Modeling and prediction for movies

Load packages

```r
options(warn=-1) # turn off warnings
library(ggplot2)
library(dplyr)
library(statsr)
library(tidyr)
library(knitr)
library(colorspace)
library(GGally)
library(ggpubr)
library(cowplot)
library(gender)
library(leaps)
```

Load data

```r
load("movies.Rdata")
```

Part 1: Data

The data set is comprised of 651 randomly sampled reviews of movies produced and released before 2016, and was obtained via the IMDB and Rotten Tomatoes APIs. The population thus corresponds to all ratings of movies on Rotten Tomatoes and IMDB that were produced and released before 2016.

The data collection methodology of random sampling is designed to represent the population, so the sample is generalizable to this population.

Because we are not actively collecting data for the study, our following analysis is strictly observational. We can not infer any causality among the variables in our analysis.

Finally, note that 651 movies is well under 10% of all movies on these websites, so we can assume that the reviews are independent.

Part 2: Research question

Question: Which features are most important in predicting the popularity of a movie?

Say we want to produce a movie from scratch. Which salient properties of the movie and its release features are most statistically important to predict how popular, as measured by viewer ratings, the movie will be?

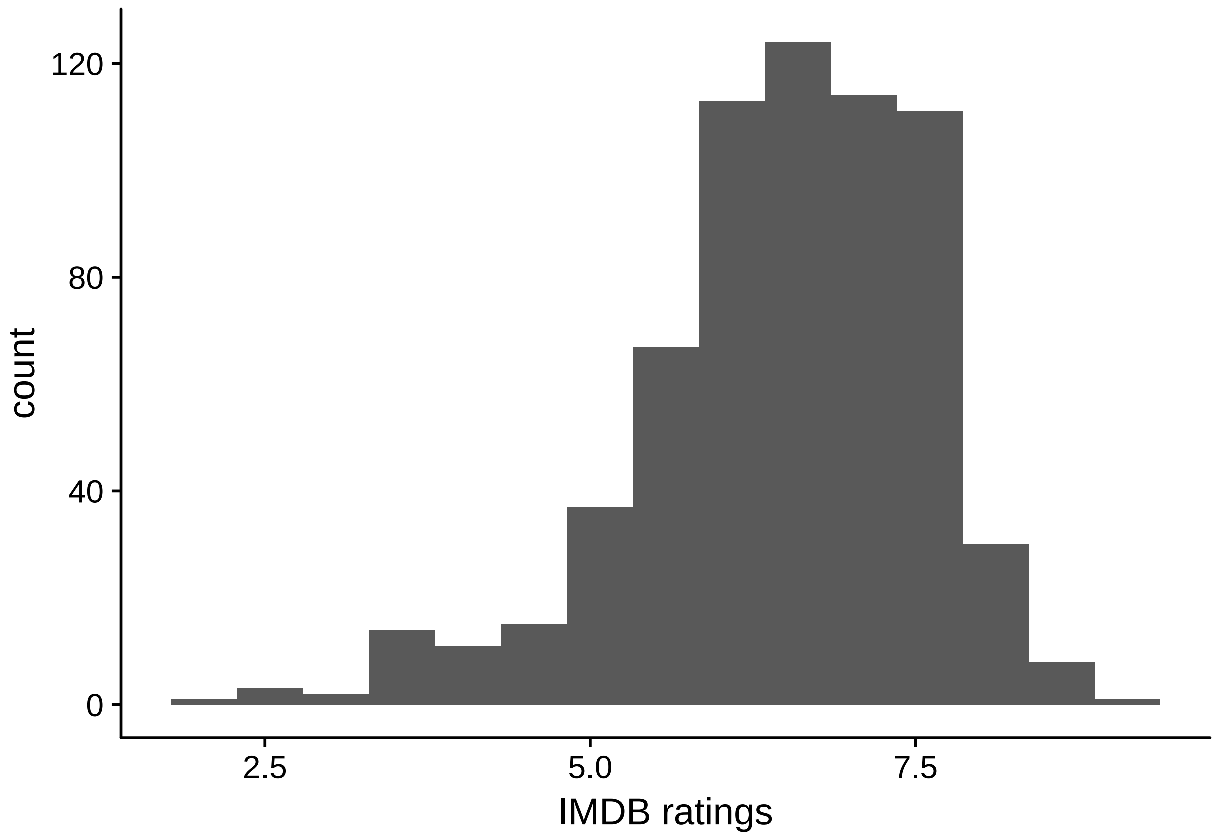
To address this question, we will build a multiple linear regression model to predict the dependent variable `imdb_rating`, which is the numerical user-rating score from the IMDB website.

In the spirit of making a predictive model, we will omit variables that "leak data" from our predictive analysis. So, we will only keep variables that we could know from the outset of producing a movie and planning its release. For example, we will drop the variable corresponding to Rotten Tomatoes user ratings, `audience_score`, and other similar variables.

Part 3: Exploratory data analysis

Below is a histgram and summary of the dependent `imdb_rating` variable, for which we will build a predictive multiple regression model. One can see that it is approximately normally distributed, with a slight left skew, a mean of 6.5, and a median of 6.6.

```
ggplot(aes(imdb_rating), data = movies) + geom_histogram(bins=15) + xlab("IMDB ratings")
```



```
summary(movies$imdb_rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.900   5.900   6.600   6.493   7.300   9.000
```

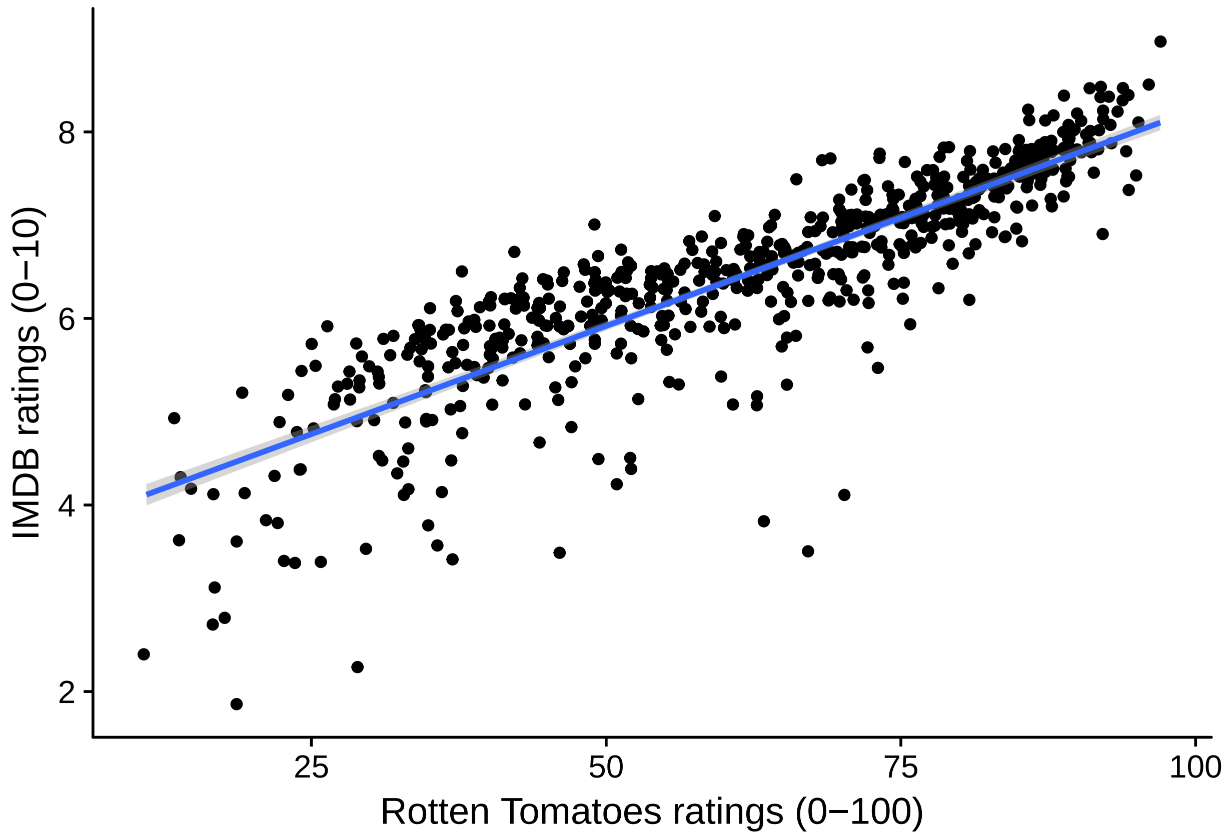
Eliminating data leakage

We identify all variables that are sources of "data leakage," meaning that there would be no way of collecting data for these variables before the movie is produced and released. Remember, we wish to create a predictive model that is relevant for someone who is creating a movie.

The chosen dependent variable for popularity is `imdb_rating` from the IMDB website, but the data frame also includes ratings data from the Rotten Tomatoes website. Surely, we expect these variables to be highly correlated, and the Rotten Tomatoes ratings data could not exist before a movie is produced. Below, we show a scatter plot of the IMDB ratings as a function of the Rotten Tomatoes ratings, and calculate their Pearson correlation coefficient.

The scatter plot shows a strong nearly-linear relationship, which is supported by a correlation of 0.8649 and a corresponding R^2 of 0.7481. Certainly, the Rotten Tomatoes ratings would serve as an excellent predictor for the IMDB ratings, but they must be eliminated for the problem at hand.

```
ggplot(movies, aes(audience_score, imdb_rating)) + geom_jitter() + xlab("Rotten Tomatoes ratings (0-100)
```



```
movies %>% summarise(cor(imdb_rating, audience_score))
```

```
## # A tibble: 1 x 1
##   `cor(imdb_rating, audience_score)`
##                                <dbl>
## 1                          0.8648652
```

We now eliminate all variables that correspond to data leakage for the problem (audience_score, imdb_num_votes, critics_score, critics_rating, audience_rating, best_pic_nom, best_pic_win, best_actor_win, best_actress_win, best_dir_win,and top200_box).

```
leakcols = c("audience_score","imdb_num_votes","critics_score","critics_rating","audience_rating","best_
cols = names(movies)
movies = movies[cols[!cols %in% leakcols]]
```

Creating new variables

Here we create new variables, or features, based on the remaining explanatory variables.

Using the year, month, and day of release of each movie, we can construct a day of the week variable (i.e. Monday, Tuesday, etc.). We do this below for both theatre and DVD release dates, calling the new variables `thtr_rel_dow` and `DVD_rel_dow`, respectively.

```
movies$thtr_rel_dow <-as.factor(weekdays(as.Date(paste(movies$thtr_rel_year, movies$thtr_rel_month, movi
movies$dvd_rel_dow <- as.factor(weekdays(as.Date(paste(movies$dvd_rel_year, movies$dvd_rel_month, movie
```

Using the R `gender` library, we can use the first name of the lead actor, from the `actor1` variable, to determine if the lead actor of the rated film is male or female. We call this new variable `ismale`, which is 1 is the actor is male, and is 0 otherwise.

```
movies$ismale <- NA
ismalefun <- function(x) {
  tmp = as.numeric(gender(strsplit(x, " ")[[1]][1])[2])
  if (tmp>0.5 | is.na(tmp)) {return(1)}
  else {return(0)} }
movies$ismale <- lapply(movies$actor1, function(x) sapply(x, ismalefun))
movies$ismale <- as.numeric(movies$ismale)
```

Next, we create two new variables, corresponding to the number of characters in the title, `titlecharnum`, and the number of words in the title, `titlewordnum`.

```
wordcountfun <- function(x) {return(length(strsplit(x," ")[[1]]))}
movies$titlewordnum <- lapply(movies$title, function(x) sapply(x, wordcountfun))
movies$titlewordnum <- as.numeric(movies$titlewordnum)
movies$titlecharnum <- nchar(movies$title)-movies$titlewordnum+1
```

In the count summaries below, we see there are 192 categorical instances of the `studio` variable out of 619 data points, or observations. This variable could end up being a very good predictor (i.e. correlate strongly with the response variable), but the regression model might not generalize well to new data because, for example, the studio(s) in the new data may be different than those in this data set. In other words, if we fit to this catagorical variable, the model could over-fit and not generalize. Let's create a new variable, corresponding to whether or not the production studio is "popular," as measured by frequency of occurances in this dataset. We call this new variable `studiopop`, and set it equal to 1 if the studio is in the top 10 most frequent studios in the data set, and 0 otherwise.

```
studiolist <- movies %>% group_by(studio) %>% summarise(n=n()) %>% arrange(desc(n))
topstudiofun <- function(x) {return(as.numeric(x %in% studiolist$studio[1:10]))}
movies$studiopop <- lapply(movies$studio, function(x) sapply(x, topstudiofun))
movies$studiopop <- as.numeric(movies$studiopop)
```

We now remove all variables that are irrelevant for constructing our multiple regression predictive model. For example, we should remove `thtr_rel_year` and `dvd_rel_year`, because we will use this model to predict ratings/popularity of a movie produced in a year that is not represented in the data set. Also, we should remove variables like `actor1`, `actor2`, etc., and `director`, because there are too many categorical instances for building a generalizable predictive model.

```
irrcols = c("title","actor1","actor2","actor3","actor4","actor5","imdb_url","rt_url","director","studio
cols = names(movies)
movies = movies[cols[!cols %in% irrcols]]
```

Although the four theatre/DVD release month/day variables are numerical, they are not ordinal in the sense that December is greater than May because 12>5. So, we convert them to factors.

```
movies$thtr_rel_month <- as.factor(movies$thtr_rel_month)
movies$thtr_rel_day <- as.factor(movies$thtr_rel_day)
movies$dvd_rel_month <- as.factor(movies$dvd_rel_month)
movies$dvd_rel_day <- as.factor(movies$dvd_rel_day)
```

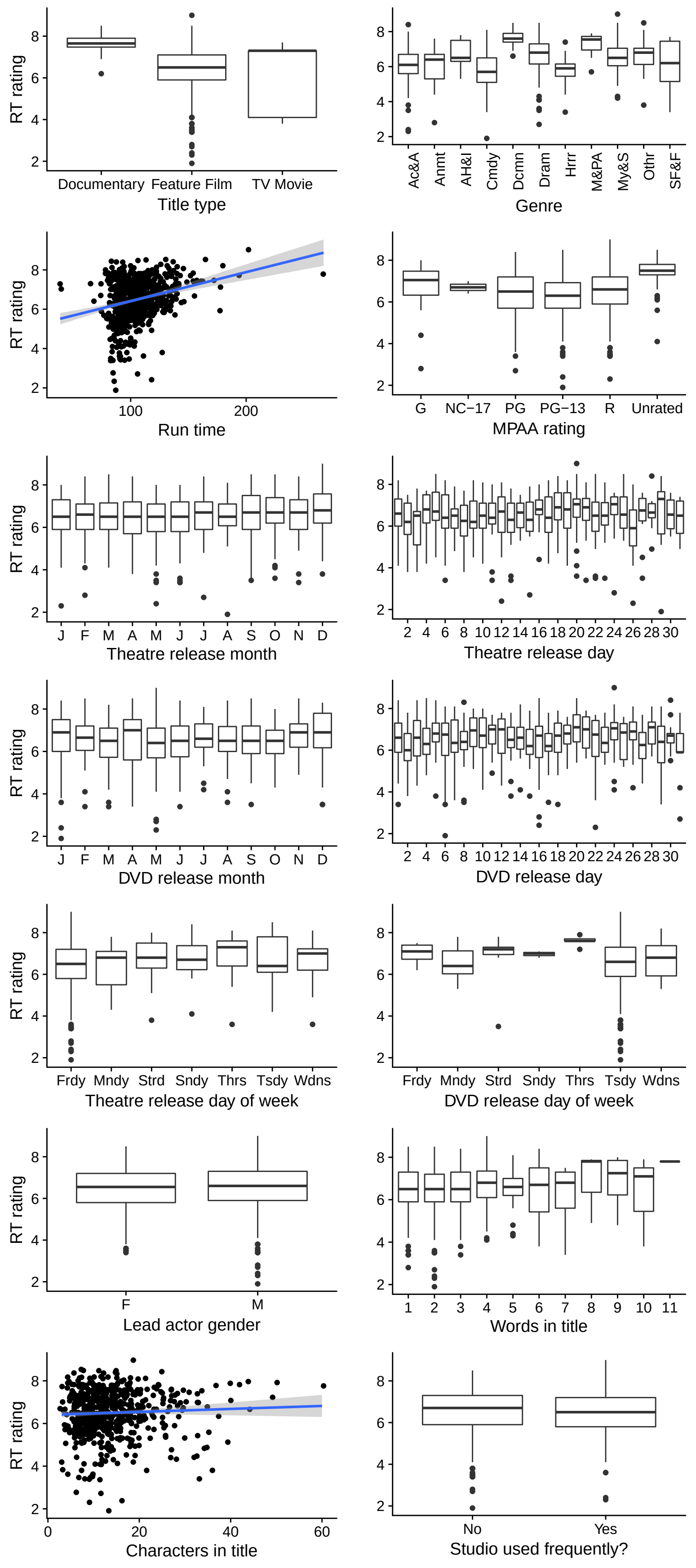
We restrict our dataset to only complete cases, and list the remaining explanatory variable names.

```
movies <- movies[complete.cases(movies),]
cols = names(movies)
movies = movies[cols[!cols %in% c("audience_score")]]
names(movies)
```

```
##  [1] "title_type"     "genre"          "runtime"        "mpaa_rating"
##  [5] "thtr_rel_month" "thtr_rel_day"   "dvd_rel_month"  "dvd_rel_day"
##  [9] "imdb_rating"    "thtr_rel_dow"   "dvd_rel_dow"    "ismale"
## [13] "titlewordnum"   "titlecharnum"   "studiopop"
```

Below, we plot the Rotten Tomatoes rating (RT rating) as a function of each one of the remaining 14 explanatory variables; all of these explanatory variables could in principle be chosen before producing and releasing a movie. Some seem to correlate with `imdb_rating` strongly, while others seem to not correlate at all. Instead of analyzing this in depth here, by predicing Pearson correlation coefficients and/or χ^2 scores from independence tests, we simply eliminate unimportant variables below in our model construction.

```
plot1 <- ggplot(aes(title_type, imdb_rating), data = movies) + geom_boxplot() + ylab("RT rating") + xlab
plot2 <- ggplot(aes(genre, imdb_rating), data = movies) + scale_x_discrete(labels=abbreviate) + theme(a
plot3 <- ggplot(aes(runtime, imdb_rating), data = movies) + geom_jitter() + ylab("RT rating") + xlab("R
plot4 <- ggplot(aes(mpaa_rating, imdb_rating), data = movies) + geom_boxplot() + ylab("") + xlab("MPAA
plot5 <- ggplot(aes(thtr_rel_month, imdb_rating), data = movies) + geom_boxplot() + ylab("RT rating") +
plot6 <- ggplot(aes(thtr_rel_day, imdb_rating), data = movies) + scale_x_discrete(breaks=seq(2, 31, 2))
plot7 <- ggplot(aes(dvd_rel_month, imdb_rating), data = movies) + geom_boxplot() + ylab("RT rating") +
plot8 <- ggplot(aes(dvd_rel_day, imdb_rating), data = movies) + scale_x_discrete(breaks=seq(2, 31, 2))
plot9 <- ggplot(aes(thtr_rel_dow, imdb_rating), data = movies) + scale_x_discrete(labels=abbreviate) +
plot10 <- ggplot(aes(dvd_rel_dow, imdb_rating), data = movies) + scale_x_discrete(labels=abbreviate) +
plot11 <- ggplot(aes(as.factor(ismale), imdb_rating), data = movies) + geom_boxplot() + ylab("RT rating
plot12 <- ggplot(aes(as.factor(titlewordnum), imdb_rating), data = movies) + scale_x_discrete(labels=ab
plot13 <- ggplot(aes(titlecharnum, imdb_rating), data = movies) + geom_jitter() + ylab("RT rating") + x
plot14 <- ggplot(aes(as.factor(studiopop), imdb_rating), data = movies) + scale_x_discrete(labels=c("No
ggarrange(plot1, plot2, plot3, plot4, plot5, plot6, plot7, plot8, plot9, plot10, plot11, plot12, plot13
```


RT rating
Documentary
Feature Film
TV Movie
Title type
Ac&A
Anmt
AH&I
Cmdy
Dcmn
Dram
Hrrr
M&PA
My&S
Othr
SF&F
Genre
Run time
G
NC−17
PG
PG−13
R
Unrated
MPAA rating
J F M A M J J A S O N D
Theatre release month
Theatre release day
DVD release month
DVD release day
Frdy
Mndy
Strd
Sndy
Thrs
Tsdy
Wdns
Theatre release day of week
DVD release day of week
F
M
Lead actor gender
Words in title
Characters in title
No
Yes
Studio used frequently?

Part 4: Modeling

We fit a multiple linear regression model called `m_imdb_rating`, and systematically eliminate unimporant variables using a stepwise backward propagation algorithm considering adjusted R^2 scores. This can be done automatically in R using the `regsubsets` function from the `leaps` library. We perform this analysis below, and print the names of the variables that maximize the adjusted R^2 score. The most relevant variables are `genre`, `runtime`, `thtr_rel_day`, and `mpaa_rating`.

```r
backward <- regsubsets(imdb_rating ~ .,data = movies, method = "backward")
backsum <- summary(backward)
names(which(backsum$which[which.max(backsum$adjr2),]))
```

```
## [1] "(Intercept)"                     "genreArt House & International"
## [3] "genreDocumentary"                "genreDrama"
## [5] "genreMusical & Performing Arts" "genreMystery & Suspense"
## [7] "runtime"                         "mpaa_ratingPG-13"
## [9] "thtr_rel_day26"
```

We now fit a multiple linear regression model using these variables, summarize the model's properties, and perform an ANOVA analysis. The results below show that all variables except for `thtr_rel_day` are statistically significant at the 0.05 confidence level, and the adjusted-R^2 score is 0.2786. This is not a very satisfactory score, in that only 27.86% of the variance of `imdb_rating` is explaned by this model. But, it's the best that can be done at the level of multiple linear regression. We note that the overall p-value is effectively zero, though, indicating that there definitely is a statistically significant relationship between this set of explanatory variables and the dependent variable `imdb_rating`.

For the categorical explanatory variables, the reference values are "Action & Adventure," "1," and "G" for `genre`, `thtr_rel_day`, and `mpaa_rating`, respectively. The model coeffients give some useful insights into the behavior of `imdb_rating`. For example, all else held equal, documentaries are rated 1.719 points higher on average than action & adventure films, movies released on the 26th day of the month are rated 0.678 points lower on average than those release on the 1st day of the month (likely correlating with the Christmas holiday), and G-rated movies are rated higher on average than all other movies by at least 0.468 points.

```r
m_imdb_rating <- lm(imdb_rating ~ genre + runtime + thtr_rel_day + mpaa_rating, data = movies)
summary(m_imdb_rating)
```

```
##
## Call:
## lm(formula = imdb_rating ~ genre + runtime + thtr_rel_day + mpaa_rating,
##     data = movies)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7247 -0.4693  0.0292  0.5906  2.2041
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     5.060881   0.339081  14.925  < 2e-16 ***
## genreAnimation                 -0.330121   0.363331  -0.909 0.363931
## genreArt House & International  0.757014   0.293228   2.582 0.010071 *
## genreComedy                    -0.110246   0.156741  -0.703 0.482103
## genreDocumentary                1.718789   0.206170   8.337 5.31e-16 ***
## genreDrama                      0.636216   0.133383   4.770 2.32e-06 ***
## genreHorror                    -0.109709   0.234002  -0.469 0.639358
## genreMusical & Performing Arts  1.138015   0.295841   3.847 0.000133 ***
## genreMystery & Suspense         0.419464   0.173266   2.421 0.015779 *
## genreOther                       0.544950   0.261906   2.081 0.037888 *
## genreScience Fiction & Fantasy  0.136363   0.350436   0.389 0.697323
## runtime                          0.013991   0.002023   6.915 1.21e-11 ***
## thtr_rel_day2                    0.140969   0.278998   0.505 0.613558
## thtr_rel_day3                   -0.249198   0.279490  -0.892 0.372960
## thtr_rel_day4                    0.315315   0.298200   1.057 0.290761
## thtr_rel_day5                    0.173828   0.232880   0.746 0.455704
## thtr_rel_day6                   -0.259636   0.228347  -1.137 0.255987
## thtr_rel_day7                    0.045026   0.218407   0.206 0.836739
## thtr_rel_day8                   -0.077286   0.223535  -0.346 0.729658
## thtr_rel_day9                    0.082139   0.248824   0.330 0.741435
## thtr_rel_day10                   0.109403   0.220481   0.496 0.619934
## thtr_rel_day11                   0.053906   0.202392   0.266 0.790066
## thtr_rel_day12                   0.037603   0.229182   0.164 0.869728
## thtr_rel_day13                  -0.354089   0.248294  -1.426 0.154368
## thtr_rel_day14                   0.081648   0.270206   0.302 0.762627
## thtr_rel_day15                  -0.192648   0.207650  -0.928 0.353911
## thtr_rel_day16                   0.283418   0.236939   1.196 0.232109
## thtr_rel_day17                  -0.068472   0.221989  -0.308 0.757848
## thtr_rel_day18                   0.444612   0.229147   1.940 0.052817 .
## thtr_rel_day19                   0.184292   0.213393   0.864 0.388142
## thtr_rel_day20                   0.206947   0.218252   0.948 0.343411
## thtr_rel_day21                   0.361639   0.217872   1.660 0.097468 .
## thtr_rel_day22                  -0.203045   0.221415  -0.917 0.359496
## thtr_rel_day23                  -0.055065   0.254672  -0.216 0.828890
## thtr_rel_day24                   0.195670   0.274761   0.712 0.476654
## thtr_rel_day25                   0.239830   0.225071   1.066 0.287050
## thtr_rel_day26                  -0.677537   0.268136  -2.527 0.011768 *
## thtr_rel_day27                   0.177633   0.242211   0.733 0.463615
## thtr_rel_day28                   0.107966   0.267851   0.403 0.687032
## thtr_rel_day29                   0.043496   0.243375   0.179 0.858219
## thtr_rel_day30                   0.092960   0.254315   0.366 0.714845
## thtr_rel_day31                  -0.040557   0.390307  -0.104 0.917275
## mpaa_ratingNC-17                -0.493983   0.706775  -0.699 0.484871
## mpaa_ratingPG                   -0.562790   0.264320  -2.129 0.033648 *
## mpaa_ratingPG-13                -0.856803   0.269788  -3.176 0.001571 **
## mpaa_ratingR                    -0.529476   0.260115  -2.036 0.042239 *
## mpaa_ratingUnrated              -0.467500   0.299973  -1.558 0.119653
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9126 on 595 degrees of freedom
## Multiple R-squared:  0.3303, Adjusted R-squared:  0.2786
## F-statistic: 6.381 on 46 and 595 DF,  p-value: < 2.2e-16
```

```r
anova(m_imdb_rating)
```

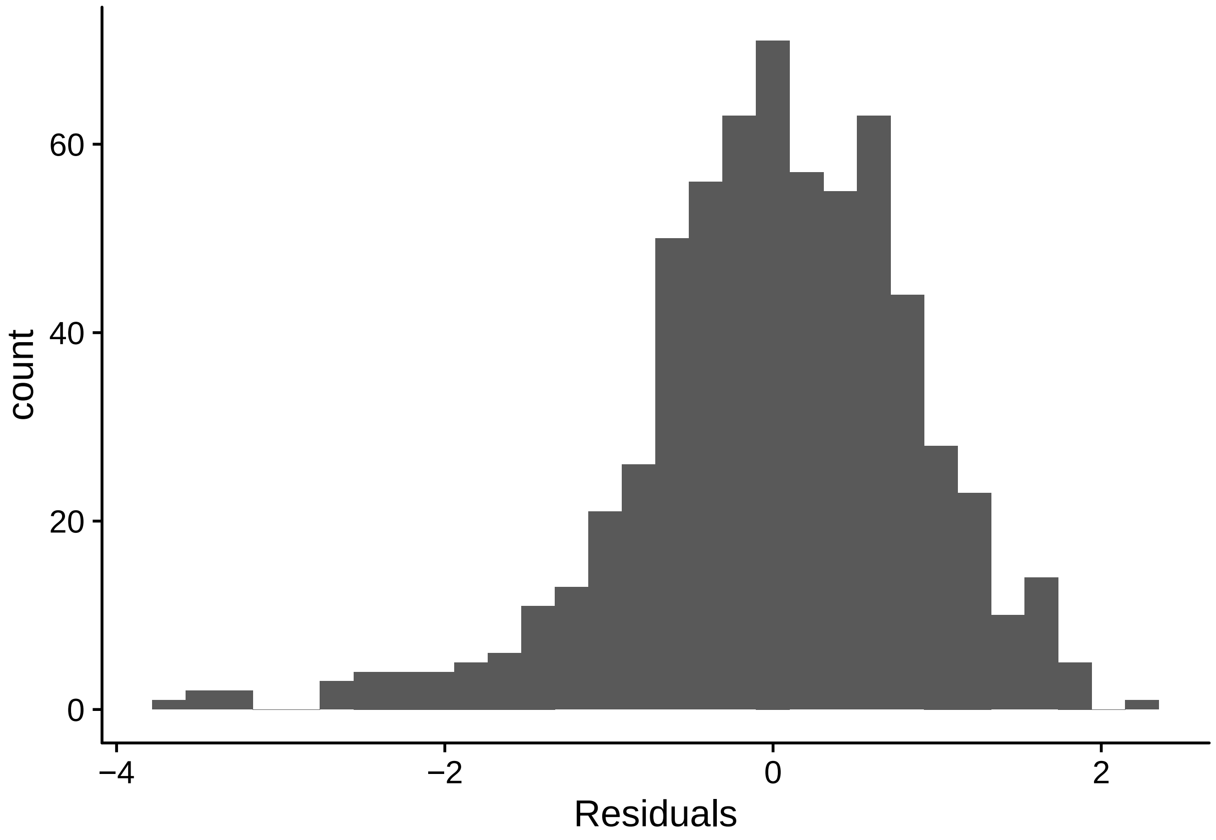
```
## Analysis of Variance Table
##
## Response: imdb_rating
##                Df Sum Sq Mean Sq F value    Pr(>F)
## genre          10 164.81  16.481 19.7890 < 2.2e-16 ***
## runtime         1  36.60  36.597 43.9423 7.593e-11 ***
## thtr_rel_day   30  29.08   0.969  1.1637  0.253121
## mpaa_rating     5  13.97   2.793  3.3542  0.005368 **
## Residuals    595 495.54   0.833
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To further analyze the model, we explore the residuals to ensure that they are normally distributed about zero and exhibit constant variability. We also check linear dependence with respect to any/all numerical explanatory variables, and ensure that the data points are independent with a visual inspection of a time series.

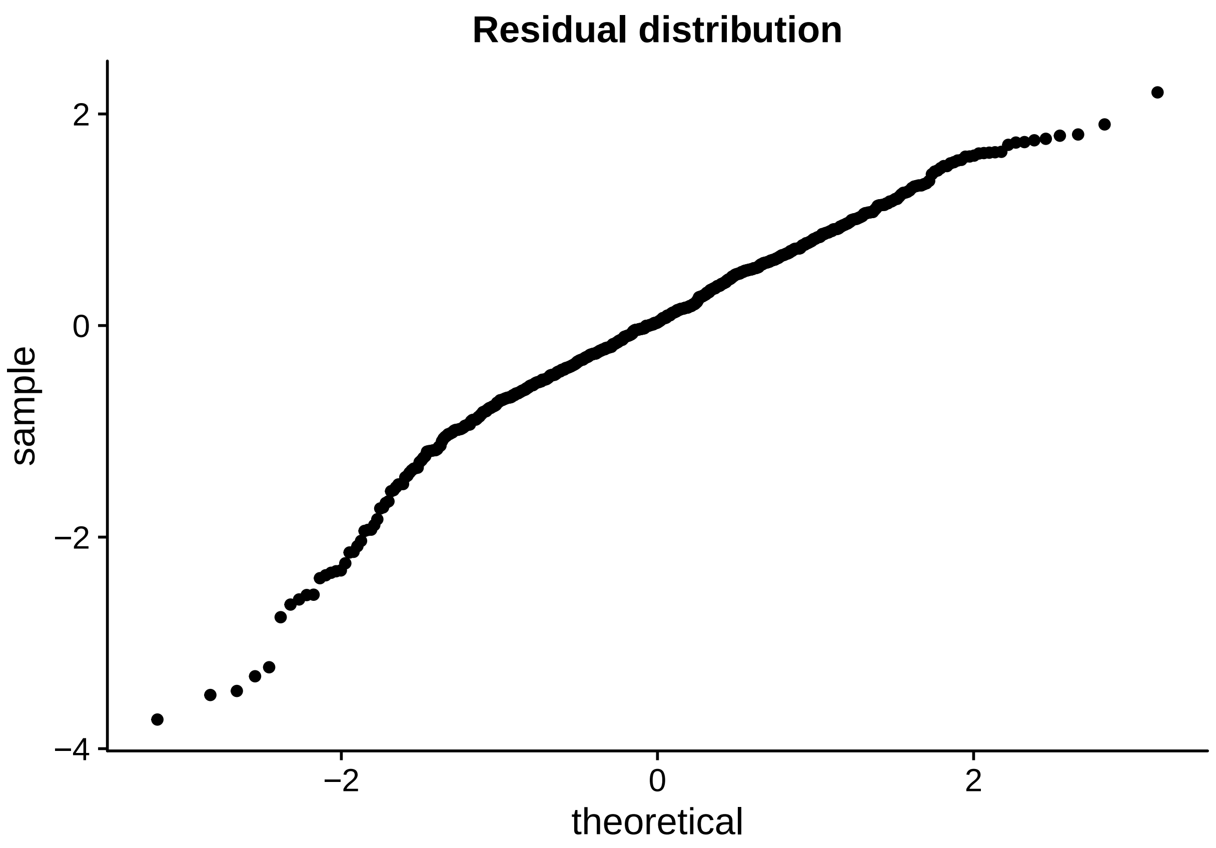
The histogram and Q-Q plot below show that the residuals are approximately normally distributed, with a slight left skew.

```r
ggplot(aes(m_imdb_rating$residuals), data = m_imdb_rating) + geom_histogram() + xlab("Residuals")
```

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



```r
ggplot(data = m_imdb_rating, aes(sample = .resid)) + stat_qq() + ggtitle("Residual distribution")
```

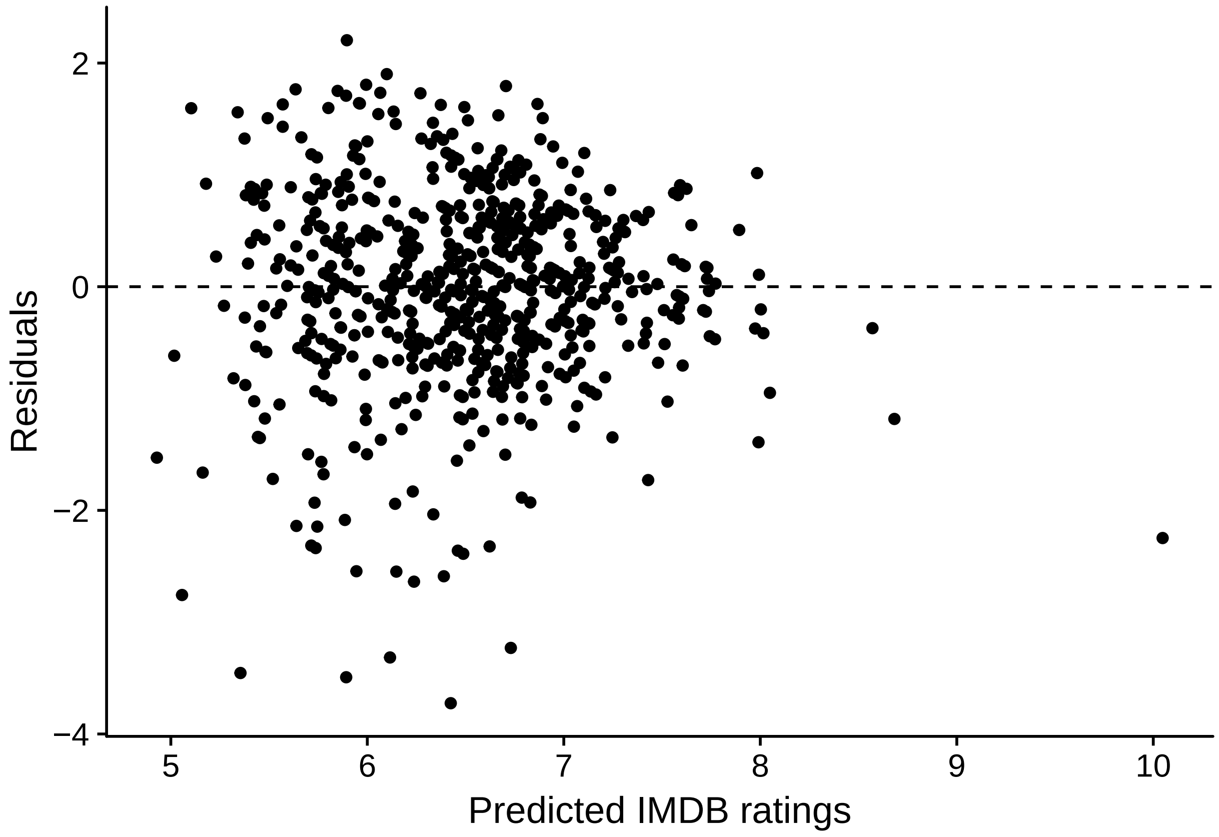


```
summary(m_imdb_rating$residuals)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.72500 -0.46930  0.02918  0.00000  0.59060  2.20400
```

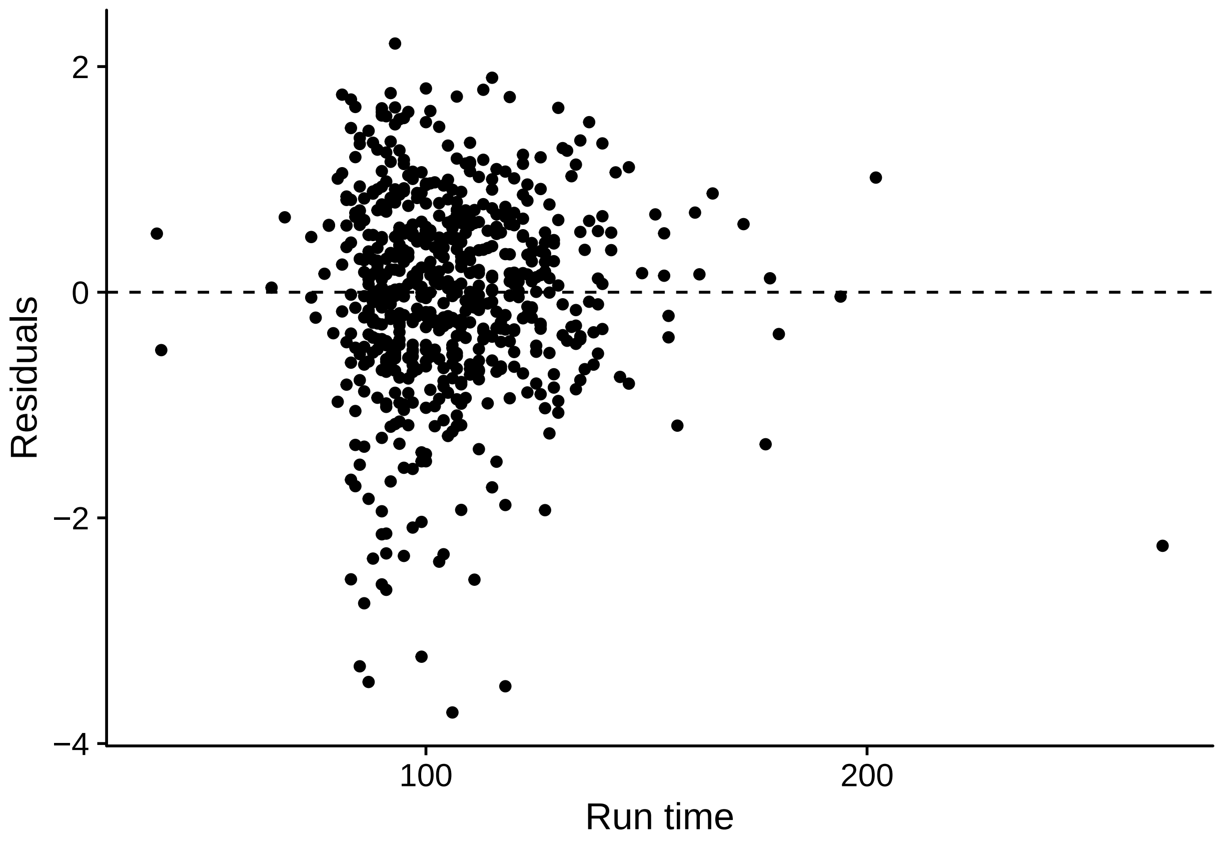
The scatter plot below shows the residuals as a function of the predicted IMDB ratings. This shows a slight "fan" shape, but it is not too noticeable, so we can conclude that the residuals have approximately constant variability.

```
ggplot(data = m_imdb_rating, aes(x = .fitted, y = .resid)) + geom_point() + geom_hline(yintercept = 0, l
```



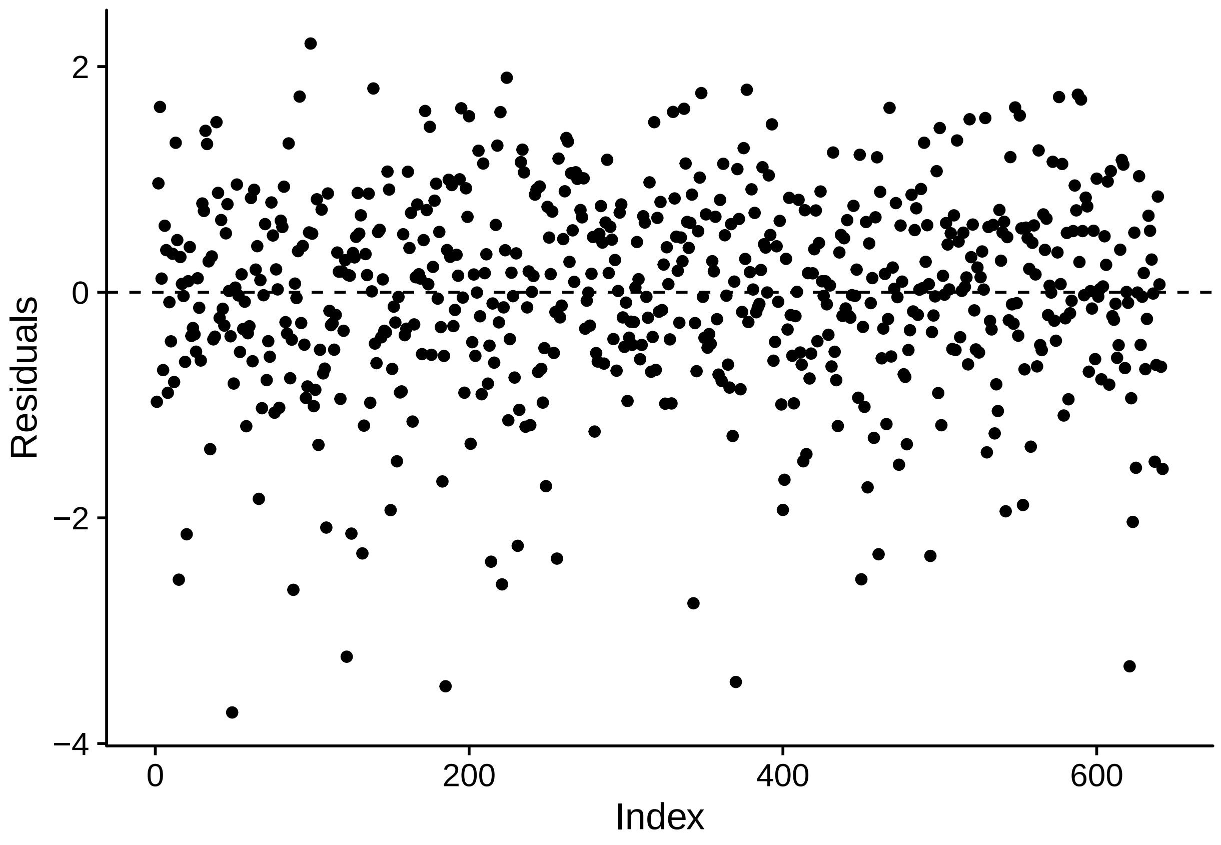
The only numerical explanatory variable is `runtime`. Below we show a scatterplot of the residuals with respect to the `runtime` variable. The data looks rather randomly scattered, indicating that the dependence is approximately linear.

```
ggplot(aes(x = movies$runtime, y = m_imdb_rating$residuals), data = m_imdb_rating) + geom_point() + xla
```



Finally, we inspect a scatter plot of the residuals with respect to the data frame index. There are no obvious time correlations, indicating that the data points are independent.

```
ggplot(aes(x = seq(1, length(m_imdb_rating$residuals)), y = m_imdb_rating$residuals), data = m_imdb_rat
```



Part 5: Prediction

Here, we predict the IMDB rating of the 2016 movie "Money Monster," chosen at random from the IMDB website:

http://www.imdb.com/title/tt2241351/

The IMDB rating is 6.5. Below, we use the multiple linear regression model `m_imdb_rating` to predict the score, and make a comparison. The predicted score, as shown below, is 6.073. The 95% confidence interval for the prediction is [4.22,7.92], meaning there is a 95% chance a movie with these features will have an IMDB score that falls in this interval. Sure enough, the movie score (6.5) does fall in this interval.

```
genre <- "Drama"
mpaa_rating <- "R"
thtr_rel_day <- as.factor(13)
runtime <- 90
newmovie <- data.frame(genre, mpaa_rating, thtr_rel_day, runtime)
RTrating_predict <- predict(m_imdb_rating, newmovie, interval = "prediction", level=.95)
RTrating_predict
```

```
##        fit      lwr      upr
## 1 6.072679 4.223801 7.921557
```

Part 6: Conclusion

To conclude, we developed a multiple linear regression model to predict the IMDB rating of a film given many variables and/or features that could be chosen before the film's production and release. The linear model seems to be a relativly good choice for a predictive model, although there is a great deal of variance in the relationship between IMDB rating and movie run time (the only ordinal numerical variable used in the model). To develop a stronger predictive model, more variables may be useful. For example, knowing the IMDB ratings of movies the actors have been in previously seems intuitively useful. First steps towards future work on this project would include generating such features.