STAT 230A Final Project Report

Andrea Padilla

11th May, 2023

Introduction

The Bechdel test was first introduced in a 1985 strip of Alison Bechdel's "Dykes to Watch Out For", a weekly comic originally published from 1983 to 2008. In the strip titled "The Rule" (see appendix), a character describes the criteria that she uses when deciding whether or not to watch a movie: (1) there are at least two women and (2) they have a conversation that is (3) not about men. For this character, a movie is not worth watching if it does not even explore the existence and humanity of people who are not men. While gender-based discrimination and violence can and do still occur in a movie that has passed the test, this character sees the test as a baseline for a movie she would be willing to consume, and thus support. This is a very important principle, especially for lesbians, who are centered in all of Bechdel's works. For Bechdel and many other lesbians, sexuality and identity are political and it is vital to decenter men in all aspects of life. Regarding the Bechdel test, NPR's Neda Ulaby states simply: "it articulates something often missing in popular culture. Not the number of women we see on screen, but the depths of their stories and the range of their concerns."

You can't always know whether a movie will pass the Bechdel test before seeing it. However, through a regression analysis, perhaps I can develop some criteria to predict whether a movie will pass or not. I'd also like to find similarities and differences in the movies that do and do not pass the test. Knowing this information would help myself and others decide whether or not to support a movie with our time and money. My results may point out what attributes are shared by movies that do not center women's experiences. We can also explore how the role of women in movies has changed over time and across genres.

Data Description

The data for this project was compiled by FiveThirtyEight and included in a TidyTuesday post, as well as an article on FiveThirtyEight titled "The Dollar-And-Cents Case Against Hollywood's Exclusion of Women". One simple but key finding from the data in the article from 538 is that in a sample of 1794 movies, "only half had at least one scene in which women talked to each other about something other than a man". The data cited by 538 comes from two sources: http://bechdeltest.com/ and http://www.the-numbers.com/. The data set used for regression contains extensive information on 1,794 movies from 1970-2013. Some variables of interest in this data set include budget, release date, revenue, and genre. There is a second set of data that exists and counts movies from 1888 to 2021, but it only includes title, release year, rating on the Bechdel test, and a few identification numbers. This second data set was only used to explore ratings over the larger time interval, which spans the entire history of cinema. If this was a long term project we could work on expanding the data set used for a larger sample size.

Two important transformations were done on the data to make it a viable source for regression. First, the outcome variable needed to be converted from words to a numeric rating. This was partially to make the regression possible and keep it clean but also because the original factor has 5 levels rather than 4. In addition to the ratings "no women", "no talk", "men", and "ok", which correspond with ratings 0-3, there is "dubious". A movie is rated as "dubious" if it is between 2 and 3, there are women talking, but the conversation they have that is not about men is brief and fleeting. I decided to turn this into a passing score (3) since the

Bechdel test is about a bare minimum amount of women's conversations and not necessarily an endorsement. Secondly, I performed a transformation to extract the genres. The original data set has one column for genre, with each movie potentially having multiple genres. I decided to make an indicator column for every genre in the data set. One limitation of this is that prediction would be difficult for a movie with a genre that isn't already a feature.

EDA

Looking at the distribution of scores over the years in Figure 1, prior to about 1920 most movies scored a zero. After 1920 we see all 4 ratings but can't quite see the proportions from this visual. Putting this information into a relative frequency table we can see the following:

```
## [1] "Relative frequencies of ratings from complete data, 1888-2021"
```

```
## 
##         0         1         2         3 
## 0.1011427 0.2194818 0.1013689 0.5780066
```

```
## [1] "Relative frequencies of ratings, 1970-2013"
```

```
## 
##         0         1         2         3 
## 0.0777027 0.2871622 0.1086712 0.5264640
```

In both tables we can see that overall 80% of scores are 1 or 3, with over half of all movies passing the test. This presents the biggest challenge of this problem: class imbalance. We see a smaller proportion of movies scoring 0, but the same goes for movies scoring a 3. This isn't a strong indication that scores have changed over time.

Before running any models I also wanted to compare scores between genres. I hypothesized that the romance genre would score better than the action genre. A dot plot likely wouldn't give much useful information due to the categorical outcome so we'll look at a table instead.

```
## [1] "Bechdel test scores for non-action movies (0) and action movies (1)"
```

```
## 
##       0   1
##   0  74  64
##   1 334 176
##   2 154  39
##   3 775 160
```

```
##    0    1 
## 1337  439
```

The right column tells us specifically how action movies perform on the Bechdel test, but the left column can help us compare to all non-action movies. The proportions of scores is not consistent across action movies and non-action movies. Below we can see that the majority, 66%, of romance movies pass the Bechdel test compared to 37% of action movies.

```
## [1] "Bechdel test scores for non-romance movies (0) and romance movies (1)"
```

```
## 
##       0   1
##   0 131   7
##   1 477  33
##   2 153  40
##   3 779 156
```

```
##    0    1
## 1540  236
```

Model Selection

The data was split into 80% training data and 20% test data in order to evaluate models on unseen data. Scaling was also performed on the year, budget, and revenue variables to get them to match the genre variables in magnitude. I didn't find any difference in accuracy when using scaled or original data but I chose to keep the scaled data.

The main models trained were multinomial logistic and proportional odds. I initially hypothesized that the proportional odds model would work best because there is an ordinal quality to the response variable. The two models handled the imbalance in classes very differently. To begin with, the multinomial logistic model only predicted scores of 1 and 3. This makes sense as they make up 80% of the data. By only predicting the two largest classes, a regular multinomial model does still retain 51% accuracy on test data. The AIC for this first model and many to follow was around 3,000, which we can compare to think about model fit later. The proportional odds model on the other hand, predicts all four classes nearly equally. Since the scores of 0 and 2 make up only 20% of the data, this gave accuracy of only about 20-25%, and it had a similar AIC.

After these initial models, I decided to adapt the weights to reflect the distribution of scores. If given more time, I would like to train the weights through cross-validation. Proportional weights only exacerbated the class imbalance for the multinomial model, and inverse weights overcompensated. As a medium between equal weights and inverse weights I selected weights that were inversely proportional to the square root of the frequency. I tried other options for weighting but didn't find anything that was significantly better, and didn't want to make it overly complicated. The weighting did not improve overall accuracy, it remains at around 53%, but the proportion of classifications when predicting is closer to the true distribution of scores. AIC for this model is around 300, only one-tenth of the original multinomial and ordinal models.

On the other hand, the proportional odds model was hardly responsive to changes in weights. Using the same weights that were best for the multinomial model produced predictions with only 25% accuracy. The AIC is even lower at only 230 but it essentially has no predictive power.

Discussion

Class imbalance is a tricky problem and I have not found a perfect solution to it. I found value and insight in both types of model. I appreciate the amount of information in the proportional odds model with the general summary function. I was able to get diagnostics on the multinomial model using the "dropterms" function from the MASS package but that information took time to find. Overall I prefer the multinomial logistic model because it is much more responsive to tuning, though model accuracy wasn't particularly better. The chi-square test results for the final multinomial model are quite high for all coefficients, but lower p-values only resulted with lower accuracy in this case. The multinomial model has a decay parameter that can be tuned through cross-validation but the results I got from that were not significant. I was optimistic that weighting would improve the proportional odds model but it was indifferent to weights.

In all models, the most significant variables consistently were year, budget, adventure, animation, action, crime, documentary, family, horror, music, romance, thriller, and war. Of those, the variables that are positively correlated with the Bechdel test score are romance, music, horror, family, and year. The positive and negative correlations are about as I expected. Horror was a bit of a surprise initially but it makes sense that women in horror movies have more pressing matters to discuss than their relationships with men. Though year is positively correlated with higher Bechdel scores, the coefficient is small which was anticipated after looking at the distribution of scores over time.

Conclusion

Imbalance in Bechdel test scores proved to be the main difficulty of this problem. While this issue was not solved, insights were gained from proportional odds and multinomial logistic models. The multinomial logistic model is most receptive to weights which leads me to believe it would perform better on other unseen data.

Additional Work & Appendix

EDA:

Reading data in and changing scores to 0-3:

```
tuesdata <- tidytuesdayR::tt_load('2021-03-09')
```

```
## --- Compiling #TidyTuesday Information for 2021-03-09 ----
```

```
## --- There are 2 files available ---
```

```
## --- Starting Download ---
```

```
##
##  Downloading file 1 of 2: `raw_bechdel.csv`
##  Downloading file 2 of 2: `movies.csv`
```

```
## --- Download complete ---
```

```
bechdel <- tuesdata$raw_bechdel
movieinfo <- tuesdata$movies

colnames(movieinfo)
```

```
##  [1] "year"          "imdb"          "title"         "test"
##  [5] "clean_test"    "binary"        "budget"        "domgross"
##  [9] "intgross"      "code"          "budget_2013"   "domgross_2013"
## [13] "intgross_2013" "period_code"   "decade_code"   "imdb_id"
## [17] "plot"          "rated"         "response"      "language"
## [21] "country"       "writer"        "metascore"     "imdb_rating"
## [25] "director"      "released"      "actors"        "genre"
## [29] "awards"        "runtime"       "type"          "poster"
## [33] "imdb_votes"    "error"
```

```
movieinfo$clean_test <- gsub("nowomen", 0, movieinfo$clean_test)
movieinfo$clean_test <- gsub("notalk", 1, movieinfo$clean_test)
movieinfo$clean_test <- gsub("men", 2, movieinfo$clean_test)
movieinfo$clean_test <- gsub("ok", 3, movieinfo$clean_test)
movieinfo$clean_test <- gsub("dubious", 3, movieinfo$clean_test)

movieinfo$clean_test <- as.numeric(movieinfo$clean_test)
movieinfo$domgross_2013 <- as.numeric(movieinfo$domgross_2013)
```

```
## Warning: NAs introduced by coercion
```

```
movieinfo$intgross_2013 <- as.numeric(movieinfo$intgross_2013)
```

```
## Warning: NAs introduced by coercion
```

```
movieinfo <- subset(movieinfo, select = c(year, title, clean_test, budget_2013, domgross_2013,
                                          intgross_2013, genre) )
```

I also removed some columns that are either repetitive, can't be used, or are beyond the scope of this project, like IMDB ID or actors.

Adding one column for each genre:

```
genres <- c("Action", "Adventure", "Animation", "Biography", "Comedy", "Crime",
            "Documentary", "Drama", "Family", "Fantasy", "History", "Horror", "Music",
            "Musical", "Mystery", "Romance", "Sci-Fi", "Sport", "Thriller",
            "Western", "War")

n <- ncol(movieinfo)
# make a column of zeroes for each genre
movieinfo[genres] <- c(0)


for (i in 1:length(genres)){
  # returns index of all movies that match the genre
  indices <- grep(genres[i], movieinfo$genre)
  colindex <- n + i

  # now set indicator to 1 for each movie that matches the genre
  movieinfo[indices, colindex] <- 1
}

movieinfo <- rename(movieinfo, Scifi = "Sci-Fi")
movieinfo = subset(movieinfo, select = -c(genre) )

# Dropping rows with NAs in budget and revenue columns as this will pose issues later
movieinfo <- drop_na(movieinfo, budget_2013:intgross_2013)
```

```
# Train-test split
trainindex <- createDataPartition(movieinfo$clean_test, p = 0.8, list = FALSE)
train_data <- movieinfo[trainindex, ]
test_data <- movieinfo[-trainindex, ]

# Scaling
scaled_train <- train_data
scaled_train[,c(1, 4:6)] <- scale(train_data[,c(1, 4:6)])

scaled_test <- test_data
scaled_test[,c(1, 4:6)] <- scale(test_data[,c(1, 4:6)])
```

Initial models

Initial multinomial model with all relevant variables

```
model1 <- multinom(train_data$clean_test ~ .-title, data = train_data, MaxNWts = 14395)
```

```
## # weights:  108 (78 variable)
## initial  value 1971.310582
## iter  10 value 1817.958285
## iter  20 value 1635.886096
## iter  30 value 1596.786996
## iter  40 value 1592.852043
## iter  50 value 1556.927470
```

```
## iter  60 value 1555.276395
## iter  70 value 1548.029597
## iter  70 value 1548.029597
## iter  80 value 1542.935605
## iter  90 value 1542.156267
## iter 100 value 1541.609177
## final  value 1541.609177
## stopped after 100 iterations
```

```r
model1
```

```
## Call:
## multinom(formula = train_data$clean_test ~ . - title, data = train_data,
##     MaxNWts = 14395)
##
## Coefficients:
##     (Intercept)         year   budget_2013 domgross_2013 intgross_2013
## 1  0.0022094072 0.0004756552  8.804751e-10  2.722304e-09 -7.969008e-10
## 2  0.0007960873 0.0002285524 -1.801381e-09  3.543629e-09 -1.571135e-09
## 3 -0.0049791589 0.0011067865 -5.259991e-09 -1.562555e-09  1.070178e-09
##         Action   Adventure    Animation    Biography      Comedy       Crime
## 1  0.07736251  0.03198692 -0.008326086  0.033381368  0.16478424  0.24593715
## 2 -0.18317963 -0.16413672 -0.033834865 -0.005482761  0.06215320  0.02226305
## 3 -0.37946278 -0.12550270 -0.141355862  0.033966351 -0.07084483 -0.38195605
##    Documentary       Drama      Family     Fantasy       History       Horror
## 1  0.006819137 0.02492108 -0.0767334 -0.14392053  0.0368070652 -0.246016895
## 2 -0.007879845 0.05011907 -0.0687427 -0.09081457  0.0002684277  0.002924835
## 3 -0.016561282 0.10632046  0.1778530  0.17322789 -0.0627283836  0.292130294
##           Music      Musical      Mystery    Romance        Scifi       Sport
## 1 -0.1252496563 -0.030799384  0.136818013 -0.2737706  0.060361458  0.05672162
## 2 -0.0003081347 -0.004671923  0.005124096  0.2364932 -0.056850405  0.02027066
## 3  0.1930223693  0.064297270 -0.052606666  0.1912293 -0.005880463 -0.07960332
##     Thriller      Western         War
## 1  0.2907161  0.077712206  0.02859097
## 2 -0.2007271 -0.006381652 -0.03308590
## 3 -0.1329629 -0.052012931 -0.09505356
##
## Residual Deviance: 3083.218
## AIC: 3239.218
```

```r
exp(coef(model1))
```

```
##   (Intercept)     year budget_2013 domgross_2013 intgross_2013    Action
## 1   1.0022118 1.000476           1             1             1 1.0804337
## 2   1.0007964 1.000229           1             1             1 0.8326186
## 3   0.9950332 1.001107           1             1             1 0.6842289
##   Adventure Animation Biography    Comedy     Crime Documentary     Drama
## 1 1.0325040 0.9917085 1.0339448 1.1791387 1.2788192   1.0068424 1.025234
## 2 0.8486260 0.9667311 0.9945322 1.0641254 1.0225127   0.9921511 1.051396
## 3 0.8820534 0.8681803 1.0345498 0.9316064 0.6825251   0.9835751 1.112178
##      Family   Fantasy   History   Horror     Music   Musical   Mystery
## 1 0.9261367 0.8659566 1.0374928 0.781909 0.8822766 0.9696701 1.1466195
## 2 0.9335669 0.9131870 1.0002685 1.002929 0.9996919 0.9953390 1.0051372
## 3 1.1946497 1.1891371 0.9391985 1.339278 1.2129099 1.0664094 0.9487531
##     Romance     Scifi     Sport  Thriller   Western       War
## 1 0.7605065 1.0622204 1.0583611 1.3373848 1.0808116 1.0290036
```

```
## 2 1.2667989 0.9447354 1.0204775 0.8181357 0.9936387 0.9674555
## 3 1.2107370 0.9941368 0.9234826 0.8754975 0.9493166 0.9093242
```

```r
# Checking predictions
predictions <- predict(model1, type = 'class')
table(predictions)/length(predictions)
```

```
## predictions
##         0         1         2         3
## 0.0000000 0.1835443 0.0000000 0.8164557
```

```r
table(train_data$clean_test)/nrow(train_data)
```

```
##
##          0          1          2          3
## 0.08016878 0.28481013 0.11040788 0.52461322
```

Initial proportional odds model

```r
model.ordinal <- ordinal::clm(factor(clean_test) ~ year + budget_2013 + domgross_2013 +intgross_2013 + A
summary(model.ordinal)
```

```
## formula:
## factor(clean_test) ~ year + budget_2013 + domgross_2013 + intgross_2013 + Action + Adventure + Anima
## data:    scaled_train
##
##  link  threshold nobs logLik   AIC     niter max.grad cond.H
##  logit flexible  1422 -1533.87 3123.74 6(1)  4.32e-12 1.9e+03
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## year           0.19853    0.05859   3.389 0.000703 ***
## budget_2013   -0.16603    0.07439  -2.232 0.025621 *
## domgross_2013 -0.12084    0.15735  -0.768 0.442512
## intgross_2013  0.16484    0.16983   0.971 0.331730
## Action        -0.61252    0.14887  -4.114 3.88e-05 ***
## Adventure     -0.18192    0.17048  -1.067 0.285924
## Animation     -0.65707    0.24123  -2.724 0.006453 **
## Biography      0.22757    0.27889   0.816 0.414500
## Comedy        -0.13878    0.12953  -1.071 0.283975
## Crime         -0.47641    0.15891  -2.998 0.002717 **
## Documentary   -1.41500    1.15579  -1.224 0.220849
## Drama          0.15531    0.12478   1.245 0.213245
## Family         0.52469    0.23965   2.189 0.028568 *
## Fantasy        0.13314    0.19296   0.690 0.490222
## History       -0.58414    0.34181  -1.709 0.087454 .
## Horror         0.71961    0.21779   3.304 0.000953 ***
## Music          1.37405    0.55410   2.480 0.013146 *
## Musical       -0.59843    0.76579  -0.781 0.434536
## Mystery       -0.20489    0.19929  -1.028 0.303882
## Romance        0.37319    0.17974   2.076 0.037864 *
## Scifi          0.01315    0.17945   0.073 0.941563
## Sport         -0.78299    0.37925  -2.065 0.038965 *
## Thriller      -0.36905    0.15405  -2.396 0.016593 *
## Western       -0.46676    0.55471  -0.841 0.400090
## War           -1.45074    0.44963  -3.227 0.001253 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##     Estimate Std. Error z value
## 0|1  -2.8597     0.1592 -17.962
## 1|2  -0.8117     0.1315  -6.174
## 2|3  -0.3023     0.1298  -2.330
```

```
ord_prdctns <- predict(model.ordinal, newdata = scaled_test, type = 'class')
table(test_data$clean_test, ord_prdctns[[1]])
```

```
##
##      0  1  2  3
##   0  7  4  6  7
##   1 27 25 28 25
##   2  5 13  9  9
##   3 43 48 51 47
```

```
mean(test_data$clean_test == ord_prdctns[[1]])
```

```
## [1] 0.2485876
```

Multinomial model with tuning for decay

```
trControl_mnl <- trainControl(method = "cv",
                              number = 10,
                              search = "grid",
                              classProbs = TRUE,
                              summaryFunction = multiClassSummary)

tuneGrid_mnl <- expand.grid(decay = seq(0, 1, by = 0.1))

model_mnl <- caret::train(make.names(clean_test) ~ year + budget_2013 + domgross_2013 +intgross_2013 + 
                         method = 'multinom',
                          maxit = 100,
                          trace = FALSE, # suppress iterations
                          tuneGrid = tuneGrid_mnl,
                          trControl = trControl_mnl,
                          na.action = na.exclude
                          )
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
model_mnl$results
```

```
##    decay  logLoss       AUC     prAUC  Accuracy      Kappa Mean_F1
## 1    0.0 1.116112 0.5935866 0.3155072 0.5274455 0.07397817     NaN
## 2    0.1 1.113524 0.6041309 0.3179839 0.5225456 0.05842034     NaN
## 3    0.2 1.113964 0.5932930 0.3118015 0.5408211 0.10391300     NaN
## 4    0.3 1.117837 0.5927153 0.3138010 0.5323605 0.07236813     NaN
## 5    0.4 1.114960 0.5906386 0.3143597 0.5331091 0.09056655     NaN
## 6    0.5 1.120554 0.5899812 0.3108778 0.5352070 0.07748984     NaN
## 7    0.6 1.119192 0.5844495 0.3079902 0.5330991 0.06919210     NaN
## 8    0.7 1.113642 0.5960491 0.3129600 0.5344779 0.08281832     NaN
```

```
## 9    0.8 1.113475 0.5922349 0.3115715 0.5408408 0.09964671     NaN
## 10   0.9 1.113561 0.5949658 0.3135730 0.5359110 0.08128136     NaN
## 11   1.0 1.110079 0.5961215 0.3153252 0.5457012 0.11740160     NaN
##    Mean_Sensitivity Mean_Specificity Mean_Pos_Pred_Value Mean_Neg_Pred_Value
## 1         0.2764490        0.7663255                 NaN           0.7863409
## 2         0.2690605        0.7634785                 NaN           0.7870269
## 3         0.2855598        0.7737486                 NaN           0.8072623
## 4         0.2747681        0.7660841                 NaN           0.7990514
## 5         0.2807868        0.7709359                 NaN           0.7951729
## 6         0.2761436        0.7674288                 NaN           0.8003401
## 7         0.2729593        0.7654698                 NaN           0.7977884
## 8         0.2778575        0.7690605                 NaN           0.7975317
## 9         0.2831136        0.7732832                 NaN           0.8069656
## 10        0.2767014        0.7688357                 NaN           0.7989793
## 11        0.2907447        0.7766964                 NaN           0.8085637
##    Mean_Precision Mean_Recall Mean_Detection_Rate Mean_Balanced_Accuracy
## 1             NaN   0.2764490           0.1318614              0.5213873
## 2             NaN   0.2690605           0.1306364              0.5162695
## 3             NaN   0.2855598           0.1352053              0.5296542
## 4             NaN   0.2747681           0.1330901              0.5204261
## 5             NaN   0.2807868           0.1332773              0.5258613
## 6             NaN   0.2761436           0.1338018              0.5217862
## 7             NaN   0.2729593           0.1332748              0.5192145
## 8             NaN   0.2778575           0.1336195              0.5234590
## 9             NaN   0.2831136           0.1352102              0.5281984
## 10            NaN   0.2767014           0.1339778              0.5227686
## 11            NaN   0.2907447           0.1364253              0.5337205
##     logLossSD      AUCSD    prAUCSD AccuracySD    KappaSD Mean_F1SD
## 1  0.02464981 0.03276650 0.02341346 0.03038088 0.06294684        NA
## 2  0.03192213 0.04582752 0.02742658 0.03227283 0.06601148        NA
## 3  0.02401314 0.04952191 0.02416842 0.01889917 0.04189400        NA
## 4  0.01813390 0.03727460 0.02185795 0.02733849 0.05474277        NA
## 5  0.02911870 0.04715438 0.02610435 0.02984243 0.06744014        NA
## 6  0.02089062 0.03951676 0.02289303 0.02245077 0.04866989        NA
## 7  0.02758840 0.04861467 0.02915454 0.03244566 0.06760521        NA
## 8  0.02894996 0.04135122 0.02408643 0.03295175 0.07672537        NA
## 9  0.03172309 0.05386663 0.02763876 0.03395123 0.07979014        NA
## 10 0.02762052 0.04439000 0.02654404 0.03174273 0.07493872        NA
## 11 0.02415774 0.03657986 0.02252566 0.01744258 0.04145420        NA
##    Mean_SensitivitySD Mean_SpecificitySD Mean_Pos_Pred_ValueSD
## 1          0.02251657        0.014089420                    NA
## 2          0.02277239        0.014900770                    NA
## 3          0.01512059        0.010113460                    NA
## 4          0.02011691        0.012256743                    NA
## 5          0.02381712        0.015583619                    NA
## 6          0.01658085        0.011661384                    NA
## 7          0.02452432        0.014635384                    NA
## 8          0.02724245        0.017832764                    NA
## 9          0.02844192        0.018536562                    NA
## 10         0.02584478        0.017370587                    NA
## 11         0.01514491        0.009548526                    NA
##    Mean_Neg_Pred_ValueSD Mean_PrecisionSD Mean_RecallSD Mean_Detection_RateSD
## 1             0.03222953               NA    0.02251657           0.007595220
## 2             0.03874699               NA    0.02277239           0.008068208
```

```
## 3             0.02137541                NA    0.01512059           0.004724791
## 4             0.03286511                NA    0.02011691           0.006834623
## 5             0.03073183                NA    0.02381712           0.007460607
## 6             0.03057131                NA    0.01658085           0.005612692
## 7             0.03605425                NA    0.02452432           0.008111415
## 8             0.03503354                NA    0.02724245           0.008237938
## 9             0.03440657                NA    0.02844192           0.008487808
## 10            0.03683861                NA    0.02584478           0.007935683
## 11            0.01711474                NA    0.01514491           0.004360645
##    Mean_Balanced_AccuracySD
## 1                0.01824510
## 2                0.01876932
## 3                0.01242123
## 4                0.01610349
## 5                0.01962571
## 6                0.01407466
## 7                0.01953585
## 8                0.02247339
## 9                0.02338308
## 10               0.02151991
## 11               0.01226611
```

The model isn't very adaptive to different values of decay, so I didn't choose any of them.

Multinomial model with proportional weights

```
inv_wts <-(table(train_data$clean_test))
case_wts <- matrix(data = 0, nrow = nrow(train_data), ncol = 1)

for (i in 1:nrow(train_data)){
  index <- train_data$clean_test[i]
  case_wts[i] <- inv_wts[index+1]
}

# Initial multinomial model with all relevant variables
model6 <- multinom(train_data$clean_test ~ year + budget_2013 + domgross_2013 +intgross_2013 + Action +
```

```
## # weights:  108 (78 variable)
## initial  value 1051068.976480
## iter  10 value 830370.416061
## iter  20 value 618918.894704
## iter  30 value 565626.400925
## iter  40 value 550793.118828
## iter  50 value 548555.706925
## iter  60 value 548153.350080
## iter  60 value 548153.350080
## iter  70 value 547753.592866
## iter  80 value 547664.930123
## iter  80 value 547664.929757
## final  value 547664.614229
## converged
```

```
model6
```

```
## Call:
```

```
## multinom(formula = train_data$clean_test ~ year + budget_2013 +
##     domgross_2013 + intgross_2013 + Action + Adventure + Animation +
##     Biography + Comedy + Crime + Documentary + Drama + Family +
##     Fantasy + History + Horror + Music + Musical + Mystery +
##     Romance + Scifi + Sport + Thriller + Western + War, data = train_data,
##     weights = case_wts, MaxNWts = 14395)
##
## Coefficients:
##     (Intercept)         year   budget_2013 domgross_2013 intgross_2013
## 1  2.865707e-05 0.0011865531  8.558339e-10  2.868148e-09 -8.948718e-10
## 2  2.710594e-06 0.0004353721 -4.171773e-09  3.578261e-09 -1.574289e-09
## 3 -3.864287e-05 0.0020137049 -6.576065e-09 -1.086465e-09  8.875771e-10
##          Action     Adventure     Animation     Biography        Comedy
## 1  0.0062511172  0.0012235340 -1.703921e-04  1.339145e-04  0.0006520644
## 2 -0.0004447592 -0.0007097594 -4.299479e-05 -5.534575e-05  0.0002214081
## 3 -0.0072860764 -0.0011333754 -4.842881e-04  2.136081e-04 -0.0003884763
##           Crime   Documentary         Drama        Family       Fantasy
## 1  0.0064539217  1.109630e-04 -0.0016664701 -0.0022299131 -0.0027196605
## 2  0.0004916088 -2.615951e-05 -0.0001981042 -0.0003043335 -0.0004865728
## 3 -0.0075011416 -1.304744e-04  0.0027470477  0.0026480879  0.0033718087
##         History        Horror         Music       Musical       Mystery
## 1  3.649144e-04 -3.405093e-03 -0.002096048 -7.695531e-04  0.0013343180
## 2  7.923165e-06 -3.958579e-06 -0.000104218 -5.755761e-05  0.0001364234
## 3 -4.235839e-04  3.428475e-03  0.002335026  8.619002e-04 -0.0014251987
##         Romance         Scifi         Sport      Thriller       Western
## 1 -0.0052734933  0.0013144985  0.0005648701  0.0056047654  8.731941e-04
## 2  0.0009473404 -0.0002598575  0.0001016169 -0.0005451464 -2.698287e-05
## 3  0.0046247334 -0.0011263513 -0.0006817309 -0.0054353724 -7.919660e-04
##             War
## 1  4.018911e-04
## 2 -6.954037e-05
## 3 -6.474466e-04
##
## Residual Deviance: 1095329
## AIC: 1095485
```

```
exp(coef(model6))
```

```
##   (Intercept)     year budget_2013 domgross_2013 intgross_2013    Action
## 1   1.0000287 1.001187           1             1             1 1.0062707
## 2   1.0000027 1.000435           1             1             1 0.9995553
## 3   0.9999614 1.002016           1             1             1 0.9927404
##   Adventure Animation Biography    Comedy     Crime Documentary     Drama
## 1 1.0012243 0.9998296 1.0001339 1.0006523 1.0064748   1.0001110 0.9983349
## 2 0.9992905 0.9999570 0.9999447 1.0002214 1.0004917   0.9999738 0.9998019
## 3 0.9988673 0.9995158 1.0002136 0.9996116 0.9925269   0.9998695 1.0027508
##      Family   Fantasy   History    Horror     Music   Musical   Mystery
## 1 0.9977726 0.9972840 1.0003650 0.9966007 0.9979061 0.9992307 1.0013352
## 2 0.9996957 0.9995135 1.0000079 0.9999960 0.9998958 0.9999424 1.0001364
## 3 1.0026516 1.0033775 0.9995765 1.0034344 1.0023378 1.0008623 0.9985758
##     Romance     Scifi     Sport  Thriller   Western       War
## 1 0.9947404 1.0013154 1.0005650 1.0056205 1.0008736 1.0004020
## 2 1.0009478 0.9997402 1.0001016 0.9994550 0.9999730 0.9999305
## 3 1.0046354 0.9988743 0.9993185 0.9945794 0.9992083 0.9993528
```

```r
# Checking predictions
predictions <- predict(model6, newdata = test_data, type = "class")
table(predictions)/length(predictions)
```

```
## predictions
##           0           1           2           3
## 0.000000000 0.005649718 0.000000000 0.994350282
```

```r
table(test_data$clean_test)/nrow(test_data)
```

```
##
##          0          1          2          3
## 0.06779661 0.29661017 0.10169492 0.53389831
```

```r
table(predictions, test_data$clean_test)
```

```
##
## predictions   0   1   2   3
##           0   0   0   0   0
##           1   0   1   1   0
##           2   0   0   0   0
##           3  24 104  35 189
```

```r
sum(diag(table(predictions, test_data$clean_test)))/nrow(test_data)
```

```
## [1] 0.5367232
```

Multinomial model with inverse weights

```r
inv_wts <-1/(table(train_data$clean_test))
case_wts <- matrix(data = 0, nrow = nrow(train_data), ncol = 1)

for (i in 1:nrow(train_data)){
  index <- train_data$clean_test[i]
  case_wts[i] <- inv_wts[index+1]
}

# Initial multinomial model with all relevant variables
model6 <- multinom(train_data$clean_test ~ year + budget_2013 + domgross_2013 +intgross_2013 + Action +
```

```
## # weights:  108 (78 variable)
## initial  value 5.545177
## iter  10 value 5.511899
## iter  20 value 5.466010
## iter  30 value 5.457204
## iter  40 value 5.454771
## iter  50 value 5.267024
## iter  60 value 5.263528
## iter  70 value 5.262338
## iter  80 value 5.262230
## final  value 5.262223
## converged
```

```r
model6
```

```
## Call:
## multinom(formula = train_data$clean_test ~ year + budget_2013 +
```

```
##     domgross_2013 + intgross_2013 + Action + Adventure + Animation +
##     Biography + Comedy + Crime + Documentary + Drama + Family +
##     Fantasy + History + Horror + Music + Musical + Mystery +
##     Romance + Scifi + Sport + Thriller + Western + War, data = train_data,
##     weights = case_wts, MaxNWts = 14395)
##
## Coefficients:
##     (Intercept)          year   budget_2013 domgross_2013 intgross_2013
## 1  1.048804e-04 -1.596316e-04  1.285223e-09  2.593517e-09 -7.362330e-10
## 2  1.140771e-05  5.013608e-05  3.396879e-10  3.373111e-09 -1.521798e-09
## 3 -3.916791e-03  1.666827e-04 -4.535825e-09 -1.714448e-09  1.051878e-09
##        Action   Adventure   Animation  Biography     Comedy       Crime
## 1  0.02360401 -0.03762878 -0.09254785 0.03979383 0.12204905  0.25984131
## 2 -0.56529948 -0.40998458 -0.15337896 0.02245948 0.20708830 -0.04933192
## 3 -0.55378225 -0.17406726 -0.16012668 0.07159227 0.05792659 -0.41910785
##     Documentary        Drama      Family    Fantasy      History      Horror
## 1 -3.973282e-05 -0.003708886 -0.02869950 -0.1473516 -0.002751339 -0.15455669
## 2 -2.298493e-02  0.196242380 -0.09708878 -0.1707374 -0.011202072  0.04194481
## 3 -1.738650e-02  0.241475259  0.14925091  0.1330598 -0.031327904  0.19567430
##         Music     Musical      Mystery    Romance       Scifi       Sport
## 1 -0.06509643 -0.01089163  0.169940606 -0.2881076  0.12189165  0.02672658
## 2  0.05545954  0.01658666  0.038994065  0.4898231 -0.11602423  0.02231292
## 3  0.14977130  0.05591436 -0.008171931  0.2047210 -0.04484316 -0.04491700
##     Thriller      Western         War
## 1  0.4035651  0.065458905 -0.02608542
## 2 -0.3936320 -0.002964635 -0.11547488
## 3 -0.1482359 -0.016271102 -0.08265845
##
## Residual Deviance: 10.52445
## AIC: 166.5244
```

```r
exp(coef(model6))
```

```
##   (Intercept)      year budget_2013 domgross_2013 intgross_2013    Action
## 1   1.0001049 0.9998404           1             1             1 1.0238848
## 2   1.0000114 1.0000501           1             1             1 0.5681900
## 3   0.9960909 1.0001667           1             1             1 0.5747718
##   Adventure Animation Biography   Comedy     Crime Documentary     Drama
## 1 0.9630704 0.9116056  1.040596 1.129810 1.2967243   0.9999603 0.996298
## 2 0.6636605 0.8578046  1.022714 1.230091 0.9518651   0.9772772 1.216822
## 3 0.8402404 0.8520358  1.074217 1.059637 0.6576333   0.9827638 1.273126
##      Family   Fantasy   History    Horror     Music   Musical   Mystery
## 1 0.9717084 0.8629905 0.9972524 0.8567949 0.9369771 0.9891675 1.1852345
## 2 0.9074754 0.8430429 0.9888604 1.0428369 1.0570262 1.0167250 1.0397643
## 3 1.1609642 1.1423183 0.9691577 1.2161307 1.1615686 1.0575071 0.9918614
##     Romance     Scifi     Sport  Thriller   Western       War
## 1 0.7496809 1.1296317 1.0270869 1.4971527 1.0676489 0.9742519
## 2 1.6320275 0.8904536 1.0225637 0.6746023 0.9970398 0.8909429
## 3 1.2271827 0.9561474 0.9560768 0.8622277 0.9838606 0.9206656
```

```r
# Checking predictions
predictions <- predict(model6, newdata = test_data, type = 'class')
table(predictions)/length(predictions)
```

```
## predictions
```

```
##         0         1         2         3
## 0.1327684 0.2655367 0.2570621 0.3446328
```

```r
table(test_data$clean_test)/nrow(test_data)
```

```
##
##          0          1          2          3
## 0.06779661 0.29661017 0.10169492 0.53389831
```

```r
table(predictions, test_data$clean_test)
```

```
##
## predictions  0  1  2  3
##           0  3 21  4 19
##           1  9 38 12 35
##           2  4 21 10 56
##           3  8 25 10 79
```

```r
sum(diag(table(predictions, test_data$clean_test)))/nrow(test_data)
```

```
## [1] 0.3672316
```

Ordinal model with scaled data and proportional weights

```r
inv_wts <-table(train_data$clean_test)
case_wts <- matrix(data = 0, nrow = nrow(train_data), ncol = 1)

for (i in 1:nrow(train_data)){
  index <- train_data$clean_test[i]
  case_wts[i] <- inv_wts[index+1]
}

ordinal_scaled <- ordinal::clm(factor(clean_test) ~ year + budget_2013 + domgross_2013 +intgross_2013 +

summary(ordinal_scaled)
```

```
## formula:
## factor(clean_test) ~ year + budget_2013 + domgross_2013 + intgross_2013 + Action + Adventure + Anima
## data:    scaled_train
##
##  link  threshold nobs   logLik     AIC        niter max.grad cond.H
##  logit flexible  758186 -522665.80 1045387.59 8(2)  9.30e-10 4.7e+03
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## year            0.215511   0.003099  69.549  < 2e-16 ***
## budget_2013   -0.252617   0.003842 -65.744  < 2e-16 ***
## domgross_2013 -0.173804   0.008271 -21.013  < 2e-16 ***
## intgross_2013  0.189393   0.008743  21.662  < 2e-16 ***
## Action        -0.440957   0.007581 -58.164  < 2e-16 ***
## Adventure     -0.160712   0.008697 -18.478  < 2e-16 ***
## Animation     -0.315662   0.012204 -25.865  < 2e-16 ***
## Biography      0.003971   0.014378   0.276    0.782
## Comedy        -0.273611   0.006733 -40.638  < 2e-16 ***
## Crime         -0.582103   0.008133 -71.571  < 2e-16 ***
## Documentary   -1.043095   0.058155 -17.936  < 2e-16 ***
```

```
## Drama          0.153544   0.006515  23.568  < 2e-16 ***
## Family         0.495175   0.012263  40.379  < 2e-16 ***
## Fantasy        0.255089   0.010122  25.200  < 2e-16 ***
## History       -0.530796   0.017917 -29.626  < 2e-16 ***
## Horror         0.893930   0.011984  74.597  < 2e-16 ***
## Music          1.710932   0.035230  48.564  < 2e-16 ***
## Musical       -0.872819   0.045074 -19.364  < 2e-16 ***
## Mystery       -0.365642   0.010229 -35.745  < 2e-16 ***
## Romance        0.490087   0.009991  49.052  < 2e-16 ***
## Scifi         -0.054664   0.009171  -5.961 2.51e-09 ***
## Sport         -0.786162   0.019362 -40.603  < 2e-16 ***
## Thriller      -0.447383   0.007760 -57.650  < 2e-16 ***
## Western       -1.185192   0.028971 -40.909  < 2e-16 ***
## War           -1.060039   0.023546 -45.019  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##      Estimate Std. Error z value
## 0|1 -4.432822   0.011032  -401.8
## 1|2 -1.426817   0.006879  -207.4
## 2|3 -1.234319   0.006817  -181.1
```

```
ordinalpredict <- predict(ordinal_scaled, newdata = scaled_test, type = 'class')
table(ordinalpredict)
```

```
## fit
##  0  1  2  3
## 89 78 98 89
```

```
table(scaled_test$clean_test, ordinalpredict$fit)
```

```
##
##      0  1  2  3
##   0  7  6  5  6
##   1 25 23 28 29
##   2  9  9 12  6
##   3 48 40 53 48
```

```
sum(diag(table(scaled_test$clean_test, ordinalpredict$fit)))/nrow(test_data)
```

```
## [1] 0.2542373
```

Ordinal model with scaled data and inverse weights

```
inv_wts <-1/table(train_data$clean_test)
case_wts <- matrix(data = 0, nrow = nrow(train_data), ncol = 1)

for (i in 1:nrow(train_data)){
  index <- train_data$clean_test[i]
  case_wts[i] <- inv_wts[index+1]
}
scaled_train <- train_data
scaled_train[,c(1, 4:6)] <- scale(train_data[,c(1, 4:6)])

scaled_test <- test_data
```

```r
scaled_test[,c(1, 4:6)] <- scale(test_data[,c(1, 4:6)])


ordinal_scaled <- ordinal::clm(factor(clean_test) ~ year + budget_2013 + domgross_2013 +intgross_2013 +
summary(ordinal_scaled)
```

```
## formula:
## factor(clean_test) ~ year + budget_2013 + domgross_2013 + intgross_2013 + Action + Adventure + Anima
## data:    scaled_train
##
##  link  threshold nobs logLik AIC   niter max.grad cond.H
##  logit flexible  4    -5.22  66.43 4(0)  9.73e-12 1.2e+03
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## year           0.17713    1.00931   0.176    0.861
## budget_2013   -0.03336    1.28118  -0.026    0.979
## domgross_2013  0.05719    2.79659   0.020    0.984
## intgross_2013  0.02928    3.05363   0.010    0.992
## Action        -0.87045    2.57366  -0.338    0.735
## Adventure     -0.23774    2.95349  -0.080    0.936
## Animation     -1.12746    4.24759  -0.265    0.791
## Biography      0.55852    5.22480   0.107    0.915
## Comedy         0.09436    2.28522   0.041    0.967
## Crime         -0.35067    2.76512  -0.127    0.899
## Documentary   -1.84445   19.90974  -0.093    0.926
## Drama          0.15944    2.20036   0.072    0.942
## Family         0.38125    4.41943   0.086    0.931
## Fantasy       -0.17033    3.39446  -0.050    0.960
## History       -0.49556    5.85742  -0.085    0.933
## Horror         0.48996    3.81919   0.128    0.898
## Music          1.17556    8.69404   0.135    0.892
## Musical       -0.47038   12.87382  -0.037    0.971
## Mystery        0.07144    3.57328   0.020    0.984
## Romance        0.37089    3.07762   0.121    0.904
## Scifi          0.08999    3.16017   0.028    0.977
## Sport         -0.78214    6.59470  -0.119    0.906
## Thriller      -0.32906    2.79300  -0.118    0.906
## Western        0.37525   10.07357   0.037    0.970
## War           -1.73993    7.51502  -0.232    0.817
##
## Threshold coefficients:
##     Estimate Std. Error z value
## 0|1  -1.5794     2.4867  -0.635
## 1|2  -0.3182     2.3354  -0.136
## 2|3   0.9077     2.3786   0.382
```

```r
ordinalpredict <- predict(ordinal_scaled, newdata = scaled_test, type = 'class')
table(ordinalpredict)
```

```
## fit
##  0  1  2  3
## 79 95 81 99
```

```
table(scaled_test$clean_test, ordinalpredict$fit)
```

```
##
##      0  1  2  3
##   0  6  8  4  6
##   1 28 29 22 26
##   2  6  8  9 13
##   3 39 50 46 54
```

```
sum(diag(table(scaled_test$clean_test, ordinalpredict$fit)))/nrow(test_data)
```

```
## [1] 0.2768362
```

Ordinal model with scaled data and inverse square root weights

```
inv_wts <-(table(train_data$clean_test))^{1/2}
case_wts <- matrix(data = 0, nrow = nrow(train_data), ncol = 1)

for (i in 1:nrow(train_data)){
  index <- train_data$clean_test[i]
  case_wts[i] <- inv_wts[index+1]
}
scaled_train <- train_data
scaled_train[,c(1, 4:6)] <- scale(train_data[,c(1, 4:6)])

scaled_test <- test_data
scaled_test[,c(1, 4:6)] <- scale(test_data[,c(1, 4:6)])


ordinal_scaled <- ordinal::clm(factor(clean_test) ~ year + budget_2013 + domgross_2013 +intgross_2013 +
summary(ordinal_scaled)
```

```
## formula:
## factor(clean_test) ~ year + budget_2013 + domgross_2013 + intgross_2013 + Action + Adventure + Anima
## data:    scaled_train
##
##  link  threshold nobs     logLik    AIC      niter max.grad cond.H
##  logit flexible  31710.36 -27729.09 55514.19 8(1)  2.11e-12 3.0e+03
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## year            0.20692    0.01362  15.195  < 2e-16 ***
## budget_2013    -0.21806    0.01711 -12.747  < 2e-16 ***
## domgross_2013 -0.15788    0.03640  -4.337 1.44e-05 ***
## intgross_2013  0.18495    0.03888   4.757 1.97e-06 ***
## Action        -0.50623    0.03398 -14.896  < 2e-16 ***
## Adventure     -0.16822    0.03890  -4.324 1.53e-05 ***
## Animation     -0.45172    0.05474  -8.252  < 2e-16 ***
## Biography      0.10074    0.06368   1.582    0.114
## Comedy        -0.21867    0.02979  -7.340 2.13e-13 ***
## Crime         -0.53770    0.03637 -14.783  < 2e-16 ***
## Documentary   -1.17243    0.26464  -4.430 9.41e-06 ***
## Drama          0.15284    0.02876   5.314 1.07e-07 ***
## Family         0.51557    0.05445   9.468  < 2e-16 ***
```

```
## Fantasy        0.21221    0.04453   4.766 1.88e-06 ***
## History       -0.56227    0.07919  -7.100 1.24e-12 ***
## Horror         0.81501    0.05139  15.858  < 2e-16 ***
## Music          1.53983    0.14047  10.962  < 2e-16 ***
## Musical       -0.73152    0.18646  -3.923 8.74e-05 ***
## Mystery       -0.29914    0.04557  -6.565 5.22e-11 ***
## Romance        0.42804    0.04264  10.039  < 2e-16 ***
## Scifi         -0.02796    0.04098  -0.682    0.495
## Sport         -0.78338    0.08669  -9.037  < 2e-16 ***
## Thriller      -0.40950    0.03489 -11.737  < 2e-16 ***
## Western       -0.86968    0.12808  -6.790 1.12e-11 ***
## War           -1.24411    0.10556 -11.786  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##     Estimate Std. Error z value
## 0|1 -3.61910    0.04116  -87.92
## 1|2 -1.11224    0.03027  -36.74
## 2|3 -0.79667    0.02991  -26.64
```

```r
ordinalpredict <- predict(ordinal_scaled, newdata = scaled_test, type = 'class')
table(ordinalpredict)
```

```
## fit
##  0  1  2  3
## 89 85 89 91
```

```r
table(scaled_test$clean_test, ordinalpredict$fit)
```

```
##
##      0  1  2  3
##   0  6  9  2  7
##   1 25 24 28 28
##   2  9  8 11  8
##   3 49 44 48 48
```

```r
sum(diag(table(scaled_test$clean_test, ordinalpredict$fit)))/nrow(test_data)
```

```
## [1] 0.2514124
```

Final model

```r
inv_wts <-1/(table(train_data$clean_test))^{1/2}
case_wts <- matrix(data = 0, nrow = nrow(train_data), ncol = 1)

for (i in 1:nrow(train_data)){
  index <- train_data$clean_test[i]
  case_wts[i] <- inv_wts[index+1]
}

#  multinomial model with all relevant variables and scaled data
scaled_ml <- multinom(scaled_train$clean_test ~ year + budget_2013 + domgross_2013 +intgross_2013 + Acti
```

```
## # weights:  108 (78 variable)
## initial  value 97.934286
```

```
## iter  10 value 86.695535
## iter  20 value 85.465718
## iter  30 value 85.251004
## iter  40 value 85.175469
## iter  50 value 85.156038
## iter  60 value 85.146823
## iter  70 value 85.143510
## iter  80 value 85.143367
## final  value 85.143354
## converged
```

```r
summary(scaled_ml)
```

```
## Call:
## multinom(formula = scaled_train$clean_test ~ year + budget_2013 +
##     domgross_2013 + intgross_2013 + Action + Adventure + Animation +
##     Biography + Comedy + Crime + Documentary + Drama + Family +
##     Fantasy + History + Horror + Music + Musical + Mystery +
##     Romance + Scifi + Sport + Thriller + Western + War, data = scaled_train,
##     weights = case_wts, MaxNWts = 14395)
##
## Coefficients:
##   (Intercept)      year budget_2013 domgross_2013 intgross_2013     Action
## 1   0.7916363 0.1324337   0.2509694     0.4302021   -0.19768446 -0.7872753
## 2   0.7591270 0.1431899   0.2088453     0.5806211   -0.46992409 -1.1596156
## 3   1.3773896 0.3378360  -0.0839754     0.1955408    0.03809185 -1.1648561
##    Adventure Animation Biography    Comedy      Crime Documentary      Drama
## 1 -0.1163783 -1.235677 1.1539136 0.6065861 -0.1148766   -1.038658 0.15532067
## 2 -0.5937438 -1.188472 0.9021298 0.2505706 -0.1641123  -16.200303 0.08525486
## 3 -0.2371807 -1.535022 1.1715869 0.2518090 -0.7204294   -2.012841 0.26912654
##        Family    Fantasy    History     Horror     Music  Musical   Mystery
## 1  0.15467553 -0.4653269 -0.3863037 -0.5546455  9.490231 5.534006 0.7215239
## 2 -0.07468928 -0.7697188 -0.4175966  0.2931095 11.125467 3.828947 0.4438785
## 3  0.63029696 -0.1643649 -0.8086729  0.4501092 11.617795 4.037116 0.2779549
##      Romance      Scifi      Sport   Thriller  Western       War
## 1 -0.1958492 0.31252248 -0.2941697  0.3692840 11.88822 -1.227371
## 2  0.8858142 0.07687752 -0.4802638 -0.6866627 11.00307 -2.613584
## 3  0.4921243 0.19706996 -1.1713601 -0.2275769 10.42191 -2.165811
##
## Std. Errors:
##   (Intercept)      year budget_2013 domgross_2013 intgross_2013   Action
## 1    1.105070 0.4320597   0.5320092      1.294054      1.298181 1.007954
## 2    1.121583 0.4818164   0.6277162      1.474608      1.548843 1.214401
## 3    1.029715 0.4313047   0.5568030      1.335871      1.336403 1.022977
##   Adventure Animation Biography   Comedy    Crime  Documentary     Drama
## 1  1.160115  1.563724  2.951809 1.039909 1.151107 5.838451e+00 0.9989746
## 2  1.429656  1.956212  3.103966 1.119668 1.316378 1.603583e-06 1.0839724
## 3  1.166882  1.557162  2.894554 1.005371 1.202072 6.486075e+00 0.9653121
##     Family  Fantasy  History   Horror    Music  Musical  Mystery  Romance
## 1 1.855467 1.358907 2.433750 1.834457 3.121735 3.729986 1.770757 1.800895
## 2 2.185054 1.617300 2.785488 1.829823 2.099456 2.866628 1.934394 1.725479
## 3 1.734813 1.270842 2.450437 1.600240 1.852761 2.443719 1.762263 1.632252
##      Scifi    Sport Thriller  Western      War
## 1 1.312865 2.935346 1.127497 2.128323 2.390795
## 2 1.556735 3.125341 1.384064 2.802835 3.998044
```

```
## 3 1.327460 3.049965 1.146724 2.874357 2.773284
##
## Residual Deviance: 170.2867
## AIC: 326.2867
```

```
exp(coef(scaled_ml))
```

```
##   (Intercept)     year budget_2013 domgross_2013 intgross_2013    Action
## 1    2.207005 1.141603   1.2852707      1.537568     0.8206288 0.4550831
## 2    2.136410 1.153949   1.2322544      1.787148     0.6250497 0.3136067
## 3    3.964539 1.401911   0.9194539      1.215968     1.0388266 0.3119676
##   Adventure Animation Biography   Comedy     Crime  Documentary    Drama
## 1 0.8901384 0.2906380  3.170577 1.834159 0.8914762 3.539293e-01 1.168032
## 2 0.5522558 0.3046866  2.464847 1.284758 0.8486467 9.210811e-08 1.088995
## 3 0.7888487 0.2154508  3.227110 1.286350 0.4865433 1.336085e-01 1.308821
##      Family   Fantasy   History    Horror     Music   Musical  Mystery
## 1 1.1672792 0.6279298 0.6795641 0.5742758  13229.85 253.15605 2.057566
## 2 0.9280318 0.4631433 0.6586279 1.3405896  67877.95  46.01406 1.558741
## 3 1.8781682 0.8484324 0.4454488 1.5684834 111056.55  56.66270 1.320427
##     Romance    Scifi     Sport  Thriller   Western        War
## 1 0.8221362 1.366869 0.7451501 1.4466984 145541.97 0.29306196
## 2 2.4249579 1.079910 0.6186202 0.5032528  60058.53 0.07327149
## 3 1.6357874 1.217829 0.3099451 0.7964612  33587.56 0.11465690
```

```
# Checking predictions
predictions <- predict(scaled_ml, newdata = scaled_test)
table(predictions)/length(predictions)
```

```
## predictions
##           0           1           2           3
## 0.064971751 0.316384181 0.008474576 0.610169492
```

```
table(test_data$clean_test)/nrow(test_data)
```

```
##
##          0          1          2          3
## 0.06779661 0.29661017 0.10169492 0.53389831
```

```
table(predictions, test_data$clean_test)
```

```
##
## predictions   0   1   2   3
##           0   3  12   0   8
##           1  12  43  14  43
##           2   0   1   1   1
##           3   9  49  21 137
```

```
sum(diag(table(predictions, test_data$clean_test)))/nrow(test_data)
```

```
## [1] 0.519774
```

```
dropterm(scaled_ml, trace = FALSE, test = "Chisq")
```

```
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.756205
## iter  20 value 85.759443
## iter  30 value 85.597739
```

```
## iter  40 value 85.535163
## iter  50 value 85.519577
## iter  60 value 85.511557
## iter  70 value 85.509750
## iter  80 value 85.509671
## final  value 85.509668
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.778292
## iter  20 value 85.746953
## iter  30 value 85.584511
## iter  40 value 85.519327
## iter  50 value 85.503443
## iter  60 value 85.495018
## iter  70 value 85.493001
## iter  80 value 85.492921
## final  value 85.492918
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.860014
## iter  20 value 85.538852
## iter  30 value 85.361927
## iter  40 value 85.284809
## iter  50 value 85.265746
## iter  60 value 85.256760
## iter  70 value 85.253493
## iter  80 value 85.253365
## final  value 85.253358
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.680507
## iter  20 value 85.649967
## iter  30 value 85.341827
## iter  40 value 85.264161
## iter  50 value 85.244498
## iter  60 value 85.235358
## iter  70 value 85.231911
## iter  80 value 85.231751
## final  value 85.231740
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 87.243424
## iter  20 value 86.296813
## iter  30 value 85.997127
## iter  40 value 85.914924
## iter  50 value 85.896132
## iter  60 value 85.886452
## iter  70 value 85.882700
## iter  80 value 85.882519
## final  value 85.882508
```

```
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.924125
## iter  20 value 85.581737
## iter  30 value 85.347709
## iter  40 value 85.274406
## iter  50 value 85.255159
## iter  60 value 85.246376
## iter  70 value 85.243274
## iter  80 value 85.243133
## final  value 85.243125
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.971582
## iter  20 value 85.980495
## iter  30 value 85.774708
## iter  40 value 85.692074
## iter  50 value 85.673961
## iter  60 value 85.664683
## iter  70 value 85.661349
## iter  80 value 85.661242
## final  value 85.661235
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.834571
## iter  20 value 85.558554
## iter  30 value 85.352349
## iter  40 value 85.285663
## iter  50 value 85.265687
## iter  60 value 85.258875
## iter  70 value 85.256692
## iter  80 value 85.256593
## final  value 85.256583
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.983495
## iter  20 value 85.636560
## iter  30 value 85.440802
## iter  40 value 85.375047
## iter  50 value 85.358612
## iter  60 value 85.349744
## iter  70 value 85.347546
## iter  80 value 85.347452
## final  value 85.347444
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 87.093393
## iter  20 value 85.796319
## iter  30 value 85.540608
```

```
## iter  40 value 85.461396
## iter  50 value 85.441996
## iter  60 value 85.433435
## iter  70 value 85.430273
## iter  80 value 85.430086
## final  value 85.430071
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.701910
## iter  20 value 85.492687
## iter  30 value 85.308829
## iter  40 value 85.259731
## iter  50 value 85.237029
## iter  60 value 85.233532
## iter  70 value 85.233399
## iter  80 value 85.233367
## final  value 85.233366
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.772754
## iter  20 value 85.458028
## iter  30 value 85.287665
## iter  40 value 85.221744
## iter  50 value 85.205004
## iter  60 value 85.196906
## iter  70 value 85.194946
## iter  80 value 85.194870
## final  value 85.194864
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.738508
## iter  20 value 85.612786
## iter  30 value 85.390601
## iter  40 value 85.314082
## iter  50 value 85.294647
## iter  60 value 85.285247
## iter  70 value 85.281948
## iter  80 value 85.281792
## final  value 85.281783
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.576621
## iter  20 value 85.648155
## iter  30 value 85.415448
## iter  40 value 85.336689
## iter  50 value 85.317430
## iter  60 value 85.308554
## iter  70 value 85.305457
## iter  80 value 85.305312
## final  value 85.305303
```

```
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.770302
## iter  20 value 85.525607
## iter  30 value 85.307925
## iter  40 value 85.232087
## iter  50 value 85.213094
## iter  60 value 85.204173
## iter  70 value 85.200988
## iter  80 value 85.200871
## final  value 85.200867
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 87.123390
## iter  20 value 85.807903
## iter  30 value 85.572945
## iter  40 value 85.494025
## iter  50 value 85.474625
## iter  60 value 85.465488
## iter  70 value 85.462141
## iter  80 value 85.461958
## final  value 85.461942
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.893062
## iter  20 value 85.724626
## iter  30 value 85.538524
## iter  40 value 85.456768
## iter  50 value 85.439233
## iter  60 value 85.426720
## iter  70 value 85.425727
## iter  80 value 85.425674
## final  value 85.425669
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.695302
## iter  20 value 85.510236
## iter  30 value 85.305818
## iter  40 value 85.218880
## iter  50 value 85.202463
## iter  60 value 85.191324
## iter  70 value 85.190483
## iter  80 value 85.190433
## final  value 85.190429
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 87.001716
## iter  20 value 85.585242
## iter  30 value 85.367918
```

```
## iter  40 value 85.294015
## iter  50 value 85.274802
## iter  60 value 85.265531
## iter  70 value 85.262201
## iter  80 value 85.262056
## final  value 85.262046
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 87.303025
## iter  20 value 85.888046
## iter  30 value 85.662655
## iter  40 value 85.586652
## iter  50 value 85.568093
## iter  60 value 85.558790
## iter  70 value 85.555658
## iter  80 value 85.555502
## final  value 85.555490
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.966571
## iter  20 value 85.498115
## iter  30 value 85.283408
## iter  40 value 85.208379
## iter  50 value 85.190181
## iter  60 value 85.181377
## iter  70 value 85.178250
## iter  80 value 85.178158
## final  value 85.178150
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.780246
## iter  20 value 85.558991
## iter  30 value 85.358117
## iter  40 value 85.280422
## iter  50 value 85.260660
## iter  60 value 85.252791
## iter  70 value 85.250008
## iter  80 value 85.249874
## final  value 85.249860
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 87.098267
## iter  20 value 85.923209
## iter  30 value 85.709594
## iter  40 value 85.637671
## iter  50 value 85.619365
## iter  60 value 85.610507
## iter  70 value 85.607934
## iter  80 value 85.607828
## final  value 85.607819
```

```
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.763712
## iter  20 value 85.597897
## iter  30 value 85.408304
## iter  40 value 85.356210
## iter  50 value 85.343815
## iter  60 value 85.337865
## iter  70 value 85.335378
## iter  80 value 85.335229
## final  value 85.335220
## converged
## # weights:  104 (75 variable)
## initial  value 97.934286
## iter  10 value 86.834021
## iter  20 value 85.886411
## iter  30 value 85.682079
## iter  40 value 85.612983
## iter  50 value 85.594268
## iter  60 value 85.585959
## iter  70 value 85.583117
## iter  80 value 85.582965
## final  value 85.582955
## converged
```

```
## Single term deletions
##
## Model:
## scaled_train$clean_test ~ year + budget_2013 + domgross_2013 +
##     intgross_2013 + Action + Adventure + Animation + Biography +
##     Comedy + Crime + Documentary + Drama + Family + Fantasy +
##     History + Horror + Music + Musical + Mystery + Romance +
##     Scifi + Sport + Thriller + Western + War
##               Df    AIC     LRT Pr(Chi)
## <none>           326.29
## year           3 321.02 0.73263  0.8655
## budget_2013    3 320.99 0.69913  0.8734
## domgross_2013  3 320.51 0.22001  0.9743
## intgross_2013  3 320.46 0.17677  0.9812
## Action         3 321.77 1.47831  0.6873
## Adventure      3 320.49 0.19954  0.9777
## Animation      3 321.32 1.03576  0.7926
## Biography      3 320.51 0.22646  0.9732
## Comedy         3 320.69 0.40818  0.9385
## Crime          3 320.86 0.57343  0.9025
## Documentary    3 320.47 0.18002  0.9807
## Drama          3 320.39 0.10302  0.9915
## Family         3 320.56 0.27686  0.9643
## Fantasy        3 320.61 0.32390  0.9555
## History        3 320.40 0.11502  0.9900
## Horror         3 320.92 0.63718  0.8879
## Music          3 320.85 0.56463  0.9045
## Musical        3 320.38 0.09415  0.9925
```

```
## Mystery        3 320.52 0.23738  0.9713
## Romance        3 321.11 0.82427  0.8437
## Scifi          3 320.36 0.06959  0.9952
## Sport          3 320.50 0.21301  0.9755
## Thriller       3 321.22 0.92893  0.8184
## Western        3 320.67 0.38373  0.9436
## War            3 321.17 0.87920  0.8304
```

"The Rule"

References

- Tidy Tuesday: https://github.com/rfordatascience/tidytuesday/blob/master/data/2021/2021-03-09/readme.md
- "The Dollar-And-Cents Case Against Hollywood's Exclusion of Women", Five Thirty Eight, https://fivethirtyeight.com/features/the-dollar-and-cents-case-against-hollywoods-exclusion-of-women/
- "The 'Bechdel Rule,' Defining Pop-Culture Character", NPR, https://www.npr.org/templates/story/story.php?storyId=94202522