HarvardX-PH125.9x: MovieLens Project Report

Predicting user ratings from MovieLens dataset

Regina Castra

2019-03-24

Contents

1 Introductory Summary **1**

2 Analysis **2**
2.1 Data Source . . . 2
2.2 Data Structure . . . 2
2.3 Data Check . . . 3
2.4 Data Analysis . . . 4

3 Modeling **7**
3.1 Model 0: Use the Average of Ratings as a Baseline Model . . . 7
3.2 Model 1: Movie Effect Model . . . 7
3.3 Model 2: Movie + User Effects Model . . . 8
3.4 Model 3: Regularized Movie Effect Model . . . 8
3.5 Model 4: Regularized Movie + User Effect Model . . . 9

4 Results **10**

5 Conclusion **10**

6 Appendix **11**
6.1 Analysis of Genre Effects . . . 11
6.2 Analysis of Release Year Effects . . . 12
6.3 Sessioninfo . . . 13

1 Introductory Summary

Recommendation systems use ratings that users have given items to make specific recommendations to users. In this project a movie recommendation system will be created, using the features in the MovieLens dataset to predict the potential movie rating a particular user gives for a movie. A 10M version of the MovieLens dataset will be used to provide a training and a validation dataset.

The goal is to train a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set. For a final test of the algorithm, movie ratings in the validation set have to be predicted as if they were unknown. RMSE will be used to evaluate how close the predictions are to the true values in the validation set. A target value of **RMSE <= 0.87750** is given.

Starting with a base model which utilizes just an average rating as a constant value, four different models taking movie and/or user effects into account are used to predict the ratings. Two of the presented models fulfill the required target value. The final model reaches a **RMSE** value of around **0.8648**.

2 Analysis

2.1 Data Source

The data is a 10M version of the MovieLens dataset and is downloaded from grouplens.org. By using the provided code from the course webpage "Create Test and Validation Sets" the data will be downloaded and separated into two datasets.

2.2 Data Structure

There are two datasets provided: **edx** as a training dataset for developing the recommendation algorithm and **validation** for a final test of the algorithm, i.e. predict movie ratings in the validation set as if they were unknown. Both datasets consist of 6 columns, the dataset **edx** has 9000061 rows, **validation** has 999993 rows. The attributes are of the following types and their names are self explanatory:

userId	movieId	rating	timestamp	title	genres
integer	double	double	integer	character	character

Each row represents a rating given by one user to one movie.

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi

The statistic values show, that the two provided datasets are quite similar: Statistic values of dataset **edx**:

userId	movieId	rating	timestamp	title	genres
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000061	Length:9000061
1st Qu.:18122	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character
Median :35743	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character
Mean :35869	Mean : 4120	Mean :3.512	Mean :1.033e+09	NA	NA
3rd Qu.:53602	3rd Qu.: 3624	3rd Qu.:4.000	3rd Qu.:1.127e+09	NA	NA
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA

Statistic values of dataset **validation**:

userId	movieId	rating	timestamp	title	genres
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:999993	Length:999993
1st Qu.:18127	1st Qu.: 653	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character
Median :35719	Median : 1835	Median :4.000	Median :1.036e+09	Mode :character	Mode :character
Mean :35878	Mean : 4121	Mean :3.512	Mean :1.033e+09	NA	NA
3rd Qu.:53649	3rd Qu.: 3633	3rd Qu.:4.000	3rd Qu.:1.127e+09	NA	NA
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA

2.3 Data Check

Neither dataset **edx** nor **validation** contain missing values. The number of **NA** entries in the datasets are:

	userId	movieId	rating	timestamp	title	genres
edx	0	0	0	0	0	0
validation	0	0	0	0	0	0

The following table shows, how many distinct values of the predictors are available for the **edx** and the **validation** dataset:

	n_userIds	n_movieIds	n_titles	n_genres
edx	69878	10677	10676	797
validation	68531	9796	9795	769

In both datasets the number of **movieId** and **title** is not the same: There seems to be one **title** with two **movieId**s. It turns out that the movie with the **title** "War of the Worlds (2005)" has two different **movieId**s in the **edx** dataset as well as in the **validation** dataset:

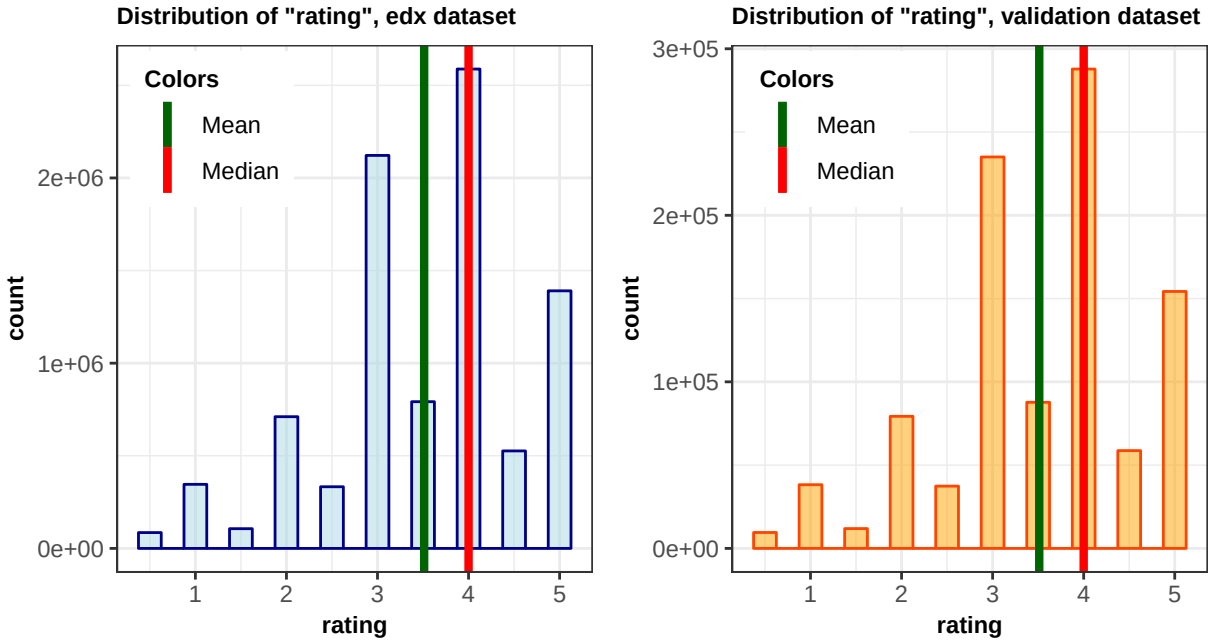
movieId	title	genres
34048	War of the Worlds (2005)	Action|Adventure|Sci-Fi|Thriller
64997	War of the Worlds (2005)	Action

The next step would be to combine these two entries into one unique entry. But it's probably not allowed to mess around with the data provided. Therefore, the two datasets remain untouched. In the following, a movie will be identified by its **movieId**. The **title** is for information only. From now on, it is thought to have two different movies with **movieId**s 34048 and 64997, even though these two **movieId**s have the same **title**.

2.4 Data Analysis

2.4.1 Data Analysis of Outcome “rating”

Plotting a histogram of the outcome **rating** for each of the given datasets reveals that both show similar distributions. Furthermore the half-number-ratings are less common than the whole-number-ratings.

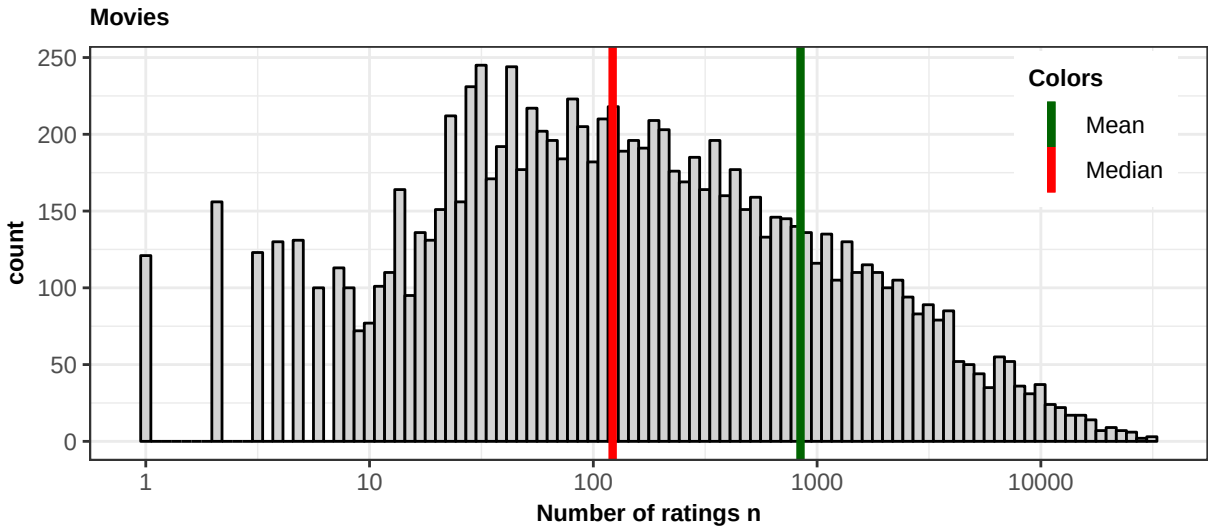


The distributions show 10 distinct values of **rating**, beginning with 0.5 in steps of 0.5 till max rating of 5 (inclusive):

rating	n_edx	n_validation	% of n_edx	% of n_validation
0.5	85420	9568	0.9	1.0
1.0	345935	38245	3.8	3.8
1.5	106379	11899	1.2	1.2
2.0	710998	79308	7.9	7.9
2.5	332783	37395	3.7	3.7
3.0	2121638	235038	23.6	23.5
3.5	792037	87727	8.8	8.8
4.0	2588021	287829	28.8	28.8
4.5	526309	58713	5.8	5.9
5.0	1390541	154271	15.5	15.4

2.4.2 Data Analysis of Predictor “movieId”

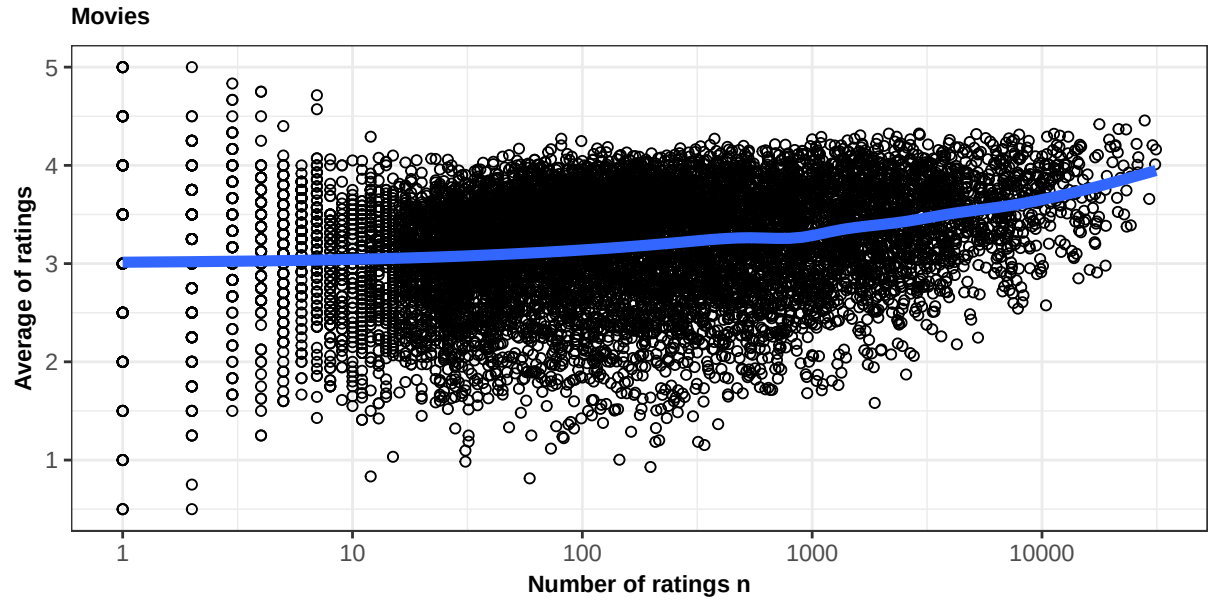
Some movies get rated more than others as shown in a histogram:



Following table lists the movies, which got the most ratings:

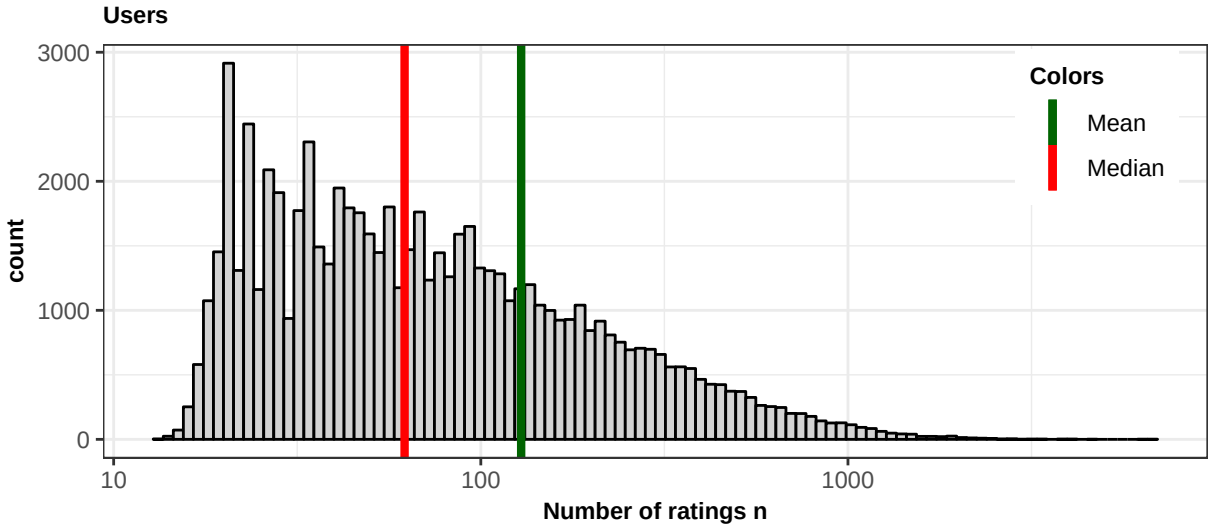
movieId	title	n	avg_rating
296	Pulp Fiction (1994)	31336	4.161731
356	Forrest Gump (1994)	31076	4.010265
593	Silence of the Lambs, The (1991)	30280	4.205086
480	Jurassic Park (1993)	29291	3.658189
318	Shawshank Redemption, The (1994)	27988	4.456928
110	Braveheart (1995)	26258	4.083194
589	Terminator 2: Judgment Day (1991)	26115	3.928394
457	Fugitive, The (1993)	26050	4.005969
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25809	4.218567

Movies with a high number of ratings tend to have higher ratings:

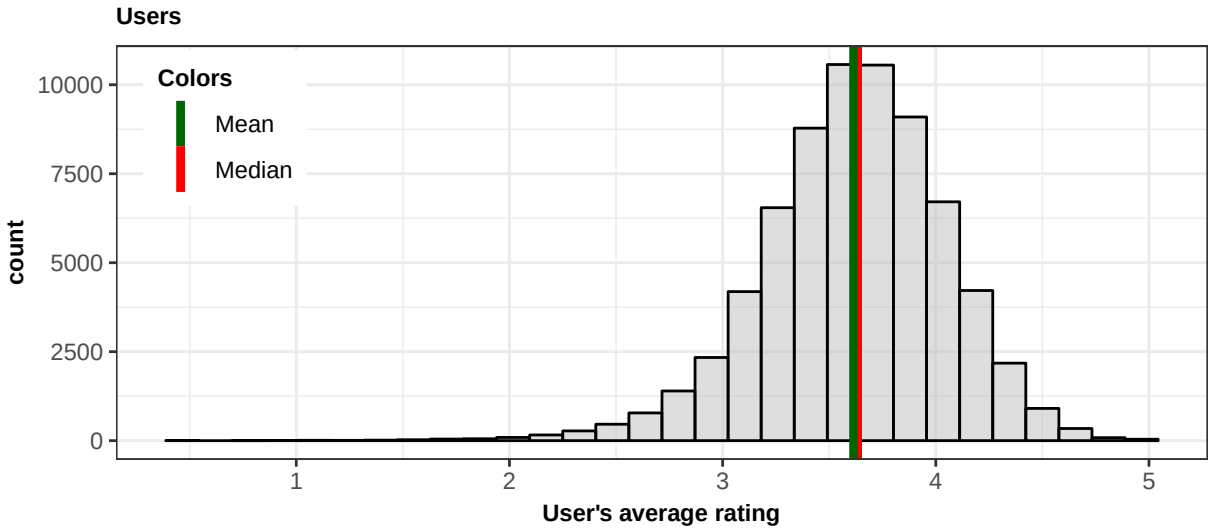


2.4.3 Data Analysis of Predictor “userId”

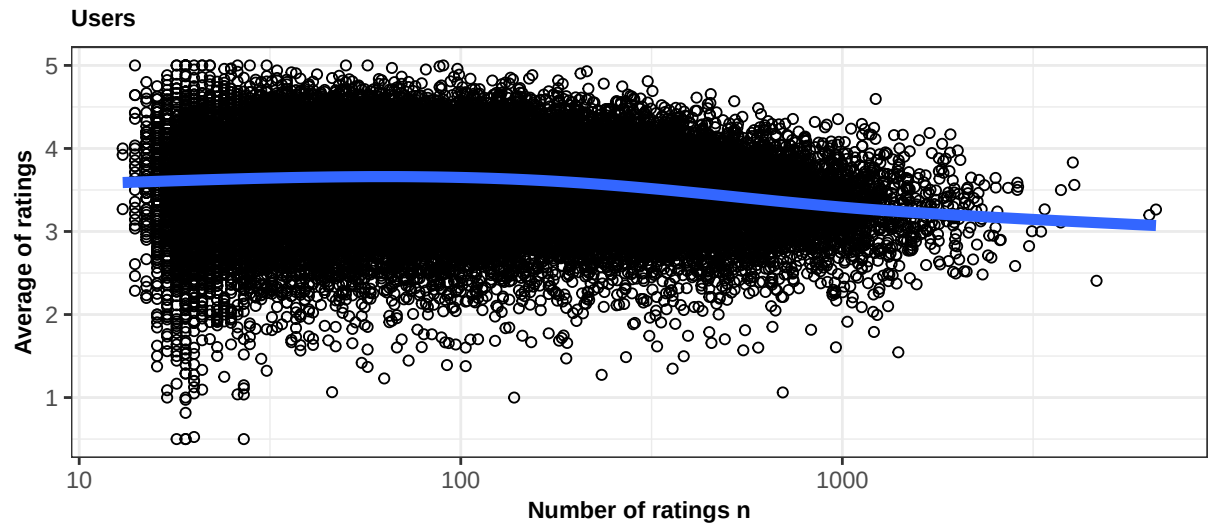
Some users are more active than others at rating movies:



Looking at the users’ average ratings, a substantial variability across users is noticeable:



There seems to be no significant correlation between average rating and number of ratings with respect of users:



3 Modeling

In order to compare different models a loss-function that computes the Residual Mean Squared (RMSE) is introduced. Using $y_{u,i}$ as the rating for movie i by user u and denoting our prediction with $\hat{y}_{u,i}$, the RMSE is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

with N being the number of user/movie combinations and the sum occurring over all these combinations. It is the typical error someone makes when predicting a movie rating. If this number is larger than 1, it means the typical error is larger than one star.

3.1 Model 0: Use the Average of Ratings as a Baseline Model

The simplest possible recommendation system is to predict the same rating for all movies regardless of user. A model that assumes the same rating for all movies and users with all the differences explained by random variation would look like

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

with ϵ_i independent errors sampled from the same distribution centered at 0 and μ the "true" rating for all movies. The estimate that minimizes the RMSE is the least squares estimate of μ and, in this case, is the average of all ratings.

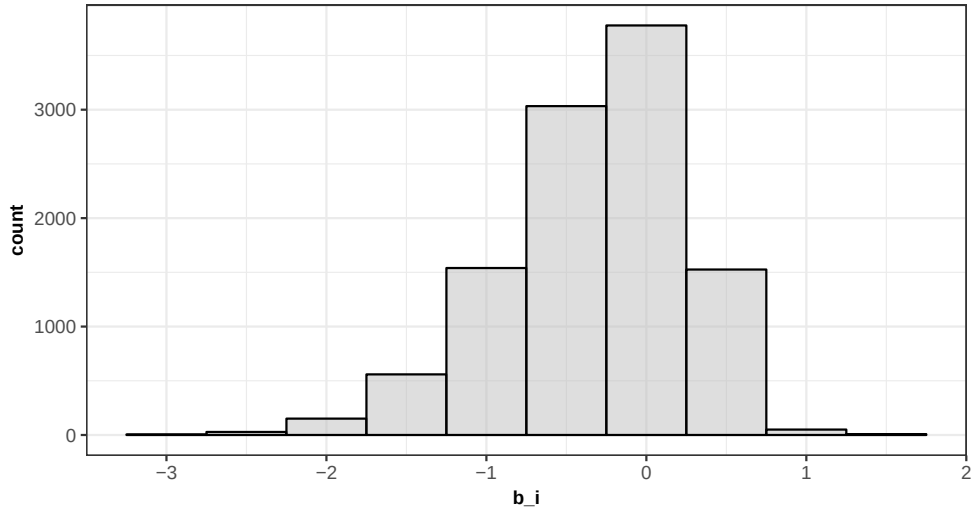
For further information, please refer to Rafael A. Irizarry's book Introduction to Data Science (2019-03-17).

3.2 Model 1: Movie Effect Model

Some movies are just generally rated higher than others. That different movies are rated differently, is confirmed by data. The previous model can be modified by adding the term b_i to represent average ranking for movie i:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

Using this model the distribution of b_i will look like the following:



For further information, please refer to Rafael A. Irizarry's book Introduction to Data Science (2019-03-17).

3.3 Model 2: Movie + User Effects Model

As already shown in the **Analysis** chapter, there is a substantial variability across users. To take this into account, the model can be further improved:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

where b_u is a user-specific effect.

For further information, please refer to Rafael A. Irizarry's book Introduction to Data Science (2019-03-17).

3.4 Model 3: Regularized Movie Effect Model

Looking at the data it can be shown that some movies are only rated by few users. Some of these users gave movies very high or very low rates. With just a few users, we have more uncertainty. Therefore, larger estimates of b_i, negative or positive, are more likely. By using regularization it is possible to penalize large estimates that are formed using small sample sizes. It is intended to fit the model

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

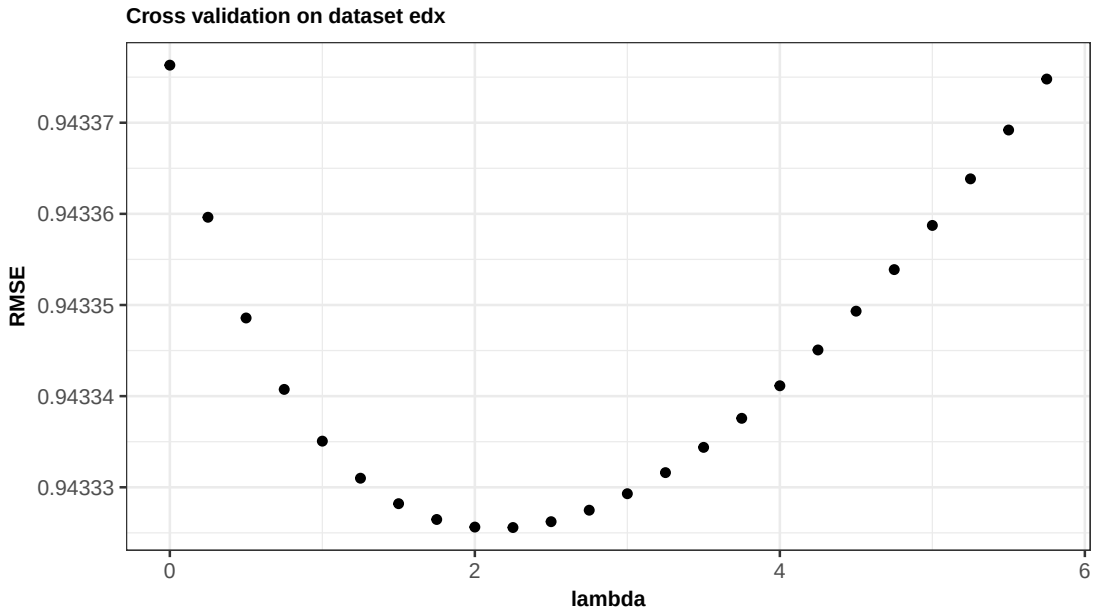
Instead of minimizing the least square equation, an equation that adds a penalty will be minimized:

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i)^2 + \lambda\sum_i b_i^2$$

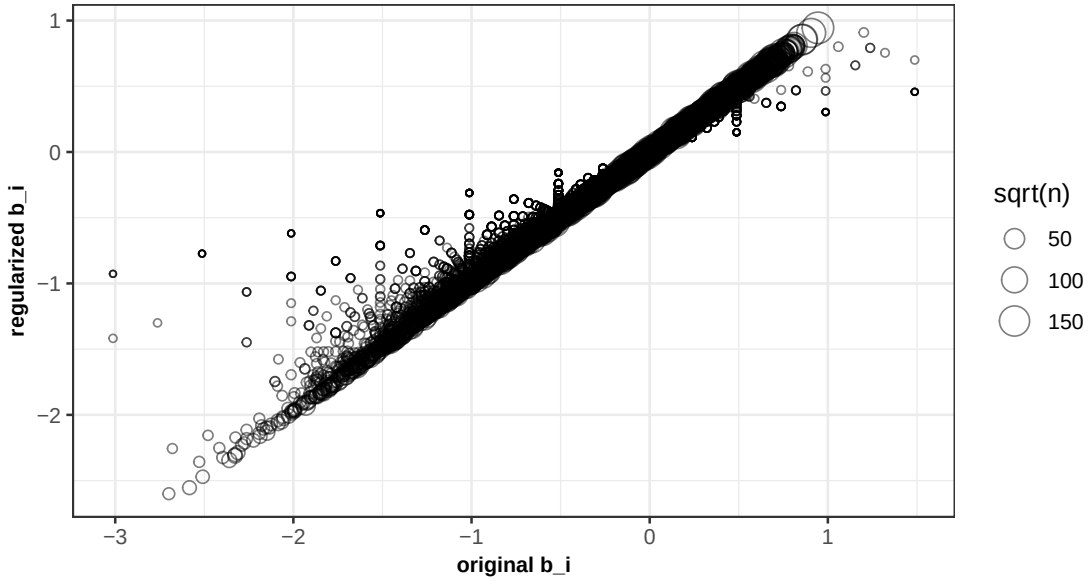
The first term is just least squares and the second is a penalty that gets larger when many b_i are large. By using calculus it can actually been shown that the values of b_i that minimize this equation are:

$$\hat{b_i}(\lambda) = \frac{1}{\lambda + n_i}\sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu})$$

where n_i is the number of ratings made for movie i and λ is the penalty. Penalty λ is a tuning parameter. It can be determined by using cross validation just on the dataset **edx**. Therefore the dataset **edx** is devided into a train set (90%) and a test set (10%). Following plot shows the resulting RMSEs for several λ used on the test set.



For dataset **edx** best value of RMSE can be achieved with $\lambda = 2.25$. Using this value, a plot of the regularized estimates versus the least squares estimates will show how the estimates shrink:



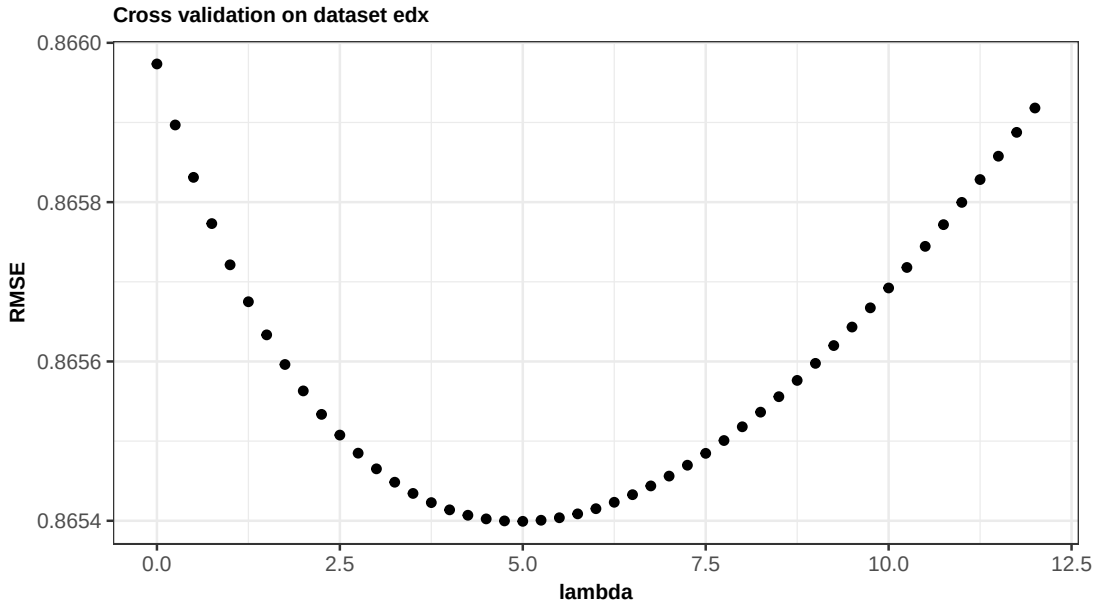
For further information, please refer to Rafael A. Irizarry's book Introduction to Data Science (2019-03-17).

3.5 Model 4: Regularized Movie + User Effect Model

Regularization can be used for the estimate of user effects as well. Following equation has to be minimized:

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2)$$

By cross validation on the dataset **edx** a different λ delivers optimum RMSE:



For dataset **edx** best value of RMSE can be achieved with $\lambda = 5$ for this kind of model.

For further information, please refer to Rafael A. Irizarry's book Introduction to Data Science (2019-03-17).

4 Results

Using above models on the **validation** dataset leads to following results:

Method	RMSE
M0: Just the average	1.0606506
M1: Movie Effect Model	0.9437046
M2: Movie + User Effects Model	0.8655329
M3: Regularized Movie Effect Model	0.9436523
M4: Regularized Movie + User Effect Model	0.8649887

Two Models create RMSE values below the given target value of **RMSE <= 0.87750**. In both cases movie and user effects are taken into account.

The **M4: Regularized Movie + User Effect Model** achieves the lowest **RMSE** value of around **0.865**.

5 Conclusion

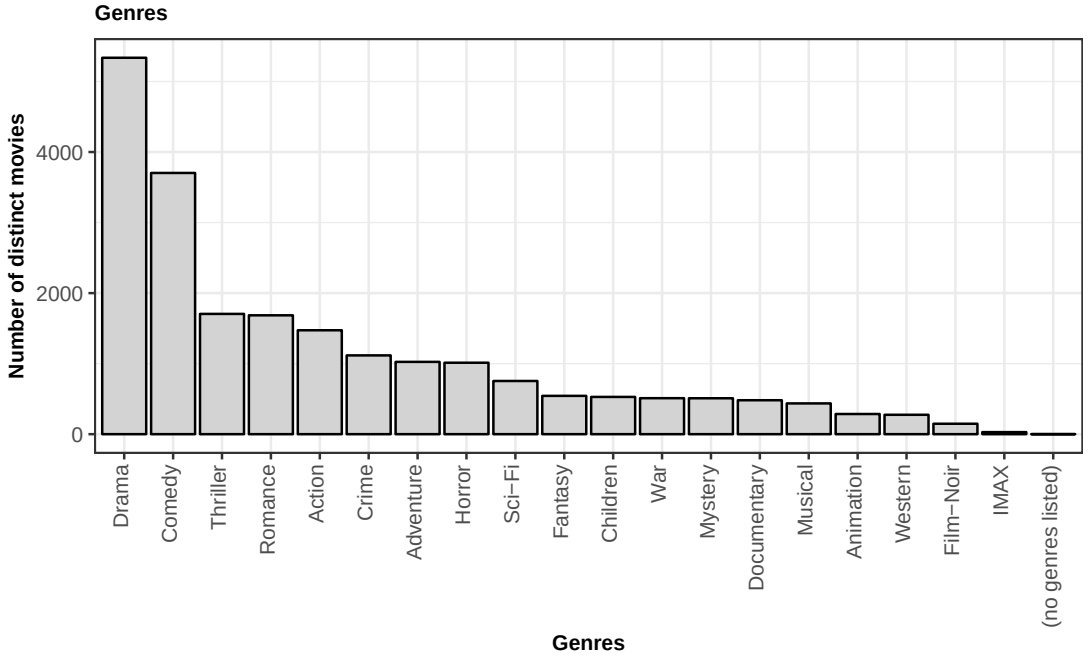
In this project a movie recommendation system was created, using the features in the **MovieLens** dataset to predict the potential movie rating a particular user gives for a movie. A machine learning algorithm was implemented using a model which takes movie and user effects into account and predicts movie ratings. The model which best fulfilled the given target achieves a **RMSE** value of around **0.865**.

It is likely that much more sophisticated methods would achieve better RMSE values. For example genre effects as well as the release year of the movie could be considered. A first analysis presented in the **Appendix** chapter show that these features would possibly improve the recommendation algorithm. The implementation is left to future work.

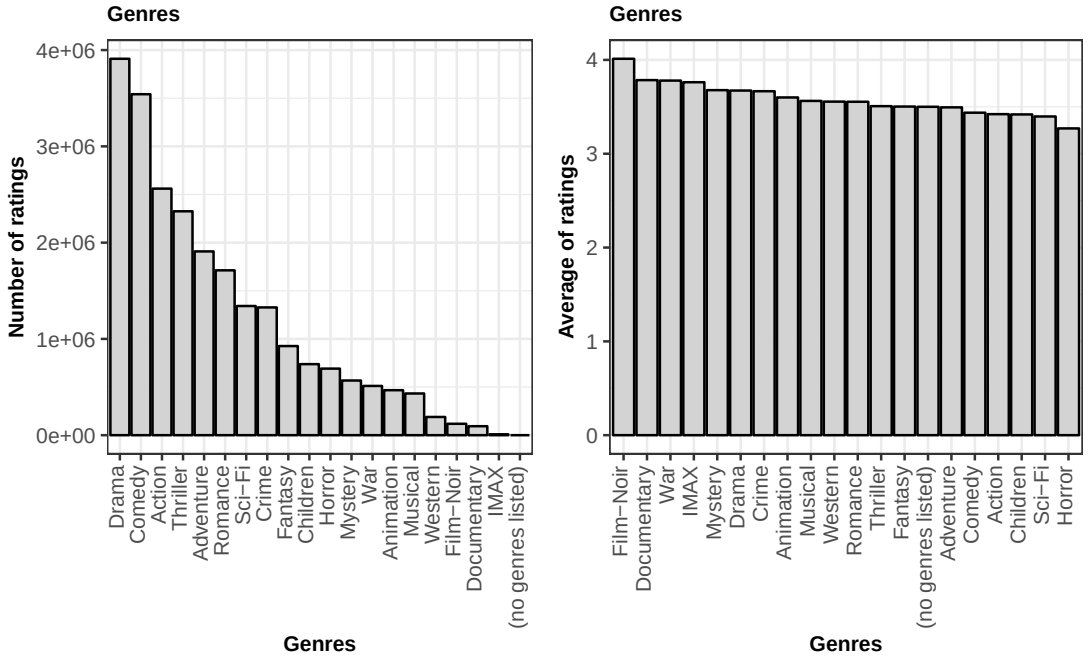
6 Appendix

6.1 Analysis of Genre Effects

Even if the **genre**s were not considered for the recommendation modeling, it is worth to check, if they can be used as a predictor:



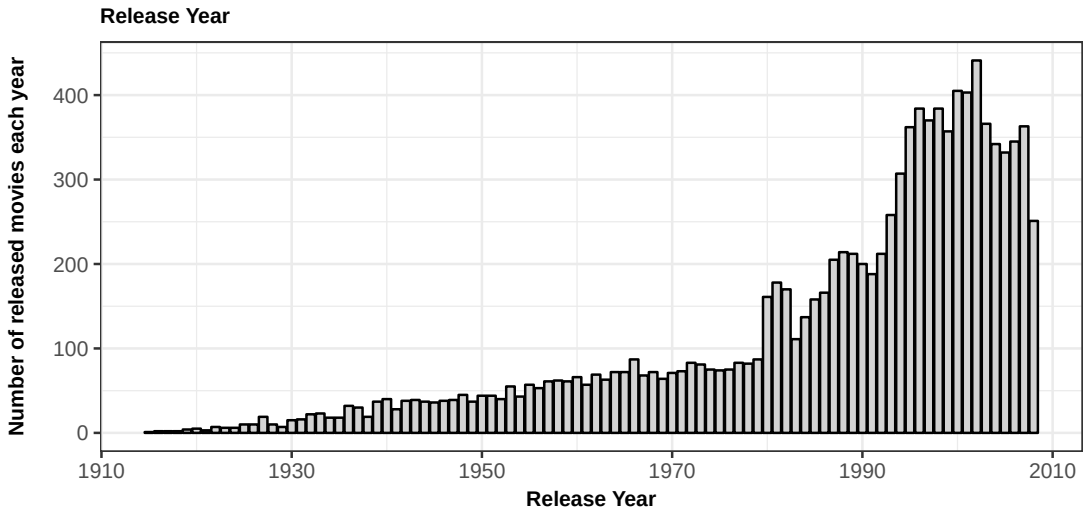
The distribution shows that movies are tagged in many cases with the genres Drama and/or Comedy. Therefore it is no wonder that movies belonging to these two genres also got the largest number of ratings:

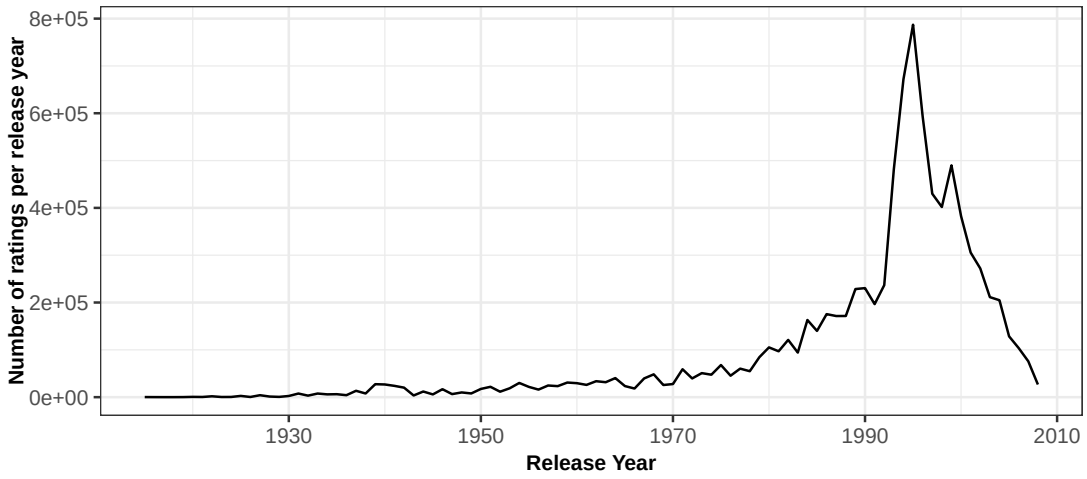


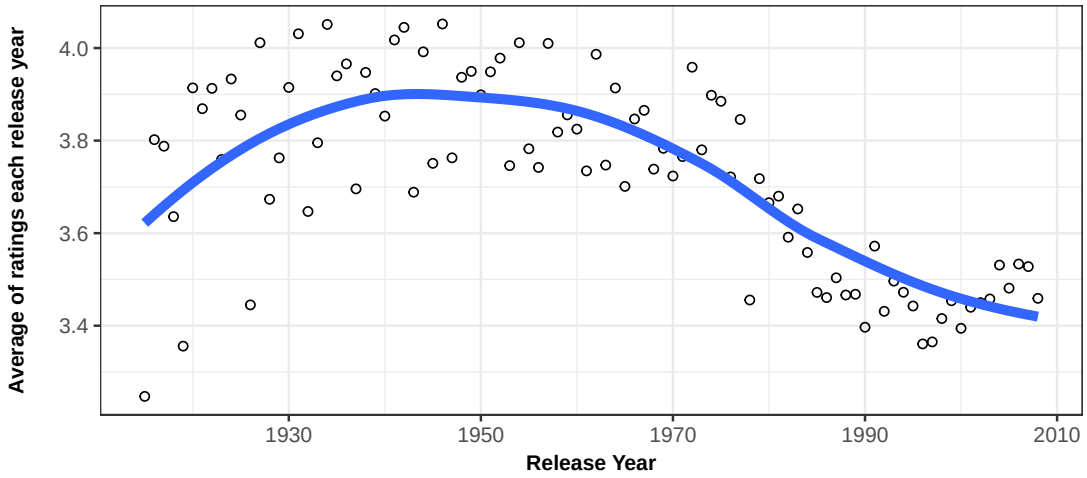
There is a connection of “average of ratings” and the genres which probably could be used to improve the recommendation algorithm.

6.2 Analysis of Release Year Effects

The movie title also contains the information of the release year. This information is not yet considered for the recommendation modeling.







It turns out that since 1940 the newer the movie the lower its rating. This insight could be used to improve the recommendation algorithm.

6.3 Sessioninfo

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.6 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## locale:
##  [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=de_DE.UTF-8        LC_COLLATE=de_DE.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=de_DE.UTF-8
##  [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] knitr_1.22      gridExtra_2.3   caret_6.0-84    lattice_0.20-38
##  [5] forcats_0.4.0   stringr_1.4.0   dplyr_0.8.0.1   purrr_0.3.2
##  [9] readr_1.3.1     tidyr_0.8.3     tibble_2.1.1    ggplot2_3.1.1
## [13] tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1         lubridate_1.7.4    class_7.3-15
##  [4] zeallot_0.1.0      utf8_1.1.4         assertthat_0.2.1
##  [7] digest_0.6.18      ipred_0.9-9        foreach_1.4.4
## [10] R6_2.4.0           cellranger_1.1.0   plyr_1.8.4
## [13] backports_1.1.4    stats4_3.6.1       evaluate_0.13
## [16] highr_0.8          httr_1.4.0         pillar_1.4.0
## [19] rlang_0.3.4        lazyeval_0.2.2     readxl_1.3.1
## [22] data.table_1.12.2  rstudioapi_0.10    rpart_4.1-15
## [25] Matrix_1.2-18      rmarkdown_1.12     labeling_0.3
## [28] splines_3.6.1      gower_0.2.0        munsell_0.5.0
## [31] broom_0.5.2        compiler_3.6.1     modelr_0.1.4
## [34] xfun_0.6           pkgconfig_2.0.2    mgcv_1.8-31
## [37] htmltools_0.3.6    nnet_7.3-12        tidyselect_0.2.5
## [40] prodlim_2018.04.18 codetools_0.2-16   fansi_0.4.0
## [43] crayon_1.3.4       withr_2.1.2        ModelMetrics_1.2.2
## [46] MASS_7.3-51.4      recipes_0.1.5      nlme_3.1-142
## [49] jsonlite_1.6       gtable_0.3.0       magrittr_1.5
## [52] scales_1.0.0       cli_1.1.0          stringi_1.4.3
## [55] reshape2_1.4.3     timeDate_3043.102  xml2_1.2.0
## [58] vctrs_0.1.0        generics_0.0.2     lava_1.6.5
## [61] iterators_1.0.10   tools_3.6.1        glue_1.3.1
## [64] hms_0.4.2          survival_3.1-8     yaml_2.2.0
```

```
## [67] colorspace_1.4-1   rvest_0.3.3        haven_2.1.0
```