Bayesian modeling and prediction for movies

Setup

Load packages

```
options(warn=-1) # turoff warnings
library(ggplot2)
library(dplyr)
library(statsr)
library(BAS)
library(tidyr)
library(knitr)
library(colorspace)
library(GGally)
library(ggpubr)
library(cowplot)
```

Load data

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

Part 1: Data

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

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

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

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

Part 2: Data manipulation

Here we create new explanatory variables, or features, from the existing explanatory variables using the `dplyr` "mutate" function.

Variable 1: `feature_film`, "Yes" if the movies is a feature film, "No" otherwise. Variable 2: `drama`, "Yes" if the movie is a drama, "No" other otherwise. Variable 3: `mpaa_rating_R`, "Yes" if the movie is rated R, "No" otherwise. Variable 4: `oscar_season`, "Yes" if the movie's theatre release was during the Oscar months of October, November, or December, "No" otherwise. Variable 5: `summer_season`, "Yes" if the movie's theatre release was during the summer months of May, June, July, or August, "No" otherwise.

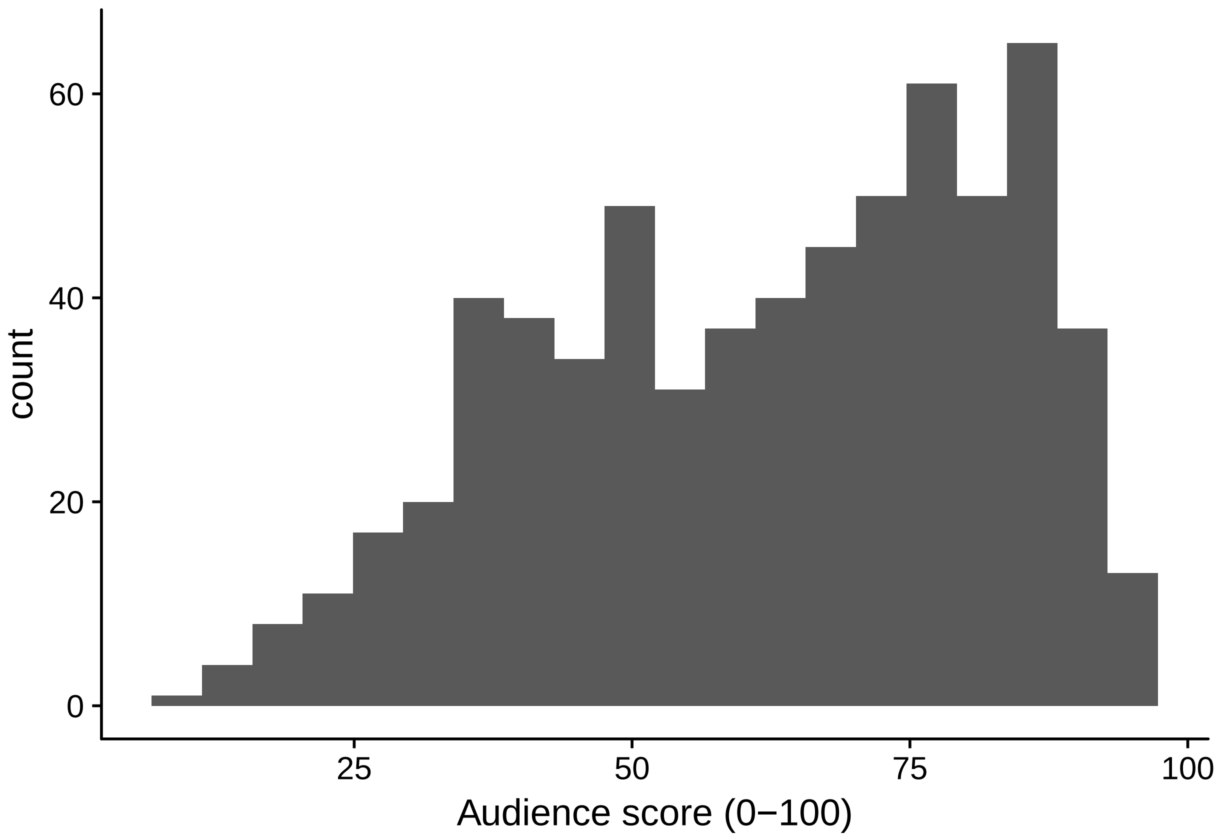
```r
movies <- movies %>% mutate(feature_film = ifelse(title_type == "Feature Film", "yes", "no"))
movies <- movies %>% mutate(drama = ifelse(genre == "Drama", "yes", "no"))
movies <- movies %>% mutate(mpaa_rating_R = ifelse(mpaa_rating == "R", "yes", "no"))
movies <- movies %>% mutate(oscar_season = ifelse(thtr_rel_month %in% c(10,11,12), "yes", "no"))
movies <- movies %>% mutate(summer_season = ifelse(thtr_rel_month %in% c(5,6,7,8), "yes", "no"))
```

Part 3: Exploratory data analysis

Here we explore the relationship between the `audience_score` variable and the new variables created above.

We first show a histogram of the dependent variable `audience_score`. The distribution is slightly bimodal, and left skewed. This is evidenced by the mean being 62.36 and the median being slightly higher, at 65.00.

```r
ggplot(aes(x=audience_score), data=movies) + geom_histogram(bins=20) + xlab("Audience score (0-100)")
```




```r
summary(movies$audience_score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.00   46.00   65.00   62.36   80.00   97.00
```

Here we summarize the proportions of "yes" and "no" instances for the new variables defined and created above. Note that 90.8% of the `feature_film` entries are "yes," indicating that most of the reviews in the data set correspond to feature films.

```r
tmp1 <- movies %>% group_by(feature_film) %>% summarize(Prop = n()/nrow(movies))
tmp2 <- movies %>% group_by(drama) %>% summarize(Prop = n()/nrow(movies))
tmp3 <- movies %>% group_by(mpaa_rating_R) %>% summarize(Prop = n()/nrow(movies))
```

```r
tmp4 <- movies %>% group_by(oscar_season) %>% summarize(Prop = n()/nrow(movies))
tmp5 <- movies %>% group_by(summer_season) %>% summarize(Prop = n()/nrow(movies))
tbl <- cbind(tmp1,tmp2[,2],tmp3[,2],tmp4[,2],tmp5[,2])
kable(tbl,col.names = c("","Var. 1","Var. 2","Var. 3","Var. 4","Var. 5"),caption = "Proportion table fo
```

Table 1: Proportion table for new variables

	Var. 1	Var. 2	Var. 3	Var. 4	Var. 5
no	0.0921659	0.53149	0.4946237	0.7066052	0.6804916
yes	0.9078341	0.46851	0.5053763	0.2933948	0.3195084

To further explore the relationship between `audience_score` and the newly created variables, we create a linear model using linear regression. In the context of statistical inference, the model shows that the `feature_film` and `drama` variables are likely very important predictors; this is evidenced by the nearly-zero p-values for the coefficients of these variables, and large sum-of-squares relative to the total residual sum-of-squares, shown in the ANOVA analysis.

```r
lm_newvar <- lm(audience_score ~ feature_film + drama + mpaa_rating_R + oscar_season + summer_season, da
summary(lm_newvar)
```

```
##
## Call:
## lm(formula = audience_score ~ feature_film + drama + mpaa_rating_R +
##     oscar_season + summer_season, data = movies)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -69.376 -13.903   1.144  14.117  40.342
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       79.1234     2.5617  30.887  < 2e-16 ***
## feature_filmyes  -25.2672     2.6889  -9.397  < 2e-16 ***
## dramayes           9.2525     1.5439   5.993 3.43e-09 ***
## mpaa_rating_Ryes   0.8017     1.5142   0.529    0.597
## oscar_seasonyes    2.7878     1.8102   1.540    0.124
## summer_seasonyes   1.9394     1.7746   1.093    0.275
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.83 on 645 degrees of freedom
## Multiple R-squared:  0.1398, Adjusted R-squared:  0.1331
## F-statistic: 20.96 on 5 and 645 DF,  p-value: < 2.2e-16
```

```r
anova(lm_newvar)
```

```
## Analysis of Variance Table
##
## Response: audience_score
##                Df Sum Sq Mean Sq F value    Pr(>F)
## feature_film    1  23081 23080.6 65.1026 3.479e-15 ***
## drama           1  13068 13068.0 36.8606 2.166e-09 ***
## mpaa_rating_R   1     87    87.4  0.2466    0.6197
## oscar_season    1    492   491.7  1.3868    0.2394
```

```
## summer_season   1    423   423.4  1.1943    0.2749
## Residuals     645 228669   354.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Below we perform chi-square tests of independence for the dependent `audience_score` variable and the explanatory `feature_film` and `drama` variables. The chi-square values are large for these tests, but the p-value for the `audience_score-feature_film` test is very small, while the p-value for the `audience_score-drama` test is not as small, and is in fact greater than 0.05, meaning that we can not reject the null hypothesis that these variables are independent with a 0.05 critical value. We also perform this test for the explanatory variables `drama` and `feature_film`, and find a p-value of nearly zero, indicating we can reject the null hypothesis that these variables are independent.

```
chisq.test(movies$audience_score, movies$feature_film)
```

```
##
##  Pearson's Chi-squared test
##
## data:  movies$audience_score and movies$feature_film
## X-squared = 148.09, df = 83, p-value = 1.482e-05
```

```
chisq.test(movies$audience_score, movies$drama)
```

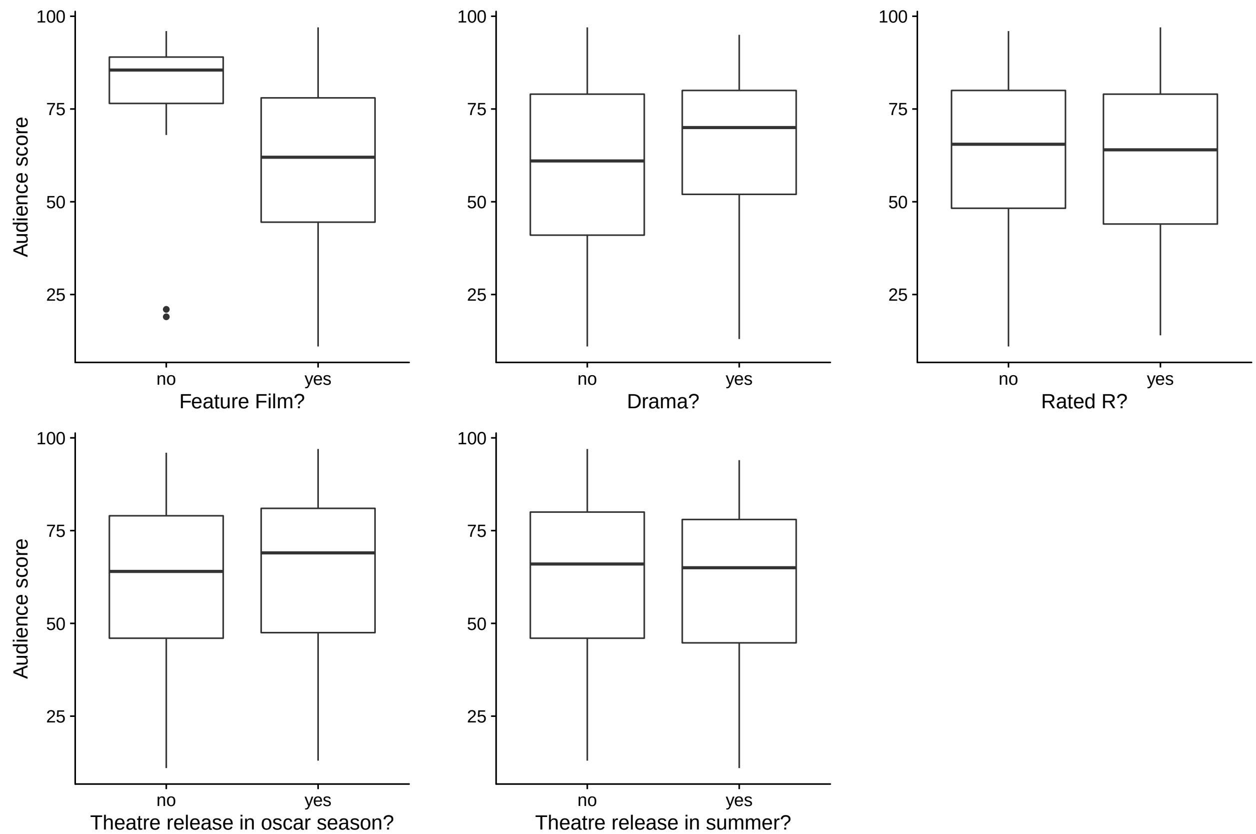
```
##
##  Pearson's Chi-squared test
##
## data:  movies$audience_score and movies$drama
## X-squared = 89.862, df = 83, p-value = 0.2843
```

```
chisq.test(movies$drama, movies$feature_film)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  movies$drama and movies$feature_film
## X-squared = 41.1, df = 1, p-value = 1.446e-10
```

Below, we show box plots of the `audience_score` distribution as a function of the "yes" and "No" catagorical values of the new variables. Once can see that variable `feature_film` appears visually to be the best predictor, followed by the `drama` variable. This is consistent with the above linear regression and chi-square analyses.

```
plot1 <- ggplot(aes(x=feature_film, y=audience_score), data=movies) + geom_boxplot() + xlab("Feature Fi
plot2 <- ggplot(aes(x=drama, y=audience_score), data=movies) + geom_boxplot() + xlab("Drama?") + ylab("
plot3 <- ggplot(aes(x=mpaa_rating_R, y=audience_score), data=movies) + geom_boxplot() + xlab("Rated R?")
plot4 <- ggplot(aes(x=oscar_season, y=audience_score), data=movies) + geom_boxplot() + xlab("Theatre rel
plot5 <- ggplot(aes(x=summer_season, y=audience_score), data=movies) + geom_boxplot() + xlab("Theatre re
ggarrange(plot1, plot2, plot3, plot4, plot5, ncol = 3, nrow = 2)
```



Part 4: Modeling

We develop a linear regression model to predict the dependent `audience_score` variable using Bayesian techniques. The model will use the following explanatory variables:

`feature_film`, `drama`, `runtime`, `mpaa_rating_R`, `thtr_rel_year`, `oscar_season`, `summer_season`, `imdb_rating`, `imdb_num_votes`, `critics_score`, `best_pic_nom`, `best_pic_win`, `best_actor_win`, `best_actress_win`, `best_dir_win`, and `top200_box`

We implement a "Bayesian Information Criteria" (BIC) prior for the model parameters, which amounts to the prior being determined entirely by the likelihoods from the data. Our model prior will be uniform, meaning that we assign equal probability to all possible linear regression models, consisting of all combinations of explanatory variables. We keep only complete cases, and bulid the model using the `bas.lm` function from the Bayesian Adaptive Sampling (BAS) library.

The best model, discussed in more detail below, has only four explanatory variables: `runtime`, `imdb_rating`, `critics_score`, and the intercept. It has a posterior probability of 0.1297, and an R-squared value of 0.7549, meaning that 75.49% of the variance of `audience_score` is explained by the model. The log of the marginal likelihood for this model is -3615.2791.

```
moviesmodel = movies[,c("audience_score","feature_film","drama","runtime","mpaa_rating_R","thtr_rel_yea
moviesmodel = moviesmodel[complete.cases(moviesmodel),]
bma_movies = bas.lm(audience_score ~ ., data = moviesmodel, prior = "BIC", modelprior = uniform())
x = summary(bma_movies)
x[,2][x[,2]!=0]
```

```
##     Intercept        runtime    imdb_rating critics_score               BF
```

```
##        1.0000        1.0000        1.0000        1.0000        1.0000
##     PostProbs            R2           dim       logmarg
##        0.1297        0.7549        4.0000    -3615.2791
```

Below we show the posterior probability that each explanatory variable is included in the linear model. The `imdb_rating` has the highest probability of 1, while perhaps surprisingly the `best_pic_win` variable, indicating whether or not the film won an award for "best picture," has the lowest posterior probability of 0.0398. The other variables included in the model have, as prescribed, relatively high posterior probabilities.

```
paste(bma_movies$namesx,": ",bma_movies$probne0)
```

```
##  [1] "Intercept :  0.999999999999999"
##  [2] "feature_filmyes :  0.0653694673575859"
##  [3] "dramayes :  0.043198334896606"
##  [4] "runtime :  0.469714769072765"
##  [5] "mpaa_rating_Ryes :  0.19984016277148"
##  [6] "thtr_rel_year :  0.0906897042179044"
##  [7] "oscar_seasonyes :  0.0750568423769872"
##  [8] "summer_seasonyes :  0.0804202302078004"
##  [9] "imdb_rating :  0.999999999999999"
## [10] "imdb_num_votes :  0.0577350218928507"
## [11] "critics_score :  0.888550556938742"
## [12] "best_pic_nomyes :  0.131191398175812"
## [13] "best_pic_winyes :  0.0398476619665834"
## [14] "best_actor_winyes :  0.144348962365823"
## [15] "best_actress_winyes :  0.141280873058576"
## [16] "best_dir_winyes :  0.0669389788393073"
## [17] "top200_boxyes :  0.0476223406247296"
```

```
postprobtbl <- data.frame(bma_movies$namesx,bma_movies$probne0)
kable(postprobtbl,col.names = c("Variable","Posterior probability"))
```

Variable	Posterior probability
Intercept	1.0000000
feature_filmyes	0.0653695
dramayes	0.0431983
runtime	0.4697148
mpaa_rating_Ryes	0.1998402
thtr_rel_year	0.0906897
oscar_seasonyes	0.0750568
summer_seasonyes	0.0804202
imdb_rating	1.0000000
imdb_num_votes	0.0577350
critics_score	0.8885506
best_pic_nomyes	0.1311914
best_pic_winyes	0.0398477
best_actor_winyes	0.1443490
best_actress_winyes	0.1412809
best_dir_winyes	0.0669390
top200_boxyes	0.0476223

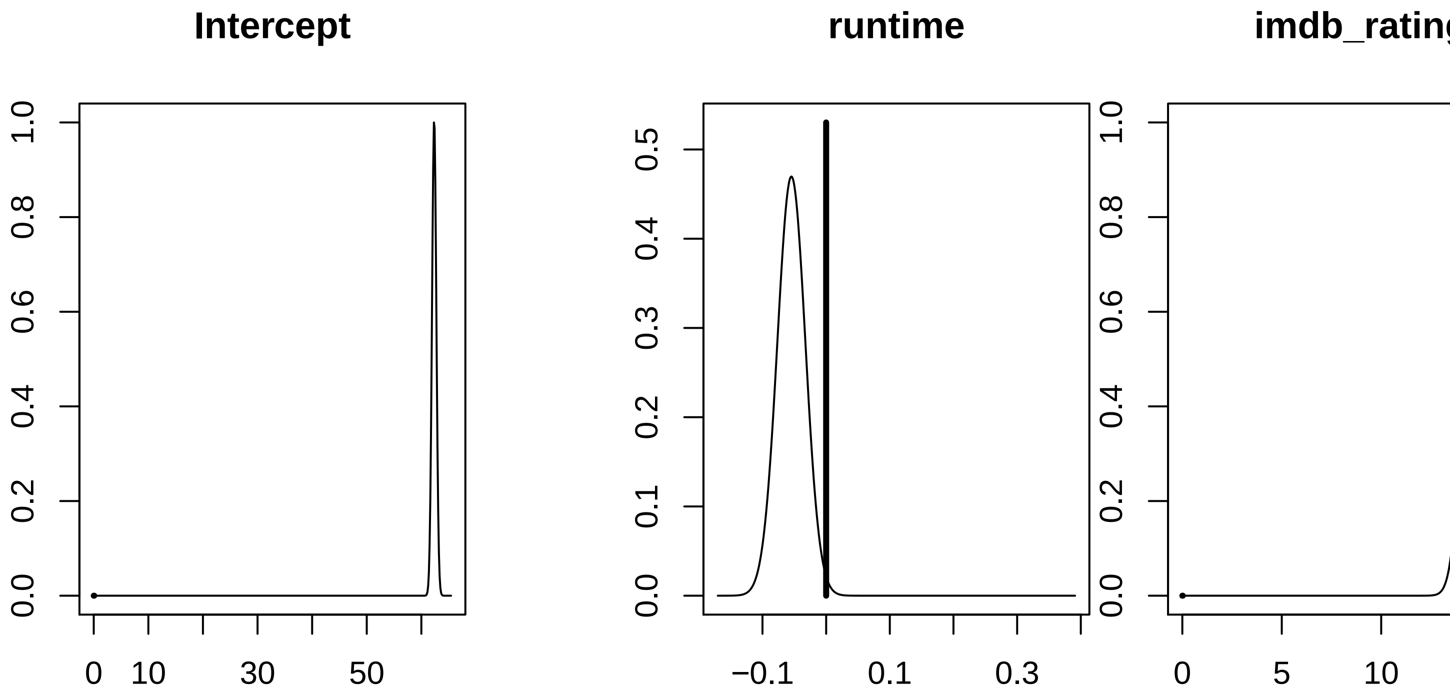
The beta model parameters (linear slopes) for each of the variables is shown below, with corresponding 95% credible interval bounds. The variable with the largest beta is `imdb_rating`, for which $\beta = 14.98$. This means that a 1 point increase in `imdb_rating` on average corresponds to a 14.98 point increase in `audience` score. Similarly, if the film is a drama it will have an audience_score value that is on average .016 higher than if it is not a drama, and for every additional minute of run time (as captured by the runtime variable) the audience_score decreases by .0257 on average.

```
coef_movies = coefficients(bma_movies)
confint(coef_movies)
```

```
##                              2.5%        97.5%          beta
## Intercept            6.159103e+01 6.314813e+01  6.234769e+01
## feature_filmyes     -1.224125e+00 1.001035e-02 -1.046908e-01
## dramayes             0.000000e+00 0.000000e+00  1.604413e-02
## runtime             -8.218014e-02 0.000000e+00 -2.567772e-02
## mpaa_rating_Ryes    -2.126625e+00 2.335434e-04 -3.036174e-01
## thtr_rel_year       -5.654207e-02 5.401352e-05 -4.532635e-03
## oscar_seasonyes     -9.000997e-01 4.261378e-03 -8.034940e-02
## summer_seasonyes     0.000000e+00 1.083938e+00  8.704545e-02
## imdb_rating          1.363707e+01 1.652688e+01  1.498203e+01
## imdb_num_votes      -2.017827e-08 1.631411e-06  2.080713e-07
## critics_score        0.000000e+00 1.057645e-01  6.296648e-02
## best_pic_nomyes      0.000000e+00 4.928373e+00  5.068035e-01
## best_pic_winyes      0.000000e+00 0.000000e+00 -8.502836e-03
## best_actor_winyes   -2.536663e+00 1.966406e-03 -2.876695e-01
## best_actress_winyes -2.987987e+00 0.000000e+00 -3.088382e-01
## best_dir_winyes     -1.428798e+00 0.000000e+00 -1.195011e-01
## top200_boxyes        0.000000e+00 0.000000e+00  8.648185e-02
## attr(,"Probability")
## [1] 0.95
## attr(,"class")
## [1] "confint.bas"
```

Below we show the posterior probability distributions for the explanatory variables included in the best model. One can see that the probabiliy that runtime is not an important variable is still greater than 0.5.

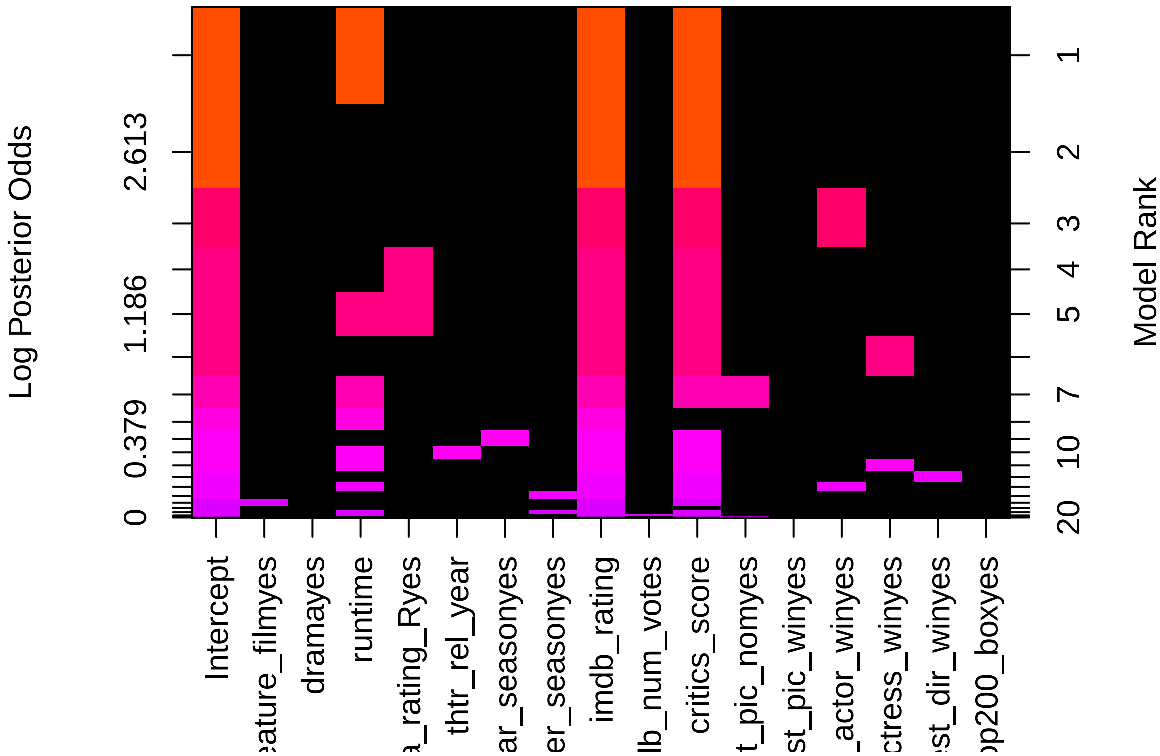
```
par(mfrow = c(1,2))
plot(coef_movies, subset=c(1,4,9,11), ask=FALSE)
```



While the best model discussed above, which corresponds to the single model that minimizes the marginal likelihood, is quite simple, Bayesian Model Averaging (BMA) provides a powerful predictive tool, and averages over all models with weights given by their posterior probabilities calcualated under Bayesian Adaptive Sampling.

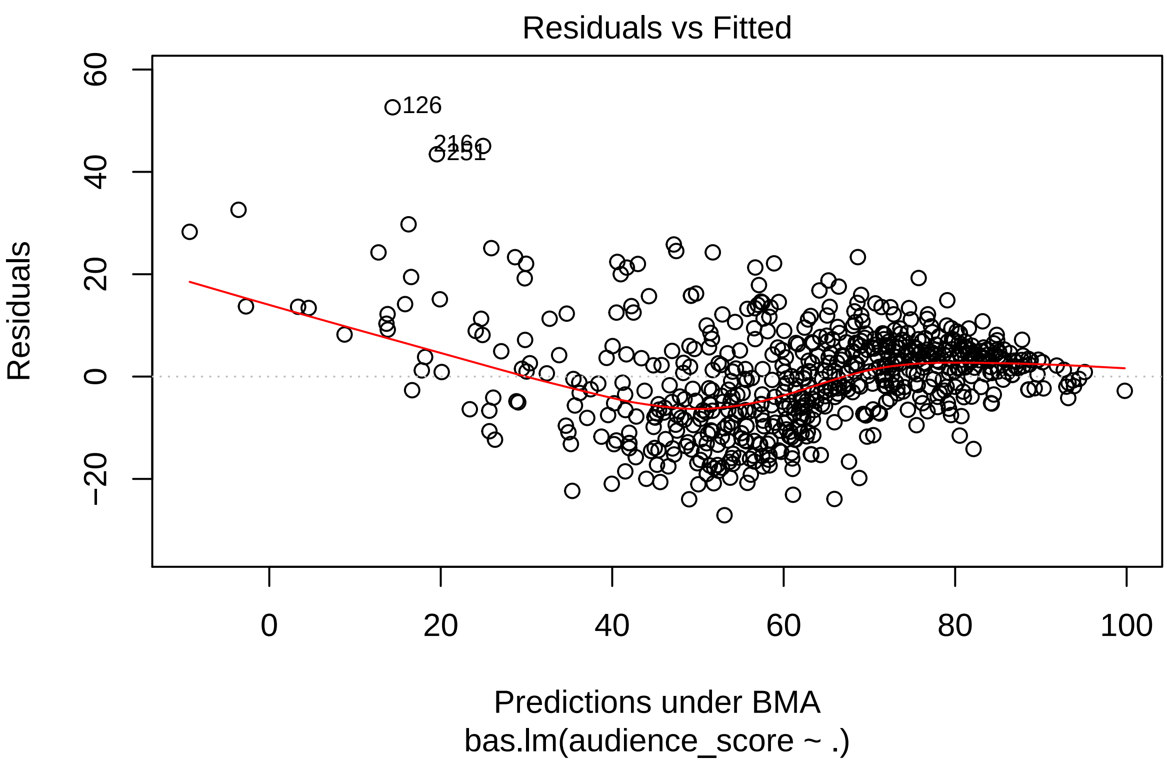
Below, we show a diagram that illustrates the best models, arranged ordinally by their posterior odds. BMA uses all of these models. As one can see, the best model, discussed above, corresponds to the row at the top of this diagram.

```
image(bma_movies, rotate=TRUE)
```



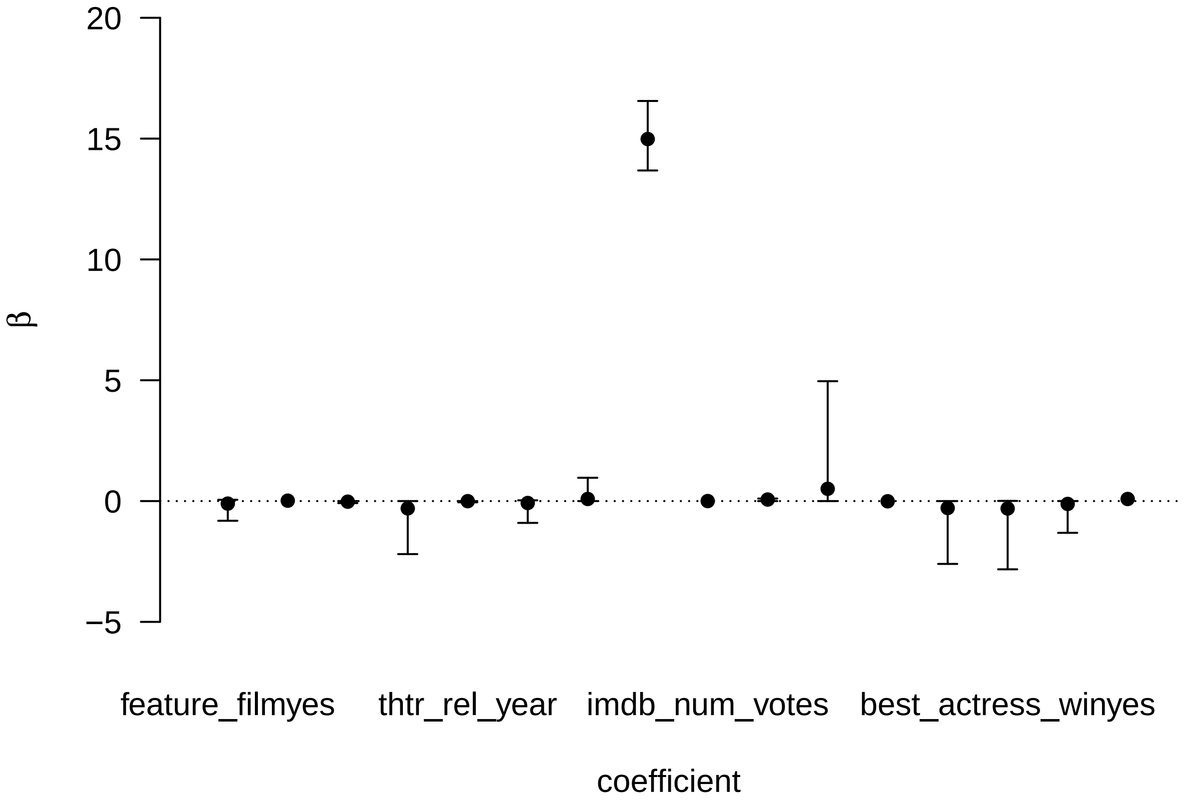
Below we plot the BMA model residuals as a function of the BMA predictions. Ideally, we would see approximately normal distribution about zero, and constant variance. This is not quite the case, and three statistical outliers have been identified.

```
plot(bma_movies, which=1)
```



Below, we plot the creidble intervals for each beta parameter in the model under BMA.

```r
plot(confint(coef_movies, parm=2:17, estimator = "BMA"))
```

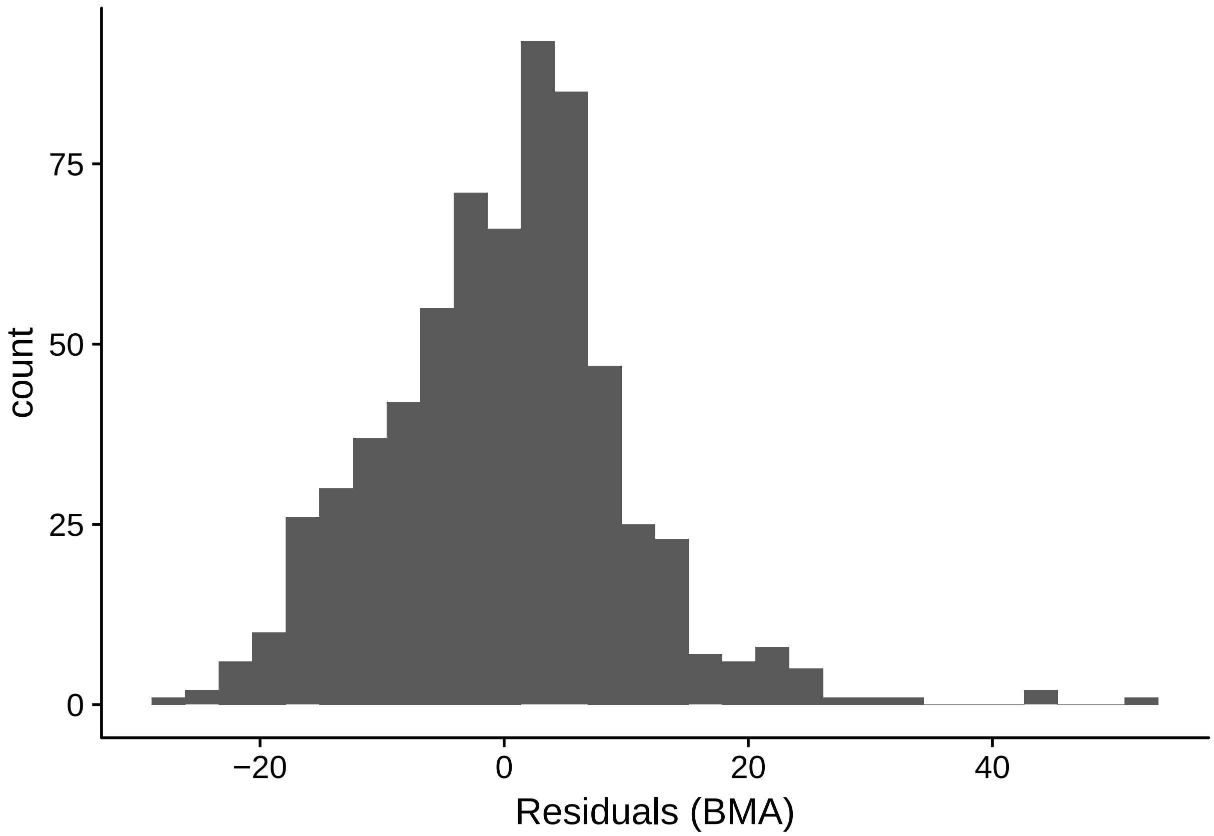



```
## NULL
```

Finally, we show a histogram of the residuals under BMA. The distribution is approximately normal, but is slightly skewed.

```r
BMA_pred_movies =  predict(bma_movies, estimator="BMA", se.fit=TRUE)
ggplot(aes(x = moviesmodel$audience_score - BMA_pred_movies$Ybma[,1]), data = moviesmodel) + geom_histog
```

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



Part 5: Prediction

Here, we use BMA to predict the Rotten Tomatoes "Audience Score" (the `audience_score` variable) of the 2016 movie "Money Monster," chosen at random from the IMDB website:

https://www.rottentomatoes.com/m/money_monster/

The Audience Score is 51. Below, we use Bayesian Model Averaging (BMA) on the Bayesian Adaptive Sampling linear model `bma_model` to predict the score, and make a comparison. The predicted score, as shown below, is 59.98. The 95% credible interval for the prediction is [38.28,79.09], meaning there is a 95% probability the audience score of this movie falls in this interval. Sure enough, the audience score (51) falls near the middle of this interval.

```
feature_film <- "yes"
drama <- "yes"
runtime <- 198
mpaa_rating_R <- "yes"
thtr_rel_year <- 2016
oscar_season <- "no"
summer_season <- "yes"
imdb_rating <- 6.5
imdb_num_votes <- 72600
critics_score <- 57
best_pic_nom <- "no"
best_pic_win <- "no"
best_actor_win <- "no"
```

```r
best_actress_win <- "no"
best_dir_win <- "no"
top200_box <- "no"
newmovie <- data.frame(feature_film, drama, runtime, mpaa_rating_R, thtr_rel_year, oscar_season, summer_
new.pred = predict(bma_movies, newdata=newmovie, estimator="BMA", se.fit=TRUE)
confint(new.pred,level=.95)
```

```
##          2.5%    97.5%     pred
## [1,] 40.25144 80.85717 59.98352
## attr(,"Probability")
## [1] 0.95
## attr(,"class")
## [1] "confint.bas"
```

Part 6: Conclusion

To conclude, we developed a multiple linear regression model to predict the Rotten Tomatoes Audience Score (corresponding to the `audience_score` variable) of a film. This model was developed using Bayesian Adaptive Sampling all possible models given the explanatory variables, and the prediction was made using Bayesian Model Averaging. The most important feature was `imdb_rating`, which is intuitive and expected, as variable simply corresponds to the user rating from a different movie website. The residuals were not quite normal, and the variance not constant over the range of predicted values, indicating that the linear model might not be best to predict Audience Scores, and assign credible intervals.