Movies Runtime density vs Decade*, and distribution, *Figure 3 - Movies Runtime distribution vs Decade*, in each decade. Figures show that the median movie Runtime is increase over time with the biggest increase in the 1920s. It shows also that the short-feature films trend started in the 1920s. Since its start, the short-feature film trend lasted till 1950s in a more balanced shape. Since 1960s number of feature films started to increase significantly relative to number of short films. This increase in feature films continues till today (Maybe becuase feature films are more profitable).

From *Figure 4 - Movies Runtime density vs Budget types* : it seems to be the movies Runtime have a normal distribution with median that increases as the Budget increases. It make sense because the bigger budget you have, the longer movie you can produce. Although some movies have very high or very low budgets, the median Runtime generally falls between 90 and 120 minutes (Maybe because the movie Runtime is not the only factor that consumes the budget).

3. Encode `Genre` column

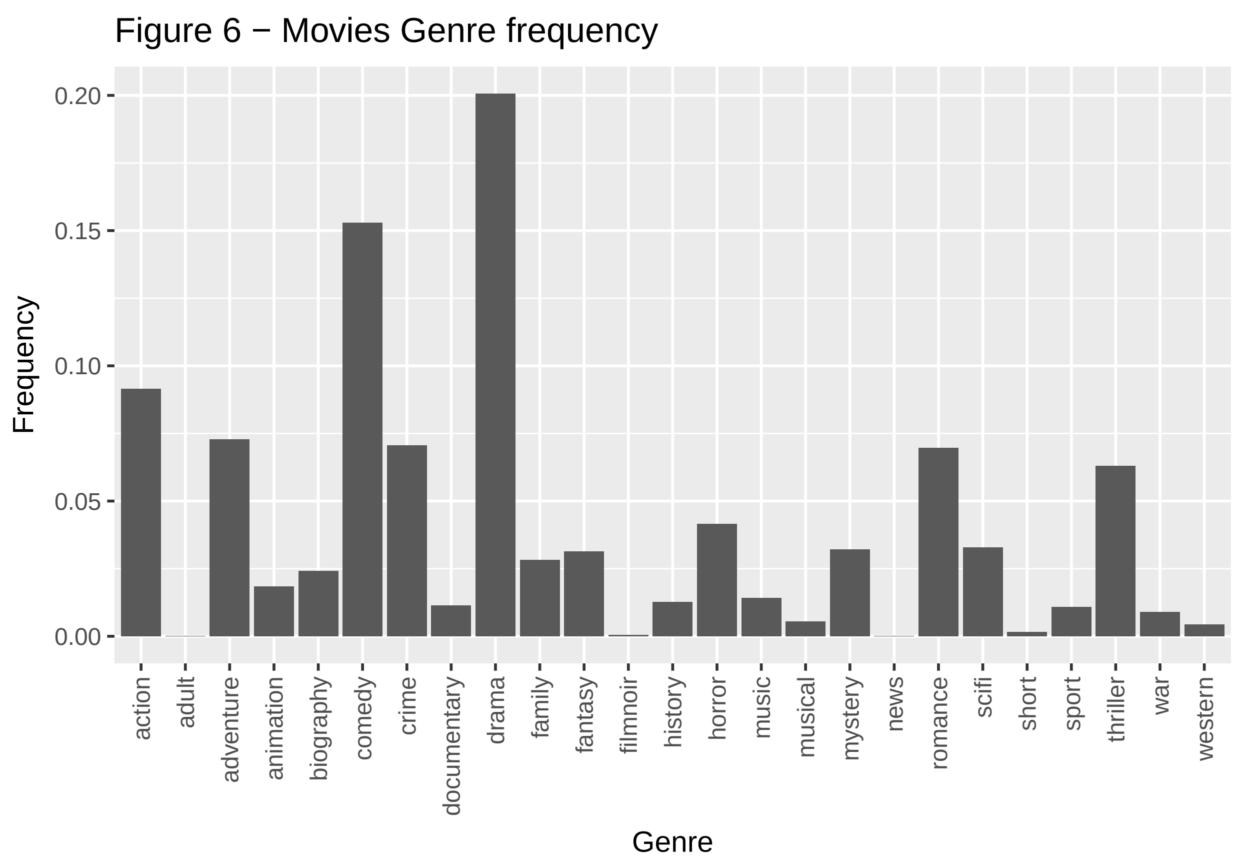
The column `Genre` represents a list of genres associated with the movie in a string format. Write code to parse each text string into a binary vector with 1s representing the presence of a genre and 0s the absence, and add it to the dataframe as additional columns. Then remove the original `Genre` column.

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).

```
# TODO: Replace Genre with a collection of binary columns
genreCorpus = VCorpus(VectorSource(df$Genre))
genreCorpus = tm_map(genreCorpus, content_transformer(tolower))
genreCorpus = tm_map(genreCorpus, removeWords, c("N/A"))
genreCorpus = tm_map(genreCorpus, removePunctuation)

genreDocumentTermMatrix = DocumentTermMatrix(genreCorpus)
genreMatrix = as.data.frame(as.matrix(genreDocumentTermMatrix))
df = merge(genreMatrix, df, by=0, all=TRUE)
```

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

```
# TODO: Select movies from top 10 most common genres and plot their relative proportions
genreFrequencies = colSums(as.matrix(genreDocumentTermMatrix))
genreFrequenciesOrdered = genreFrequencies[order(genreFrequencies,decreasing = TRUE)]

topGenres = genreFrequenciesOrdered[1:10]
topGenresName = names(topGenres)

genresDataFrame = data.frame(names(genreFrequenciesOrdered), genreFrequenciesOrdered/sum(genreFrequenci
names(genresDataFrame) = c("Genre","Frequency")

chart <- ggplot(genresDataFrame) + geom_bar(aes(x = Genre, y = Frequency), stat = "identity") + theme(a
print(chart)
```

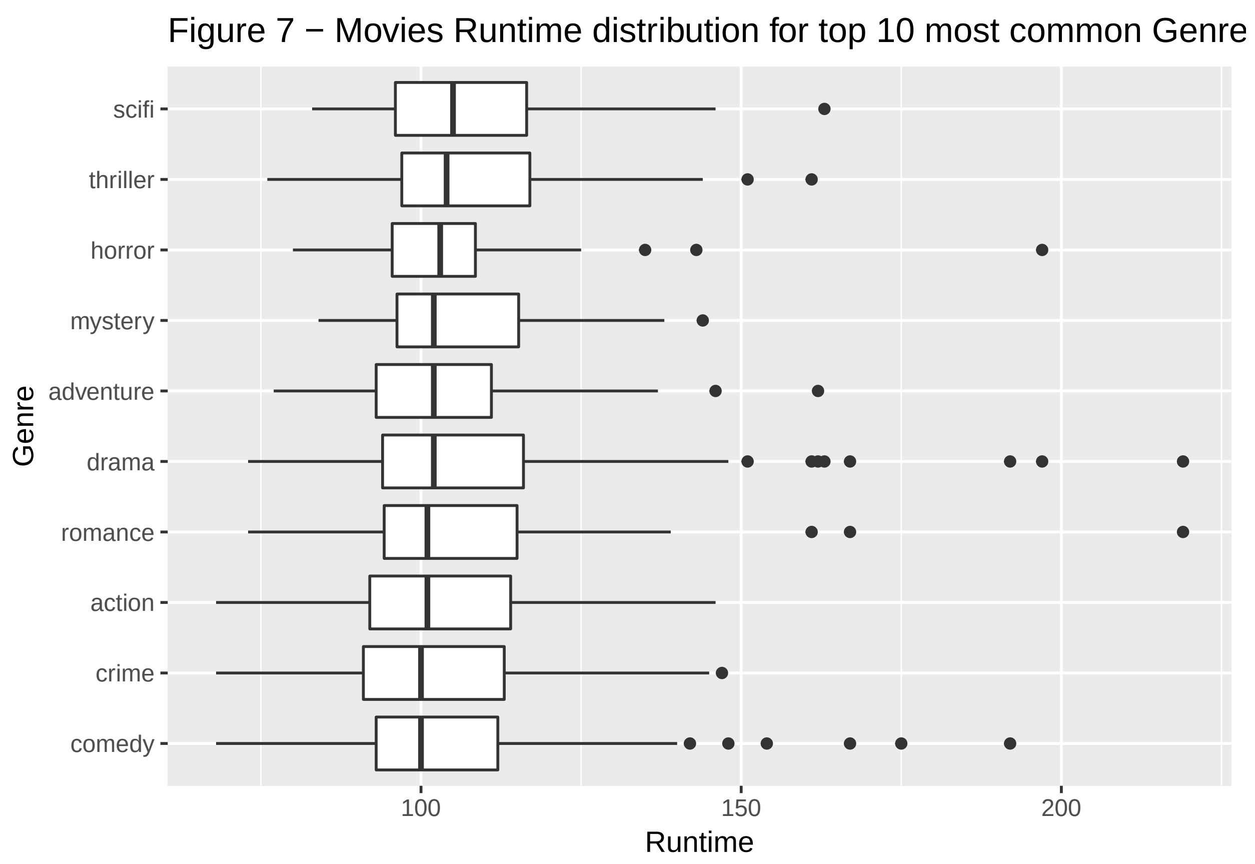


Examine how the distribution of `Runtime` changes across genres for the top 10 most common genres.

```
# TODO: Plot Runtime distribution for top 10 most common genres

# Let's prepare our data
meltedTopGenresRuntime = melt(df, id.vars = c("Title","Runtime"), measure.vars = c(topGenresName))
meltedTopGenresRuntime = meltedTopGenresRuntime[meltedTopGenresRuntime$value!=0,]
meltedTopGenresRuntime = meltedTopGenresRuntime[!is.na(meltedTopGenresRuntime$Runtime),]
```

Now we can examine the Runtime distribution for top 10 most common Genres

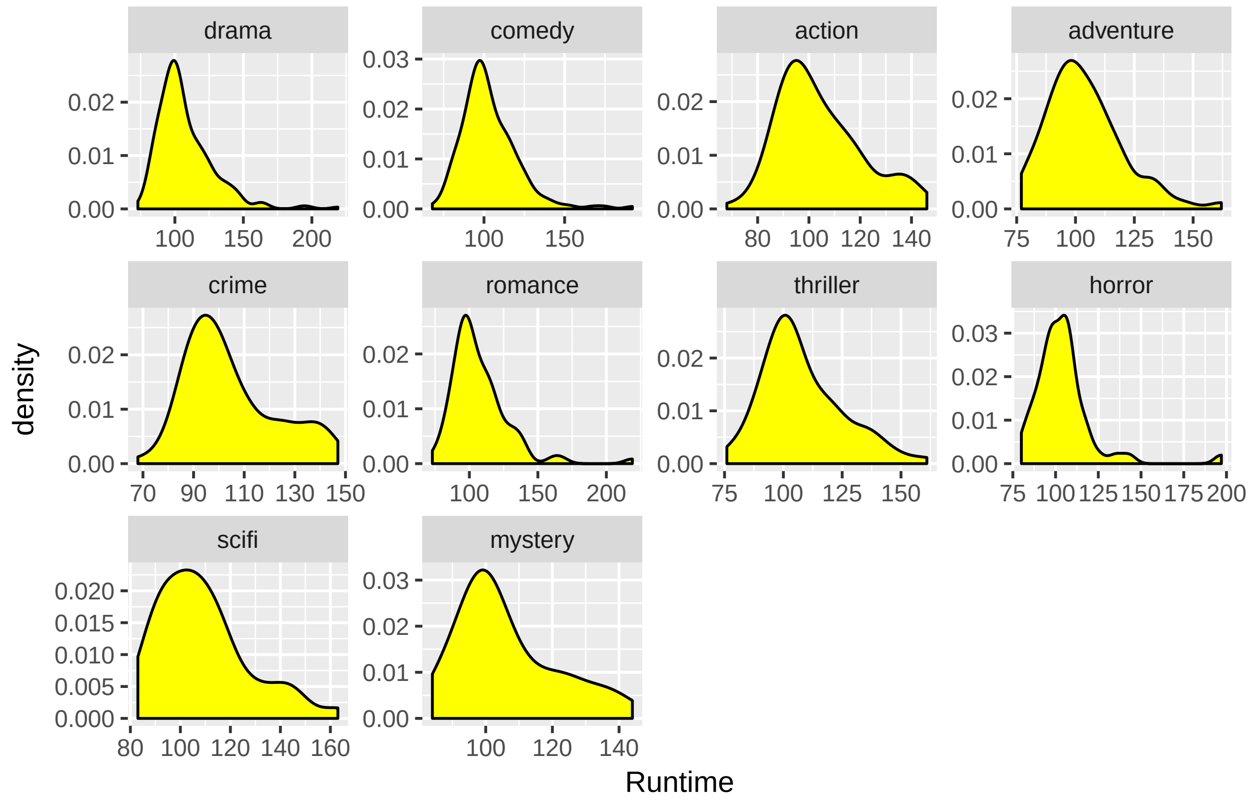
```
chart <- ggplot(meltedTopGenresRuntime, aes(reorder(variable, Runtime, median), Runtime)) + geom_boxplo
  scale_y_continuous(labels = scales::comma) + xlab("Genre") + ggtitle("Figure 7 - Movies Runtime distr
print(chart)
```



And also the Runtime density

```
chart <- ggplot(meltedTopGenresRuntime) + geom_density(aes(x=Runtime), fill='yellow') + facet_wrap(~var
  ggtitle("Figure 8 - Movies Runtime density for top 10 most common Genres")
print(chart)
```

Figure 8 − Movies Runtime density for top 10 most common Genres



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

A: We can notice from *Figure 7 - Movies Runtime distribution for top 10 most common Genres* that top 10 most common movie genres have a median close to 100 minutes, with comedy have the lowest median and sci-fi have the highest median. We can also notice from *Figure 8 - Movies Runtime density for top 10 most common Genres* that all top 10 most common genres have one clear peak which could be helpful if predictions need to be make in the future.

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 source is 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.

```r
# TODO: Remove rows with Released-Year mismatch
df$YearReleased <- as.numeric(format(as.Date(df$Released, format = "%Y-%m-%d"), "%Y"))
temp <- df
df <- df[
  (df$Year != df$YearReleased) |
    (is.na(df$YearReleased) & !is.na(df$Year)) |
    (!is.na(df$YearReleased) & is.na(df$Year)),]
```

The number of removed rows is

```
print(nrow(temp)-nrow(df))
```

```
## [1] 3718
```

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

A: In order to find Released-Year mismatch, we need to parse Released column and extract the year part of it and assign it to a new column YearReleased. Then we remove the rows that YearReleased don't equal Year OR one and only one of the two columns (Year, YearReleased) is NA. We removed 3718 rows

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.

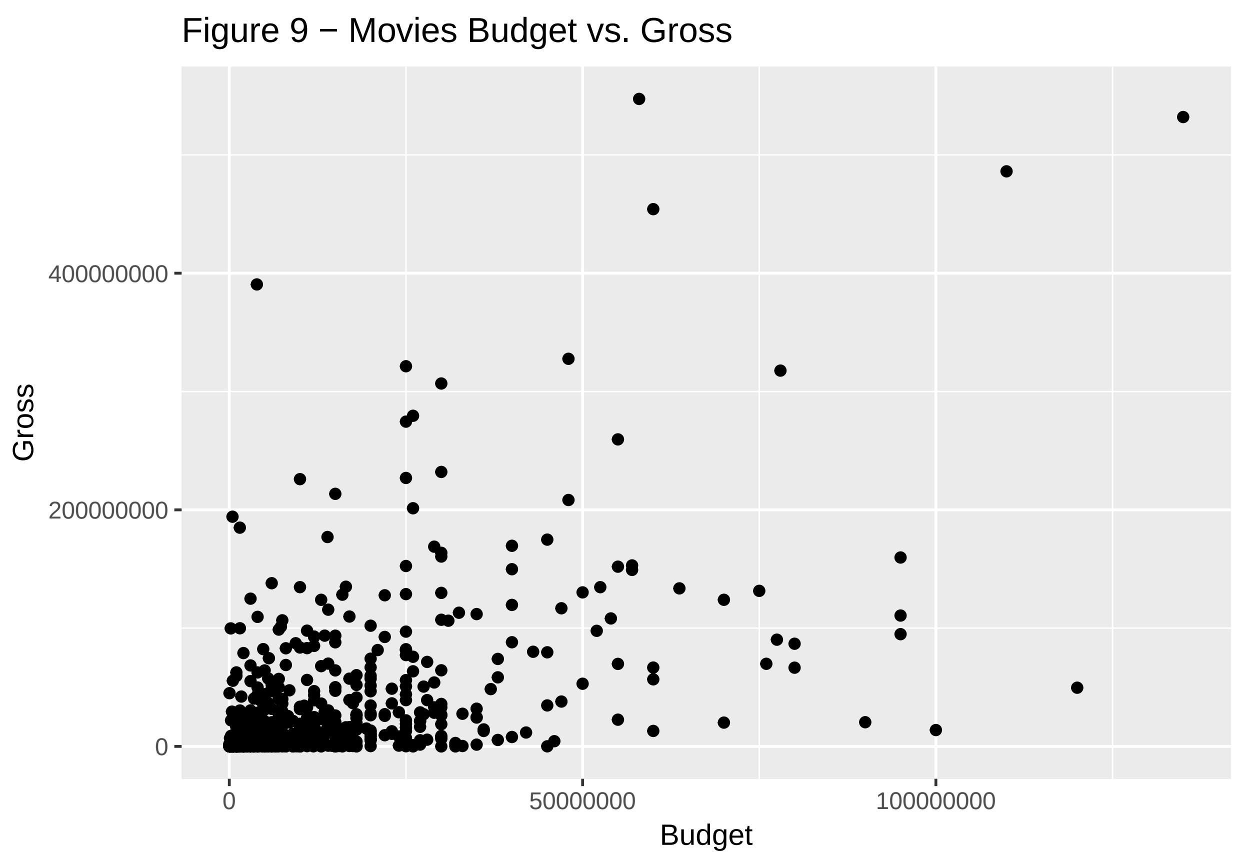
Note: To get a meaningful relationship, you may have to partition the movies into subsets such as short vs. long duration, or by genre, etc.

```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
options(scipen=999)
```

Answer Part 1 - Gross Revenue vs Budget

```
chart <- ggplot(df, aes(Budget, Gross)) + geom_point() + ggtitle("Figure 9 - Movies Budget vs. Gross")
print(chart)
```

```
## Warning: Removed 4027 rows containing missing values (geom_point).
```

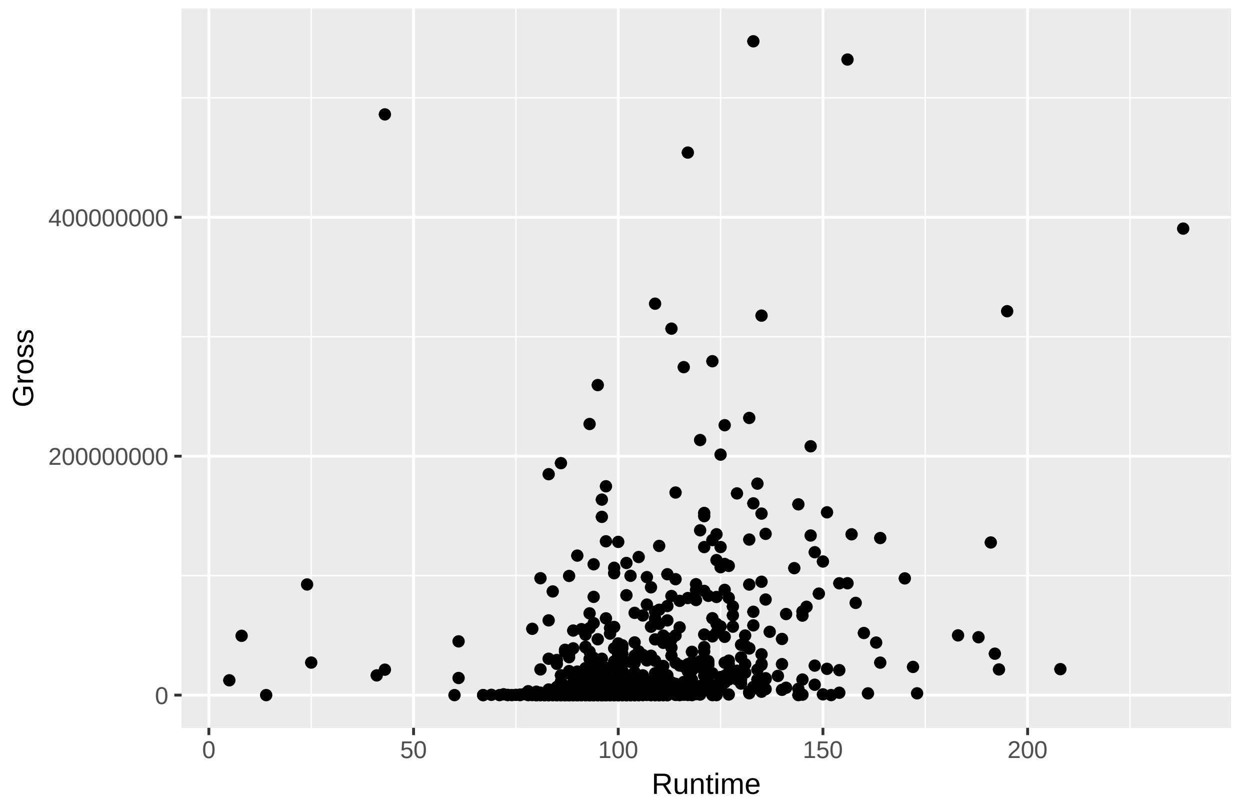


### Answer Part 2 - Gross Revenue vs Runtime

```
chart <- ggplot(df, aes(Runtime, Gross)) + geom_point() + ggtitle("Figure 10 - Movies Runtime vs. Gross
print(chart)
```

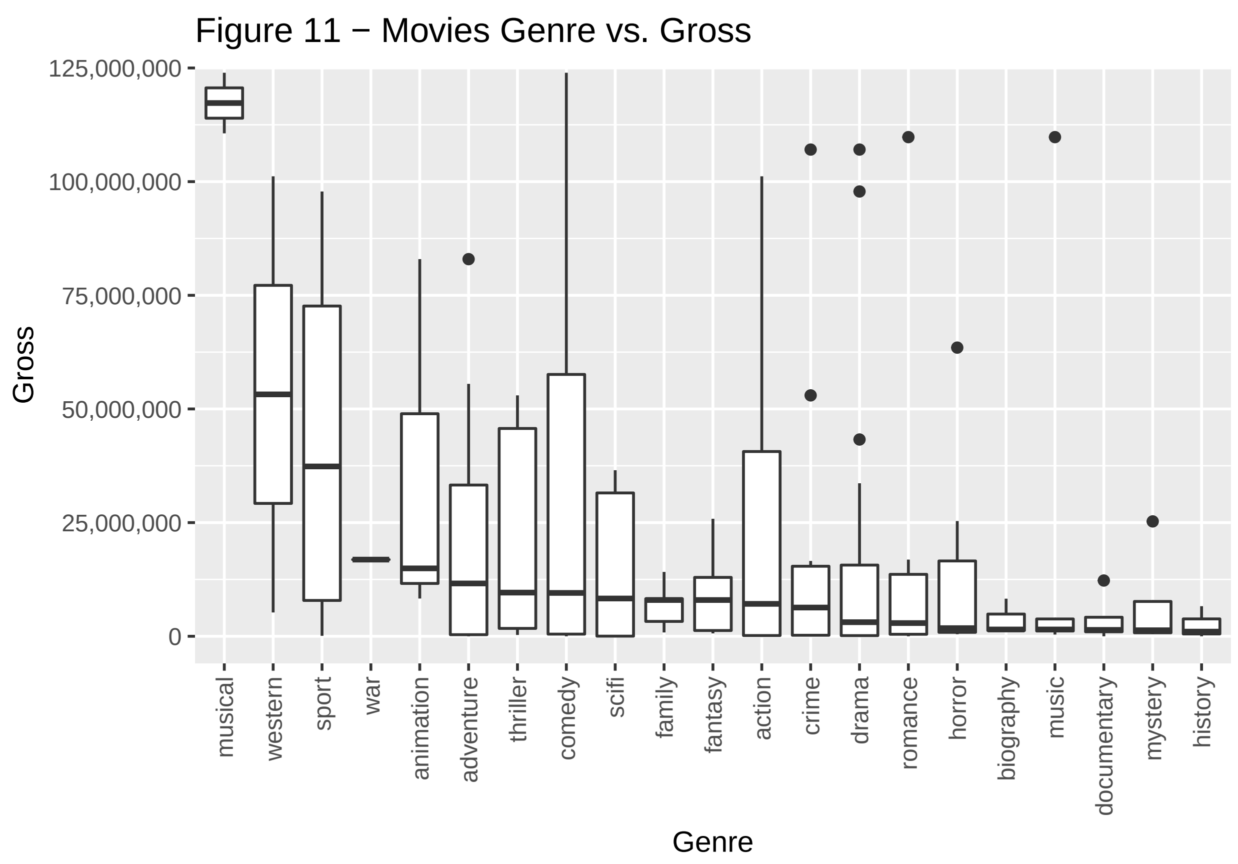
```
## Warning: Removed 4027 rows containing missing values (geom_point).
```

Figure 10 − Movies Runtime vs. Gross



### Answer Part 3 - Gross Revenue vs Genre

```
# Let's do so preparations first
meltedTopGenresGross = melt(df, id.vars = c("Title","Gross"), measure.vars = c(names(genreFrequencies))
meltedTopGenresGross = meltedTopGenresGross[meltedTopGenresGross$value!=0,]
meltedTopGenresGross = meltedTopGenresGross[!is.na(meltedTopGenresGross$Gross),]

# let's use this gross distribution to do a visual zoom on our blot to focus on non-outliers
zeroToFifthQ = boxplot.stats(meltedTopGenresGross$Gross)$stats[c(1, 5)]
chart <- ggplot(meltedTopGenresGross, aes(reorder(variable, -Gross,median), Gross)) +
  geom_boxplot() +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5)) +
  xlab("Genre") +
  coord_flip() +
  coord_cartesian(ylim = zeroToFifthQ*2) +
  scale_y_continuous(labels = scales::comma) +
  ggtitle("Figure 11 - Movies Genre vs. Gross")
print(chart)
```

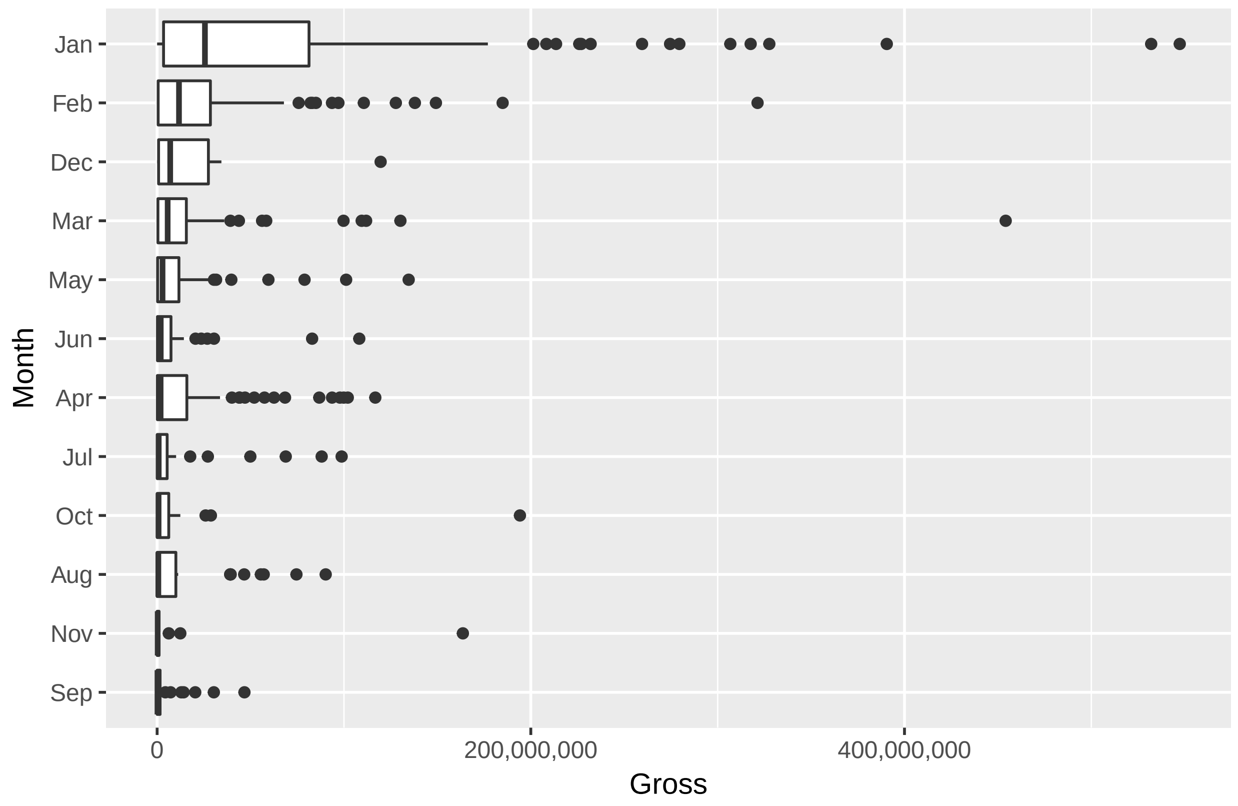


Q: Did you find any observable relationships or combinations of Budget/Runtime/Genre that result in high Gross revenue? If you divided the movies into different subsets, you may get different answers for them - point out interesting ones.

A: It seems from *Figure 9 - Movies Budget vs. Gross* and *Figure 10 - Movies Runtime vs. Gross* that there is no direct clear relationship between Budget and Gross nor Runtim and Gross. Most movies' budget falls below 50000000 and Gross below 200000000. Most movies' runtime falls between 75 and 150 minutes. Within this budget range and runtime range, there is no clear correlation between Budget and Gross nor Runtim and Gross. It seems from *Figure 11 - Movies Genre vs. Gross* that (musical, western, sport) genres have the highest median gross; on the other hand (documentry, mystery, history) have the lowest gross

```
# TODO: Investigate if Gross Revenue is related to Release Month
df$Month = as.numeric(format(as.Date(df$Released, format = "%Y-%m-%d"), "%m"))

GetMonthName = function(num){
  switch(as.numeric(num),"Jan","Feb","Mar","Apr","May","Jun", "Jul","Aug","Sep","Oct","Nov","Dec")
}

df$Month = factor(
  lapply(as.numeric(df$Month, units="months"), GetMonthName),
  levels=c("Jan","Feb","Mar","Apr","May","Jun", "Jul","Aug","Sep","Oct","Nov","Dec"))

cleanDF = df[!is.na(df$Month),]
chart <- ggplot(cleanDF, aes(reorder(Month, Gross, median), Gross)) + geom_boxplot() + coord_flip() +
  scale_y_continuous(labels = scales::comma) + xlab("Month") + ggtitle("Figure 10 - Movies Gross distri
print(chart)
```

Figure 10 − Movies Gross distribution vs. release Month



We see from *Figure 10 - Movies Gross distribution vs. release Month* that movies Gross is related to release Month with (Jan, Feb, Dec) represent the top months in median Gross and (Aug, Nov, Sep) are the lowest in median Gross.

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.

```
# TODO: Convert Awards to 2 numeric columns: wins and nominations
# Let's process Awards column
df$AwardsNum <- sapply(df$Awards,
                       function(awards) as.numeric( unlist(
                         regmatches(awards, gregexpr("+[0-9]+", awards))[[1]]
                         ) ) )
df$Wins = 0
df$Nominations = 0
rowsWithInvalidAwards <- 0
for(i in 1:nrow(df)){
  #print(paste(i, df[i,]$Awards, sep = " == > "))
  # 1 number found : find out if it is win or nomination
  if( (length(df[i,]$AwardsNum[[1]])!=3) & (length(df[i,]$AwardsNum[[1]])!=2) & (length(df[i,]$AwardsNu
    if(length(grep("win",df[i,]$Awards))>0){
      #print(paste(as.character(df[i,]$AwardsNum[[1]][1]) , "Win", sep=" "))
      df[i,]$Wins = as.numeric(df[i,]$AwardsNum[[1]][1])
      }
    if(length(grep("nomination",df[i,]$Awards))>0){
      #print(paste(as.character(df[i,]$AwardsNum[[1]][1]) , "nomination", sep=" "))
      df[i,]$Nominations = as.numeric(df[i,]$AwardsNum[[1]][1])
      }
  }
  # 2 numbers found : most probably the first is win and the second is nomination
  if(length(df[i,]$AwardsNum[[1]])==2){
    if((length(grep("win",df[i,]$Awards))>0) || (length(grep("won",df[i,]$Awards))>0)|| (length(grep("W
      #print(paste(as.character(df[i,]$AwardsNum[[1]][1]) , "Win", sep=" "))
      df[i,]$Wins = as.numeric(df[i,]$AwardsNum[[1]][1])
    }
    if(length(grep("nomination",df[i,]$Awards))>0){
      #print(paste(as.character(df[i,]$AwardsNum[[1]][2]) , "nomination", sep=" "))
      df[i,]$Nominations = as.numeric(df[i,]$AwardsNum[[1]][2])
    }
  }
  # 3 numbers found : most probably the first is total or another award, the second is win, and the thi
  if(length(df[i,]$AwardsNum[[1]])==3){
    if((length(grep("win",df[i,]$Awards))>0) || (length(grep("won",df[i,]$Awards))>0)|| (length(grep("W
      #print(paste(as.character(df[i,]$AwardsNum[[1]][2]) , "Win", sep=" "))
      df[i,]$Wins = as.numeric(df[i,]$AwardsNum[[1]][2])
    }
    if(length(grep("nomination",df[i,]$Awards))>0){
      #print(paste(as.character(df[i,]$AwardsNum[[1]][3]) , "nomination", sep=" "))
      df[i,]$Nominations = as.numeric(df[i,]$AwardsNum[[1]][3])
    }
  }
  # no numbers found : set both win and nomination to 0, increment the rowsWithInvalidAwards counter
  if(length(df[i,]$AwardsNum[[1]])==0){
    df[i,]$Wins = 0
    df[i,]$Nominations = 0
    rowsWithInvalidAwards = rowsWithInvalidAwards + 1
  }

  if(length(unlist(df[i,]$Wins))==1)
    df[i,]$Wins = as.numeric(unlist(df[i,]$Wins))
  else
    print(paste(df[i,]$Awards, df[i,]$Wins, sep = " ==> " ))
  if(length(unlist(df[i,]$Nominations))==1)
    df[i,]$Nominations = as.numeric(unlist(df[i,]$Nominations)[[1]])
  else
    print(paste(df[i,]$Nominations, df[i,]$Nominations, sep = " ==> " ))
}
```

Number of rows that have non-zero wins nominations is 675

```
print(nrow(df) - rowsWithInvalidAwards)
```

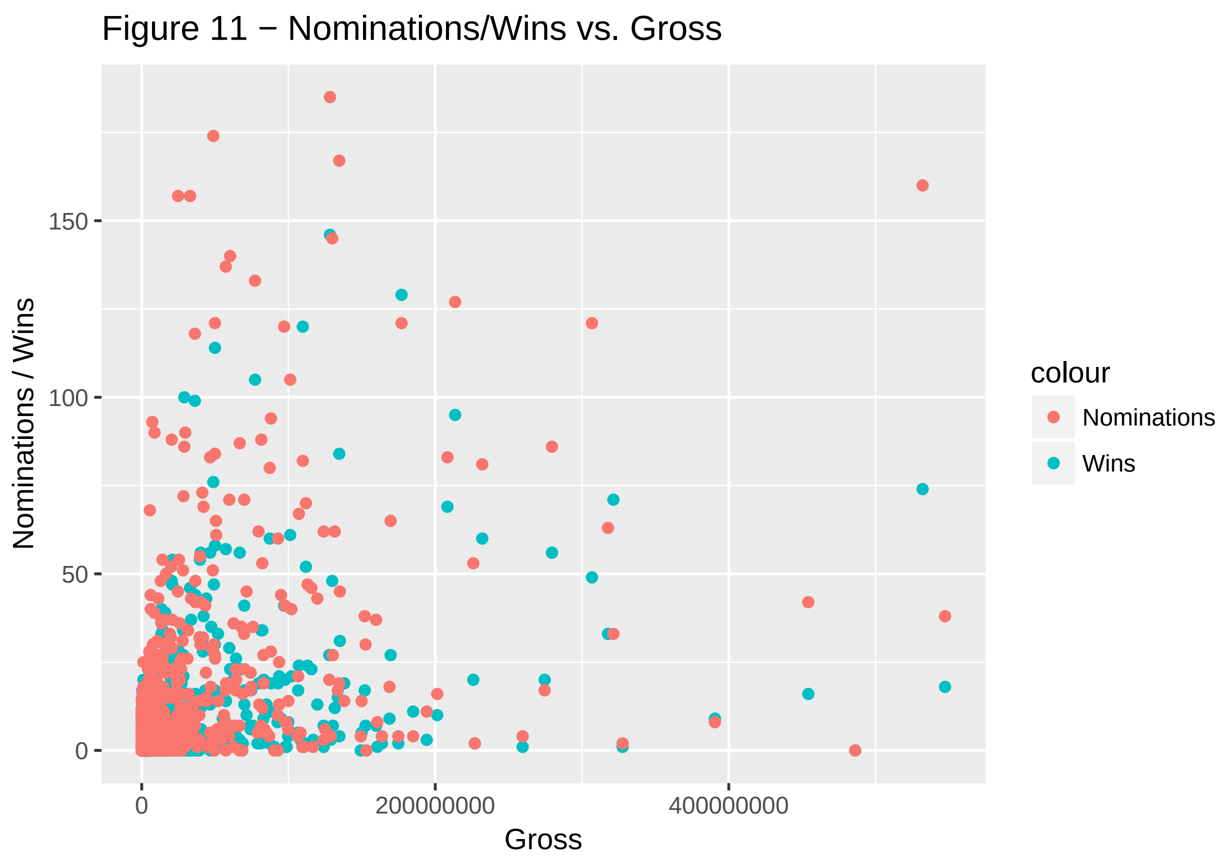
```
## [1] 675
```

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

A: First I counted how many numbers exist in the Awards column using regular expressions. Based on number of regular expression matches (each match represent one number within the Awards column) we decide the value for Wins and Nominations column. If there is one number: use regular expression to decide if this number is for Wins or Nominations. If there are two numbers: use regular expressions to make sure the first one goes to Wins if any of the following exist("Win", "win", "won"), use regular expression to make sure the second one goes to Nominations if ("nomination") exists. If there are three numbers: use regular expressions to make sure the second one goes to Wins if any of the following exist("Win", "win", "won"), use regular expression to make sure the third one goes to Nominations if ("nomination") exists. If there are no numbers: count this row as Invalid row. This counter will be used to answer the second part of this question. I found 674 rows with valid non-zero wins or nominations.

```r
# TODO: Plot Gross revenue against wins and nominations
ggplot(df, aes(Gross)) +
  geom_point(data=df, aes(y=Wins, color="Wins")) +
  geom_point(data=df, aes(y=Nominations, color="Nominations")) +
  labs(title="Figure 11 - Nominations/Wins vs. Gross",
       x="Gross",
       y="Nominations / Wins")
```

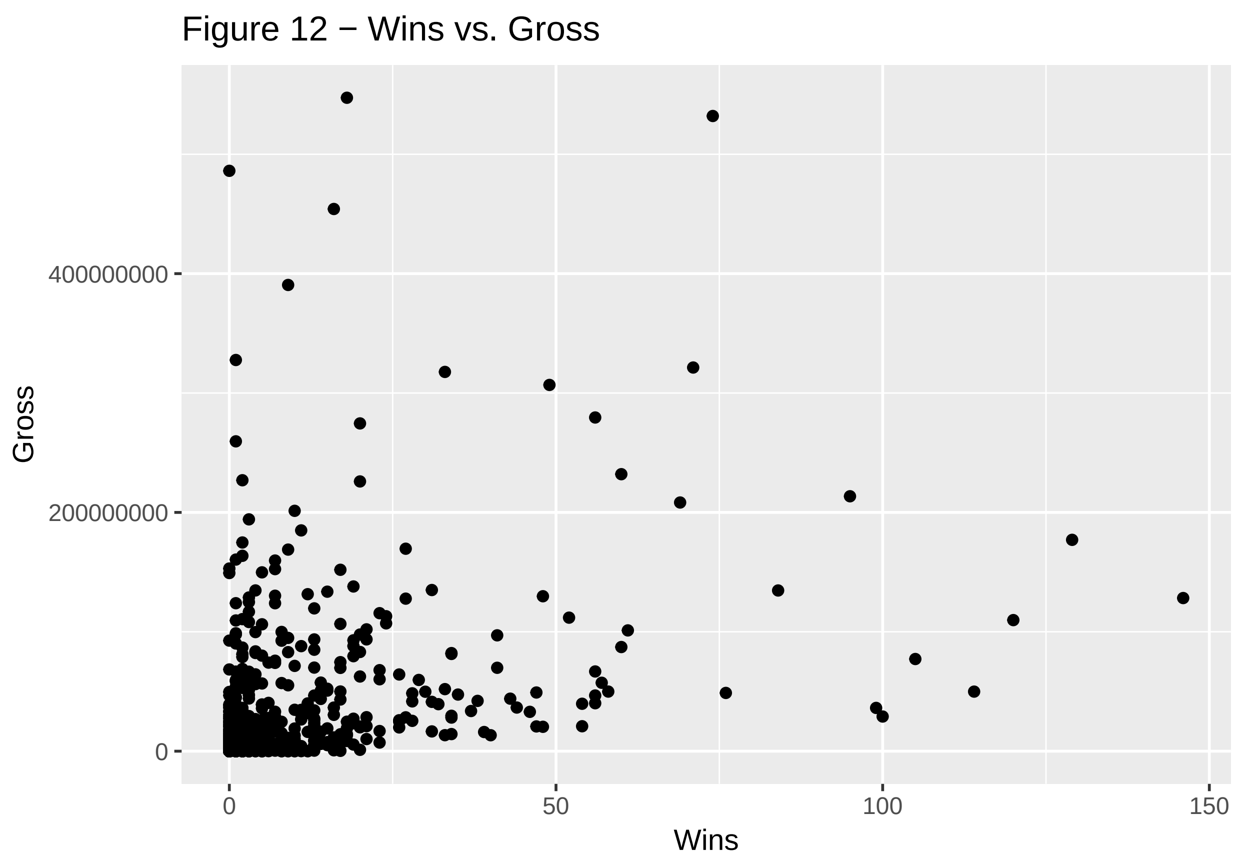
```
## Warning: Removed 4027 rows containing missing values (geom_point).

## Warning: Removed 4027 rows containing missing values (geom_point).
```



Let's see Wins alone against Gross

```r
ggplot(df) +
  geom_point(data=df, aes(x=Wins, y=Gross)) +
  labs(title="Figure 12 - Wins vs. Gross",
       x="Wins",
       y="Gross")
```

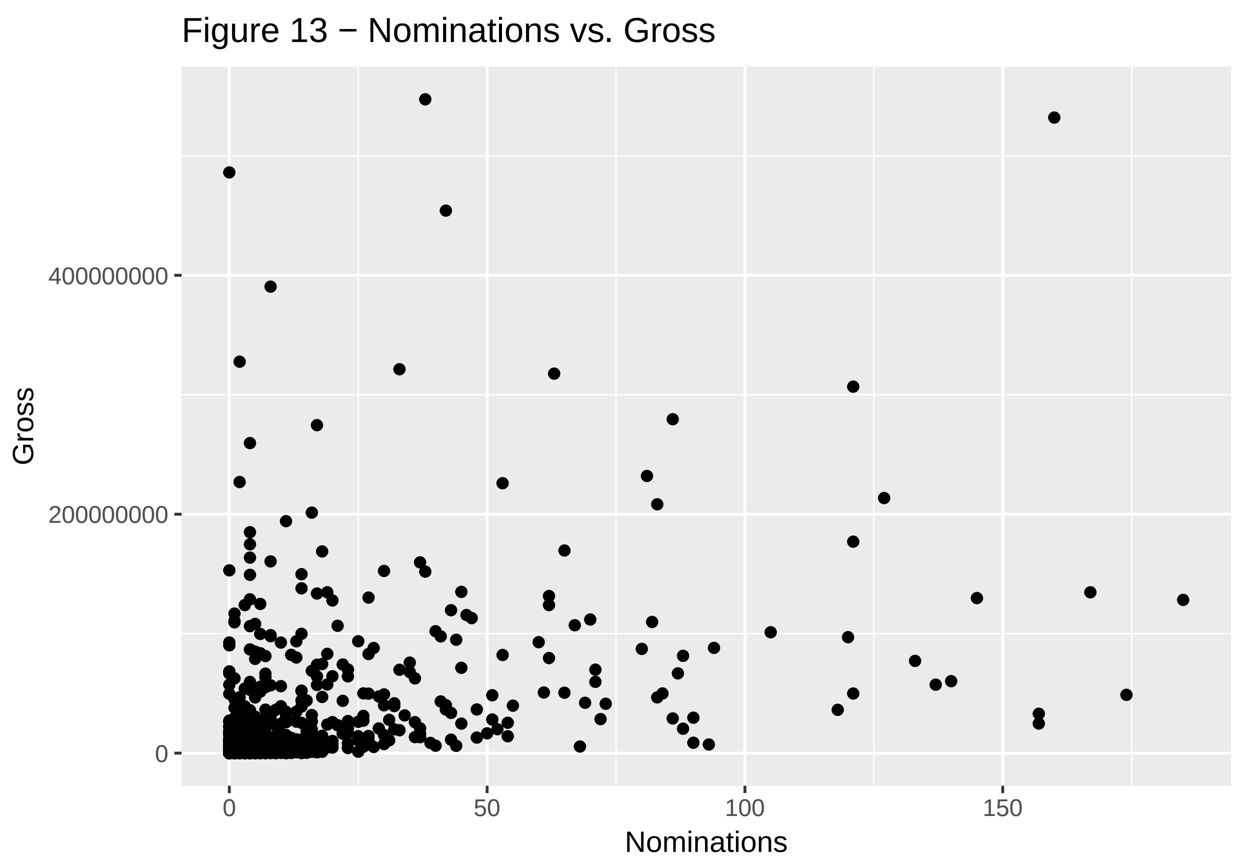
```
## Warning: Removed 4027 rows containing missing values (geom_point).
```



Let's see Nominations alone against Gross

```
ggplot(df) +
  geom_point(data=df, aes(x=Nominations, y=Gross)) +
  labs(title="Figure 13 - Nominations vs. Gross",
       x="Nominations",
       y="Gross")
```

```
## Warning: Removed 4027 rows containing missing values (geom_point).
```



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

A: We can notice from Figures 11, Figure 12, and Figure 13 that neither wins nor nominations correlate with Gross. Both wins and nominations fall below 50 regardless to Gross.

7. Movie ratings from IMDb and Rotten Tomatoes

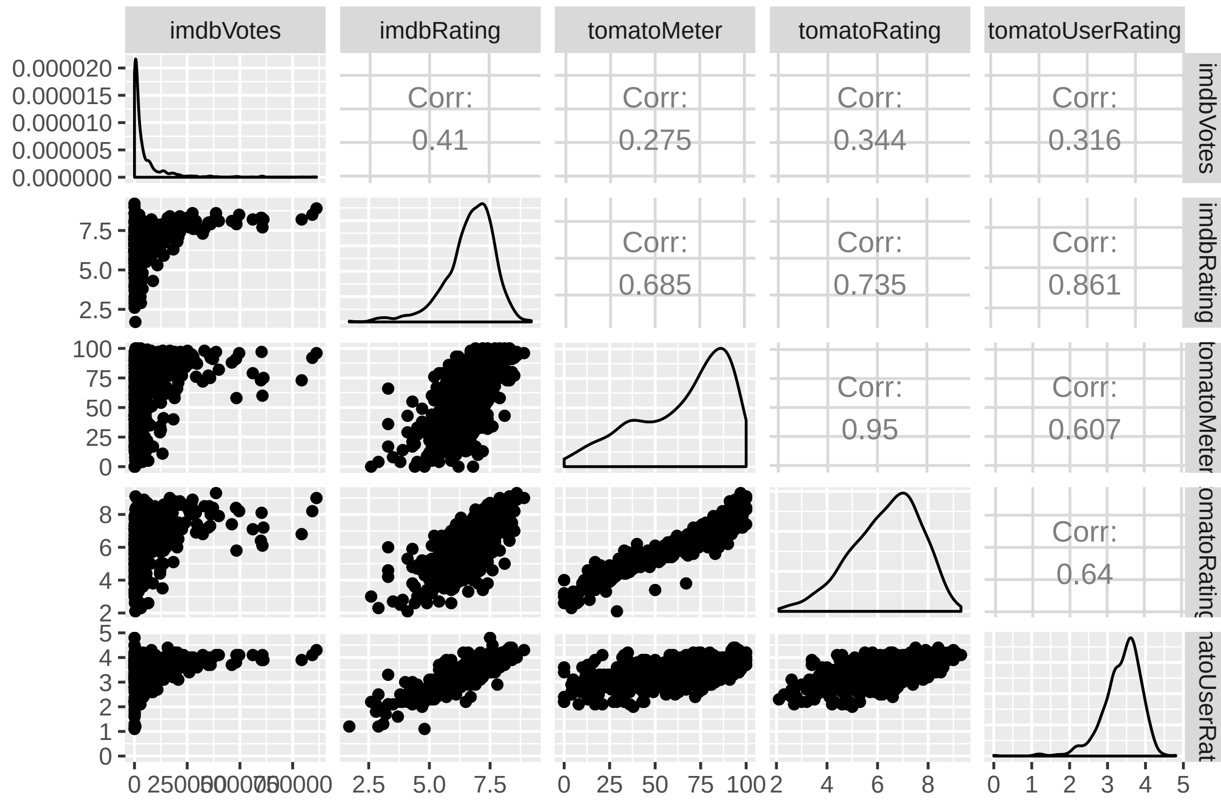
There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by several variables pre-fixed by `tomato`). Read up on such ratings on the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

Investigate the pairwise relationships between these different descriptors using graphs.

```r
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
chart <- ggpairs(df[, c("imdbVotes","imdbRating","tomatoMeter","tomatoRating", "tomatoUserRating")],
                 title = "Figure 14 - Pairwise relationships between IMDB and tomato ratings.")
print(chart)
```

```
## Warning: Removed 4033 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4033 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4125 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4125 rows containing missing values
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4076 rows containing missing values

## Warning: Removed 4033 rows containing missing values (geom_point).

## Warning: Removed 4033 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4125 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4125 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4076 rows containing missing values

## Warning: Removed 4125 rows containing missing values (geom_point).

## Warning: Removed 4125 rows containing missing values (geom_point).

## Warning: Removed 4125 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4125 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4128 rows containing missing values

## Warning: Removed 4125 rows containing missing values (geom_point).

## Warning: Removed 4125 rows containing missing values (geom_point).

## Warning: Removed 4125 rows containing missing values (geom_point).

## Warning: Removed 4125 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 4128 rows containing missing values

## Warning: Removed 4076 rows containing missing values (geom_point).

## Warning: Removed 4076 rows containing missing values (geom_point).

## Warning: Removed 4128 rows containing missing values (geom_point).

## Warning: Removed 4128 rows containing missing values (geom_point).

## Warning: Removed 4074 rows containing non-finite values (stat_density).
```

Figure 14 − Pairwise relationships between IMDB and tomato ratings.



We can notice from *Figure 14 - Pairwise relationships between IMDB and tomato ratings.* that most rating columns have a bell shaped distribution with some sort skewness except imdbVotes. We can also notice that tomatoRating and tomatoMeter have high correlation 95 which make sense given that tomatoMeter is the percentage of critics giving positive review and tomatoRating is overall critics average. tomatoUserRating and imdbRating follows that with 86.1 correlation which make sense since both provided by users. tomatoRating and imdbRating follows that with 73.5 correlation. Lowest correlation happend to be between tomatoMeter and imdbVotes.

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

A: We can notice from *Figure 14 - Pairwise relationships between IMDB and tomato ratings.* that tomatoRating and imdbRating have 73.5 correlation, both have similar bell shapped distributions, both have peaks close to 7. Although there is high correlation, but it is not as high as the 86.1 correlation between tomatoUserRating and imdbRating.

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).

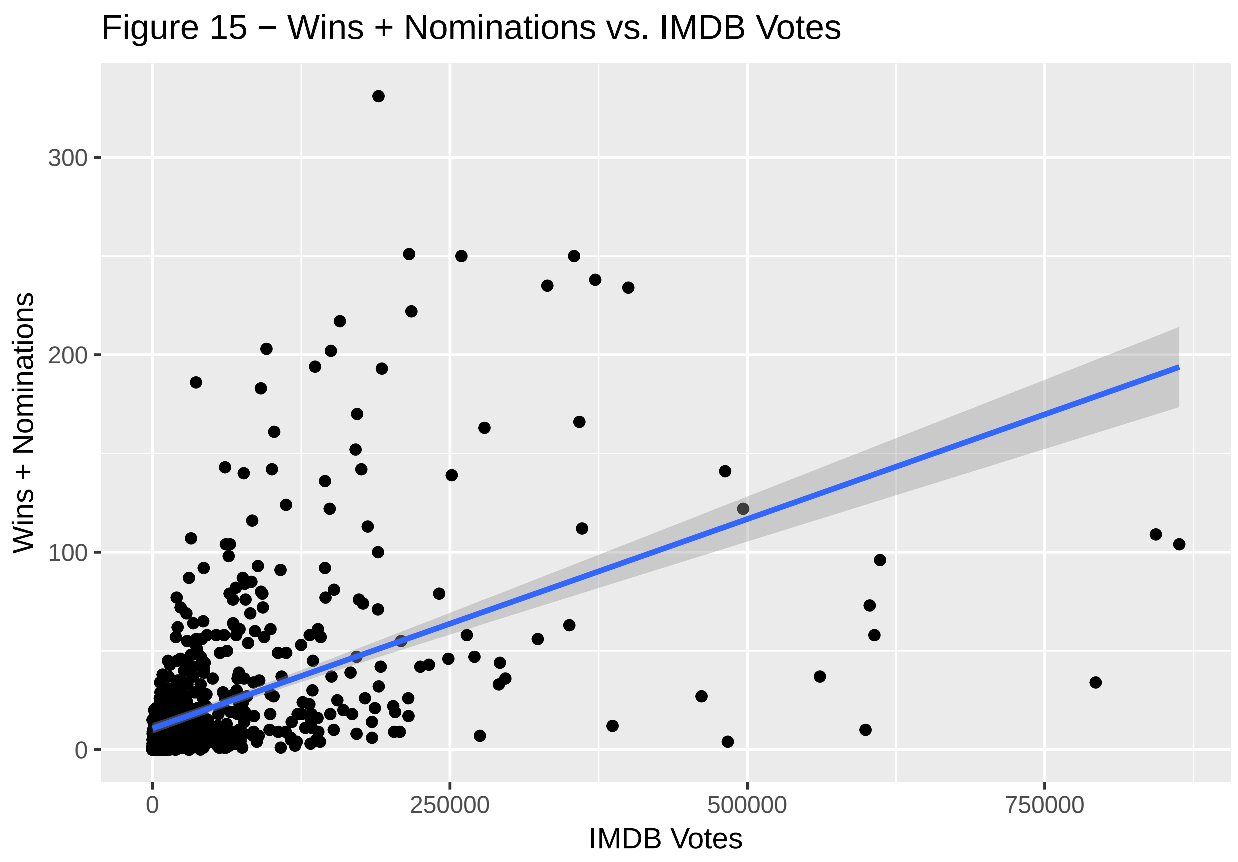
```
# TODO: Show how ratings and awards are related
#Will study Win + Nominations against ("imdbVotes","imdbRating","tomatoMeter","tomatoRating", "tomatoUs
```

Win + Nominations vs. imdbVotes

```
chart <- ggplot(df) +
  geom_point(aes(x=imdbVotes,Wins+Nominations)) +
  geom_smooth(aes(x=imdbVotes, Wins+Nominations), method="lm") +
  labs(title="Figure 15 - Wins + Nominations vs. IMDB Votes",
       x="IMDB Votes",
       y="Wins + Nominations")
print(chart)
```

```
## Warning: Removed 4033 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 4033 rows containing missing values (geom_point).
```



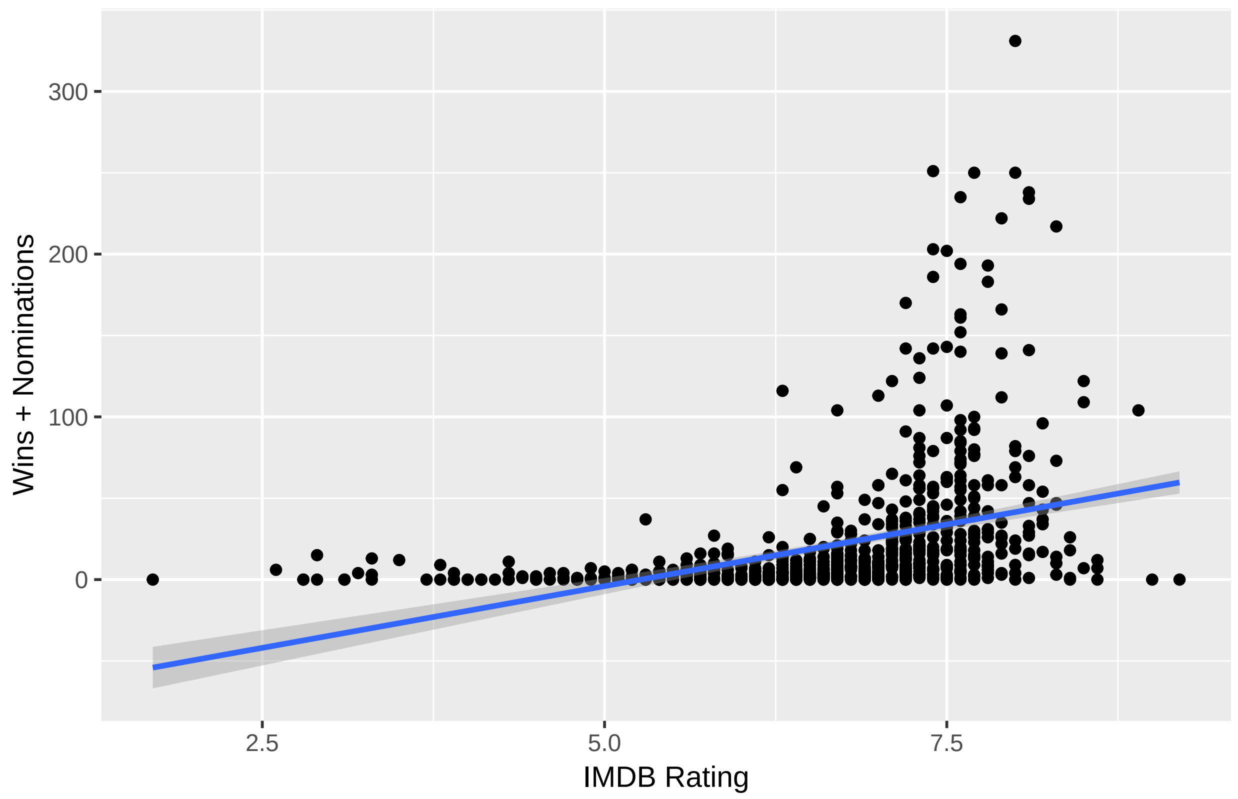
### Win + Nominations vs. imdbRating

```
chart <- ggplot(df) +
  geom_point(aes(x=imdbRating,Wins+Nominations)) +
  geom_smooth(aes(x=imdbRating, Wins+Nominations), method="lm") +
  labs(title="Figure 16 - Wins + Nominations vs. IMDB Rating",
       x="IMDB Rating",
       y="Wins + Nominations")
print(chart)
```

```
## Warning: Removed 4033 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 4033 rows containing missing values (geom_point).
```

Figure 16 − Wins + Nominations vs. IMDB Rating

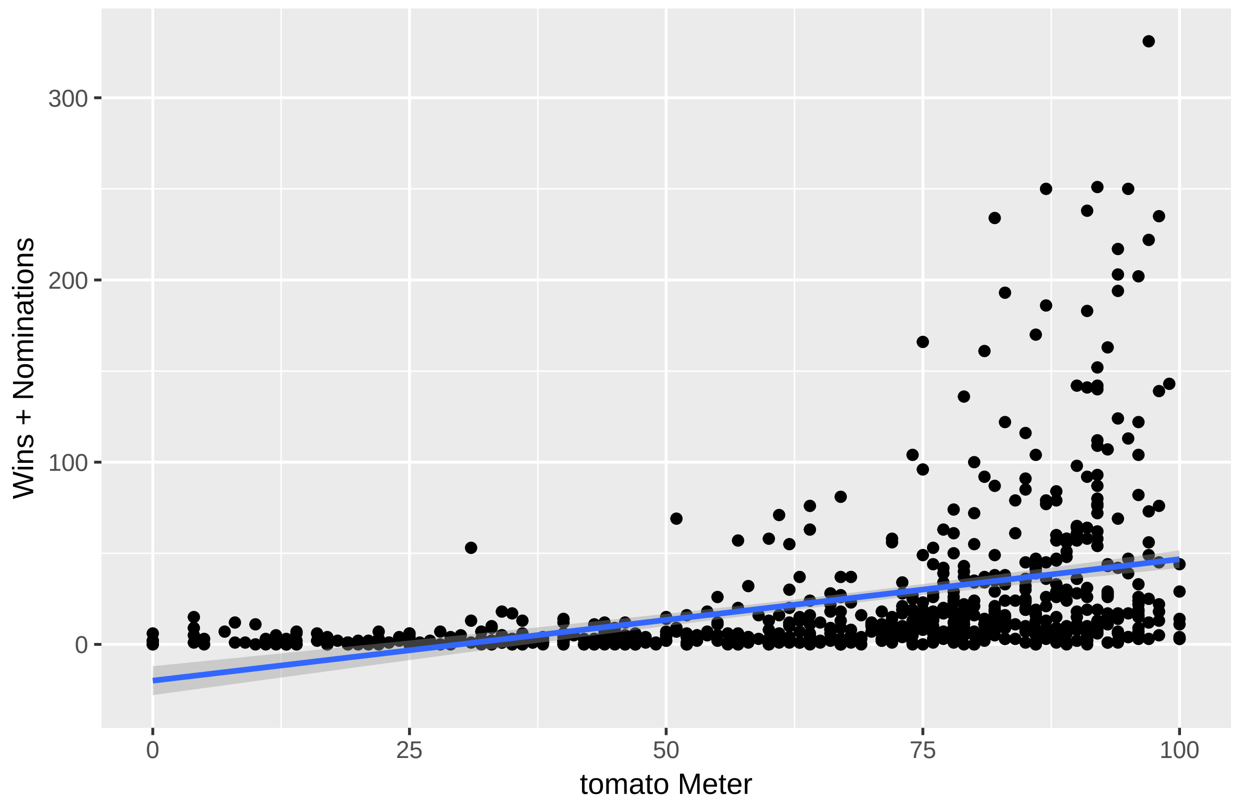


### Win + Nominations vs. tomatoMeter

```
chart <- ggplot(df) +
  geom_point(aes(x=tomatoMeter,Wins+Nominations)) +
  geom_smooth(aes(x=tomatoMeter, Wins+Nominations), method="lm") +
  labs(title="Figure 17 - Wins + Nominations vs. tomato Meter",
       x="tomato Meter",
       y="Wins + Nominations")
print(chart)
```

```
## Warning: Removed 4125 rows containing non-finite values (stat_smooth).

## Warning: Removed 4125 rows containing missing values (geom_point).
```

Figure 17 − Wins + Nominations vs. tomato Meter

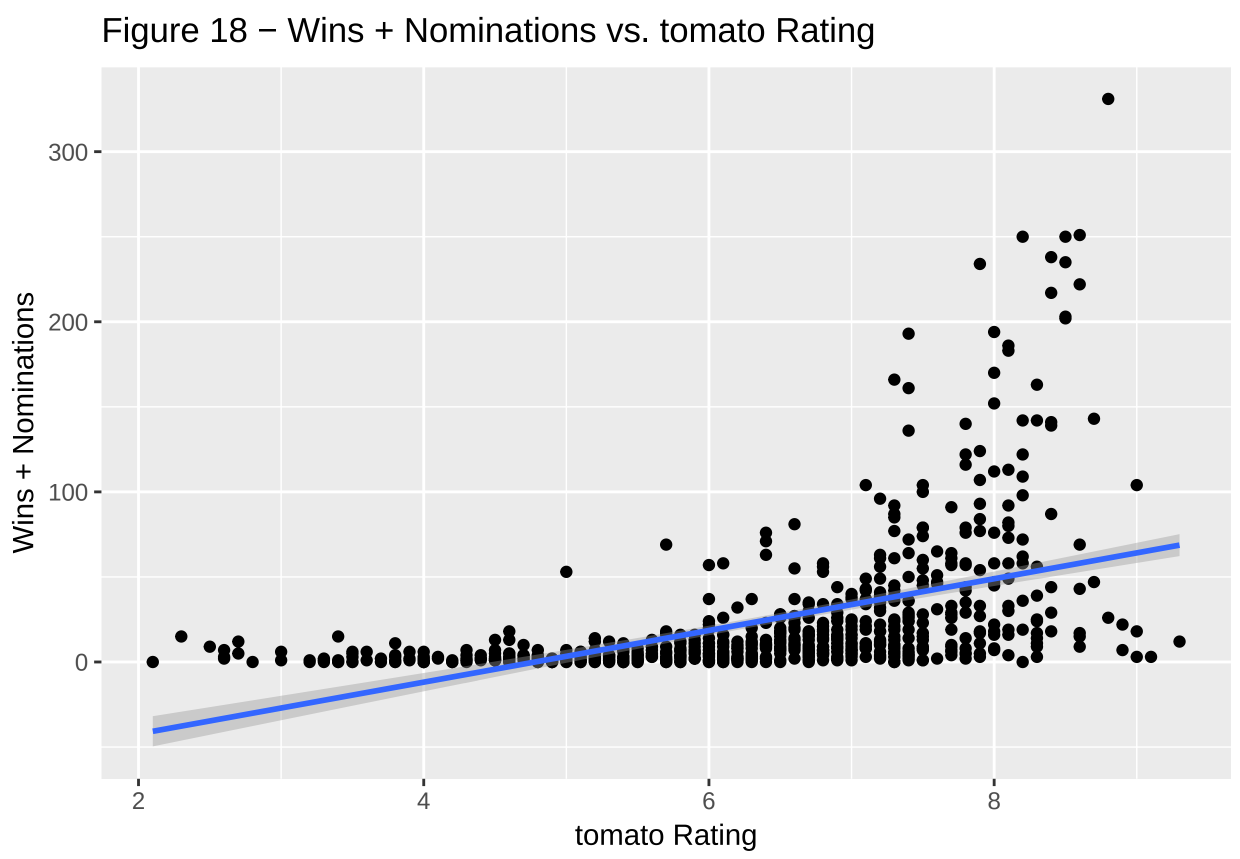


### Win + Nominations vs. tomato Rating

```
chart <- ggplot(df) +
  geom_point(aes(x=tomatoRating,Wins+Nominations)) +
  geom_smooth(aes(x=tomatoRating, Wins+Nominations), method="lm") +
  labs(title="Figure 18 - Wins + Nominations vs. tomato Rating",
       x="tomato Rating",
       y="Wins + Nominations")
print(chart)
```

```
## Warning: Removed 4125 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 4125 rows containing missing values (geom_point).
```

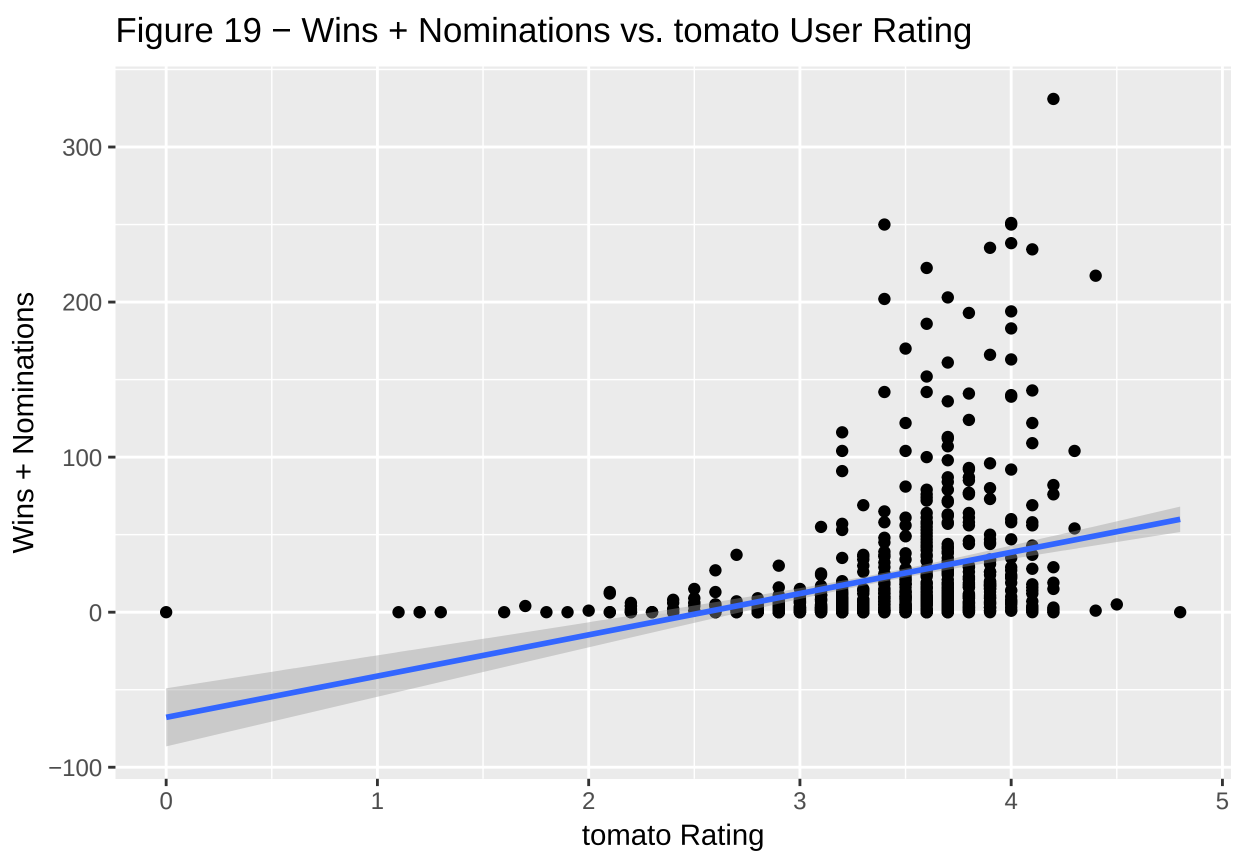


### Win + Nominations vs. tomato User Rating

```
chart <- ggplot(df) +
  geom_point(aes(x=tomatoUserRating,Wins+Nominations)) +
  geom_smooth(aes(x=tomatoUserRating, Wins+Nominations), method="lm") +
  labs(title="Figure 19 - Wins + Nominations vs. tomato User Rating",
       x="tomato Rating",
       y="Wins + Nominations")
print(chart)
```

```
## Warning: Removed 4074 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 4074 rows containing missing values (geom_point).
```



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: All ratings have clear linear correlation with movies' Wins + Nominations; so they can be good predictors for movie award model. The highest correlation is between IMDB votes and total award wins/nomination.

9. Expected insights

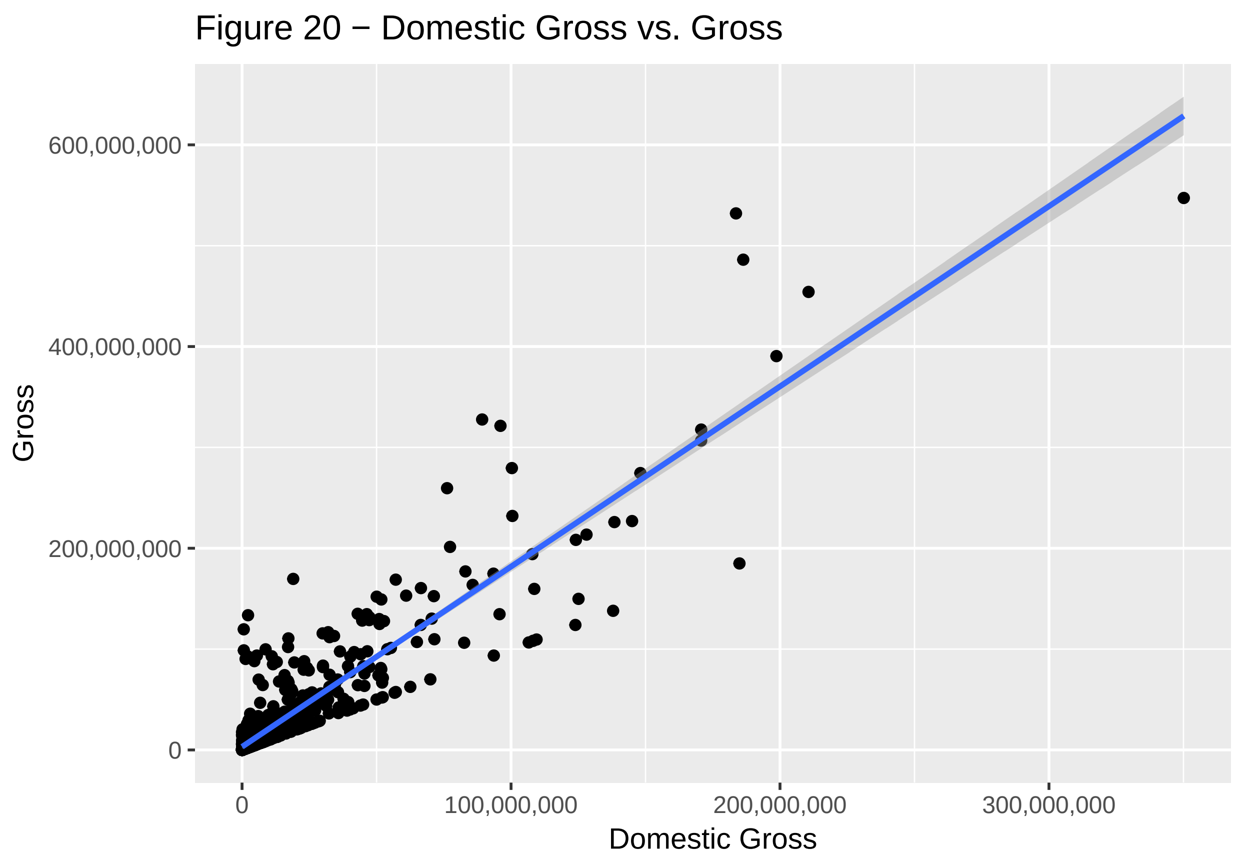
Come up with two new insights (backed up by data and graphs) that is expected. Here a new means insights that are not an immediate consequence of one of the above tasks. You may use any of the columns already explored above or a different one in the dataset, such as `Title`, `Actors`, etc.

```
# TODO: Find and illustrate two expected insights
```

Expected insight #1 - Domestic Gross vs Gross

```
chart <- ggplot(df) +
  geom_point(aes(x=Domestic_Gross,Gross)) +
  geom_smooth(aes(x=Domestic_Gross,Gross), method="lm") +
  scale_x_continuous(labels = scales::comma) +
  scale_y_continuous(labels = scales::comma) +
  labs(title="Figure 20 - Domestic Gross vs. Gross",
     x="Domestic Gross",
     y="Gross")
print(chart)
```

```
## Warning: Removed 4027 rows containing non-finite values (stat_smooth).

## Warning: Removed 4027 rows containing missing values (geom_point).
```



Q: Expected insight #1.

A: As it is clear from *Figure 13 - Domestic Gross vs. Gross* that is a linear relationship between Domestic Gross and Gross.

Expected insight #2 - Observations regarding how much the dataset is focused on USA English movies

```
print(paste("%", (( nrow(df[(df$Language == "English"),]) / nrow(df) ) * 100), "of movies are only in En
```

```
## [1] "% 94.2016980741354 of movies are only in English."
```

```
print(paste("%", (( nrow(df[(df$Country == "USA"),]) / nrow(df) ) * 100), "of movies are made in USA.",
```

```
## [1] "% 91.4889211016774 of movies are made in USA."
```

```
print(paste("%", (( nrow(df[((df$Gross + df$Domestic_Gross - df$Budget) < 0),])/nrow(df) ) * 100), "of m
```

```
## [1] "% 90.9919237937461 of movies are losing money."
```

Q: Expected insight #2.

A: The following statistics show how much the data set is focused on movies produced in USA in English: % 94.2 of movies are only in English. % 91.5 of movies are made in USA. % 90.9 of movies are losing money.

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.

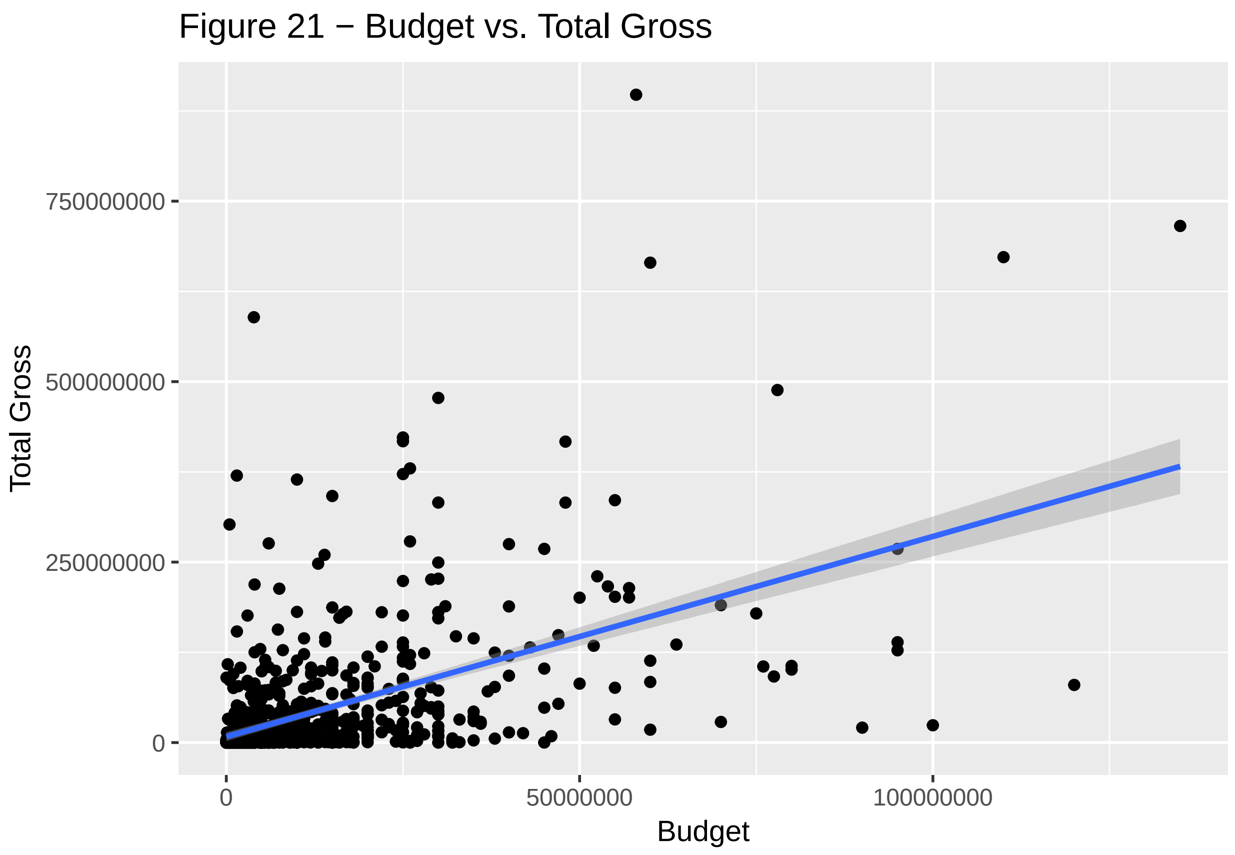
```
# TODO: Find and illustrate one unexpected insight
```

### Investigating Budget vs. Total Gross

```
chart <- ggplot(df) +
  geom_point(aes(x=Budget,Gross + Domestic_Gross)) +
  geom_smooth(aes(x=Budget, Gross + Domestic_Gross),method="lm") +
  labs(title="Figure 21 - Budget vs. Total Gross",
       x="Budget",
       y="Total Gross")
print(chart)
```

```
## Warning: Removed 4027 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 4027 rows containing missing values (geom_point).
```

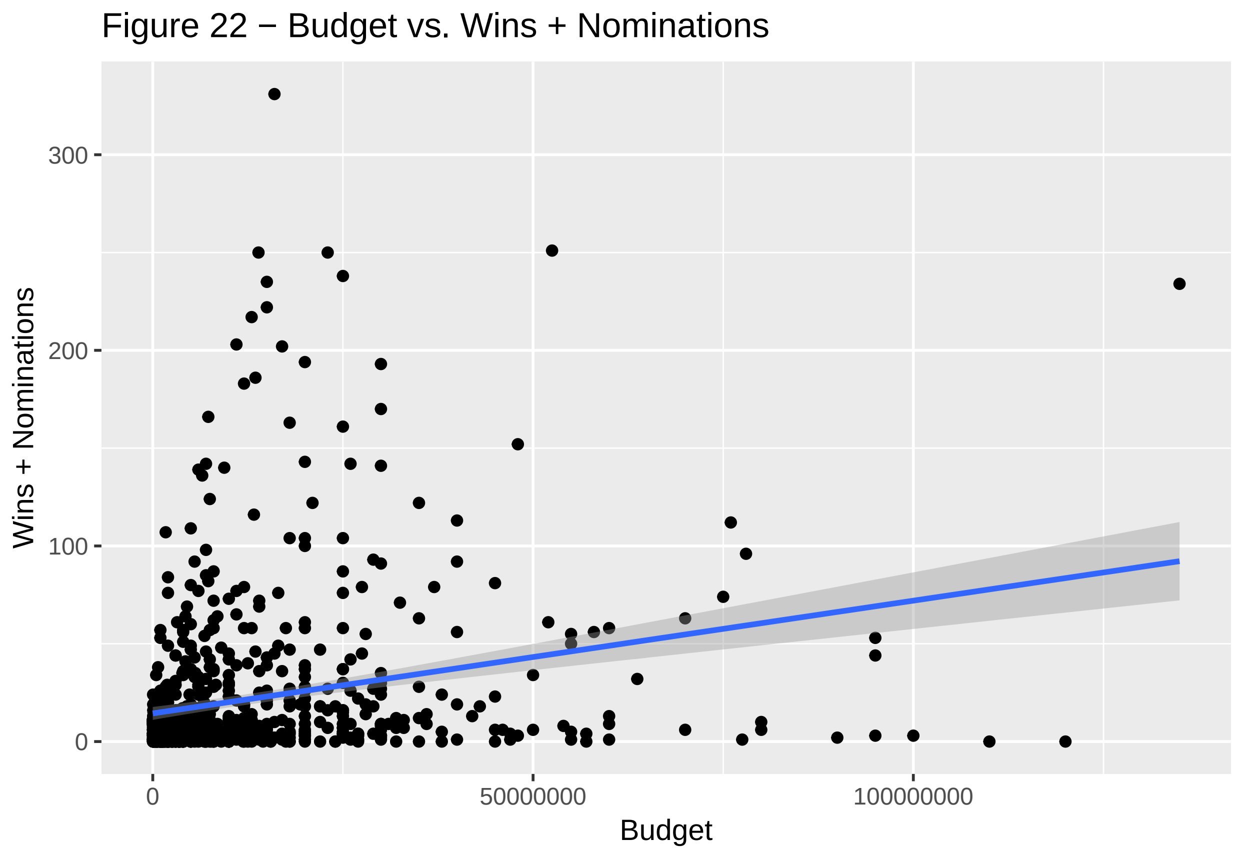


### Investigating Budget vs. Total Award wins and nominations

```
chart <- ggplot(df) +
  geom_point(aes(x=Budget, Wins+Nominations)) +
  geom_smooth(aes(x=Budget, Wins+Nominations),method="lm") +
  labs(title="Figure 22 - Budget vs. Wins + Nominations",
       x="Budget",
       y="Wins + Nominations")
print(chart)
```

```
## Warning: Removed 4027 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 4027 rows containing missing values (geom_point).
```



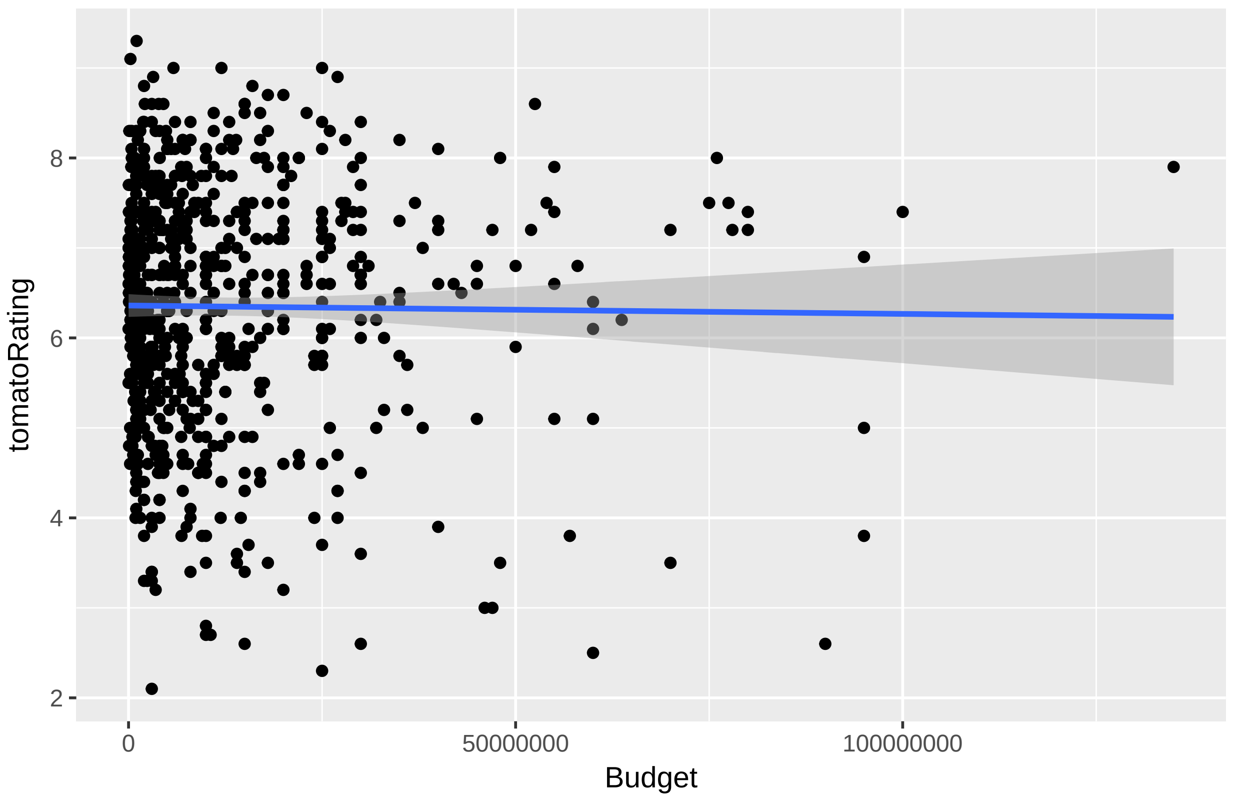
### Investigating Budget vs. tomatoRating (Critic reviews)

```
chart <- ggplot(df) +
  geom_point(aes(x=Budget, tomatoRating )) +
  geom_smooth(aes(x=Budget, tomatoRating),method="lm") +
  labs(title="Figure 23 - Budget vs. tomato Rating",
       x="Budget",
       y="tomatoRating")
print(chart)
```

```
## Warning: Removed 4125 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 4125 rows containing missing values (geom_point).
```

Figure 23 − Budget vs. tomato Rating



Q: Unexpected insight.

A: As we notice from Figures 20, 21, 22: Budget is highly correlated with total gross and award winning and nominations but not with average critics ratings. This means that increasing the movie budget will increase its gross and also its awards' chances, but it is irrelavant to critics reviews of the movie. I was expecting the awards' wins/nominations to correlate with tomatoRating, but it doesn't.