# By Xiaodong Gu (xgu60)

title: ‘Project 1: Explore and Prepare Data’ subtitle: |- CSE6242 - Data and Visual Analytics - Spring 2017 Due: Sunday, March 5, 2017 at 11:59 PM UTC-12:00 on T-Square output: html_document: default html_notebook: default pdf_document: default —

*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 â□□second windowâ□□ streaming rights).

# Instructions

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

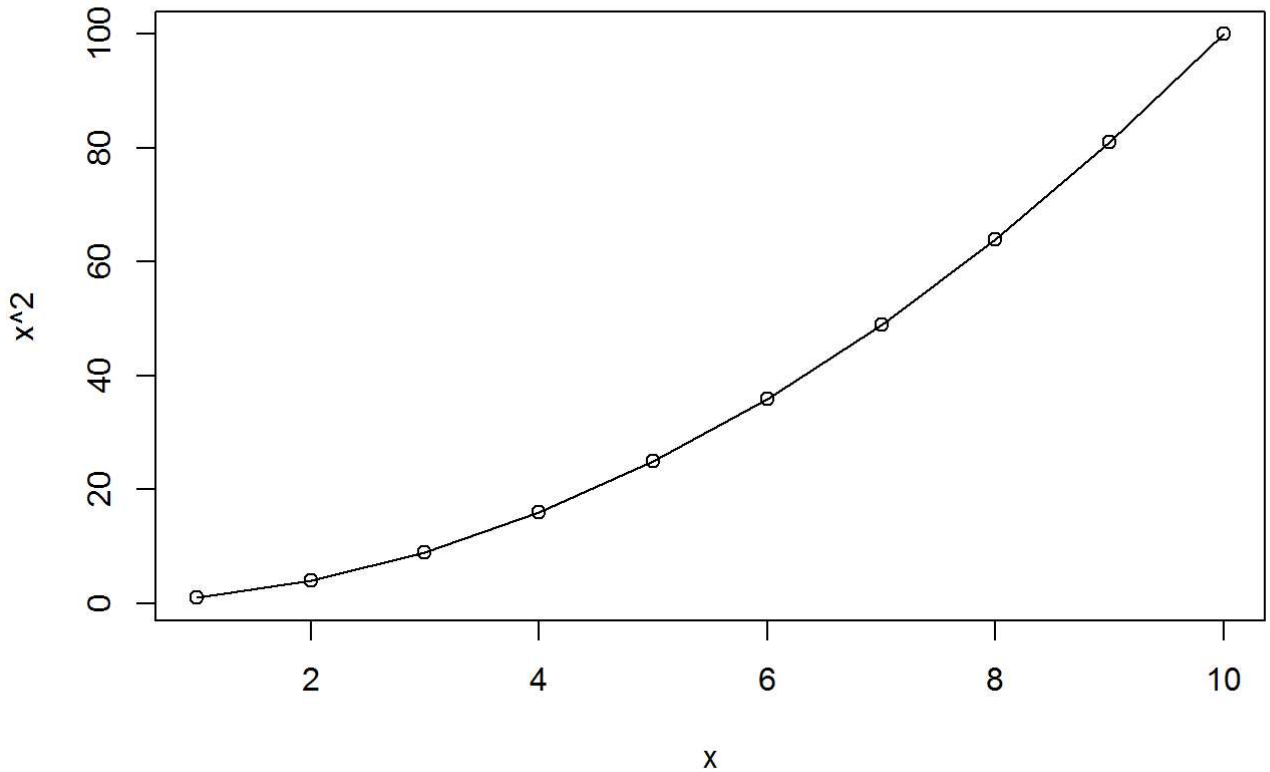
When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

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

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

Plots appear inline too:

```
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:

```
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)
library(GGally)
library(reshape)
```

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.

```
# 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**: 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.

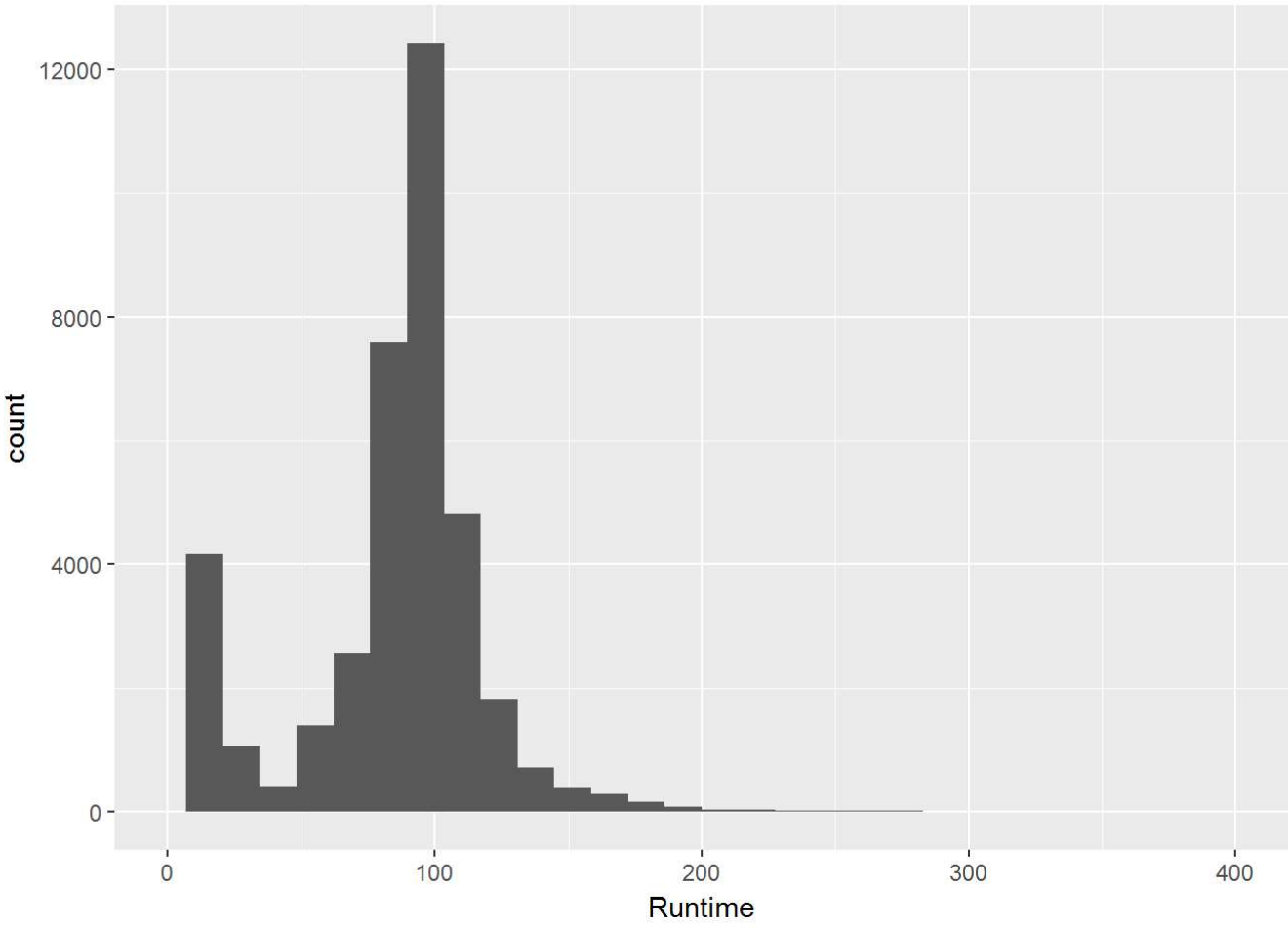
```
# TODO: Replace df$Runtime with a numeric column containing the runtime in minutes
#df$Runtime = strtoi(do.call(rbind, strsplit(df$Runtime, " ", fixed = TRUE))[, 1])
calTime = function(str){
  if(str == "N/A"){
    return(NA)
  }

  strList = unlist(strsplit(str, " "))
  h.pos = grep("h", strList)
  if(!is.na(h.pos[1])){
    h.num = as.numeric(strList[h.pos[1] - 1])
  } else{
    h.num = 0
  }
  min.pos = grep("min", strList)
  if(!is.na(min.pos[1])){
    min.num = as.numeric(strList[min.pos[1] - 1])
  } else{
    min.num = 0
  }
  return (h.num * 60 + min.num)
}

df$Runtime = mapply(calTime, df$Runtime)
```

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.

```
# TODO: Investigate the distribution of Runtime values and how it varies by Year and Budget
plot2.1 = ggplot(df, aes(Runtime)) +
  geom_histogram() +
  xlim(0, 400)

plot2.2 = ggplot(df, aes(factor(floor(df$Year / 10)), Runtime)) +
  geom_boxplot()

plot2.3 = ggplot(df[!is.na(df$Runtime) & !is.na(df$Budget), ], aes(Runtime, Budget)) +
  geom_point() +
  geom_smooth(method="lm")

plot2.4 = ggplot(df[df$Runtime<250 & !is.na(df$Runtime), ], aes(Runtime, Budget)) +
  geom_point() +
  #xlim(0, 350) +
  facet_wrap(~factor(floor(Runtime/50)), scales="free") +
  geom_smooth(method="lm")

plot2.1
```

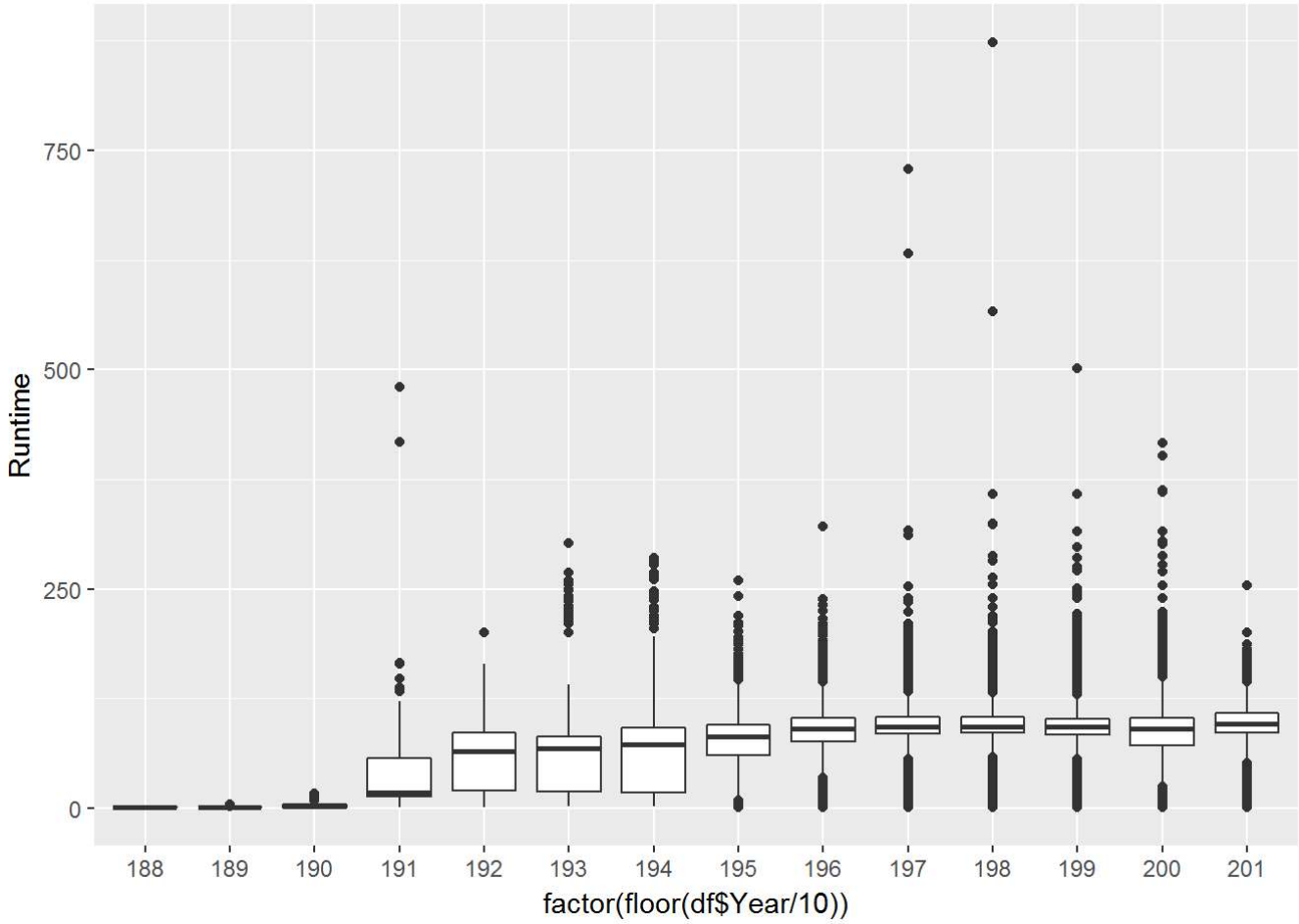
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 760 rows containing non-finite values (stat_bin).
```

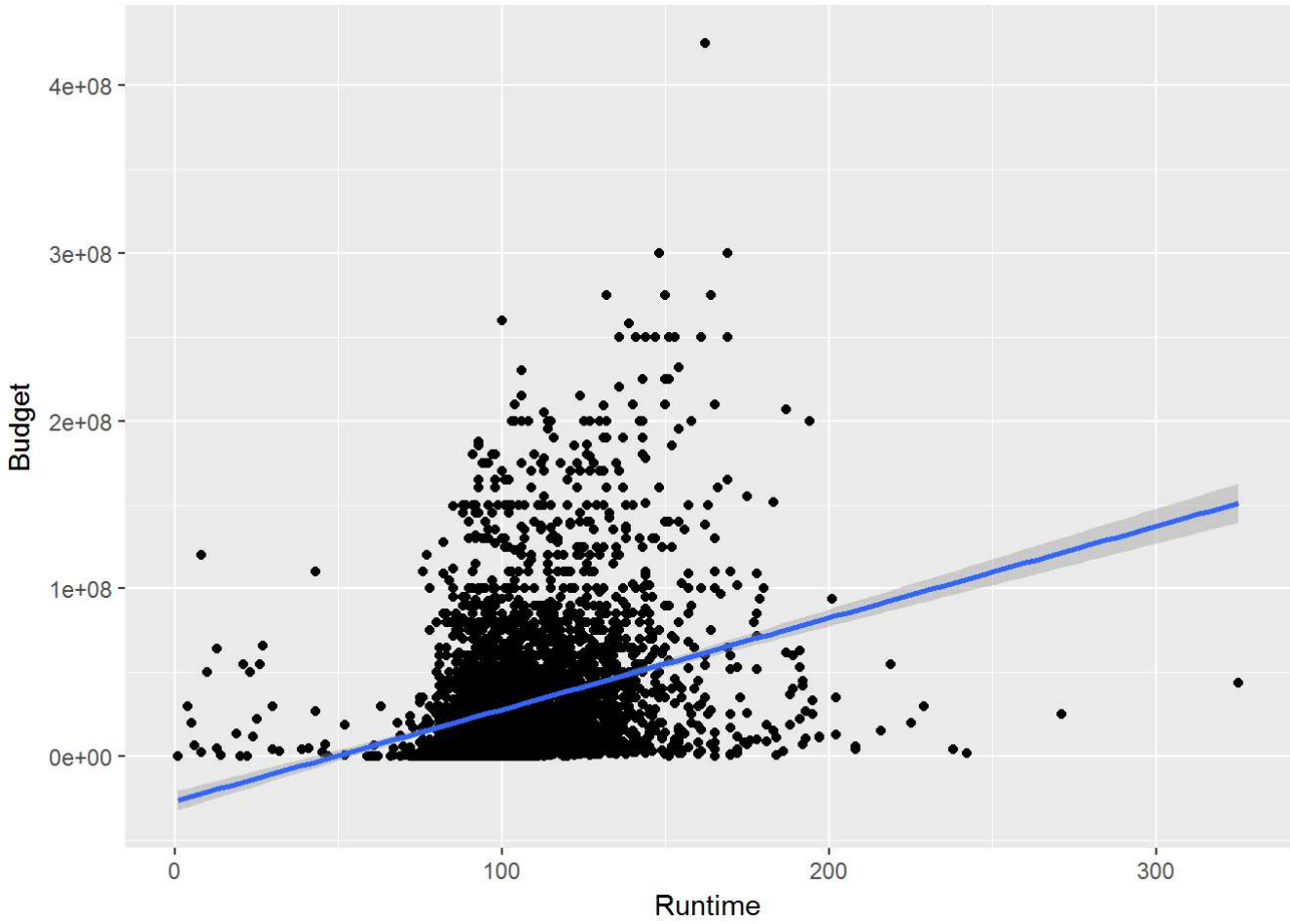


```
plot2.2
```

```
## Warning: Removed 751 rows containing non-finite values (stat_boxplot).
```



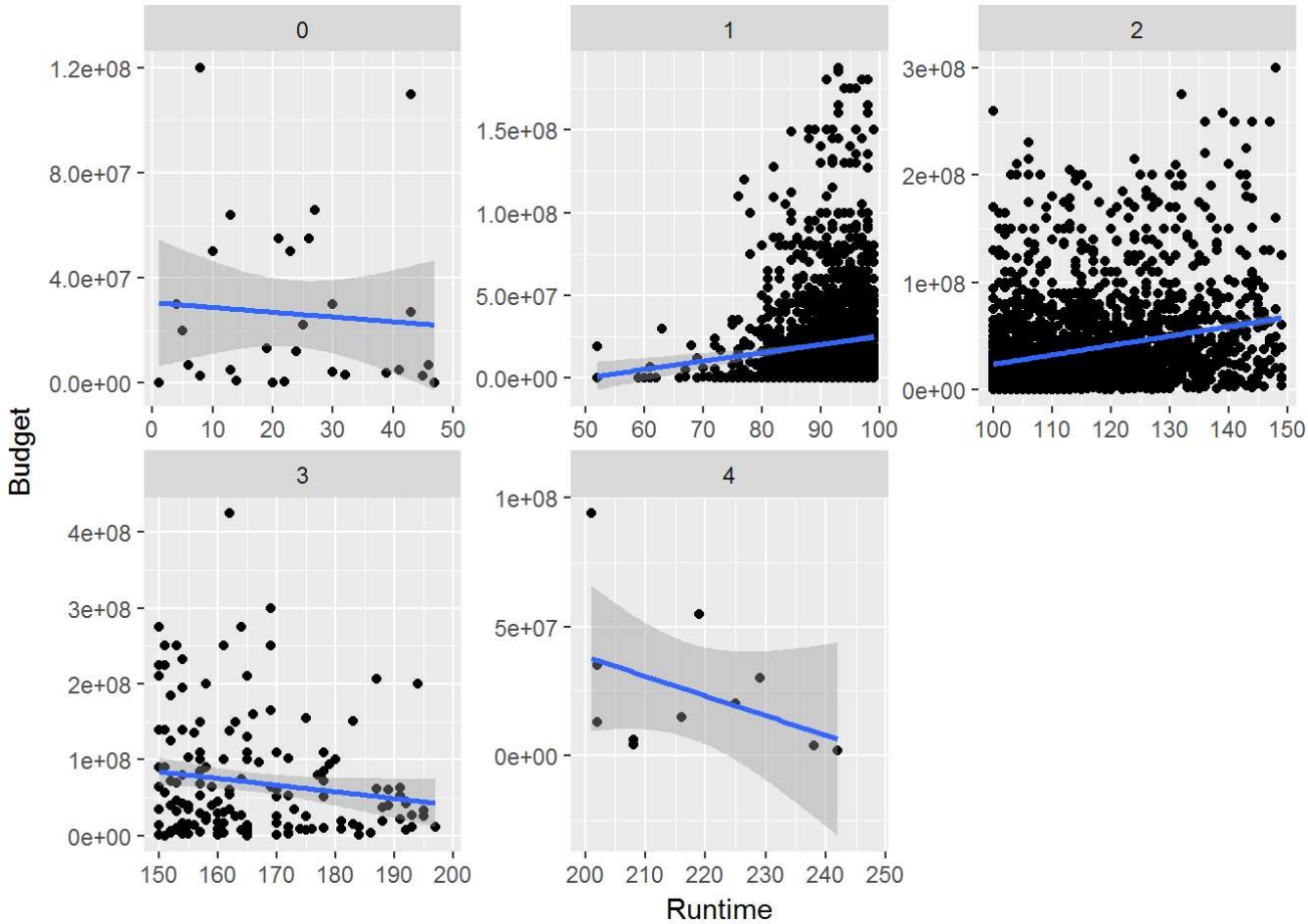

```
plot2.3
```




```
plot2.4
```

```
## Warning: Removed 34675 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 34675 rows containing missing values (geom_point).
```




```
cor(df$Runtime, df$Budget, use="complete")
```

```
## [1] 0.2867817
```

*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**: 1. There is a bi-model distribution for Runtime of movies. There are more than 4000 movies have short runtime. The counts of movies decrease and increase and further decrase with runtime. More than 12000 movies have runtime between 90-100 min. There are also a few movies have very long runtime. 2. Before 1900, runtime for movies are very short. From 1900 to 1950, the runtime of movies increase, and there are big variations of runtime between movies since the interquartile ranges (IQR) are big. Since 1950, most movies have similar runtime. These movies have similar median runtime, and IQR are small. However, there are also lots of outlier points. 3. Buget is positively correlated with runtime. It is reasonalbe that longer movies need much more money to produce. However, the correlation is not very strong (ce = 0.2867). However, if you factor Runtime to < 50, 50-100, 100-150, 150-200, 200-250 min, Budget is positively correlated with Runtime when the Runtime is between 50-100 and 100-150, and negatively correlated with Runtime when it is less than 50, and larger than 150 min.

# 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).

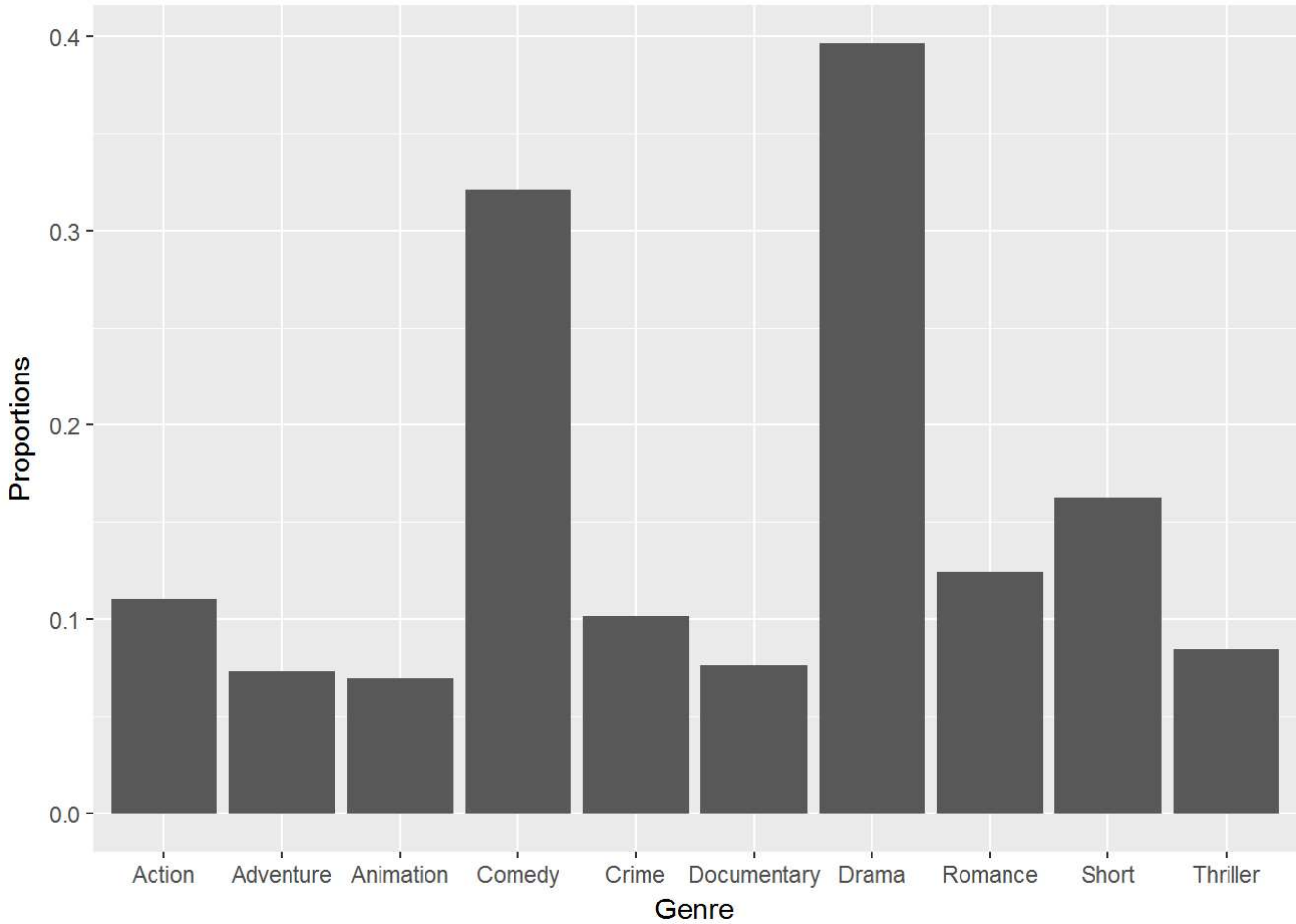
```
# TODO: Replace Genre with a collection of binary columns
#get a vector contains all genres
genre.all = unlist(strsplit(df$Genre, ", "))
genre.unique = unique(genre.all)
genre.unique = genre.unique[genre.unique != "N/A"]

#create new column, and insert 0 or 1
counts = c()
for(ge in genre.unique){
  colname = paste("G", ge, sep = ".")
  values = as.numeric(grepl(ge, df$Genre))
  df[, colname] = values
  counts = append(counts, sum(values))
}
df$Genre = NULL
```

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
genre.stat = data.frame(Genre = genre.unique, Counts = counts, Proportions = counts / nrow(df))
genre.stat = genre.stat[rev(order(genre.stat$Counts)),]
genre.stat[1:10,]
```

```
##          Genre Counts Proportions
## 6        Drama  15859    0.396475
## 8       Comedy  12849    0.321225
## 4        Short   6516    0.162900
## 3      Romance   4975    0.124375
## 15      Action   4413    0.110325
## 14       Crime   4062    0.101550
## 5     Thriller   3380    0.084500
## 1  Documentary   3051    0.076275
## 11   Adventure   2928    0.073200
## 20   Animation   2788    0.069700
```

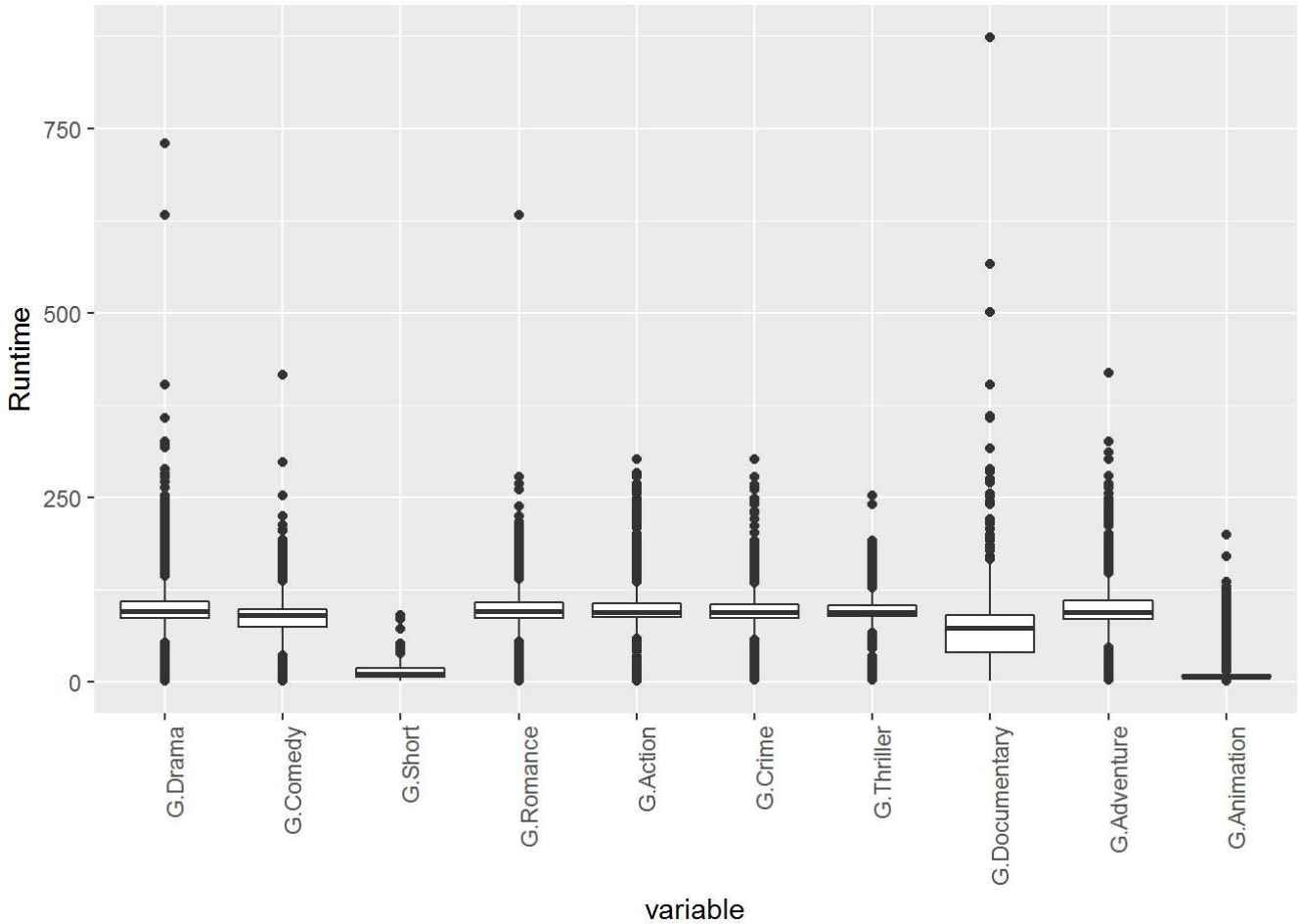
```
plot3.1 = ggplot(genre.stat[1:10,], aes(Genre, Proportions)) +
  geom_bar(stat="identity")

plot3.1
```




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
#create a new dataframe has Runtime and the top 10 Genres
df.new = df[, c(5, 44, 46, 42, 41, 53, 52, 43, 39, 49, 58)]
#melt the new dataframe
df.nm = melt(df.new, id="Runtime")

plot3.2 = ggplot(df.nm[df.nm$value == 1, ], aes(variable, Runtime)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

plot3.2
```

```
## Warning: Removed 809 rows containing non-finite values (stat_boxplot).
```




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

**A**: Short and Animation movies have small Runtime, and their IQRs are very narrow. Documentary movies have low median Runtime and wider IQR compared with others. The left 7 genres of movies have similar median Runtime and IQRs. There are some Drama, Romance and Documenray movies have very large Runtime.

It is expected that Short movies have small Runtime, but it is unexpected that Animation movies also have small Runtime. It is also surpring that some Drama and Documentary have very long Runtime (> 10 hours).

# 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.

```
# TODO: Remove rows with Released-Year mismatch
#get original row number
row.num1 = nrow(df)

#add column that match Year and Released
df$match = as.numeric(mapply(grepl, pattern=df$Year, x=df$Released))
df$match = df$match + as.numeric(is.na(df$Year) | is.na(df$Released))

#add column to tell whether Gross is NA
df$Gross.NA = as.numeric(is.na(df$Gross))

#new dataframe remove all dismatch rows
df.ma = df[df$match == 1, ]

#new dataframe dismatch Year and Released, but has Gross value
df.dmg = df[df$match == 0 & df$Gross.NA == 0,]

#new dataframe has Gross value
df.g = df[df$Gross.NA == 0,]

#if the row number of df.dmg large than 10% of total rows of df.g, add some rows back
set.seed(1)
df.dmg2= df.dmg[runif(nrow(df.dmg), 0.0, 1.0) > (nrow(df.g) * 0.1 / nrow(df.dmg)), ]
df = rbind(df.ma, df.dmg2)

df$match = NULL
df$Gross.NA = NULL

#row number of dataframe
row.num2 = nrow(df)

"Total rows have been removed:"
```

```
## [1] "Total rows have been removed:"
```

```
row.num1 - row.num2
```

```
## [1] 5418
```

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

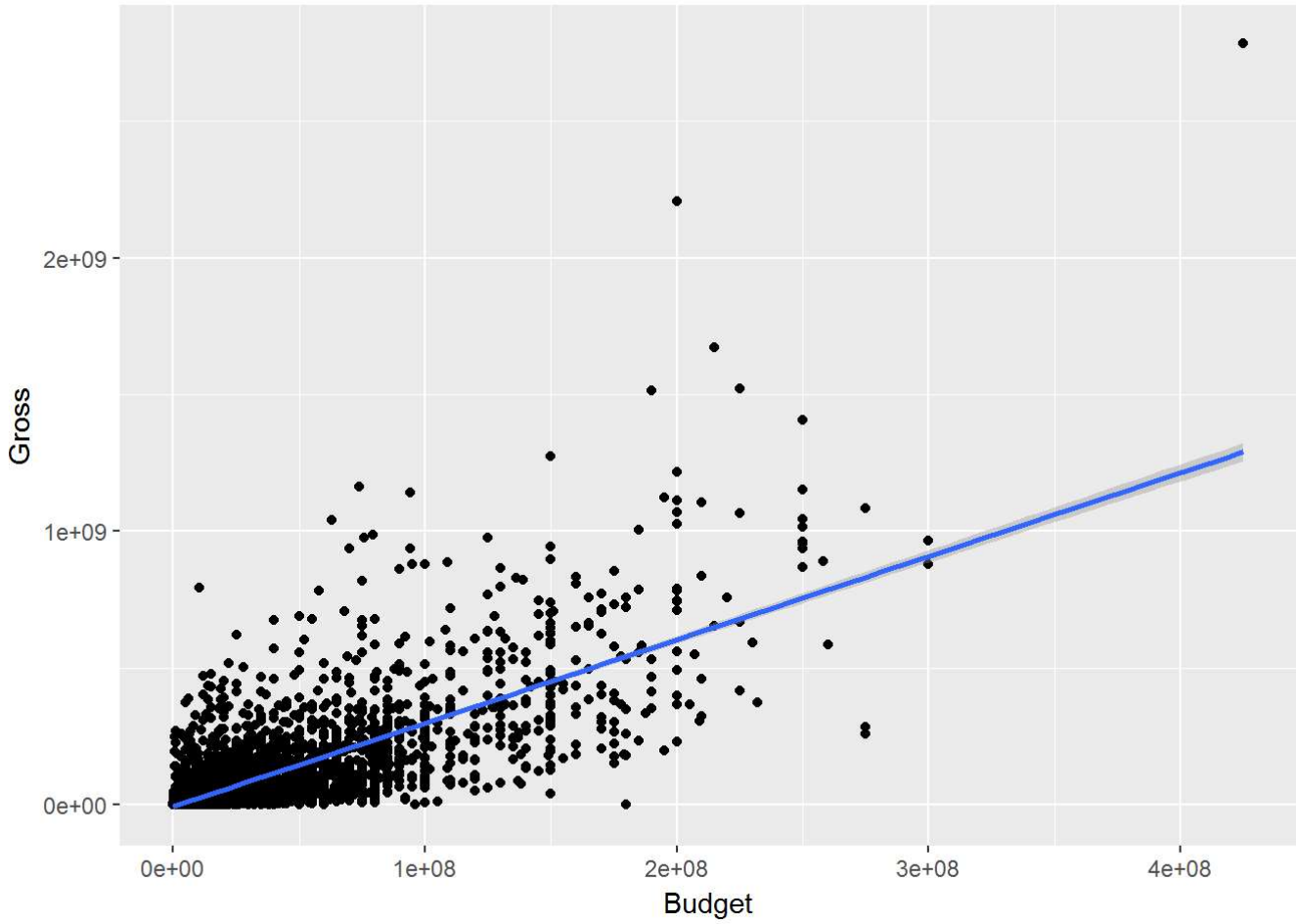
**A**: 1. generate a dataframe (df.ma) remove all dismatch year and released rows. 2. generate another dataframe (df.dmg) that dismatch year and released, but with gross value. 3. computer the number of 10% of rows with gross value. 4. if the row number in df.dmg is larger than 10% of rows with gross value, randomly select that portion and add back to df.ma. Finally, 5418 rows have been removed.

# 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.

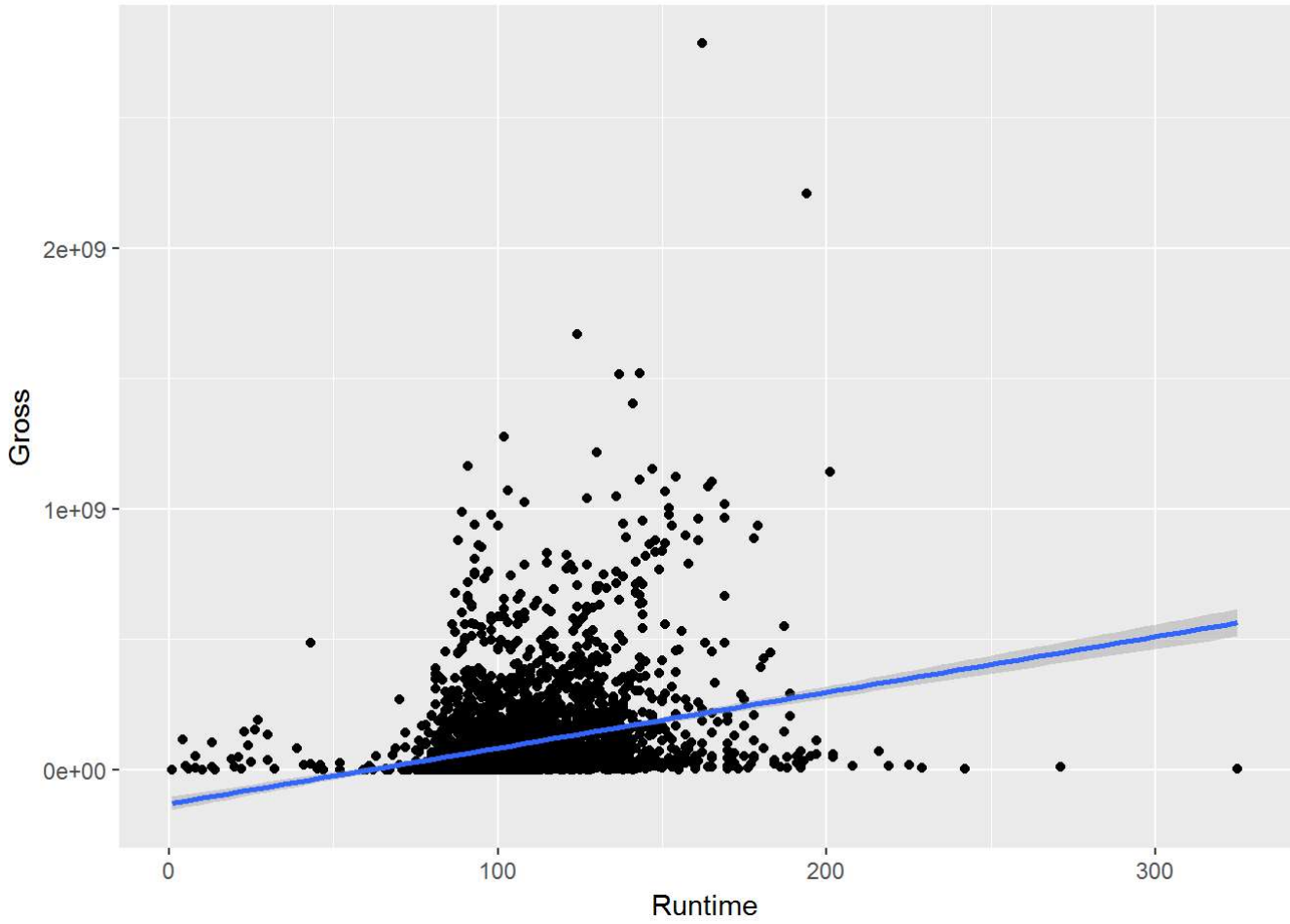
```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
#plot Gross vs Budget
plot5.1 = ggplot(df[!is.na(df$Gross), ], aes(Budget, Gross)) +
  geom_point() +
  geom_smooth(method="lm")

#plot Gross vs Runtime
plot5.2 = ggplot(df[!is.na(df$Gross), ], aes(Runtime, Gross)) +
  geom_point() +
  geom_smooth(method="lm")

#plot Gross vs Runtime
plot5.3 = ggplot(df[!is.na(df$Gross), ], aes(Runtime, Gross)) +
  geom_point() +
  facet_wrap(~factor(floor(Runtime/50)), scales="free") +
  geom_smooth(method="lm")

#create a new dataframe to plot
df.new = df[!is.na(df$Gross), c(5, 35, 37, 39:66)]
df.nm = melt(df.new, id = c("Runtime", "Budget", "Gross"))
df.nm = df.nm[df.nm$value == 1, ]

#plot Gross vs Budget in different genres
plot5.4 = ggplot(df.nm, aes(Budget, Gross)) +
  geom_point() +
  facet_wrap(~variable, scales="free") +
  geom_smooth(method="lm")

#plot Gross vs Runtime in different genres
plot5.5 = ggplot(df.nm, aes(Runtime, Gross)) +
  geom_point() +
  facet_wrap(~variable, scales="free") +
  geom_smooth(method="lm")

#plot Gross vs Genre
plot5.6 = ggplot(df.nm, aes(variable, Gross)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

plot5.1
```




```
plot5.2
```

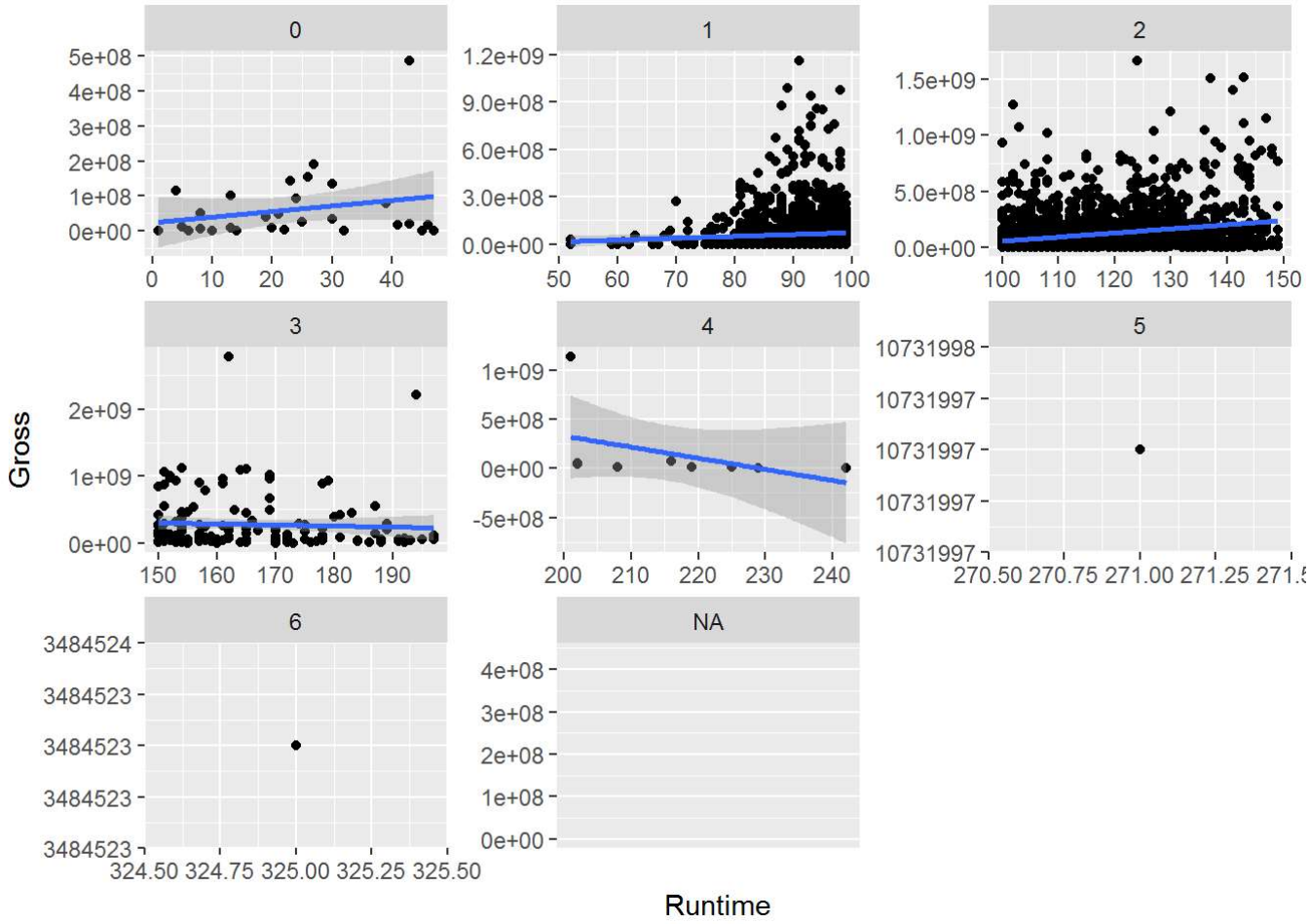
```
## Warning: Removed 38 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 38 rows containing missing values (geom_point).
```



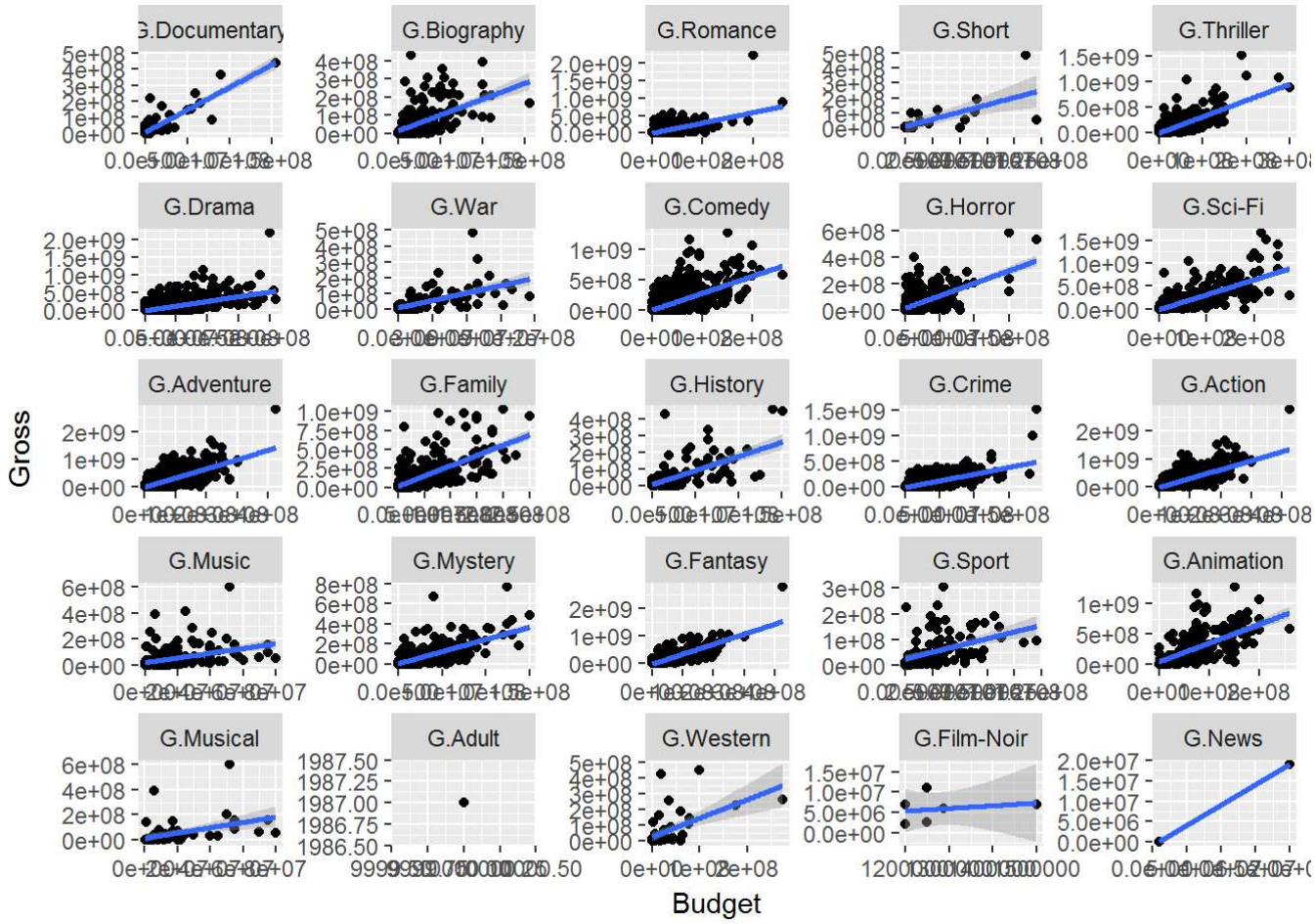

```
plot5.3
```

```
## Warning: Removed 38 rows containing non-finite values (stat_smooth).

## Warning: Removed 38 rows containing missing values (geom_point).
```




```
plot5.4
```

```
## Warning in qt((1 - level)/2, df): NaNs produced
```

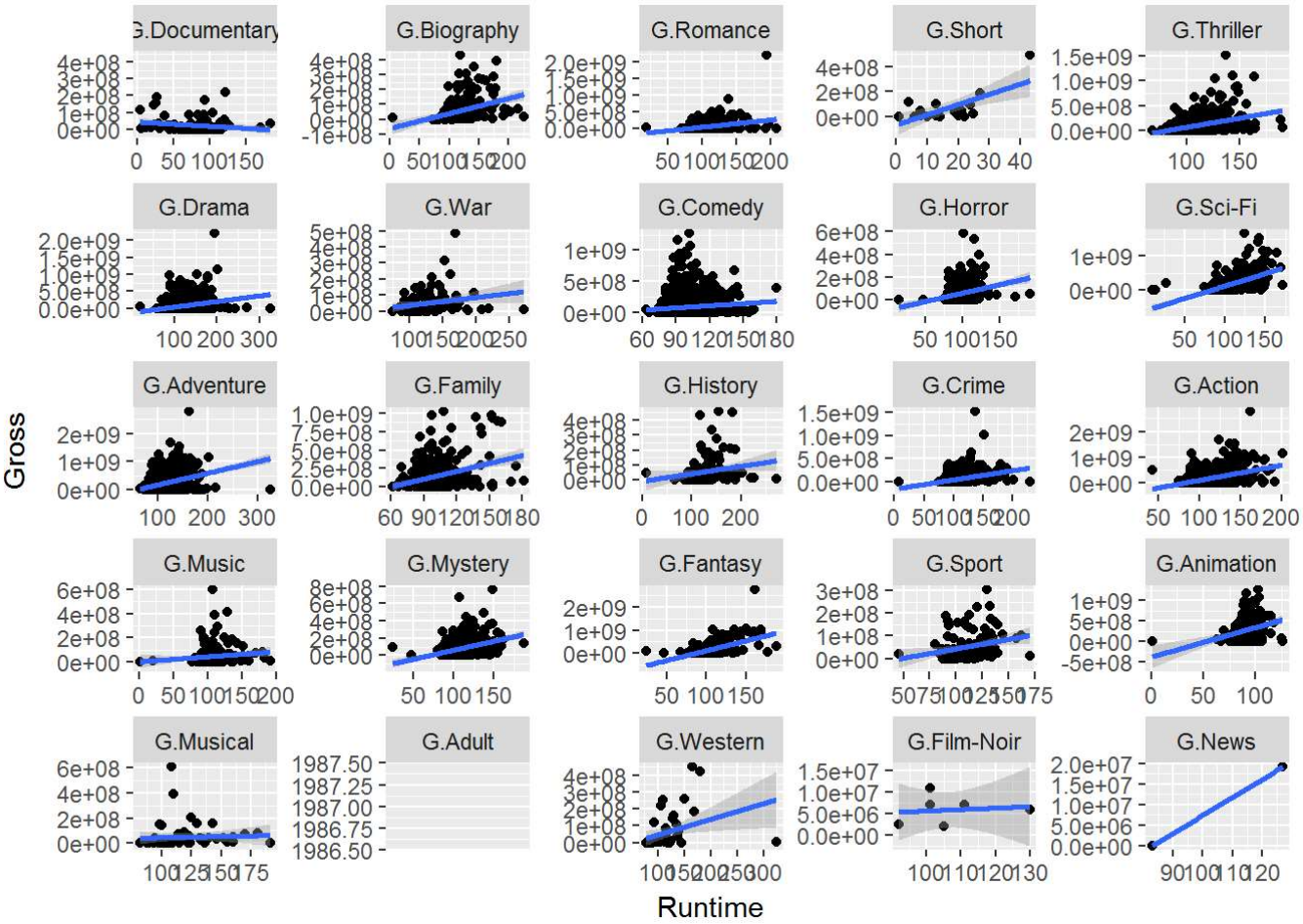



```
plot5.5
```

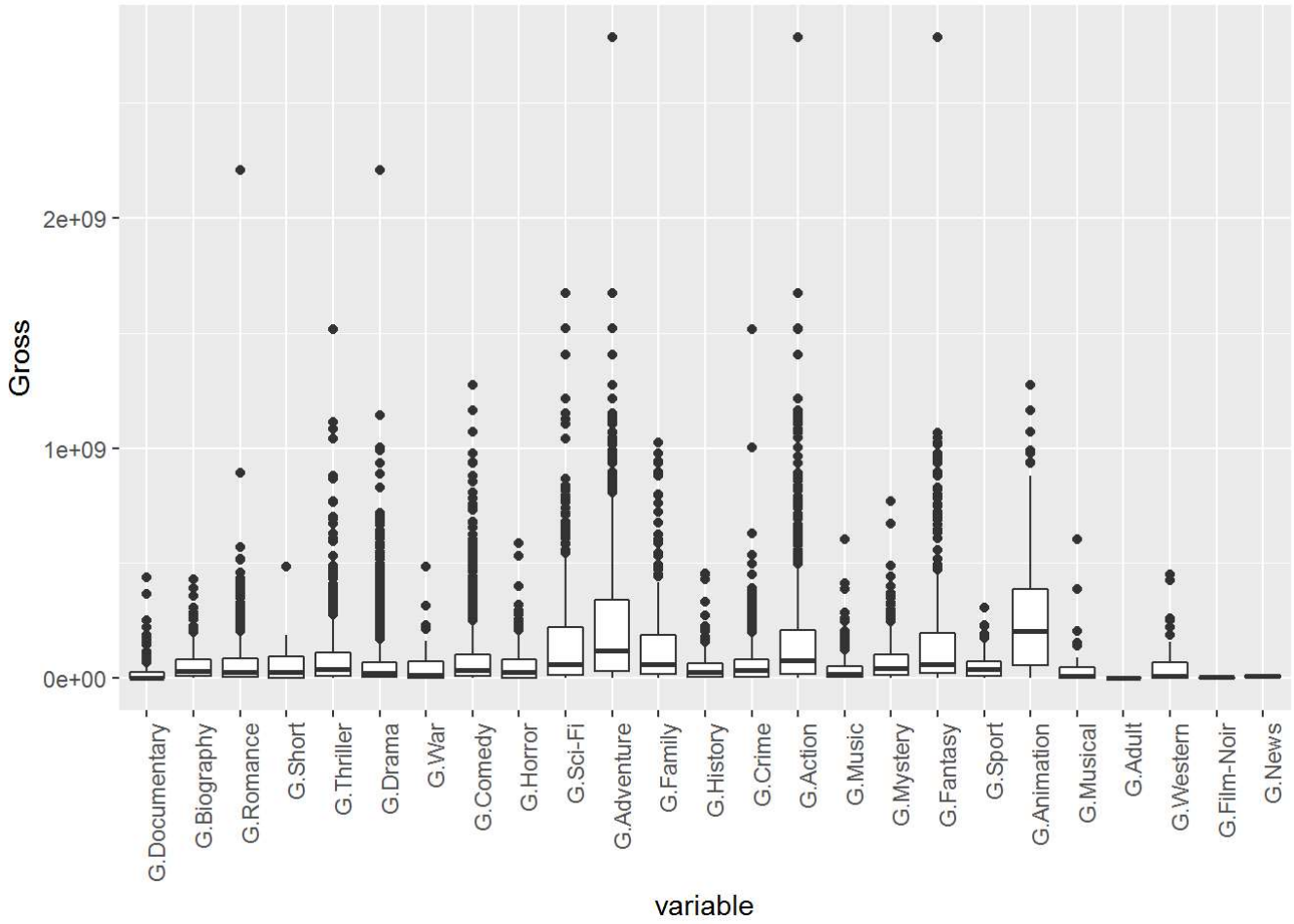
```
## Warning: Removed 63 rows containing non-finite values (stat_smooth).
```

```
## Warning in qt((1 - level)/2, df): NaNs produced
```

```
## Warning: Removed 63 rows containing missing values (geom_point).
```


G.Documentary
G.Biography
G.Romance
G.Short
G.Thriller
G.Drama
G.War
G.Comedy
G.Horror
G.Sci-Fi
G.Adventure
G.Family
G.History
G.Crime
G.Action
G.Music
G.Mystery
G.Fantasy
G.Sport
G.Animation
G.Musical
G.Adult
G.Western
G.Film-Noir
G.News
Gross
Runtime

```
plot5.6
```

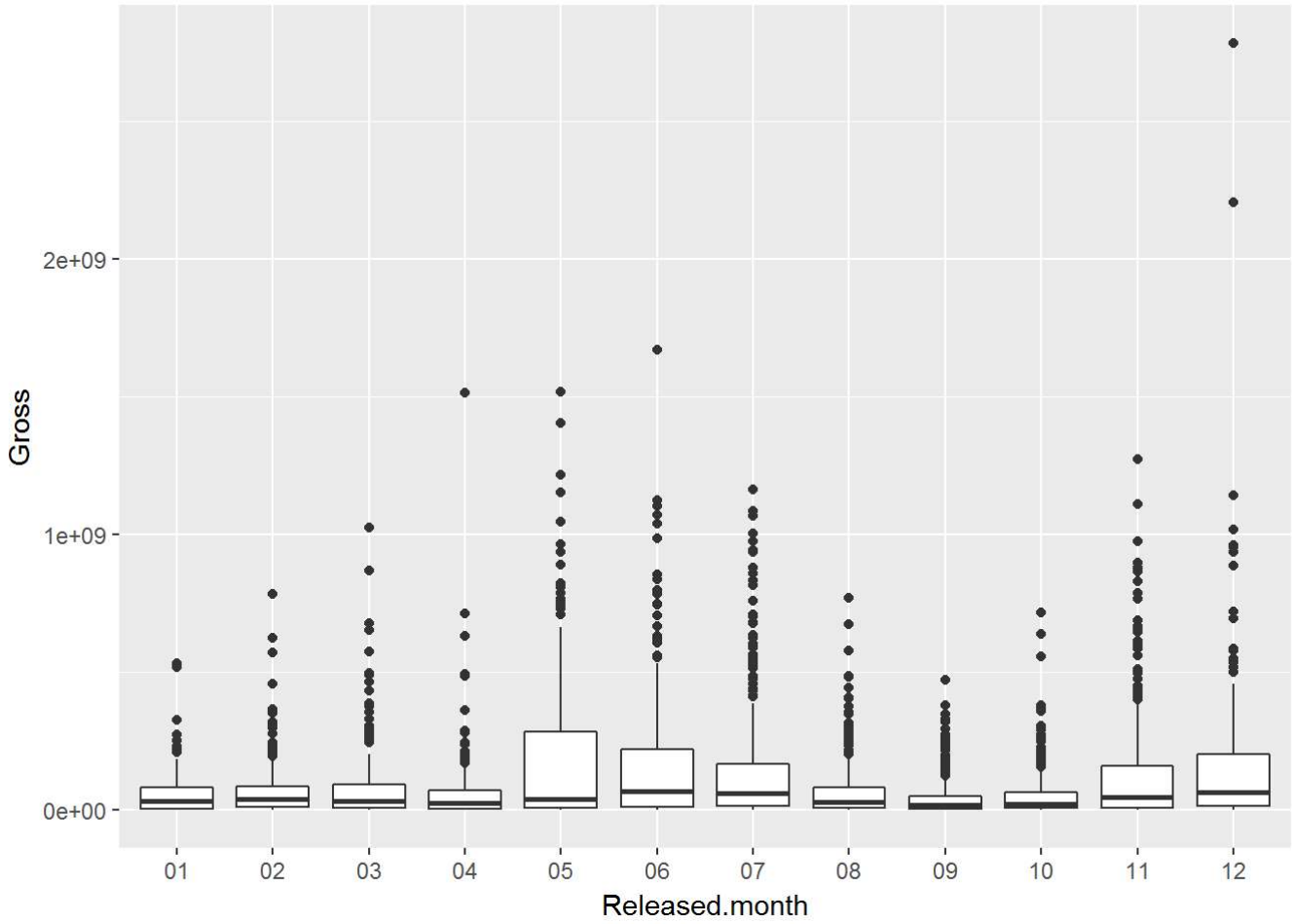



**Q**: Did you find any observable relationships or combinations of Budget/Runtime/Genre that result in high Gross revenue? If you divided the movies into different subsets, you may get different answers for them - point out interesting ones.

**A**: 1. Generally, there are positive correlations between Gross and Budget, or Gross and Runtime. 2. However, for subsets of movies with different Runtime, different correlations are found. For movies with Runtime less than 150 min, there are positive correlations between Gross and Runtime, and there are no or negative correlations for movies with longer Runtime. 3. There is a positive correlation between Gross revenue and Budget in all Genres. High budget movies are more likely have high gross revenue. The solpes are different, which means it is worth to increase budget to some genres (Adventure, Fantasy, Action) of movies to get high gross revenue. 4. For some genres of movies (Sci-Fi, Adventure, Fantasy), there is a positive correlation between Gross Revenue and Runtime, while for some genres of movies (Documentary, War, Musical), the correlation is very week. 5. Animation and Adventure movies have high median Gross revenues, followed by Sci-Fi, Action, Family and Fantasy movies. There are also lots of outliers for each genres of movies, which have very high Gross revenues.

```
# TODO: Investigate if Gross Revenue is related to Release Month
df.new = df[!is.na(df$Released), ]
df.new$Released.month = mapply(function(x) unlist(strsplit(x, "-"))[2], as.character(df.new$Rele
ased))
plot5.4 = ggplot(df.new, aes(Released.month, Gross)) +
  geom_boxplot()
plot5.4
```

```
## Warning: Removed 25581 rows containing non-finite values (stat_boxplot).
```




# 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.

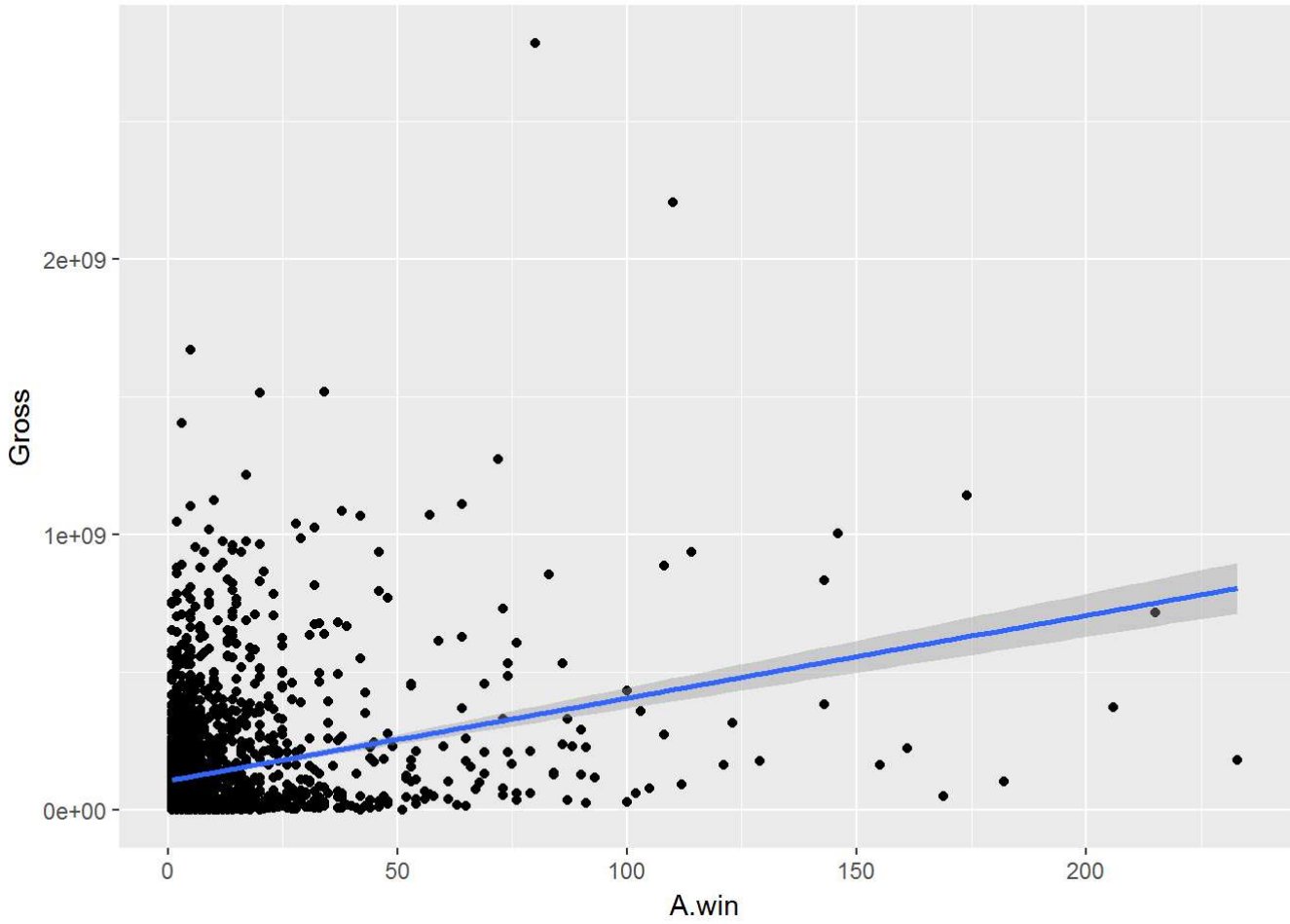
```
# TODO: Convert Awards to 2 numeric columns: wins and nominations
#count wins from a string
countWin = function(str){
  str.list = unlist(strsplit(str, " "))
  m1 = grep("win", str.list)
  return (as.numeric(str.list[m1[1] - 1]))
}

#count nominations from a string
countNomination = function(str){
  str.list = unlist(strsplit(str, " "))
  m2 = grep("nomination", str.list)
  return (as.numeric(str.list[m2[1] - 1]))
}

#add two columns for wins and nominations
df$A.win = mapply(countWin, df$Awards)
df$A.nomination = mapply(countNomination, df$Awards)

# rows had valid wins or nominations
nrow(df[!is.na(df$A.win) | !is.na(df$A.nomination), ])
```

```
## [1] 11947
```

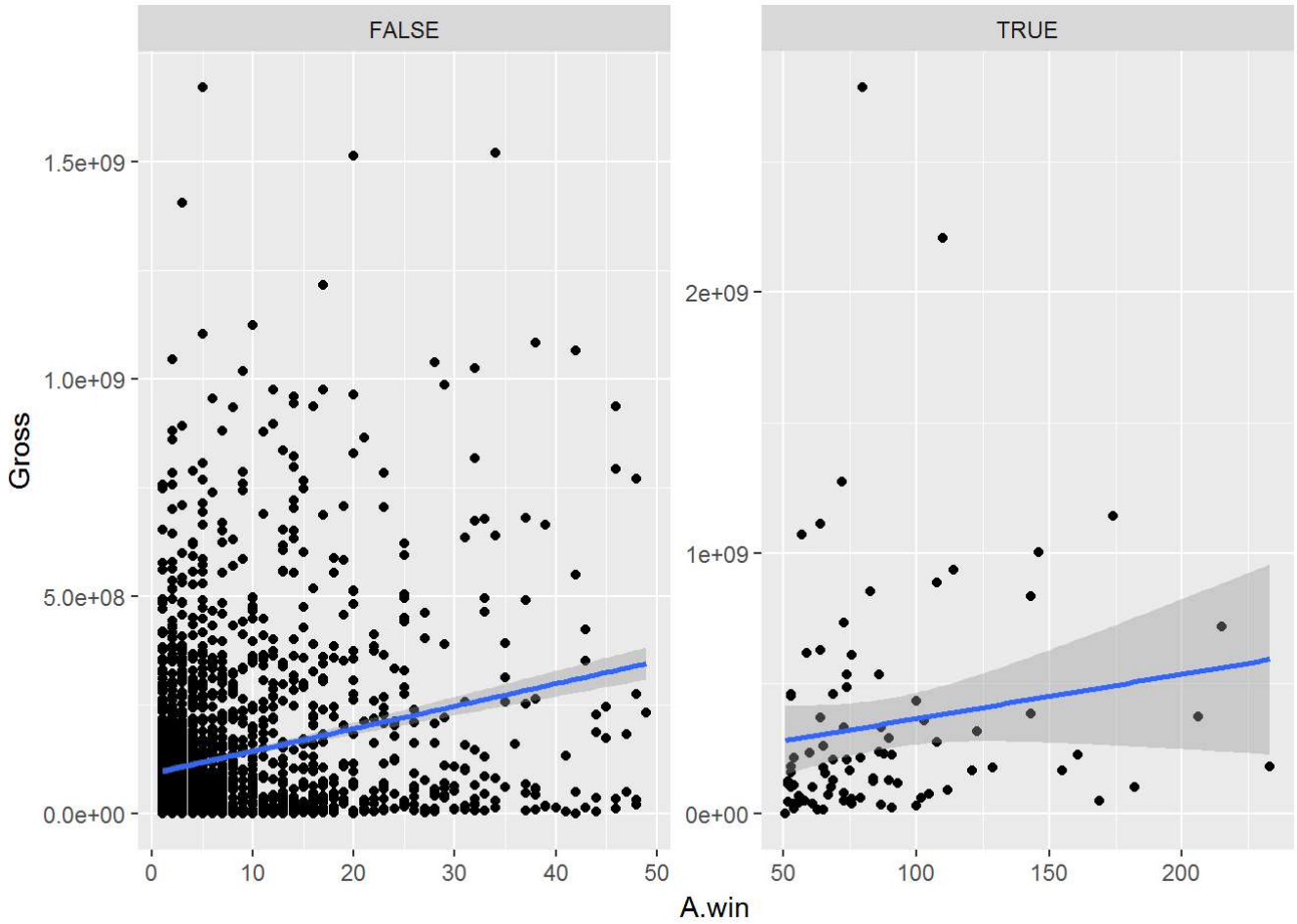
**Q**: How did you construct your conversion mechanism? How many rows had valid/non-zero wins or nominations?

**A**: I create two functions: countWin and countNomination, and mapply them. The mechanisms are simple, I first split the string by space, then search the string “win”. Once find the string, the string before that string is the number of wins. There are 11106 rows had valid/non-zero wins or nominations.

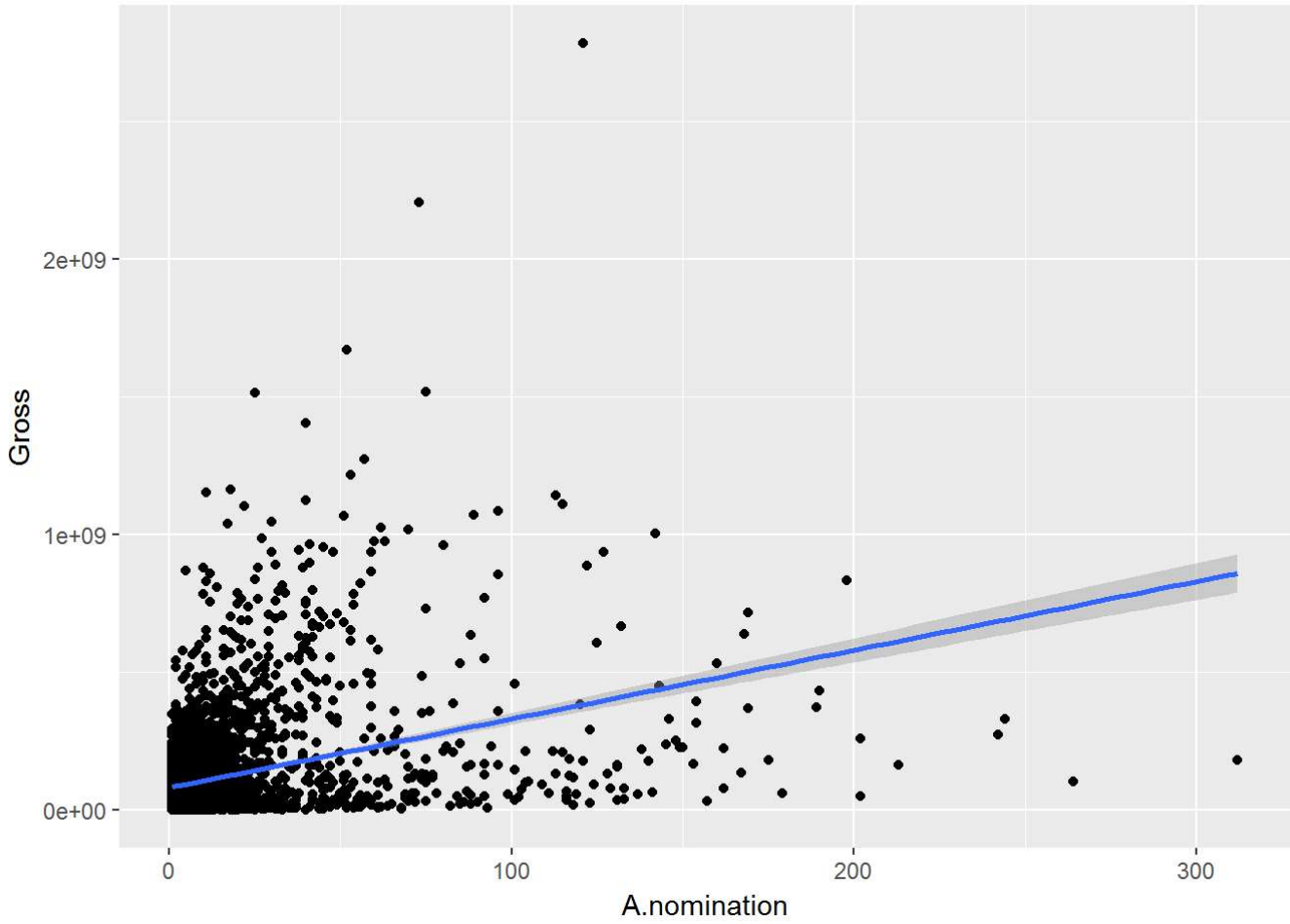
```
# TODO: Plot Gross revenue against wins and nominations
plot6.1 = ggplot(df[!is.na(df$Gross) & !is.na(df$A.win),], aes(A.win, Gross)) +
  geom_point() +
  geom_smooth(method="lm")

plot6.2 = ggplot(df[!is.na(df$Gross) & !is.na(df$A.win),], aes(A.win, Gross)) +
  geom_point() +
  facet_wrap(~(A.win > 50), scales="free") +
  geom_smooth(method="lm")

plot6.3 = ggplot(df[!is.na(df$Gross) & !is.na(df$A.nomination),], aes(A.nomination, Gross)) +
  geom_point() +
  geom_smooth(method="lm")

plot6.4 = ggplot(df[!is.na(df$Gross) & !is.na(df$A.nomination),], aes(A.nomination, Gross)) +
  geom_point() +
  facet_wrap(~(A.nomination > 100), scales="free") +
  geom_smooth(method="lm")

plot6.1
```



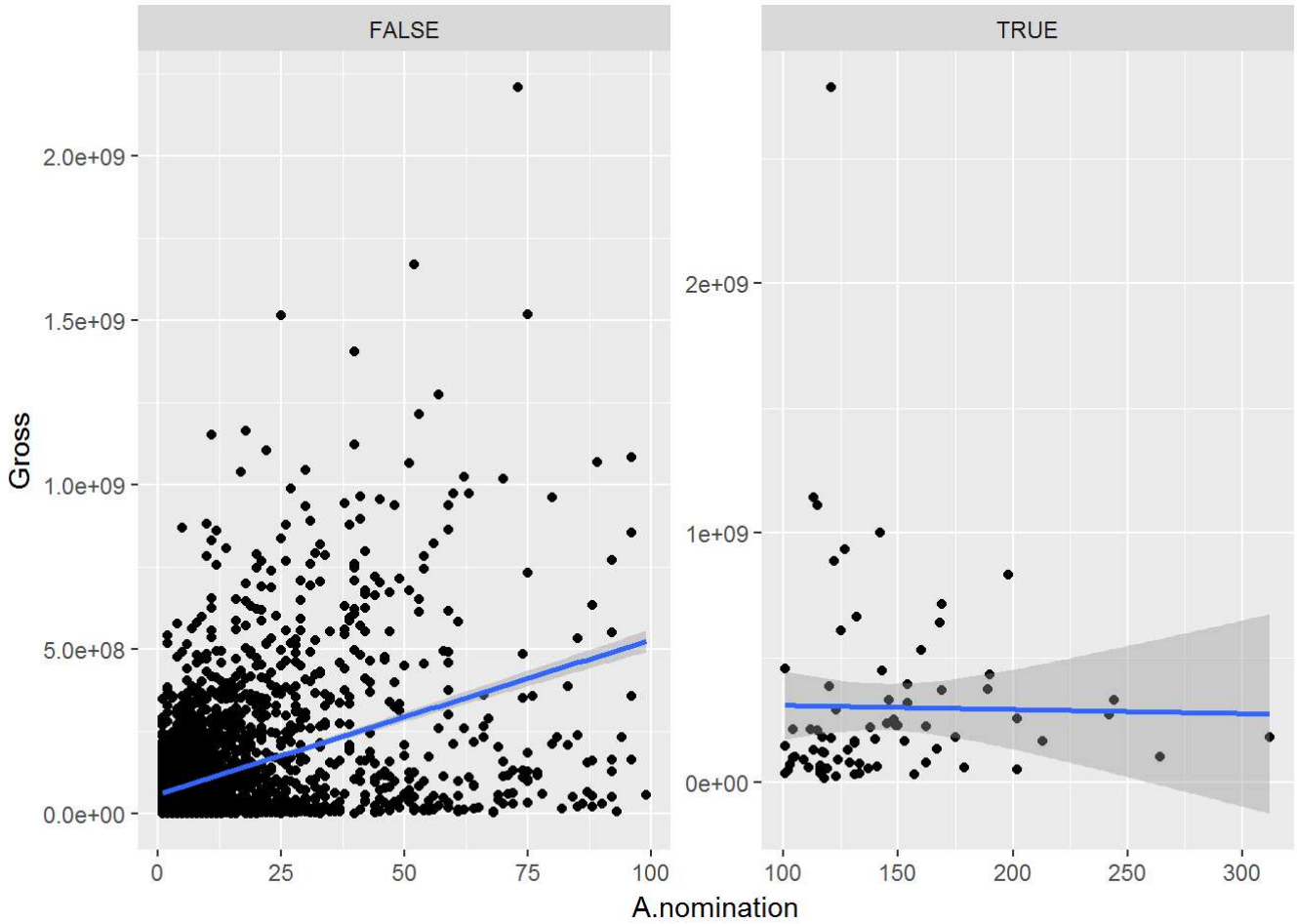

```
plot6.2
```


FALSE
TRUE
Gross
1.5e+09
1.0e+09
5.0e+08
0.0e+00
2e+09
1e+09
0e+00
0
10
20
30
40
50
50
100
150
200
A.win

```
plot6.3
```




```
plot6.4
```




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

**A**: The scatter plots show movies with more wins and nominations have higher gross revenues. However, this correlation decrease when wins > 50 or nominations > 100.

# 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)).

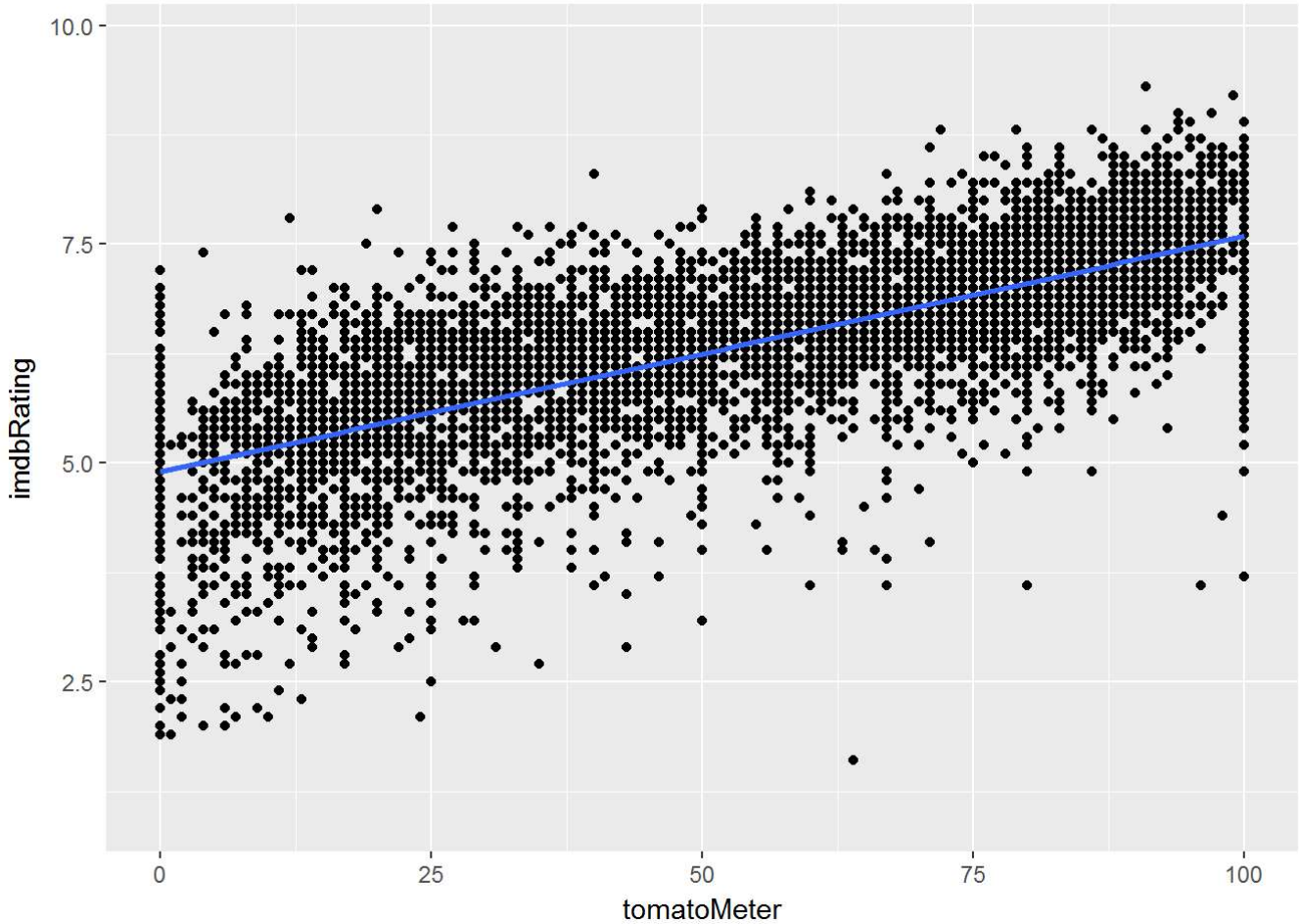
Investigate the pairwise relationships between these different descriptors using graphs.

```
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
#imdbRating vs tomatoMeter
plot7.1 = ggplot(df, aes(tomatoMeter, imdbRating)) +
  geom_point() +
  geom_smooth(method="lm")

#imdbRating vs tomatoRating
plot7.2 = ggplot(df, aes(tomatoRating, imdbRating)) +
  geom_point() +
  geom_smooth(method="lm")

#imdbRating vs tomatoRotten
plot7.3 = ggplot(df, aes(tomatoRotten, imdbRating)) +
  geom_point() +
  geom_smooth(method="lm")

#imdbRating vs tomatoFresh
plot7.4 = ggplot(df, aes(tomatoFresh, imdbRating)) +
  geom_point() +
  geom_smooth(method="lm")

#imdbRating vs tomatoImage
plot7.5 = ggplot(df, aes(tomatoImage, imdbRating)) +
  geom_boxplot()


#imdbRating vs tomatoUserMeter
plot7.6 = ggplot(df, aes(tomatoUserMeter, imdbRating)) +
  geom_point() +
  geom_smooth(method="lm")

#imdbRating vs tomatoUserRating
plot7.7 = ggplot(df, aes(tomatoUserRating, imdbRating)) +
  geom_point() +
  geom_smooth(method="lm")

#imdbRating histogram
plot7.8 = ggplot(df, aes(imdbRating)) +
  geom_histogram()

plot7.9 = ggplot(df, aes(tomatoMeter)) +
  geom_histogram()

plot7.10 = ggplot(df, aes(tomatoRating)) +
  geom_histogram()

plot7.11 = ggplot(df, aes(tomatoFresh)) +
  geom_histogram()

plot7.12 = ggplot(df, aes(tomatoRotten)) +
  geom_histogram()
```

```
plot7.1
```

```
## Warning: Removed 26800 rows containing non-finite values (stat_smooth).
```

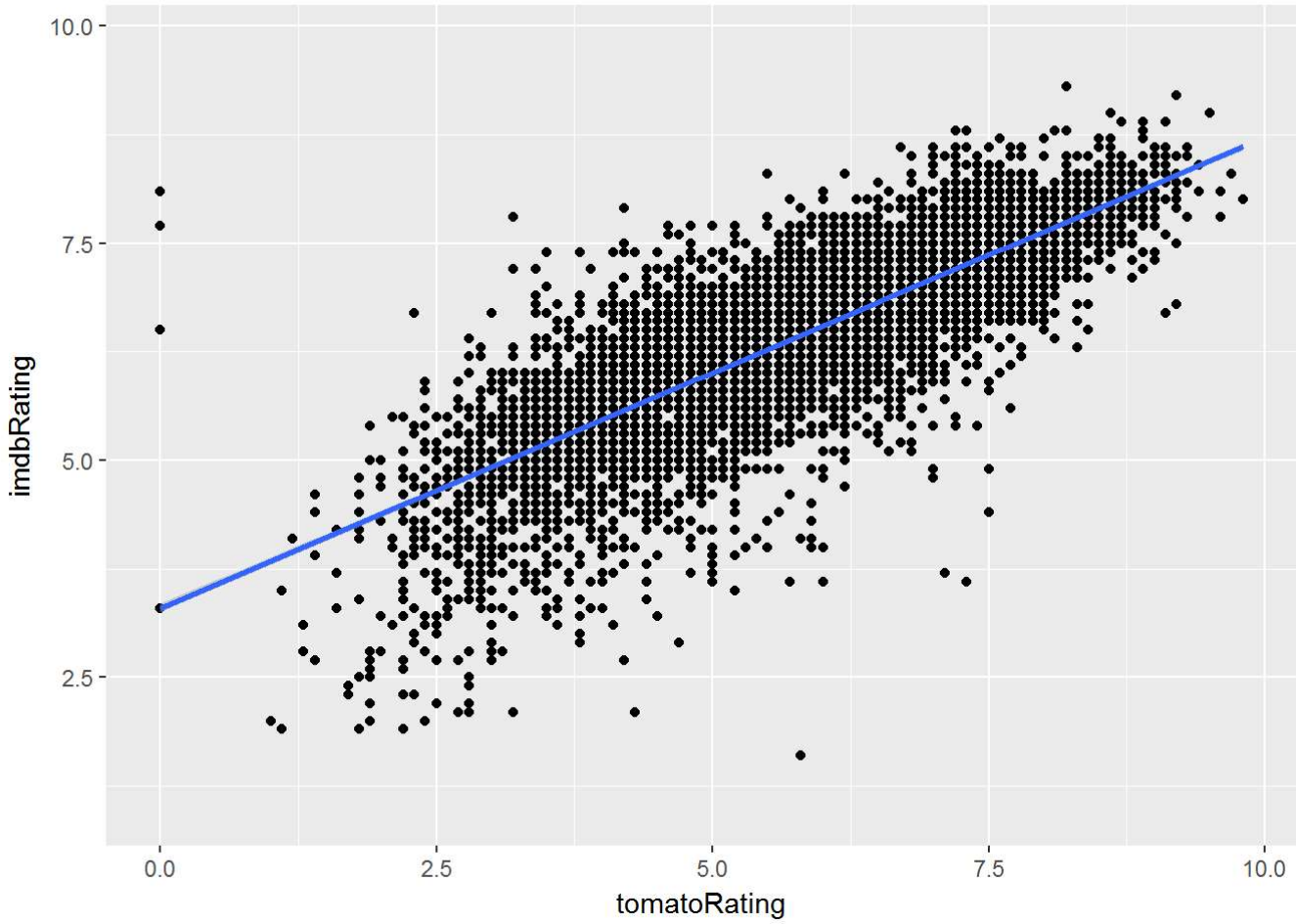
```
## Warning: Removed 26800 rows containing missing values (geom_point).
```




```
plot7.2
```

```
## Warning: Removed 26807 rows containing non-finite values (stat_smooth).
```

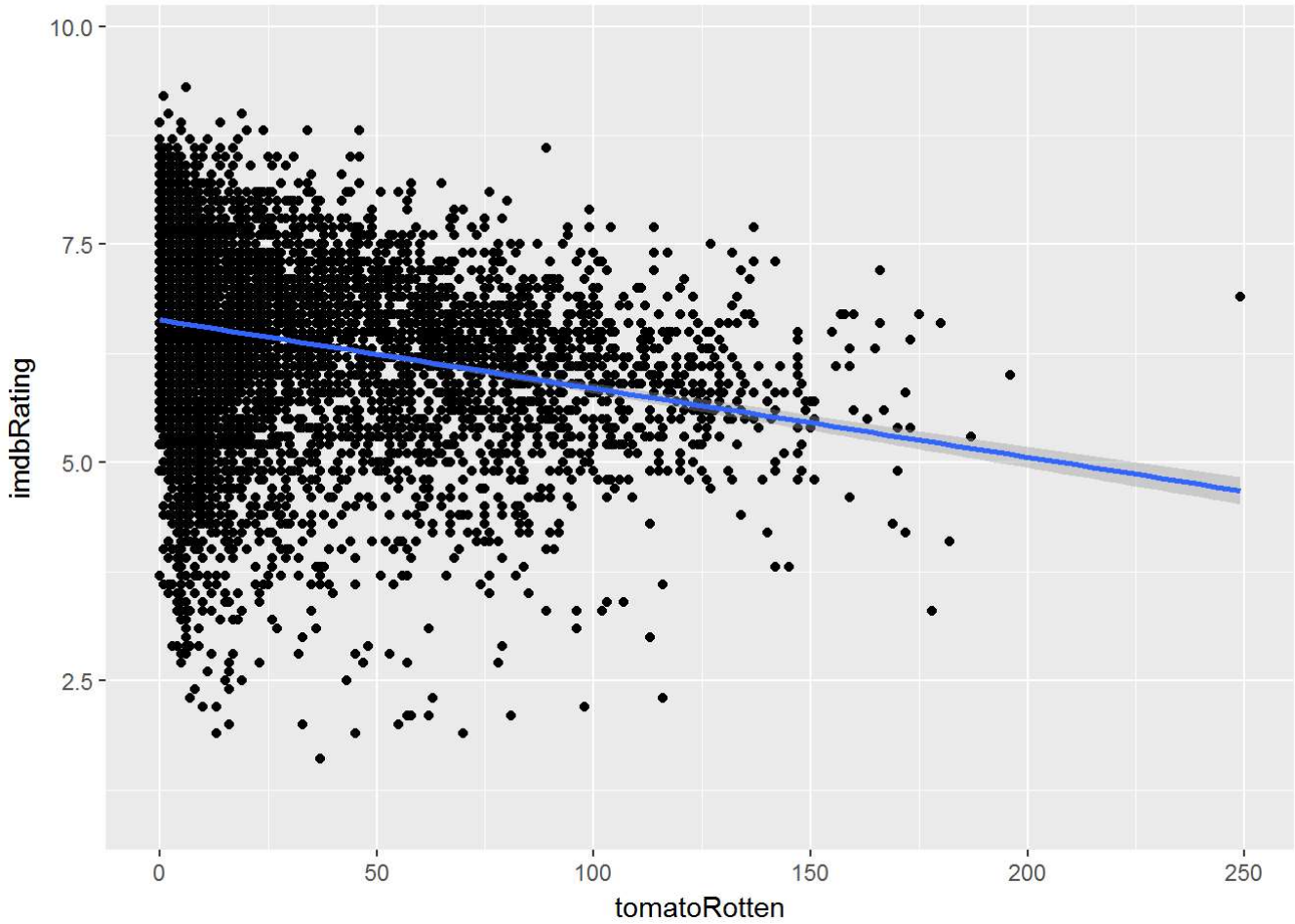
```
## Warning: Removed 26807 rows containing missing values (geom_point).
```




```
plot7.3
```

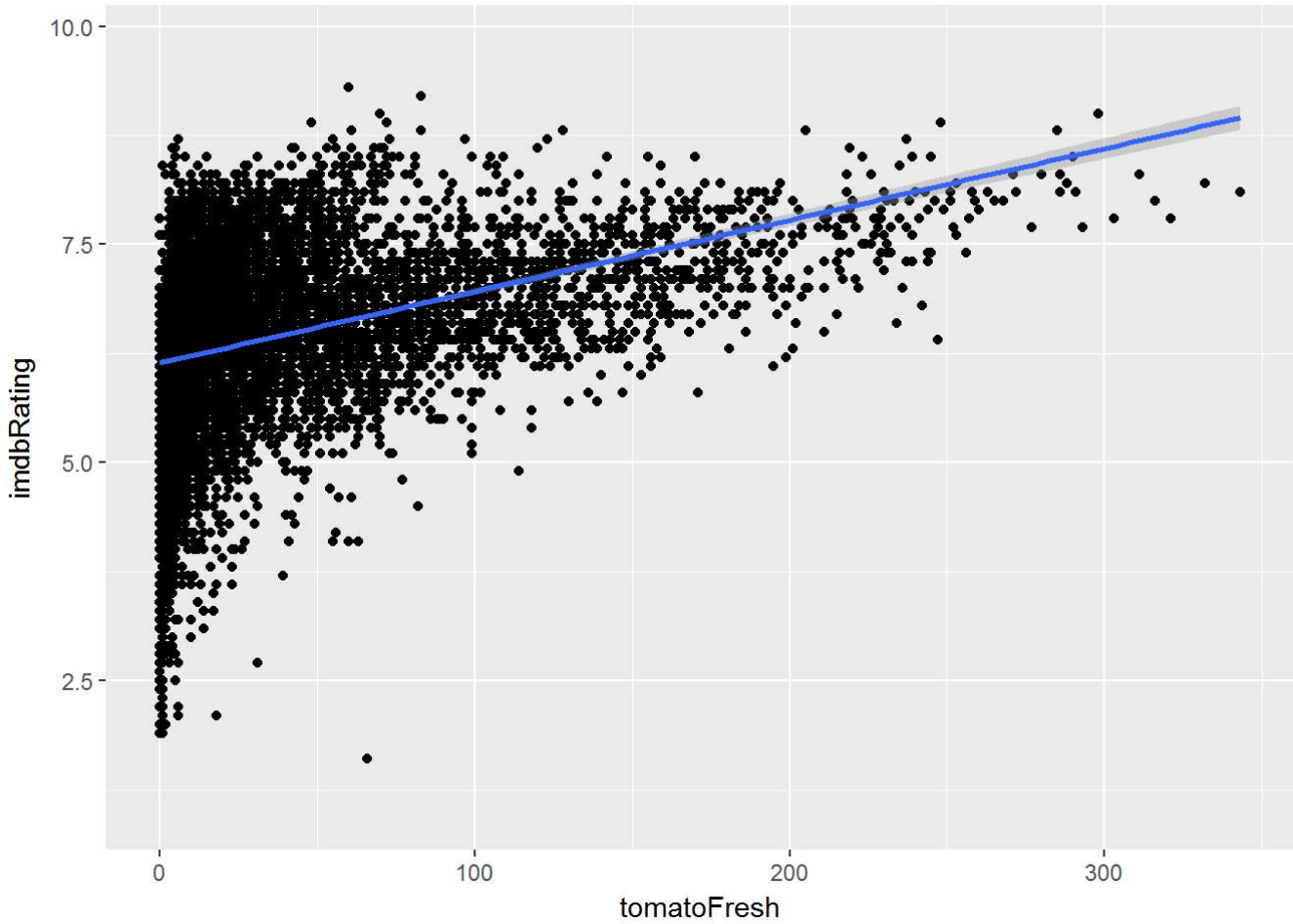
```
## Warning: Removed 26765 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 26765 rows containing missing values (geom_point).
```



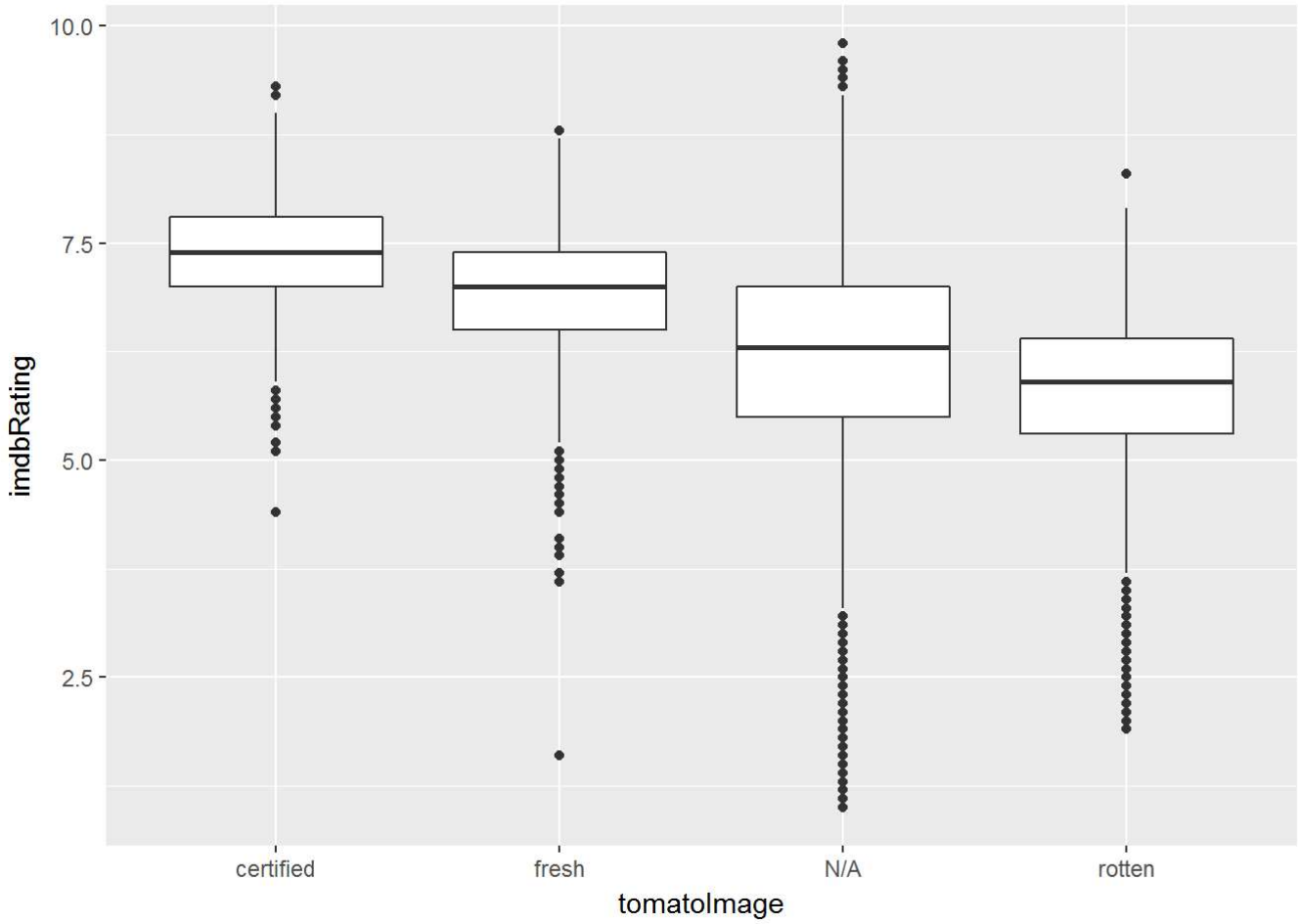

```
plot7.4
```

```
## Warning: Removed 26765 rows containing non-finite values (stat_smooth).

## Warning: Removed 26765 rows containing missing values (geom_point).
```




```
plot7.5
```

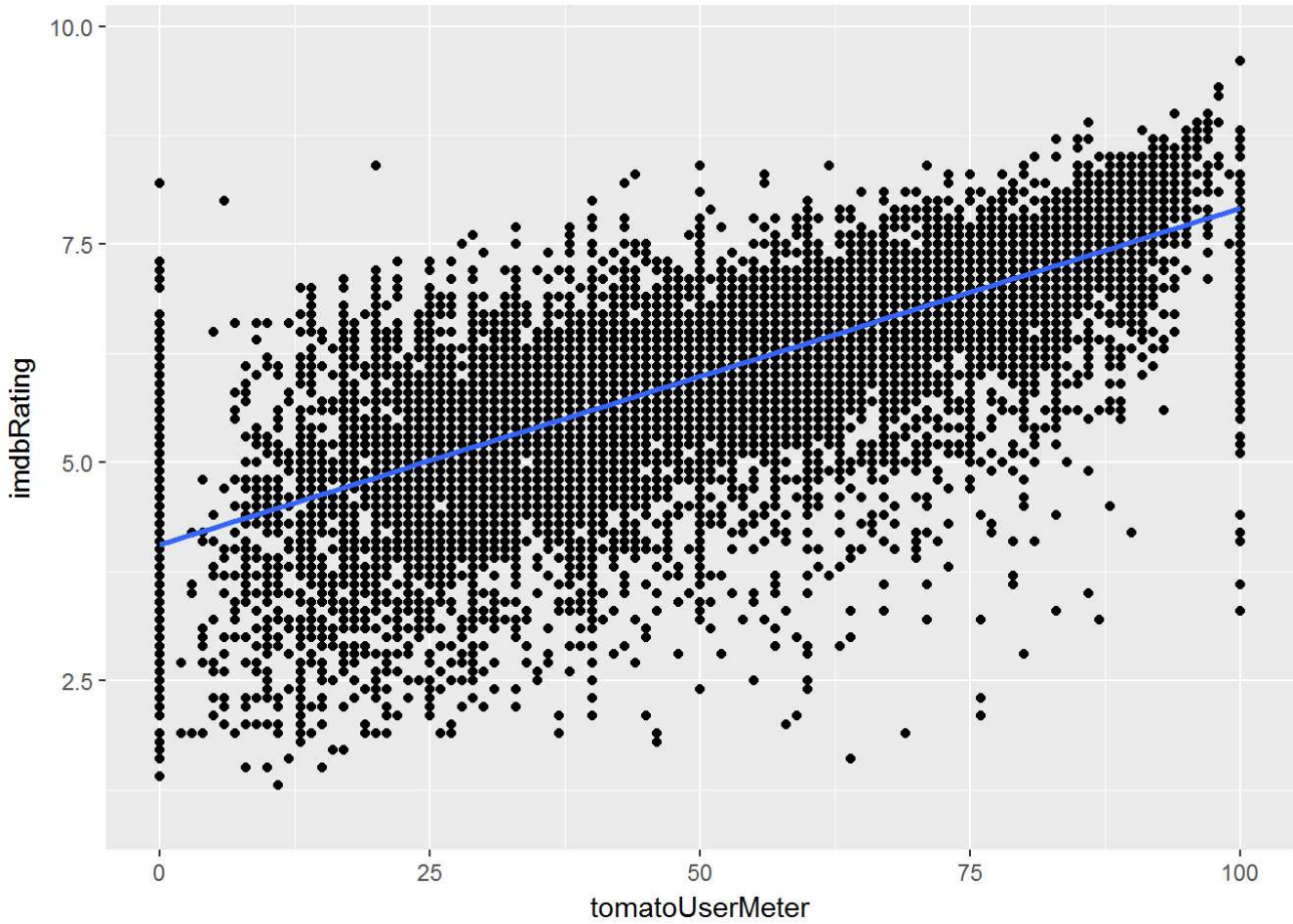
```
## Warning: Removed 1123 rows containing non-finite values (stat_boxplot).
```




```
plot7.6
```

```
## Warning: Removed 19744 rows containing non-finite values (stat_smooth).
```

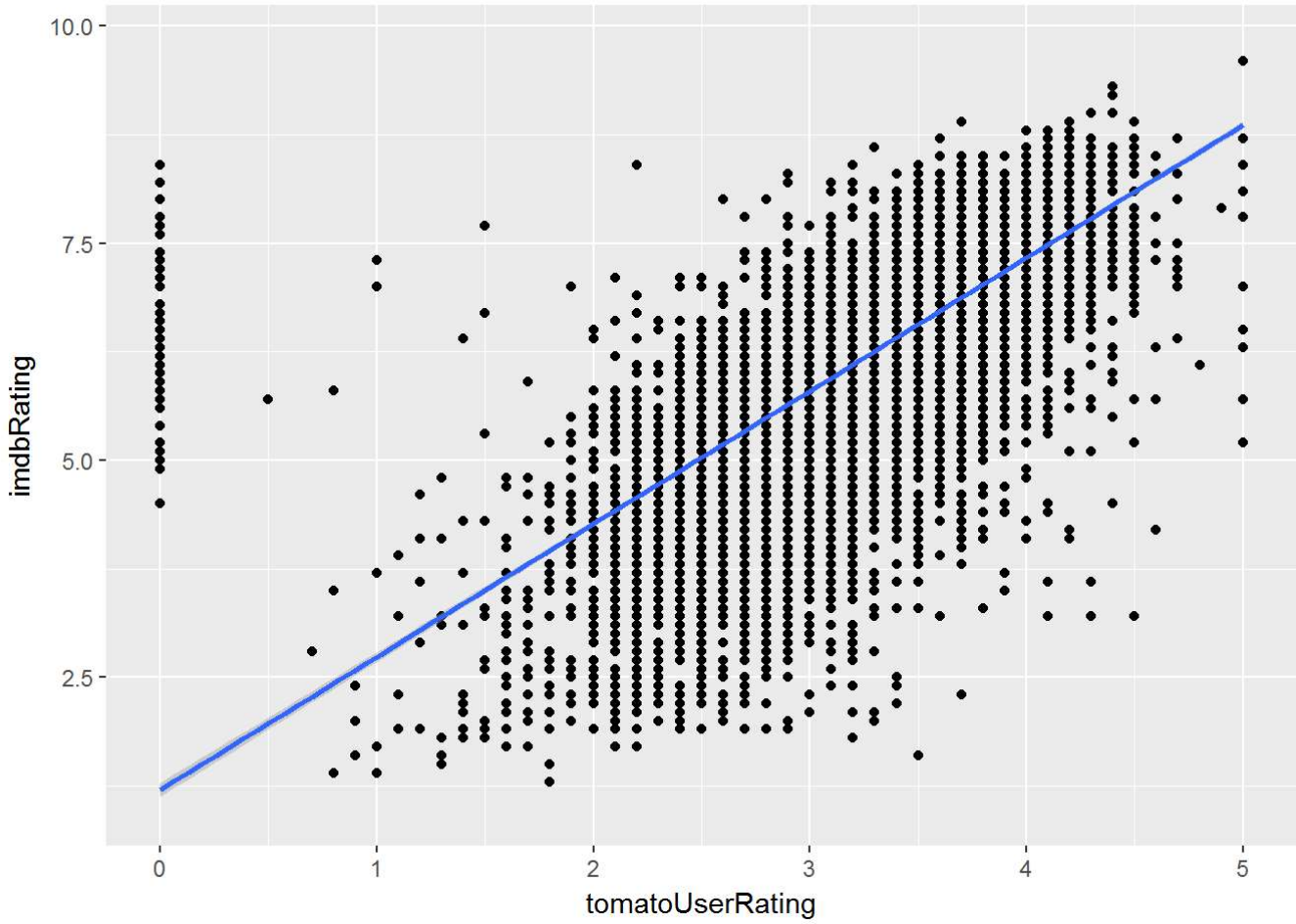
```
## Warning: Removed 19744 rows containing missing values (geom_point).
```




```
plot7.7
```

```
## Warning: Removed 19675 rows containing non-finite values (stat_smooth).
```

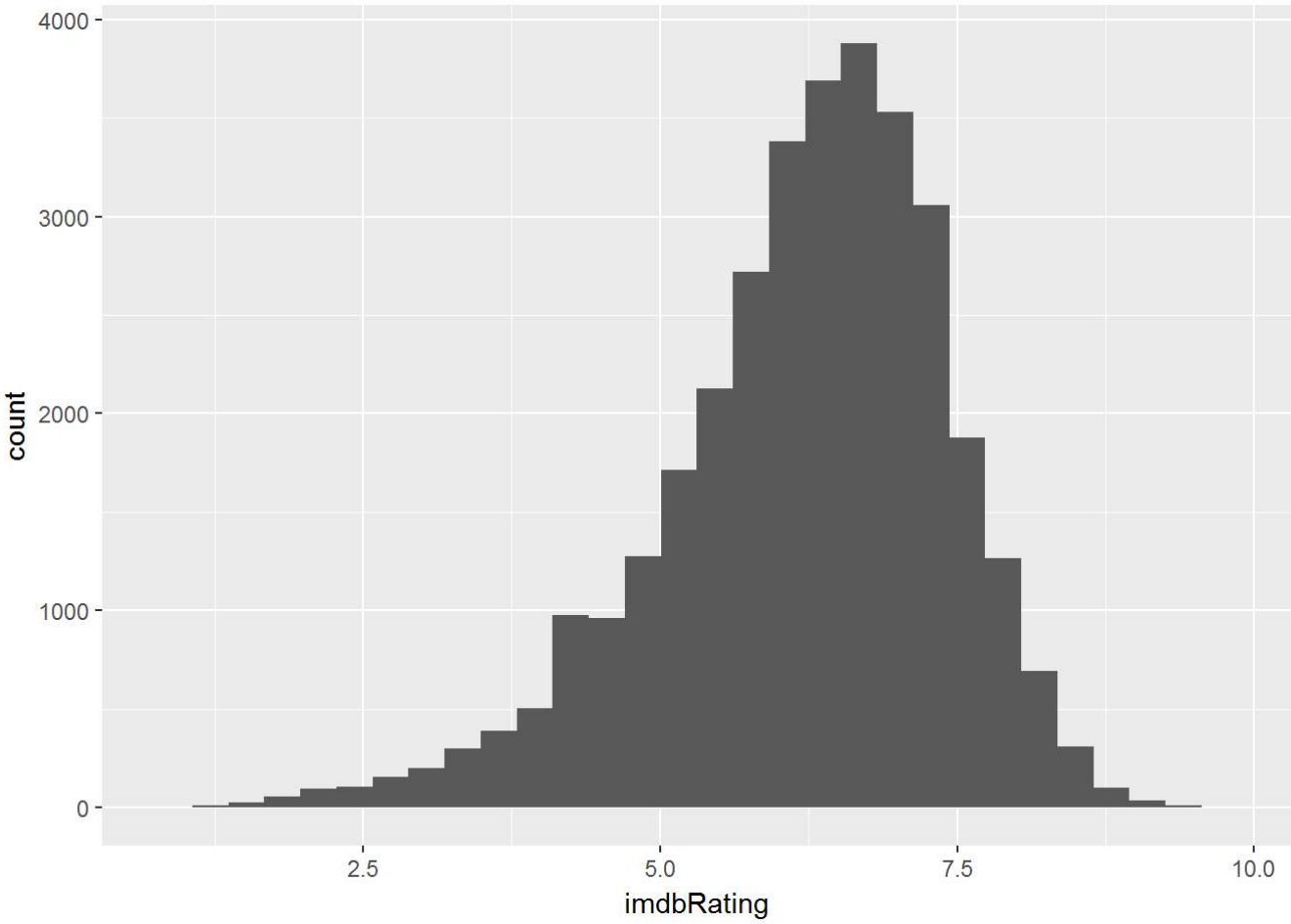
```
## Warning: Removed 19675 rows containing missing values (geom_point).
```




```
plot7.8
```

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

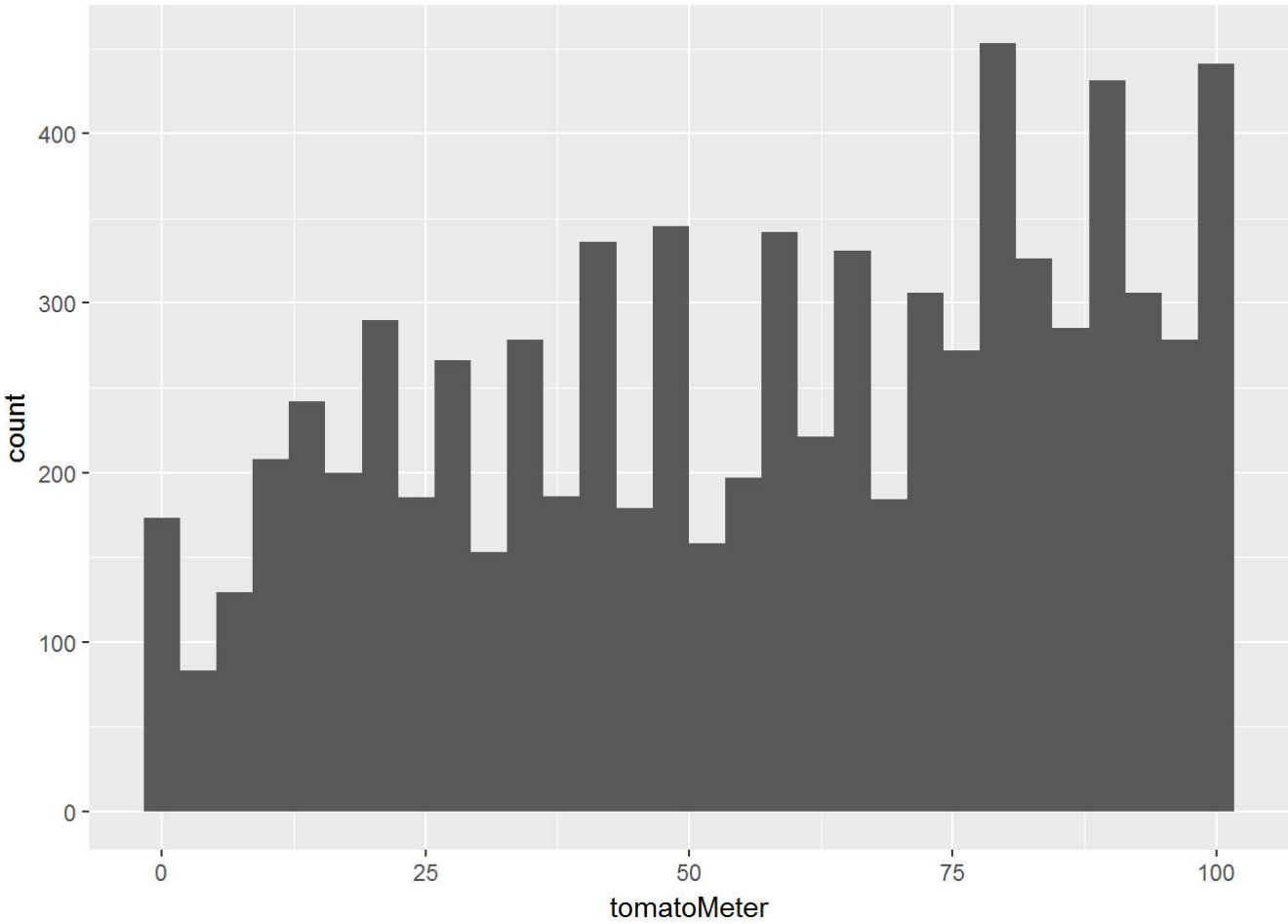
```
## Warning: Removed 1123 rows containing non-finite values (stat_bin).
```




```
plot7.9
```

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

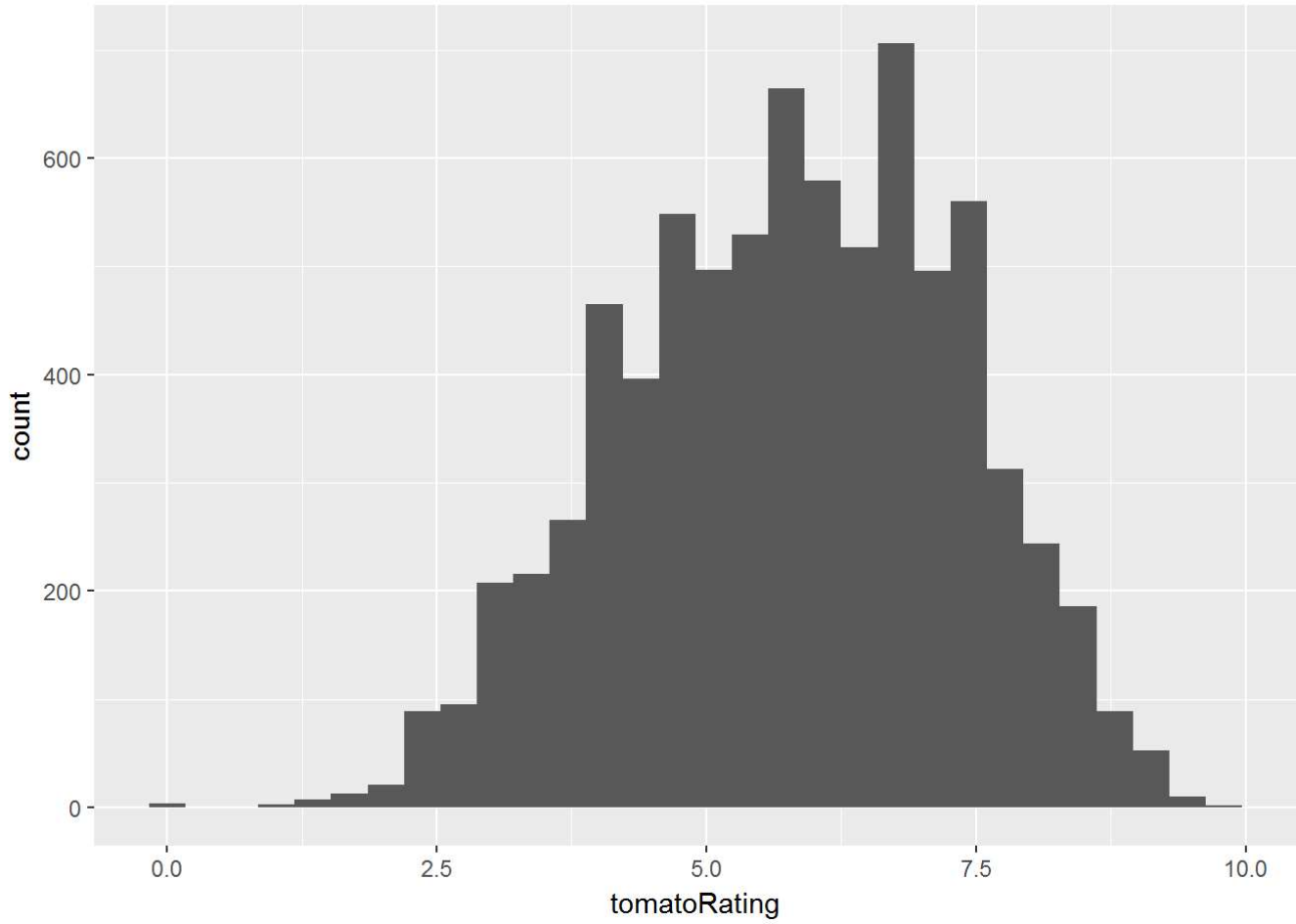
```
## Warning: Removed 26798 rows containing non-finite values (stat_bin).
```




```
plot7.10
```

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

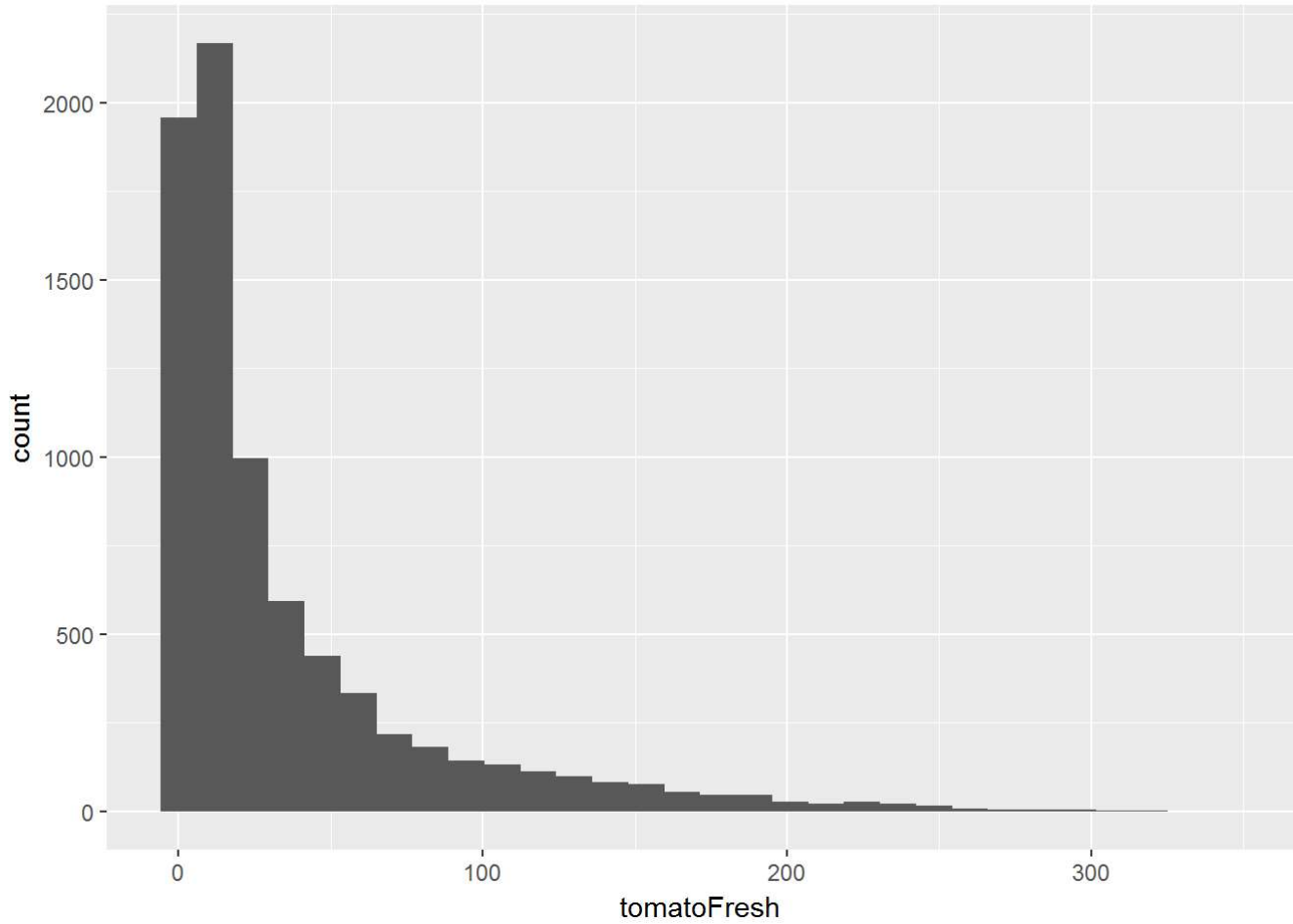
```
## Warning: Removed 26805 rows containing non-finite values (stat_bin).
```




```
plot7.11
```

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

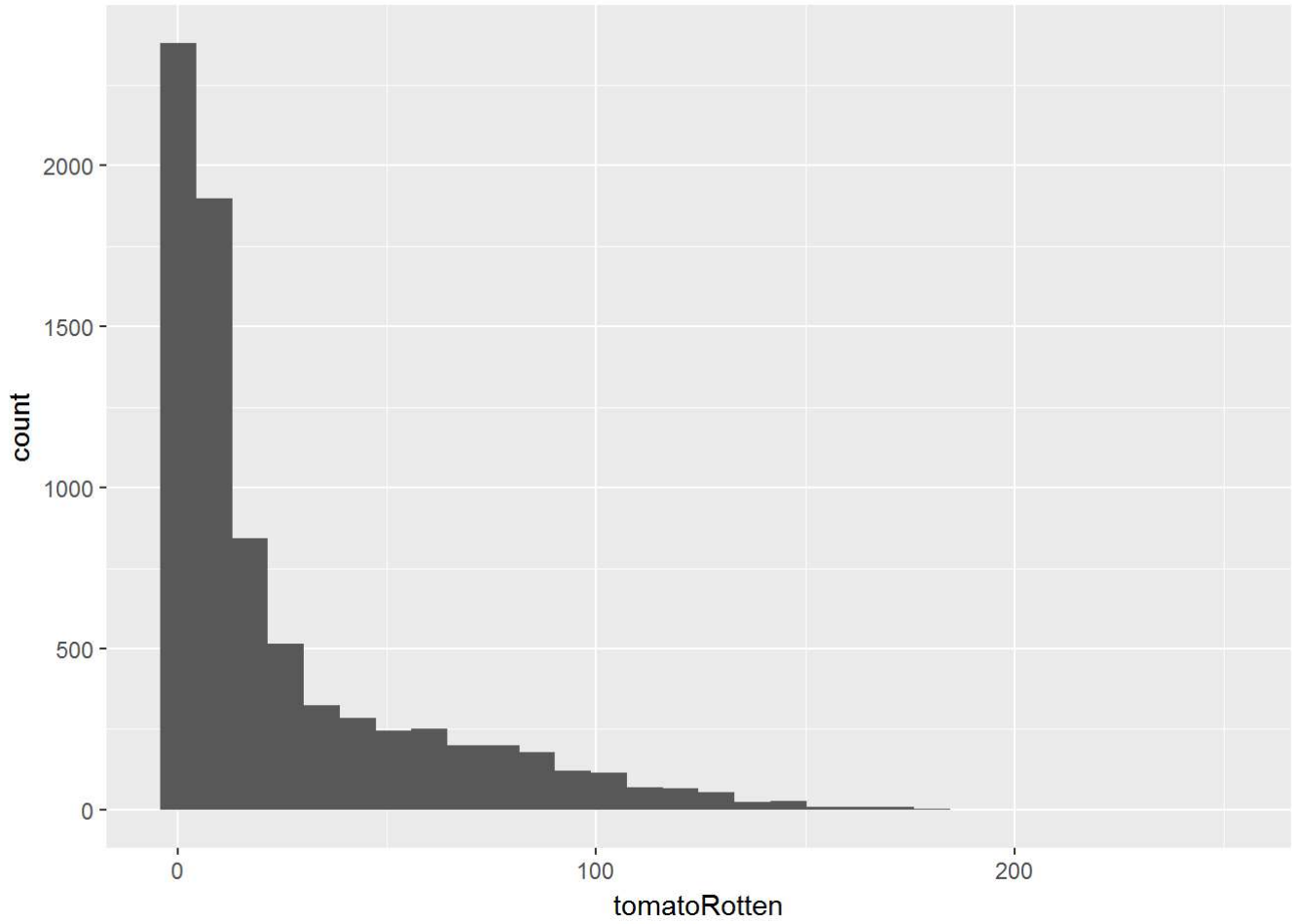
```
## Warning: Removed 26763 rows containing non-finite values (stat_bin).
```




```
plot7.12
```

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

```
## Warning: Removed 26763 rows containing non-finite values (stat_bin).
```




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

**A**: Similarities: there are positive correlations between imdbRating and tomatoMeter,tomatoRating, tomatoUserMeter, tomatoUserRating. The imdbRating is positive correlated with freshTomato, but negative correlated with tomatoRotten. The box plot of imdbRating vs tomatoImage shows certified movies have highest imdbRating, followed by fress, N/A, and rotten.

Difference: using histogram, imdbRating shows a left skewed gaussian distribution with peak at 6.2. the tomatoMeter is more like a uniform distribution. Both tomatoFresh and tomatoRotten have perks at zero, and long tails at right.

# 8. Ratings and awards

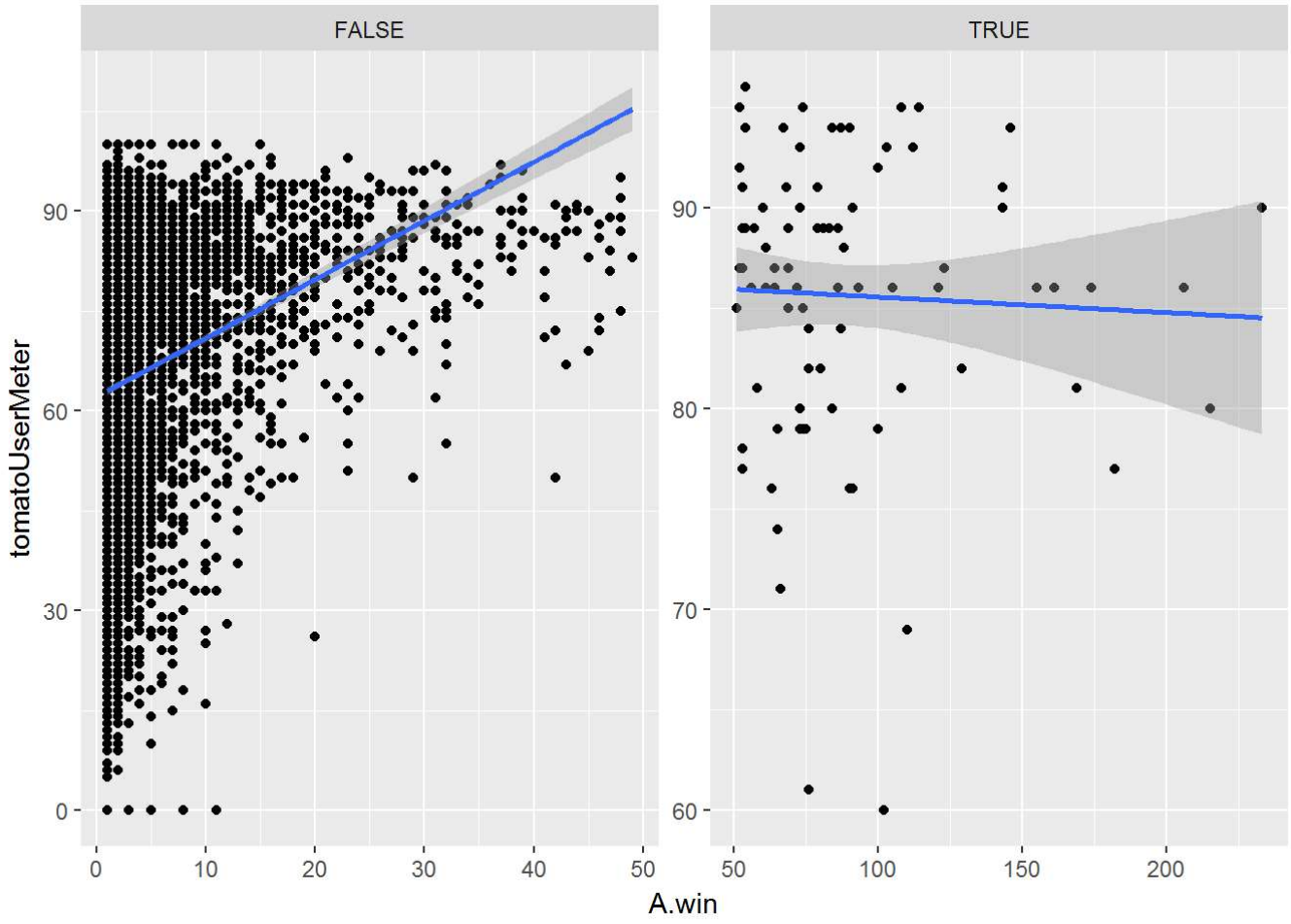
These ratings typically reflect the general appeal of the movie to the public or gather opinions from a larger body of critics. Whereas awards are given by professional societies that may evaluate a movie on specific attributes, such as artistic performance, screenplay, sound design, etc.

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
plot8.1 = ggplot(df[!is.na(df$A.win),], aes(A.win, tomatoUserMeter)) +
  geom_point() +
  facet_wrap(~(A.win>50), scales="free") +
  geom_smooth(method="lm")

plot8.2 = ggplot(df, aes(factor(!is.na(A.win)), tomatoUserMeter)) +
  geom_boxplot()

plot8.3 = ggplot(df[!is.na(df$A.nomination), ], aes(A.nomination, tomatoUserMeter)) +
  geom_point() +
  facet_wrap(~(A.nomination>100), scales="free") +
  geom_smooth(method="lm")

plot8.4 = ggplot(df, aes(factor(!is.na(A.nomination)), tomatoUserMeter)) +
  geom_boxplot()


plot8.1
```

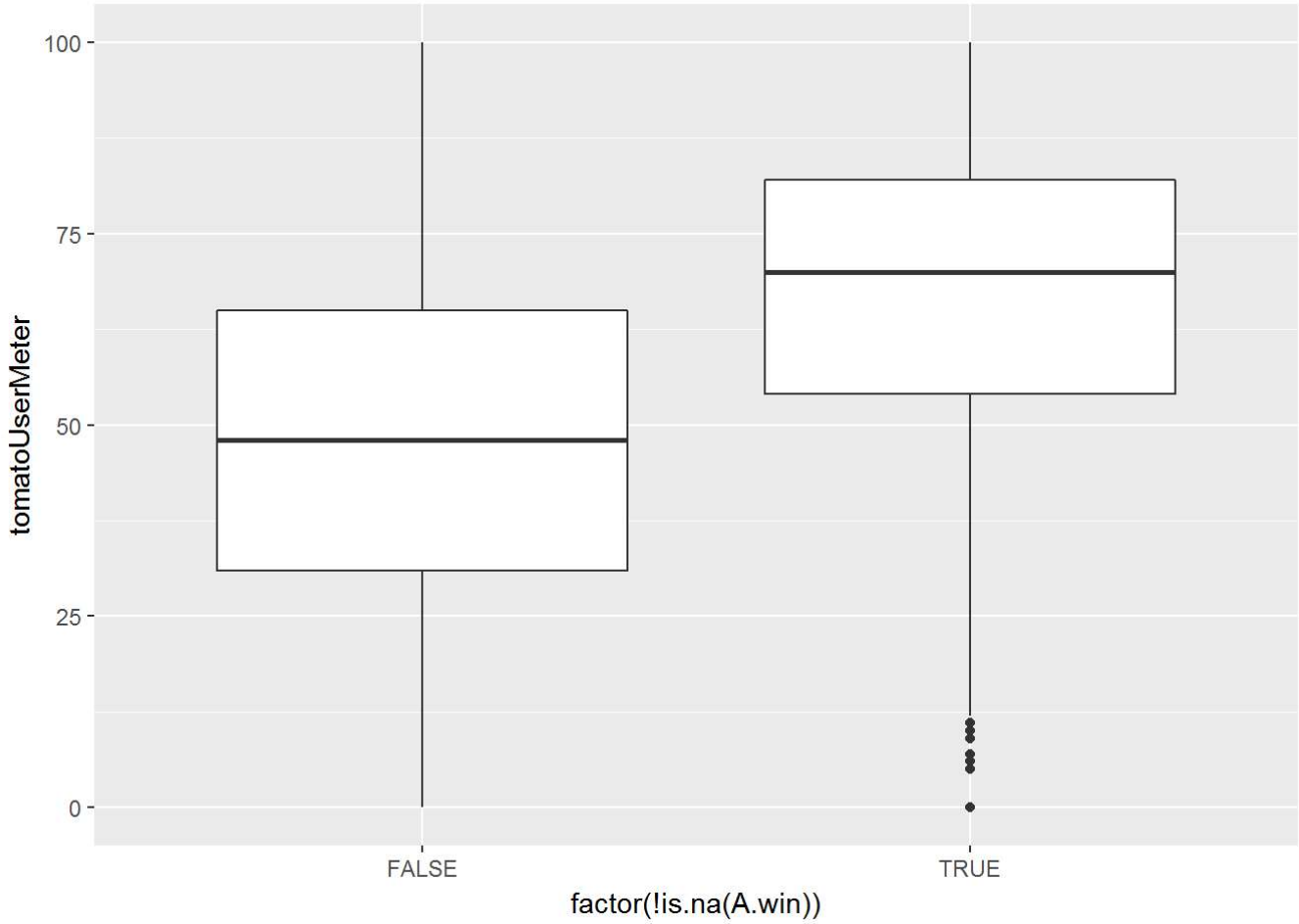
```
## Warning: Removed 3062 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 3062 rows containing missing values (geom_point).
```



```
plot8.2
```

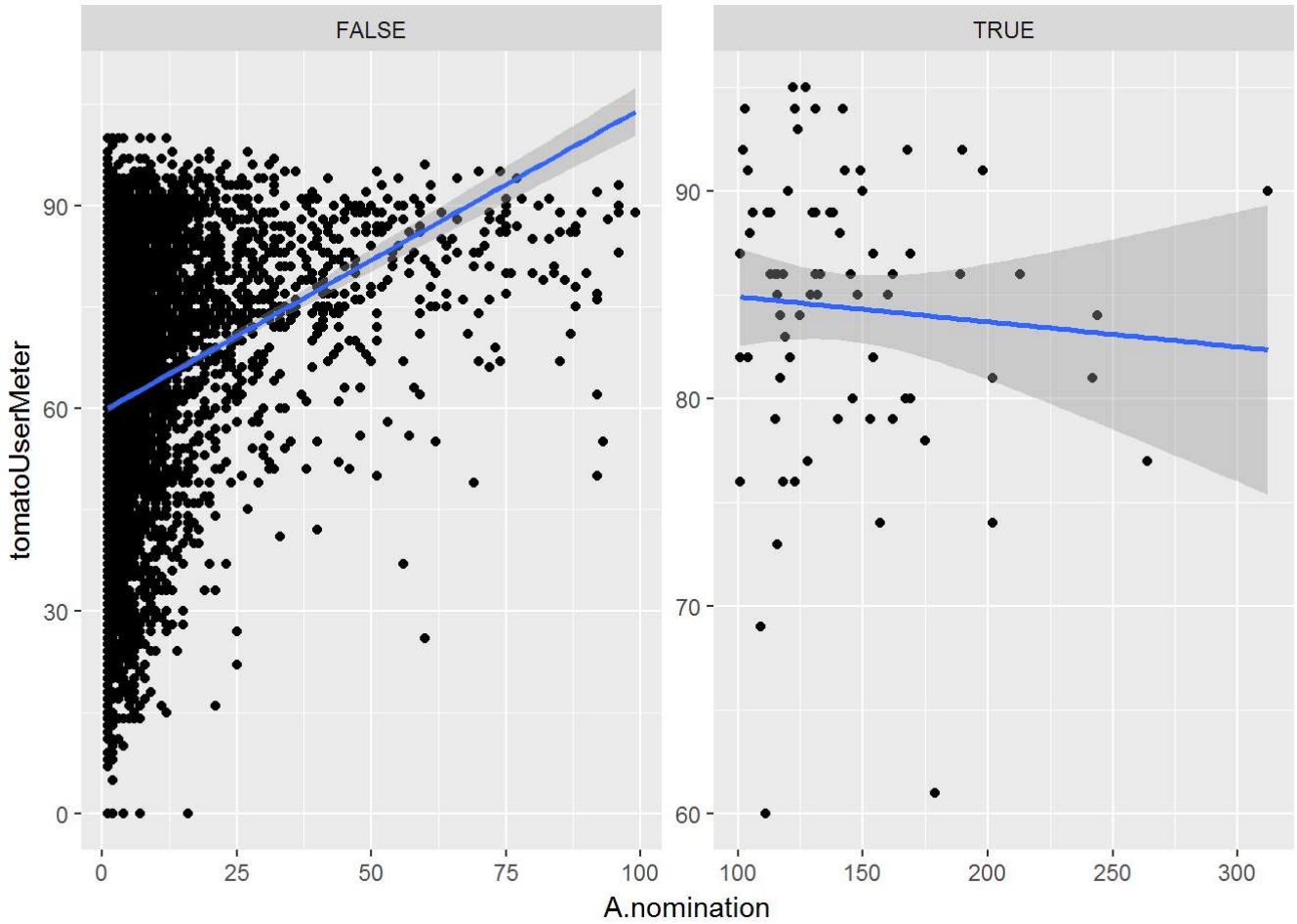
```
## Warning: Removed 19740 rows containing non-finite values (stat_boxplot).
```




```
plot8.3
```

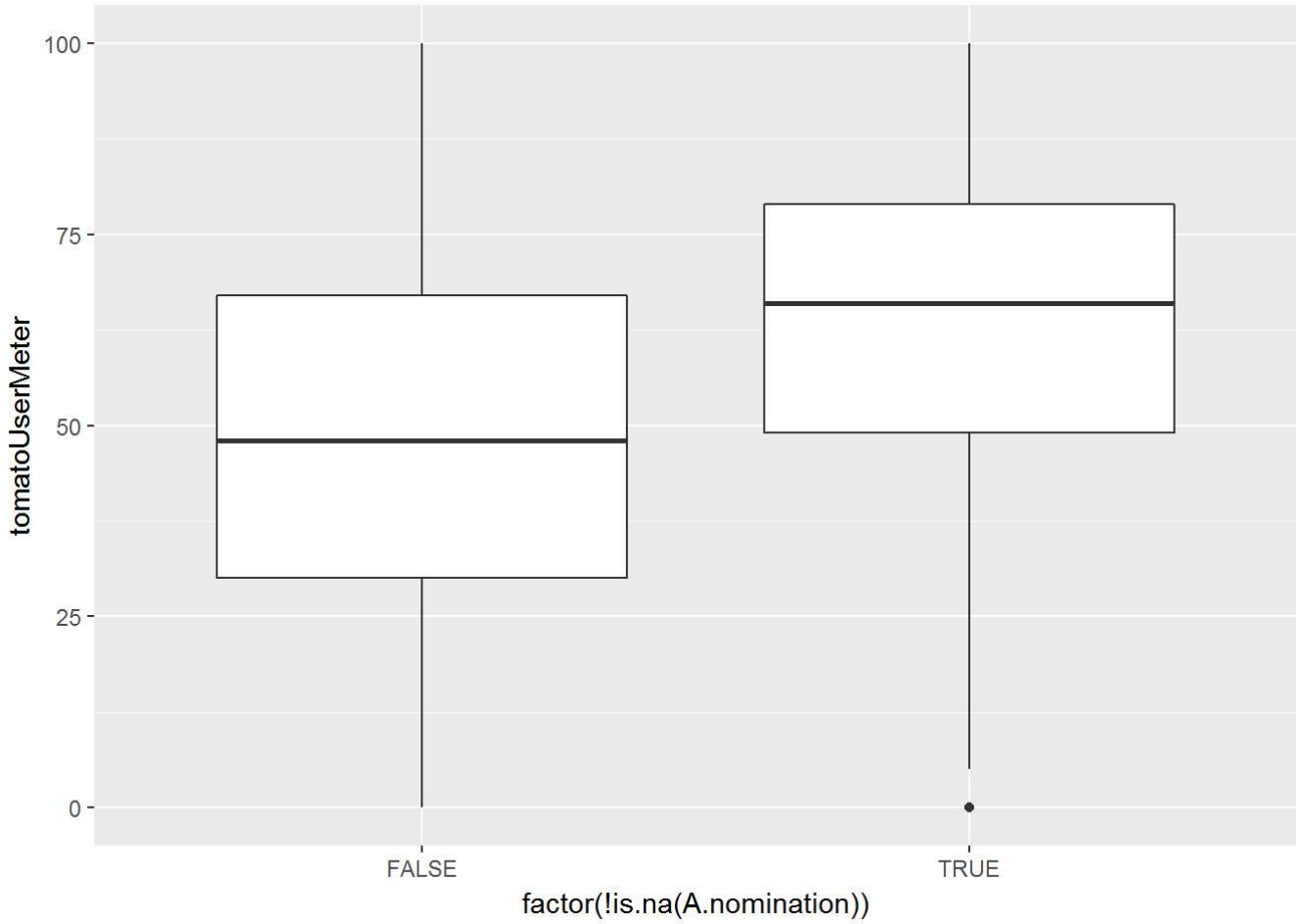
```
## Warning: Removed 2239 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 2239 rows containing missing values (geom_point).
```




```
plot8.4
```

```
## Warning: Removed 19740 rows containing non-finite values (stat_boxplot).
```




**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**: 1. Different ratings are used to predict the success of a movie in winning awards or nominations. It seems tomatoUserMeter provides the best prediction. if a movie has tomatoUserMeter higher than 60, it has great chance to win an award, if the value is lower than 50, then has very low chance to win an award. Same for nominations. 2. For movies have wins below 50, or nomination below 100, there are strong correlation between ratings and wins or nominations. For movies have wins larger than 50 or nomination larger than 100, there are weak or none correlations between ratings and wins or nominations.

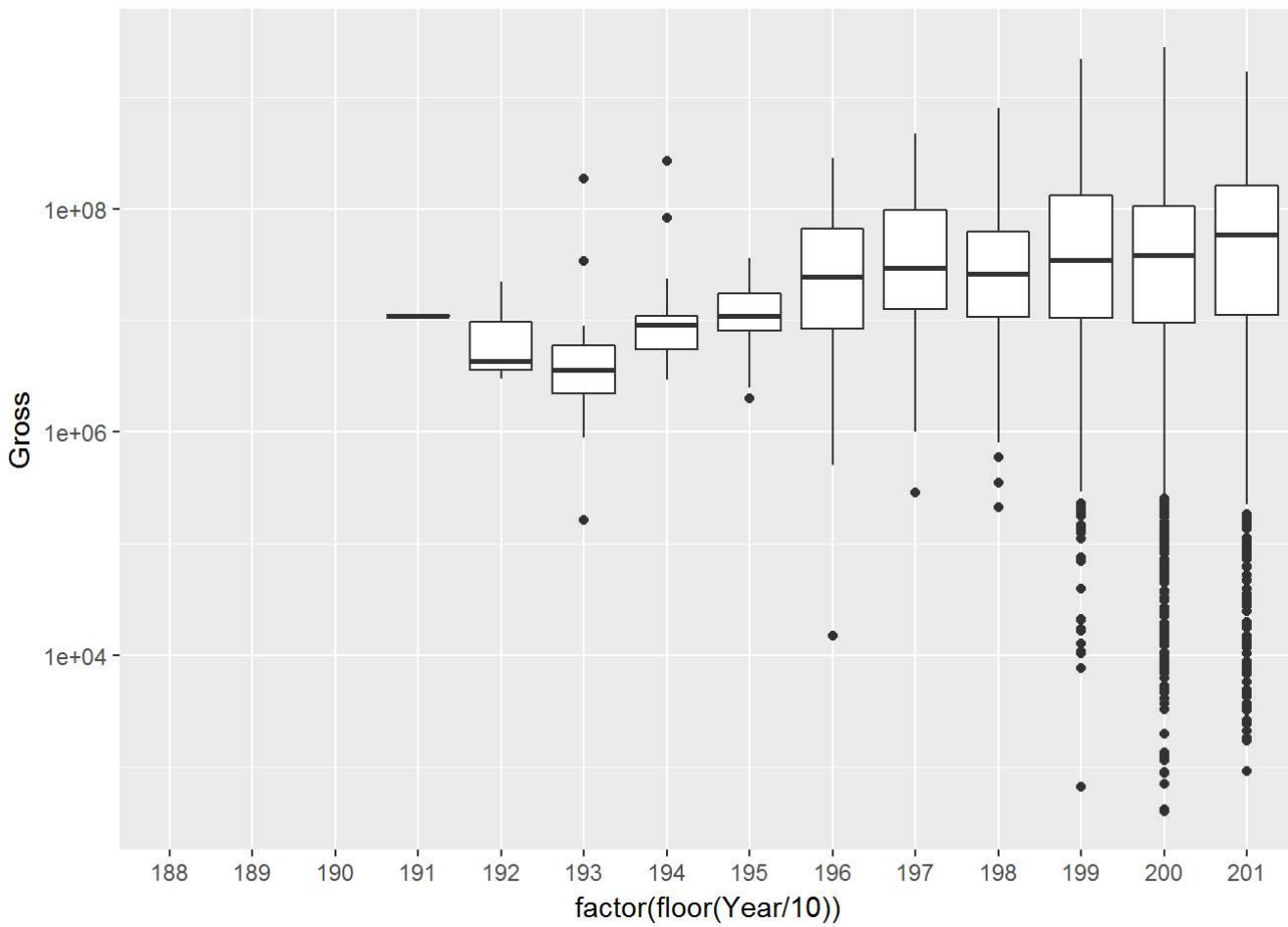
# 9. Expected insights

Come up with two new insights (backed up by data and graphs) that is expected. Here â□□newâ□□ 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
#plot show Gross increase with years
plot9.1 = ggplot(df, aes(factor(floor(Year/10)), Gross)) +
  geom_boxplot() +
  scale_y_log10()

#plot shows movies directed by James Cameron have high Gross revenue.
plot9.2 = ggplot(df, aes(factor(Director == "James Cameron"), Gross)) +
  geom_boxplot()

plot9.1
```

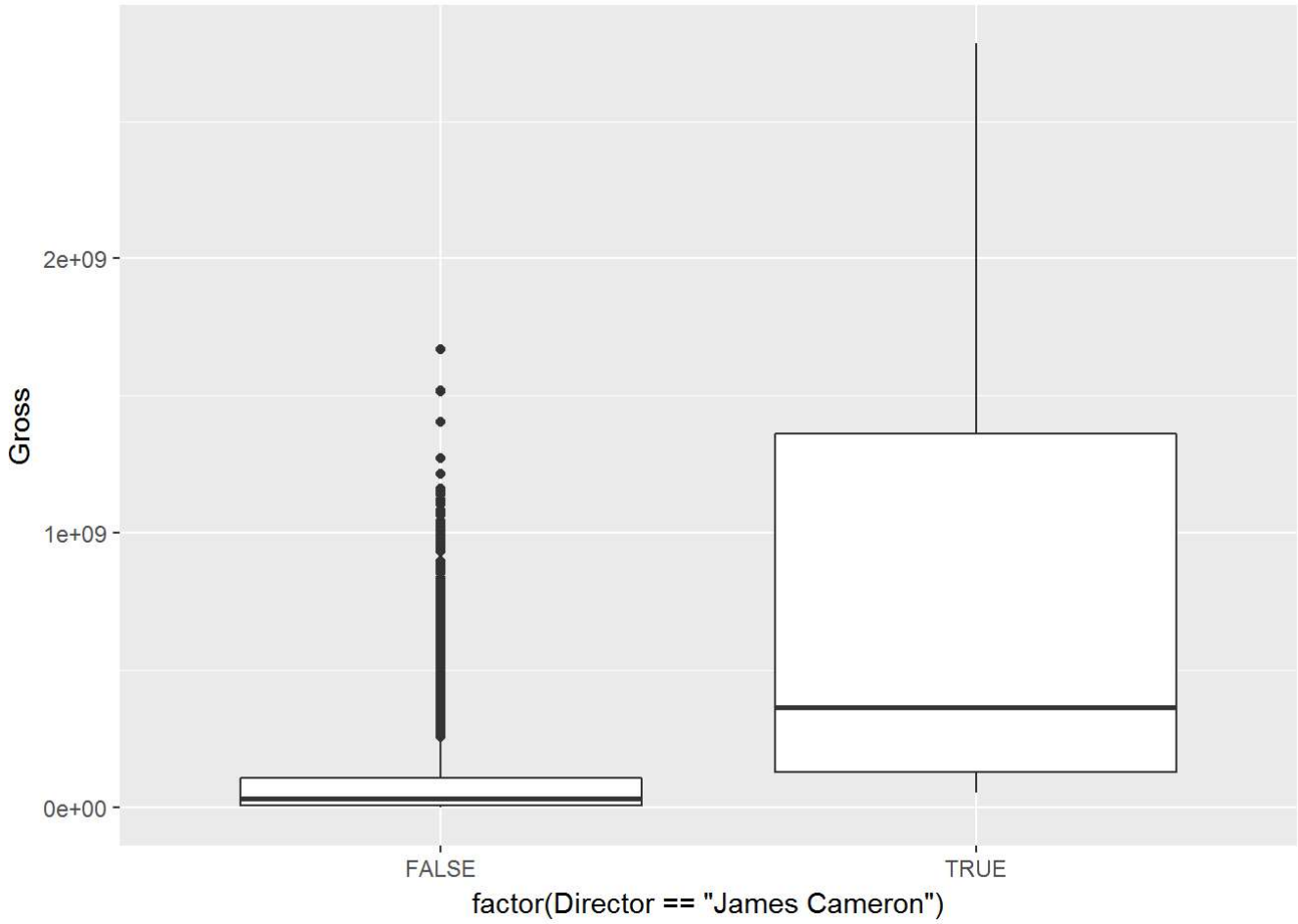
```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 30728 rows containing non-finite values (stat_boxplot).
```




```
plot9.2
```

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




**Q**: Expected insight #1.

**A**: Gross revenues increase with years. This may due to 1) the inflation, 2) movies are more popular than earlier years.

**Q**: Expected insight #2.

**A**: Movies directed by famous directors have high gross revenues. For example, the movies directed by James Cameron have very high gross revenues comapared with the rest movies.

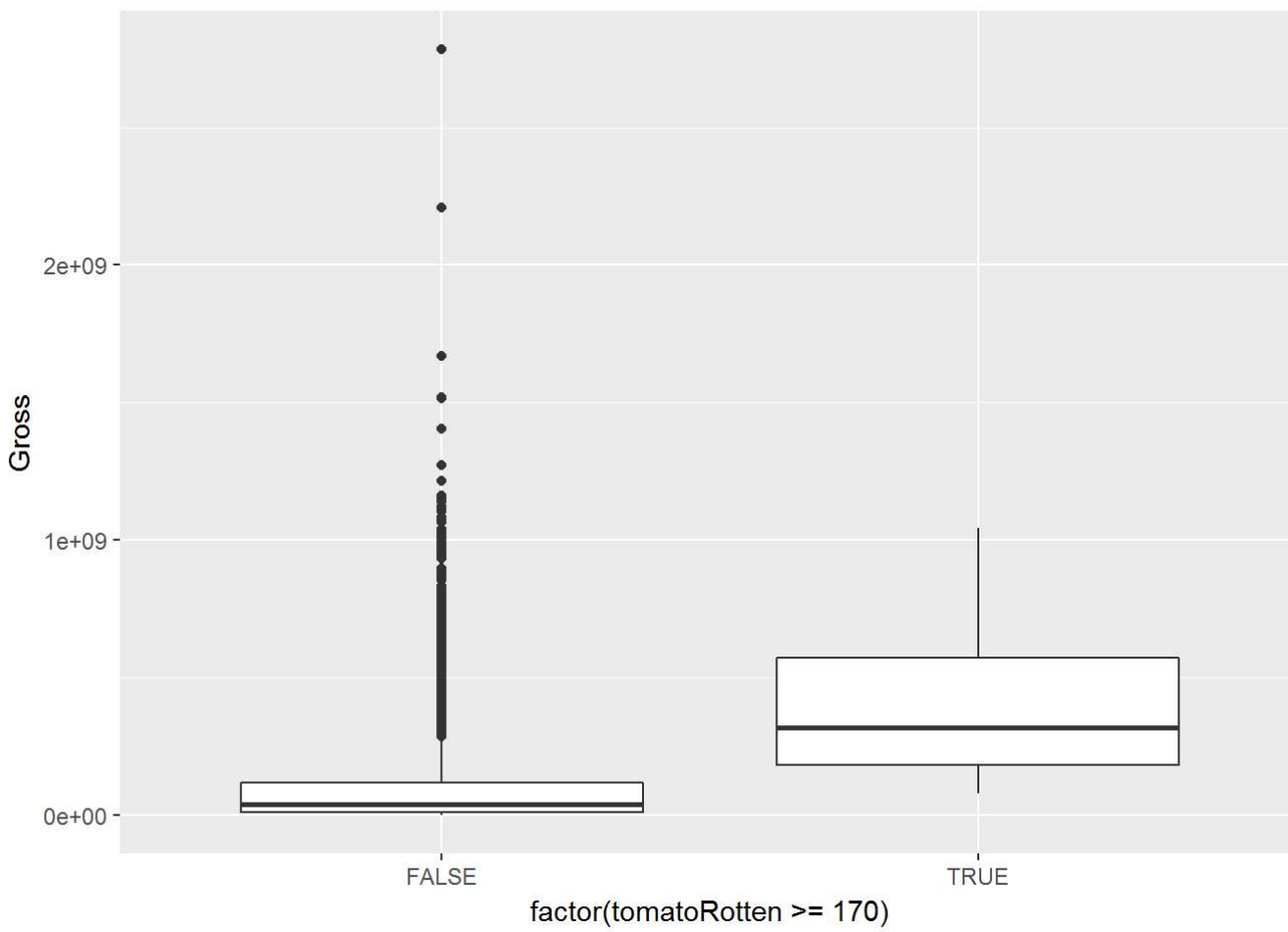
# 10. Unexpected insight

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.

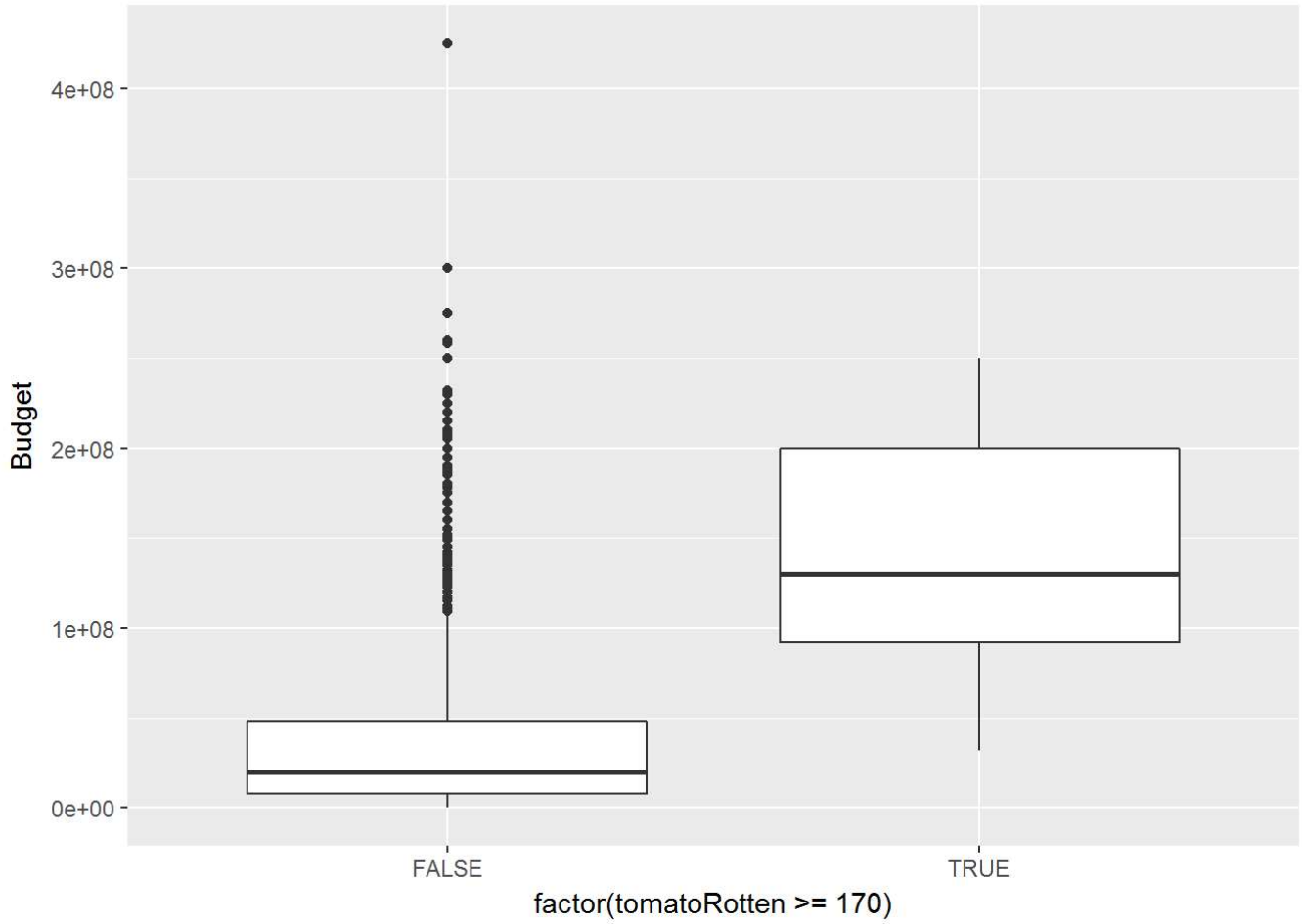
```
# TODO: Find and illustrate one unexpected insight
df.new = df[!is.na(df$Gross) & !is.na(df$tomatoRotten & !is.na(df$Budget)), c(37, 24, 35)]
df.new = df.new[rev(order(df.new$tomatoRotten)),]
df.new[1:10, ]
```

```
##             Gross tomatoRotten  Budget
## 1985    868160194          249 2.5e+08
## 110978  836519699          196 2.1e+08
## 110240  304249198          187 9.2e+07
## 17662   570978470          182 4.0e+07
## 110996  365491792          180 2.0e+08
## 448      82145379          178 1.0e+08
## 110569 1045663875          175 2.5e+08
## 111417   77593167          173 3.2e+07
## 17718    91678442          173 4.2e+07
## 110751  319713881          172 1.5e+08
```

```
plot10.1 = ggplot(df.new, aes(factor(tomatoRotten >= 170), Gross)) +
  geom_boxplot()

plot10.2 = ggplot(df.new, aes(factor(tomatoRotten >= 170), Budget)) +
  geom_boxplot()

plot10.1
```




```
plot10.2
```




**Q**: Unexpected insight.

**A**: Movies with large number of tomatoRotten are usually bad movies. Interestingly, the top 10 movies with highest tomatoRotten number have much higher gross revenues comapred the rest movies. Possible explanation is that these are popular and high budget movies, thus have so many bad reviews. The explanation is supported by the boxplot of budgets, that these ten movies have much higher budgets compared with average movies.