“Draw me like one of your French girls”

A report on (linear) modelling for IMDb score prediction

Owen Jones, Artem Yurlov, Sam Lewis, Dave Robjohns

Spring 2017

Abstract

In the following report we attempt to produce a mathematical model capable of using information about a film to predict its IMDb score.

This turns out to be a difficult thing to do; at least, it is difficult to make consistently accurate predictions, owing to the huge amount of variance in the data. As one might expect, film seems to be highly subjective.

After creating an initial model, we make various attempts to:

a. improve its accuracy, by numerically transforming the information we provide it with
b. simplify it, by only providing it with a subset of the available information and trying to achieve a similar level of performance

Satisfying both of these goals is not an easy task, given that making progress towards one goal generally leads to making sacrifices in the other. The eventual result is something of a compromise, where we have ensured that we are not overfitting our training data but maintain a reasonable degree of accuracy when making predictions.

In the final section we briefly explore film genres and some trends that become apparent in the data when films are separated by genre.

1. Data exploration

The data are provided in a data frame, `imdb`, which contains information about a subset of 3638 films from the IMDb database.

```r
load("imdb.rda")
str(imdb)
```

We are ultimately interested in the relationship between the IMDb score (recorded as `score`) and the rest of the information which is available.

```r
summary(imdb$score)
hist(imdb$score, xlim = c(0, 10), prob = TRUE, breaks = 30)
```

Looking at the distribution of scores we can see the mean rating of the films in our dataset is between 6 and 7; this is a consequence of our tendency to treat 6 or 7 as an “average” film rating, rather than 5 (which would arguably be more logical from a mathematician’s point of view!).

It will also be useful to see the relationship between the score and each of the predictors which might be used in our models.

The code used to produce the following plots can be found in Appendix A.

1.1 Replacing incorrect values

One seemingly erroneous point is immediately noticeable: the aspect value of 16, which lies well out to the right of the other points. We can easily find which film it belongs to.

```
imdb[imdb$aspect == 16, ]
```

Consulting the entry for this film in the online IMDb database, it turns out that the aspect should in fact be 16:9, so we shall amend this.

```
imdb$aspect[3539] <- 1.78
```

Let's have a look at another plot which might reveal a potentially interesting relationship among some predictors.

```
plot(imdb$budget, imdb$gross, pch = 20, main = "Gross vs Budget")
```

There is a smattering of outlying values in the plot. Further investigation reveals that one of these, the clear outlier in the bottom-right, is in fact another misentered value.

```
imdb[imdb$budget > 3.5e+08, ]
```

Again consulting the online database, it seems the budget for this film was actually \$85m, not \$390m as currently recorded. This is easily changed.

```
imdb$budget[959] <- 8.5e+07
```

1.2 Duplicated entries

There are also multiple titles which appear more than once in the dataset.

```
# Create a data frame of the duplicated entries
dups <- group_by(imdb, title) %>%
    summarise(count = n()) %>%
    filter(count > 1)
nrow(dups)
# So there are 93 titles appearing more than once...
# Empirically, it seems different values of uservotes is the main culprit

# See if uservotes is only difference between duplicates
setdiff(dups$title,
        imdb$title[select(imdb, -c(uservotes)) %>%
                       duplicated()])

# No, it isn't in two cases!

filter(imdb, title == "Brothers" |  title == "Cinderella")
# Brothers: uservotes, actor3 are different
# Cinderella: uservotes, userreviews are different
# In both cases, it won't matter which record we keep

# We'll remove duplicates by taking the record with largest uservotes for each
# title (i.e. the most recent record)
imdc <- group_by(imdb, title) %>%
    slice(which.max(uservotes)) %>%
    ungroup()
```

We now have a new data frame, `imdc`, which is identical to `imdb` but with each title now only appearing once. All the following analyses and models will be based on this tidied dataset.

1.3 Aside: exaggerating hating

If a film has a high number of user votes but a low score, it would suggest that the film is perhaps unfairly treated as a result of "viral" negative coverage. We can attempt to identify any such films by inspecting the Score vs User Votes plot and looking for points towards the lower-right corner.

In fact, the films with higher numbers of user votes all have high scores, so there is no clear indication of any "exaggerated dislike" for certain films. However we will see some evidence to the contrary later in Section 2.1.

2. Initial model

We are now in a position to begin training linear models on the data. Initially we will train a model on the full set of predictor variables.

```
fulldf <- select(imdc, title, score, year, duration, gross, budget,
                 criticreviews, uservotes, userreviews, aspect, rating,
                 country, color)

# Exclude title, which is a label rather than a predictor
fullmod <- lm(score ~ . - title, fulldf)

# Inspect the model
summary(fullmod)
```

Immediately we notice that the factor variables `country` and `rating` are each expanded into many dummy variables by `lm()`, and that many of these new dummy predictors appear to contribute very little. Moreover, the fairly poor adjusted R^2 score indicates that the model is currently not explaining much of the variance in the data.

Before continuing, it would be wise to check the diagnostic plots of the model.

The issues with our model become more obvious. The negative trend in the residuals-fitted plot shows that the model predicts high scores too generously. The spread of variance in residuals is also very uneven. This is largely a consequence of the concentration of film scores at approximately 6: there are a great deal more points there, so we are likely to see a higher variance. Nonetheless we may be able to reduce the severity of the issue.

The Q-Q plot of standardised residuals also reveals that the distribution of residuals is most certainly not normal. Considering the fact that normally-distributed residuals form important underlying assumption of the entire linear modelling process, this is something we must try to remedy. Additionally, the residuals-leverage plot reveals multiple points with unusually high leverage.

Working from this base model we can now begin to take steps to improve the model's performance.

2.1 Transforming numeric predictors

By its very design, a linear model will perform better the closer to linear the relationship between each numeric predictor and the response. However, it appears that the relationship between some of the predictors and the score is **not** linear; and therefore we may be able to improve the performance of the model by somehow transforming the data.

The decision of which transformations to apply is largely an empirical process. Given that the model is consistently overpredicting high scores, we try squaring the response variable to encourage the model to correct this.

We also apply square root transformations to some of the predictors. For count variables such as `uservotes` this choice can be explained, at least in part, by the fact that counts are often approximately Poisson distributed, and square-rooting is an effective method of improving the symmetry of this distribution. For non-count variables such as `duration`, square-rooting simply improves the fit of the resulting model.

The following plots show linear regression lines for the original and transformed predictors and response.

See Appendix B for the code used to generate these plots.

Now we can compare the performance of linear models fitted on the untransformed and transformed numerical predictors.

```r
# Train models on variations of (numeric) data:
#   s = score^2 as response
#   t = transformed predictors

df <- select(imdc, title, score, year, duration, gross, budget,
             criticreviews, uservotes, userreviews)

tdf <- select(imdc, title, score, year, duration, gross, budget,
              criticreviews, uservotes, userreviews) %>%
    transmute(title, score, year, durationsr = sqrt(duration),
              grosssr = sqrt(gross), budget,
              criticreviewssr = sqrt(criticreviews),
              uservotessr = sqrt(uservotes), userreviewssr = sqrt(userreviews))

sdf <- select(imdc, title, score, year, duration, gross, budget,
              criticreviews, uservotes, userreviews) %>%
    mutate(score = score^2) %>% rename(sqscore = score)

tsdf <- select(imdc, title, score, year, duration, gross, budget,
               criticreviews, uservotes, userreviews) %>%
    transmute(title, sqscore = (score^2), year, durationsr = sqrt(duration),
              grosssr = sqrt(gross), budget,
              criticreviewssr = sqrt(criticreviews),
              uservotessr = sqrt(uservotes), userreviewssr = sqrt(userreviews))

lmod <- lm(score ~ . - title, df)
tlmod <- lm(score ~ . - title, tdf)
slmod <- lm(sqscore ~ . - title, sdf)
tslmod <- lm(sqscore ~ . - title, tsdf)

# Compare RSS for the models
sum(residuals(lmod)^2)
sum((sqrt(sdf$sqscore) - sqrt(fitted(slmod)))^2)
sum(residuals(tlmod)^2)
sum((sqrt(tsdf$sqscore) - sqrt(fitted(tslmod)))^2)

# {squared score} ~ {transformed predictors} is best, as hoped!
```

We can also compare the diagnostic plots of the new model with those of the base model.

```r
par(mfrow = c(2, 2))
plot(tslmod)
```

The new model shows significant improvements. We still have a slight negative trend in the residuals-fitted plot but the effect is markedly less pronounced (the issue with spread of variance remains, but it is all but impossible to fix this for the reasons described earlier). The Q-Q plot shows that the model's residuals are now distributed much closer to normally.

The high-leverage points have also disappeared, although we will see shortly that this is due to the omission of factor predictors in this model.

There are two outliers, particularly evident in the Q-Q plot: the films appearing in rows 1415 and 3475 of `tsdf`.

```r
tsdf[c(1415, 3477), ]
```

Justin Bieber: Never Say Never makes an appearance, along with the documentary *Winged Migration* – at opposite ends of the spectrum in terms of score. In particular, according to our model the former should have a much higher score than it actually does. Given that the featured celebrity tends to divide opinion (one of this report's authors created an IMDb account with the sole intention of giving that particular film a score of 1), it is perhaps not surprising that our model is overly sympathetic in its predicted score.

Having said that, Mr Bieber does actually appear in another film in the dataset (*Zoolander 2*), and although the model still rates the film more highly than IMDb users do, the residual is much smaller than for *Never Say Never*.

```r
residuals(tslmod)[which(imdc$title == "Zoolander 2")]
```

So perhaps certain users (aforementioned author included) exaggerate their dislike for certain *films* after all, rather than disapproving of certain *individuals* appearing in those films.

Continuing with our bid to improve the model, let's re-fit with the two outlying points omitted and look at the summary.

```r
tslmod2 <- lm(sqscore ~ . - title, tsdf, subset = -c(1415, 3477))
summary(tslmod2)
```

This has a better adjusted R^2 than `tslmod`, suggesting it was sensible to remove the outlying points. It also has a better adjusted R^2 than `fullmod`, despite the greater number of predictors and therefore information `fullmod` was trained on. This is almost certainly due to the expansion of the factor variables in `fullmod`; we will address this issue in Section 3.

2.2 Interactions

Given the large number of predictors, there are an almighty number of possible interactions which could be added to the model. First let's check for strongly-correlated predictors. The upper triangle of the following figure is the correlation matrix of the numeric predictors; the lower half is scatterplots of each pair; and the diagonal shows the distribution of each predictor.

```r
library(psych)
pairs.panels(tsdf[c("year", "durationsr", "grosssr", "budget",
                    "criticreviewssr", "uservotessr", "userreviewssr")])
```

We will add the five strongest correlations as interactions in our model. Three of these are obvious linear relationships between user and critic votes and reviews. The others can be easily interpreted in the context of film: high-budget films are often well-advertised, leading to large gross takings at the box office and concurrently many votes from filmgoers.

2.3 Re-fitting the full model

We now re-fit `fullmod` by returning the factor variables to `tsdf`, adding the interaction predictors and removing the two outlying points we identified earlier.

```r
fulldf <- mutate(tsdf, color = imdc$color, aspect = imdc$aspect,
                 country = imdc$country, rating = imdc$rating)
fullmod <- lm(sqscore ~ . - title + uservotessr:userreviewssr +
                  criticreviewssr:uservotessr + criticreviewssr:userreviewssr +
                  grosssr:budget + grosssr:uservotessr,
              fulldf, subset = -c(1415, 3477))
summary(fullmod)
```

The adjusted R^2 score is still low, but is significantly better than the first time we fitted the model. Of course, the score a given film receives from a user is highly subjective; for such a difficult task, the adjusted R^2 is not terrible. In the following section we will aim to further improve and simplify the model.

3. Simplifying the initial model

Due to the inclusion of many-levelled factor predictors, the current model contains a very large number of predictors. Many of these are likely to be statistically insignificant.

Moreover, on closer inspection many levels contain only one data point: `country` is perhaps the most extreme example.

```r
# Check distribution of country
summary(imdc$country)
```

Hence many predictors only apply to a single point. This is virtually the textbook definition of overfitting, which will lead to poor results when the model is used to predict unseen scores. Certainly we must ensure that these one-point predictors at the very least are removed from any simplified models.

3.1 Selecting smaller models with AIC

Given that we currently have over 50 predictors in the model, a “top-down” `step()`-style approach is unlikely to reduce our model to a satisfactory size. Moreover, `step()` keeps staunchly to its numeric-assessment-based approach, and therefore does a poor job of recognising the overfitting endemic in our model and removing the offending predictors.

Therefore we adopt a somewhat more natural approach, founded on the principle of Occam’s Razor, whereby we will add predictors to an initially empty model. The `regsubsets()` function in the `leaps` package allows us to find the “best” model (according to Akaike’s An Information Criterion, or AIC) of each size. We find such models up to an arbitrary limit of 25 predictors - a generous limit given that we wish to find a “simple” model.

```r
# Set up a data frame with transformed predictors
imdd <- select(imdc, title, score, year, duration, gross, budget,
               criticreviews, uservotes, userreviews, color, aspect, country,
               rating) %>%
    transmute(title, sqscore = (score^2), year, durationsr = sqrt(duration),
              grosssr = sqrt(gross), budget,
              criticreviewssr = sqrt(criticreviews),
              uservotessr = sqrt(uservotes), userreviewssr = sqrt(userreviews),
              color, aspect, country, rating) %>%
    filter(!(title %in% c("Justin Bieber: Never Say Never",
                          "Winged Migration")))

# Let's find the best model of each size up to 25 predictors
library(leaps)
modselect <- regsubsets(sqscore ~ . - title + uservotessr:userreviewssr +
                            criticreviewssr:uservotessr +
                            criticreviewssr:userreviewssr +
                            grosssr:budget + grosssr:uservotessr,
                        imdd, nvmax = 25, really.big = TRUE)
```

We can now calculate the AIC of each of these models according to the formula

$$-2\{\text{maximised log-likelihood}\} + 2\{\text{number of parameters, including intercept}\} + (\text{constant})$$

which for linear models is equivalent to

$$n \log\left(\frac{RSS}{n}\right) + 2\{\text{number of parameters, including intercept}\} + (\text{constant})$$

We can subsequently plot these AIC scores against model size.

```r
n <- nrow(imdd)
ms_out <- summary(modselect)

plot(1:25, n*log(ms_out$rss/n) + 2*(2:26), type = "o", main = "AIC vs size",
     xlab = "Number of predictors (not including intercept)", ylab = "AIC")
```

Notice that over this range of sizes, adding predictors consistently reduces (improves) the AIC score of the model – `step()` would have faltered long before reaching this stage. Therefore this is not a simple case of choosing the model with the smallest AIC score and we shall have to be slightly more subjective in our selection of a smaller model.

We take the approach that two distinct "elbows" can be seen in the plot, at 3 and (less markedly) 9 predictors. After each of these points the rate of improvement of the AIC score of the model with the addition of each extra predictor becomes smaller. Again abiding by Occam's Razor we select these model sizes to study in more detail.

Firstly we consider the "reduced" model with 9 predictors.

```r
# Which predictors are used?
which(ms_out$which[9, ])

# We need to make a binary variable
imdd$ratingPG13 <- as.numeric(imdd$rating == "PG-13")

rmod <- lm(sqscore ~ year + durationsr + grosssr + budget + criticreviewssr +
               uservotessr + userreviewssr + grosssr:budget + ratingPG13, imdd)
summary(rmod)
```

Notice that this model performs almost as well as `fullmod` did earlier, according to adjusted R^2. Additionally the troublesome one-point predictors have all been eliminated. The only factor-level predictor to be included (`ratingPG13`) is also the least significant.

Let's also inspect the "tiny" model with only 3 predictors.

```r
which(ms_out$which[3, ])
tmod <- lm(sqscore ~ budget + uservotessr + durationsr, imdd)
summary(tmod)

# Compare AIC of smaller models against a model with intercept only
nullmod <- lm(sqscore ~ 1, imdd)

AIC(fullmod, rmod, tmod, nullmod)

# Common-sense check that AIC() is doing what we think it is: difference between
# hand-calculated values for tmod and rmod
3539*log(summary(msel)$rss[3]/3539) + 2*4 -
```

```r
    (3539*log(summary(msel)$rss[9]/3539) + 2*10)
# Difference between AIC() values
AIC(tmod) - AIC(rmod)
# All is well! Only difference is in positive constant in definition of AIC
```

Observe that `rmod` has an AIC score very close to `fullmod`, but uses approximately six times fewer predictors and avoids the overfitting issues we saw earlier. In addition `tmod` performs almost as well using only 3 predictors.

We *can* try to compare the models with an F-ratio test. The small p-value suggests that the predictors we drop to obtain `rmod` were in fact significant.

```r
anova(rmod, fullmod)
```

However because the F-test is dependent on the RSS of the models and because `fullmod` overfits the data, the result must be interpreted with caution.

The decision to use 9 predictors in `rmod` was, as mentioned earlier, somewhat empirical. As it turns out, the extra predictor in the best model with 10 predictors was `countryUK` – indicating that whether a film comes from the UK or not is the most useful country predictor. In this sense, then, the UK is indeed number one.

3.2 Manually "collapsing" factor predictors

We can take a different approach to reduce the number of levels in factor predictors which involves manually grouping similar levels - for example, grouping ratings into "kid-friendly" and "not kid-friendly". Details can be found in Appendix C, but seeing as we ultimately did not manage to make significant improvements to the models from Section 3.1 we do not include the results here.

4. Assessing the reduced model (`rmod`)

4.1 Interpretation of the model

The best model would ideally be one which is consistent with intuition. Let us describe how a film viewer might predict the quality of a film, and give some (deliberately extreme and subjective) reasoning to explain the intuition:

- Older films are better (according to widespread parental opinion)
- Films that made more profit are better (if they were bad no one would go to watch them)
- If a user decided to review a film he must think it's bad (because you don't review a film if it is widely acclaimed to be good, unless you are a critic)
- If there are more critics reviewing a film than users then it's good and vice versa (critics can give their professional opinion because, at least in theory, they know exactly what they are talking about)
- The larger a film's ($uservotes - userreviews$) the better it is (because you don't have time to leave a vote for a mediocre film, and for a bad film you leave a review to warn others)

Now let's put this into an equation with positive constants only and compare that equation to our model:

$$score \approx c_1\, year + c_2\, (gross - budget) + c_3\, criticreviews + c_4\, uservotes - c_5\, userreviews$$

Notice how the equation above is linear and unbounded, so films with better parameters will receive linearly better scores; actual score is in the interval $[1, 10]$ and definitely levels off at the top end. To adjust for the level-off we can predict the square of the score (as detailed earlier) and then convert it into normal score later: this also partially fixes unboundedness as it's "harder" for films to get higher scores.

$$score^2 \approx c_1\, year + c_2\, (gross - budget) + c_3\, criticreviews + c_4\, uservotes - c_5\, userreviews$$

Looking at `rmod`, we see that we have 9 statistically significant predictors which are precisely the ones mentioned above, but also duration, whether it is rated PG-13, and the interaction between gross and budget. We can try to explain these extra predictors intuitively too, and then after doing so we can predict the signs of each of the coefficients in the fitted model.

A problem with the current pseudo-model is that it's very susceptible to the "film-so-bad-it's-good" principle. To account for this we amend the 2nd point we made earlier to:

- films that have higher $(gross \times budget) - gross - budget$ are better (meaning only films that have both high gross and high budget receive good scores)

And we can add a point about rating:

- PG13-rated films score worse (since plotlines are more predictable because directors and actors are trying to appeal to a younger audience)

Additionally, an unashamedly non-statistical survey of the report's authors suggested that longer films are strongly preferred to shorter films.

So we now predict the model to look something like this (recall all constants c_i here are positive):

$$\begin{aligned} score^2 \approx c_1\, year + c_2\, (gross \times budget) + c_3\, criticreviews + c_4\, uservotes - c_5 userreviews \\ + c_6\, duration - c_7\, gross - c_8\, budget - c_9\, ratingPG13 \end{aligned}$$

The signs of coefficients of the linear model are as predicted – that is, as we had hoped, the predictors in our fitted model have the same effect on predictions as our intuition suggests.

4.2 Generalisation to unseen data

We unfortunately do not have the space in this report to adequately investigate whether the model performs well in general on unseen data. However we shall check a single case: Alfred Hitchcock's *Vertigo*, which has an IMDb score of 8.4.

```r
# Data obtained from IMDb/Wikipedia
vertigo <- data.frame(1958, sqrt(128), sqrt(3200000), 2479000, sqrt(201),
                      sqrt(267005), sqrt(706), 1)
names(vertigo) <- names(coef(rmod)[2:9])

# Actual score is 8.4
sqrt(predict(rmod, vertigo, interval = "prediction"))
sqrt(predict(tmod, vertigo, interval = "prediction"))
```

In this specific case, `rmod` is virtually spot-on with a prediction of 8.40, with a 95% prediction interval of (7.28, 9.39). The simplistic `tmod` makes a prediction of 7.72, with a 95% prediction interval of (6.38, 8.86), so `rmod` outperforms `tmod` in terms of both accuracy and certainty in its predicton. Although the accuracy is encouraging, we reiterate that much more test data would be needed to carry out a thorough assessment of the model's performance.

5. Further analysis: General genre-al *je ne sais quoi*

Often when selecting a film to watch, the genre is an important consideration. If we break down the films in the dataset according to genre then we may be able to reveal some information about the differences between genres.

We'll create violin plots to show the spread of data and will highlight a couple of points of interest about each plot.

See Appendix D for the code used to generate the plots in this section.

5.1 Score

There is an obvious trend here: educational films (documentaries, histories, biographies etc.) have higher scores on average. We suggest that the reason is that anything educational, of any format, is valued based on the information that it carries. In case of a film it is also designed to be entertaining or at least aesthetically pleasing, which ideally renders it “less boring” than a lesson and easier to consume than a book. Then the rating of such a film is not a rating of how good the actors are or how good is the plot but instead of how “not boring” it is; and in most cases, it’s not boring at all, and therefore scores highly on IMDb.

5.2 Budget

Note that (in this plot only) we have made the width of the violins uniform across all genres, so we can better see the trends within each genre rather than across genres.

Of particular note here are documentaries, which usually have very low budgets. Also notice the proportion of animations which are high-budget, which is relatively high compared to the other genres. This suggests processing power is more expensive than actors - and generalising, that computers are better than humans. We shall make no attempt to disprove the corollary that `rmod` is better than any given film critic.

5.3 Year

Here the recent boom in creation of documentaries is particularly evident - we have seen already that they are cheap to produce, and according to IMDb score they are well-received by viewers. We can also see a gradual but steady decline in the production of Westerns in recent years, perhaps due to directors being wary of controversy around the stereotypes they traditionally portray.

Appendix A: Plots of score vs predictors

```r
# Select relevant columns from imdb; take numeric columns and put into "long"
# data format, ready for plotting
plotdf <- select(imdb, title, score, year, duration, gross, budget, criticreviews,
             uservotes, userreviews, country, rating, color, aspect) %>%
    gather(-c(title, score, country, rating, color, aspect),
           key = "var", value = "value")

# Set up plot
p <- ggplot(plotdf, aes(x = value, y = score)) + geom_point(alpha = 0.07) +
    facet_wrap(~ var, scales = "free")

# Actually plot it
p + labs(title = "Plots of IMDb score against numerical predictors") +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"))

# Plot the non-numeric things
p1 <- qplot(country, score, data = imdb, alpha = I(0.1)) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = rel(0.8)))
p2 <- qplot(rating, score, data = imdb, alpha = I(0.1)) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = rel(0.8)))
p3 <- qplot(color, score, data = imdb, alpha = I(0.1))
p4 <- qplot(aspect, score, data = imdb, alpha = I(0.1))
grid.arrange(p1, p2, p3, p4, ncol = 2,
             top = textGrob("Plots of IMDb score against categorical predictors",
                            gp = gpar(fontface = "bold")))
```

Appendix B: Plots with linear regression lines

```r
#==== As-given ====

p + geom_smooth(method = "lm", col = "red") +
    labs(title = "Score vs Predictors") +
    theme(plot.title = element_text(hjust = 0.5, size = rel(1.5),
                                    face = "bold"))


#==== Linearised ====

# Try making relationships more linear
betterplotdf <- select(imdc, title, score, year, duration, gross, budget,
                       criticreviews, uservotes, userreviews, country, rating,
                       color, aspect) %>%
    transmute(title, sqscore = (score^2), year, durationsr = sqrt(duration),
              grosssr = sqrt(gross), budget,
              criticreviewssr = sqrt(criticreviews),
              uservotessr = sqrt(uservotes),
              userreviewssr = sqrt(userreviews),
              country, rating, color, aspect) %>%
    gather(-c(title, sqscore, country, rating, color, aspect),
           key = "var", value = "value")

bp <- ggplot(betterplotdf, aes(x = value, y = sqscore)) +
    geom_point(alpha = 0.07) + geom_smooth(method = "lm", col = "magenta") +
    facet_wrap(~ var, scales = "free") +
    labs(title = "Squared Score vs Linearised Predictors") +
    theme(plot.title = element_text(face = "bold", hjust = 0.5,
                                    size = rel(1.5)))

bp


#==== Aside: how did we decide on linearisation transformations? ====

# Mostly empirical! Looking at plots and trying to spot trends

# Example of predictor which is hard to linearise: duration. How should we
# transform it?

#   Base:
m1 <- lm((score^2) ~ duration - 1, imdc)
#   Square root:
m2 <- lm((score^2) ~ sqrt(duration) - 1, imdc)
#   Something fancier: square root of distance from most common duration:
m3 <- lm((score^2) ~ sqrt(abs(duration-90)) - 1, imdc)

# Compare RSS for these models
anova(m1, m2, m3)
# m2 has smallest RSS, so we'll square-root duration
```

Appendix C: Grouping similar factor levels

```
# Let's move on to our factor variables...

# aspect is given to us as a numeric, but really it should probably be a factor
# since it only takes certain values
summary(factor(imdc$aspect))
ggplot(imdc, aes(factor(aspect), score)) + geom_point(alpha = 0.2)

# A lot of these aspects only have 1 point... clearly a model trained on this
# factor will overfit our data!
# Let's see if we can remove this issue and simultaneously simplify our factor
# by 'collapsing' the factor i.e. collecting similar groups
imdc$standardaspect <- imdc$aspect %in% c(1.85, 2.35)

# Let's compare some models
amod <- lm(score ~ aspect - 1, imdc)
bmod <- lm(score ~ factor(aspect) - 1, imdc)
cmod <- lm(score ~ standardaspect - 1, imdc)
# Fit a constant model for comparison
nmod <- lm(score ~ 1, imdc)
AIC(amod, bmod, cmod, nmod)

# standardaspect is, perhaps surprisingly, almost as good as aspect - and it
# removes the overfitting problem


# Now let's do something similar for rating:
summary(imdc$rating)
ggplot(imdc, aes(rating, score)) + geom_point(alpha = 0.2)

# "Kid-friendly" films might get worse scores? ...
imdc$kidfriendly <- imdc$rating %in% c("G", "GP", "PG", "PG-13")

dmod <- lm(score ~ rating - 1, imdc)
emod <- lm(score ~ kidfriendly - 1, imdc)
AIC(dmod, emod, nmod)

# So the kidfriendly model has a marginally better AIC - it is much simpler too!
# Again this is likely to reduce the problem of overfitting, so this is good :)


# And now for country...
summary(imdc$country)
ggplot(imdc, aes(country, score)) + geom_point(alpha = 0.2) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))

# LOADS of countries with only 1 film. Surefire recipe for extreme overfitting!
# Perhaps it's significant if a country has or hasn't released many films
imdc$manyfilms <- NULL
countrycounts <- group_by(imdc, country) %>%
    summarise(manyfilms = (n() > 5))
```

```r
imdc <- right_join(imdc, countrycounts, by = "country")

fmod <- lm(score ~ country - 1, imdc)
gmod <- lm(score ~ manyfilms - 1, imdc)
AIC(fmod, gmod, nmod)

# That doesn't seem to be much good. It's worse than the null model!
# What about grouping countries by continent?
library(countrycode)
imdc$continent <- factor(countrycode(imdc$country, "country.name", "continent"))
summary(imdc$continent)

# We've managed to group up countries, so we shouldn't overfit too badly
hmod <- lm(score ~ continent - 1, imdc)
AIC(fmod, hmod, nmod)

# Much better than before. Still worse than the model using country, but we know
# that that model is overfitting, so this seems to be a good compromise!


# Our last factor variable is color. This is already a binary factor so we can
# happily leave it as it is!
```

Appendix D: Plots in Section 5

```
gens <- strsplit(imdc$genres, "[|]")

# summary(factor(unlist(gens)))

genscores <- data.frame("score" = imdc$score)
genbudgets <- data.frame("budget" = imdc$budget)
genyears <- data.frame("year" = imdc$year)

for (i in 1:nrow(imdc)) {
    for (j in 1:length(gens[[i]])) {
        genscores[i, j+1] <- gens[[i]][j]
        genbudgets[i, j+1] <- gens[[i]][j]
        genyears[i, j+1] <- gens[[i]][j]
    }
}

keyvalue1 <- gather(genscores, key = score, value = genre, na.rm = TRUE)
keyvalue2 <- gather(genbudgets, key = budget, value = genre, na.rm = TRUE)
keyvalue3 <- gather(genyears, key = year, value = genre, na.rm = TRUE)
```

```
ggplot(keyvalue1, aes(x = genre, y = score)) +
    geom_violin(scale = "area", fill = "#dddddd") +
    labs(x = "Genre", y = "Score") + theme_bw() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0))
```

```
ggplot(keyvalue2, aes(x = genre, y = budget)) +
    geom_violin(scale = "width", fill = "#dddddd") +
    labs(x = "Genre", y = "Budget") + theme_bw() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0))
```

```
ggplot(keyvalue3, aes(x = genre, y = year)) +
    geom_violin(scale = "area", fill = "#dddddd") +
    labs(x = "Genre", y = "Year") + theme_bw() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0))
```