Project 1

JOON HYUNG IM

2022-12-04

Define objectives of writeup

I will use the data of 1000 movies from 2006 to 2016 listed on the IMDB site. The data includes variables such as Rank, Title, Genre, Director, Actors, Year, Runtime, Rating, Votes, Revenue, and Metascore. Using this data, I create a model to predict movie ratings.

Data (Munging - EDA stage)

Data inport/gathering

```
data <- read.csv("02. Regression/Regression/data.csv")
imdb <- data
```

Data summarization

```
head(imdb)
```

```
##   Rank                   Title     Genre            Director
## 1    1 Guardians of the Galaxy   Fantasy           James Gunn
## 2    2              Prometheus     Drama         Ridley Scott
## 3    3                   Split  Thriller   M. Night Shyamalan
## 4    4                    Sing Animation Christophe Lourdelet
## 5    5           Suicide Squad   Fantasy           David Ayer
## 6    6          The Great Wall   Fantasy          Yimou Zhang
##                                                                    Actors
## 1                      Chris Pratt, Vin Diesel, Bradley Cooper, Zoe Saldana
## 2    Noomi Rapace, Logan Marshall-Green, Michael Fassbender, Charlize Theron
## 3           James McAvoy, Anya Taylor-Joy, Haley Lu Richardson, Jessica Sula
## 4 Matthew McConaughey,Reese Witherspoon, Seth MacFarlane, Scarlett Johansson
## 5                        Will Smith, Jared Leto, Margot Robbie, Viola Davis
## 6                            Matt Damon, Tian Jing, Willem Dafoe, Andy Lau
##   Year Runtime..Minutes. Rating  Votes Revenue..Millions. Metascore
## 1 2014               121    8.1 757074            333.13        76
## 2 2012               124    7.0 485820            126.46        65
## 3 2016               117    7.3 157606            138.12        62
## 4 2016               108    7.2  60545            270.32        59
## 5 2016               123    6.2 393727            325.02        40
## 6 2016               103    6.1  56036             45.13        42
```

```
str(imdb)
```

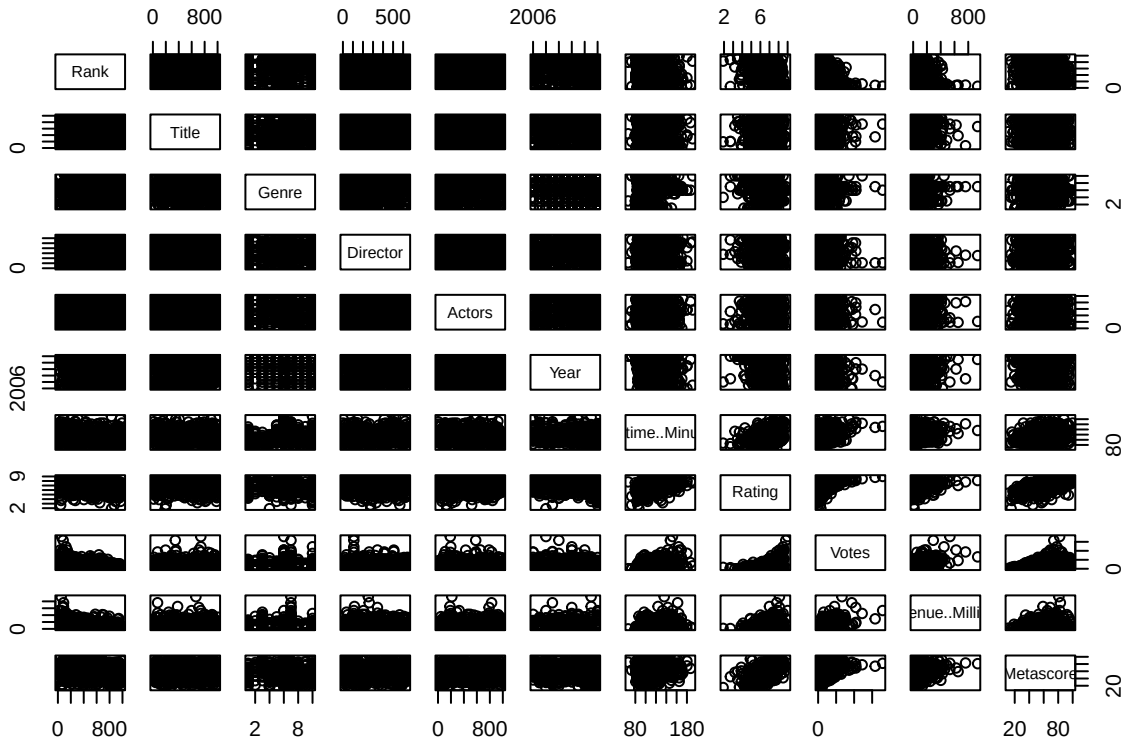
```
## 'data.frame':    1000 obs. of  11 variables:
##  $ Rank              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Title             : chr  "Guardians of the Galaxy" "Prometheus" "Split" "Sing" ...
##  $ Genre             : chr  "Fantasy" "Drama" "Thriller" "Animation" ...
##  $ Director          : chr  "James Gunn" "Ridley Scott" "M. Night Shyamalan" "Christophe Lourdelet"
##  $ Actors            : chr  "Chris Pratt, Vin Diesel, Bradley Cooper, Zoe Saldana" "Noomi Rapace, Lo
##  $ Year              : int  2014 2012 2016 2016 2016 2016 2016 2016 2016 2016 ...
##  $ Runtime..Minutes. : int  121 124 117 108 123 103 128 89 141 116 ...
##  $ Rating            : num  8.1 7 7.3 7.2 6.2 6.1 8.3 6.4 7.1 7 ...
##  $ Votes             : int  757074 485820 157606 60545 393727 56036 258682 2490 7188 192177 ...
##  $ Revenue..Millions.: num  333 126 138 270 325 ...
##  $ Metascore         : int  76 65 62 59 40 42 93 71 78 41 ...
```

```
summary(imdb)
```

```
##       Rank           Title              Genre              Director
##  Min.   :   1.0   Length:1000        Length:1000        Length:1000
##  1st Qu.: 250.8   Class :character   Class :character   Class :character
##  Median : 500.5   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 500.5
##  3rd Qu.: 750.2
##  Max.   :1000.0
##
##     Actors               Year      Runtime..Minutes.     Rating
##  Length:1000        Min.   :2006   Min.   : 66.0     Min.   :1.900
##  Class :character   1st Qu.:2010   1st Qu.:100.0     1st Qu.:6.200
##  Mode  :character   Median :2014   Median :111.0     Median :6.800
##                     Mean   :2013   Mean   :113.2     Mean   :6.723
##                     3rd Qu.:2016   3rd Qu.:123.0     3rd Qu.:7.400
##                     Max.   :2016   Max.   :191.0     Max.   :9.000
##
##      Votes         Revenue..Millions.   Metascore
##  Min.   :     61   Min.   :  0.00     Min.   : 11.00
##  1st Qu.:  36309   1st Qu.: 13.27     1st Qu.: 47.00
##  Median : 110799   Median : 47.98     Median : 59.50
##  Mean   : 169808   Mean   : 82.96     Mean   : 58.99
##  3rd Qu.: 239910   3rd Qu.:113.72     3rd Qu.: 72.00
##  Max.   :1791916   Max.   :936.63     Max.   :100.00
##                    NA's   :128        NA's   :64
```

Data visualization

```
plot(imdb)
```



Data preparation (tidying/engineering - human readable)

Director consists of 644 factors. Since there are too many factors compared to the data size, it is judged to be meaningless. Instead, I added a new variable named Director_count. Similarly, Actors lists the names of four main actors in each film. This is also meaningless with too many factors, so I added a variable called Actors_count. Genre specifies at least 1 to 3 genres. Difficulties arose in distinguishing duplicates, so the films were divided into 10 genres.

In this data, there are 128 missing values in Revenue, 64 in Metascore, and 1 in Actors_count. For Actors, 999 observations out of 1000 have the names of four main actors. However, in one observation, only the names of three main actors are specified. I don't think it's reasonable to compare 4 Actors_count and 3 Actors_count with each other, so if the main actor is the same as the observed value in which 3 actors are specified, Na is included in Actors_count. A regression analysis was created to predict Revenue, replacing missing values with predicted values. To improve the accuracy of the regression model, WLS was used. Metascore has 64 missing values and Actors_count has 1 missing value. A total of 65, corresponding to 6.5% of 1000 observations. The graph on the left is the rating distribution when missing values are included, and the graph on the right is the rating distribution after removing data containing 65 missing values. Since the distributions before and after removing missing values matched and the missing values were less than 10%, data with missing values in Metascore and Actors_count were removed.

Outliers were removed. At Votes, data greater than 800000 were removed. There are 12 data larger than 800000. Revenue removed data greater than 440. There are 9 data greater than 440. Seventeen data were removed through the process of removing outliers in Revenue and Votes. Therefore, after removing the outliers, there are a total of 918 remaining data. I will make a regression model with 918 data. #### transformations - center/scale - dummy variables Both Votes and Revenue are graphs skewed to the right. Therefore, I use the square root for Votes and Revenue to make the distribution more normal.

Data preparation (tidying/engineering/dummy variables- machine readable)

```
#################### dummy variable - Actors ####################
four_Actors <- as.character(imdb$Actors)
four_Actors <- strsplit(four_Actors, split = ",")

trim <- function(x) gsub("^\\s+|\\s+$", "", x)
for (i in 1:1000) {
  for (j in 1:4) {
    four_Actors[[i]][j] <- trim(four_Actors[[i]][j])
  }
}

Actor1 <- c()
Actor2 <- c()
Actor3 <- c()
Actor4 <- c()
for (i in 1:1000) {
  Actor1 <- rbind(Actor1, four_Actors[[i]][1])
  Actor2 <- rbind(Actor2, four_Actors[[i]][2])
  Actor3 <- rbind(Actor3, four_Actors[[i]][3])
  Actor4 <- rbind(Actor4, four_Actors[[i]][4])
}

Actor1_table <- table(Actor1)
Actor1_table <- as.data.frame(Actor1_table)
str(Actor1_table)
```

```
## 'data.frame':    525 obs. of  2 variables:
##  $ Actor1: Factor w/ 525 levels "Aamir Khan","Aaron Paul",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Freq  : int  3 1 1 4 1 1 1 1 9 1 ...
```

```
Actor2_table <- table(Actor2)
Actor2_table <- as.data.frame(Actor2_table)
#str(Actor2_table)

Actor3_table <- table(Actor3)
Actor3_table <- as.data.frame(Actor3_table)
#str(Actor3_table)

Actor4_table <- table(Actor4)
Actor4_table <- as.data.frame(Actor4_table)
#str(Actor4_table)

names(Actor1_table)[1] <- c("Actor")
names(Actor2_table)[1] <- c("Actor")
names(Actor3_table)[1] <- c("Actor")
names(Actor4_table)[1] <- c("Actor")

Actor_count1 <- merge(x = Actor1_table, y = Actor2_table, by = "Actor", all = TRUE)
Actor_count2 <- merge(x = Actor3_table, y = Actor4_table, by = "Actor", all = TRUE)
Actor_count <- merge(x = Actor_count1, y = Actor_count2, by = "Actor", all = TRUE)
```

```
Actor_count[, c("Freq.x.x", "Freq.y.x", "Freq.x.y", "Freq.y.y")][is.na(Actor_count[, c("Freq.x.x", "Fre

Count <- c()
for (i in 1:1985) {
  Count <- rbind(Count, Actor_count$Freq.x.x[i] + Actor_count$Freq.y.x[i] + Actor_count$Freq.x.y[i] + A
}
Actor_count <- cbind(Actor_count, Count)

Actor1_count <- c()
Actor2_count <- c()
Actor3_count <- c()
Actor4_count <- c()
for (i in 1:1000) {
  Actor1_count <- rbind(Actor1_count, Actor_count[Actor_count$Actor == Actor1[i],]$Count)
  Actor2_count <- rbind(Actor2_count, Actor_count[Actor_count$Actor == Actor2[i],]$Count)
  Actor3_count <- rbind(Actor3_count, Actor_count[Actor_count$Actor == Actor3[i],]$Count)
  Actor4_count <- rbind(Actor4_count, Actor_count[Actor_count$Actor == Actor4[i],]$Count)
}
```

```
## Warning in rbind(Actor4_count, Actor_count[Actor_count$Actor == Actor4[i], :
## number of columns of result is not a multiple of vector length (arg 2)
```

```
Actors_count <- c()
for (i in 1:1000) {
  Actors_count <- rbind(Actors_count, Actor1_count[i] + Actor2_count[i] + Actor3_count[i] + Actor4_coun
}

imdb <- cbind(imdb, Actors_count)


#################### dummy variable - Director ####################
Director_table <- table(imdb$Director)
Director_table <- as.data.frame(Director_table)
Director_count <- c()
for (i in 1:1000) {
  Director_count <- rbind(Director_count, Director_table[Director_table$Var1 == imdb$Director[i],])
}
imdb <- cbind(imdb, Director_count$Freq)


#################### dummy variable - Genre ####################
Genre_table <- table(imdb$Genre)
Genre_table <- as.data.frame(Genre_table)
Genre_count <- c()
for (i in 1:1000) {
  Genre_count <- rbind(Genre_count, Genre_table[Genre_table$Var1 == imdb$Genre[i],])
}

imdb <- cbind(imdb, Genre_count$Freq)
################################################################################
#################### rename ####################
names(imdb) <- c('Rank', 'Title', 'Genre', 'Director', 'Actors', 'Year', 'Runtime', 'Rating', 'Votes',
data <- imdb
```

```r
imdb <- imdb[,-c(2:5)]
################################################################################
############### na-value ###############
colSums(is.na(imdb))
```

```
##           Rank           Year        Runtime         Rating          Votes
##              0              0              0              0              0
##        Revenue      Metascore   Actors_count Director_count    Genre_count
##            128             64              1              0              0
```

```r
############### na-value : Revenue - regression ###############
no_revenue_data <- imdb %>% filter(is.na(Revenue))
yes_revenue_data <- imdb %>% filter(!is.na(Revenue))
yes_revenue_data <- na.omit(yes_revenue_data)
revenue_model1 <- lm(Revenue ~ Rank + Year + Runtime + Rating + Votes + Metascore + Actors_count + Direc
revenue_model2 <- step(revenue_model1, direction="both")
```

```
## Start:  AIC=7264.68
## Revenue ~ Rank + Year + Runtime + Rating + Votes + Metascore +
##     Actors_count + Director_count + Genre_count
##
##                   Df Sum of Sq     RSS    AIC
## - Actors_count     1       103 4806016 7262.7
## - Director_count   1       588 4806500 7262.8
## - Metascore        1      9534 4815446 7264.3
## <none>                         4805912 7264.7
## - Rank             1     32924 4838837 7268.4
## - Year             1     61565 4867477 7273.3
## - Runtime          1     64725 4870637 7273.9
## - Rating           1    103246 4909159 7280.5
## - Genre_count      1    301937 5107849 7313.7
## - Votes            1   1888415 6694328 7540.1
##
## Step:  AIC=7262.7
## Revenue ~ Rank + Year + Runtime + Rating + Votes + Metascore +
##     Director_count + Genre_count
##
##                   Df Sum of Sq     RSS    AIC
## - Director_count   1       510 4806526 7260.8
## - Metascore        1      9816 4815831 7262.4
## <none>                         4806016 7262.7
## + Actors_count     1       103 4805912 7264.7
## - Rank             1     32835 4838851 7266.4
## - Year             1     61593 4867609 7271.4
## - Runtime          1     64913 4870929 7271.9
## - Rating           1    103442 4909458 7278.5
## - Genre_count      1    301855 5107871 7311.7
## - Votes            1   2005230 6811246 7552.6
##
## Step:  AIC=7260.79
## Revenue ~ Rank + Year + Runtime + Rating + Votes + Metascore +
##     Genre_count
##
```

```
##                   Df Sum of Sq     RSS    AIC
## - Metascore        1      9727 4816253 7260.5
## <none>                         4806526 7260.8
## + Director_count   1       510 4806016 7262.7
## + Actors_count     1        26 4806500 7262.8
## - Rank             1     33963 4840489 7264.7
## - Year             1     61270 4867796 7269.4
## - Runtime          1     70902 4877428 7271.0
## - Rating           1    104865 4911391 7276.9
## - Genre_count      1    302471 5108997 7309.9
## - Votes            1   2093959 6900485 7561.5
##
## Step:  AIC=7260.48
## Revenue ~ Rank + Year + Runtime + Rating + Votes + Genre_count
##
##                   Df Sum of Sq     RSS    AIC
## <none>                         4816253 7260.5
## + Metascore        1      9727 4806526 7260.8
## + Director_count   1       422 4815831 7262.4
## + Actors_count     1       217 4816036 7262.4
## - Rank             1     34906 4851159 7264.5
## - Year             1     62108 4878361 7269.2
## - Runtime          1     67261 4883514 7270.1
## - Rating           1    108736 4924989 7277.2
## - Genre_count      1    293002 5109255 7307.9
## - Votes            1   2100469 6916722 7561.4
```

```
summary(revenue_model1)
```

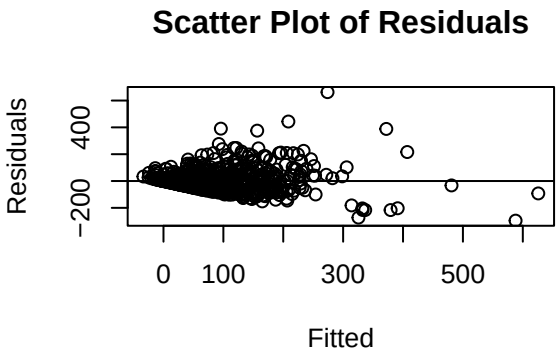
```
##
## Call:
## lm(formula = Revenue ~ Rank + Year + Runtime + Rating + Votes +
##     Metascore + Actors_count + Director_count + Genre_count,
##     data = yes_revenue_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -292.62  -40.87  -10.25   23.39  659.46
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -6.642e+03  2.078e+03  -3.197 0.001444 **
## Rank           -2.677e-02  1.125e-02  -2.380 0.017526 *
## Year            3.353e+00  1.030e+00   3.255 0.001181 **
## Runtime         5.681e-01  1.702e-01   3.337 0.000884 ***
## Rating         -1.932e+01  4.583e+00  -4.215 2.77e-05 ***
## Votes           3.633e-04  2.015e-05  18.027  < 2e-16 ***
## Metascore       2.744e-01  2.142e-01   1.281 0.200606
## Actors_count   -4.774e-02  3.578e-01  -0.133 0.893888
## Director_count  6.422e-01  2.019e+00   0.318 0.750466
## Genre_count    -1.311e-01  1.818e-02  -7.208 1.28e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

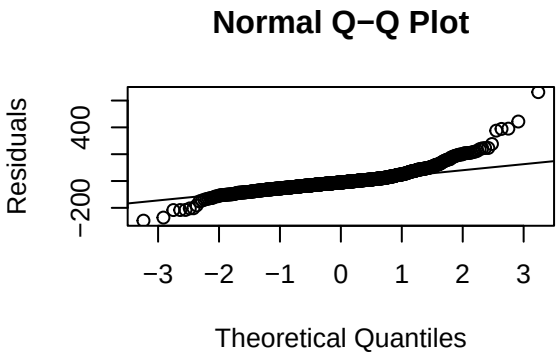
```
## Residual standard error: 76.23 on 827 degrees of freedom
## Multiple R-squared:  0.474,  Adjusted R-squared:  0.4683
## F-statistic:  82.8 on 9 and 827 DF,  p-value: < 2.2e-16
```

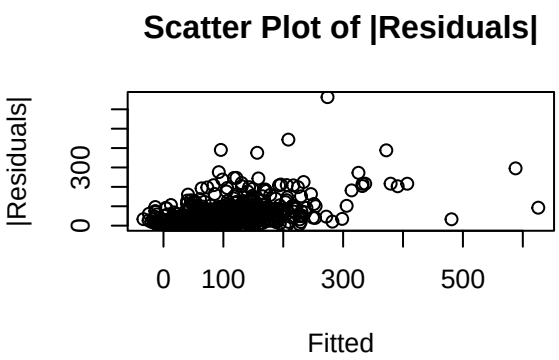
```
summary(revenue_model2)
```

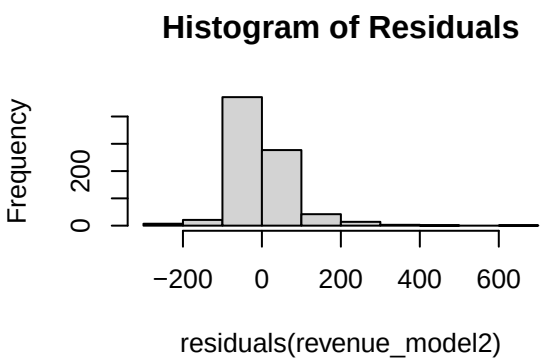
```
##
## Call:
## lm(formula = Revenue ~ Rank + Year + Runtime + Rating + Votes +
##     Genre_count, data = yes_revenue_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -295.39  -39.94  -10.74   20.83  662.68
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.579e+03  2.046e+03  -3.215 0.001353 **
## Rank        -2.740e-02  1.117e-02  -2.453 0.014386 *
## Year         3.318e+00  1.014e+00   3.272 0.001114 **
## Runtime      5.599e-01  1.645e-01   3.405 0.000694 ***
## Rating      -1.591e+01  3.674e+00  -4.329 1.68e-05 ***
## Votes        3.642e-04  1.914e-05  19.026  < 2e-16 ***
## Genre_count -1.276e-01  1.795e-02  -7.106 2.58e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 76.18 on 830 degrees of freedom
## Multiple R-squared:  0.4729, Adjusted R-squared:  0.4691
## F-statistic: 124.1 on 6 and 830 DF,  p-value: < 2.2e-16
```

```
# Model3. Assumption
{par(mfcol = c(2,2))
plot(fitted(revenue_model2), residuals(revenue_model2), xlab = 'Fitted', ylab = 'Residuals', main = 'Sca
abline(h=0)
plot(fitted(revenue_model2), abs(residuals(revenue_model2)), xlab = 'Fitted', ylab = '|Residuals|', mai
qqnorm(residuals(revenue_model2), ylab = 'Residuals')
qqline(residuals(revenue_model2))
hist(residuals(revenue_model2), main = 'Histogram of Residuals')
}
```










```
### Transformation - WLS ###
absresid <- abs(residuals(revenue_model2))
lmfitw0 <- lm(absresid ~ Year + Runtime + Rating + Votes + Genre_count, data = yes_revenue_data)
w <- 1/fitted(lmfitw0)^2

revenue_model2.wls <- lm(Revenue ~ Year + Runtime + Rating + Votes + Genre_count, data = yes_revenue_da
summary(revenue_model2.wls)
```

```
##
## Call:
## lm(formula = Revenue ~ Year + Runtime + Rating + Votes + Genre_count,
##     data = yes_revenue_data, weights = w)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3257 -0.8143 -0.2869  0.4434  7.4522
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.171e+03  7.009e+02  -3.097 0.002023 **
## Year         1.109e+00  3.477e-01   3.190 0.001476 **
## Runtime      3.205e-01  8.844e-02   3.624 0.000307 ***
## Rating      -1.184e+01  1.583e+00  -7.477 1.93e-13 ***
## Votes        3.886e-04  1.855e-05  20.953  < 2e-16 ***
## Genre_count -4.638e-02  1.078e-02  -4.304 1.88e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.344 on 831 degrees of freedom
## Multiple R-squared:  0.5144, Adjusted R-squared:  0.5115
## F-statistic: 176.1 on 5 and 831 DF,  p-value: < 2.2e-16
```

```
### Predict ###
# pred = predict(revenue_model3.wls, yes_revenue_data)
# cor(pred, yes_revenue_data$Revenue)

pred_value = predict(revenue_model2.wls, no_revenue_data)

count = 1
for (i in 1:1000) {
  if (is.na(imdb$Revenue[i])) {
    imdb$Revenue[i] <- pred_value[count]
    count = count + 1
  }
}
############### na-value : Metascore ###############
temp_imdb <- na.omit(imdb)
sapply(imdb, function(x) sum(is.na(x)))
```

```
##           Rank           Year        Runtime         Rating          Votes
##              0              0              0              0              0
##        Revenue      Metascore   Actors_count Director_count    Genre_count
##              0             64              1              0              0
```

```
sapply(temp_imdb, function(x) sum(is.na(x)))
```

```
##           Rank           Year        Runtime         Rating          Votes
##              0              0              0              0              0
##        Revenue      Metascore   Actors_count Director_count    Genre_count
##              0              0              0              0              0
```

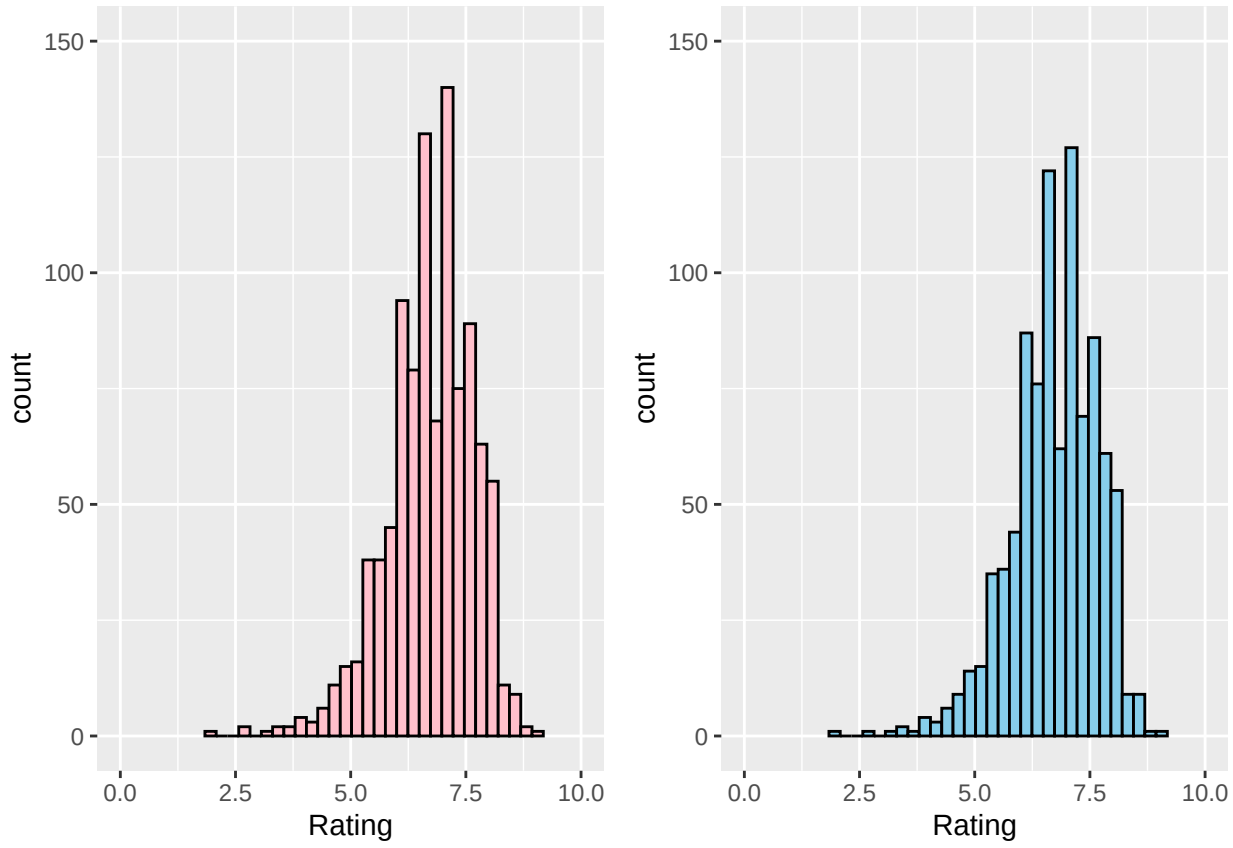
```
str(temp_imdb)
```

```
## 'data.frame':    935 obs. of  10 variables:
##  $ Rank          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Year          : int  2014 2012 2016 2016 2016 2016 2016 2016 2016 2016 ...
##  $ Runtime       : int  121 124 117 108 123 103 128 89 141 116 ...
##  $ Rating        : num  8.1 7 7.3 7.2 6.2 6.1 8.3 6.4 7.1 7 ...
##  $ Votes         : int  757074 485820 157606 60545 393727 56036 258682 2490 7188 192177 ...
##  $ Revenue       : num  333 126 138 270 325 ...
##  $ Metascore     : int  76 65 62 59 40 42 93 71 78 41 ...
##  $ Actors_count  : num  31 29 15 30 25 18 26 8 16 23 ...
##  $ Director_count: int  3 8 6 1 3 1 2 1 1 2 ...
##  $ Genre_count   : int  162 406 120 45 162 162 406 55 56 406 ...
##  - attr(*, "na.action")= 'omit' Named int [1:65] 26 27 28 40 43 48 104 118 124 155 ...
##   ..- attr(*, "names")= chr [1:65] "26" "27" "28" "40" ...
```

```
library(ggplot2)
beforeRemoving <- ggplot(imdb, aes(x = Rating)) +
```

```
  geom_histogram(fill = "pink", color = "black") +
  coord_cartesian(xlim = c(0, 10), ylim = c(0, 150))

afterRemoving <- ggplot(temp_imdb, aes(x = Rating)) +
  geom_histogram(fill = "skyblue", color = "black") +
  coord_cartesian(xlim = c(0, 10), ylim = c(0, 150))

grid.arrange(beforeRemoving, afterRemoving, nrow=1, ncol=2)
```

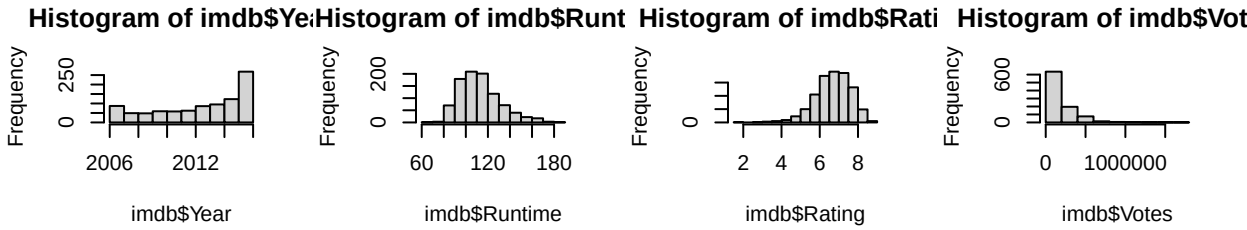
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

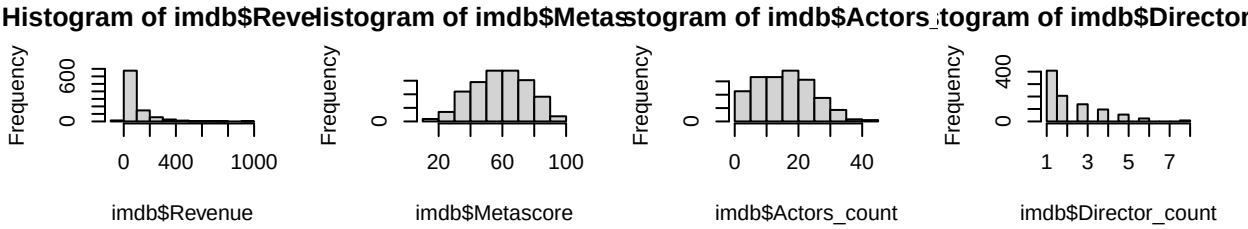


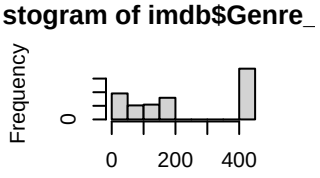

```
imdb <- temp_imdb
```

```
{
par(mfrow=c(3,4))
hist(imdb$Year)
hist(imdb$Runtime)
hist(imdb$Rating)
hist(imdb$Votes)
hist(imdb$Revenue)
hist(imdb$Metascore)
```

```
hist(imdb$Actors_count)
hist(imdb$Director_count)
hist(imdb$Genre_count)
}
```



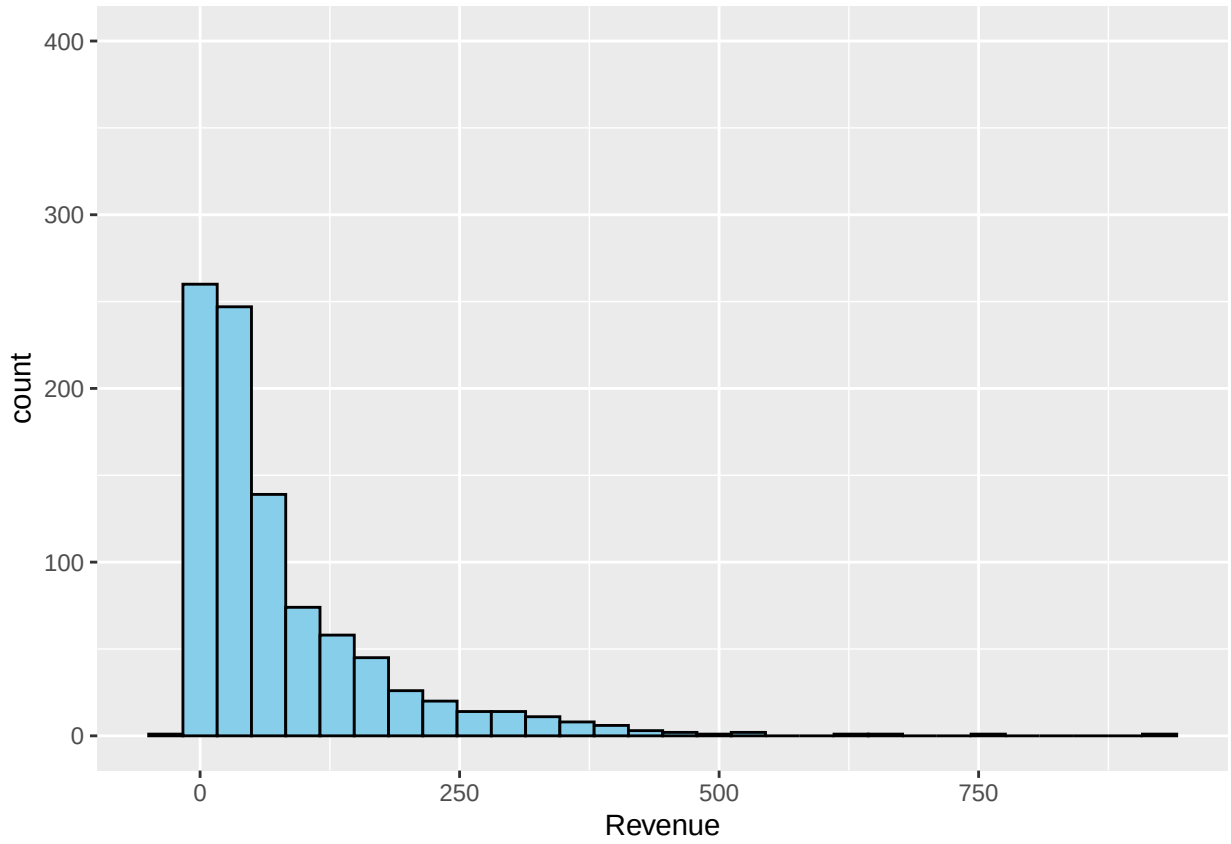




visualization

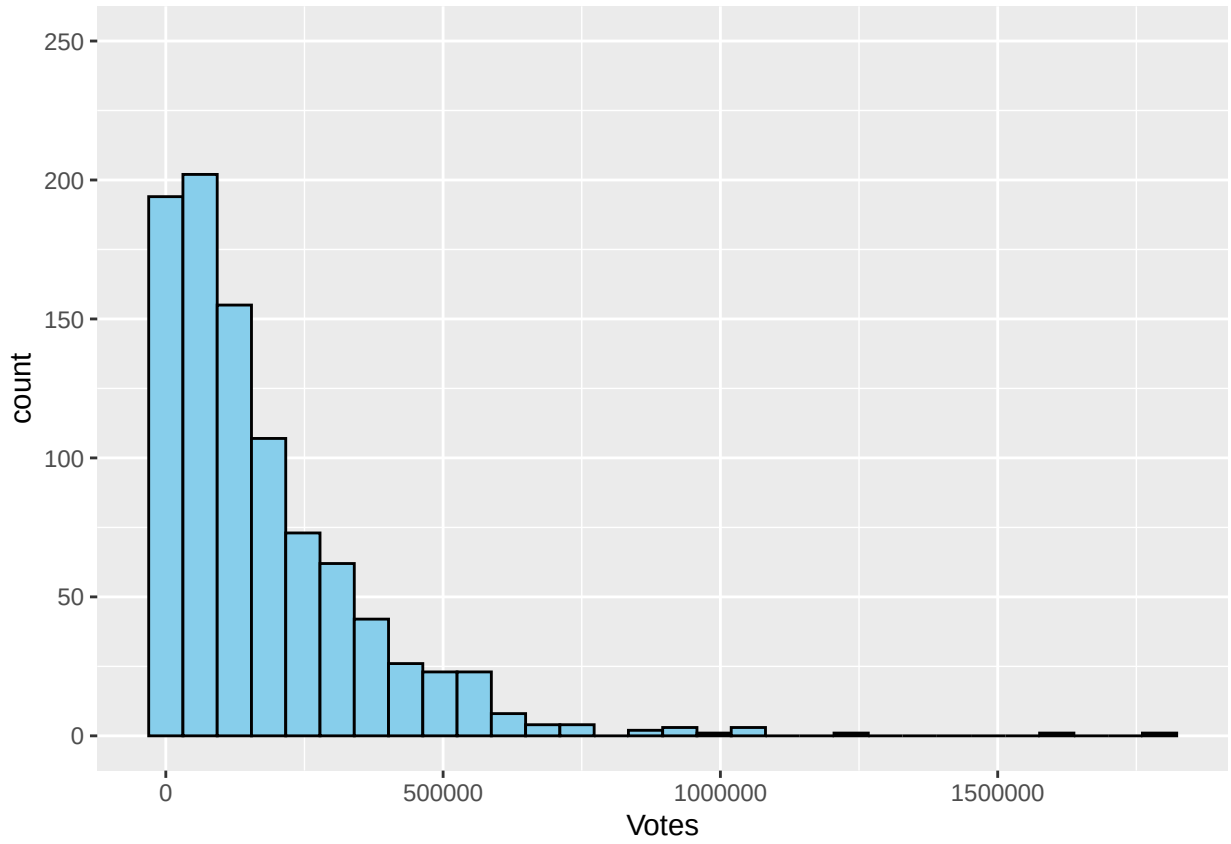
```
outlier_imdb <- imdb

ggplot(imdb, aes(x = Revenue)) +
  geom_histogram(fill = "skyblue", color = "black") +
  coord_cartesian(ylim = c(0, 400))
```

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




```
ggplot(imdb, aes(x = Votes)) +
  geom_histogram(fill = "skyblue", color = "black") +
  coord_cartesian(ylim = c(0, 250))
```

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




```
length(which(imdb$Revenue > 430))
```

```
## [1] 9
```

```
length(which(imdb$Votes > 800000))
```

```
## [1] 12
```

```
outlier_imdb <- subset(imdb, imdb$Revenue < 430)
outlier_imdb <- subset(imdb, imdb$Votes < 800000)
```

```
trans_imdb <- outlier_imdb
trans_imdb <- transform(trans_imdb, Votes = sqrt(Votes))
trans_imdb <- transform(trans_imdb, Revenue = sqrt(Revenue+30))

summary(outlier_imdb)
```

transformations - center/scale

```
##       Rank             Year         Runtime          Rating
##  Min.   :   1.0   Min.   :2006   Min.   : 66.0   Min.   :1.900
```

```
##  1st Qu.: 255.5   1st Qu.:2010   1st Qu.:100.0   1st Qu.:6.200
##  Median : 502.0   Median :2014   Median :110.0   Median :6.800
##  Mean   : 503.4   Mean   :2013   Mean   :112.7   Mean   :6.708
##  3rd Qu.: 750.5   3rd Qu.:2016   3rd Qu.:123.0   3rd Qu.:7.400
##  Max.   :1000.0   Max.   :2016   Max.   :187.0   Max.   :8.600
##      Votes           Revenue         Metascore       Actors_count
##  Min.   :    61   Min.   :-20.68   Min.   : 11.00   Min.   : 4.00
##  1st Qu.: 40443   1st Qu.: 13.26   1st Qu.: 46.00   1st Qu.: 9.00
##  Median :113599   Median : 41.96   Median : 59.00   Median :16.00
##  Mean   :163340   Mean   : 75.46   Mean   : 58.79   Mean   :16.17
##  3rd Qu.:241534   3rd Qu.:100.48   3rd Qu.: 72.00   3rd Qu.:22.00
##  Max.   :757074   Max.   :936.63   Max.   :100.00   Max.   :44.00
##  Director_count   Genre_count
##  Min.   :1.000   Min.   : 16.0
##  1st Qu.:1.000   1st Qu.: 55.0
##  Median :2.000   Median :162.0
##  Mean   :2.221   Mean   :217.8
##  3rd Qu.:3.000   3rd Qu.:406.0
##  Max.   :8.000   Max.   :406.0
```

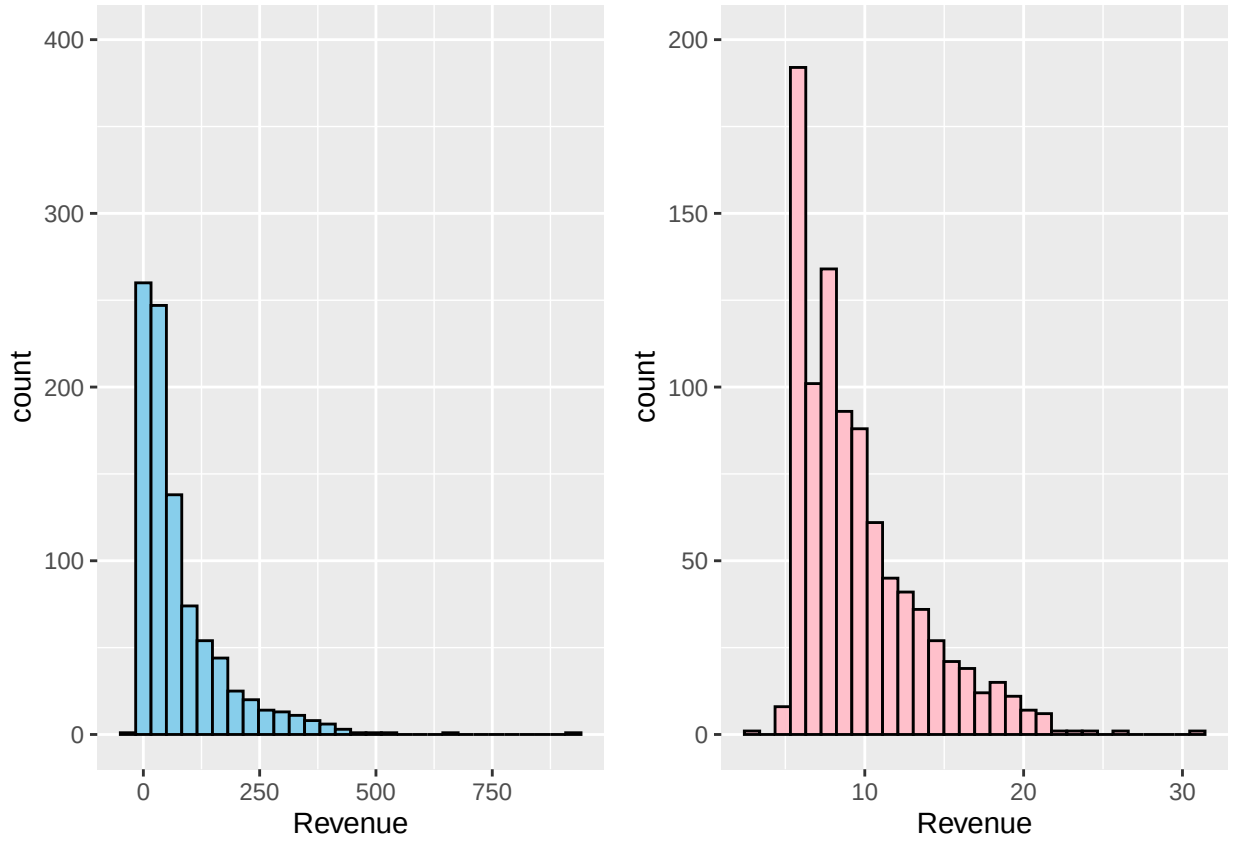
```
summary(trans_imdb)
```

```
##       Rank            Year         Runtime          Rating
##  Min.   :   1.0   Min.   :2006   Min.   : 66.0   Min.   :1.900
##  1st Qu.: 255.5   1st Qu.:2010   1st Qu.:100.0   1st Qu.:6.200
##  Median : 502.0   Median :2014   Median :110.0   Median :6.800
##  Mean   : 503.4   Mean   :2013   Mean   :112.7   Mean   :6.708
##  3rd Qu.: 750.5   3rd Qu.:2016   3rd Qu.:123.0   3rd Qu.:7.400
##  Max.   :1000.0   Max.   :2016   Max.   :187.0   Max.   :8.600
##      Votes           Revenue         Metascore       Actors_count
##  Min.   :  7.81   Min.   : 3.053   Min.   : 11.00   Min.   : 4.00
##  1st Qu.:201.10   1st Qu.: 6.577   1st Qu.: 46.00   1st Qu.: 9.00
##  Median :337.04   Median : 8.483   Median : 59.00   Median :16.00
##  Mean   :352.09   Mean   : 9.525   Mean   : 58.79   Mean   :16.17
##  3rd Qu.:491.46   3rd Qu.:11.423   3rd Qu.: 72.00   3rd Qu.:22.00
##  Max.   :870.10   Max.   :31.091   Max.   :100.00   Max.   :44.00
##  Director_count   Genre_count
##  Min.   :1.000   Min.   : 16.0
##  1st Qu.:1.000   1st Qu.: 55.0
##  Median :2.000   Median :162.0
##  Mean   :2.221   Mean   :217.8
##  3rd Qu.:3.000   3rd Qu.:406.0
##  Max.   :8.000   Max.   :406.0
```

```
beforeTransRevenue <- ggplot(outlier_imdb, aes(x = Revenue)) +
  geom_histogram(fill = "skyblue", color = "black") +
  coord_cartesian(ylim = c(0, 400))
afterTransRevenue <- ggplot(trans_imdb, aes(x = Revenue)) +
  geom_histogram(fill = "pink", color = "black") +
  coord_cartesian(ylim = c(0, 200))

grid.arrange(beforeTransRevenue, afterTransRevenue, nrow=1, ncol=2)
```

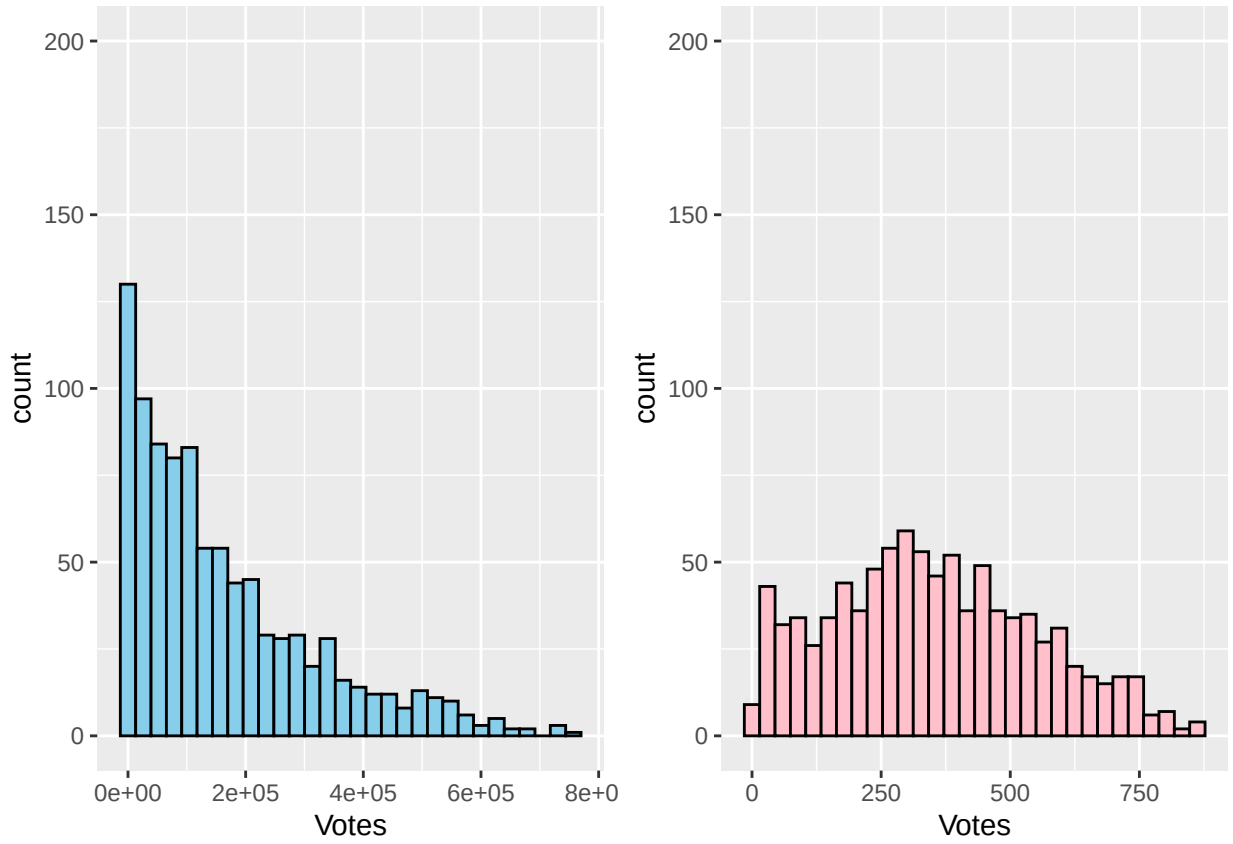
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

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

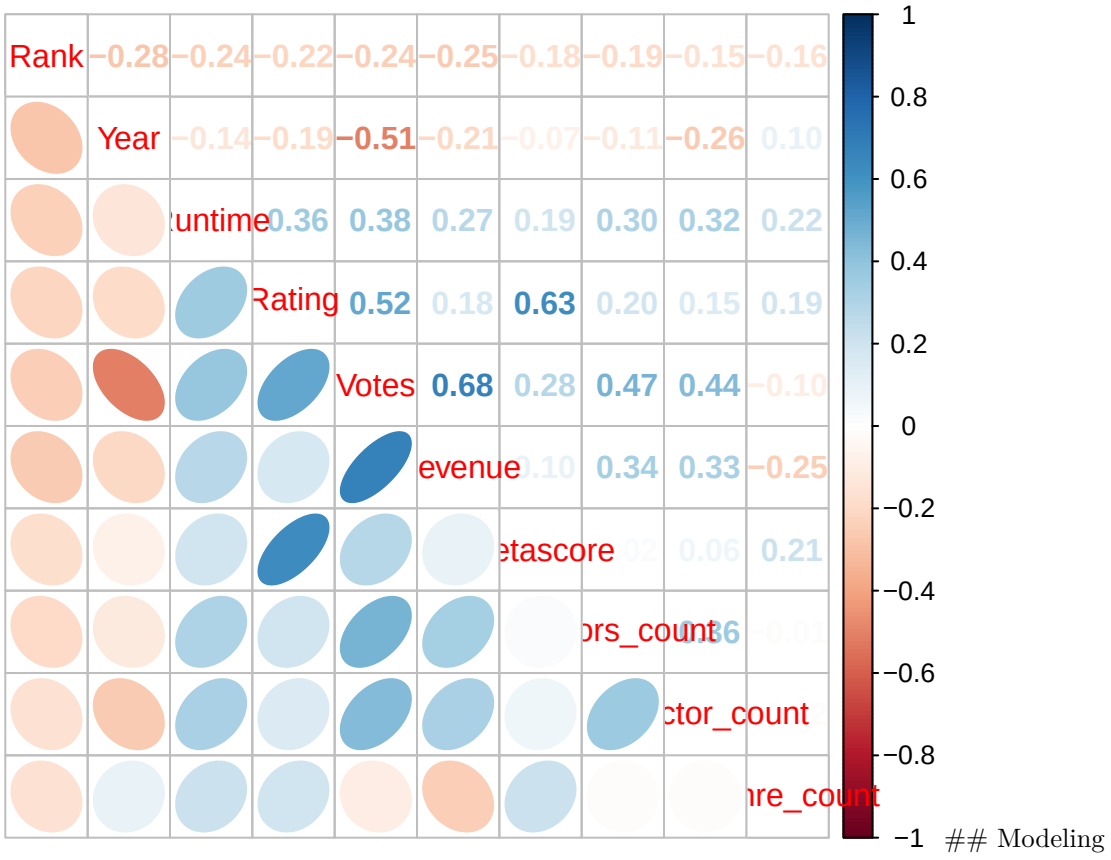



```
beforeTransVotes <- ggplot(outlier_imdb, aes(x = Votes)) +
  geom_histogram(fill = "skyblue", color = "black") +
  coord_cartesian(ylim = c(0, 200))

afterTransVotes <- ggplot(trans_imdb, aes(x = Votes)) +
  geom_histogram(fill = "pink", color = "black") +
  coord_cartesian(ylim = c(0, 200))

grid.arrange(beforeTransVotes, afterTransVotes, nrow=1, ncol=2)
```

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




```
imdb <- trans_imdb
M <- cor(imdb)
corrplot.mixed(M, upper = "number", lower = "ellipse")
```



Modeling

```
str(imdb)
```

```
## 'data.frame':    923 obs. of  10 variables:
##  $ Rank          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Year          : int  2014 2012 2016 2016 2016 2016 2016 2016 2016 2016 ...
##  $ Runtime       : int  121 124 117 108 123 103 128 89 141 116 ...
##  $ Rating        : num  8.1 7 7.3 7.2 6.2 6.1 8.3 6.4 7.1 7 ...
##  $ Votes         : num  870 697 397 246 627 ...
##  $ Revenue       : num  19.1 12.5 13 17.3 18.8 ...
##  $ Metascore     : int  76 65 62 59 40 42 93 71 78 41 ...
##  $ Actors_count  : num  31 29 15 30 25 18 26 8 16 23 ...
##  $ Director_count: int  3 8 6 1 3 1 2 1 1 2 ...
##  $ Genre_count   : int  162 406 120 45 162 162 406 55 56 406 ...
```

```
p<-lm(Rating~.,data=imdb)
summary(p)
```

```
##
## Call:
## lm(formula = Rating ~ ., data = imdb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4103 -0.3011  0.0299  0.3516  1.8823
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -4.319e+01  1.741e+01  -2.480  0.01331 *
## Rank           -2.099e-05  8.399e-05  -0.250  0.80271
## Year            2.351e-02  8.633e-03   2.724  0.00658 **
## Runtime         6.933e-03  1.285e-03   5.394 8.77e-08 ***
## Votes           2.668e-03  1.926e-04  13.849  < 2e-16 ***
## Revenue        -5.796e-02  7.578e-03  -7.648 5.16e-14 ***
## Metascore       2.461e-02  1.277e-03  19.282  < 2e-16 ***
## Actors_count   -8.252e-04  2.832e-03  -0.291  0.77084
## Director_count -4.280e-02  1.578e-02  -2.713  0.00680 **
## Genre_count     3.150e-04  1.413e-04   2.229  0.02606 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6035 on 913 degrees of freedom
## Multiple R-squared:  0.5751, Adjusted R-squared:  0.571
## F-statistic: 137.3 on 9 and 913 DF,  p-value: < 2.2e-16
```

```
a<-step(p,direction=c("both","backward","forward"))
```

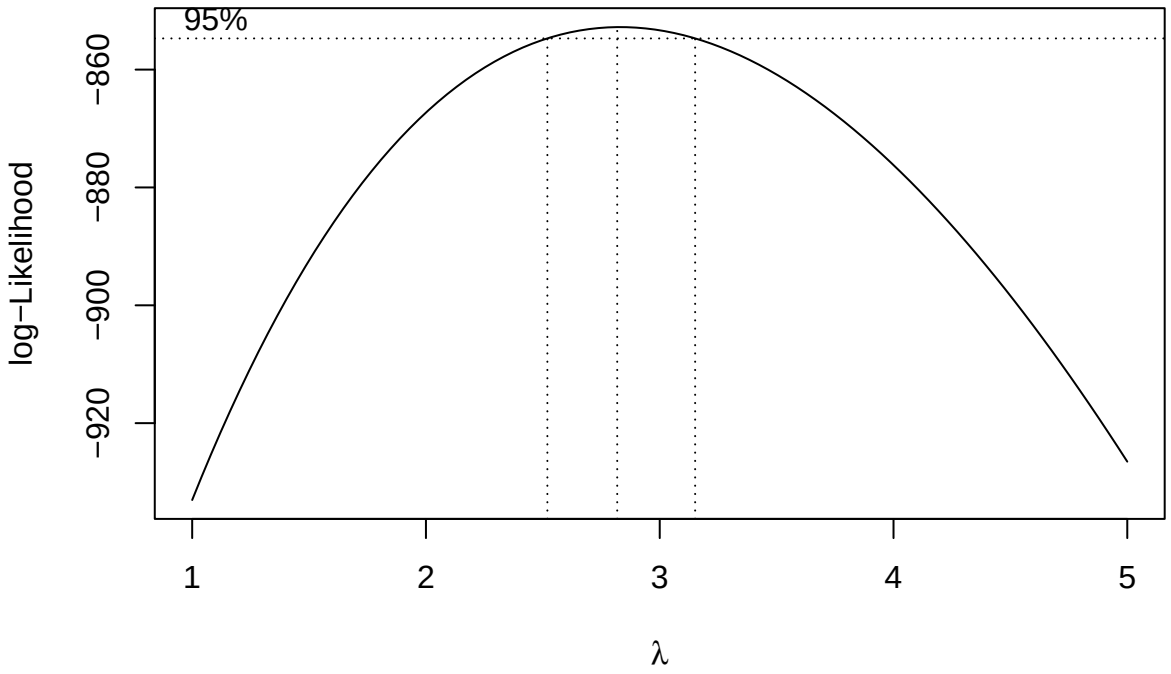
```
## Start:  AIC=-922.32
## Rating ~ Rank + Year + Runtime + Votes + Revenue + Metascore +
##     Actors_count + Director_count + Genre_count
##
##                  Df Sum of Sq    RSS     AIC
## - Rank            1     0.023 332.54 -924.25
## - Actors_count    1     0.031 332.55 -924.23
## <none>                        332.52 -922.32
## - Genre_count     1     1.809 334.33 -919.31
## - Director_count  1     2.680 335.20 -916.91
## - Year            1     2.702 335.22 -916.85
## - Runtime         1    10.598 343.12 -895.36
## - Revenue         1    21.305 353.83 -867.00
## - Votes           1    69.848 402.37 -748.33
## - Metascore       1   135.412 467.93 -609.00
##
## Step:  AIC=-924.25
## Rating ~ Year + Runtime + Votes + Revenue + Metascore + Actors_count +
##     Director_count + Genre_count
##
##                  Df Sum of Sq    RSS     AIC
## - Actors_count    1     0.031 332.57 -926.17
## <none>                        332.54 -924.25
## + Rank            1     0.023 332.52 -922.32
## - Genre_count     1     1.914 334.46 -920.96
## - Director_count  1     2.659 335.20 -918.90
## - Year            1     3.675 336.22 -916.11
## - Runtime         1    10.751 343.29 -896.89
## - Revenue         1    21.307 353.85 -868.93
## - Votes           1    75.197 407.74 -738.09
## - Metascore       1   135.918 468.46 -609.96
##
## Step:  AIC=-926.17
```

```
## Rating ~ Year + Runtime + Votes + Revenue + Metascore + Director_count +
##     Genre_count
##
##                   Df Sum of Sq    RSS     AIC
## <none>                         332.57 -926.17
## + Actors_count     1     0.031 332.54 -924.25
## + Rank             1     0.023 332.55 -924.23
## - Genre_count      1     1.905 334.48 -922.89
## - Director_count   1     2.860 335.43 -920.26
## - Year             1     3.676 336.25 -918.02
## - Runtime          1    10.753 343.33 -898.80
## - Revenue          1    21.276 353.85 -870.93
## - Votes            1    83.315 415.89 -721.82
## - Metascore        1   139.842 472.42 -604.20
```

```
summary(a)
```

```
##
## Call:
## lm(formula = Rating ~ Year + Runtime + Votes + Revenue + Metascore +
##     Director_count + Genre_count, data = imdb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4023 -0.3032  0.0257  0.3577  1.8817
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -4.433e+01  1.523e+01  -2.910  0.00370 **
## Year            2.407e-02  7.568e-03   3.180  0.00152 **
## Runtime         6.918e-03  1.272e-03   5.439 6.87e-08 ***
## Votes           2.661e-03  1.758e-04  15.140  < 2e-16 ***
## Revenue        -5.775e-02  7.548e-03  -7.651 5.06e-14 ***
## Metascore       2.469e-02  1.259e-03  19.615  < 2e-16 ***
## Director_count -4.339e-02  1.547e-02  -2.805  0.00514 **
## Genre_count     3.195e-04  1.396e-04   2.289  0.02228 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6029 on 915 degrees of freedom
## Multiple R-squared:  0.5751, Adjusted R-squared:  0.5718
## F-statistic: 176.9 on 7 and 915 DF,  p-value: < 2.2e-16
```

```
y<-imdb$Rating
x1<-imdb$Year
x2<-imdb$Runtime
x3<-imdb$Votes
x4<-imdb$Revenue
x5<-imdb$Metascore
x6<-imdb$Director_count
x7<-imdb$Genre_count


bc <- boxcox(lm(a), lambda = seq(1, 5, 0.1))
```




```
lambda <- bc$x[which.max(bc$y)]
aboxcox <- lm(((y^lambda-1)/lambda) ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = imdb)
summary(aboxcox)
```

```
##
## Call:
## lm(formula = ((y^lambda - 1)/lambda) ~ x1 + x2 + x3 + x4 + x5 +
##     x6 + x7, data = imdb)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -62.61 -10.67  -0.77  10.08  65.79
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.429e+03  4.359e+02  -3.278 0.001084 **
## x1           7.093e-01  2.166e-01   3.275 0.001097 **
## x2           2.162e-01  3.640e-02   5.940 4.04e-09 ***
## x3           8.268e-02  5.031e-03  16.435  < 2e-16 ***
## x4          -1.759e+00  2.160e-01  -8.143 1.26e-15 ***
## x5           7.629e-01  3.602e-02  21.180  < 2e-16 ***
## x6          -1.580e+00  4.426e-01  -3.570 0.000375 ***
## x7           1.040e-02  3.994e-03   2.603 0.009387 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.25 on 915 degrees of freedom
## Multiple R-squared:  0.6139, Adjusted R-squared:  0.6109
## F-statistic: 207.8 on 7 and 915 DF,  p-value: < 2.2e-16
```

```r
b<-lm(y ~ x1 + x2 + x3 + x4 + x5 + x6,data=imdb)
summary(b)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6, data = imdb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4147 -0.3024  0.0261  0.3484  1.8280
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.709e+01  1.522e+01  -3.094 0.002034 **
## x1           2.544e-02  7.562e-03   3.364 0.000802 ***
## x2           7.688e-03  1.229e-03   6.254 6.15e-10 ***
## x3           2.662e-03  1.762e-04  15.108  < 2e-16 ***
## x4          -6.203e-02  7.330e-03  -8.462  < 2e-16 ***
## x5           2.526e-02  1.236e-03  20.437  < 2e-16 ***
## x6          -4.308e-02  1.550e-02  -2.779 0.005566 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6043 on 916 degrees of freedom
## Multiple R-squared:  0.5726, Adjusted R-squared:  0.5698
## F-statistic: 204.6 on 6 and 916 DF,  p-value: < 2.2e-16
```

```r
resid <- residuals(b)
absresid <- abs(resid)

# find the linear relationship between |residual| vs X
lmfitw0 <- lm(absresid ~ x1 + x2 + x3 + x4 + x5 + x6, data = imdb)

# weight is proportion to inverse of variance
w <- 1/fitted(lmfitw0)^2

# fit WLS
wlm <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6, weights = w)
summary(wlm)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6, weights = w)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4751 -0.7293  0.0508  0.8878  5.4504
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.714e+01  1.209e+01  -2.245  0.02502 *
## x1           1.560e-02  6.009e-03   2.596  0.00959 **
```

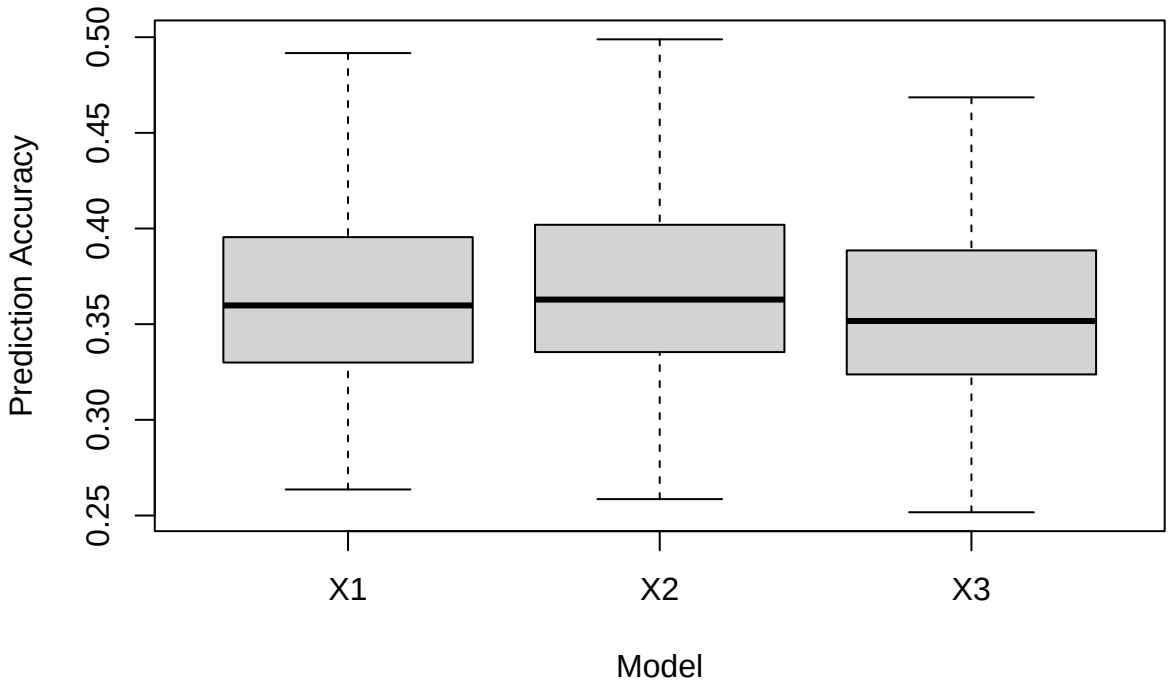
```
## x2           7.089e-03  9.805e-04   7.230 1.02e-12 ***
## x3           2.336e-03  1.269e-04  18.399  < 2e-16 ***
## x4          -4.808e-02  5.608e-03  -8.574  < 2e-16 ***
## x5           2.369e-02  1.112e-03  21.300  < 2e-16 ***
## x6          -4.571e-02  1.035e-02  -4.417 1.12e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.335 on 916 degrees of freedom
## Multiple R-squared:  0.6868, Adjusted R-squared:  0.6847
## F-statistic: 334.8 on 6 and 916 DF,  p-value: < 2.2e-16
```

```
##Predict##
set.seed(777)
acc.result<-matrix(NA,3,100)

for (i in 1:100) {
  tt<- sample(923,650, replace= FALSE)
  train.set<- imdb[tt,]
  test.set<- imdb[-tt,]

  #str(train.set)
  lmfit1<-lm(Rating ~ Year + Runtime + Votes + Revenue + Metascore + Director_count + Genre_count, trai
  acc.result[1, i] <- round(mean((test.set$Rating - predict(lmfit1, test.set))^2),4)

  lmfit2 <- lm(Rating ~ Year + Runtime + Votes + Revenue + Metascore + Director_count, train.set)
  resid <- residuals(lmfit2)
  absresid <- abs(resid)

  # find the linear relationship between |residual| vs X
  lmfitw0 <- lm(absresid ~ Year + Runtime + Votes + Revenue + Metascore + Director_count, data = train.s

  # weight is proportion to inverse of variance
  w <- 1/fitted(lmfitw0)^2
  lmfit2<- lm(Rating ~ Year + Runtime + Votes + Revenue + Metascore + Director_count, weight = w, data =
  acc.result[2, i] <- round(mean((test.set$Rating - predict(lmfit2, test.set))^2),5)

  # lmfit3<-lm(y~x1+x2+x3+x4+x5+x6+x7, data= train.set)
  bc <- boxcox(lmfit1, lambda = seq(-5, 5, 0.1), plotit = FALSE)
  lambda <- bc$x[which.max(bc$y)]
  lmfit3 <- lm(((Rating^lambda-1)/lambda) ~ Year + Runtime + Votes + Revenue + Metascore +
                 Director_count + Genre_count, data = train.set)

  a<-(predict(lmfit3, test.set)*lambda+1)^(1/lambda)
  acc.result[3, i] <- round(mean((test.set$Rating -a)^2),5)
}
acc.result
```

```
##         [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]
## [1,] 0.36580 0.30220 0.30550 0.34810 0.36550 0.39750 0.40280 0.29860 0.38330
## [2,] 0.36522 0.30829 0.30760 0.35629 0.39312 0.39736 0.40544 0.30143 0.38224
## [3,] 0.36501 0.31277 0.29683 0.34372 0.35506 0.38443 0.40213 0.28353 0.37914
##        [,10]   [,11]   [,12]   [,13]   [,14]   [,15]   [,16]   [,17]   [,18]
## [1,] 0.35850 0.39300 0.41770 0.31590 0.31460 0.34770 0.34410 0.40740 0.32360
```

```
## [2,] 0.35200 0.40167 0.41747 0.32589 0.31982 0.35648 0.34849 0.40219 0.32484
## [3,] 0.35742 0.38528 0.41502 0.32378 0.31961 0.35052 0.33497 0.40255 0.32480
##        [,19]   [,20]   [,21]   [,22]   [,23]   [,24]   [,25]   [,26]   [,27]
## [1,] 0.31350 0.45350 0.38860 0.32960 0.39740 0.29530 0.31150 0.44980 0.40650
## [2,] 0.30382 0.45938 0.37458 0.33568 0.40171 0.30672 0.31111 0.46864 0.42370
## [3,] 0.29730 0.45629 0.38047 0.33288 0.38329 0.28897 0.31633 0.45076 0.39925
##        [,28]   [,29]   [,30]   [,31]   [,32]   [,33]   [,34]   [,35]   [,36]
## [1,] 0.41980 0.35380 0.38620 0.35280 0.33410 0.33820 0.39490 0.38330 0.33980
## [2,] 0.42231 0.36292 0.40278 0.35057 0.33513 0.34099 0.40259 0.38058 0.34126
## [3,] 0.41048 0.34661 0.38272 0.34326 0.32231 0.33804 0.38680 0.37459 0.34338
##        [,37]   [,38]   [,39]   [,40]   [,41]   [,42]   [,43]   [,44]   [,45]
## [1,] 0.42170 0.33910 0.41640 0.33270 0.35090 0.45580 0.37380 0.33200 0.37110
## [2,] 0.42769 0.35082 0.42678 0.33757 0.35845 0.46561 0.37407 0.33797 0.37147
## [3,] 0.41005 0.33000 0.40727 0.33743 0.34929 0.44446 0.36140 0.32190 0.37423
##        [,46]   [,47]   [,48]   [,49]   [,50]   [,51]   [,52]   [,53]   [,54]
## [1,] 0.34550 0.34430 0.34560 0.33070 0.36810 0.41670 0.32010 0.39610 0.31490
## [2,] 0.35771 0.34478 0.35042 0.33964 0.36435 0.41325 0.32232 0.40236 0.32017
## [3,] 0.34784 0.34447 0.33693 0.33840 0.35100 0.40353 0.32367 0.38781 0.31537
##        [,55]   [,56]   [,57]   [,58]   [,59]   [,60]   [,61]   [,62]   [,63]
## [1,] 0.39090 0.29040 0.49170 0.40700 0.39480 0.31930 0.38410 0.33940 0.31820
## [2,] 0.39946 0.29253 0.49888 0.41487 0.40148 0.30084 0.39554 0.34709 0.32609
## [3,] 0.39610 0.27336 0.46855 0.40835 0.38931 0.31105 0.37926 0.33358 0.30415
##        [,64]   [,65]   [,66]   [,67]   [,68]   [,69]   [,70]   [,71]   [,72]
## [1,] 0.45020 0.42870 0.31760 0.33030 0.35680 0.32500 0.43120 0.34450 0.26850
## [2,] 0.44869 0.44342 0.32080 0.33829 0.35855 0.32154 0.43327 0.35898 0.26670
## [3,] 0.44109 0.43549 0.30749 0.31514 0.36395 0.31271 0.42515 0.34706 0.25801
##        [,73]   [,74]   [,75]   [,76]  [,77]   [,78]   [,79]   [,80]   [,81]
## [1,] 0.38110 0.36110 0.29470 0.37560 0.3761 0.35150 0.30750 0.39000 0.33480
## [2,] 0.38219 0.36279 0.28958 0.38825 0.3820 0.35321 0.30922 0.39361 0.33941
## [3,] 0.37185 0.34755 0.28094 0.37718 0.3686 0.34227 0.29598 0.38652 0.33594
##        [,82]   [,83]   [,84]   [,85]   [,86]   [,87]   [,88]   [,89]   [,90]
## [1,] 0.37190 0.35050 0.40950 0.31230 0.40530 0.39290 0.44360 0.34680 0.41070
## [2,] 0.38210 0.35043 0.40995 0.31017 0.40622 0.39467 0.45474 0.33993 0.41604
## [3,] 0.36791 0.32955 0.40125 0.29919 0.40869 0.39349 0.44182 0.34051 0.41605
##        [,91]   [,92]   [,93]   [,94]   [,95]   [,96]   [,97]   [,98]   [,99]
## [1,] 0.31310 0.36370 0.41610 0.37330 0.26360 0.30040 0.31140 0.37170 0.36840
## [2,] 0.30188 0.37197 0.41931 0.36394 0.25857 0.31087 0.31163 0.37733 0.38089
## [3,] 0.30324 0.35232 0.40535 0.36573 0.25170 0.28863 0.29961 0.36392 0.36390
##       [,100]
## [1,] 0.41760
## [2,] 0.42070
## [3,] 0.39763
```

```r
acc <- data.frame(t(acc.result))
boxplot(acc, xlab = 'Model', ylab = "Prediction Accuracy")
```




```
apply(acc.result, 1, mean)
```

```
## [1] 0.3631970 0.3670436 0.3574247
```

Evaluation - Discussion of what you saw/did

Through variable selection, I selected variables. The optimal model was to use 7 variables, and the BP Test was used to test the equal variance of this model. However, because the assumptions were not met, models using WLS and Box-Cox were created. All three models have good R-squared values. To validate these models, I divided the data into test set and train set. Finally, the accuaracy was calculated through prediction. As a result, the third model is the best. The model using Box-Cox is the best model for predicting movie ratings.