MovieLens Project

Su Lin

Sep 18, 2019

Dear peers, before we get started, I would like to thank you for your time and effort spent on my project report. If any concerns or comments, please feel free to shoot me an email at su.linwower@hotmail.com. ***

1. Project Overview

The project is for Capstone, the final course of *Edx HarvardX Data Science Professional Certificate Program*, which I like very much, and I have learnt and enjoyed much along the journey with Professor Irizarry and my peers.

The aim of this project is to built a movie recommdation system using the *10M version of the MovieLens dataset*, which will use about 90% of the data to train a machine learning algorithm, and predict movie ratings for all movie-user combinations in the rest of data which is a validation or test set. The 10M MovieLens data includes 10,000,054 ratings for 10,677 movies by 69,878 users. Our project goal is to minimize the loss functiom RMSE, better lower than 0.8649.

The method used here is much the same as Professor Irizarry instructed during the eighth course of the program series, Machine Learning, which is the combination of baseline predictors, regularization and matrix factorization. Code details can be found in the corresponding R script file. Key steps performed are as following.

1. Create train and validation sets (same code as provided in the course)
2. Data wrangling and exploration
3. Train with cross-validation to built a model
4. Predict on test set and evaluate with RMSE

The PC used here is a Intel(R) Core(TM) i5-6200U 2.30GHz, RAM 8.00G laptop.

To speed up the process of converting the R Markdown file to PDF document, most code chunks for intermediate results are omitted here, and results produced for presenting in the report are saved in the working directory. For code details, please refer to the corresponding R script file.

2. Analysis

Create train and validation sets

Following the code provided in the course, omitted here (included in the R script file), two data sets are created with **edx** for training and **validation** for testing, basic information of the two data sets are shown as below. All varaible names are very plain, except *timestamp* which is rating date and time stored as integer.

Table 1: Sample data from edx

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

	userId	movieId	rating	timestamp	title	genres
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

Table 2: Dimensions of edx and validation

	no of records	no of variables
edx	9000055	6
validation	999999	6

The two date sets are saved as Rda files under the working directory for future use to avoid downloading the same data every time.

Data wrangling and exploration

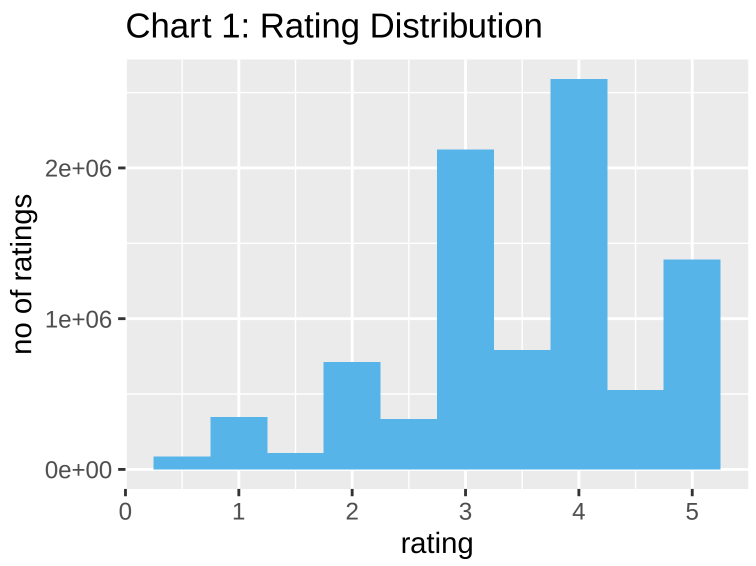
As shown above, the two data sets are quite tidy, but still can be further cleaned to faciliate the analysis and reduce the file size. After wrangling, **edx** set is saved as **training_set**, and **validation** set is saved as **test_set**. Below is how **training_set** looks, and **test_set** shares the same structure.

Table 3: Sample data from training_set

userId	movieId	rating	rating_date	release_year	tidy_title	genres
1	122	5	1996-08-02	1992	Boomerang	Comedy|Romance
1	185	5	1996-08-02	1995	Net, The	Action|Crime|Thriller
1	292	5	1996-08-02	1995	Outbreak	Action|Drama|Sci-Fi|Thriller
1	316	5	1996-08-02	1994	Stargate	Action|Adventure|Sci-Fi
1	329	5	1996-08-02	1994	Star Trek: Generations	Action|Adventure|Drama|Sci-Fi
1	355	5	1996-08-02	1994	Flintstones, The	Children|Comedy|Fantasy

The two cleaned data sets are also saved as Rda files to built the model during the following days. From now on, only **training_set** is used for exploration and training, as for **test_set**, it will be used for validation after the model is complete, so we pretend that we do not have this data set now.

First of all, let us start with some EDA (exploratory data analysis) with data visualisation and basic summary of statistics of the data.



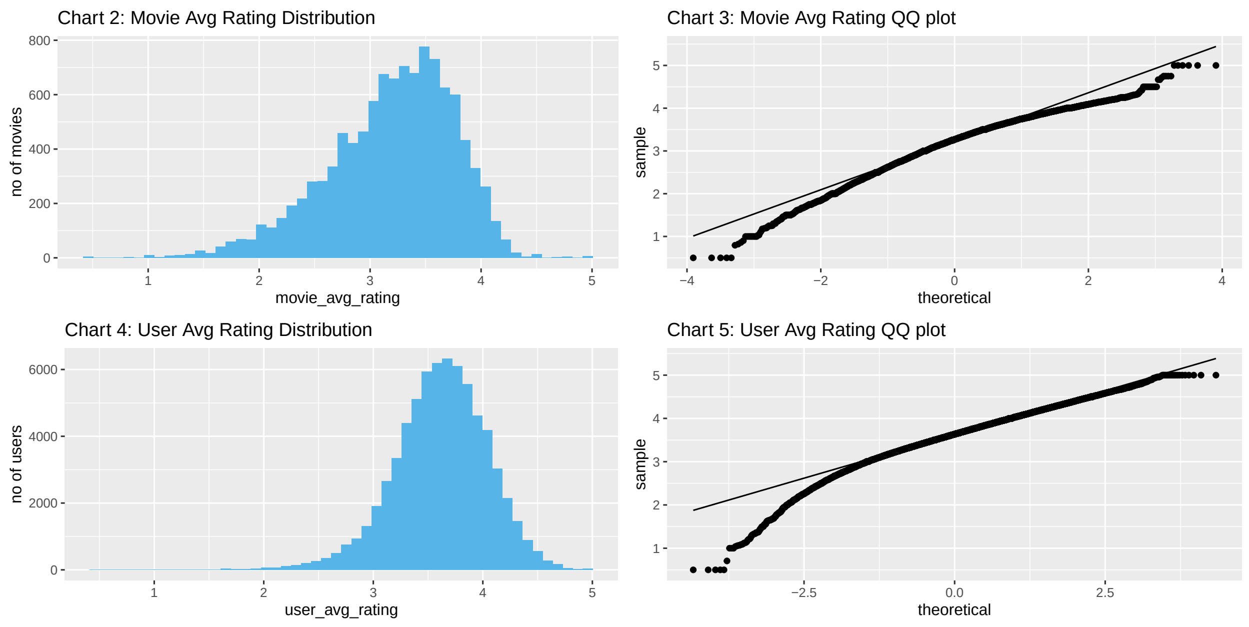


Table 4: Summary of Statistics

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
rating	0.5	3.000000	4.000000	3.512465	4.000000	5
movie_avg_rating	0.5	2.844262	3.267857	3.191736	3.609375	5
user_avg_rating	0.5	3.357143	3.635135	3.613602	3.902597	5

From above histograms and basic statistics, we know that 3-4 ratings predominate in the **training_set** with mean at 3.512465. For the average rating of each movie, it is clearly left-skewed based on the QQ plot (Pic.3). We see that average rating varies from movie to movie, which makes sence, cause most movies are moderate, and great movies are less than boring ones. The distribution of average rating of each user is also left_skewed, and the rating also has a user effect or bias.

Train with cross-validation to built a model

Naive model

With above findings and what we have learnt during the Machine Learning course*[1]*, we build our first and simplest model by predicting all movie-user combinations with overall average rating $\hat{\mu}$, then our model is

$$r_{u,i} = \mu + \varepsilon_{u,i} \tag{1}$$

where $r_{u,i}$ is the real rating for user u and movie i, μ is the real average rating for all movie-user combinations, and $\varepsilon_{u,i}$ is independent error with 0 mean. We learnt that the least squares estimate of μ is the overall average rating of our **training_set**, $\hat{\mu}$, so our predicted $r_{u,i}$ is $\hat{r_{u,i}} = \hat{\mu}$. Based on the definition of RMSE as below with N being the number of user-movie combinations, we can evaluate our model's performance.

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

```
[1] "Overall average rating is 3.51246520160155"
```

```
[1] "RMSE is 1.06120181029262"
```

From above, we know that naive model's error is larger than one star, and is quite bigger than our goal of lowering it to less than 0.8649. Since EDA reveals that rating varies from movie to movie, and from user to user, we should also consider those effects in our model.

Baseline model

The predictors in below model are called baseline predictors in *[2]*.

$$\hat{r_{u,i}} = \hat{\mu} + \hat{b_u} + \hat{bi} \tag{3}$$

where $\hat{b_u}$ is the estimated value of user bias b_u, and $\hat{b_i}$ is the estimated value of movie bias b_i. In order to "penalize large estimates that are formed using small sample sizes"*[1]*, regularization is used. These two estimates can be got by solving the least squares problem as below with gradient descent.

$$\min_{b_u, b_i} \sum_{u,i}(r_{u,i} - \hat{\mu} - \hat{b_u} - \hat{b_i})^2 + \lambda(\sum_u \hat{b_u}^2 + \sum_i \hat{b_i}^2) \tag{4}$$

Other than gradient descent, there is another way, much easier and quicker, to estimate the parameters by decoupling the calculation of the b_i's from the calculation of the b_u's as shown below*[2]*, where $R(i)$ is the set of users who rated movie i, and $R(u)$ is the set of movies that user u have rated.

We use this decoupling method to calculate bias parameters through out this report which is the same as what we have learnt during Machine Learning course.

$$\begin{aligned} \hat{b_i} &= \frac{\sum_{u \in R(i)}(r_{u,i} - \hat{\mu})}{\lambda + |R(i)|} \\ \hat{b_u} &= \frac{\sum_{i \in R(u)}(r_{u,i} - \hat{\mu} - \hat{b_i})}{\lambda + |R(u)|} \end{aligned} \tag{5}$$

Since λ, the regularization parameter, is tunable, we can use cross-validation to find the best one that minimizes cross-validation set's RMSE. In order to do this without using **test_set**, a **train_set** (for training) and a **cv_set** (for cross-validation) are created from **training_set** via below codes.

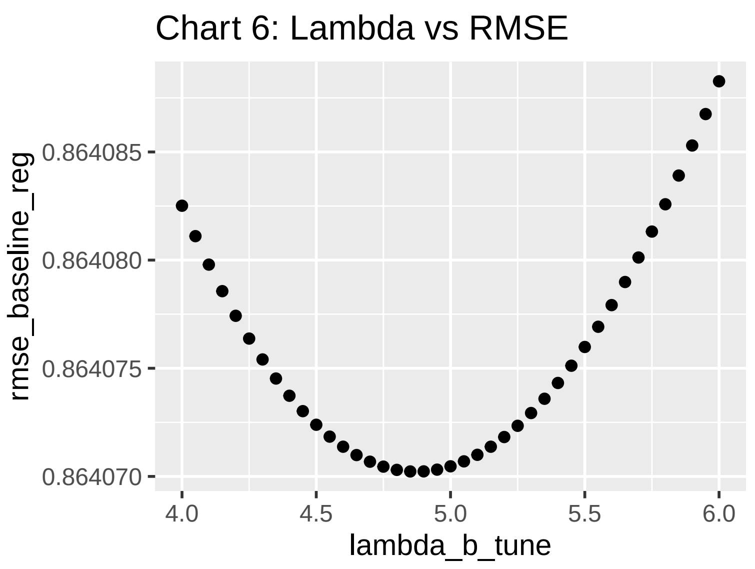
```r
# train_set and cv_set are created from training_set
set.seed(1)
cv_index <- createDataPartition(y = training_set$rating, times = 1, p = 0.1, list = FALSE)
train_set <- training_set[-cv_index,]
```

```r
temp <- training_set[cv_index,]

# make sure userId and movieId in cv_set are also in train_set
cv_set <- temp %>% semi_join(train_set, by = "movieId") %>% semi_join(train_set, by = "userId")

# add rows removed from cv_set back into train_set
removed <- anti_join(temp, cv_set)
train_set <- rbind(train_set, removed)
rm(cv_index, removed, temp)
```

During the tuning process, we found below relationship between λ and RMSE, so when $\lambda = 4.85$, we get the minimun RMSE for cross-validation set. Now we can insert the best λ we got into formulas (5) to get $\hat{b_u}$ and $\hat{b_i}$ and check the performance of our baseline predictors model with RMSE on **test_set**.



```
[1] "The best lambda is 4.85"
```

```
[1] "RMSE for baseline predictors model is 0.864818973159809"
```

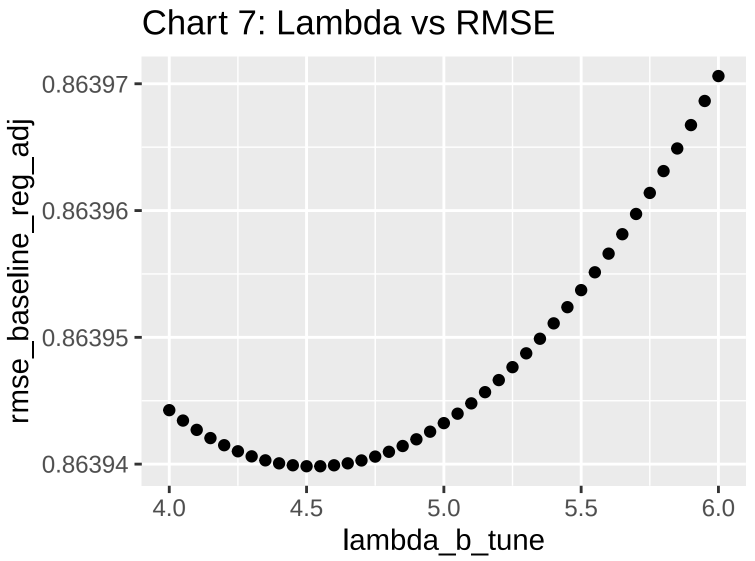
The RMSE is 0.864819 which achieves our project goal, but there are some unreasonable predictions detected in the results such as:

Table 5: Unreasonable Predictions

userId	movieId	rating	tidy_title	rating_predicted	movie_bias_reg	user_bias_reg	global_mean
50560	3593	0.5	Battlefield Earth	-0.4097493	-1.9392147	-1.98300	3.512465
36022	50	5.0	Usual Suspects, The	5.9978525	0.8531973	1.63219	3.512465

known from our data, the highest rating is 5, and the lowest is 0.5, so the prediction 5.9979 from user 36022 for movie 50 is out of rage. Because global mean $\hat{\mu}$ for all movie-user combinations is 3.5125, and moive 50's average rating is 0.8532 higher than global mean, indicating it may be a good movie; further more user 36022's average bias is 1.63219, indicating he or she is not a very choosy viewer; so the summation of these three exceed 5. On the contrary, negative movie bias and user bias may lead to negative ratings or ratings below 0.5, the lower bound.

Such cases damage our predicting performance, since no mather how much a user loves or hates a certain movie, 5 or 0.5 is the bound that can be given. So we adjust extreme values based on the bounds, that is if predicted rating is less than 0.5, make it 0.5; if it is higher than 5, make it 5.



```
[1] "The best lambda is 4.55"
```

After this adjustment, our λ is 4.55, and we can insert this λ into formulas (5) again to get new $\hat{b_u}$ and $\hat{b_i}$.

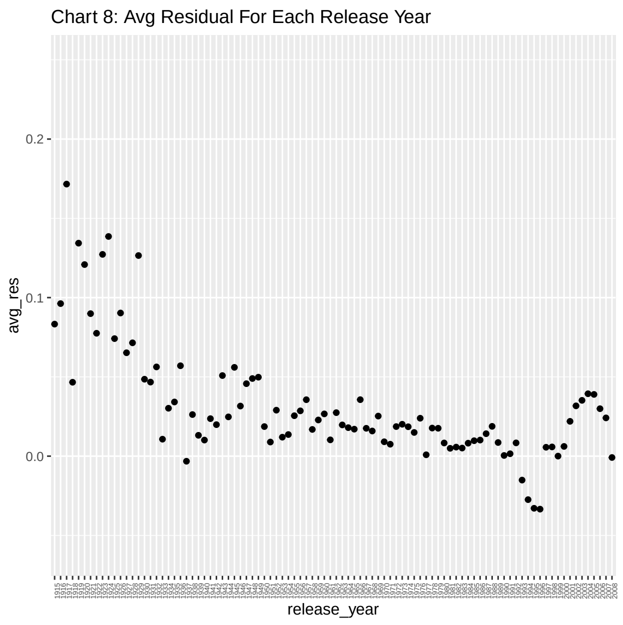
```
[1] "RMSE for baseline predictors model is 0.864710066143266"
```

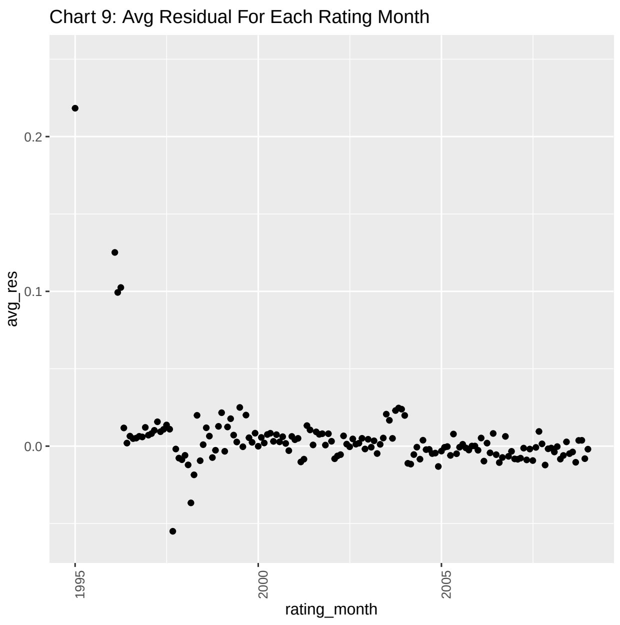
The corresponding RMSE is 0.864710, a litter better than previous one, but let's see if we can do better.

Baseline + time biases model

We can further explore the residuals from our model as below.

$$res_{u,i} = r_{u,i} - \hat{r_{u,i}} = r_{u,i} - \hat{\mu} - \hat{b_u} - \hat{b_i} \tag{6}$$





Judging from both scatter plots, a clear declining trend can be seen for average residuals by release year, and a little bit declining trend for average residuals by rating month. So we will take these two new biases into our new model as below, where $\hat{b_{ry}}$ is release year bias, and $\hat{b_{rm}}$ is rating month bias.

$$\hat{r_{u,i}} = \hat{\mu} + \hat{b_u} + \hat{b_i} + \hat{b_{ry}} + \hat{b_{rm}} \tag{7}$$

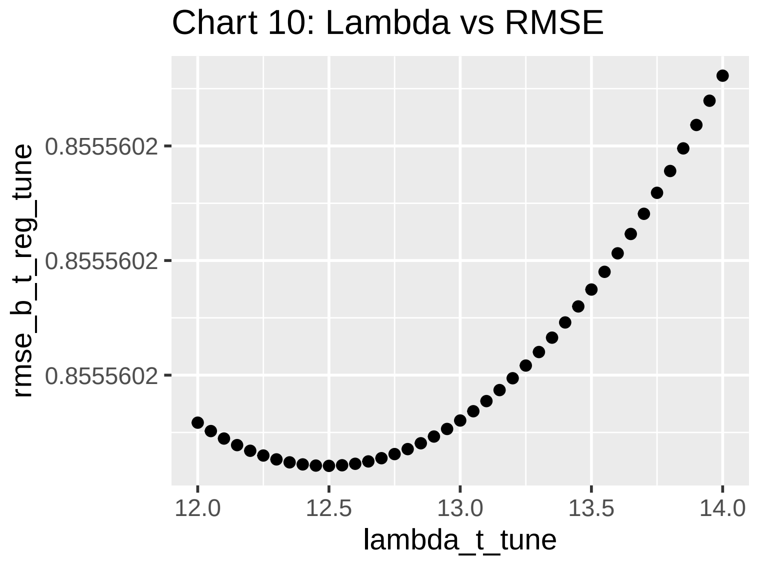
We still use regularization to get release year bias estimate and rating month bias estimate with respect to our baseline model's residual, and cross-validate with **cv_set**

$$res_{u,i} = \hat{b_{ry}} + \hat{b_{rm}} \tag{8}$$

Similar as formula (5), these two estimates can be got by decoupling the calculation of the b_{ry}'s from the calculation of the b_{rm}'s as shown below, where $R(ry)$ is the set of ratings given by all users to all movies with certain release year ry, and $R(rm)$ is the set of ratings given in certain rating month rm by all users to all movies.

$$\begin{aligned} \hat{b_{ry}} &= \frac{\sum_{u,i \in R(ry)}(r_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u})}{\lambda_t + |R(ry)|} \\ \hat{b_{rm}} &= \frac{\sum_{u,i \in R(rm)}(r_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u} - \hat{b_{ry}})}{\lambda_t + |R(rm)|} \end{aligned} \tag{9}$$

Still λ_t, the regularization parameter, is tunable, we can use cross-validation to find the best one that minimizes the RMSE for cross-validation set.



```
[1] "The best lambda is 12.5"
```

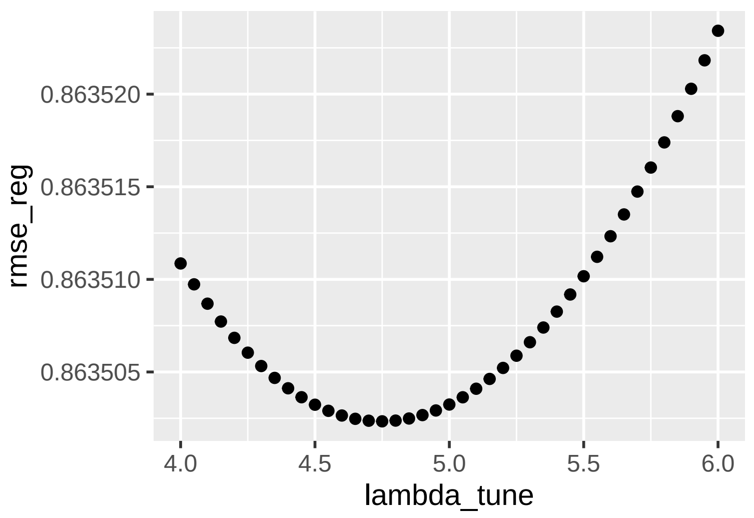
The best λ_t that minimizes RMSE is 12.5, and we can get $\hat{b_{ry}}$ and $\hat{b_{rm}}$ as regularized estimate of release year bias and regularized estimate of rating month bias respectively, based on formulas (9). Again we validate with **test_set** to see the performance of this baseline + time biases model.

```
[1] "RMSE for baseline + time biases model is 0.864266493128666"
```

The corresponding RMSE drops to 0.8642665, a litter better than beseline model's RMSE 0.864710.

By now, we have two unsymmetrical regularization parameters, λ and λ_t, what if we regularize them symmetrically? Let's see if this can improve the performance. The formulas used are still the same as (5) and (9), but with the same regularization parameter λ.

Chart 11: Lambda vs RMSE



```
[1] "The best lambda is 4.75"
```

```
[1] "RMSE for baseline + time predictor model is 0.864265562992227"
```

The best λ is 4.75, and corresponding RMSE is 0.8642656, slightly better than the unsymmetrical regularization's RMSE 0.8642665. So the symmetrical regularization way is adopted, and its residual (10) is further analysed to see if there are any latent factors we have not taken into account.

$$res_{u,i} = r_{u,i} - \hat{r_{u,i}} = r_{u,i} - \hat{\mu} - \hat{b_u} - \hat{b_i} - \hat{b_{ry}} - \hat{b_{rm}} \qquad (10)$$

Baseline + time biases + SVD model

To do so, some popular movies' and active users' ratings are seleted from **training_set**. Active users means users who rated more than 300 movies, and popular movies are shown at the y axis in below data images. A residual image and an actual rating image are plotted for these movies, from which we can see how different users think of these well-known movies.

Chart 12: Residual Image for Interactions between Users and Movies

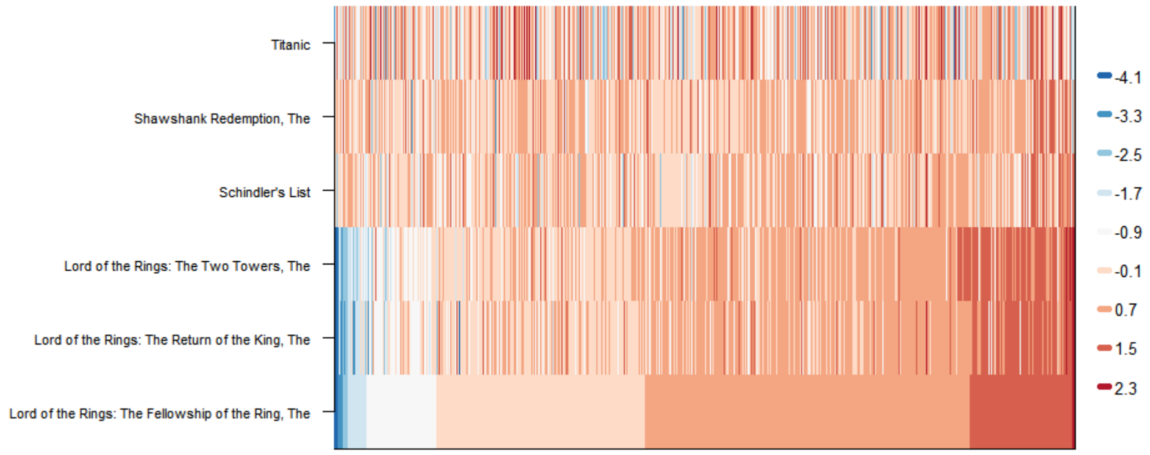


Chart 13: Actual Rating Image for Interactions between Users and Movies

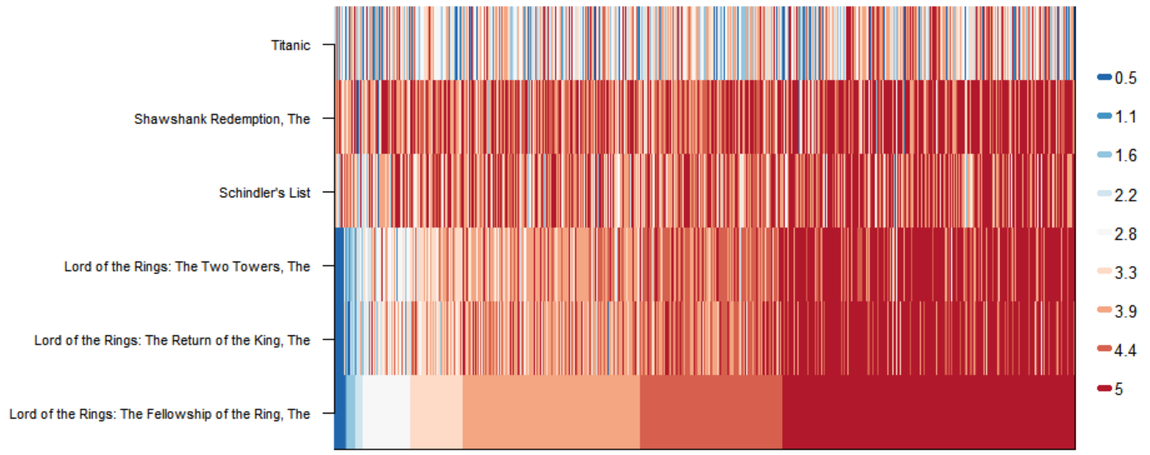


Table 6: Interactions between Users and Movies

tidy_title	userId 70078	userId 24544
Lord of the Rings: The Fellowship of the Ring, The	5.0	5.0
Lord of the Rings: The Return of the King, The	5.0	5.0
Lord of the Rings: The Two Towers, The	5.0	5.0
Schindler's List	5.0	3.5
Shawshank Redemption, The	5.0	2.0
Titanic	0.5	4.5

From the residual image and actual rating image, we see different users do show divergent preference. For example, some people like *Lord of the Rings* series very much (I am a big fun too), but some people are just not into them. *Titanic*, well-known box office record maker, is clearly not loved by everyone, for example, user 70078 gave *Titanic* a 0.5, but enjoyed *Lord of the Rings* series a lot. *Schindler's List* and *The Shawshank Redemption* are very classic to some, but still not everyone feels the same, for instance, to user 24544, it was just so so. We are not surprised to know this, because in real life it is normal that we have different tastes, but from both images, we see that our baseline + time bias model did not capture those differences, since the pattern of different users' preference is still there, although it is weakened in the residual image, compared with in the actual rating image. It is not surprising to know this, because each parameter we got by now represents only one factor (from user perspective, or movie perspective, or time), and we need a way to express those interactions between users and movies.

Singular Value Decomposition (SVD) is widely used in Recommendation Systems to find latent factor model, such as these interactions between users and movies. For details, please refer to the eighth course of the program series, Machine Learning, and there is plenty of related info online.

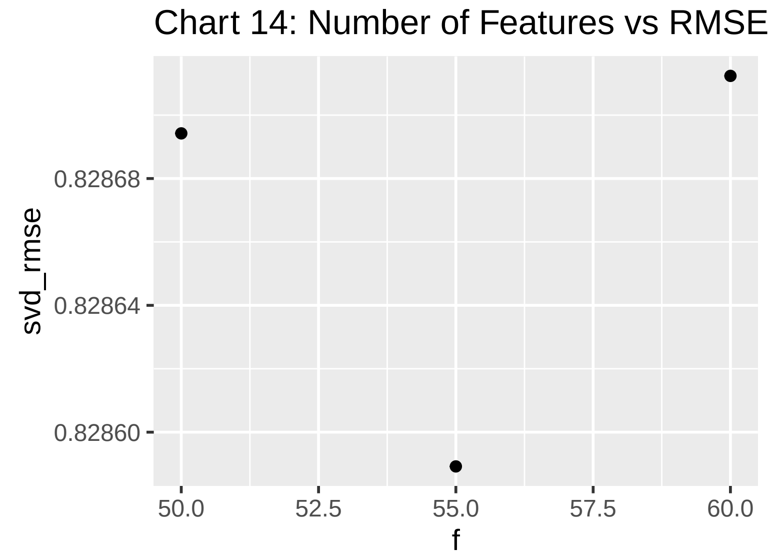
Here we have our residual matrix Y which is the matrix format of userId/movieId/res triplets in **train_set** with userId as row names and movieId as column names. We want to decompose Y, a $m \times n$ real matrix, as the product of three matrix as below, where U is a $m \times f$ user feature matrix, V is a $n \times f$ movie feature matrix, and Σ is a $f \times f$ diagonal matrix with singular values on the diagonal. m is the number of users in **train_set**; n is the number of movies; and f is the number of top factors/features contributing the most of all features (or the number of top singular values).

$$Y_{m,n} = U_{m,f}\Sigma_f {V_{n,f}}^T \tag{11}$$

Since we have all users and movies in our **train_set**, so U and V have all user features and movie features that we need to predict residuals of baseline + time biases model for **test_set**.

Because our **train_set** is very big, it will crash R and the laptop, if trying to convert userId/movieId/res triplets to matrix or use *svd* function in *base* package to do SVD for the matrix, here we use *Matrix* package to convert the triplets to sparse matrix, then implement SVD on the sparse matrix with the help of *irlba* package*[3]*.

Since f is a tunable parameter, so as always, we find the best f by cross-validating on the **cv_set**. below chart shows that the minimun RMSE happens at $f = 55$.



3. Results

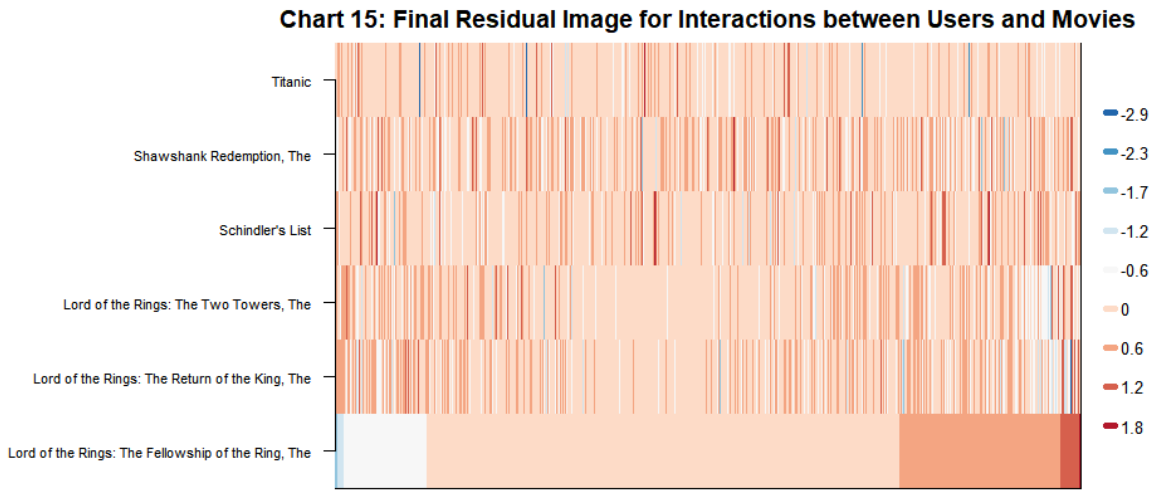
Predict on test set and evaluate with RMSE

Finally our model is a hybrid model with the combination of baseline predictors, time bias predictors and SVD, shown as below, where $\vec{u_u}$ is the feature vector of user u, and $\vec{v_i}$ is the feature vector of movie i.

$$\hat{r_{u,i}} = \hat{\mu} + \hat{b_u} + \hat{b_i} + \hat{b_{ry}} + \hat{b_{rm}} + \vec{u_u}\Sigma_{55}\vec{v_i}^T \tag{11}$$

Before we validate with **test_set**, we want to see if our model takes those user-movie interactions into account or not, or in other words, if the preference pattern is eliminated or further weakened. We again check the residual image from this model with the same data set we used to produce the previous two data images (chart 12 & 13), this time we only generate the image of the final model's residual. Final residual's formula and images are as below.

$$res_{u,i} = r_{u,i} - \hat{r_{u,i}} = r_{u,i} - (\hat{\mu} + \hat{b_u} + \hat{b_i} + \hat{b_{ry}} + \hat{b_{rm}} + \vec{u_u}\Sigma_{55}\vec{v_i}^T) \tag{12}$$



From the image, we can see the preference pattern is weakened considerably compared with the residual image of the baseline + time biases model, indicating the SVD model works.

As always, let's check its performance by predicting on **test_set** and evaluating the result with RMSE.

```
[1] "RMSE for baseline + time biases + svd model is 0.836897261274131"
```

RMSE drops from 0.8642656 to 0.836897, and meets our project requirement of being lower than 0.8649.

Additional : predicting with the help of *recosystem* package

We can also use *recosystem* package to build the latent factor model, which is a R wrapper of the *LIBMF library*.

```
[1] "RMSE for latent factor model with recosystem package is 0.793563926655046"
```

The package produces more accurate model than the one built in this report, since the final RMSE on **test_set** is 0.793564. For details, please refer to references *[4]*. Code details can be found in the corresponding R file in the project package.

4. Conclusion

summary of the report

The report is for Edx HarvardX Data Science Capstone project, which aims to build a movie recommdation system and meet the project requirement of lowering the RMSE on **test_set** to less than 0.8649. It derived three models along the analysis: baseline model, baseline + time biases model and baseline + time biases + SVD model, and demonstrated the advantage of hybrid models and reason why it makes sense to do so. The baseline + time biases + SVD model achieved the project's goal with final RMSE at 0.836897.

The report also mentioned another more accurate way of building latent factor model with the help of *recosystem* package.

Limitations and future work

Limitations

The report just derived very basic and simple models for the purpose of movie rating prediction, which is very entry-level. In fact, there are many advanced techniques and algorithms published in literatures and acdemic papers, so there is much to learn and explore.

Future work

Recommendation Systems are widely used in a variety of areas, such as what products to buy, what news to read, what music to listen and so on. Netflix, YouTube, Amazon and Yahoo Music all have their own Recommendation Systems in order to understand and serve their customers better. Hope after completing this program, we have the opportunity to apply these methods learnt in our work and discover more.

References

[1] Rafael A. Irizarry : Introduction to Data Science: Data Analysis and Prediction Algorithms with R, 2019

[2] Yehuda Koren : The BellKor Solution to the Netflix Grand Prize, 2009

[3] Jim Baglama, Lothar Reichel, B. W. Lewis : Package 'irlba', 2019

[4] Yixuan Qiu, Chih-Jen Lin, Yu-Chin Juan, Wei-Sheng Chin, Yong Zhuang, Bo-Wen Yuan, Meng-Yuan Yang, and other contributors : Package 'recosystem', 2017