Data Analytics - Project Presentation
29/05/2018

Musical Instruments Recommendation

Alessia Ruggeri

Introduction

For my Data Analytics Project, my task is to **build two recommendation systems** using a Musical Instruments dataset and to use them to **predict the product ratings** for some users based on other users' ratings.

Two datasets have been given to me: the first one contains information about **user preferences** on musical instruments (ratings); the second one contains information about each rated item (**metadata**).

Exploration and description of data (1)

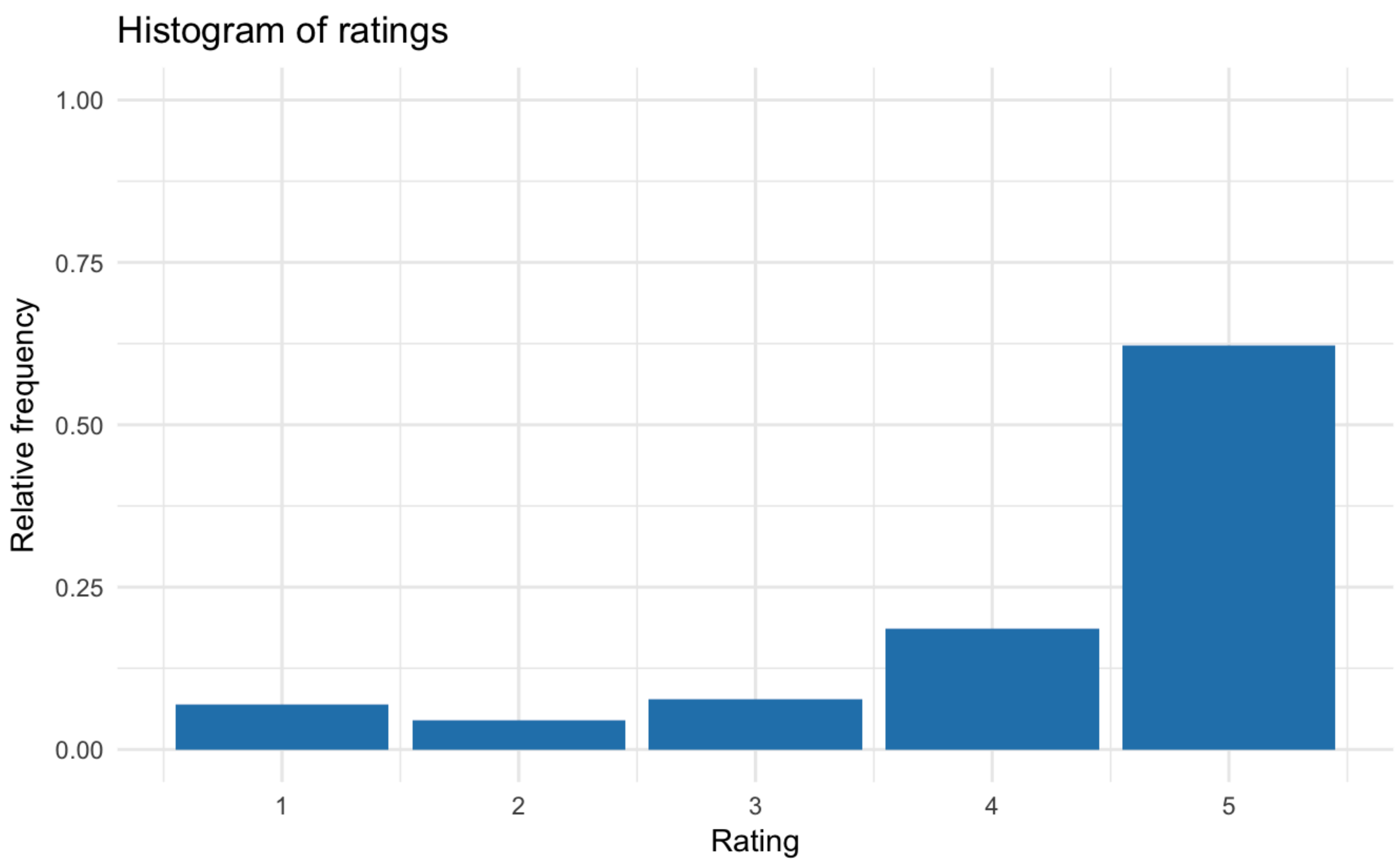
The **main dataset** contains four attributes, which are *user*, *item*, *rating* and *timestamp*, and 500,175 observations, each one representing an item's rating given by a user. This is the most interesting dataset, since it relates users and items by the corresponding rating.

The **second dataset** contains nine attributes and 84,901 observations, each one representing the metadata of one item. This dataset doesn't contain very interesting data for my purpose, furthermore more than a half of the metadata values are NA and can't be used.

Exploration and description of data (2)

Ratings' **standard descriptive statistics**:

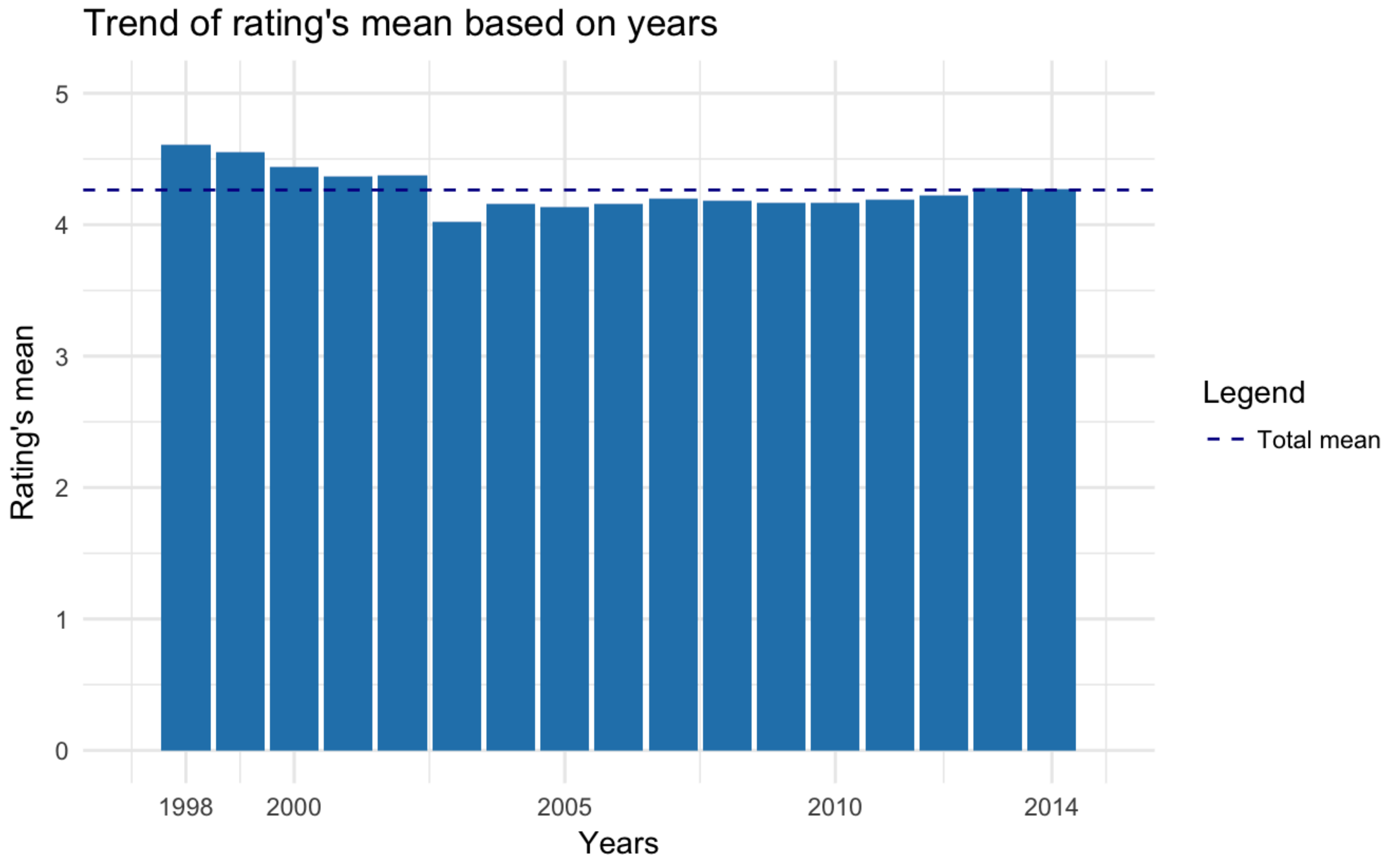
Min.	1.000
1st Qu.	4.000
Median	5.000
Mean	4.244
3rd Qu.	5.000
Max.	5.000
St. Dev.	1.203

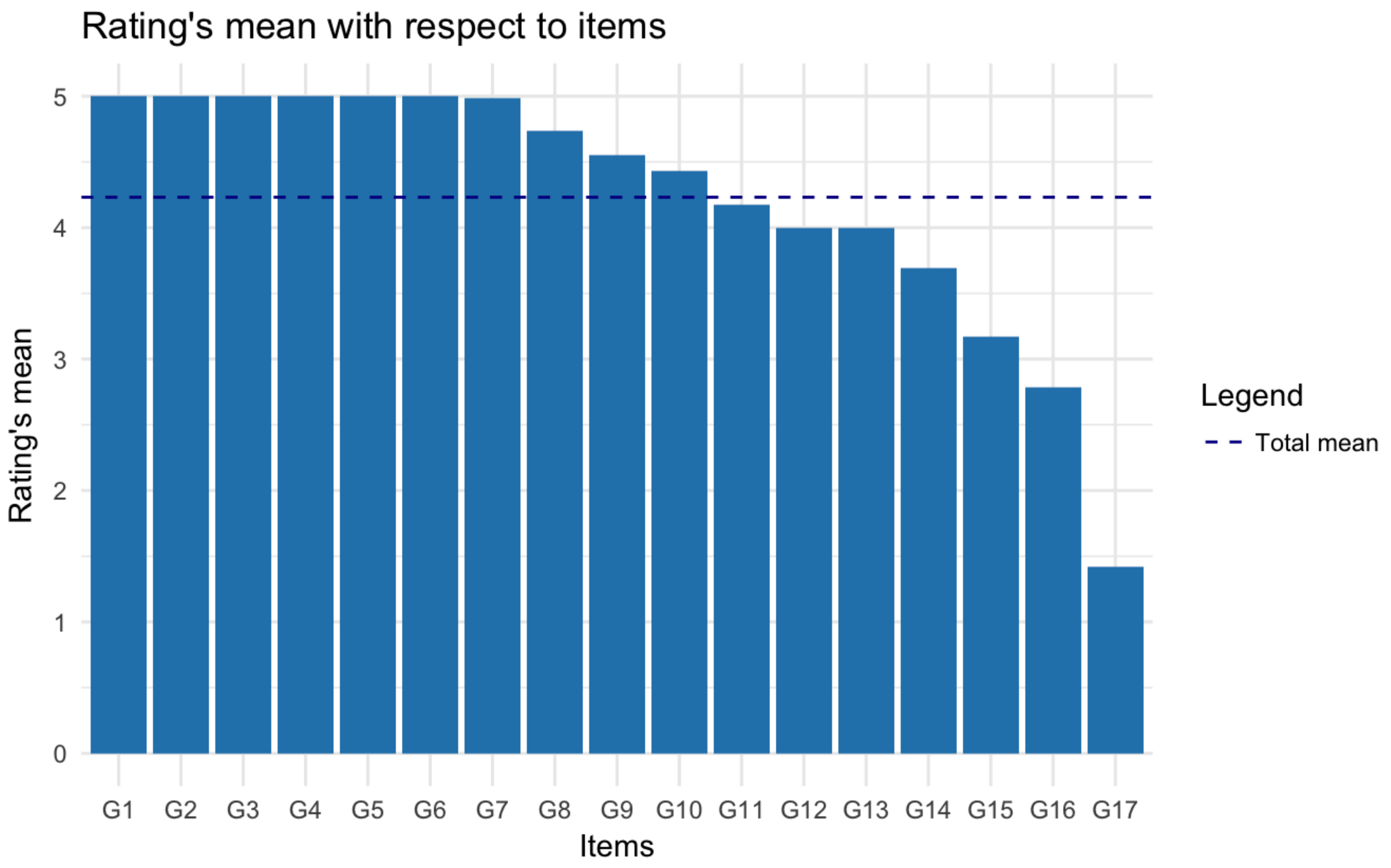


1	2	3	4	5
7.0%	4.5%	7.7%	18.7%	62.1%

Exploration and description of data (3)

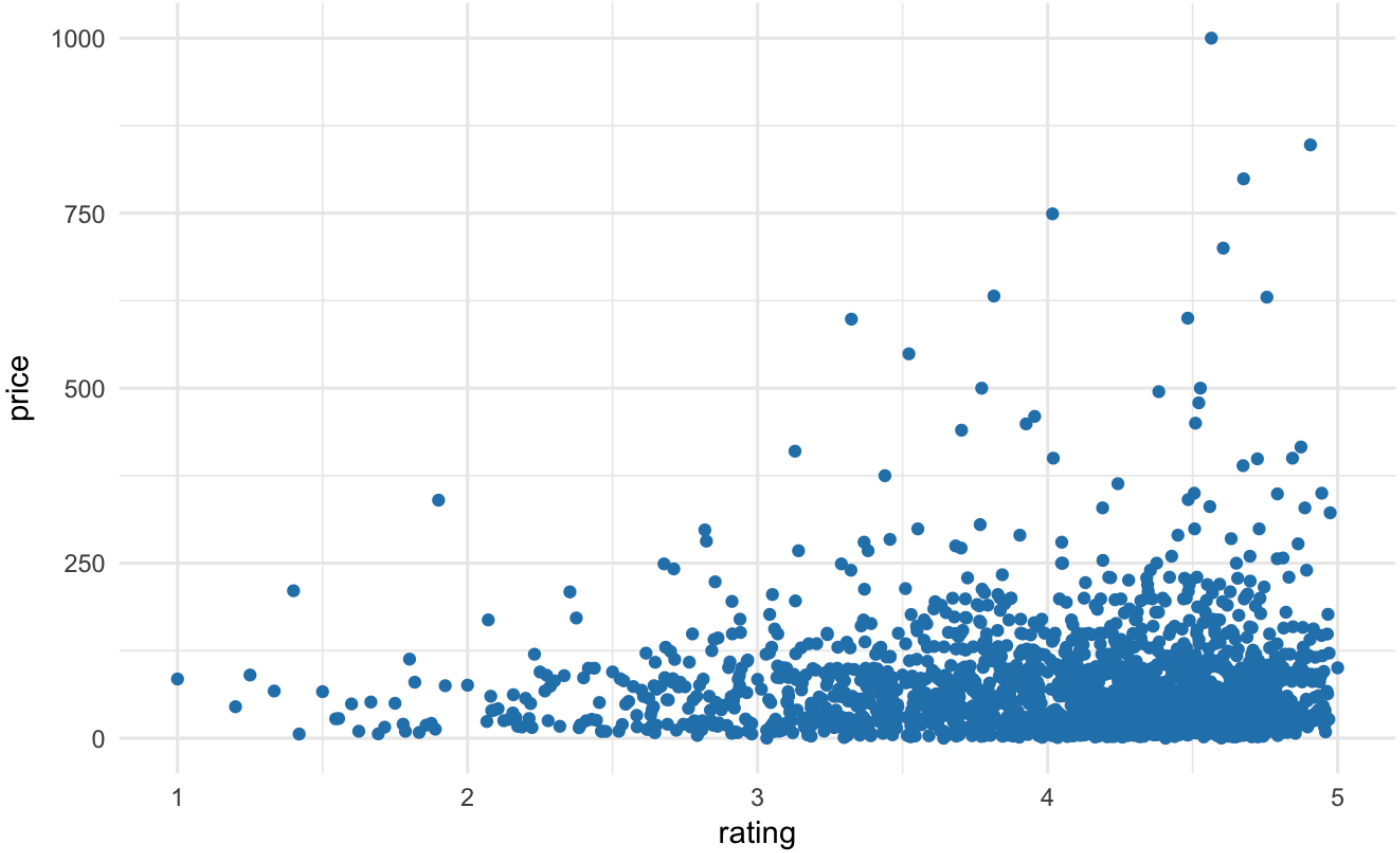
Some other **data representations**:





Exploration and description of data (4)

Relation between the ratings and the prices of items:



	rating	price
rating	1.0000	0.0097
price	0.0097	1.0000

The **correlation matrix** shows that there is no linear correlation between the variables *rating* and *price*. However, from the graph we can deduct the presence of a **non-linear relation** between the two variables.

Recommendation systems (1)

In order to build the two recommendation systems required, I used **collaborative filtering** (CF) in both cases: one of them makes use of the *user-based collaborative filtering* (UBCF), the other makes use of the *item-based collaborative filtering* (IBCF).

Collaborative filtering is an algorithm that uses given rating data by many users for many items as the basis for predicting missing ratings or for creating a list of items with the top-N items to recommend to a given user, called the active user.

The **UBCF** algorithms bases the recommendations on the similarity between users, while the **IBCF** algorithms bases the recommendations on the similarity between items.

Recommendation systems (2)

First, I **pre-processed data** by filtering users and items in order to speed up computation and reduce the sparsity of the utility matrix. Then, I build the **utility matrix**, where each row represents a user, each column an item and into each user-item cell it is stored the rating of that user for that item.

```
utility_matrix = dcast(rating, formula = user~item, value.var = 'rating')
```

After that, I created an **evaluation scheme** that determines what and how data is used for training and testing and that allows me to evaluate the two models at the end. I split the dataset using 75% of users for the training set and 25% for the test set, I considered good ratings only the ratings equal to 5 and I set to 4 the value of given ratings for each user in the test set.

```
e = evaluationScheme(utility_m, method = "split", train = 0.75, given = 4, goodRating = 5)
```

Recommendation systems (3)

I created the **two recommendation systems** by using the training set and setting the method, depending on whether the UBCF or the IBCF has to be used. Then, I used the created *Recommenders* in order to predict the missing ratings in the test set through the function *predict*.

```
# --- Recommendation system using UBCF
rUBCF = Recommender(getData(e, "train"), method = "UBCF")
pUBCF = predict(rUBCF, getData(e, "known"), type = "ratings")
# --- Recommendation system using IBCF
rIBCF = Recommender(getData(e, "train"), method = "IBCF")
pIBCF = predict(rIBCF, getData(e, "known"), type = "ratings")
```

Finally, I **evaluated the predictions** of the two models through an evaluation matrix.

```
# Error between prediction and unknown part of the test data
error = rbind(UBCF = calcPredictionAccuracy(pUBCF, getData(e, "unknown")),
                            IBCF = calcPredictionAccuracy(pIBCF, getData(e, "unknown")))
```

Recommendation systems (4)

	RMSE	MSE	MAE
UBCF	0.9631215	0.9276031	0.6462931
IBCF	1.3906379	1.9338738	0.9206462

In general, **the UBCF works better than the IBCF** with every configuration I tried, since all the three measures of error are higher in the UBCF than in the IBCF .

In all the different configurations I tried for the data filtering, obviously the higher the minimum number of ratings for each user and item, the lower the error of the predictions.

Thank you for your attention

Alessia Ruggeri