

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")  
ncol(movies)
```

```
## [1] 20
```

```
#Note that the budget and revenue columns are in United States dollars
```

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"
```

```
movies[1:2, ]
```

```

##      budget
## 1 237000000
## 2 300000000
##
genres
## 1 [{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]
## 2                               [{"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 28, "name": "Action"}]
##
homepage      id
## 1           http://www.avatarmovie.com/ 19995
## 2 http://disney.go.com/disneypictures/pirates/ 285
##
keywords
## 1 [{"id": 1463, "name": "culture clash"}, {"id": 2964, "name": "future"}, {"id": 3386, "name": "space war"}, {"id": 3388, "name": "space colony"}, {"id": 3679, "name": "society"}, {"id": 3801, "name": "space travel"}, {"id": 9685, "name": "futuristic"}, {"id": 9840, "name": "romance"}, {"id": 9882, "name": "space"}, {"id": 9951, "name": "alien"}, {"id": 10148, "name": "tribe"}, {"id": 10158, "name": "alien planet"}, {"id": 10987, "name": "cgi"}, {"id": 11399, "name": "marine"}, {"id": 13065, "name": "soldier"}, {"id": 14643, "name": "battle"}, {"id": 14720, "name": "love affair"}, {"id": 165431, "name": "anti war"}, {"id": 193554, "name": "power relations"}, {"id": 206690, "name": "mind and soul"}, {"id": 209714, "name": "3d"}]
## 2
[{"id": 270, "name": "ocean"}, {"id": 726, "name": "drug abuse"}, {"id": 911, "name": "exotic island"}, {"id": 1319, "name": "east india trading company"}, {"id": 2038, "name": "love of one's life"}, {"id": 2052, "name": "traitor"}, {"id": 2580, "name": "shipwreck"}, {"id": 2660, "name": "strong woman"}, {"id": 3799, "name": "ship"}, {"id": 5740, "name": "alliance"}, {"id": 5941, "name": "calypso"}, {"id": 6155, "name": "afterlife"}, {"id": 6211, "name": "fighter"}, {"id": 12988, "name": "pirate"}, {"id": 157186, "name": "swashbuckler"}, {"id": 179430, "name": "aftercreditsstinger"}]
##
original_language          original_title
## 1             en                      Avatar
## 2             en Pirates of the Caribbean: At World's End
##
overview
## 1 In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization.
## 2 Captain Barbosa, long believed to be dead, has come back to life and is headed to the edge of the Earth with Will Turner and Elizabeth Swann. But nothing is quite as it seems.
##
popularity
## 1 150.4376
## 2 139.0826
##
production_companies
## 1 [{"name": "Ingenious Film Partners", "id": 289}, {"name": "Twentieth Century Fox Film Corporation", "id": 306}, {"name": "Dune Entertainment", "id": 444}, {"name": "Lightstorm Entertainment", "id": 574}]
## 2                               [{"name": "Walt Disney Pictures", "id": 2}, {"name": "Jerry Bruckheimer Films", "id": 130}, {"name": "Second Mate Productions", "id": 19936}]
##
production_countries

```

```

## 1 [{"iso_3166_1": "US", "name": "United States of America"}, {"iso_3166_1": "GB", "name": "United Kingdom"}]
## 2 [{"iso_3166_1": "US", "name": "United States of America"}]
## release_date revenue runtime
## 1 2009-12-10 2787965087 162
## 2 2007-05-19 9610000000 169
## spoken_languages
## 1 [{"iso_639_1": "en", "name": "English"}, {"iso_639_1": "es", "name": "Espa\u00f1ol"}]
## 2 [{"iso_639_1": "en", "name": "English"}]
## status tagline
## 1 Released Enter the World of Pandora.
## 2 Released At the end of the world, the adventure begins.
## title vote_average vote_count
## 1 Avatar 7.2 11800
## 2 Pirates of the Caribbean: At World's End 6.9 4500

```

#Based on the column names as well as the structure of the columns' data, we decide to remove some 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

```
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 nu
  #mber 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 are vary significantly in topic and dsecription.
movies = subset(movies, select = -keywords)
```

```
#Tells us whether the movie was originally in English or not. It is hypothesized that English sp
#eaking 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)

```

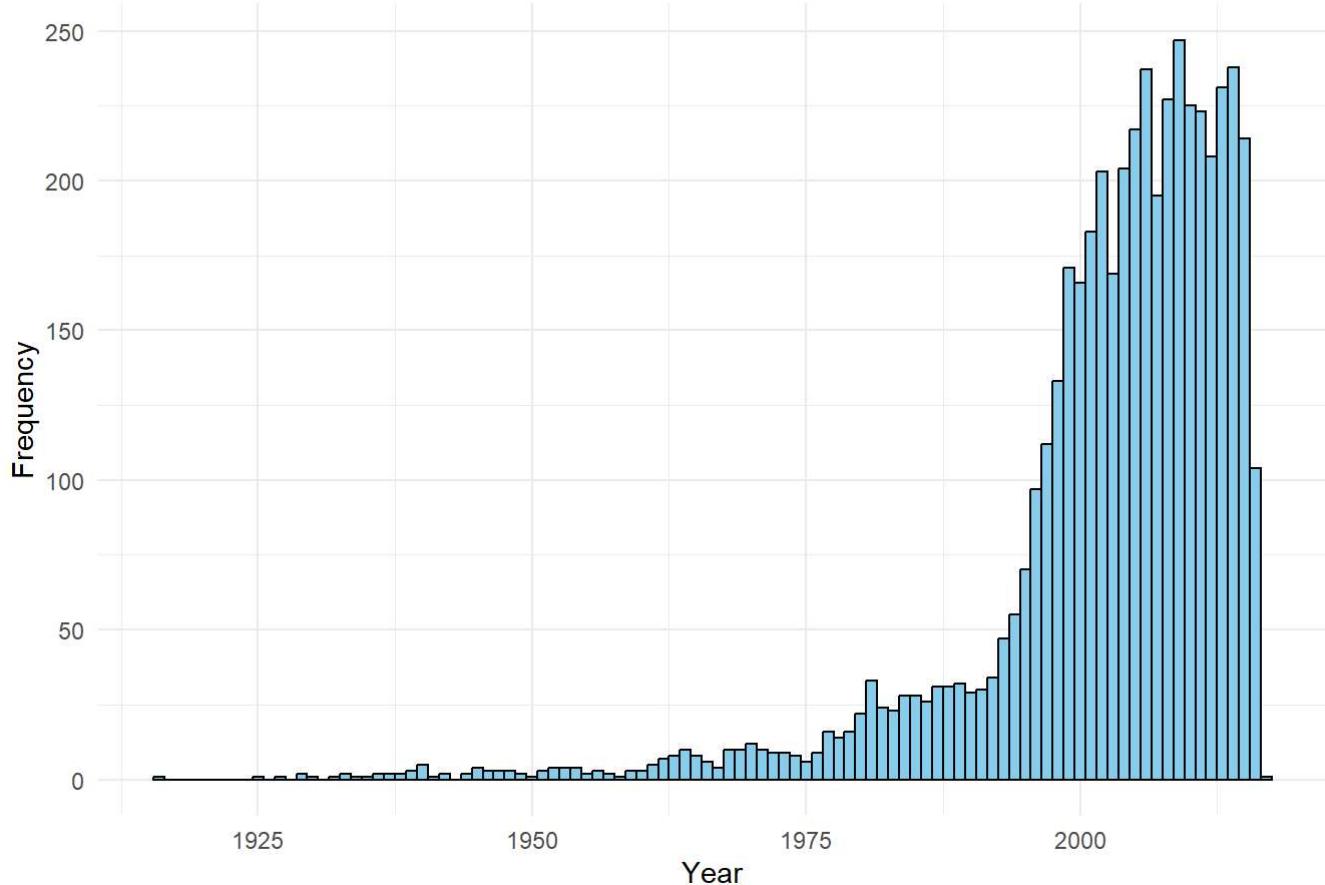
```
##  
## Attaching package: 'lubridate'
```

```
## 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()`).
```

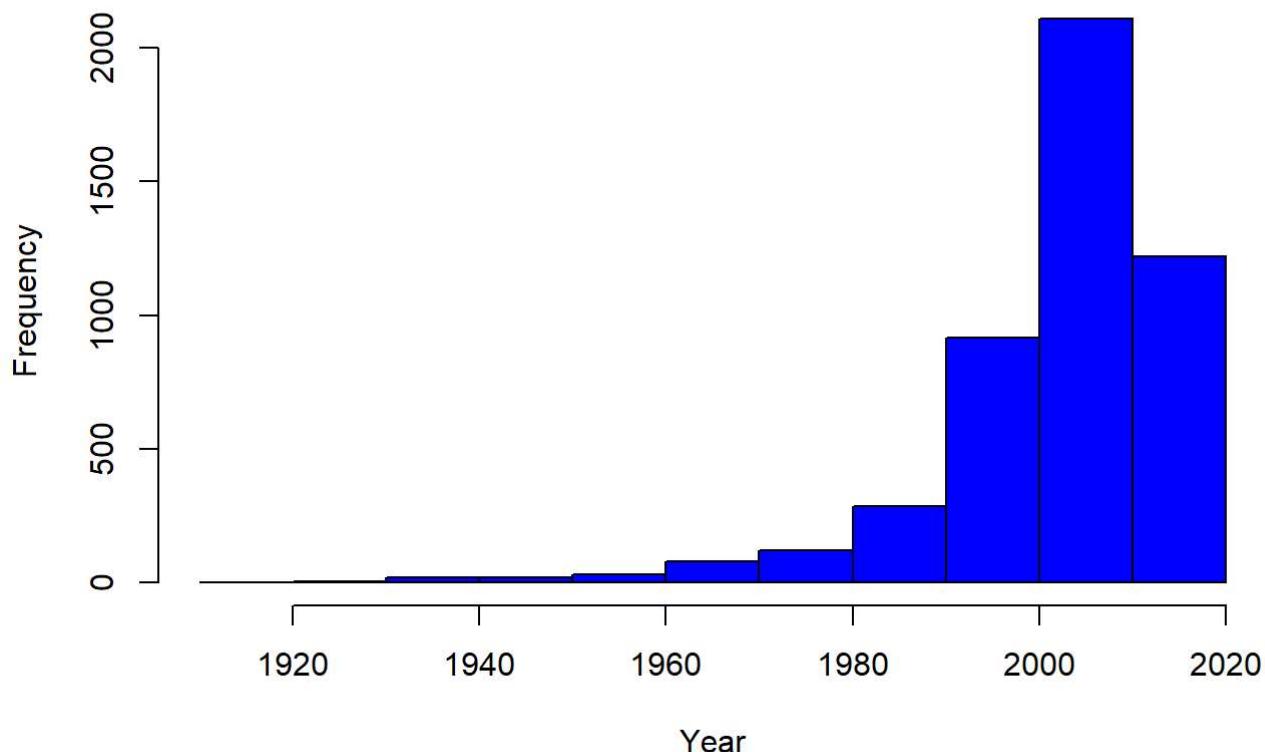
Distribution of Movies by Release Year



Adjusting data for inflation

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

Histogram of Movies by Release Year



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

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

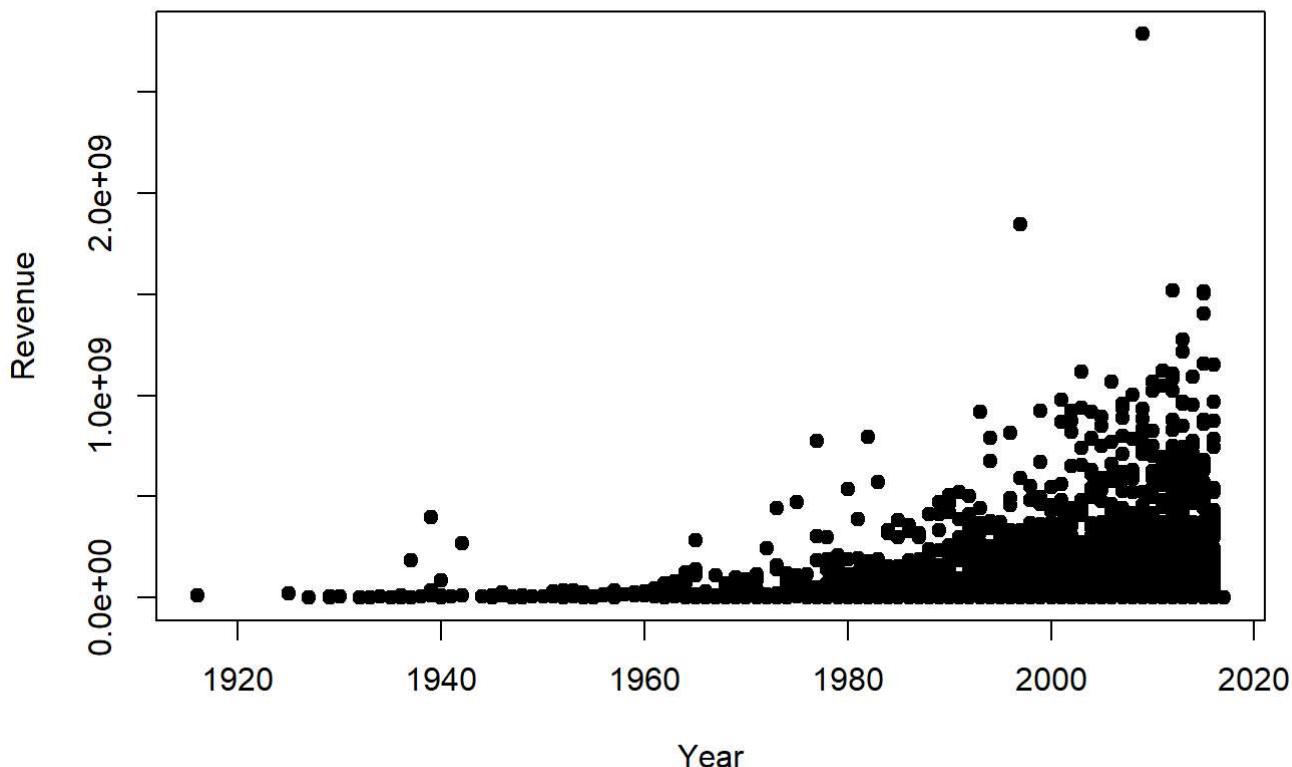
```
movies[movies$year == max(movies$year, na.rm = TRUE), ][,15]
```

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

#From the histogram and the lines of code above, we see that the dataset includes movies from 1916 to 2017.

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

Plot of Movie Revenue and Release Year



```
#We also see from the plot above that the most profitable movie according to the dataset was Avatar, 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 time, when adjusted for inflation, and it is in our dataset.
```

```
movies[movies$title=="Gone with the Wind",][,c(6,9)]
```

```
##      revenue    title
## 1 3814 400176459 Gone with the Wind
```

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

#CPI dataset copy and pasted from: <https://www.usinflationcalculator.com/inflation/consumer-price-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[1:10,]
```

```
##   Year Jan Feb Mar Apr May June July Aug Sep Oct Nov Dec Avg
## 1 1913 9.8 9.8 9.8 9.8 9.7 9.8 9.9 9.9 10.0 10.0 10.1 10.0 9.9
## 2 1914 10.0 9.9 9.9 9.8 9.9 9.9 10.0 10.2 10.2 10.2 10.1 10.2 10.1 10.0
## 3 1915 10.1 10.0 9.9 10.0 10.1 10.1 10.1 10.1 10.1 10.2 10.3 10.3 10.3 10.1
## 4 1916 10.4 10.4 10.5 10.6 10.7 10.8 10.8 10.9 11.1 11.3 11.5 11.6 10.9
## 5 1917 11.7 12.0 12.0 12.6 12.8 13.0 12.8 13.0 13.3 13.5 13.5 13.7 12.8
## 6 1918 14.0 14.1 14.0 14.2 14.5 14.7 15.1 15.4 15.7 16.0 16.3 16.5 15.1
## 7 1919 16.5 16.2 16.4 16.7 16.9 16.9 17.4 17.7 17.8 18.1 18.5 18.9 17.3
## 8 1920 19.3 19.5 19.7 20.3 20.6 20.9 20.8 20.3 20.0 19.9 19.8 19.4 20.0
## 9 1921 19.0 18.4 18.3 18.1 17.7 17.6 17.7 17.7 17.5 17.5 17.4 17.3 17.9
## 10 1922 16.9 16.9 16.7 16.7 16.7 16.7 16.8 16.6 16.6 16.7 16.8 16.9 16.8
##    Dec.Dec Avg.Avg
## 1      -      -
## 2      1.0     1.0
## 3      2.0     1.0
## 4     12.6     7.9
## 5     18.1    17.4
## 6     20.4    18.0
## 7     14.5    14.6
## 8      2.6    15.6
## 9    -10.8   -10.5
## 10   -2.3    -6.1
```

#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, CPI_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 an error occurs
  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]

```

```
#Before we adjust for inflation, we see that there a lot of movies with their revenues as 0, presumably from being not recorded, and observe that some movies possess suspiciously low budgets as well.
```

```
#To rectify this, we will remove movies with revenue lower than $100, since they are likely data entry mistakes or entered as a joke.
```

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

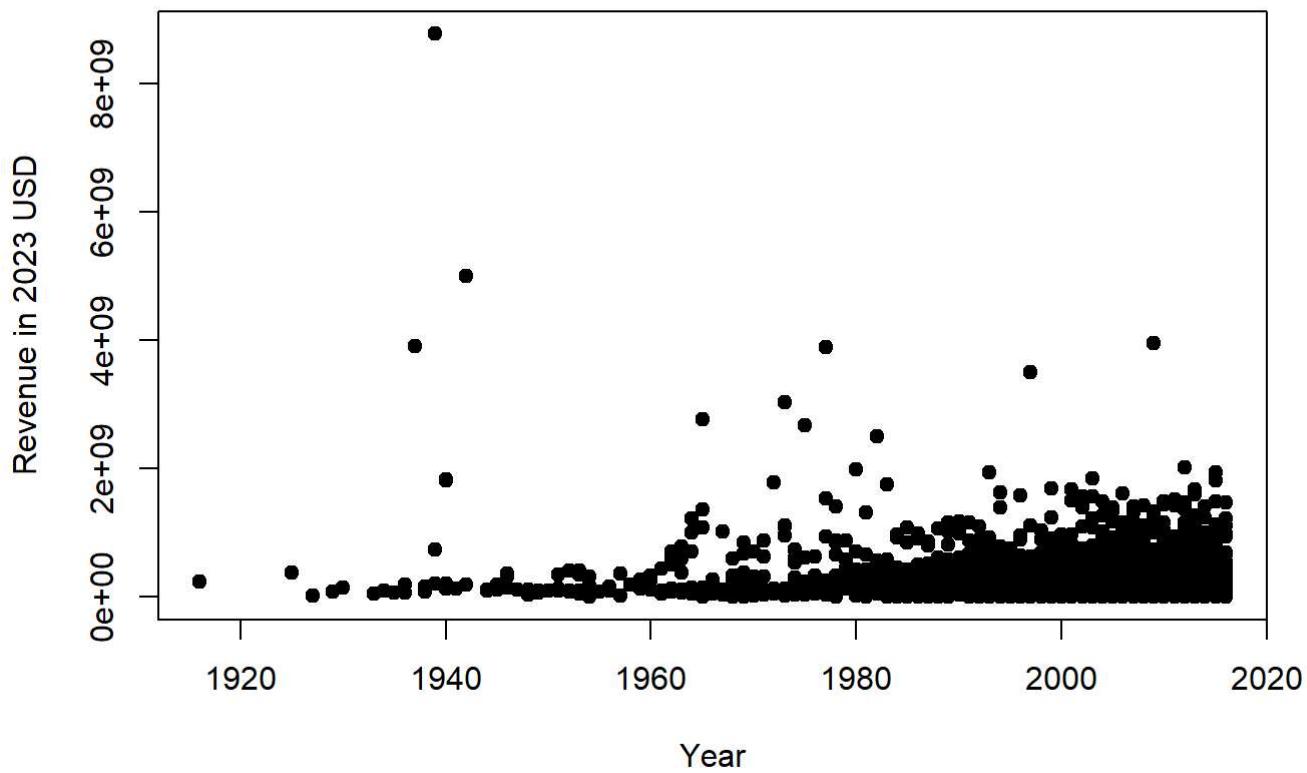
```
#Removing movies with earnings less than $1000
```

```
movies = movies[movies$revenue >= 1000, ]
```

```
movies = adjust_for_inflation(movies, "revenue", "year", CPI, "adjusted_revenue")
```

```
plot(movies$year, movies$adjusted_revenue, pch=19, main = "Plot of Movie Revenue Adjusted for Inflation and Release Year", xlab = "Year", ylab = "Revenue in 2023 USD")
```

Plot of Movie Revenue Adjusted for Inflation and Release Year



```
movies[order(-movies$adjusted_revenue), ][c(1:10),c(9,17)]
```

```

##                               title adjusted_revenue
## 3814      Gone with the Wind      8772271037
## 4442          Bambi        4999489663
## 1            Avatar        3959683122
## 4260 Snow White and the Seven Dwarfs 3912997600
## 2913       Star Wars        3898767715
## 26           Titanic        3502714063
## 2809     The Exorcist        3028532995
## 3237   The Sound of Music        2768573504
## 2810          Jaws        2665598794
## 2968 E.T. the Extra-Terrestrial    2503641778

```

#We see that the adjusted earnings differ from the list found on wikipedia. It's likely that the source used to record these earnings is different than the one on Wikipedia.

Last stretch of cleaning

```

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

```

```

#Removing movies with budgets less than $1000, for the same reason we removed movies with revenues of less than $1000
table(movies$budget)[1:50]

```

```

##
##      0      1     10    218    7000    8000   10000   12000   15000   20000   27000
##     140      1      1      1      1      1      1      1      1      1      1      1
##  31192  60000  65000  85000 100000 114000 125000 130000 134005 150000 160000
##      1      2      2      1      5      1      1      1      1      2      2
## 1800000 2000000 2100000 220000 225000 230000 245000 250000 300000 312000 325000
##      1      4      1      1      1      1      1      5      6      1      1
## 3500000 379000 385907 400000 439000 450000 500000 560000 600000 609000 650000
##      2      1      1      3      1      1     10      1      2      1      1
## 700000 730000 750000 777000 800000 806948
##      5      1      2      1      4      1

```

```

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 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)
```

```
#The data cleaning is now finished!
```

```

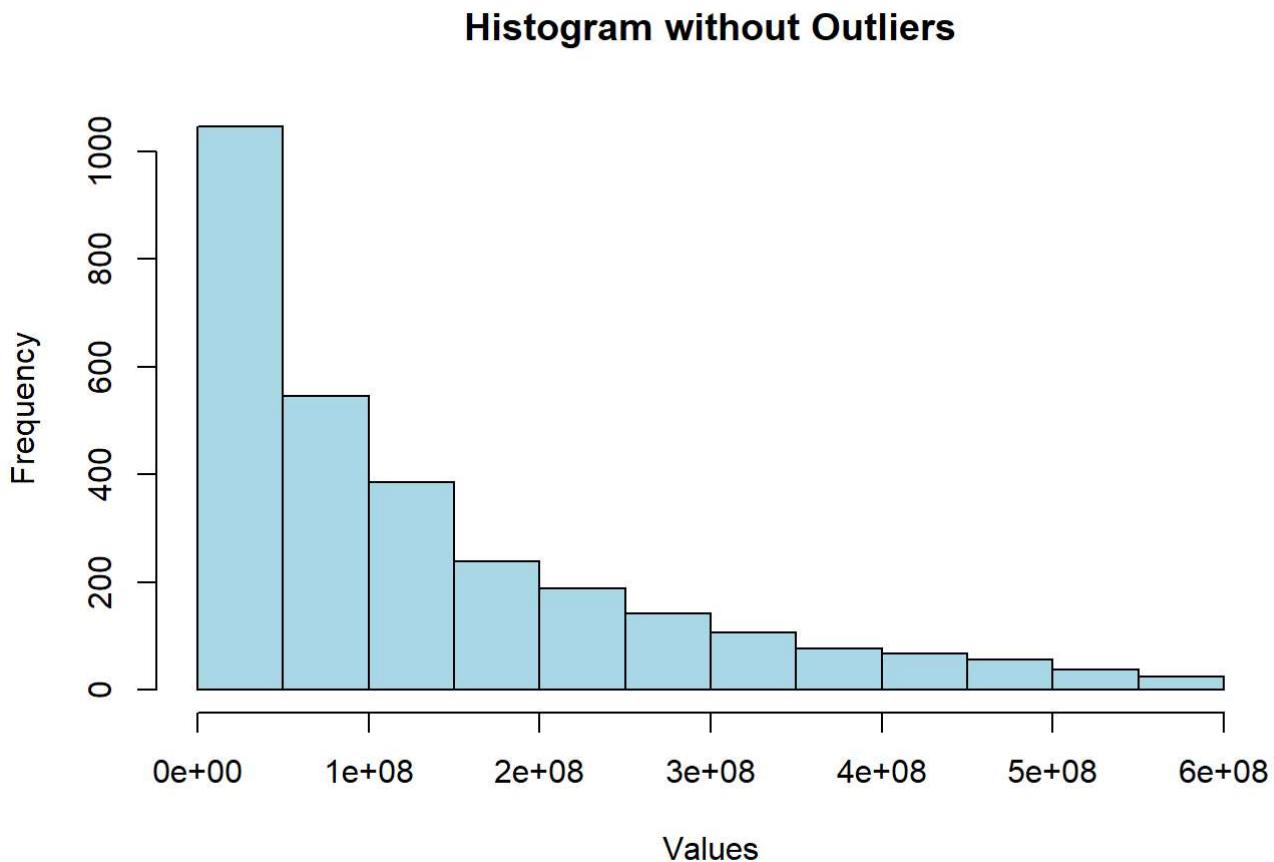
# Identify outliers using IQR
Q1 <- quantile(movies$adjusted_revenue, 0.25)
Q3 <- quantile(movies$adjusted_revenue, 0.75)
IQR_value <- IQR(movies$adjusted_revenue)

# Define Lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR_value
upper_bound <- Q3 + 1.5 * IQR_value

# Remove outliers
data_cleaned <- movies[movies$adjusted_revenue >= lower_bound & movies$adjusted_revenue <= upper_bound,]

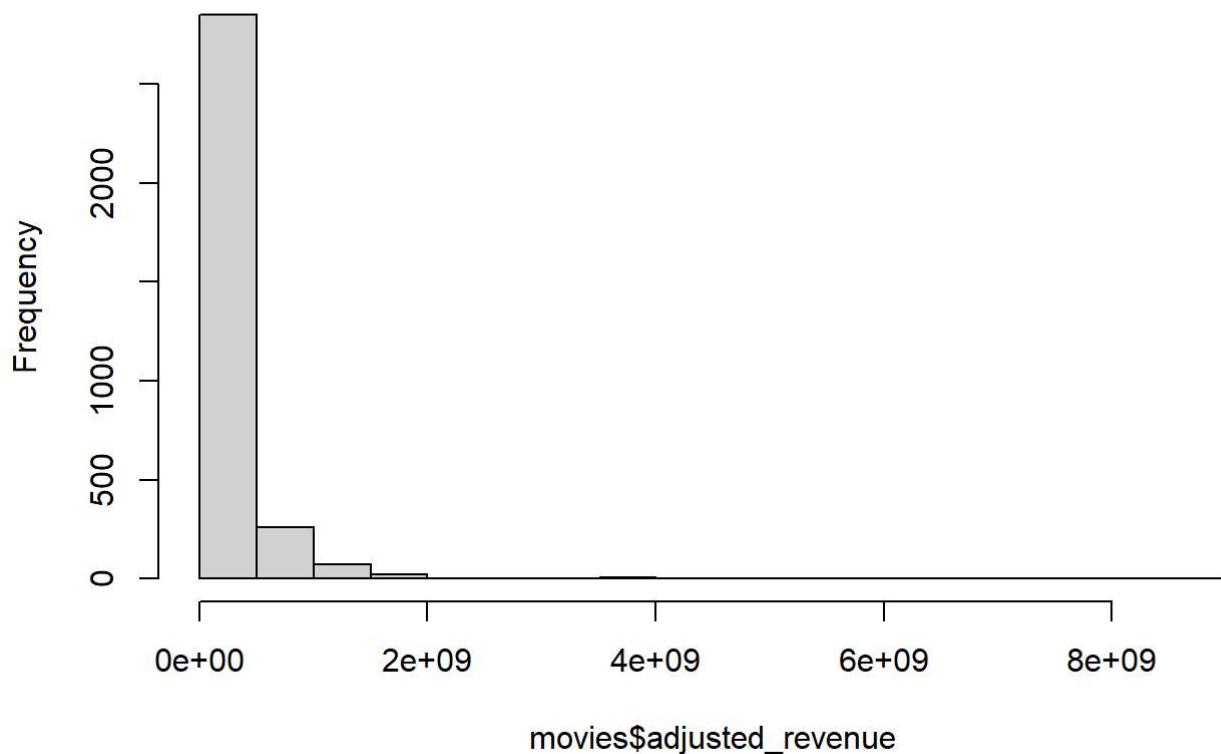
# Create histogram without outliers
hist(data_cleaned$adjusted_revenue, main="Histogram without Outliers", xlab="Values", col="light blue")

```



```
hist(movies$adjusted_revenue)
```

Histogram of movies\$adjusted_revenue



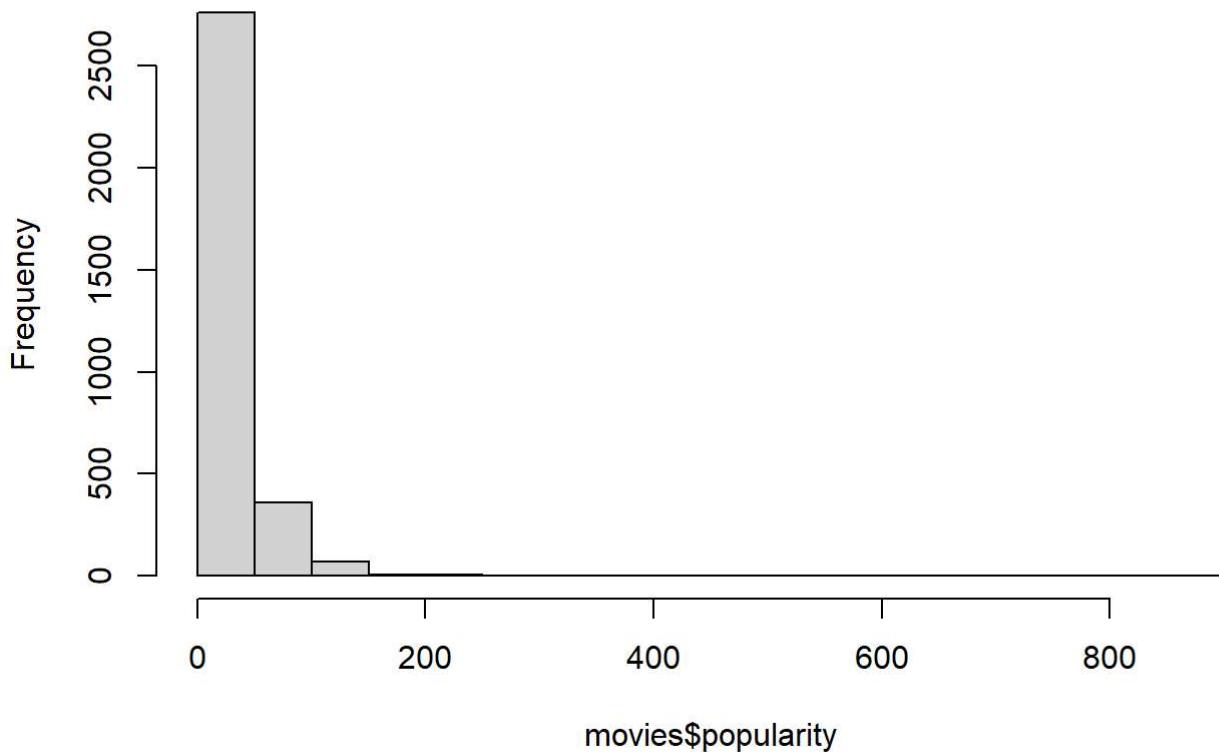
#Might have to account for the dataset potentially holding movies that performed really well in the past, which may skew results

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

Doing multiple linear regression first Want to predict for adjusted revenue

```
hist(movies$popularity)
```

Histogram of 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, because 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 using vote_count either

```
movies = subset(movies, select=-c(popularity, vote_count))
```

Checking 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
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 the 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_n
ame, " 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 = response_var)
abline(linear_model, col="black")

#Residuals Plot
plot(linear_model$fitted.values, linear_model$residuals, xlab="Fitted values", ylab = "Residuals", 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)
}

}

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

```

```

#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)

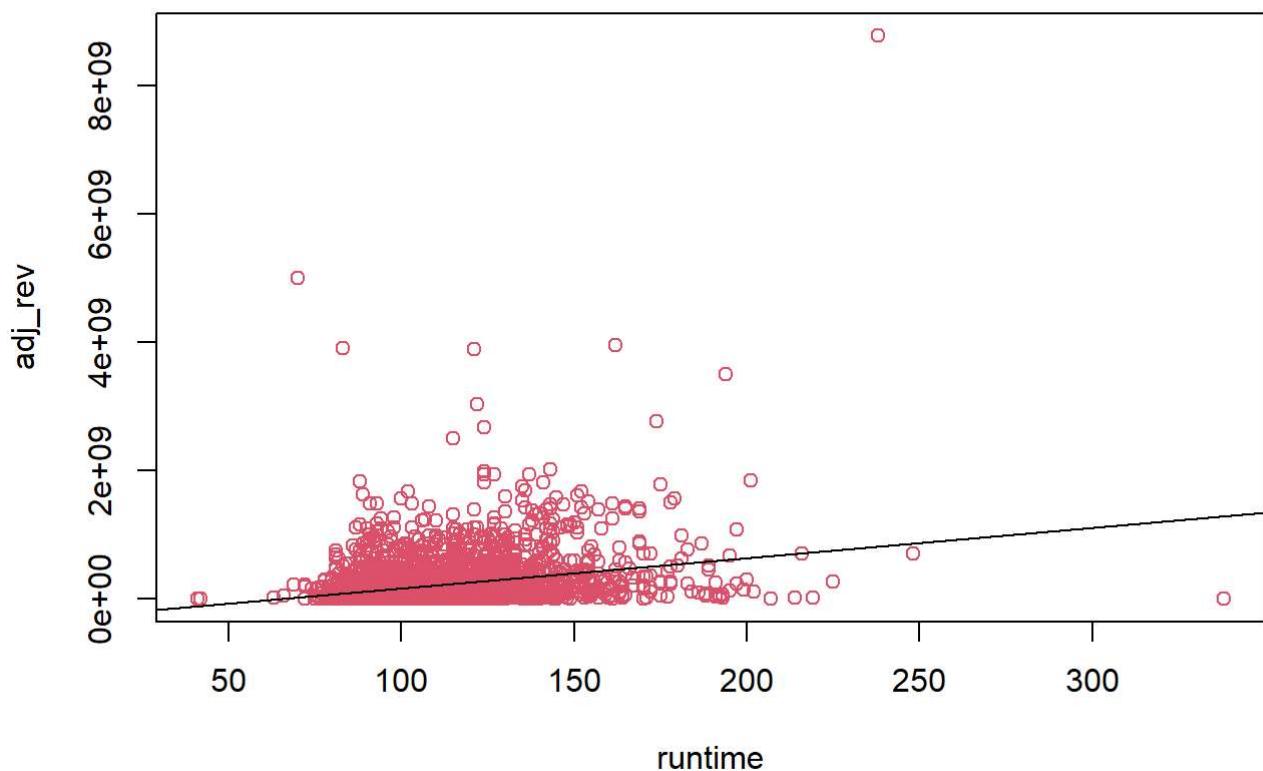
```

```

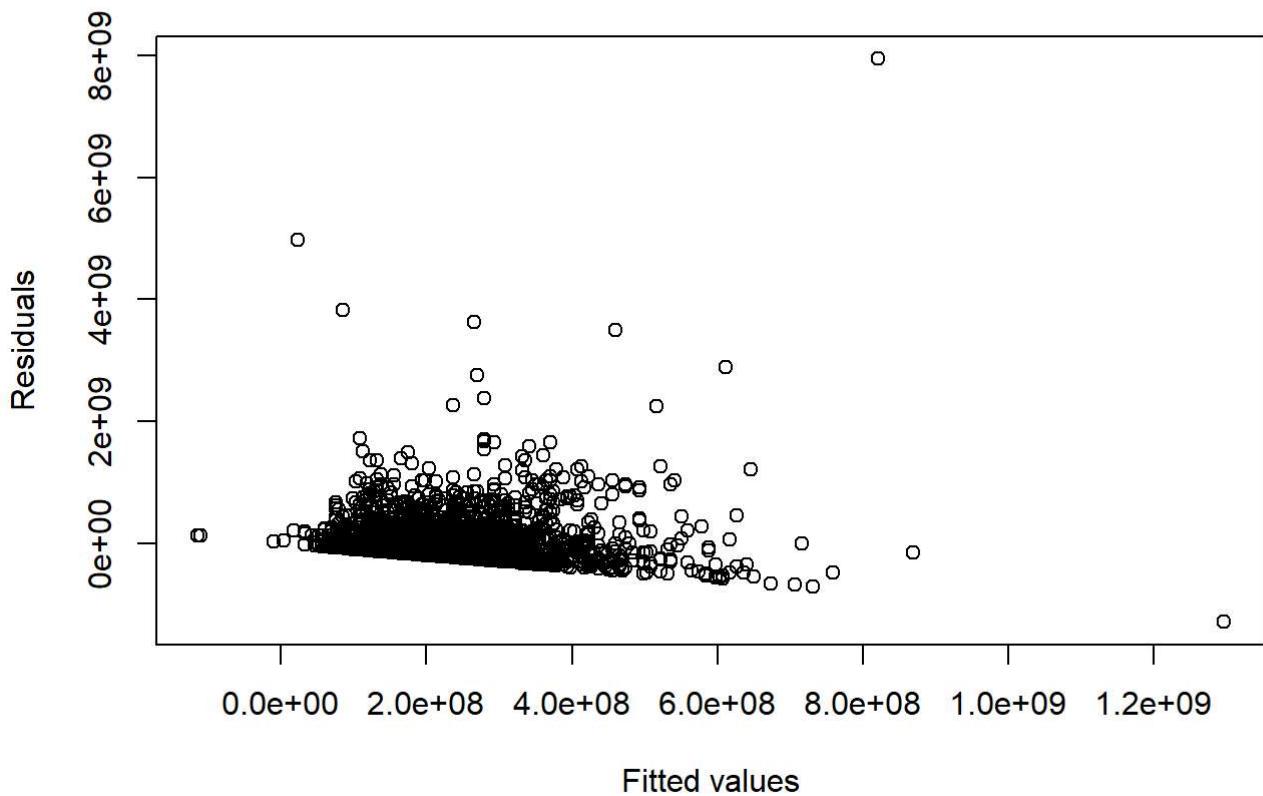
check_regression_assumptions(movies_regression, "adj_rev", F, NA)

```

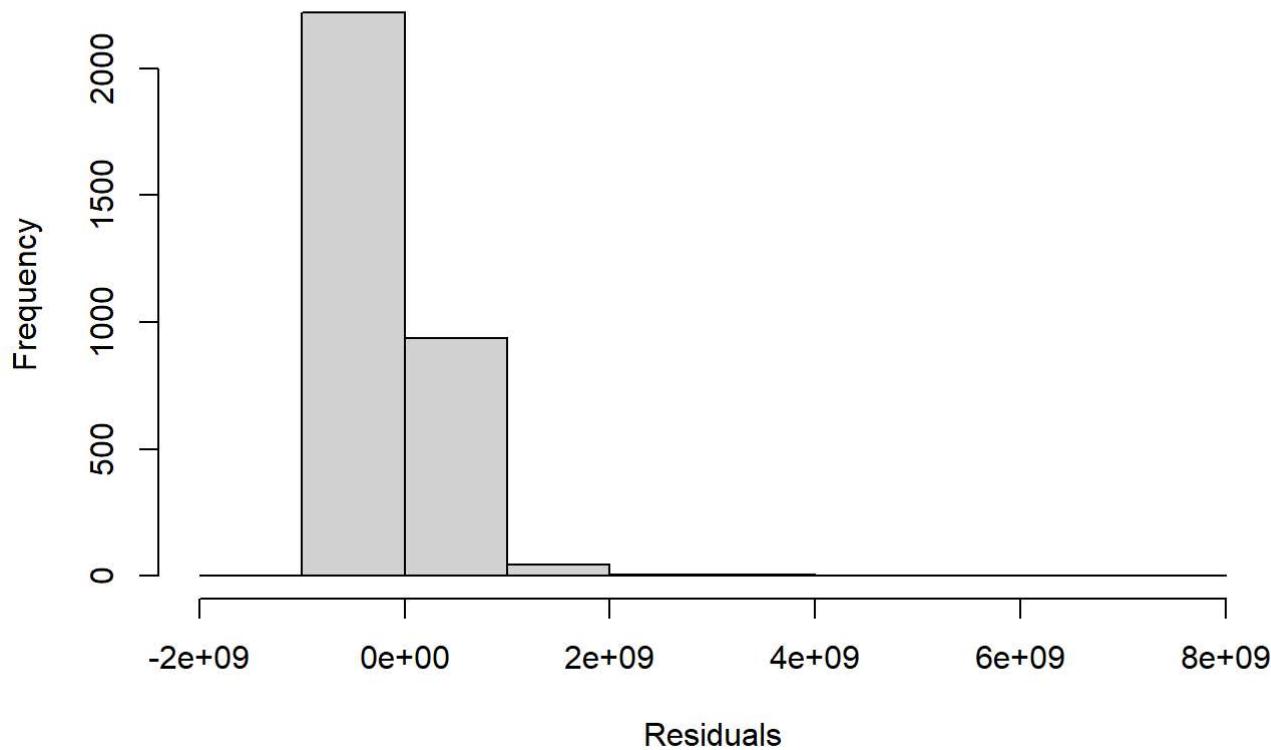
Plot of adj_rev vs runtime With Outliers



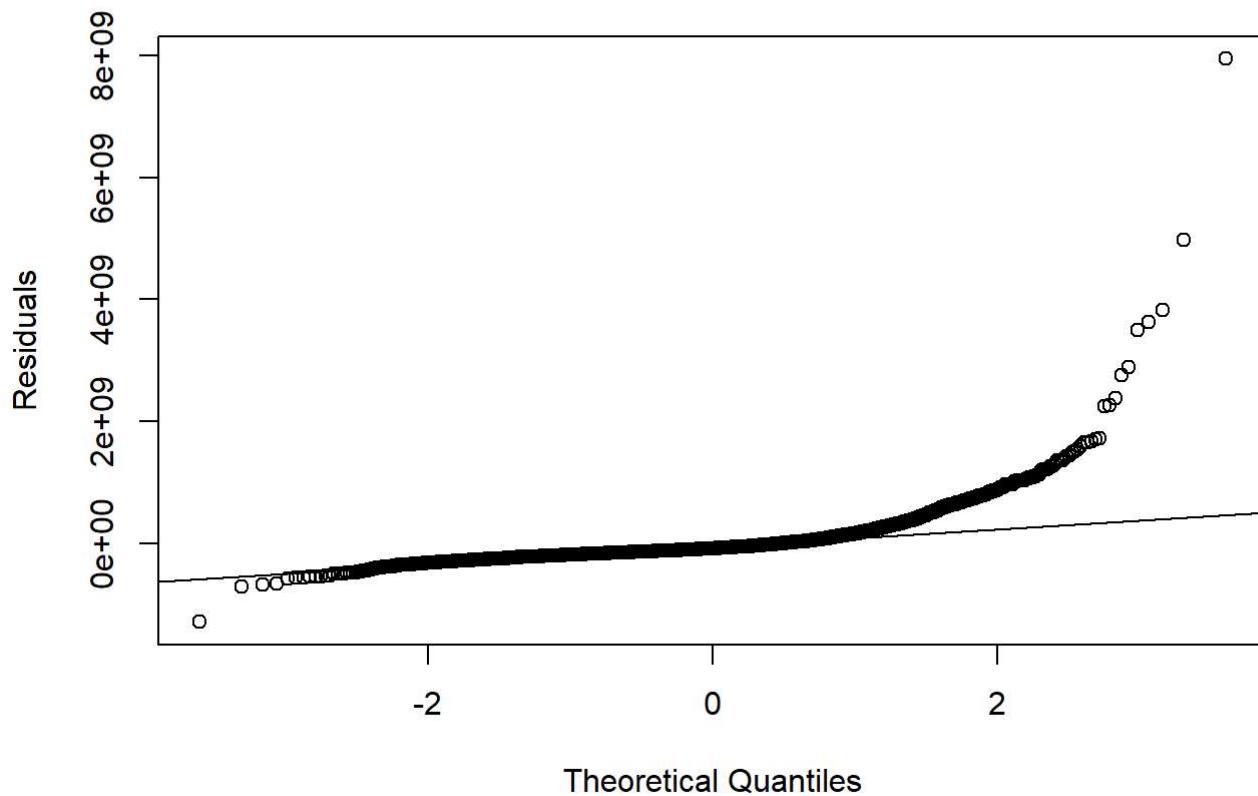
Residuals vs Fitted Values for adj_rev vs runtime With Outliers



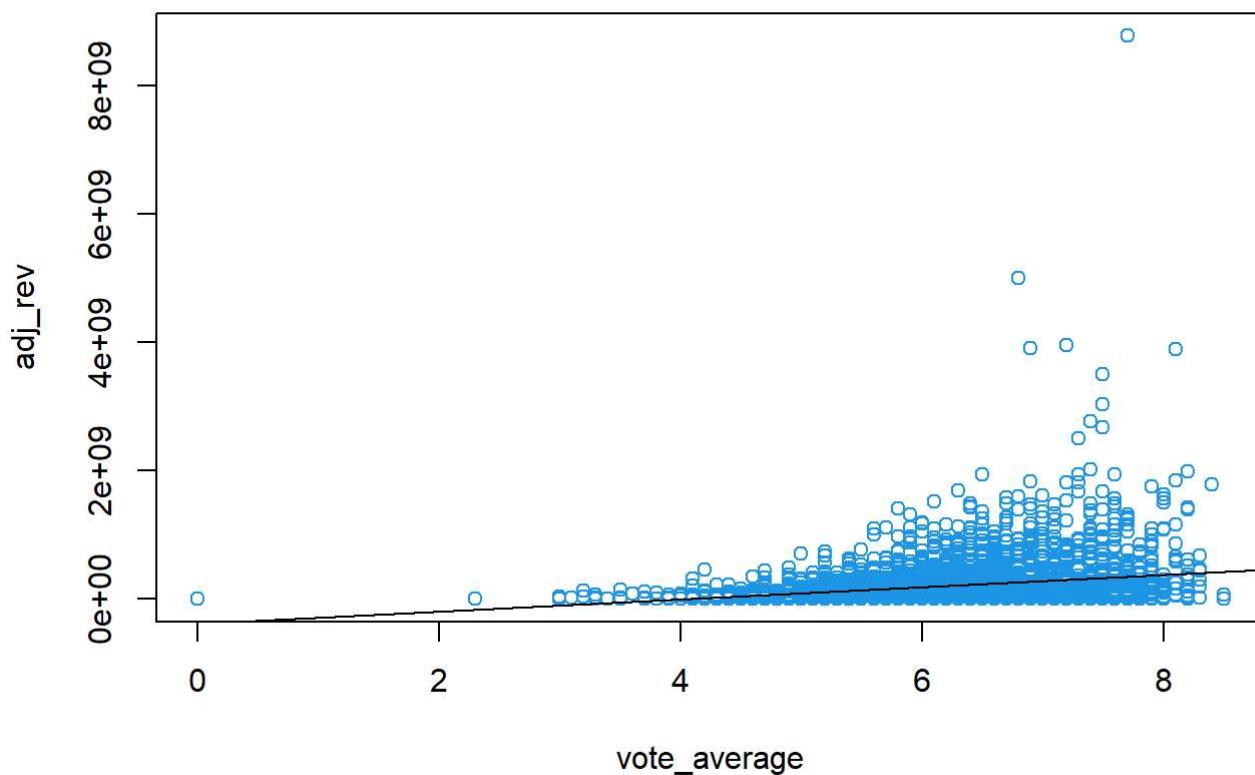
Histogram of the Residuals for adj_rev vs runtime With Outliers



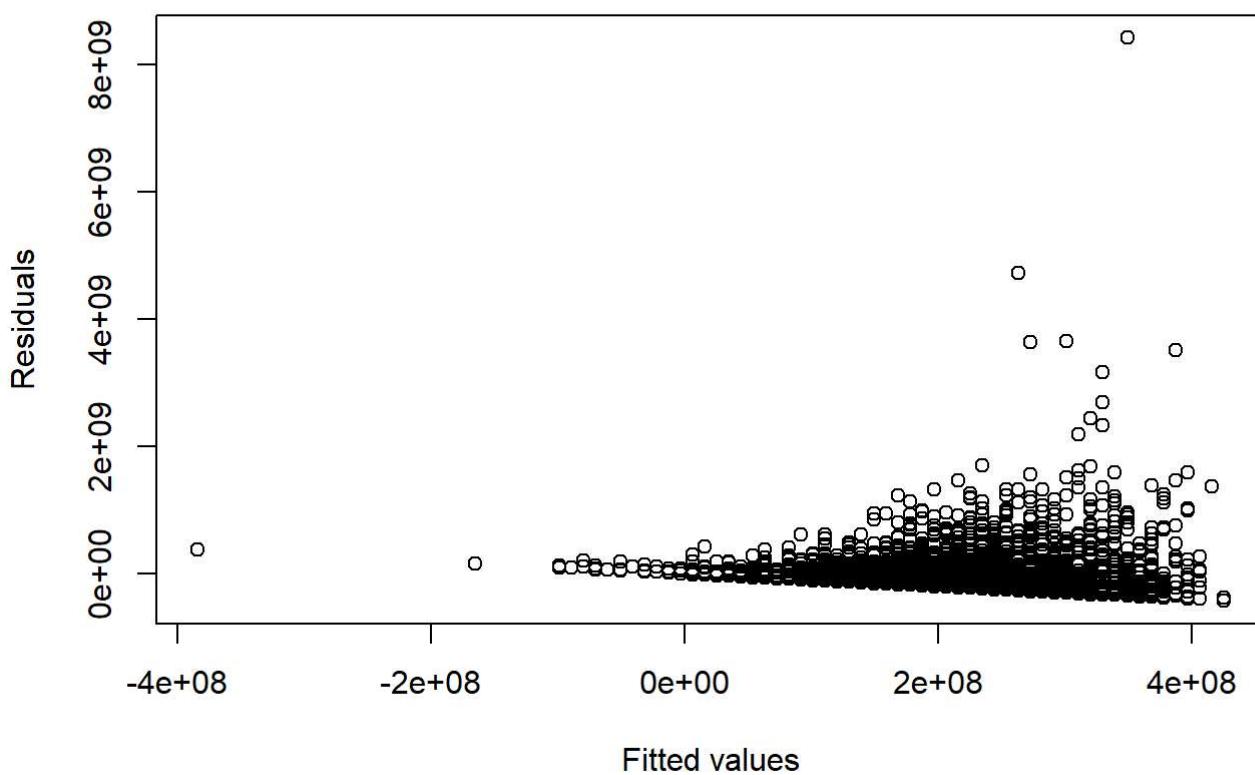
Normal Q-Q plot of the Residuals for adj_rev vs runtime With Outliers



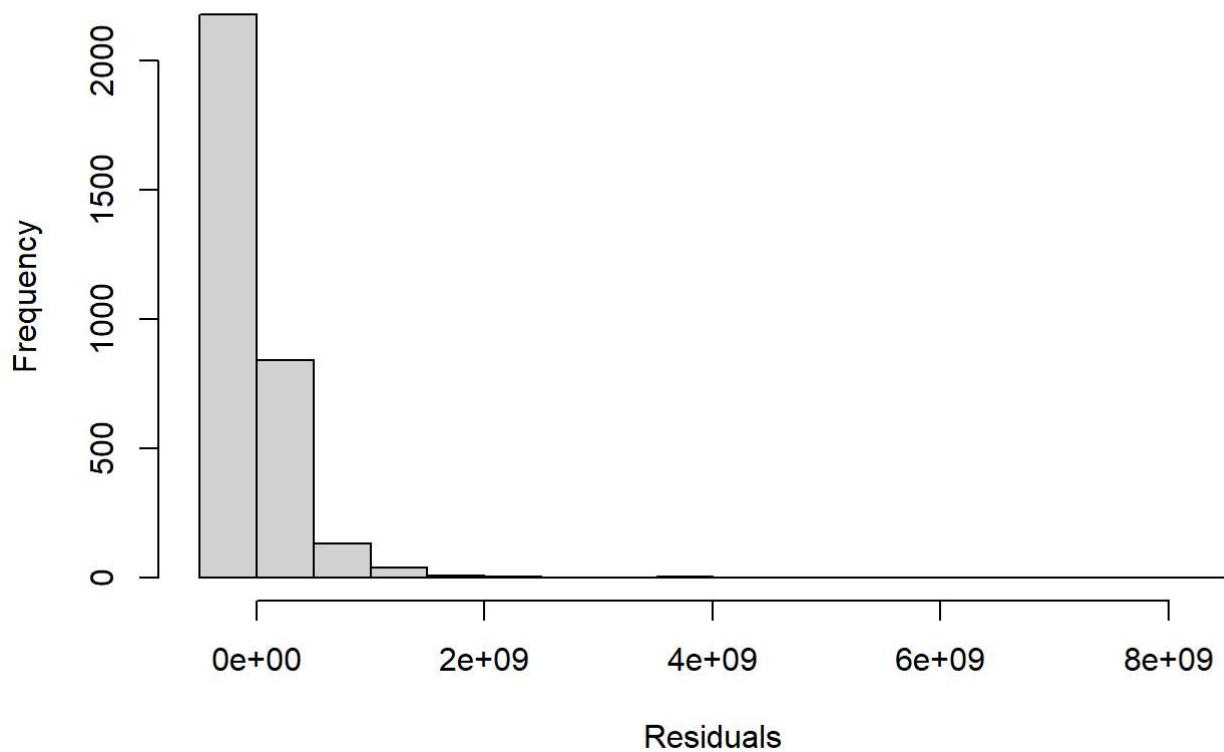
Plot of adj_rev vs vote_average With Outliers



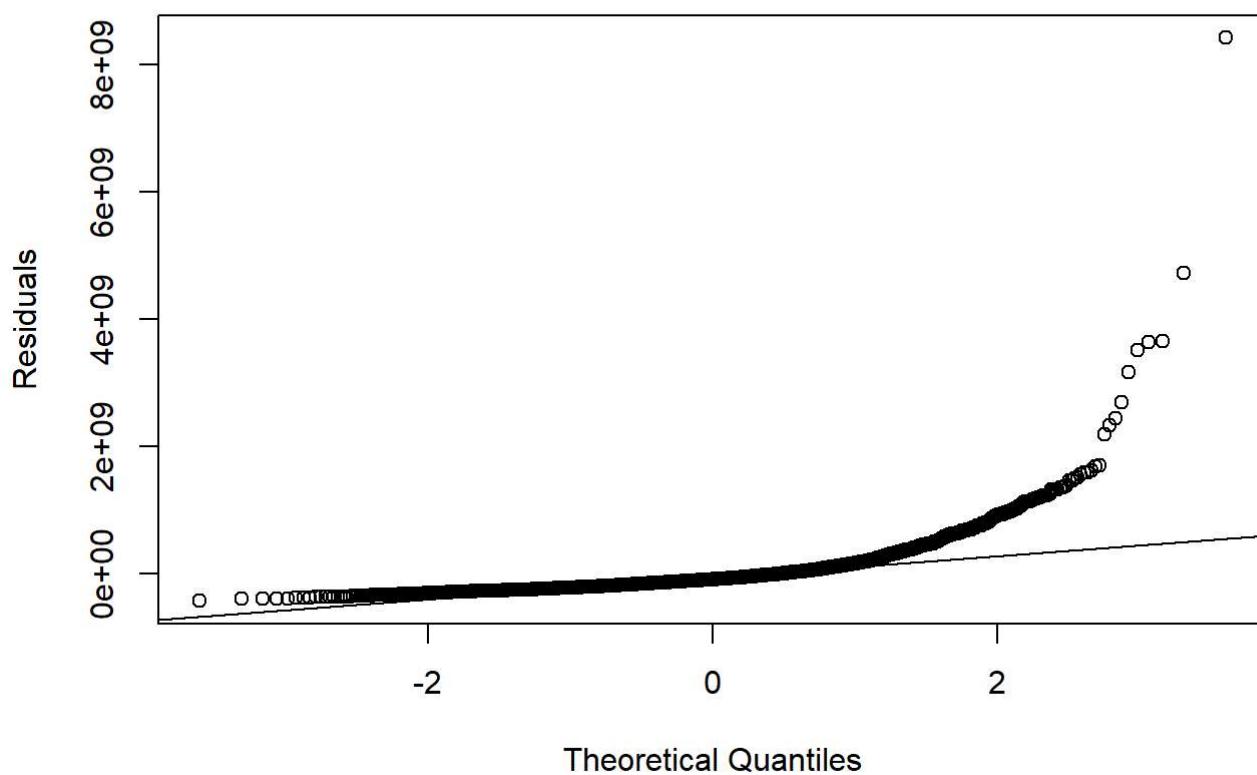
Residuals vs Fitted Values for adj_rev vs vote_average With Outliers



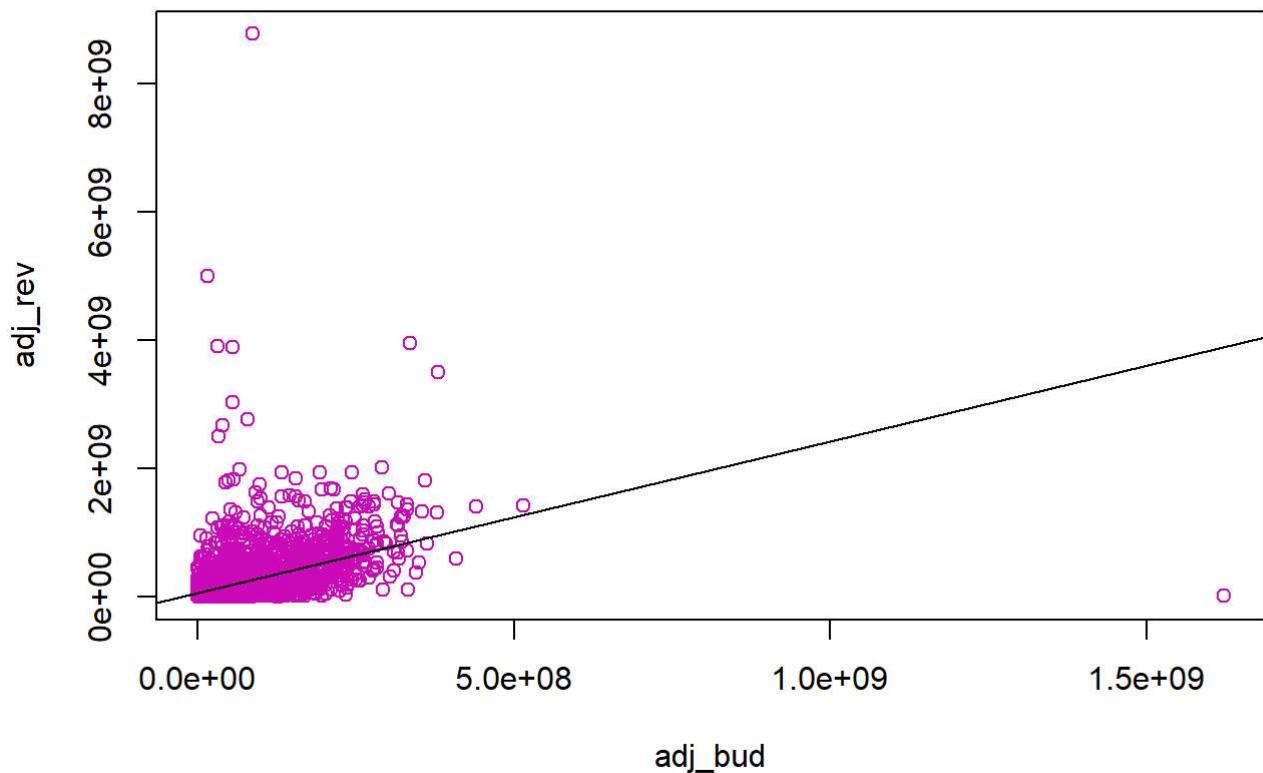
Histogram of the Residuals for adj_rev vs vote_average With Outliers



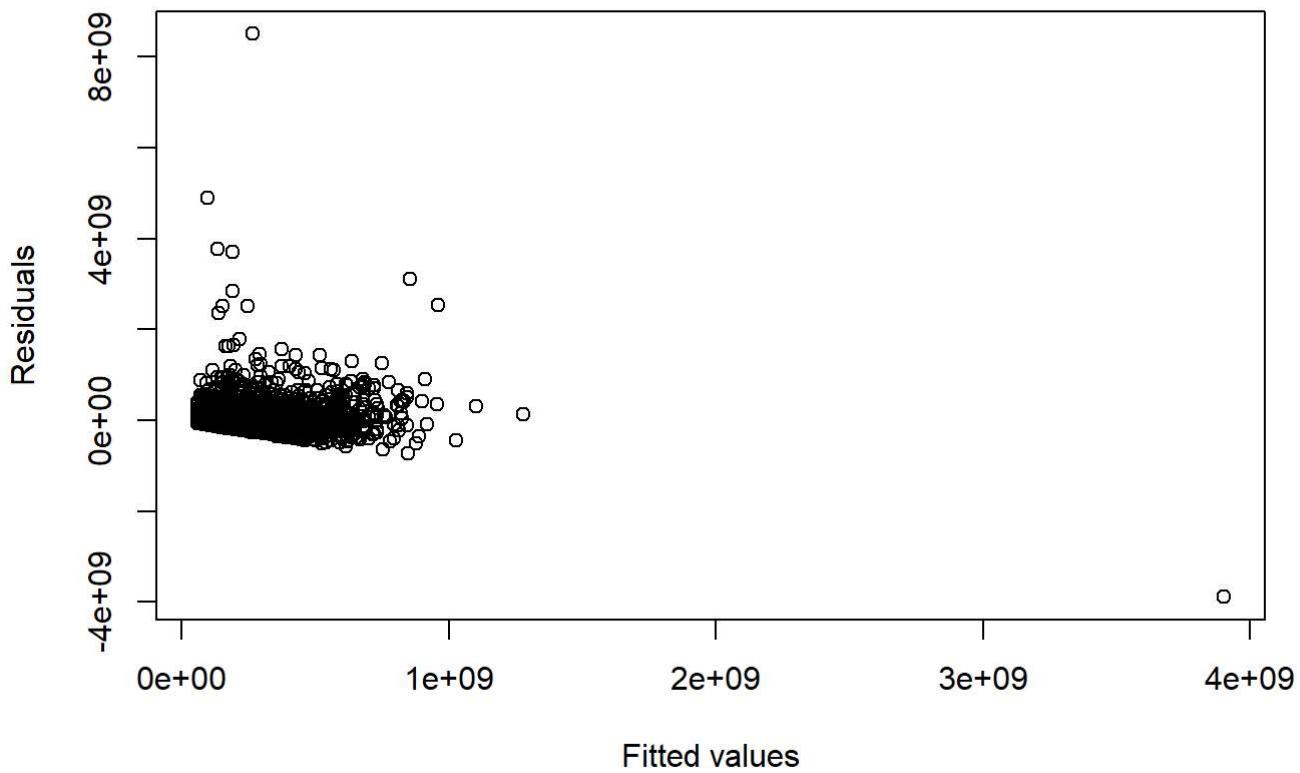
Normal Q-Q plot of the Residuals for adj_rev vs vote_average With Outliers



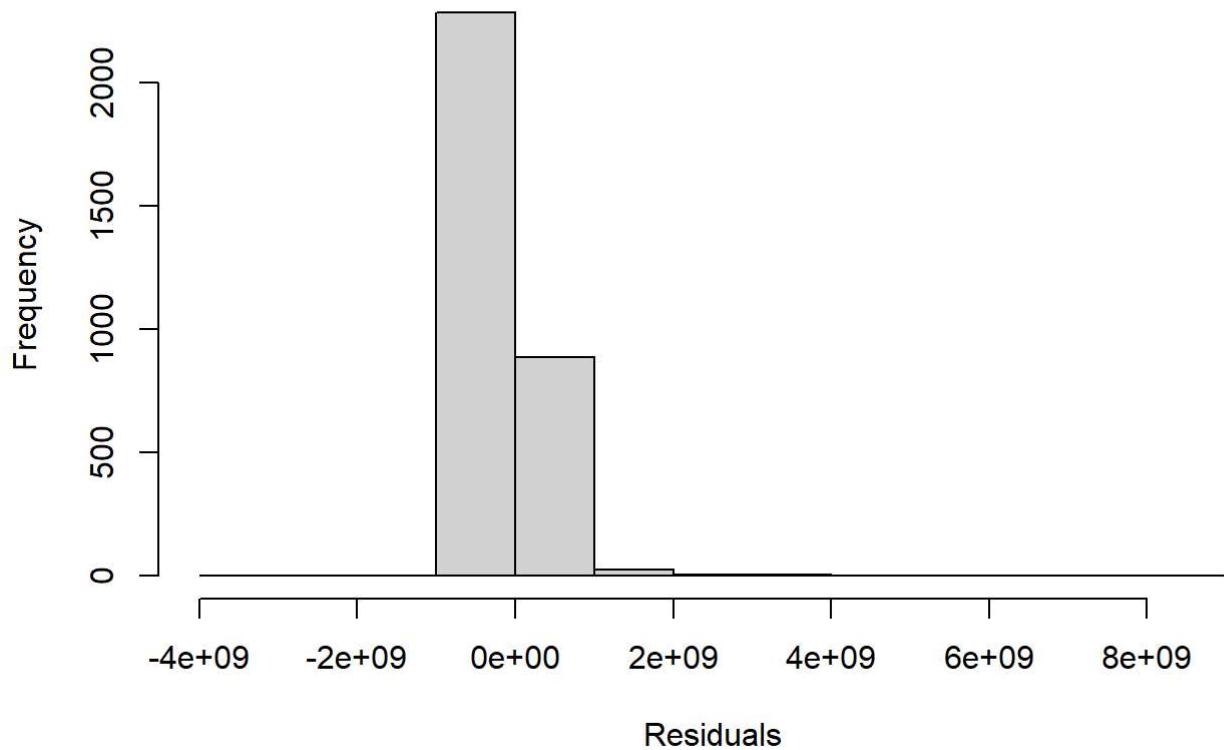
Plot of adj_rev vs adj_bud With Outliers



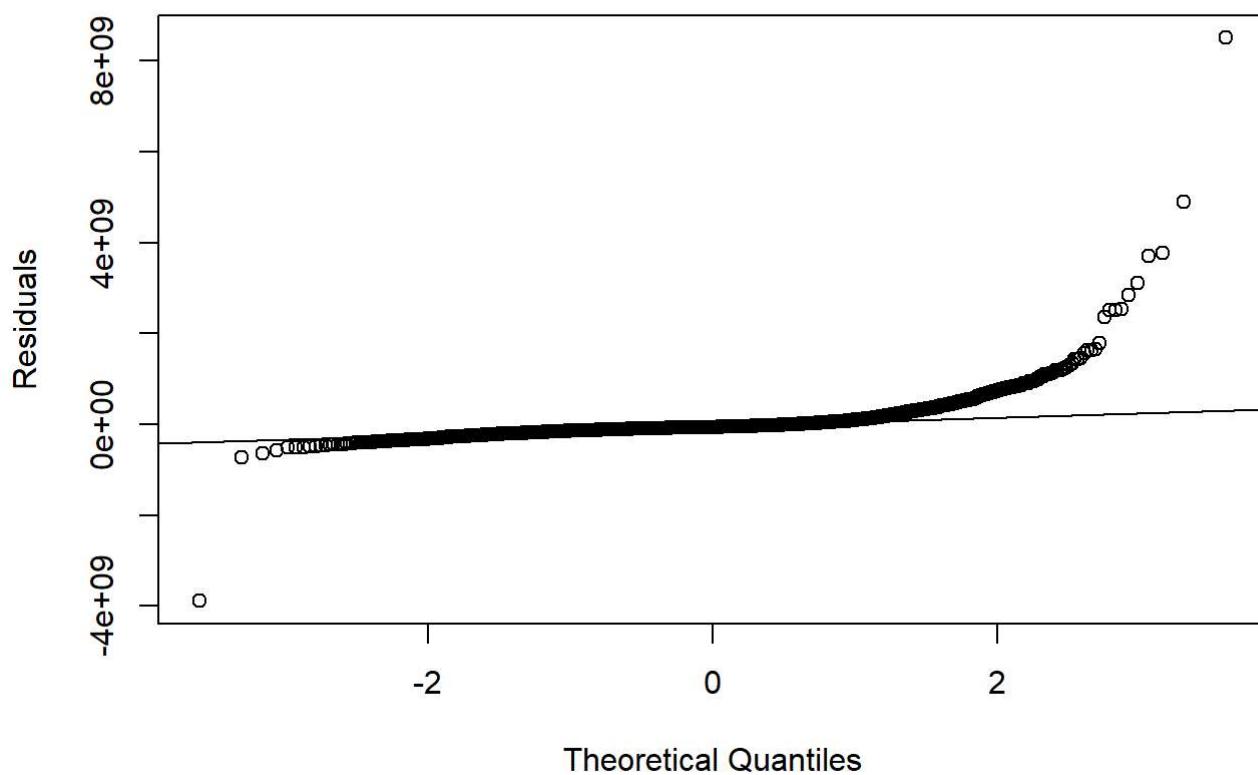
Residuals vs Fitted Values for adj_rev vs adj_bud With Outliers



Histogram of the Residuals for adj_rev vs adj_bud With Outliers



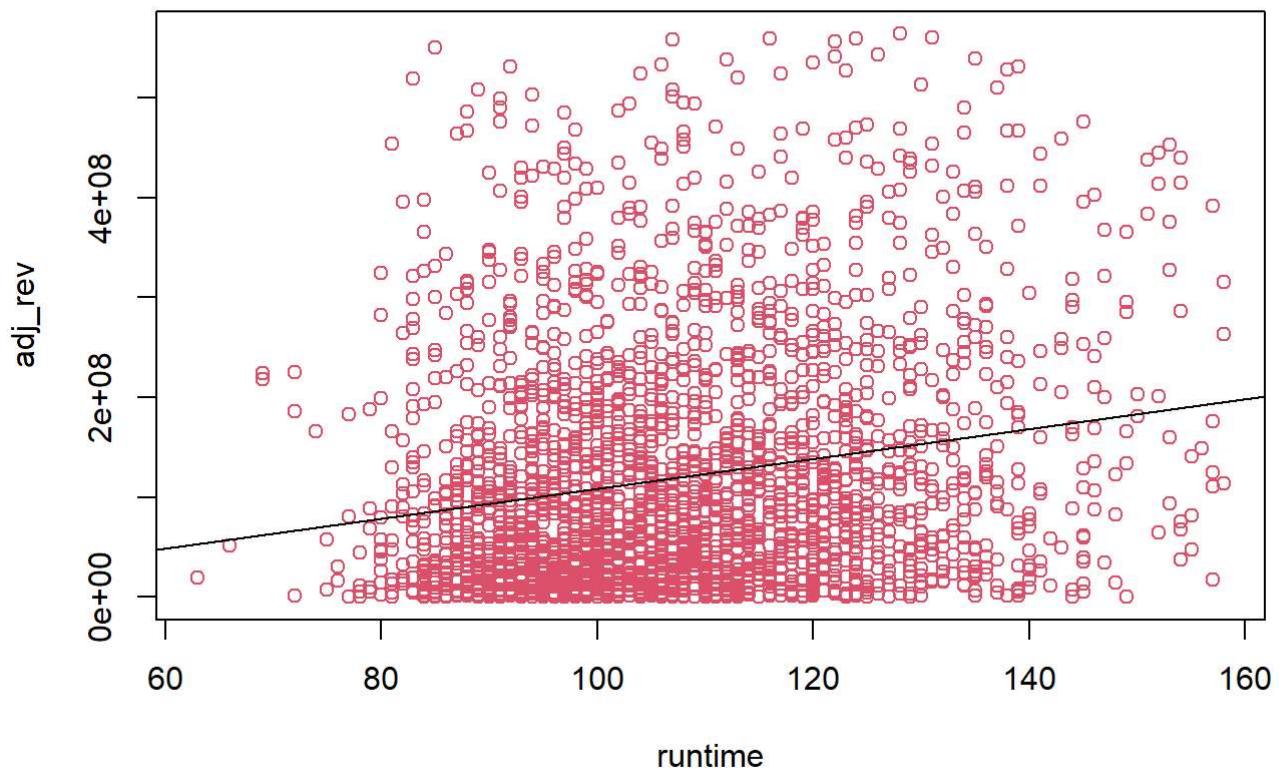
Normal Q-Q plot of the Residuals for adj_rev vs adj_bud With Outliers



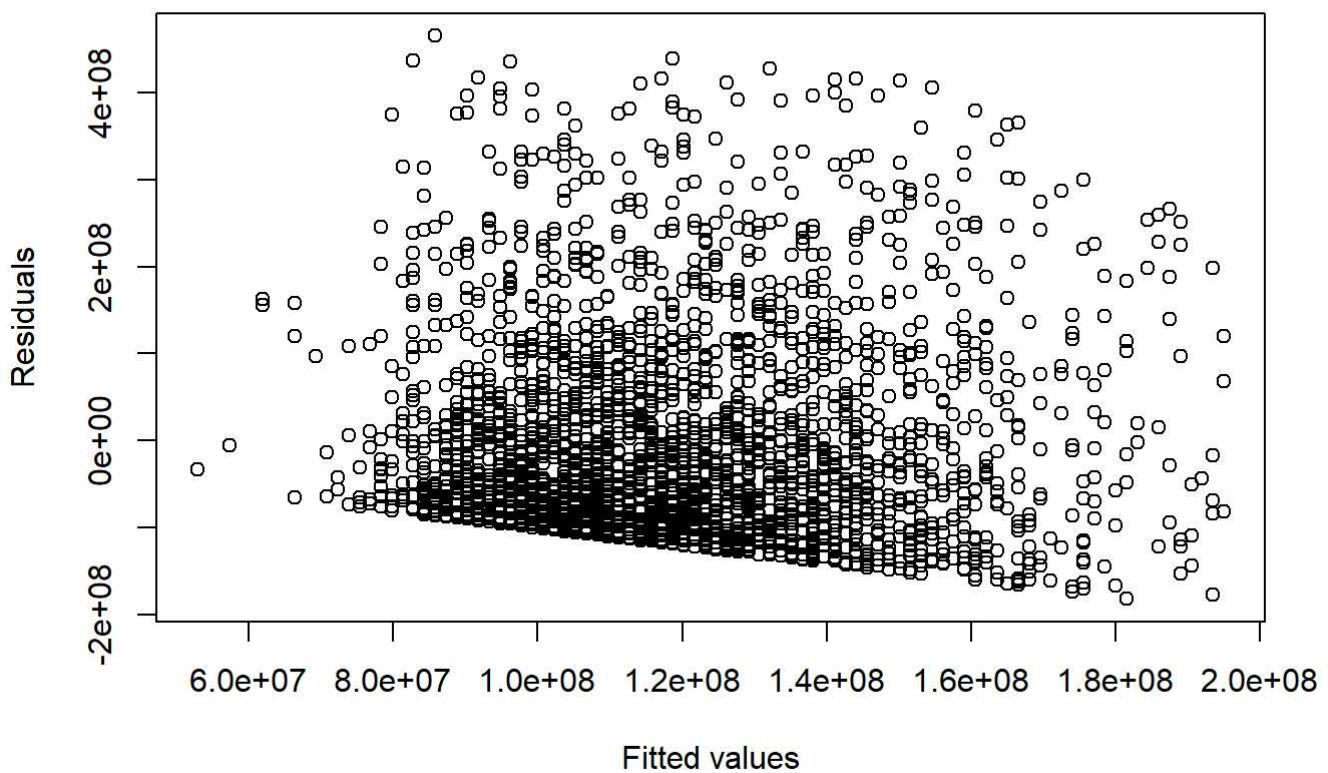
#We observe that most of the plots, which we use to check the assumptions of regression, are significantly distorted for when we don't remove outliers from our dataset. To better understand if the simple Linear models follow the assumptions of regression, we will remove the outliers from our data and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

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

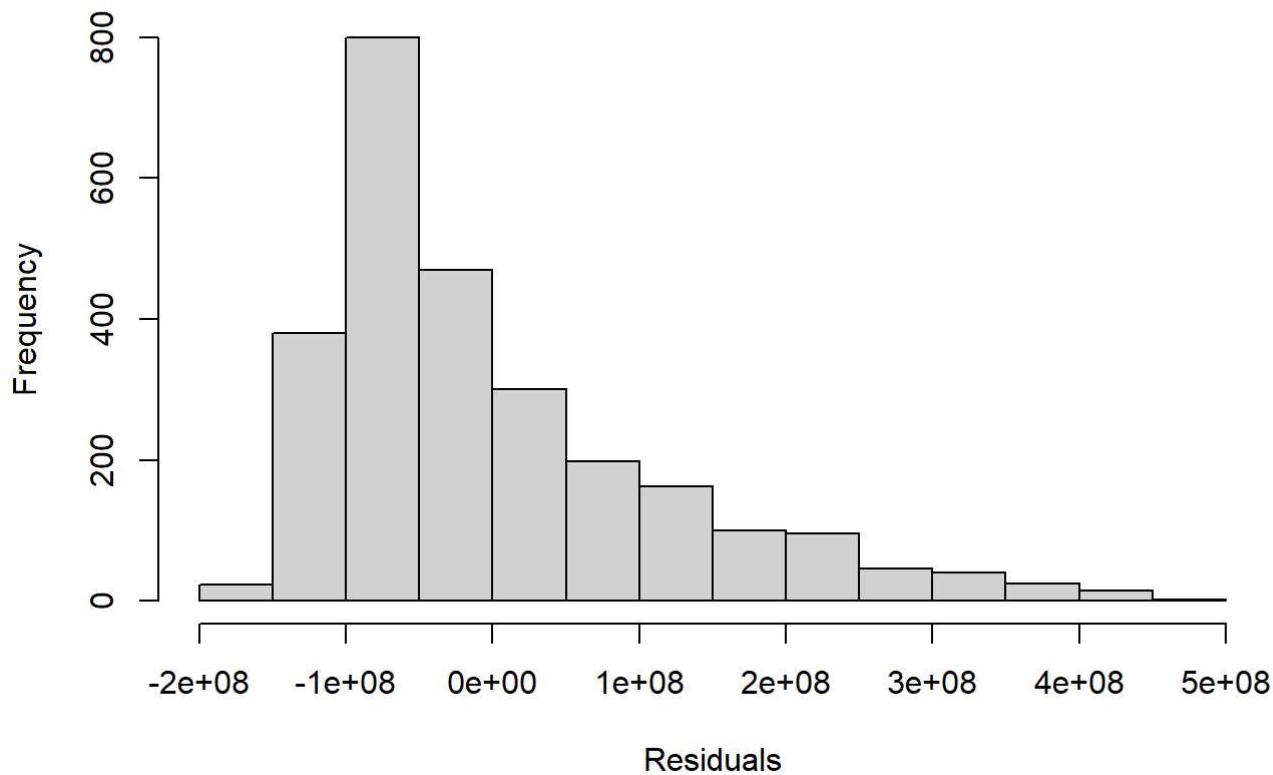
Plot of adj_rev vs runtime Without Outliers



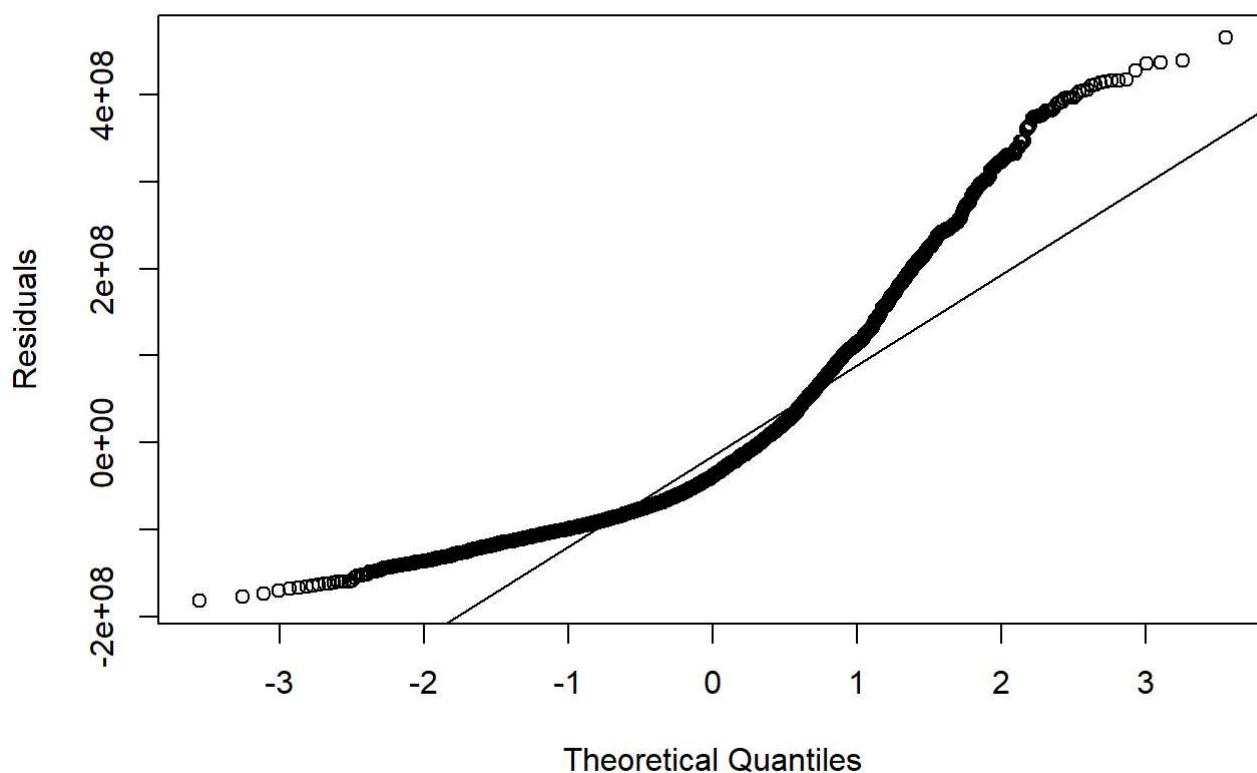
Residuals vs Fitted Values for adj_rev vs runtime Without Outliers



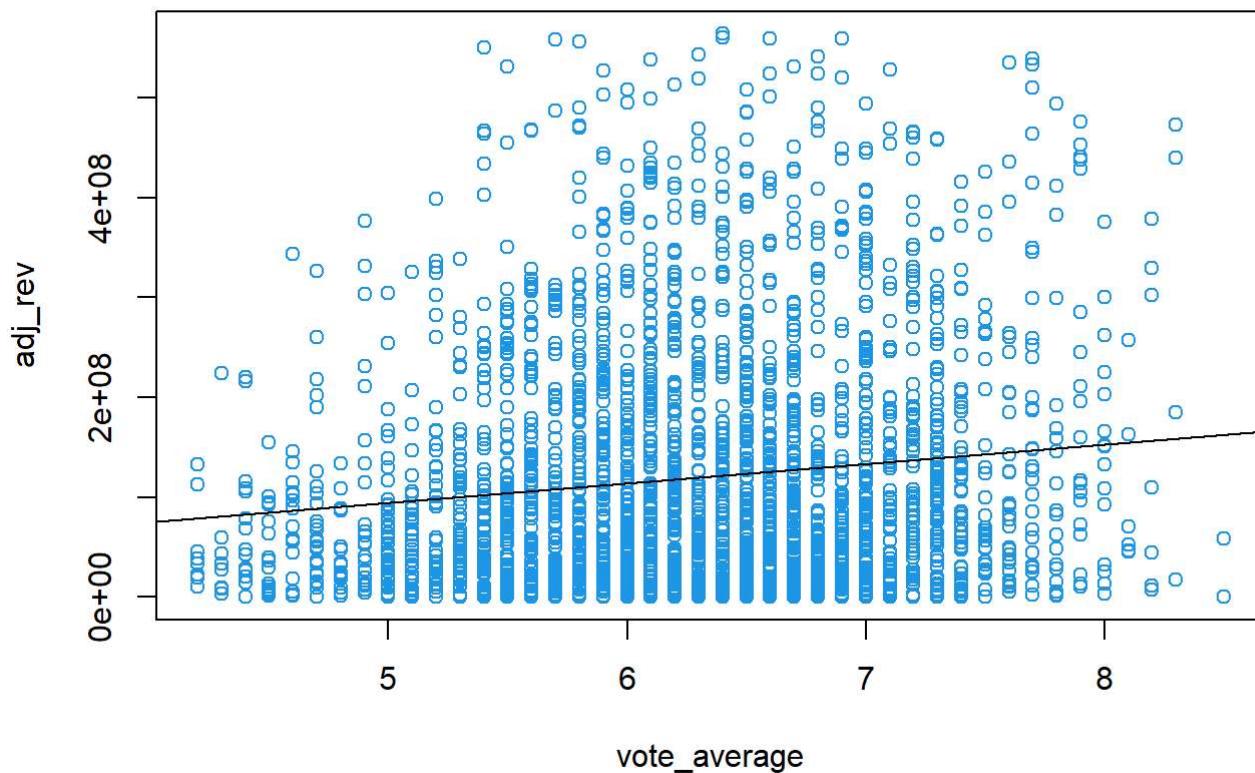
Histogram of the Residuals for adj_rev vs runtime Without Outliers



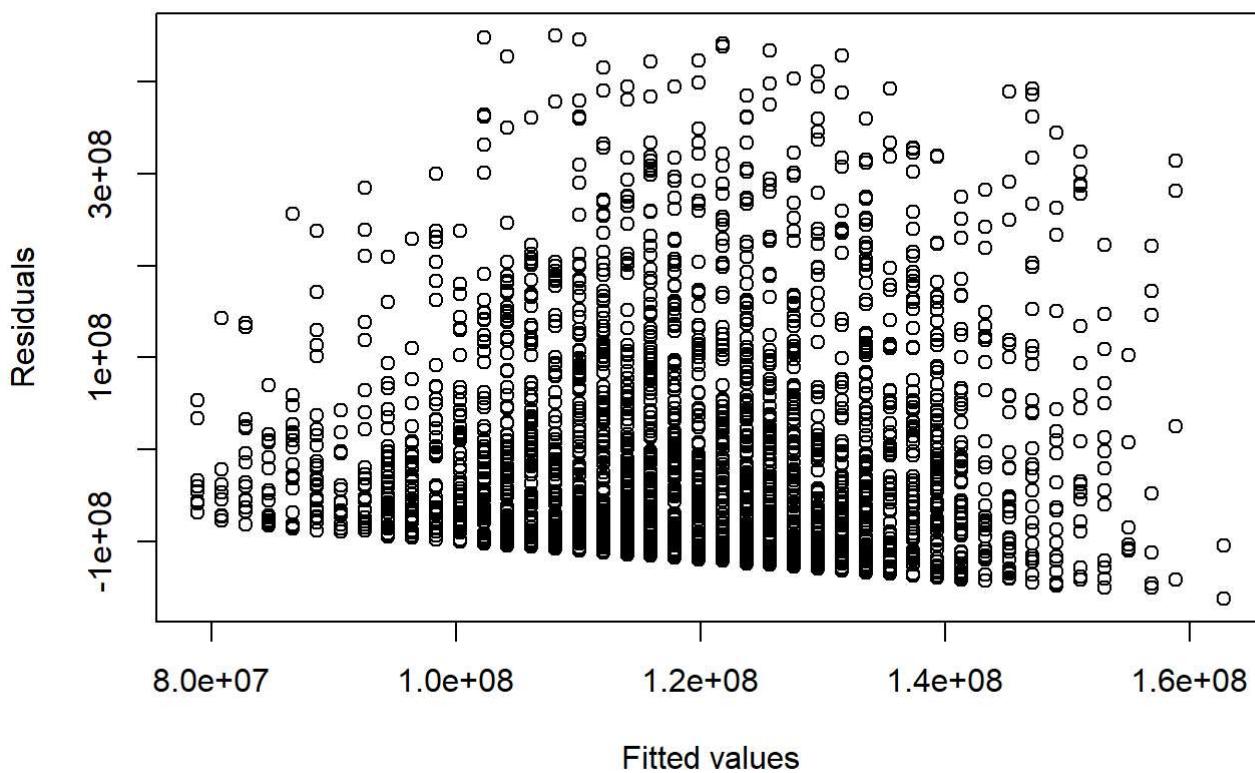
Normal Q-Q plot of the Residuals for adj_rev vs runtime Without Outlier



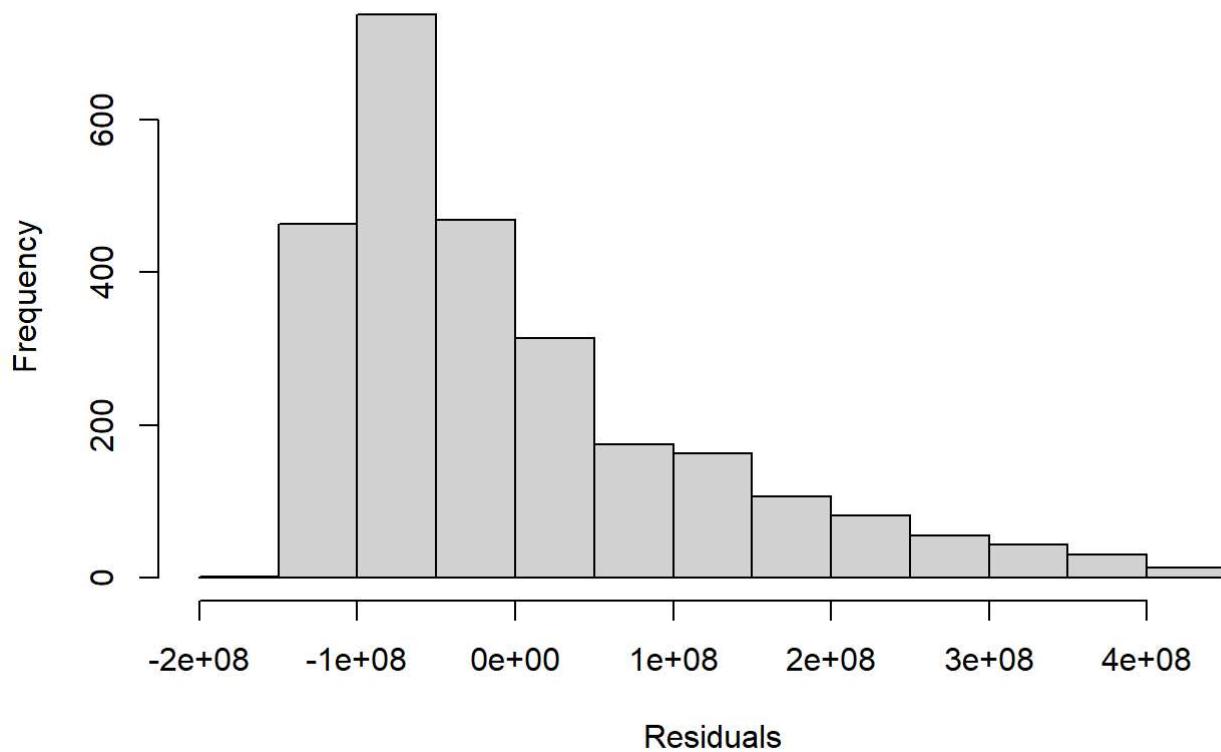
Plot of adj_rev vs vote_average Without Outliers



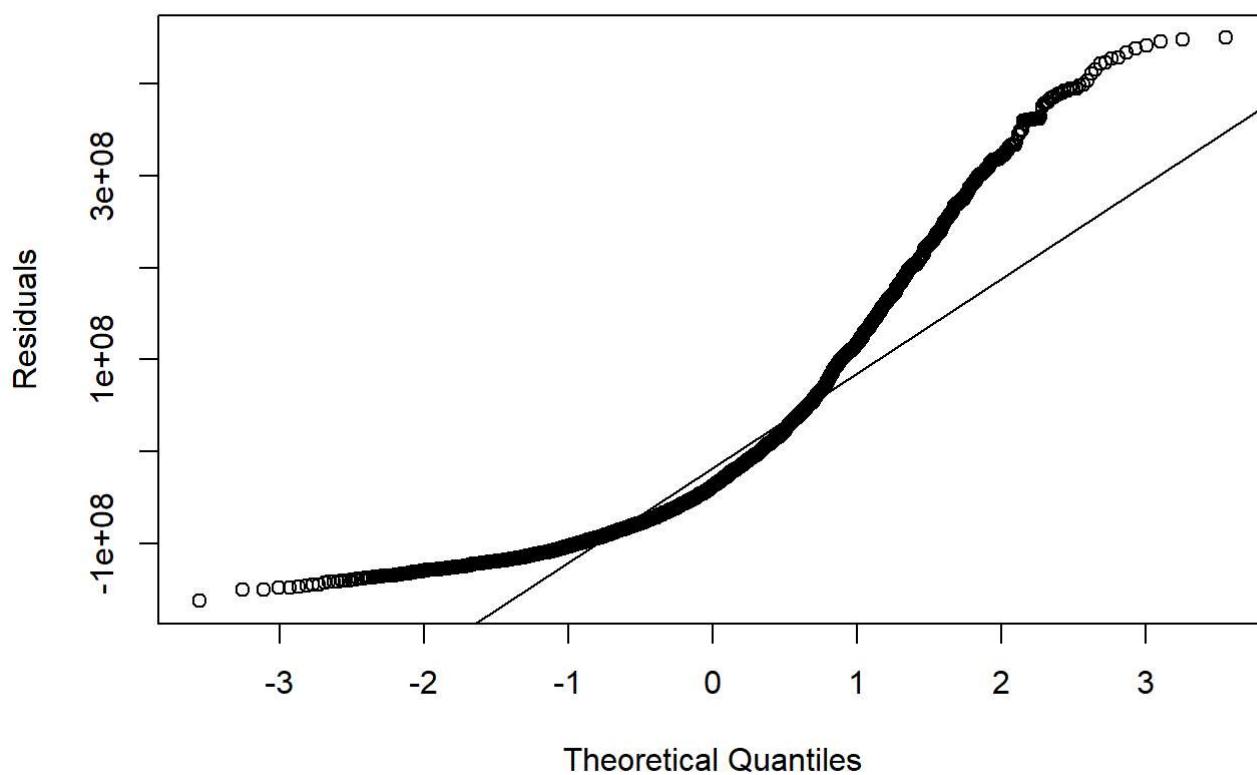
Residuals vs Fitted Values for adj_rev vs vote_average Without Outliers



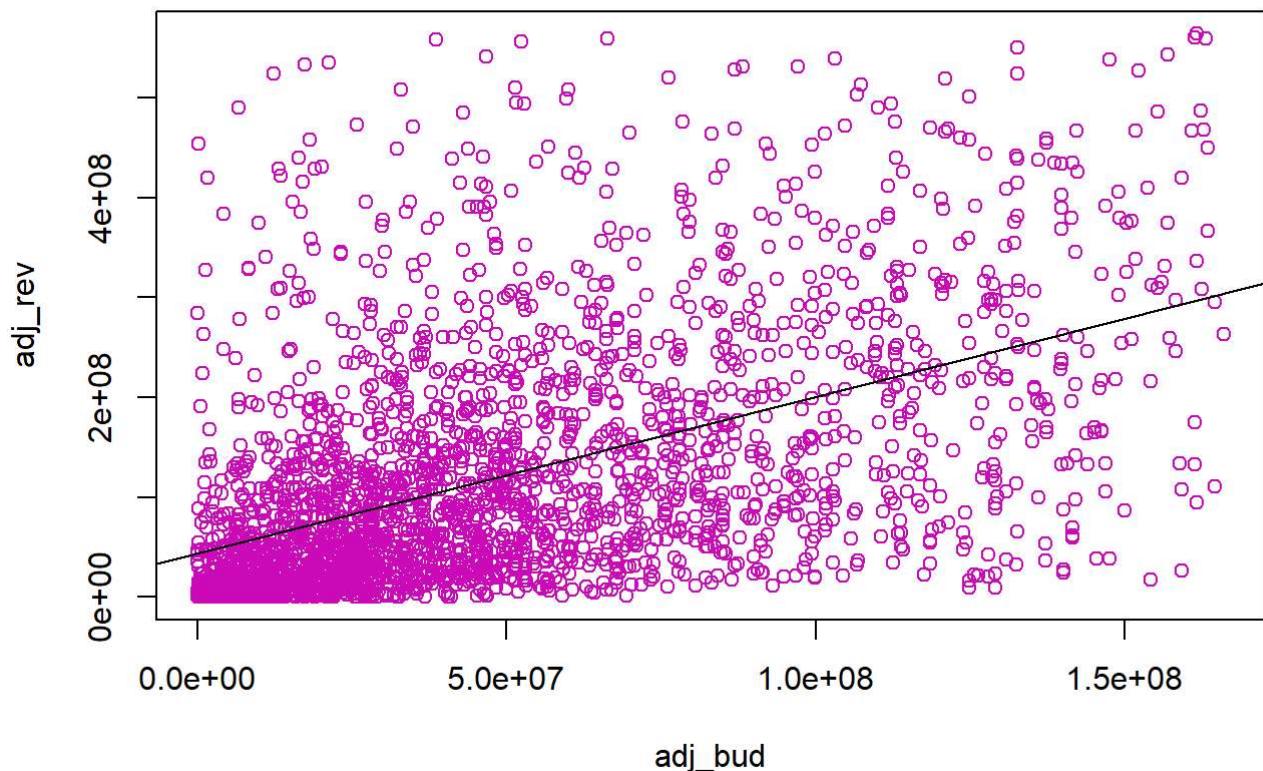
Histogram of the Residuals for adj_rev vs vote_average Without Outlier



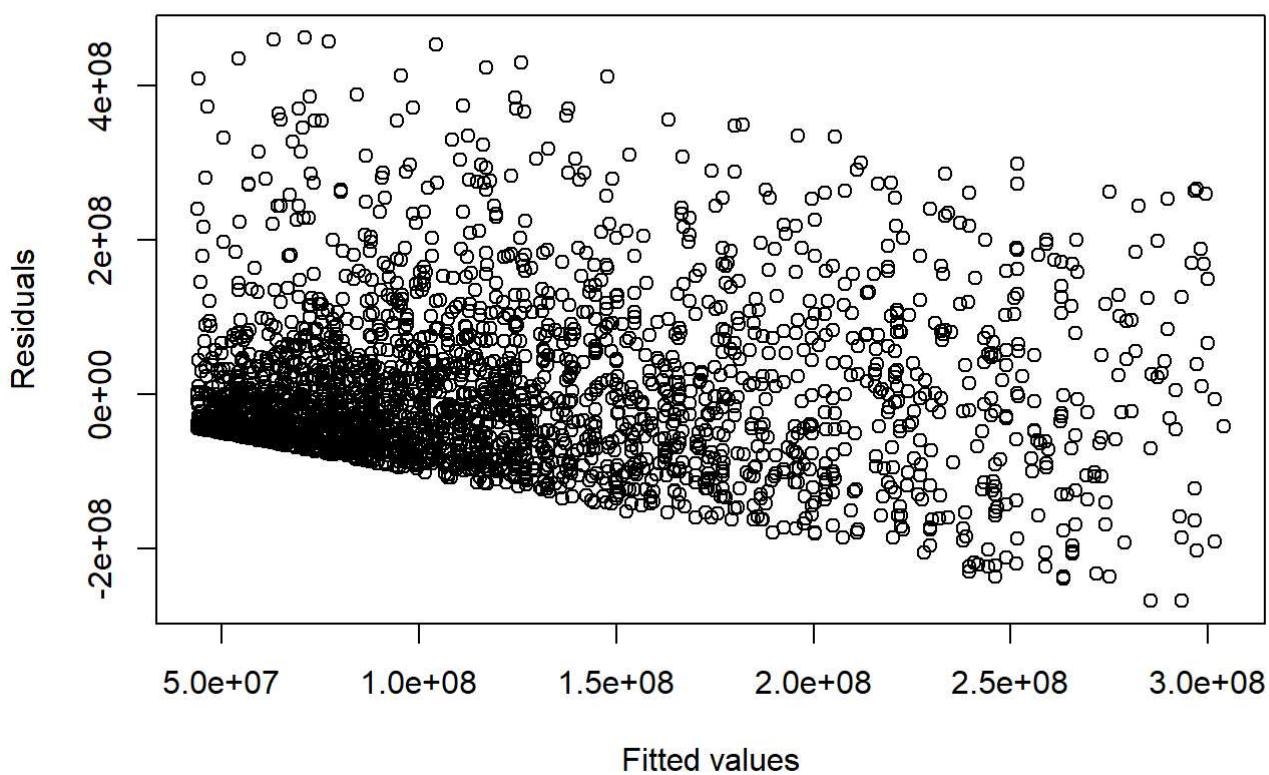
Normal Q-Q plot of the Residuals for adj_rev vs vote_average Without Outlier



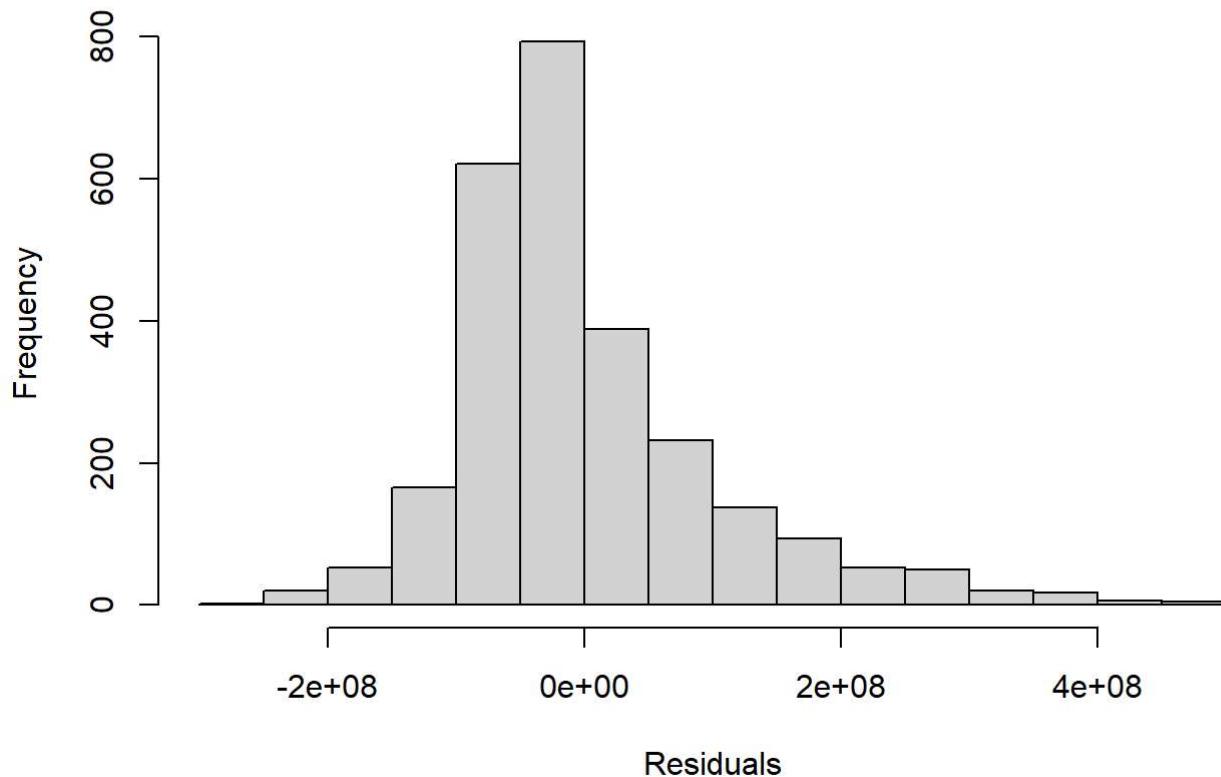
Plot of adj_rev vs adj_bud Without Outliers



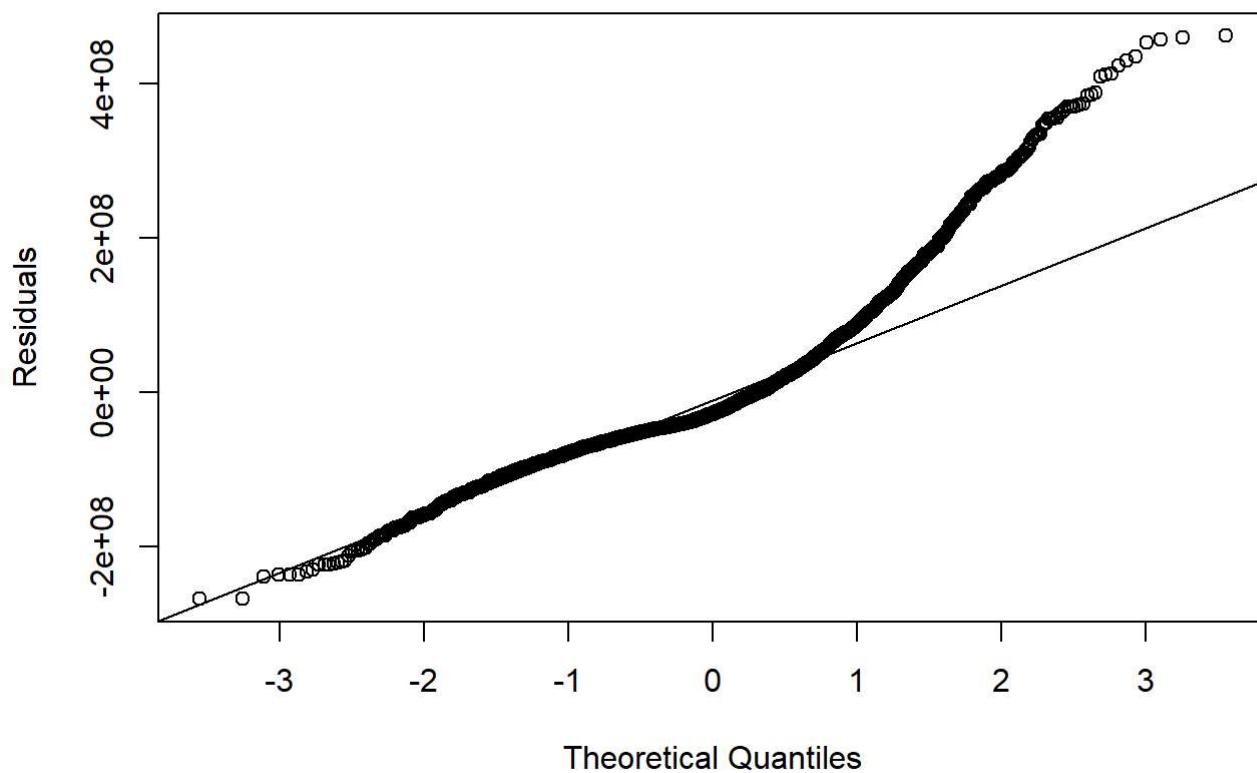
Residuals vs Fitted Values for adj_rev vs adj_bud Without Outliers



Histogram of the Residuals for adj_rev vs adj_bud Without Outliers



Normal Q-Q plot of the Residuals for adj_rev vs adj_bud Without Outliers



#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 and end of the data for variables runtime and vote_average; and that the residuals are right-skewed for adjusted_budget.

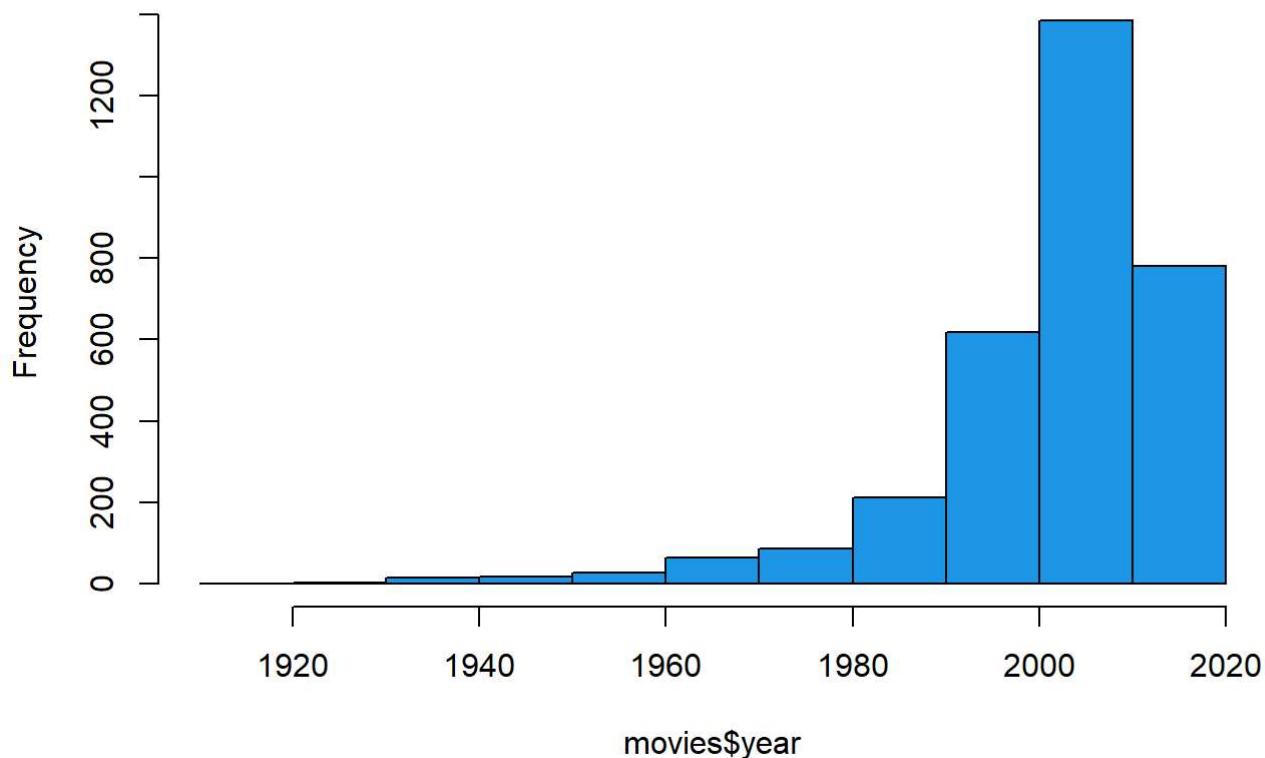
#We also see from the Residuals vs Fitted Values plots that the residuals appear to be trending downwards.

#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

```
hist(movies$year, col=20)
```

Histogram of movies\$year



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

```

#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'

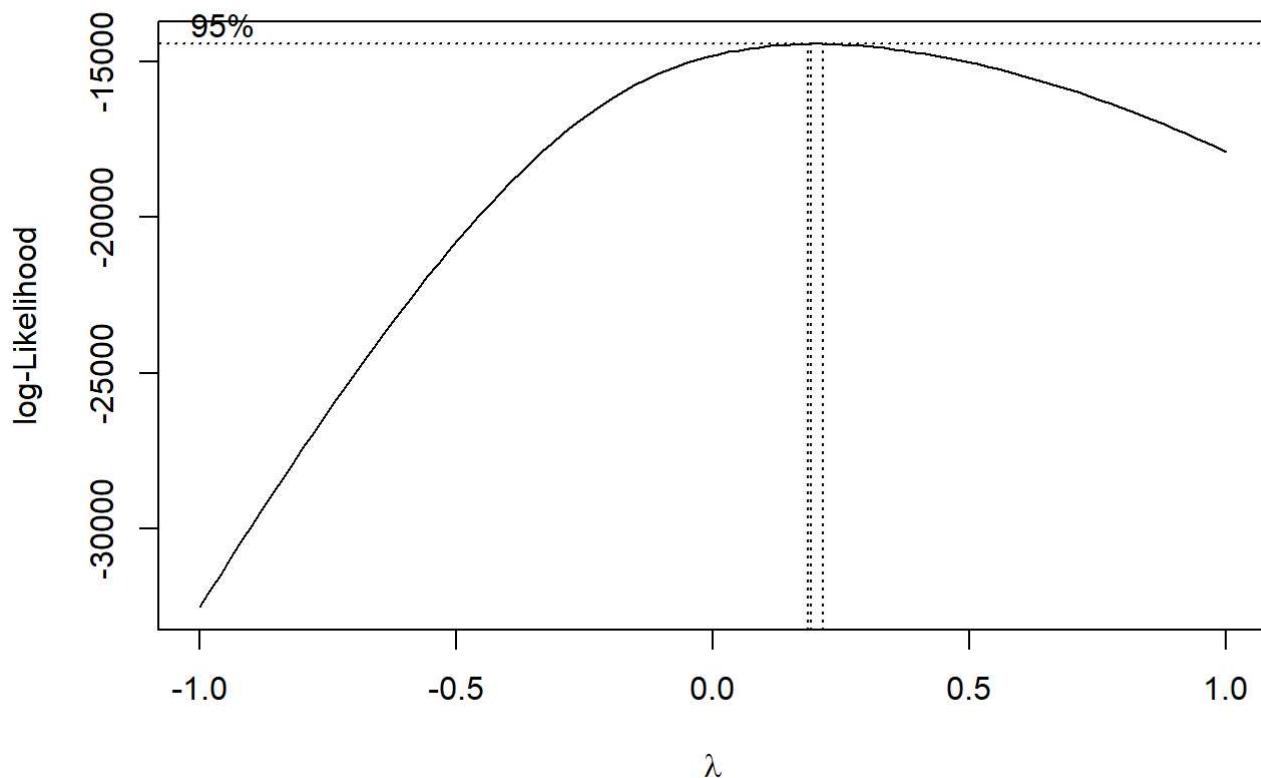
```

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

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

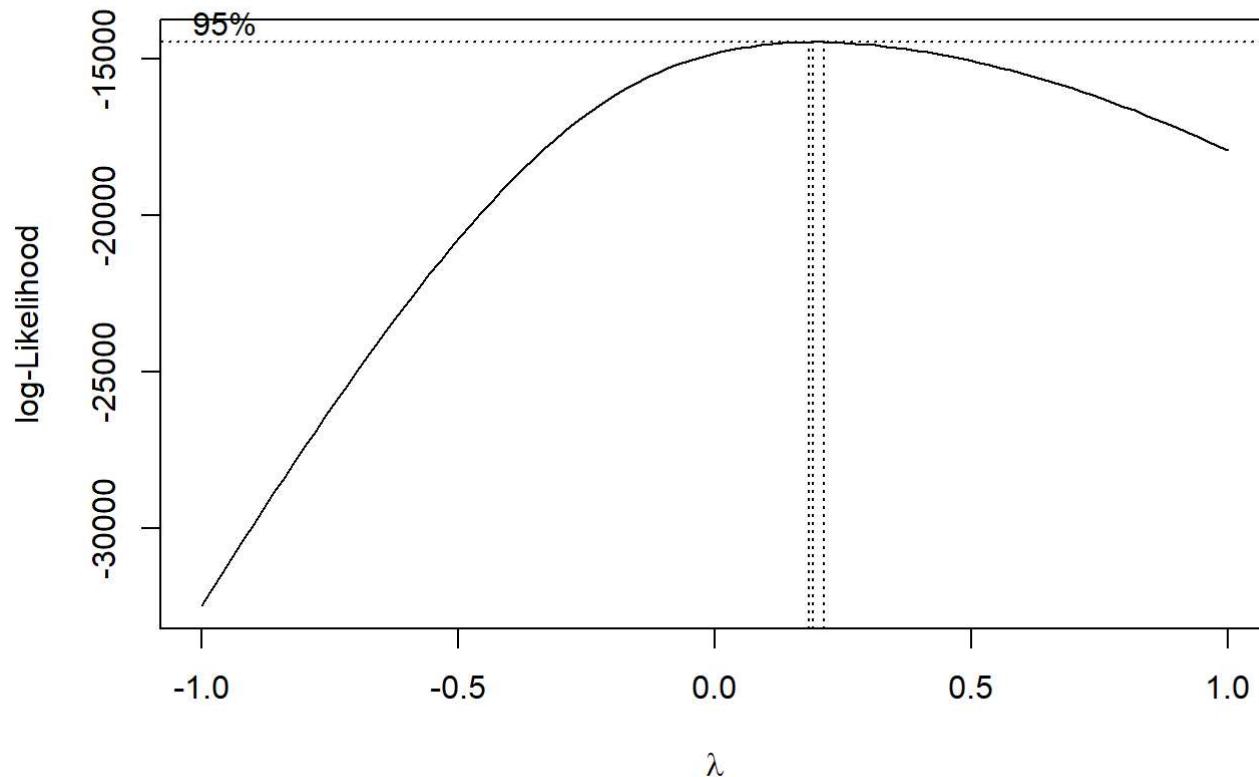
```
## [1] "Predictor: runtime"  
## [1] "Formula: ~"           "Formula: adj_rev" "Formula: runtime"  
## [1] "Linear model created successfully."
```

Box-Cox for adj_rev vs runtime Without Outliers



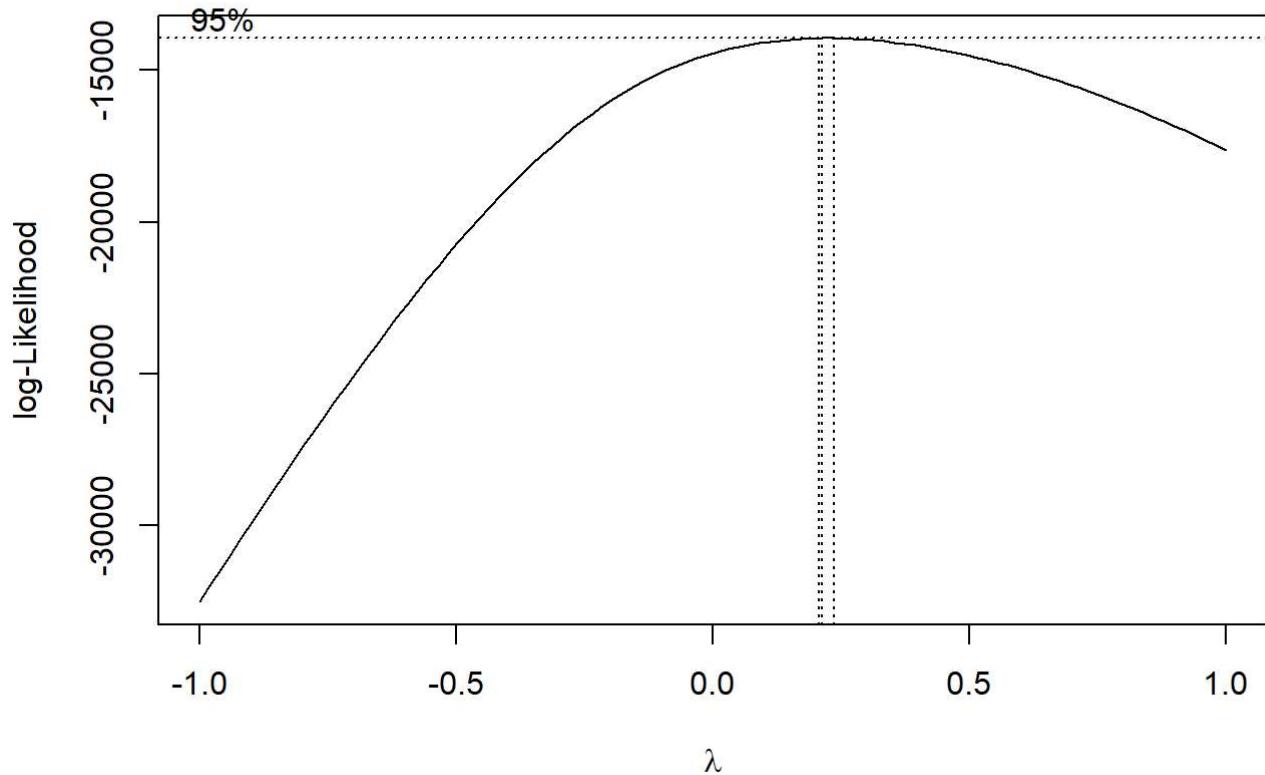
```
## [1] "Predictor: vote_average"  
## [1] "Formula: ~"           "Formula: adj_rev"      "Formula: vote_average"  
## [1] "Linear model created successfully."
```

Box-Cox for adj_rev vs vote_average Without Outliers



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

Box-Cox for adj_rev vs adj_bud Without Outliers



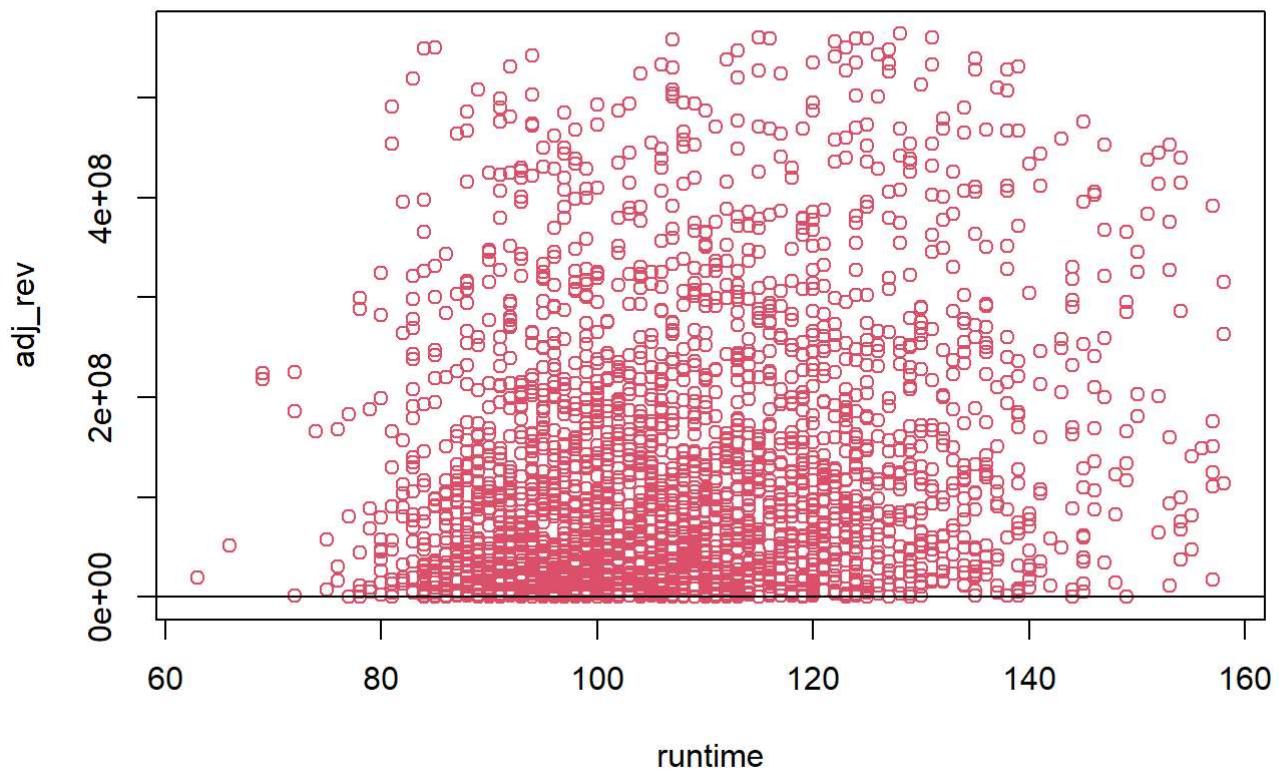
#From the plots, we observe that the optimal values for run_time and vote_average are 1/6. When comparing these variables with adj_rev, we will be applying a transformation of 1/6 to the response.

#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.

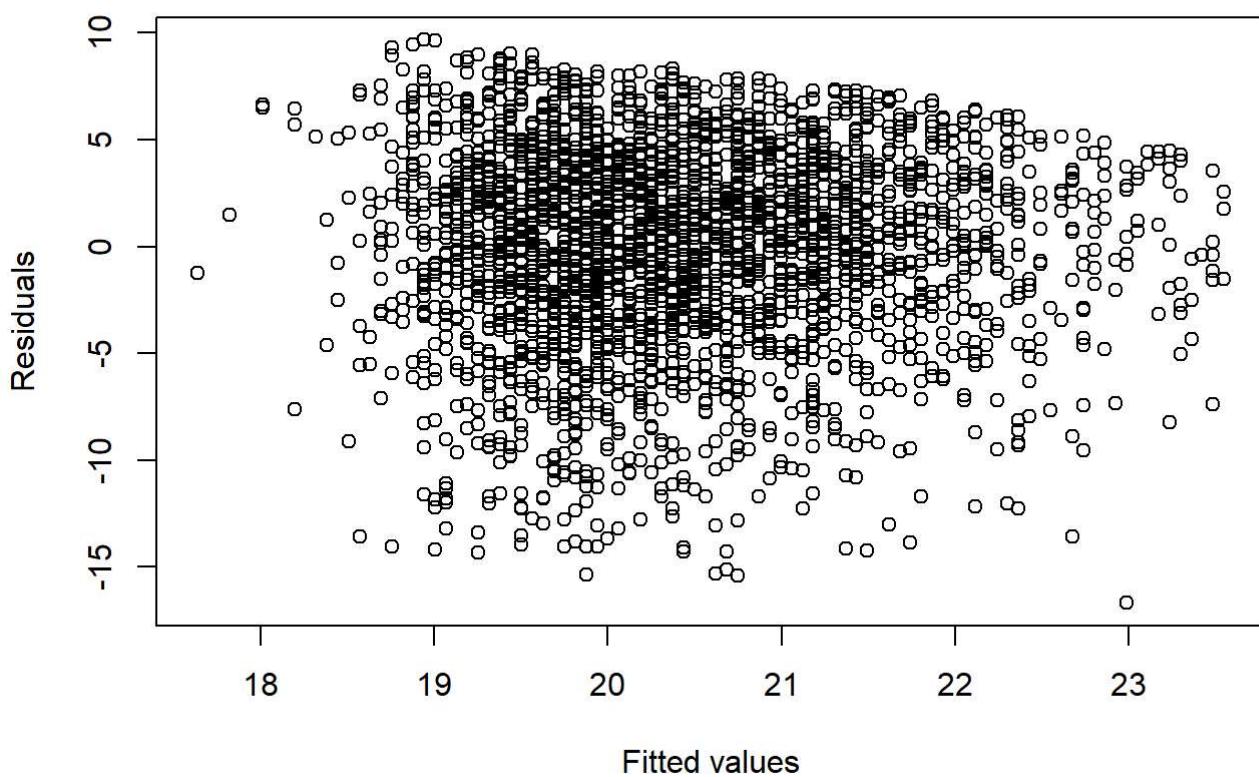
```
movies_sixth_root = subset(movies_regression, select=-adj_bud)
movies_fourth_root = subset(movies_regression, select=-c(runtime, vote_average))
```

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

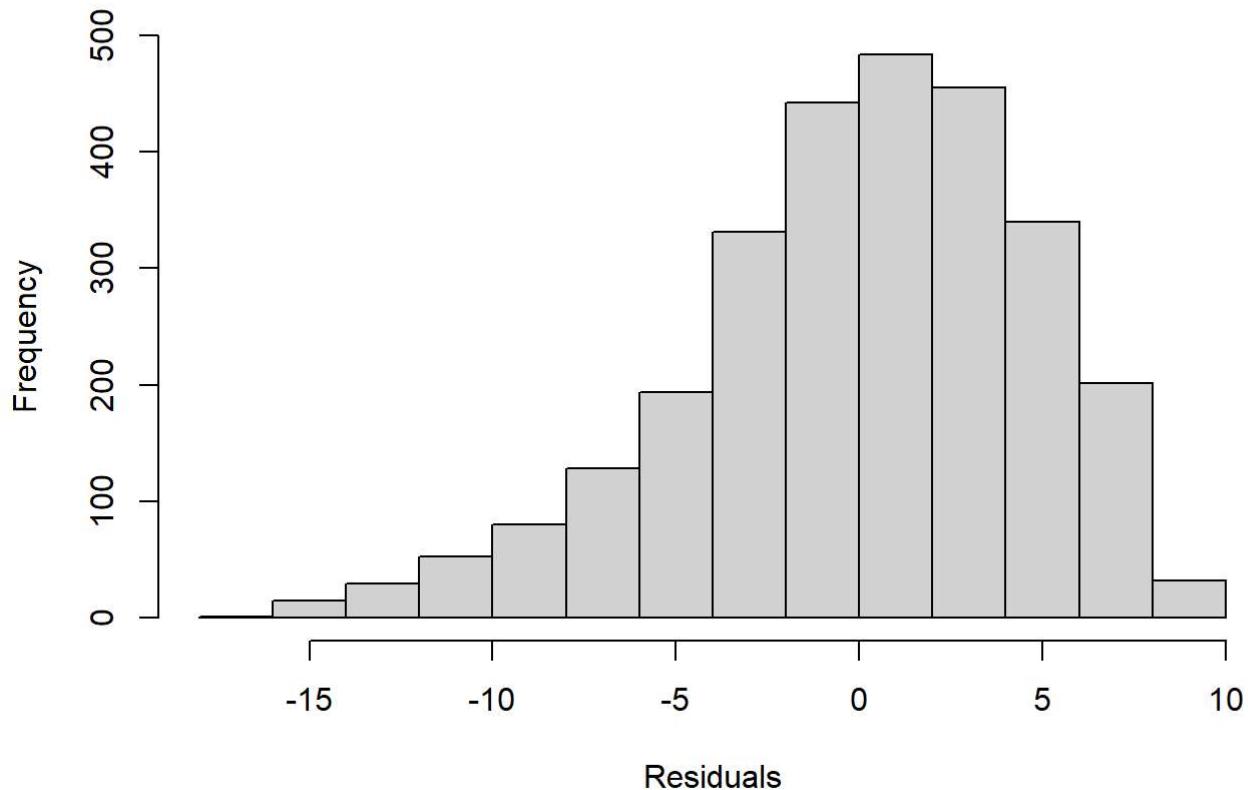
Plot of adj_rev vs runtime Without Outliers



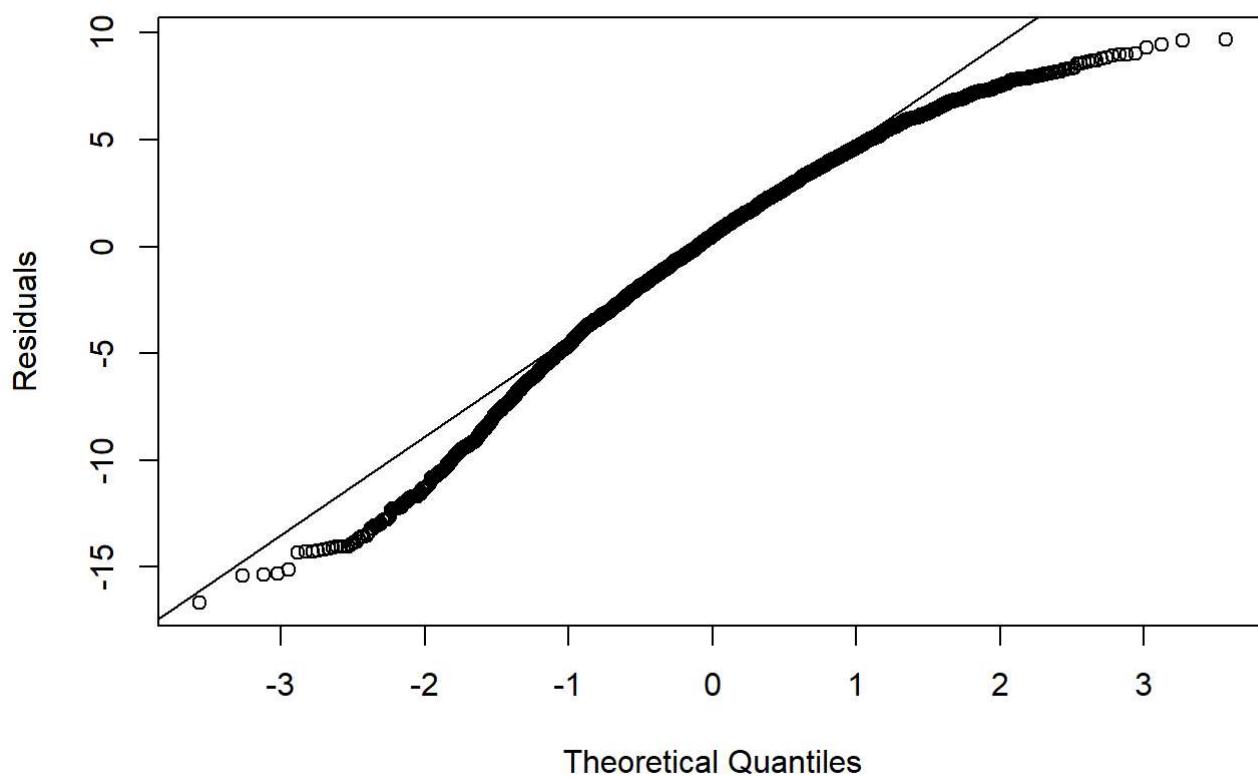
Residuals vs Fitted Values for adj_rev vs runtime Without Outliers



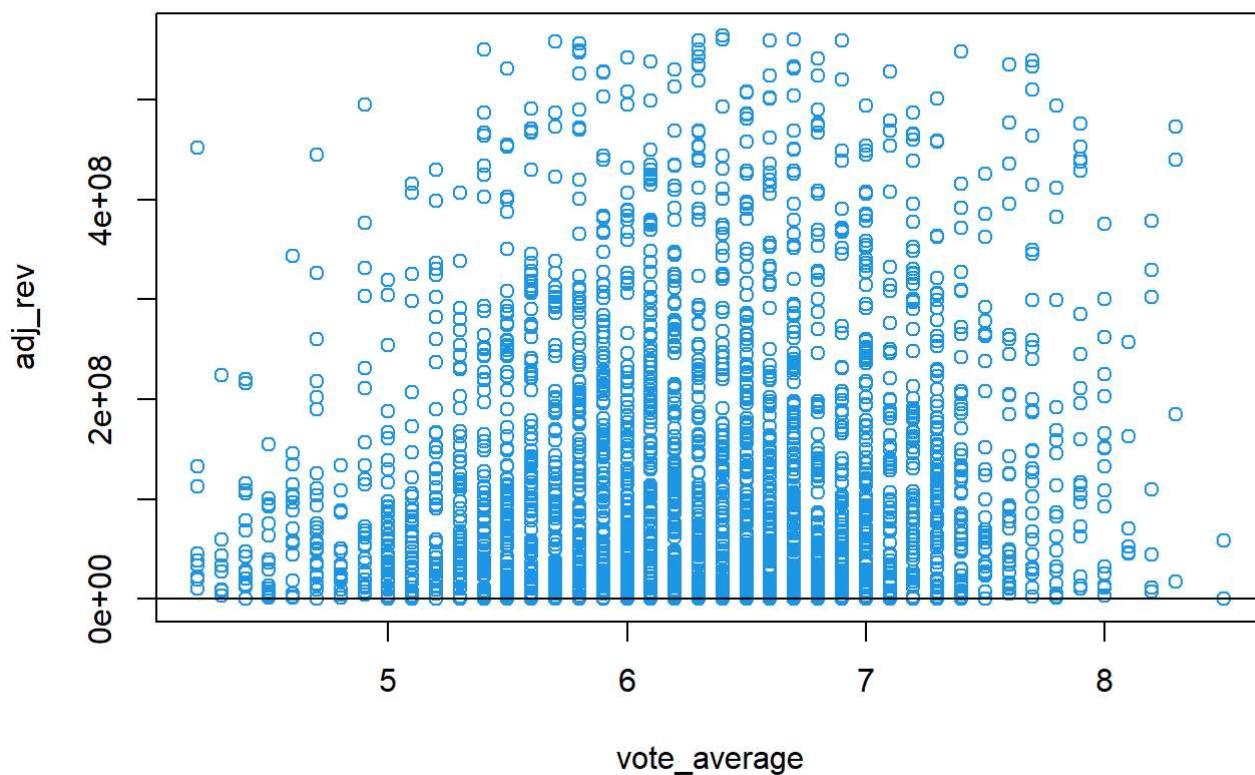
Histogram of the Residuals for adj_rev vs runtime Without Outliers



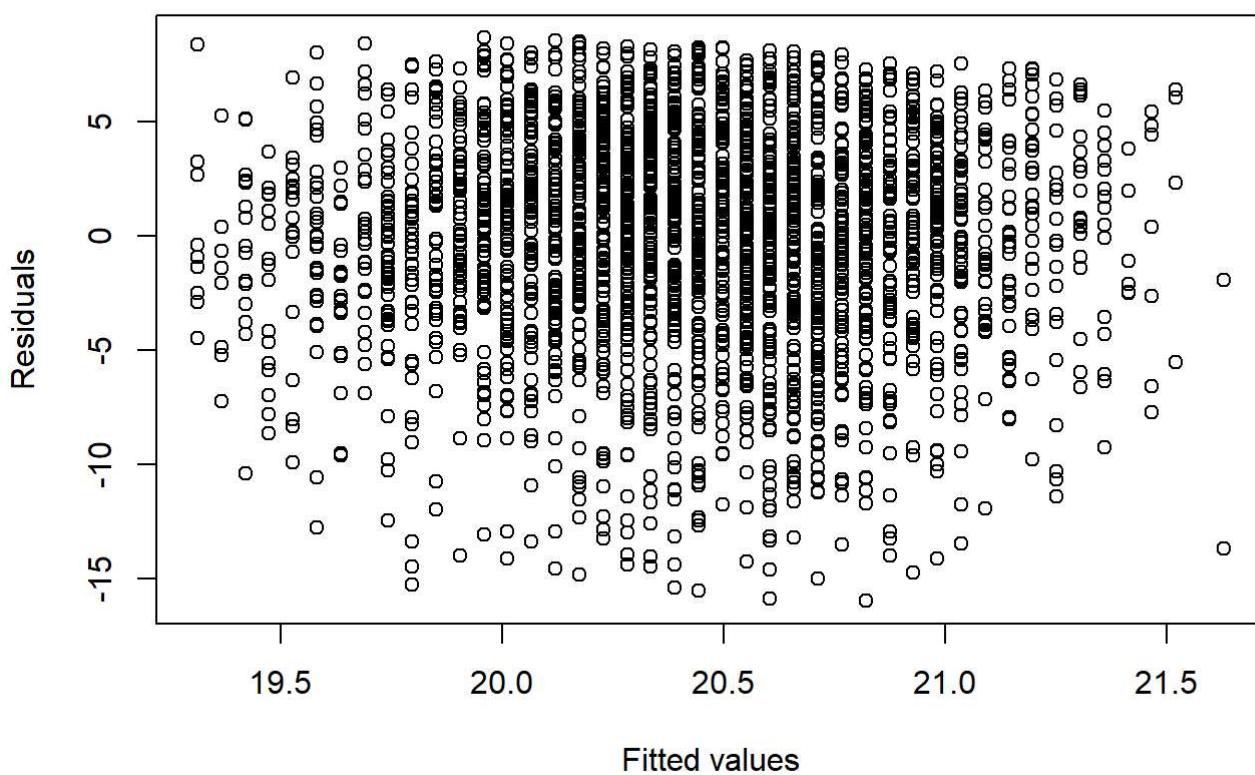
Normal Q-Q plot of the Residuals for adj_rev vs runtime Without Outlier



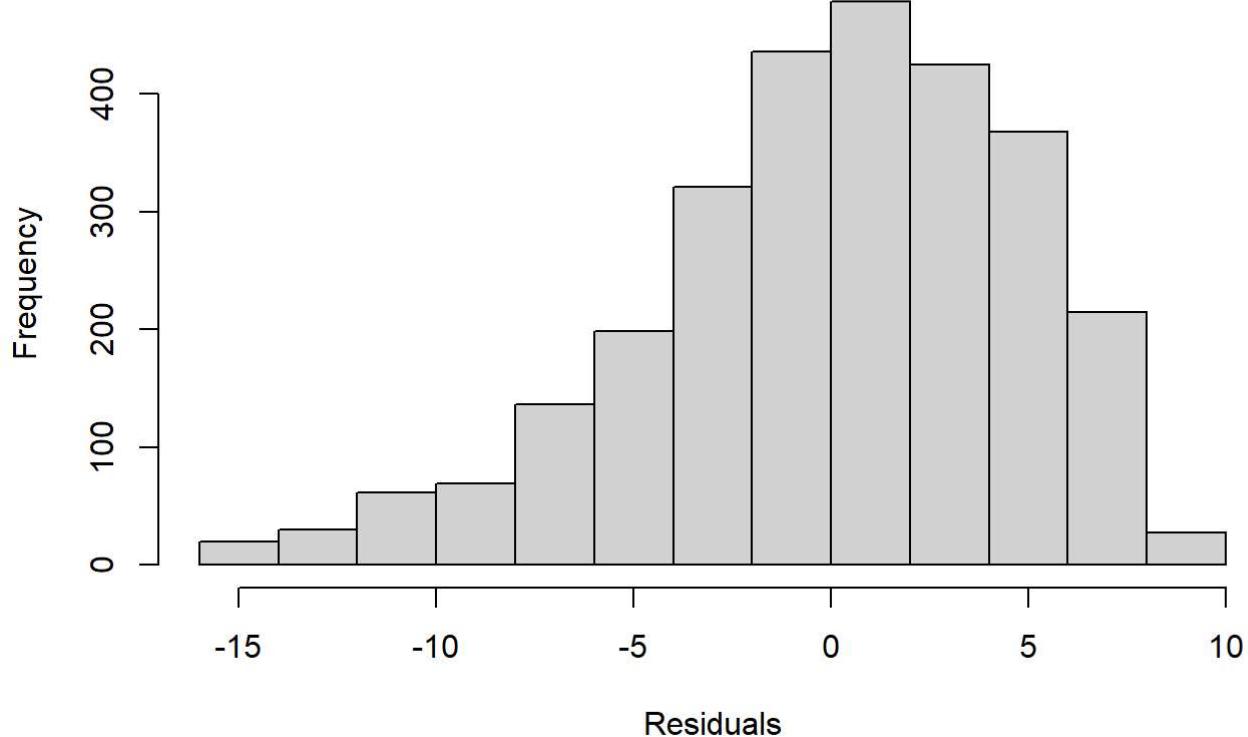
Plot of adj_rev vs vote_average Without Outliers



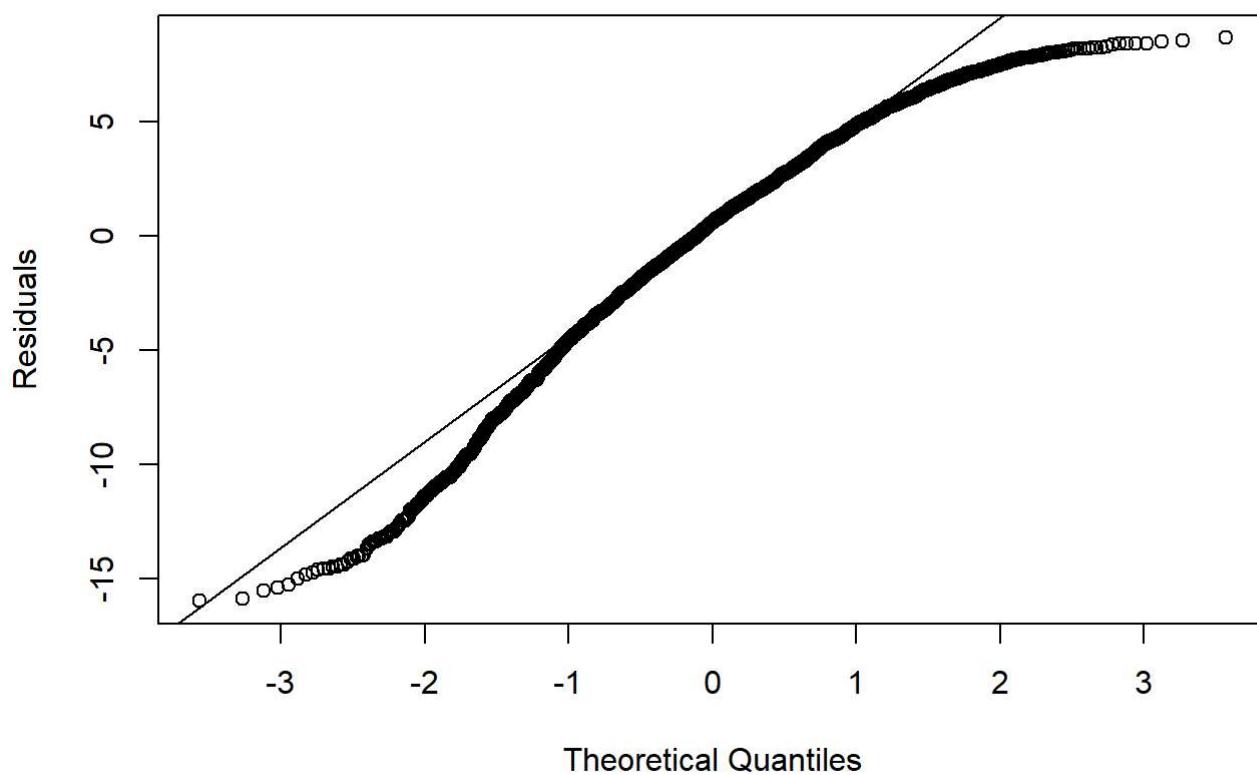
Residuals vs Fitted Values for adj_rev vs vote_average Without Outliers



Histogram of the Residuals for adj_rev vs vote_average Without Outlier

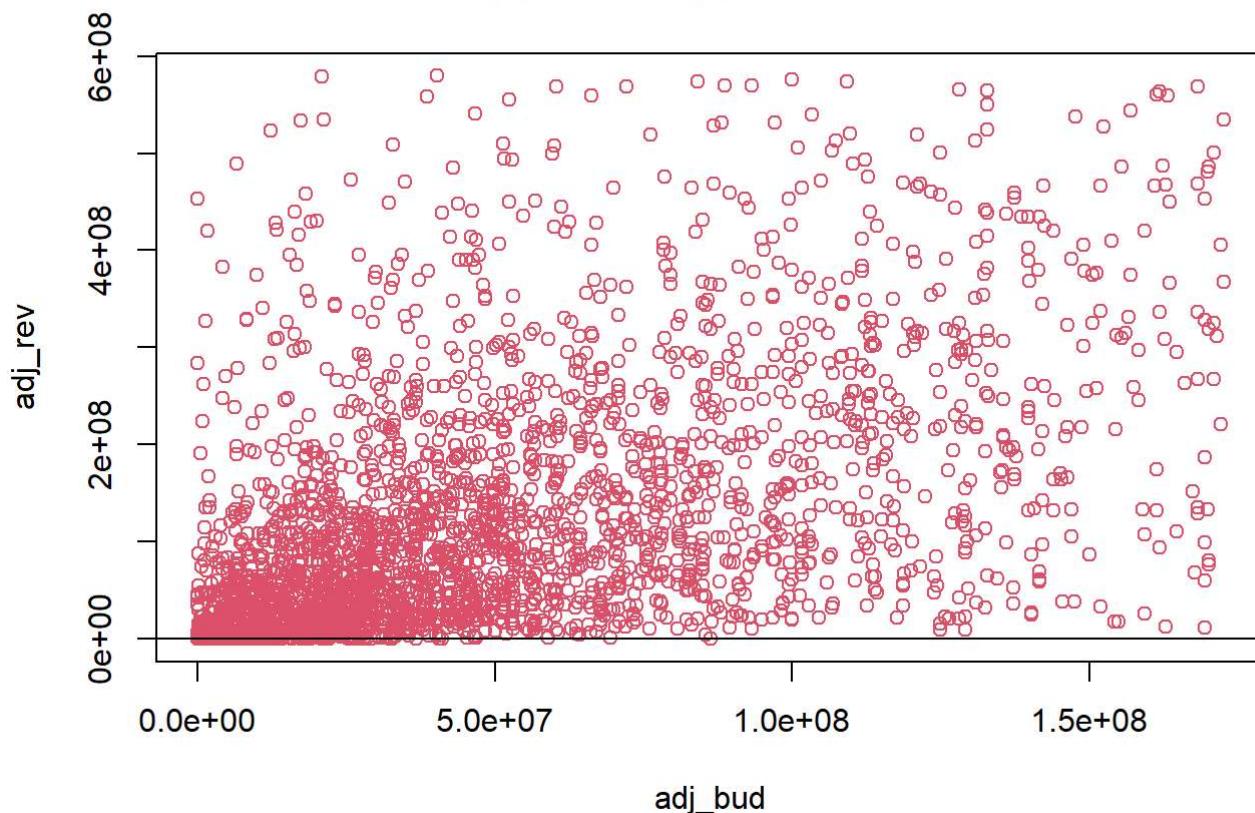


Normal Q-Q plot of the Residuals for adj_rev vs vote_average Without Outlier

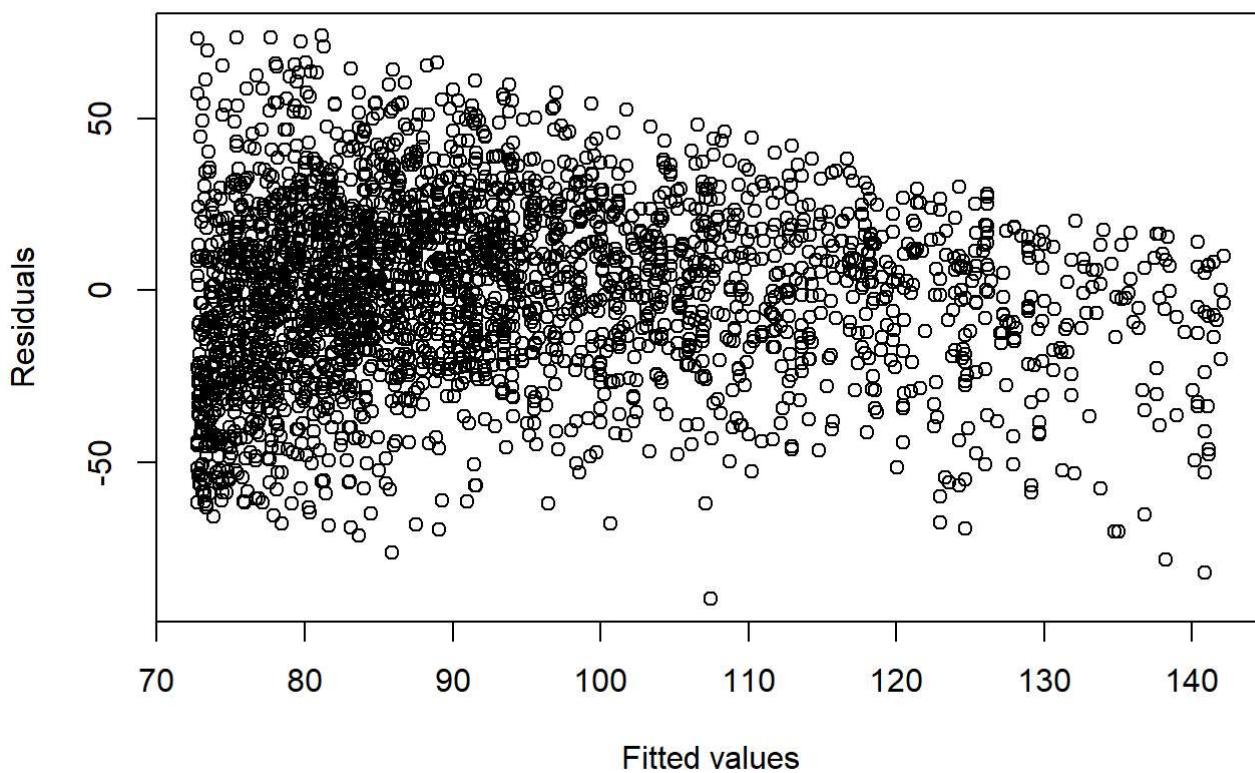


```
check_regression_assumptions(movies_fourth_root, "adj_rev", T, 1/4)
```

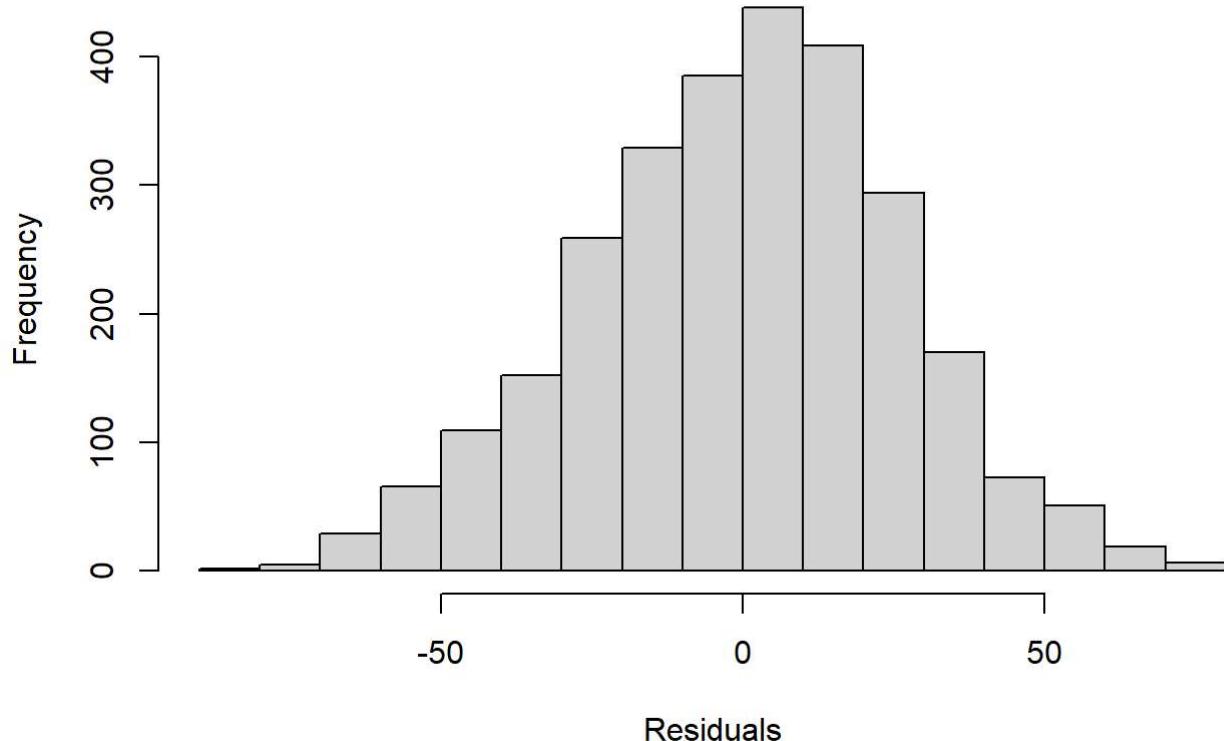
Plot of adj_rev vs adj_bud Without Outliers



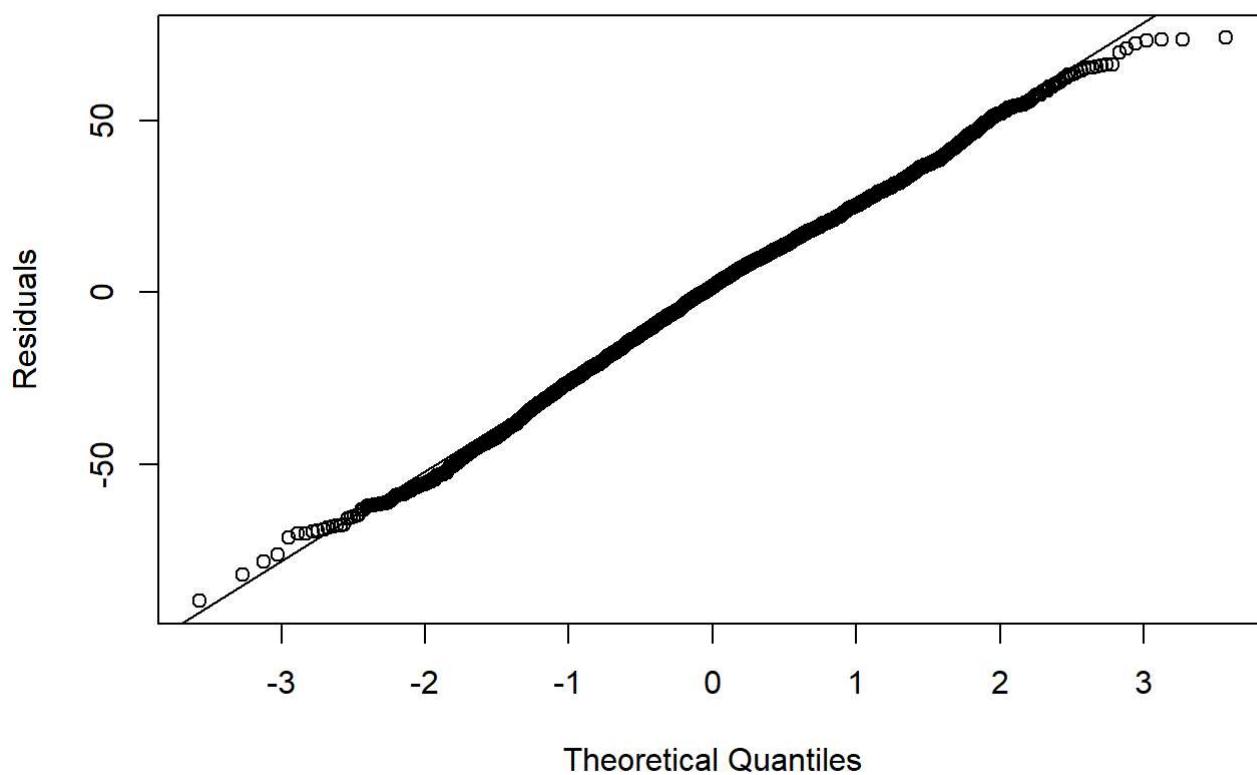
Residuals vs Fitted Values for adj_rev vs adj_bud Without Outliers



Histogram of the Residuals for adj_rev vs adj_bud Without Outliers



Normal Q-Q plot of the Residuals for adj_rev vs adj_bud Without Outliers



#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.