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

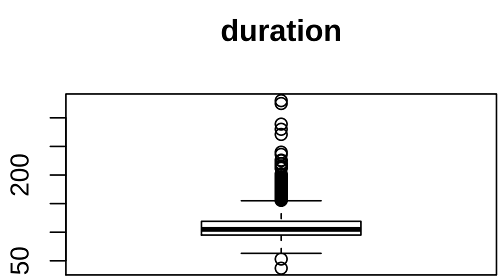
```
##     duration     director_facebook_likes actor_3_facebook_likes
```

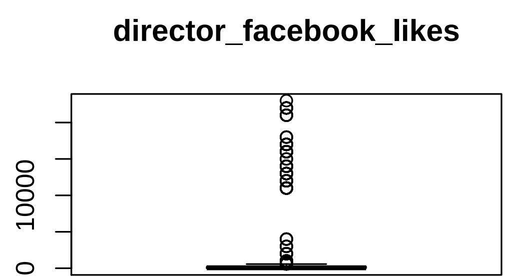
```
##  Min.   : 37.0   Min.   :    0.0         Min.   :    0.0
##  1st Qu.: 95.0   1st Qu.:   11.0         1st Qu.:  233.0
##  Median :105.0   Median :   62.0         Median :  467.0
##  Mean   :109.5   Mean   :  911.3         Mean   :  836.2
##  3rd Qu.:119.0   3rd Qu.:  235.0         3rd Qu.:  723.0
##  Max.   :330.0   Max.   :23000.0         Max.   :23000.0
##
##  actor_1_facebook_likes     gross           movie_title
##  Min.   :     0.0       Min.   :      703   Length:3042
##  1st Qu.:   811.2       1st Qu.: 11787482   Class :character
##  Median :  2000.0       Median : 34264376   Mode  :character
##  Mean   :  8241.5       Mean   : 57651658
##  3rd Qu.: 13000.0       3rd Qu.: 75074326
##  Max.   :640000.0       Max.   :760505847
##
##  num_voted_users   cast_total_facebook_likes facenumber_in_poster
##  Min.   :     22   Min.   :     0            Min.   : 0.000
##  1st Qu.:  19117   1st Qu.:  2210            1st Qu.: 0.000
##  Median :  54463   Median :  4517            Median : 1.000
##  Mean   : 108285   Mean   : 12340            Mean   : 1.419
##  3rd Qu.: 132124   3rd Qu.: 16904            3rd Qu.: 2.000
##  Max.   :1689764   Max.   :656730            Max.   :43.000
##
##    content_rating     budget            title_year
##  R         :1333   Min.   :      218   Min.   :1929
##  PG-13     :1110   1st Qu.: 10725000   1st Qu.:1999
##  PG        : 472   Median : 25000000   Median :2004
##  G         :  70   Mean   : 40319361   Mean   :2003
##  Not Rated:  18   3rd Qu.: 55000000   3rd Qu.:2010
##  Unrated  :  13   Max.   :300000000   Max.   :2016
##  (Other)  :  26
##  actor_2_facebook_likes   imdb_score
##  Min.   :     0.0       Min.   :1.600
##  1st Qu.:   436.0       1st Qu.:5.800
##  Median :   729.5       Median :6.500
##  Mean   :  2180.3       Mean   :6.383
##  3rd Qu.:  1000.0       3rd Qu.:7.100
##  Max.   :137000.0       Max.   :9.300
##
```

	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes
duration	1.0000000	0.2104197	0.1448777	0.0912903
director_facebook_likes	0.2104197	1.0000000	0.1219467	0.0868426
actor_3_facebook_likes	0.1448777	0.1219467	1.0000000	0.2483043
actor_1_facebook_likes	0.0912903	0.0868426	0.2483043	1.0000000
num_voted_users	0.3705768	0.3190331	0.2818195	0.1741973
cast_total_facebook_likes	0.1349956	0.1172865	0.4830033	0.9459350
facenumber_in_poster	0.0065845	-0.0523321	0.1042739	0.0538466
budget	0.2988689	0.0942904	0.2747815	0.1551897
title_year	-0.1086958	-0.0580504	0.1277213	0.0914452
actor_2_facebook_likes	0.1504159	0.1192872	0.5521997	0.3798140
imdb_score	0.3819342	0.2225461	0.0882029	0.1178984

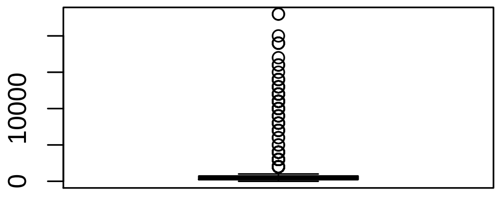
	num_voted_users	cast_total_facebook_likes	facenumber_in_poster	budget
duration	0.3705768	0.1349956	0.0065845	0.2988689
director_facebook_likes	0.3190331	0.1172865	-0.0523321	0.0942904
actor_3_facebook_likes	0.2818195	0.4830033	0.1042739	0.2747815
actor_1_facebook_likes	0.1741973	0.9459350	0.0538466	0.1551897
num_voted_users	1.0000000	0.2486828	-0.0441983	0.4054595
cast_total_facebook_likes	0.2486828	1.0000000	0.0750811	0.2362870
facenumber_in_poster	-0.0441983	0.0750811	1.0000000	-0.0267742
budget	0.4054595	0.2362870	-0.0267742	1.0000000
title_year	0.0241674	0.1256809	0.0873375	0.2412454
actor_2_facebook_likes	0.2524944	0.6319688	0.0625703	0.2526741
imdb_score	0.5089320	0.1377072	-0.0694804	0.0713682

	title_year	actor_2_facebook_likes
duration	-0.1086958	0.1504159
director_facebook_likes	-0.0580504	0.1192872
actor_3_facebook_likes	0.1277213	0.5521997
actor_1_facebook_likes	0.0914452	0.3798140
num_voted_users	0.0241674	0.2524944
cast_total_facebook_likes	0.1256809	0.6319688
facenumber_in_poster	0.0873375	0.0625703
budget	0.2412454	0.2526741
title_year	1.0000000	0.1253783
actor_2_facebook_likes	0.1253783	1.0000000
imdb_score	-0.1504498	0.1274387

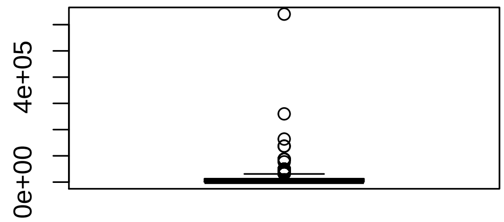




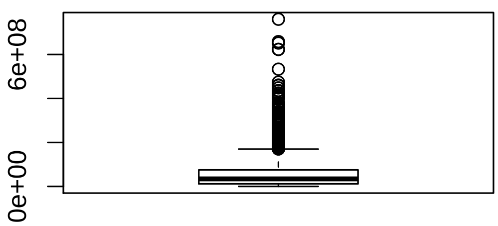
actor_3_facebook_likes



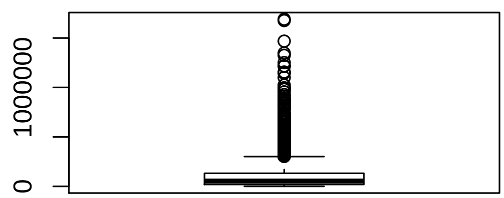
actor_1_facebook_likes



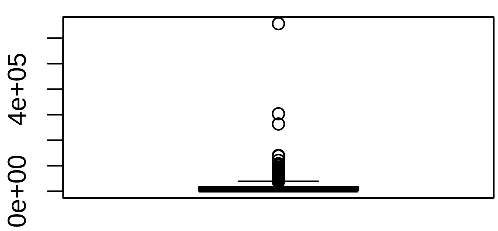
gross



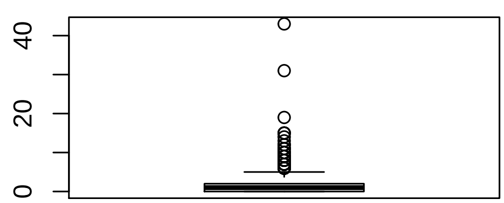
num_voted_users

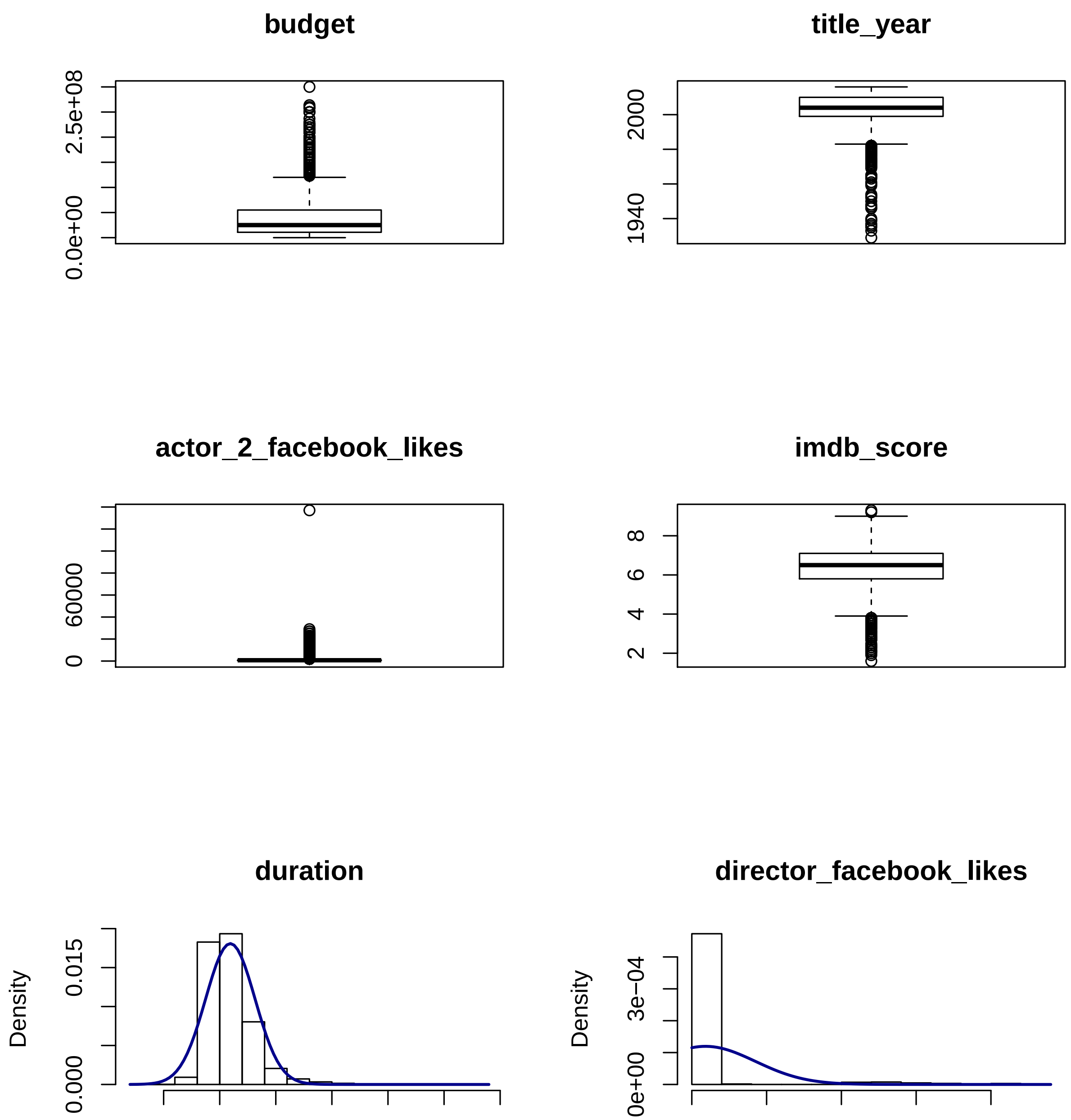


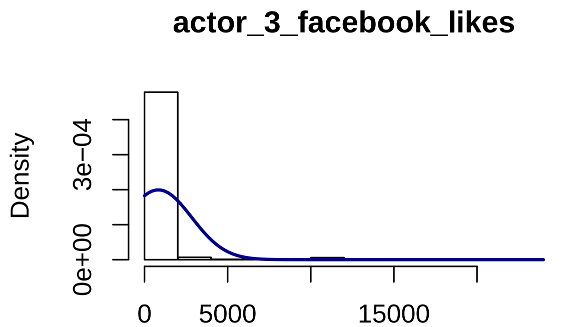
cast_total_facebook_likes

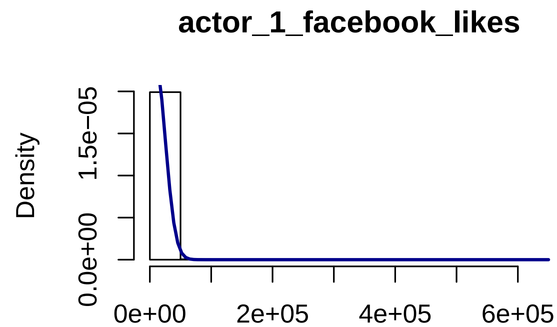


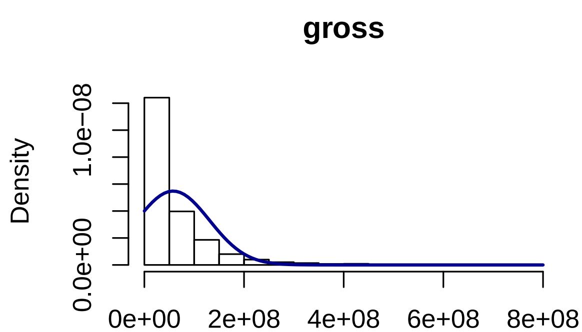
facenumber_in_poster

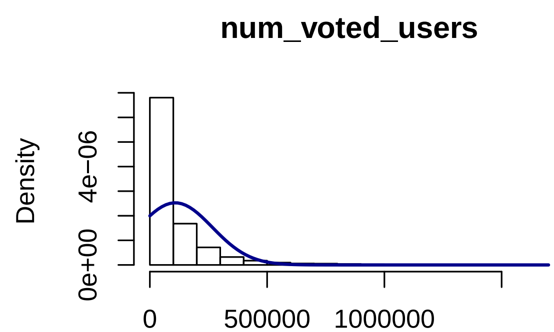


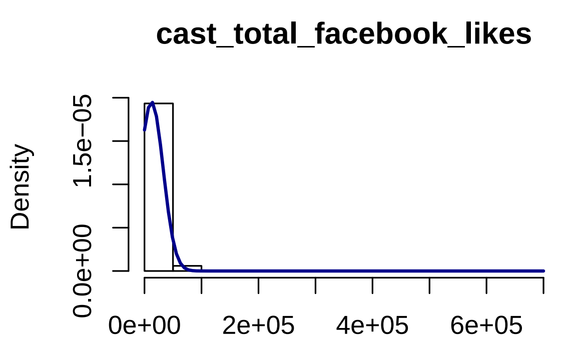

budget
0.0e+00
2.5e+08
title_year
1940
2000
actor_2_facebook_likes
0
60000
imdb_score
2
4
6
8
duration
Density
0.000
0.015
50
100
200
300
director_facebook_likes
Density
0e+00
3e−04
0
5000
15000

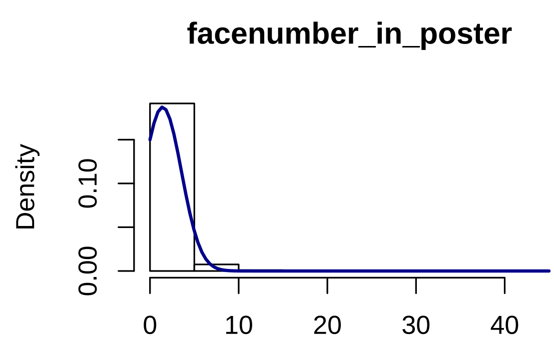

actor_3_facebook_likes
Density
0e+00
3e−04
0
5000
15000

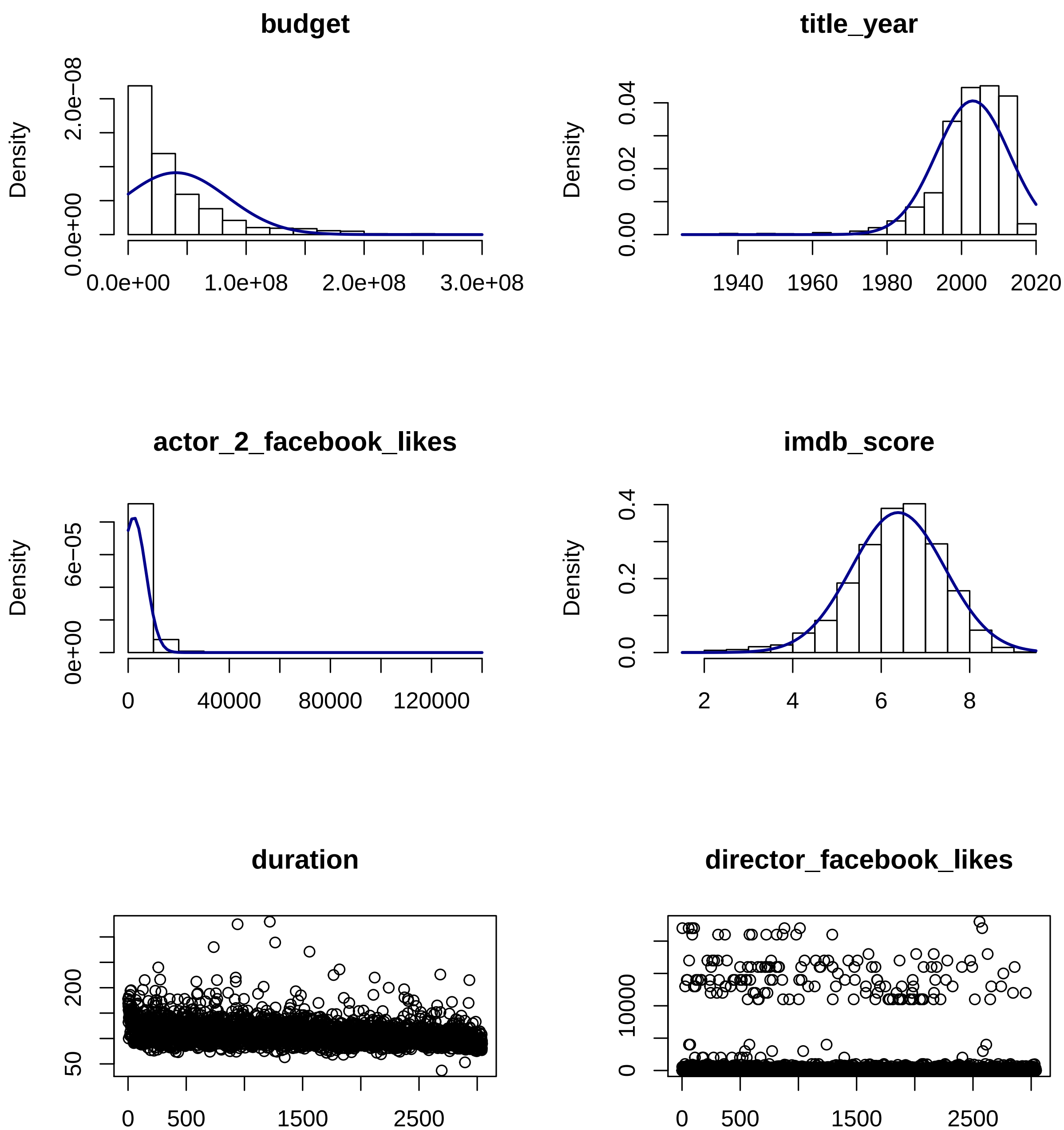

actor_1_facebook_likes
Density
0.0e+00
1.5e−05
0e+00
2e+05
4e+05
6e+05

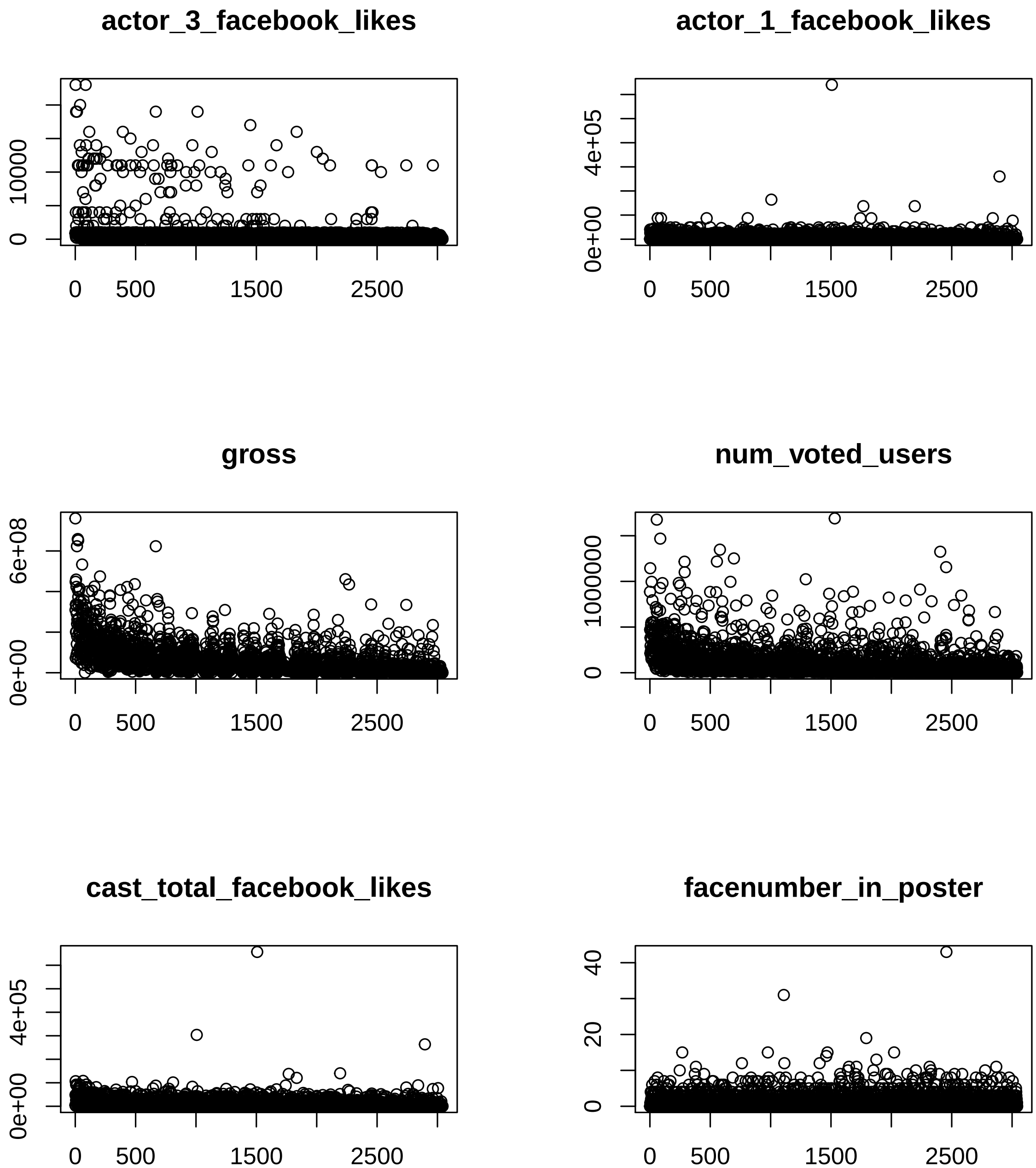

gross
Density
0.0e+00
1.0e−08
0e+00
2e+08
4e+08
6e+08
8e+08

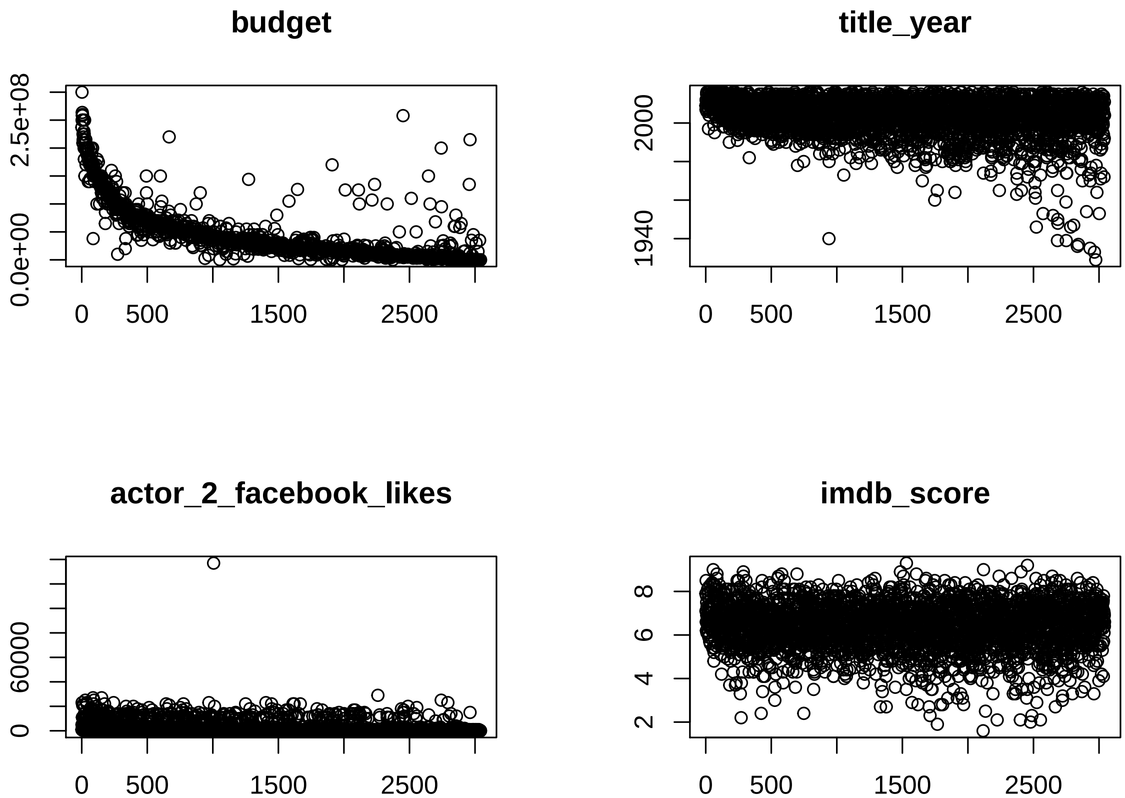

num_voted_users
Density
0e+00
4e−06
0
500000
1000000


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
0.0e+00
2.0e−08
0.0e+00
1.0e+08
2.0e+08
3.0e+08
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
500
1500
2500
director_facebook_likes
0
10000
0
500
1500
2500


actor_3_facebook_likes
0
10000
0
500
1500
2500
actor_1_facebook_likes
0e+00
4e+05
0
500
1500
2500
gross
0e+00
6e+08
0
500
1500
2500
num_voted_users
0
1000000
0
500
1500
2500
cast_total_facebook_likes
0e+00
4e+05
0
500
1500
2500
facenumber_in_poster
0
20
40
0
500
1500
2500



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
## 2502                      1                         1
##  520                      1                         1
##   10                      1                         1
##    1                      1                         1
##    6                      1                         1
##    2                      1                         1
##    1                      0                         0
##                           1                         1
##      actor_2_facebook_likes actor_3_facebook_likes director_facebook_likes
## 2502                      1                      1                       1
##  520                      1                      1                       0
##   10                      1                      0                       1
##    1                      1                      0                       0
##    6                      0                      0                       1
##    2                      0                      0                       0
##    1                      0                      0                       1
##                           9                     20                     523
```

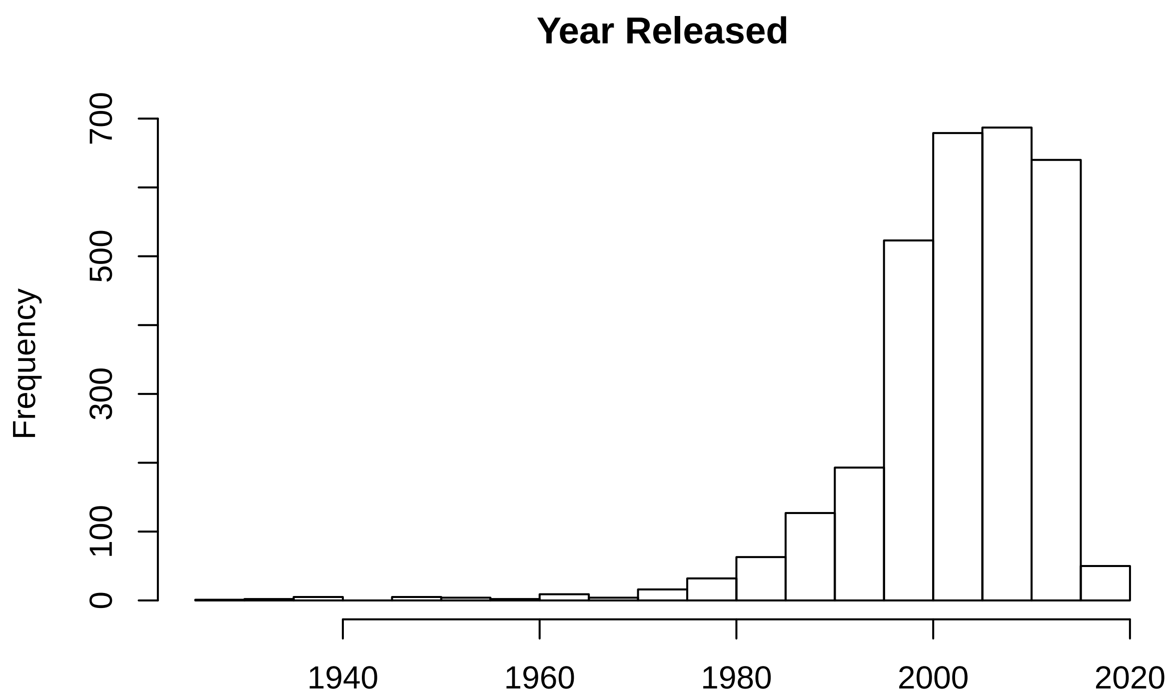
```
##
## 2502   0
##  520   1
##   10   1
##    1   2
##    6   2
##    2   3
##    1   4
##      554
```

```
##     duration     director_facebook_likes actor_3_facebook_likes
##  Min.   : 37.0   Min.   :    2           Min.   :    2.0
##  1st Qu.: 95.0   1st Qu.:   31           1st Qu.:  233.0
##  Median :105.0   Median :   96           Median :  467.0
##  Mean   :109.5   Mean   : 1133           Mean   :  836.4
##  3rd Qu.:119.0   3rd Qu.:  295           3rd Qu.:  723.0
##  Max.   :330.0   Max.   :23000           Max.   :23000.0
##
##  actor_1_facebook_likes     gross           movie_title
##  Min.   :      2.0       Min.   :      703   Length:3042
##  1st Qu.:    811.2       1st Qu.: 11787482   Class :character
##  Median :   2000.0       Median : 34264376   Mode  :character
##  Mean   :   8241.7       Mean   : 57651658
##  3rd Qu.:  13000.0       3rd Qu.: 75074326
##  Max.   :640000.0        Max.   :760505847
##
##  num_voted_users   cast_total_facebook_likes facenumber_in_poster
##  Min.   :     22   Min.   :     2            Min.   : 0.000
##  1st Qu.:  19117   1st Qu.:  2210            1st Qu.: 0.000
##  Median :  54463   Median :  4517            Median : 1.000
##  Mean   : 108285   Mean   : 12340            Mean   : 1.419
##  3rd Qu.: 132124   3rd Qu.: 16904            3rd Qu.: 2.000
##  Max.   :1689764   Max.   :656730            Max.   :43.000
##
##    content_rating     budget             title_year
##  R        :1333   Min.   :      218   Min.   :1929
##  PG-13    :1110   1st Qu.: 10725000   1st Qu.:1999
##  PG       : 472   Median : 25000000   Median :2004
##  G        :  70   Mean   : 40319361   Mean   :2003
##  Not Rated:  18   3rd Qu.: 55000000   3rd Qu.:2010
##  Unrated  :  13   Max.   :300000000   Max.   :2016
##  (Other)  :  26
##  actor_2_facebook_likes   imdb_score
##  Min.   :      2.0       Min.   :1.600
##  1st Qu.:    436.0       1st Qu.:5.800
##  Median :    729.5       Median :6.500
##  Mean   :   2180.4       Mean   :6.383
##  3rd Qu.:   1000.0       3rd Qu.:7.100
##  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:



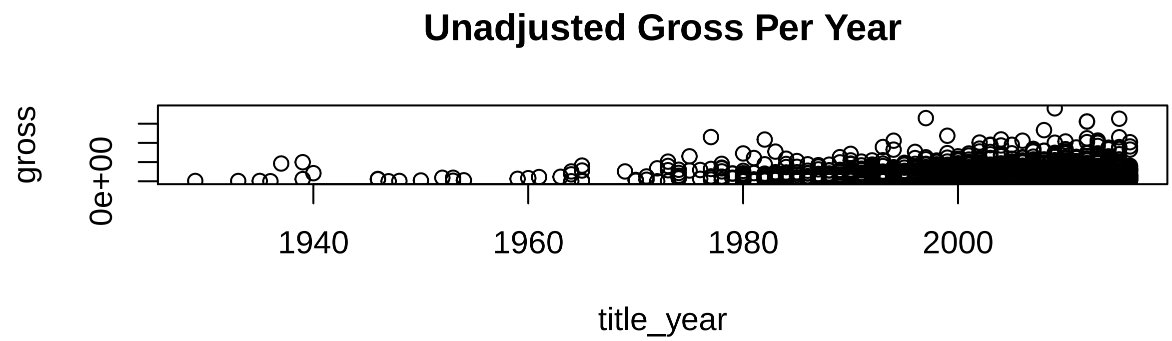
As you can see, the vast majority came from 1990s and above, but we can't discredit the movies from previous year. In order to accurately portray elements from the past, we have instituted a rate of inflation calculation. Using the consumer price index (for our part here we are making a crucial assumption, 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.

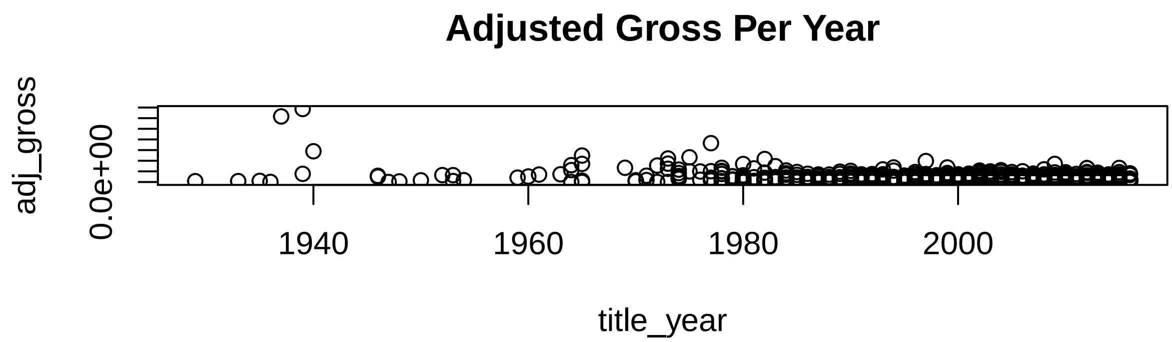
```
movies <- merge(x = movies, y = cpi, by = "title_year")
movies$adj_gross <- with(movies, (240/cpi * gross))
movies$adj_budget <- with(movies, (240/cpi * budget))
movies$adj_margin <- with(movies, adj_gross-adj_budget)
```

```
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")
```





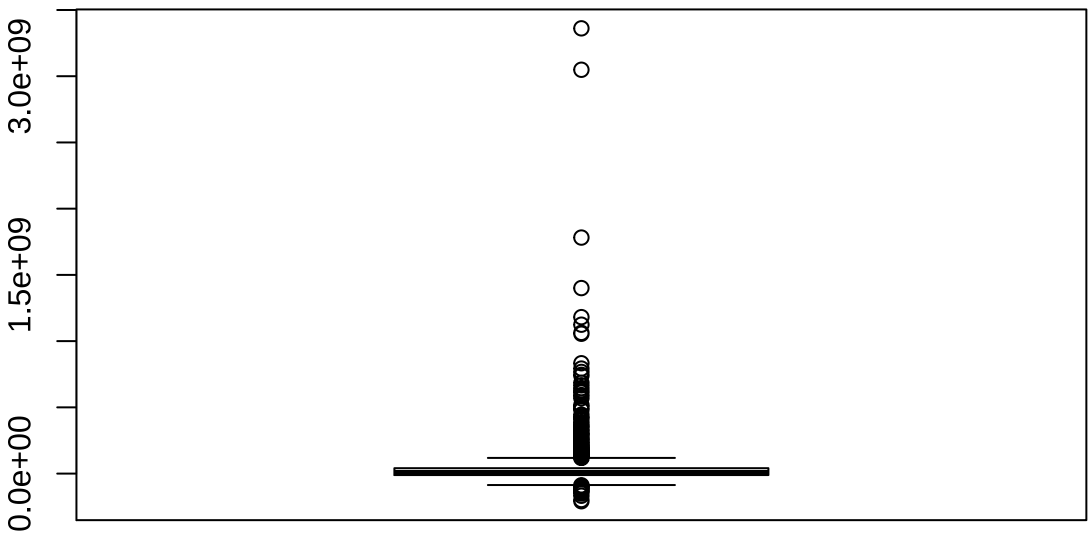
From the above graphs, we can see that the adjustment for the gross did indeed create a more uniformed dataset (where as before we saw movies increasing over the years). As a point of interest, the movies that made over a billion dollars are shown below:

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

```
##                            movie_title     gross  adj_gross
## 5     Snow White and the Seven Dwarfs 184925485 3082091417
## 7                  Gone with the Wind 198655278 3430019188
## 8                           Pinocchio  84300000 1445142857
## 26                The Sound of Music 163214286 1243537417
## 39                       The Exorcist 204565000 1105756757
## 48                               Jaws 260000000 1159851301
## 53 Star Wars: Episode IV - A New Hope 460935665 1825487782
## 90          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
## -4.5230  -1.1134   0.5148   1.0651   1.8657
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 7.975e+01  1.065e+01   7.488 6.99e-14 ***
## title_year                 -3.934e-02  5.298e-03  -7.425 1.13e-13 ***
## duration                   -1.330e-02  2.471e-03  -5.384 7.28e-08 ***
```

```
## director_facebook_likes   -2.695e-05  1.393e-05  -1.936   0.0529 .
## actor_3_facebook_likes    -1.209e-04  7.464e-05  -1.620   0.1052
## actor_1_facebook_likes    -1.180e-04  5.019e-05  -2.351   0.0187 *
## num_voted_users            8.596e-06  6.465e-07  13.297  < 2e-16 ***
## cast_total_facebook_likes  1.138e-04  5.015e-05   2.270   0.0232 *
## facenumber_in_poster       4.059e-02  2.221e-02   1.827   0.0676 .
## actor_2_facebook_likes    -1.192e-04  5.245e-05  -2.272   0.0231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3307.1  on 2432  degrees of freedom
## Residual deviance: 2959.2  on 2423  degrees of freedom
## AIC: 2979.2
##
## Number of Fisher Scoring iterations: 5
```

```r
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] 0.7447325
```

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.

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

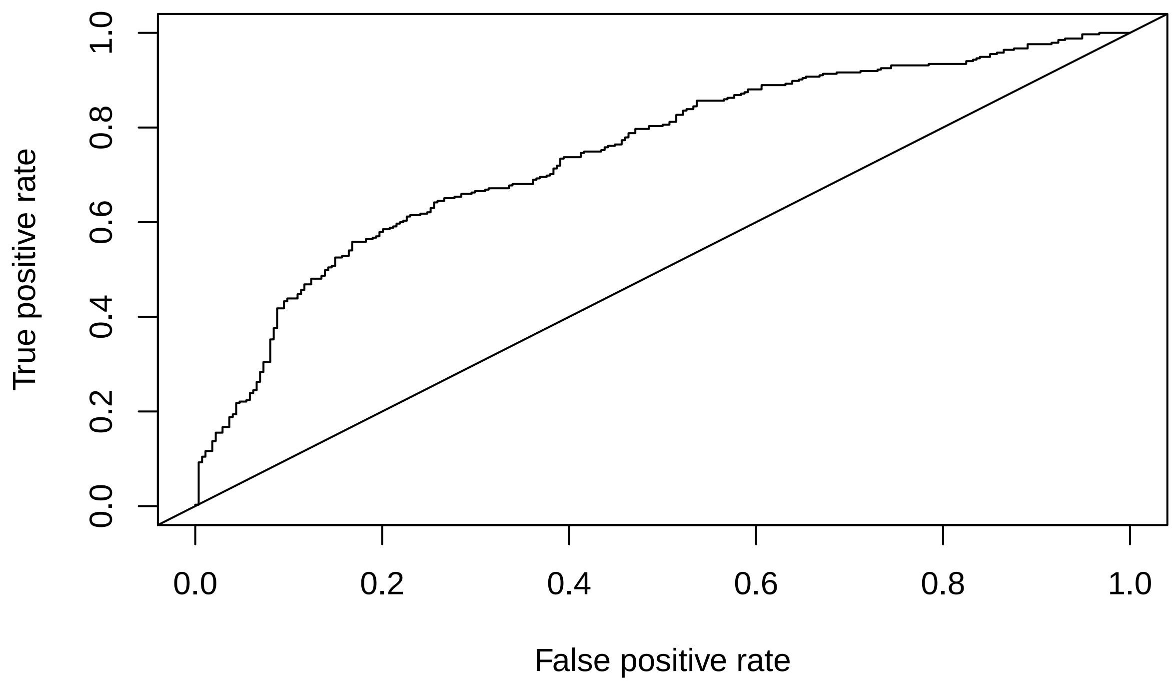
```
## Start:  AIC=2979.19
## money ~ title_year + duration + director_facebook_likes + actor_3_facebook_likes +
##     actor_1_facebook_likes + num_voted_users + cast_total_facebook_likes +
##     facenumber_in_poster + actor_2_facebook_likes
##
##                             Df Deviance    AIC
## <none>                           2959.2 2979.2
## - actor_3_facebook_likes     1   2961.8 2979.8
## - facenumber_in_poster       1   2962.6 2980.6
## - director_facebook_likes    1   2962.9 2980.9
## - cast_total_facebook_likes  1   2964.6 2982.6
## - actor_2_facebook_likes     1   2964.6 2982.6
## - actor_1_facebook_likes     1   2965.0 2983.0
## - duration                   1   2989.3 3007.3
## - title_year                 1   3021.2 3039.2
## - num_voted_users            1   3242.6 3260.6
```

```r
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 + facenumber_in_poster + actor_2_facebook_likes,
##     family = binomial(link = "logit"), data = train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5230  -1.1134   0.5148   1.0651   1.8657
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                7.975e+01  1.065e+01   7.488 6.99e-14 ***
## title_year                -3.934e-02  5.298e-03  -7.425 1.13e-13 ***
## duration                  -1.330e-02  2.471e-03  -5.384 7.28e-08 ***
## director_facebook_likes   -2.695e-05  1.393e-05  -1.936   0.0529 .
## actor_3_facebook_likes    -1.209e-04  7.464e-05  -1.620   0.1052
## actor_1_facebook_likes    -1.180e-04  5.019e-05  -2.351   0.0187 *
## num_voted_users            8.596e-06  6.465e-07  13.297  < 2e-16 ***
## cast_total_facebook_likes  1.138e-04  5.015e-05   2.270   0.0232 *
## facenumber_in_poster       4.059e-02  2.221e-02   1.827   0.0676 .
## actor_2_facebook_likes    -1.192e-04  5.245e-05  -2.272   0.0231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3307.1  on 2432  degrees of freedom
## Residual deviance: 2959.2  on 2423  degrees of freedom
## AIC: 2979.2
##
## Number of Fisher Scoring iterations: 5
```

```r
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] 0.7447325
```

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
## -543916599  -34116145  -14970355   11323207 3253842604
##
## Coefficients:
```

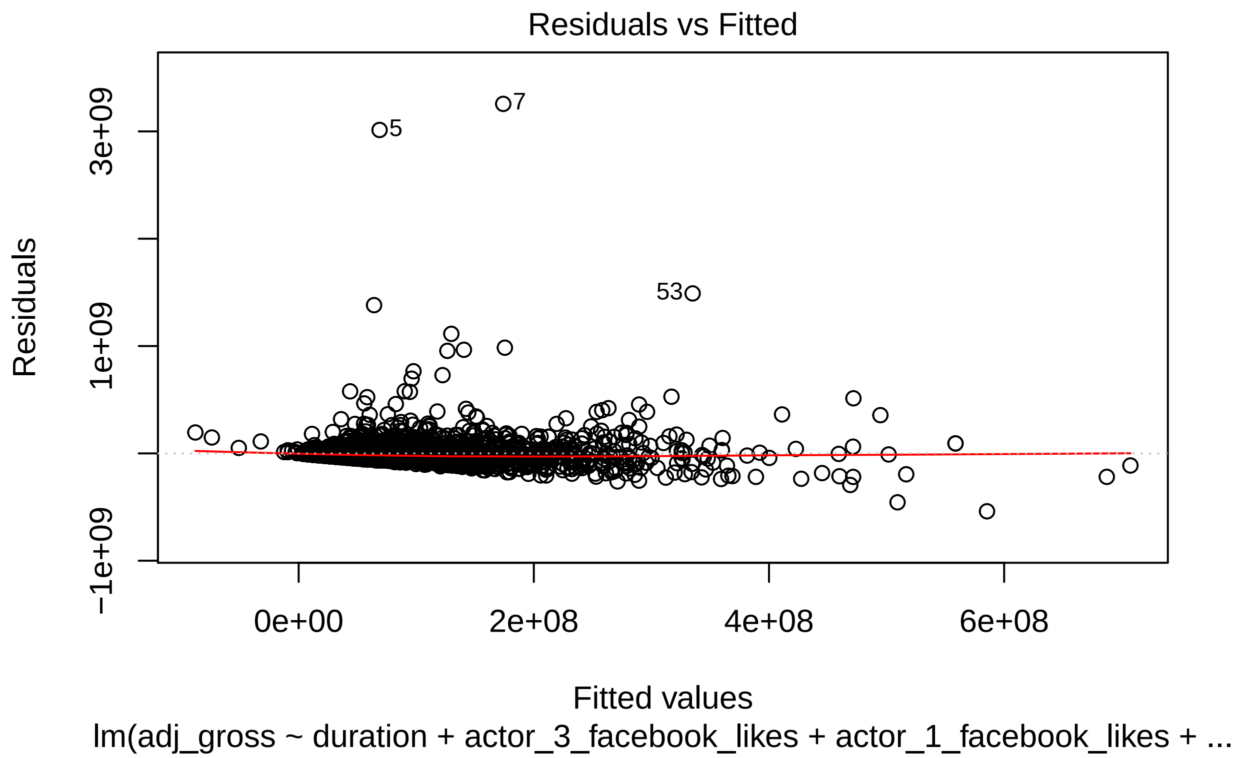
```
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -2.495e+07  1.177e+07  -2.121 0.034042 *
## duration                   3.912e+05  1.127e+05   3.472 0.000523 ***
## director_facebook_likes   -3.780e+02  6.444e+02  -0.586 0.557601
## actor_3_facebook_likes    -9.492e+03  3.174e+03  -2.990 0.002811 **
## actor_1_facebook_likes    -6.898e+03  1.929e+03  -3.577 0.000353 ***
## num_voted_users            3.140e+02  1.685e+01  18.632  < 2e-16 ***
## cast_total_facebook_likes  6.610e+03  1.925e+03   3.433 0.000605 ***
## facenumber_in_poster      -1.100e+06  1.052e+06  -1.046 0.295859
## actor_2_facebook_likes    -7.269e+03  2.039e+03  -3.565 0.000369 ***
## adj_budget                 6.197e-01  5.056e-02  12.256  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 122100000 on 3032 degrees of freedom
## Multiple R-squared:  0.2619, Adjusted R-squared:  0.2597
## F-statistic: 119.5 on 9 and 3032 DF,  p-value: < 2.2e-16
```

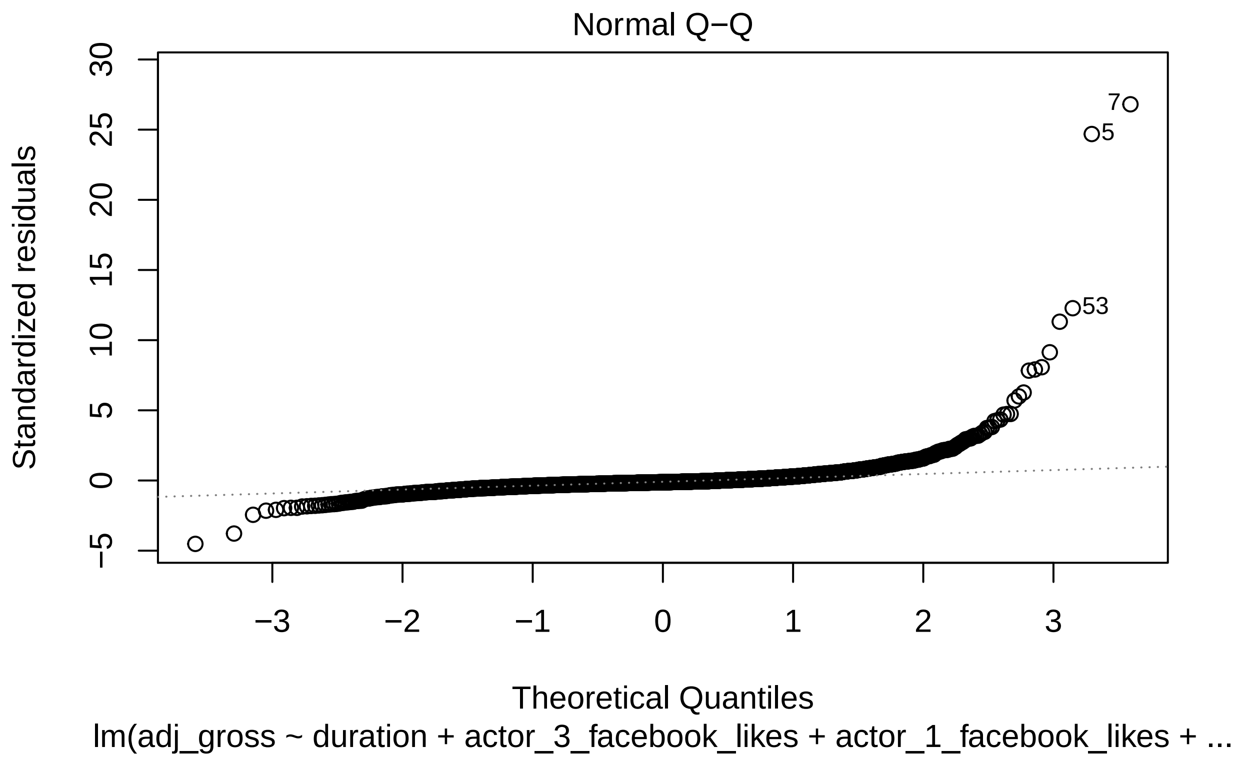
```r
m1_back <- step(m1, trace = 0)
summary(m1_back)
```

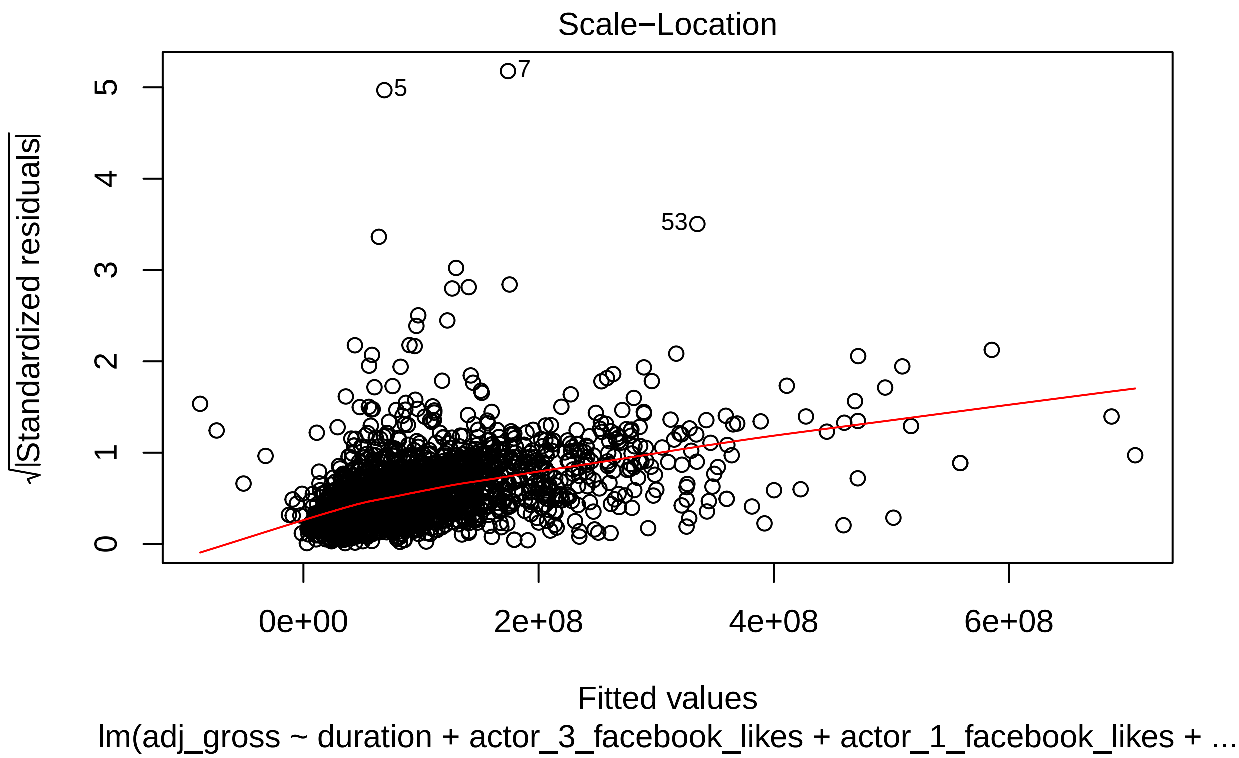
```
##
## Call:
## lm(formula = adj_gross ~ duration + actor_3_facebook_likes +
##     actor_1_facebook_likes + num_voted_users + cast_total_facebook_likes +
##     actor_2_facebook_likes + adj_budget, data = movies_new)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -539507906  -34108279  -15248268   11621440 3256031799
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -2.541e+07  1.164e+07  -2.184 0.029066 *
## duration                   3.784e+05  1.115e+05   3.393 0.000699 ***
## actor_3_facebook_likes    -9.658e+03  3.169e+03  -3.048 0.002324 **
## actor_1_facebook_likes    -6.906e+03  1.926e+03  -3.586 0.000341 ***
## num_voted_users            3.127e+02  1.633e+01  19.154  < 2e-16 ***
## cast_total_facebook_likes  6.612e+03  1.923e+03   3.438 0.000593 ***
## actor_2_facebook_likes    -7.293e+03  2.035e+03  -3.584 0.000344 ***
## adj_budget                 6.249e-01  5.034e-02  12.414  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 122100000 on 3034 degrees of freedom
## Multiple R-squared:  0.2616, Adjusted R-squared:  0.2598
## F-statistic: 153.5 on 7 and 3034 DF,  p-value: < 2.2e-16
```

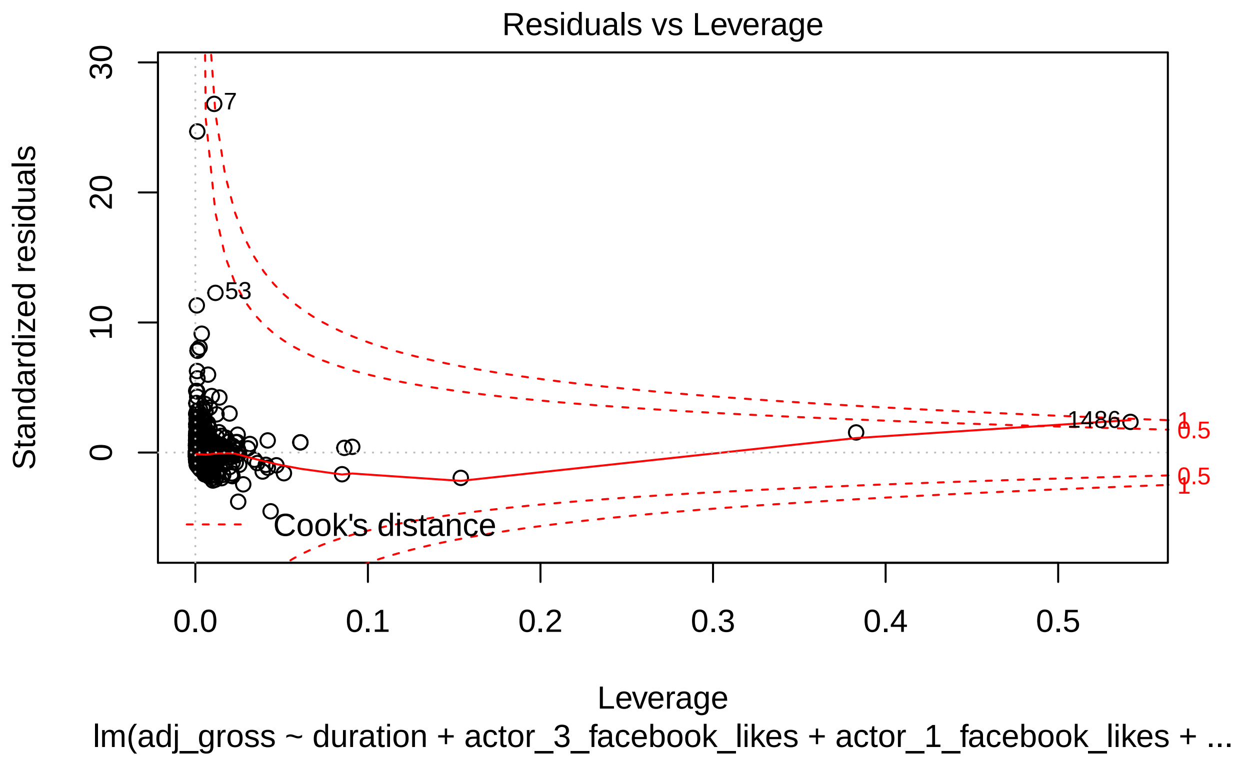
```r
gross_p <- predict(m1_back, newdata = movies_new, type = "response")

plot(m1_back)
```

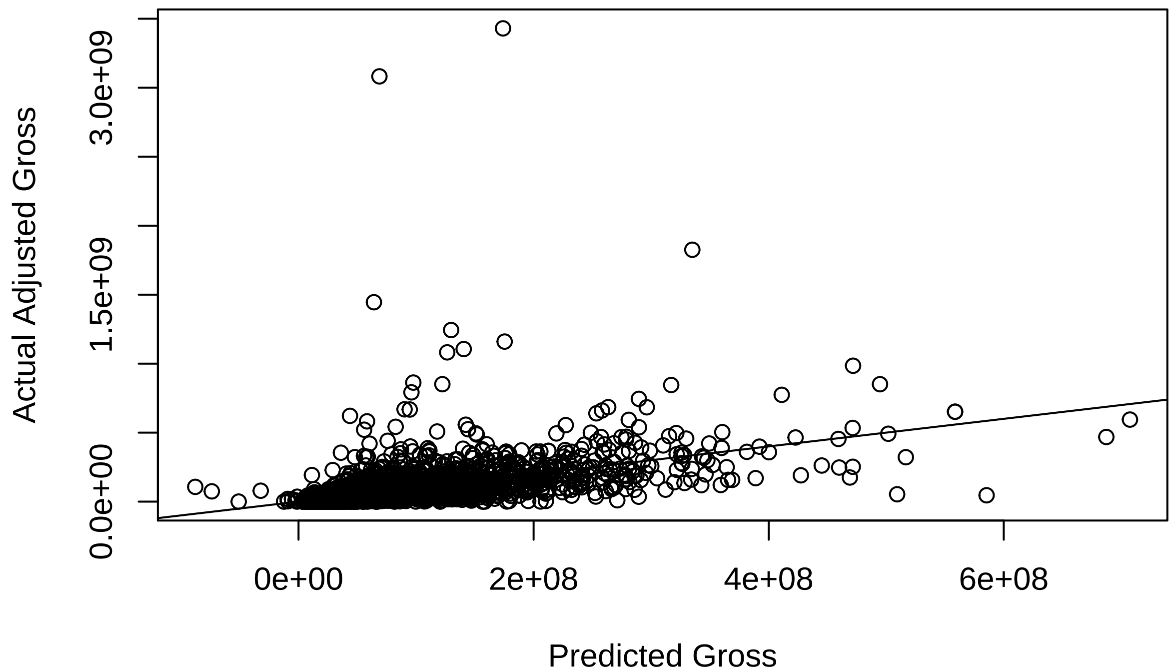

Residuals vs Fitted
Residuals
3e+09
1e+09
−1e+09
0e+00
2e+08
4e+08
6e+08
Fitted values
lm(adj_gross ~ duration + actor_3_facebook_likes + actor_1_facebook_likes + ...
5
7
53


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(adj_gross ~ duration + actor_3_facebook_likes + actor_1_facebook_likes + ...
−5
0
5
10
15
20
25
30
−3
−2
−1
0
1
2
3
7
5
53


Scale−Location
√|Standardized residuals|
0
1
2
3
4
5
7
5
53
0e+00
2e+08
4e+08
6e+08
Fitted values
lm(adj_gross ~ duration + actor_3_facebook_likes + actor_1_facebook_likes + ...




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




```r
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 G

head(movies_p)
```

```
##                        Movie Title Actual Adjusted Budget
## 1             The Broadway Melody                5288372
## 2                     42nd Street                8167442
## 3                         Top Hat               10668613
## 4                   Modern Times               25899281
## 5 Snow White and the Seven Dwarfs               33333333
## 6                The Wizard of Oz               48345324
##   Actualy Adjusted Gross Predicted Gross Actualy Profit Margin
## 1               39181395        17094968             0.8650285
## 2               42790698        17006814             0.8091304
## 3               52554745        15808622             0.7970000
## 4                2818619        68491707            -8.1886428
## 5             3082091417        68826380             0.9891848
## 6              383354452       139935227             0.8738887
##   Predicted Profit Margin
## 1               0.6906474
## 2               0.5197547
## 3               0.3251396
```

```
## 4                0.6218625
## 5                0.5156896
## 6                0.6545164
```

Smooth Operators - All Done!