Project 1: Explore and Prepare Data

Code

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 IMDb'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 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 "second window" 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

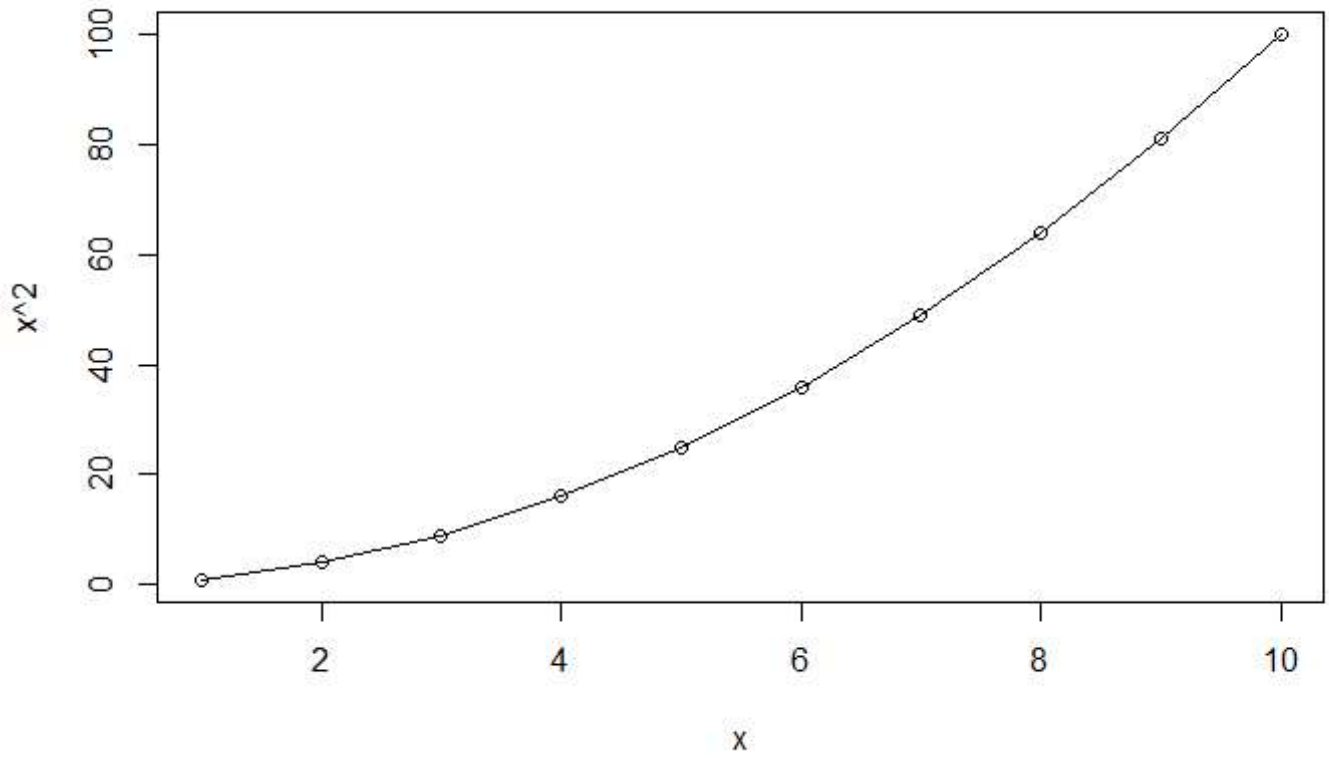
```
x = 1:10
print(x^2)
```

```
 [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

```
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"            "Released"         "Runtime"
 [6] "Genre"            "Director"         "Writer"           "Actors"           "Plot"
[11] "Language"         "Country"          "Awards"           "Poster"           "Metascore"
[16] "imdbRating"       "imdbVotes"        "imdbID"           "Type"             "tomatoMeter"
[21] "tomatoImage"      "tomatoRating"     "tomatoReviews"    "tomatoFresh"      "tomatoRotten"
[26] "tomatoConsensus"  "tomatoUserMeter"  "tomatoUserRating" "tomatoUserReviews" "tomatoURL"
[31] "DVD"              "BoxOffice"        "Production"       "Website"          "Response"
[36] "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

```
install.packages("tokenizers")
```

```
Installing package into 愀牶C:/Users/BrunoWan/Documents/R/win-library/3.2愀牷
(as 愀牶lib愀牷 is unspecified)
also installing the dependency 愀牶SnowballC愀牷

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/SnowballC_0.5.1.zip'
Content type 'application/zip' length 3076520 bytes (2.9 MB)
downloaded 2.9 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/tokenizers_0.1.4.zip'
Content type 'application/zip' length 493089 bytes (481 KB)
downloaded 481 KB
```

```
package ‘SnowballC’ successfully unpacked and MD5 sums checked
package ‘tokenizers’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\BrunoWan\AppData\Local\Temp\Rtmp86LeP0\downloaded_packages
```

Hide

```
library("tokenizers")
```

```
package 愀牶tokenizers愀牷 was built under R version 3.2.5
```

Hide

```
library(ggplot2)
library(GGally)
library(stringr)
```

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=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.

Hide

```
# TODO: Replace df$Runtime with a numeric column containing the runtime in minutes
# Check the text feature of the Runtime column
unique(str_extract_all(df$Runtime, "[A-z]+"))
```

```
[[1]]
[1] "min"

[[2]]
[1] "N" "A"

[[3]]
[1] "h"   "min"

[[4]]
[1] "h"
```

Hide

```
#Replace df$Runtime with a numeric column containing the runtime in minutes
for(i in c(1:nrow(df))){
  if(is.na(str_extract_all(df[i,"Runtime"], "[A-z]+"))){
    df[i,"Runtime"]=as.numeric(str_extract_all(df[i,"Runtime"], "[[:digit:]]+"))
  }else if(length(str_extract_all(df[i,"Runtime"], "[A-z]+")[[1]])==2){
    df[i,"Runtime"]=(as.numeric(str_extract_all(df[i,"Runtime"], "[[:digit:]]+")[[1]][1]))*60 +
(as.numeric(str_extract_all(df[i,"Runtime"], "[[:digit:]]+")[[1]][2]))
  }else if(str_extract_all(df[i,"Runtime"],"[A-z]+")=="h"){
    df[i,"Runtime"]=(as.numeric(str_extract_all(df[i,"Runtime"], "[[:digit:]]+")))*60
  }else{
    df[i,"Runtime"]=as.numeric(str_extract_all(df[i,"Runtime"], "[[:digit:]]+"))
  }
}
```

Hide

```
# Convert the data type of Runtime from character to numeric
df$Runtime=as.numeric(df$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.

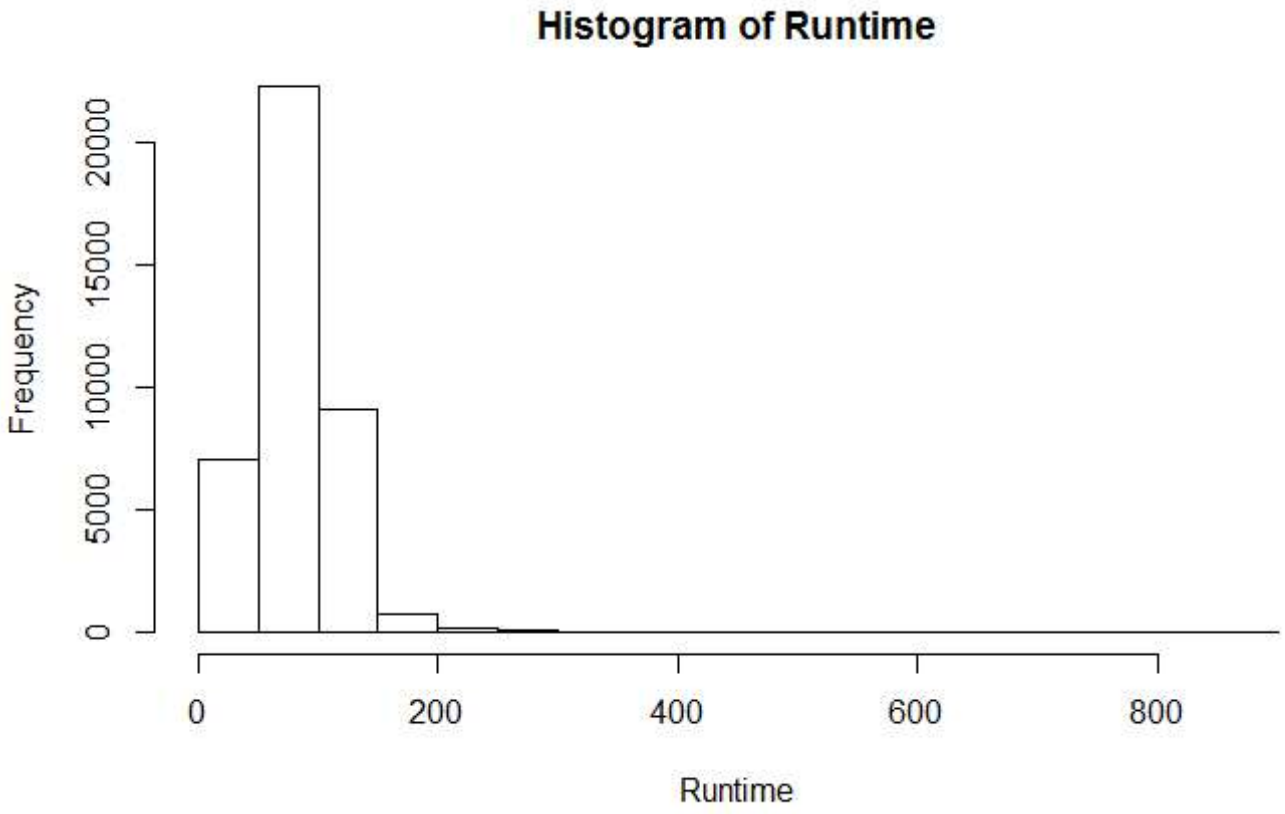
Hide

```
# 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.    NA's
   1.00   72.00   90.00   81.79  101.00  873.00     751
```

Hide

```
hist(df$Runtime, xlab="Runtime", main="Histogram of Runtime")
```



Hide

```
# Subset the original dataset to get a new dataset for the analysis
runtime=df[,c("Year", "Budget", "Runtime")]
```

Hide

```
# Get some statistical features of the three columns
min(runtime$Year)
```

```
[1] 1888
```

Hide

```
max(runtime$Year)
```

```
[1] 2018
```

Hide

```
min(runtime[!is.na(runtime$Budget), "Budget"])
```

```
[1] 1100
```

Hide

```
max(runtime[!is.na(runtime$Budget), "Budget"])
```

```
[1] 425000000
```

Hide

```
nrow(runtime[!is.na(runtime$Budget),])
```

```
[1] 4558
```

Hide

```
nrow(runtime[!is.na(runtime$Year),])
```

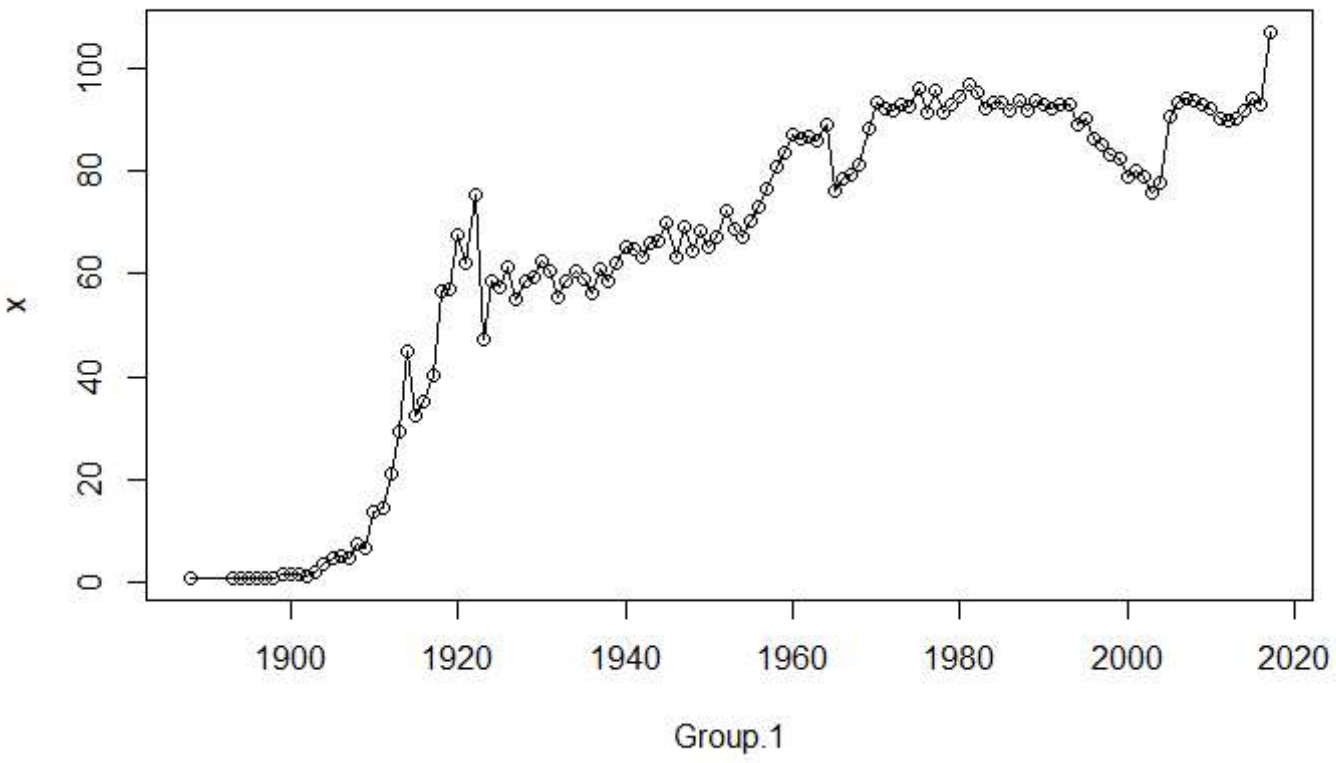
```
[1] 40000
```

Hide

```
# From the dataset, we can discover that
# Budget only contains 4520 valid records
# Year contain 400000 valid records
# Year span is from 1888 to 2017
```

Hide

```
# To check the distribution of Runtime over Year and Budget. We need to omit the record with null values
runtime=runtime[!is.na(runtime$Runtime),]
# To fairly compare the Runtime over Year, we need to aggregate the Runtime by Year and Mean
plot(aggregate(runtime[, "Runtime"], list(runtime$Year), mean), type="o")
```



Hide

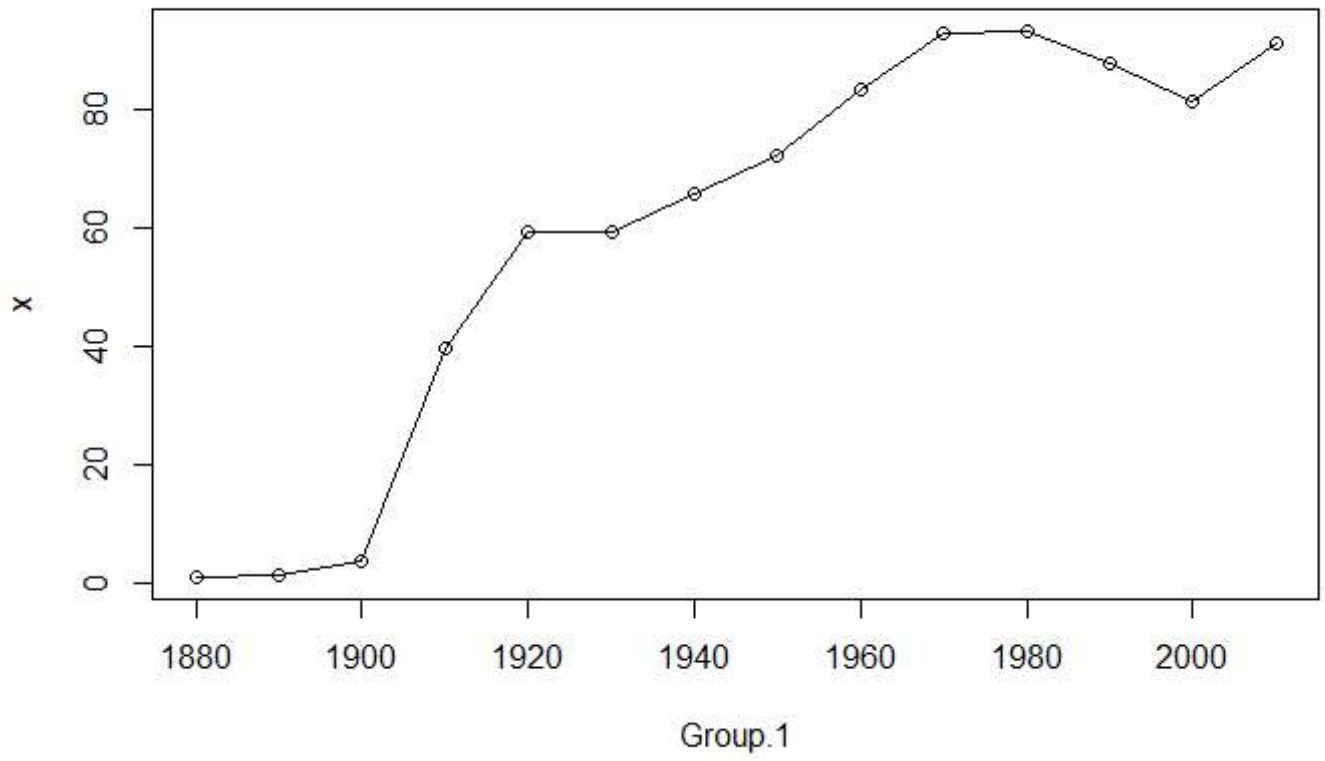
```
# From the plot above, we can discover that there fluctuations over years and there are some years missing
# To get an insight over the general trend of Runtime over Year, we can smooth the data by cut Year into B
ins of decades
runtime$Year_bin=cut(runtime$Year, c(1880, seq(1890, 2020, by=10)), dig.lab=0, labels=FALSE, right=FALSE)
# Change the bin Bin number to the start year of that decade
runtime$Year_cut=1880+(runtime$Year_bin-1)*10
# Then aggregate the "Runtime" by decade by mean
aggregate(runtime[, "Runtime"], list(runtime$Year_cut), mean, na.action=na.omit)
```

Group.1 <dbl>	x <dbl>
1880	1.000000
1890	1.147059
1900	3.528926
1910	39.614973
1920	59.271768
1930	59.317490
1940	65.890879
1950	72.172252
1960	83.489428
1970	92.931782

1-10 of 14 rows Previous **1** 2 Next

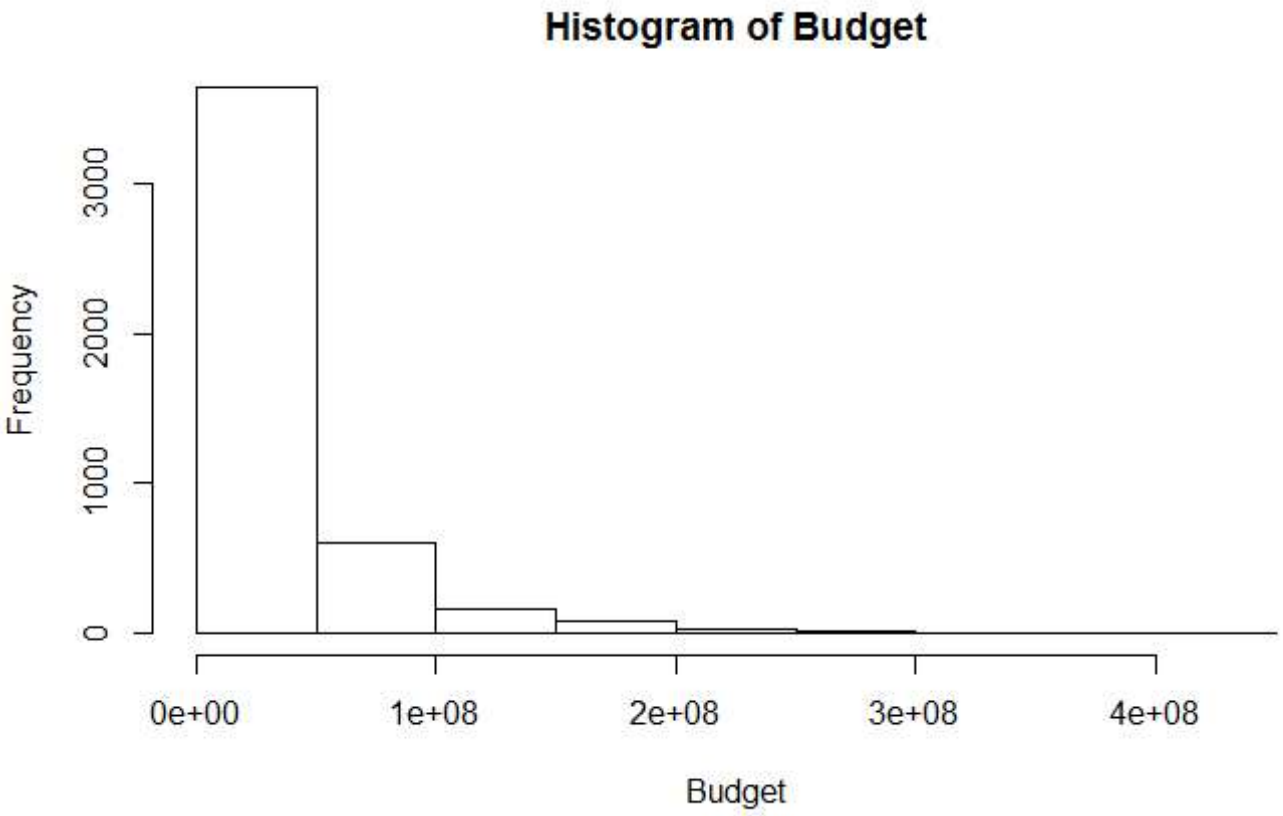
Hide

```
#Display the average Runtime data for the movies in different decades
#need to double check how to omit the NA values at the first place then plot them
plot(aggregate(runtime[,"Runtime"], list(runtime$Year_cut), mean), type="o")
```



Hide

```
#Check the distribution of Budget, we can discover that the Budget is right skewed. Most movie cost less t
han 100 million dollars.
hist(runtime[!is.na(runtime$Budget), "Budget"], main=paste("Histogram of Budget"), xlab="Budget")
```



Hide

```
# For analysis purpose, we use log of Budget instead of Budget because the log of the Budget is more close
d to normal distribution
runtime$Budget_log=log10(runtime$Budget)
min(runtime[!is.na(runtime$Budget_log), "Budget_log"])
```

```
[1] 3.041393
```

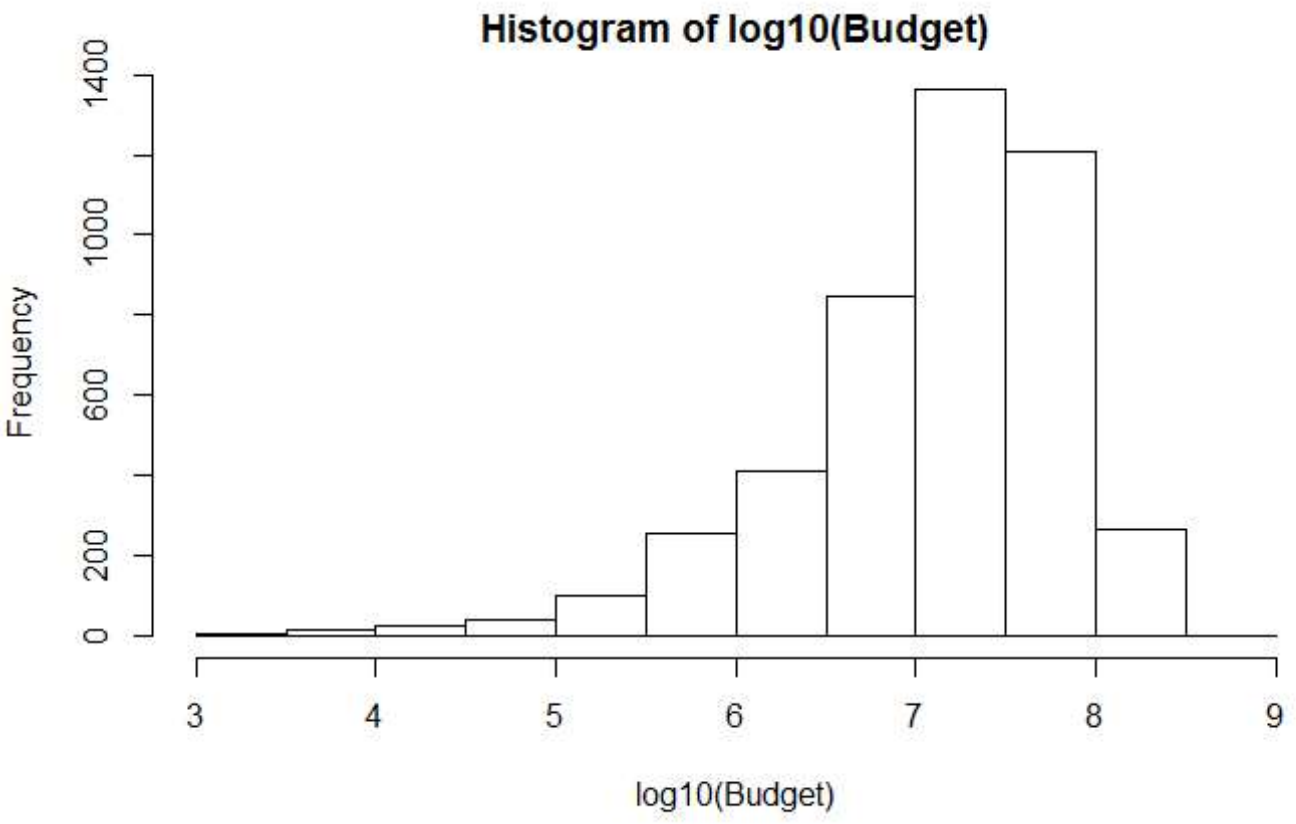
Hide

```
max(runtime[!is.na(runtime$Budget_log), "Budget_log"])
```

```
[1] 8.628389
```

Hide

```
hist(runtime[!is.na(runtime$Budget_log), "Budget_log"], xlab="log10(Budget)", main="Histogram of log10(Bud
get)")
```



Hide

```
# For analysis purpose, we cut the log of Budget into Bins
runtime$Budget_log_bin=cut(runtime$Budget_log, c(3, seq(3.5, 9,by=0.5)), dig.lab=0, labels=FALSE, right=FA
LSE)
# Then aggregate the Runtime by Budget by Budget log by means
aggregate(runtime[,3], list(runtime$Budget_log_bin*0.5+3), mean)
```

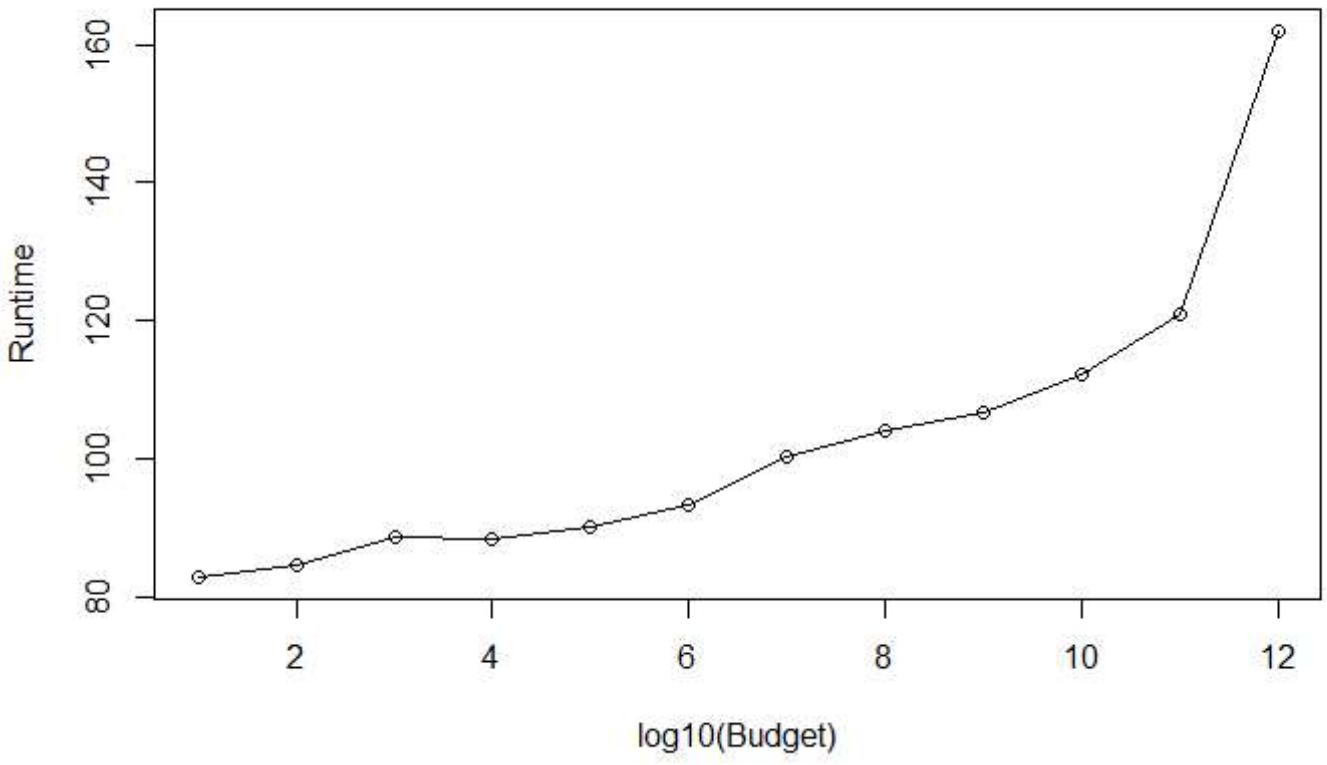
Group.1	x
<dbl>	<dbl>

3.5	83.00000
4.0	84.55556
4.5	88.72000
5.0	88.40741
5.5	90.11607
6.0	93.35625
6.5	100.25149
7.0	103.99564
7.5	106.79698
8.0	112.23997

1-10 of 12 rows Previous **1** 2 Next

Hide

```
plot(aggregate(runtime[,3], list(runtime$Budget_log_bin),mean), type="o", xlab="log10(Budget)", ylab="Runtime")
```



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: The distribution of Runtime of movies is close to normal distribution with some outliers. The median is 90 and mean is 81.44. The runtime of movies is centred on an hour and a half. After process, there is a obvious positive correlation between Budget and Runtime. As Budget increases, the runtime increases as well.

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
# We list all of the genres
unique(unlist(tokenize_regex(df$Genre,pattern=", ")))
```

```
 [1] "Documentary" "Biography"   "Romance"     "Short"       "Thriller"    "Drama"       "War"
 [8] "Comedy"      "Horror"      "Sci-Fi"      "Adventure"   "Family"      "History"     "Crime"
[15] "Action"      "Music"       "Mystery"     "Fantasy"     "Sport"       "Animation"   "Musical"
[22] "N/A"         "Talk-Show"   "Adult"       "Western"     "Film-Noir"   "Reality-TV"  "News"
[29] "Game-Show"
```

Hide

```
#Combine the genre with the original dataset
column_genre_names=unique(unlist(tokenize_regex(df$Genre,pattern=", ")))
genre=data.frame(matrix(nrow=nrow(df), ncol=length(column_genre_names)))
colnames(genre)=column_genre_names
genre[]=0
for(val in column_genre_names){
  genre[grep(val, df[, "Genre"]), val]=1
}
genre_vectored=cbind(df, genre)
```

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
genre_sum=colSums(genre)
print(genre_sum)
```

```
Documentary   Biography     Romance       Short    Thriller       Drama         War      Comedy      Horro
r
       3051        1109        4975        6516        3380       15859        1070       12849         271
6
     Sci-Fi   Adventure      Family     History       Crime      Action       Music     Mystery      Fantas
y
       1624        2928        2653         830        4062        4413        2550        1642         140
0
      Sport   Animation     Musical         N/A   Talk-Show       Adult     Western   Film-Noir  Reality-T
V
        525        2788        1387         986           4         422        1314         352
 6
       News   Game-Show
         20           2
```

Hide

```
#Sort the genres based on the number of occurences
genre_sort=sort(genre_sum, decreasing=TRUE)
print(genre_sort)
```

```
      Drama      Comedy       Short     Romance      Action       Crime    Thriller Documentary   Adventur
e
      15859       12849        6516        4975        4413        4062        3380        3051         292
8
  Animation      Horror      Family       Music     Mystery      Sci-Fi     Fantasy     Musical     Wester
n
       2788        2716        2653        2550        1642        1624        1400        1387         131
4
  Biography         War         N/A     History       Sport       Adult   Film-Noir        News  Reality-T
V
       1109        1070         986         830         525         422         352          20
 6
  Talk-Show   Game-Show
          4           2
```

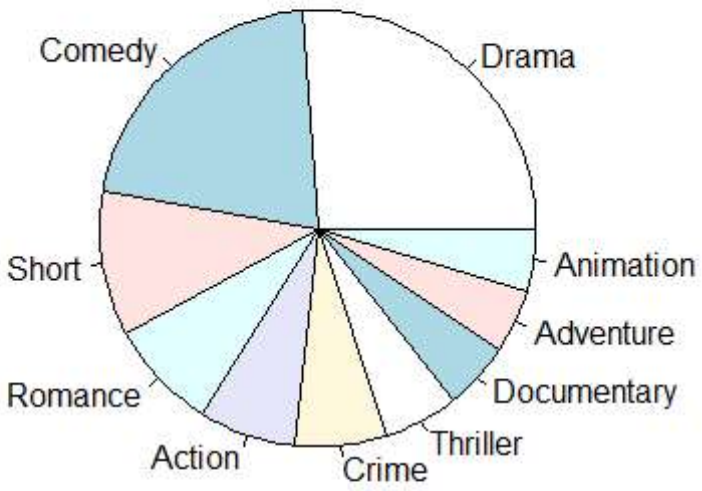
Hide

```
#Take the top 10 genres
genre_sort_names=names(genre_sort)
genre_top=genre_sort[1:10]
genre_prop=genre_top/nrow(df)
print(genre_prop)
```

```
      Drama      Comedy       Short     Romance      Action       Crime    Thriller Documentary   Adventur
e
   0.396475    0.321225    0.162900    0.124375    0.110325    0.101550    0.084500    0.076275    0.07320
0
  Animation
   0.069700
```

Hide

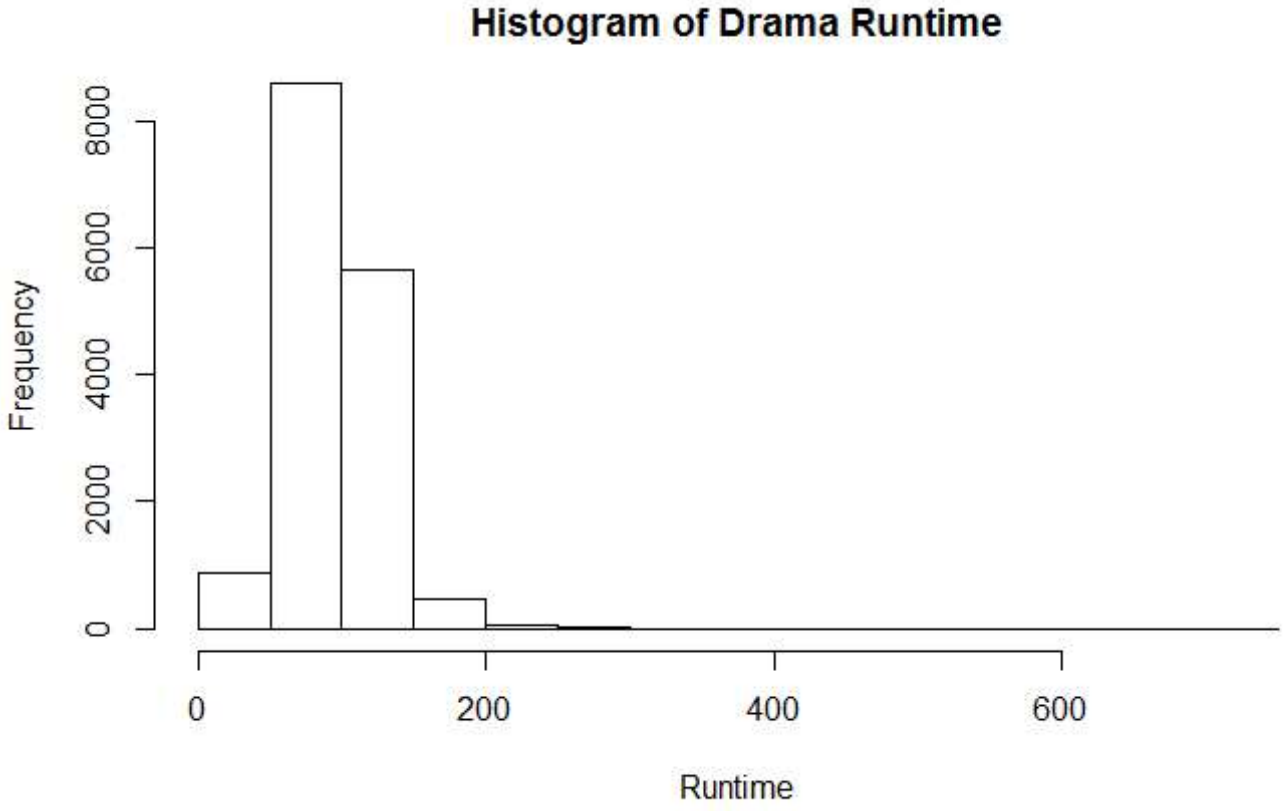
```
library(ggplot2)
pie(genre_prop)
```

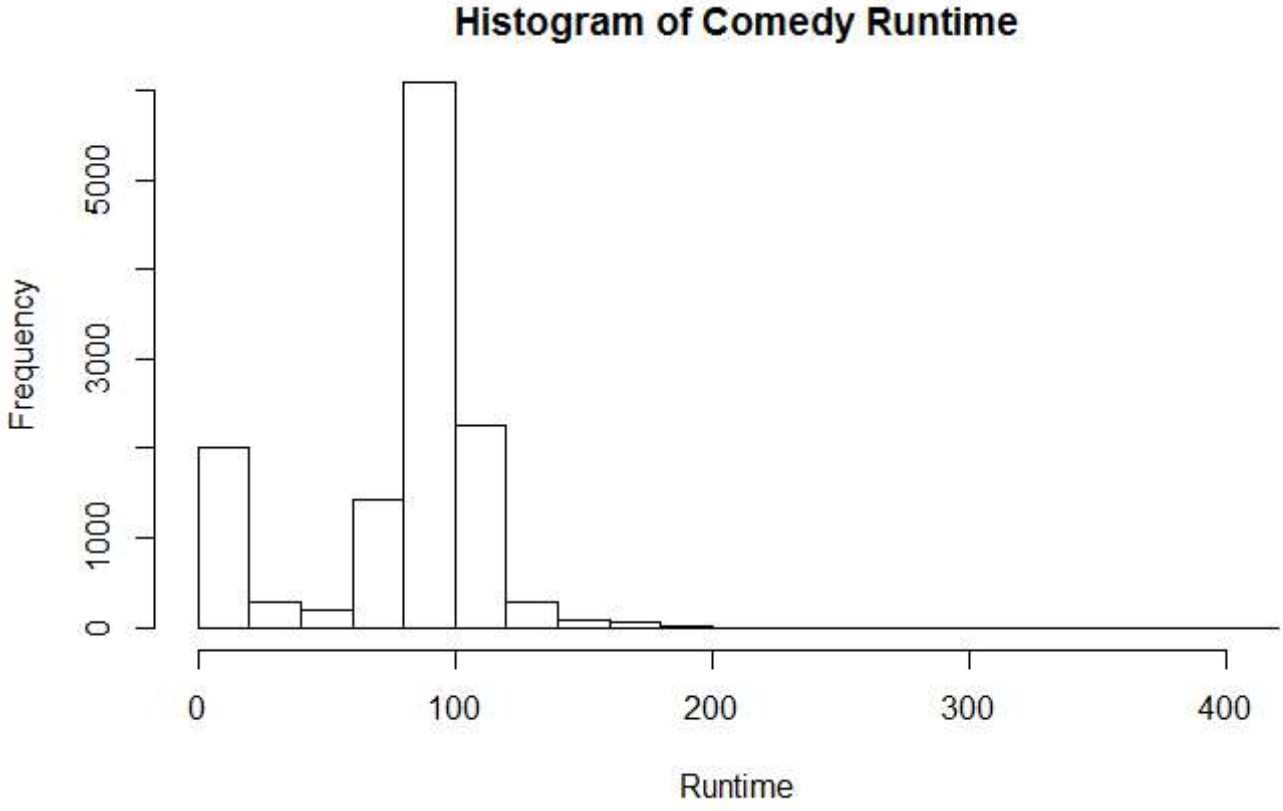


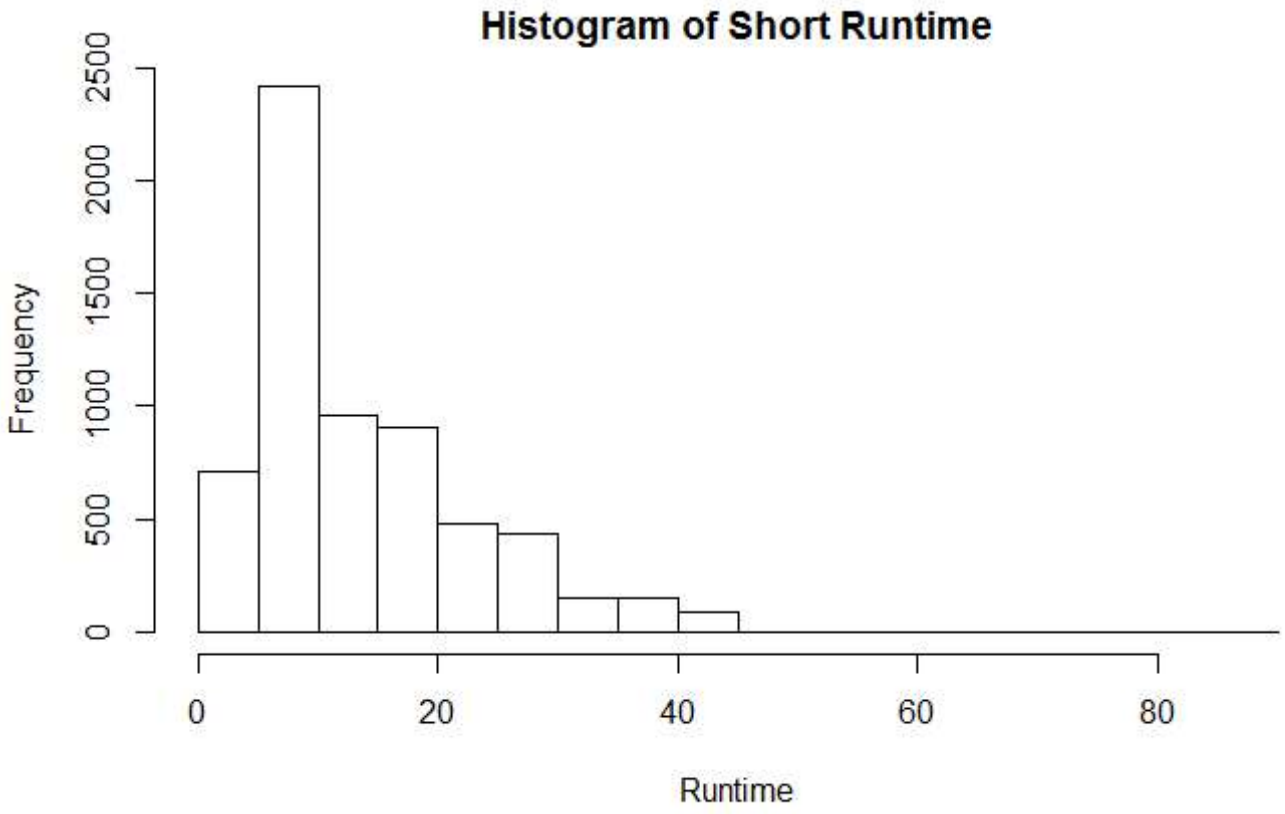
Examine how the distribution of `Runtime` changes across genres for the top 10 most common genres.

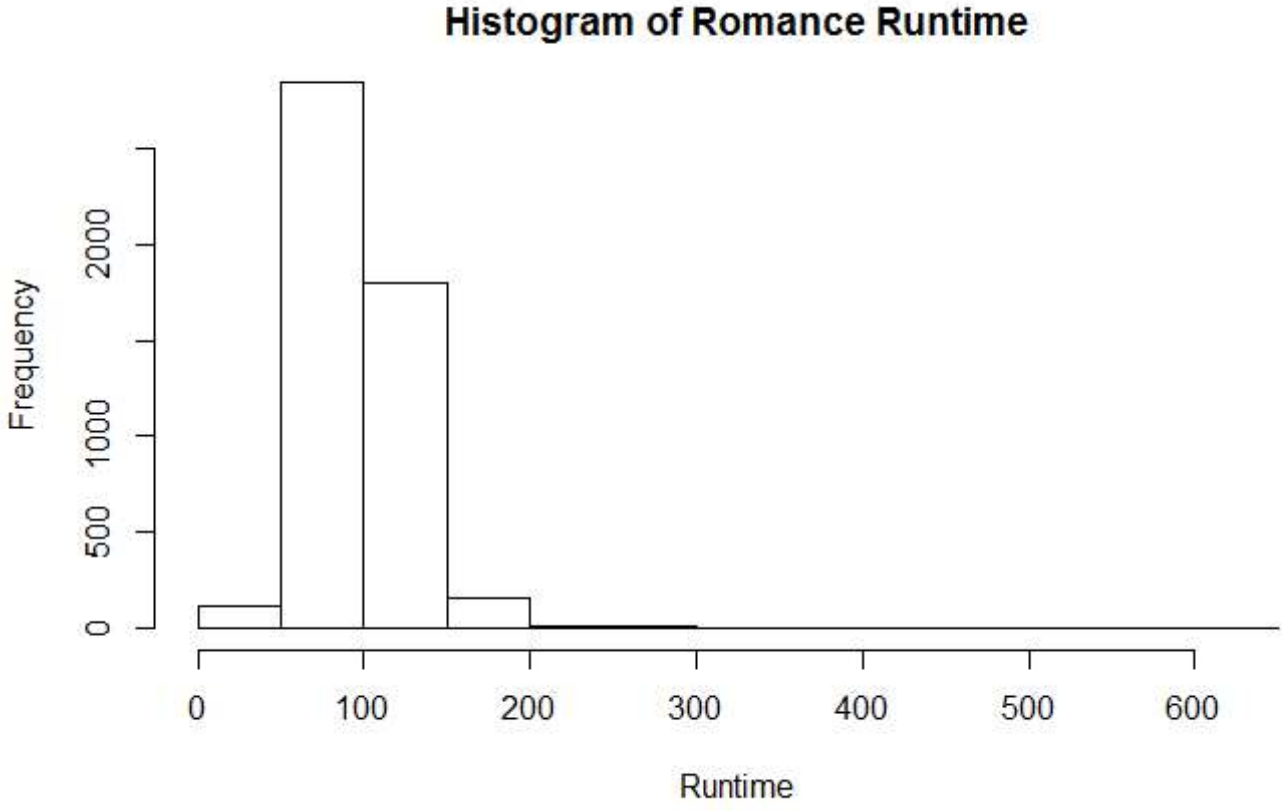
Hide

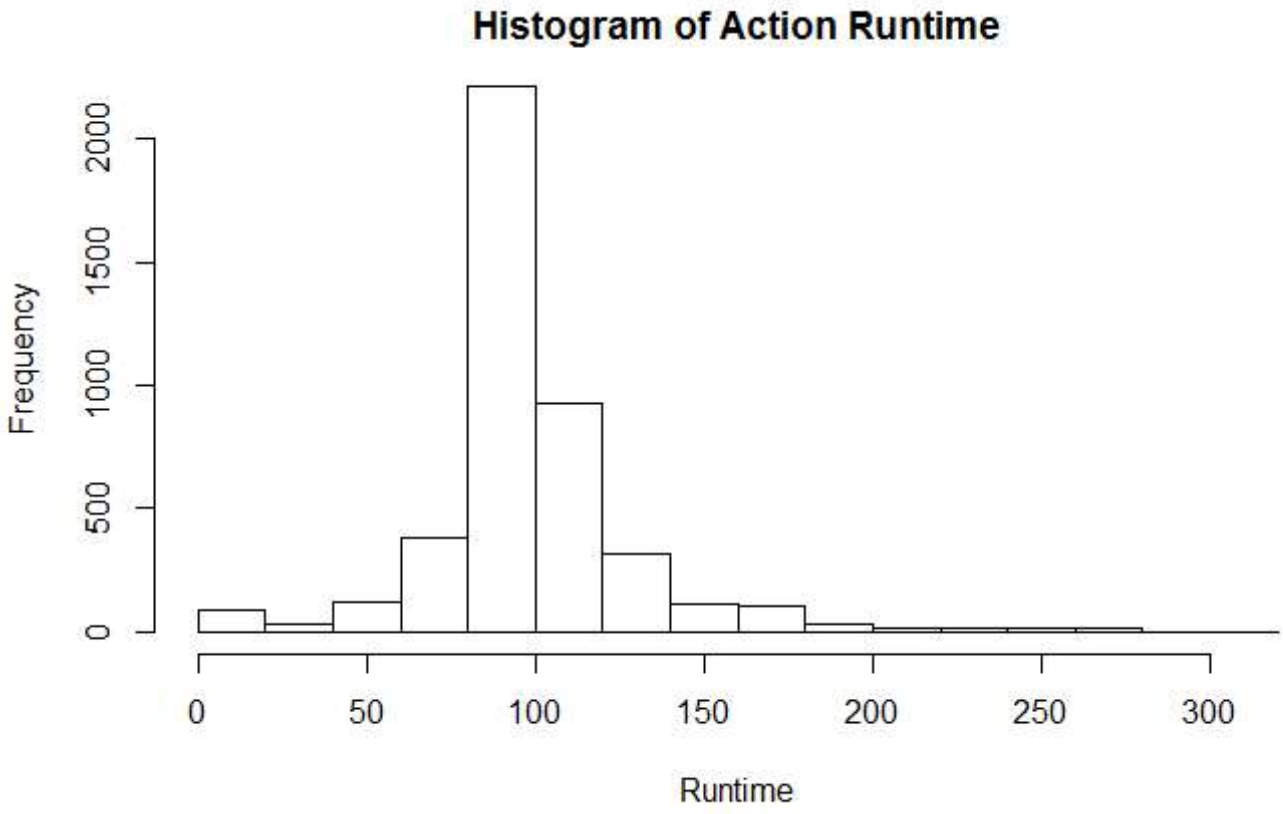
```
# TODO: Plot Runtime distribution for top 10 most common genres
# Create individual data frame for each genre
for(gen in genre_sort_names[1:10]){
  assign(gen, genre_vectored[genre_vectored[, gen]==1,])
}
# Create histogram for each genre and display their runtime distribution
for( gen in genre_sort_names[1:10]){
  hist(get(gen)[!is.na(get(gen)$Runtime), "Runtime"], main=paste("Histogram of", gen, "Runtime"), xlab="Ru
ntime")

}
```

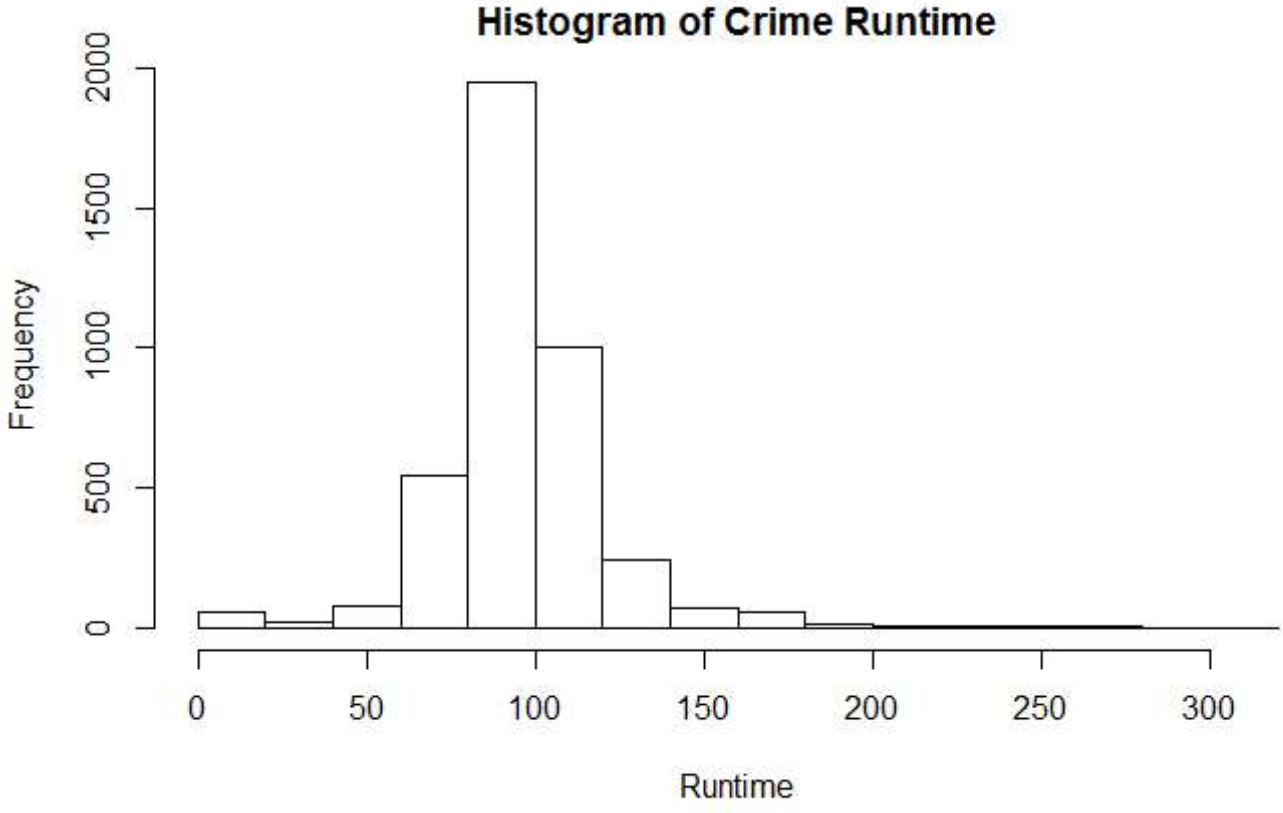

Histogram of Drama Runtime
Frequency
0
2000
4000
6000
8000
0
200
400
600
Runtime

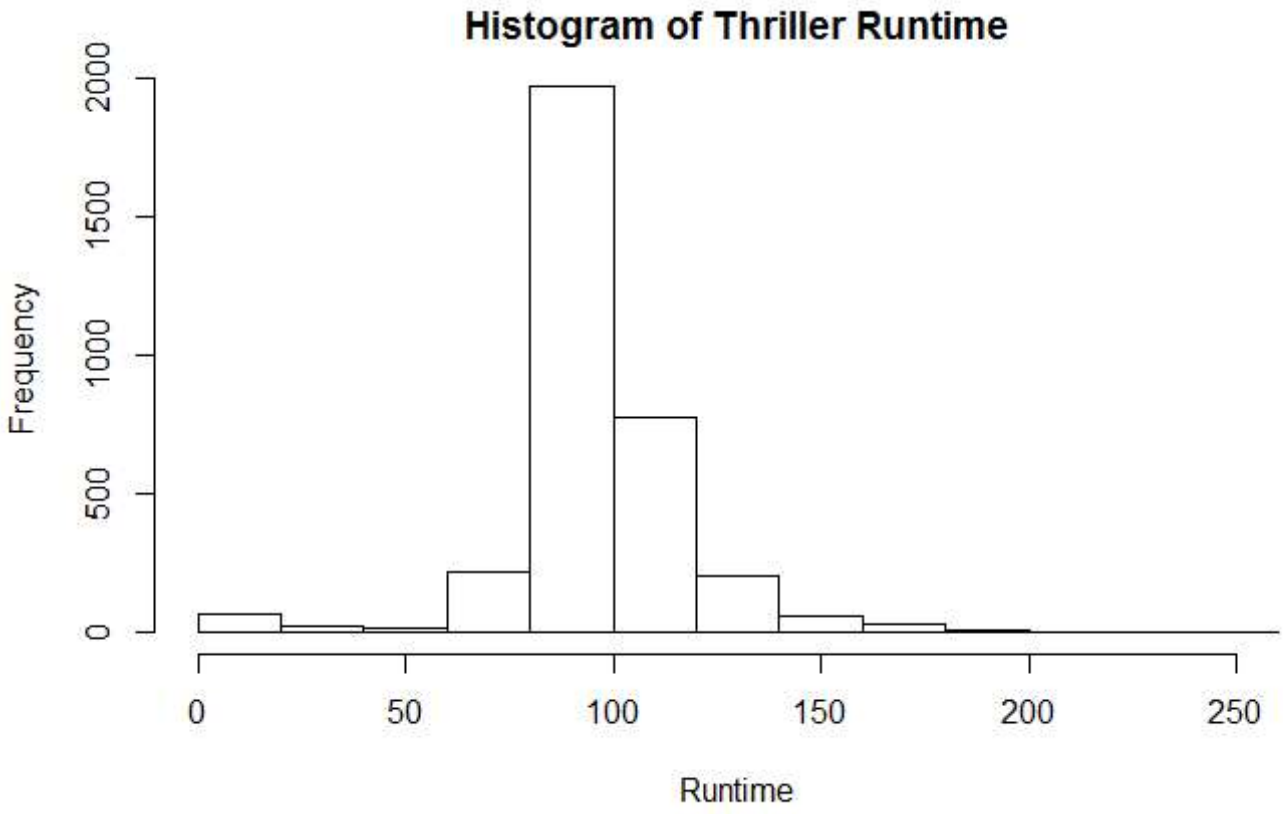

Histogram of Comedy Runtime
Frequency
0
1000
3000
5000
0
100
200
300
400
Runtime

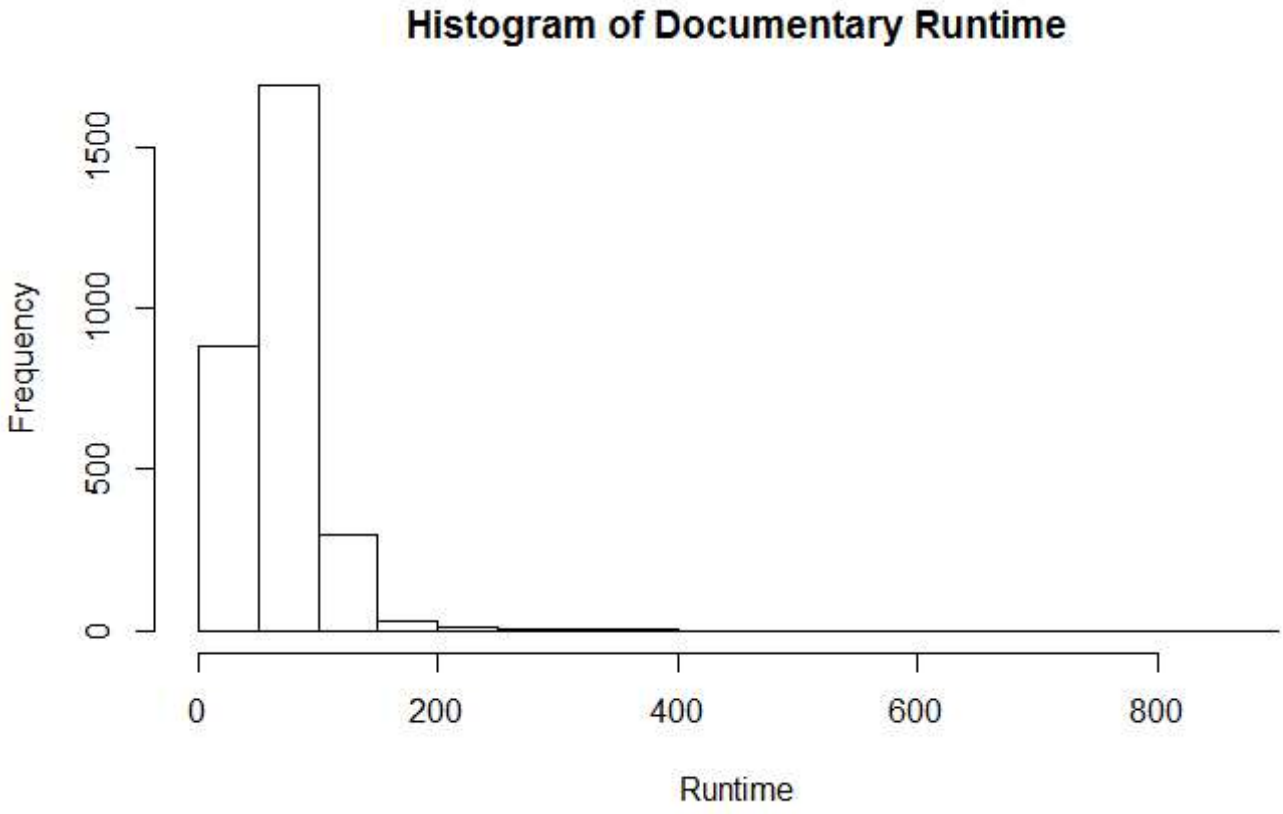

Histogram of Short Runtime
Frequency
0
500
1000
1500
2000
2500
0
20
40
60
80
Runtime

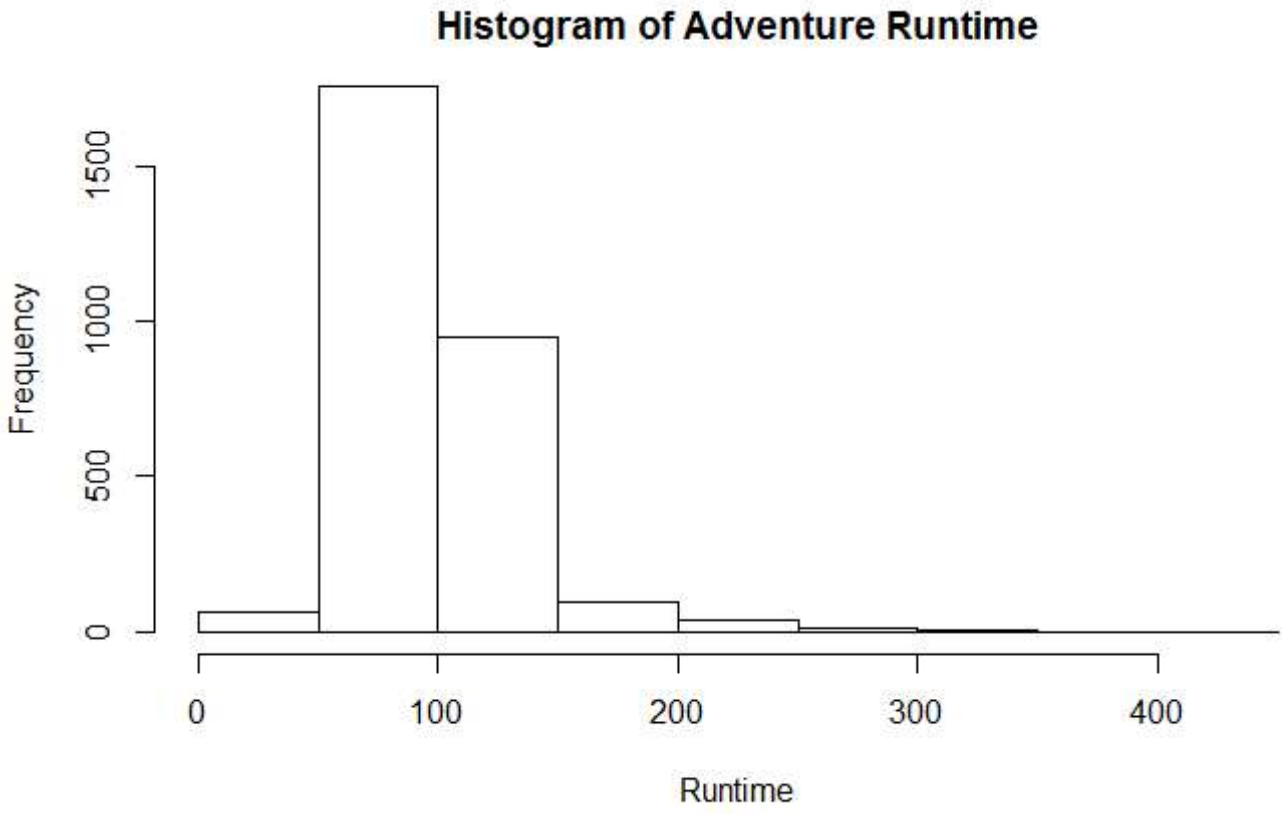

Histogram of Romance Runtime
Frequency
0
500
1000
2000
0
100
200
300
400
500
600
Runtime

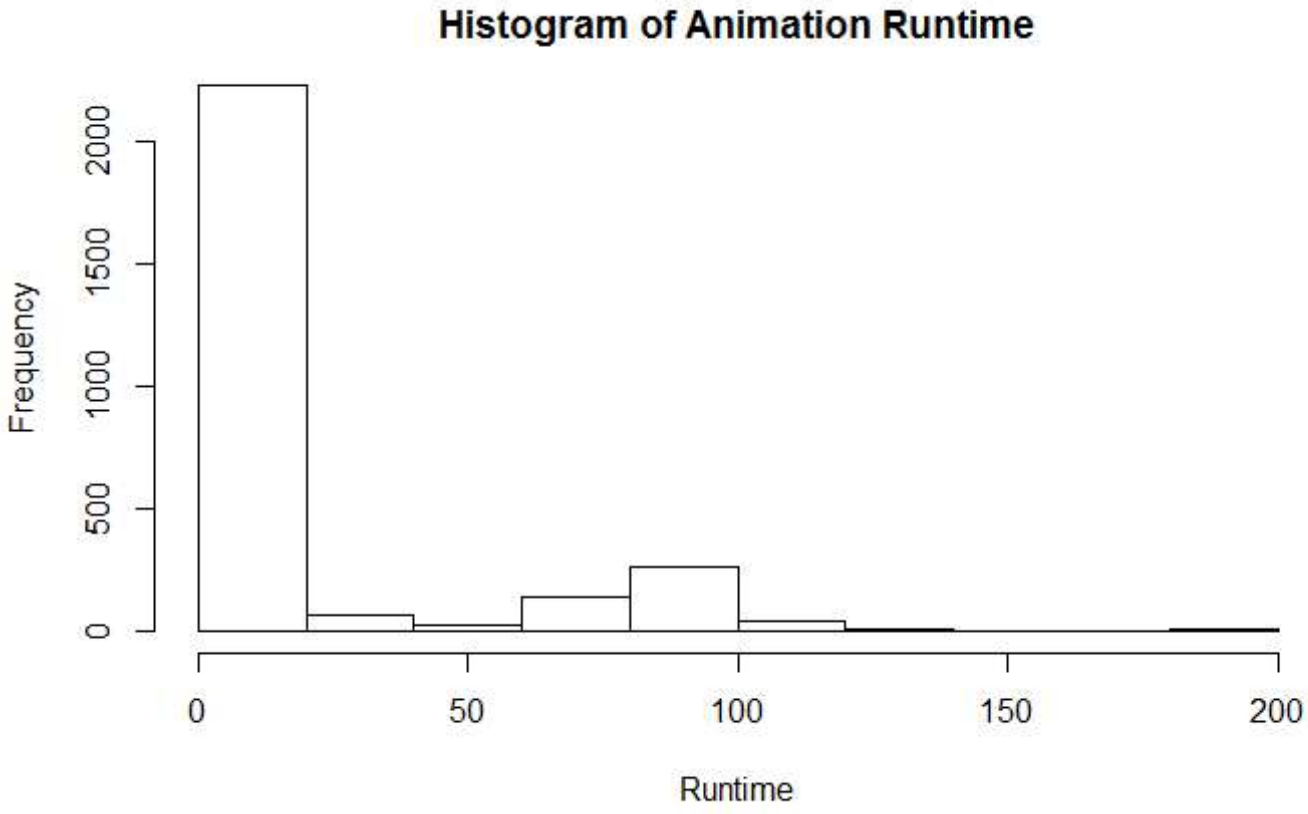

Histogram of Action Runtime
Frequency
0
500
1000
1500
2000
0
50
100
150
200
250
300
Runtime


Histogram of Crime Runtime
Frequency
0
500
1000
1500
2000
0
50
100
150
200
250
300
Runtime


Histogram of Thriller Runtime
Frequency
0
500
1000
1500
2000
0
50
100
150
200
250
Runtime


Histogram of Documentary Runtime
Frequency
0
500
1000
1500
0
200
400
600
800
Runtime





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

A: For most of genre, runtime has a normal-like distribution Expected: Most of genre have a distribution close to normal distribution. Unexpected: Comedy and Animation has a distribution showing a strong skewness.

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'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
# omit NA rows
test=df[setdiff(rownames(df), rownames(df[is.na(df$Released),])),]
test_match=test[abs((test$Year-as.numeric(substr(test$Released, 1, 4))))<=1,]
test_dismatch=df[setdiff(rownames(df), rownames(df[rownames(test_match),])),]
```

Hide

```
#Count the records with Gross value in matched records and dismatched records to ensure the "less than 10%
 removement" requirement is met
nrow(test_match[!is.na(test_match$Gross), ])
```

```
[1] 4422
```

Hide

```
nrow(test_dismatch[!is.na(test_dismatch$Gross),])
```

```
[1] 136
```

Hide

```
nrow(test_dismatch[!is.na(test_dismatch$Gross),])/nrow(test_match[!is.na(test_match$Gross), ])
```

```
[1] 0.03075531
```

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

A: At first, we exclude the records with missing Released column, then we do the calculation of the difference between "Year" column and "Released" column. Then we exclude the records which the difference of "Year" column and the "Released" column is bigger than 1.

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
g_revenue=df[!is.na(df$Gross),]
g_revenue=g_revenue[!is.na(g_revenue$Budget),]
```

Hide

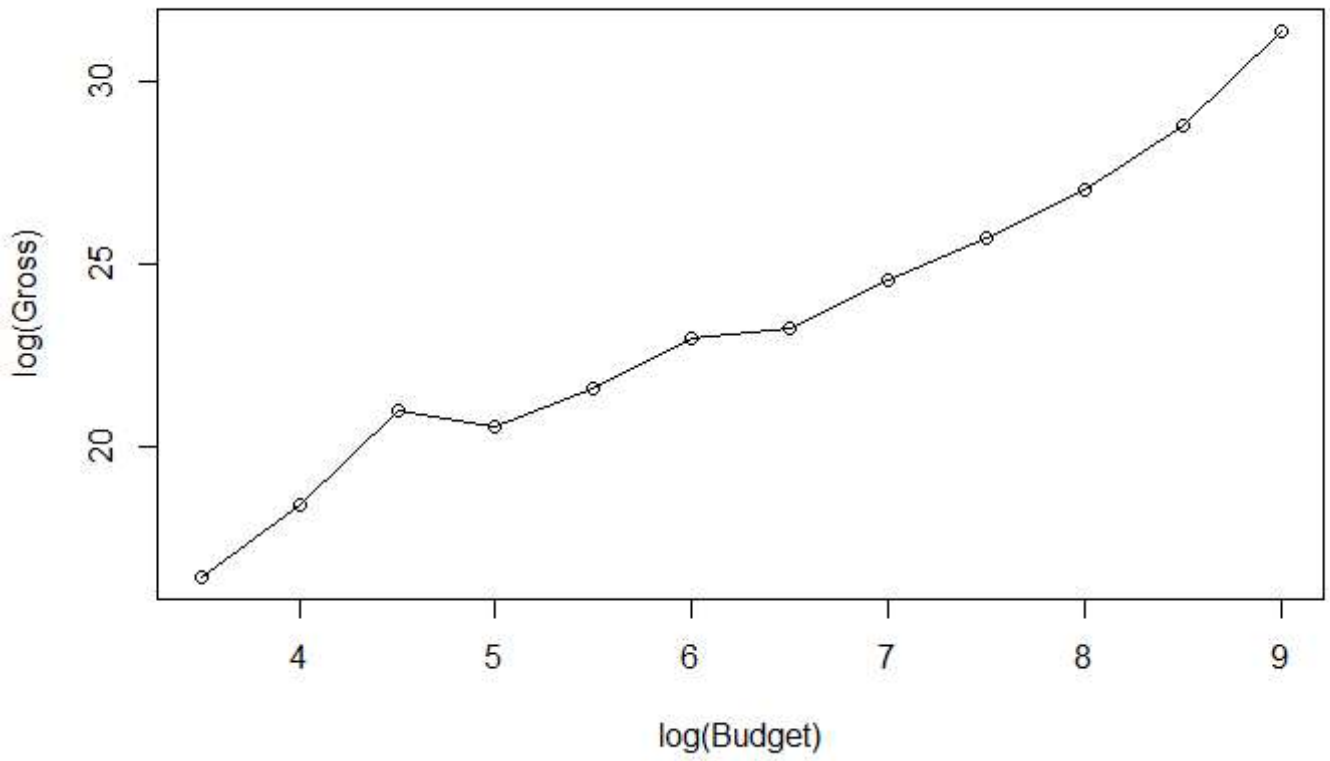
```
g_revenue$Budget_log=log10(g_revenue$Budget)
g_revenue$Budget_log_bin=cut(g_revenue$Budget_log, c(3, seq(3.5, 9, by=0.5)), dig.lab=0, labels=FALSE,righ
t=FALSE)
g_revenue_agg=aggregate(g_revenue[,"Gross"], list(g_revenue$Budget_log_bin*0.5+3), mean)
g_revenue_agg$Gross_log=log2(g_revenue_agg[,2])
print(g_revenue_agg)
```

Group.1 <dbl>	x <dbl>	Gross_log <dbl>
3.5	90520.5	16.46596
4.0	357028.7	18.44568
4.5	2080540.3	20.98853
5.0	1523486.3	20.53895
5.5	3217110.2	21.61733
6.0	8138650.1	22.95636
6.5	9809817.7	23.22579
7.0	24408259.0	24.54087
7.5	55004919.8	25.71306
8.0	137590591.9	27.03581

1-10 of 12 rows Previous **1** 2 Next

Hide

```
plot(g_revenue_agg[,c(1,3)], type="o", xlab="log(Budget)", ylab="log(Gross)")
```

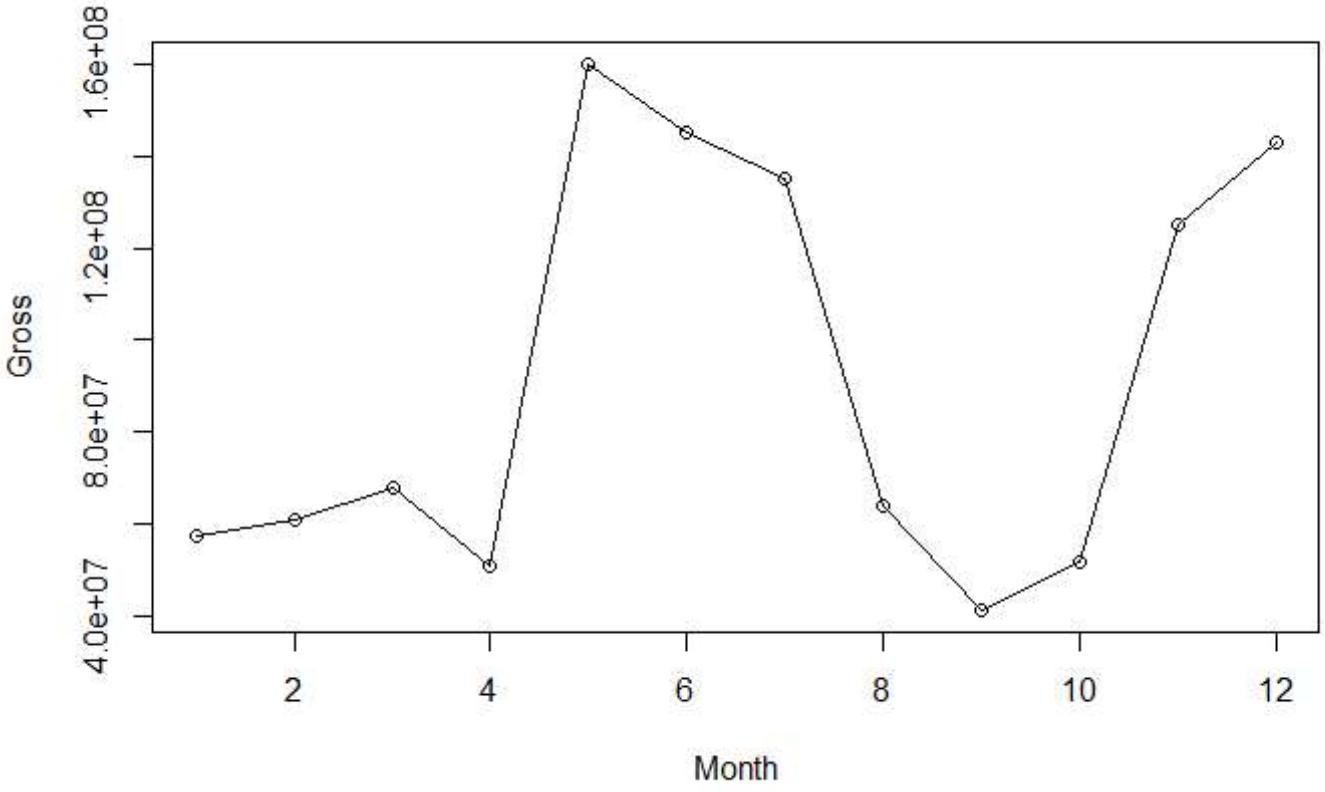


Q: Did you find any observable relationships or combinations of Budget/Runtime/Genre that result in high Gross revenue? If you divided the movies into different subsets, you may get different answers for them - point out interesting ones.

A: We can discover there is a positive correlation between the Budget and Gross value

Hide

```
# TODO: Investigate if Gross Revenue is related to Release Month
# Retrieve the month from the "Released" column
g_revenue$month=substr(g_revenue$Released, 6, 7)
# Aggregate the Gross by month, we can discover the movies are much more successful in summer and Year End
plot(aggregate(g_revenue$Gross, list(g_revenue$month), mean), type="o", xlab="Month", ylab="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.

Hide

```
# TODO: Convert Awards to 2 numeric columns: wins and nominations
# Process the "Awards" column and fill in the "wins" and "nominations" columns
for (i in c(1:nrow(df))){
    win=0
    nomination=0
  if(grepl("Won",df[i, "Awards"])){
    win=as.numeric(str_extract(str_extract(df[i,"Awards"], "Won ([\\d]+)"), "[[:digit:]]+"))
    win=win+as.numeric(str_extract(str_extract(df[i,"Awards"], "([\\d]+) win"), "[[:digit:]]+"))
    nomination=nomination+as.numeric(str_extract(str_extract(df[i,"Awards"], "([\\d]+) nomination"), "[[:d
igit:]]+"))
  }else if(grepl("Nominated",df[i, "Awards"])){
    nomination=as.numeric(str_extract(str_extract(df[i,"Awards"], "Nominated for ([\\d]+)"), "[[:digit:]]
+"))
    win=win+as.numeric(str_extract(str_extract(df[i,"Awards"], "([\\d]+) win"), "[[:digit:]]+"))
    nomination=nomination+as.numeric(str_extract(str_extract(df[i,"Awards"], "([\\d]+) nomination"), "[[:d
igit:]]+"))
  }else{
    win=win+as.numeric(str_extract(str_extract(df[i,"Awards"], "([\\d]+) win"), "[[:digit:]]+"))
    nomination=nomination+as.numeric(str_extract(str_extract(df[i,"Awards"], "([\\d]+) nomination"), "[[:d
igit:]]+"))
  }
    df[i,"wins"]=win
    df[i, "nominations"]=nomination
    if(is.na(df[i,"wins"])){df[i,"wins"]=0}
    if(is.na(df[i,"nominations"])){df[i,"nominations"]=0}
}
```

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

A: 1. If the records contains the word “Won”, then take the number and add it to the variable “win”. Then check the number after the word “win”, add it to the variable “win”. Then check the number after the word “nomination”, add it to the variable “nomination”. 2. Else if the record contains the word “Nominated”, then take the number and add it to the variable “nomination”. Then check the number after the word “win”, add it to the variable “win”. Then check the number after the word “nomination”, add it to the variable “nomination” 3. Else Then check the number after the word “win”, add it to the variable “win”. Then check the number after the word “nomination”, add it to the variable “nomination” 4. Set rest as 0

Hide

```
# TODO: Plot Gross revenue against wins and nominations
gross_awards=df[!is.na(df$Gross),]
aggregate(gross_awards[,"Gross"], list(gross_awards[,"wins"]), mean)
```

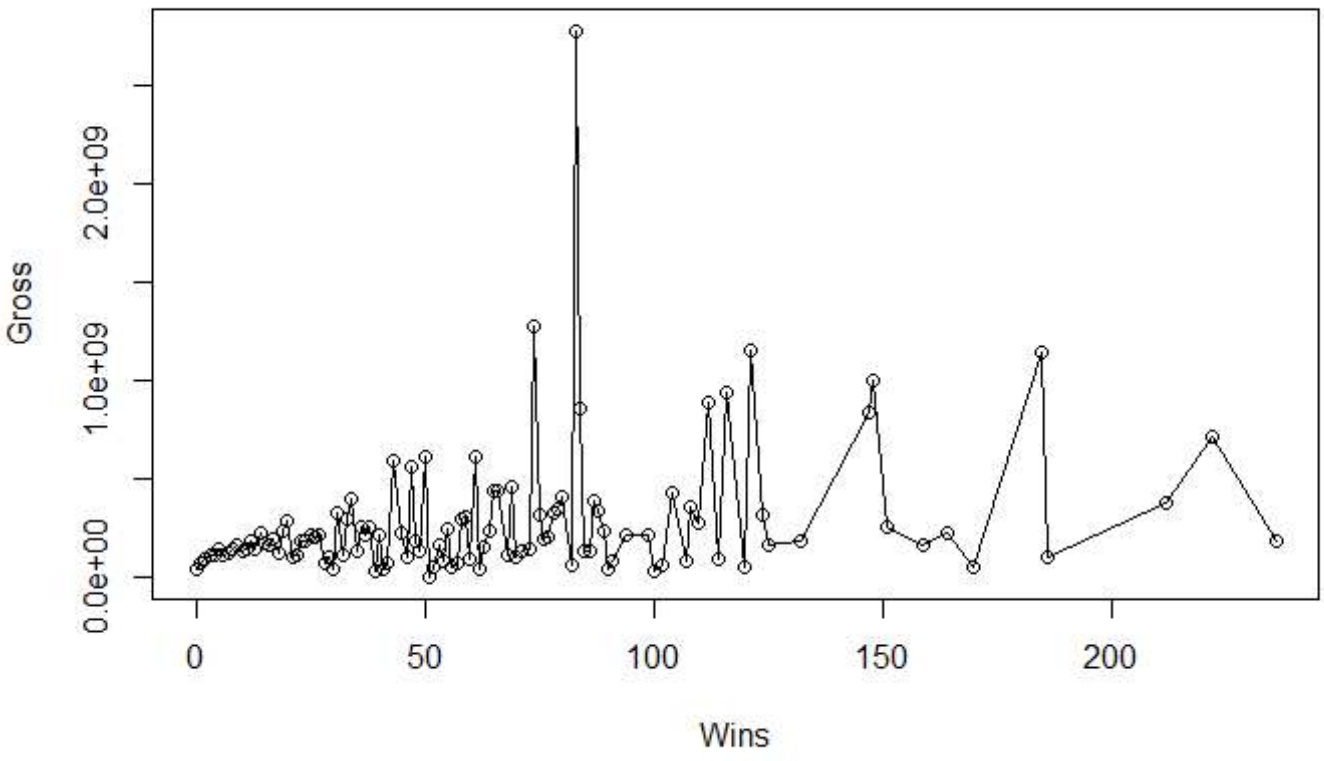
Group.1 <dbl>	x <dbl>
0	35158984
1	68518775
2	94132507
3	105520317
4	108213320
5	137769494

Group.1 <dbl>	x <dbl>
6	112433853
7	123320079
8	142410636
9	157471273

1-10 of 115 rows Previous **1** 2 3 4 5 6 ... 12 Next

Hide

```
plot(aggregate(gross_awards[,"Gross"], list(gross_awards[,"wins"]), mean), type="o", xlab="Wins", ylab="Gr
oss")
```



Hide

```
aggregate(gross_awards[,"Gross"], list(gross_awards[,"nominations"]), mean)
```

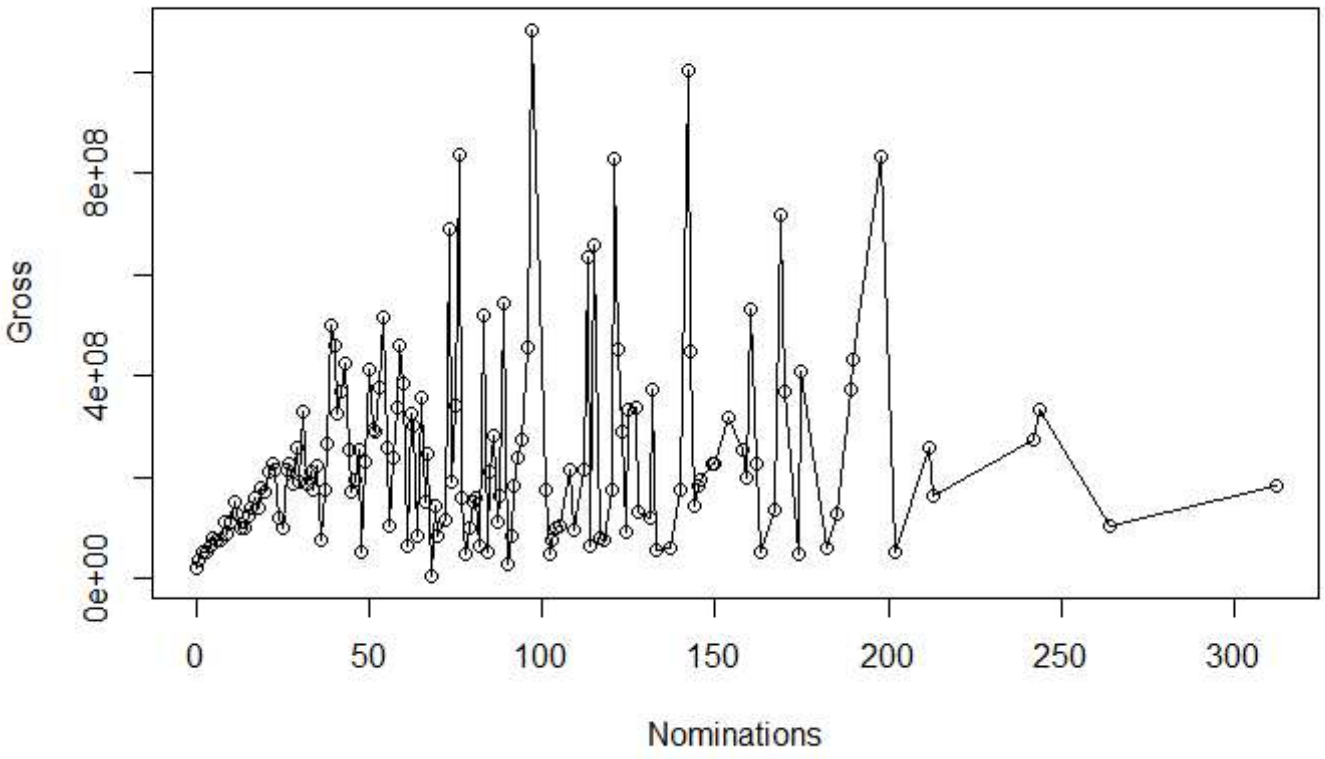
Group.1 <dbl>	x <dbl>
0	19708399
1	35999823
2	51725138
3	51652178

Group.1 <dbl>	x <dbl>
4	64230197
5	77729361
6	75852102
7	73475344
8	110629355
9	88323196

1-10 of 152 rows Previous **1** 2 3 4 5 6 ... 16 Next

Hide

```
plot(aggregate(gross_awards[,"Gross"], list(gross_awards[,"nominations"]), mean), type="o", xlab="Nominati
ons", ylab="Gross")
```



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

A: We can discover that movies with nomination and wins around 70 to 150 have highest Gross revenue. As nomination and wins decreases or increases from this range, the Gross revenue decreases.

7. Movie ratings from IMDb and Rotten Tomatoes

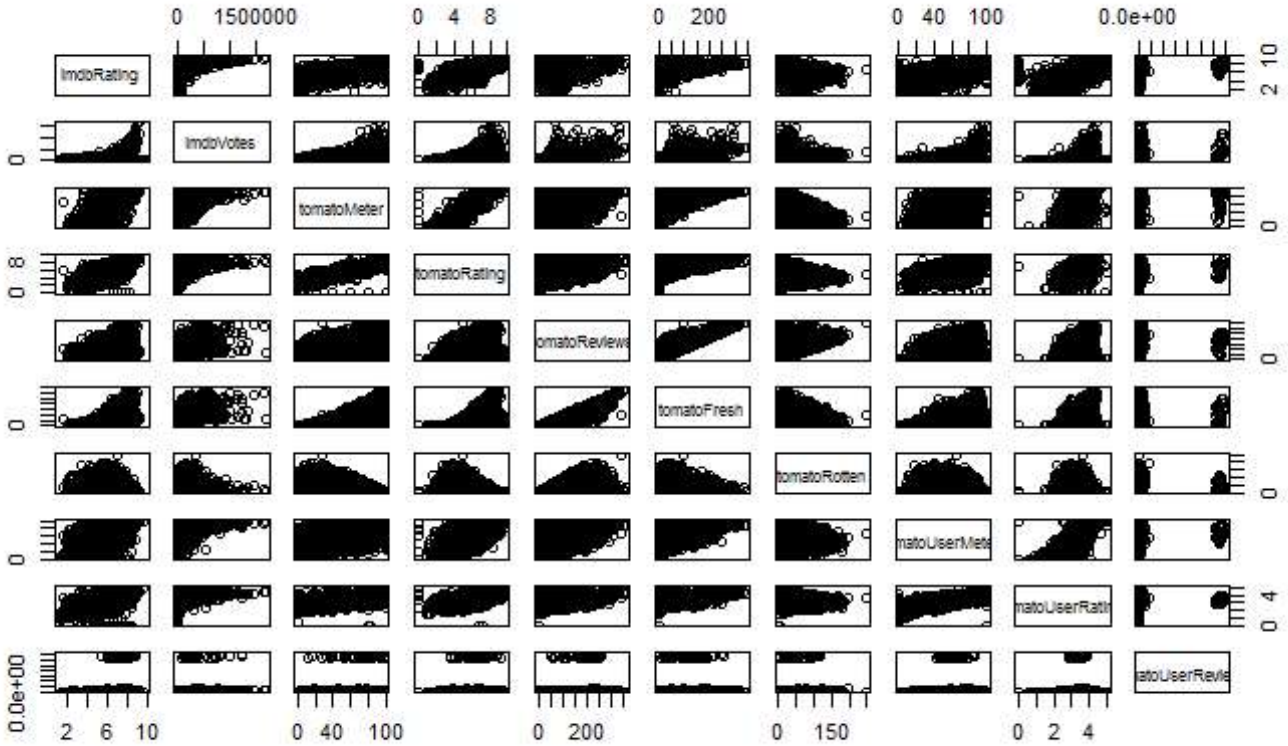
There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by several variables pre-fixed by `tomato`). Read up on such ratings on the 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
pairs(df[,c("imdbRating", "imdbVotes", "tomatoMeter", "tomatoRating","tomatoReviews" , "tomatoFresh"
 ,"tomatoRotten" ,"tomatoUserMeter" , "tomatoUserRating","tomatoUserReviews")])
```



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

A: IMDB ratings does not have a strong self correlation, however, tomato rating shows a very strong self correlations. For example, TomatoReviews almost form a linear relationship with TomatoFresh. Similarities: IMDB and Rotten Tomatoes all centralized over a range of rating scores. Diffirences: There are some Rotten Tomatoes ratings which distributed over two polar, like TomotoUserReviews.

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.

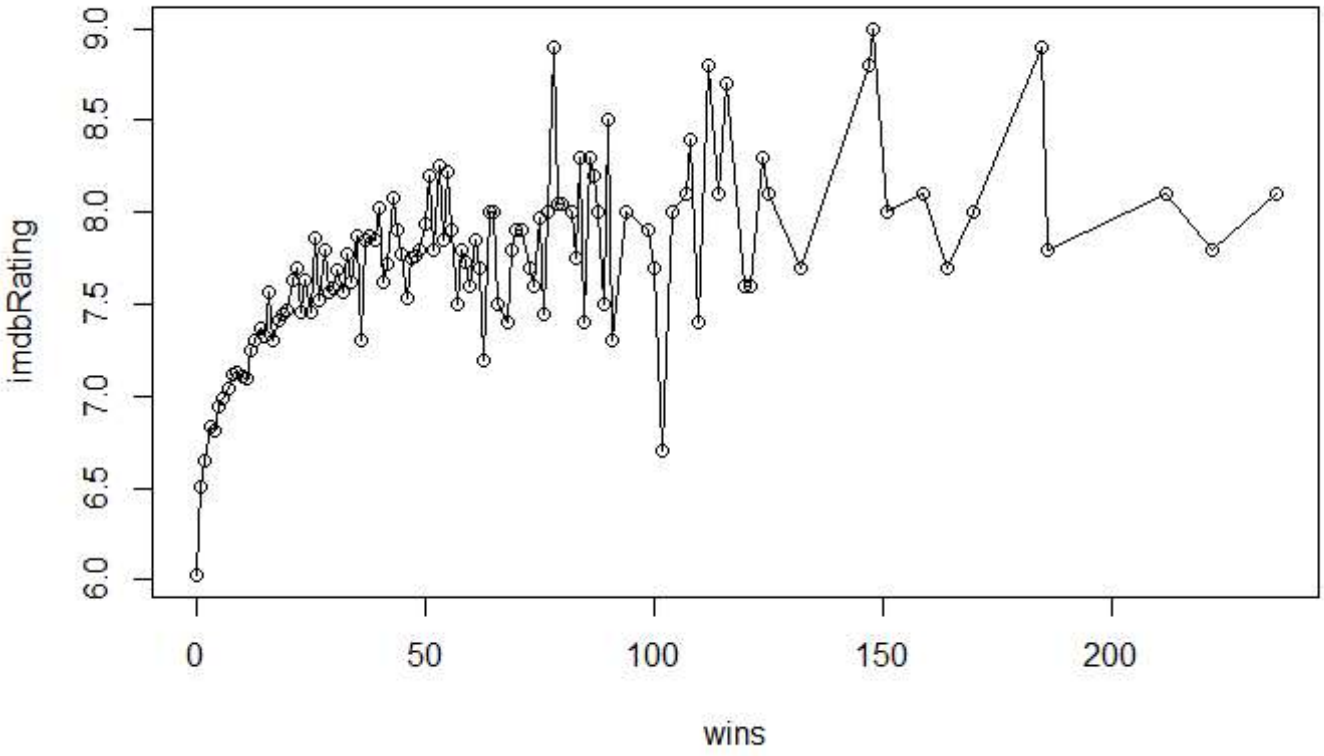
Study the relationship between ratings and awards using graphs (awards here refers to wins and/or nominations).

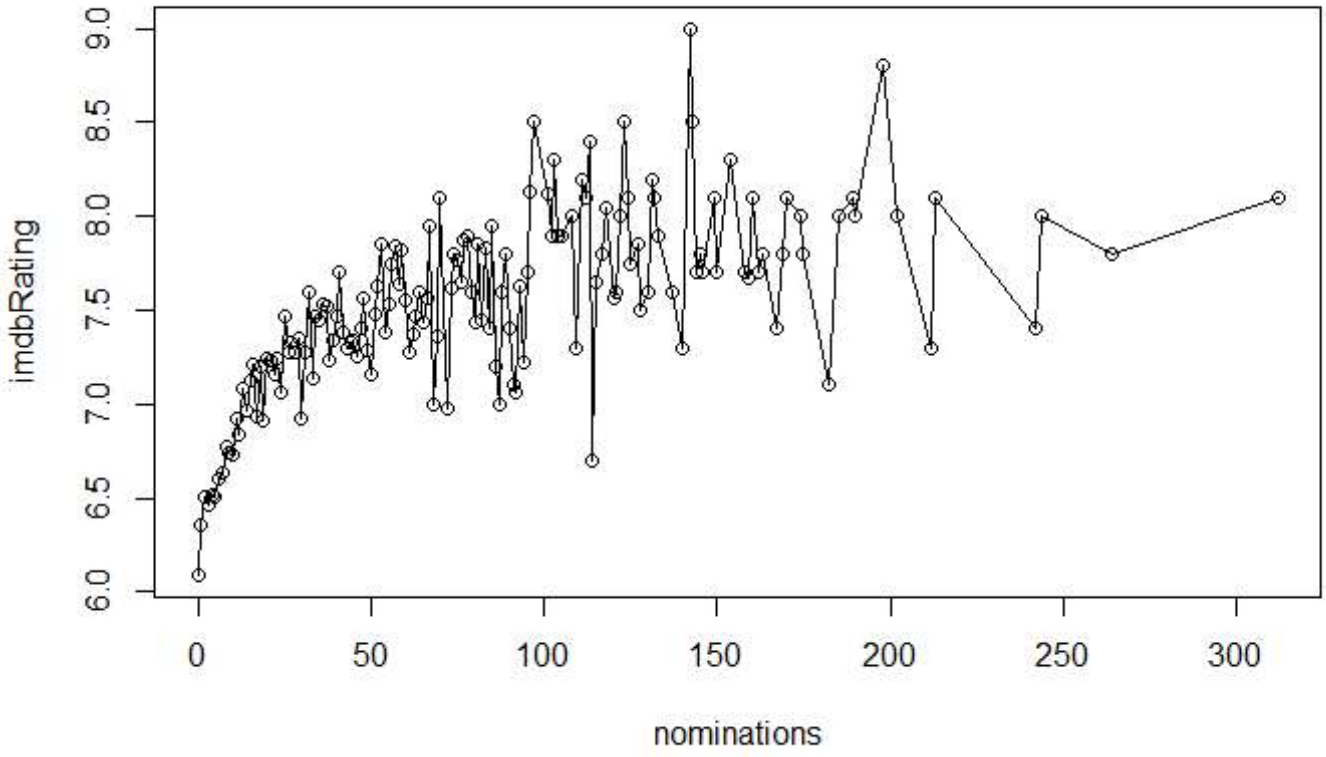
Hide

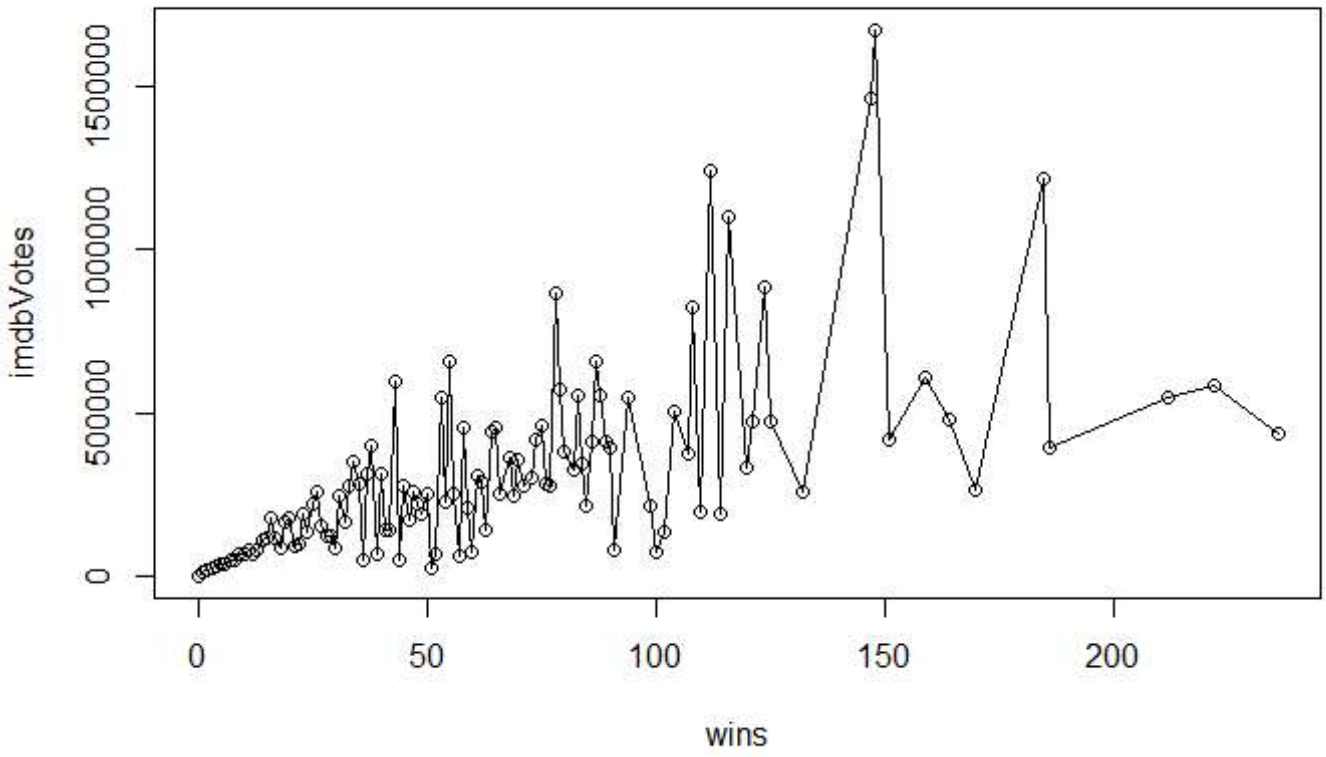
```
# TODO: Show how ratings and awards are related
ratings=c("imdbRating", "imdbVotes", "tomatoMeter", "tomatoRating","tomatoReviews" , "tomatoFresh"
,"tomatoRotten" ,"tomatoUserMeter" , "tomatoUserRating","tomatoUserReviews")
for(rating in ratings){
  assign(rating, df[!is.na(df[,rating]),c(rating, "wins","nominations")])
}
```

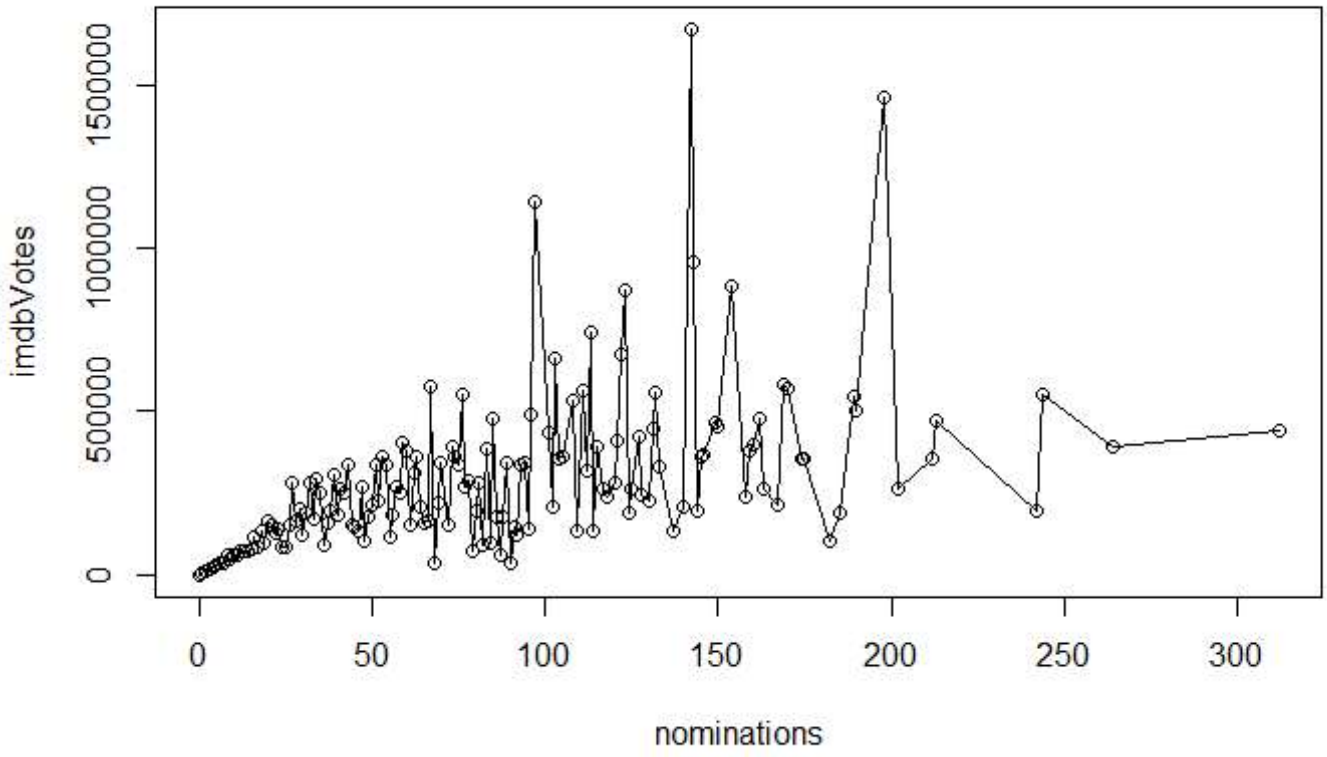
Hide

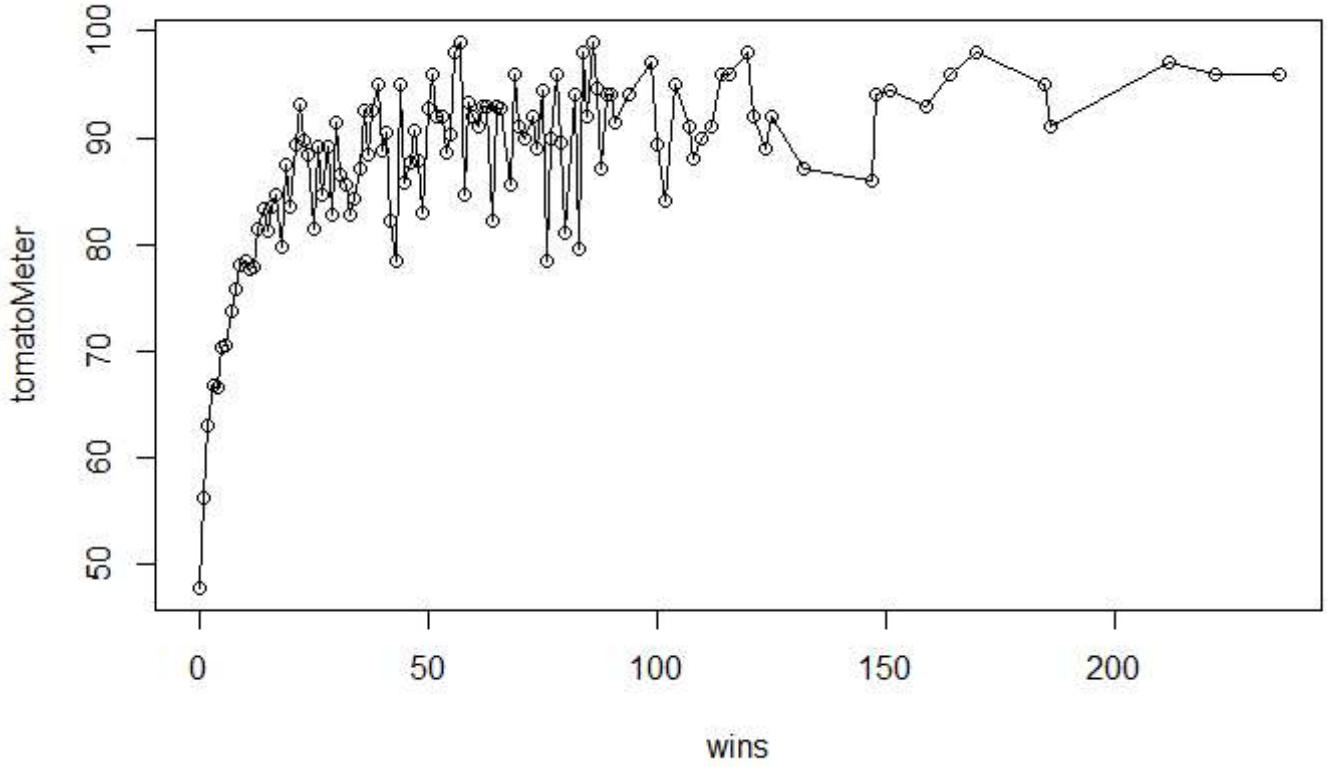
```
# For each rating, display their distribution over wins and nominations
for(rating in ratings){
  plot(aggregate(get(rating)[,rating], list(get(rating)[,"wins"]), mean), type="o", xlab="wins", ylab=rati
ng)
  plot(aggregate(get(rating)[,rating], list(get(rating)[,"nominations"]), mean), type="o", xlab="nominatio
ns", ylab=rating)
}
```

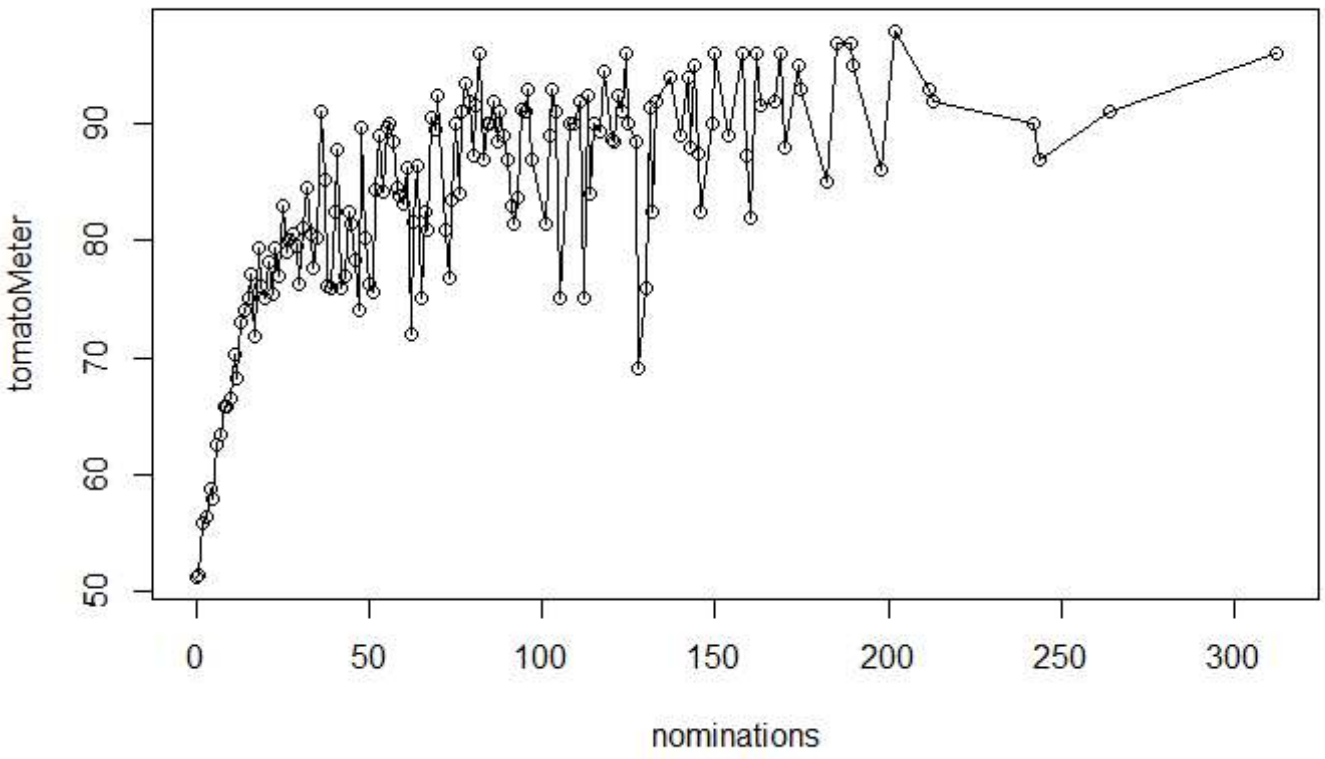


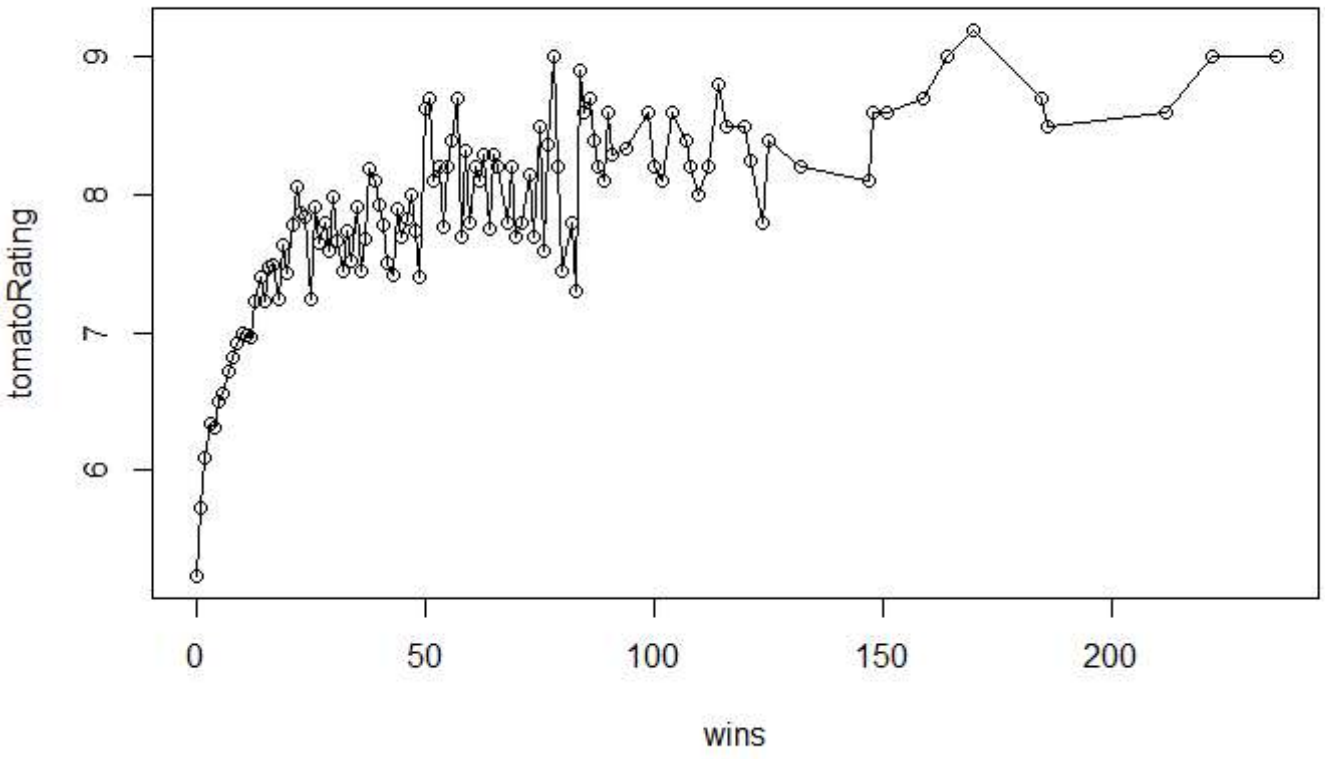

imdbRating
6.0
6.5
7.0
7.5
8.0
8.5
9.0
0
50
100
150
200
250
300
nominations

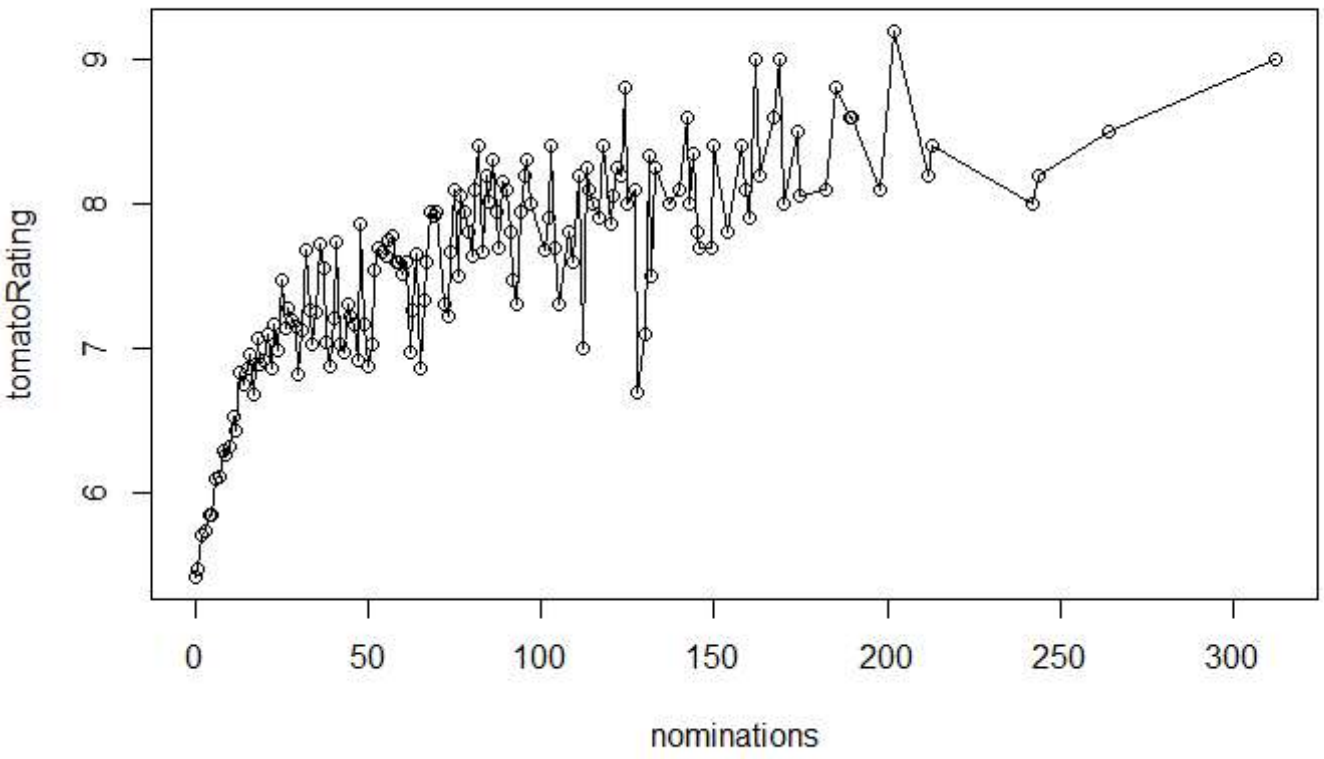

imdbVotes
0
500000
1000000
1500000
0
50
100
150
200
wins

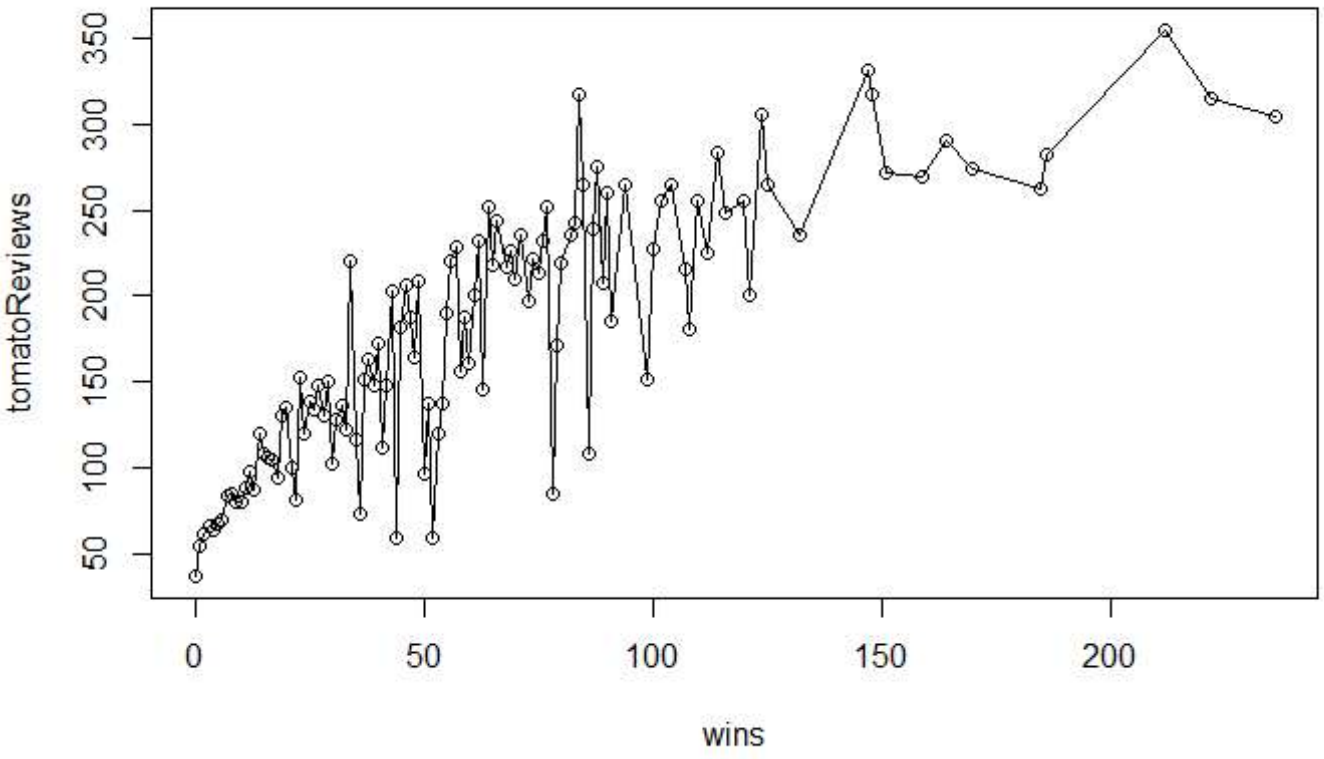

imdbVotes
0
500000
1000000
1500000
0
50
100
150
200
250
300
nominations

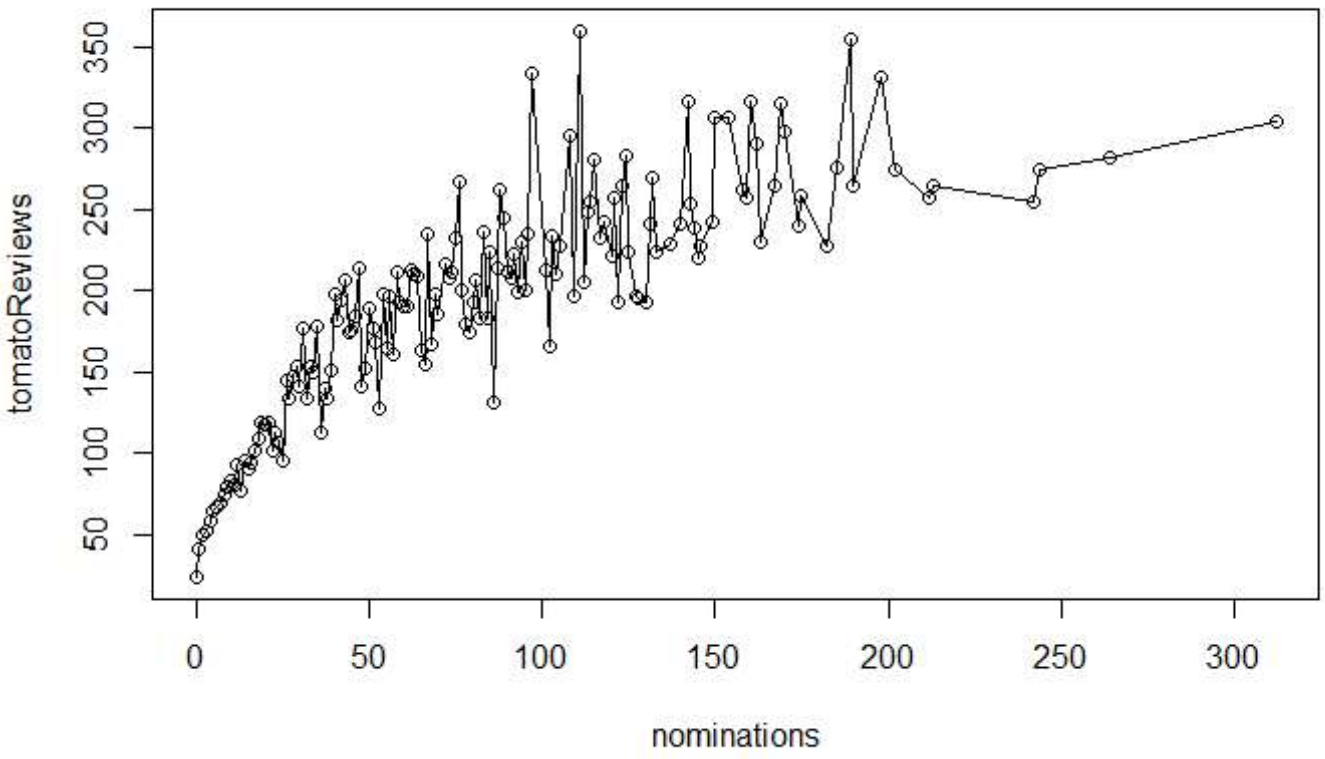

tomatoMeter
50
60
70
80
90
100
0
50
100
150
200
wins

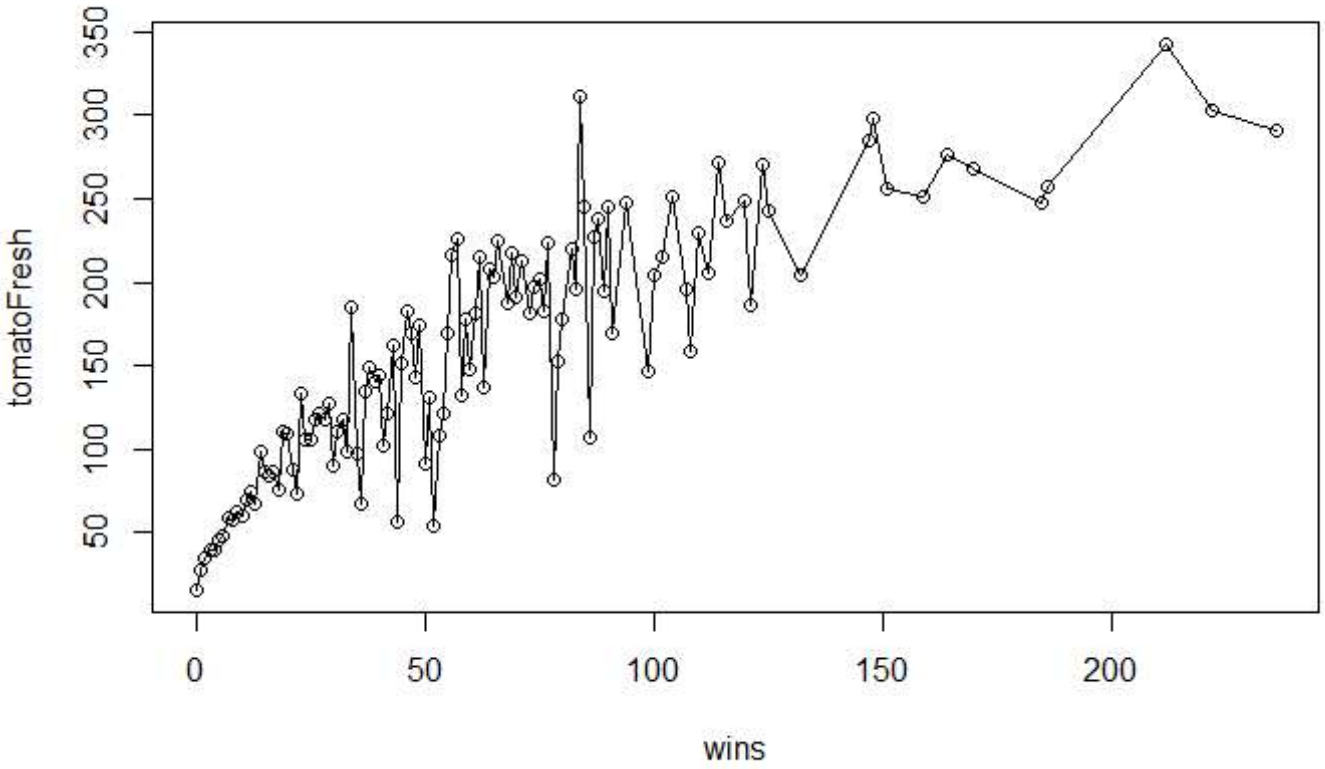

tomatoMeter
50
60
70
80
90
0
50
100
150
200
250
300
nominations

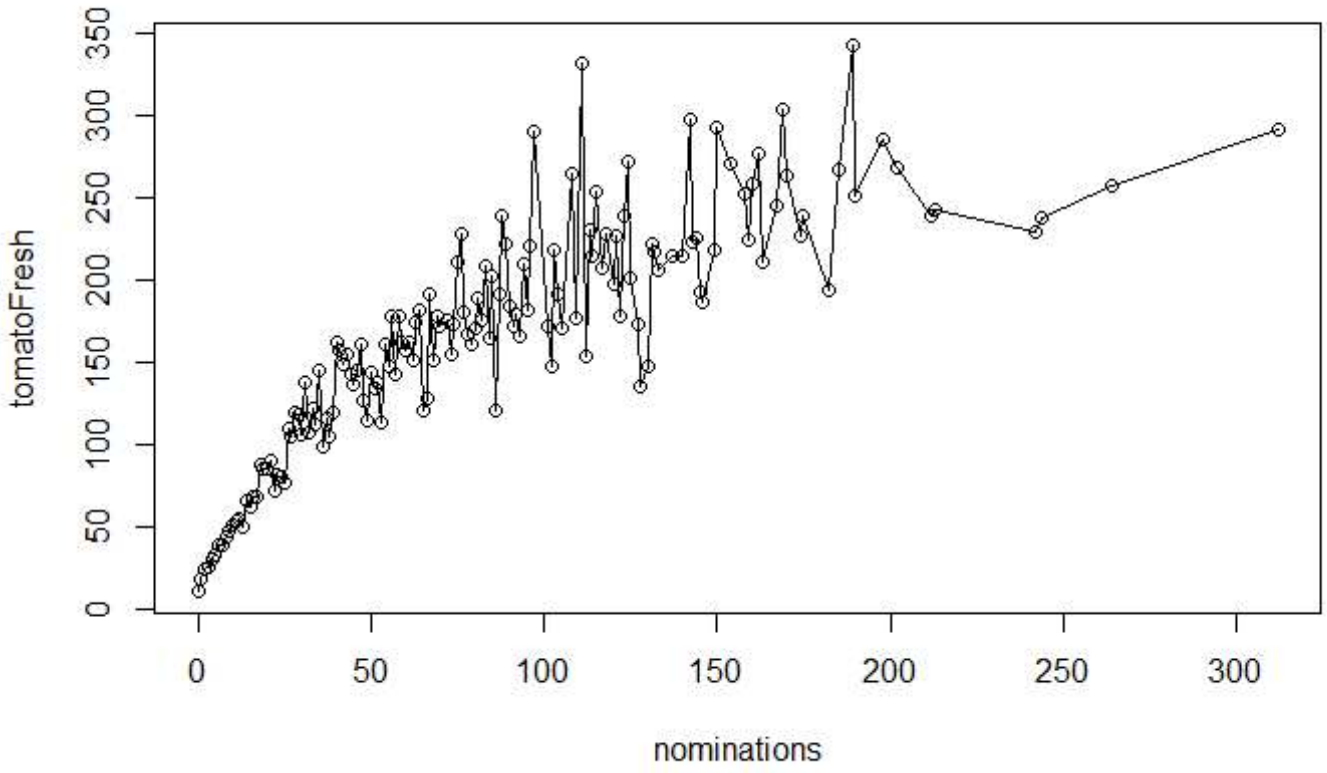

tomatoRating
6
7
8
9
0
50
100
150
200
wins

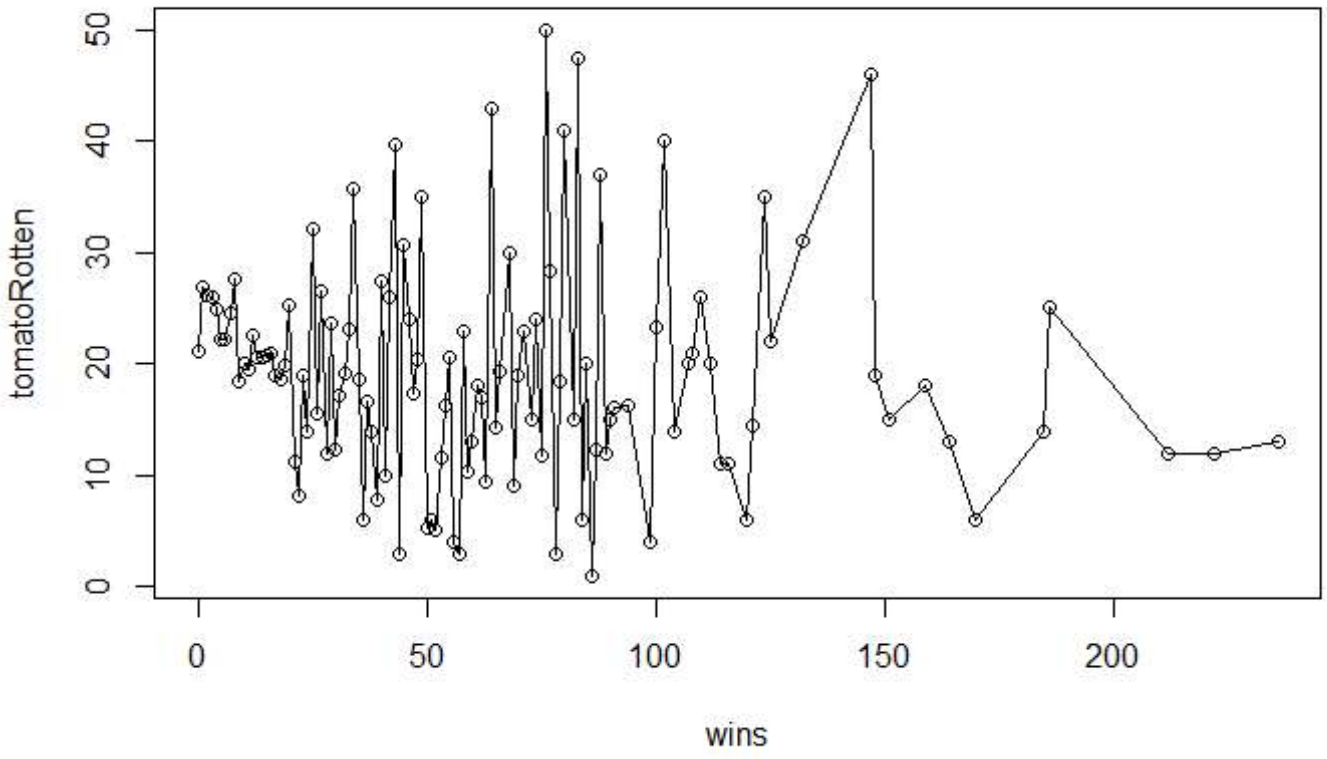

tomatoRating
6
7
8
9
0
50
100
150
200
250
300
nominations

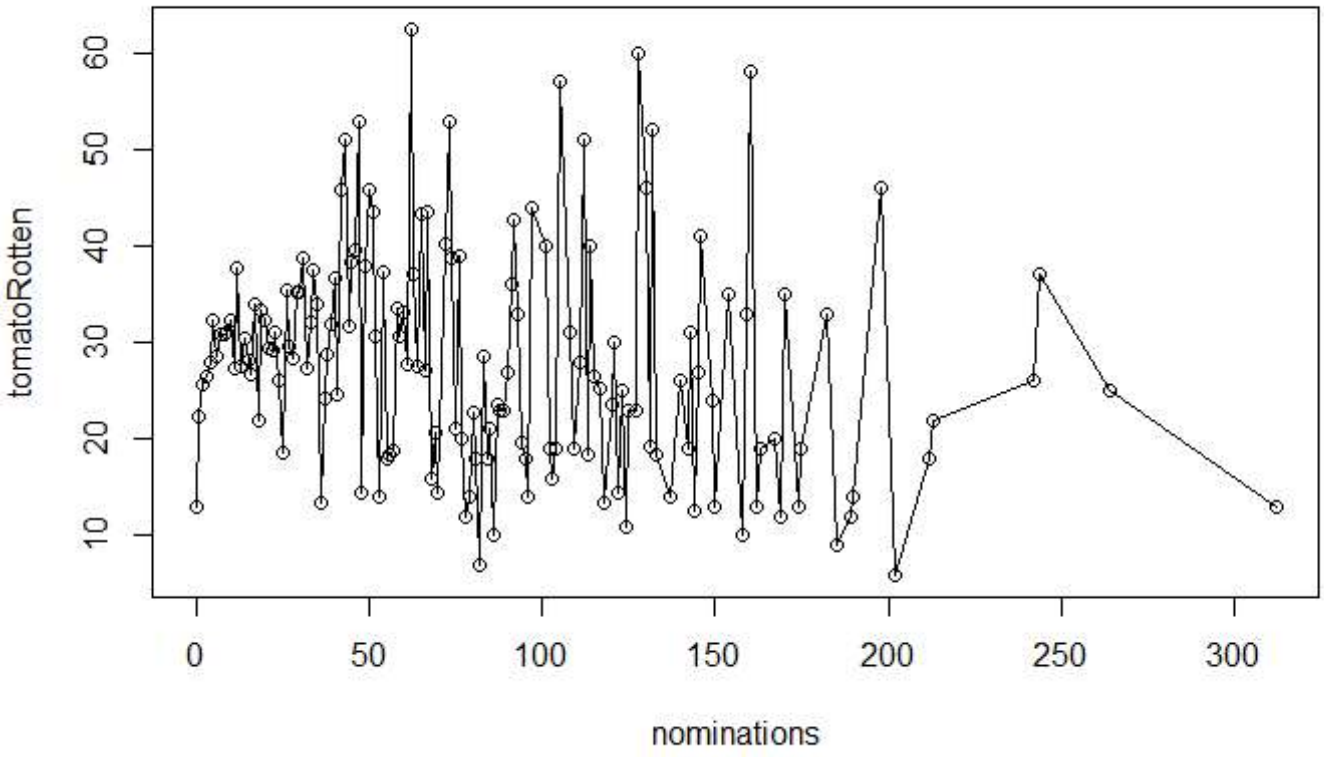

tomatoReviews
50
100
150
200
250
300
350
0
50
100
150
200
wins

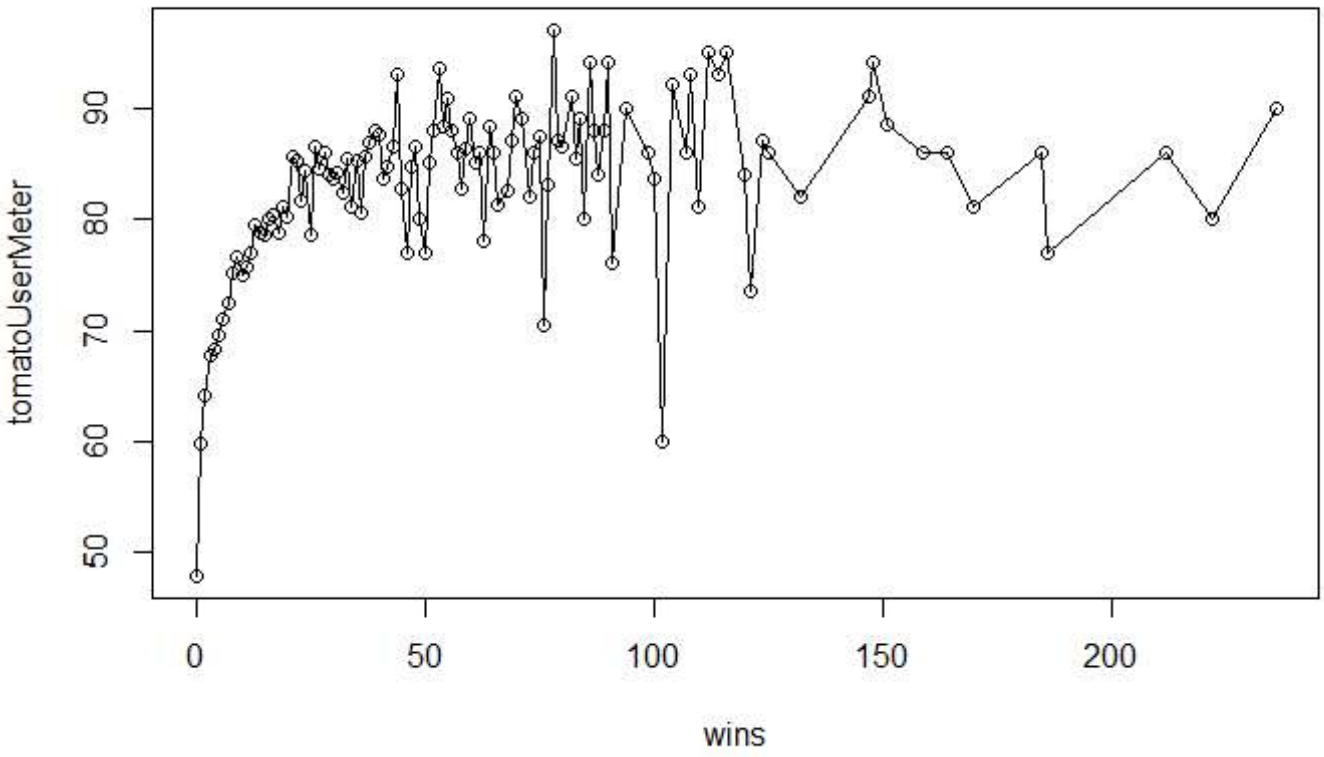

tomatoReviews
50
100
150
200
250
300
350
0
50
100
150
200
250
300
nominations

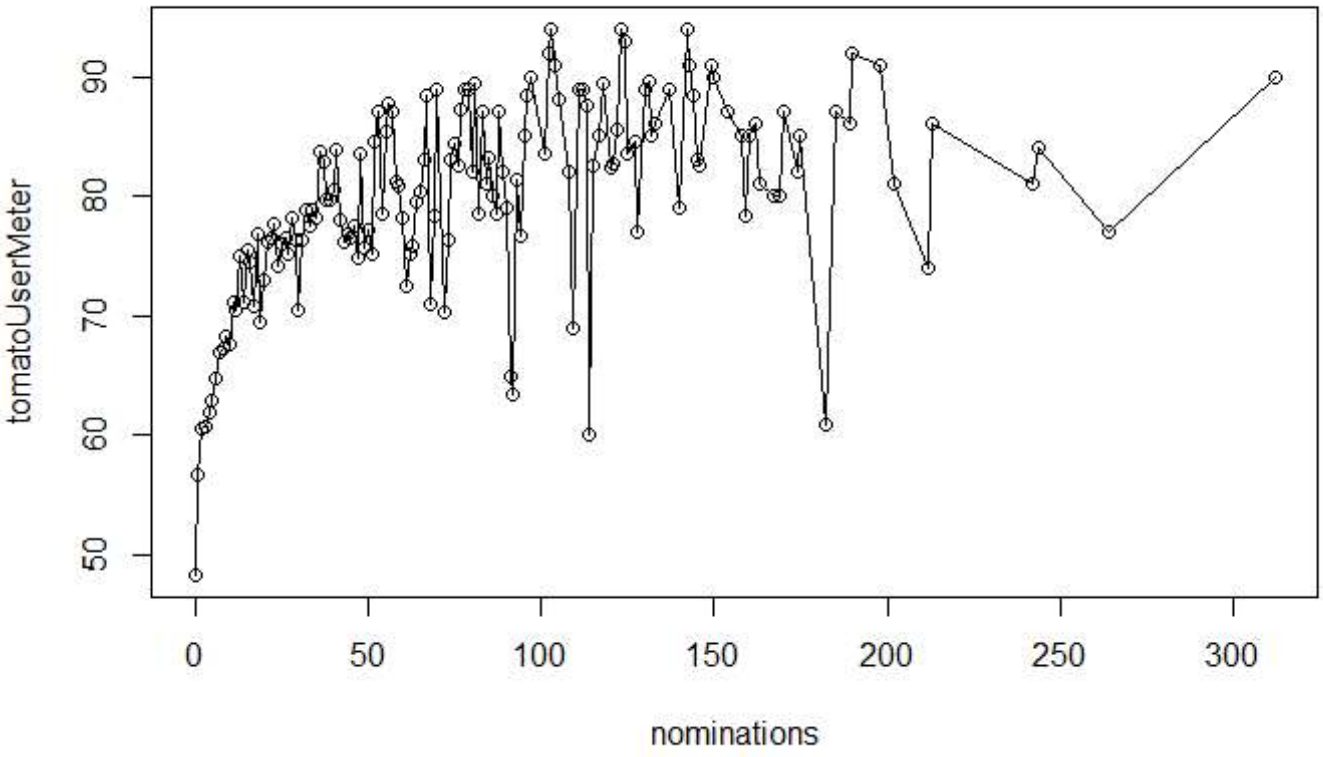

tomatoFresh
50
100
150
200
250
300
350
0
50
100
150
200
wins

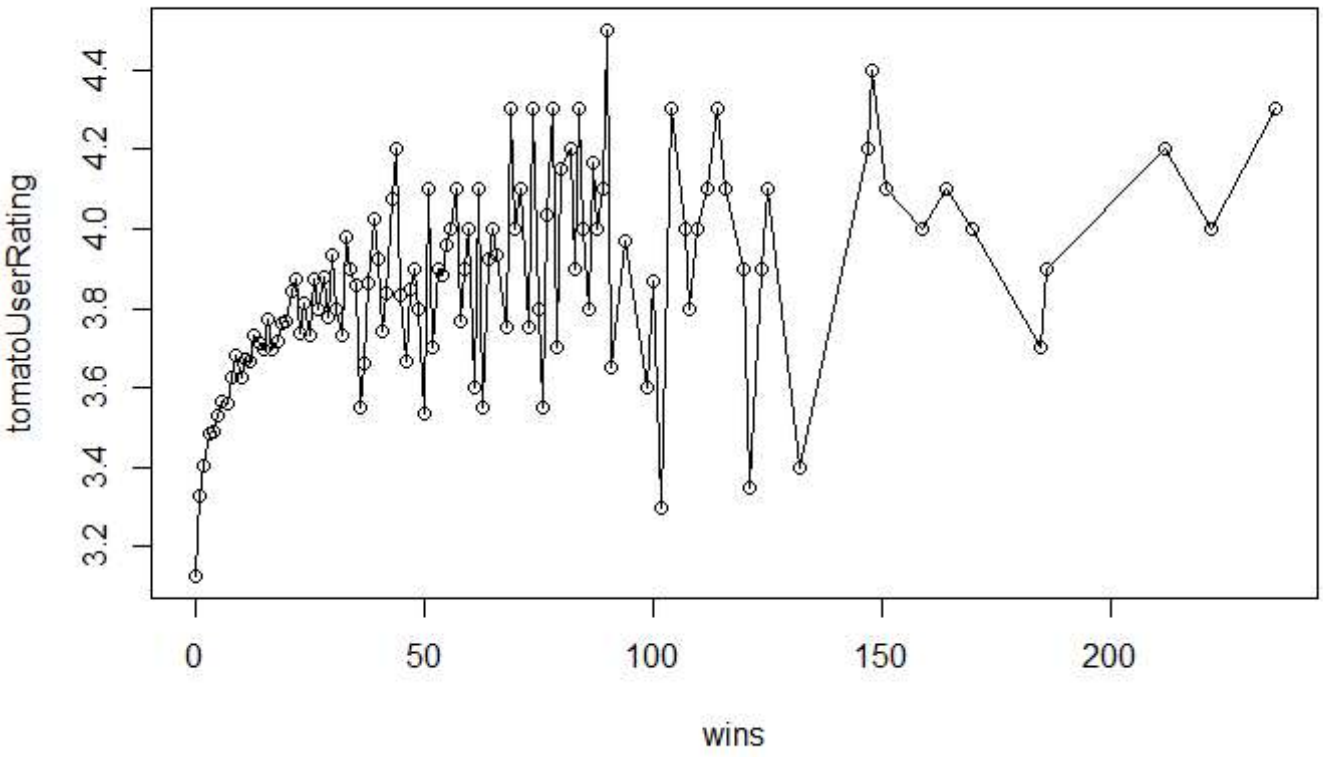

tomatoFresh
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
nominations

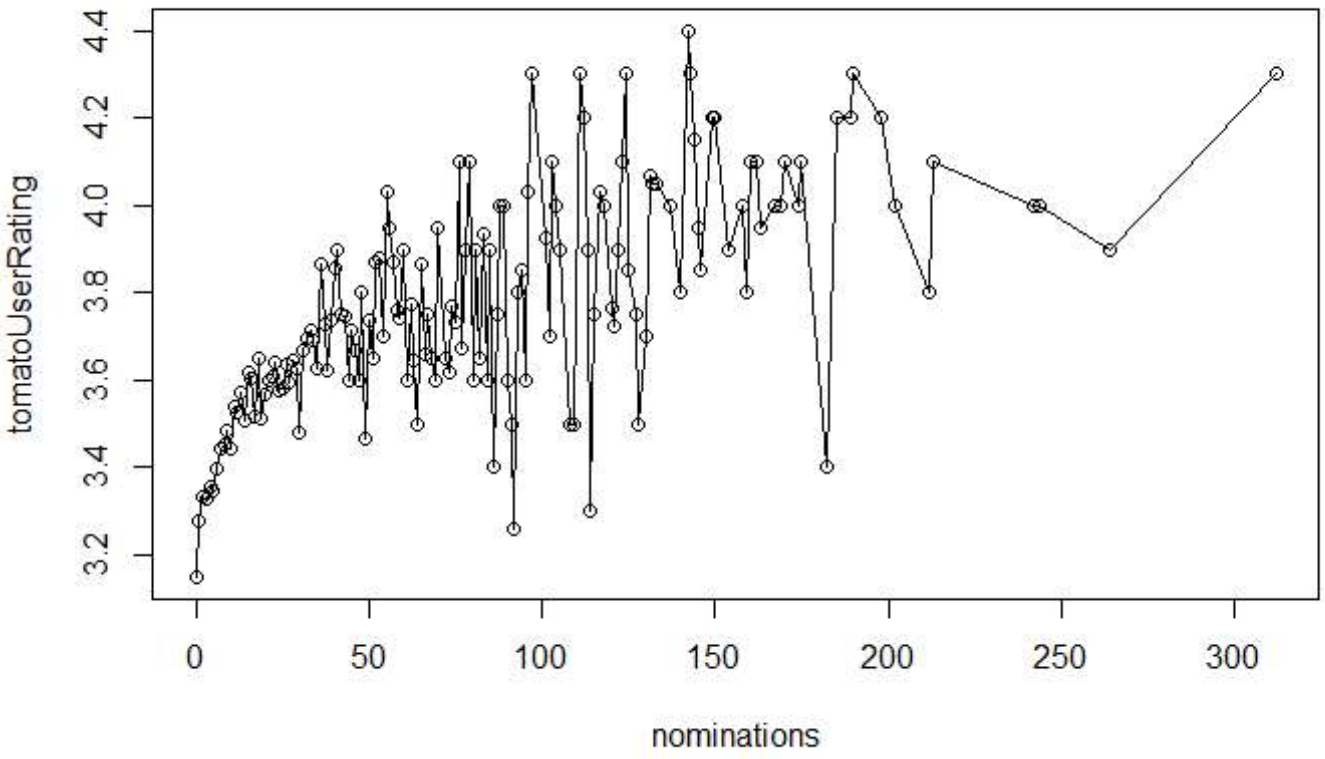

tomatoRotten
0
10
20
30
40
50
0
50
100
150
200
wins

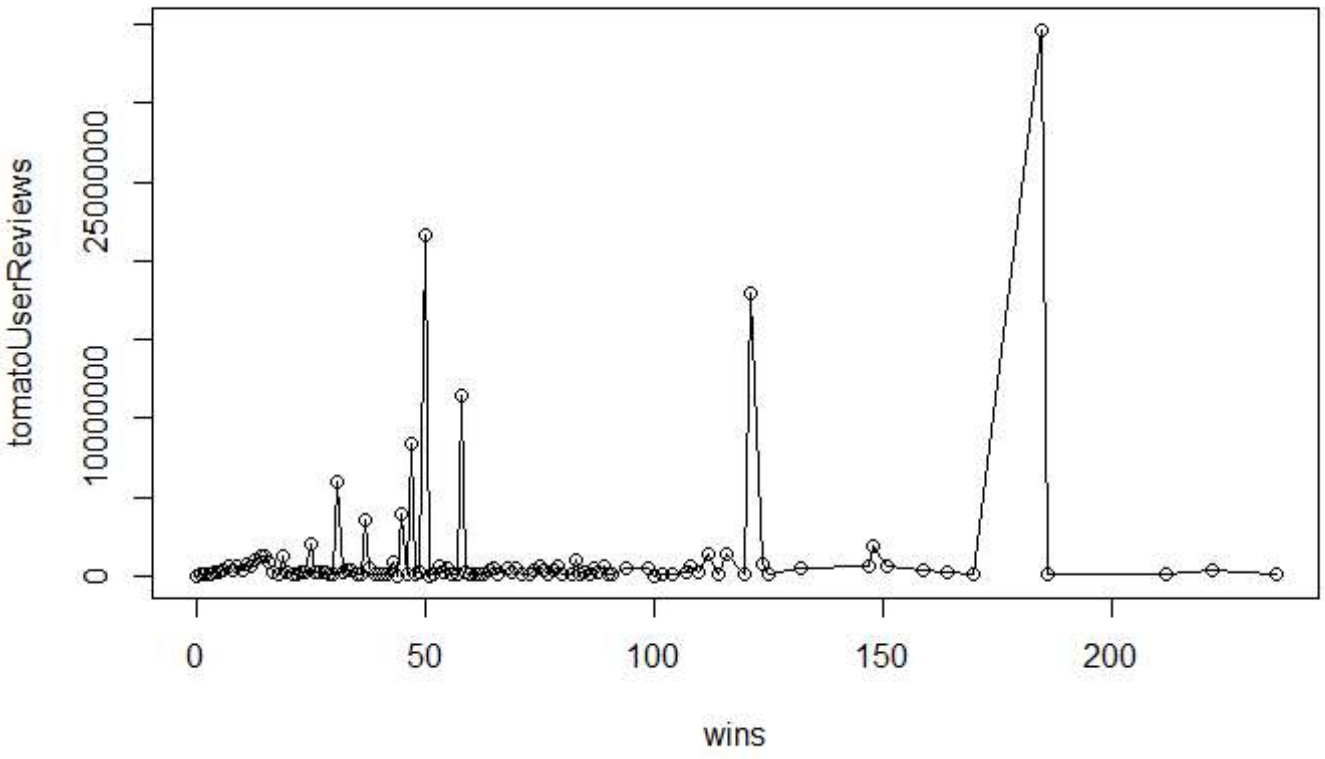

tomatoRotten
10
20
30
40
50
60
0
50
100
150
200
250
300
nominations

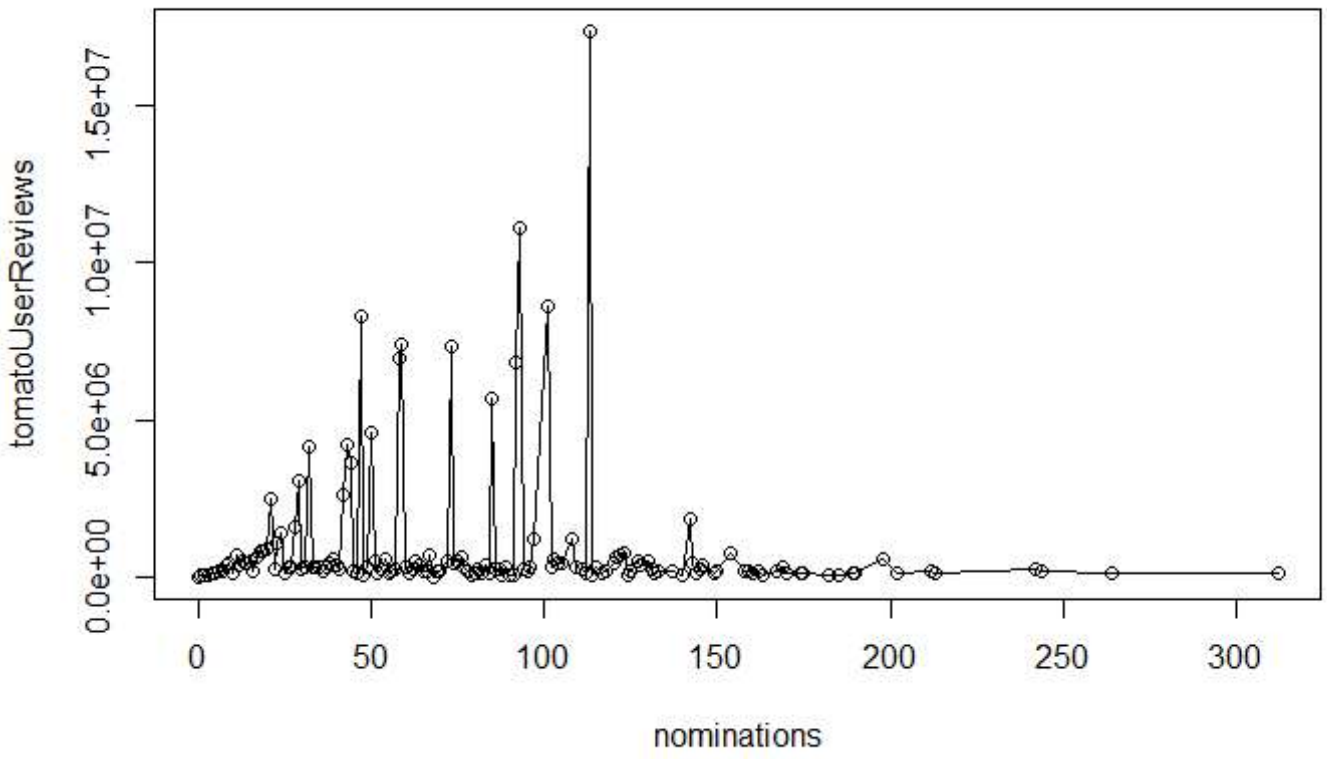

tomatoUserMeter
50
60
70
80
90
0
50
100
150
200
wins


tomatoUserMeter
50
60
70
80
90
0
50
100
150
200
250
300
nominations


tomatoUserRating
3.2
3.4
3.6
3.8
4.0
4.2
4.4
0
50
100
150
200
wins


tomatoUserRating
3.2
3.4
3.6
3.8
4.0
4.2
4.4
0
50
100
150
200
250
300
nominations


tomatoUserReviews
0
10000000
25000000
0
50
100
150
200
wins



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: We can find that, except “tomatoUserReviews” and “tomatoRotten”, all of other reviews has a strong positive correlation between wins/nominations and review scores.

9. Expected insights

Come up with two new insights (backed up by data and graphs) that is expected. Here “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.

Hide

```
# TODO: Find and illustrate two expected insights
unique(unlist(tokenize_regex(df$Country,pattern=", ")))
```

```
 [1] "USA"                    "UK"                    "France"
 [4] "Canada"                 "Germany"               "Norway"
 [7] "Spain"                  "India"                 "Switzerland"
[10] "Algeria"                "N/A"                   "New Zealand"
[13] "Russia"                 "Philippines"           "Mexico"
[16] "Italy"                  "Netherlands"           "Denmark"
[19] "Australia"              "Finland"               "Sweden"
[22] "Brazil"                 "Argentina"             "Portugal"
[25] "Morocco"                "Hong Kong"             "Belgium"
[28] "South Africa"           "Palestine"             "Israel"
[31] "Singapore"              "Bulgaria"              "Panama"
[34] "Turkey"                 "Ireland"               "Austria"
[37] "Iran"                   "Guatemala"             "China"
[40] "Luxembourg"             "Egypt"                 "Taiwan"
[43] "Hungary"                "Romania"               "Greece"
[46] "Chile"                  "Costa Rica"            "Japan"
[49] "South Korea"            "Colombia"              "Tunisia"
[52] "Croatia"                "Czech Republic"        "West Germany"
[55] "Yugoslavia"             "Poland"                "Albania"
[58] "Indonesia"              "Ecuador"               "Paraguay"
[61] "Uruguay"                "Belarus"               "Soviet Union"
[64] "Azerbaijan"             "Dominican Republic"    "Iceland"
[67] "Fiji"                   "Malaysia"              "Cuba"
[70] "Bahamas"                "Malta"                 "Slovenia"
[73] "Cyprus"                 "Estonia"               "Latvia"
[76] "Lithuania"              "Slovakia"              "Iraq"
[79] "Serbia and Montenegro"  "Peru"                  "Venezuela"
[82] "Thailand"               "Ukraine"               "Korea"
```

```
 [85] "Cambodia"                      "Kazakhstan"                      "Czechoslovakia"
 [88] "Burkina Faso"                  "Afghanistan"                     "Aruba"
 [91] "Georgia"                       "Bosnia and Herzegovina"          "Jamaica"
 [94] "East Germany"                  "Armenia"                         "Chad"
 [97] "Uzbekistan"                    "Haiti"                           "Gabon"
[100] "Nigeria"                       "Jordan"                          "Mali"
[103] "Republic of Macedonia"         "United Arab Emirates"            "Lebanon"
[106] "Federal Republic of Yugoslavia" "The Democratic Republic Of Congo" "Kyrgyzstan"
[109] "Guinea"                        "Bangladesh"                      "Tanzania"
[112] "Kenya"                         "Saudi Arabia"                    "Ghana"
[115] "Honduras"                      "Benin"                           "Namibia"
[118] "Netherlands Antilles"          "Turks And Caicos Islands"        "Zimbabwe"
[121] "Ethiopia"                      "Qatar"                           "Tajikistan"
[124] "Puerto Rico"                   "Vietnam"                         "Serbia"
[127] "Pakistan"                      "Senegal"                         "Côte d'Ivoire"
[130] "Montenegro"                    "Cameroon"                        "Trinidad and Tobago"
[133] "Zaire"                         "Guinea-Bissau"                   "Mongolia"
[136] "Turkmenistan"                  "Liechtenstein"                   "Papua New Guinea"
[139] "Syria"                         "Nicaragua"                       "North Korea"
[142] "Gibraltar"                     "Mauritania"                      "Monaco"
[145] "Antarctica"                    "Micronesia"                      "Solomon Islands"
[148] "Vanuatu"                       "Niger"                           "Angola"
[151] "Nepal"                         "Bermuda"                         "Isle Of Man"
[154] "Moldova"                       "Bolivia"                         "Guyana"
[157] "Libya"
```

Hide

```
#Combine the genre with the original dataset
column_country_names=unique(unlist(tokenize_regex(df$Country,pattern=", ")))
country=data.frame(matrix(nrow=nrow(df), ncol=length(column_country_names)))
colnames(country)=column_country_names
country[]=0
for(val in column_country_names){
  country[grep(val, df[, "Country"]), val]=1
}
```

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
country_sum=colSums(country)
#Sort the genres based on the number of occurences
country_sort=sort(country_sum, decreasing=TRUE)
print(country_sort)
```

USA	UK	France
25063	3635	2291
Germany	Canada	Italy
1982	1977	1236
Japan	India	Spain
780	725	714
Australia	N/A	West Germany
676	562	483
Soviet Union	Sweden	Hong Kong
431	398	398
Netherlands	Denmark	Poland
387	381	361
Mexico	Belgium	Brazil
294	270	245
Switzerland	Argentina	Finland
244	227	226
Yugoslavia	Norway	Russia
216	203	198
Ireland	Austria	Korea
198	192	186
South Korea	Israel	Philippines
184	175	169
China	Greece	South Africa
142	131	130
New Zealand	Bulgaria	Hungary
124	120	118
Romania	Czechoslovakia	Turkey
100	91	89
Czech Republic	Taiwan	Portugal
89	86	77
Iran	Luxembourg	Chile
76	56	48
Thailand	Cuba	Indonesia
45	43	41
East Germany	Iceland	Egypt
39	32	30
Croatia	Peru	Slovenia
26	24	23
United Arab Emirates	Singapore	Federal Republic of Yugoslavia
23	21	21
Morocco	Malaysia	Ukraine
20	19	17
Colombia	Slovakia	Estonia
16	16	15
Tunisia	Senegal	Algeria
14	14	12
Venezuela	Burkina Faso	Palestine
11	11	10
Kazakhstan	Bosnia and Herzegovina	Uruguay
10	10	9
Georgia	Jamaica	Lebanon
9	8	8
Puerto Rico	Liechtenstein	Panama
8	8	7
Dominican Republic	Armenia	Mali

7	7	7
Latvia	Lithuania	Guinea
6	6	6
Belarus	Cyprus	Ghana
5	5	5
Vietnam	Serbia	Pakistan
5	5	5
Niger	Bahamas	Uzbekistan
5	4	4
Haiti	Nigeria	Republic of Macedonia
4	4	4
Kenya	Zimbabwe	Costa Rica
4	4	3
Albania	Malta	Iraq
3	3	3
Chad	Jordan	Bangladesh
3	3	3
Tajikistan	Cameroon	Guinea-Bissau
3	3	3
Isle Of Man	Guatemala	Ecuador
3	2	2
Azerbaijan	Cambodia	Afghanistan
2	2	2
Aruba	Kyrgyzstan	Tanzania
2	2	2
Saudi Arabia	Namibia	Qatar
2	2	2
Montenegro	Trinidad and Tobago	Zaire
2	2	2
Papua New Guinea	Mauritania	Monaco
2	2	2
Antarctica	Angola	Paraguay
2	2	1
Fiji	Serbia and Montenegro	Gabon
1	1	1
The Democratic Republic Of Congo	Honduras	Benin
1	1	1
Netherlands Antilles	Turks And Caicos Islands	Ethiopia
1	1	1
Côte d'Ivoire	Mongolia	Turkmenistan
1	1	1
Syria	Nicaragua	North Korea
1	1	1
Gibraltar	Micronesia	Solomon Islands
1	1	1
Vanuatu	Nepal	Bermuda
1	1	1
Moldova	Bolivia	Guyana
1	1	1
Libya		
1		

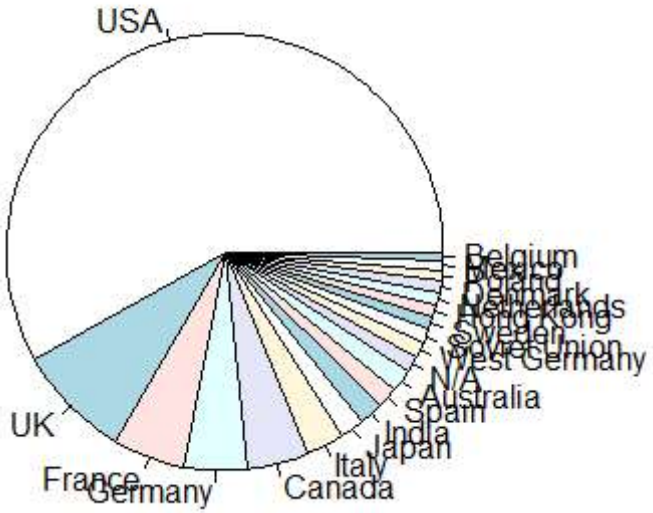
Hide

```
#Calculate the percentage of movies by country
country_top=country_sort[1:20]
country_prop=country_top/nrow(df)
print(country_prop)
```

```
       USA          UK      France     Germany      Canada       Italy       Japan       India
 0.626575    0.090875    0.057275    0.049550    0.049425    0.030900    0.019500    0.018125
     Spain   Australia         N/A West Germany Soviet Union      Sweden   Hong Kong Netherlands
 0.017850    0.016900    0.014050    0.012075    0.010775    0.009950    0.009950    0.009675
   Denmark      Poland      Mexico     Belgium
 0.009525    0.009025    0.007350    0.006750
```

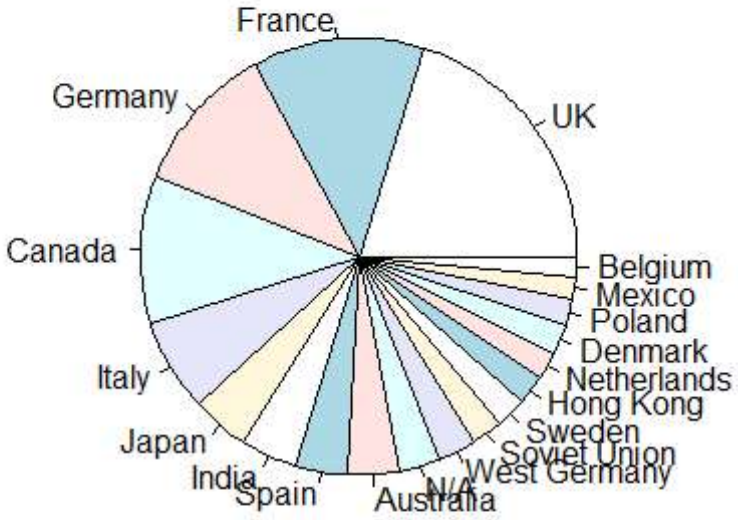
Hide

```
#Plot the number by pie chart
pie(country_prop)
```



Hide

```
#Take the USA out and replot the data
pie(country_sort[2:20]/nrow(df))
```



Q: Expected insight #1.

A: Since USA has the biggest movie industry in the world, it is very normal to see that over half of the movies was produced by USA.

Q: Expected insight #2.

A: With few exceptions, movies all come from developed countries. Since movie industry requires a lot of investment, it is normal to see the absence of developing countries and under-developed countries.

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
# To investigate the relationship between gross revenue and countries
country_gross=cbind(df[, c("Gross", "Domestic_Gross", "Budget")], country)
# Calculate the the percentage of domestic gross over overall gross
country_gross$gross_perc=country_gross$Domestic_Gross/country_gross$Gross
```

Hide

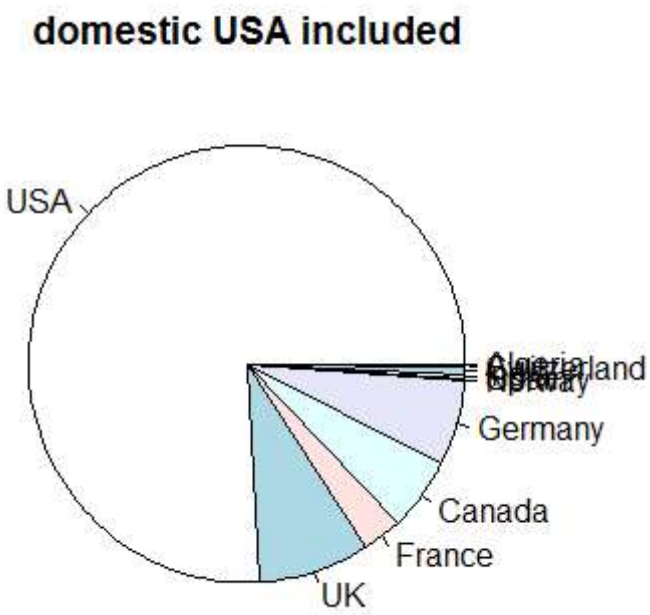
```
domestic_market=country_gross[!is.na(country_gross$gross_perc),]
domestic_market=domestic_market[domestic_market$gross_perc>0.5,]
international_market=country_gross[!is.na(country_gross$gross_perc),]
international_market=international_market[international_market$gross_perc<=0.5,]
```

Hide

```
domestic_market=domestic_market[, !(names(domestic_market) %in% c("Gross", "Domestic_Gross", "Budget", "gr
oss_perc"))]
international_market=international_market[, !(names(international_market) %in% c("Gross",
"Domestic_Gross", "Budget", "gross_perc"))]
```

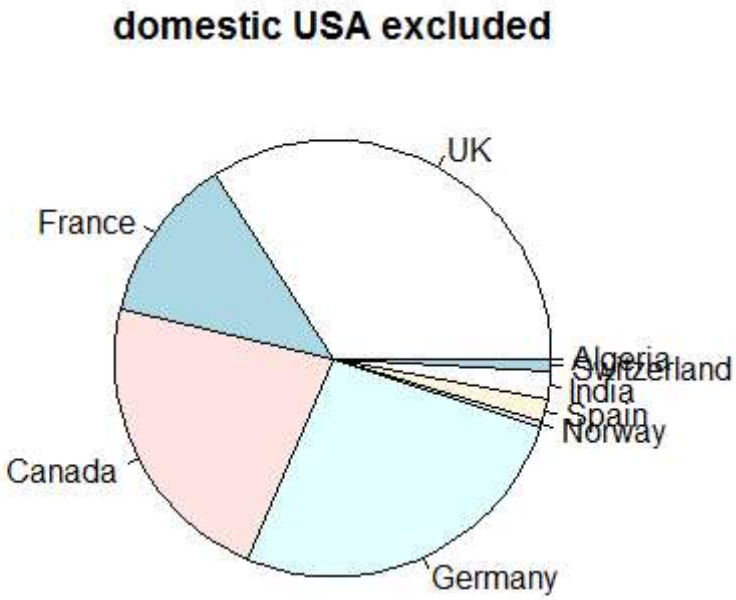
Hide

```
pie(colSums(domestic_market)[1:10], main="domestic USA included")
```



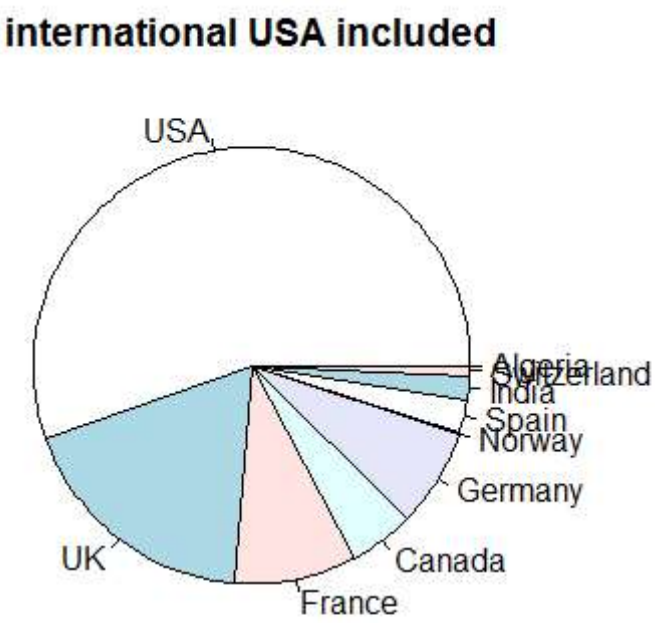
Hide

```
pie(colSums(domestic_market)[2:10], main="domestic USA excluded")
```



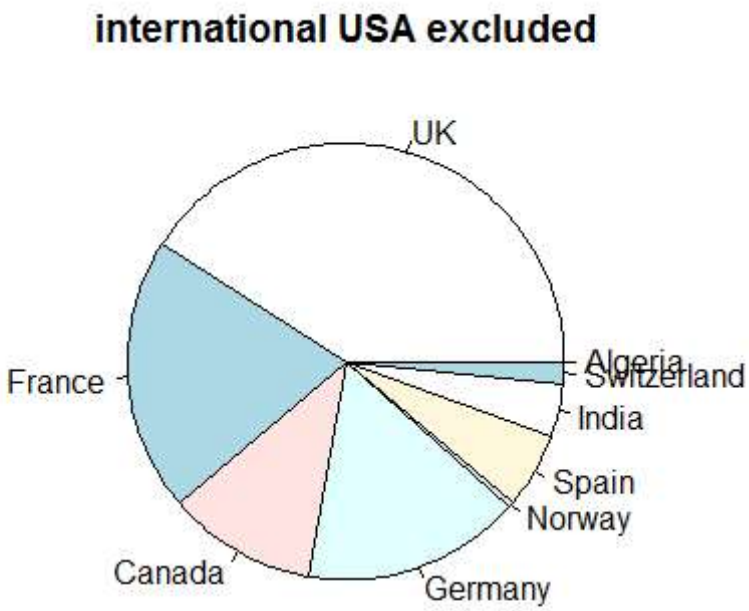
Hide

```
pie(colSums(international_market)[1:10], main="international USA included")
```



Hide

```
pie(colSums(international_market)[2:10], main="international USA excluded")
```



Q: Unexpected insight.

A: Even though USA movie industry is very international, however, most of its movies focus on domestic market. From the first pie chart above, we can find USA account for more than 75% domestic focusing movies compared to about a half in the third pie chart.