DATA621-FinalProject-SmoothOperators

Rob Hodde, Matt Farris, Jeffrey Burmood, Bin Lin

5/26/2017

Problem Description

Our final project will explore, analyze and model a data set containing information on approximately 5,000 movies. The dataset contains movie data extracted from the IMDB website and is available on Kaggle.com.

The project will develop predictive models for two questions:

1) Will the movie make money, lose money, or break even (approximately)?
2) What is the anticipated gross margin (profit) for the movie?

Data Exploration

Data Exploration

To this point we've removed the data columns for the variables that we will not be using in the analysis. The columns remaining in the data set are the following:

```
##  [1] "duration"                "director_facebook_likes"
##  [3] "actor_3_facebook_likes"  "actor_1_facebook_likes"
##  [5] "gross"                   "movie_title"
##  [7] "num_voted_users"         "cast_total_facebook_likes"
##  [9] "facenumber_in_poster"    "content_rating"
## [11] "budget"                  "title_year"
## [13] "actor_2_facebook_likes"  "imdb_score"
```

After exploring the data, we noticed there is a scattering of NAs across the variables. Due to the relatively low number of total NAs, we choose to remove all rows with NAs, leaving 3,828 rows of data.

Next we will explore the nature of the data for the variables we will be using in the analysis.

VAR	TYPE
duration	integer
director_facebook_likes	integer
actor_3_facebook_likes	integer
actor_1_facebook_likes	integer
gross	integer
movie_title	character
num_voted_users	integer
cast_total_facebook_likes	integer
facenumber_in_poster	integer
content_rating	character
budget	double
title_year	integer
actor_2_facebook_likes	integer
imdb_score	double

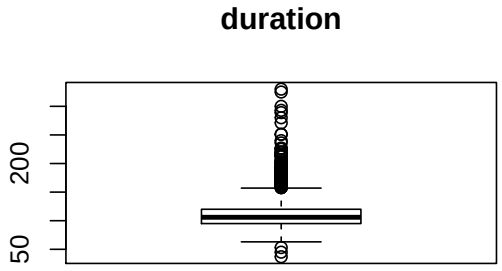
```
##     duration   director_facebook_likes actor_3_facebook_likes
```

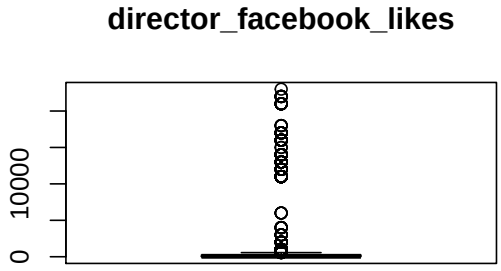
```
##  Min.   : 37   Min.   :    0.0         Min.   :    0.0
##  1st Qu.: 95   1st Qu.:   10.0         1st Qu.:  188.8
##  Median :106   Median :   60.0         Median :  433.0
##  Mean   :110   Mean   :  792.9         Mean   :  761.8
##  3rd Qu.:120   3rd Qu.:  232.5         3rd Qu.:  690.0
##  Max.   :330   Max.   :23000.0         Max.   :23000.0
##
##  actor_1_facebook_likes     gross           movie_title
##  Min.   :     0.0       Min.   :      162   Length:3828
##  1st Qu.:   737.5       1st Qu.:  7452337   Class :character
##  Median :  1000.0       Median : 28854152   Mode  :character
##  Mean   :  7664.1       Mean   : 51694432
##  3rd Qu.: 12250.0       3rd Qu.: 66004138
##  Max.   :640000.0       Max.   :760505847
##
##  num_voted_users   cast_total_facebook_likes facenumber_in_poster
##  Min.   :     22   Min.   :     0            Min.   : 0.000
##  1st Qu.:  18267   1st Qu.:  1880            1st Qu.: 0.000
##  Median :  52380   Median :  3962            Median : 1.000
##  Mean   : 103908   Mean   : 11396            Mean   : 1.379
##  3rd Qu.: 125643   3rd Qu.: 16128            3rd Qu.: 2.000
##  Max.   :1689764   Max.   :656730            Max.   :43.000
##
##    content_rating     budget            title_year
##  R        :1736   Min.   :2.180e+02   Min.   :1927
##  PG-13    :1326   1st Qu.:1.000e+07   1st Qu.:1999
##  PG       : 574   Median :2.500e+07   Median :2005
##  G        :  89   Mean   :4.548e+07   Mean   :2003
##  Not Rated:  40   3rd Qu.:5.000e+07   3rd Qu.:2010
##  Unrated  :  24   Max.   :1.222e+10   Max.   :2016
##  (Other)  :  39
##  actor_2_facebook_likes   imdb_score
##  Min.   :     0.0       Min.   :1.600
##  1st Qu.:   373.8       1st Qu.:5.900
##  Median :   677.0       Median :6.600
##  Mean   :  1994.6       Mean   :6.459
##  3rd Qu.:   975.0       3rd Qu.:7.200
##  Max.   :137000.0       Max.   :9.300
##
```

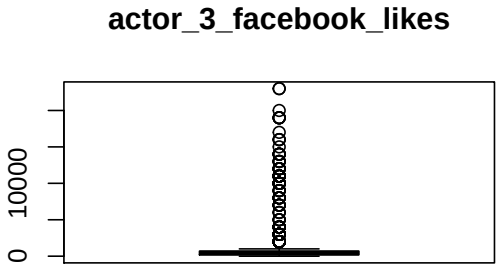
	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes
duration	1.0000000	0.1822411	0.1279962	0.0863409
director_facebook_likes	0.1822411	1.0000000	0.1184843	0.0905543
actor_3_facebook_likes	0.1279962	0.1184843	1.0000000	0.2526590
actor_1_facebook_likes	0.0863409	0.0905543	0.2526590	1.0000000
num_voted_users	0.3434487	0.3013255	0.2697667	0.1817812
cast_total_facebook_likes	0.1232351	0.1197195	0.4895509	0.9450371
facenumber_in_poster	0.0263907	-0.0478417	0.1055483	0.0614101
budget	0.0696018	0.0189881	0.0408678	0.0173849
title_year	-0.1311001	-0.0464926	0.1144145	0.0929673
actor_2_facebook_likes	0.1311685	0.1172937	0.5540722	0.3910139
imdb_score	0.3655775	0.1915761	0.0661996	0.0939598

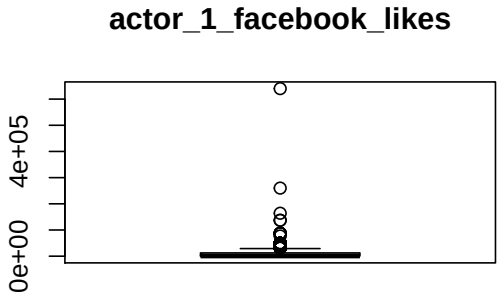
	num_voted_users	cast_total_facebook_likes	facenumber_in_poster	budget
duration	0.3434487	0.1232351	0.0263907	0.0696018
director_facebook_likes	0.3013255	0.1197195	-0.0478417	0.0189881
actor_3_facebook_likes	0.2697667	0.4895509	0.1055483	0.0408678
actor_1_facebook_likes	0.1817812	0.9450371	0.0614101	0.0173849
num_voted_users	1.0000000	0.2516946	-0.0324633	0.0678793
cast_total_facebook_likes	0.2516946	1.0000000	0.0837393	0.0298442
facenumber_in_poster	-0.0324633	0.0837393	1.0000000	-0.0215767
budget	0.0678793	0.0298442	-0.0215767	1.0000000
title_year	0.0172947	0.1230087	0.0716142	0.0452068
actor_2_facebook_likes	0.2473172	0.6424574	0.0720087	0.0367048
imdb_score	0.4792715	0.1073363	-0.0671658	0.0298854

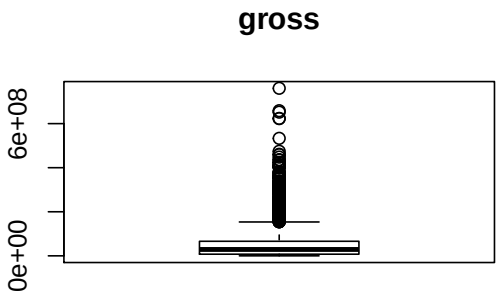
	title_year	actor_2_facebook_likes
duration	-0.1311001	0.1311685
director_facebook_likes	-0.0464926	0.1172937
actor_3_facebook_likes	0.1144145	0.5540722
actor_1_facebook_likes	0.0929673	0.3910139
num_voted_users	0.0172947	0.2473172
cast_total_facebook_likes	0.1230087	0.6424574
facenumber_in_poster	0.0716142	0.0720087
budget	0.0452068	0.0367048
title_year	1.0000000	0.1186388
actor_2_facebook_likes	0.1186388	1.0000000
imdb_score	-0.1357930	0.1031776

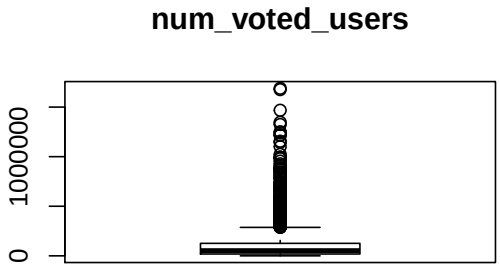


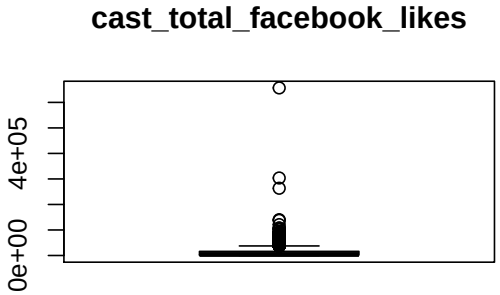


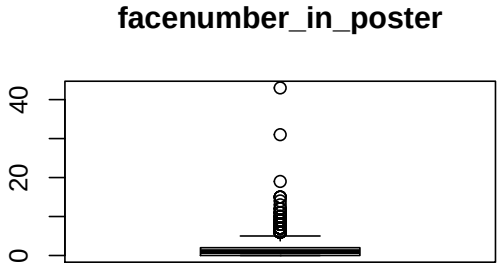

actor_3_facebook_likes
0
10000

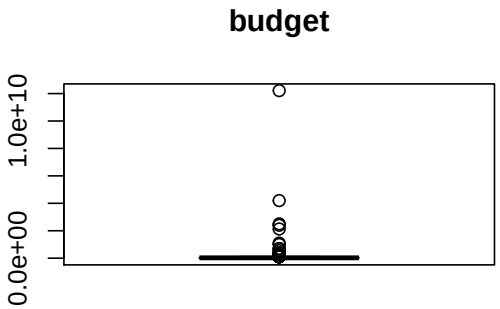

actor_1_facebook_likes
0e+00
4e+05

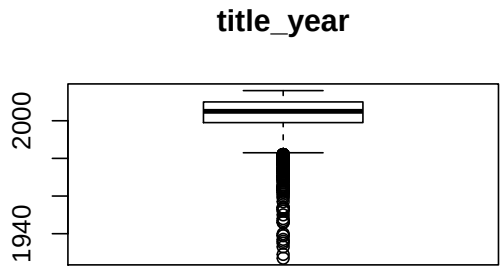

gross
0e+00
6e+08

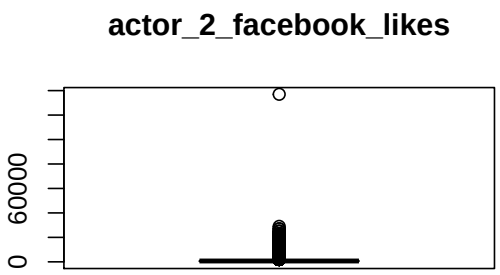

num_voted_users
0
1000000

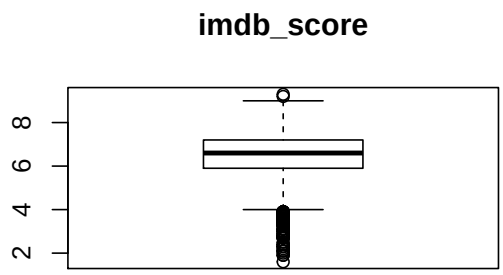

cast_total_facebook_likes
0e+00
4e+05

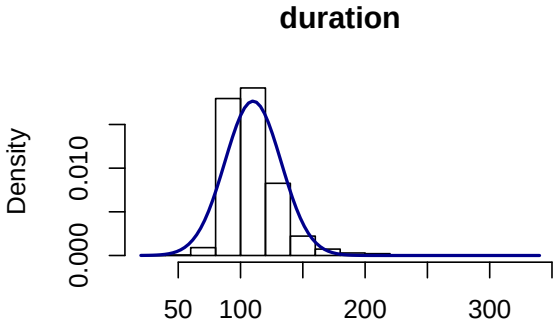

facenumber_in_poster
0
20
40

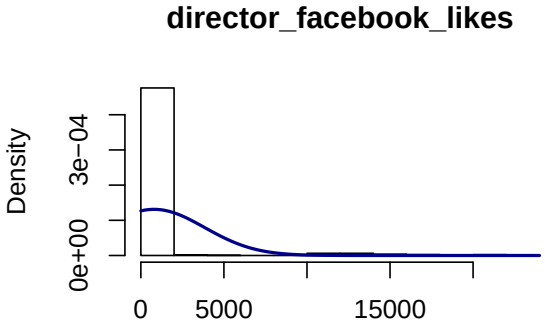

budget
0.0e+00
1.0e+10

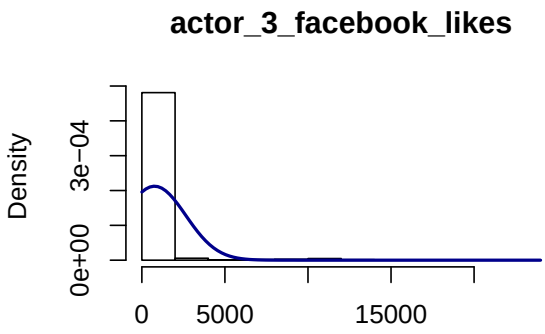

title_year
1940
2000

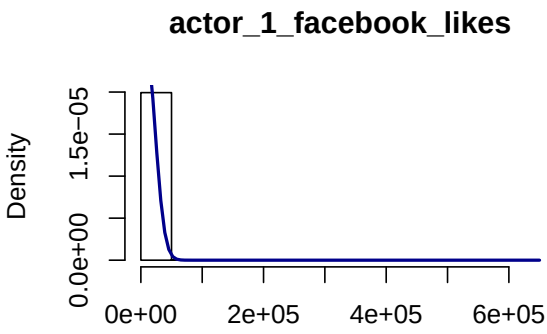

actor_2_facebook_likes
0
60000

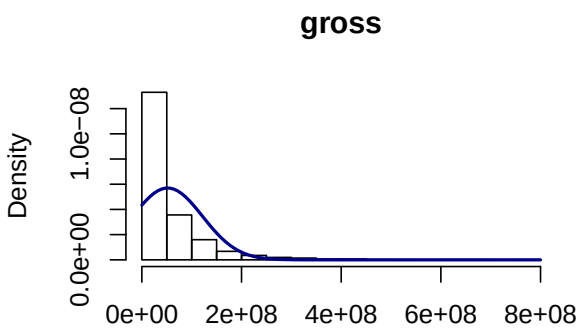

imdb_score
2
4
6
8

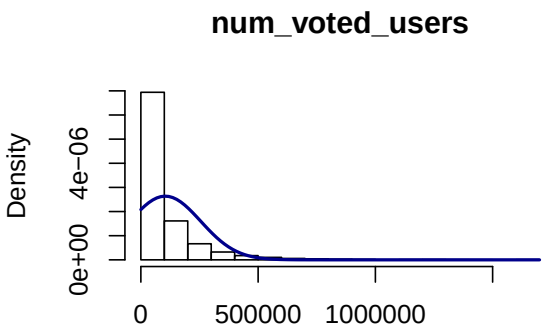

duration
Density
0.000
0.010
50
100
200
300

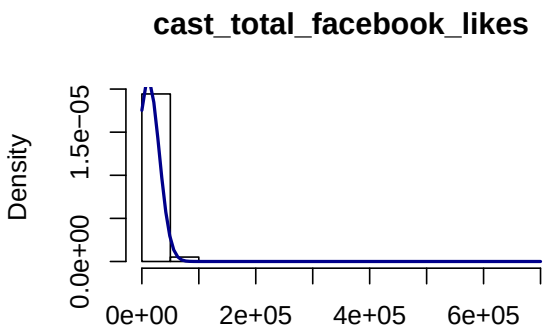

director_facebook_likes
Density
0e+00
3e−04
0
5000
15000

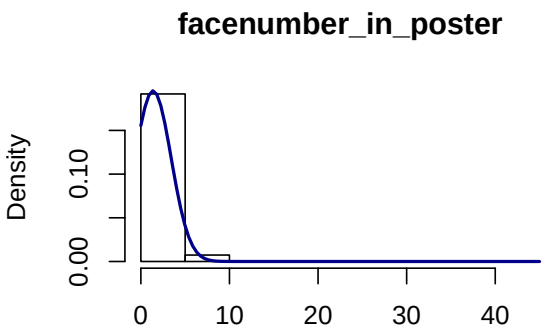

actor_3_facebook_likes
Density
0e+00
3e−04
0
5000
15000

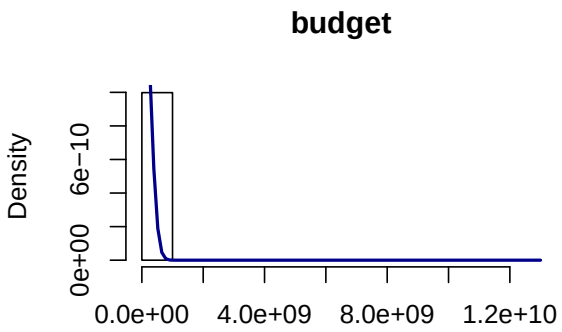

actor_1_facebook_likes
Density
0.0e+00
1.5e−05
0e+00
2e+05
4e+05
6e+05

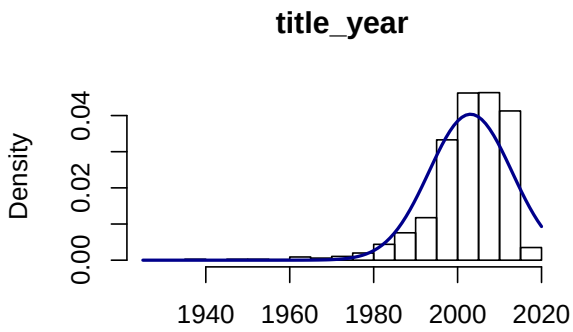

gross
Density
0.0e+00
1.0e−08
0e+00
2e+08
4e+08
6e+08
8e+08

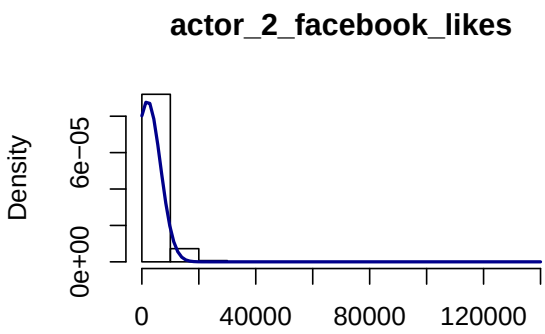

num_voted_users
Density
0e+00
4e−06
0
500000
1000000

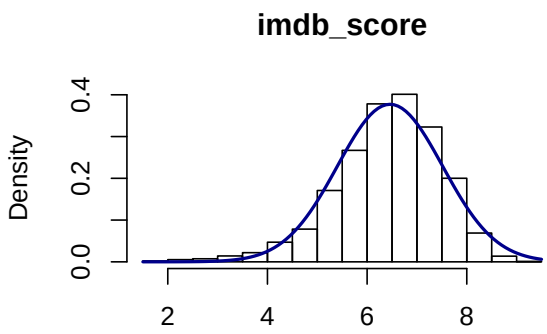

cast_total_facebook_likes
Density
0.0e+00
1.5e−05
0e+00
2e+05
4e+05
6e+05

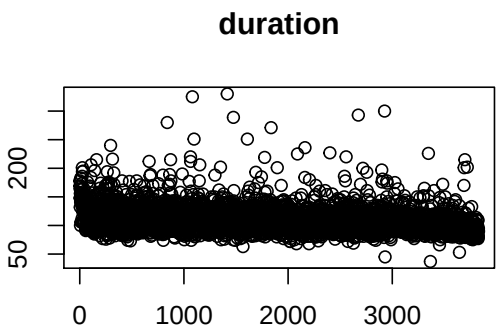

facenumber_in_poster
Density
0.00
0.10
0
10
20
30
40

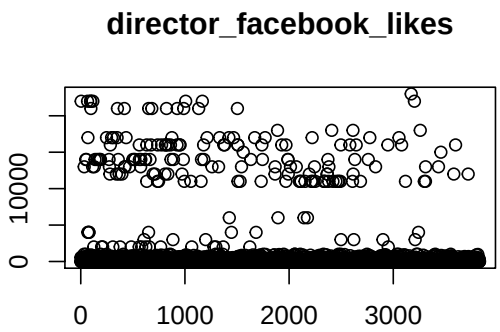

budget
Density
0e+00
6e−10
0.0e+00
4.0e+09
8.0e+09
1.2e+10

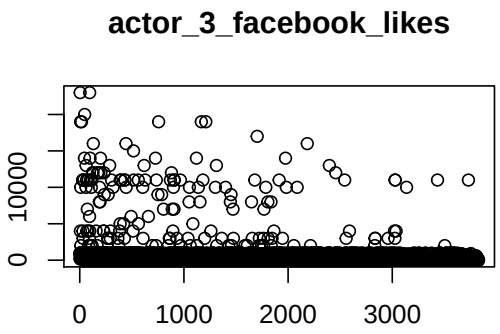

title_year
Density
0.00
0.02
0.04
1940
1960
1980
2000
2020

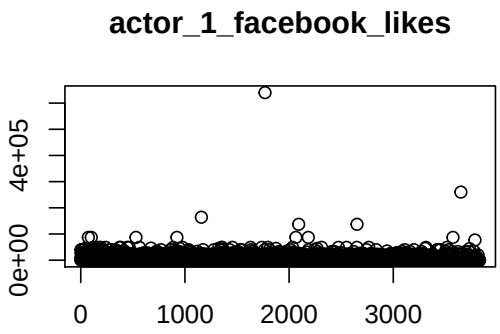

actor_2_facebook_likes
Density
0e+00
6e−05
0
40000
80000
120000

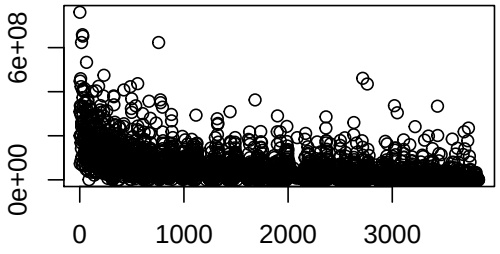

imdb_score
Density
0.0
0.2
0.4
2
4
6
8

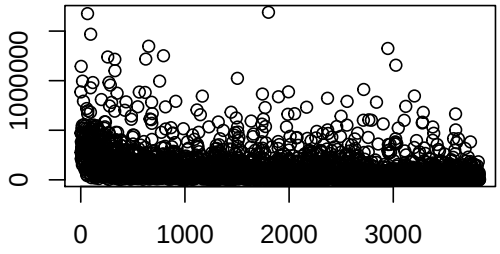

duration
50
200
0
1000
2000
3000

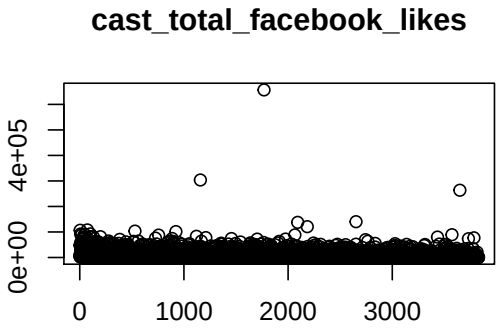

director_facebook_likes
0
10000
0
1000
2000
3000

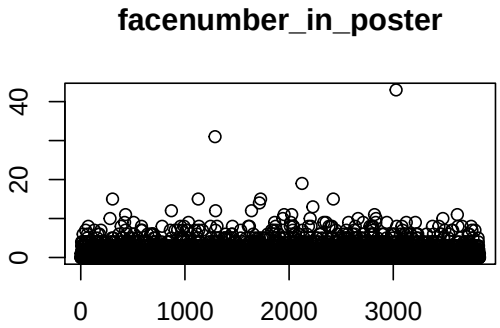

actor_3_facebook_likes
0
10000
0
1000
2000
3000

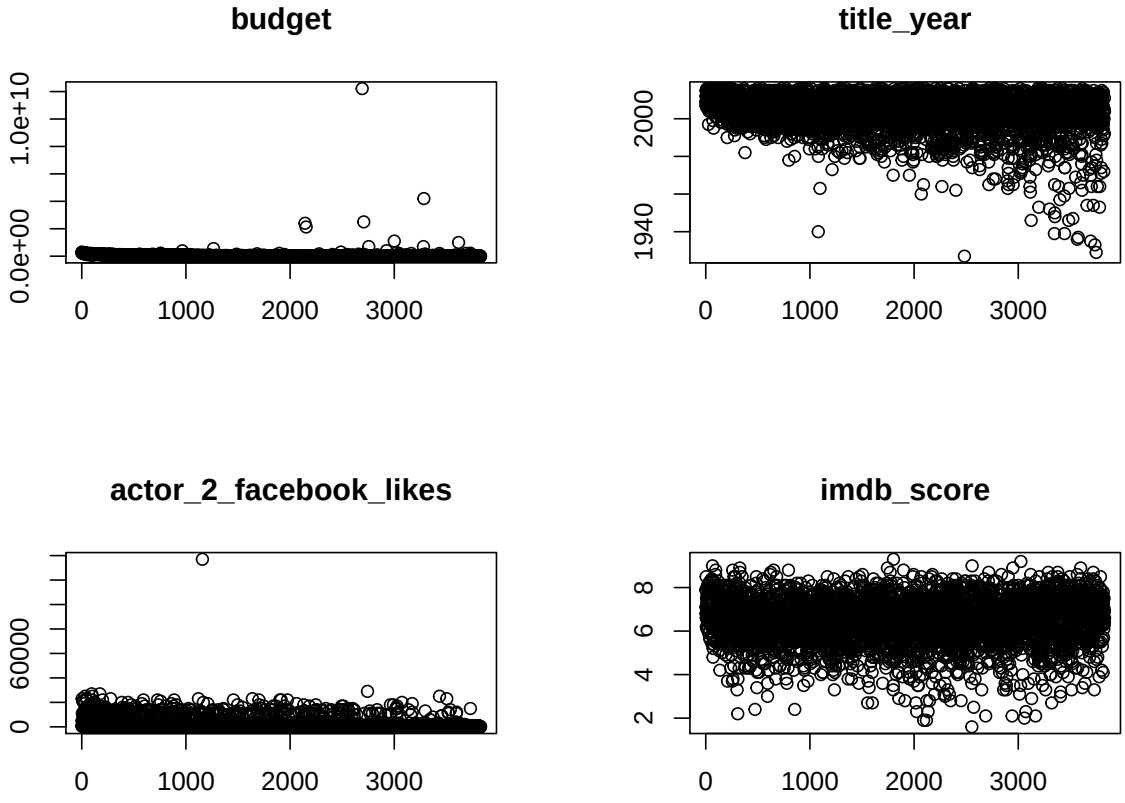

actor_1_facebook_likes
0e+00
4e+05
0
1000
2000
3000


gross
0e+00
6e+08
0
1000
2000
3000


num_voted_users
0
1000000
0
1000
2000
3000


cast_total_facebook_likes
0e+00
4e+05
0
1000
2000
3000


facenumber_in_poster
0
20
40
0
1000
2000
3000



As we can see from the plots and statistical summary, most of the variables have a reasonable distribution except those variable associated with the Facebook likes. There are five variables related to Facebook likes that are highly skewed due to a large number of zeros. At this point we assume these zeros represent NAs in the Facebook data.

Next, we'll use the mice package to impute the Facebook likes data for the zeros/NAs.

```
##      actor_1_facebook_likes cast_total_facebook_likes
## 3142                      1                         1
##  656                      1                         1
##   14                      1                         1
##    3                      1                         1
##    9                      1                         1
##    2                      1                         1
##    1                      0                         0
##    1                      0                         0
##                           2                         2
##      actor_2_facebook_likes actor_3_facebook_likes director_facebook_likes
## 3142                      1                       1                       1
##  656                      1                       1                       0
##   14                      1                       0                       1
##    3                      1                       0                       0
##    9                      0                       0                       1
##    2                      0                       0                       0
##    1                      0                       0                       1
```

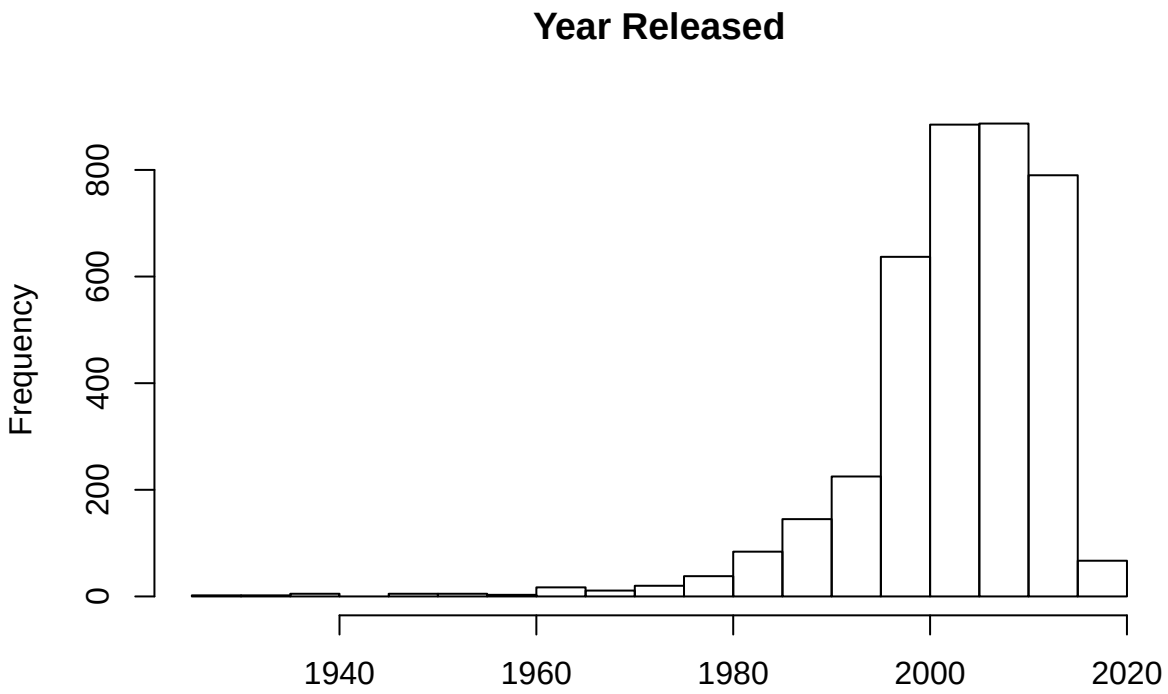
```
##    1                     0                     0                     0
##                         13                    30                   662
##
## 3142   0
##  656   1
##   14   1
##    3   2
##    9   2
##    2   3
##    1   4
##    1   5
##      709
```

```
##     duration   director_facebook_likes actor_3_facebook_likes
##  Min.   : 37   Min.   :    2.0         Min.   :    2
##  1st Qu.: 95   1st Qu.:   31.0         1st Qu.:  189
##  Median :106   Median :   98.0         Median :  433
##  Mean   :110   Mean   :  978.8         Mean   :  762
##  3rd Qu.:120   3rd Qu.:  309.0         3rd Qu.:  690
##  Max.   :330   Max.   :23000.0         Max.   :23000
##
##  actor_1_facebook_likes     gross           movie_title
##  Min.   :     2.0       Min.   :      162   Length:3828
##  1st Qu.:   739.5       1st Qu.:  7452337   Class :character
##  Median :  1000.0       Median : 28854152   Mode  :character
##  Mean   :  7677.2       Mean   : 51694432
##  3rd Qu.: 13000.0       3rd Qu.: 66004138
##  Max.   :640000.0       Max.   :760505847
##
##  num_voted_users   cast_total_facebook_likes facenumber_in_poster
##  Min.   :     22   Min.   :     2            Min.   : 0.000
##  1st Qu.:  18267   1st Qu.:  1884            1st Qu.: 0.000
##  Median :  52380   Median :  3965            Median : 1.000
##  Mean   : 103908   Mean   : 11409            Mean   : 1.379
##  3rd Qu.: 125643   3rd Qu.: 16144            3rd Qu.: 2.000
##  Max.   :1689764   Max.   :656730            Max.   :43.000
##
##    content_rating     budget            title_year
##  R        :1736   Min.   :2.180e+02   Min.   :1927
##  PG-13    :1326   1st Qu.:1.000e+07   1st Qu.:1999
##  PG       : 574   Median :2.500e+07   Median :2005
##  G        :  89   Mean   :4.548e+07   Mean   :2003
##  Not Rated:  40   3rd Qu.:5.000e+07   3rd Qu.:2010
##  Unrated  :  24   Max.   :1.222e+10   Max.   :2016
##  (Other)  :  39
##  actor_2_facebook_likes   imdb_score
##  Min.   :     2.0       Min.   :1.600
##  1st Qu.:   374.0       1st Qu.:5.900
##  Median :   677.5       Median :6.600
##  Mean   :  1994.8       Mean   :6.459
##  3rd Qu.:   975.0       3rd Qu.:7.200
##  Max.   :137000.0       Max.   :9.300
##
```

Data Preparation

Data Preparation

One of the big issues faced in when using this dataset is the time frame. These movies were collected over the past 80+ years, and the following shows our distribution over time:



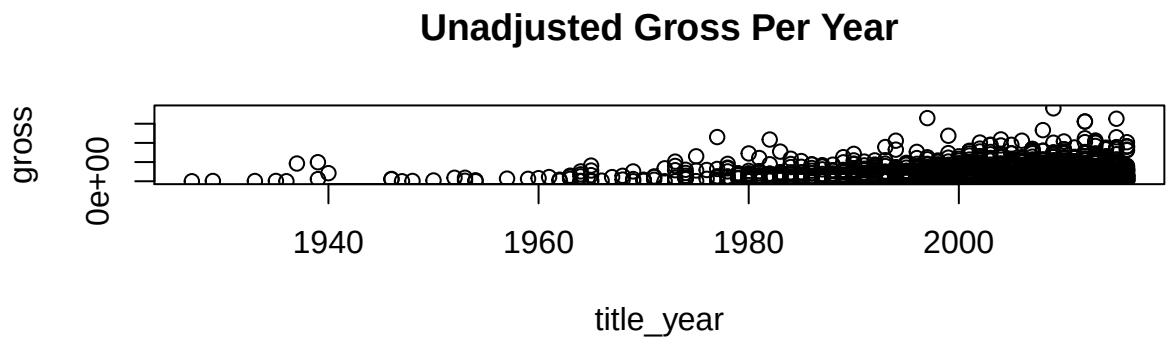
As you can see, the vast majority came from 1990s and above, but we can't discredit the movies from previous year. In order to accurately portray elements from the past, we have instituted a rate of inflation calculation. Using the consumer price index (for our part here we are making a crucial assumption, that all dollars are calculated based on US currency, and we are ignoring even more complex foreign exchange rates of the time), we can calculate the gross value per year. As a basis of comparison, we are using the CPI index from 2016, as the last movie was made in 2016.

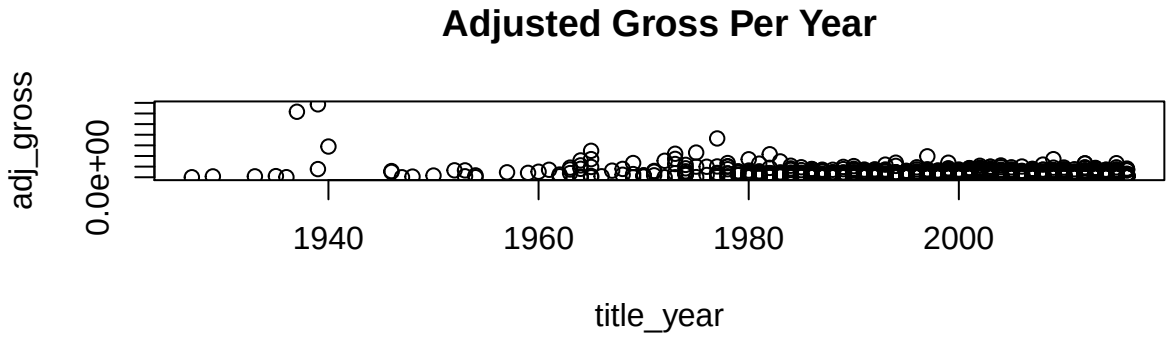
```
movies <- merge(x = movies, y = cpi, by = "title_year")
movies$adj_gross <- with(movies, (240/cpi * gross))
movies$adj_budget <- with(movies, (240/cpi * budget))
movies$adj_margin <- with(movies, adj_gross-adj_budget)
```

```
attach(movies)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     cpi
```

```
par(mfrow=c(2,1))
plot(title_year,gross, main="Unadjusted Gross Per Year")
plot(title_year,adj_gross,main="Adjusted Gross Per Year")
```





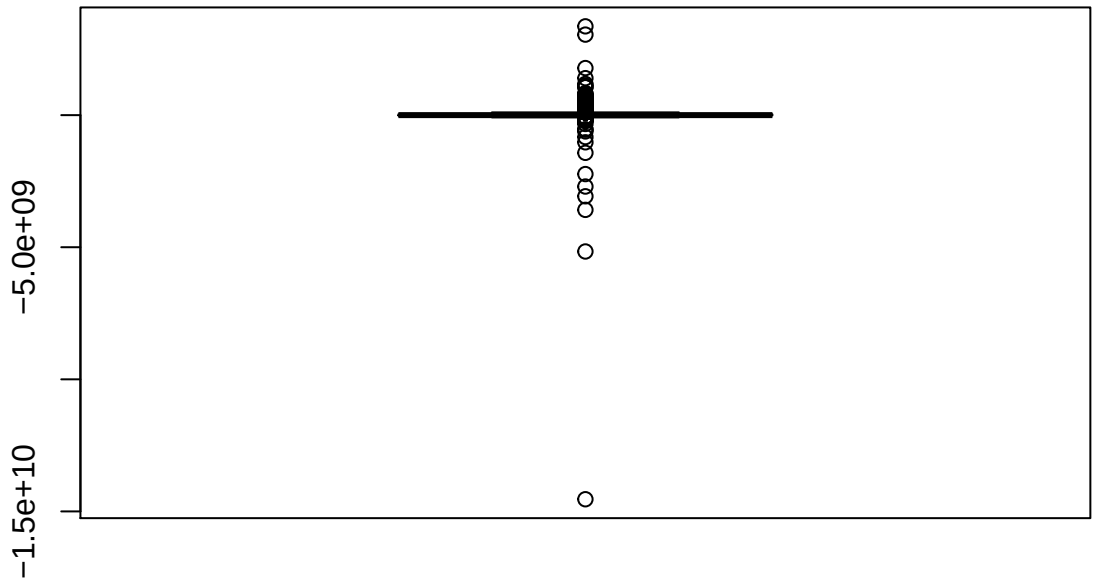
From the above graphs, we can see that the adjustment for the gross did indeed create a more uniformed dataset (where as before we saw movies increasing over the years). As a point of interest, the movies that made over a billion dollars are shown below:

```
highest_gross <- subset(movies, adj_gross > 1000000000, select=c("movie_title", "gross", "adj_gross"))
highest_gross
```

```
##                            movie_title     gross  adj_gross
## 6      Snow White and the Seven Dwarfs 184925485 3082091417
## 7                   Gone with the Wind 198655278 3430019188
## 9                             Pinocchio  84300000 1445142857
## 37                 The Sound of Music 163214286 1243537417
## 59                      The Exorcist 204565000 1105756757
## 69                              Jaws 260000000 1159851301
## 78  Star Wars: Episode IV - A New Hope 460935665 1825487782
## 133         E.T. the Extra-Terrestrial 434949459 1081739587
```

A quick Google search indicates that the above movies are consistently listed the top grossing movies of all time. Furthermore, our "estimated adjusted gross" mimics the findings that we see with adjusted gross (for the most part, there are two schools of thought on how to adjust gross, using ticket prices or our method adjusting based on CPI). Though our dollar amount vary slightly from other sources, any variance is consistent across our datase.

```
boxplot(movies$adj_margin)
```



Build Models

Build Models

Binomial Regression

Our first model we want to investigate is whether or not we can predict if film will make money given the cast and direction. To do this, we decided to create a binary regression model, transforming our adjusted

```
#Creating the Binomial Model
bin_movie <- glm(money ~ ., family=binomial(link='logit'),data=train)
summary(bin_movie)
```

```
##
## Call:
## glm(formula = money ~ ., family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5245  -1.0589   0.3362   1.0895   2.0627
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 8.565e+01  9.207e+00   9.303  < 2e-16 ***
## title_year                 -4.223e-02  4.578e-03  -9.225  < 2e-16 ***
## duration                   -1.636e-02  2.139e-03  -7.647 2.06e-14 ***
```

```
## director_facebook_likes   -2.614e-05  1.405e-05  -1.861  0.06278 .
## actor_3_facebook_likes    -2.381e-04  7.505e-05  -3.173  0.00151 **
## actor_1_facebook_likes    -2.184e-04  5.016e-05  -4.355 1.33e-05 ***
## num_voted_users            8.816e-06  5.703e-07  15.459  < 2e-16 ***
## cast_total_facebook_likes  2.133e-04  5.000e-05   4.266 1.99e-05 ***
## facenumber_in_poster       2.986e-02  2.163e-02   1.381  0.16741
## actor_2_facebook_likes    -2.256e-04  5.268e-05  -4.284 1.84e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4228.4  on 3061  degrees of freedom
## Residual deviance: 3741.9  on 3052  degrees of freedom
## AIC: 3761.9
##
## Number of Fisher Scoring iterations: 5
```

```
pred_col <- c(1,2,3,4,5,7,8,9,11)
p <- predict(bin_movie, newdata=test, type = "response")
pr <- prediction(p, test$money)
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.7714305
```

Using all the prediction variables at hand, the model accurately predicts 76% of the time. Using backward stepwise regression, we attempted to remove some variables that may not have had significance in our model.

```
backward <- step(bin_movie)
```

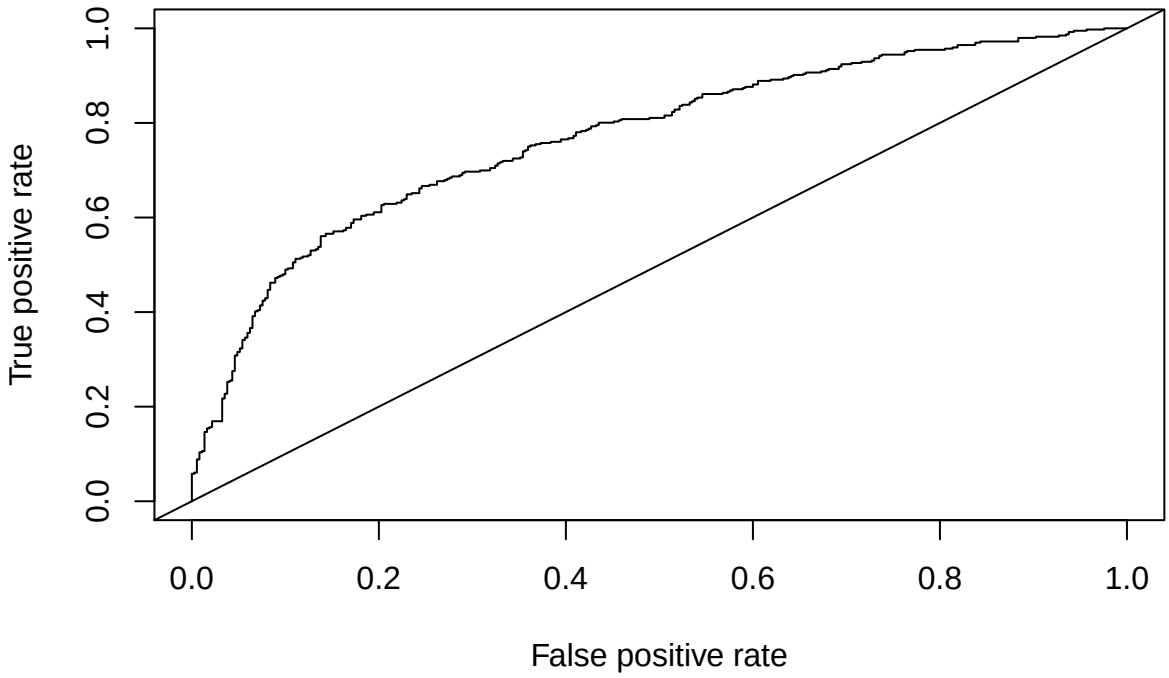
```
## Start:  AIC=3761.86
## money ~ title_year + duration + director_facebook_likes + actor_3_facebook_likes +
##     actor_1_facebook_likes + num_voted_users + cast_total_facebook_likes +
##     facenumber_in_poster + actor_2_facebook_likes
##
##                             Df Deviance    AIC
## - facenumber_in_poster       1   3743.8 3761.8
## <none>                           3741.9 3761.9
## - director_facebook_likes    1   3745.3 3763.3
## - actor_3_facebook_likes     1   3752.1 3770.1
## - cast_total_facebook_likes  1   3761.4 3779.4
## - actor_2_facebook_likes     1   3761.7 3779.7
## - actor_1_facebook_likes     1   3762.3 3780.3
## - duration                   1   3805.3 3823.3
## - title_year                 1   3837.4 3855.4
## - num_voted_users            1   4120.1 4138.1
##
## Step:  AIC=3761.77
## money ~ title_year + duration + director_facebook_likes + actor_3_facebook_likes +
##     actor_1_facebook_likes + num_voted_users + cast_total_facebook_likes +
##     actor_2_facebook_likes
##
##                             Df Deviance    AIC
## <none>                           3743.8 3761.8
```

```
## - director_facebook_likes    1   3747.3 3763.3
## - actor_3_facebook_likes     1   3754.2 3770.2
## - cast_total_facebook_likes  1   3764.1 3780.1
## - actor_2_facebook_likes     1   3764.3 3780.3
## - actor_1_facebook_likes     1   3765.0 3781.0
## - duration                   1   3806.2 3822.2
## - title_year                 1   3837.9 3853.9
## - num_voted_users            1   4120.2 4136.2
```

```
summary(backward)
```

```
##
## Call:
## glm(formula = money ~ title_year + duration + director_facebook_likes +
##     actor_3_facebook_likes + actor_1_facebook_likes + num_voted_users +
##     cast_total_facebook_likes + actor_2_facebook_likes, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5012  -1.0609   0.3267   1.0856   2.0383
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 8.466e+01  9.154e+00   9.248  < 2e-16 ***
## title_year                 -4.173e-02  4.551e-03  -9.169  < 2e-16 ***
## duration                   -1.618e-02  2.130e-03  -7.597 3.03e-14 ***
## director_facebook_likes    -2.643e-05  1.403e-05  -1.884  0.05957 .
## actor_3_facebook_likes     -2.403e-04  7.504e-05  -3.203  0.00136 **
## actor_1_facebook_likes     -2.227e-04  5.021e-05  -4.436 9.17e-06 ***
## num_voted_users             8.749e-06  5.669e-07  15.432  < 2e-16 ***
## cast_total_facebook_likes   2.177e-04  5.004e-05   4.351 1.35e-05 ***
## actor_2_facebook_likes     -2.300e-04  5.269e-05  -4.364 1.28e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4228.4  on 3061  degrees of freedom
## Residual deviance: 3743.8  on 3053  degrees of freedom
## AIC: 3761.8
##
## Number of Fisher Scoring iterations: 5
```

```
p <- predict(backward, newdata=test, type="response")
pr <- prediction(p, test$money)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
auc_back <- performance(pr, measure = "auc")
auc_back <- auc_back@y.values[[1]]
plot(prf)
abline(a = 0, b = 1)
```




```
auc_back
```

```
## [1] 0.7697857
```

Profit Margin Model

```
#Elminate title_year, gross, budget, cpi
movies_new <- Filter(is.numeric, movies)
profit_margin <- movies_new$adj_margin / movies_new$adj_gross
movies_new <- cbind(movies_new, profit_margin)

movies_new <- subset(movies_new,select = -c(1, 6, 10, 12, 13))

##Also exclude adj_margin profit_margin when building models for gross prediction, because they are simp
m1 <- lm(adj_gross ~. - adj_margin - profit_margin, data = movies_new)
summary(m1)
```

```
##
## Call:
## lm(formula = adj_gross ~ . - adj_margin - profit_margin, data = movies_new)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -632723971  -36574797  -18508723   14399942 3243481422
##
## Coefficients:
```

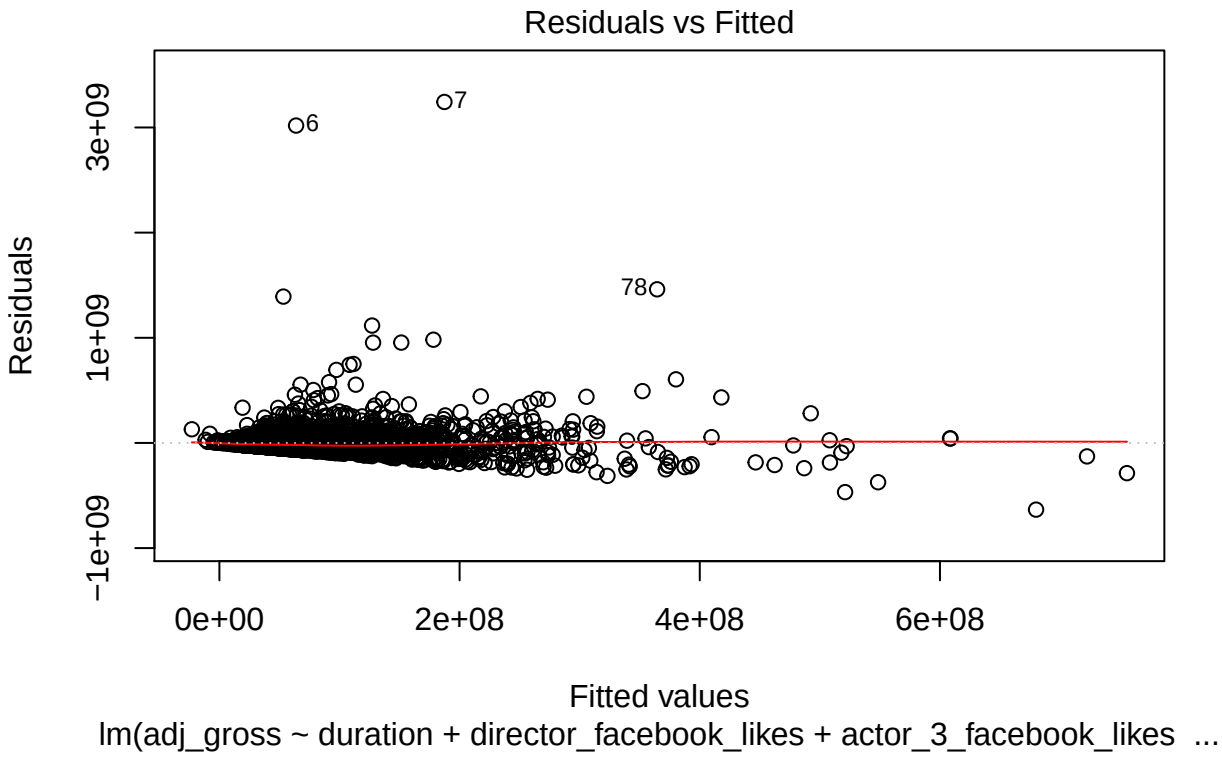
```
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -3.367e+07  9.614e+06  -3.502 0.000466 ***
## duration                    6.017e+05  8.889e+04   6.769 1.49e-11 ***
## director_facebook_likes    -1.410e+03  5.889e+02  -2.395 0.016672 *
## actor_3_facebook_likes     -1.427e+04  2.871e+03  -4.970 6.98e-07 ***
## actor_1_facebook_likes     -1.143e+04  1.721e+03  -6.642 3.54e-11 ***
## num_voted_users             3.647e+02  1.415e+01  25.772  < 2e-16 ***
## cast_total_facebook_likes   1.122e+04  1.715e+03   6.541 6.90e-11 ***
## facenumber_in_poster       -2.004e+06  9.245e+05  -2.167 0.030261 *
## actor_2_facebook_likes     -1.125e+04  1.819e+03  -6.187 6.79e-10 ***
## adj_budget                  9.072e-03  6.899e-03   1.315 0.188599
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 115600000 on 3818 degrees of freedom
## Multiple R-squared:  0.2322, Adjusted R-squared:  0.2304
## F-statistic: 128.3 on 9 and 3818 DF,  p-value: < 2.2e-16
```

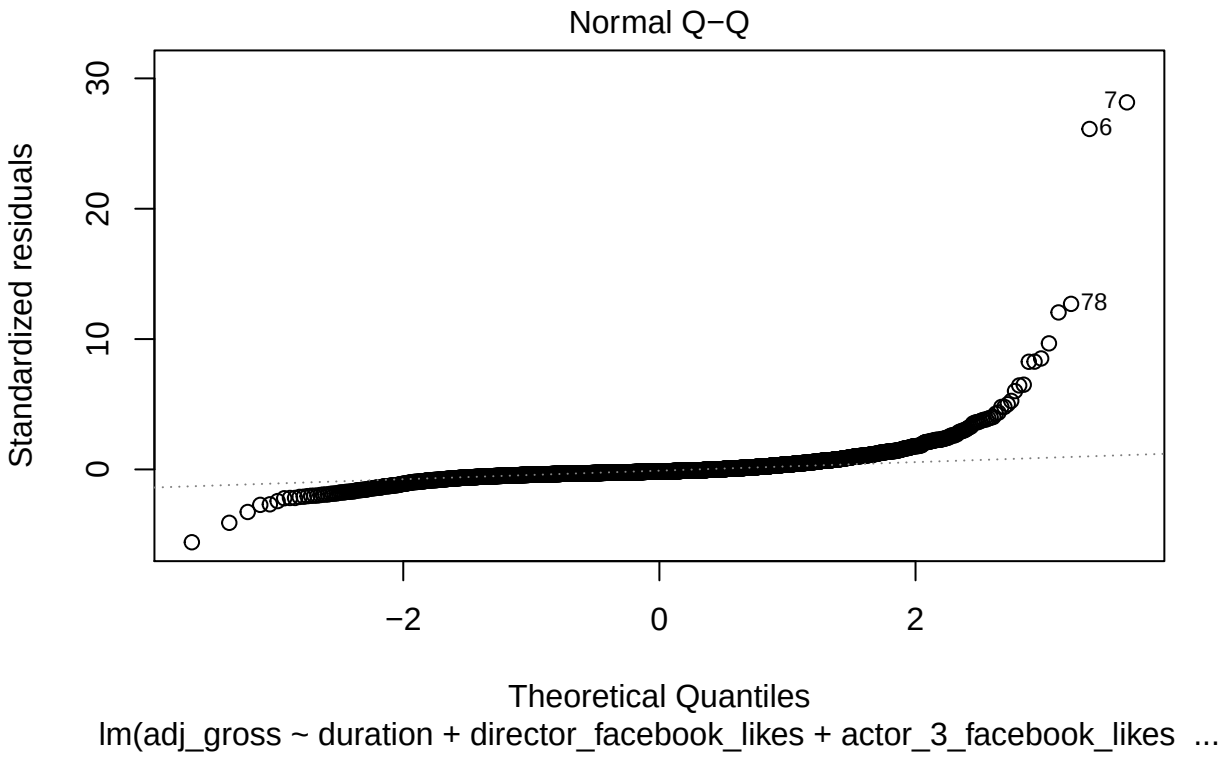
```
m1_back <- step(m1, trace = 0)
summary(m1_back)
```

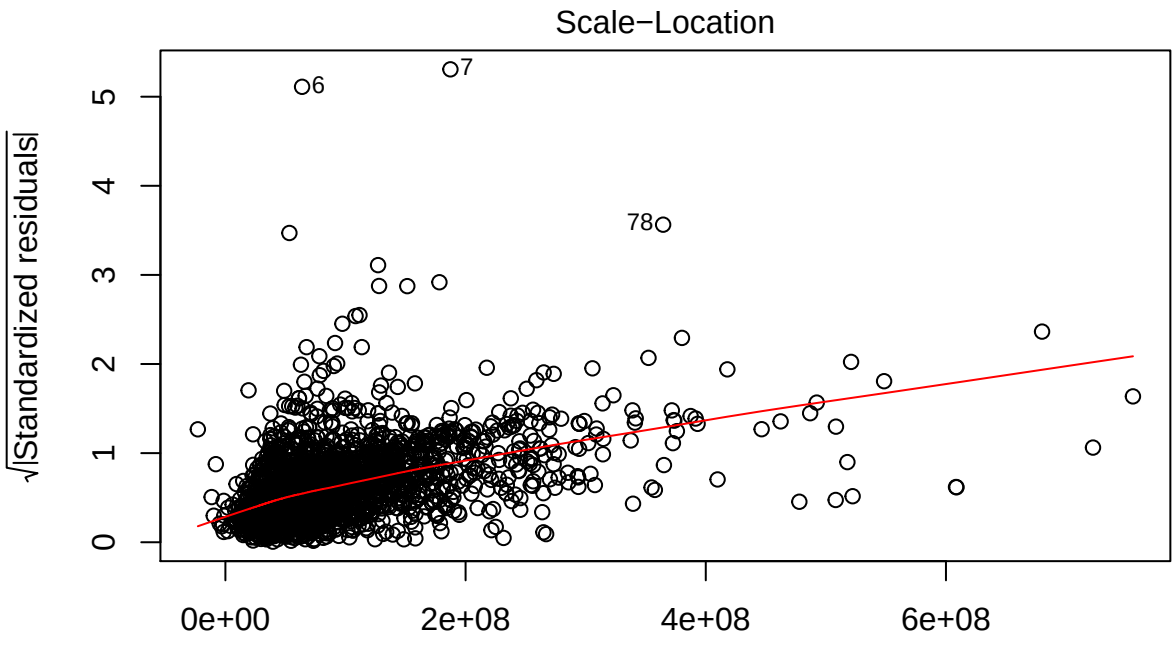
```
##
## Call:
## lm(formula = adj_gross ~ duration + director_facebook_likes +
##     actor_3_facebook_likes + actor_1_facebook_likes + num_voted_users +
##     cast_total_facebook_likes + facenumber_in_poster + actor_2_facebook_likes,
##     data = movies_new)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -634125413  -36721553  -18643066   14401963 3242652872
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -3.399e+07  9.611e+06  -3.537  0.00041 ***
## duration                    6.091e+05  8.872e+04   6.865 7.72e-12 ***
## director_facebook_likes    -1.415e+03  5.889e+02  -2.403  0.01631 *
## actor_3_facebook_likes     -1.427e+04  2.871e+03  -4.968 7.05e-07 ***
## actor_1_facebook_likes     -1.144e+04  1.721e+03  -6.647 3.41e-11 ***
## num_voted_users             3.653e+02  1.414e+01  25.827  < 2e-16 ***
## cast_total_facebook_likes   1.123e+04  1.715e+03   6.546 6.68e-11 ***
## facenumber_in_poster       -2.037e+06  9.242e+05  -2.204  0.02757 *
## actor_2_facebook_likes     -1.126e+04  1.819e+03  -6.189 6.69e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 115600000 on 3819 degrees of freedom
## Multiple R-squared:  0.2319, Adjusted R-squared:  0.2303
## F-statistic: 144.1 on 8 and 3819 DF,  p-value: < 2.2e-16
```

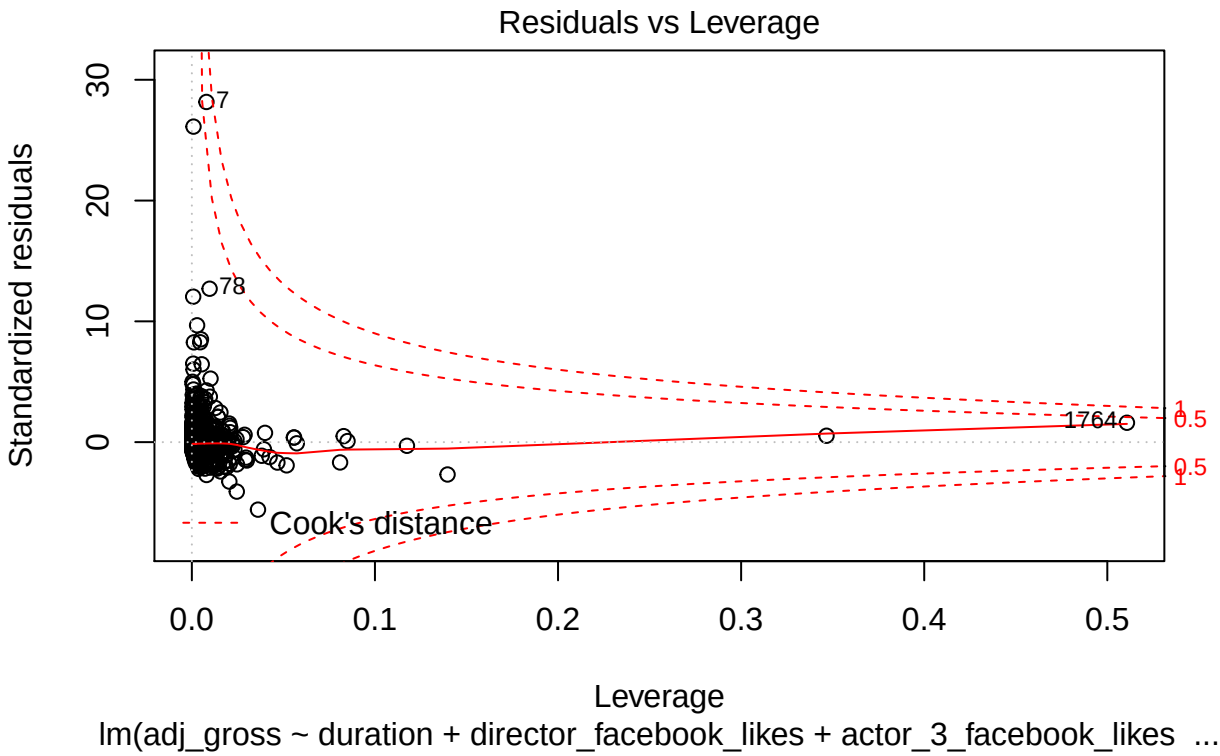
```
gross_p <- predict(m1_back, newdata = movies_new, type = "response")

plot(m1_back)
```

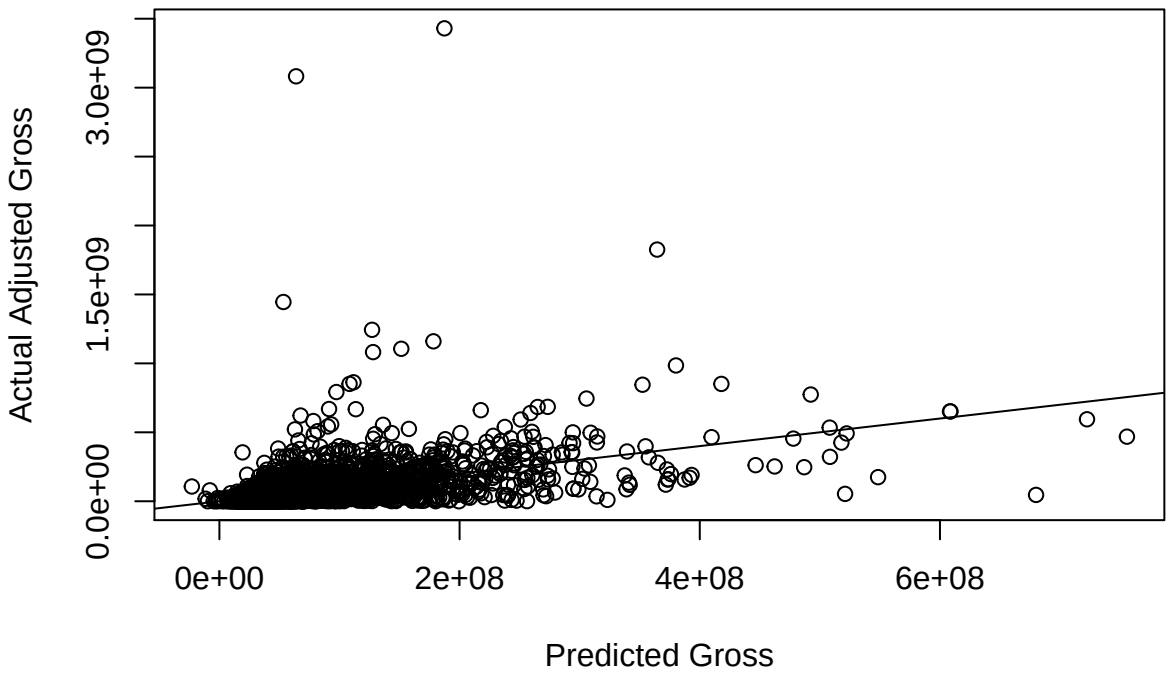

Residuals vs Fitted
Residuals
3e+09
1e+09
−1e+09
0e+00
2e+08
4e+08
6e+08
Fitted values
lm(adj_gross ~ duration + director_facebook_likes + actor_3_facebook_likes ...
6
7
78


Normal Q−Q
Standardized residuals
30
20
10
0
7
6
78
−2
0
2
Theoretical Quantiles
lm(adj_gross ~ duration + director_facebook_likes + actor_3_facebook_likes ...


Scale−Location
√|Standardized residuals|
0
1
2
3
4
5
6
7
78
0e+00
2e+08
4e+08
6e+08
Fitted values
lm(adj_gross ~ duration + director_facebook_likes + actor_3_facebook_likes ...




```
plot(x = gross_p, y = movies_new$adj_gross, xlab = "Predicted Gross", ylab = "Actual Adjusted Gross")
abline(a=0,b=1)
```




```
profit_margin_p <- (gross_p - movies_new$adj_budget) / gross_p

movies_p <- data.frame(movies$movie_title, movies_new$adj_budget, movies_new$adj_gross, gross_p, movies_

colnames(movies_p) <- c("Movie Title", "Actual Adjusted Budget", "Actualy Adjusted Gross", "Predicted Gr

head(movies_p)
```

```
##                       Movie Title Actual Adjusted Budget
## 1                       Metropolis                82758621
## 2             The Broadway Melody                 5288372
## 3                     42nd Street                 8167442
## 4                         Top Hat                10668613
## 5                    Modern Times                25899281
## 6 Snow White and the Seven Dwarfs                33333333
##   Actualy Adjusted Gross Predicted Gross Actualy Profit Margin
## 1               364620.7        92279310          -225.9718177
## 2             39181395.3        12244900             0.8650285
## 3             42790697.7        21372711             0.8091304
## 4             52554744.5        16112839             0.7970000
## 5              2818618.7        69436696            -8.1886428
## 6           3082091416.7        63867341             0.9891848
##   Predicted Profit Margin
## 1               0.1031725
## 2               0.5681163
## 3               0.6178565
```

```
## 4               0.3378812
## 5               0.6270087
## 6               0.4780848
```

Smooth Operators - All Done!