What makes a Hollywood movie profitable?

Jimmy Ng

November 14, 2018

Data Preparation

```
> # load packages
> library(plyr)
> library(lubridate)
> library(tidyverse)
>
> # load file from github
> # source from tidytuesday - they have a new dataset coming out every Tuesday
> github <- "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data"
> date <- "2018-10-23"
> csv <- "movie_profit.csv"
> myfile <- paste(github, date, csv, sep = "/")
> df <- readr::read_csv(myfile)
>
> ###################
> ### have a look ###
> ###################
> head(df)
## # A tibble: 6 x 9
##      X1 release_date movie production_budg~ domestic_gross worldwide_gross
##   <int> <chr>        <chr>            <dbl>          <dbl>           <dbl>
## 1     1 6/22/2007    Evan~        175000000      100289690       174131329
## 2     2 7/28/1995    Wate~        175000000       88246220       264246220
## 3     3 5/12/2017    King~        175000000       39175066       139950708
## 4     4 12/25/2013   47 R~        175000000       38362475       151716815
## 5     5 6/22/2018    Jura~        170000000      416769345      1304866322
## 6     6 8/1/2014     Guar~        170000000      333172112       771051335
## # ... with 3 more variables: distributor <chr>, mpaa_rating <chr>,
## #   genre <chr>
> dim(df)  # [1] 3401    9
## [1] 3401    9
> str(df)
## Classes 'tbl_df', 'tbl' and 'data.frame':    3401 obs. of  9 variables:
##  $ X1               : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ release_date     : chr  "6/22/2007" "7/28/1995" "5/12/2017" "12/25/2013" ...
##  $ movie            : chr  "Evan Almighty" "Waterworld" "King Arthur: Legend of the Sword" "47 Ronin
##  $ production_budget: num  1.75e+08 1.75e+08 1.75e+08 1.75e+08 1.70e+08 1.70e+08 1.70e+08 1.70e+08 1
##  $ domestic_gross   : num  1.00e+08 8.82e+07 3.92e+07 3.84e+07 4.17e+08 ...
##  $ worldwide_gross  : num  1.74e+08 2.64e+08 1.40e+08 1.52e+08 1.30e+09 ...
##  $ distributor      : chr  "Universal" "Universal" "Warner Bros." "Universal" ...
##  $ mpaa_rating      : chr  "PG" "PG-13" "PG-13" "PG-13" ...
##  $ genre            : chr  "Comedy" "Action" "Adventure" "Action" ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 9
##   .. ..$ X1                : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ release_date     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ movie            : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ production_budget: list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ domestic_gross   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ worldwide_gross  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ distributor      : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ mpaa_rating      : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ genre            : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
>
> ############
> # clean-up #
> ############
> # check duplication: is there any movie that's duplicated in this data set?
> plyr::count(df, "movie") %>%
+         dplyr::filter(freq >1)
##            movie freq
## 1 Tau ming chong    2
>
> # what is this movie?
> df %>%
+         dplyr::filter(movie == "Tau ming chong") %>%
+         print
## # A tibble: 2 x 9
##      X1 release_date movie production_budg~ domestic_gross worldwide_gross
##   <int> <chr>        <chr>            <dbl>          <dbl>           <dbl>
## 1  2974 4/2/2010     Tau ~          4000000         129078        38899792
## 2  2975 4/2/2010     Tau ~          4000000         129078        38899792
## # ... with 3 more variables: distributor <chr>, mpaa_rating <chr>,
## #   genre <chr>
> # id == 2974, 2975
>
> # let's remove either one of these identical rows
> df <- df %>%
+         dplyr::filter(X1 != 2974)
>
> # is there any movie that has 0 or even negative domestic/worldwide gross?
> df %>%
+         dplyr::filter(domestic_gross <=0 | worldwide_gross <= 0)
## # A tibble: 66 x 9
##       X1 release_date movie production_budg~ domestic_gross worldwide_gross
##    <int> <chr>        <chr>            <dbl>          <dbl>           <dbl>
##  1    31 12/21/2018   Aqua~        160000000              0              0
##  2   229 3/15/2019    Wond~        100000000              0              0
##  3  1031 11/11/2016   USS ~         40000000              0        1641255
##  4  1089 4/14/2017    Quee~         36000000              0        1578543
##  5  1184 3/13/2015    The ~         35000000              0          11106
##  6  1360 12/14/2007   Good~         30000000              0        2717302
##  7  1446 3/17/2015    Acci~         26000000              0         135436
##  8  1567 7/8/2011     Iron~         25000000              0        5297411
##  9  1826 3/31/2004    The ~         20000000              0        5918742
## 10  1827 8/29/2014    Dweg~         20000000              0              0
## # ... with 56 more rows, and 3 more variables: distributor <chr>,
## #   mpaa_rating <chr>, genre <chr>
> # there are 66 of these in this data set (as of Nov 7)
> # some of them have not been released yet, like the Aquaman!
>
> # let's not remove them but create a flag for each of these variables
> df$domestic_flag <- ifelse(df$domestic_gross <=0, 0, 1); sum(df$domestic_flag)
## [1] 3334
> df$worldwide_flag <- ifelse(df$worldwide_gross <=0, 0, 1); sum(df$worldwide_flag)
## [1] 3364
>
> # let's rename the X1 column and rename it as a movie id column
> names(df)[1] <- "movie_id"
>
> # change "date": turn the release_date column as date data type; add release day, month, year
> df <- df %>%
+         mutate(release_date = lubridate::mdy(release_date),
+                release_day = lubridate::wday(release_date,
+                                              week_start = getOption("lubridate.week.start", 1)),
+                release_month = lubridate::month(release_date),
+                release_year = lubridate::year(release_date))
>
> # rescale the production_budget, domestic_gross & worldwide_gross (by dividing 1 million)
> # so that they are easier to read and/or visualize
> df <- df %>%
+         mutate(production_budget = production_budget / 1000000,
+                domestic_gross = domestic_gross / 1000000,
+                worldwide_gross = worldwide_gross / 1000000)
>
> # change mpaa_rating & genre data type from character to factor
> df <- df %>%
+         mutate(mpaa_rating = factor(mpaa_rating,
+                                     levels = c("G", "PG", "PG-13", "R")),
+                genre = as.factor(genre))
>
> # complete.cases - remove all NA's
> dfComplete <- df[complete.cases(df), ]
>
> # let's look at the clean data set
> head(dfComplete)
## # A tibble: 6 x 14
##   movie_id release_date movie production_budg~ domestic_gross
##      <int> <date>       <chr>             <dbl>          <dbl>
## 1        1 2007-06-22   Evan~              175          100.
## 2        2 1995-07-28   Wate~              175           88.2
## 3        3 2017-05-12   King~              175           39.2
## 4        4 2013-12-25   47 R~              175           38.4
## 5        5 2018-06-22   Jura~              170          417.
## 6        6 2014-08-01   Guar~              170          333.
## # ... with 9 more variables: worldwide_gross <dbl>, distributor <chr>,
## #   mpaa_rating <fct>, genre <fct>, domestic_flag <dbl>,
## #   worldwide_flag <dbl>, release_day <dbl>, release_month <dbl>,
## #   release_year <dbl>
> dim(dfComplete)  # [1] 3230   14
## [1] 3230   14
> str(dfComplete)
## Classes 'tbl_df', 'tbl' and 'data.frame':    3230 obs. of  14 variables:
##  $ movie_id         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ release_date     : Date, format: "2007-06-22" "1995-07-28" ...
##  $ movie            : chr  "Evan Almighty" "Waterworld" "King Arthur: Legend of the Sword" "47 Ronin
##  $ production_budget: num  175 175 175 175 170 170 170 170 170 170 ...
##  $ domestic_gross   : num  100.3 88.2 39.2 38.4 416.8 ...
##  $ worldwide_gross  : num  174 264 140 152 1305 ...
##  $ distributor      : chr  "Universal" "Universal" "Warner Bros." "Universal" ...
##  $ mpaa_rating      : Factor w/ 4 levels "G","PG","PG-13",..: 2 3 3 3 3 3 3 3 3 1 ...
##  $ genre            : Factor w/ 5 levels "Action","Adventure",..: 3 1 2 1 1 1 1 1 2 2 ...
##  $ domestic_flag    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ worldwide_flag   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ release_day      : num  5 5 5 3 5 5 5 5 5 3 ...
##  $ release_month    : num  6 7 5 12 6 8 5 4 7 11 ...
##  $ release_year     : num  2007 1995 2017 2013 2018 ...
```

Part 1 - Introduction

The Hollywood movie industry is a multi-billion dollar industry that is so popular and influential worldwide. The industry makes a long-lasting contribution to our pop culture. A successful movie possesses the ability to capture and reshape the audiences' imagination and identities for years. It can effectively carry its messages to its audiences across time and space without restricting to any language or cultural relevancy. It's such an extraordinary industry; however, like all other industries, it is also driven by monetary gains and financial risks.

Essentially, what makes a movie profitable? Set aside creativity and aesthetic quality, can we predict a movie's box office success purely based on quantitative variables? If so, such equation can have significant impact to transform one of the most lucrative businesses in the world.

Part 2 - Data

Data collection: the data set (csv) is cloned from github, and it is complied from a social science project "tidytuesday" - a weekly social data project in R. As the name described, "tidytuesday" would post a data set on github every Tuesday. This movie data set is from Oct 23, 2018 and the original data is come from numbers.com - a movie industry data website that tracks box office revenue in a systematic, algorithmic way. This movie data set provides various categorical and numerical variables about any major Hollywood released movie since 1936 to present. Box office revenue and production budget are reported in USD and scaled to current monetary value.

Sources of data: https://github.com/rfordatascience/tidytuesday/tree/master/data/2018-10-23 https://thomasmock.netlify.com/post/tidytuesday-a-weekly-social-data-project-in-r/ https://www.the-numbers.com/research-analysis

Cases: each case in this data set represents a movie. After removing one duplicated entry, we are left with 3400 cases from the original data set; however, we need to filter out movies that are not yet released as of this moment (such as the Aquaman won't be released until December 14, 2018 in the United States). In addition, we need to remove cases that have missing values. Finally, we are left with 3230 complete cases (movies) and we are going to split it into "train" and "test" data sets.

Variables: primarily, we want to predict worldwide_gross from production_budget (numeric), mpaa_rating (factor), genre (factor), release_month (we should treat it as factor), and release_year (will group release years and transform the variable as factor).

```
library(plyr)
variable <- lapply(dfComplete, class) %>%
        plyr::ldply(., data.frame)
variable
##                   .id    X..i..
## 1            movie_id   integer
## 2        release_date      Date
## 3               movie character
## 4   production_budget   numeric
## 5      domestic_gross   numeric
## 6     worldwide_gross   numeric
## 7         distributor character
## 8         mpaa_rating    factor
## 9               genre    factor
## 10      domestic_flag   numeric
## 11     worldwide_flag   numeric
## 12        release_day   numeric
## 13      release_month   numeric
## 14       release_year   numeric
```

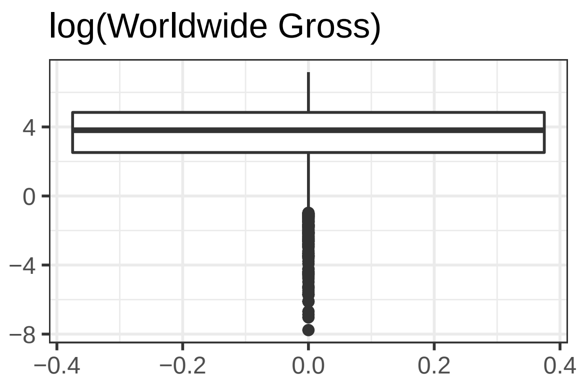
Type of study: this is an observational study with no interference of the box office.

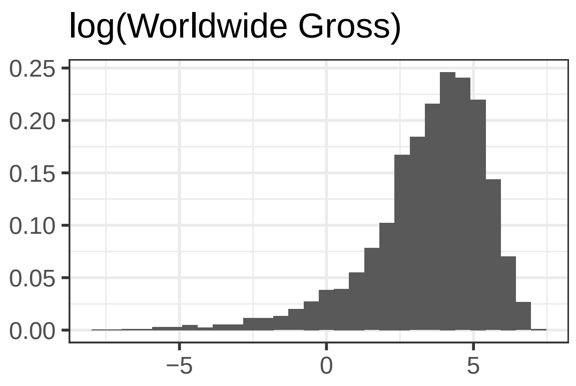
Scope of inference: generalizability / causality - the data is collected through historical record and this is not an experimental design. We can establish correlation, but it is not suitable to justify a causal relation in this observational study (as we do not control the production budget, genre or release month of these movies). However, the result can generate important insights for the industry. For instance, our regression model can explain how much variance that certain variables are held accountable for the box office, while holding everything else constant. We can distill from the data and figure out whether there's anything of significant interest that can impact the box office.

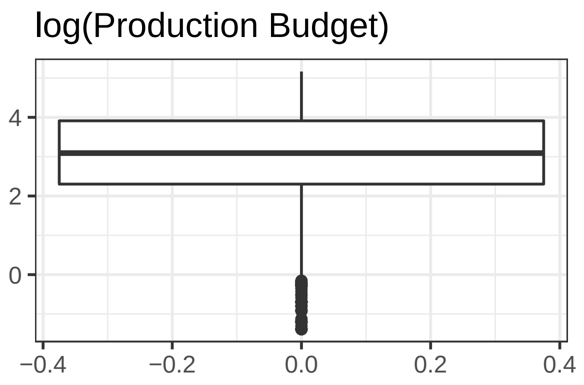
Part 3 - Exploratory data analysis

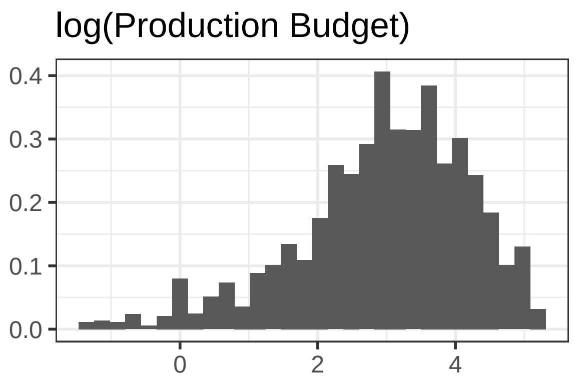
```
################################################
# we are interested in the following variables,

# dependent/predicted variable: worldwide_gross
# independent/predicting variables: production_budget, mpaa_rating, genre, release_month, release_year

###################################################
# let's first look at profit and production_budget
# profit
```

```
dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        dplyr::select(worldwide_gross) %>%
        summary
##  worldwide_gross
##  Min.   :   0.0004
##  1st Qu.:  12.4073
##  Median :  45.1106
##  Mean   :  99.2777
##  3rd Qu.: 126.5470
##  Max.   :1304.8663

# production_budget
dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        dplyr::select(production_budget) %>%
        summary
##  production_budget
##  Min.   :  0.25
##  1st Qu.: 10.00
##  Median : 22.00
##  Mean   : 34.77
##  3rd Qu.: 50.00
##  Max.   :175.00

#################################################################################
# there are many outliners in both profit and production_budget (not surprising)
# let's do "log" transformation for both variables
dfComplete <- dfComplete %>%
        mutate(wg.log = log(worldwide_gross),
               pb.log = log(production_budget))

###########################################
# take a look of them after transformation
library(gridExtra)
##
## Attaching package: 'gridExtra'
## The following object is masked from 'package:dplyr':
##
##     combine

## worldwide_gross ##
box.wg <- dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        ggplot(data = ., aes(y = wg.log)) +
        geom_boxplot() +
        ggtitle("log(Worldwide Gross)") +
        theme_bw() +
        labs(x = "", y = "")

hist.wg <- dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        ggplot(data = ., aes(x = wg.log, stat(density))) +
        geom_histogram() +
        ggtitle("log(Worldwide Gross)") +
        theme_bw() +
        labs(x = "", y = "")

## production_budget ##
box.pb <- dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        ggplot(data = ., aes(y = pb.log)) +
        geom_boxplot() +
        ggtitle("log(Production Budget)") +
        theme_bw() +
        labs(x = "", y = "")

hist.pb <- dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        ggplot(data = ., aes(x = pb.log, stat(density))) +
        geom_histogram() +
        ggtitle("log(Production Budget)") +
        theme_bw() +
        labs(x = "", y = "")

gridExtra::grid.arrange(box.wg,
                        hist.wg,
                        box.pb,
                        hist.pb,
                        ncol = 2)
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



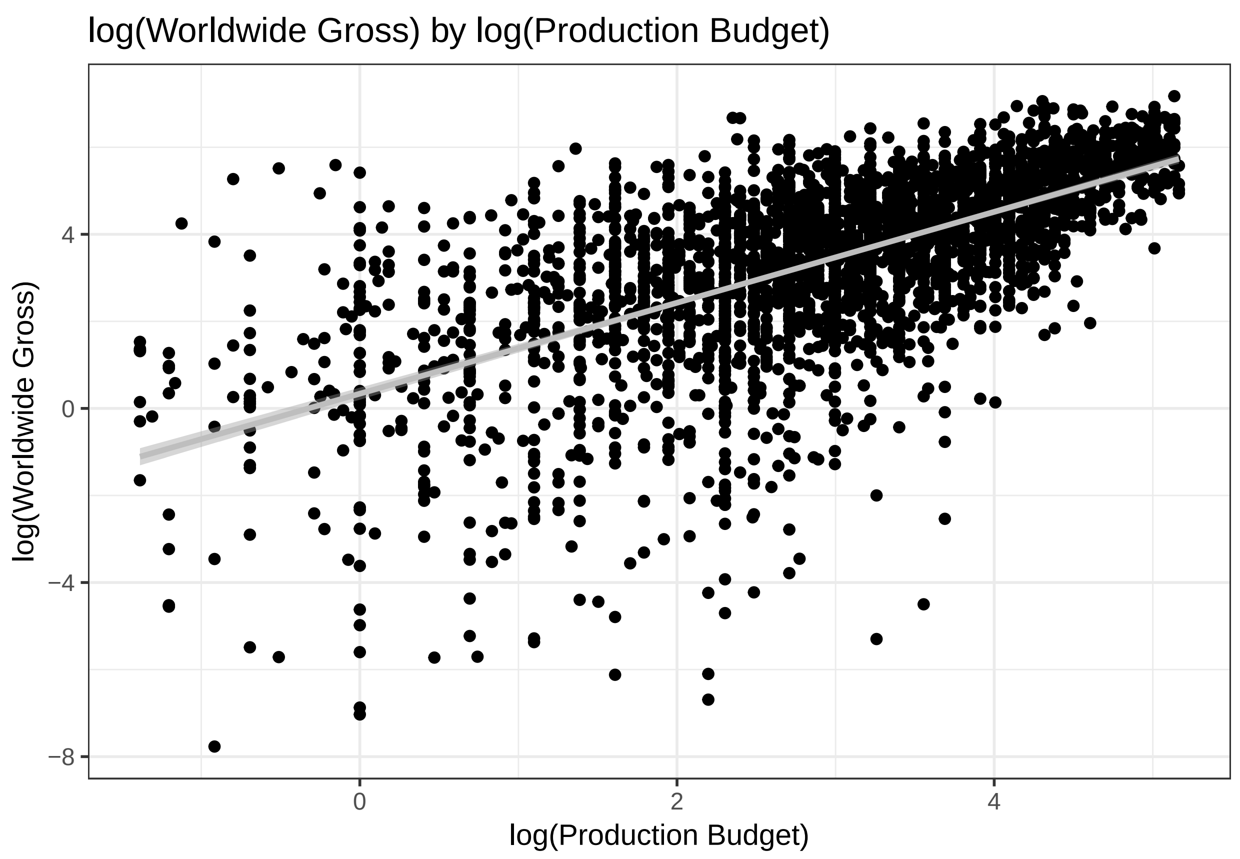




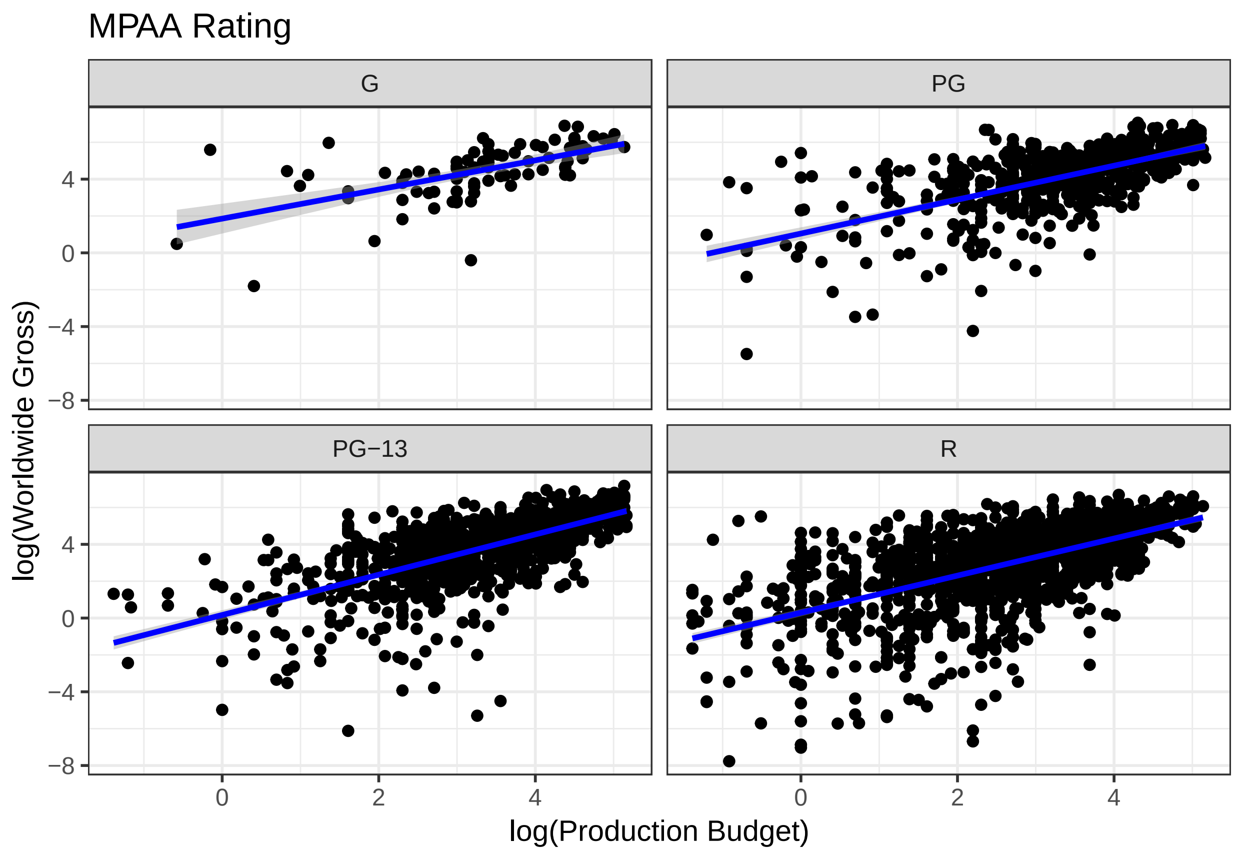


```
##########################################
# let's look at the categorical variables
dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        dplyr::select(mpaa_rating, genre, release_month, release_year) %>%
        lapply(., ftable)
## $mpaa_rating
##      G   PG PG-13    R
##
##     84  560  1082 1476
##
## $genre
##   Action Adventure Comedy Drama Horror
##
##      532       465    769  1172    264
##
## $release_month
##    1   2   3   4   5   6   7   8   9  10  11  12
##
##  181 217 267 230 211 286 258 276 282 328 305 361
##
## $release_year
##   1936 1939 1940 1942 1943 1953 1954 1956 1957 1959 1960 1962 1963 1964 1965 1967 1968 1969 1972 1973
##
##      1    1    1    1    1    1    1    1    1    1    1    2    2    3    3    1    1    1    1    4
```

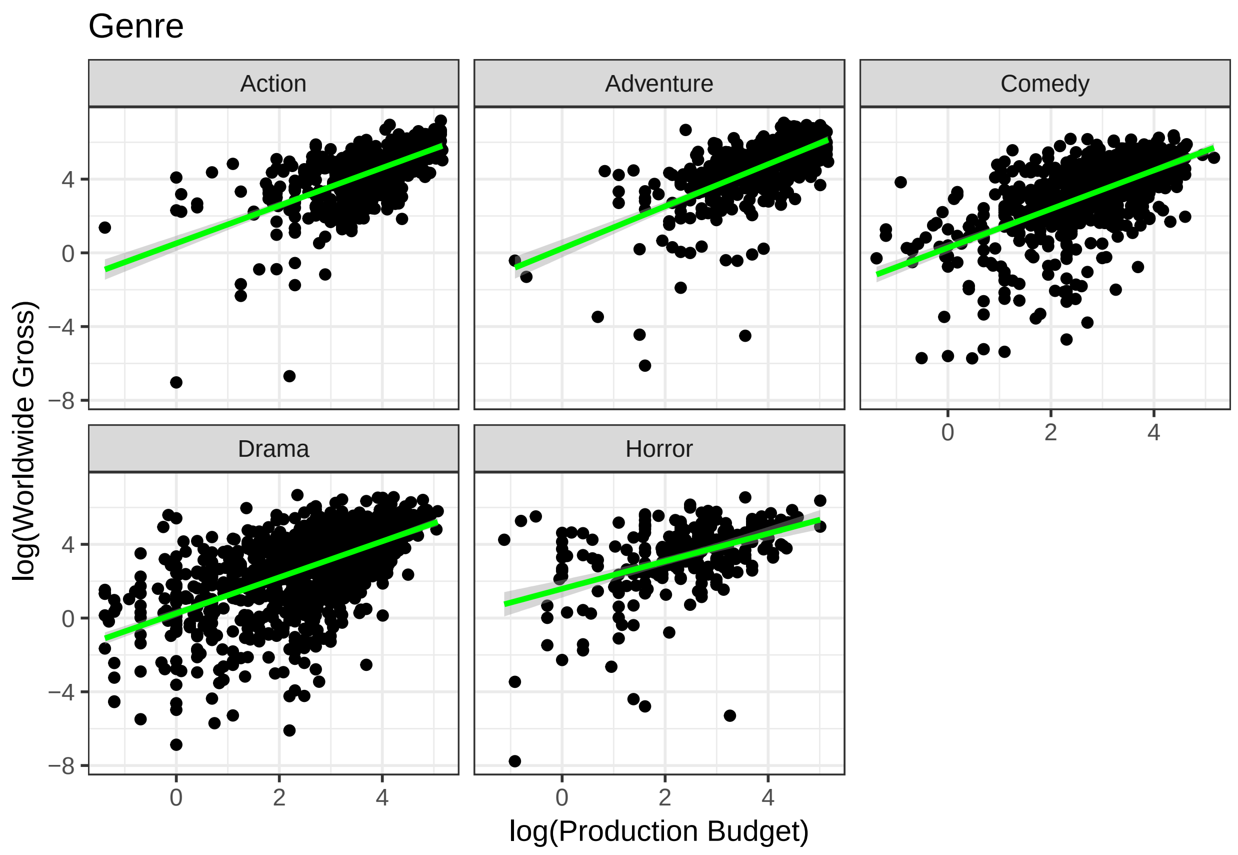
```r
# we already transformed mpaa_rating and genre from character to factor
# now let's do the same for release_month and release_year, and let's call it release_decade
dfComplete <- dfComplete %>%
        mutate(release_month = as.factor(release_month),
               release_decade =
                       if_else(release_year < 1980, "1970s or earlier",
                               if_else(release_year >= 1980 & release_year < 1990, "1980s",
                                       if_else(release_year >= 1990 & release_year < 2000, "1990s",
                                               if_else(release_year >= 2000 & release_year < 2010, "200
                                                       if_else(release_year >= 2010, "2010s", "n/a"))))
               release_decade = as.factor(release_decade))

# let's look at these categorical variables again
dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        dplyr::select(mpaa_rating, genre, release_month, release_decade) %>%
        lapply(., ftable)
## $mpaa_rating
##     G   PG PG-13    R
##
##    84  560  1082 1476
##
## $genre
##  Action Adventure Comedy Drama Horror
##
##     532       465    769  1172    264
##
## $release_month
##    1   2   3   4   5   6   7   8   9  10  11  12
##
##  181 217 267 230 211 286 258 276 282 328 305 361
##
## $release_decade
##  1970s or earlier 1980s 1990s 2000s 2010s
##
##                59   142   588  1375  1038

#################################################################
# let's visualize the data - worldwide_gross ~ production_budget
# let's focus more on wg.log (y-axis) vs pb.log (x-axis) by different categorical variables

# wg.log vs pb.log
dfComplete %>%
    dplyr::filter(worldwide_flag == 1) %>%
    ggplot(data = ., aes(x = pb.log, y = wg.log)) +
    geom_point() +
    geom_smooth(method = "lm", col = "grey", se = T) +
    ggtitle("log(Worldwide Gross) by log(Production Budget)") +
    labs(x = "log(Production Budget)", y = "log(Worldwide Gross)") +
    theme_bw()
```



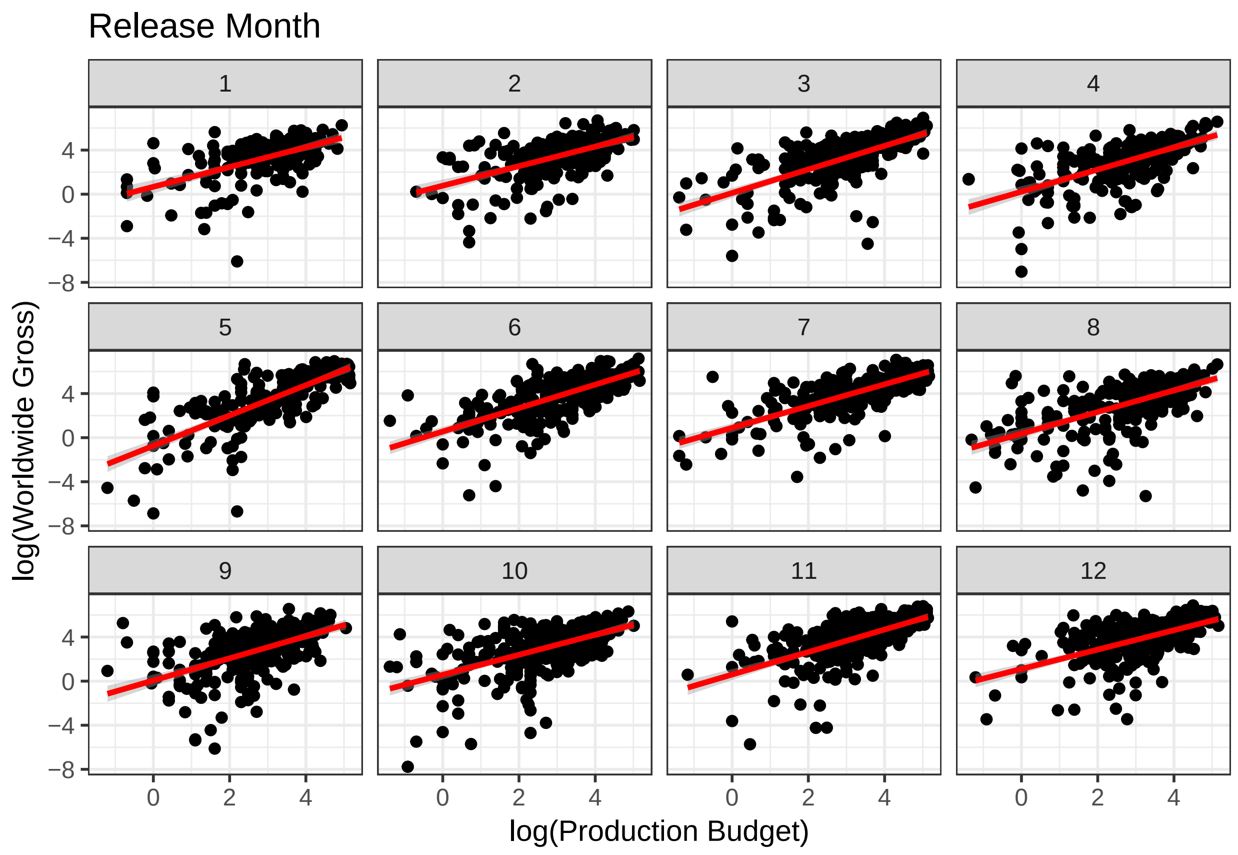
```
# wg.log vs pb.log by mpaa_rating
dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        ggplot(data = ., aes(x = pb.log, y = wg.log)) +
        # ggplot(data = ., aes(x = production_budget, y = worldwide_gross)) +
        geom_point() +
        geom_smooth(method = "lm", col = "blue", se = T) +
        facet_wrap(~mpaa_rating) +
        ggtitle("MPAA Rating") +
        labs(x = "log(Production Budget)", y = "log(Worldwide Gross)") +
        theme_bw()
```



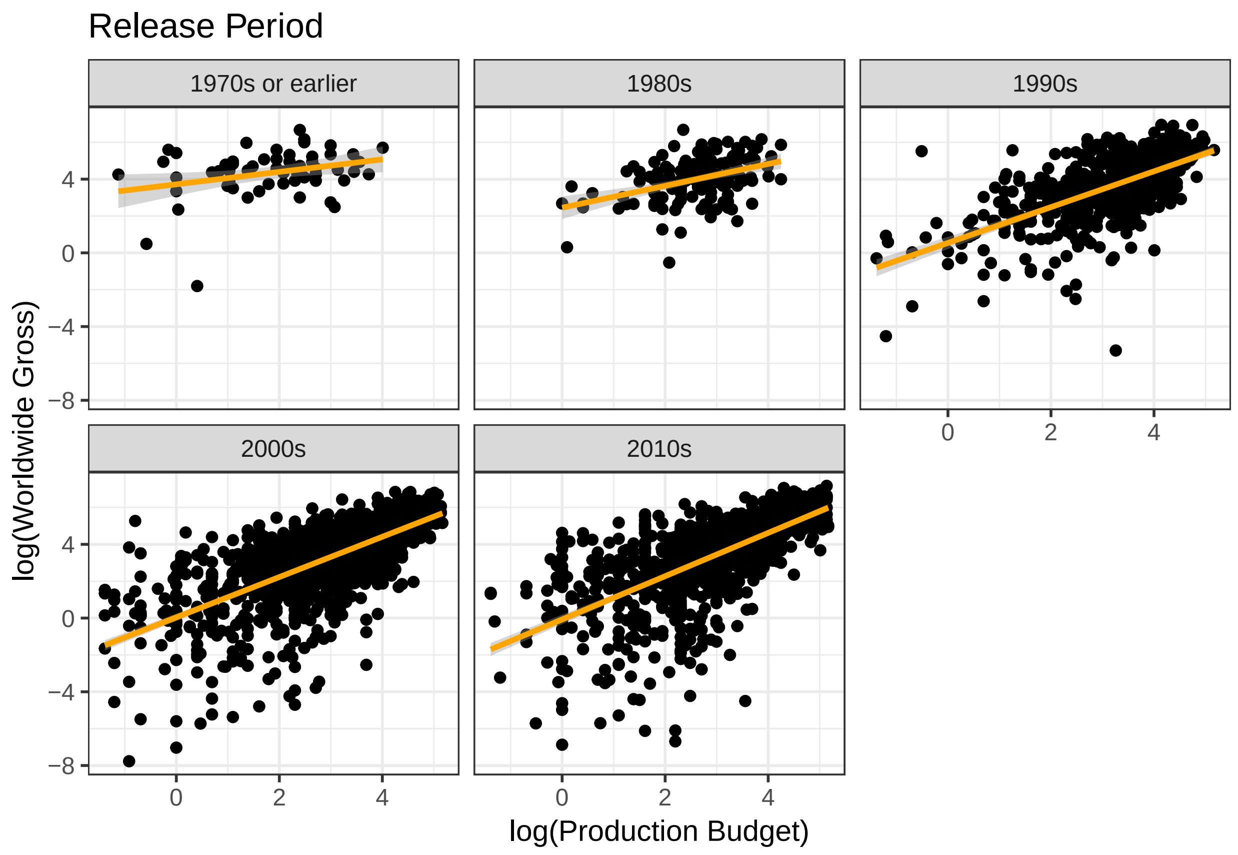
```
# wg.log vs pb.log by genre
dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        ggplot(data = ., aes(x = pb.log, y = wg.log)) +
        geom_point() +
        geom_smooth(method = "lm", col = "green", se = T) +
        facet_wrap(~genre) +
        ggtitle("Genre") +
        labs(x = "log(Production Budget)", y = "log(Worldwide Gross)") +
        theme_bw()
```



```r
# wg.log vs pb.log by release_month
dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        ggplot(data = ., aes(x = pb.log, y = wg.log)) +
        geom_point() +
        geom_smooth(method = "lm", col = "red", se = T) +
        facet_wrap(~release_month) +
        ggtitle("Release Month") +
        labs(x = "log(Production Budget)", y = "log(Worldwide Gross)") +
        theme_bw()
```



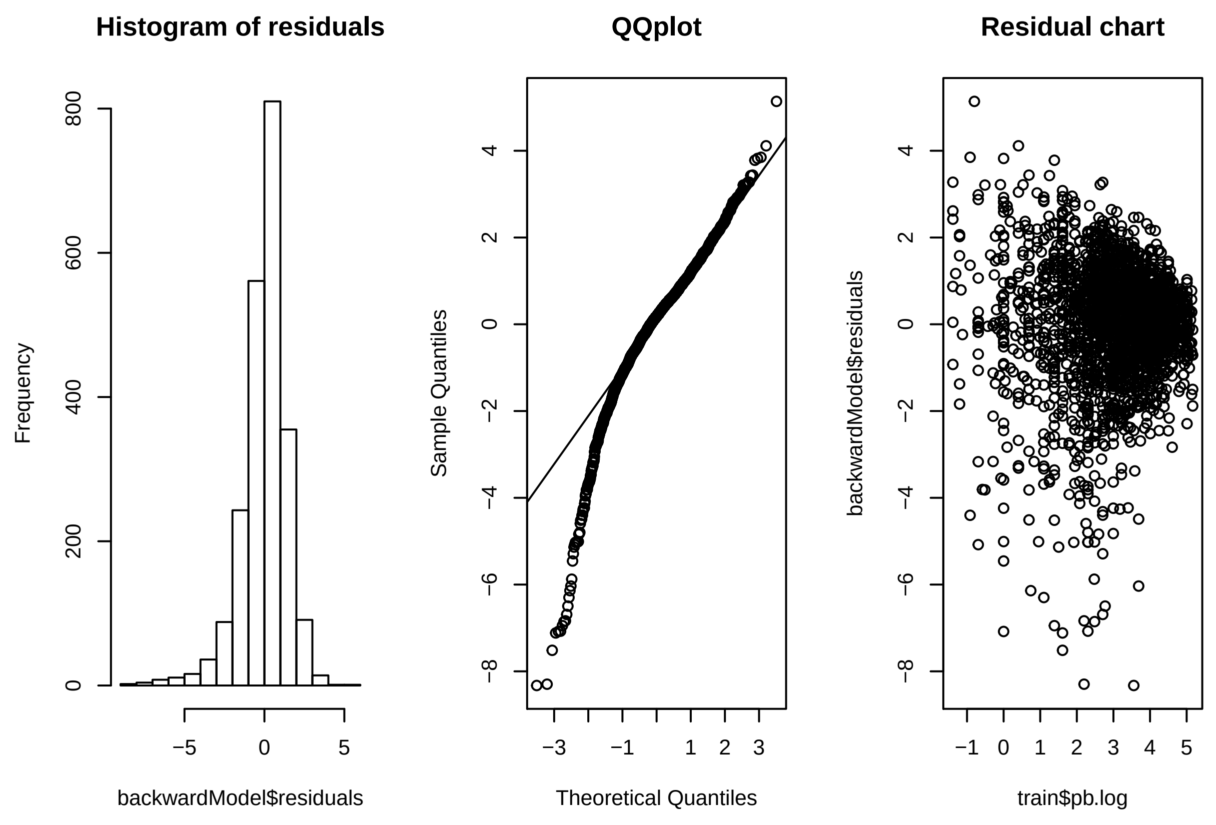
```
# wg.log vs pb.log  by release_decade
dfComplete %>%
        dplyr::filter(worldwide_flag == 1) %>%
        ggplot(data = ., aes(x = pb.log, y = wg.log)) +
        geom_point() +
        geom_smooth(method = "lm", col = "orange", se = T) +
        facet_wrap(~release_decade) +
        ggtitle("Release Period") +
        labs(x = "log(Production Budget)", y = "log(Worldwide Gross)") +
        theme_bw()
```



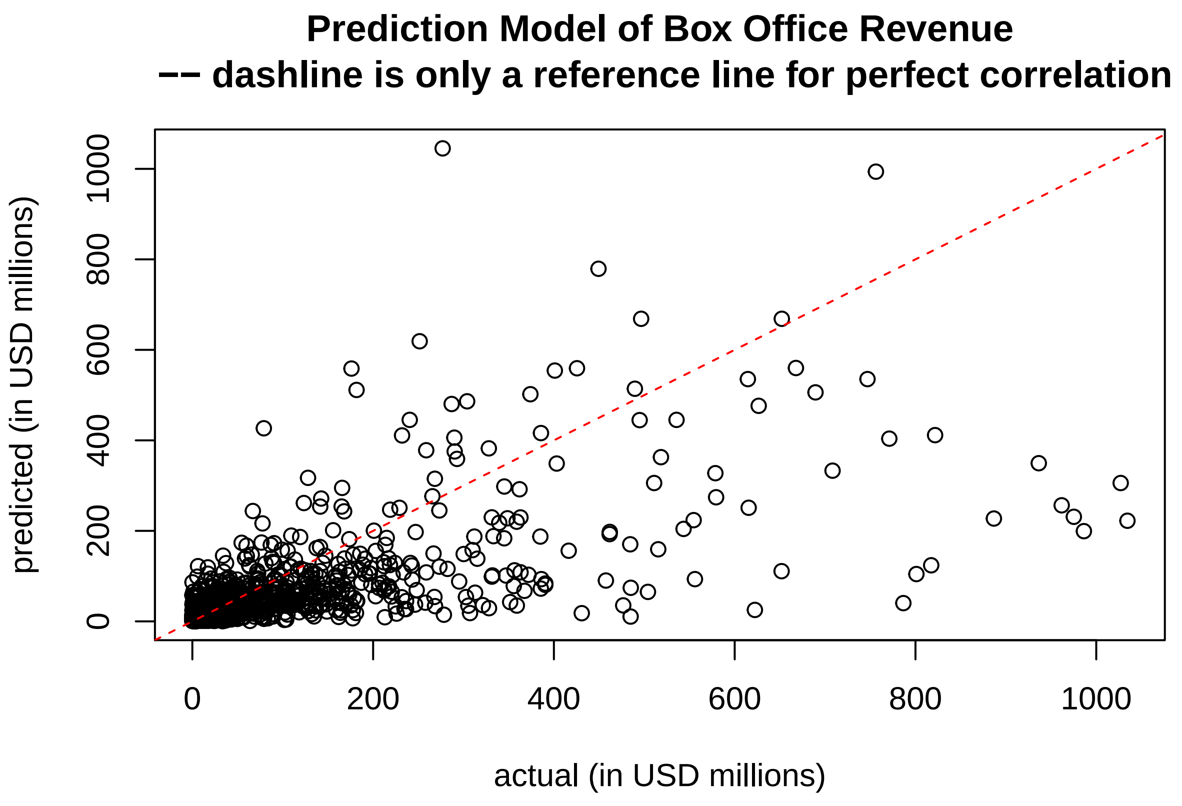
Part 4 - Inference

```
################################################################
#################### regression model #########################
################################################################

################################
# split the data, train + test
set.seed(1234)
dfCompleteFlag <- dfComplete %>%
        dplyr::filter(worldwide_flag == 1)
partition <- sample(x = nrow(dfCompleteFlag),
                    size = nrow(dfCompleteFlag) * 0.7)
train <- dfCompleteFlag[partition, ]
test <- dfCompleteFlag[-partition, ]

###################
# build full model
# build interaction between log(production_budget) with everything else
library(broom)
fullModel <- lm(wg.log ~ pb.log * (mpaa_rating + genre + release_month + release_decade),
                data = train)
broom::tidy(fullModel) %>%
        dplyr::filter(p.value < .05) %>%
        dplyr::arrange(p.value)
## # A tibble: 11 x 5
##    term                       estimate std.error statistic  p.value
##    <chr>                         <dbl>     <dbl>     <dbl>    <dbl>
##  1 release_decade2010s          -3.50      0.478     -7.32 3.42e-13
##  2 release_decade2000s          -3.20      0.470     -6.81 1.23e-11
##  3 release_decade1990s          -2.51      0.503     -4.99 6.51e- 7
##  4 (Intercept)                   3.50      0.771      4.54 5.85e- 6
##  5 pb.log:release_decade2010s    0.889     0.206      4.31 1.68e- 5
##  6 genreHorror                   1.47      0.397      3.70 2.20e- 4
##  7 pb.log:release_decade2000s    0.745     0.204      3.65 2.72e- 4
##  8 pb.log:genreHorror           -0.349     0.124     -2.82 4.86e- 3
##  9 pb.log:release_month5         0.391     0.140      2.80 5.22e- 3
## 10 release_month5               -1.21      0.441     -2.74 6.10e- 3
## 11 pb.log:release_decade1990s    0.561     0.211      2.66 7.96e- 3

# try backward stepwise regression
# calculate AIC
library(MASS)
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
backwardModel <- MASS::stepAIC(fullModel, direction = "backward")
## Start:  AIC=1699.86
## wg.log ~ pb.log * (mpaa_rating + genre + release_month + release_decade)
##
##                          Df Sum of Sq    RSS    AIC
## - pb.log:mpaa_rating      3     2.549 4594.9 1695.1
## <none>                                4592.4 1699.9
## - pb.log:release_month   11    52.868 4645.2 1703.5
## - pb.log:genre            4    26.462 4618.8 1704.7
## - pb.log:release_decade   4    86.661 4679.0 1733.8
##
## Step:  AIC=1695.1
## wg.log ~ pb.log + mpaa_rating + genre + release_month + release_decade +
##     pb.log:genre + pb.log:release_month + pb.log:release_decade
##
##                          Df Sum of Sq    RSS    AIC
## <none>                                4594.9 1695.1
## - pb.log:release_month   11    52.094 4647.0 1698.4
## - pb.log:genre            4    29.044 4624.0 1701.2
## - mpaa_rating             3    26.922 4621.8 1702.2
## - pb.log:release_decade   4    92.860 4687.8 1731.9
summary(backwardModel)
##
## Call:
## lm(formula = wg.log ~ pb.log + mpaa_rating + genre + release_month +
##     release_decade + pb.log:genre + pb.log:release_month + pb.log:release_decade,
##     data = train)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -8.3263 -0.6427  0.1843  0.8531  5.1379
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  3.77353    0.61160   6.170 8.11e-10 ***
## pb.log                       0.32226    0.23256   1.386 0.165987
## mpaa_ratingPG               -0.26336    0.20578  -1.280 0.200741
## mpaa_ratingPG-13            -0.44083    0.21659  -2.035 0.041937 *
## mpaa_ratingR                -0.59075    0.21826  -2.707 0.006850 **
## genreAdventure              -0.50780    0.45233  -1.123 0.261722
## genreComedy                  0.40510    0.34575   1.172 0.241462
## genreDrama                   0.13490    0.32814   0.411 0.681034
## genreHorror                  1.47526    0.39228   3.761 0.000174 ***
## release_month2               0.21969    0.48473   0.453 0.650429
## release_month3              -0.29120    0.44108  -0.660 0.509204
## release_month4               0.06971    0.43718   0.159 0.873323
## release_month5              -1.22402    0.44066  -2.778 0.005521 **
## release_month6               0.15553    0.43443   0.358 0.720375
## release_month7               0.43290    0.44801   0.966 0.334020
## release_month8               0.19622    0.42781   0.459 0.646525
## release_month9              -0.63921    0.43607  -1.466 0.142834
## release_month10             -0.19031    0.41317  -0.461 0.645128
## release_month11             -0.14537    0.47662  -0.305 0.760395
## release_month12              0.18468    0.45922   0.402 0.687613
## release_decade1980s         -0.91305    0.63722  -1.433 0.152037
## release_decade1990s         -2.52400    0.46393  -5.440 5.90e-08 ***
## release_decade2000s         -3.20170    0.43005  -7.445 1.39e-13 ***
## release_decade2010s         -3.50889    0.43409  -8.083 1.03e-15 ***
## pb.log:genreAdventure        0.08232    0.11467   0.718 0.472910
## pb.log:genreComedy          -0.13643    0.09568  -1.426 0.154035
## pb.log:genreDrama           -0.12622    0.08963  -1.408 0.159197
## pb.log:genreHorror          -0.35291    0.12172  -2.899 0.003777 **
## pb.log:release_month2       -0.04168    0.15463  -0.270 0.787542
## pb.log:release_month3        0.05519    0.14232   0.388 0.698220
## pb.log:release_month4       -0.02368    0.14592  -0.162 0.871127
## pb.log:release_month5        0.39535    0.13970   2.830 0.004698 **
## pb.log:release_month6        0.03776    0.13728   0.275 0.783301
## pb.log:release_month7       -0.02149    0.14023  -0.153 0.878231
## pb.log:release_month8       -0.04839    0.13956  -0.347 0.728824
## pb.log:release_month9        0.14558    0.14460   1.007 0.314152
## pb.log:release_month10       0.01456    0.13823   0.105 0.916097
## pb.log:release_month11       0.10648    0.14856   0.717 0.473604
## pb.log:release_month12       0.04843    0.14545   0.333 0.739175
## pb.log:release_decade1980s  0.21424    0.26015   0.824 0.410306
## pb.log:release_decade1990s  0.56325    0.20076   2.806 0.005066 **
## pb.log:release_decade2000s  0.74445    0.19407   3.836 0.000129 ***
## pb.log:release_decade2010s  0.89066    0.19480   4.572 5.09e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.446 on 2198 degrees of freedom
## Multiple R-squared:  0.4988, Adjusted R-squared:  0.4892
## F-statistic: 52.09 on 42 and 2198 DF,  p-value: < 2.2e-16

#########################
# (1) linearity, (2) nearly normal residuals, (3) homoscedasticity

# linear scatter plot (checked already from above EDA)
# a strong positive correlation between production budget and worldwide gross
# however, there are patterns hidden underneath the two variables
# for examples,
# PG, PG-13 generate the most revenue
# Action, Adventure draw the biggest box office, but
# Horror is the most profitable movie genre
# summer (May, June & July) is always the best time to roll out blockbuster movies!
# therefore, we are building interaction between log(production_budget) with everything else

par(mfrow = c(1, 3))
# hist of residuals
hist(backwardModel$residuals, main = "Histogram of residuals")

# qqplot
qqnorm(backwardModel$residuals, main = "QQplot")
qqline(backwardModel$residuals)

# residual plot
plot(backwardModel$residuals ~ train$pb.log, main = "Residual chart")
```

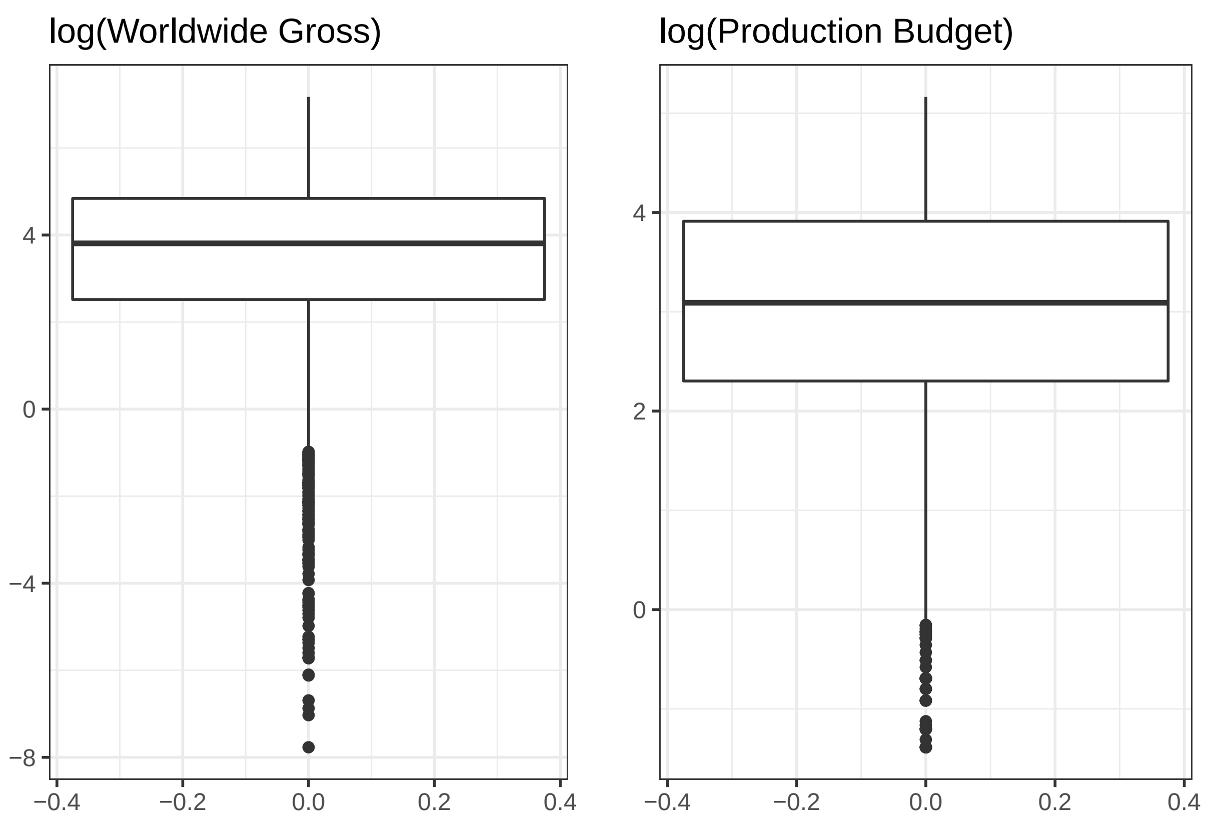


```
par(mfrow = c(1, 1))

######################
# prediction accuracy
# apply model to test data set
testModel <- predict(backwardModel, test)

# convert back to original value
testModel <- exp(testModel)
test$predicted <- testModel

# visualize the prediction
plot(test$predicted ~ test$worldwide_gross,
     main = "Prediction Model of Box Office Revenue\n -- dashline is only a reference line for perfect c
     xlab = "actual (in USD millions)",
     ylab = "predicted (in USD millions)")
abline(a = 0, b = 1, col = "red", lty = 2)
```

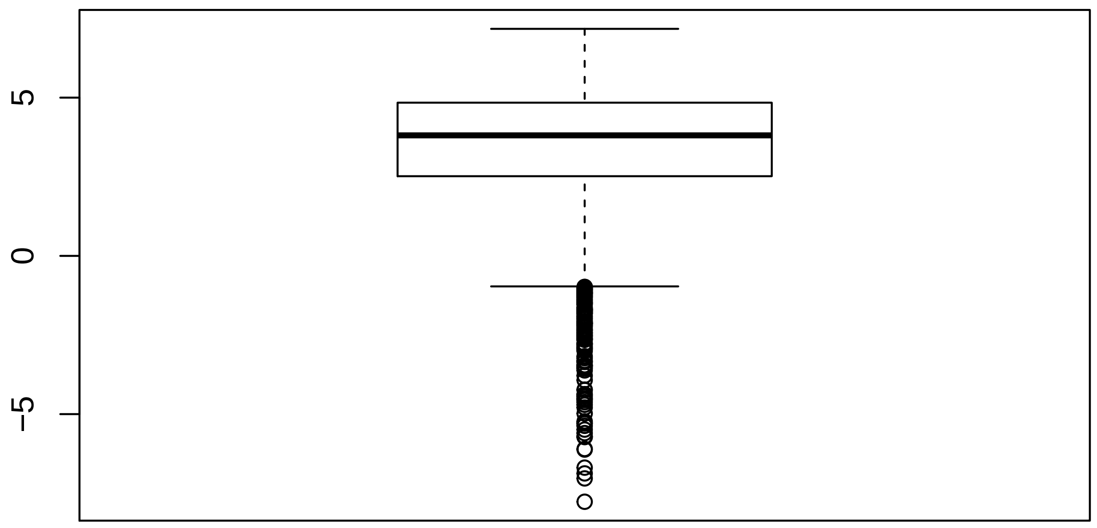


```
# calculate RMSE (Root Mean Square Error)
library(Metrics)
rmse(actual = test$worldwide_gross, predicted = test$predicted)
## [1] 123.7525

####################
##### comments #####
```

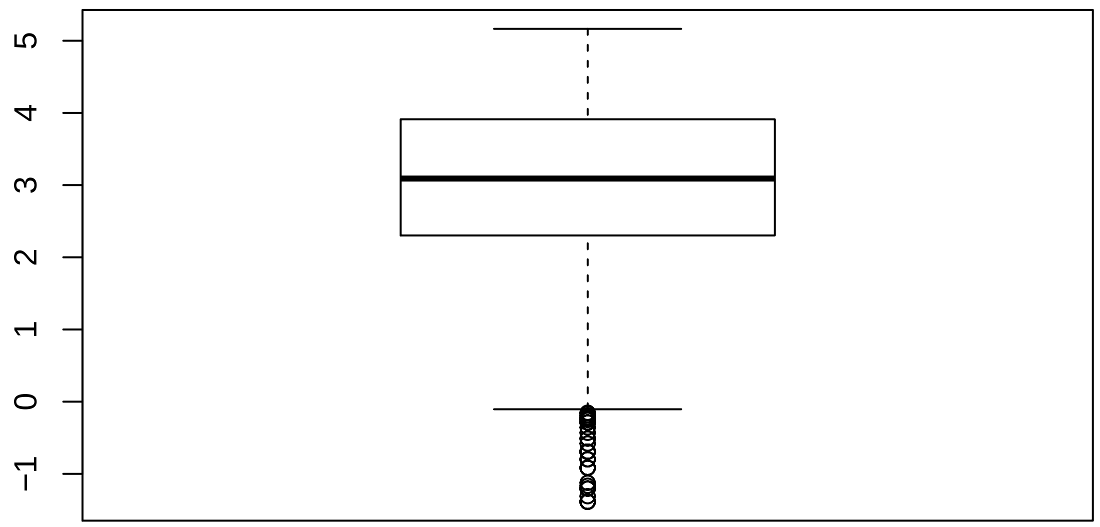
```
####################
# residuals heavily skewed to the left from the histogram
# qqplot does not support normality of the residuals either
# the final plot (actual vs. predicted) shows majority of the movies are underestimated!
# the RMSE - it is interpreted as how far on an average, the residuals are from zero
# It nullifies squared effect of MSE and provides the result in original units as data
# Let's try to minimize it!



#########################################################################
#################### rebuild regression model ########################
#########################################################################

##########################
# let's rebuild the model
# by first removing all the outliners from the lower end of the sprectum

# recall from above EDA,
gridExtra::grid.arrange(box.wg, box.pb, ncol = 2)
```



```
# dplyr::filter outliners
wg.log.out <- boxplot(dfCompleteFlag$wg.log)$out
```



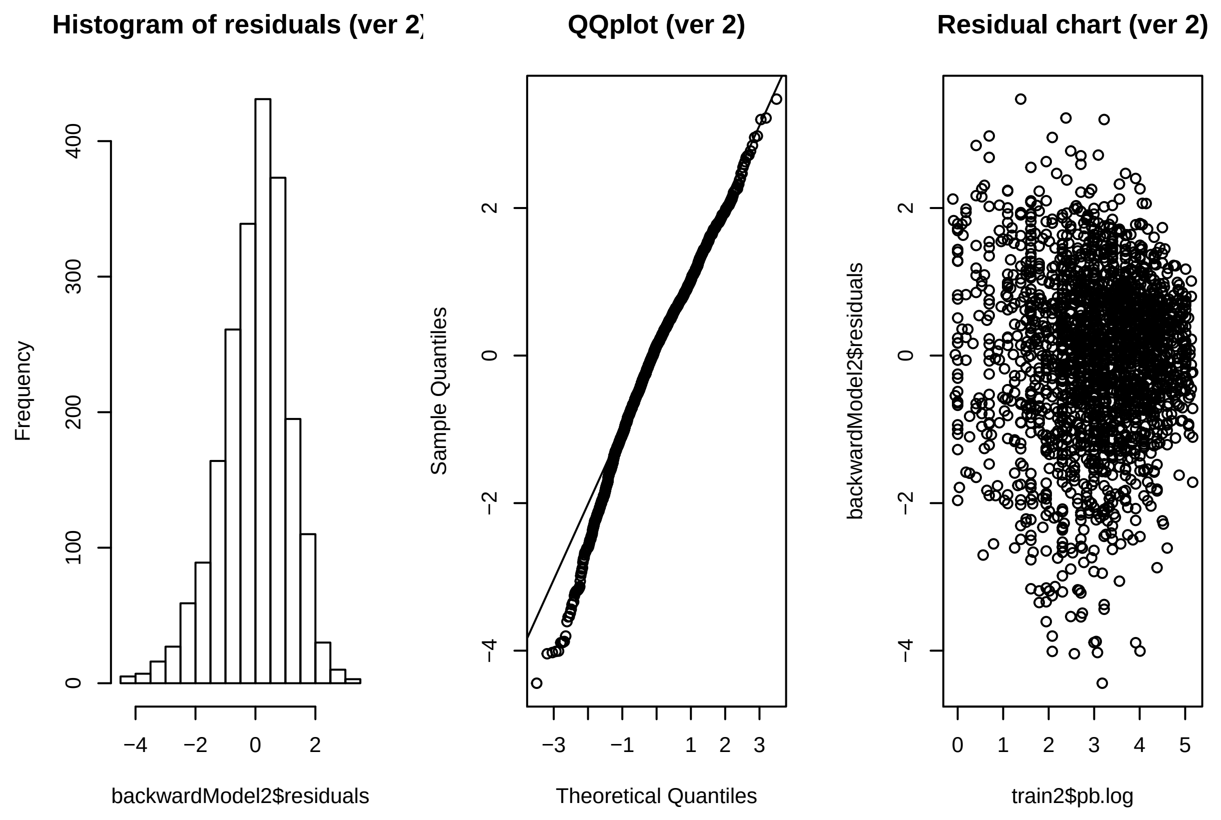
```
pb.log.out <- boxplot(dfCompleteFlag$pb.log)$out
```



```
dfCompleteFlag2 <- dfCompleteFlag %>%
        dplyr::filter(!(wg.log %in% wg.log.out) & !(pb.log %in% pb.log.out))
dim(dfCompleteFlag2)
## [1] 3028   17

#####################################
# split the data, train + test again
set.seed(1234)

partition2 <- sample(x = nrow(dfCompleteFlag2),
                     size = nrow(dfCompleteFlag2) * 0.7)
train2 <- dfCompleteFlag2[partition2, ]
test2 <- dfCompleteFlag2[-partition2, ]

#########################
# build full model again
# build interaction between log(production_budget) with everything else
library(broom)
fullModel2 <- lm(wg.log ~ pb.log * (mpaa_rating + genre + release_month + release_decade),
                data = train2)
broom::tidy(fullModel2) %>%
        dplyr::filter(p.value < .05) %>%
        dplyr::arrange(p.value)
## # A tibble: 16 x 5
##    term                       estimate std.error statistic  p.value
##    <chr>                         <dbl>     <dbl>     <dbl>    <dbl>
##  1 release_decade2000s          -3.07      0.470     -6.54 7.89e-11
##  2 release_decade2010s          -2.98      0.475     -6.29 3.94e-10
##  3 release_decade1990s          -2.89      0.494     -5.84 5.90e- 9
##  4 genreHorror                   1.76      0.353      4.98 6.94e- 7
##  5 (Intercept)                   3.77      0.763      4.95 8.14e- 7
##  6 pb.log:genreHorror           -0.429     0.110     -3.89 1.03e- 4
##  7 pb.log:release_decade2010s    0.801     0.211      3.80 1.48e- 4
##  8 pb.log:release_decade2000s    0.740     0.210      3.53 4.23e- 4
##  9 pb.log:release_month5         0.444     0.128      3.47 5.33e- 4
## 10 pb.log:release_month6         0.415     0.126      3.30 9.88e- 4
## 11 pb.log:release_decade1990s    0.685     0.215      3.19 1.42e- 3
## 12 release_month5               -1.18      0.410     -2.87 4.15e- 3
## 13 release_month9               -1.09      0.389     -2.80 5.20e- 3
## 14 release_month6               -1.08      0.405     -2.66 7.77e- 3
## 15 pb.log:release_month9         0.304     0.127      2.39 1.71e- 2
## 16 release_decade1980s          -1.38      0.601     -2.30 2.15e- 2

# try backward stepwise regression
# calculate AIC
library(MASS)
backwardModel2 <- MASS::stepAIC(fullModel2, direction = "backward")
## Start:  AIC=556.25
## wg.log ~ pb.log * (mpaa_rating + genre + release_month + release_decade)
##
##                          Df Sum of Sq    RSS    AIC
## - pb.log:mpaa_rating     3     2.586 2640.6 552.33
## <none>                               2638.0 556.25
## - pb.log:genre           4    19.524 2657.6 563.88
## - pb.log:release_decade  4    28.429 2666.5 570.96
## - pb.log:release_month  11    49.871 2687.9 573.94
##
## Step:  AIC=552.33
## wg.log ~ pb.log + mpaa_rating + genre + release_month + release_decade +
##     pb.log:genre + pb.log:release_month + pb.log:release_decade
##
##                          Df Sum of Sq    RSS    AIC
## <none>                               2640.6 552.33
## - mpaa_rating            3     9.286 2649.9 553.77
## - pb.log:genre           4    24.069 2664.7 563.55
## - pb.log:release_decade  4    27.875 2668.5 566.58
## - pb.log:release_month  11    50.741 2691.4 570.66
summary(backwardModel2)
##
## Call:
## lm(formula = wg.log ~ pb.log + mpaa_rating + genre + release_month +
##     release_decade + pb.log:genre + pb.log:release_month + pb.log:release_decade,
##     data = train2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4411 -0.6519  0.1369  0.7318  3.4765
##
## Coefficients:
```

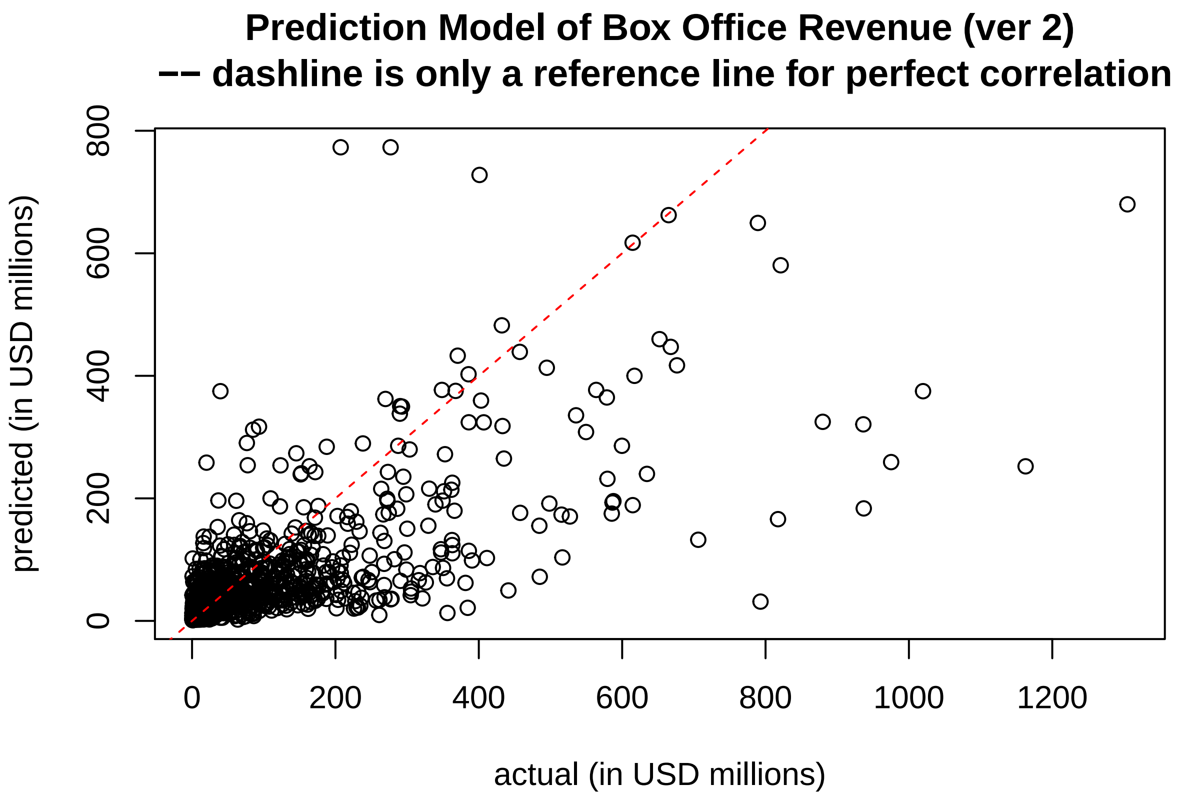
```
##                              Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                  4.199834   0.585550   7.172 1.02e-12 ***
## pb.log                       0.086896   0.231449   0.375 0.707369    
## mpaa_ratingPG               -0.212484   0.160071  -1.327 0.184512    
## mpaa_ratingPG-13            -0.240341   0.169998  -1.414 0.157575    
## mpaa_ratingR                -0.358490   0.170932  -2.097 0.036090 *  
## genreAdventure               0.334600   0.396892   0.843 0.399297    
## genreComedy                  0.405730   0.303566   1.337 0.181518    
## genreDrama                   0.150713   0.283514   0.532 0.595068    
## genreHorror                  1.826400   0.345419   5.287 1.37e-07 ***
## release_month2              -0.161454   0.426318  -0.379 0.704936    
## release_month3              -0.661513   0.407293  -1.624 0.104492    
## release_month4              -0.633595   0.407439  -1.555 0.120082    
## release_month5              -1.184716   0.409730  -2.891 0.003874 ** 
## release_month6              -1.093630   0.403597  -2.710 0.006790 ** 
## release_month7              -0.320219   0.415559  -0.771 0.441047    
## release_month8              -0.053859   0.402007  -0.134 0.893435    
## release_month9              -1.096763   0.389225  -2.818 0.004881 ** 
## release_month10             -0.122081   0.380341  -0.321 0.748259    
## release_month11             -0.002421   0.402371  -0.006 0.995201    
## release_month12              0.209641   0.415631   0.504 0.614039    
## release_decade1980s         -1.217863   0.588462  -2.070 0.038616 *  
## release_decade1990s         -2.721194   0.477239  -5.702 1.35e-08 ***
## release_decade2000s         -2.908479   0.450924  -6.450 1.39e-10 ***
## release_decade2010s         -2.832129   0.455366  -6.219 6.02e-10 ***
## pb.log:genreAdventure       -0.051876   0.099023  -0.524 0.600418    
## pb.log:genreComedy          -0.138774   0.084429  -1.644 0.100395    
## pb.log:genreDrama           -0.149079   0.077444  -1.925 0.054368 .  
## pb.log:genreHorror          -0.450394   0.107578  -4.187 2.95e-05 ***
## pb.log:release_month2        0.087334   0.135668   0.644 0.519818    
## pb.log:release_month3        0.227064   0.129387   1.755 0.079421 .  
## pb.log:release_month4        0.184920   0.132606   1.395 0.163314    
## pb.log:release_month5        0.445021   0.127683   3.485 0.000502 ***
## pb.log:release_month6        0.420986   0.124979   3.368 0.000770 ***
## pb.log:release_month7        0.222733   0.126997   1.754 0.079606 .  
## pb.log:release_month8        0.046284   0.128548   0.360 0.718846    
## pb.log:release_month9        0.306618   0.127271   2.409 0.016075 *  
## pb.log:release_month10       0.029707   0.125870   0.236 0.813444    
## pb.log:release_month11       0.118962   0.125045   0.951 0.341537    
## pb.log:release_month12       0.085241   0.129626   0.658 0.510875    
## pb.log:release_decade1980s  0.331096   0.248733   1.331 0.183293    
## pb.log:release_decade1990s  0.636179   0.210808   3.018 0.002577 ** 
## pb.log:release_decade2000s  0.693053   0.205648   3.370 0.000765 ***
## pb.log:release_decade2010s  0.755448   0.206607   3.656 0.000262 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.128 on 2076 degrees of freedom
## Multiple R-squared:  0.4991, Adjusted R-squared:  0.489 
## F-statistic: 49.26 on 42 and 2076 DF,  p-value: < 2.2e-16

########################################
# regression diagnostic - check again!!
```

```r
# (1) linearity, (2) nearly normal residuals, (3) homoscedasticity

# linear scatter plot (checked already)

par(mfrow = c(1, 3))

# hist of residuals
hist(backwardModel2$residuals, main = "Histogram of residuals (ver 2)")

# qqplot
qqnorm(backwardModel2$residuals, main = "QQplot (ver 2)")
qqline(backwardModel2$residuals)

# residual plot
plot(backwardModel2$residuals ~ train2$pb.log, main = "Residual chart (ver 2)")
```



```r
par(mfrow = c(1, 1))

######################
# prediction accuracy
# apply model to test data set
testModel2 <- predict(backwardModel2, test2)

# convert back to original value
testModel2 <- exp(testModel2)
```

```
test2$predicted <- testModel2

# visualize the prediction
plot(test2$predicted ~ test2$worldwide_gross,
     main = "Prediction Model of Box Office Revenue (ver 2)\n -- dashline is only a reference line for p
     xlab = "actual (in USD millions)",
     ylab = "predicted (in USD millions)")
abline(a = 0, b = 1, col = "red", lty = 2)
```



```
# calculate RMSE (Root Mean Square Error)
library(Metrics)
rmse(actual = test2$worldwide_gross, predicted = test2$predicted)
## [1] 120.768
```

Part 5 - Conclusion

```
summary(backwardModel2)
##
## Call:
## lm(formula = wg.log ~ pb.log + mpaa_rating + genre + release_month +
##     release_decade + pb.log:genre + pb.log:release_month + pb.log:release_decade,
##     data = train2)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max 
## -4.4411 -0.6519  0.1369  0.7318  3.4765 
## 
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                 4.199834   0.585550   7.172 1.02e-12 ***
## pb.log                      0.086896   0.231449   0.375 0.707369    
## mpaa_ratingPG              -0.212484   0.160071  -1.327 0.184512    
## mpaa_ratingPG-13           -0.240341   0.169998  -1.414 0.157575    
## mpaa_ratingR               -0.358490   0.170932  -2.097 0.036090 *  
## genreAdventure              0.334600   0.396892   0.843 0.399297    
## genreComedy                 0.405730   0.303566   1.337 0.181518    
## genreDrama                  0.150713   0.283514   0.532 0.595068    
## genreHorror                 1.826400   0.345419   5.287 1.37e-07 ***
## release_month2             -0.161454   0.426318  -0.379 0.704936    
## release_month3             -0.661513   0.407293  -1.624 0.104492    
## release_month4             -0.633595   0.407439  -1.555 0.120082    
## release_month5             -1.184716   0.409730  -2.891 0.003874 ** 
## release_month6             -1.093630   0.403597  -2.710 0.006790 ** 
## release_month7             -0.320219   0.415559  -0.771 0.441047    
## release_month8             -0.053859   0.402007  -0.134 0.893435    
## release_month9             -1.096763   0.389225  -2.818 0.004881 ** 
## release_month10            -0.122081   0.380341  -0.321 0.748259    
## release_month11            -0.002421   0.402371  -0.006 0.995201    
## release_month12             0.209641   0.415631   0.504 0.614039    
## release_decade1980s        -1.217863   0.588462  -2.070 0.038616 *  
## release_decade1990s        -2.721194   0.477239  -5.702 1.35e-08 ***
## release_decade2000s        -2.908479   0.450924  -6.450 1.39e-10 ***
## release_decade2010s        -2.832129   0.455366  -6.219 6.02e-10 ***
## pb.log:genreAdventure      -0.051876   0.099023  -0.524 0.600418    
## pb.log:genreComedy         -0.138774   0.084429  -1.644 0.100395    
## pb.log:genreDrama          -0.149079   0.077444  -1.925 0.054368 .  
## pb.log:genreHorror         -0.450394   0.107578  -4.187 2.95e-05 ***
## pb.log:release_month2       0.087334   0.135668   0.644 0.519818    
## pb.log:release_month3       0.227064   0.129387   1.755 0.079421 .  
## pb.log:release_month4       0.184920   0.132606   1.395 0.163314    
## pb.log:release_month5       0.445021   0.127683   3.485 0.000502 ***
## pb.log:release_month6       0.420986   0.124979   3.368 0.000770 ***
## pb.log:release_month7       0.222733   0.126997   1.754 0.079606 .  
## pb.log:release_month8       0.046284   0.128548   0.360 0.718846    
## pb.log:release_month9       0.306618   0.127271   2.409 0.016075 *  
## pb.log:release_month10      0.029707   0.125870   0.236 0.813444    
## pb.log:release_month11      0.118962   0.125045   0.951 0.341537    
## pb.log:release_month12      0.085241   0.129626   0.658 0.510875    
## pb.log:release_decade1980s  0.331096   0.248733   1.331 0.183293    
## pb.log:release_decade1990s  0.636179   0.210808   3.018 0.002577 ** 
## pb.log:release_decade2000s  0.693053   0.205648   3.370 0.000765 ***
## pb.log:release_decade2010s  0.755448   0.206607   3.656 0.000262 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.128 on 2076 degrees of freedom
## Multiple R-squared:  0.4991, Adjusted R-squared:  0.489 
## F-statistic: 49.26 on 42 and 2076 DF,  p-value: < 2.2e-16

broom::tidy(backwardModel2) %>%
        dplyr::filter(p.value < .05) %>%
        dplyr::arrange(p.value)
## # A tibble: 17 x 5
##    term                       estimate std.error statistic  p.value
##    <chr>                         <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)                   4.20      0.586      7.17 1.02e-12
##  2 release_decade2000s          -2.91      0.451     -6.45 1.39e-10
##  3 release_decade2010s          -2.83      0.455     -6.22 6.02e-10
##  4 release_decade1990s          -2.72      0.477     -5.70 1.35e- 8
##  5 genreHorror                   1.83      0.345      5.29 1.37e- 7
##  6 pb.log:genreHorror           -0.450     0.108     -4.19 2.95e- 5
##  7 pb.log:release_decade2010s    0.755     0.207      3.66 2.62e- 4
##  8 pb.log:release_month5         0.445     0.128      3.49 5.02e- 4
##  9 pb.log:release_decade2000s    0.693     0.206      3.37 7.65e- 4
## 10 pb.log:release_month6         0.421     0.125      3.37 7.70e- 4
## 11 pb.log:release_decade1990s    0.636     0.211      3.02 2.58e- 3
## 12 release_month5               -1.18      0.410     -2.89 3.87e- 3
## 13 release_month9               -1.10      0.389     -2.82 4.88e- 3
## 14 release_month6               -1.09      0.404     -2.71 6.79e- 3
## 15 pb.log:release_month9         0.307     0.127      2.41 1.61e- 2
## 16 mpaa_ratingR                 -0.358     0.171     -2.10 3.61e- 2
## 17 release_decade1980s          -1.22      0.588     -2.07 3.86e- 2
```

In conclusion, there's a very strong evidence that there's a positive correlation between production budget and worldwide gross. However, we must put it into the right context. It's always about an interaction between production budget AND release month or genre or release decade. Overall, the model is capable of predicting about 50% of the variance (adjusted R-squared: 0.489) with only a handful of predictors. However, after log-transformation and removing outliners, the principle of homoscedasticity still does not seem to sustain. Although ver2 of the linear model looks much better (by comparing the three diagnostic charts with the first attempt), and the AIC as well as RMSE are reduced to smaller numbers, there are still many noises in the data set that cripple the usefulness of the model. Perhaps applying a different type of GLM methodology will be better fit for this data set. In addition, we should have included more movies from the 70s and earlier, as well as 80s and 90s. The data set does not seem to be generated by a random sampling of all historically released movies. Perhaps there's a bias in data collection, i.e. including only high-profile or more well-known (but not necessarily financially successful) movies in this data set, and that skews the result and prevents us from building a useful model that can apply to the general population. A more comprehensive data set will provide tremendous help in building a better model. For example, cast list, directors, number of theatres, movie reviews, etc.

Nevertheless, the result is still interesting and pointing to some unexpected insights. For example, we all think that summer (especially in May and June) is the best time for box office success. However, it is true only for big budget movie. In other words, releaseing a small budget movie in the summer most likely does not guarantee anything (if not resulting in box office flop). That's most likely due to the fact that audiences are more willing to go see movies in the summer, but they only rather spend their money on big budget movies (assuming better quality and CGI effects). Timing AND buget together mean everything. In addition, a surprising insight is movie genre. The most profitable genre turns out to be "Horror". Small budget horror movie is so profitable and that's why Hollywood keeps producing them! It's much safer to bet on a small budget horror movie than a small or big budget comedy or drama.

Finally, since both production budget and worldwide gross are log-transformed, that adds difficulty in interpreting the result. We must remember to convert the number after applying the model to the original data.