DS Capstone Report

Grzegorz Wierzchowski

Compiled: 2019-06-03

Contents

Chapter 1

Prerequisites

The document is prepared in **R Markdown** language and should be compiled using the **bookdown** package. This package can be installed from CRAN or Github:

```r
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Source code for this document is available in GitHub repository: https://github.com/gwierzchowski/. Doucument with its source code is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License.

Included R code launches installation of necessary R packages during document build process if they are not already installed. But to start build process those components must be installed:

- R base program (version 3.5 or higher)
- R package `bookdown` (see instruction above)
- R Studio IDE (recommended)

To compile this document to PDF, you also need *LaTex* with most commonly used packages. MS Windows users are recommended to install *TinyTeX* (which includes *XeLaTeX*): https://yihui.name/tinytex/. Linux users can use *Texlive* contained in repositories of most Linux distributions. Mac users - please refer to `bookdown` package and `pandoc` program instructions.

In order to properly install several R packages, C compiler must be avaialble in the system (may require additional installation on Windows systems) as well as some additional libraries or `-dev` packages may be required. Please refer to particular packages' documentation (or search the web) in case of problems.

Instruction how to build PDF file from sources:

- Download document sources from https://github.com/gwierzchowski/.
- Open main project file: `Capstone.Rproj` with R Studio.
- Open file: `_output.yml` and un-comment either: `latex_engine: xelatex` (on Windows) or `latex_engine: pdflatex` (on Linux).
- Use **Build Book** option avaialble in **build** tab (on right upper panel in default R Studio layout). Output document is stored in `_book` sub-folder of main project folder.

Notes:

- Source repository contain the file `RData\genres_users_r.RData` which stores heavy calculations results, and is placed to reduce build time considerably. With this file, document build takes about 1h, without this file (in such case all calculations are performed) it may take even 9-10h.

- The file: `02-exploration.Rmd` contains switch: `FirstTime <- TRUE`. Turn it to `FALSE` if you run this buld for the second time - to avoid already done download and extraction of data files.
- I performed calculations on Linux system with memory: 16GB RAM.
- Used R version: 3.6.0 with up-to-date packages as of May 2019.

Expected source format for the *Capstone Course* is `.Rmd` file, so I provide such file as concatenation of all chapters (the build intemediate file). It can also be compiled directly, but to get full content with correct references I recommend above process that involves `bookdown` package.

To keep document compact, I do not list all accompanied sorce code, only the most important parts. Full source code which is being run during this document build is avaialble in source files in: https://github.com/gwierzchowski/.

Chapter 2

Introduction

2.1 Problem definition

This report ia a required document for the final part of *Data Science Professional Certificate Program* provided by HarwardX (Irizarry, 2018a). The excercise consist in creating a program (also called a model) which can predict a rating which particular user most probably would assign to parcicular not yet watched movie, based on their historical ratings for other movies or other users' ratings of this movie.

In practice we will use data set provided by *GroupLens research lab* (Sou, 2018) which contains 10 million ratings applied to 10,681 movies by 71,567 users. Data set was randomly split into two separate sets: training set (refered in later code as `edx`) - about 90% of data, and validation set (`validation` in code) - 10% of data. In addition splitting algorithm ensured that all users and movies contained in validation set have some data in training set. Training set was used to develop the rating prediction program and tune its parameters. Validation set was apparently used to check how good is particular version of program (the model) - i.e. it was only used to select the best model (the one nominated as final solution in section 5.1) among developed models. I honor the rule that none of validation set data can affect on any algorithms or internal parameters of any considered model.

The goal of excercise is to develop program which will calculate ratings equal or as close as possible to real ratings given by users. In practice, to measure (rate) concrete program - we will let it to calculate ratings for users and movies contained in validation set and compare those values with ratings stored in validation set. According rules established by Course Vendor the value of RMSE (root mean squared error) is considered as measure of how predictions are close to real values (the *loss function*). It is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

Where N is number of records in validation set, $\hat{y}$ is predicted value and y is real value in validation set. So I will be trying to minimize this function. Values below 0.87 are considered as good by Course Vendor. In addition to this basic measure, I will be also calculating an overall accuracy as second, helper measure. It is defined as:

$$\text{ACCU} = \frac{\text{number of exact predictions}}{\text{number of all predictions}}$$

In many places in this report I will be using modified code from (Irizarry, 2018b). Because this is basic book for entire *Data Science Series Program* I will refrain from citing this book at every place in following text.

2.2 Work plan

Before I started working on problem I had following rough plan.

1. At first try to use basic technicks like movie and user effect and see what results they bring.
2. Then try to use additional information given in data files not explicitly used in the course and book material - like movie genres and movie year of issue (embedded in most movie titles).
3. Than, to check if regularization and introducing panalties shrinking less popular moviews to average would have any positive effect.
4. Then, would see if more advanced technics like matrix factoirization could be applied to our problem. I suspect that there could be performance of memory problems with such relativly big data set, whole working on personal PC.
5. Then some internet based research could be done to check if there are some other approaches which could be applied to current problem.

Chapter 3

Data Exploration

3.1 Data retrieval and preparation

In order to prapare data for building recomendation system I performed following operations:

- downloaded MovieLens 10M data file as zip-file
- extracted necessary files
- loaded data to data-frames (`movies` and `ratings`)
- joined those 2 data frames into one data-frame: `movielens`
- splited data into training (`edx`) and validation (`validation`) sets at rate 90/10.
- tuned sets in the way that validation data set contains only 'known' users and movies (i.e. the ones that have at least one record in traing set)
- removed spare objects enabling garbage collector to free mamory

There are two modifications in my data pre-processing procedures in compare to ones published by Course Provider:

- I noticed that there is year of the movie's premiere given in braces inside all the titles, like in "Toy Story (1995)". I have extracted it to seperate field: `year`, and then further factorize it as field `year_class`.
- To reduce memory usage during processing I excluded movie titles from joined data-frame. I do not use titles during rating predictions.

3.2 Initial analysis of training data set

In this chapter I present some basic statistics and analysis of training data set performed in order to detarmine predictors that could be usable for inproving acuracy of rating prediction model which I will be building.

3.2.1 Global

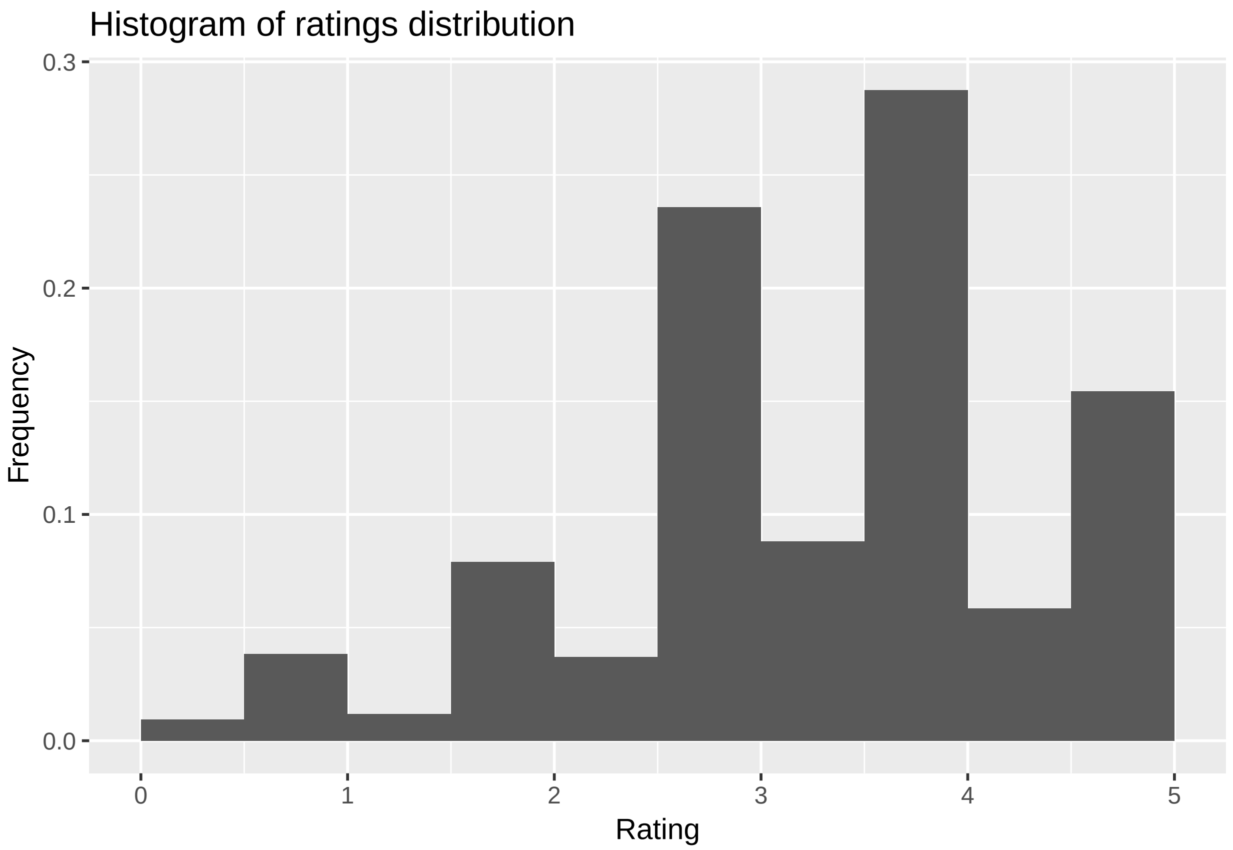
Here is how our raw data looks like:

Table 3.1: A sample of training data

userId	movieId	rating	timestamp	genres	year_class
1	122	5	838985046	Comedy|Romance	early 90's
1	185	5	838983525	Action|Crime|Thriller	late 90's
1	231	5	838983392	Comedy	early 90's
1	292	5	838983421	Action|Drama|Sci-Fi|Thriller	late 90's
1	316	5	838983392	Action|Adventure|Sci-Fi	early 90's
1	329	5	838983392	Action|Adventure|Drama|Sci-Fi	early 90's

Table 3.2: Basic characteristics of our traing data set

Measure	Value
Number of records	9000061
Total mean of ratings	3.51246397107753
Levels of ratings	0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5

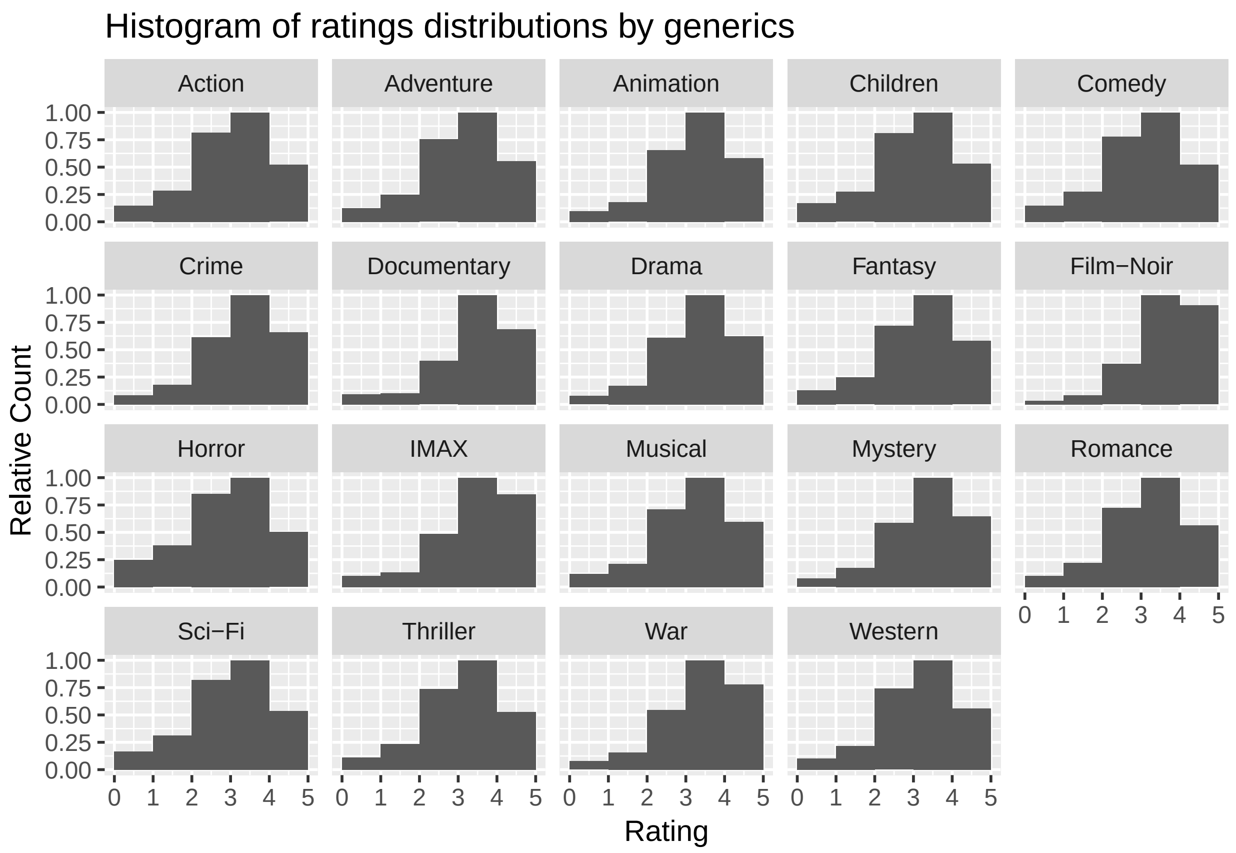


The distribution of ratings is not symmetrical. The value "4" as most frequent one and whole values are more frequent than fractions. Also distrubution of fraction-only values seems to be similar to distribution of whole values. This means that when we will be rounding predictions from floating values colculated by model to levels of actual ratings (which are in 0.5 steps) we will get better results if we take into consideration this fraction vs. whole disproportion instead of just simply rounding to nearest half-fraction values.

The fraction of half-stars in all ratings is equal: 0.2047684.

3.2.2 By Generics

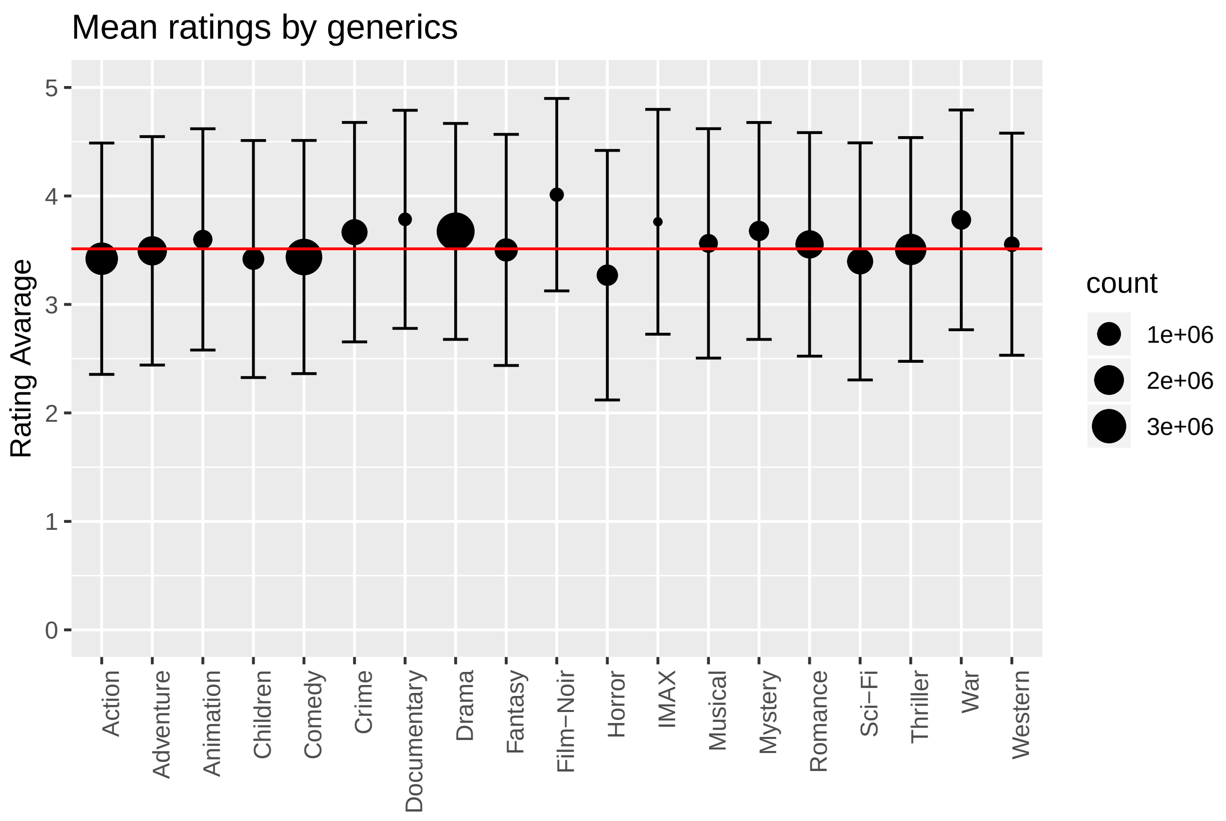
Generics assigned to movies were not extensivly used in example models provided during the course. My intuition was that it may play some substantial role when users are rating movies. So I decided to extract indivudual generics from combined vector attached to every movie and create ratings distribution historgam for every generic.



To make smaller histograms more clear, I made two changes in compare to previous histogram:

- ratings on histograms are groupped to whole values
- y-dimension is calculated in the way that highest bar is scalled to 1 and others are relative to it. In this way all are readable independently on number of actual ratings within each category.

For all catagories, the most frequent rating is “4”, but there are slight differencies when we look at hights of “3” and “5” bars (conpare for instance “Film-Noir” and “Horror”). Because this chart hides information about ratings counts in each catagory, I decided to do one more graph that shows this information and also how much avarage rating for each category diverge from total ratings avarage (marked below with red line).

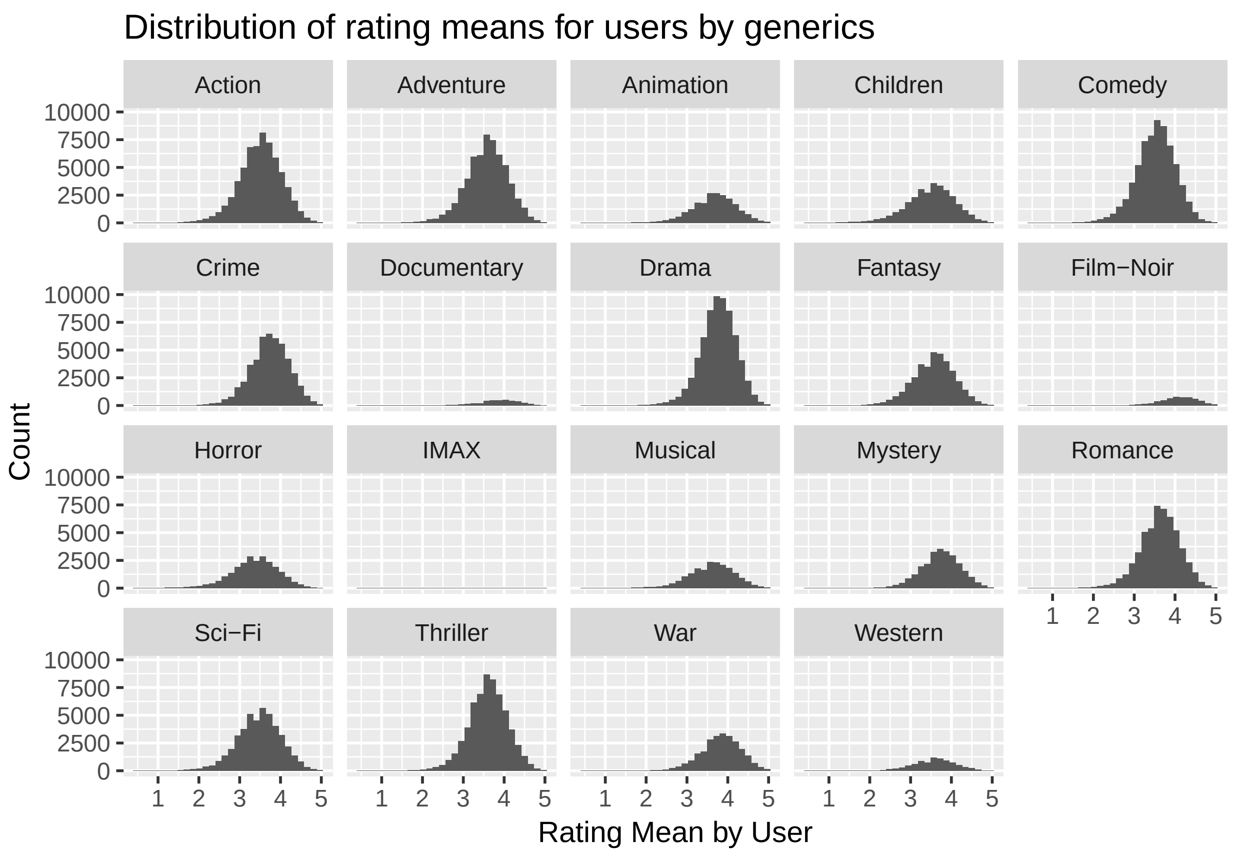


This chart shows that mean ratings in each category (espacially those more popular) does not differ that much from total mean, and also that variability within each generics is pretty much the same.

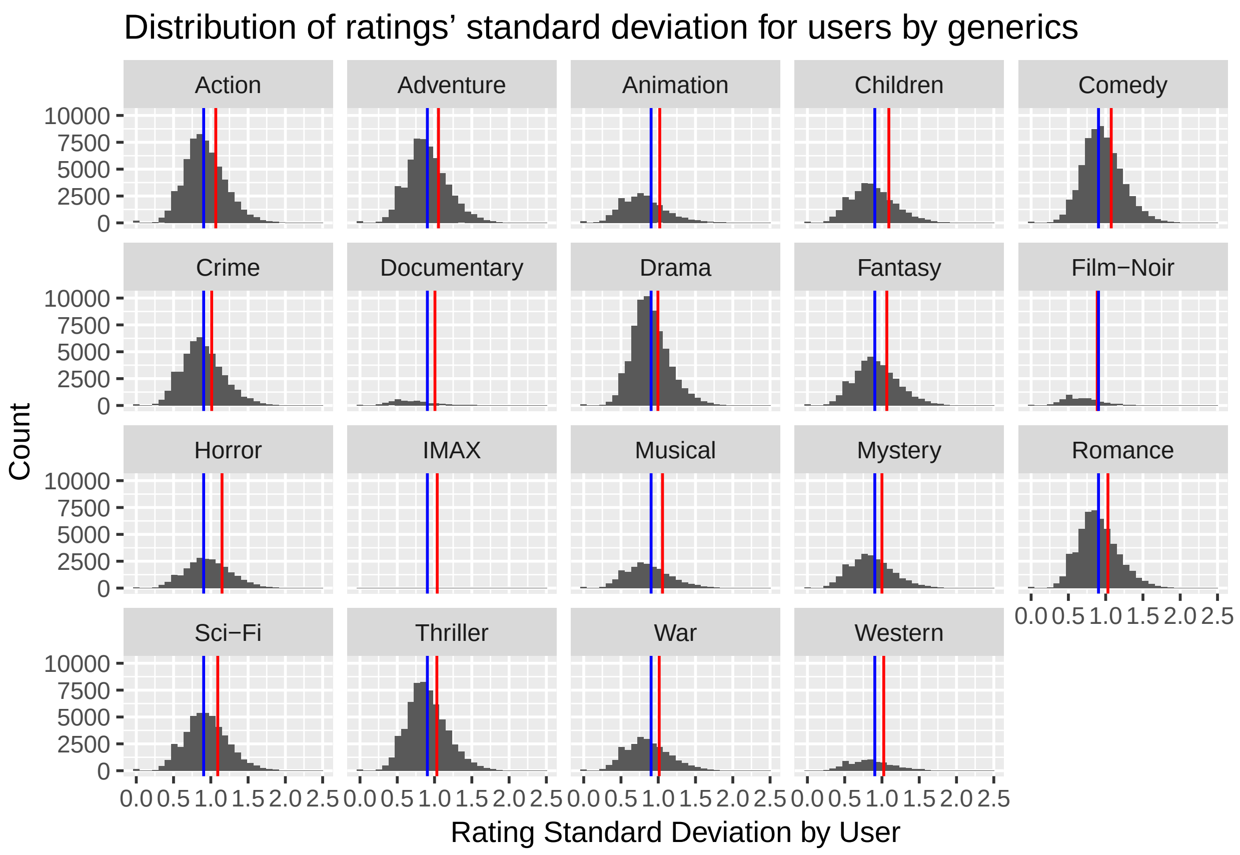
This little surpriced me bacause I expected that assigned rating is more related to generics (kind of movie). However after I thought a little, I realized that this was my auto-suggestion. I.e. I, most probably subconsciously expected that people (generally ?!?) like kind of movies that I personally like. But people are different and like different films. Otherwise it could happen that some movie catagories would dominate industry. Kind of interesting discovery. The other observation is that some kind of movies are in average higher rated that others, but are in the same time relatively rare - what could maybe be some pointer for producers. The last thing which is little strange are the extremas: "Film-Noir" - highest rated and "Horror" - lowest rated, while they are in my humble opinion a little bit similar in its nature. So there may be some different factor which influance the rating.

3.2.3 By Generics and Users

Following my findings described in prevoius section - that different users may like different kinds of movies I would like to check if it is visible in the data.



Below and next charts only take into consideration users, which rated at least 6 movies within particular category. Total avarages for each generic are shown on chart **??**. All distributions are more-less symetrical and looks normally. They are also not very steep, what means (???) that users are watching and rating movies from different categories.



The lines on above chart have following menaing:

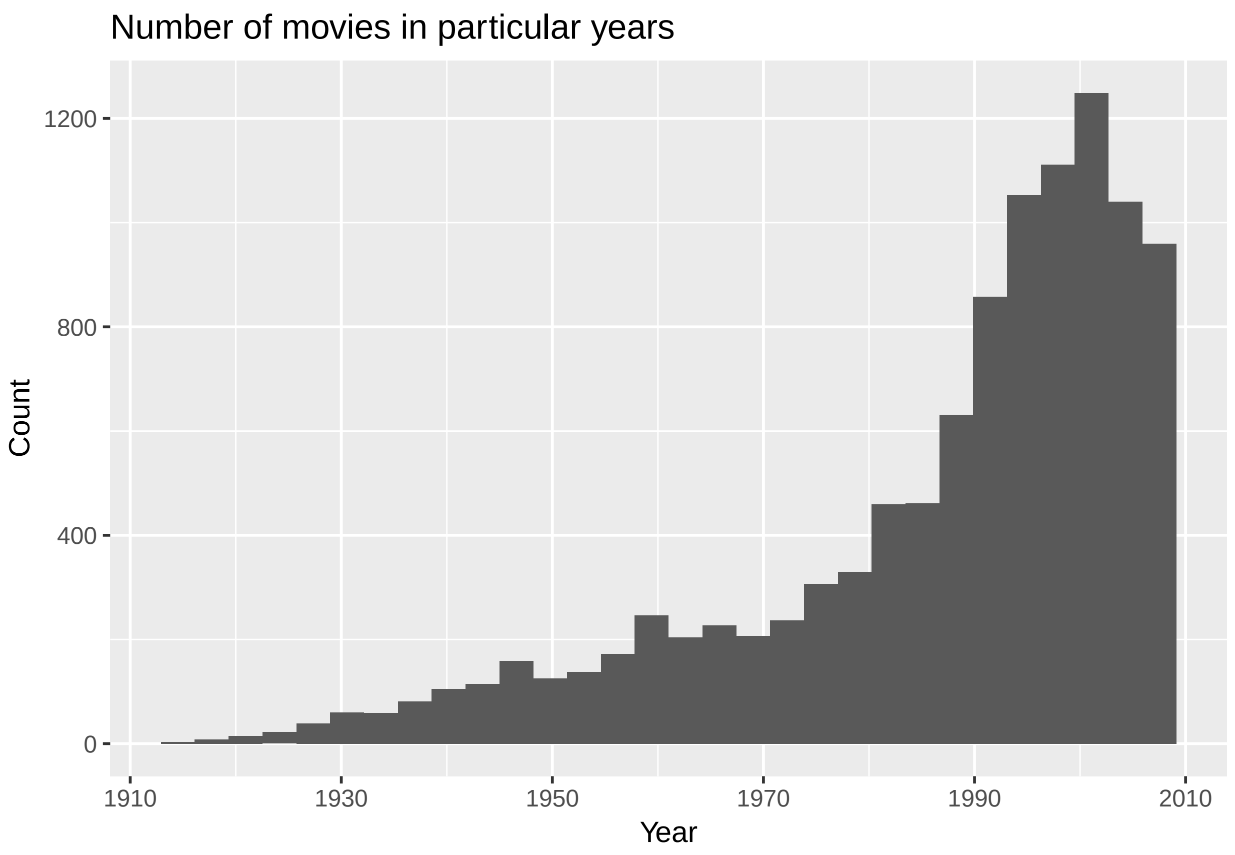
- red: common standard deviation of ratings for each category (also shown as error bars in chart **??**)
- blue: average standard deviation of ratings of one user for each category

Total standard deviation of all ratings is equal: 0.8635249.

All distributions are slightly asymetrically shifted to left, and in all cases the blue line is at left side of red. This means that ratings within each category are less variable for particular users than generally. This mens that movie generic togather with user ID may bring some information into the expected rating. So it may be sensible to take this factor into consideration in the prediction model.

3.2.4 By Year

Lets repeat similar analysis and try to use `year` data.

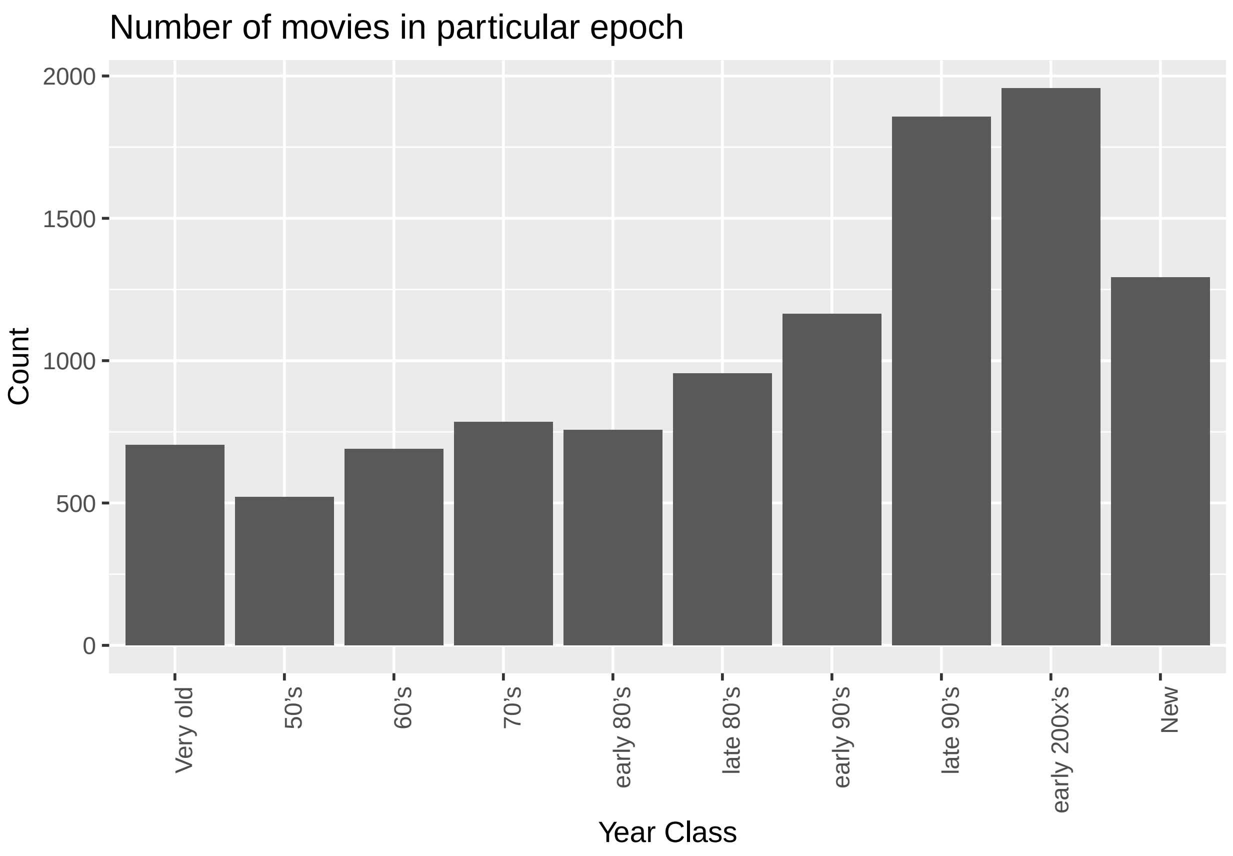


As expected year of movie production is distributed not evenly. So it may be hard to use raw year number in prediction model. So I revised initial data pre-processing routines and added factorized year class (years groped by 10 or 5 - further called “epoch”) instead of raw year.

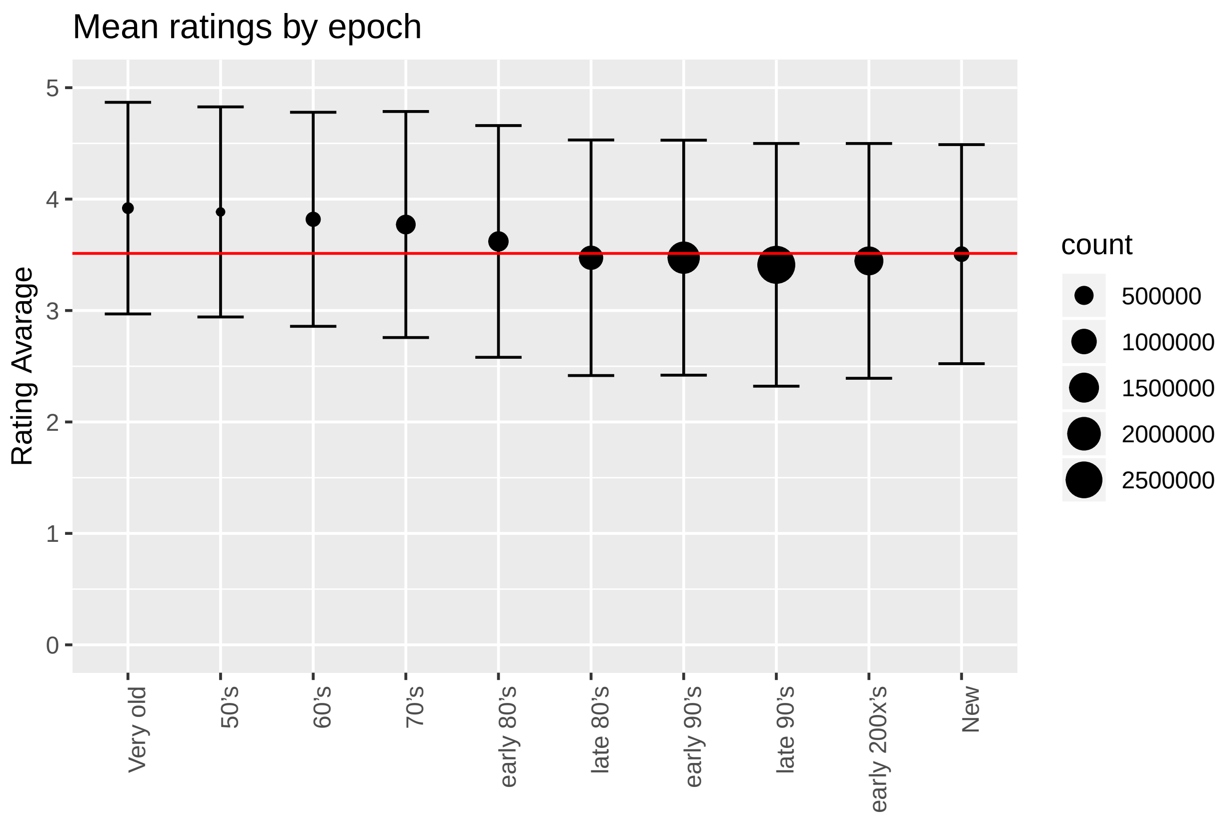
Here is my groupping function:

```r
clasify_year <- function(year) {
  if (year < 1950) {return ("Very old")}
  if (year < 1960) {return ("50's")}
  if (year < 1970) {return ("60's")}
  if (year < 1980) {return ("70's")}
  if (year < 1985) {return ("early 80's")}
  if (year < 1990) {return ("late 80's")}
  if (year < 1995) {return ("early 90's")}
  if (year < 2000) {return ("late 90's")}
  if (year < 2005) {return ("early 200x's")}
  "New"
}
```

After this change information about production year of movies is more evenly distributed.

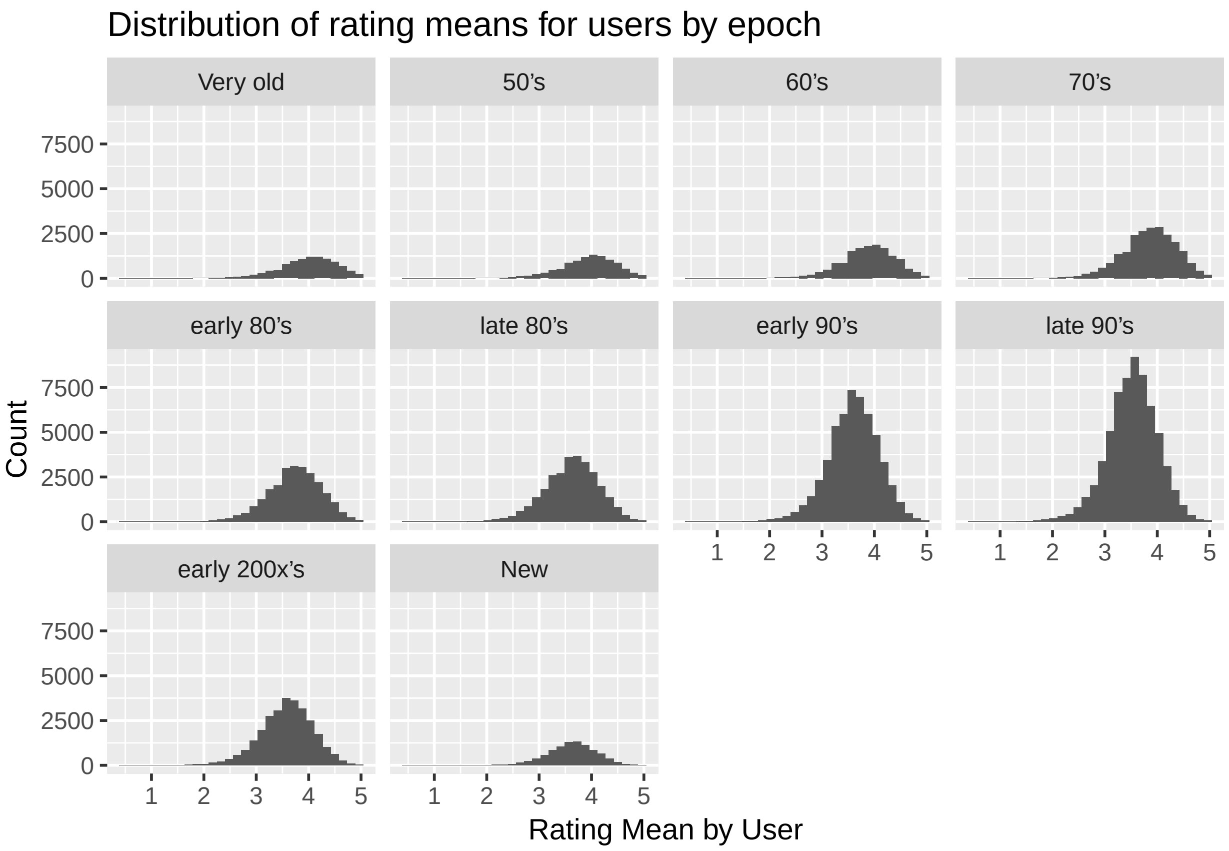


Let's create chart similar to **??**), but which will show how rating is changing for particular epochs. As on refered graph, red line marks total rating avarage.

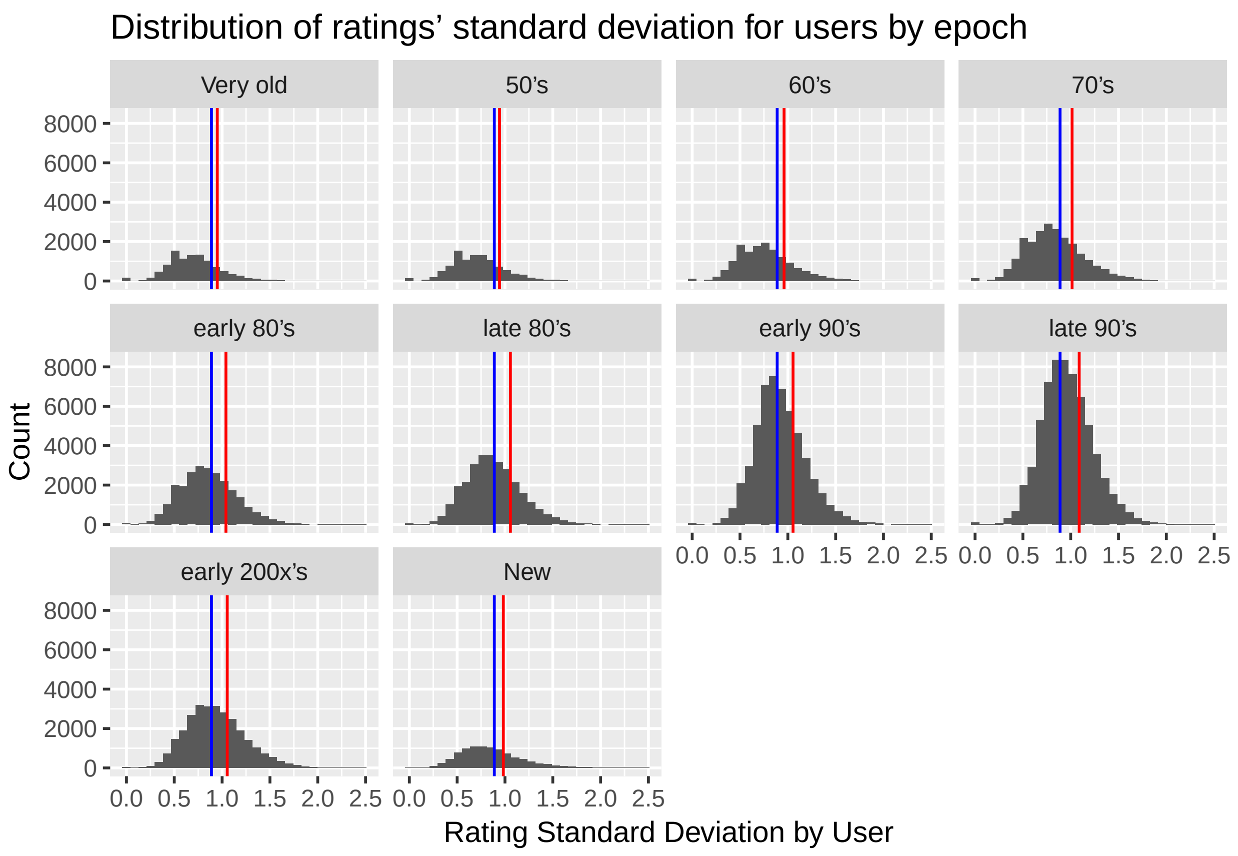


Note that counts on this and previous chart are for different entities. On chart **??**) count refers to *number of movies* in each epoch, while on above chart count stands for *number of ratings* of movie of particular epoch. It is evident from this chart that older movies tends to be rated higher, but in the same time they are also far less frequent. So total impact of epoch in correct prediction may not be that big. So I decided to continue analysis and check if maybe some users likes more movies from certain epoch.

3.2.5 By Year and User



Below and next charts only take into consideration users, which rated at least 6 movies within particular epoch. Distributions of youngest movies are more-less symetrical and looks normally, while older ones are little shifted to right.



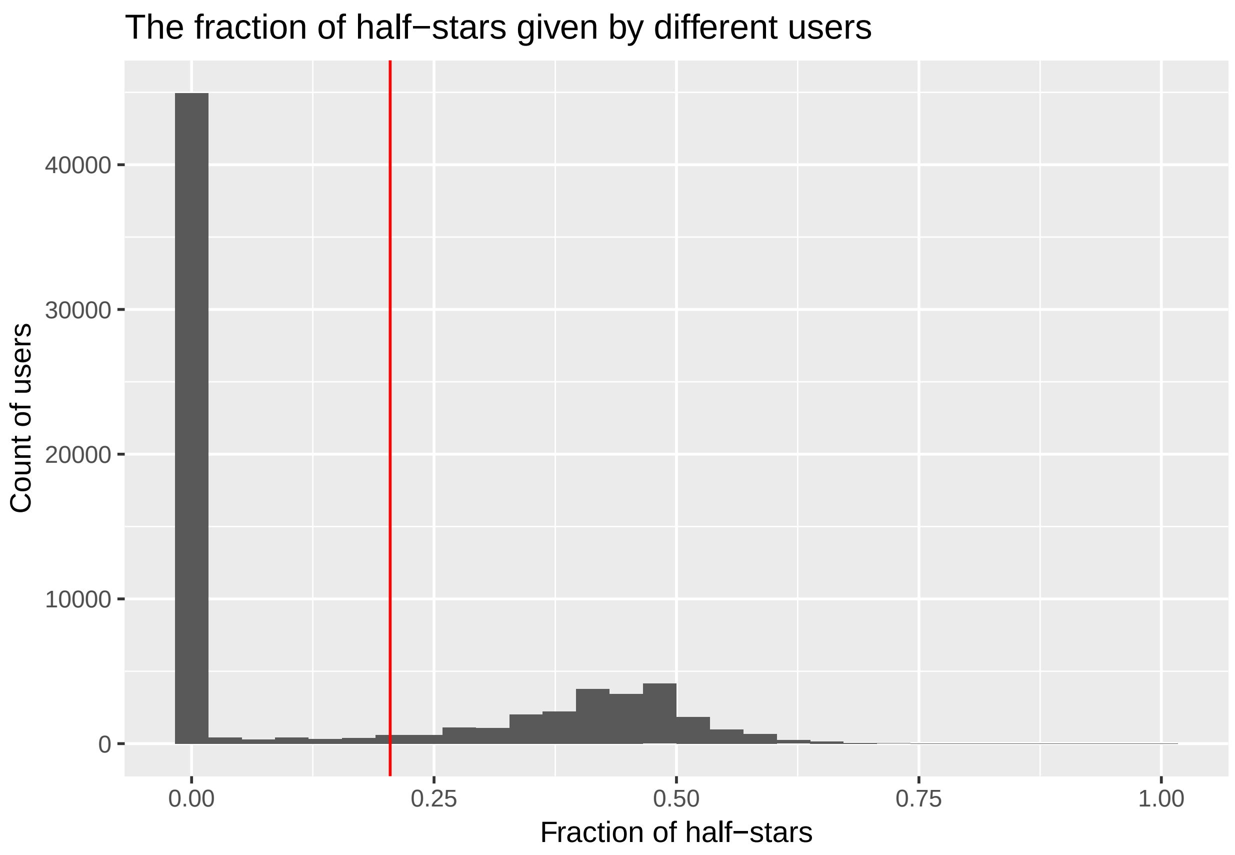
The lines on above chart have following menaing:

- red: common standard deviation of ratings for each epoch (also shown as error bars in chart **??**)
- blue: average standard deviation of ratings of one user for each epoch

All distributions are slightly asymetrically shifted to left. For older movies the blue line is close to red one, but for more frequently rated movies there is slight difference and standard deviation per user is smaller that common one. This mens that movie epoch togather with user ID may bring some information into the expected rating. So it may be sensible to take this factor into consideration in the prediction model.

3.2.6 Fractional Ratings

We have seen in Global Statistics section (3.2.1) that users tend to give whole stars ratings more then half-stars, but does this depends on particular users? Let’s check this.



Red line marks total half-star frequency.

This chart shows that users are divided into two separate groups:

- those who never use half stars
- those who give half-star ratings with randon (close to 0.5) frequency

This is interesting observation and can highly improve prediction acuracy of our model.

Chapter 4

Models

I this chapter I describe process of building and improving prediction models, starting from simplest ones to more advanced. Initially I will be following steps described in the course official book (Irizarry, 2018b), but later will try to use some results described in prevoius chapter and further modify model based on early results. In this chapter, we will be using following symbols and abbreviations:

- U - set of all users contained in data set. Its element (i.e. particular user) will be denoted by u.
- I - set of all movies contained in data set. Its element (i.e. particular movie) will be denoted by i.
- $T,\ V$ - we have divided entire data set into two distinct sets: training T and validation V. Assuming that there may be only one (or none) rating of given movie by given user, we can define those sets as subbects of respective indexes pairs. Following conditions are met:

$$T \subsetneq U \times I,\ V \subsetneq U \times I,\ T \cap V = \emptyset$$

- $Y_{u,i}$ - rating given by user u for movie i. This synbol denotes random variable and is used only to describe models.
- $y_{u,i}$ - observed rating given by user u for movie i ($(u,i) \in T \cup V$).
- $\epsilon_{u,i}$ - random fluctuation - random variable with small variance and centered around zero. This symbol only appears in model description. In actual preditions we replace it with its expected value, which is 0.
- $\rho_1(), \rho_2(), \rho_u()$ - ratings' rounding functions (to full-star or half-star).
- RMSE, ACCU - loss functions that will measure how good is model, see section 2.1. I will be rounding results of those functions to three meaningfull digits after comma.

More symbols are introduced for particular models.

4.1 Naive models

I will calculate measures for simplest possible "model", which is constant prediction for all movies and users equal to total ratings avarage. This will be our starting, referrence point.

4.1.1 Mean of ratings

Our model will be simple:

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

where μ is mean of all ratings from training set.

RMSE is equal 1.061. Values are not rounded to allowed ratings so ACCU does not have sense here.

Table 4.1: Results (naive models)

Method	RMSE	ACCU
Just the average	1.061	NA
Just the modal	1.168	0.288

4.1.2 Modal rating

Our model will be equally simple:

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

where m is mode of all ratings from training set (i.e. most frequent rating).

RMSE is equal 1.168 and accuracy is 0.288. The most frequent value is 4.

4.1.3 Summary

4.2 Movie effect

There are generally better and worse movies, and better ones tends to be rated higher than worse. We will take this into account in our model:

$$Y_{u,i} = \rho_{\bullet}(\mu + b_i + \epsilon_{u,i})$$

where b_i is movie effect - i.e. some modifier (with avarage zero) which is dependent on particular movie, and $\rho_{\bullet}$ stands for rounding function which will transform float-numeric calculation into allowed value (in 0.5 steps). The bullet sign means that we will be trying several rounding functions.

4.2.1 Rounding functions

At first we will check model without any rounding, so:

$$\rho_0(r) = r$$

Then we will check most obvoius rounding strategy i.e. to round calculated rating prediction to nearest allowed value - i.e. the value from set $0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5$. Let's call it ρ_1. Respective R code:

```r
rho1 <- function(rating) {
  round(rating * 2) / 2
}
```

Looking at histogram **??** we see a problem with this approach: users far more frequently give whole star ratings then half-star. So next tried rounding function will round to whole stars: ρ_2 with the code:

```r
rho2 <- function(rating) {
  round(rating, 0)
}
```

Finally we will try to utilize real proportion of half-stars in ratings from training set to more precise rounding function:

$$\rho_3(r) = \begin{cases} \rho_1(r) & \text{if } |r - \rho_1(r)| < \tau/2 \\ \rho_2(r) & \text{if } |r - \rho_1(r)| \geq \tau/2 \end{cases}$$

where τ is the proportion of half-star ratings to all ratings which is for traing data 0.205. The code:

```
rho3 <- function(rating) {
  r1 <- rho1(rating)
  ifelse(abs(rating - r1) < thres_halfs / 2, r1, rho2(rating))
}
```

4.2.2 Predictions

From mathematics we know that RMSE measure will be minimal if we assume movie factor as avarage rating difference from total mean taken for all users who rated given movie:

$$b_i = \text{Avg}_u(Y_{u,i} - \mu)$$

Following table list measures of predictions calculated according our model with applied different rounding functions:

Table 4.2: Movie effect with applied different rounding functions.

$\rho_\bullet$	RMSE	ACCU
ρ_0	0.944	NA
ρ_1	0.955	0.225
ρ_2	0.984	0.326
ρ_3	0.966	0.289

Results are interesting. It looks like trying to optimize RMSE is in contradiction to optimizing ACCU. It may also mean that maybe applied rounding functions are still not ideal. I will try to improve rounding function further in next models.

4.2.3 Summary

We will add two models from prevoius section to our results list - those which the best optimize RMSE and ACCU.

Method	RMSE	ACCU
Just the average	1.061	NA
Just the modal	1.168	0.288
+ Movie effect (no rounding)	0.944	NA
+ Movie effect (rounding rho2)	0.984	0.326

Note that for models without rounding accuracy measure does not make sense because by definition it will be very low.

4.3 User effect

Some users may be more demanding than others, and some may just like almost every movie or be very polite while rating. Also the mental maening of particular grade may be different (i.e. four stars for some may be high rate while for others just avarage). We will try to consider this factor now.

$$Y_{u,i} = \rho_\bullet(\mu + b_i + b_u + \epsilon_{u,i})$$

where new factor: b_u is user effect - in similar way like movie effect in last section.

4.3.1 Rounding functions

Beside general level of rating there is also one more factor which is very user specific. It is user's inclination for giving half-star's ratings. Some peple are perfoectionists and like do the things with highest precision - those will more likely will give more precise grades (i.e. fraction grades), whole other may not care that much or just see half-strs as not estetic thing - those very rarely or just never wil give fraction rates. This ssms to be strongly confirmed on graph **??** where users are more-less grouped into those never giving fractions rates and those using fraction rates with frequency close to 50%. So it makes sense to use different rounding function for those two groups. So we will modify slightly our prevoius rounding function:

$$\rho_u(r) = \begin{cases} \rho_1(r) & \text{if } |r - \rho_1(r)| < \tau(u)/2 \\ \rho_2(r) & \text{if } |r - \rho_1(r)| \geq \tau(u)/2 \end{cases}$$

where $\tau(u)$ is the proportion of half-star ratings calculated for particular user (unlike ρ_3 where it was constant). The code:

```
rfrac_counts <- edx %>%
  filter(rating != round(rating)) %>%
  group_by(userId) %>%
  summarize(rfrac = n())

thres_halfs_by_user <- edx %>%
  group_by(userId) %>%
  summarize(rtot = n()) %>%
  left_join(rfrac_counts, by = "userId") %>%
  mutate(rfrac = ifelse(is.na(rfrac), 0, rfrac)) %>%
  mutate(halfs = ifelse(rtot > 0, rfrac / rtot, thres_halfs)) %>%
  select(-rtot, -rfrac)

rho_u <- function(rating, user) {
  r1 <- rho1(rating)
  ifelse(abs(rating - r1) < filter(thres_halfs_by_user, userId == user)$halfs / 2, r1, rho2(rating))
}
rm(rfrac_counts)
```

Note: This code was given only for ilustration. Using such defined `rho_u` would be very unefficient bacause of nested filtering. While calculating predistion we will left-join `thres_halfs_by_user` frame to validation set instead and then calcute rounded value - what will give equivalent result.

4.3.2 Predictions

As user effect factor we will take average of difference of ratings from predictions of our prevoius model (including movie effect) taken by movies for given user:

$$b_u = \text{Avg}_{\{i:\ (u,i)\in T\}}(y_{u,i} - (\mu + b_i))$$

Following table list measures of predictions calculated according our model with applied different rounding functions:

Table 4.3: User effect with applied different rounding functions.

$\rho_\bullet$	RMSE	ACCU
ρ_0	0.866	NA
ρ_1	0.877	0.248

$\rho_\bullet$	RMSE	ACCU
ρ_2	0.912	0.361
ρ_3	0.889	0.318
ρ_u	0.896	0.356

Results generally improved but are suprising I expected that ρ_u should be pretty good now, but it's RMSE is second to worst and accuracy is still worse that ρ_2. I wonder if there is still some fundamental, not accounted factor.

4.3.3 Summary

Like recently we will add two models from prevoius section to our results list - those which the best optimize RMSE and ACCU.

Method	RMSE	ACCU
Just the average	1.061	NA
Just the modal	1.168	0.288
+ Movie effect (no rounding)	0.944	NA
+ Movie effect (rounding rho2)	0.984	0.326
+ User effect (no rounding)	0.866	NA
+ User effect (rounding rho2)	0.912	0.361

Note that for models without rounding accuracy measure does not make sense because by definition it will be very low.

4.4 Year of production effect

Results shown in section 3.2.5 seems to suggest that movie year of production may have any impact on ratings given by some users. So let's check if adding this factor to our model will improve it.

$$Y_{u,i} = \rho_\bullet(\mu + b_i + b_u + b_{u,e} + \epsilon_{u,i})$$

where $b_{u,e}$ is year's (epoch's) efect calculated per given user from his/her ratings of movies that were produced in similar time frame that movie whose rating is being predicted. We will calculate it as average of those ratings:

$$b_{u,e} = \text{Avg}_{\{i:\ E(i)\ni e \text{ and } (u,i)\in T\}}(y_{u,i} - (\mu + b_i + b_u))$$

Here $E(i)$ is set of movies that are produced in same time frame as given movie i. We do not use per year ganularity, but rather widen "epochs" as described in 3.2.4. If this value can not be calculated for some record from training set we will assume it to be zero.

4.4.1 Predictions

Following table list measures of predictions calculated according our model with applied different rounding functions:

Table 4.4: Epoch effect with applied different rounding functions.

$\rho_\bullet$	RMSE	ACCU
ρ_0	0.868	NA
ρ_1	0.88	0.251

$\rho_\bullet$	RMSE	ACCU
ρ_2	0.914	0.366
ρ_3	0.892	0.323
ρ_u	0.898	0.361

To my surprise RMSE measures degradated a bit, while Acuracy little imroved. It looks like ading movie year of production to model introduced more noise to predictions and are not very helpfull. I suppose that it was because many users have rated very few or just single movies from given epoch, and this influenced too much on prediction model. To quickly check that (without making heavy computations) I decided to try cutting-off (bring down to zero) movie epoch parameter in case user rated less than 10 movies from given epoch.

Here is the code of this transformation (see last mutate instruction):

```
epoch_by_user_avgs_test <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(year_class, userId) %>%
  summarize(b_ue = mean(rating - mu - b_i - b_u), cnt = n()) %>%
  mutate(b_ue = ifelse(cnt < 10, 0, b_ue))
```

And results:

$\rho_\bullet$	RMSE	ACCU
ρ_0	0.861	NA
ρ_1	0.873	0.251
ρ_2	0.908	0.365
ρ_3	0.885	0.323
ρ_u	0.892	0.361

This time RMSE measures have improved and are also better than those without date of movie production effect. It means that movie production date may be a meaningfull factor but needs to be regularized. This case shows that regularization may be in general important process. So I decided to reimplement all prevoius models, but this time performing regularization instead of taking just means.

4.5 Regularization

Regularization means that we want our movie, user or year effects to be less affected by potential random fluctuations. Random fluctuation happens when the mean is taken from relatively small number of samples. In such case we would like calculated effect to be smaller - i.e. closer to zero, while when taken from bigger sample - closer to actual average. This is being achived by changing our effect calculation rule from e.g.

```
  summarize(b_i = mean(rating - mu))
```

to:

```
  summarize(b_i = sum(rating - mu) / (n() + lambda))
```

where `lambda` is some positive parameter (if zero it turnes out to be regular mean). It is being appointed experimantally in the following way:

- we split our training set (`edx`) into 2 separate sub-sets: `edx_training` and `edx_test` in the way that training set is bigger then test (e.g. 90% / 10%) and all movies and users present in test set are also present in training set.

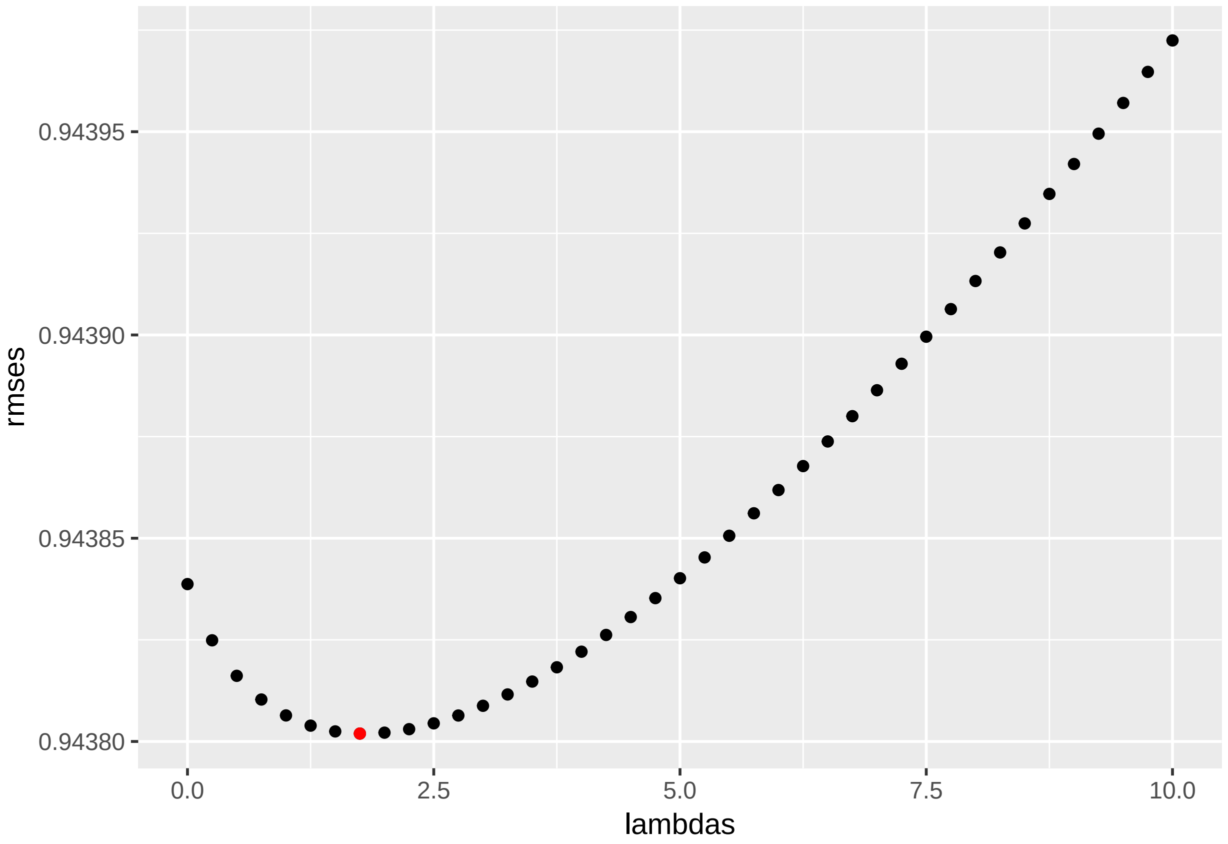
- we assume that `lanbda` belongs to certain set of discrete values (e.g. the range between certain values at certain step).
- for every `lambda` value from our set we calculate effect based on training set and check RMSE against test set.
- we choose `lambda` for which RMSE is minimal.

Unlike in the course book, we will try to find optimal `lambda` separately for every kind of effect.

4.6 Movie effect with regularization

We will be checking following lambda values:

```
lambdas <- seq(0, 10, 0.25)
```



Minimum is reached for $\lambda_i = 1.75$ which we then apply in our model.

Following table compares measures calculated with and without regularization (marked with **reg/mean** postfix respactively):

Table 4.6: Movie effect with regularization.

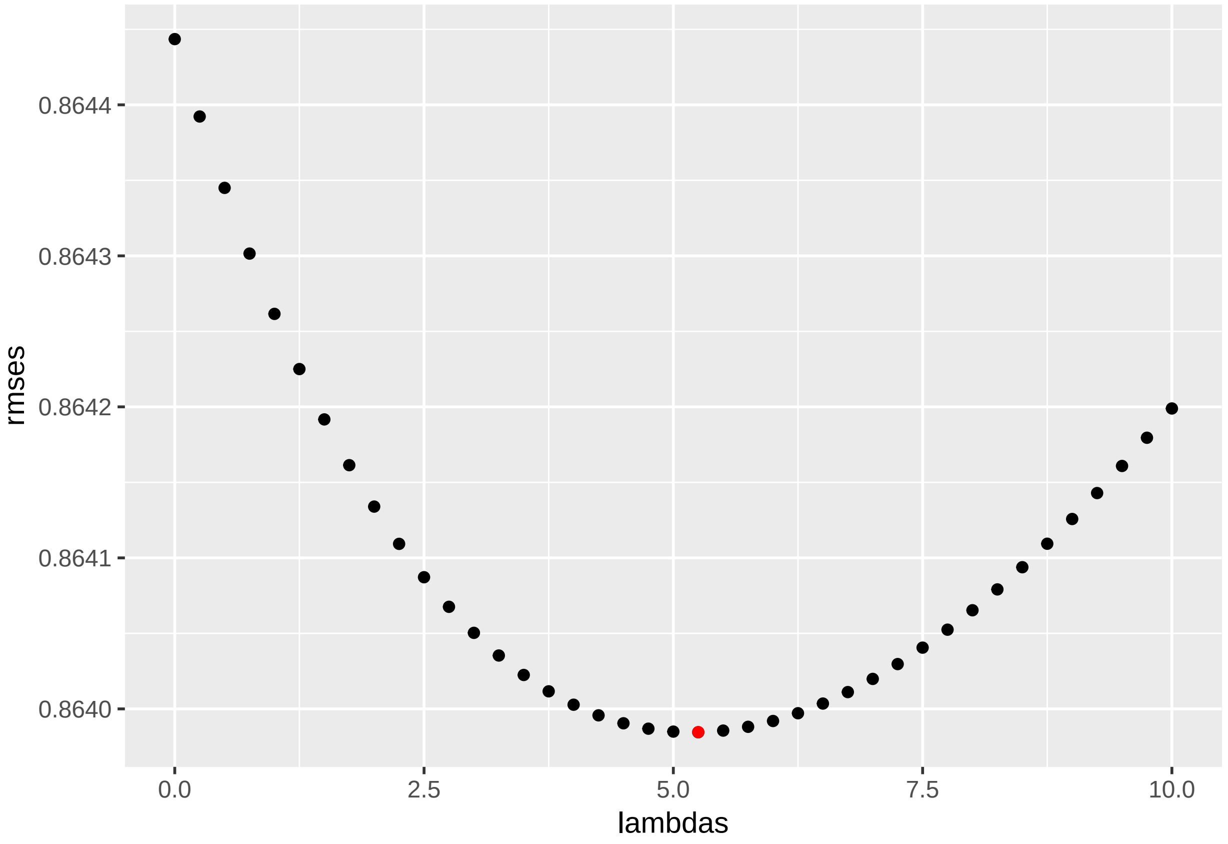
$\rho_\bullet$	RMSE reg	ACCU reg	RMSE mean	ACCU mean
ρ_0	0.944	NA	0.944	NA
ρ_1	0.955	0.225	0.955	0.225
ρ_2	0.984	0.326	0.984	0.326
ρ_3	0.966	0.289	0.966	0.289

Results are pretty much the same, so regularization did not helped much in this case.

4.7 User effect with regularization

We will be checking the same lambda values:

```
lambdas <- seq(0, 10, 0.25)
```



Minimum is reached for $\lambda_u = 5.25$ which we then apply in our model.

Following table compares measures calculated with and without regularization:

Table 4.7: User effect with regularization.

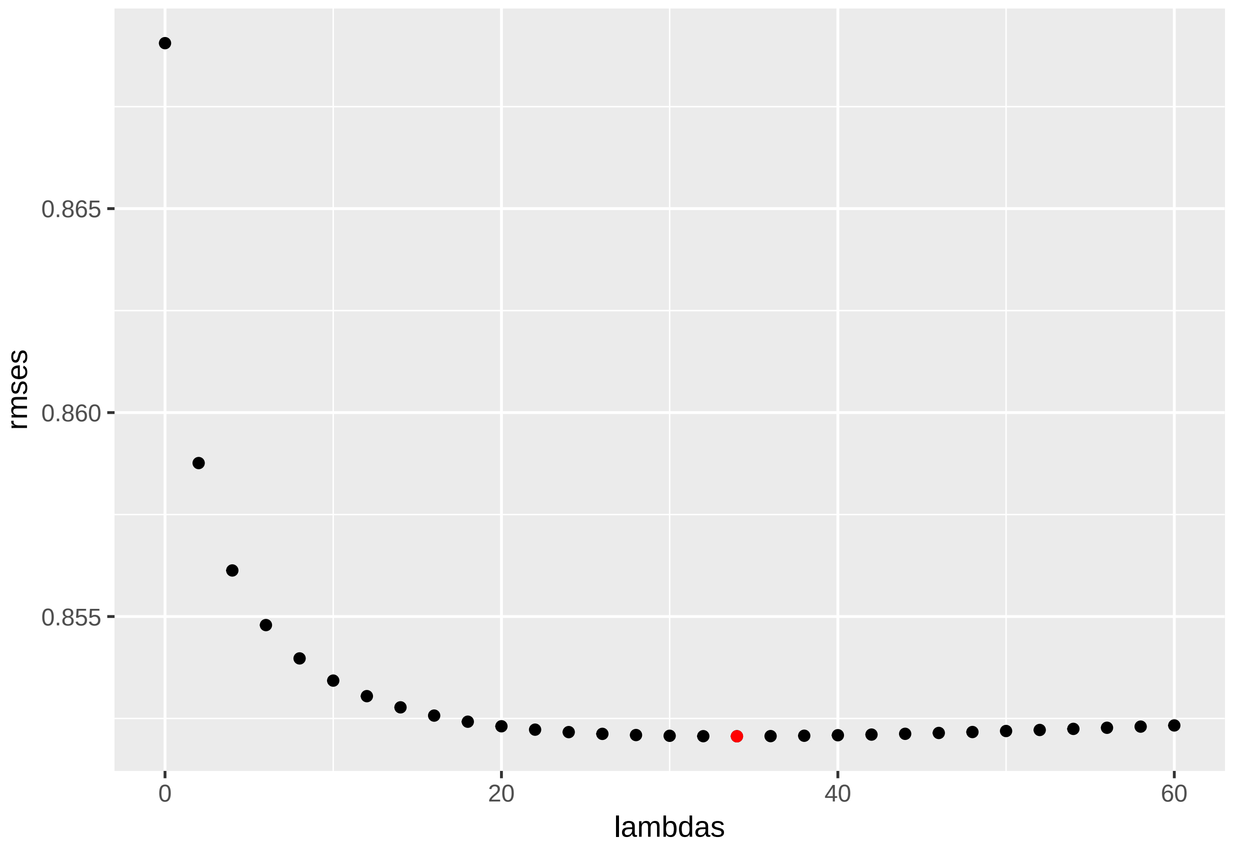
$\rho_\bullet$	RMSE reg	ACCU reg	RMSE mean	ACCU mean
ρ_0	0.865	NA	0.866	NA
ρ_1	0.877	0.248	0.877	0.248
ρ_2	0.911	0.36	0.912	0.361
ρ_3	0.889	0.318	0.889	0.318
ρ_u	0.895	0.356	0.896	0.356

Results are pretty much the same, so regularization also did not helped much in this case.

4.8 Year of production effect with regularization

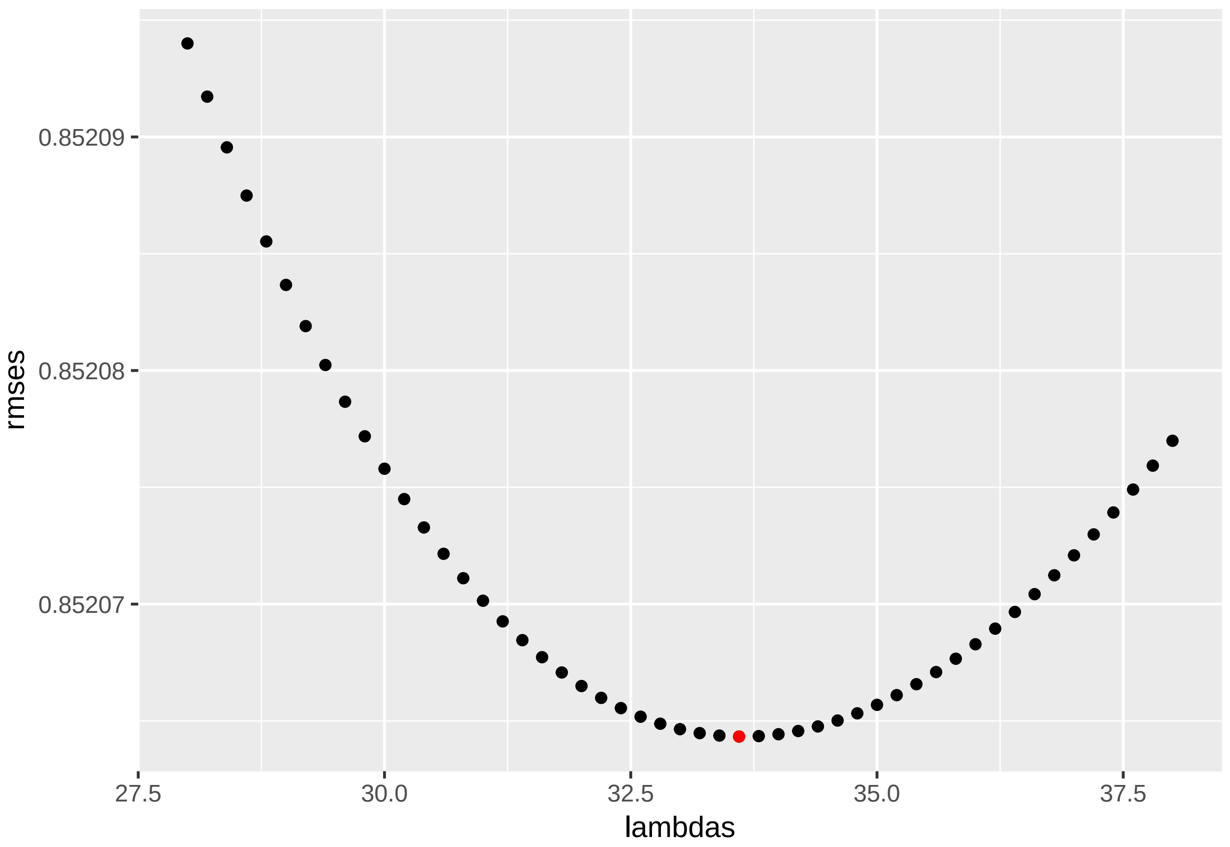
We will be checking at first more wide range of lambda values:

```
lambdas <- seq(0, 60, 2)
```



Minimum is reached for value 34. We will then try to check values around this rough minimal value to estamate parameter with bigger precision:

```
lambdas <- seq(28, 38, 0.2)
```



Minimum is reached for $\lambda_e = 33.6$ which we then apply in our model.

Following table compares measures calculated with and without regularization:

Table 4.8: Epoch effect with regularization.

$\rho_\bullet$	RMSE reg	ACCU reg	RMSE mean	ACCU mean
ρ_0	0.857	NA	0.868	NA
ρ_1	0.869	0.251	0.88	0.251
ρ_2	0.904	0.364	0.914	0.366
ρ_3	0.881	0.322	0.892	0.323
ρ_u	0.888	0.36	0.898	0.361

Those results prove that regularization allowed us to improve results considerably.

4.8.1 Summary

We have finally work-out model that futher improved our results - so let's add it to our sumary list.

Method	RMSE	ACCU
Just the average	1.061	NA
Just the modal	1.168	0.288
+ Movie effect (no rounding)	0.944	NA
+ Movie effect (rounding rho2)	0.984	0.326
+ User effect (no rounding)	0.866	NA
+ User effect (rounding rho2)	0.912	0.361
+ Epoch regularized effect (no rounding)	0.857	NA
+ Epoch regularized effect (rounding rho2)	0.904	0.364

4.9 Generics effect

Some users may prefer certain kind of movies represented by generics assiciated to movies. Results shown in section 3.2.3 seems to confirm that. So let's add this factor to our model based on regularized effects. In below formulas we will add acute sign to effects that are computed with regularization (like b_i') to distinguish them from regular mean-based (like b_i).

$$Y_{u,i} = \rho_\bullet(\mu + b_i' + b_u' + b_{u,e}' + \overline{b_{u,G(i)}} + \epsilon_{u,i})$$

where $G(i)$ is set of genres associated with the given movie i, and:

$$\overline{b_{u,G(i)}} = \text{Avg}_{\{g \in G(i)\}}(b_{u,g})$$

i.e. it is mean of genre effects for genres of given movie and characteristic for given user. We will calculate it as average of diffeneces between known ratings for given user of other movies with the same genre and prediction of recent model:

$$b_{u,g} = \text{Avg}_{\{i:\ G(i) \ni g \text{ and } (u,i) \in T\}}(y_{u,i} - (\mu + b_i' + b_u' + b_{u,e}'))$$

Our model require a lot of calculation, so for performance reasons we will not use regularization here.

4.9.1 Predictions

Following table list measures of predictions calculated according to our model with applied different rounding functions:

Table 4.9: Generics effect with applied different rounding functions.

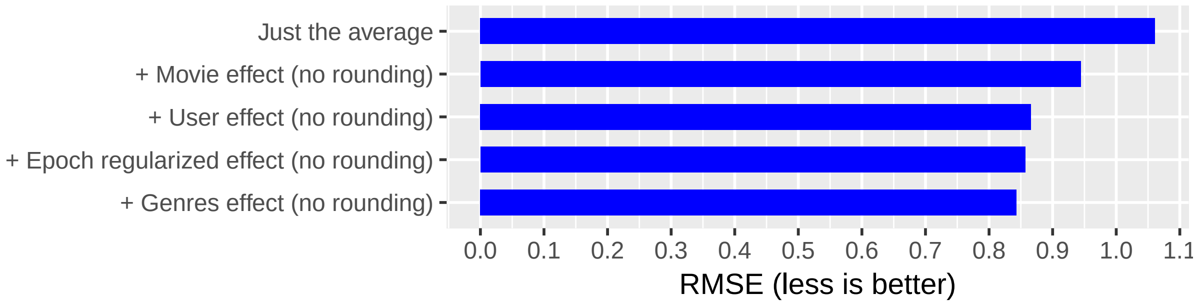
$\rho_\bullet$	RMSE	ACCU
ρ_0	0.843	NA
ρ_1	0.856	0.256
ρ_2	0.891	0.372
ρ_3	0.868	0.328
ρ_u	0.875	0.368

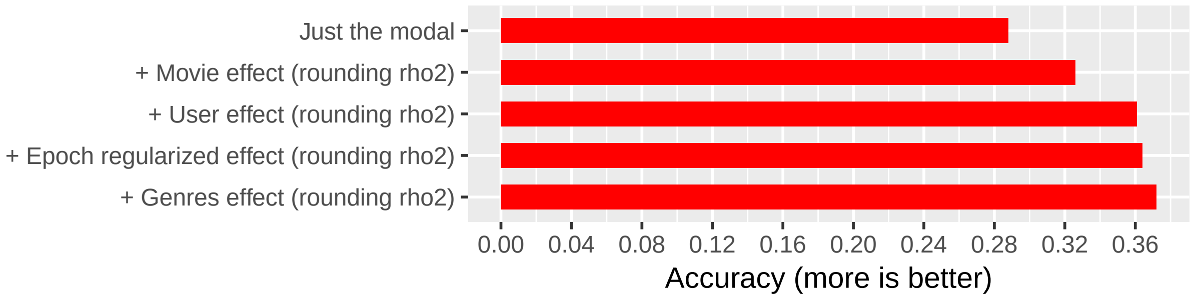
4.9.2 Summary

We will add recent result, and here is final list of our models:

Method	RMSE	ACCU
Just the average	1.061	NA
Just the modal	1.168	0.288
+ Movie effect (no rounding)	0.944	NA
+ Movie effect (rounding rho2)	0.984	0.326
+ User effect (no rounding)	0.866	NA
+ User effect (rounding rho2)	0.912	0.361
+ Epoch regularized effect (no rounding)	0.857	NA
+ Epoch regularized effect (rounding rho2)	0.904	0.364
+ Genres effect (no rounding)	0.843	NA
+ Genres effect (rounding rho2)	0.891	0.372

4.9.3 Graphical overview of results





Chapter 5

Conclusion

5.1 Final Model

Our final movie rating prodiction model takes folowing characteristics into consideration:

- how movie was rated by other users
- how user rated all other movies
- how user rated movies coming from similar year decade
- how user rated movies that have similar generics that movie to be rated
- how much user is inclined to give half-star ratings

The rating estimation is a sum of total mean value from all ratings, and correction coefficients for all above factors called also 'effects'. In case of first three points above, the effect is calculated as:

$$b(\text{group}) = \frac{\sum_{\text{all movies and users in group}}(y_{i,u} - \hat{y}_{i,u})}{n(\text{group}) + \lambda}$$

where $y_{i,u}$ is actual rating, $\hat{y}_{i,u}$ is predicion of current model (i.e. model before applying currently calculated effect), $n(\text{group})$ is number of elements in group, and λ is a constant which depands on kind of effect and is caclulated by try-check (training) process, where different values are tried and the one which minimize RMSE is selected. All calculations are performed only on training set: `edx`. Note that if $\lambda = 0$ this calculataion narrowns down to siple average. But usually (aspecially for year of production effect), better results were achived when this parameter was a positive contant. The idea behind this formula is that it gives value close to average for bigger gropus and is closer to zero for smaller groups (for which average might not represent any trend but rather be more random value).

For the forth point (generics effect) calculation is more complicated, because every movie has many assigned generics and this particular set of generics might be different then for other movies from validation set. So we first divide our training set by particular generics (duplicating rating records), and then calculate averages for particular user and generic. Therefore when calculating prediction for particular movie and user we take average of those caculated averages. Because of proformance reasons we do not try to regularize results and estimate special λ value for this case. This is described in more detail in section 4.9 and prevoius sections.

Such model when applied to validation set gave us RMSE measure equal to **0.843** when taking raw calculated predictions without rounding. For rounded predictions maximal achived RMSE was: 0.856 - when rounding was performed to closed allowed rating. Maximal obtained accuracy was when we rounded predictions to whole stars, and it was: 0.372. All caculation routines with full source code is contained as R code ambeded in this report sources, which are available from repository: https://github.com/gwierzchowski/.

5.2 Ideas for futher researches

The time for course final capstone excercise is limited, and also the time that I can sacrifice on it is very limited. Because of those constraints I did not managed to realize some ideas that I initally intended. Most important are:

- try to utilize somehow the movie rating time, assuming that user preferences might change, some kind of time limited "fashion" may impact ratings etc.
- try to somehow join together into groups movies that have similar titles (e.g. movies from the same 'series' should land in the same group). Then estimate some 'effect' for such groups, assuming that users would rate movies from same group or just movies from the same series similarly.
- try to deeper analyze material in the course where fectoring was used, and maybe perform some factoring on snmaller data set and try to use it to improve prediction (this might be non-triviel bacause of computational complexity).
- I was not very happy with results obtaned using rounding function that is based on calculation how frequently particular users give half star's ratings (I noticed that it is highly user-dependent) - the $\rho_u(r)$ function. It would be interesting to deeper investigate why this funtion in a way that I implemented it does not improve results that much (and is even worse than just rounding to full stars).
- I noticed that the data that we worked with (*GroupLens research lab* (Sou, 2018)) also contained a "movie tag" imformation. It was not extracted by corse official data prepration script, so probably cound not be used in official result, but for an extercise it would be worth to check if it could improve our predictions further.

I belive that using some of those ideas might slightly improve result. Hovewer I noticed that adding more and more ingredients to the model improves the result to a lesser extent, as it can be seen on graph **??**. So it looks like there might be some limit for predicton acuracy coming from fact that actual user ratings is laden with some random effect and that we predict ratings for some new users that simply did not shown their preferences yet bacause they have rated too few movies.

5.3 Final words

I would like to take opportunity and express my appreciation to all persons imvolved into preparation of this excelent "Data Scientists Proffesional" series of courses, espacially to *prof. Rafael A. Irizarry* - the main author. I am also very grateful that this course and all materials have been made public.

I have learned a lot. Gained also some hands-on experiance with R programming language which I did not known before. I like very much RStudio programming environment, but honestly I'm not very big fan of R language, espacially because of its strange syntax, unclear type system, sometimes not clear help descriptions for packages, but mostly for slowness and one-thread approach. I'm also not satisfied from `bookdown` and `rmarkdown` packages. While HTML back-end seems to be usually correct, however PDF one is frequetly misplaced (e.g. tables or pictures are placed in wrong places on page, or table or picture titles are not printed, bibliography is hard to get dome right, etc). Looking forward I think I will be trying to use languages like Julia or Python for my next data science related learning efforts. In particular I'd like to learn more about Neural Networks.

Used packages:

- **?**
- Wickham (2017)
- from Jed Wing et al. (2019)
- Wickham (2019)
- Wickham and Henry (2019)

Bibliography

(2018). Movielens 10m dataset.

from Jed Wing, M. K. C., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., the R Core Team, Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., Candan, C., and Hunt., T. (2019). *caret: Classification and Regression Training.* R package version 6.0-84.

Irizarry, R. A. (2018a). Harvardx data science series - capstone.

Irizarry, R. A. (2018b). *Introduction to Data Science.*

Wickham, H. (2017). *tidyverse: Easily Install and Load the 'Tidyverse'.* R package version 1.2.1.

Wickham, H. (2019). *stringr: Simple, Consistent Wrappers for Common String Operations.* R package version 1.4.0.

Wickham, H. and Henry, L. (2019). *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions.* R package version 0.8.3.