Bayesian modeling and prediction for movies

Setup

Load packages

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

load("movies.Rdata")
```

Part 1: Data

The data set is comprised of 651 randomly sampled movies produced and released before 2016.

Two important points must be made about the data under examination before beginning:

- This is an observational study, not an experiment, and it may therefore prove difficult to determine casuality. This does not mean that any predictors identified will not be meaningful, but we should be aware that any variables of interest were not intentionally isolated and randomized.

- We should keep in mind that any findings may not be generalizable to the movie-going population as a whole. These data are drawn specifically from Rotten Tomatoes and IMDB, and the variable we're predicting is the audience score on Rotten Tomatoes. It should not be assumed that the people who take the time to rate films on Rotten Tomatoes form a representative random sample of the general population of interest. Therefore, any references to a film's "predicted score" or other similar statements should be interpreted as its ***predicted score for the Rotten Tomatoes audience***, and ***not*** the predicted score for all movie goers. Great care - and additional analysis - should be undertaken before any results presented here are used to inform broader strategy.

Part 2: Data manipulation

Several new features were generated for analysis from the underlying data:

- **feature_film:** Indicator variable; records whether or not the film was a feature. Levels: {"yes", "no"}

- **drama:** Indicator variable; records whether or not the film was a drama. Levels: {"yes", "no"}

- **mpaa_rating_R:** Indicator variable; records whether or not the film was rated R. Levels: {"yes", "no"}

- **oscar_season:** Indicator variable; records whether or not the film was released during Oscar Season (Oct, Nov, or Dec). Levels: {“yes”, “no”}
- **summer_season:** Indicator variable; records whether or not the film was released during the summer season (May, Jun, Jul, or Aug). Levels: {“yes”, “no”}

```
movies <- mutate(movies, feature_film = ifelse(title_type == "Feature Film", "yes", "n
o"))
movies$feature_film <- as.factor(movies$feature_film)

movies <- mutate(movies, drama = ifelse(genre == "Drama", "yes", "no"))
movies$drama <- as.factor(movies$drama)

movies <- mutate(movies, mpaa_rating_R = ifelse(mpaa_rating == "R", "yes", "no"))
movies$mpaa_rating_R <- as.factor(movies$mpaa_rating_R)

movies <- mutate(movies, oscar_season = ifelse(thtr_rel_month %in% c(10,11,12), "yes",
"no"))
movies$oscar_season <- as.factor(movies$oscar_season)

movies <- mutate(movies, summer_season = ifelse(thtr_rel_month %in% c(5,6,7,8), "yes",
"no"))
movies$summer_season <- as.factor(movies$summer_season)
```

We will also filter out the columns that will not be used in our analysis, and set aside the last two movies for use in the “Prediction” section:

```
  movies <- movies[,c("title", "audience_score", "feature_film", "drama", "runtime", "m
paa_rating_R", "thtr_rel_year", "oscar_season", "summer_season", "imdb_rating", "imdb_n
um_votes", "critics_score", "best_pic_nom", "best_pic_win", "best_actor_win", "best_act
ress_win", "best_dir_win", "top200_box")]
  test_movies <- movies[650:651,]
  movies <- movies[1:649,]
```

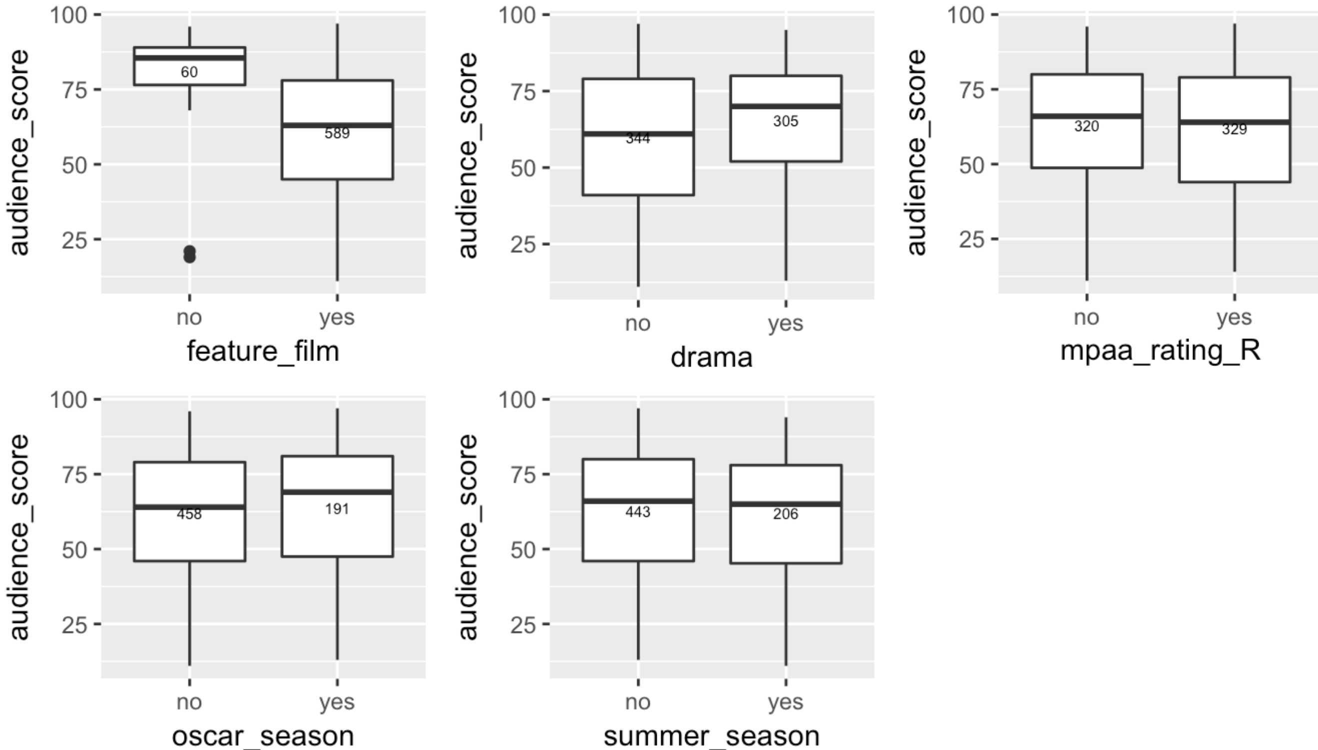
There is one piece of missing data here - the runtime for the film *The End of America*, which is 73 minutes. That error has been fixed:

```
  movies[ which(movies$title == "The End of America"), "runtime"] = 73
```

Part 3: Exploratory data analysis

1. Exploration of New Features

A set of box plots provide a quick summary of the new features created in section 2. The number of observations in each category is printed on each box plot.



It is immediately clear that feature_film has the largest impact on the audience_score of a film - and also that it is the most unevenly distributed, with “yes” having nearly 10 times the number of observations as “no”.

With the exception of the two outliers, the all the observed values for the non-feature films lie above the median of the feature film values. These outliers are almost certainly not random at this sample size, as shown below:

```
  non_feature_without_outliers <- movies[ which(movies$feature_film == "no" & movies$au
dience_score > 50),]
  non_feature_mean <- mean(non_feature_without_outliers$audience_score)
  non_feature_sd <- sd(non_feature_without_outliers$audience_score)
  probability_of_outlier <- 2*pnorm(25, non_feature_mean, non_feature_sd)^60
```

The probability of observing a single outlier that extreme by chance alone is **0**, which argues in favor of feature_film being a reliable predictor.

2. Analysis of Variable Correlation

It is also worth examining the degree of correlation between the original variables and new features, so that we can consider removing highly correlated variables while modeling.

```
  cor_matrix <- cor(as.data.frame(lapply(movies[,!colnames(movies) %in% c("audience_sco
re", "title")], as.integer)))
  cor_matrix <- cor_matrix[apply(cor_matrix, MARGIN = 1, function(x) any(x > .4 & x <
1)), ]
  cor_matrix[, apply(cor_matrix, MARGIN = 2, function(x) any(x > .4 & x < 1))]
```

```
##               imdb_rating critics_score best_pic_nom best_pic_win
## imdb_rating     1.0000000     0.7480915    0.2142347    0.1429456
## critics_score   0.7480915     1.0000000    0.1959177    0.1233197
## best_pic_nom    0.2142347     0.1959177    1.0000000    0.4750179
## best_pic_win    0.1429456     0.1233197    0.4750179    1.0000000
```

Here we can see that **best_pic_nom and best_pic_win** are correlated (which makes sense, given that only a few movies are nominated each year). More importantly, we see that **critics_score and imdb_rating** are highly correlated (~.75).

Part 4: Modeling

We will start by building all possible models starting with a uniform prior and using the Bayes Information Criterion as our metric.

```
audience_models <- bas.lm(audience_score ~ . -title, data = movies,
                   prior = "BIC",
                   modelprior = uniform())
  audience_models
```

```
##
## Call:
## bas.lm(formula = audience_score ~ . - title, data = movies, prior = "BIC",     model
prior = uniform())
##
##
##  Marginal Posterior Inclusion Probabilities:
##           Intercept      feature_filmyes             dramayes
##             1.00000              0.06109              0.04328
##             runtime     mpaa_rating_Ryes        thtr_rel_year
##             0.41989              0.18617              0.10497
##     oscar_seasonyes     summer_seasonyes          imdb_rating
##             0.08203              0.08043              1.00000
##      imdb_num_votes        critics_score      best_pic_nomyes
##             0.05439              0.87119              0.12376
##     best_pic_winyes    best_actor_winyes  best_actress_winyes
##             0.03982              0.15022              0.14609
##     best_dir_winyes        top200_boxyes
##             0.06950              0.07081
```

The result is unsurprising, and not particularly informative - the most included predictors by far are imdb_rating and critics_score. The coefficient is positive, and the correlation is made clearer by the size difference between them (given one, the second provides little new information):

```
##                 2.5  %    97.5  %        beta
## imdb_rating   13.81911 16.6808256 15.06832420
## critics_score  0.00000  0.1030213  0.06045053
```

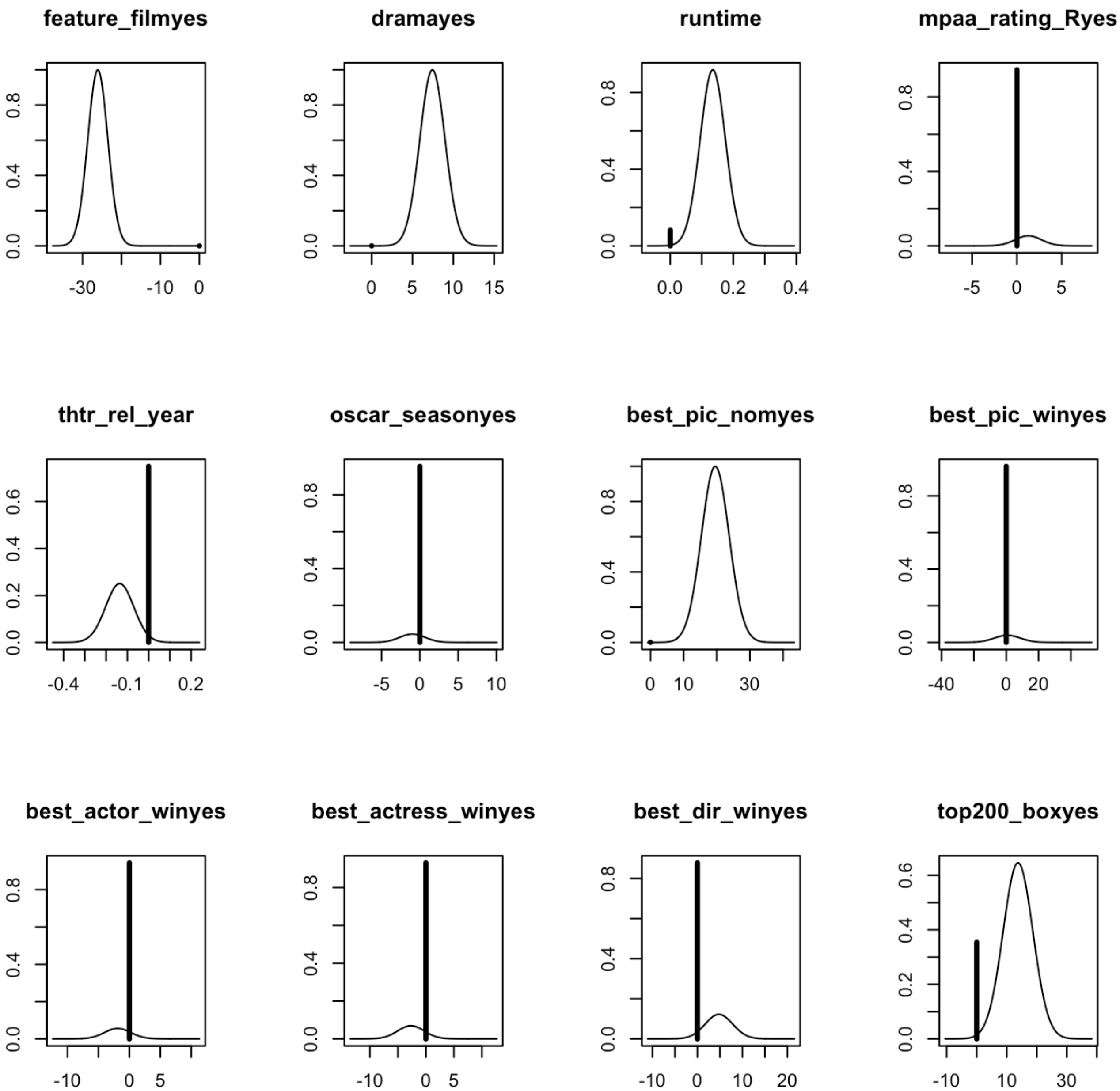
While it is useful from a generalizability perspective to know that the audience score is generally in agreement with the scores from IMDB and critics, it isn't very useful in terms of the business logic - that is to say, knowing that a film's rating can be predicted by how other people have rated it doesn't help us answer "what attributes make a movie popular."

Therefore, we will build the models again, this time removing critics_score, imdb_rating, and imdb_num_votes as predictors:

```
  no_other_scores_models <- bas.lm(audience_score ~ . -title -critics_score -imdb_ratin
g -imdb_num_votes, data = movies,
                 prior = "BIC",
                 modelprior = uniform())
  no_other_scores_models
```

```
##
## Call:
## bas.lm(formula = audience_score ~ . - title - critics_score -     imdb_rating - imdb
_num_votes, data = movies, prior = "BIC",     modelprior = uniform())
##
##
##  Marginal Posterior Inclusion Probabilities:
##           Intercept      feature_filmyes            dramayes
##             1.00000              1.00000             0.99978
##             runtime     mpaa_rating_Ryes        thtr_rel_year
##             0.91812              0.05416             0.25030
##     oscar_seasonyes     summer_seasonyes      best_pic_nomyes
##             0.04530              0.06393             0.99948
##     best_pic_winyes    best_actor_winyes  best_actress_winyes
##             0.03856              0.05594             0.06961
##     best_dir_winyes        top200_boxyes
##             0.12225              0.64563
```

Here we start to see some interesting results. As shown below, the best predictors (and the only ones with credible intervals that don't include zero) are feature_film, drama, best_pic_nom, runtime and (just barely) top200_box.


feature_filmyes
dramayes
runtime
mpaa_rating_Ryes
thtr_rel_year
oscar_seasonyes
best_pic_nomyes
best_pic_winyes
best_actor_winyes
best_actress_winyes
best_dir_winyes
top200_boxyes
0.0
0.4
0.8
0.2
0.6
-30
-10
0
5
10
15
0.0
0.2
0.4
-5
-0.4
-0.1
0.2
30
-40
20
-10

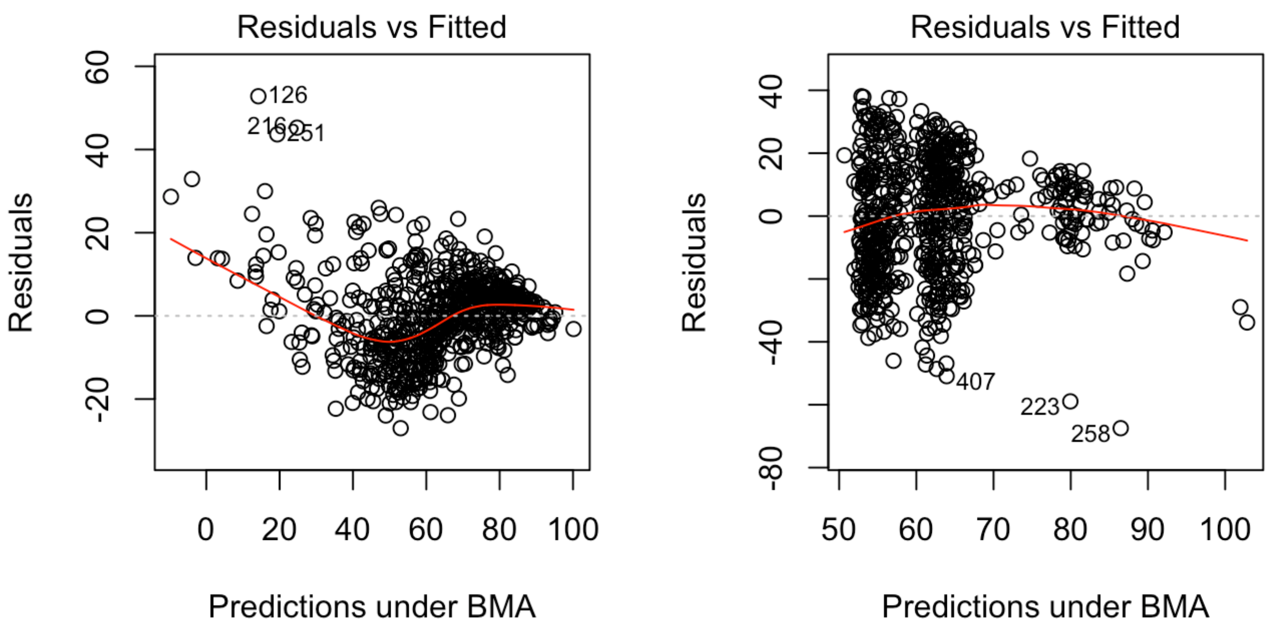
```
##                          2.5  %      97.5  %         beta
## Intercept           61.03599638  63.83596251  62.42372881
## feature_filmyes     -31.42564907 -21.26527557 -26.10642001
## dramayes             4.46942709  10.53934601   7.44514942
## runtime              0.00000000   0.19888999   0.12385617
## mpaa_rating_Ryes    -0.09789422   0.02604717   0.06836463
## thtr_rel_year       -0.19697086   0.00000000  -0.03409065
## oscar_seasonyes      0.00000000   0.00000000  -0.04308922
## summer_seasonyes     0.00000000   1.24060406   0.10351329
## best_pic_nomyes     11.27868296  27.82338066  19.56472817
## best_pic_winyes      0.00000000   0.00000000   0.02691529
## best_actor_winyes   -0.46900498   0.03487621  -0.10610149
## best_actress_winyes  -2.11635527   0.00000000  -0.18548264
## best_dir_winyes      0.00000000   6.15881793   0.58623379
## top200_boxyes        0.00000000  21.00863905   8.88807450
## attr(,"Probability")
## [1] 0.95
## attr(,"class")
## [1] "confint.bas"
```

That said, R2 is quite low across the board, topping out at 0.205. This indicates that even the models with the best BIC do not predict score very well:

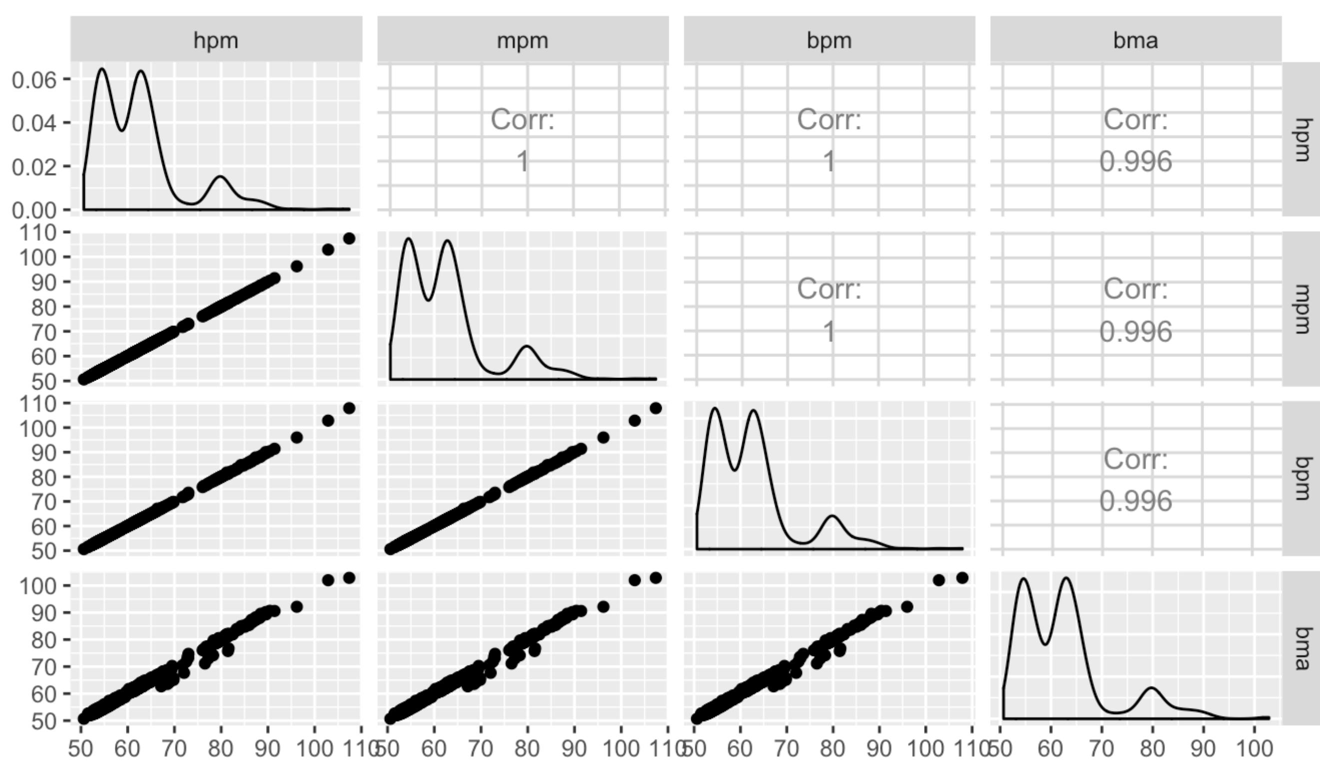
```
##      Intercept feature_filmyes dramayes runtime mpaa_rating_Ryes
## [1,]         1               1        1       1                0
## [2,]         1               1        1       1                0
## [3,]         1               1        1       1                0
## [4,]         1               1        1       1                0
## [5,]         1               1        1       1                0
##      thtr_rel_year oscar_seasonyes summer_seasonyes best_pic_nomyes
## [1,]             0               0                0               1
## [2,]             0               0                0               1
## [3,]             1               0                0               1
## [4,]             1               0                0               1
## [5,]             0               0                0               1
##      best_pic_winyes best_actor_winyes best_actress_winyes best_dir_winyes
## [1,]               0                 0                   0               0
## [2,]               0                 0                   0               0
## [3,]               0                 0                   0               0
## [4,]               0                 0                   0               0
## [5,]               0                 0                   0               1
##      top200_boxyes        BF PostProbs     R2 dim   logmarg
## [1,]             1 1.0000000    0.2806 0.2050   6 -3997.042
## [2,]             0 0.5589602    0.1569 0.1956   5 -3997.624
## [3,]             1 0.2899972    0.0814 0.2099   7 -3998.280
## [4,]             0 0.2201969    0.0618 0.2013   6 -3998.555
## [5,]             1 0.1267125    0.0356 0.2078   7 -3999.108
```

This is especially striking when compared to the full set, where the model with the highest posterior probability (which, again, is based solely on the other ratings) has an R2 of **0.7549**.

The residuals of the two models sets are both fairly normally distributed, but both have some odd behavior at the upper and lower ends (for example, the full set predicts 3 negative scores, while the other predicts two > 100 scores).



Nevertheless, we will move forward with the non-ratings model, as it has the most practical application. Using that set, we'll select a model by reviewing the highest probability, median probability, and best predictive models:



With all methods in agreement, we will move forward with the highest probability model, which has coefficients:

```
##           Intercept     feature_filmyes            dramayes
##          62.4237288         -26.0357711           7.4211660
##             runtime    mpaa_rating_Ryes       thtr_rel_year
##           0.1318971           0.0000000           0.0000000
##     oscar_seasonyes    summer_seasonyes     best_pic_nomyes
##           0.0000000           0.0000000          19.0900292
##     best_pic_winyes   best_actor_winyes best_actress_winyes
##           0.0000000           0.0000000           0.0000000
##     best_dir_winyes       top200_boxyes
##           0.0000000          13.6600468
```

And standard deviations:

```
##           Intercept     feature_filmyes            dramayes
##           0.7103829           2.5478805           1.5139397
##             runtime    mpaa_rating_Ryes       thtr_rel_year
##           0.0391881           0.0000000           0.0000000
##     oscar_seasonyes    summer_seasonyes     best_pic_nomyes
##           0.0000000           0.0000000           4.0865517
##     best_pic_winyes   best_actor_winyes best_actress_winyes
##           0.0000000           0.0000000           0.0000000
##     best_dir_winyes       top200_boxyes
##           0.0000000           4.9508262
```

Part 5: Prediction

Now we will use the model to predict some new movies. I have to confess here - I was unable to get predict() to work for a single row, as I continuously got this error:

```
  ##Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) : contrasts can b
e applied only to factors with 2 or more levels
```

Therefore, I'm predicting on two films rather than just one. These two were left out of the sample at the beginning, for use here.

```
  pred <- predict(no_other_scores_models, test_movies, estimator = "HPM", se.fit =
TRUE, nsim = 10000)
```

This gives the following result:

```
  mu <- confint(pred, parm="mean")
  pred_interval <- confint(pred, parm="pred")
  head(cbind(mu,pred_interval),2)
```

```
##        2.5  %  97.5  %     mean   2.5  %   97.5  %     pred
## [1,] 68.05917 74.19129 71.12523 35.45618 106.79429 71.12523
## [2,] 52.25116 56.61195 54.43155 18.82769  90.03541 54.43155
```

Which is not a very good prediction, given that the true audience_scores were 51 and 34 - so it's way over in both cases.

The full model, which includes the other scores, yields the following:

```
pred <- predict(audience_models, test_movies, estimator = "HPM", se.fit = TRUE, nsim
= 10000)
mu <- confint(pred, parm="mean")
pred_interval <- confint(pred, parm="pred")
head(cbind(mu,pred_interval),2)
```

```
##          2.5  %  97.5  %     mean    2.5  %  97.5  %     pred
## [1,] 49.11964 51.30833 50.21398 30.498507 69.92946 50.21398
## [2,] 24.26939 26.95218 25.61079  5.880057 45.34152 25.61079
```

As expected, the credible interval around the prediction is much tighter, and the predictions are also much closer to the actual. But that's because, again, it's mostly predicting based on what other websites and reviewers said, which isn't particularly informative.

Part 6: Conclusion

There are two possible ways to approach this problem based on the business needs:

1. A reasonably accurate model that includes ratings from other websites (but therefore mostly just tells you what other people are saying); or
2. An inaccurate model that doesn't include ratings from other websites. This may give more insight into the business problem, but I wouldn't be confident enough in its predictions to base many decisions on it.

In short, both approaches are flawed, and it would be a good idea to reevaluate our approach - most likely by collecting more potential predictors, if possible.