Modeling and prediction for movies

Setup

Load packages

```
library(ggplot2)
library(dplyr)
library(statsr)
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 3.4.3
```

Load data

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

Part 1: Data

The data set contains 651 randomly sampled movies that were produced and released prior to 2016, including additional information that has been gathered from movie websites IMBD and Rotten Tomatoes.

Consequently, we are dealing with observational data. Since random sampling was utilized to generate the sample, we can generalize our findings to all movies. However, since this is not an experimental study, we cannot infer causality from the data. While there are advanced methods to discover causal relationships in observational data, they were not covered in the course and will not be used in this analysis.

Part 2: Research question

We will use the data in order to predict the IMDB rating using a subset of the other predictors in the data set. More specifically, we will focus on the question:

Which properties of movies are associated with high IMDB ratings and how large is their association with the expected IMDB rating?

This question is interesting for tweo reasons: first, it gives insight about which characteristics are valued at IMDB. Second, it also gives us insight about the correlation between the IMDB score and the received score on Rotten Tomatoes. This can explain whether the scores received on both websites are similar or not.

The properties we are considering are only those included in the original data set and additional properties than can be derived using those variables. To ensure staying within the scope of the project, no external data will be gathered.

Part 3: Exploratory data analysis

Our analysis begins with a few plots to obtain a feeling for the data at hand. First, let us investigate the distribution of rankings on IMDB and Rotten Tomatoes, which will be abbreviated as RT for the remainder of the analysis.

```
summary(select(movies, imdb_rating, critics_score, audience_score))
```

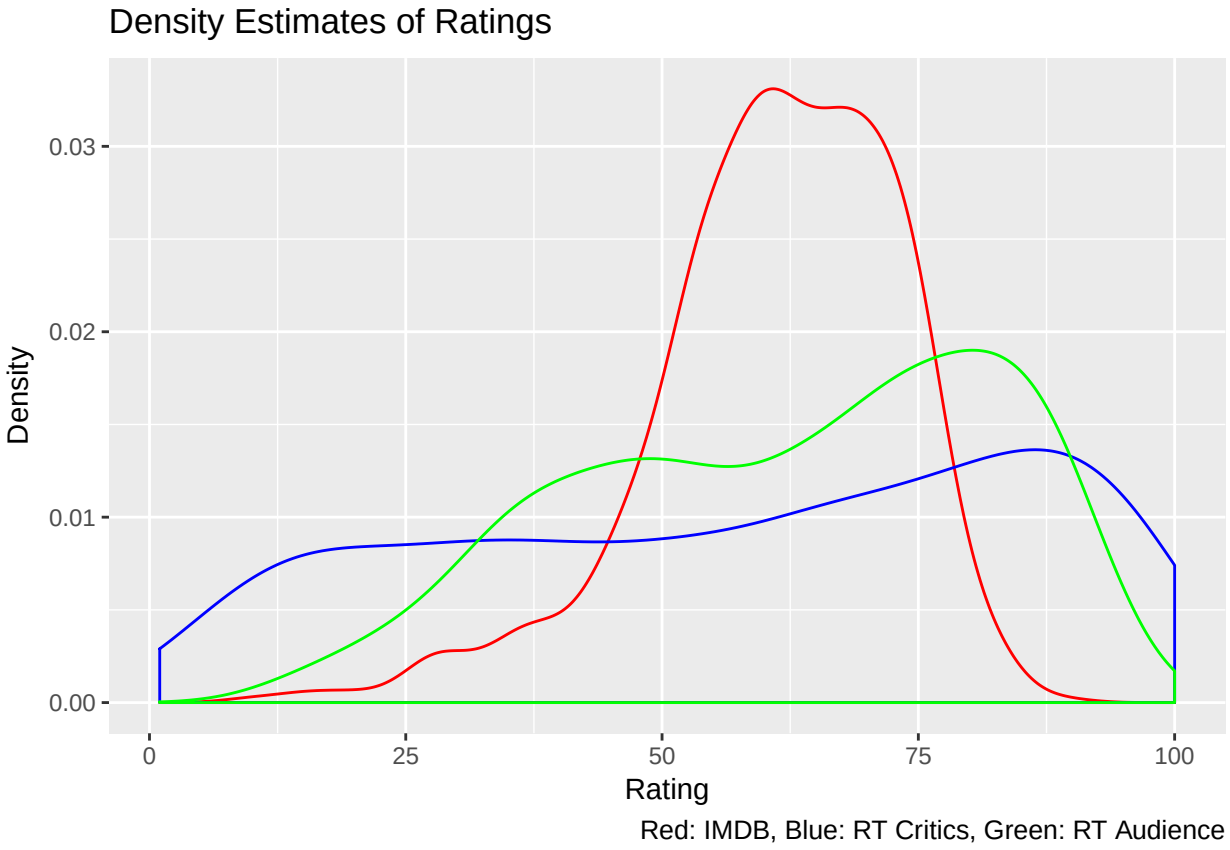
```
##   imdb_rating    critics_score    audience_score
##  Min.   :1.900   Min.   :  1.00   Min.   :11.00
##  1st Qu.:5.900   1st Qu.: 33.00   1st Qu.:46.00
##  Median :6.600   Median : 61.00   Median :65.00
##  Mean   :6.493   Mean   : 57.69   Mean   :62.36
##  3rd Qu.:7.300   3rd Qu.: 83.00   3rd Qu.:80.00
##  Max.   :9.000   Max.   :100.00   Max.   :97.00
```

Our first observation is that the data is on different scales. A short test on the websites revealed that the scores on IMDB range from 1 (minimum) to 10 (maximum). The RT scores on the other hand are according to my research on a 0-100 scale. This means we will have to scale them to be comparable. We will use a simple linear scaling that transforms IMDB ratings to a 0 to 100 scale instead of a 1 to 10 scale.

```
movies$imdb_rating <- 100/9 * movies$imdb_rating - 100/9
```

Let us plot density estimates of the three ratings into a single plot.

```
ggplot(data = movies) +
  geom_density(mapping = aes(x = imdb_rating), color = "red") +
  geom_density(mapping = aes(x = critics_score), color = "blue") +
  geom_density(mapping = aes(x = audience_score), color = "green") +
  xlab("Rating") +
  ylab("Density") +
  labs(title = "Density Estimates of Ratings",
       caption = "Red: IMDB, Blue: RT Critics, Green: RT Audience")
```



As we can see, the RT ratings have far more spread, especially the audience ones. Next, we will look at pairwise correlations. To do so, we will utilize the ggpairs function. However, we need to dispose of variables that we will not include.

There a few variables that we should omit in the first step for several reasons.

Title, Actor columns, Director, Studio: While there are probably a few titles, actors, directors and studios that are guaranteed to receive good ratings, we will not consider them due to potential overfitting.

Date columns: We will convert the date columns into single date-format columns.

Best Picture Oscar: We will merge these variables into one variable with three levels: "No", "Nominated" and "Received" due to collinearity of the two columns.

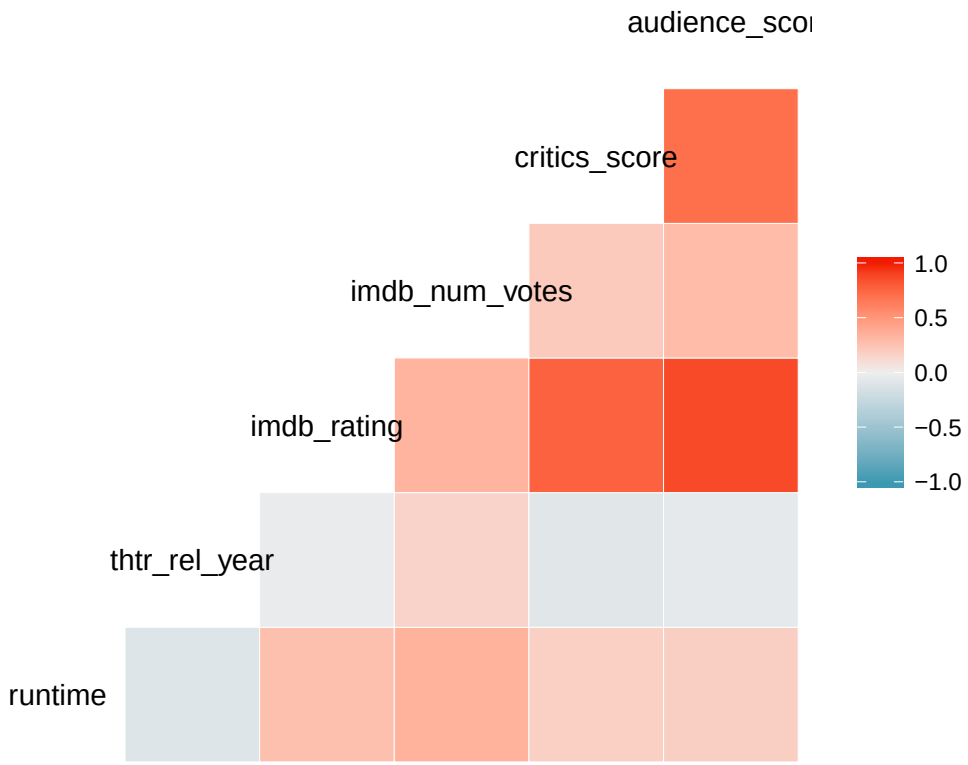
Other variables might be removed later on.

We will start doing the data manipulations mentioned above.

```
movies <- movies %>%
  mutate(date_theater_release = as.POSIXct(paste(movies$thtr_rel_year, movies$thtr_rel_month, movies$th
  select(-c(thtr_rel_month, thtr_rel_day, dvd_rel_year, dvd_rel_month, dvd_rel_day, actor1, actor2, act
```

To avoid issues with collinearity, we should drop some highly correlated covariates. We can use the ggcorr function to calculate correlation between our numerical predictors. Please note that the following code produces a warning because not all columns contain numerical data when warnings are not suppressed.

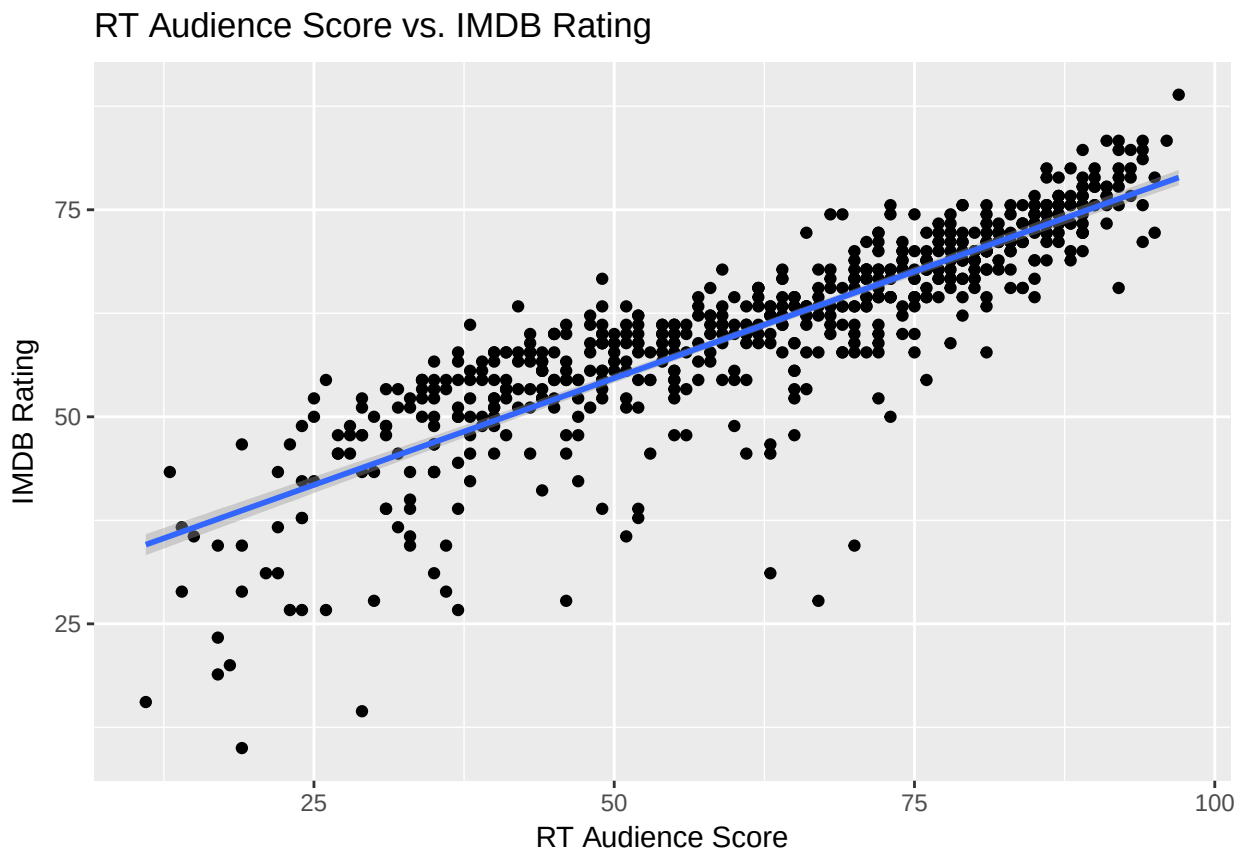
```
ggcorr(movies)
```



We can see that the three score variables are strongly correlated, so the audience score will probably be a valuable predictor for the IMDB rating. However, since critics score and audience score are also strongly correlated, we should remove the critics score from the model.

Our most important predictor will probably be the rating on Rotten Tomatoes, so we should make a plot:

```
ggplot(movies, aes(x = audience_score, y = imdb_rating)) +
  geom_point() +
  geom_smooth(method = "lm") +
  xlab("RT Audience Score") +
  ylab("IMDB Rating") +
  labs(title = "RT Audience Score vs. IMDB Rating")
```



The correlation is easily visible in the visualization.

Next, we should think about the critics rating and the audience rating and their respective scores. Let us summarise their relationships:

```
movies %>%
  select(critics_rating, critics_score) %>%
  group_by(critics_rating) %>%
  summarise(min_score = min(critics_score), max_score = max(critics_score))
```

```
## # A tibble: 3 x 3
##    critics_rating min_score max_score
##            <fctr>     <dbl>     <dbl>
## 1 Certified Fresh        72       100
## 2           Fresh        60       100
## 3          Rotten         1        59
```

```
movies %>%
  select(audience_rating, audience_score) %>%
  group_by(audience_rating) %>%
  summarise(min_score = min(audience_score), max_score = max(audience_score))
```

```
## # A tibble: 2 x 3
##   audience_rating min_score max_score
##            <fctr>     <dbl>     <dbl>
## 1         Spilled        11        59
## 2         Upright        60        97
```

Our observation is confirmed by some research on the official RT website: The cutoff for the ratings is 60,

and Certified Fresh is a special rating that does not only require a score of at least 60 but also several other factors such as a number of reviews by recognized reviewers. Since little additional information is provided by these variables, we will exclude them from our further analysis. We will later examine whether there is special need to include these variables.

Next, our two date columns are moderately correlated:

```
with(data = movies, expr = cor(as.integer(date_DVD_release), as.integer(date_theater_release), use = "c
```

```
## [1] 0.6610352
```

So we should remove one of them as well. We will keep the theater_release_date as probably most important reviews are based on the movie theater version. We can also remove the theater release year from the data set since this information is already included in the date of theater release. We will also convert this date to an integer, measuring the number of days since 1970-01-01.

```
movies <- movies %>%
  select(-c(critics_rating, audience_rating, date_DVD_release, thtr_rel_year)) %>%
  mutate(t = as.integer(date_theater_release)) %>%
  select(-date_theater_release)
```

Let us check whether there is an association between rating and the top200_box variable.

```
movies %>%
  select(imdb_rating, audience_score, top200_box) %>%
  group_by(top200_box) %>%
  summarise(avg_imdb_rating = mean(imdb_rating), avg_audience_score = mean(audience_score))
```

```
## # A tibble: 2 x 3
##   top200_box avg_imdb_rating avg_audience_score
##       <fctr>           <dbl>              <dbl>
## 1         no        60.86478           62.07547
## 2        yes        68.22222           74.53333
```

It seems that top200 box movies have higher IMDB ratings on average, so this might carry information. However, since this effect is also contained in the audience score, we can remove the top200 box variable due to its association with the audience score.

```
movies <- movies %>%
  select(-top200_box)
```

The remaining predictors are more difficult to rate. The number of variables is low enough to start with the model selection, so we will start with the modeling step.

Part 4: Modeling

We will now start building the model. We will stay within the scope of the course and use linear regression without any additional preprocessing such as scaling the data or using dimensionality reduction methods and without regularization methods. For model selection, we will use backwards feature selection with the p-value as selection criterion. We have to be careful with overfitting, so using a separate test data set would be beneficial. We could probably receive slightly better results when using the adjusted R^2 as a criterion. However, since the assignment requires manual feature selection, we will use the method that requires us to build fewer models. In a more serious pattern, we should follow a more precise technique such as using a designated test set, the R^2 criterion or even a best subset selection which is still feasible with only 12 predictors and suitable preprocessing steps.

We start with the full model, including all predictors. We remove the predictor with the highest p-value and use the remaining predictors to build a new model. Repeating this step until all predictors with p-values greater than the threshold are removed yields the final model. Note that we will use 95% significance for our predictors.

```
fit <- lm(data = movies, formula = imdb_rating ~ title_type + genre + runtime + mpaa_rating +
            imdb_num_votes + audience_score + best_actor_win + best_actress_win +
            best_dir_win + t + best_picture_oscar)
```

For our first model, we can see that the time factor is highly non-significant, so we drop it from our model and fit the same model again.

```
fit <- lm(data = movies, formula = imdb_rating ~ title_type + genre + runtime + mpaa_rating +
            imdb_num_votes + audience_score + best_actor_win + best_actress_win +
            best_dir_win + best_picture_oscar)
```

Next, we remove the best picture Oscar from the model.

```
fit <- lm(data = movies, formula = imdb_rating ~ title_type + genre + runtime + mpaa_rating +
            imdb_num_votes + audience_score + best_actor_win + best_actress_win +
            best_dir_win)
```

The least significant predictor now is best actor win, so we remove it as well.

```
fit <- lm(data = movies, formula = imdb_rating ~ title_type + genre + runtime + mpaa_rating +
            imdb_num_votes + audience_score + best_actress_win + best_dir_win)
```

The best actress Oscar is now the least significant, so we remove it.

```
fit <- lm(data = movies, formula = imdb_rating ~ title_type + genre + runtime + mpaa_rating +
            imdb_num_votes + audience_score + best_dir_win)
```

The best director is the least significant predictor now, so we can remove it as well.

```
fit <- lm(data = movies, formula = imdb_rating ~ title_type + genre + runtime + mpaa_rating +
            imdb_num_votes + audience_score)
```

The next feature to remove is the title type.

```
fit <- lm(data = movies, formula = imdb_rating ~ genre + runtime + mpaa_rating +
            imdb_num_votes + audience_score)
```

All other predictors are significant. Please note that for a factor variable with more than two levels the absence of any significant associated dummy variables is required for it to be removed in feature selection based on p-value.

We will now have a look at the complete summary for this model:

```
summary(fit)
```

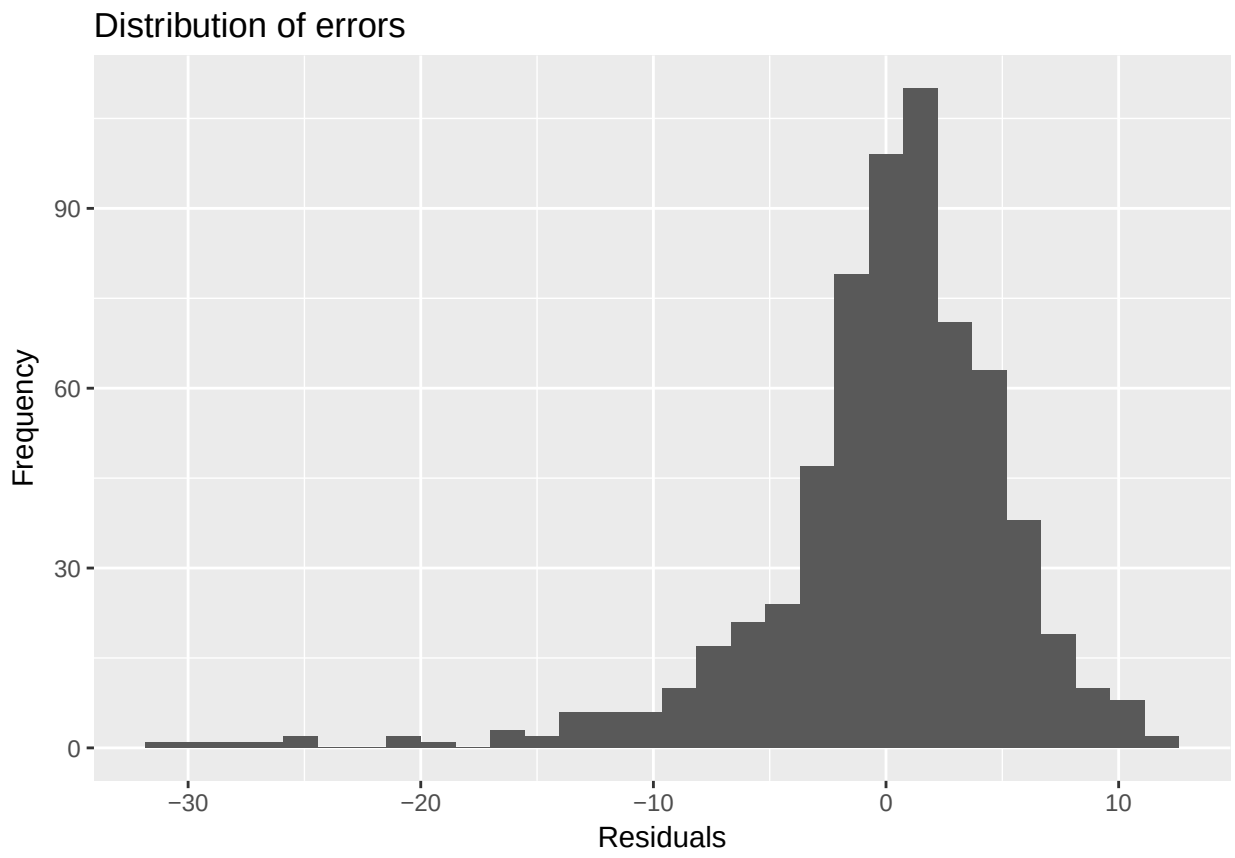
```
##
## Call:
## lm(formula = imdb_rating ~ genre + runtime + mpaa_rating + imdb_num_votes +
##     audience_score, data = movies)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -31.2692  -1.9423   0.7072   2.9978  11.6599
##
## Coefficients:
```

```
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     2.681e+01  2.037e+00  13.165  < 2e-16 ***
## genreAnimation                 -5.171e+00  2.170e+00  -2.383  0.01749 *
## genreArt House & International  3.137e+00  1.692e+00   1.855  0.06412 .
## genreComedy                    -9.890e-01  9.268e-01  -1.067  0.28637
## genreDocumentary                6.190e+00  1.277e+00   4.846 1.59e-06 ***
## genreDrama                      2.368e+00  8.040e-01   2.946  0.00334 **
## genreHorror                     2.244e+00  1.389e+00   1.615  0.10675
## genreMusical & Performing Arts  2.714e+00  1.810e+00   1.499  0.13435
## genreMystery & Suspense         4.249e+00  1.028e+00   4.135 4.03e-05 ***
## genreOther                      6.982e-01  1.569e+00   0.445  0.65646
## genreScience Fiction & Fantasy -1.213e+00  1.982e+00  -0.612  0.54072
## runtime                         5.339e-02  1.288e-02   4.145 3.86e-05 ***
## mpaa_ratingNC-17               -2.153e-01  4.214e+00  -0.051  0.95927
## mpaa_ratingPG                  -2.596e+00  1.532e+00  -1.694  0.09067 .
## mpaa_ratingPG-13               -3.226e+00  1.574e+00  -2.049  0.04088 *
## mpaa_ratingR                   -2.198e+00  1.519e+00  -1.447  0.14847
## mpaa_ratingUnrated             -2.016e+00  1.740e+00  -1.159  0.24689
## imdb_num_votes                  1.028e-05  2.280e-06   4.507 7.82e-06 ***
## audience_score                  4.552e-01  1.324e-02  34.385  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.563 on 631 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.7929, Adjusted R-squared:  0.787
## F-statistic: 134.2 on 18 and 631 DF,  p-value: < 2.2e-16
```

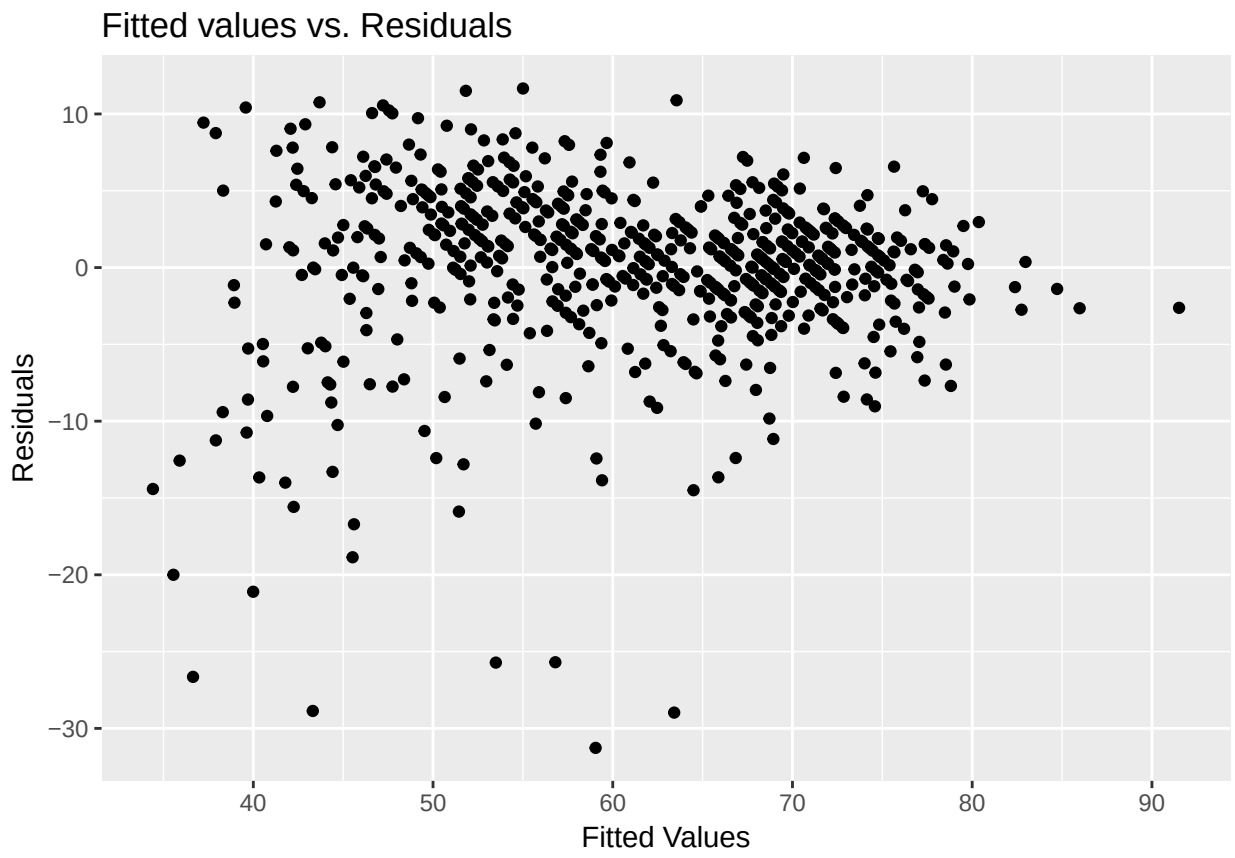
As we can see, we have an adjusted R^2 of 0.787 which is not a bad result. However, we cannot conclude the validity of the model without performing model diagnostics.

```
ggplot(data = fit, aes(x = .resid)) +
  geom_histogram() +
  xlab("Residuals") +
  ylab("Frequency") +
  labs(title = "Distribution of errors")
```

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



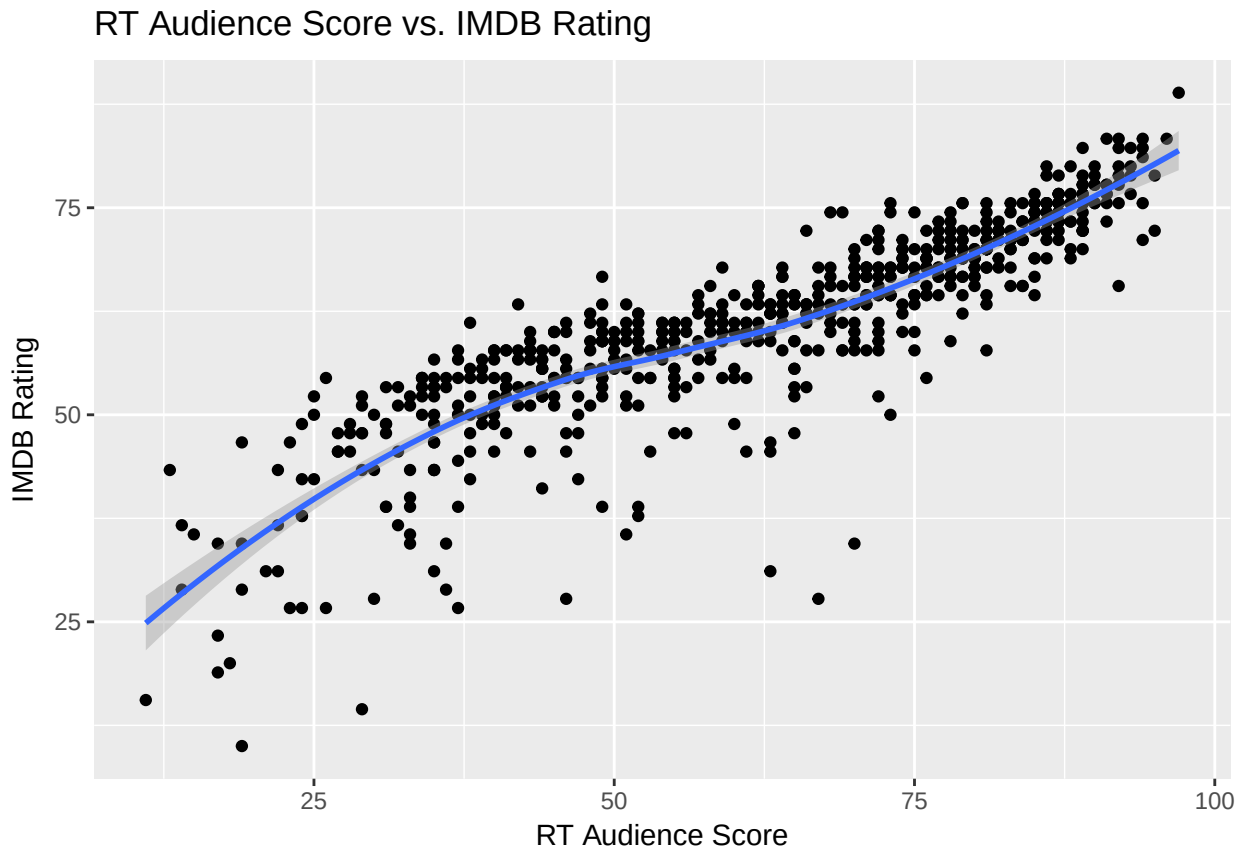

```
ggplot(data = fit, aes(x = .fitted, y = .resid)) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  labs(title = "Fitted values vs. Residuals")
```



As we can see, the distribution of the residuals is left-skewed and the residuals have a characteristic fan shape, so we have to deal with heteroscedasticity. We will make an attempt to find the reasons for this issue. It might in fact be the case that the residuals appear to have varying standard deviations because there is an unseen nonlinear relationship. To see this, we plot the main predictor and the response in a scatter plot once again, this time using a smoothing curve:

```
ggplot(movies, aes(x = audience_score, y = imdb_rating)) +
  geom_point() +
  geom_smooth(method = "auto") +
  xlab("RT Audience Score") +
  ylab("IMDB Rating") +
  labs(title = "RT Audience Score vs. IMDB Rating")
```

```
## `geom_smooth()` using method = 'loess'
```



Indeed, we can observe that the relationship looks less linear below the 35 mark and above the 85 score mark. We can think about fitting a model including a transformed feature. The plot above should remind of a third degree polynomial. Consequently, we should try and add squared and cubic variations of the audience score.

```
fit <- lm(data = movies, formula = imdb_rating ~ genre + runtime + mpaa_rating +
            imdb_num_votes + audience_score + I(audience_score^2) + I(audience_score^3))
summary(fit)
```

```
##
## Call:
## lm(formula = imdb_rating ~ genre + runtime + mpaa_rating + imdb_num_votes +
##     audience_score + I(audience_score^2) + I(audience_score^3),
##     data = movies)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -30.2663  -1.9343   0.6222   2.9363  15.1014
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     2.174e-01  4.035e+00   0.054 0.957052
## genreAnimation                 -4.284e+00  2.088e+00  -2.052 0.040547 *
## genreArt House & International  3.362e+00  1.623e+00   2.072 0.038711 *
## genreComedy                    -1.212e+00  8.901e-01  -1.361 0.173989
## genreDocumentary                6.346e+00  1.241e+00   5.114 4.20e-07 ***
## genreDrama                      2.739e+00  7.728e-01   3.545 0.000422 ***
## genreHorror                     1.803e+00  1.334e+00   1.351 0.177046
```

```
## genreMusical & Performing Arts  2.951e+00  1.745e+00   1.691 0.091312 .
## genreMystery & Suspense         4.219e+00  9.858e-01   4.279 2.17e-05 ***
## genreOther                      1.199e+00  1.507e+00   0.796 0.426436
## genreScience Fiction & Fantasy  2.930e-02  1.910e+00   0.015 0.987767
## runtime                         5.200e-02  1.237e-02   4.202 3.03e-05 ***
## mpaa_ratingNC-17               -4.986e-01  4.047e+00  -0.123 0.901991
## mpaa_ratingPG                  -2.622e+00  1.470e+00  -1.784 0.074940 .
## mpaa_ratingPG-13               -3.315e+00  1.512e+00  -2.193 0.028682 *
## mpaa_ratingR                   -2.379e+00  1.458e+00  -1.632 0.103155
## mpaa_ratingUnrated             -1.924e+00  1.669e+00  -1.153 0.249445
## imdb_num_votes                  9.749e-06  2.281e-06   4.275 2.21e-05 ***
## audience_score                  2.081e+00  2.195e-01   9.480  < 2e-16 ***
## I(audience_score^2)            -2.939e-02  4.126e-03  -7.122 2.91e-12 ***
## I(audience_score^3)             1.633e-04  2.427e-05   6.729 3.84e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.336 on 629 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:   0.81,  Adjusted R-squared:  0.804
## F-statistic: 134.1 on 20 and 629 DF,  p-value: < 2.2e-16
```

We can see that the new polynomial features are highly significant, so we will not start over with feature elimination. Let us look at the residuals in a bit more detail:

```
head(movies[order(-I(abs(fit$residuals))), c("imdb_rating", "audience_score",
                                             "critics_score", "genre", "title_type")], 10)
```

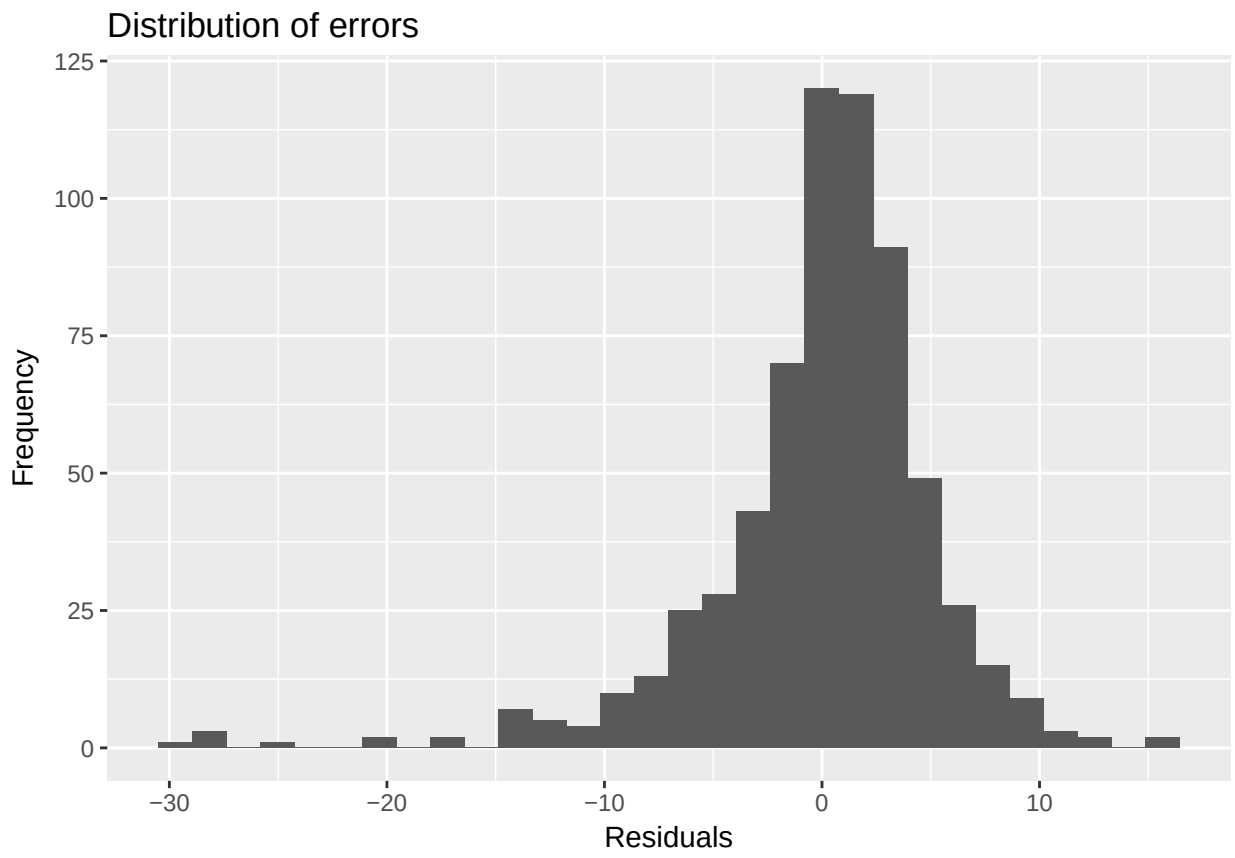
```
## # A tibble: 10 x 5
##    imdb_rating audience_score critics_score                    genre
##          <dbl>          <dbl>         <dbl>                   <fctr>
##  1    27.77778             67             4                   Comedy
##  2    76.66667             87           100              Documentary
##  3    34.44444             70            29                    Drama
##  4    27.77778             46            35                    Drama
##  5    31.11111             63            10                   Comedy
##  6    40.00000             33            27 Art House & International
##  7    26.66667             37             3                   Comedy
##  8    28.88889             36            15                   Comedy
##  9    53.33333             34            31       Action & Adventure
## 10    64.44444             64            53       Mystery & Suspense
## # ... with 1 more variables: title_type <fctr>
```

As we can see, the movies that have been predicted the worst are movies for which different scores vary extremely. It is a difficult task to rank these movies correctly with the data at hand. An interesting thing to notice is that the mispredictions are mostly associated with movies that received low scores.

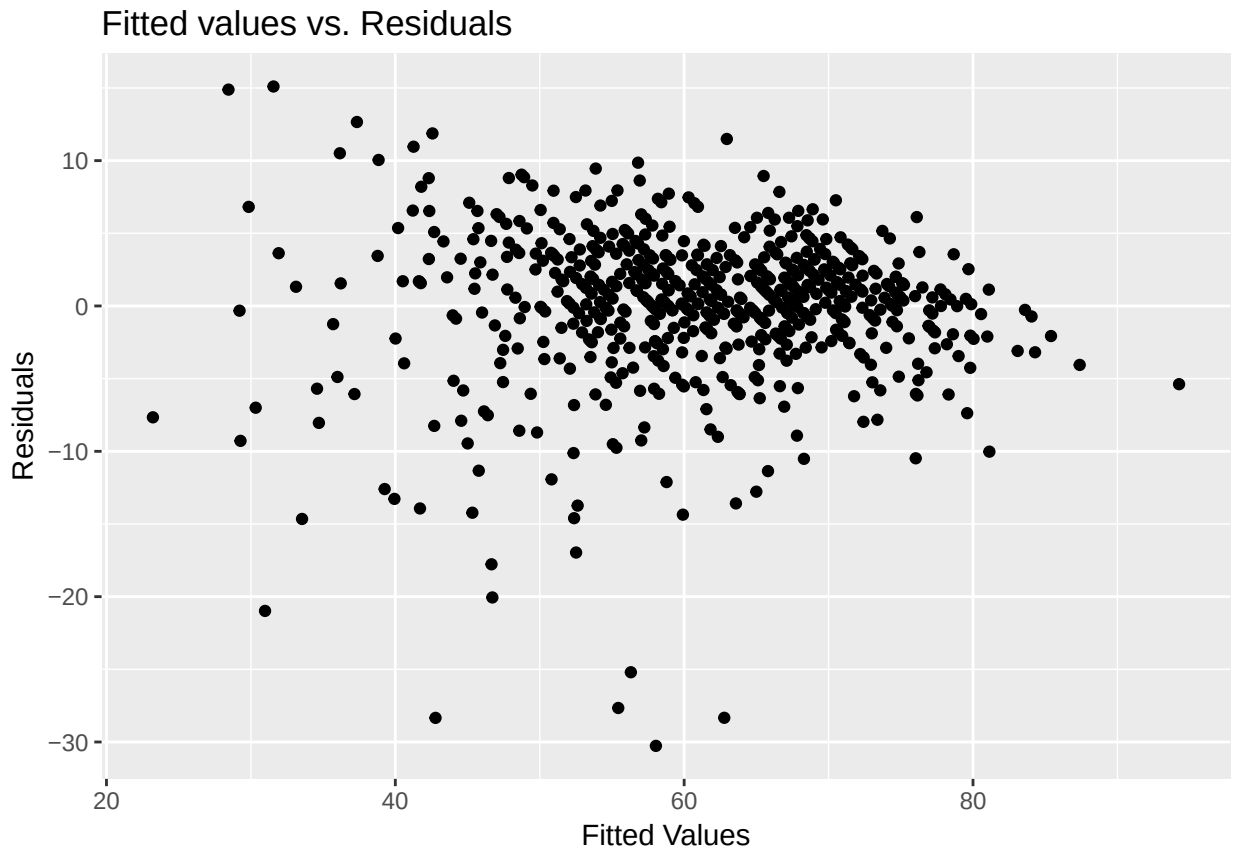
Let us do some final diagnostics for this model.

```
ggplot(data = fit, aes(x = .resid)) +
  geom_histogram() +
  xlab("Residuals") +
  ylab("Frequency") +
  labs(title = "Distribution of errors")
```

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




```
ggplot(data = fit, aes(x = .fitted, y = .resid)) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  labs(title = "Fitted values vs. Residuals")
```



We can interpret the coefficients of the linear model to find insight about associations between the predictors and the response. Let us begin with the interpretation of the coefficients for the genre. The baseline value is the genre "Action and Adventure". For example, horror movies are expected to be rated by 1.803 (unscaled) score points better than the baseline genre. Scaling back to the original scale, that translates to a better score of 1.16227, on average. The interpretation of the other coefficient for the factor variable mpaa rating is similar. The baseline level is "G". The coefficients for numerical variables such as runtime are interpreted as follows: For each additional minute runtime, the (unscaled) rating is expected to be 0.052 higher, on average. For the number of IMDB votes, the interpretation is similar. For the audience score, it is a bit more difficult. Since we use polynomial features, the increase is linear in each of the three powers of the audience score. For example, an increase of the third power of the audience score by 1 is associated with an increase of the (unscaled) IMDB rating by 0.0001633, on average. The total effect on an increase in the audience score is, however, dependent on the current level of that variable.

To finalize this section, we have to note that the normality assumption seems to be violated by the left-skew of the residuals. Also, we still have a problem with heteroscedasticity. This means that our prediction intervals will be incorrect. For low fitted values, they will be too narrow and for large fitted values they will be too wide. One could try and calculate approximate standard deviations for the errors based on the size of the fitted values in order to obtain a better understanding. * * *

Part 5: Prediction

In this part, we are going to predict the IMDB rating for a movie released in 2016 that is not included in the data set. For this, we will consider the movie "Arrival". Using the data gathered from the IMDB and RT websites (http://www.imdb.com/title/tt2543164/, https://www.rottentomatoes.com/m/arrival_2016), we can make the prediction (and transform the result back to the original scale):

```
arrival <- data.frame(genre = "Mystery & Suspense", runtime = 116, mpaa_rating = "PG-13", imdb_num_votes
unscaled_rating <- predict(fit, newdata = arrival, interval = "prediction")
scaled_rating <- 9/100 * unscaled_rating + 1
scaled_rating
```

```
##        fit      lwr      upr
## 1 7.589217 6.626645 8.551789
```

As we can see, the true IMDB rating (which is 8.0 as of 2017/12/23) lies well within the prediction interval of 6.6 to 8.6. Even the point estimate 7.59 is quite good, So the prediction here is pretty accurate. Note that the prediction interval is a little inflated by the heteroscedasticity.

The prediction interval takes into account both the uncertainty in the parameter estimation and the uncertainty of the random error term. If the model met all four assumptions of the ordinary least squares estimator (i.e. linearity in the predictors, uncorrelated error terms, normality of the errors, homoscedasticity), we would be 95% confident that the true value is inside this interval. Note that this is a slightly different interpretation than confidence intervals in inference, where we would expect 95% of confidence intervals to capture the true mean. Here we are also estimating the variance of the random error that follows a normal distribution under the OLS assumptions.

Part 6: Conclusion

The analysis showed that we would expect documentaries with long runtime and an mpaa rating of "G" that receive high audiences scores on Rotten Tomatoes and many votes on IMDB to have extraordinarily high IMDB ratings. This answers the original research question. Also, the model is easily interpretable.

However, there are a few shortcomings of the model. First and foremost, the OLS assumptions are not met completely resulting in erroneous standard errors for the predictors. This also leads to slightly incorrect prediction intervals. Nonetheless, the model yielded useful insight into the characteristics that are associated with movies that receive high ratings on IMDB.

To improve the model, several steps could be done. The apparent heteroscedasticity could in fact could have a different root cause, for example a non-linearity in the predictors or bias introduced by omitted variables. Using backwards feature selection with adjusted R^2 as feature elimination criterion might yield slightly better results.

In terms of raw predictive power, using preprocessing steps such as the singular value decomposition to ensure perpendicular predictors. Also, utilizing more powerful models such as Random Forests could yield more accurate predictions.

Another thing we found was that the largest residuals came from movies where the scores on RT and IMDB where extremely diverging. One could try and capture this effect in another predictor.