**Group #234: Movie Recommender System**

|  |  |  |
| --- | --- | --- |
| First Name | Last Name | Email |
| Sagar | Ippili | sippili@hawk.iit.edu |
| Ridhima | Bhalerao | rbhalerao@hawk.iit.edu |
| Siddhi | Kulkarni | skulkarni13@hawk.iit.edu |

Table of Contents

[**1.** **Introduction** 2](#_Toc8121450)

[**2.** **Data** 2](#_Toc8121451)

[**3.** **Problems to be Solved** 2](#_Toc8121452)

[**4.** **KDD** 2](#_Toc8121453)

[4.1. Data Processing 2](#_Toc8121454)

[4.2. Data Mining Methods and Processes: Recommender Systems 4](#_Toc8121455)

[**5. Evaluations and Results** 15](#_Toc8121458)

[5.1. Evaluation Methods 15](#_Toc8121459)

[5.2. Results and Findings 15](#_Toc8121460)

[**6. Conclusions and Future Work** 17](#_Toc8121466)

[6.1. Conclusions 17](#_Toc8121467)

[6.2. Limitations 17](#_Toc8121468)

[6.3. Potential Improvements or Future Work 17](#_Toc8121469)

# **Introduction**

Movie lovers just like us, would wait for Friday to come. While it is fun to just go to a movie theatre and hope to be pleasantly surprised by a new release, many people prefer for recommendations as they don't want to waste time and money for a low rated movie. So, we can make this easier for the movie buffs to overcome the dilemma of movie selection on a Friday eve by helping them with recommendations based on the ratings information from the previous user’s data.

# **Data**

MovieLens dataset is the collection of 2.7M movie reviews of 280,000 users. Users were selected randomly and here we focus on the types of movies each customer prefers. Depending on those preferences, our application can recommend new movies to the customer. The dataset consists of following columns:

* **Movie ID:** This is the ID assigned to each movie included in the dataset.
* **User ID:** This is the ID assigned to each user selected randomly.
* **Rating:** This is the rating value out of 5 assigned to each movie.
* **Year:** This is the year in which the movie was released.
* **Genres:** This column specifies type of movie such as action, comedy, etc.
* **Tags:** This column consists of tags i.e. user generated metadata for the movie.

This is a huge dataset (approx. > 2.7 million), and using this information, we predict the movies to customers according to their preferences using a recommender system.

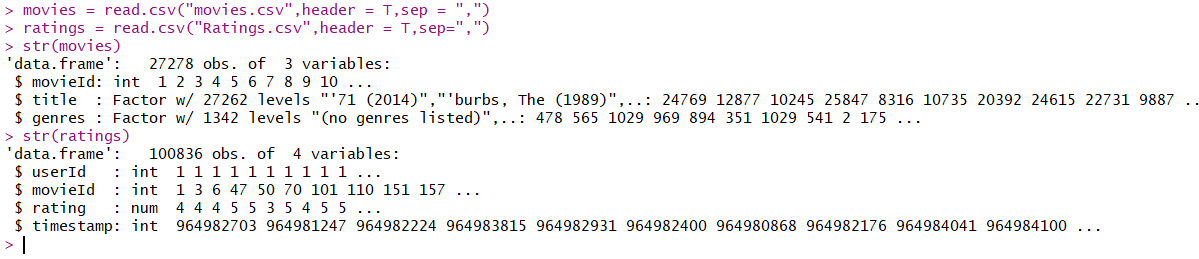
# **Problems to be Solved**

Here, the problem is of data overloading. It becomes difficult for a person to understand an issue and make decisions that can be caused by the presence of too much information. Here, we have huge information or dataset of movies, movie ratings and reviews. In this case, proper categorization and extraction of this information is necessary to get expected and understandable outcome.

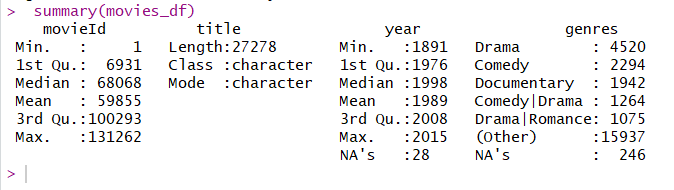
# **KDD**

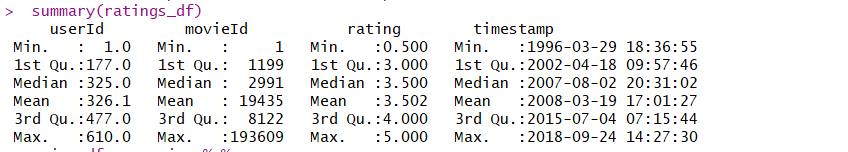
## Data Processing

1. **Importing dataset:**

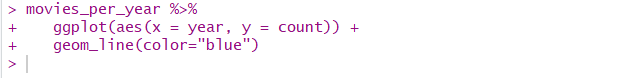


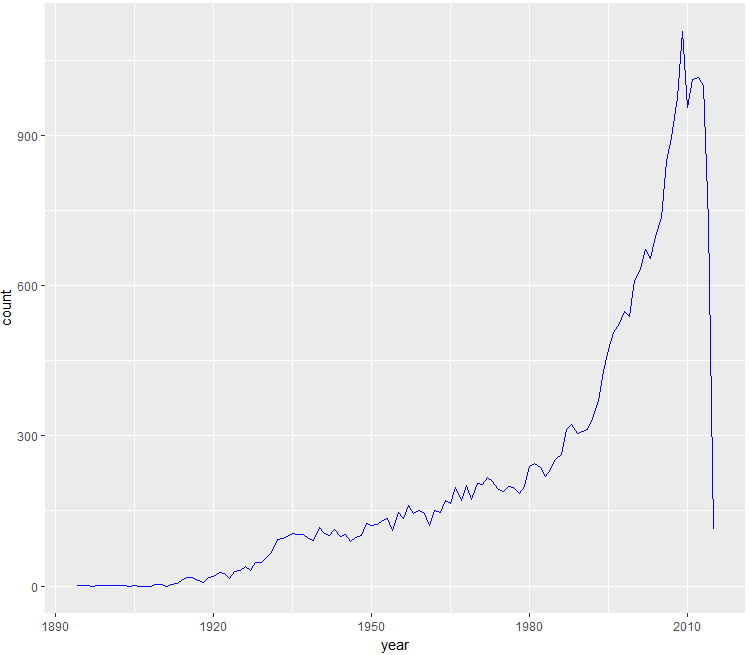
1. **Understanding Dataset**:



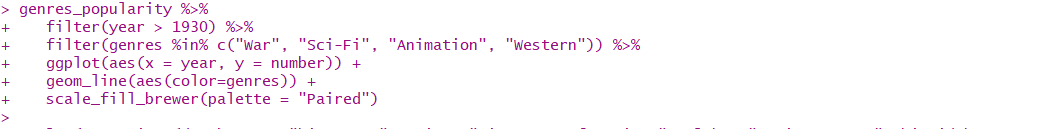


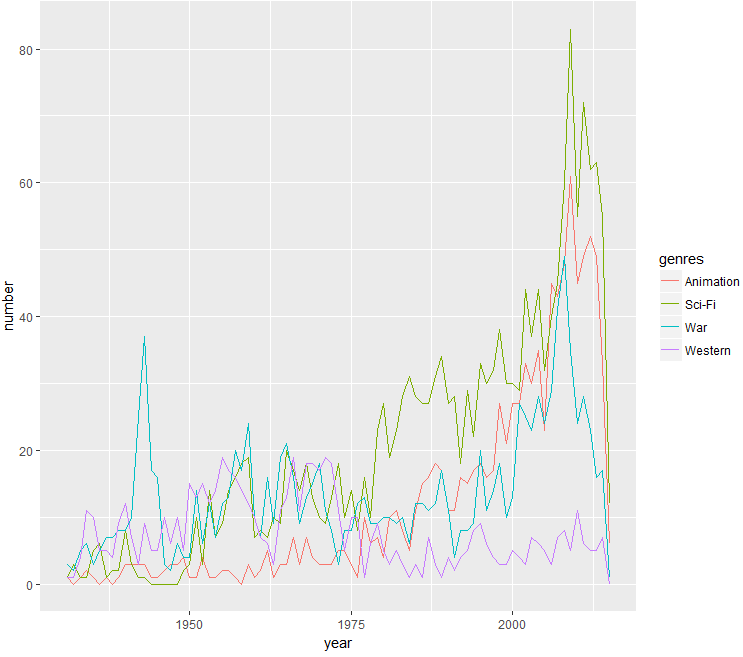
1. **Movies Per Year:**



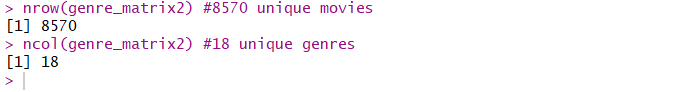


1. **Genre Popularity:**





1. **Unique Movies and Unique Ratings:**



## Data Mining Methods and Processes: Recommender Systems

* The goal of a recommender system is to help users – usually consumers – find what they want and discover new information that will help them. Recommender systems are ubiquitous now in online marketing – for books, music, health care services, the arts.
* Recommender systems may follow different paradigms. There are mainly two approaches used in Recommender systems:

1. Traditional Recommender Systems
2. New types of Recommender Systems

### Traditional Recommender Systems

There are two methods to construct traditional recommender systems:

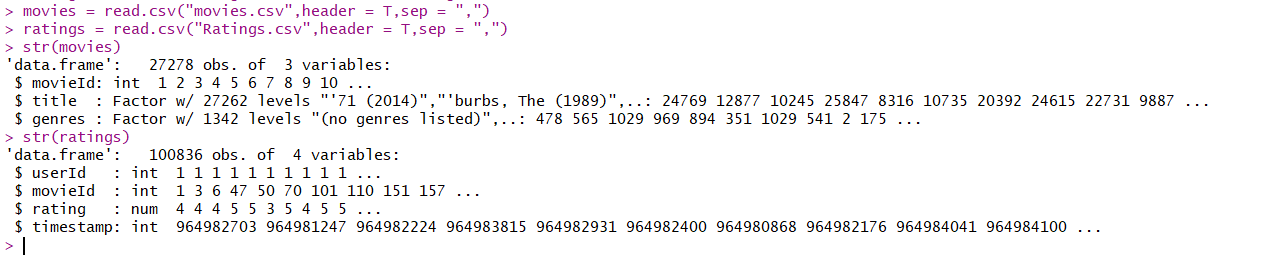
1. Collaborative Filtering: The user will be recommended items that people with similar tastes and preferences liked in the past, e.g., movie recsys. There are two categories of CF:

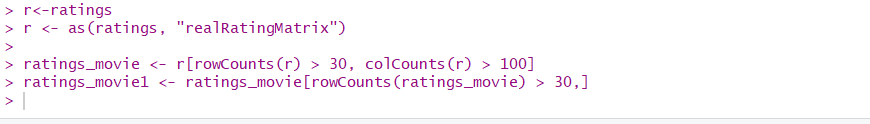
* User Based Collaborative Filtering: Measure the similarity between target users and other users. For each user, these steps are taken:

1. Measure how similar each user is to other users. As with IBCF, popular similarity measures are Pearson correlation and cosine similarity.
2. Identify the most similar users. Choices can be based similarity to the top k users (k-nearest\_neighbors).
3. Rate the items purchased by the most similar users, either by averaging or weighting the nearest users.

**Building Model**:

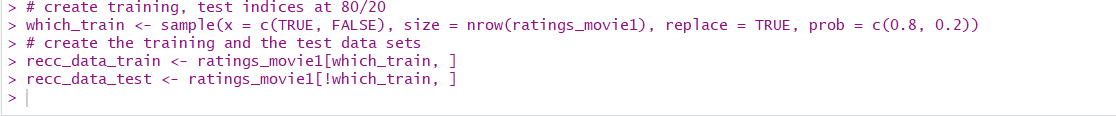
1. Loading dataset, pruning and converting into matrix:



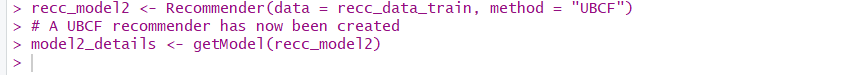


1. Splitting dataset into training and testing dataset. As dataset is large, we will perform hold out evaluation.

The Training and Testing dataset for all the models is the same which is as follows:



1. Building model using function ‘Recommender’ and predicting on test dataset:

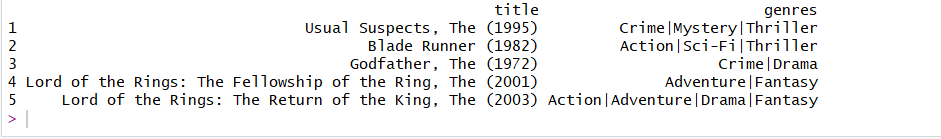




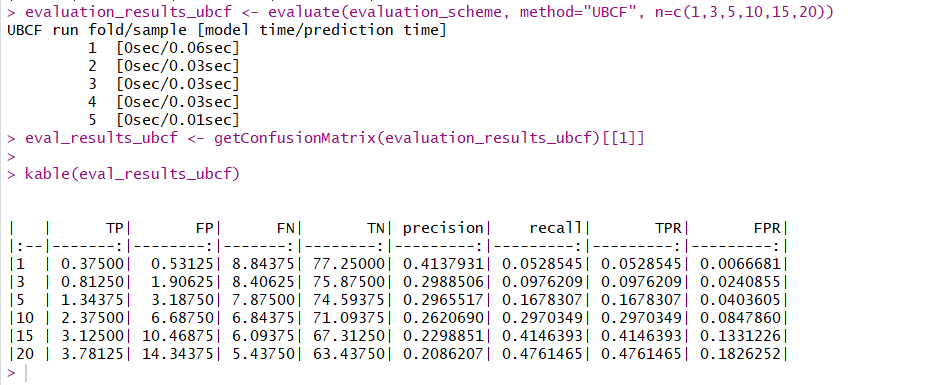




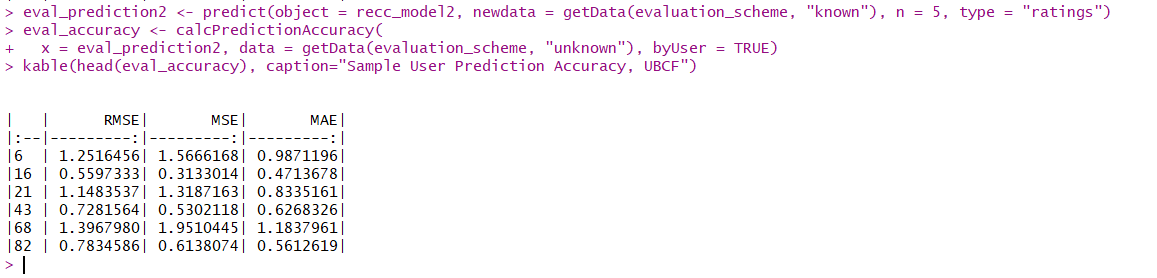
1. Predicted movies using UBCF are as follows:



1. Finding accuracies: We have calculated precision and recall as follows:



RMSE values are as follows:



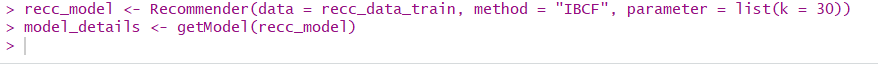
* Item-based Collaborative Filtering: Measure the similarity between the items that target users’ rates/interacts with and other items. The core algorithm is based on these steps:

1. For each two items, measure how similar based on user ratings.
2. For each item, identify the k-most similar items.
3. For each user, identify the items that are most similar to the user’s purchases.

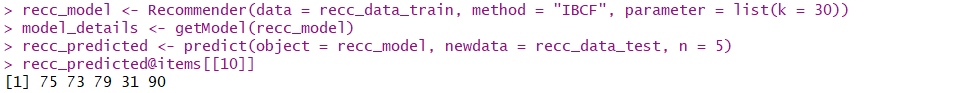
**Building Model:**

The train and test dataset used for building model using IBCF is same as datasets used in UBCF model. Therefore, all the steps are same as above till building the model.

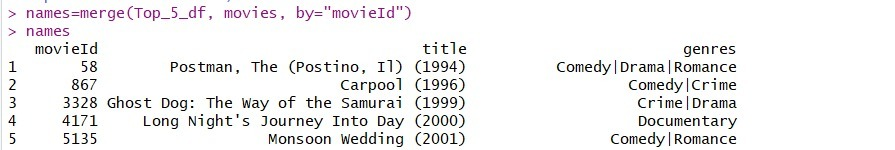
1. The model for IBCF is built using ‘Recommender’ function as follows:



1. The prediction and results are as below:

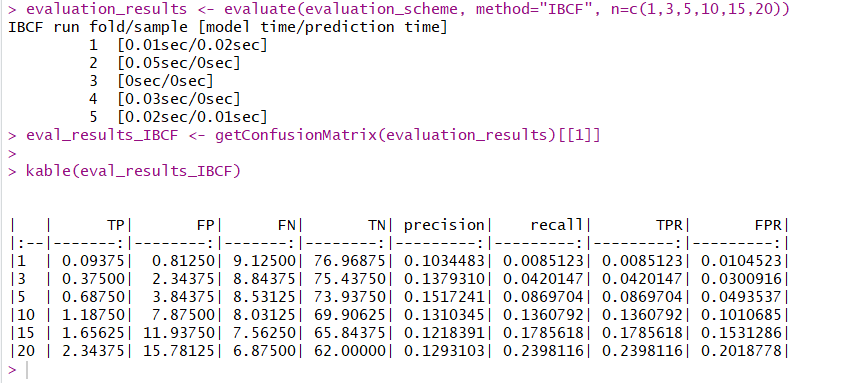


1. Recommended movies are as follows:

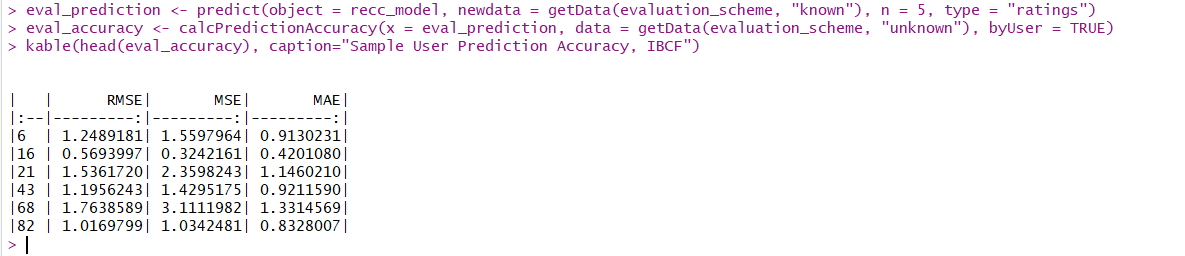


1. Finding Accuracies:

Here, I have used ‘evaluate’ function. Also, precision and recall values are obtained as follows:



1. RMSE values are as follows:



* Matrix Factorization: The matrix factorization involves decomposing a higher dimension matrix into two lower dimension matrices. We consider the latent factors for movies and users respectively:

**P (m x k) Q (k x n) Movies (n)**

**User Factors**



~

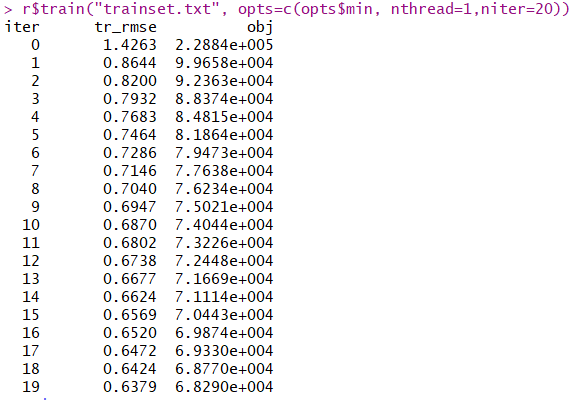
**Users (m)**

**Movie Factors**

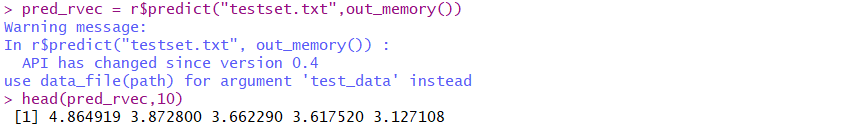
**Building model:**

1. Creating a recommender system object using ‘**recosystem’** library and building model by optimizing the parameters to achieve best results:

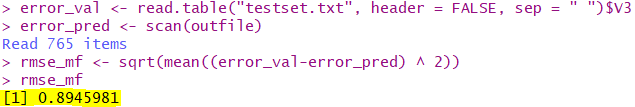




1. Generating Top 5 predictions:



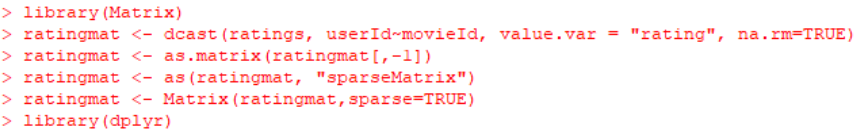
1. Calculating the RMSE:



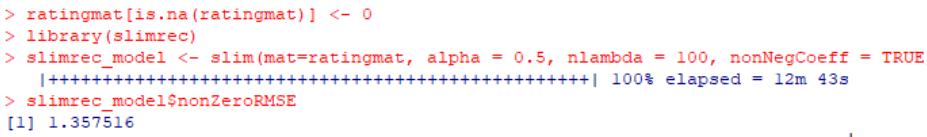
* Sparse Linear Method: SLIM produces top N recommendations by aggregating the user’s movies and rating profiles to generate the results. SLIM is a recommender model and cannot be compared with other models based on RMSE value. We built CF recommender system using SLIM and made the top 5 recommendations for a user.

**Building Model:**

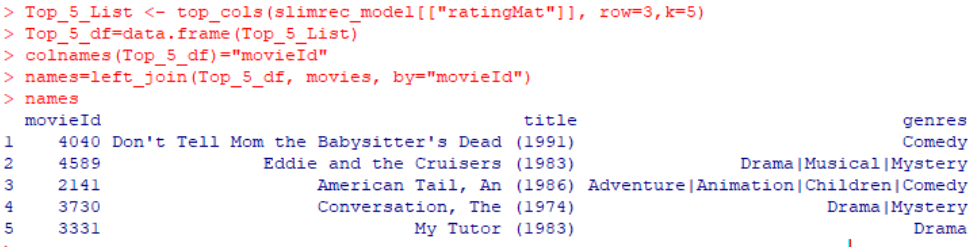
1. Import data and create a sparse matrix



1. Use ‘slimrec’ library for model building



1. List top 5 recommendations

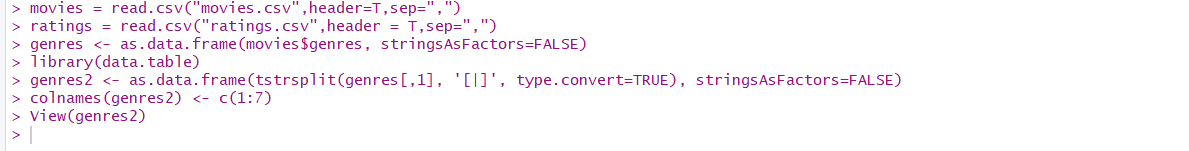


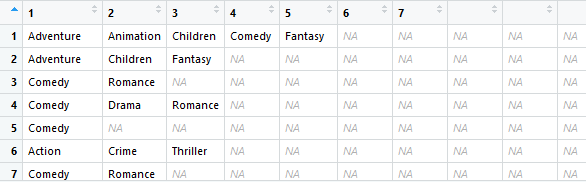
1. Content-based Filtering:

Content-based Filtering approach involves analyzing an item a user interacted with and giving recommendations that are similar in content to that item. For a movie recommendation engine, a content-based approach would be to recommend movies that are of highest similarity based on its features, such as genres, actors, directors, year of production, etc.

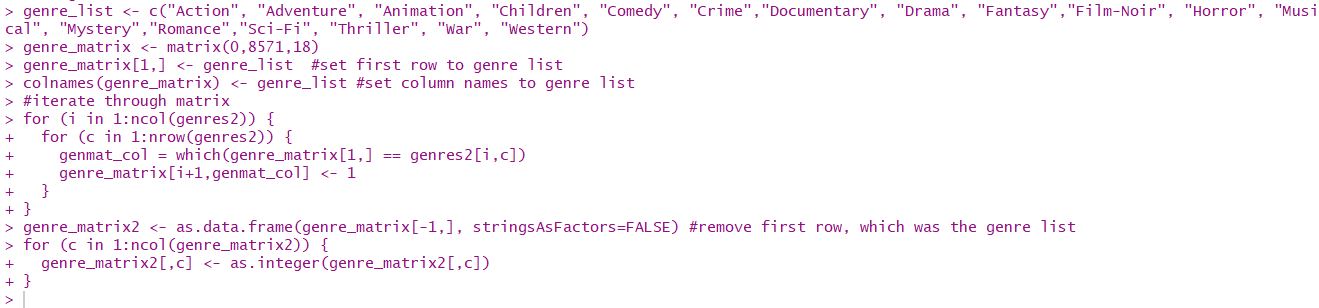
**Building Model**:

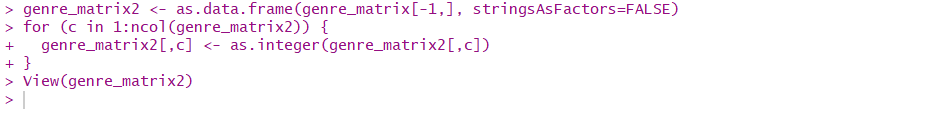
1. Loading and preprocessing dataset:

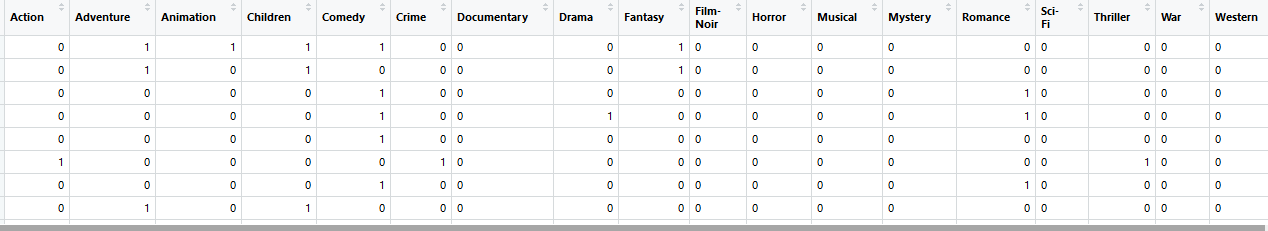




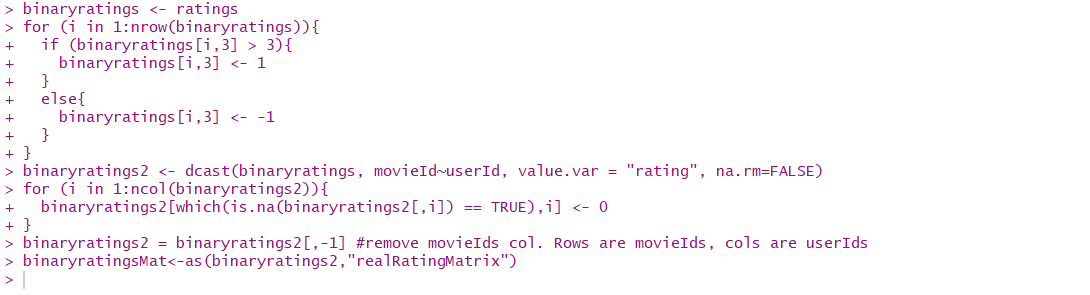
1. Finding unique movies and genres and assigning the value accordingly:



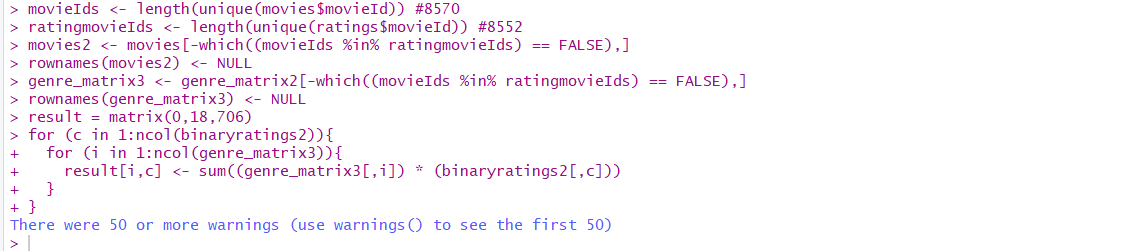




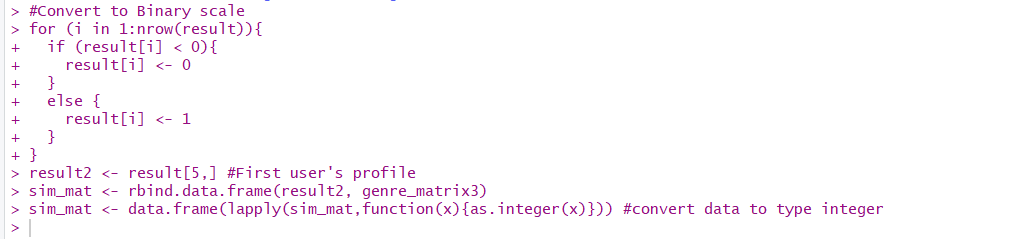
1. Converting into matrix and removing duplicating values:

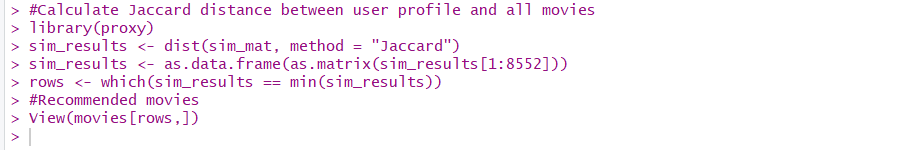


1. Calculating dot product of user profile:

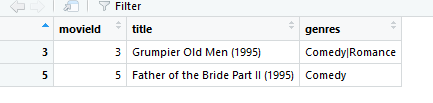


1. Converting into binary scale and finding similarity:





1. Result for content-based filtering:



### New Types of Recommender Systems:

New of Recommender system consists of different types such as,

1. Context Aware Recommender Systems
2. Multi-Criteria Recommender Systems
3. Group Recommender Systems
4. Human Factor Based Recommender Systems
5. Health Recommender Systems

Here, we have performed Context Aware Recommender systems.

* Context Aware Recommender Systems:

Context-aware recommender systems (CARS) generate more relevant recommendations by adapting them to the specific contextual situation of the user. Here we have built context-aware recommender systems using two techniques as follows:

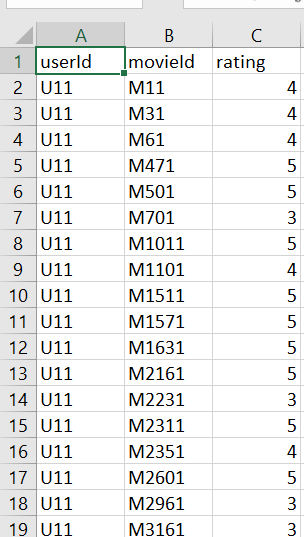
1. **Contextualized data preprocessed using prefiltering approach and traditional collaborative filtering algorithms**

As part of this technique, the data is first made contextualized and traditional collaborative filtering models as mentioned below are built.

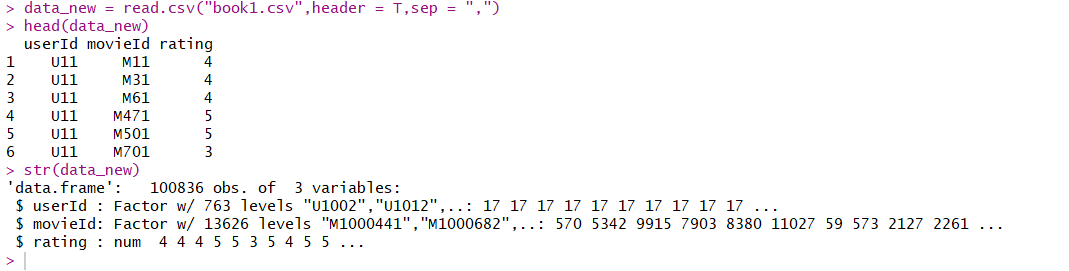
* + 1. IBCF
    2. UBCF
    3. Matrix Factorization

**Building Models**:

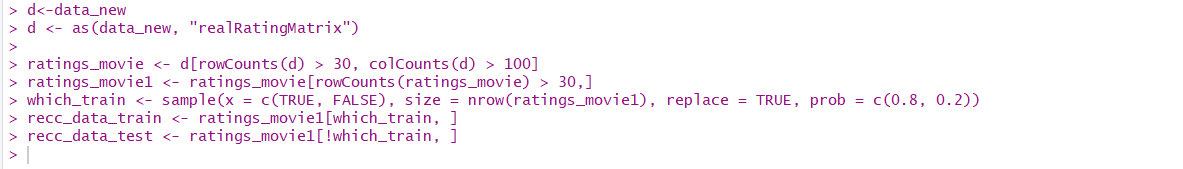
1. Using IBCF:
2. First, we contextualized data by using User-Item/UI splitting approach. We used to excel functions to contextualize the data. The dataset looks as below:



1. Loading dataset:

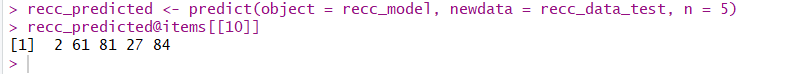


1. Splitting dataset into training and testing and matrix transformation:

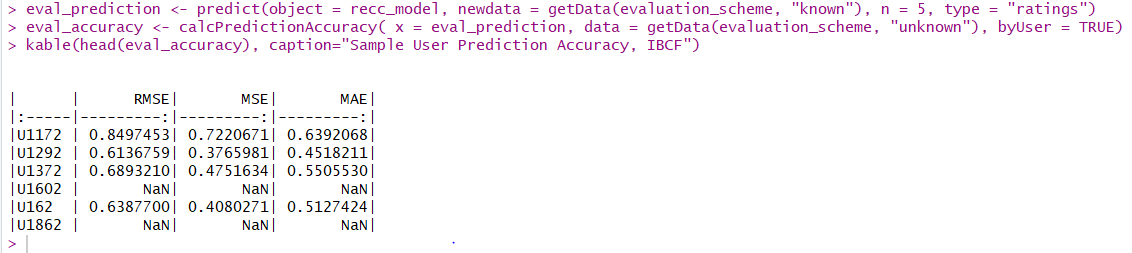


1. Building model and predicting on test dataset:





1. Finding accuracy using ‘calcPredictionAccuracy’ function

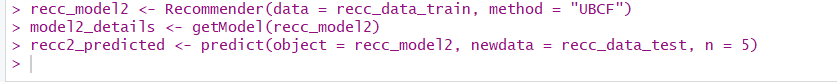


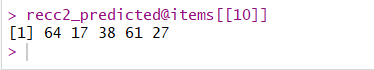
The RMSE of the Context-Aware UBCF is **0.6696**

1. Using UBCF:

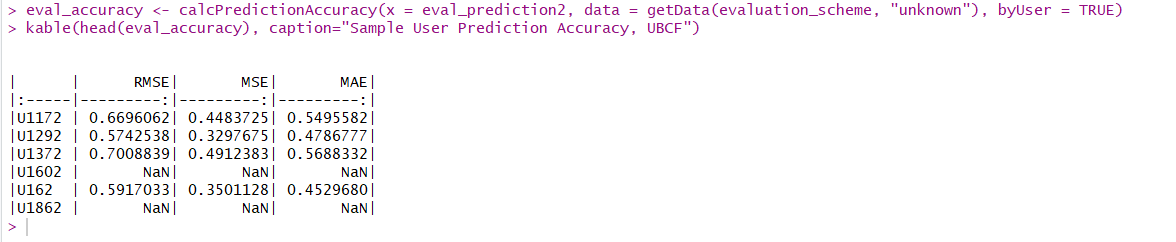
The dataset used for UBCF method is same as dataset used for IBCF also, models are built on same train and test dataset.

Building and predicting model using UBCF:



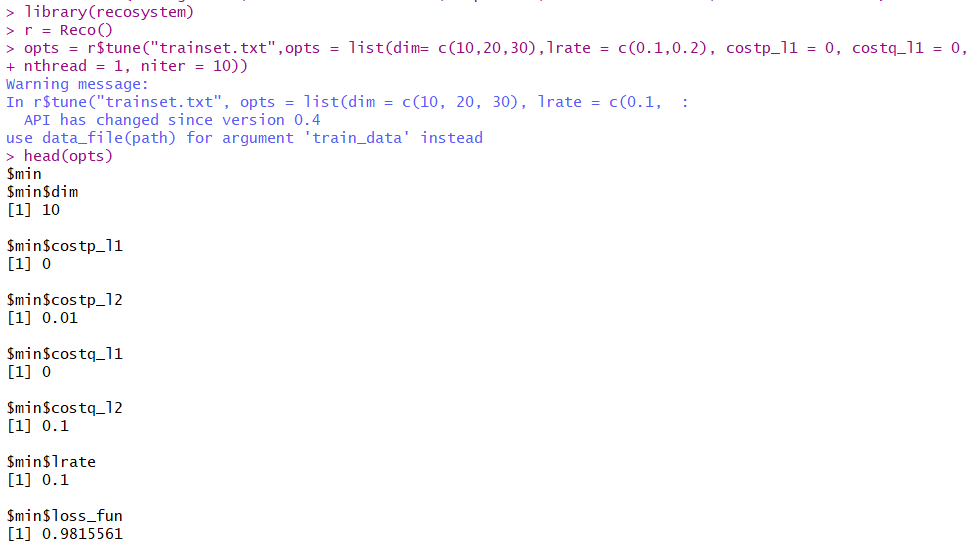


Finding accuracy using ‘calcPredictionAccuracy’ function:



1. **Context-Aware Matrix Factorization**:

Building modeland generating predictions using the Matrix Factorization method:



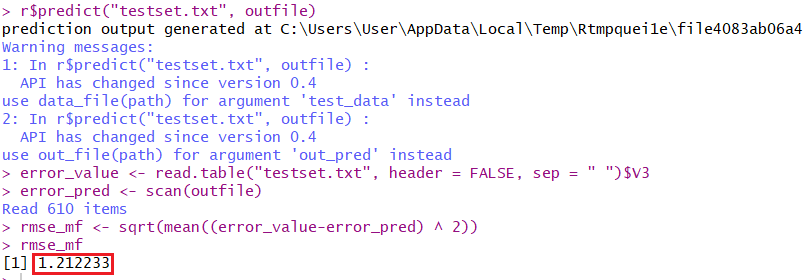
A close up of a piece of paper

Description automatically generated

A screenshot of a cell phone

Description automatically generated

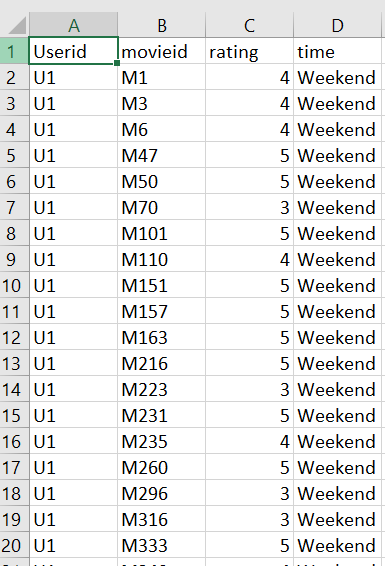
Finding out the RMSE value for this model:



1. **Context-Aware Recommender Models using the CARSkit library**

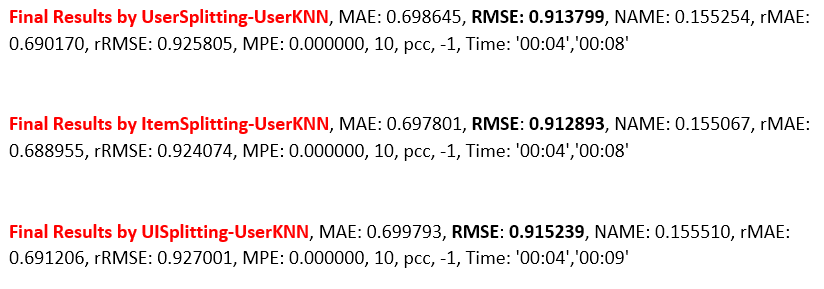
As part of this technique, the data is contextualized as shown below, and different CARS are built using the CASA approaches like UserSplitting, ItemSplitting and UISplitting, as well as an independent model using the Tensor Factorization is built, as we are having only one context, i.e. time. Models built can be seen below.

**Contextualized Data:**



* 1. **UserKNN:**

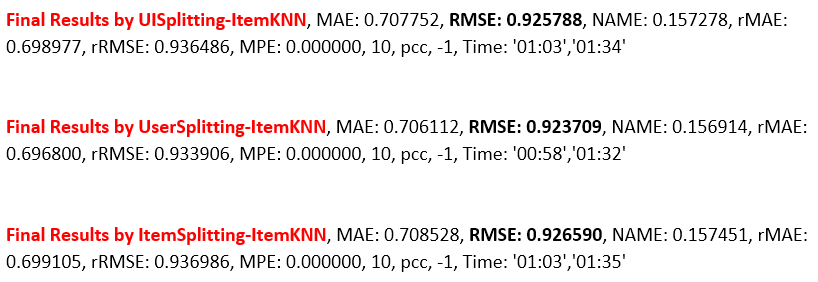
UserKNN algorithm is used here with all three splitting approaches.



From the different splitting approaches, we can see that UserKNN model built using Item Splitting approach is the best model with lowest RMSE **0.9128**.

* 1. **ItemKNN:**

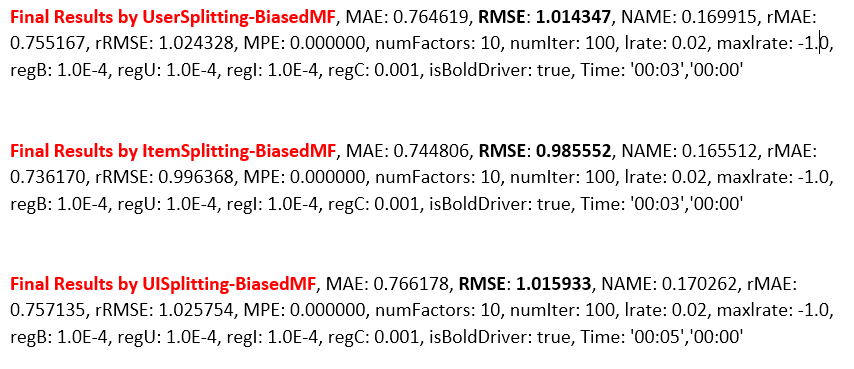
ItemKNN algorithm is used here with all three splitting approaches.



From the different splitting approaches, we can see that ItemKNN model built using User Splitting approach is the best model with lowest RMSE **0.9237**.

* 1. **BiasedMF:**

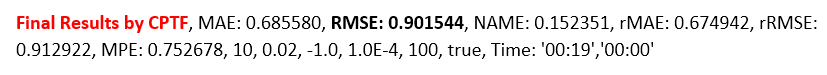
BiasedMF algorithm is used here with all three splitting approaches.



From the different splitting approaches, we can see that BiasedMF model built using Item Splitting approach is the best model with lowest RMSE **0.9855**.

* 1. Independent Model (CPTF):

Our dataset having only one context, i.e time and is independent of any other contexts, we have built an independent contextual modeling using the Tensor Factorization Optimization and Candecomp/Parafac (CP) decomposition technique. The result is as below.



The Contextual Modeling recommender system using Tensor Factorization and CP decomposition that resulted the least RMSE amongst the Context-Aware models, i.e **0.9015.**

# **5. Evaluations and Results**

## 5.1. Evaluation Methods

For evaluation purposes, we used RMSE values of different models to compare amongst them to find the best performing model.

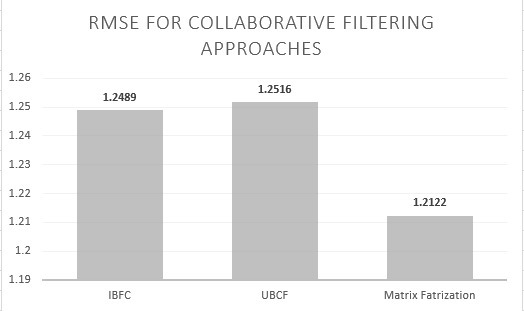
## 5.2. Results and Findings

We will now compare the models built as part of both traditional and new recommender systems and find the better performing model amongst them.



### Traditional Recommender Systems (Collaborative Filtering)

Below is the bar chart with different RMSE values for traditional recommender systems.



From the graph, we can see that Collaborative Filtering using Matrix Factorization has the lowest RMSE value of **1.2122**, when compared to the remaining. We can say that Collaborative Filtering using Matrix Factorization is the best recommender model under traditional recommender systems.

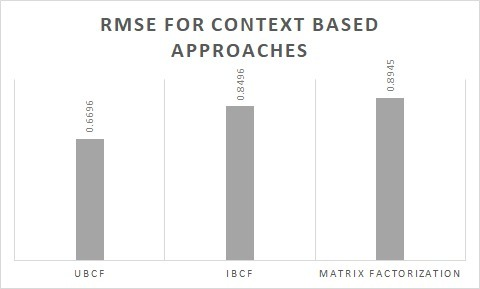
### New Recommnder Systems (Context-Aware)

As part of Context-Aware Recommender Systems, we have contextualized the data and built traditional recommender models on the contextualized data as part of the Prefiltering technique, followed by models built using the CARSkit library.

Below are the visualization that compares the models for both the techniques.

1. **Contextualized Prefiltering approach**

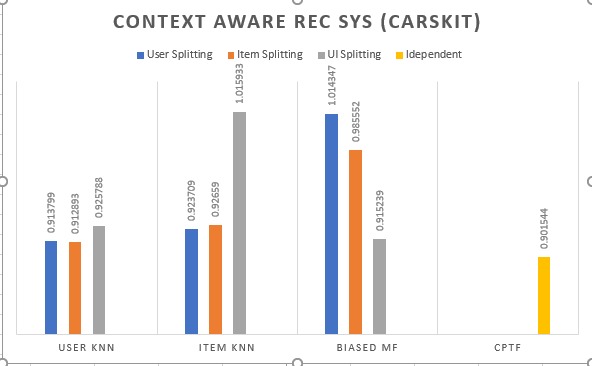
Below is the bar chart with different RMSE values for Context-Aware recommender systems built using the prefiltering technique.



From the graph, the contextualized recommendations from User base KNN CF model has the least RMSE value of **0.6696**. We can say that Context-Aware UBCF is the best recommender model.

1. **Using CARSkit library**

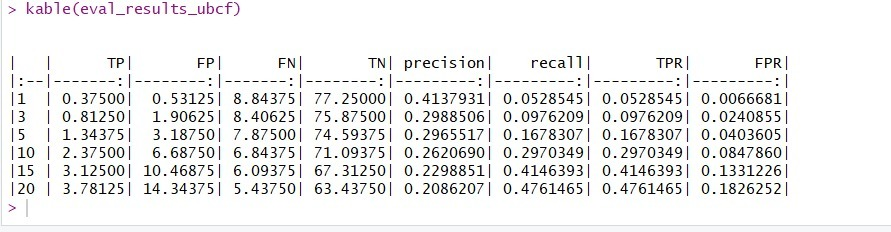
Below is the bar chart with different RMSE values for new recommender systems, i.e. Context-Aware models using the CASA approach like UserSplitting, ItemSplitting and UISplitting.

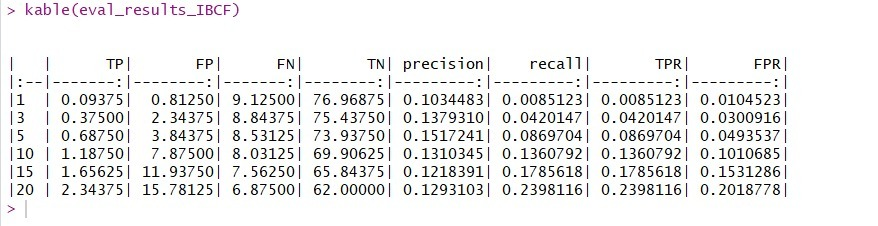


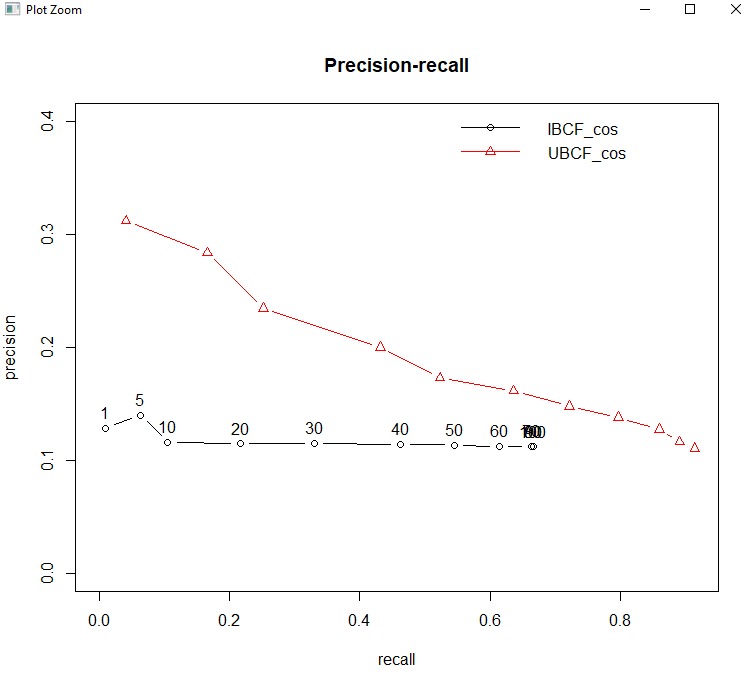
From the graph, different context-aware collaborative filtering models are built using different CASA approaches. But, independent context modeling (CPTF) outperformed the remaining values with the least RMSE of **0.9015** and stood as the best model.

1. **Comparing Precision and Recall:**

Precision and recall values for UBCF and IBCF are as follows:







From above graph we can say that UBCF model has grater recall value at 5. Therefore, UBCF model is best as compared to IBCF.

# **6. Conclusions and Future Work**

## 6.1. Conclusions

1. Built different traditional recommender systems using collaborative filtering approach and evaluated the CF model built using matrix factorization algorithm to be the best.
2. Built a content-based recommender system, that recommended user different set of movies based on the item-item similarity.
3. Built different context-aware recommender systems using contextual-prefiltering approach and evaluated the UBCF model on contextualized data to be the best

## 6.2. Limitations

1. Since we just had timestamp to consider as one of the contexts, we were only able to implement independent context-aware systems, although not that reliable.
2. We could have built more accurate and relevant context-aware systems if more than one context were available.

## 6.3. Potential Improvements or Future Work

1. The **LibRec** library can be used to work on different collaborative filtering approaches to build faster.
2. Dependent Contextual Modeling can be performed if the data set can be incorporated with different contexts like companion, place, etc.
3. Further exploration needed w.r.to Sparse-Linear Method that can build more accurate and generate recommendations faster by aggregation.