TMDB Movies Regression Analysis

2024-10-05

Dataset downloaded from: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata (https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata)

```
#We want to predict how much money a movie has made from the predictor variables in this dataset
credits = read.csv("C:/Users/ryne0/Downloads/tmdb_5000_credits.csv")
movies = read.csv("C:/Users/ryne0/Downloads/tmdb_5000_movies.csv")
```

Data Cleaning

```
colnames(movies)
```

```
##  [1] "budget"              "genres"              "homepage"
##  [4] "id"                  "keywords"            "original_language"
##  [7] "original_title"      "overview"            "popularity"
## [10] "production_companies" "production_countries" "release_date"
## [13] "revenue"             "runtime"             "spoken_languages"
## [16] "status"              "tagline"             "title"
## [19] "vote_average"        "vote_count"
```

```
#Showing a row of the data
movies[1,]
```

```
##      budget
## 1 237000000
##
genres
## 1 [{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantas
y"}, {"id": 878, "name": "Science Fiction"}]
##                       homepage    id
## 1 http://www.avatarmovie.com/ 19995
##
keywords
## 1 [{"id": 1463, "name": "culture clash"}, {"id": 2964, "name": "future"}, {"id": 3386, "nam
e": "space war"}, {"id": 3388, "name": "space colony"}, {"id": 3679, "name": "society"}, {"id":
3801, "name": "space travel"}, {"id": 9685, "name": "futuristic"}, {"id": 9840, "name": "romanc
e"}, {"id": 9882, "name": "space"}, {"id": 9951, "name": "alien"}, {"id": 10148, "name": "trib
e"}, {"id": 10158, "name": "alien planet"}, {"id": 10987, "name": "cgi"}, {"id": 11399, "name":
"marine"}, {"id": 13065, "name": "soldier"}, {"id": 14643, "name": "battle"}, {"id": 14720, "nam
e": "love affair"}, {"id": 165431, "name": "anti war"}, {"id": 193554, "name": "power relation
s"}, {"id": 206690, "name": "mind and soul"}, {"id": 209714, "name": "3d"}]
##   original_language original_title
## 1                en         Avatar
##
overview
## 1 In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique miss
ion, but becomes torn between following orders and protecting an alien civilization.
##   popularity
## 1   150.4376
##
production_companies
## 1 [{"name": "Ingenious Film Partners", "id": 289}, {"name": "Twentieth Century Fox Film Corpo
ration", "id": 306}, {"name": "Dune Entertainment", "id": 444}, {"name": "Lightstorm Entertainme
nt", "id": 574}]
##                                                                                          produ
ction_countries
## 1 [{"iso_3166_1": "US", "name": "United States of America"}, {"iso_3166_1": "GB", "name": "Un
ited Kingdom"}]
##   release_date    revenue runtime
## 1   2009-12-10 2787965087     162
##                                                                      spoken_languages
## 1 [{"iso_639_1": "en", "name": "English"}, {"iso_639_1": "es", "name": "Espa\\u00f1ol"}]
##     status                     tagline  title vote_average vote_count
## 1 Released Enter the World of Pandora. Avatar          7.2      11800
```

```
#Based on the column names as well as the structure of the columns' data, we decide to remove so
me columns as their inclusion wouldn't improve predictions.

movies = subset(movies, select = -c(overview, homepage, tagline))



#Checks if the original title matches the title that the movie was published with
#1 if the original title does match, 0 if it doesn't


movies$same_title = ifelse(movies$title == movies$original_title, 1, 0)
movies = subset(movies, select = -c( original_title))
num_rows_include_NA = nrow(movies)


#Dropping any rows that have NA values
movies = movies[complete.cases(movies),]
num_rows_no_NA = nrow(movies)

print(paste0("The number of rows that were removed due to containing NA values were ", num_rows_
include_NA - num_rows_no_NA))
```

```
## [1] "The number of rows that were removed due to containing NA values were 2"
```

```
#Date is in character form, need to change it to Date form
movies$release_date = as.Date(movies$release_date)



not_matching_rows = movies[movies$title != movies$original_title,]
#not_matching_rows[,c(6,15)]

na_rows = movies[!complete.cases(movies),]
```

Functions to clean data further

```r
library(stringr)
#Transform the production_countries column into a count of the number of countries the movie was filmed in.
#It is thought that the more countries a movie is filmed the better the movie may perform in the box office, due to it having settings which are more scenic and appealing to a wider audience.



num_production_countries = function(movies_dataset){
  #The production_countries column in the dataset is clean and formatted in set notation, where the countries involved in the film's production are enclosed within curly brackets {}. So the number of these brackets plus one will be the number of countries that were filmed in


  #Preallocate the vector to make filling it more efficient
  production_countries_count = rep(0, nrow(movies_dataset))


  for (i in seq_along(movies_dataset$production_countries))

    #Counting the number of occurrences of {
    production_countries_count[i] =  str_count(movies_dataset$production_countries[i], "\\{")


  #Add the count as a new column
  movies_dataset$production_countries_new = production_countries_count
  return(movies_dataset)
}



movies = num_production_countries(movies)



#Removing the keywords column because there are too many unique phrases for our model to account for, and that the phrases vary significantly in topic and description.
movies = subset(movies, select = -keywords)



#Tells us whether the movie was originally in English or not. It is hypothesized that English speaking movies will perform better since English is the most known language in the world.
movies$is_english_movie = ifelse(movies$original_language=="en", 1, 0)


#Also remove production_countries and spoken_languages since they are no longer needed
movies = subset(movies, select = -c(spoken_languages, production_countries, original_language))
```

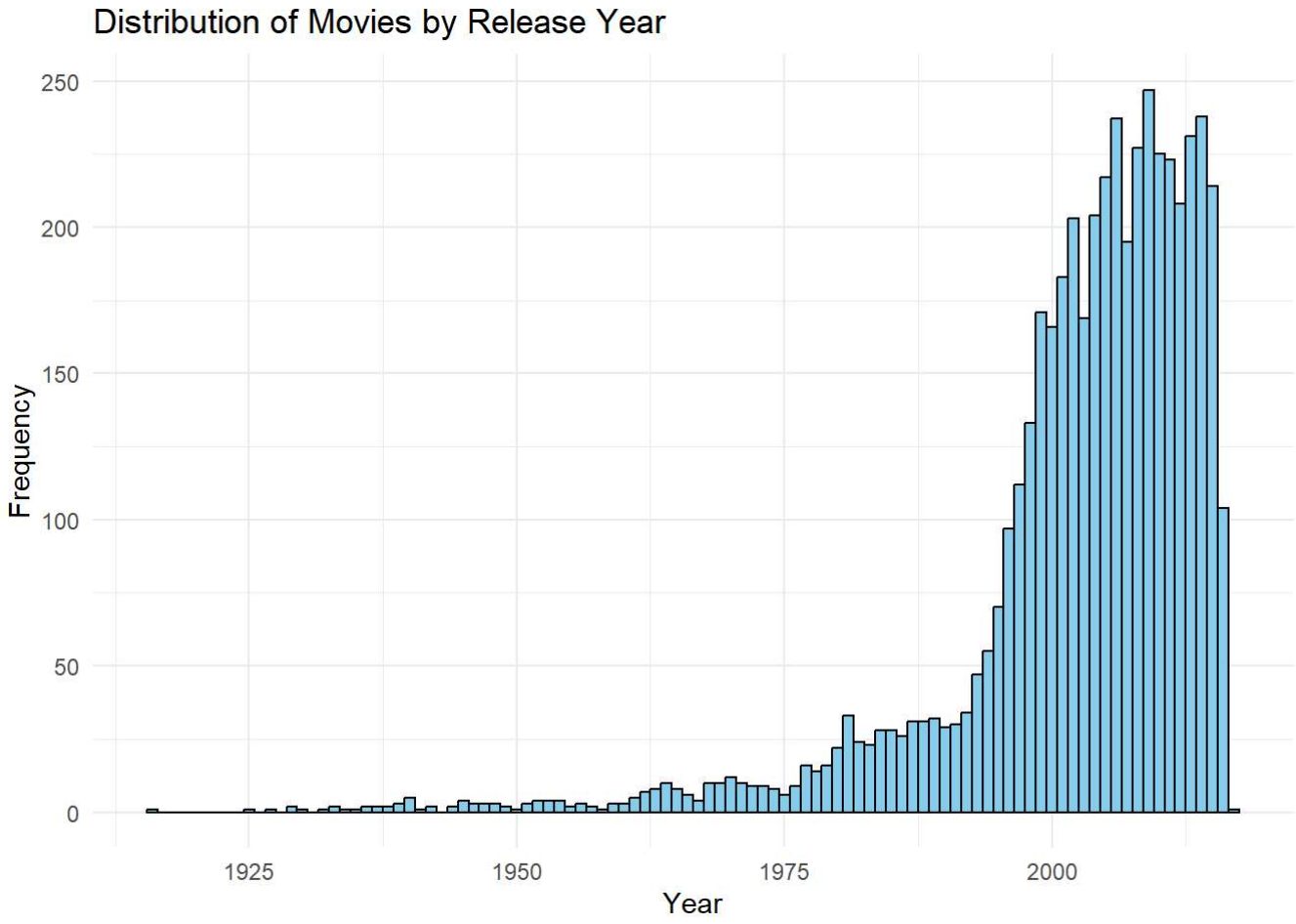
```
#Values of 1 will be assigned to action and adventure movies
#2 will be given to drama, comedy and horror
#3 will be thriller and romantic comedy movies
#and 4 will be all the rest

#This decision is supported by data from box office revenue on movies, such as at the following link:
#https://www.statista.com/statistics/188658/movie-genres-in-north-america-by-box-office-revenue-since-1995/

categorize_movies_by_genre = function(movies_dataset) {

  #Preallocate the vector to make filling it more efficient
  genre_type = rep(0, nrow(movies_dataset))


  #If a movie has multiple genres, it will be assigned the value of the genre which gives it the lowest number
  for (i in seq_along(movies_dataset$genres)) {

    if (grepl("Action|Adventure", movies_dataset$genres[i]))
      genre_type[i] = 1

    else if (grepl("Drama|Comedy|Horror", movies_dataset$genres[i]))
      genre_type[i] = 2

    else if (grepl("Thriller|Romantic Comedy", movies_dataset$genres[i]))
      genre_type[i] = 3

    else
      genre_type[i] = 4

  }

  movies_dataset$genre_type = genre_type
  return(movies_dataset)
}



movies = categorize_movies_by_genre(movies)


#Drop the genres column since it is not needed for predicting anymore
movies = subset(movies, select = -genres)


library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

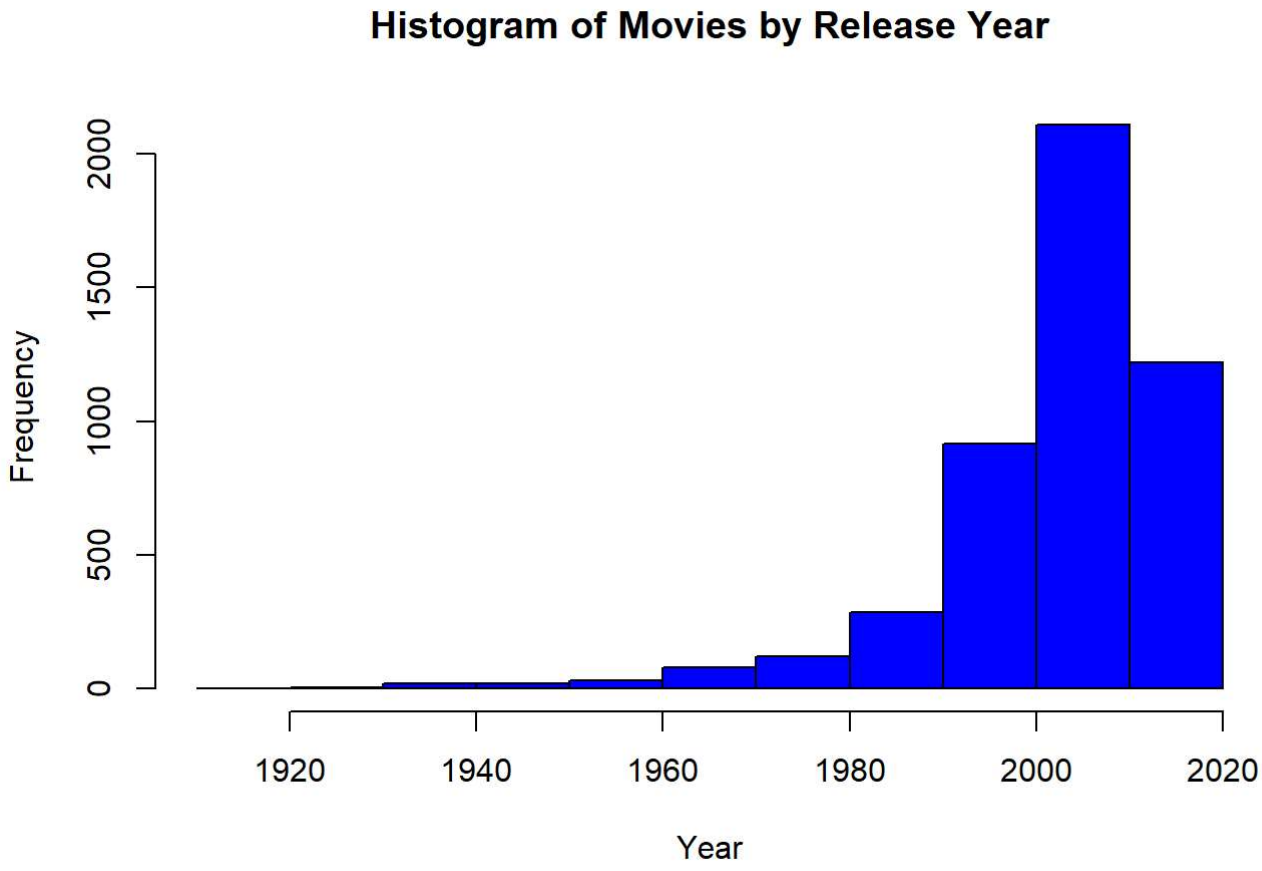
```
movies$year = year(movies$release_date)

#Plot histogram of release years
ggplot(movies, aes(x = year)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Movies by Release Year",
       x = "Year",
       y = "Frequency") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_bin()`).
```



Adjusting data for inflation

```
hist(movies$year, col="blue", main="Histogram of Movies by Release Year", xlab="Year")
```



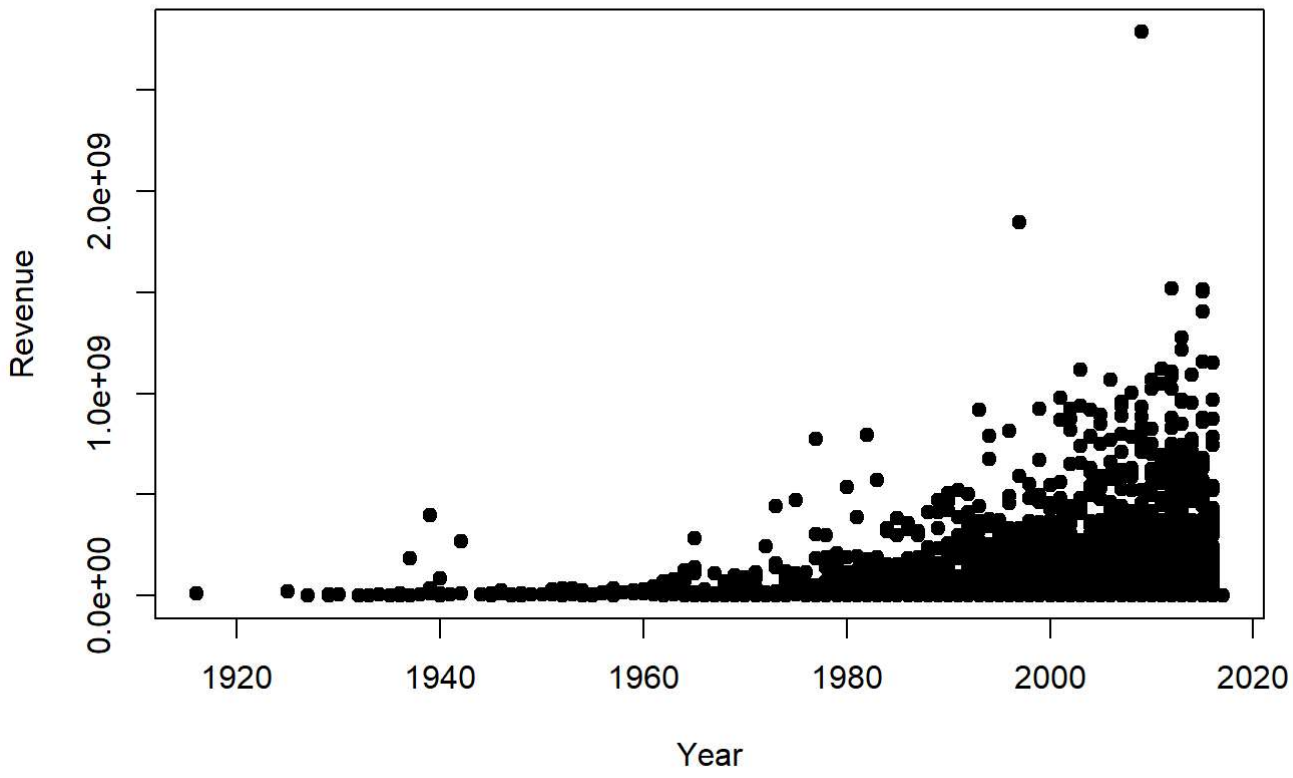

```
#Gets the oldest and newest movie in the dataset
movies[movies$year== min(movies$year, na.rm = TRUE),][,15]
```

```
## [1] NA  2
```

```
movies[movies$year== max(movies$year, na.rm = TRUE),][,15]
```

```
## [1]  2 NA
```

```
#From the histogram and the lines of code above, we see that the dataset includes movies from 19
16 to 2017.


plot(movies$year, movies$revenue, pch=19, main = "Plot of Movie Revenue and Release Year", xlab
= "Year", ylab = "Revenue")
```




```
#We also see from the plot above that the most profitable movie according to the dataset was Ava
tar, as evidenced by the following query.
movies[movies$revenue== max(movies$revenue, na.rm = TRUE),][,c(6,9)]
```

```
##      revenue  title
## 1 2787965087 Avatar
```

```
#But from the internet, we know that Gone with the Wind was the most profitable movie of all tim
e, when adjusted for inflation, and it is in our dataset.
movies[movies$title== "Gone with the Wind",][,c(6,9)]
```

```
##        revenue              title
## 3814 400176459 Gone with the Wind
```

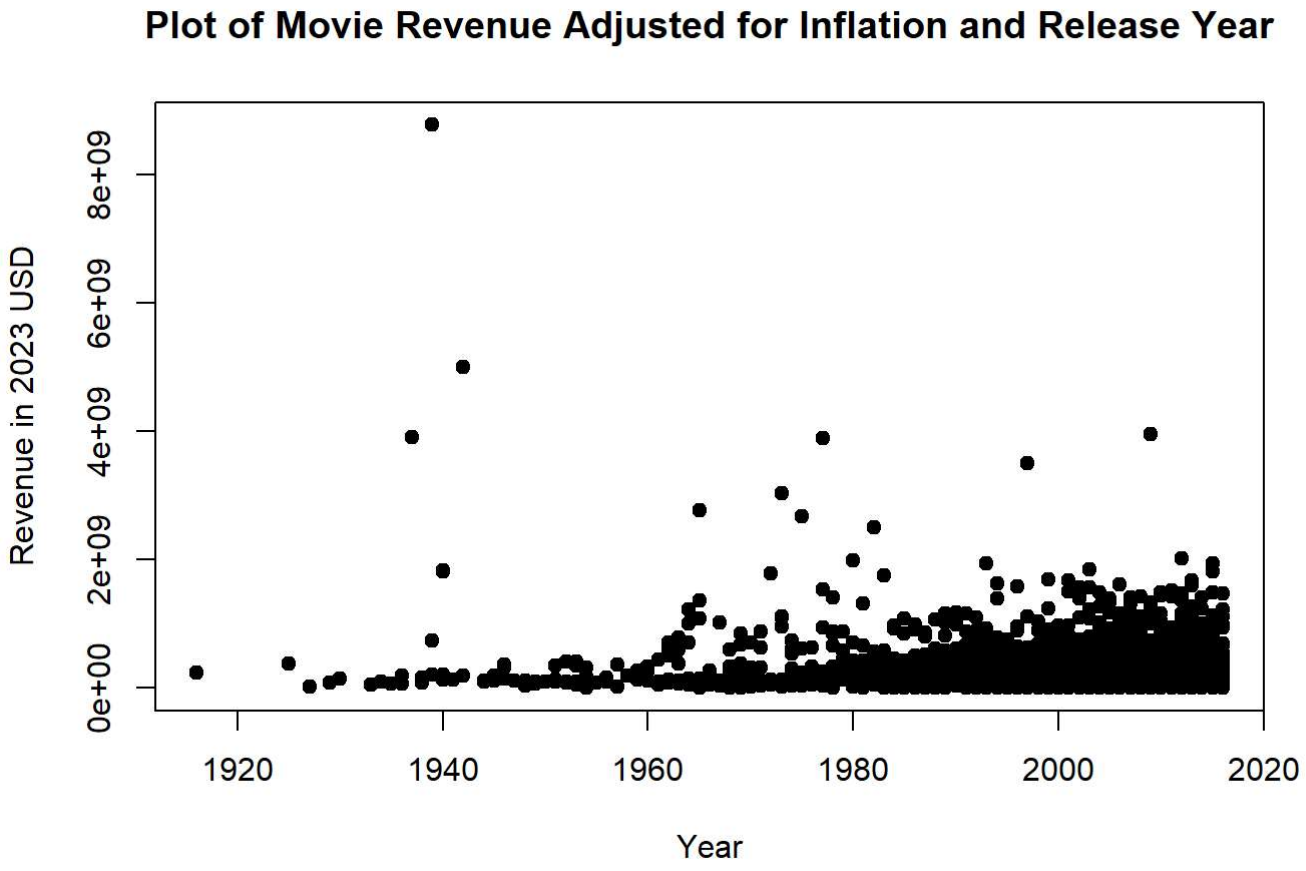
```
#So we now know that the dataset does not adjust movie revenue for inflation, and because of thi
s, we will adjust movie revenue by inflation ourselves by converting the revenue gained to how m
uch it would be in 2023.



#CPI dataset copy and pasted from:   https://www.usinflationcalculator.com/inflation/consumer-pr
ice-index-and-annual-percent-changes-from-1913-to-2008/

CPI = read.table("C:/Users/ryne0/Downloads/CPI 1913 to 2023.txt", sep = "\t", header = TRUE)


#Showing what the dataset looks like. Note that it goes from 1913 to 2023, inclusive.
CPI[1,]
```

```
##   Year Jan Feb Mar Apr May June July Aug Sep Oct  Nov Dec Avg Dec.Dec Avg.Avg
## 1 1913 9.8 9.8 9.8 9.8 9.7  9.8  9.9 9.9  10  10 10.1  10 9.9       -       -
```

```r
#We will be using the Avg column to scale our Revenue column to its worth in 2023.



adjust_for_inflation = function(year_and_revenue_dataset, money_column_name, year_column_name, C
PI_dataset, new_col_name){
  #Shortening the name to make the code more readable
  yard = year_and_revenue_dataset


  #Have to remove NA values in year and revenue columns before running, otherwise ane error occu
rs
  yard = yard[!is.na(yard[[money_column_name]]) & !is.na(yard[[year_column_name]]), ]


  #Gets the CPI for 2023
  CPI_2023 = CPI_dataset[CPI_dataset$Year==2023, 14]






  adjusted_revenue_2023 = mapply(function(money, year)
    {
      CPI_year = CPI_dataset[CPI_dataset$Year == year, 14]
      money * (CPI_2023 / CPI_year)
    },
    yard[[money_column_name]], yard[[year_column_name]])


  #Adding the adjusted revenue to the dataset
  yard[[new_col_name]] = unlist(adjusted_revenue_2023)

  return(yard)
}




table(movies$revenue)[1:50]
```

```
##
##     0     5     7    10    11    12    13    14    15    16    23    25    46
##  1425     1     2     1     2     2     1     1     1     1     1     2     1
##    51    92   103   126   203   792  1632  3330  6399  7202  9069 10000 10018
##     1     1     1     1     1     1     1     1     1     1     1     1     1
## 10508 14870 14873 17472 17479 20380 22000 23000 25000 30471 30905 31081 32000
##     1     1     1     1     1     1     1     1     1     1     1     1     1
## 32251 33245 40542 42145 44462 45000 45779 46488 46495 53086 54207
##     1     1     1     1     1     1     1     1     1     1     1
```

```r
#Before we adjust for inflation, we see that there a lot of movies with their revenues as 0, pre
sumably from being not recorded, and observe that some movies possess suspiciously low budgets a
s well.
#To rectify this, we will remove movies with revenue lower than $100, since they are likely data
entry mistakes.


#$1000 was picked to account for older movies appearing to earn less because of inflation.


#Removing movies with earnings less than $1000
movies = movies[movies$revenue>=1000,]


#We now adjust our revenue column for inflation
movies = adjust_for_inflation(movies, "revenue", "year", CPI, "adjusted_revenue")

#Remove revenue since it is now scaled for inflation
movies = subset(movies, select = -revenue)


#Showing the budgets after we have adjusted it for inflation
plot(movies$year, movies$adjusted_revenue, pch=19, main = "Plot of Movie Revenue Adjusted for In
flation and Release Year", xlab = "Year", ylab = "Revenue in 2023 USD")
```



More Cleaning

```
#Removing movies with budgets less than $1000, for the same reason we removed movies with revenu
es of less than $1000
movies = movies[movies$budget>=1000,]
movies = adjust_for_inflation(movies, "budget", "year", CPI, "adjusted_budget")



#Remove budget since we have adjusted the inflation for it
movies = subset(movies, select = -budget)


movies[movies$status!="Released",]
```

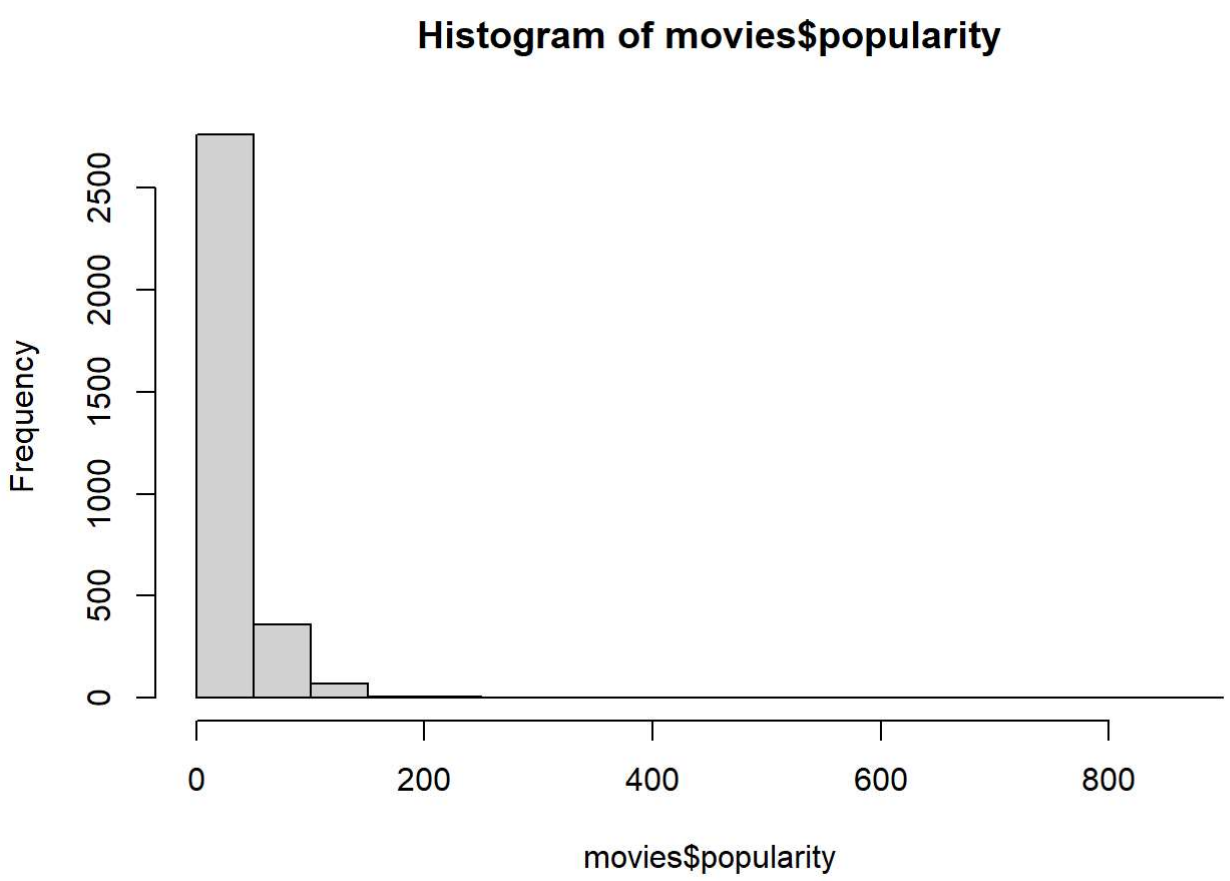
```
##         id popularity production_companies release_date runtime          status
## 4179 50875   1.699101                   []   2011-08-26     109 Post Production
##              title vote_average vote_count same_title production_countries_new
## 4179 Higher Ground          5.3         14          1                        1
##      is_english_movie genre_type year adjusted_revenue adjusted_budget
## 4179                1          2 2011          1140210         2709197
```

```
#Only one movie whose status is not "Released". Remove that movie and remove the 'status' column as it won't give us information for our model.


movies = movies[movies$status=="Released",]
movies = subset(movies, select = -status)




#Remove the id column since it is not relevant to us
movies = subset(movies, select = -id)



#Convert production_companies column into a boolean. The new column will be 1 if the movie was made by 1 or more major film studio.

#Rationale: It is thought that because the major film studios are reputable, experienced in film making, and have extensive distribution networks, that a movie produced by such a studio would perform better in sales because of better and increased marketing and theatre showtimes.



#These are the names of the major studios in the dataset. Major film studios were identified from Wikipedia: https://en.wikipedia.org/wiki/Major_film_studios#
#Will be including all present and past major studios

major_film_studio_names = c("Twentieth Century Fox Film Corporation", "Walt Disney Pictures", "Warner Bros.", "Sony Pictures Entertainment", "Paramount Pictures", "United Artists", "Metro-Goldwyn-Mayer", "RKO", "Universal Pictures", "New Line Cinema", "20th Century Studios", "Columbia Pictures", "TriStar Pictures")




is_made_by_major_studio = function(dataset, major_studios, studio_column){

  # Apply the function to check for each row
  is_made = sapply(dataset[[studio_column]], function(studio_data) {


    #These lines before the ifelse statement are necessary because the production_companies is saved in a json format which is incompatible with default string detection in R.
    #ChatGPT begins

    # Parse the studio data, assuming it's in a JSON-like format
    studios <- tryCatch(fromJSON(studio_data), error = function(e) NULL)

    # Extract studio names if parsing was successful, otherwise fallback to raw data
    if (!is.null(studios) && "name" %in% names(studios)) {
      studio_names <- studios$name  # Extract the studio names
```

```
    }
    else
        #Use raw data if parsing fails
        studio_names <- as.character(studio_data)

    #ChatGPT ends


    #paste turns the major_studios vector into a single string separated by |
    ifelse(any(grepl(paste(major_studios, collapse = "|"), studio_names)), 1, 0)

  })

  # Add a new column to the dataset
  dataset$made_by_major_studio = is_made

  return(dataset)
}


#Have to coerce production_companies to be a string, otherwise an error would occur when calling
it into is_made_by_major_studio
movies$production_companies = as.character(movies$production_companies)



movies = is_made_by_major_studio(movies, major_film_studio_names, "production_companies")
table(movies$same_title)
```

```
##
##    0    1
##  105 3105
```

```
#Drop the production companies column now that transformed it into a predictor variable.
movies = subset(movies, select = -production_companies)



hist(movies$popularity)
```




```
movies[movies$popularity==max(movies$popularity),]
```

```
##     popularity release_date runtime   title vote_average vote_count same_title
## 547   875.5813   2015-06-17      91 Minions          6.4       4571          1
##     production_countries_new is_english_movie genre_type year adjusted_revenue
## 547                        1                1          1 2015       1487058893
##     adjusted_budget made_by_major_studio
## 547        95132197                    1
```

```
movies[movies$popularity==max(movies$popularity),]
```

```
##     popularity release_date runtime   title vote_average vote_count same_title
## 547   875.5813   2015-06-17      91 Minions          6.4       4571          1
##     production_countries_new is_english_movie genre_type year adjusted_revenue
## 547                        1                1          1 2015       1487058893
##     adjusted_budget made_by_major_studio
## 547        95132197                    1
```

```
movies[order(-movies$popularity), ][c(1:20),c(1,4,11)]
```

```
##      popularity                                                  title year
## 547    875.5813                                                 Minions 2015
## 96     724.2478                                            Interstellar 2014
## 789    514.5700                                                Deadpool 2016
## 95     481.0986                                 Guardians of the Galaxy 2014
## 128    434.2786                                      Mad Max: Fury Road 2015
## 29     418.7086                                          Jurassic World 2015
## 200    271.9729 Pirates of the Caribbean: The Curse of the Black Pearl 2003
## 83     243.7917                          Dawn of the Planet of the Apes 2014
## 201    206.2272                The Hunger Games: Mockingjay - Part 1 2014
## 89     203.7346                                            Big Hero 6 2014
## 109    202.0426                                      Terminator Genisys 2015
## 27     198.3724                           Captain America: Civil War 2016
## 3866   192.5288                                               Whiplash 2014
## 66     187.3229                                   The Dark Knight 2008
## 271    167.9329                                       The Martian 2015
## 97     167.5837                                         Inception 2010
## 125    165.1254                                            Frozen 2013
## 10     155.7905                  Batman v Superman: Dawn of Justice 2016
## 1      150.4376                                            Avatar 2009
## 663    146.7574                                          Fight Club 1999
```

```
#We observe that modern movies tend to have a higher popularity than older ones. This is because
the method TMDB uses to calculates popularity favours giving modern movies a larger score, becau
se of recently available metrics such as social media activity and number of visits to the TMDB
website for that movie.
#Because of this confounding variable, we will not be using popularity.


#And because vote_count skews the total number of votes to favour newer movies, we will not be u
sing vote_count either


movies = subset(movies, select=-c(popularity, vote_count))
#The data cleaning is now finished!
```

Building Functions to Check Assumptions

```
outlier_removal = function(dataset, outlier_column){

  #oc means outlier column
  oc = dataset[[outlier_column]]


  #Getting the first and third quartiles for our outlier column
  oc_Q1 = quantile(oc, 0.25)
  oc_Q3 = quantile(oc, 0.75)


  #Gets the interquartile range
  oc_IQR = oc_Q3 - oc_Q1


  #Determine the outlier bounds
  oc_lower_bound = oc_Q1 - 1.5 * oc_IQR
  oc_upper_bound = oc_Q3 + 1.5 * oc_IQR


  #Removes the outliers from our dataset
  dataset = dataset[oc >= oc_lower_bound & oc <= oc_upper_bound,]


  return(dataset)
}


#remove_outliers is a boolean
#trans_val is short for transformation value
check_regression_assumptions = function(dataset_regression, response_var, remove_outliers, trans
_val){

  #Removes outliers when set to true. Removes outliers from both the predictor variable(s) and t
he response variable
  if (remove_outliers){
    for(i in 1:ncol(dataset_regression)){
      dataset_regression = outlier_removal(dataset_regression, names(dataset_regression)[i])
    }
  }

  response = dataset_regression[[response_var]]



  #Runs true if there is only one predictor
  if (ncol(dataset_regression) == 2) {
    dataset_no_response = dataset_regression[ , !(names(dataset_regression) %in% response_var)]


    if (is.vector(dataset_no_response)) {
      predictor_name = names(dataset_regression)[names(dataset_regression) != response_var]
```

```
      #In this line of code, dataset_no_response is a vector because there was only one predicto
r
      dataset_no_response = setNames(data.frame(dataset_no_response), predictor_name)
    }
  }
  else {
    dataset_no_response = dataset_regression[ , !(names(dataset_regression) %in% response_var)]
  }


  for(i in 1:ncol(dataset_no_response)){

    #Gets the name of the predictor variable
    x_axis_name = names(dataset_no_response)[i]



    #Runs if we want to remove outliers. Changes the names of the plots
    if (remove_outliers){
      scatterplot_name = paste0("Plot of ",response_var," vs ", x_axis_name, " Without Outlier
s")
      residual_plot_name = paste0("Residuals vs Fitted Values for ",response_var," vs ", x_axis_
name, " Without Outliers")

      histogram_name = paste0("Histogram of the Residuals for ",response_var," vs ", x_axis_nam
e, " Without Outliers")
      qqnorm_name = paste0("Normal Q-Q plot of the Residuals for ",response_var," vs ", x_axis_n
ame, " Without Outliers")
    }


    else{
      scatterplot_name = paste0("Plot of ",response_var," vs ", x_axis_name, " With Outliers")
      residual_plot_name = paste0("Residuals vs Fitted Values for ",response_var," vs ", x_axis_
name, " With Outliers")

      histogram_name = paste0("Histogram of the Residuals for ",response_var," vs ", x_axis_nam
e, " With Outliers")
      qqnorm_name = paste0("Normal Q-Q plot of the Residuals for ",response_var," vs ", x_axis_n
ame, " With Outliers")

    }

    predictor = dataset_no_response[,i]


    #Making our simple linear model
    if (is.na(trans_val))
      linear_model = lm(response~predictor)

    else
```

```
      linear_model = lm(response^trans_val~predictor)



    #Scatterplot of the response vs the predictor variable with the least squares line
    plot(predictor, response, col = 10*i, main = scatterplot_name, xlab = x_axis_name, ylab = re
sponse_var)
    abline(linear_model, col="black")


    #Residuals Plot
    plot(linear_model$fitted.values, linear_model$residuals, xlab="Fitted values", ylab = "Resid
uals", main = residual_plot_name)


    #Histogram plot
    hist(linear_model$residuals, main = histogram_name, xlab = "Residuals")


    #qqnorm plot
    qqnorm(linear_model$residuals, main = qqnorm_name, ylab="Residuals")
    qqline(linear_model$residuals)
  }
}
```

We will only predict for continuous predictors first. Models involving categorical predictors will be included later.

Checking Assumptions

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

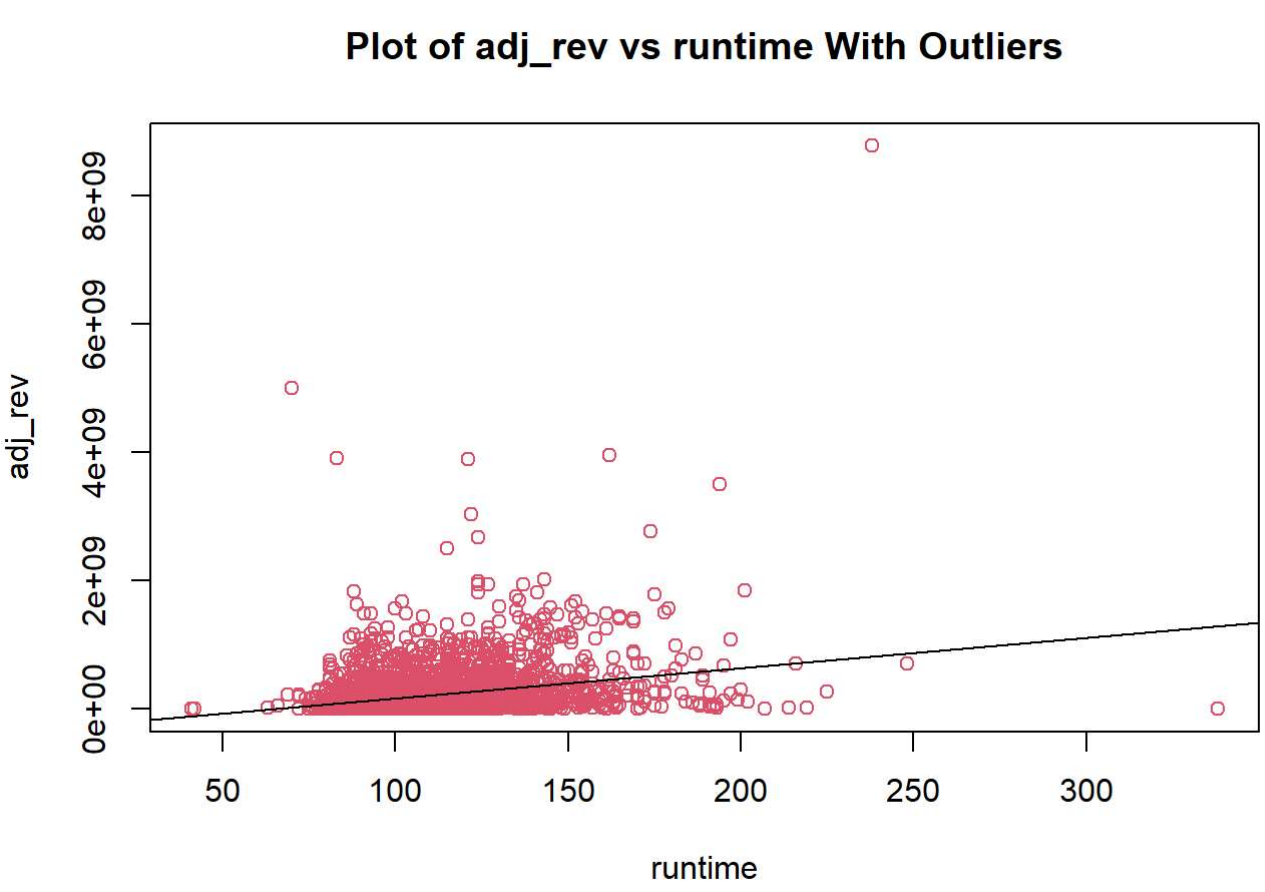
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

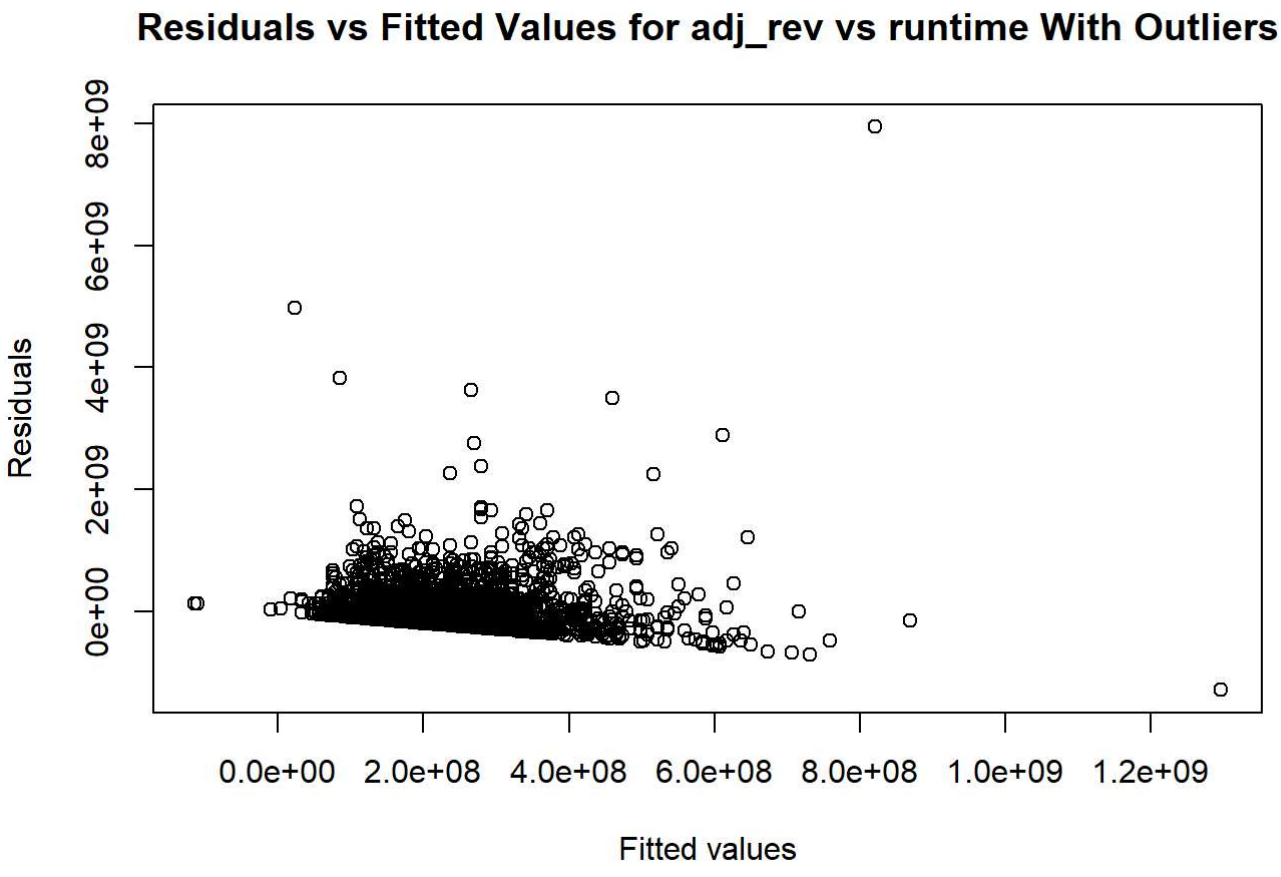
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

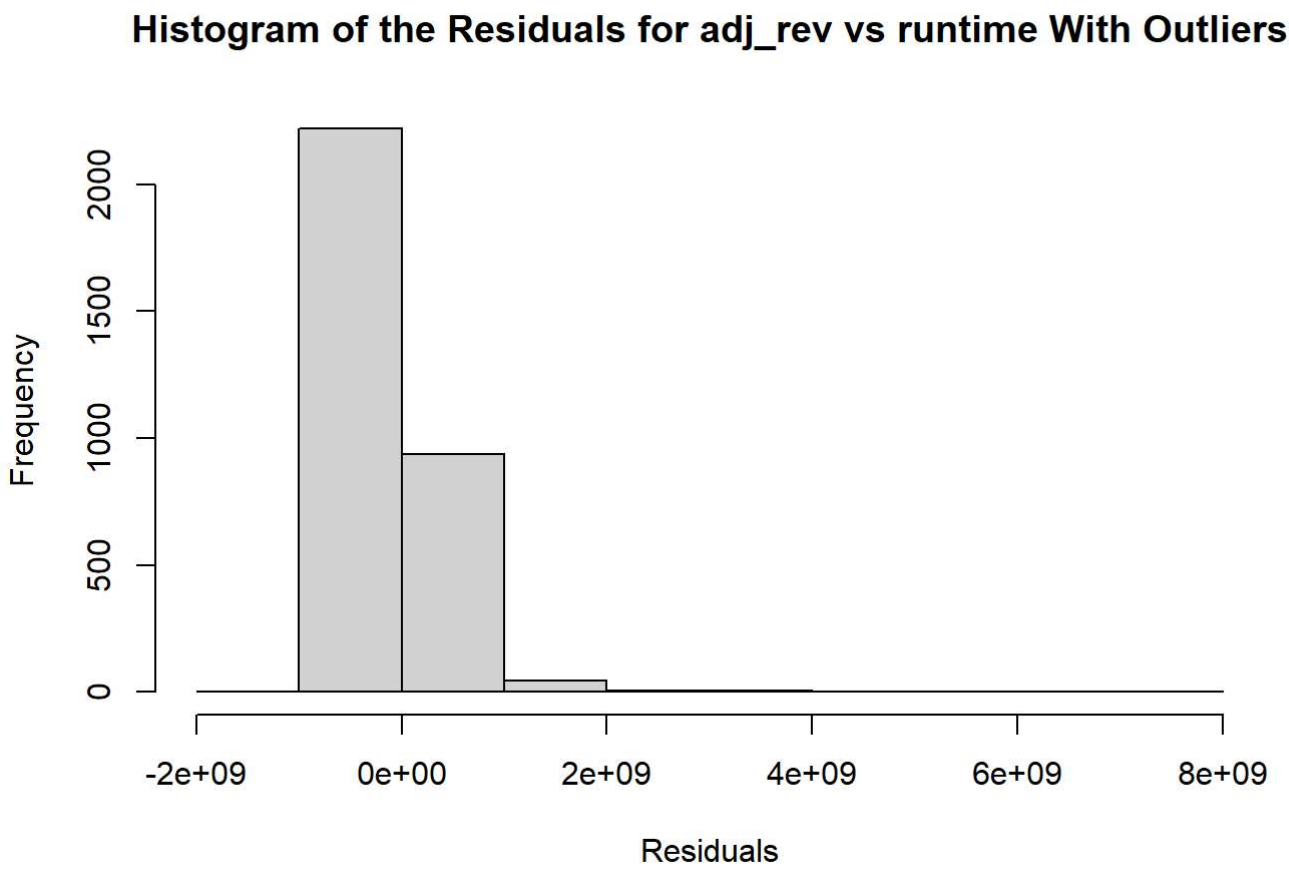
```
colnames(movies)
```

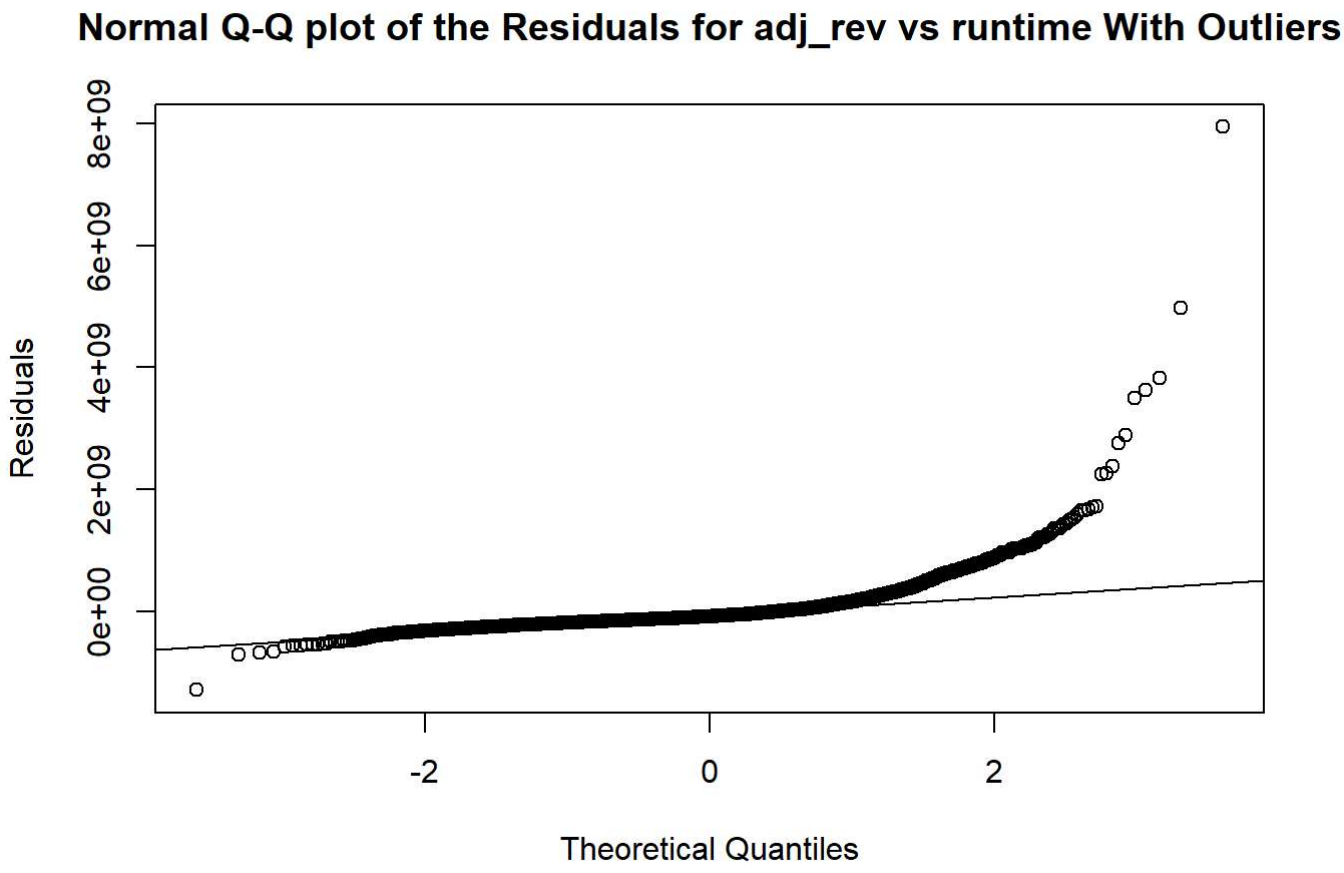
```
##  [1] "release_date"             "runtime"
##  [3] "title"                    "vote_average"
##  [5] "same_title"               "production_countries_new"
##  [7] "is_english_movie"         "genre_type"
##  [9] "year"                     "adjusted_revenue"
## [11] "adjusted_budget"          "made_by_major_studio"
```

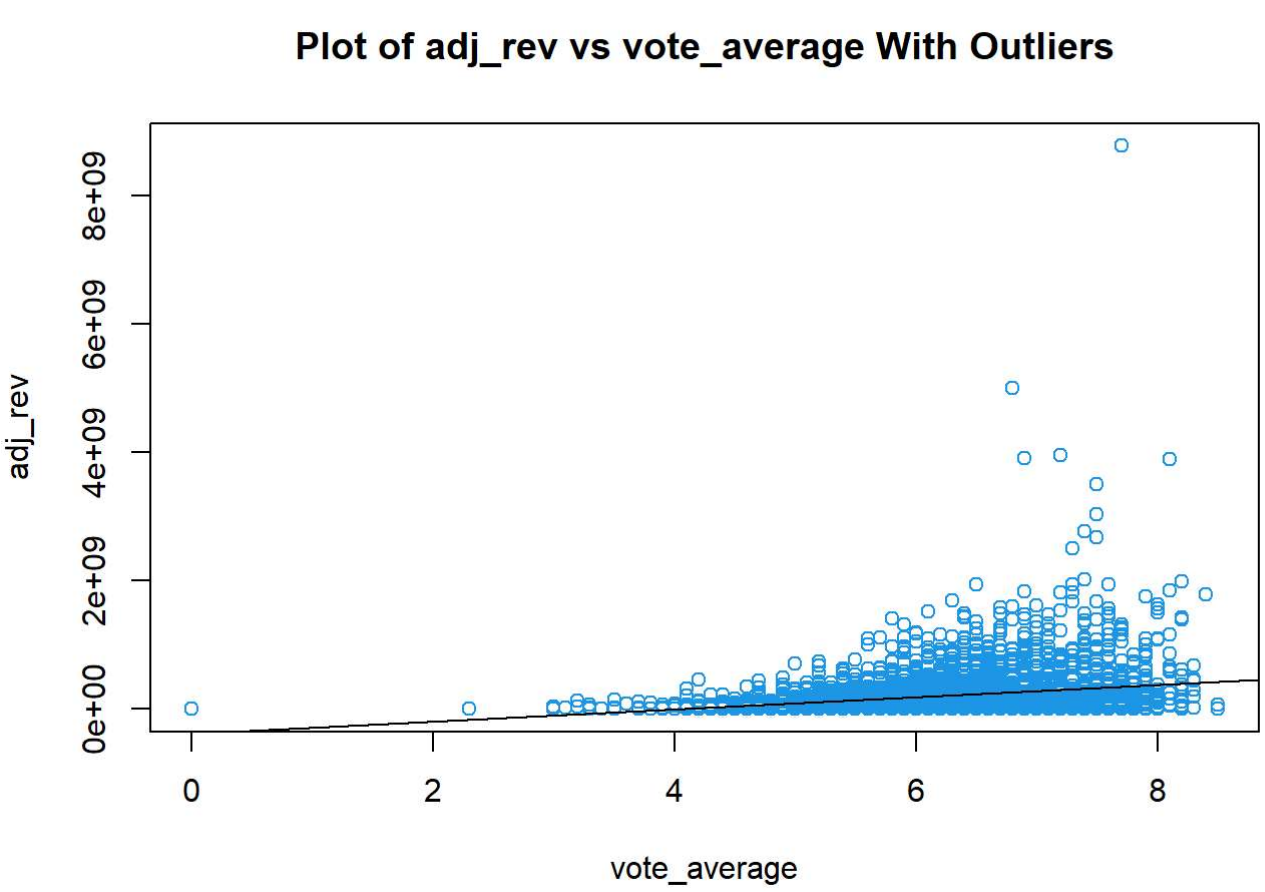
```
#Extract the numerical variables out of the dataset to predict for adjusted_revenue
movies_regression = movies[,c(2,4,10,11)]


#Rename these columns to make title of plots more readable
movies_regression = movies_regression %>%
  rename("adj_rev" = adjusted_revenue, "adj_bud" = adjusted_budget)



check_regression_assumptions(movies_regression, "adj_rev", F, NA)
```

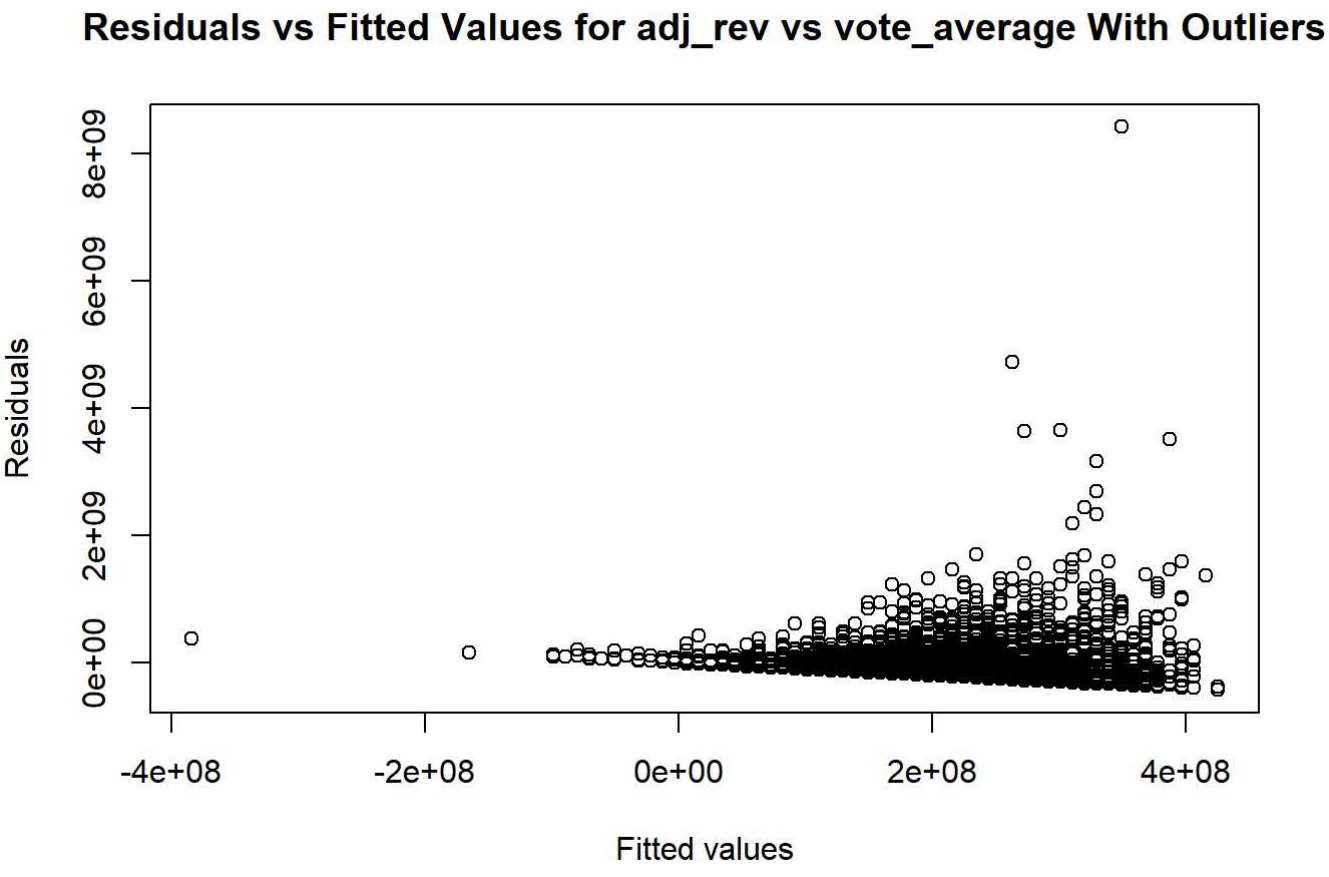

Plot of adj_rev vs runtime With Outliers
adj_rev
0e+00
2e+09
4e+09
6e+09
8e+09
50
100
150
200
250
300
runtime

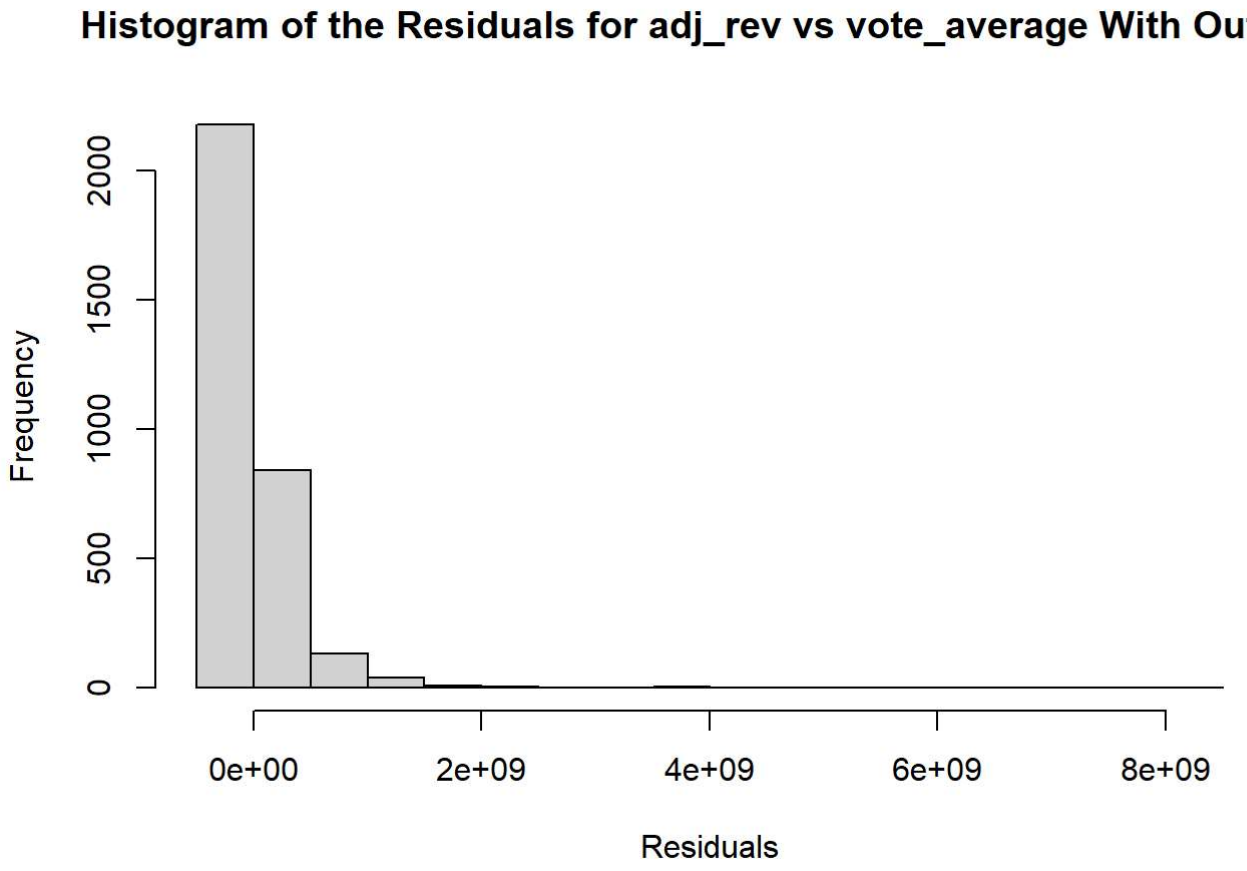

Residuals vs Fitted Values for adj_rev vs runtime With Outliers
Residuals
0e+00
2e+09
4e+09
6e+09
8e+09
0.0e+00
2.0e+08
4.0e+08
6.0e+08
8.0e+08
1.0e+09
1.2e+09
Fitted values

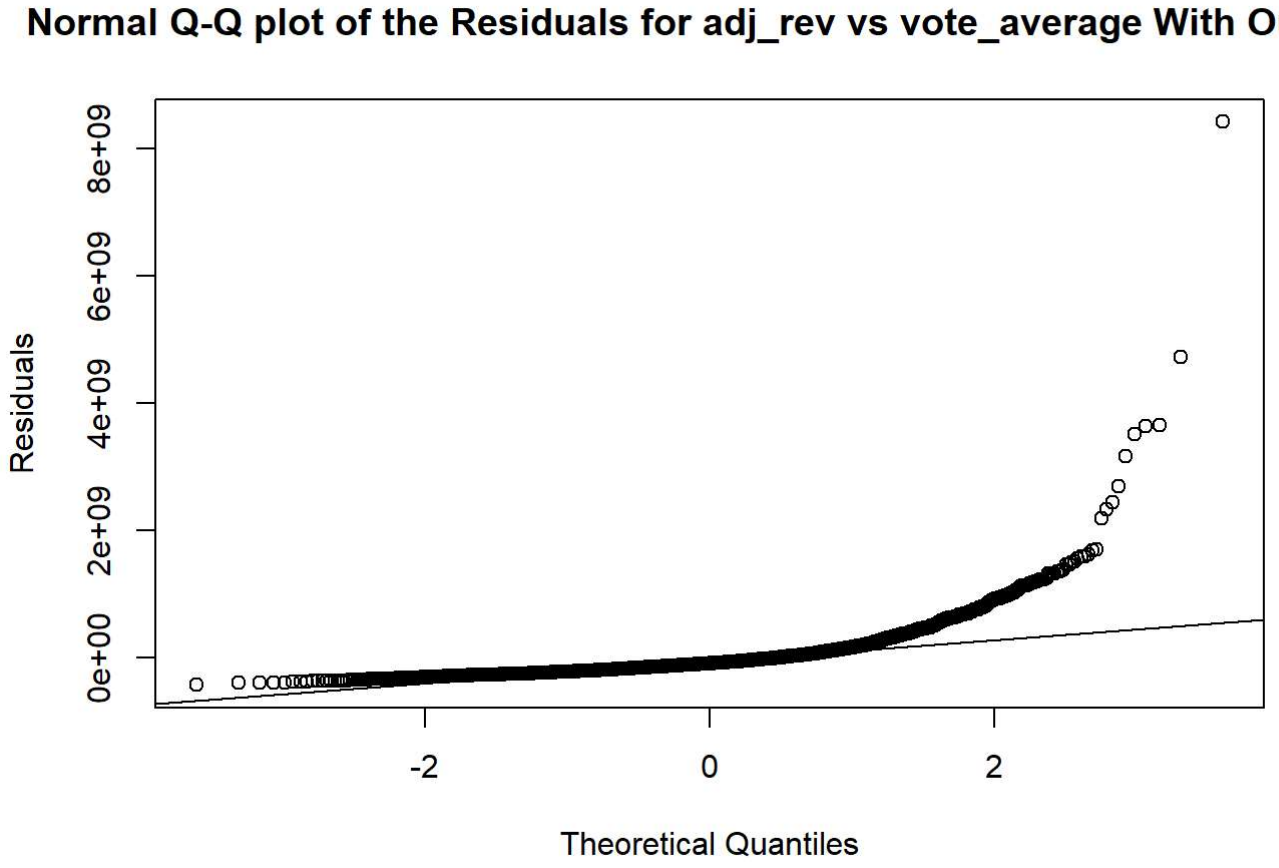

Histogram of the Residuals for adj_rev vs runtime With Outliers
Frequency
0
500
1000
1500
2000
-2e+09
0e+00
2e+09
4e+09
6e+09
8e+09
Residuals

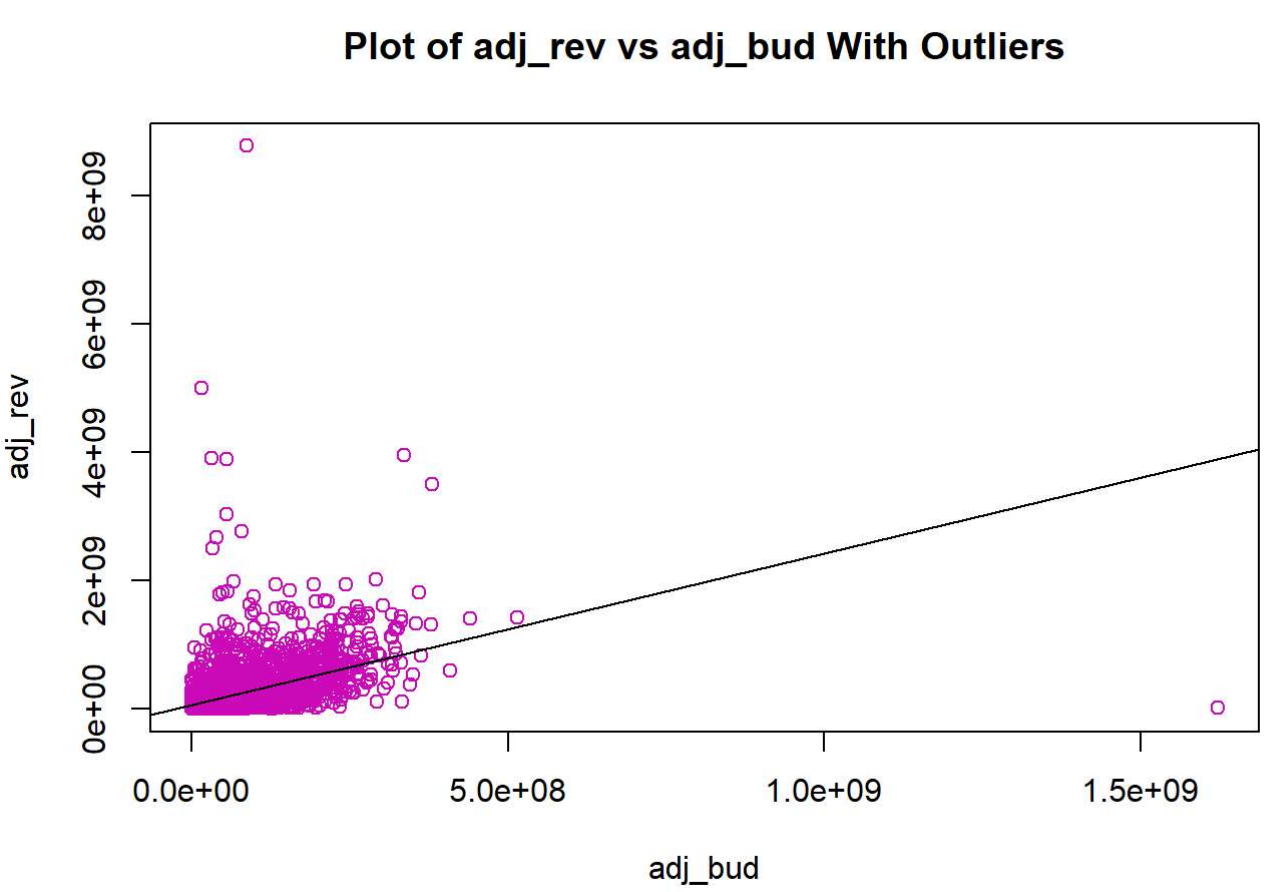

Normal Q-Q plot of the Residuals for adj_rev vs runtime With Outliers
Residuals
0e+00
2e+09
4e+09
6e+09
8e+09
-2
0
2
Theoretical Quantiles

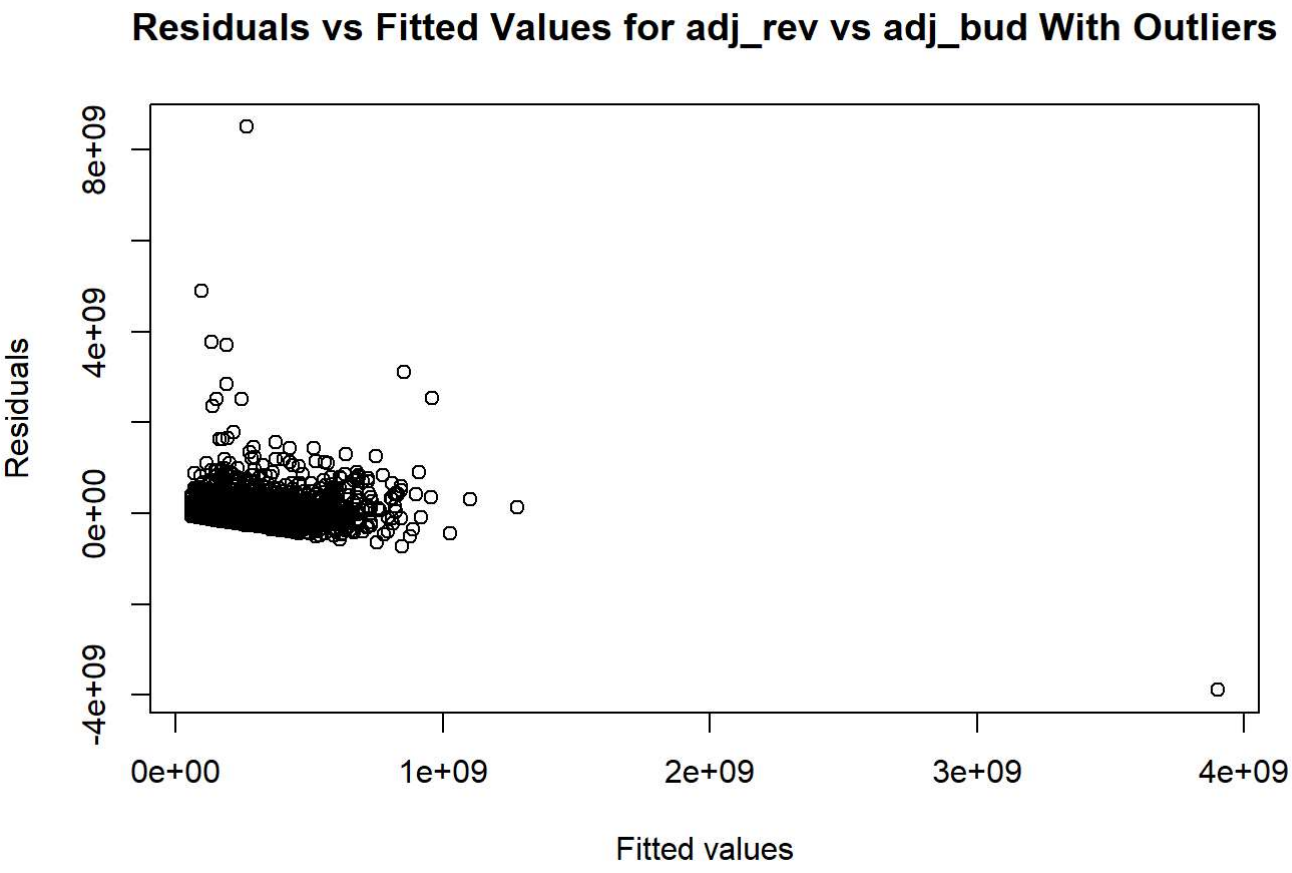

Plot of adj_rev vs vote_average With Outliers
adj_rev
0e+00
2e+09
4e+09
6e+09
8e+09
0
2
4
6
8
vote_average


Residuals vs Fitted Values for adj_rev vs vote_average With Outliers
Residuals
0e+00
2e+09
4e+09
6e+09
8e+09
-4e+08
-2e+08
0e+00
2e+08
4e+08
Fitted values

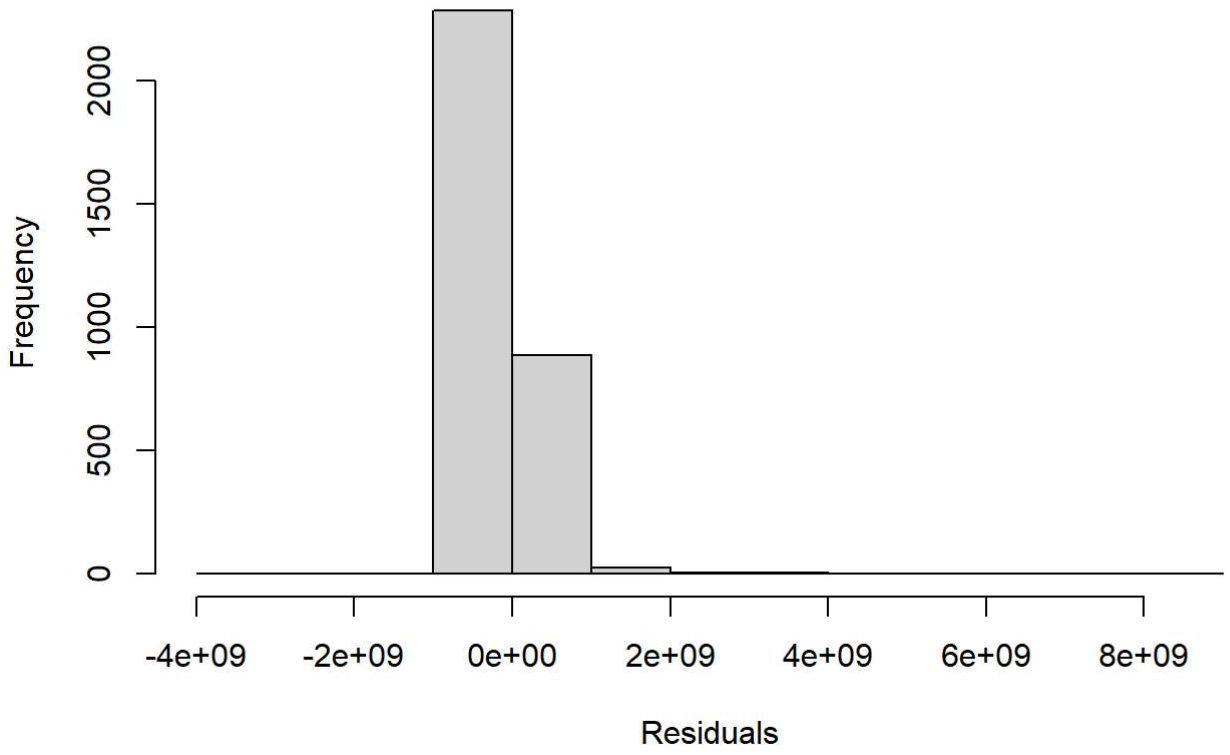

Histogram of the Residuals for adj_rev vs vote_average With Outliers
Frequency
0
500
1000
1500
2000
0e+00
2e+09
4e+09
6e+09
8e+09
Residuals


Normal Q-Q plot of the Residuals for adj_rev vs vote_average With Outlie
Residuals
0e+00
2e+09
4e+09
6e+09
8e+09
-2
0
2
Theoretical Quantiles

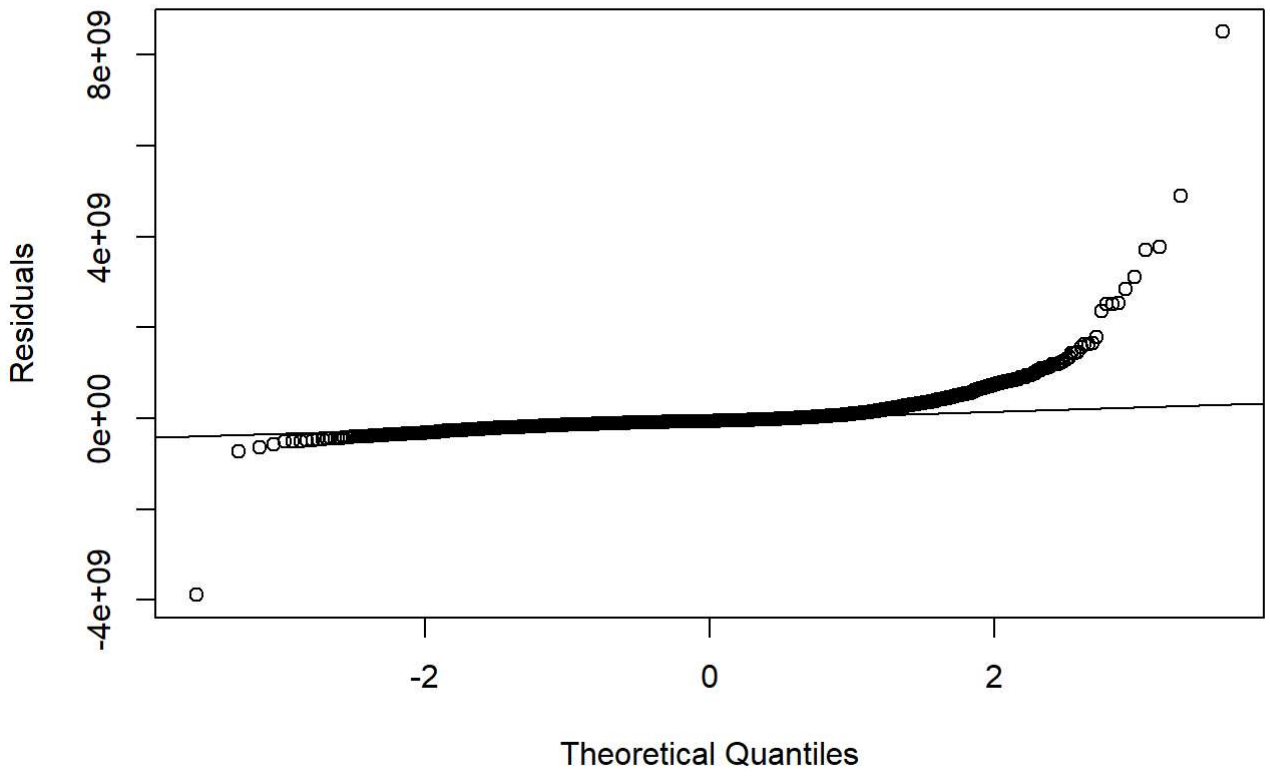

Plot of adj_rev vs adj_bud With Outliers
adj_rev
0e+00
2e+09
4e+09
6e+09
8e+09
0.0e+00
5.0e+08
1.0e+09
1.5e+09
adj_bud


Residuals vs Fitted Values for adj_rev vs adj_bud With Outliers
Residuals
-4e+09
0e+00
4e+09
8e+09
0e+00
1e+09
2e+09
3e+09
4e+09
Fitted values

Histogram of the Residuals for adj_rev vs adj_bud With Outliers

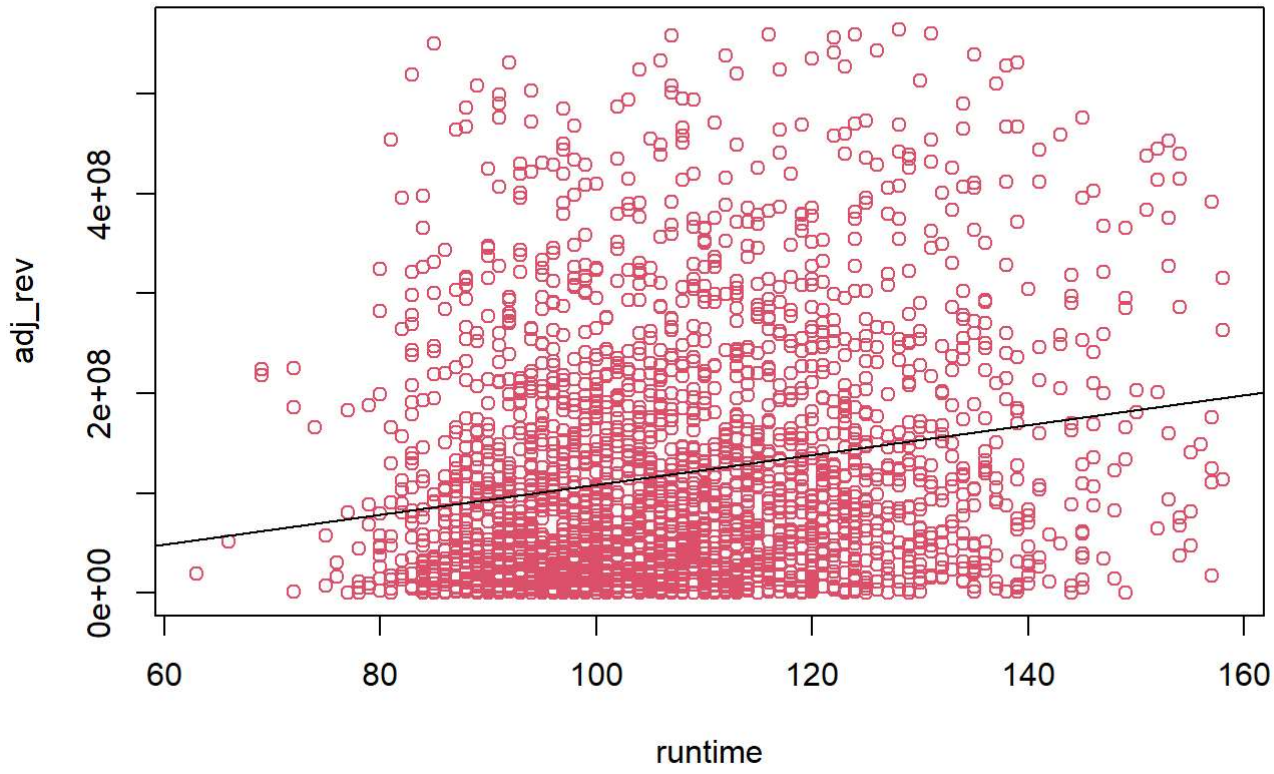


Normal Q-Q plot of the Residuals for adj_rev vs adj_bud With Outliers

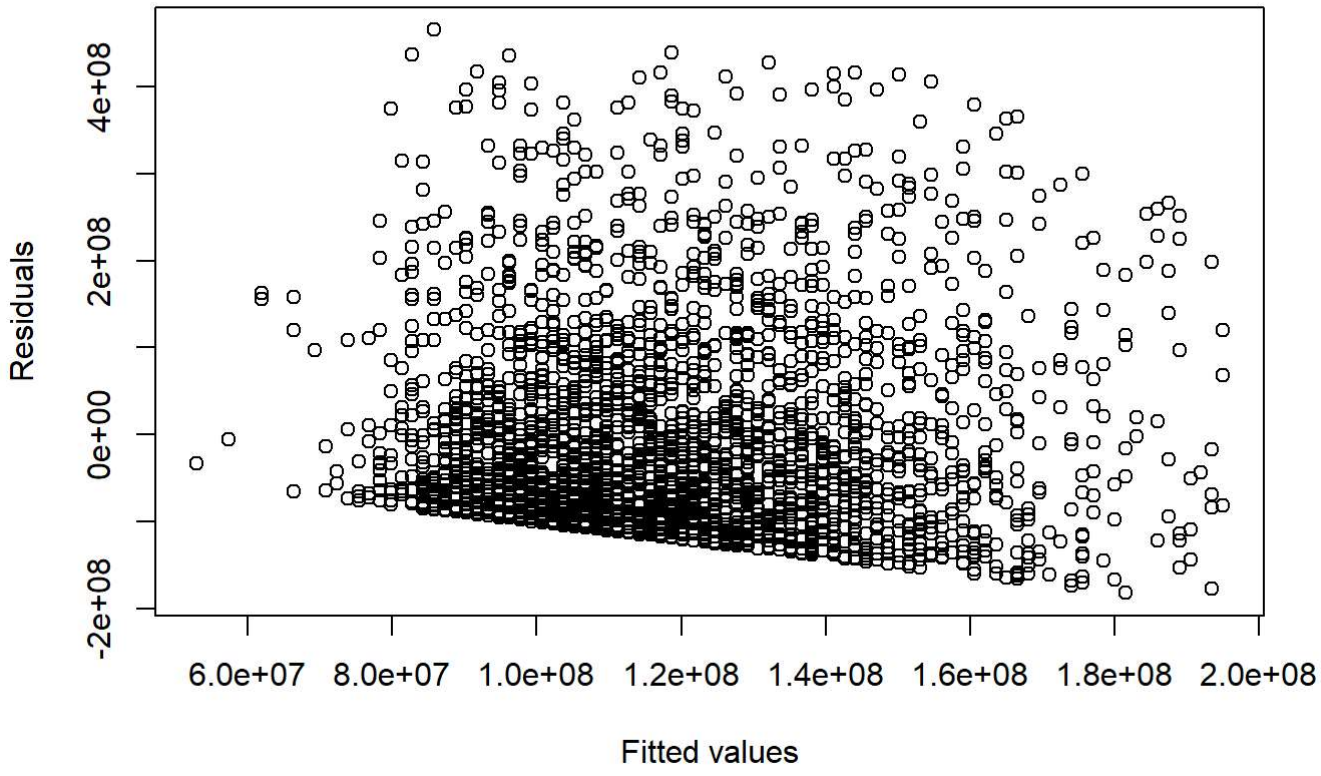


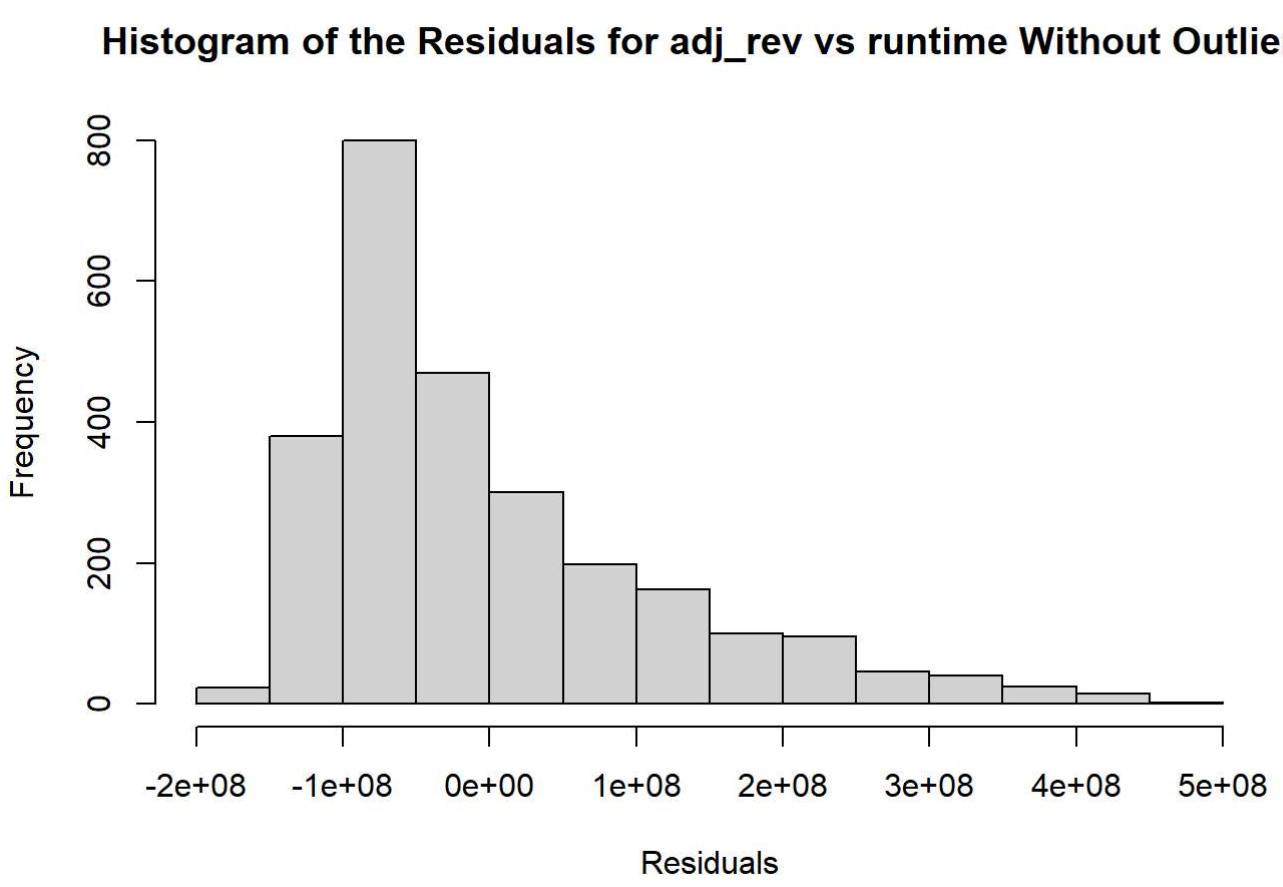
```
#We observe that most of the plots, which we use to check the assumptions of regression, are sig
nificantly distorted for when we don't remove outliers from our dataset. To better understand if
the simple linear models follow the assumptions of regression, we will remove the outliers from
out data and make the plots again


check_regression_assumptions(movies_regression, "adj_rev", T, NA)
```

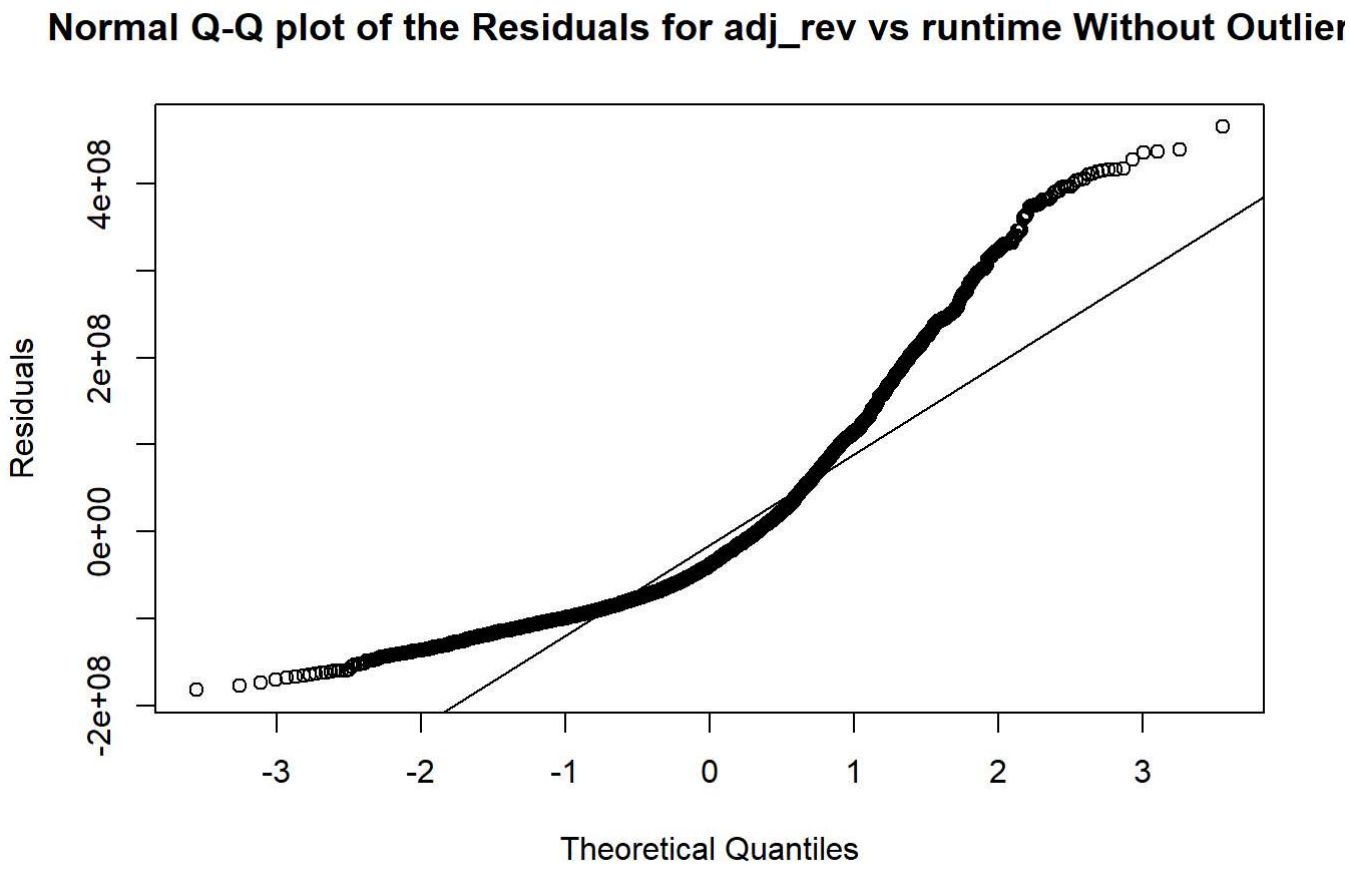
Plot of adj_rev vs runtime Without Outliers

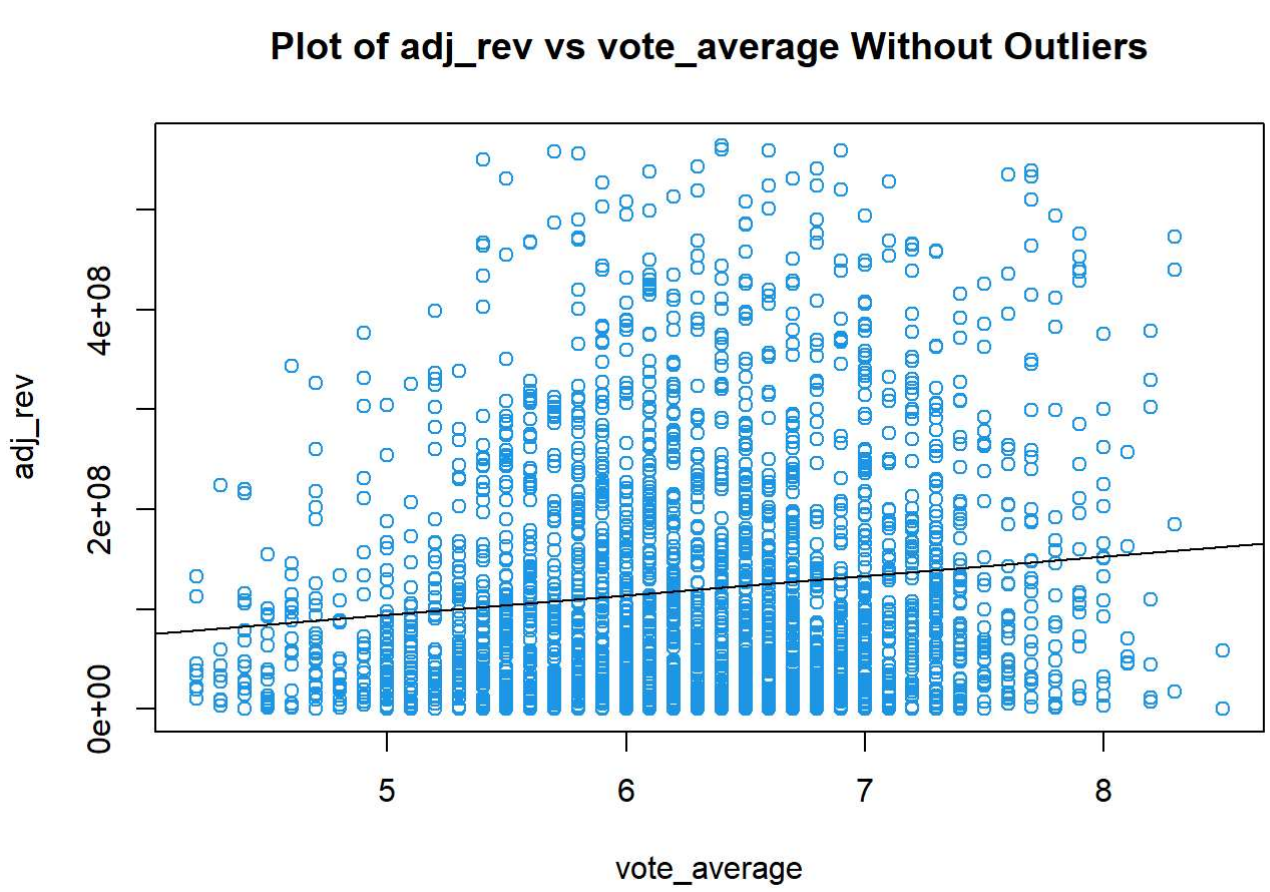


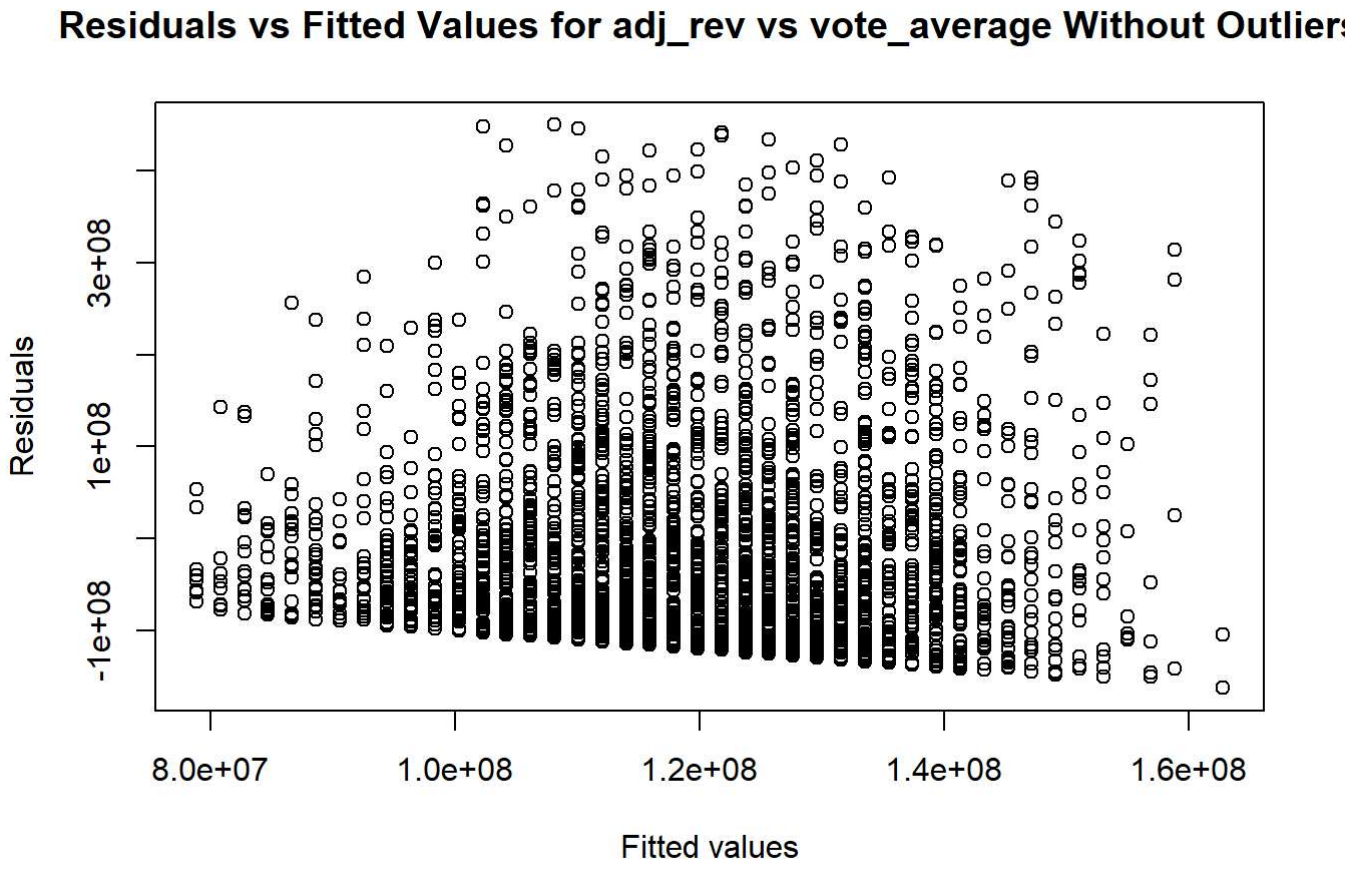
Residuals vs Fitted Values for adj_rev vs runtime Without Outliers



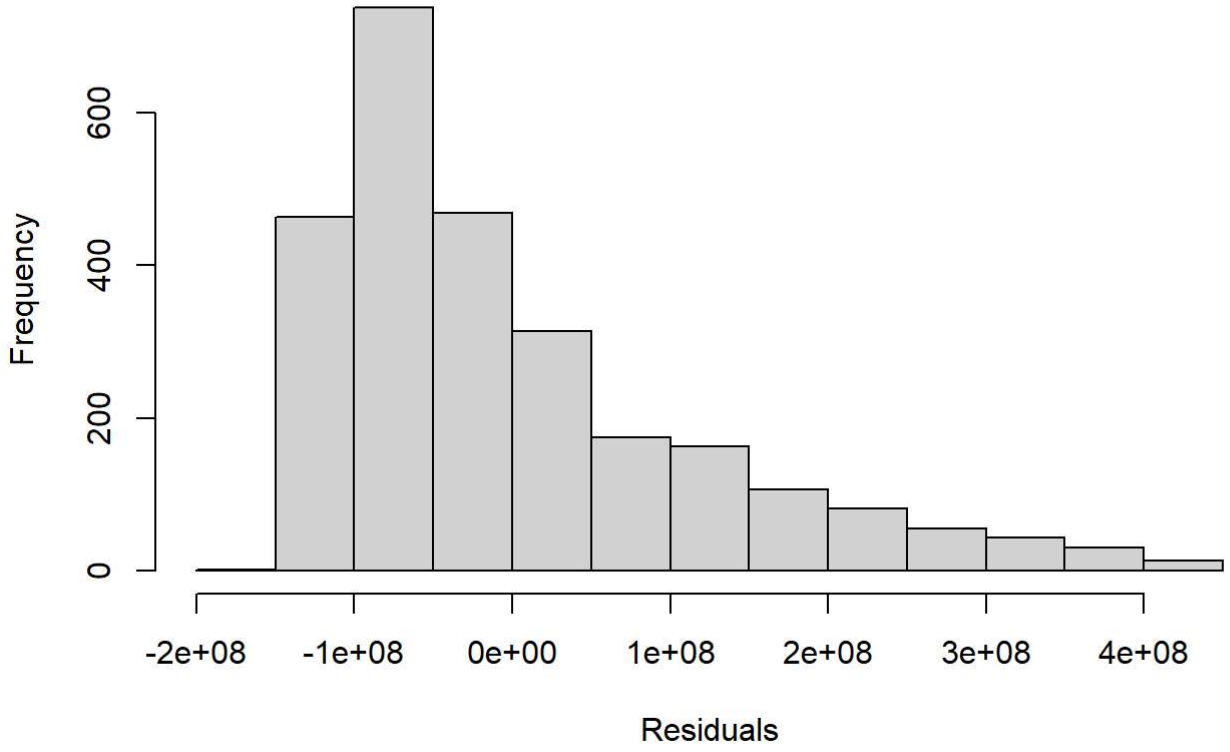

Histogram of the Residuals for adj_rev vs runtime Without Outliers
Frequency
0
200
400
600
800
-2e+08
-1e+08
0e+00
1e+08
2e+08
3e+08
4e+08
5e+08
Residuals


Normal Q-Q plot of the Residuals for adj_rev vs runtime Without Outlier
Residuals
-2e+08
0e+00
2e+08
4e+08
-3
-2
-1
0
1
2
3
Theoretical Quantiles

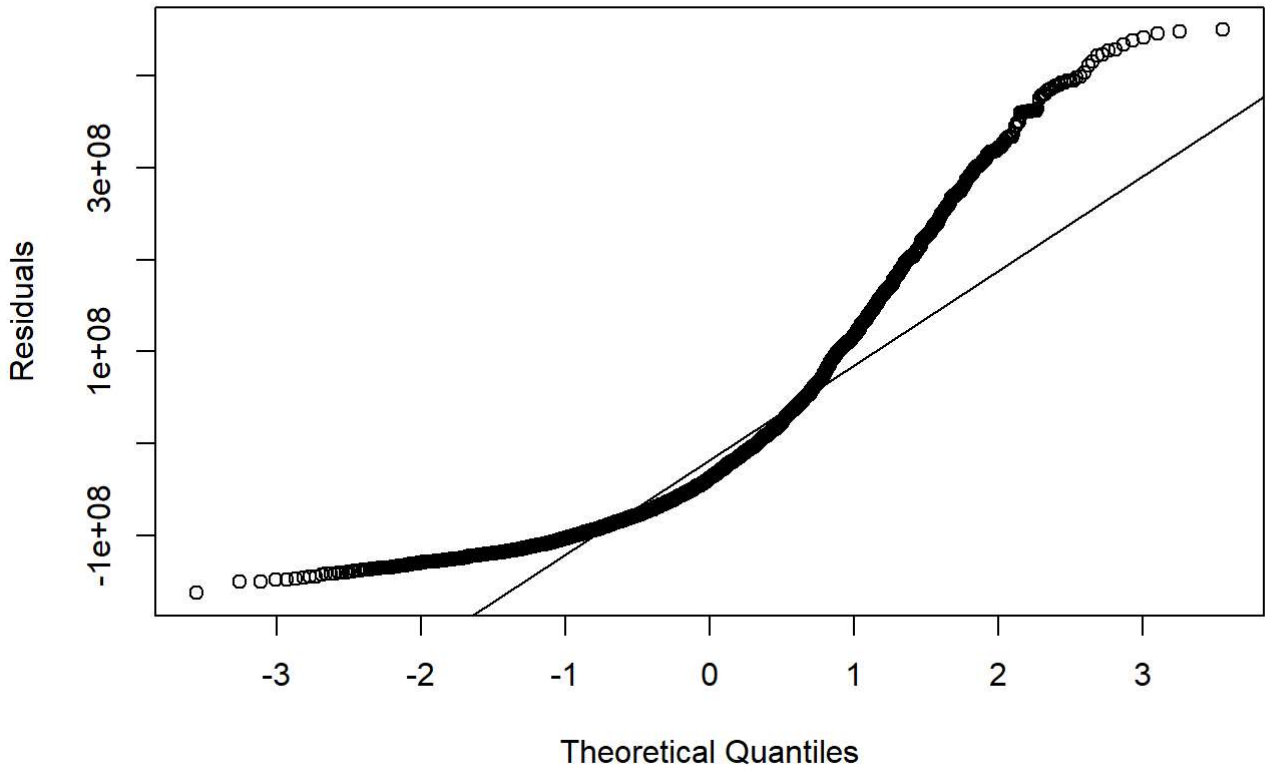

Plot of adj_rev vs vote_average Without Outliers
adj_rev
0e+00
2e+08
4e+08
5
6
7
8
vote_average


Residuals vs Fitted Values for adj_rev vs vote_average Without Outliers
Residuals
-1e+08
1e+08
3e+08
8.0e+07
1.0e+08
1.2e+08
1.4e+08
1.6e+08
Fitted values

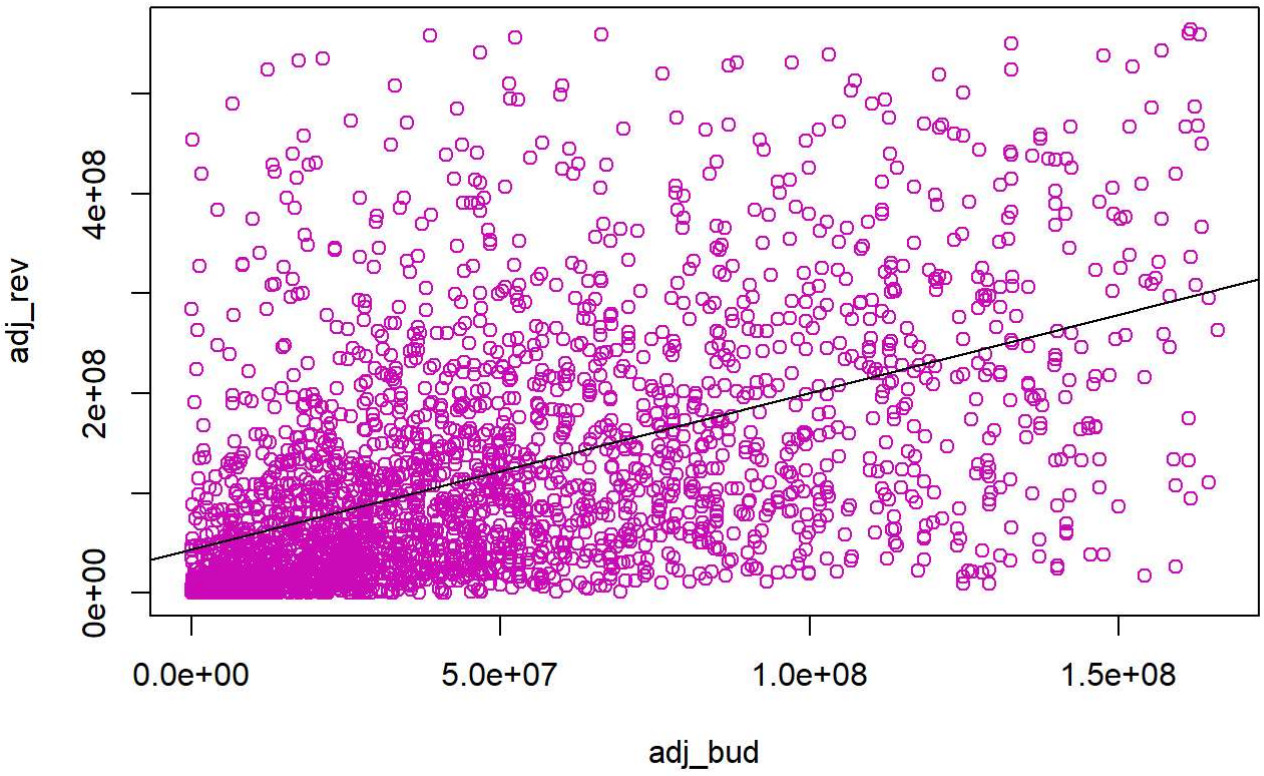
Histogram of the Residuals for adj_rev vs vote_average Without Outlier



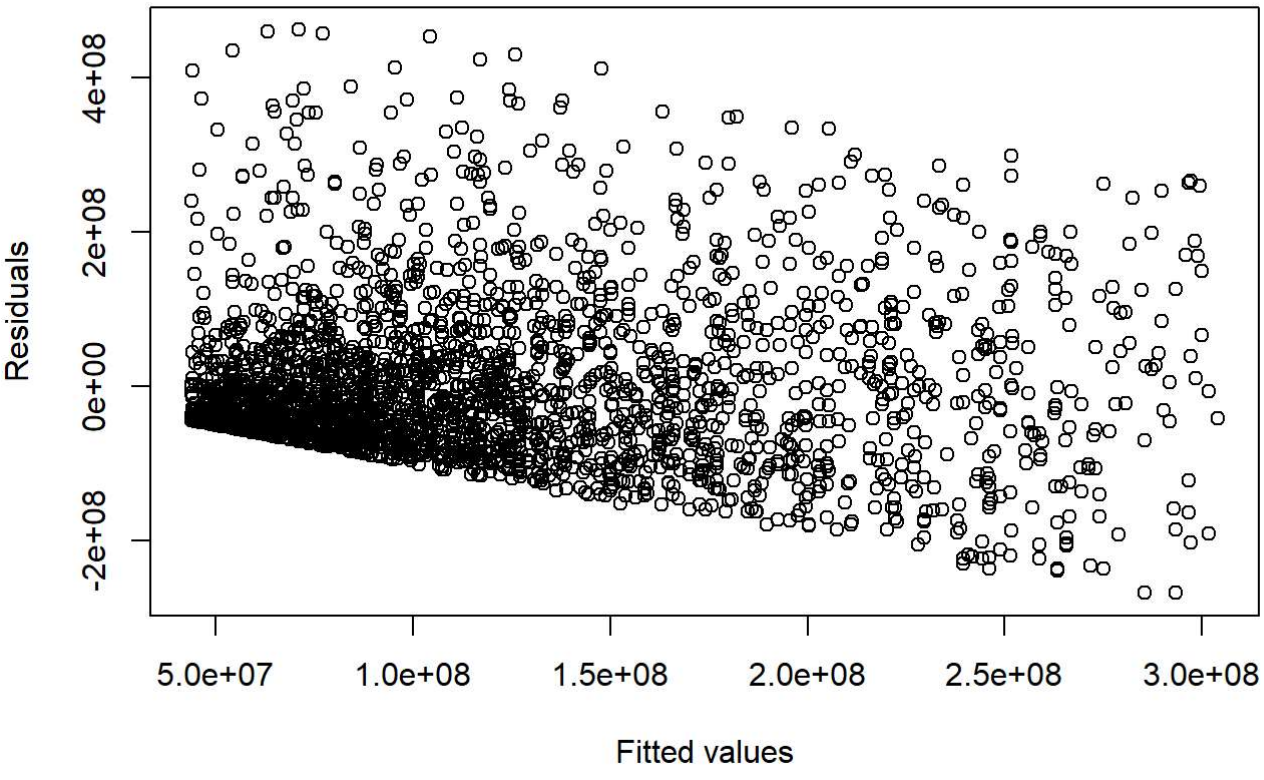
Normal Q-Q plot of the Residuals for adj_rev vs vote_average Without Out

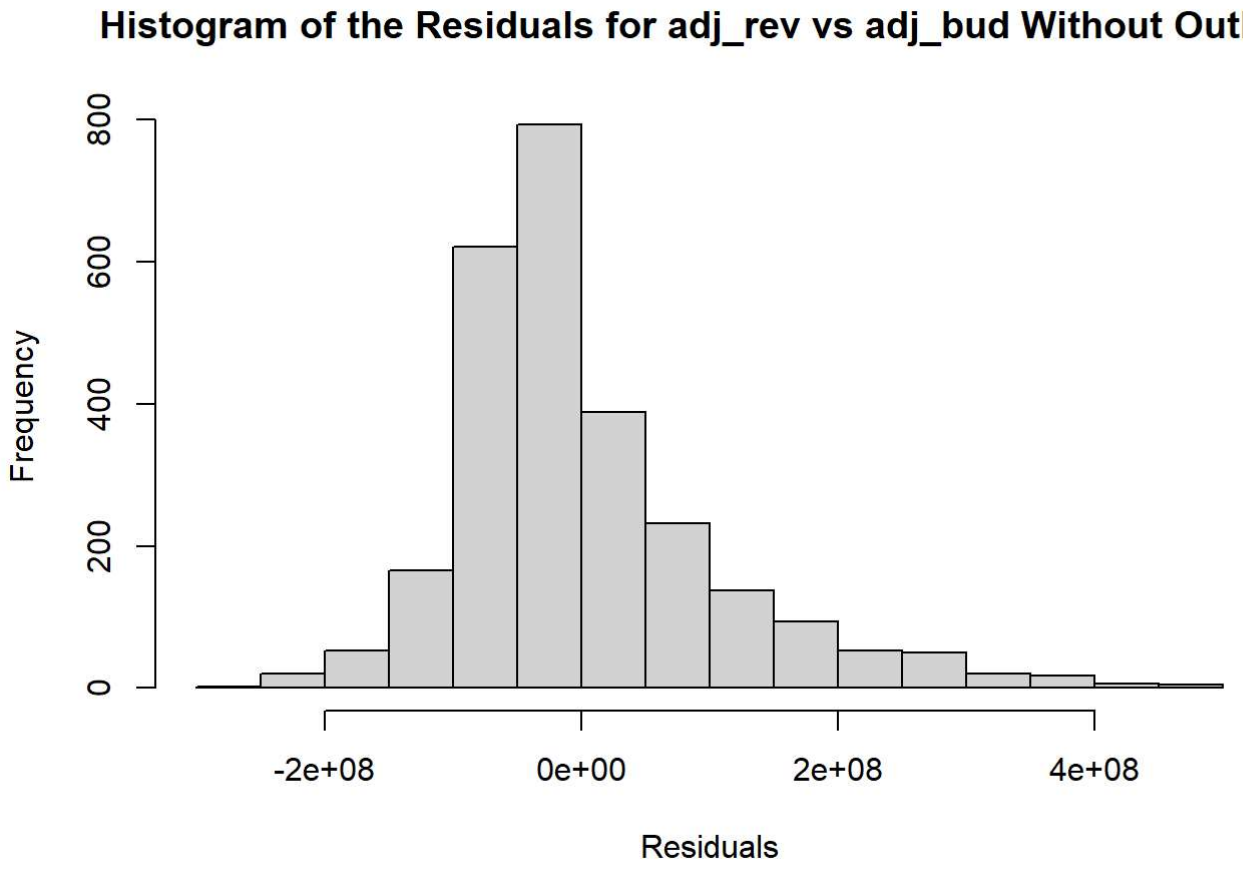


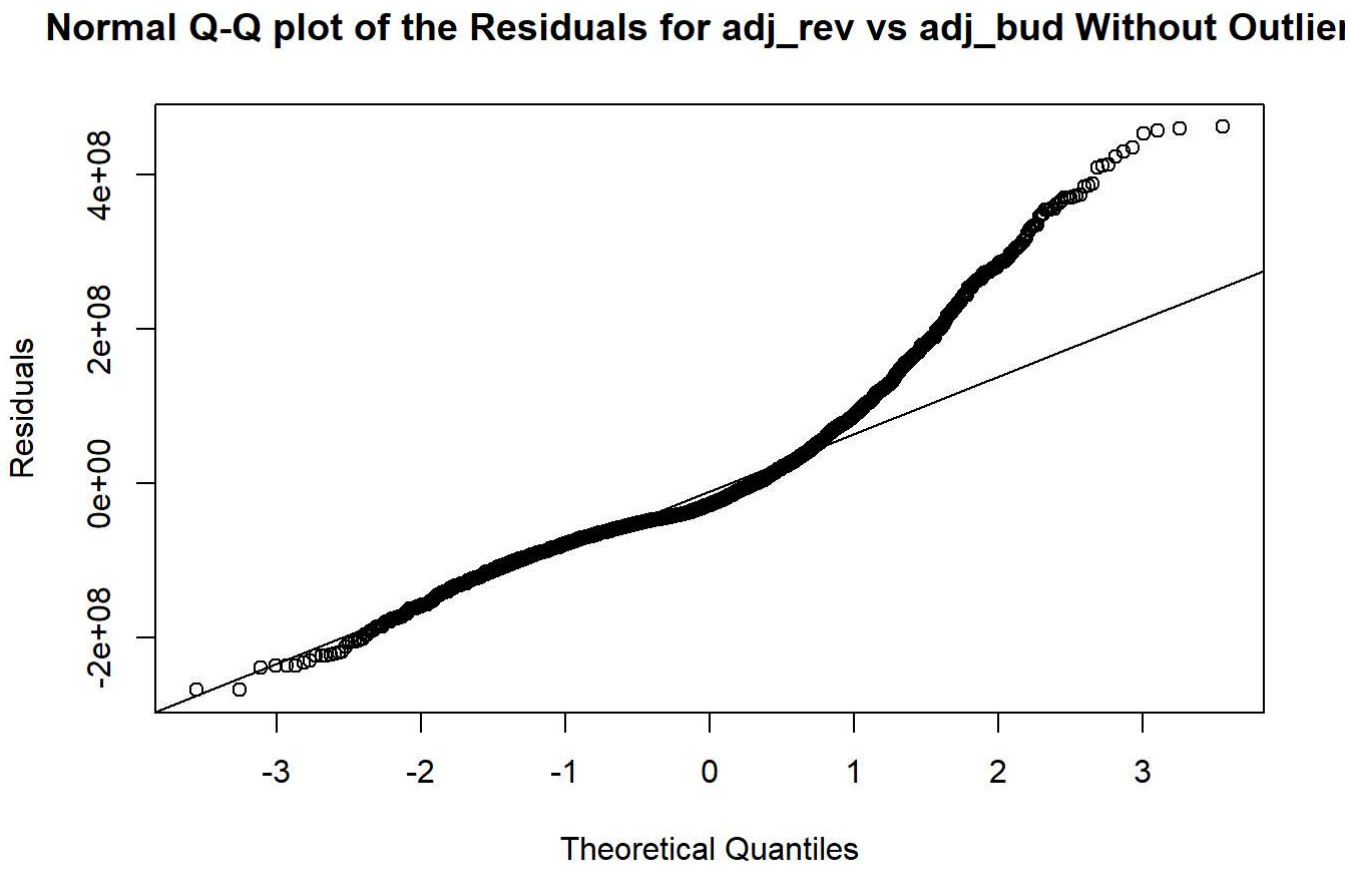
Plot of adj_rev vs adj_bud Without Outliers



Residuals vs Fitted Values for adj_rev vs adj_bud Without Outliers




Histogram of the Residuals for adj_rev vs adj_bud Without Outliers
Frequency
0
200
400
600
800
-2e+08
0e+00
2e+08
4e+08
Residuals


Normal Q-Q plot of the Residuals for adj_rev vs adj_bud Without Outlier
Residuals
-2e+08
0e+00
2e+08
4e+08
-3
-2
-1
0
1
2
3
Theoretical Quantiles

```
#The following comments are for the plots without outliers.


#From the Normal Q-Q plots, we observe that: the residuals are above the line at the beginning a
nd end of the data for variables runtime and vote_average; and that the residuals are right-skew
ed for adjusted_budget.
#We also see from the Residuals vs Fitted Values plots that the residuals appear to be trending
downwards.

#For each predictor variable, at least one of the assumptions of regression is violated. But we
can still apply transformations to the data, which can hopefully cause one or more predictors to
follow each assumption.


#We will be applying a cube root transformation to runtime and vote average because it is heavy
on the tails
```

Using the box-cox method the data to fit regression assumptions

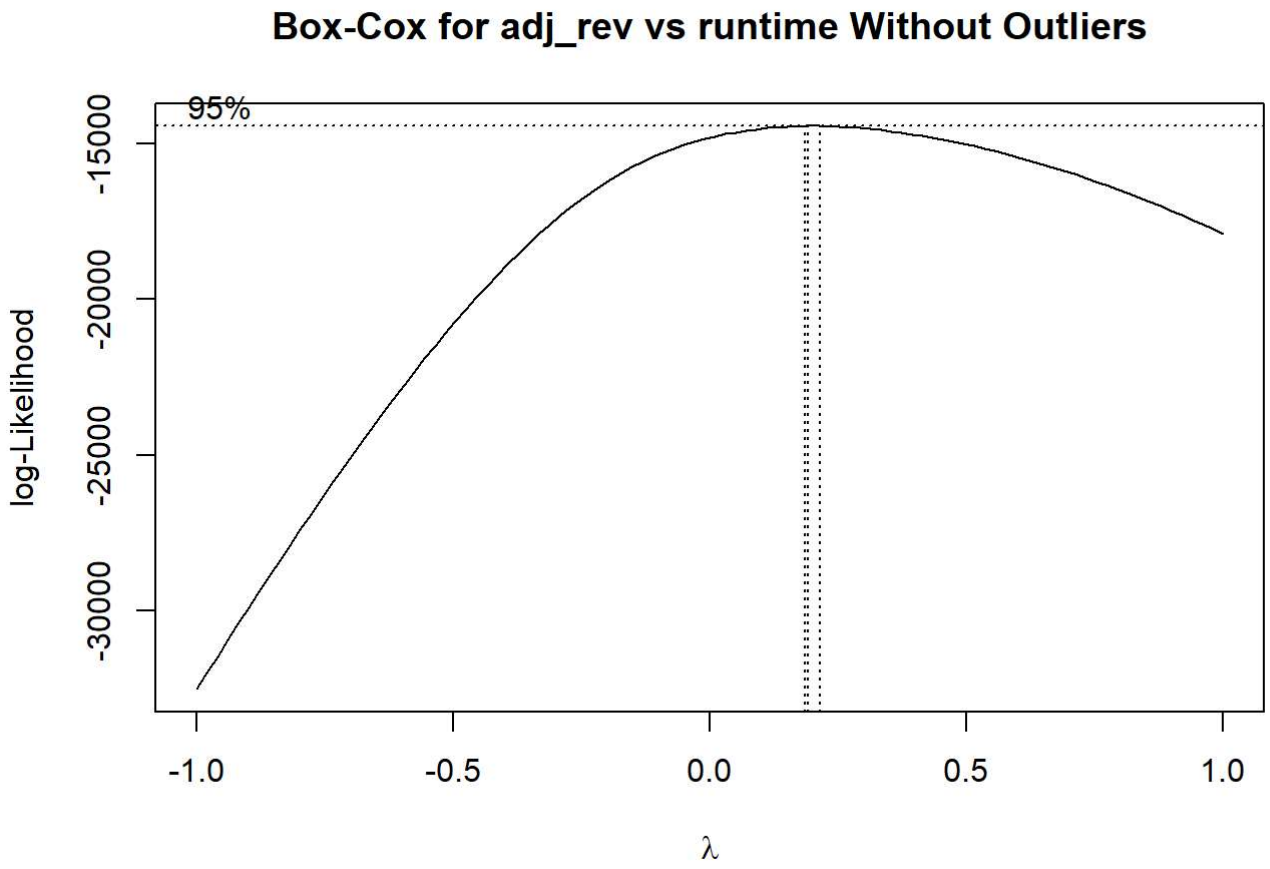
```r
#fotv stands for find optimal transformation value
#lower limit and upper limit are the limits for our transformation

fotv <- function(lower_limit, upper_limit, increment, dataset_regression, response_var) {

  # Ensure dataset_regression is a dataframe
  if (!is.data.frame(dataset_regression)) {
    stop("Error: dataset_regression must be a dataframe.")
  }

  # Check if the response variable exists in the dataset
  if (!response_var %in% colnames(dataset_regression)) {
    stop("Error: Response variable not found in dataset.")
  }

  dataset_no_response <- dataset_regression[, !(names(dataset_regression) %in% response_var)]

  # Loop through each predictor variable to generate Box-Cox plots
  for (i in 1:ncol(dataset_no_response)) {
    predictor_name <- names(dataset_no_response)[i]

    print(paste("Predictor:", predictor_name))  # Debug print

    # Dynamically create the formula
    formula <- as.formula(paste(response_var, "~", predictor_name))

    print(paste("Formula:", formula))  # Debug print

    # Run linear model with explicit environment and data passing
    linear_model <- lm(formula, data = dataset_regression, x = TRUE, y = TRUE, qr = TRUE)

    print("Linear model created successfully.")  # Debug print

    # Create the plot title
    plot_title <- paste("Box-Cox for", response_var, "vs", predictor_name, "Without Outliers")

    # Perform Box-Cox transformation with a title
    boxcox(linear_model, lambda = seq(lower_limit, upper_limit, increment))
    title(main = plot_title)
  }
}




library("MASS")
```

```
##
## Attaching package: 'MASS'
```

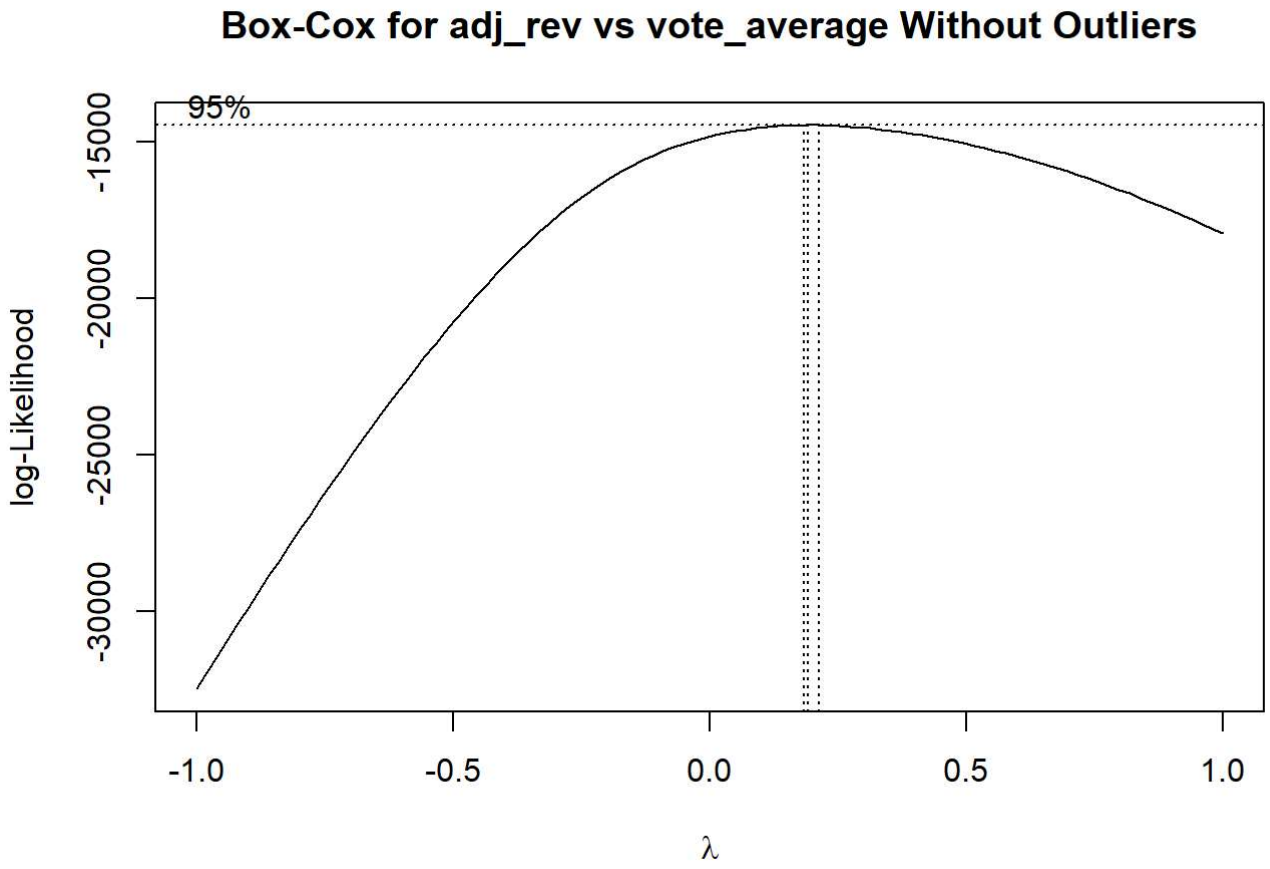
```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
fotv(-1, 1, 0.1, movies_regression, "adj_rev")
```

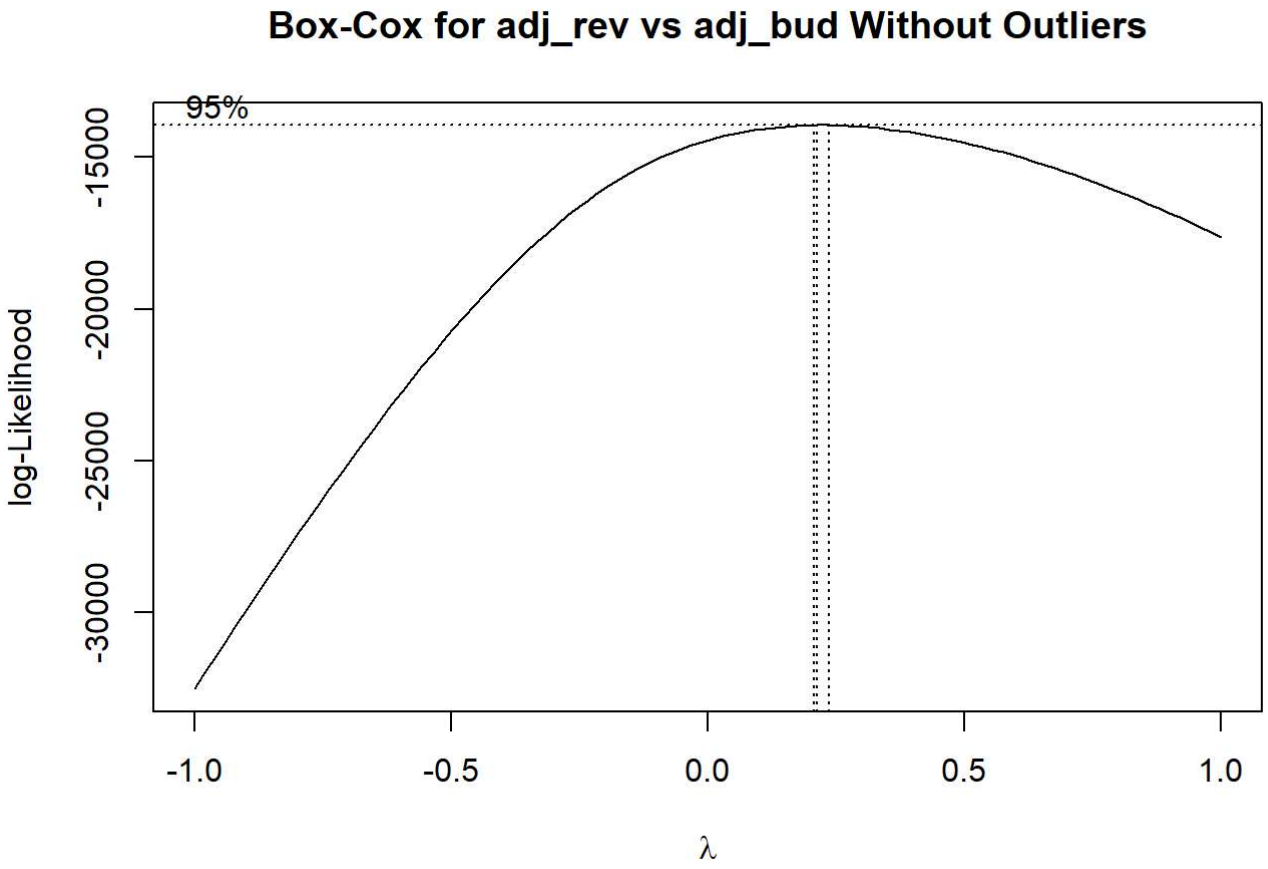
```
## [1] "Predictor: runtime"
## [1] "Formula: ~"       "Formula: adj_rev" "Formula: runtime"
## [1] "Linear model created successfully."
```




```
## [1] "Predictor: vote_average"
## [1] "Formula: ~"            "Formula: adj_rev"      "Formula: vote_average"
## [1] "Linear model created successfully."
```

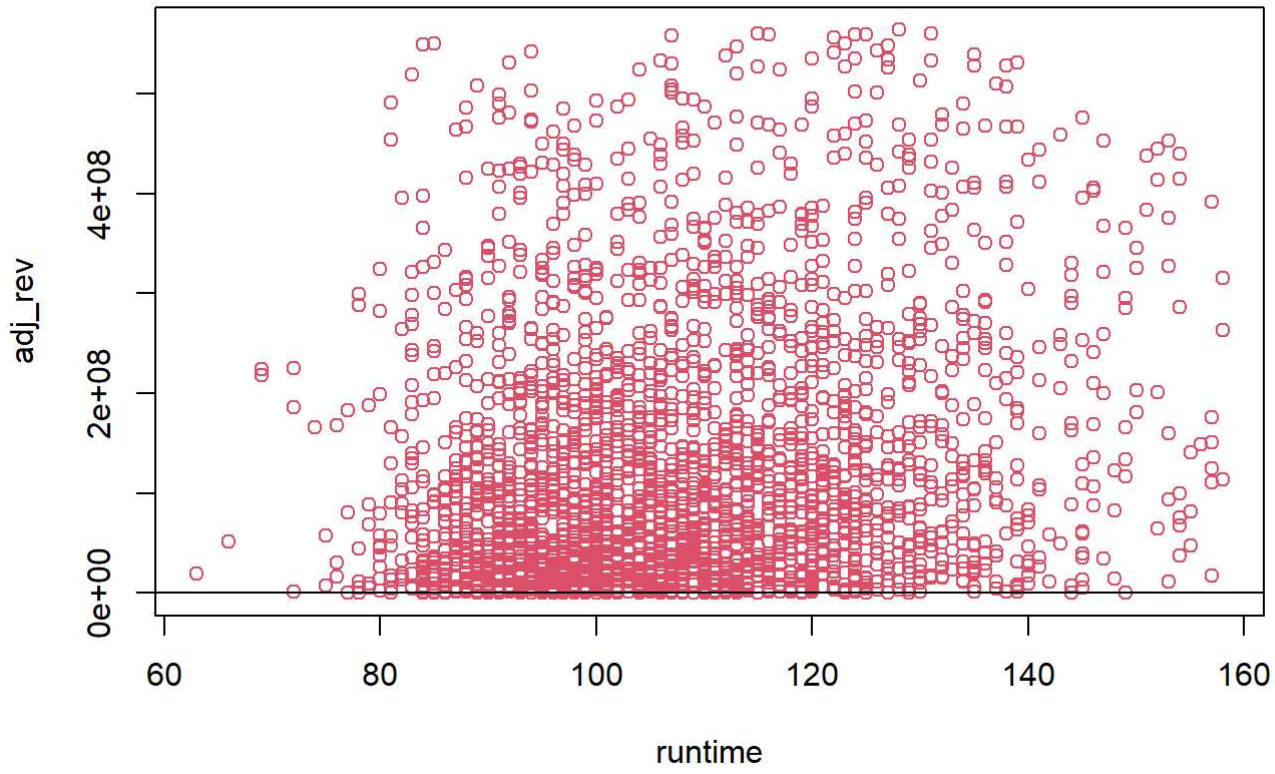



```
## [1] "Predictor: adj_bud"
## [1] "Formula: ~"       "Formula: adj_rev" "Formula: adj_bud"
## [1] "Linear model created successfully."
```

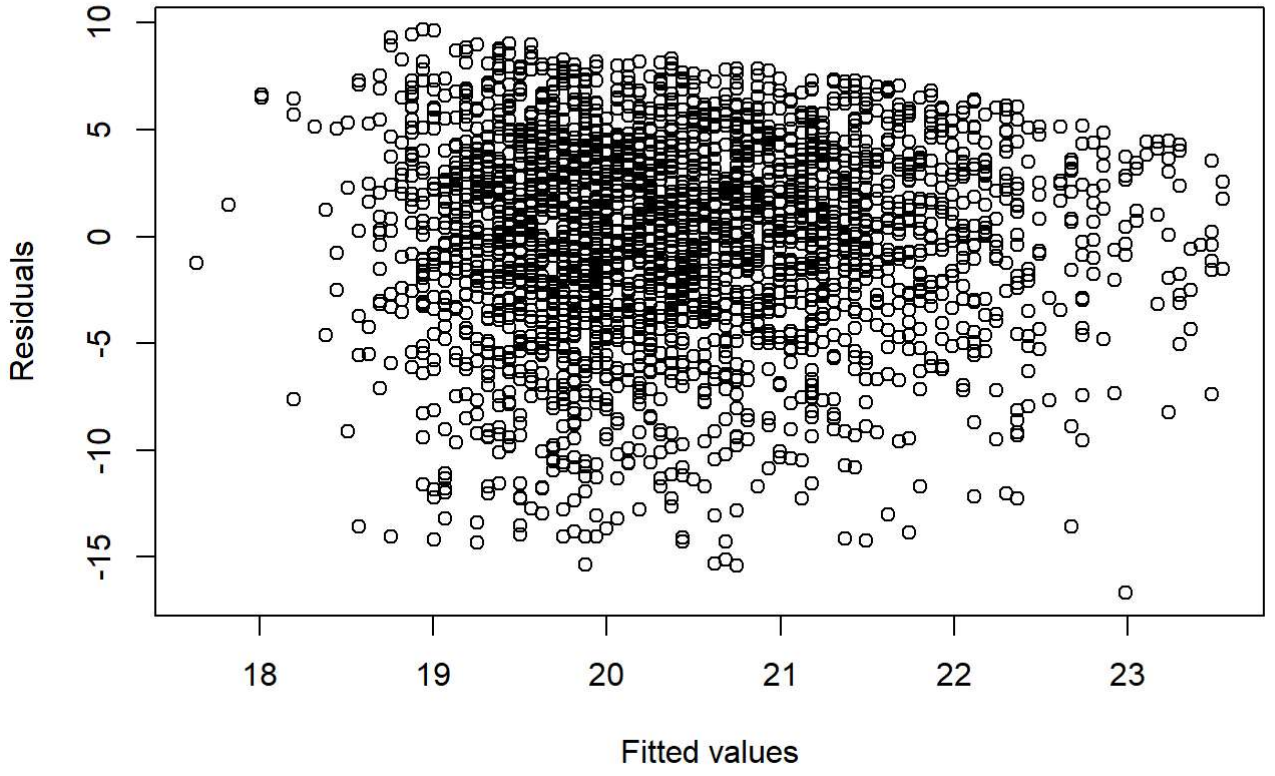


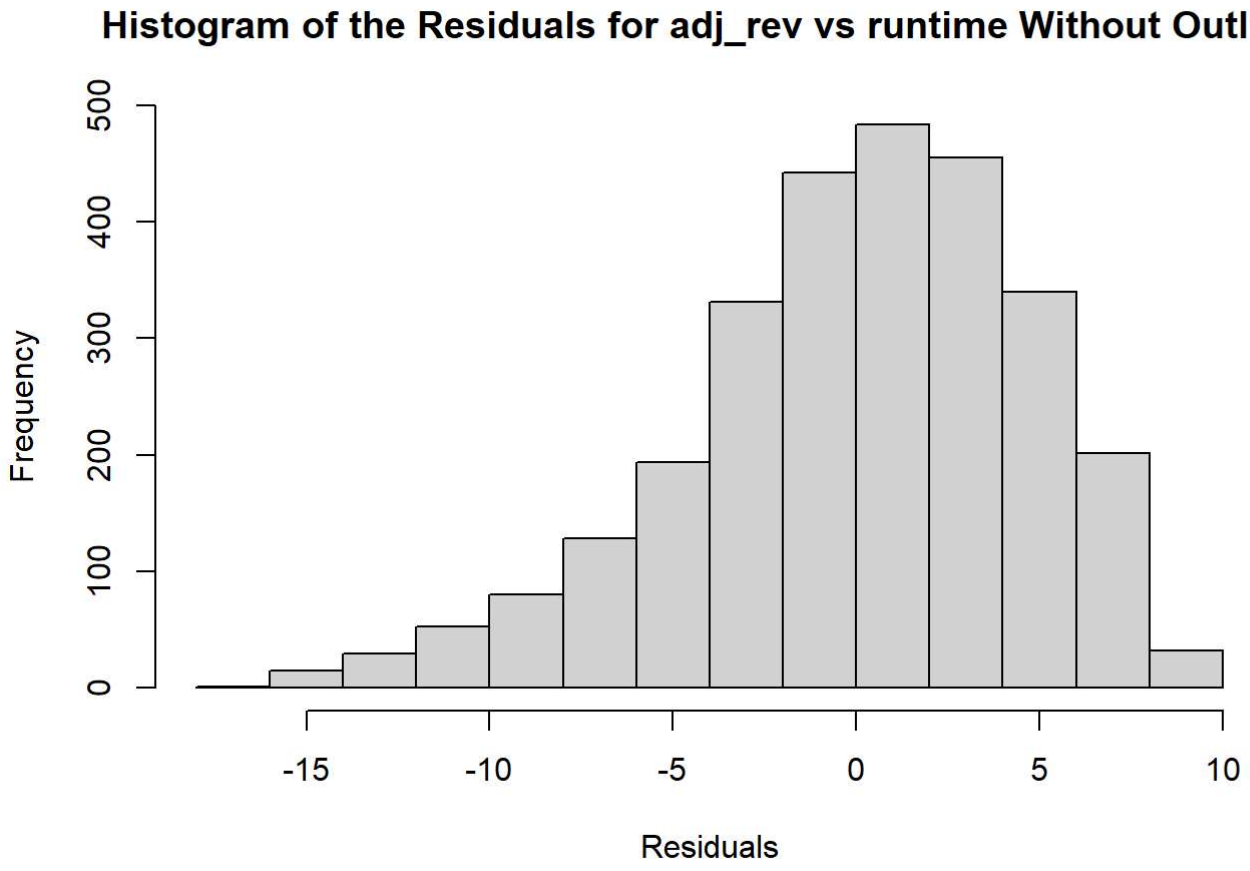

```
#From the plots, we observe that the optimal values for run_time and vote_average are 1/6. When comparing these variables with adj_rev, we will be applying a transformation of 1/6 to the response.
#We also see that the optimal value for adj_bud is 1/4, so we will be applying a transformation of 1/4 to the adj_rev when comparing it with adj_bud.



movies_sixth_root = subset(movies_regression, select=-adj_bud)
movies_fourth_root = subset(movies_regression, select=-c(runtime, vote_average))



check_regression_assumptions(movies_sixth_root, "adj_rev", T, 1/6)
```

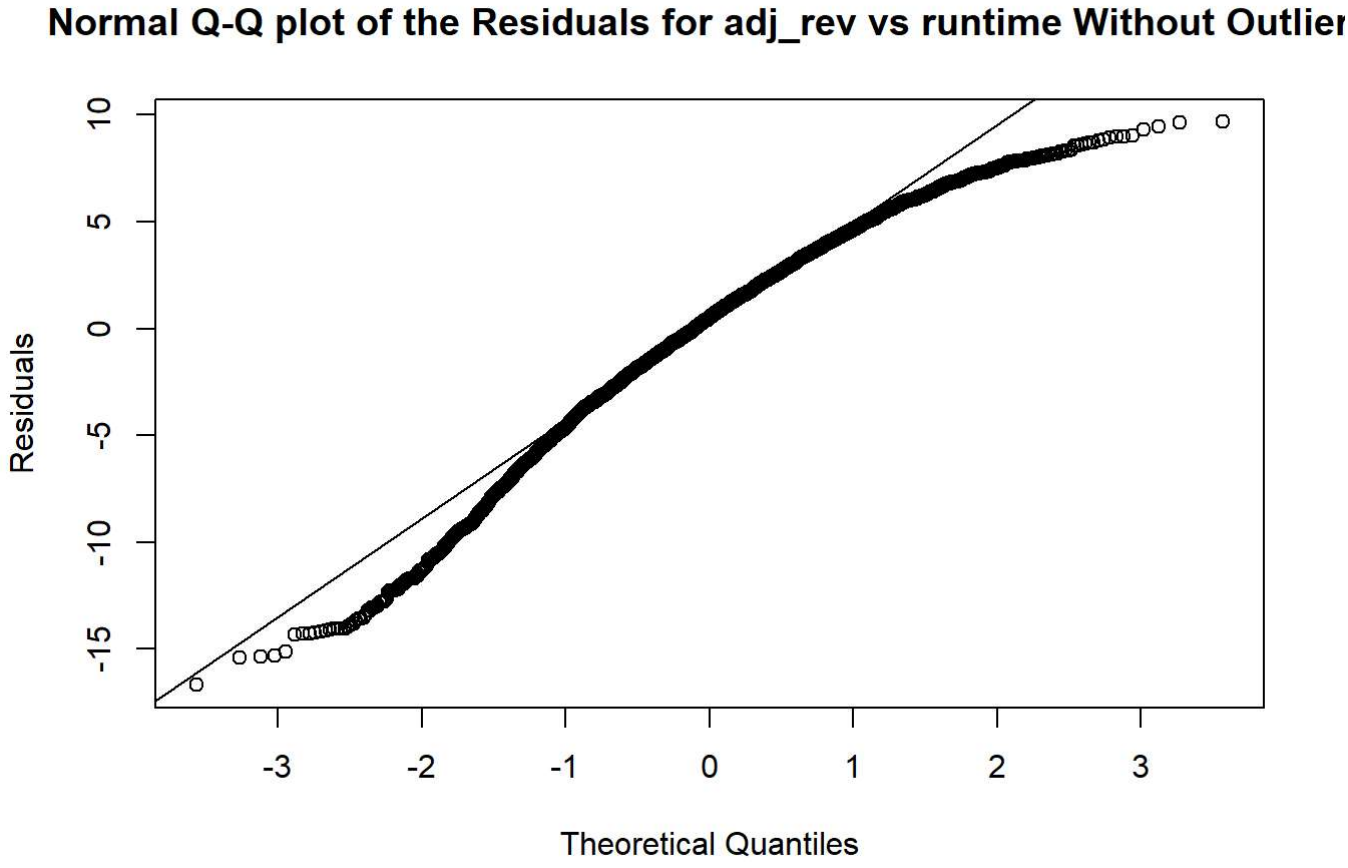
Plot of adj_rev vs runtime Without Outliers

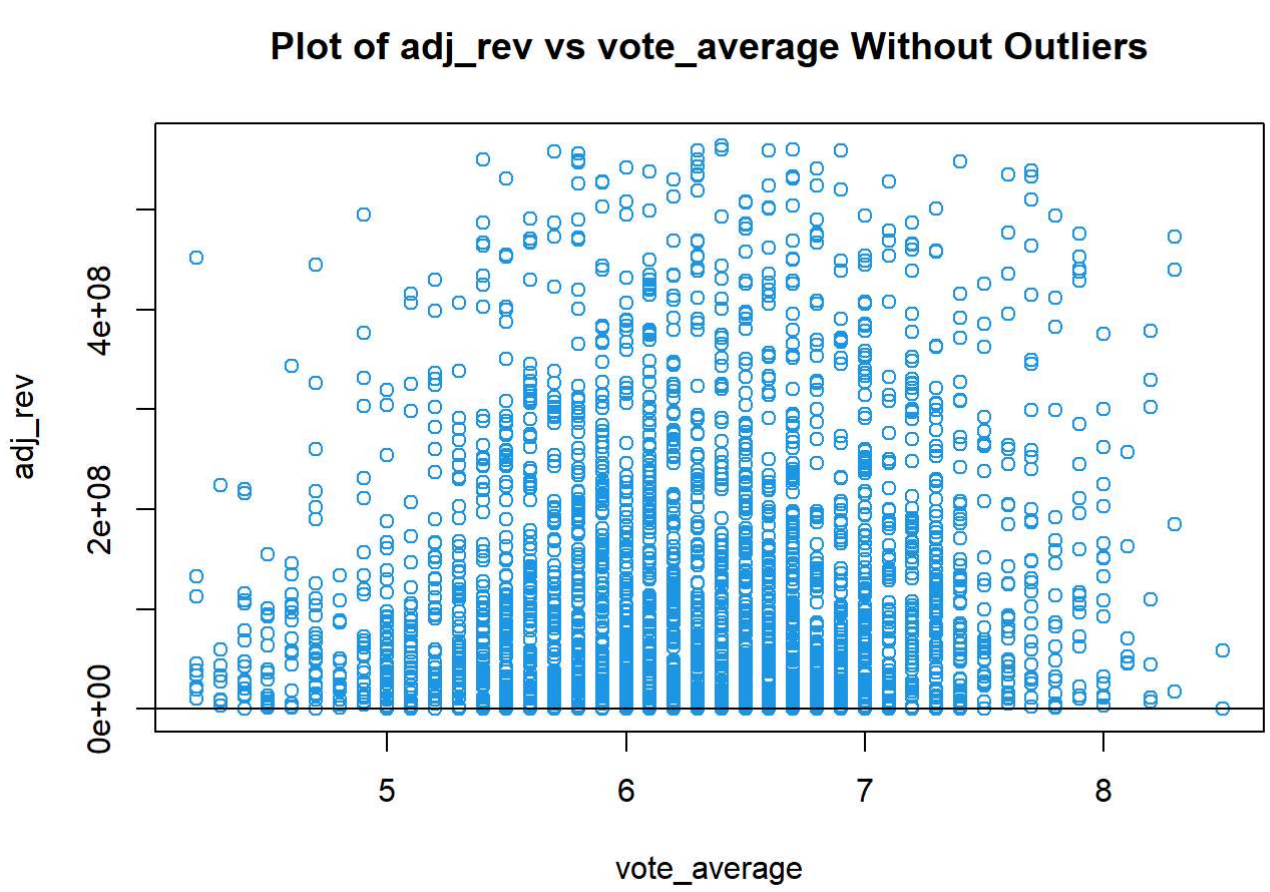


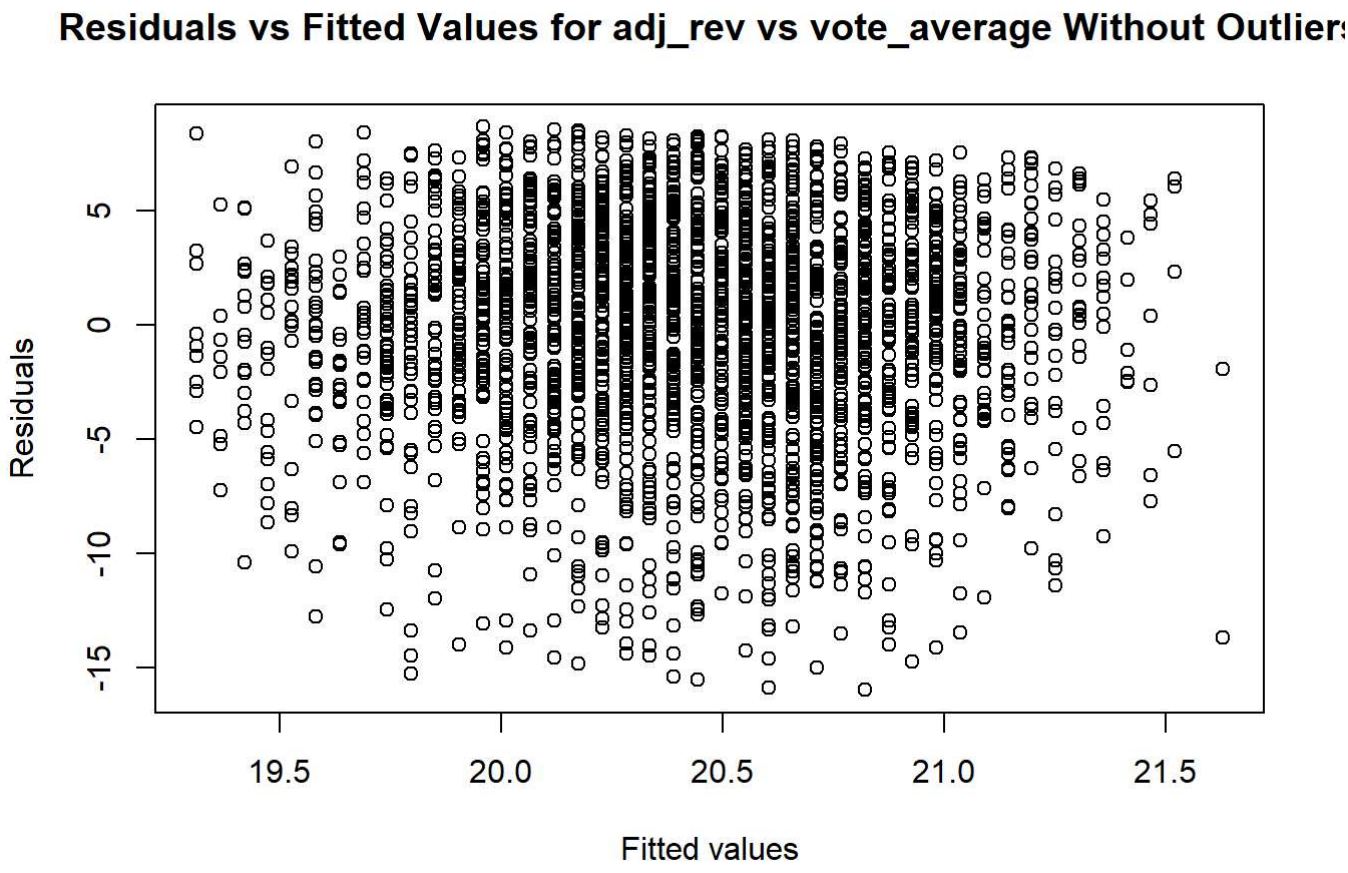
Residuals vs Fitted Values for adj_rev vs runtime Without Outliers



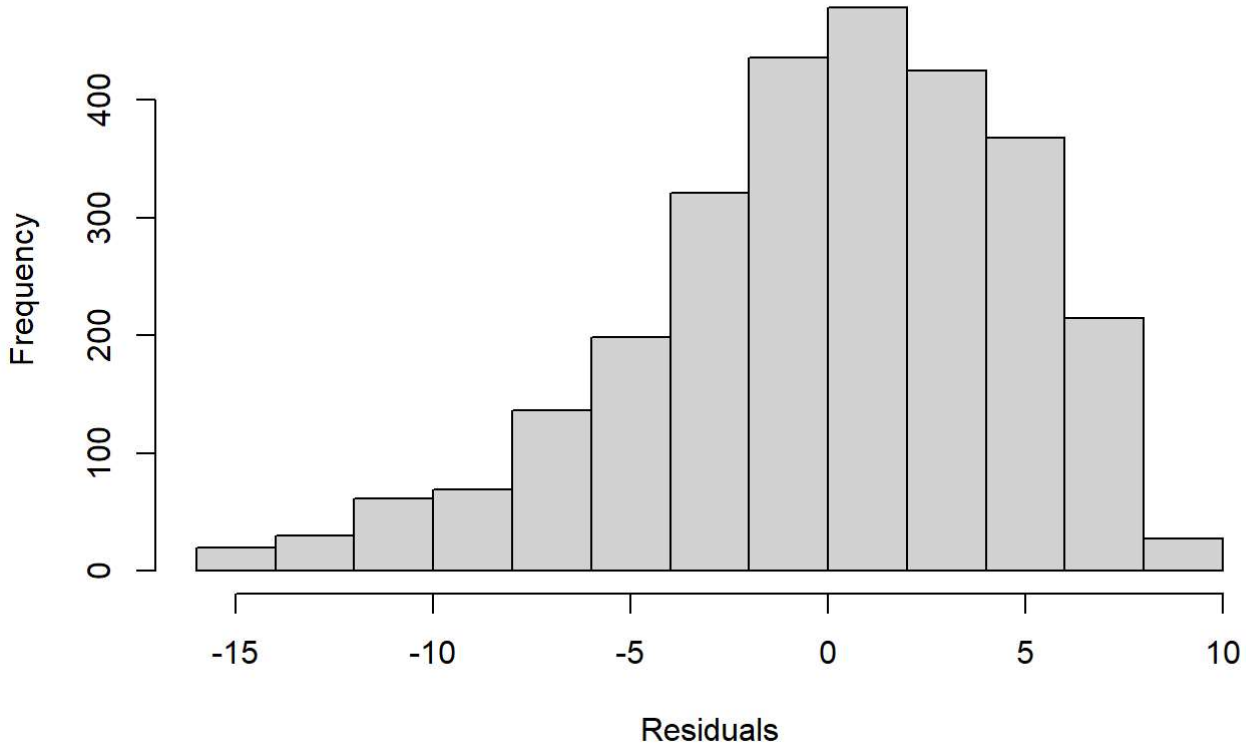

Histogram of the Residuals for adj_rev vs runtime Without Outliers
Frequency
0
100
200
300
400
500
-15
-10
-5
0
5
10
Residuals


Normal Q-Q plot of the Residuals for adj_rev vs runtime Without Outlier
Residuals
-15
-10
-5
0
5
10
-3
-2
-1
0
1
2
3
Theoretical Quantiles

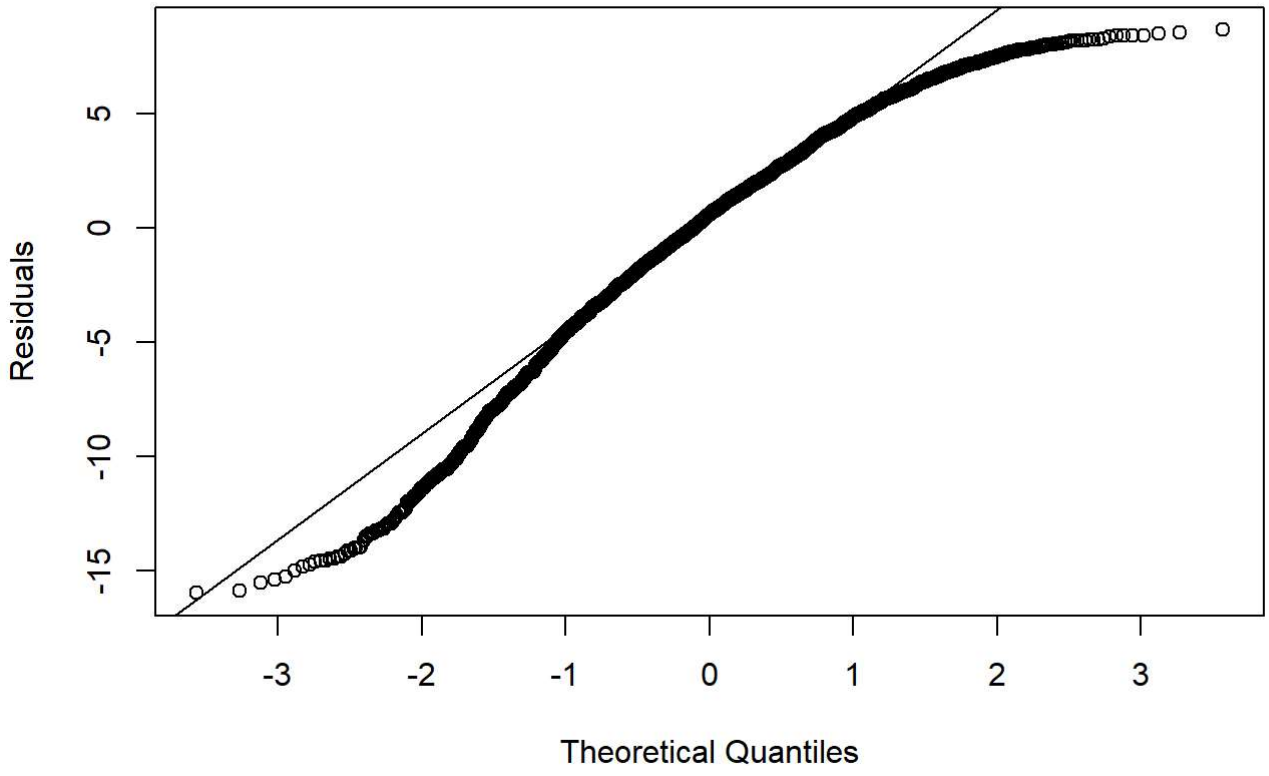

Plot of adj_rev vs vote_average Without Outliers
adj_rev
0e+00
2e+08
4e+08
5
6
7
8
vote_average


Residuals vs Fitted Values for adj_rev vs vote_average Without Outliers
Residuals
-15
-10
-5
0
5
19.5
20.0
20.5
21.0
21.5
Fitted values

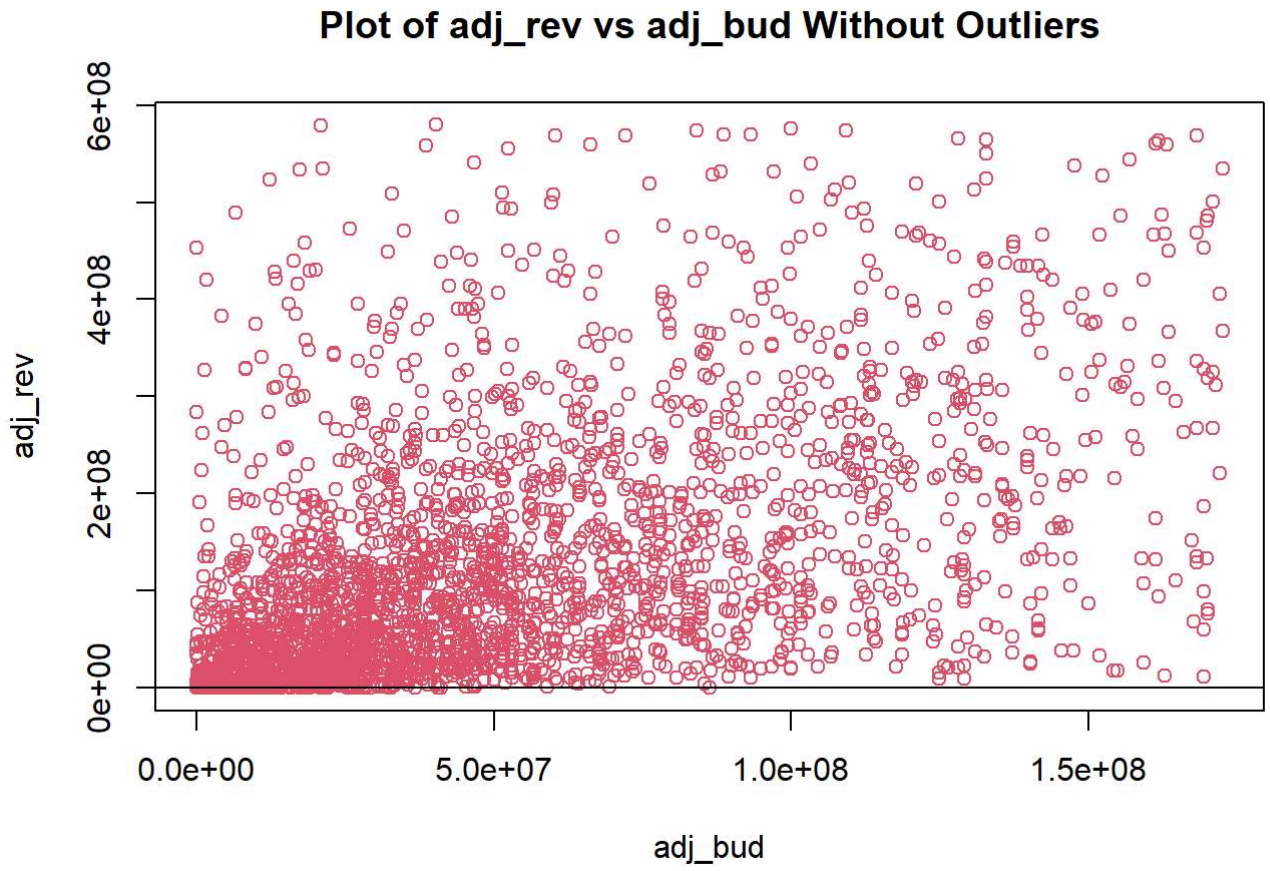
Histogram of the Residuals for adj_rev vs vote_average Without Outlier

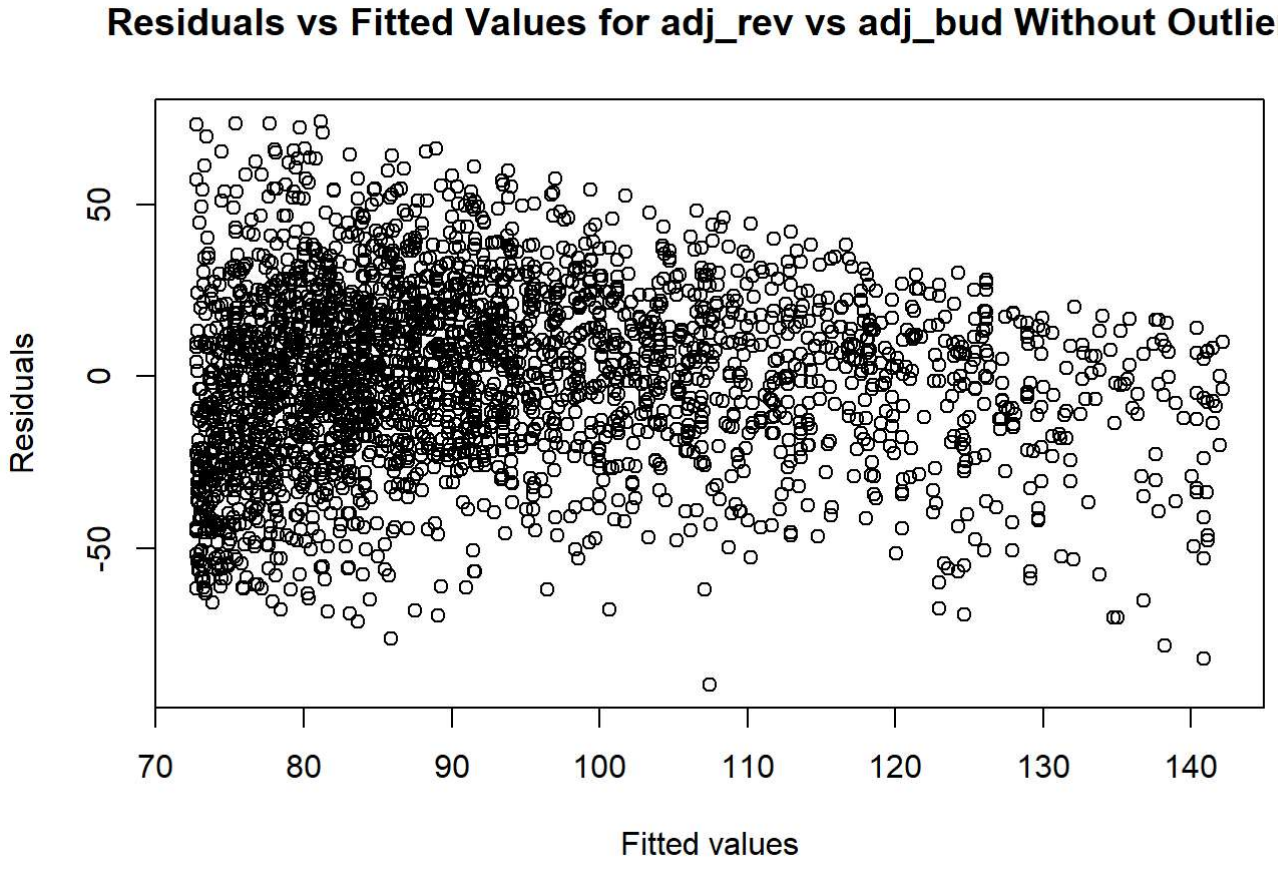


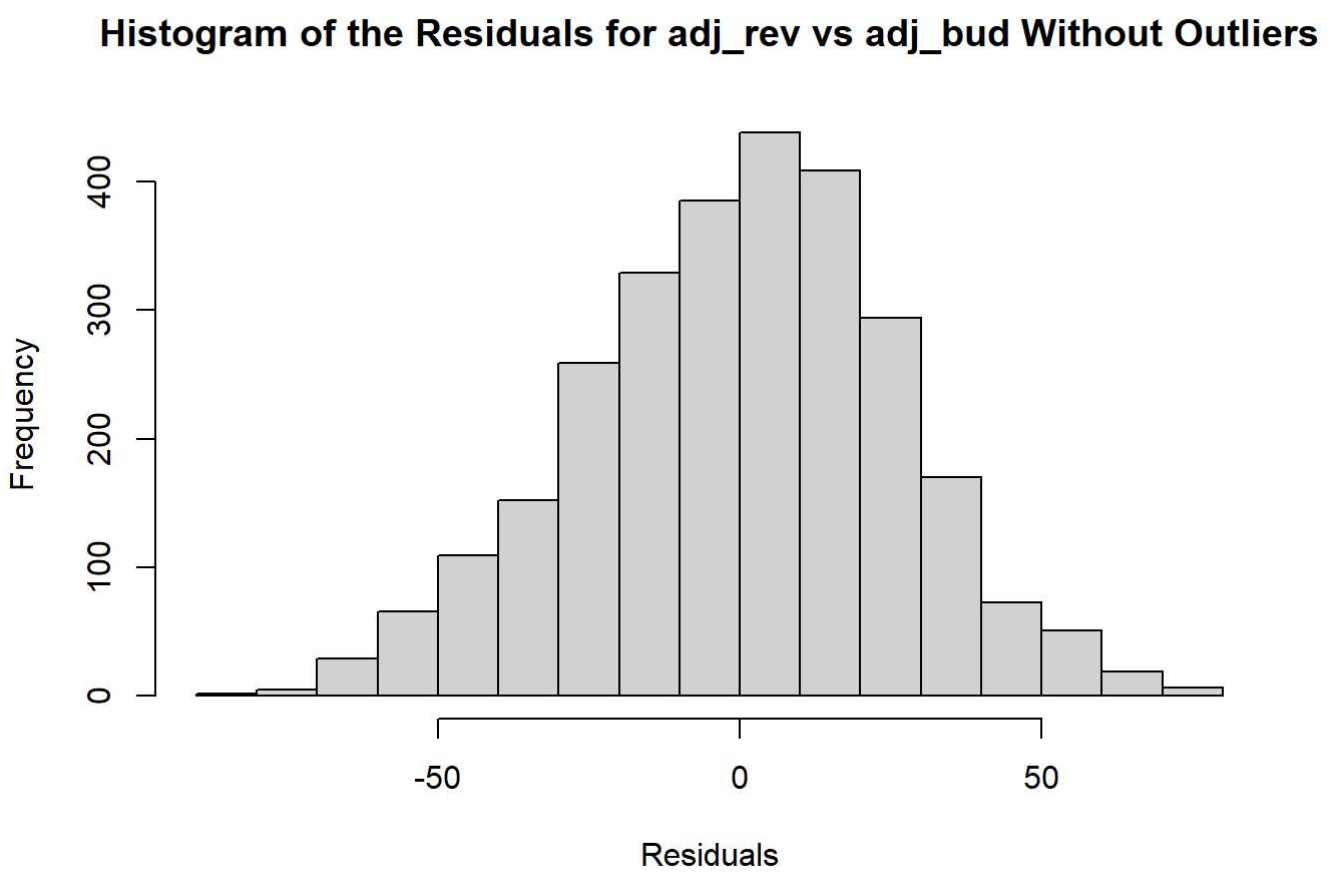
Normal Q-Q plot of the Residuals for adj_rev vs vote_average Without Out

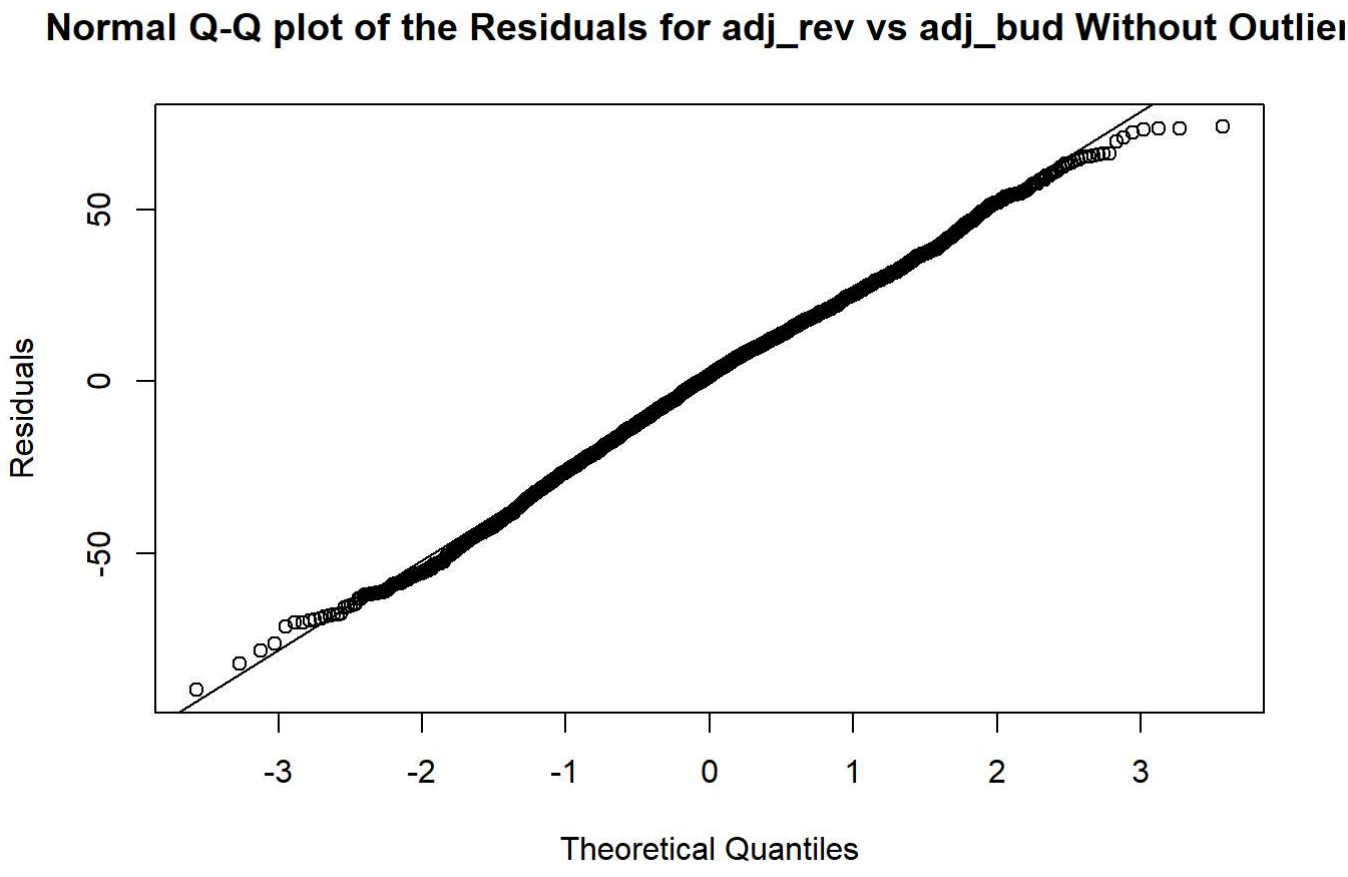


```
check_regression_assumptions(movies_fourth_root, "adj_rev", T, 1/4)
```


Plot of adj_rev vs adj_bud Without Outliers
adj_rev
0e+00
2e+08
4e+08
6e+08
0.0e+00
5.0e+07
1.0e+08
1.5e+08
adj_bud


Residuals vs Fitted Values for adj_rev vs adj_bud Without Outliers
Residuals
-50
0
50
70
80
90
100
110
120
130
140
Fitted values


Histogram of the Residuals for adj_rev vs adj_bud Without Outliers
Frequency
0
100
200
300
400
-50
0
50
Residuals


Normal Q-Q plot of the Residuals for adj_rev vs adj_bud Without Outlier
Residuals
-50
0
50
-3
-2
-1
0
1
2
3
Theoretical Quantiles

```
#As we can see from the plots, the relationship between our predictor variables and adj_rev has significantly improved!

#We can see from the plots that the variance is constant for both runtime and vote_average, and that the errors are independent.
#For these variables, the normality of the residuals is more open to interpretation as the tails deviate somewhat from the line in the Q-Q plot. But because the vast majority of the points fall on or close to the line, it is acceptable to say that the normality assumption for the residuals has been met.


#For adj_bud, there appears to be a somewhat of a funneling in the residuals, especially towards the top part of the Residuals vs Fitted plot.

#Because of this, we won't be using adj_bud given that it violates the constant variance assumption.
```

Comments: From here we have two options: We can remove adj_bud and have a model with only the predictor variables vote_average and runtime. Or, we can keep adj_bud, and see if its assumptions become satisfied in the full model with all predictors.

We will test both options separately.

Building functions to make models

```r
train_test = function(movies_data){

  #t_t is short for train_test
  t_t_indices = sample(seq_len(nrow(movies_data)), size = 0.7 * nrow(movies_data))

  train_data = movies_data[t_t_indices, ]
  test_data = movies_data[-t_t_indices, ]

  return(list(training_data = train_data, testing_data = test_data))
}


#mrm stands for multiple regression models
mrm_generator = function(movies_data, response_var, predictor_formula, remove_outliers){

  #t_t is short for train_test
  t_t_data = train_test(movies_data)


  formula = as.formula(paste(response_var, "~", predictor_formula))
  print(formula)

  mrm = lm(formula, data = t_t_data$training_data)


  #Removing outliers by first running the original one
  if(remove_outliers){
    influencers = influence.measures(mrm)
    influential_rows = which(influencers$is.inf[, "hat"])

    #Removing the influential points to improve our model
    t_t_data$training_data = t_t_data$training_data[-influential_rows, ]
    mrm = lm(formula, data = t_t_data$training_data)
  }

  predictions = predict(mrm, newdata = t_t_data$testing_data)

  #Gets the root MSE
  rmse = sqrt(mean((t_t_data$testing_data[[response_var]] - predictions)^2))


  return(list(rmse = rmse, model = mrm))
}

#Shows the model's summary, and rsq and rmse values
model_comparison = function(given_model){

  model_summary = summary(given_model$model)
  print(model_summary)


  print(given_model$rmse)
```

```
  print(model_summary$r.squared)
}
```

Option 1:

Our model has the formula:

$$y^{1/4} = \beta_0 + \beta_1 x_{va} + \beta_2 x_{rt} + \epsilon$$

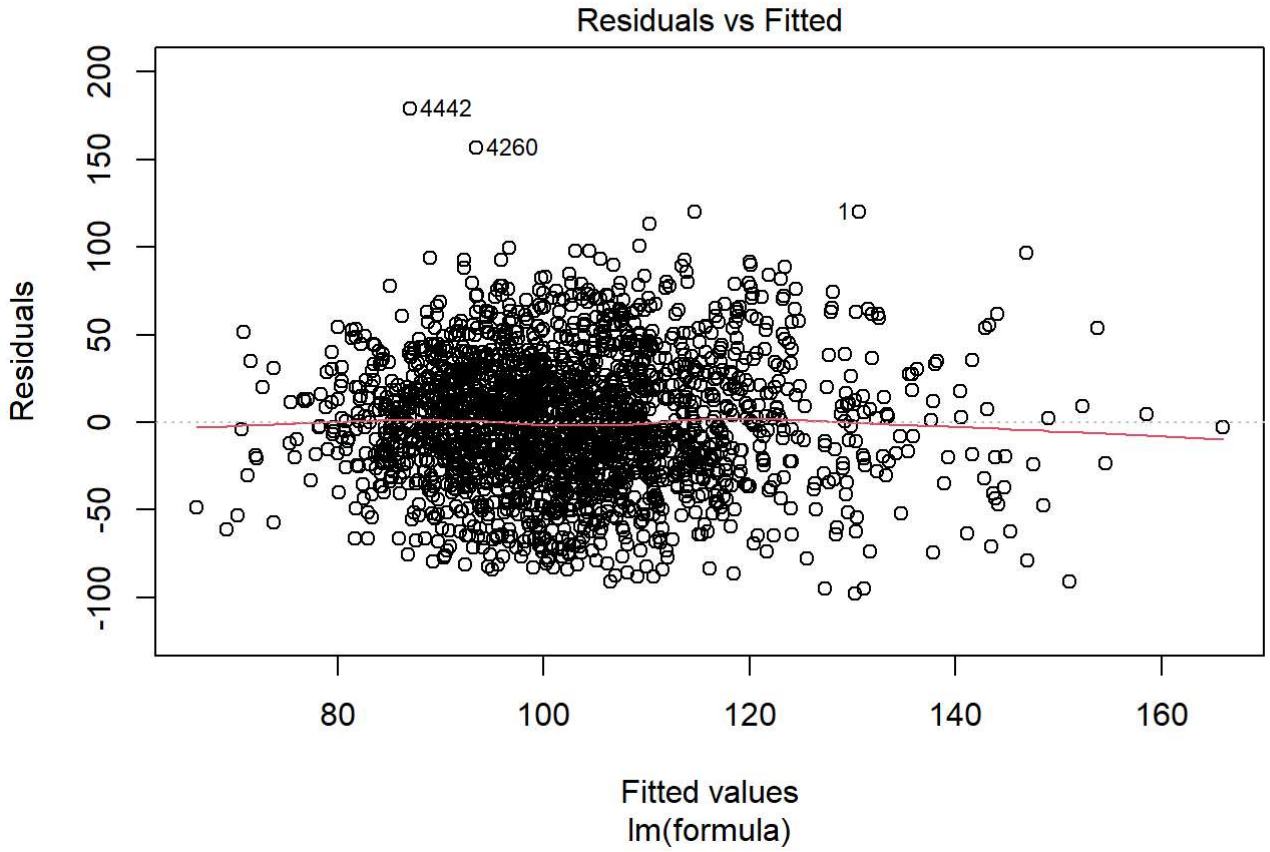
Where va stands for vote_average and rt stands for runtime.

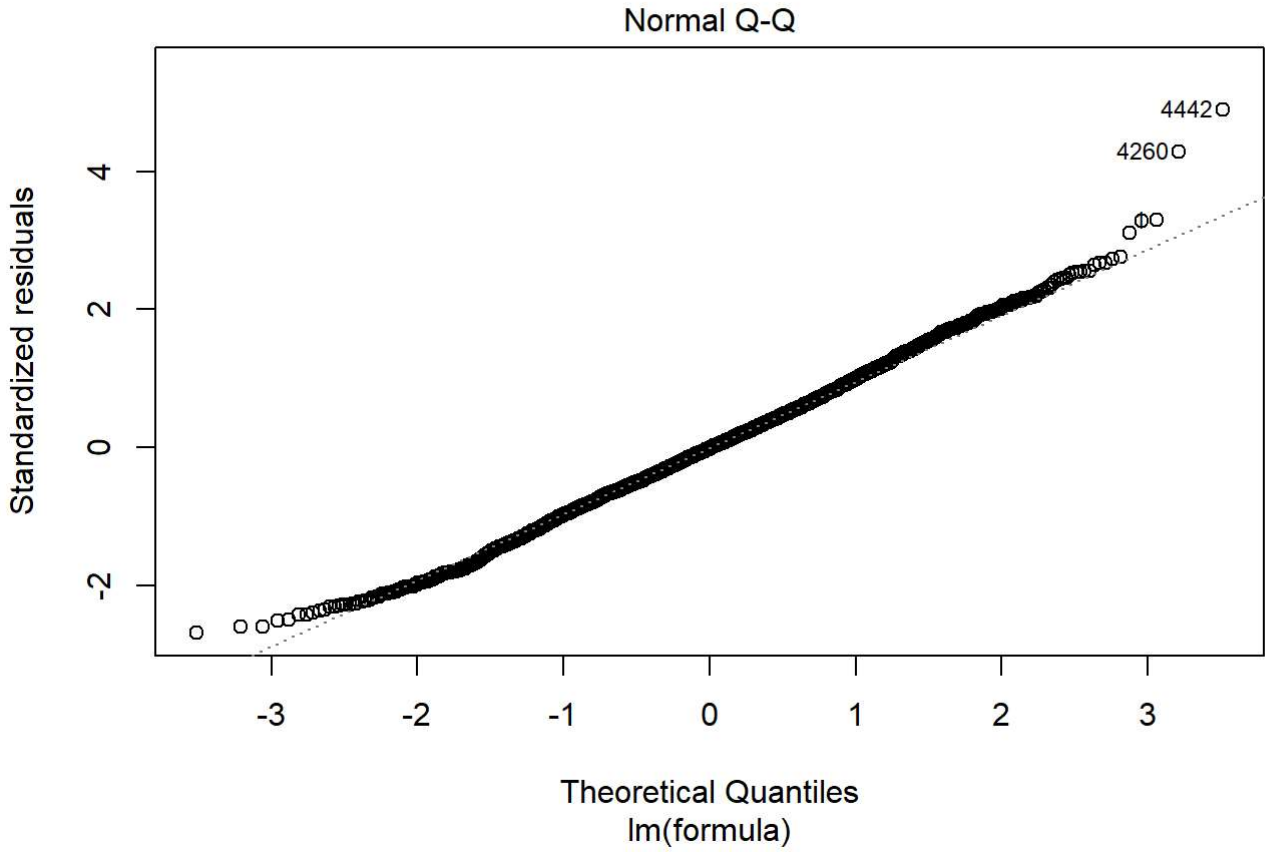
No interaction term will be made between vote_average and runtime because the two are intuitively independent from each other.

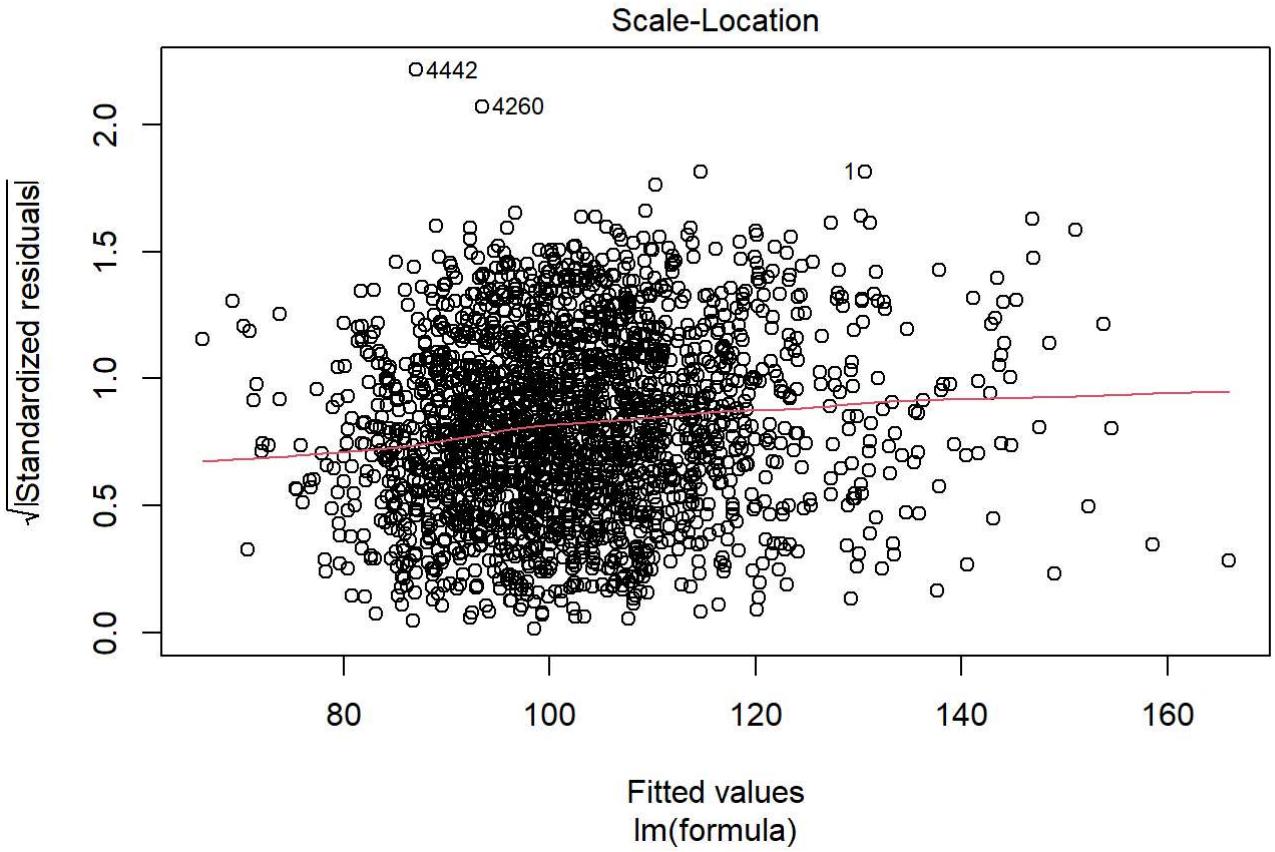
```
#Removing adjusted_budget.
#opt1 means option1
movies_regression_opt1 = subset(movies_regression, select = -adj_bud)


#Taking the fourth root of adj_rev, our response variable
movies_regression_opt1[,3] = movies_regression_opt1[,3]^(1/4)


opt1 = mrm_generator(movies_regression_opt1, "adj_rev", "runtime + vote_average", F)
```

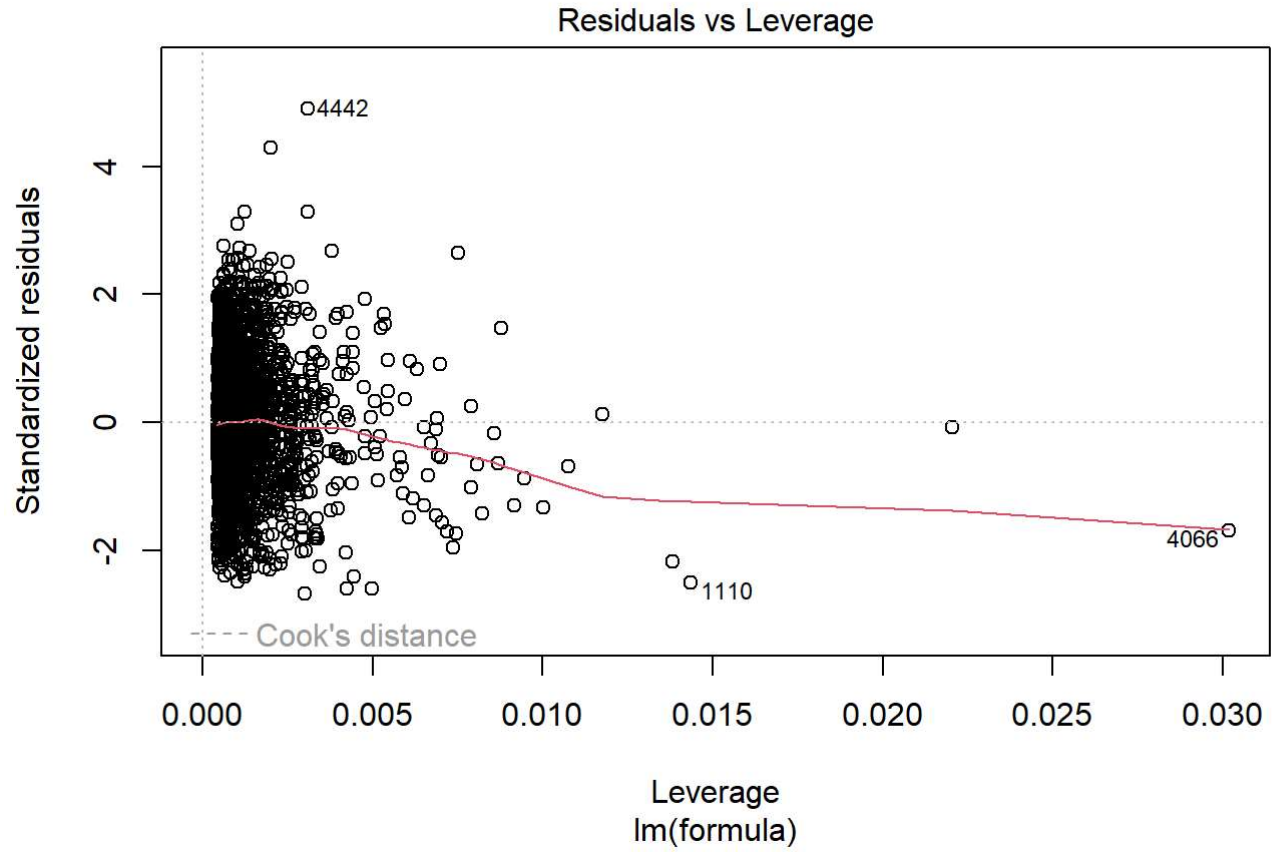
```
## adj_rev ~ runtime + vote_average
## <environment: 0x000001d2997916d0>
```

```
plot(opt1$model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(formula)
4442
4260
1
200
150
100
50
0
-50
-100
80
100
120
140
160


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(formula)
4442
4260
4
2
0
-2
-3
-2
-1
0
1
2
3


Scale-Location
4442
4260
1
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
80
100
120
140
160
Fitted values
lm(formula)


Residuals vs Leverage
4442
4066
1110
Cook's distance
Standardized residuals
-2
0
2
4
0.000
0.005
0.010
0.015
0.020
0.025
0.030
Leverage
lm(formula)

```
#We observe from the plots of opt1 that there are outliers.
#Let us remove the outliers in our response and predictor to try to create a better model
opt1_no_outlier = mrm_generator(movies_regression_opt1, "adj_rev", "runtime + vote_average", T)
```

```
## adj_rev ~ runtime + vote_average
## <environment: 0x000001d29b0dce68>
```

```
model_comparison(opt1); model_comparison(opt1_no_outlier)
```

```
##
## Call:
## lm(formula = formula, data = t_t_data$training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.056 -23.992  -0.421  23.252 178.861
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.95570    5.98229   2.166   0.0304 *
## runtime       0.44683    0.04019  11.118  < 2e-16 ***
## vote_average  6.29608    0.95554   6.589  5.5e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.54 on 2244 degrees of freedom
## Multiple R-squared:  0.1059, Adjusted R-squared:  0.1051
## F-statistic: 132.9 on 2 and 2244 DF,  p-value: < 2.2e-16
##
## [1] 37.17606
## [1] 0.1058992
```

```
##
## Call:
## lm(formula = formula, data = t_t_data$training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.608 -24.079  -0.167  23.968 158.237
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.48838    6.68912   0.970    0.332
## runtime       0.48657    0.04899   9.933  < 2e-16 ***
## vote_average  6.52141    1.03836   6.280 4.06e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.46 on 2176 degrees of freedom
## Multiple R-squared:  0.0921, Adjusted R-squared:  0.09127
## F-statistic: 110.4 on 2 and 2176 DF,  p-value: < 2.2e-16
##
## [1] 36.10792
## [1] 0.09210332
```

We observe that the

$$r^2$$

value for the original model is higher, and the rmse (root mean squared error) value is lower for the original model. Because of this, we conclude that the original model is better than the newer one.

Option 2:

Interaction terms will be created between vote_average and adj_bud, and runtime and adj_bud for the following reasons:

There may be a relationship between a movie's rating and its budget, with higher budget movies being thought to produce a movie rating. It is thought that the longer a movie is, the larger its budget needs to be, generally speaking.

Our model has the formula:

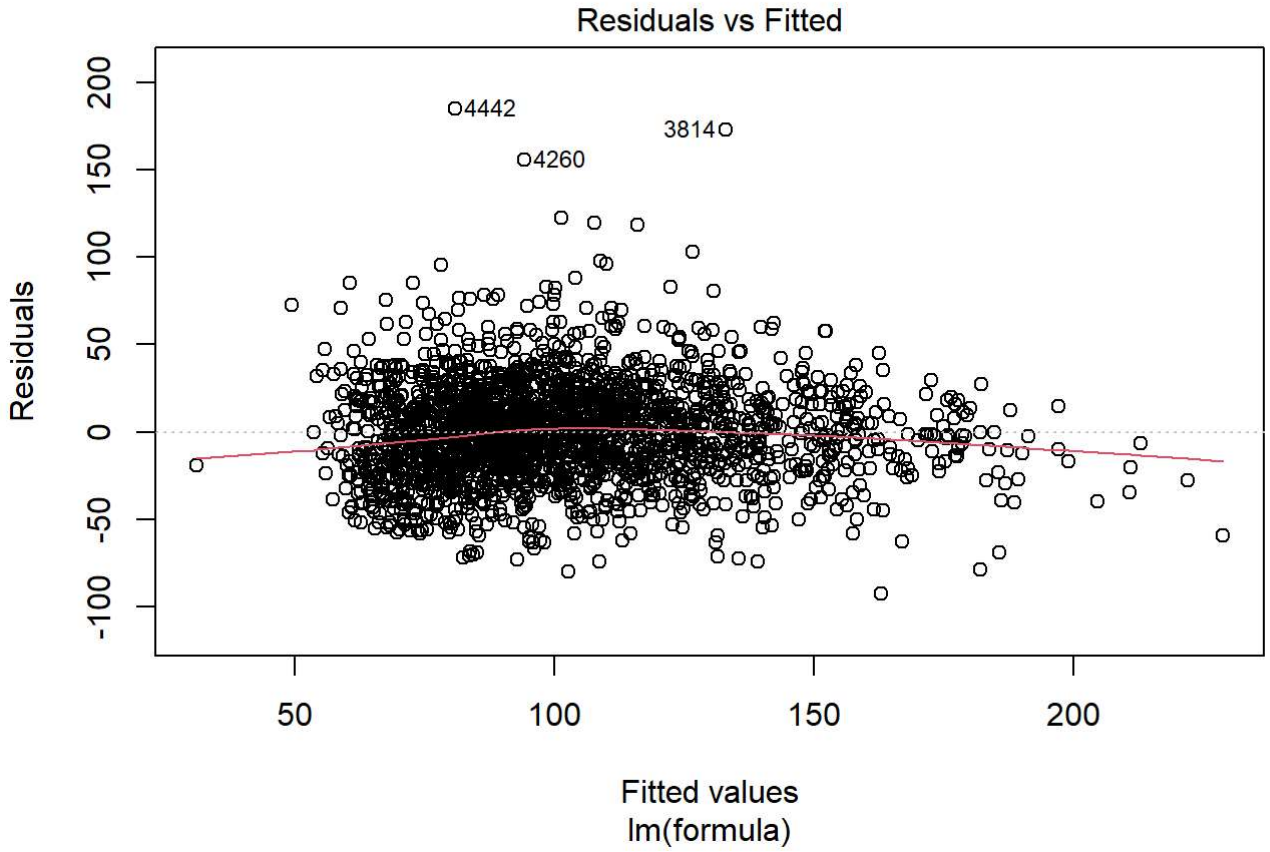
$$y^{1/4} = \beta_0 + \beta_1 x_{va} + \beta_2 x_{rt} + \beta_3 x_{ab}^{2/3} + \beta_4 x_{va} x_{ab}^{2/3} + \beta_5 x_{rt} x_{ab}^{2/3} + \epsilon$$

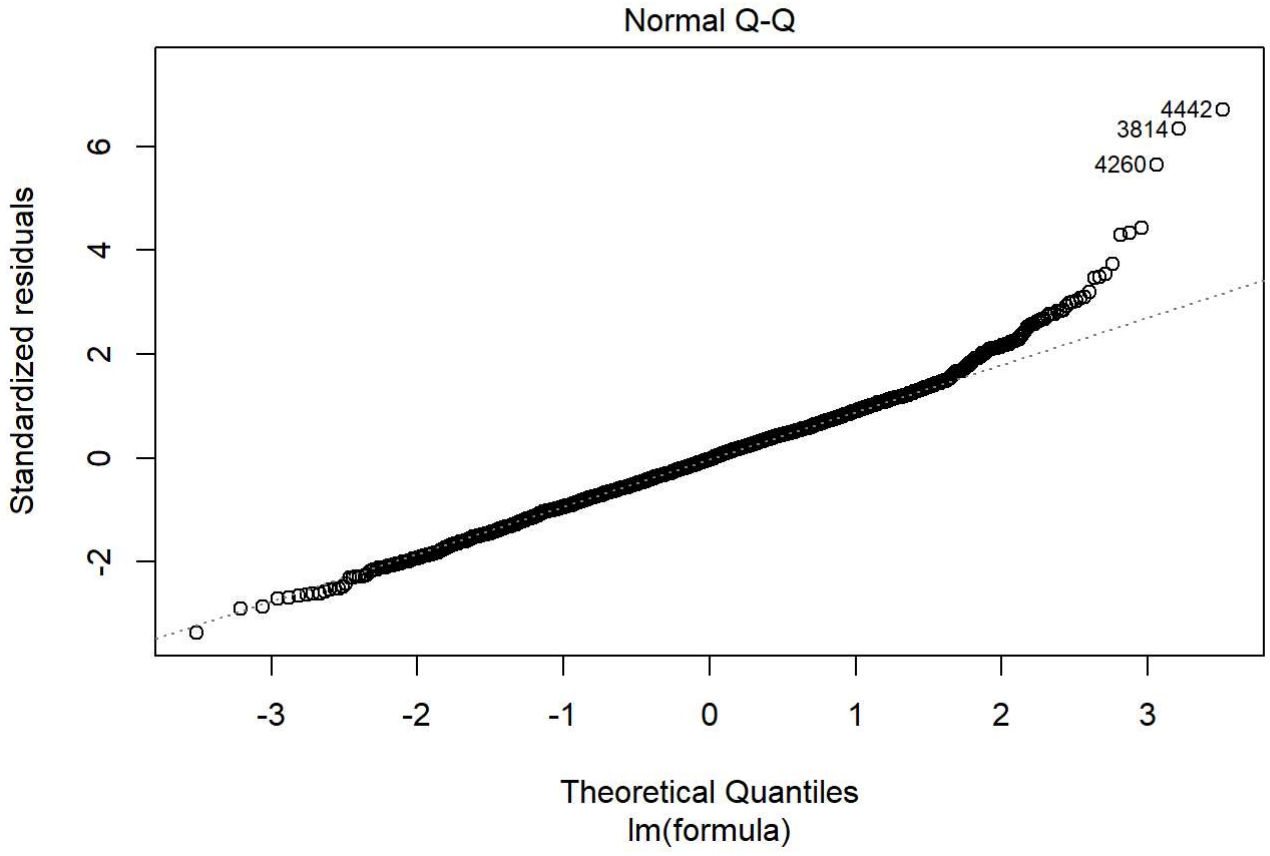
Where va stands for vote_average, rt stands for runtime, and ab stands for adj_bud.

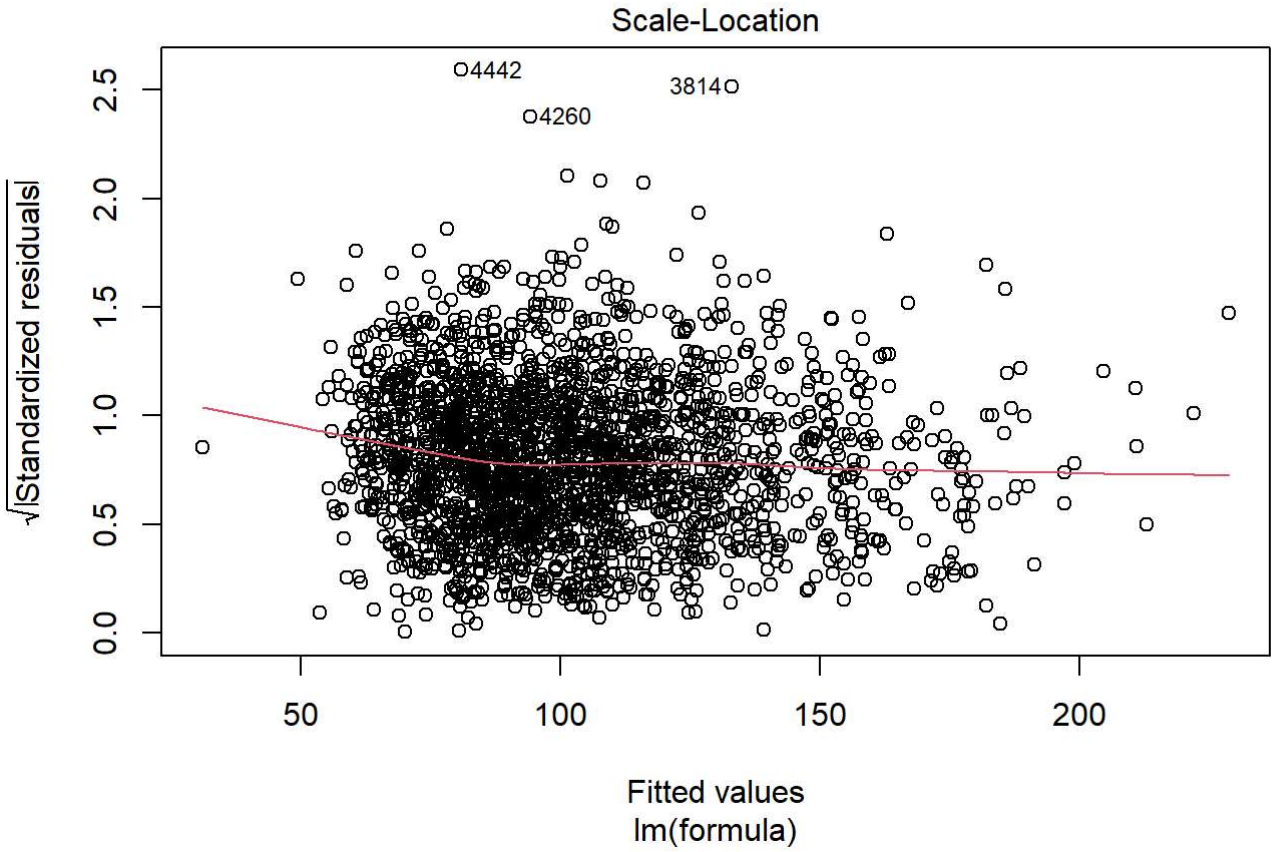
```
#Stating the code for consistency
movies_regression_opt2 = movies_regression


#Taking the fourth root of adj_rev, our response variable
movies_regression_opt2[,3] = movies_regression_opt2[,3]^(1/4)
movies_regression_opt2[,4] = movies_regression_opt2[,4]^(2/3)



opt2_predictor_formula = "runtime + vote_average + adj_bud + vote_average:adj_bud + runtime:adj_
bud"
opt2 = mrm_generator(movies_regression_opt2, "adj_rev", opt2_predictor_formula, F)
```

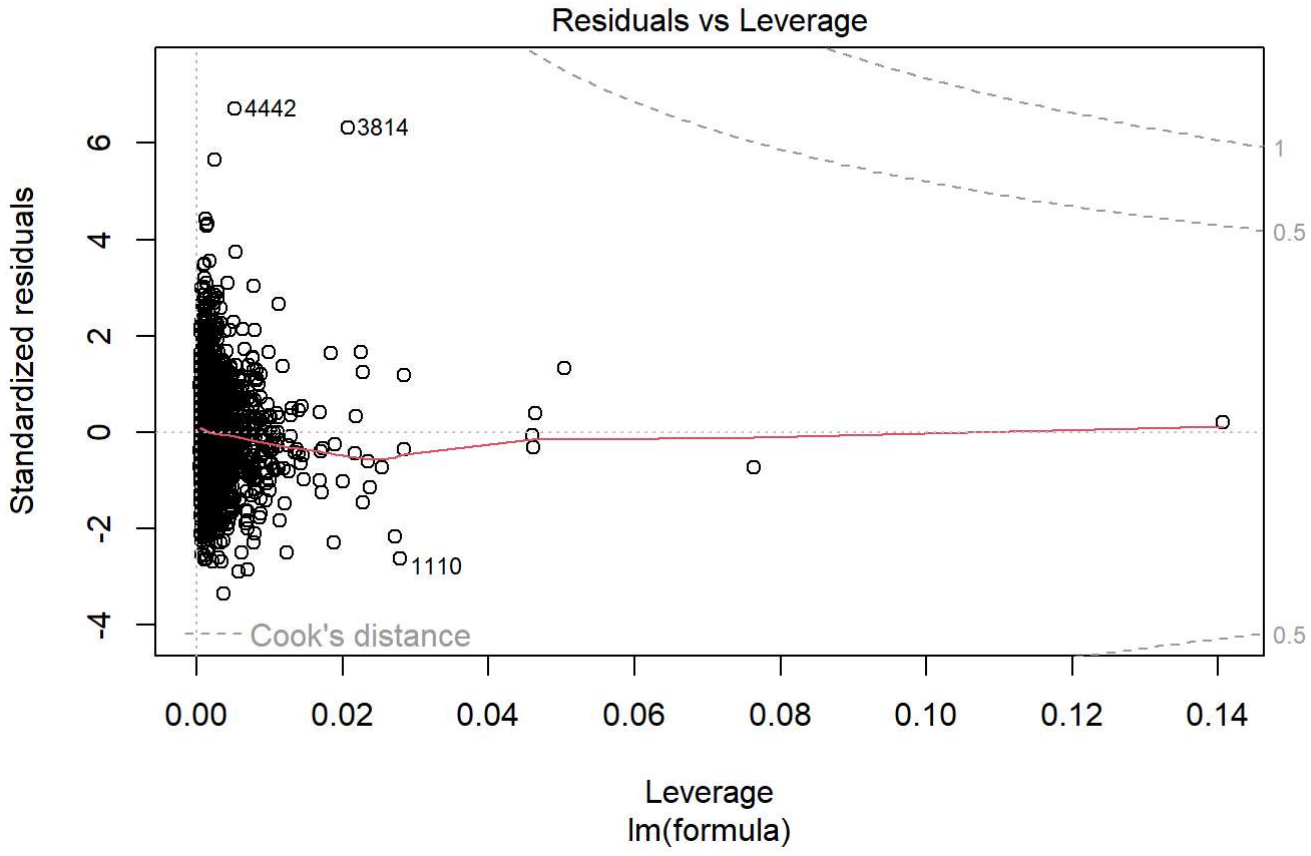
```
## adj_rev ~ runtime + vote_average + adj_bud + vote_average:adj_bud +
##     runtime:adj_bud
## <environment: 0x000001d29bf323e0>
```

```
plot(opt2$model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(formula)
4442
3814
4260
200
150
100
50
0
-50
-100
50
100
150
200


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(formula)
4442
3814
4260
6
4
2
0
-2
-3
-2
-1
0
1
2
3


Scale-Location
4442
3814
4260
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
50
100
150
200
Fitted values
lm(formula)


Residuals vs Leverage
4442
3814
1110
Cook's distance
1
0.5
0.5
Standardized residuals
-4
-2
0
2
4
6
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Leverage
lm(formula)

```
#We observe from the plots of opt2 that there are outliers, similar to opt1.
#Let us remove the outliers in our response and predictor to try to create a better model
opt2_no_outlier = mrm_generator(movies_regression_opt2, "adj_rev", opt2_predictor_formula, T)
```

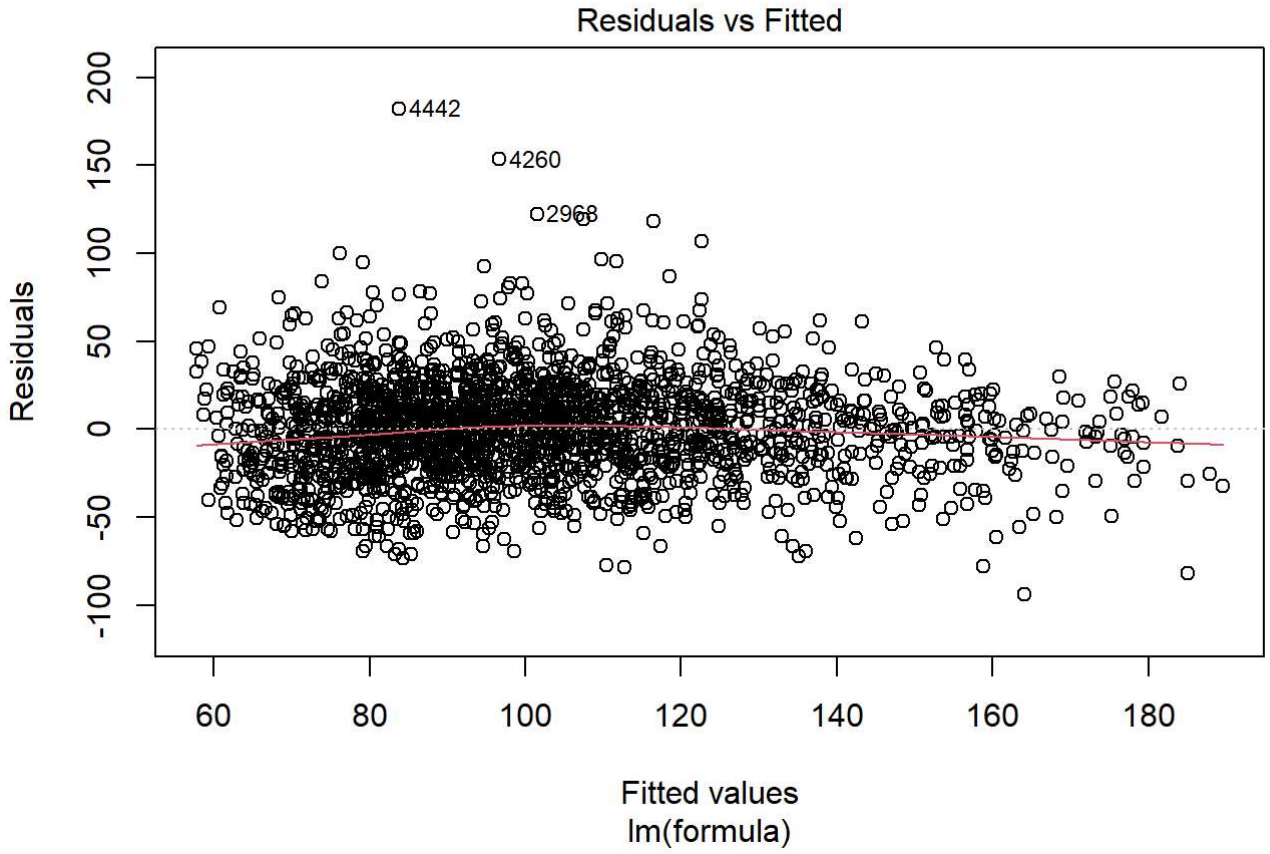
```
## adj_rev ~ runtime + vote_average + adj_bud + vote_average:adj_bud +
##     runtime:adj_bud
## <environment: 0x000001d29c467ea8>
```

```
model_comparison(opt2); model_comparison(opt2_no_outlier)
```

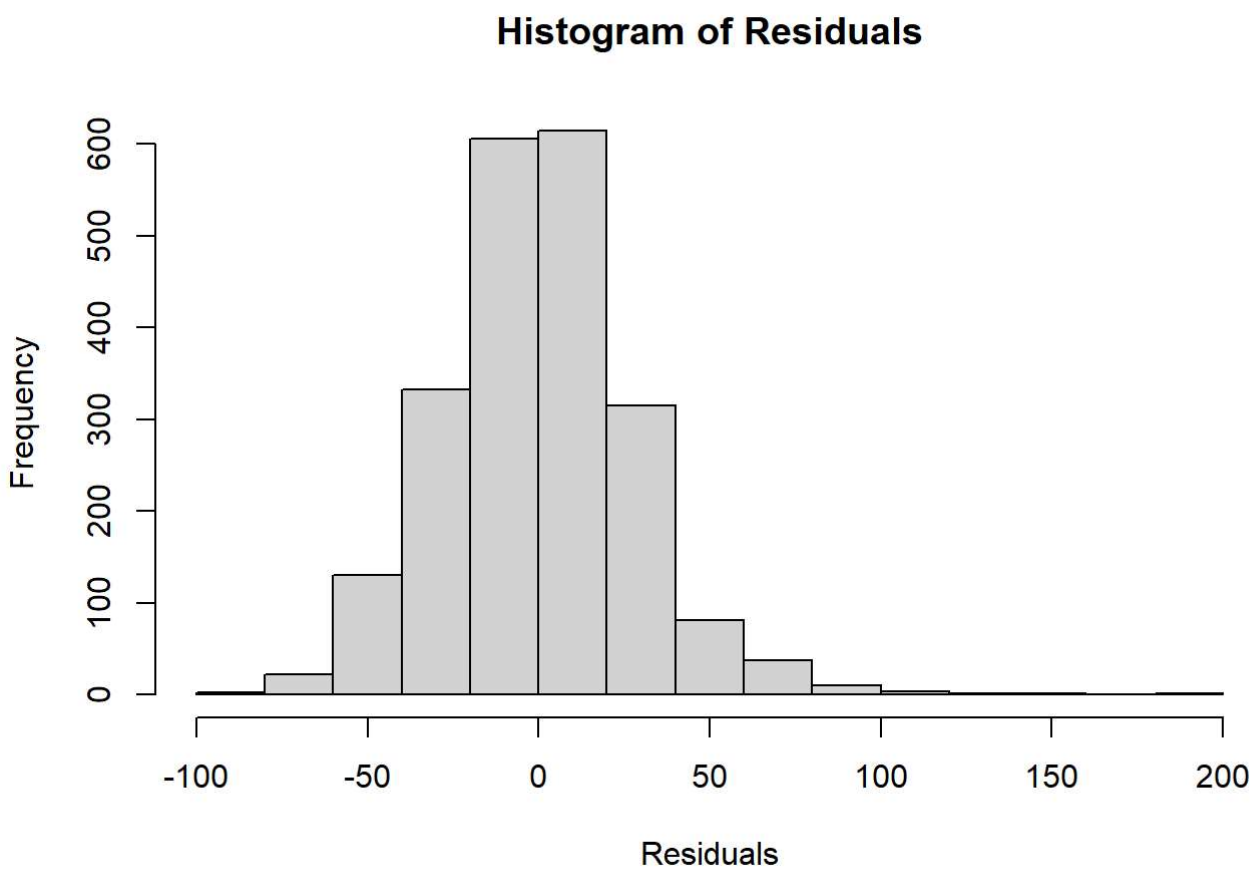
```
##
## Call:
## lm(formula = formula, data = t_t_data$training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -92.910 -17.809  -0.759  16.073 185.061
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.394e+01  8.662e+00   1.609 0.107675
## runtime               1.620e-01  5.917e-02   2.739 0.006218 **
## vote_average          5.231e+00  1.256e+00   4.165 3.23e-05 ***
## adj_bud               7.713e-05  5.037e-05   1.531 0.125799
## vote_average:adj_bud  4.413e-05  7.535e-06   5.856 5.43e-09 ***
## runtime:adj_bud      -8.979e-07  2.643e-07  -3.397 0.000692 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.64 on 2241 degrees of freedom
## Multiple R-squared:  0.4931, Adjusted R-squared:  0.492
## F-statistic:   436 on 5 and 2241 DF,  p-value: < 2.2e-16
##
## [1] 30.45633
## [1] 0.4931221
```

```
##
## Call:
## lm(formula = formula, data = t_t_data$training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -94.152 -17.818  -0.506  16.393 182.061
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.632e+01  1.000e+01   2.631  0.00858 **
## runtime               9.711e-02  7.218e-02   1.345  0.17864
## vote_average          4.376e+00  1.482e+00   2.953  0.00318 **
## adj_bud               2.261e-05  6.153e-05   0.367  0.71337
## vote_average:adj_bud  5.590e-05  9.900e-06   5.647 1.85e-08 ***
## runtime:adj_bud      -1.043e-06  3.886e-07  -2.683  0.00735 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.78 on 2150 degrees of freedom
## Multiple R-squared:  0.4479, Adjusted R-squared:  0.4466
## F-statistic: 348.8 on 5 and 2150 DF,  p-value: < 2.2e-16
##
## [1] 26.02317
## [1] 0.4478995
```

```
#We see that including adj_bud has significantly improved our model, resulting in a higher r^2 v
alue and a lower RMSE value for when outliers are both removed and retained.

#Because the model which excludes outliers has a lower RMSE value, we will use it as our final m
odel.
#But we must check that it satisfies all the assumptions of regression to ensure that we can saf
ely use it.



#Getting only the residual plot
plot(opt2_no_outlier$model, which = 1)
```




```
#Examining the histogram for normality of the residuals
hist(residuals(opt2_no_outlier$model), main = "Histogram of Residuals", xlab = "Residuals")
```




```
#The plot of the residuals and the histogram of residuals shows that all the assumptions of regr
ession are satisfied. The variance is constant, and the residuals are independent and normal.
```

We observed that the p-value of adj_bud was very high, at 0.7213. The code below will determine if removing adj_bud as a predictor will improve our model.

```
opt2_predictor_formula = "runtime + vote_average + vote_average:adj_bud + runtime:adj_bud"
opt2_no_outlier_no_adj_bud = mrm_generator(movies_regression_opt2, "adj_rev", opt2_predictor_for
mula, T)
```

```
## adj_rev ~ runtime + vote_average + vote_average:adj_bud + runtime:adj_bud
## <environment: 0x000001d2973370a8>
```

```
model_comparison(opt2_no_outlier_no_adj_bud)
```

```
##
## Call:
## lm(formula = formula, data = t_t_data$training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -97.066 -17.365  -0.493  16.305 182.483
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.640e+01  4.924e+00   5.361 9.15e-08 ***
## runtime               1.048e-01  6.777e-02   1.547 0.121967
## vote_average          4.077e+00  1.132e+00   3.601 0.000324 ***
## vote_average:adj_bud  6.246e-05  6.656e-06   9.383  < 2e-16 ***
## runtime:adj_bud      -1.134e-06  3.715e-07  -3.051 0.002305 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.06 on 2158 degrees of freedom
## Multiple R-squared:  0.469,  Adjusted R-squared:  0.468
## F-statistic: 476.6 on 4 and 2158 DF,  p-value: < 2.2e-16
##
## [1] 27.7431
## [1] 0.469032
```

```
#We see that the RMSE and r squared values both increased. But because the increase in r squared is so low and the RMSE value increased, we will keep adj_bud in our final model for better model accuracy.
```

Comments to self: Model formula technically isn't linear. Revisit the wording later.

#Might have to account for the dataset potentially holding movies that performed really well in the past, which may skew results

Could compare if movies released in the winter months do better than the summer