Data Science: Capstone Report

Executive Summary

Data Science Course Colleagues, the goal of this project is to accurately predict the rating users will give to movies with a RMSE less than 0.87750. This is the motivation to reinforce past learning and to explorer solutions to real-world data-science problems not covered in class, e.g., large data sets, limited computing resources, and changing requirements. Personally, this project was also motivation for me to explore the features of R Markdown that I have not used in the past.

The key steps are importing the data set, creating and tuning a model, and evaluating the model against an "unknown" data set. The raw data was imported using the provided script. This script creates an edx data set used to build a model and a validation data set used for final evaluation. I split the edx data set into a train_set and a test_set in order to avoid making any modelling decisions with the validation set. Then model parameters are tuned and evaluated against the test_set. Finally, the model makes predictions for the users and movies in the previously unseen validation set to obtain the final RMSE score of 0.8606 meeting the requirement.

Methods

The data required no cleaning. There were no missing values and all existing values appeared reasonable. I depended heavily on prior published work with the data set (*The BellKor Solution to the Netflix Grand Prize*, Yehuda Koren, August 2009) to guide additional predictor creation and visualization.

I choose an ensemble model with three sub-models: a very simplified version of BellKor Netflix solution (referred to as the "custom" built model) that extends the model developed during class, a tree-based gradient boosting model (XGBoost), and a neural network model (NNET). I did not have the computing resources to analyze the entire data. I had to work with various sample sizes. For example, in order to tune parameters for XGBoost and NNET I used a random sample of 1 million observations. I assume that the sample represents the full data set well enough to result in a reasonably good choice of parameters.

The general organization of the code is

- Build the data sets
- For each model
  - Train
  - Generate predictions
  - Add to ensemble
- Evaluate the ensemble to generate the final predictions.

First I build the data sets. I used the provided script which downloads the raw data and takes care of correctly building the edx and validation data sets. I saved these as .rds objects. I then take the edx data set and split it into a train and test set. I ignore the validation set until the final model run thru.

It's important not to use the validation data during model development. So, I substitute the generic variable "eval_set" in the models where we would, in class, use the variable "test_set". While developing the model I set eval_set equal to test_set. But, when running the final validation I set eval_set equal to validation.

I built the code to be modular in order to accept additional models. This makes it easier to plug-in additional models without damaging the code's overall structure [1]. So, for each model I train using the train_set, predict using the test_set during model development and the validation during final the run, and add the prediction to the ensemble. Finally, I evaluate the ensemble by taking the average of each model's prediction.

[1] Indeed, during development I swapped in/out several models. In the early stages I was experimenting. Several of the models didn't run well enough given my computing resources to be included. Others, such as the "Recommender Labs" package, I wasn't able to get very good results from - certainly due to my inexperience.

Results

The ensemble produced predictions with an RMSE of **0.8606**. I was able to achieve an RMSE of **0.8606** which meets the required RMSE using only the model provided in class. However, since my goal was to explore and practice I implemented the full ensemble of three. The XGBoost and NNET models performed only slightly better than the custom model:

Model	RMSE
Ensemble	0.8606
Custom	0.8623
XGBoost	0.8609
NNet	0.8608

This surprised me despite Bell and Koren's concluding remarks on baseline predictors:

> Out of the numerous new algorithmic contributions, I would like to highlight one – those humble baseline predictors (or biases), which capture main effects in the data. While the literature mostly concentrates on the more sophisticated algorithmic aspects, we have learned that an accurate treatment of main effects is probably at least as significant as coming up with modeling breakthroughs.

Conclusion

In conclusion, I was able to successfully create a model that meets the RMSE requirement. I learned a lot during this project:

- How to work with large data set (sampling, model parameter guessing). The 10 million records in the data set are much too large for me to work with directly when exploring the data or developing the model. Choosing smaller samples (from 10,000 to 1 million) was critical.
- Microsoft Azure virtual machines. My home computer was fine during the course. But, it is unable to create the edx or validation sets. My only option was to create a more powerful virtual machine - I choose Microsoft's Azure environment. My final virtual machine had 8 processors and 32 GB of memory. More powerful machines were available but their cost was prohibitive.
- Parallel processing in R (`doParallel` package). I learned that R does have the packages to support multiple CPUs. This was especially useful during Caret training to determine the optimum model parameters. Evaluating parameters went from 10-15 hours to 2-3 hours.
- I enhanced my knowledge of Git distributed version control. Using the Atlassian Sourcetree front end for Git along with Github was essential for me to complete this project. I could not have effectively worked on the same code using my physical home PC and the Azure VM without a central repository. Not only did it allow me easy access from both computers, it saved me countless times by allowing me to revert to previous code versions after a long walk down a dead-end development path.

There are several improvements to consider. I would do more work to possibly include the movie Genre as a predictor. Perhaps as a component of the movie bias but also perhaps as a component to the user bias. I would implement user bias as a function of time as in the Netflix competition model. Finally, I would consider the number of movies a user reviews together (also part of the Netflix model) as a possible predictor.

A detailed exploratory data analysis follows.

Data Exploration

Summary

The data is from the GroupLens research lab at the University of Minnesota. The data consists of **9000055** observations of **6** variables. The data spans the years from **1995** to **2009**.

The `userId` variable is an integer representation of an actual MovieLens userId which has been anonymized to protect privacy. The `movieId` is the actual MovieLens ID. Ratings are made on a 5-star scale with half-star increments. Time-stamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. The title includes the title and year the movie was released. The `genres` variable is a concatenation of all genres which apply to a specific movie.

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                         genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

Missing data, label encoding, and factorizing variables

Do any variables contain missing values?

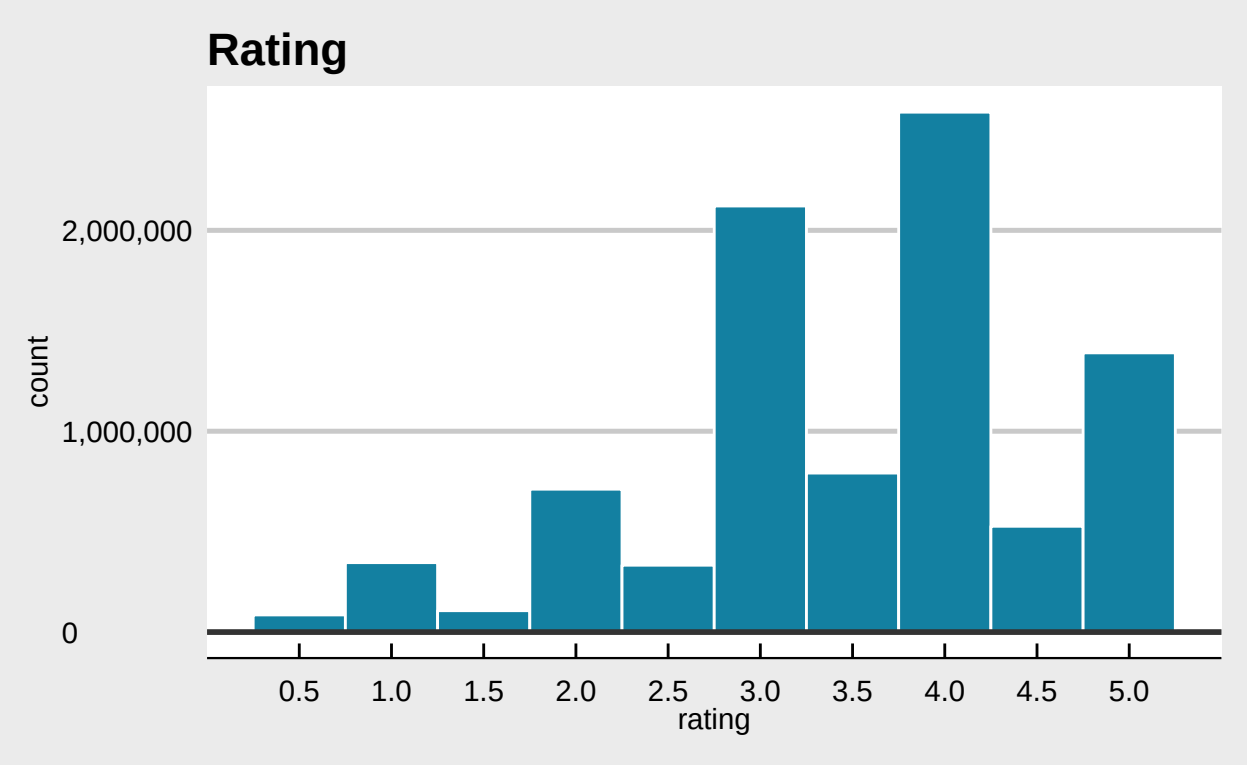
There are 0 columns with at least one NA.

There are no labels that are ordinal and should be encoded as a number.

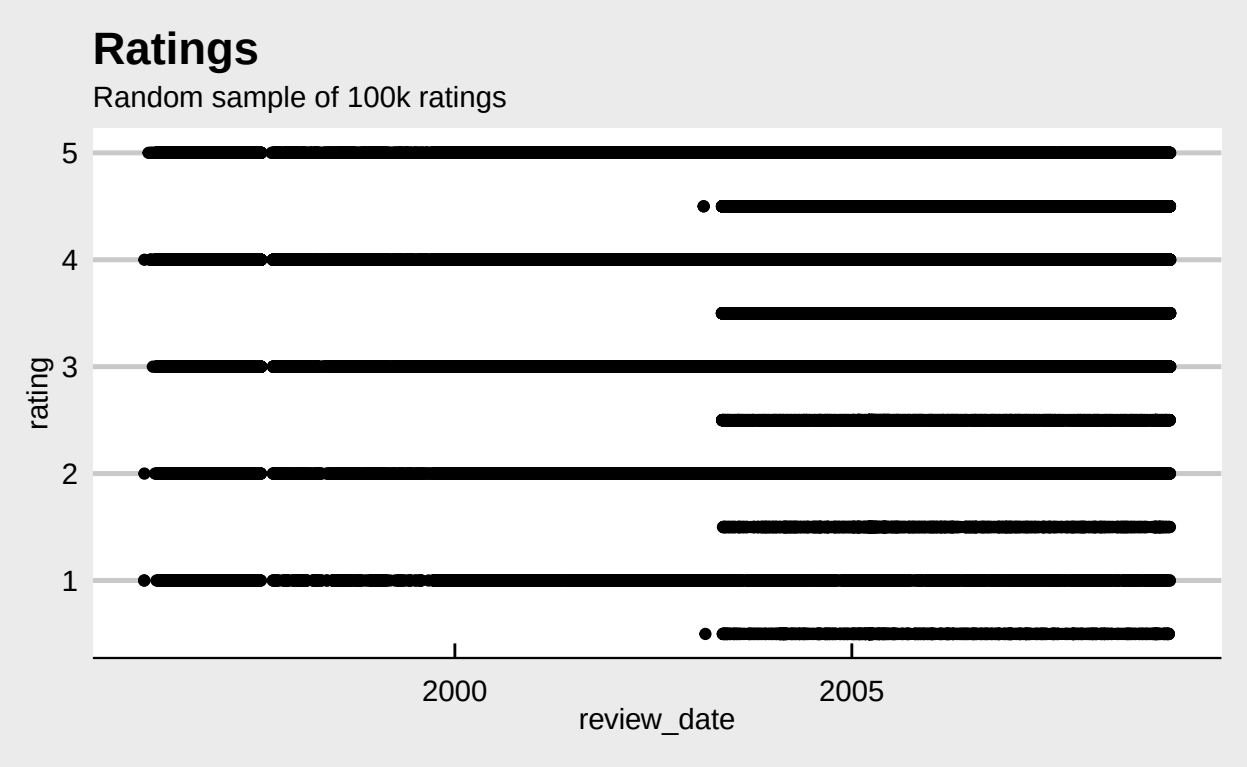
We will convert `userId` and `movieId` from an integer to a factor. We'll also separate the `genres` and store this result in a new data frame - we'll need to refer to this data frame a couple of times and don't want to redo this computationally expensive process. We'll create a `review_date` variable from `timestamp` for the same reason.

Response Variable

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```



The `rating` response variable shows a predominance of whole-star ratings. This may be due at least two factors: the rating data does not include half-star ratings until 2003 and users may have a strong preference for whole-star ratings. Looking at either the whole-star or half-star ratings we see a somewhat normal distribution which is strongly left skewed.



Prior to May, 2003, there are no 1/2-star ratings. But, this does not appear to effect the rating distribution. It may be useful in a model as if a predicted rating is prior to May, 2003. For example, if a model predicts 3.8 for a movie reviewed before May 2003 it may be better to round this to 4 stars. Of course, if the actual rating is 3 stars then rounding will hurt. So, this idea needs some exploration.

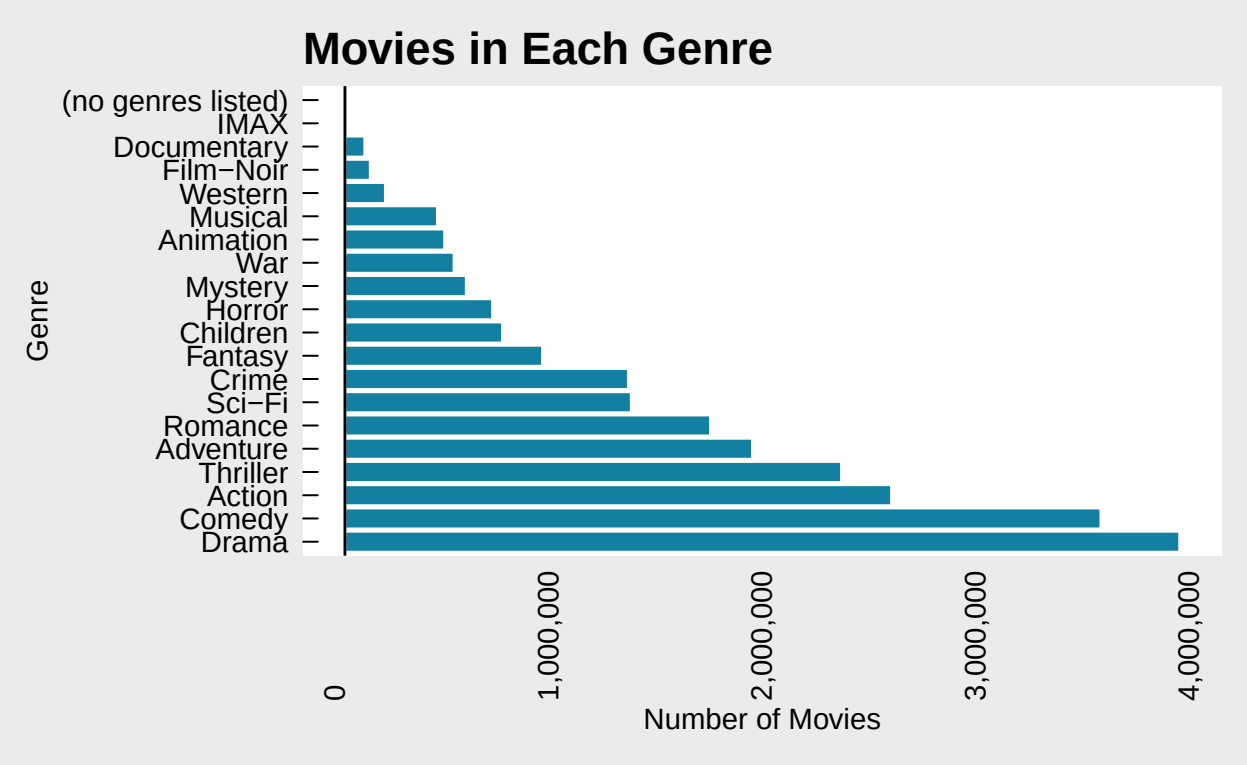
Other Variables

In the data set there are:

- **69878** unique users
- **10677** unique movies rated

Genres

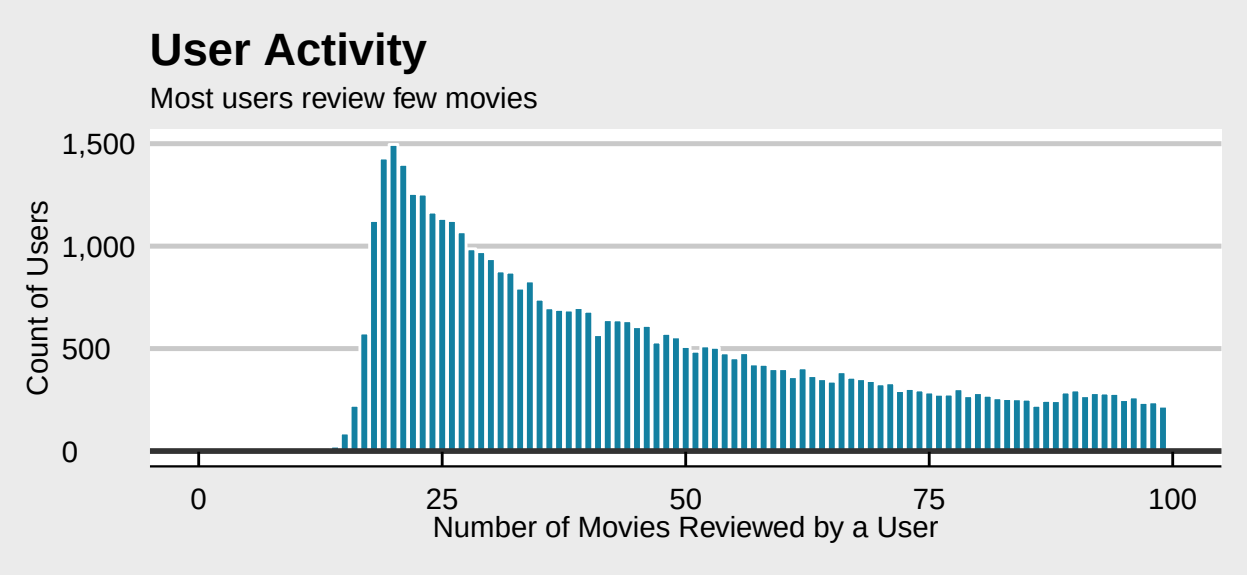
Number of reviewed movies in each genre. Note that a movie may be tagged with several genres, e.g., the movie "Boomerang" is classified as both a "Comedy" and a "Romance."



Which movies did not have a genre listed?

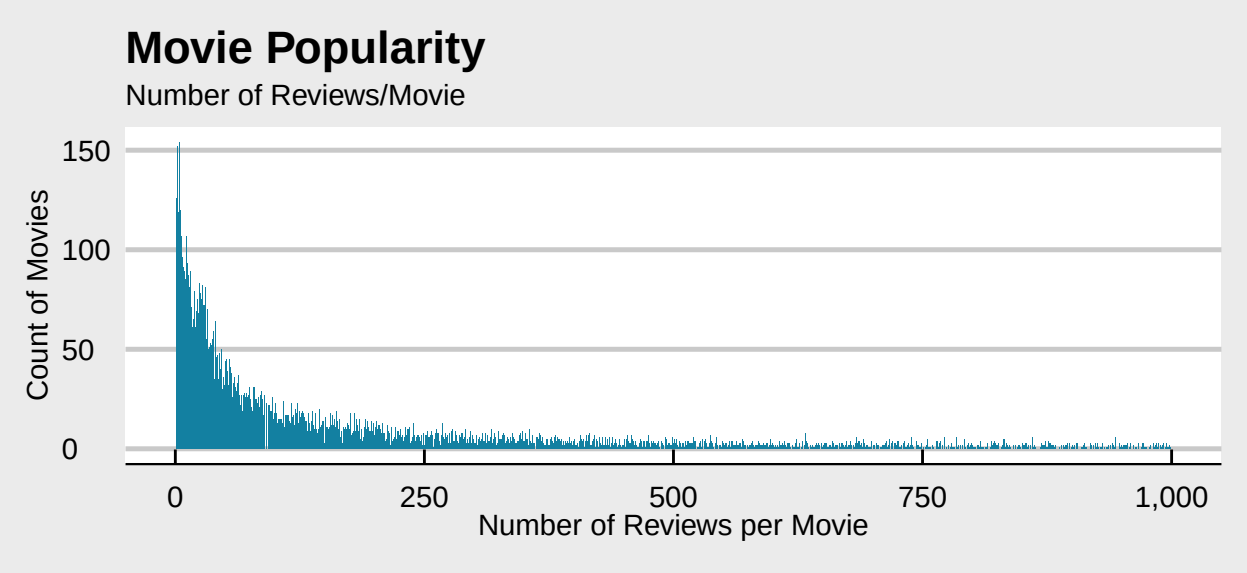
```
##   userId movieId rating  timestamp                title             genres
## 1   7701    8606    5.0 1190806786 Pull My Daisy (1958) (no genres listed)
## 2  10680    8606    4.5 1171170472 Pull My Daisy (1958) (no genres listed)
## 3  29097    8606    2.0 1089648625 Pull My Daisy (1958) (no genres listed)
## 4  46142    8606    3.5 1226518191 Pull My Daisy (1958) (no genres listed)
## 5  57696    8606    4.5 1230588636 Pull My Daisy (1958) (no genres listed)
## 6  64411    8606    3.5 1096732843 Pull My Daisy (1958) (no genres listed)
## 7  67385    8606    2.5 1188277325 Pull My Daisy (1958) (no genres listed)
##   review_date
## 1  2007-09-26
## 2  2007-02-11
## 3  2004-07-12
## 4  2008-11-12
## 5  2008-12-29
## 6  2004-10-02
## 7  2007-08-28
```

Users



Movies Reviewed by Users, truncated to show only the left side of the extremely right-skewed distribution.

Movies



How are the reviews/movie distributed?

What are the most frequently reviewed movies...

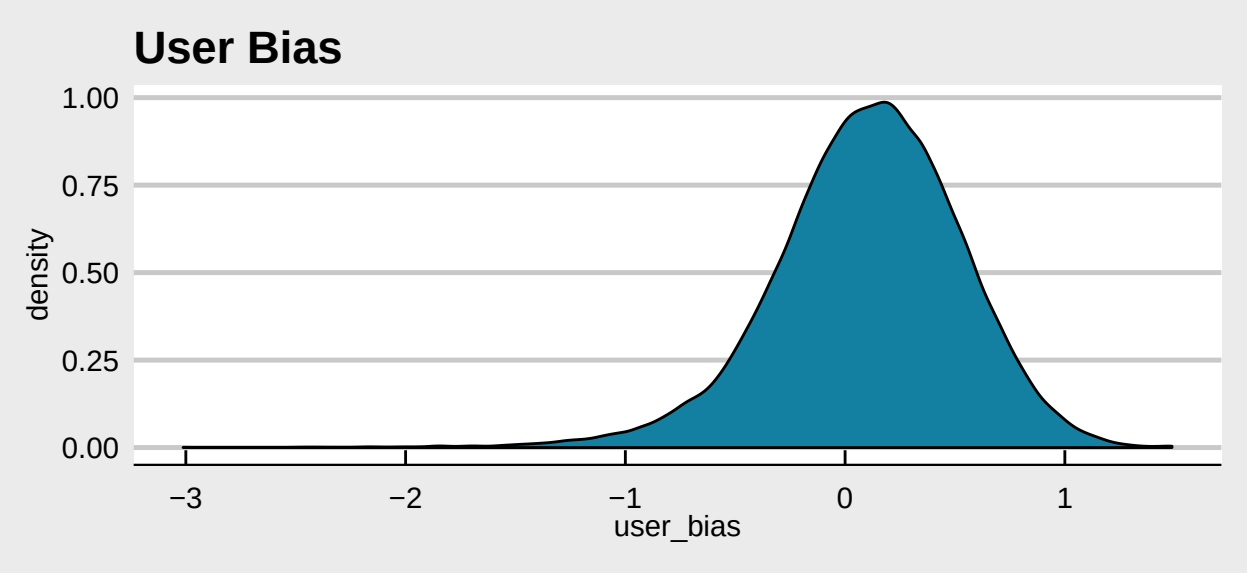
```
## # A tibble: 10 x 2
##    title                                 count
##    <chr>                                 <int>
##  1 Pulp Fiction (1994)                   31362
##  2 Forrest Gump (1994)                   31079
##  3 Silence of the Lambs, The (1991)      30382
##  4 Jurassic Park (1993)                  29360
##  5 Shawshank Redemption, The (1994)      28015
##  6 Braveheart (1995)                     26212
```

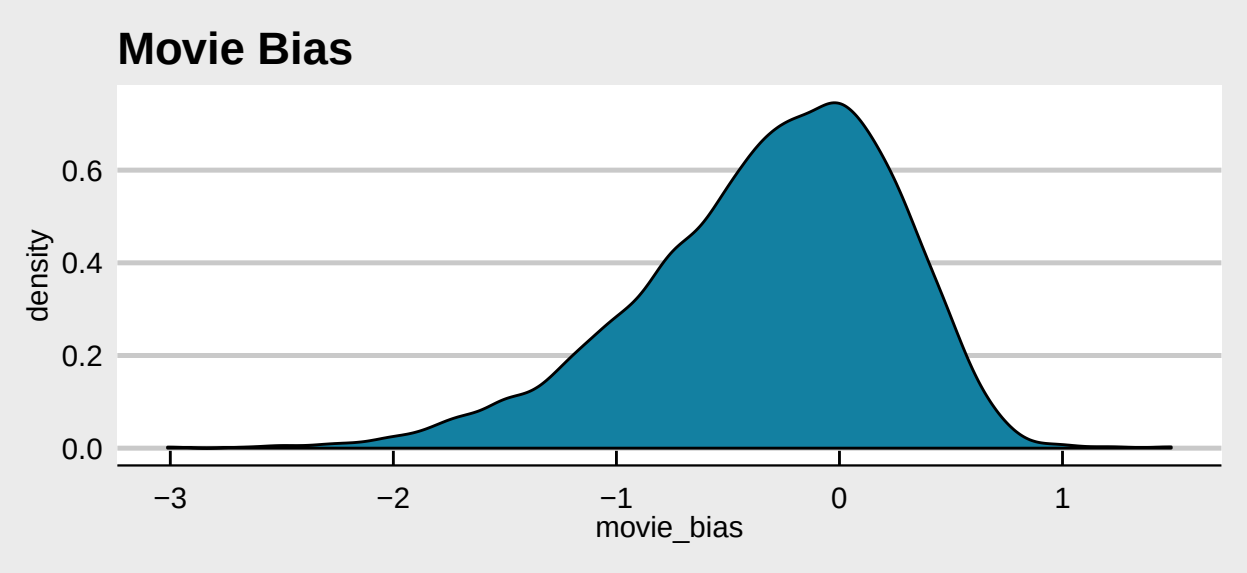
```
##  7 Fugitive, The (1993)                                          25998
##  8 Terminator 2: Judgment Day (1991)                             25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10 Apollo 13 (1995)                                              24284
```

. . . and least frequently reviewed movies?

```
## # A tibble: 10 x 2
##    title                                                          count
##    <chr>                                                          <int>
##  1 1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)                      1
##  2 100 Feet (2008)                                                    1
##  3 4 (2005)                                                           1
##  4 Accused (Anklaget) (2005)                                          1
##  5 Ace of Hearts (2008)                                               1
##  6 Ace of Hearts, The (1921)                                          1
##  7 Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio d~     1
##  8 Africa addio (1966)                                                1
##  9 Aleksandra (2007)                                                  1
## 10 Bad Blood (Mauvais sang) (1986)                                    1
```

All users do not rate movies using the same internal scale. Some users will generally rate high and others generally low. This is called `user_bias` and is the difference between the average rating and this particular user's rating. We can look for this bias by exploring the distribution of user ratings deviation from the mean of the population. We do the same for each movie - some movies are generally rated better than other movies - to establish a `movie_bias`. We see both of these are fairly normally distributed.

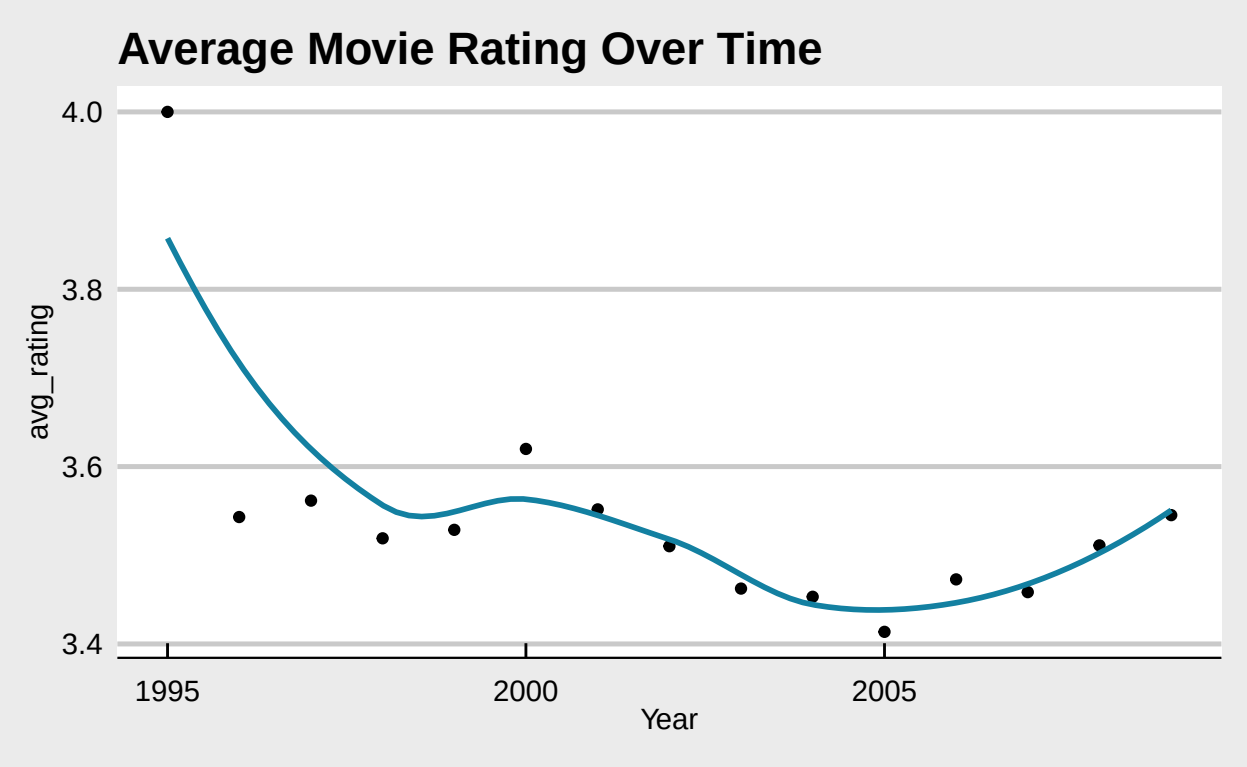


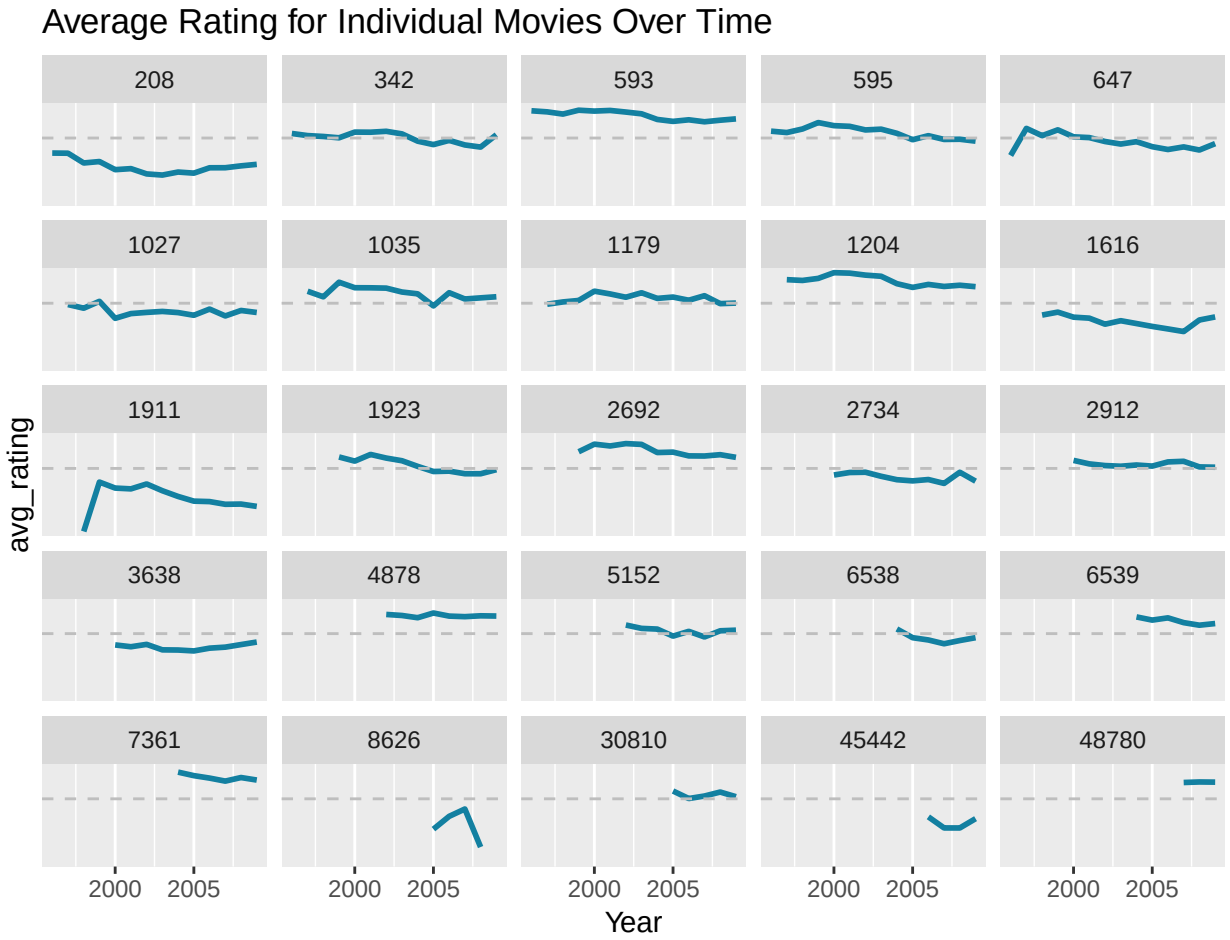


Timestamp

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "1995-01-09" "2000-01-01" "2002-10-24" "2002-09-21" "2005-09-15"
##         Max.
## "2009-01-05"
```

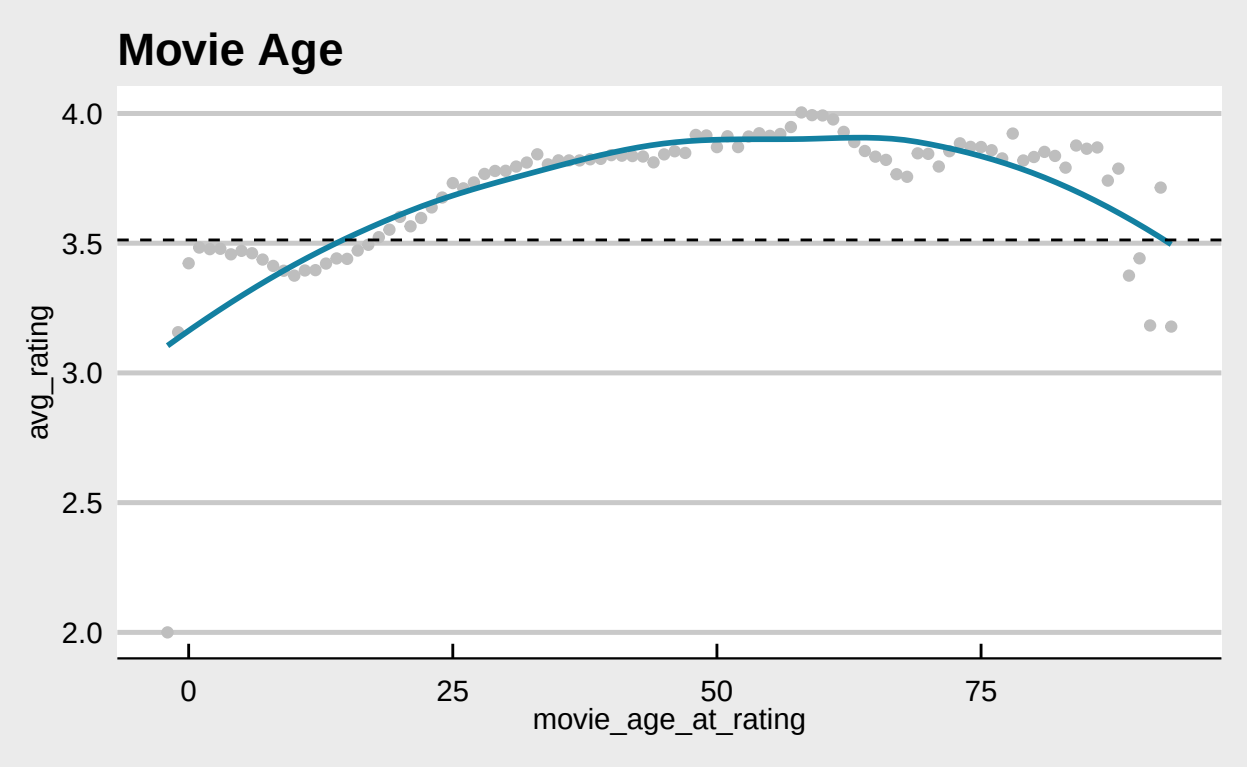
Does the overall average movie rating change over time?



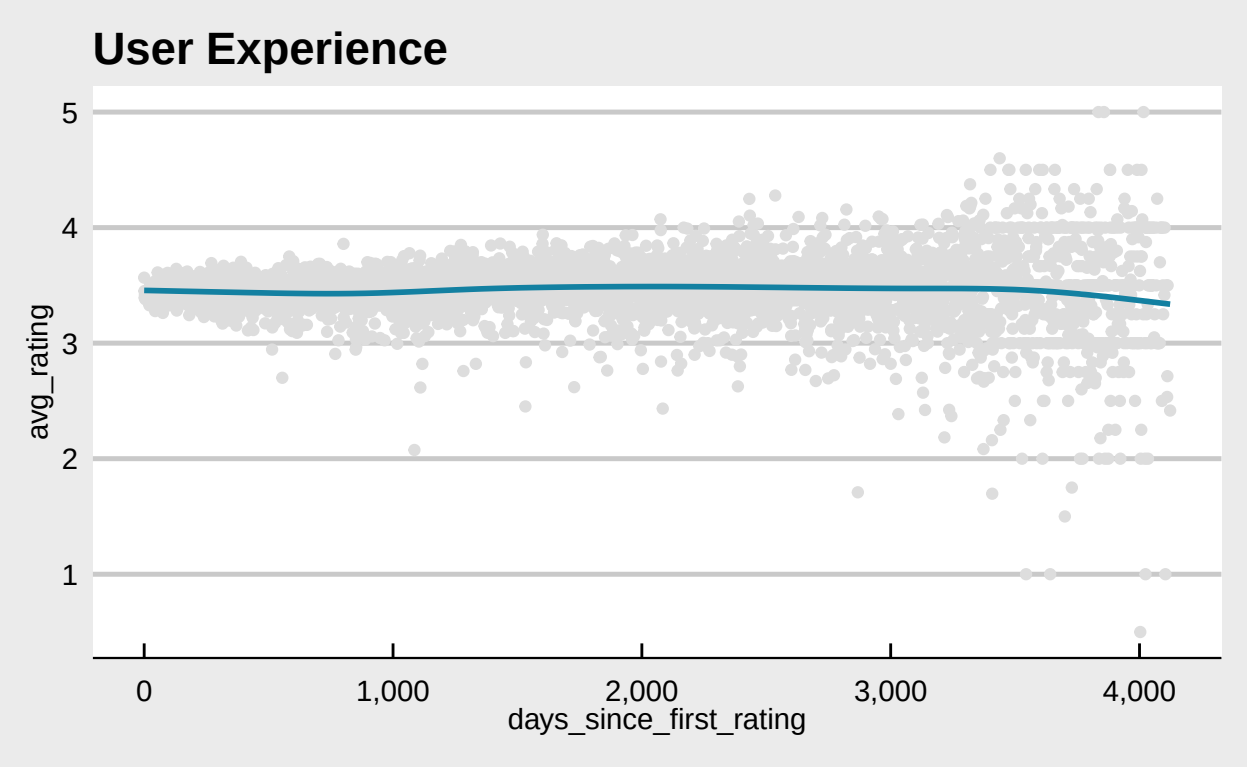


Does a particular movie's `average rating` change over time? There are too many movies to visualize this effectively so we will select a small random sample. For reference, the dashed horizontal line is the overall average rating.

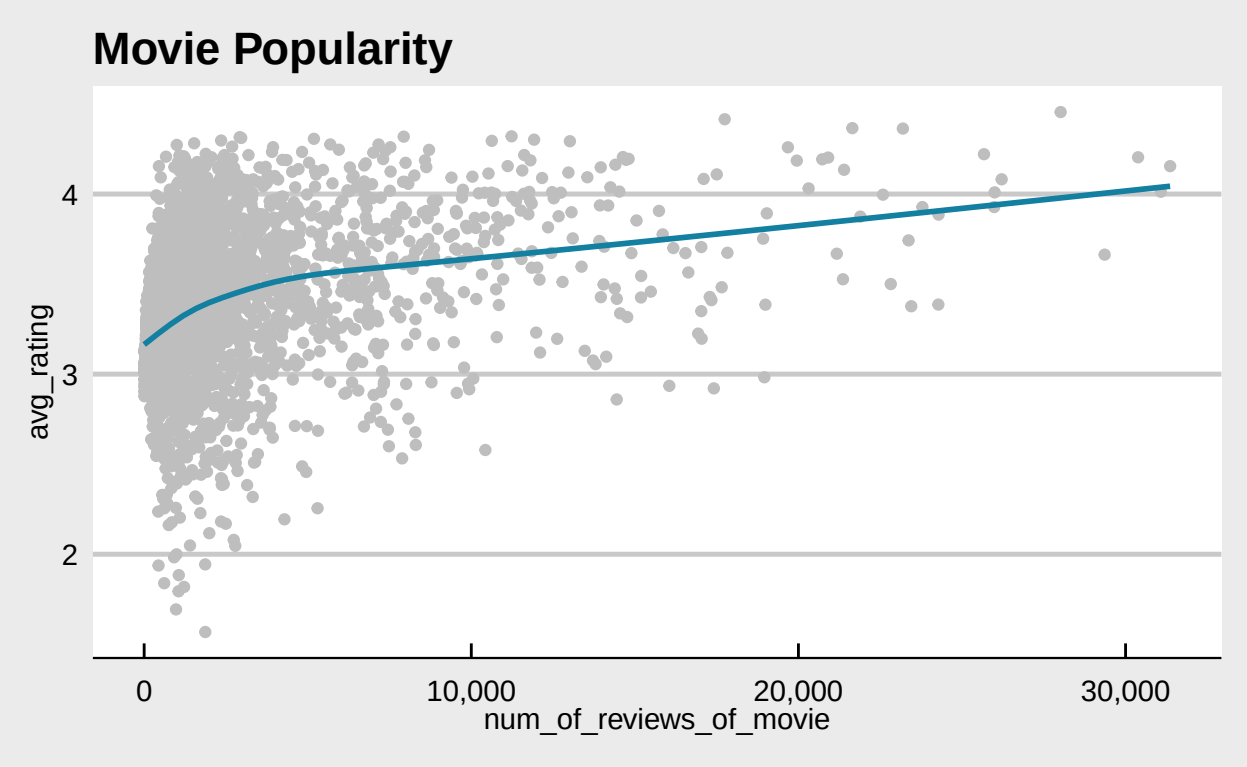
Possible Predictors to Create



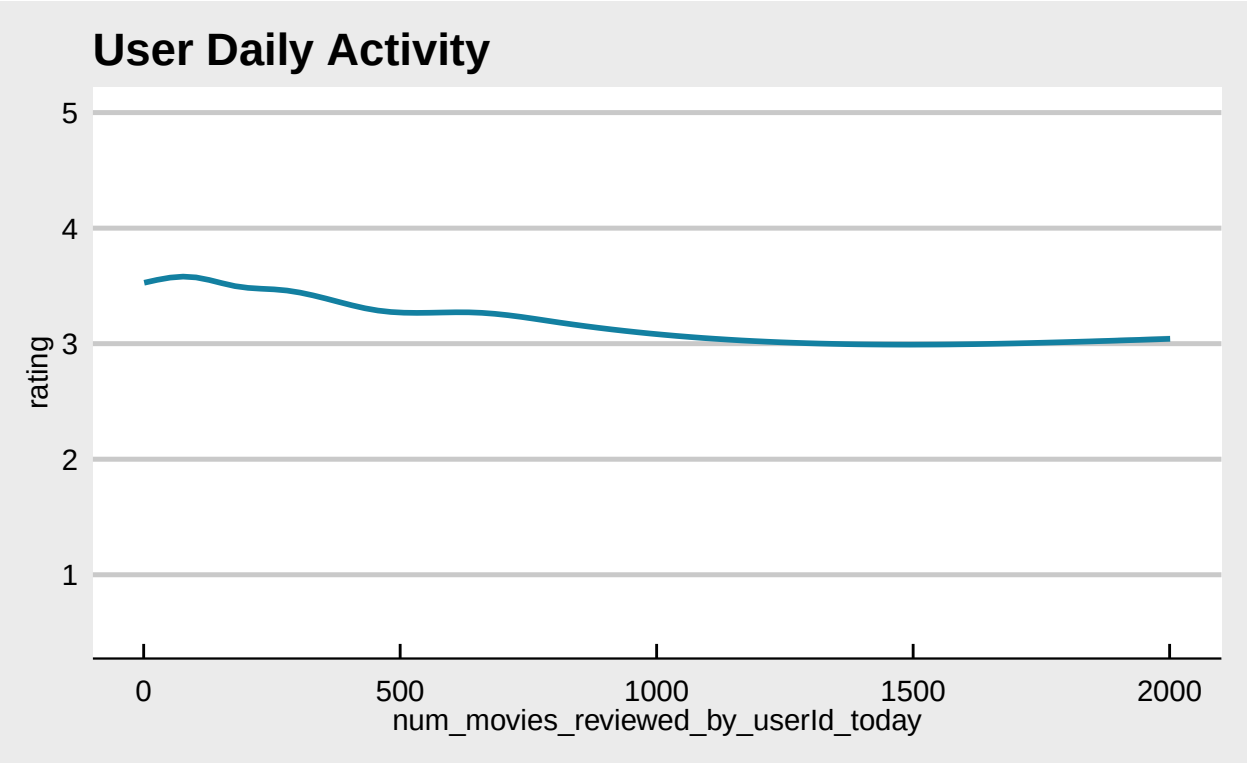
How does rating change related to the `movie's age` at the time of the review? For reference, the dashed horizontal line is the overall average rating. There are several data points that show reviews **before** the release date. Perhaps these are bad data points or were reviewed by critics prior to being released.



How does rating change related to `user's length of time reviewing` at the time of the review?



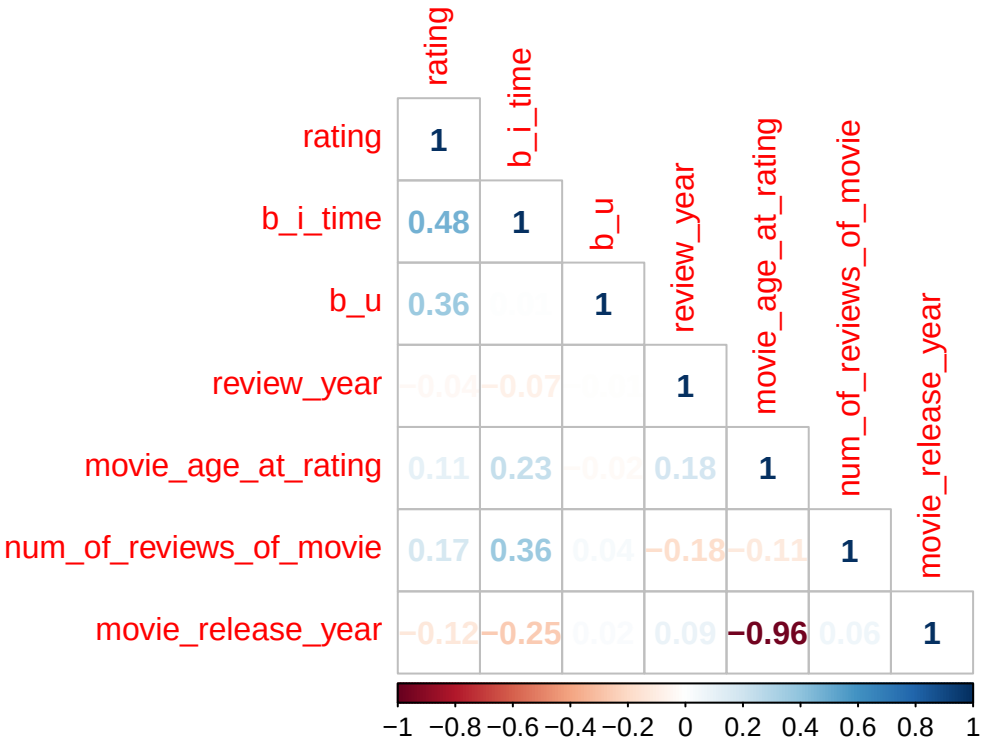
Is there a relationship between the `number of reviews a movie received` (popularity) and the rating of that movie? For reference, the dashed horizontal line is the overall average rating.



How does the number of movies reviewed by a user on a particular date (frequency) effect rating?

Proposed final predictors

Pairwise correlations among proposed predictors



First thing we notice is that the two strongest predictors of rating are the item and user biases. They don't correlate with each other so we will use both of them in any model. We see that the proposed predictors `movie_release_year` and `movie_age_at_rating` are strongly negatively correlated. This makes sense as movies released in years far in the past are older! Neither is better correlated with `rating` so we'll use `movie_age_at_rating` since it's a positive correlation and to me it seems more intuitive.

We'll use all proposed predictors except `review_year` and `movie_release_year` in our models.