Recommender system

1 Introduction

The first project is about recommender systems, which are a subclass of information filtering systems that seek to predict the "rating" or "preference" of a user (according to wiki: link.). In our case we are interested in movie ratings of users and that's what we want to predict. There are many ways to achieve that – we've picked only 3 methods that are based on mathematical theory.

2 Dataset

Our dataset comes from site grouplens and contains basic information about users: their id's, their ratings of a movie (which has its own id) and some other information that's irrelevant in our case. There are around 10^5 ratings from around 600 users and 9000 movies. There are a lot of ratings missing, most of the users rated only a few movies.

userId	movieId	rating	timestamp
1	1	4.0	964982703

Table 2.1: First row of `ratings.csv`.

2.1 Data preparation

To properly train and test our models, we need to split this dataset, so that training set contains around 90% of ratings of each user.

For our convinience we have converted the training data into sparse matrix $\mathbf{Z}_{n\times d}$ where n is the number of users and d is the number of movies, which contains ratings of movies. By analogy we have created matrix $\mathbf{T}$ that contains ratings from test set. Throughout the whole report we will use the same notation.

3 Quality of the system

Let $\mathcal{T}$ denote a set of pairs with ratings present in a test set. Assuming that algorithm after training on $\mathbf{Z}$ computes $\mathbf{Z}'$, then the quality is computed as root-mean square error:

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}|}\sum_{(u,m)\in\mathcal{T}}(\mathbf{Z}'[u,m]-\mathbf{T}[u,m])^2}.$$

4 Non-negative matrix factorization

Non-negative matrix factorization (abbrv. NMF) aims at approximating $n \times d$ non-negative matrix $\mathbf{Z}$ as a product of two matrices, $\mathbf{W}_{n\times r}$ and $\mathbf{H}_{r\times d}$. NMF's goal is to minimize the distance between $\mathbf{Z}$ and $\mathbf{WH}$, so in mathematical terms:

$$\underset{\mathbf{WH}}{arg\,\min} dist(\mathbf{Z},\mathbf{WH}), \quad s.t\ \mathbf{WH} \geq 0.$$

Often the Frobenius norm is considered:

$$dist_{Frob}(\mathbf{Z},\mathbf{WH}) = \sum_{i=1}^{n}\sum_{j=1}^{d}(\mathbf{Z}(i,j)-\mathbf{WH}(i,j))^2.$$

Popularity of this norm is related to assumption of $\mathbf{Z}$ being of form $\mathbf{WH}+\mathbf{N}$ where $\mathbf{N}$ is Gaussian noise. Optimal approximation can be computed efficiently using truncated SVD (which will be discussed later in sec. 5). It's also easy to reformulate problem when we are missing values – the sum is calculated only on non-missing values.

5 Singular value decomposition

Singular valur decomposition (abbrv. SVD) is based on two theorems: SVD and truncated SVD. The first states that any $\mathbf{Z}_{n\times d}$ matrix can be written as a product of three matrices:

$$\mathbf{Z} = \mathbf{U}_{n\times d}\mathbf{\Lambda}^{1/2}_{d\times d}\mathbf{V}^T_{d\times d}.$$

The second one states that taking first r columns of each of product matrices $\mathbf{Z_r} = \mathbf{U}_r\mathbf{\Lambda}_r^{1/2}\mathbf{V}_r^T$ gives an error:

$$\|\mathbf{Z} - \mathbf{Z_r}\|^2 = \sum_{i=r+1}^{d} \lambda_i.$$

Where λ_i are eigenvalues from matrix $\mathbf{\Lambda}$. Given small eigenvalues the error is also small and matrix $\mathbf{Z_r}$ a good approximation of $\mathbf{Z}$.

6 Singular value decomposition 2

Singular value decomposition 2 is a modification of SVD (sec. 5), that relies on repeating the algorithm until convergance. It leaves known values and changes missing values according to SVD. For this algorithm we need to replace missing values and we will start with 0. Let $SVD_r[\mathbf{Z}] = \mathbf{W}_r\mathbf{H}_r$ and set $Z^{(0)} = SVD_r[\mathbf{Z}]$:

$$Z^{(n+1)} = SVD_r\begin{bmatrix}\mathbf{Z} & \text{where } z_{ij} > 0\\ \mathbf{Z}^{(n)} & \text{elswehere}\end{bmatrix}$$

Theoreticaly the algorithm is not proved to converge, but it often does, especially in our case. We've chosen to stop iterating when RMSE between iterations is small.

7 Stochastic gradient descent

8 Parameters

The main task of this project was to fit parameters of the models, so that they give the best RMSE (sec. 3). There are 2 parameters to fit: imputed value and number of dimension r (reduction of original matrix). Fitting of those parameters was done by creating models and comparing their RMSE's. Parameteres that gave the smallest RMSE were choosen for each model.

As for imputed values, we've choosen 5 different aproaches: 0, median over all the ratings, average over all the ratings, median for each user and average for each user. Parameter r was searched over $1, 2, \ldots, 20$ and $40, 60, \ldots, 200$.

8.1 Results

All of the results are presented in appendix (A), in this section we will focus only on the best results. In case of imputed value user average always coincides with decrease of RMSE for every model, which is presented in tables 8.1. Selection of parameter r was different in each case. Besides computing RMSE, we've also calculated the time that is needed to complete the algorithm. Both graphs of RMSE and time versus r are presented in figures 8.1 and 8.2, vertical lines mark the smallest RMSE.

	0	avg	med	user avg	user med
NMF	6	17	12	18	16
SVD1	7	10	6	11	4
SVD2	3	2	2	2	2

(a) Selected parameter r for each case.

	0	avg	med	user avg	user med
NMF	2.774	1.043	1.168	0.936	0.980
SVD1	2.739	1.041	1.167	0.934	0.979
SVD2	1.366	0.934	0.953	0.899	0.910

(b) RMSE for best r in each case.

	0	avg	med	user avg	user med
NMF	2.296	5.427	4.403	5.182	5.776
SVD1	0.400	0.374	0.465	0.416	0.290
SVD2	9.835	9.635	13.924	17.309	17.466

(c) Time for best r in each case.

Table 8.1: Results of parameter selection.

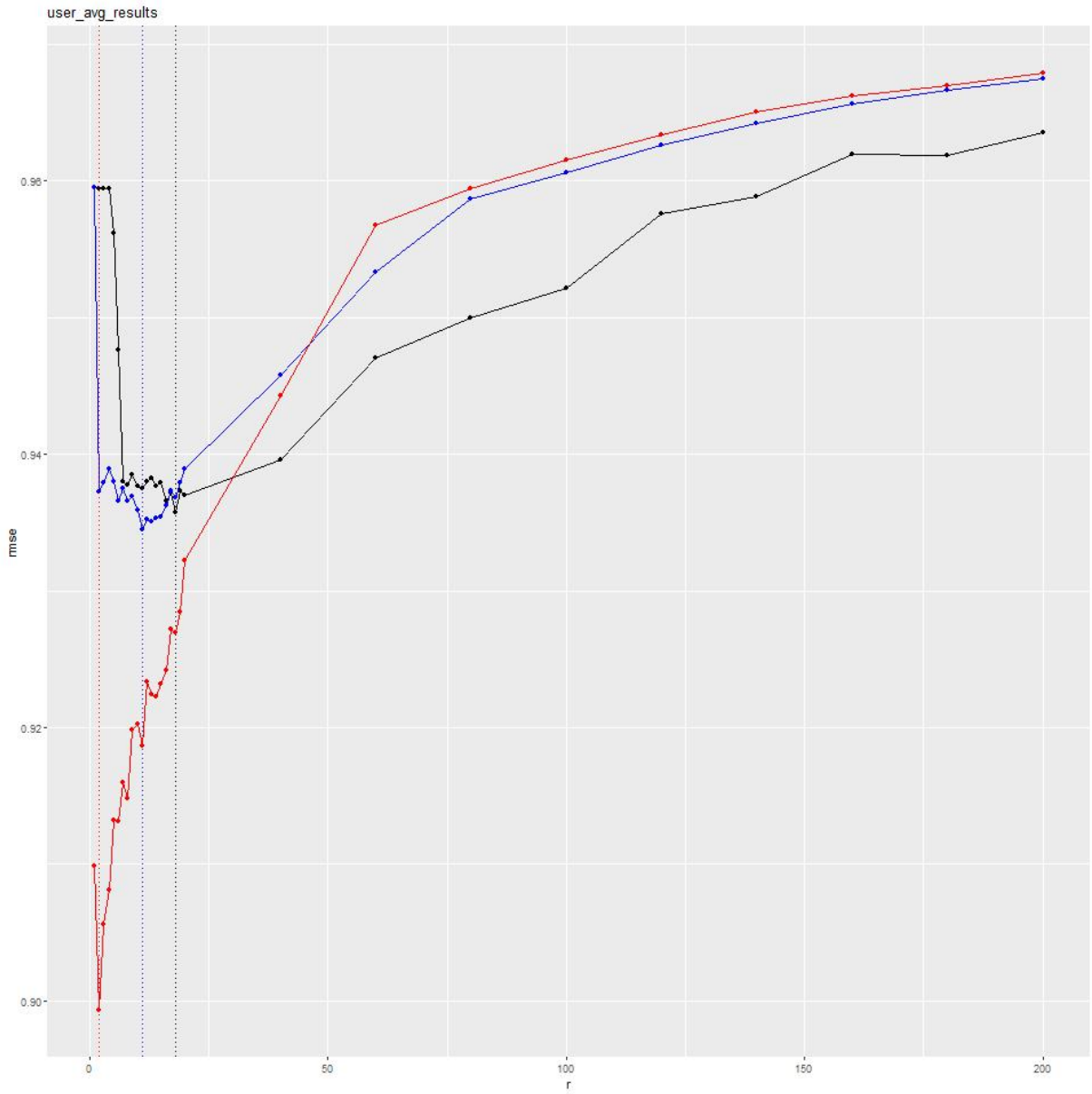


Figure 8.1: RMSE of models with missing values imputed as average of corresponding user. Black is NMF, blue is SVD1, red is SVD2.

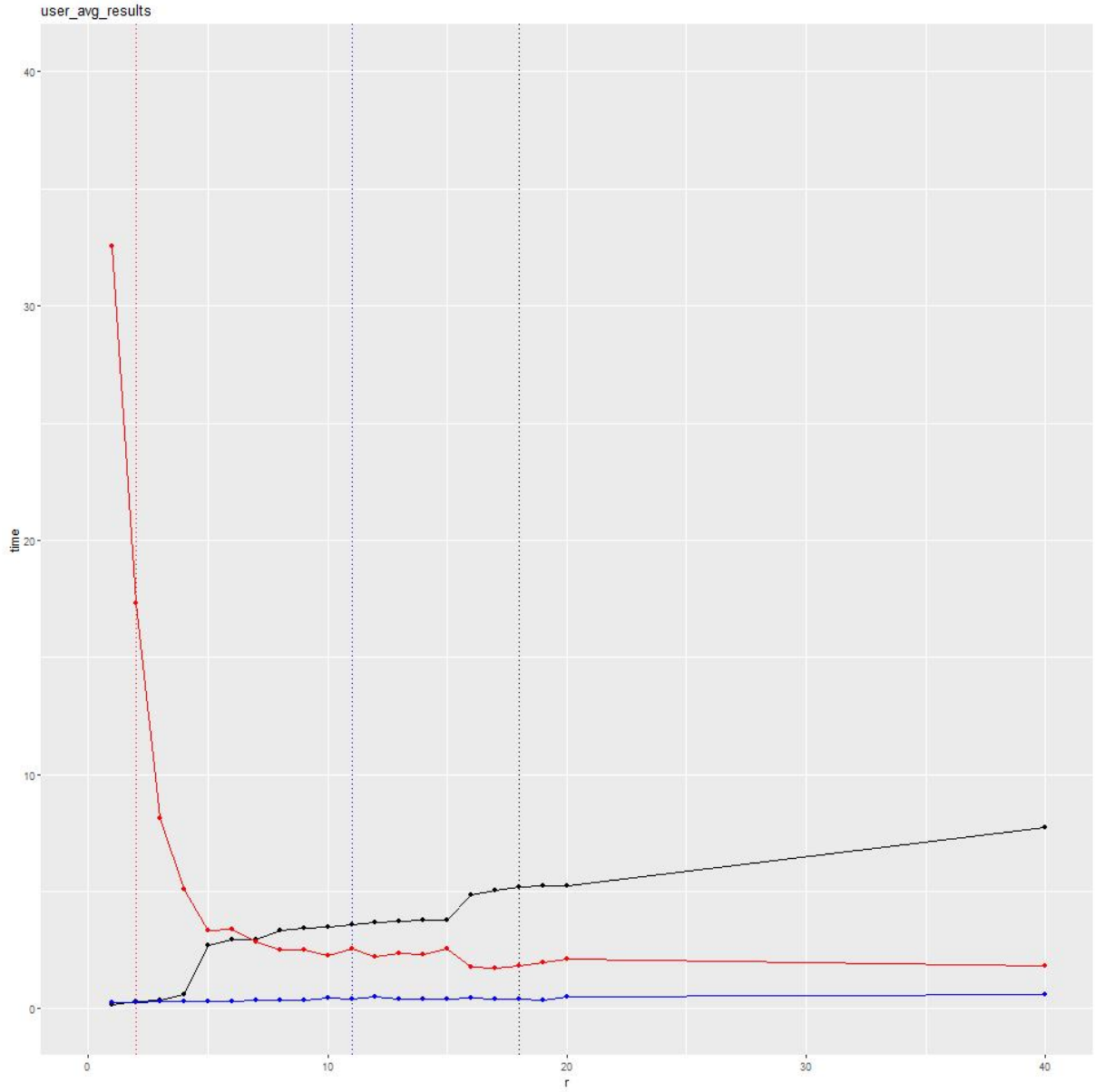


Figure 8.2: Time of models with missing values imputed as average of corresponding user. Black is NMF, blue is SVD1, red is SVD2.

9 Conclusions

9.1 RMSE

General conclusion that can be drawn is that each algorithm can reach decent results, that do not differ much. Every tested algorithm gets to around 0.9 in its best conditions. What's more interesting, those conditions also coincide.

Every algorithm performed best with imputed values as user averages and it gave the algorithm tremendous advantage over the basic model with imputing zeros. Starting with zeros gave results that varied a lot from others and were on a verge of being useful. Median values were always a bit worse than its counterparts with average.

Parameter r differed more, but always stayed in range $[1, 20]$ which is great for reducing number of computations. NMF and SVD1 needed more dimensions to reach lower RMSE, than SVD2, which is understandable, because of its iterative approach. With more dimensions SVD2 needed less iterations and because of that its RMSE was close SVD1.

9.2 Time

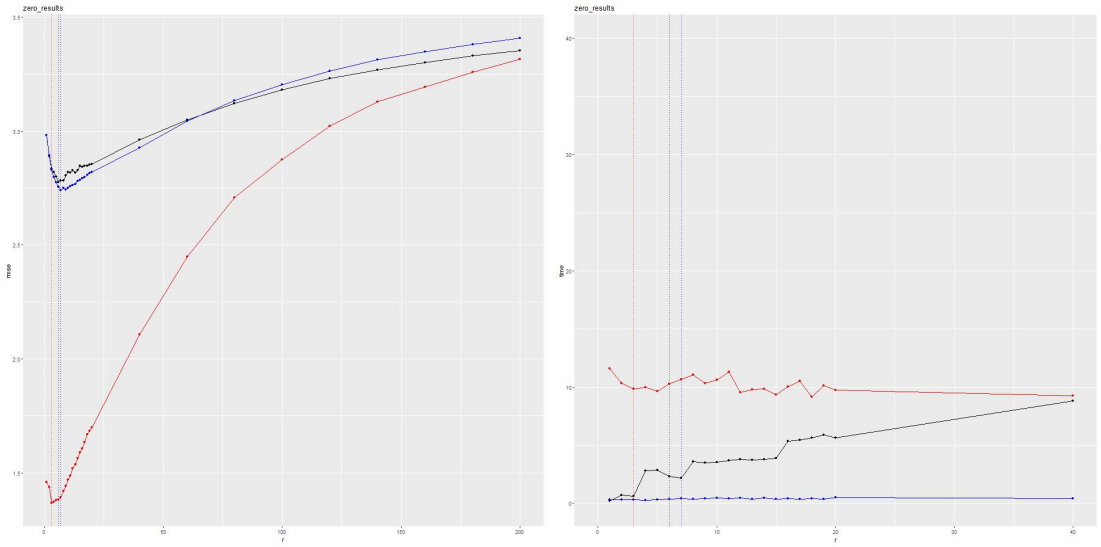
The graphs that present time necessary to complete algorithm are cut, to show only first 50 values of r, partialy because of not much happening in further functions and partialy because the best results were on interval $[1, 20]$. NMF time was always growing and took drasticaly more time than SVD's. SVD1's time grew slowly and linearly and SVD2's time got close to SVD1 because less iterations were needed. In case of low r SVD2 needed more iterations, probably because of less complicated model.

9.3 Final thoughts

The best algorithm turned out to be SVD2 and the quickest SVD1. SVD2 takes considerably longer than SVD2 in lower dimensions, but gives better results there, so the choice of algorithm depends on budget of time. If there are planned many computations it could be better to choose SVD1 as it gives similar, but always worse results. NMF is not worth considering as it is always the slowest and has the highest RMSE.

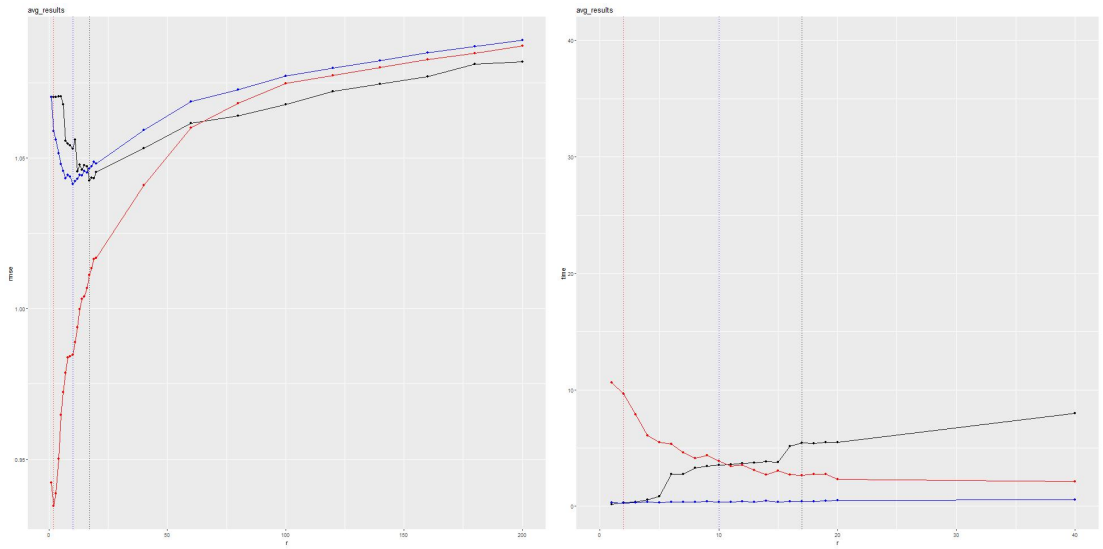
Appendices

Appendix A



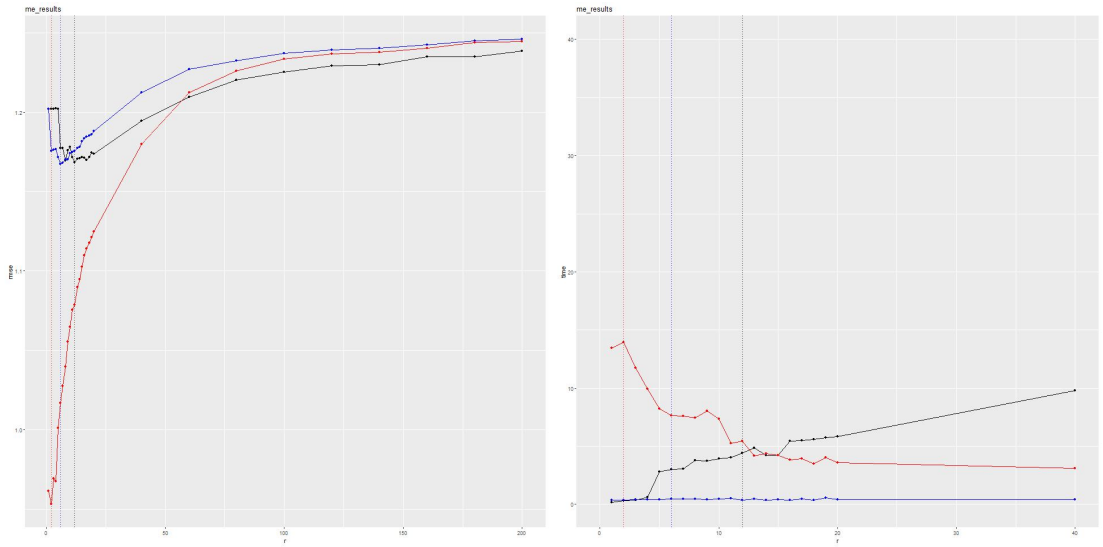
(a) RMSE. (b) Time.

Figure A.1: Results of models with missing values imputed as 0. Black is NMF, blue is SVD1, red is SVD2.



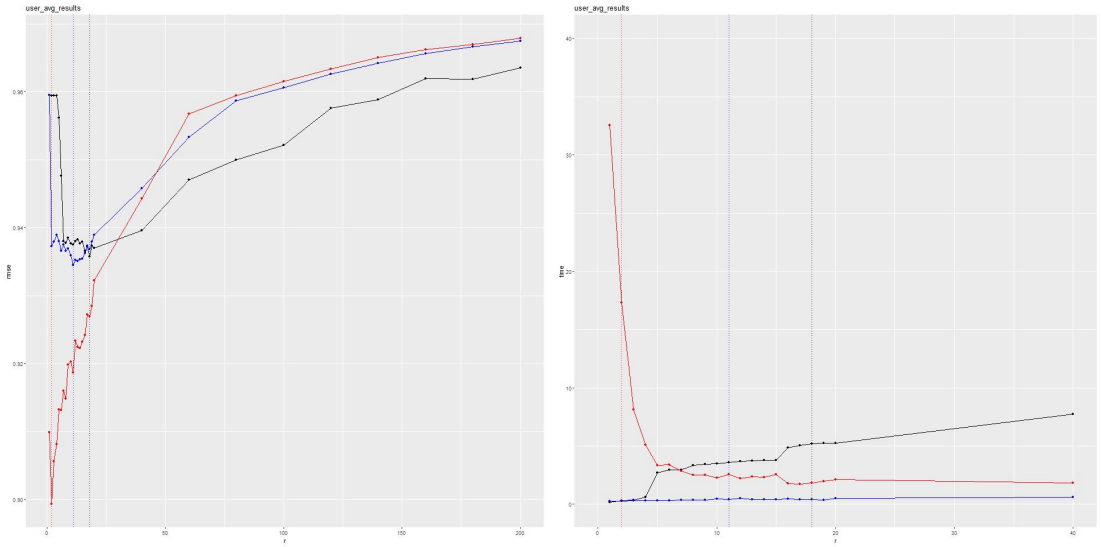
(a) RMSE. (b) Time.

Figure A.2: Results of models with missing values imputed as average over all ratings. Black is NMF, blue is SVD1, red is SVD2.



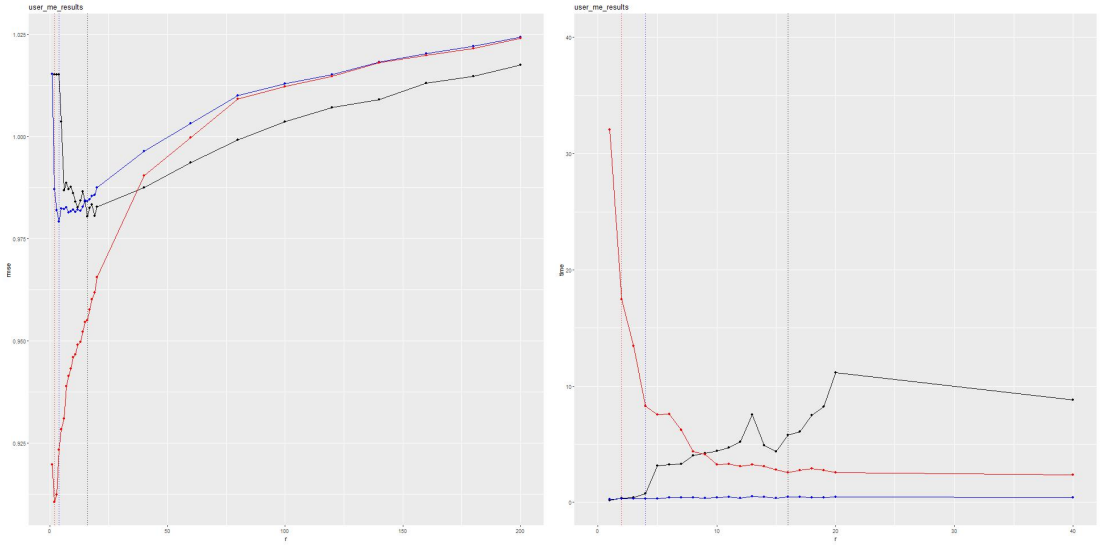
(a) RMSE. (b) Time.

Figure A.3: Results of models with missing values imputed as median over all ratings. Black is NMF, blue is SVD1, red is SVD2.



(a) RMSE. (b) Time.

Figure A.4: Results of models with missing values imputed as average of corresponding user. Black is NMF, blue is SVD1, red is SVD2.



(a) RMSE. (b) Time.

Figure A.5: Results of models with missing values imputed as median of corresponding user. Black is NMF, blue is SVD1, red is SVD2.