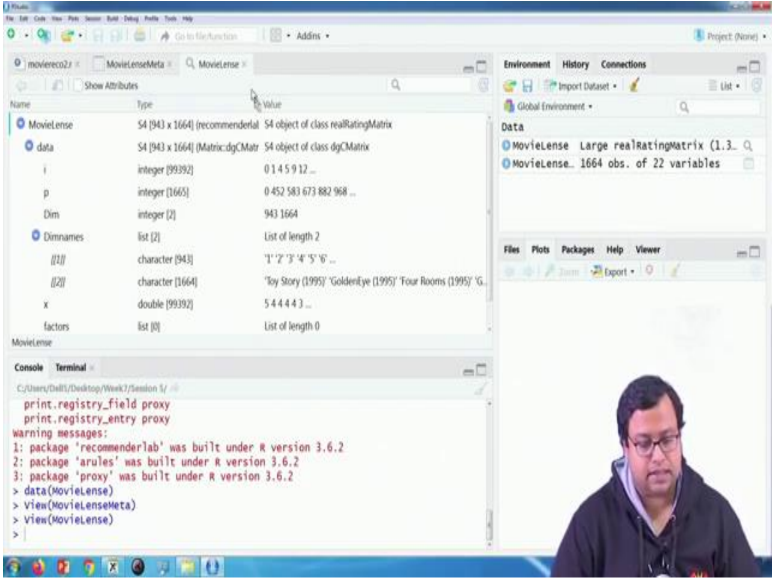
Marketing Analytics
Prof. Swagato Chatterjee
Vinod Gupta School of Management
Indian Institute of Technology, Kharagpur
Lecture 39: Recommendation Engine and Retail Analytics (Contd.)

Hello everybody, welcome to Marketing Analytics Course, this is Dr. Swagato Chatterjee from VGSOM from IIT, Kharagpur who is taking this course and this particular video is on Recommendation Engine and we are in week seven and this in this particular video we will actually discuss about one particular example of recommendation engine again with movie rating data.

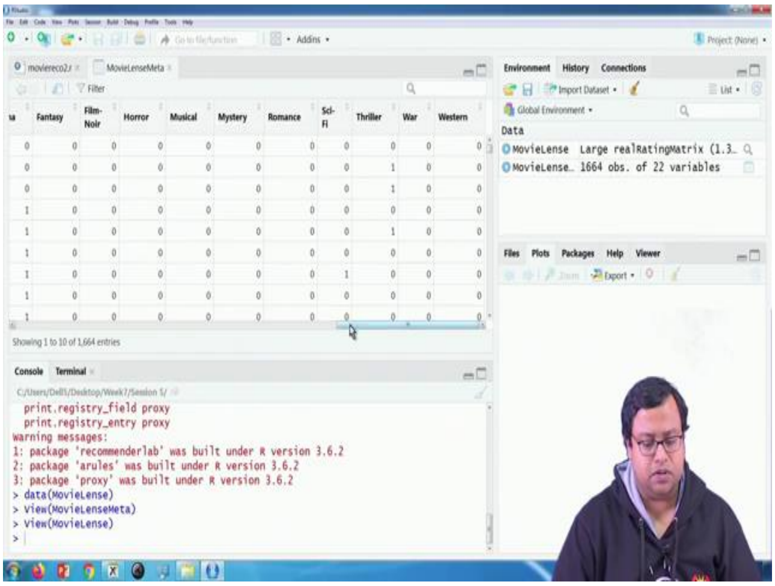
(Refer Slide Time: 0:38)



Name	Type	Value
MovieLense	S4 [943 x 1664] (recommenderlal	S4 object of class realRatingMatrix
data	S4 [943 x 1664] (Matrix::dgCMatr	S4 object of class dgCMatrix
i	integer [99392]	0 1 4 5 9 12 ...
p	integer [1665]	0 452 583 673 882 968 ...
Dim	integer [2]	943 1664
Dimnames	list [2]	List of length 2
[[1]]	character [943]	'1' '2' '3' '4' '5' '6' ...
[[2]]	character [1664]	'Toy Story (1995)' 'GoldenEye (1995)' 'Four Rooms (1995)' 'G...
x	double [99392]	5 4 4 4 4 3 ...
factors	list [0]	List of length 0

```
Data
MovieLense  Large realRatingMatrix (1.3…
MovieLense… 1664 obs. of 22 variables
```

```
C:/Users/Dell5/Desktop/Week7/Session 5/
  print.registry_field proxy
  print.registry_entry proxy
Warning messages:
1: package 'recommenderlab' was built under R version 3.6.2
2: package 'arules' was built under R version 3.6.2
3: package 'proxy' was built under R version 3.6.2
> data(MovieLense)
> View(MovieLenseMeta)
> View(MovieLense)
> 
```

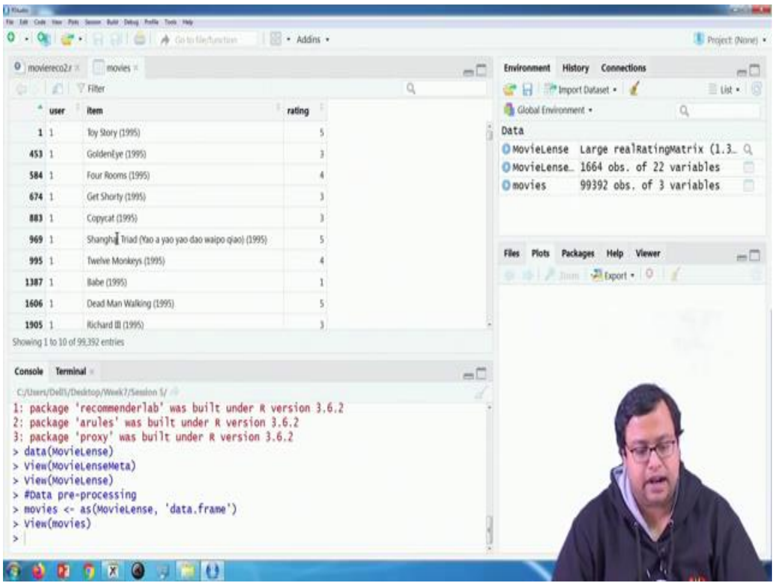


a	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0

Showing 1 to 10 of 1,664 entries

```
Data
MovieLense  Large realRatingMatrix (1.3…
MovieLense… 1664 obs. of 22 variables
```

```
C:/Users/Dell5/Desktop/Week7/Session 5/
  print.registry_field proxy
  print.registry_entry proxy
Warning messages:
1: package 'recommenderlab' was built under R version 3.6.2
2: package 'arules' was built under R version 3.6.2
3: package 'proxy' was built under R version 3.6.2
> data(MovieLense)
> View(MovieLenseMeta)
> View(MovieLense)
> 
```



	user	item	rating
1	1	Toy Story (1995)	5
453	1	GoldenEye (1995)	3
584	1	Four Rooms (1995)	4
674	1	Get Shorty (1995)	3
883	1	Copycat (1995)	3
969	1	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)	5
995	1	Twelve Monkeys (1995)	4
1387	1	Babe (1995)	1
1606	1	Dead Man Walking (1995)	5
1905	1	Richard III (1995)	3

Showing 1 to 10 of 99,392 entries

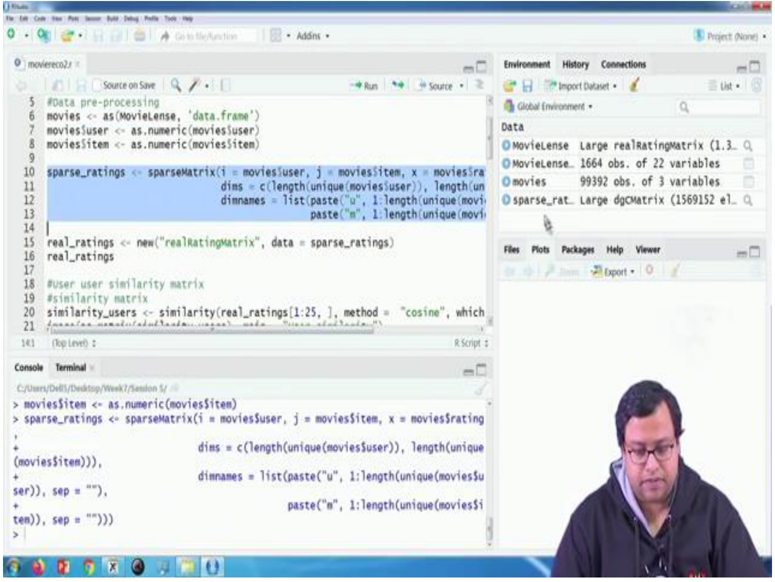
```
Data
MovieLense     Large realRatingMatrix (1.3…
MovieLense…    1664 obs. of 22 variables
movies         99392 obs. of 3 variables
```

```
C:/Users/Dell1/Desktop/Week7/Session 5/
1: package 'recommenderlab' was built under R version 3.6.2
2: package 'arules' was built under R version 3.6.2
3: package 'proxy' was built under R version 3.6.2
> data(MovieLense)
> View(MovieLenseMeta)
> View(MovieLense)
> #Data pre-processing
> movies <- as(MovieLense, 'data.frame')
> View(movies)
> 
```

So, like the previous one we will again set working data to source file location, put my Recommender lab library. The data in this case is inbuilt called MovieLense. So, data is equal to MovieLense and the MovieLense data has basically again the various movies, their years, and etcetera and the genres, remember this is something that we have created in the last video and MovieLense is the matrix which is like looks like this so, this is something that is there in MovieLense dataset already available with us.

Now, I have in the previous video have manually created all of these things so here our focus is not to manually create, is already there in the dataset. So, we will directly jump into the picture. So, what I am doing here is that the pre processing first so, MovieLense, I am changing it to a sparse datasets, sparse is a specific form of dataset. So, movie as that means it changes it to sparse dataset it looks like this, the user, the item and the rating that you are getting, the user ID that item ID and the rating ID. Now, I am changing user to numeric and item to numeric data so both are numeric, fair enough.

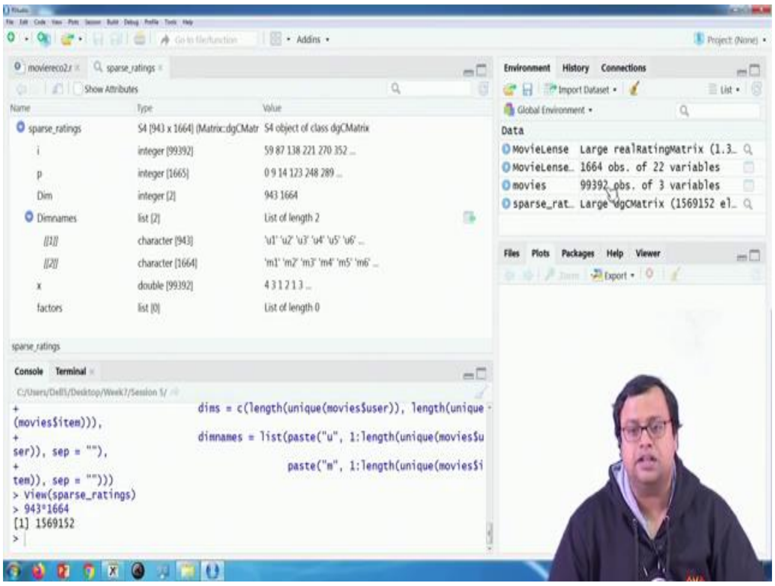
(Refer Slide Time: 2:02)



```
#Data pre-processing
movies <- as(MovieLense, 'data.frame')
movies$user <- as.numeric(movies$user)
movies$item <- as.numeric(movies$item)

sparse_ratings <- sparseMatrix(i = movies$user, j = movies$item, x = movies$ra
                               dims = c(length(unique(movies$user)), length(un
                               dimnames = list(paste("u", 1:length(unique(movi
                                               paste("m", 1:length(unique(movi

real_ratings <- new("realRatingMatrix", data = sparse_ratings)
real_ratings

#User user similarity matrix
#similarity matrix
similarity_users <- similarity(real_ratings[1:25, ], method = "cosine", which
```

```
> movies$item <- as.numeric(movies$item)
> sparse_ratings <- sparseMatrix(i = movies$user, j = movies$item, x = movies$rating
,
+                                dims = c(length(unique(movies$user)), length(unique
(movies$item))),
+                                dimnames = list(paste("u", 1:length(unique(movies$u
ser)), sep = ""),
+                                                paste("m", 1:length(unique(movies$i
tem)), sep = "")))
>
```

```
Data
MovieLense    Large realRatingMatrix (1.3…
MovieLense…   1664 obs. of 22 variables
movies        99392 obs. of 3 variables
sparse_rat…   Large dgCMatrix (1569152 el…
```



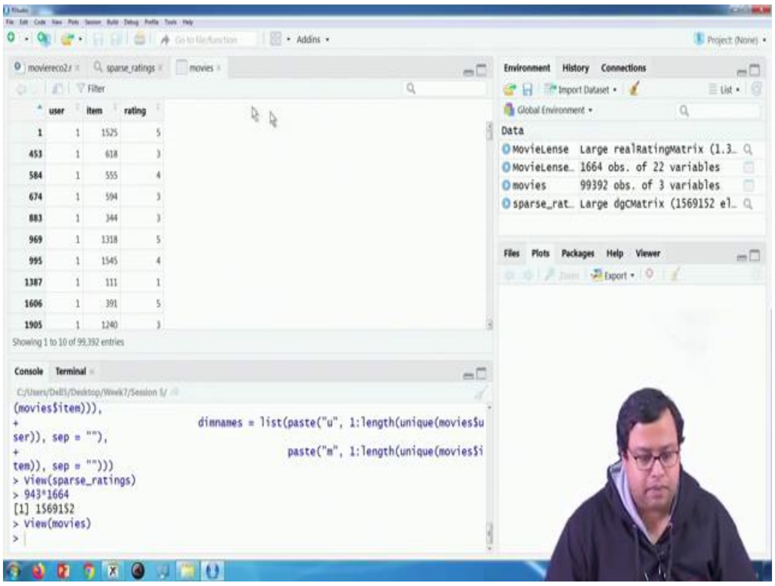
```
Name            Type                            Value
sparse_ratings  S4 [943 x 1664] (Matrix:dgCMatr S4 object of class dgCMatrix
  i             integer [99392]                 59 87 138 221 270 352 …
  p             integer [1665]                  0 9 14 123 248 289 …
  Dim           integer [2]                     943 1664
  Dimnames      list [2]                        List of length 2
    [[1]]       character [943]                 'u1' 'u2' 'u3' 'u4' 'u5' 'u6' …
    [[2]]       character [1664]                'm1' 'm2' 'm3' 'm4' 'm5' 'm6' …
  x             double [99392]                  4 3 1 2 1 3 …
  factors       list [0]                        List of length 0
```

```
+                                dims = c(length(unique(movies$user)), length(unique
(movies$item))),
+                                dimnames = list(paste("u", 1:length(unique(movies$u
ser)), sep = ""),
+                                                paste("m", 1:length(unique(movies$i
tem)), sep = "")))
> View(sparse_ratings)
> 943*1664
[1] 1569152
>
```

Now what is sparse ratings? Sparse ratings is a sparse matrix where i is = user ID, j = item ID and x = user rating. So, this is how I am creating a sparse matrix instead of sparse dataset and all the rest of the things remain same. So, now I have created a sparse matrix which is a big matrix which looks like this. So, carefully see, these are the dimension names you want to, user1 to user 943 and this is movie1 to movie1664, there are 1664 movies. So basically, if you see 943 into 1664, will create 156915 to this many elements.

Now these many elements, these many reviews are not there, actually if you check the large matrix, there are how many reviews? There are 99,392 reviews, unique reviews.

(Refer Slide Time: 3:01)

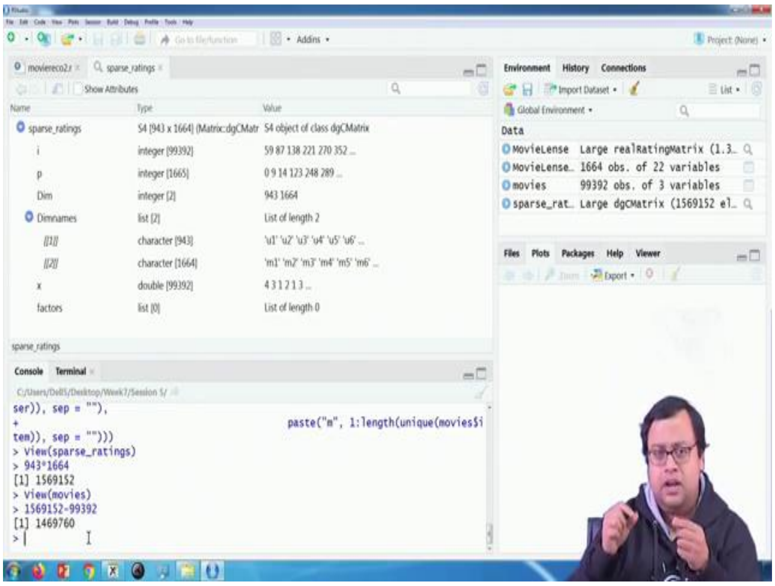


	user	item	rating
1	1	1525	5
453	1	618	3
584	1	555	4
674	1	594	3
883	1	344	3
969	1	1318	5
995	1	1545	4
1387	1	111	1
1606	1	391	5
1905	1	1240	3

Showing 1 to 10 of 99,392 entries

```
(movies$item))),
+                                    dimnames = list(paste("u", 1:length(unique(movies$u
ser)), sep = ""),
+                                                  paste("m", 1:length(unique(movies$i
tem)), sep = "")))
> View(sparse_ratings)
> 943*1664
[1] 1569152
> View(movies)
>
```

```
Data
MovieLense  Large realRatingMatrix (1.3…
MovieLense… 1664 obs. of 22 variables
movies      99392 obs. of 3 variables
sparse_rat… Large dgCMatrix (1569152 el…
```



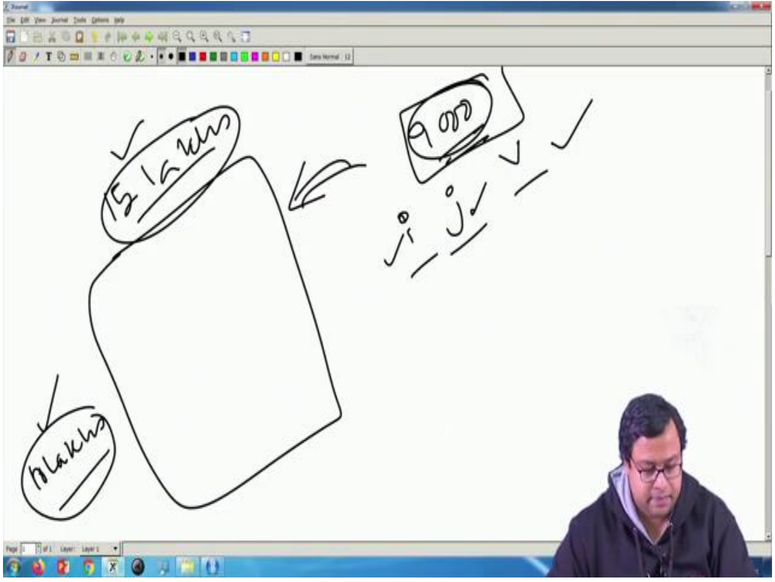
Name	Type	Value
sparse_ratings	S4 [943 x 1664] (Matrix:dgCMatr	S4 object of class dgCMatrix
i	integer [99392]	59 87 138 221 270 352 ...
p	integer [1665]	0 9 14 123 248 289 ...
Dim	integer [2]	943 1664
Dimnames	list [2]	List of length 2
[[1]]	character [943]	'u1' 'u2' 'u3' 'u4' 'u5' 'u6' ...
[[2]]	character [1664]	'm1' 'm2' 'm3' 'm4' 'm5' 'm6' ...
x	double [99392]	4 3 1 2 1 3 ...
factors	list [0]	List of length 0

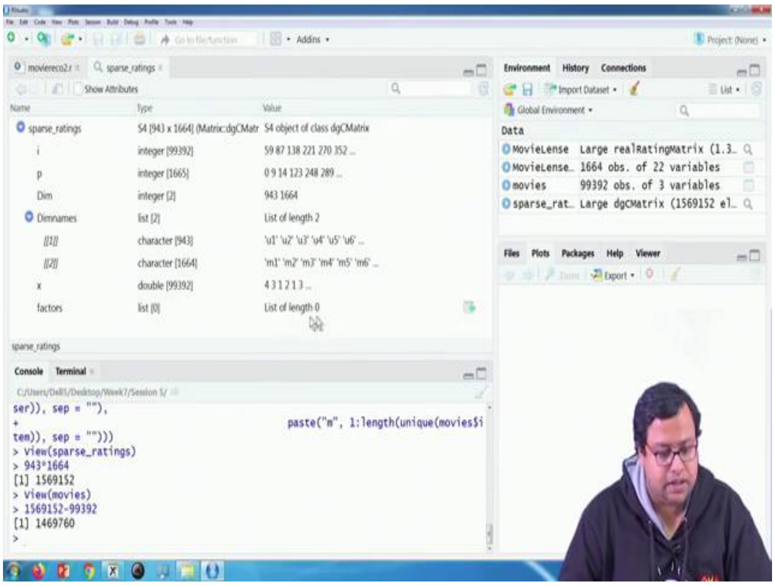
```
ser)), sep = ""),
+                                                  paste("m", 1:length(unique(movies$i
tem)), sep = "")))
> View(sparse_ratings)
> 943*1664
[1] 1569152
> View(movies)
> 1569152-99392
[1] 1469760
>
```

```
Data
MovieLense  Large realRatingMatrix (1.3…
MovieLense… 1664 obs. of 22 variables
movies      99392 obs. of 3 variables
sparse_rat… Large dgCMatrix (1569152 el…
```

If you check this particular dataset, each column is one unique review, each row is one unique review. So, there are 99,392 reviews but I have 1569152 this much is the value so 1569152 - 99,392 gives me so, most of the places of blank. Now, if I put zeros there and etcetera that will create a huge matrix, that much matrix, the large matrix I might not need because so, all I would rather know that okay, this is a matrix, I can create those the same thing.

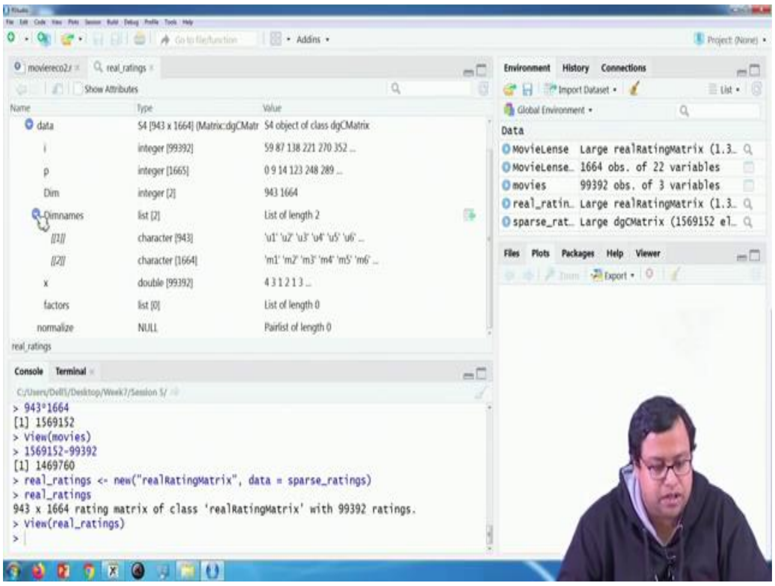
(Refer Slide Time: 3:40)




Name	Type	Value
sparse_ratings	S4 [943 x 1664] (Matrix:dgCMatr	S4 object of class dgCMatrix
i	integer [99392]	59 87 138 221 270 352 ...
p	integer [1665]	0 9 14 123 248 289 ...
Dim	integer [2]	943 1664
Dimnames	list [2]	List of length 2
[[1]]	character [943]	'u1' 'u2' 'u3' 'u4' 'u5' 'u6' ...
[[2]]	character [1664]	'm1' 'm2' 'm3' 'm4' 'm5' 'm6' ...
x	double [99392]	4 3 1 2 1 3 ...
factors	list [0]	List of length 0

```
ser)), sep = ""),
+                                              paste("m", 1:length(unique(movies$i
tem)), sep = "")))
> View(sparse_ratings)
> 943*1664
[1] 1569152
> View(movies)
> 1569152-99392
[1] 1469760
>
```

```
Data
MovieLense   Large realRatingMatrix (1.3…
MovieLense…  1664 obs. of 22 variables
movies       99392 obs. of 3 variables
sparse_rat…  Large dgCMatrix (1569152 el…
```


Name	Type	Value
data	S4 [943 x 1664] (Matrix:dgCMatr	S4 object of class dgCMatrix
i	integer [99392]	59 87 138 221 270 352 ...
p	integer [1665]	0 9 14 123 248 289 ...
Dim	integer [2]	943 1664
Dimnames	list [2]	List of length 2
[[1]]	character [943]	'u1' 'u2' 'u3' 'u4' 'u5' 'u6' ...
[[2]]	character [1664]	'm1' 'm2' 'm3' 'm4' 'm5' 'm6' ...
x	double [99392]	4 3 1 2 1 3 ...
factors	list [0]	List of length 0
normalize	NULL	Pairlist of length 0

```
> 943*1664
[1] 1569152
> View(movies)
> 1569152-99392
[1] 1469760
> real_ratings <- new("realRatingMatrix", data = sparse_ratings)
> real_ratings
943 x 1664 rating matrix of class 'realRatingMatrix' with 99392 ratings.
> View(real_ratings)
>
```

```
Data
MovieLense   Large realRatingMatrix (1.3…
MovieLense…  1664 obs. of 22 variables
movies       99392 obs. of 3 variables
real_ratin…  Large realRatingMatrix (1.3…
sparse_rat…  Large dgCMatrix (1569152 el…
```

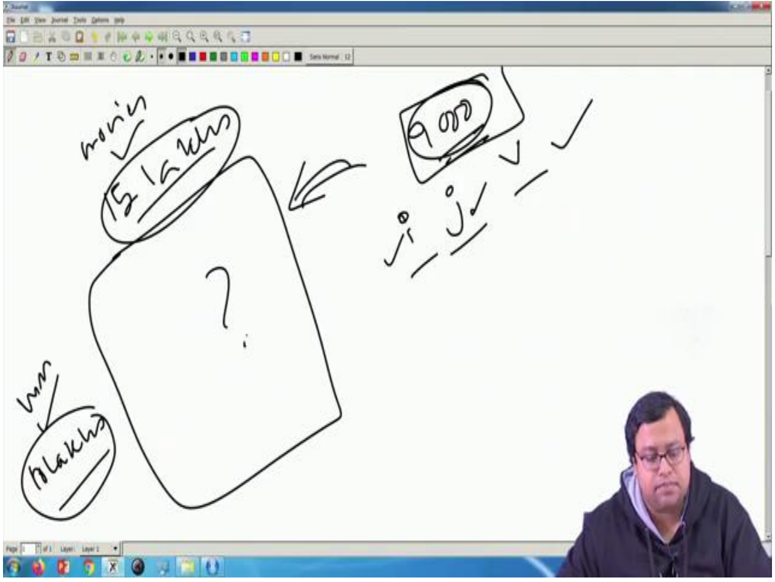
The same information I am create like this, that this is a matrix which has let us say 10 lakh rows here, I do not know and 15 lakh columns here, I do not know. And I will tell that in this 10 lakhs, 15 lakhs, there are only let us say 9,000 elements which are actually positive elements or valid elements, all other elements are zero. So, if I just tell you that there are 10 lakhs and 15 lakhs and there are 9,000 elements, for each element I am giving you i, j and value, that means the row number, column number and the value of that, then you can create this matrix. You can create, it will be very big, but you can create on your own.

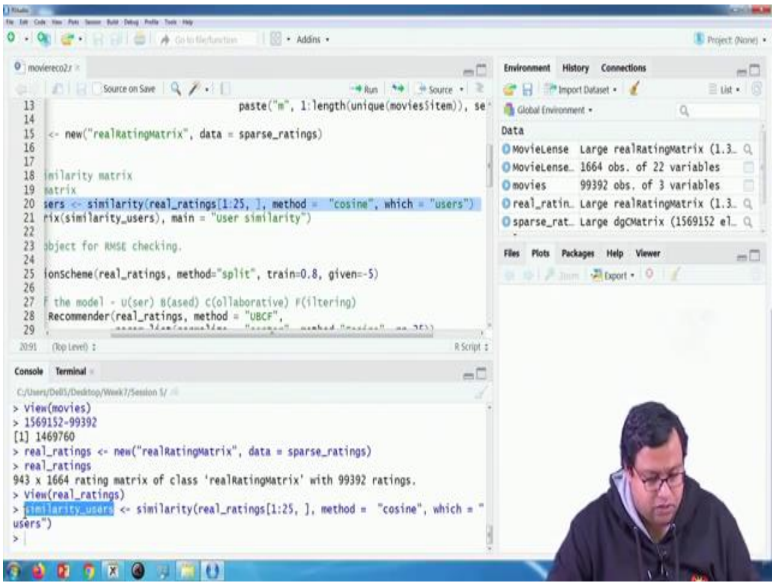
So, all the information that you need to create such a matrix is how many rows, how many columns, this number, and then how many elements and the element position and the element value. So, if you get that the matrix is created, you do not have to create the matrix. So, that is what is here. In sparse matrix, it is giving you the i, the p that means the integer, this is I think the position if I am not wrong, this is the cell number 1665, there are 1665 movies. So, the movies numbers I would say and this is a dimension, just one minute.

So, currently, it is absolutely zero, everything is zero and then I will populate it slowly so, real ratings is equal to new real rating matrix, data is equal to sparse ratings. So, if I just populate it, real ratings, this will give me the same thing whether ratings figures are given, see the rating figures are given here 4, 3, 1, 2 etcetera, etcetera, that is the rating figure that is given and correspondingly the u1, u2, u3 and the character names, the dimension names are given and the position will also be given.

So, these are the positions, this 993 is the position and this is basically the corresponding values at that position. So, that has been given to you in this sparse matrix. Now, based on this sparse matrix what I am getting?

(Refer Slide Time: 6:04)

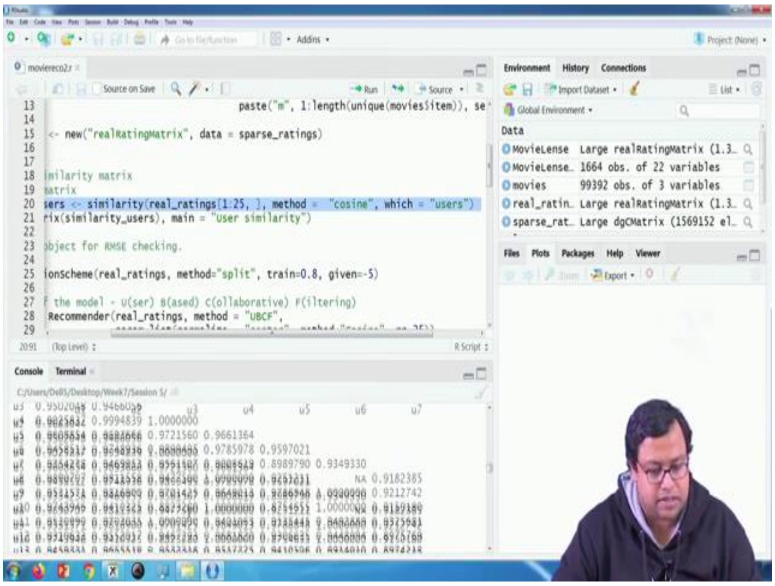




```
                                   paste("m", 1:length(unique(movies$item)), se
 <- new("realRatingMatrix", data = sparse_ratings)


imilarity matrix
atrix
sers <- similarity(real_ratings[1:25, ], method =  "cosine", which = "users")
rix(similarity_users), main = "User similarity")

bject for RMSE checking.

ionScheme(real_ratings, method="split", train=0.8, given=-5)

F the model - U(ser) B(ased) C(ollaborative) F(iltering)
 Recommender(real_ratings, method = "UBCF",
```

```
Data
MovieLense   Large realRatingMatrix (1.3…
MovieLense…  1664 obs. of 22 variables
movies       99392 obs. of 3 variables
real_ratin…  Large realRatingMatrix (1.3…
sparse_rat…  Large dgCMatrix (1569152 el…
```

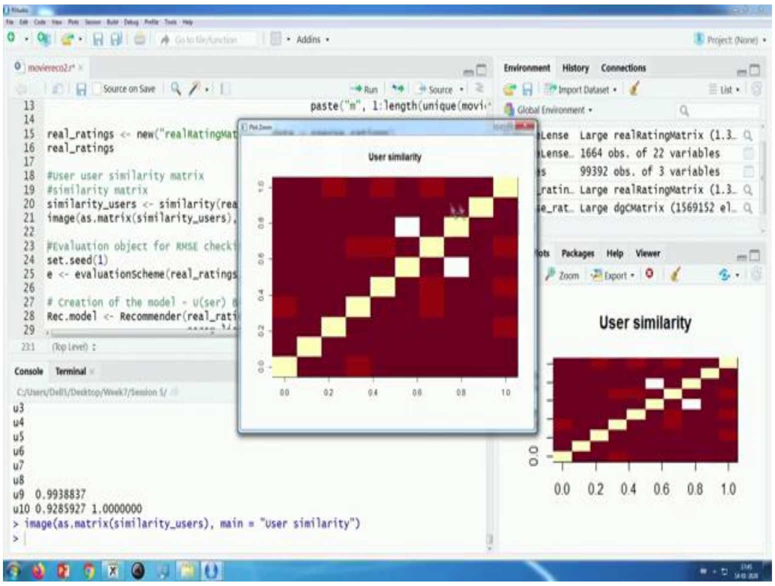
```
C:/Users/Dell5/Desktop/Week7/Session 5/
> View(movies)
> 1569152-99392
[1] 1469760
> real_ratings <- new("realRatingMatrix", data = sparse_ratings)
> real_ratings
943 x 1664 rating matrix of class 'realRatingMatrix' with 99392 ratings.
> View(real_ratings)
> similarity_users <- similarity(real_ratings[1:25, ], method =  "cosine", which = "
users")
>
```



```
                                   paste("m", 1:length(unique(movies$item)), se
 <- new("realRatingMatrix", data = sparse_ratings)


imilarity matrix
atrix
sers <- similarity(real_ratings[1:25, ], method =  "cosine", which = "users")
rix(similarity_users), main = "User similarity")

bject for RMSE checking.

ionScheme(real_ratings, method="split", train=0.8, given=-5)

F the model - U(ser) B(ased) C(ollaborative) F(iltering)
 Recommender(real_ratings, method = "UBCF",
```

```
Data
MovieLense   Large realRatingMatrix (1.3…
MovieLense…  1664 obs. of 22 variables
movies       99392 obs. of 3 variables
real_ratin…  Large realRatingMatrix (1.3…
sparse_rat…  Large dgCMatrix (1569152 el…
```

```
C:/Users/Dell5/Desktop/Week7/Session 5/
u3  0.9502048 0.9466055        u3        u4        u5        u6        u7
u4  [illegible] 0.9994839 1.0000000
u5  [illegible] [illegible] 0.9721560 0.9661364
u6  [illegible] [illegible] [illegible] 0.9785978 0.9597021
u7  [illegible] [illegible] [illegible] [illegible] 0.8989790 0.9349330
u8  [illegible] [illegible] [illegible] [illegible] [illegible]        NA 0.9182385
u9  [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] 0.9212742
u10 [illegible] [illegible] [illegible] 1.0000000 [illegible] 1.0000000 [illegible]
u11 [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
u12 [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
u13 [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
```



```
                                                paste("m", 1:length(unique(movi

real_ratings <- new("realRatingMat
real_ratings

#User user similarity matrix
#similarity matrix
similarity_users <- similarity(rea
image(as.matrix(similarity_users),

#Evaluation object for RMSE checki
set.seed(1)
e <- evaluationScheme(real_ratings

# Creation of the model - U(ser) B
Rec.model <- Recommender(real_rati
```

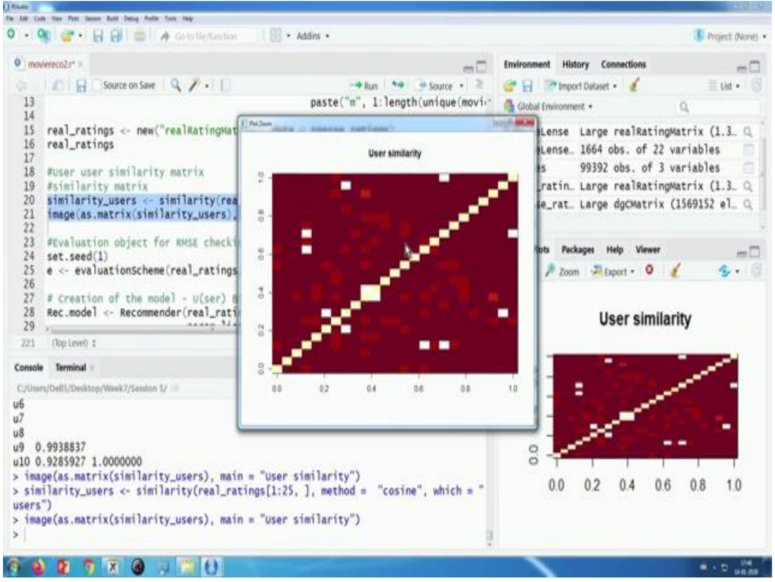
```
u3
u4
u5
u6
u7
u8
u9  0.9938837
u10 0.9285927 1.0000000
> image(as.matrix(similarity_users), main = "User similarity")
>
```

```
Lense      Large realRatingMatrix (1.3…
Lense…     1664 obs. of 22 variables
s          99392 obs. of 3 variables
_ratin…    Large realRatingMatrix (1.3…
e_rat…     Large dgCMatrix (1569152 el…
```

Basically I am getting something like this, where there are, these are my users and these are my movies and these are my ratings. So now from there, I can create a similarity matrix either user user or item item. So here I am creating a similarity ratings method is equal to cosine and which is equal to user so, I am taking each users at a time and creating a similarity matrix.

So, if I just run this, the similarity of users, first 25 I have taken, this is the similarity of the first 25 users corresponding matrix. So, this looks a little bit bad, let us do it for first 10 or first 10, and you will be able to understand this. This is the similarity matrix for first 10 guys, and if I just find out the image, that is the image, so, the more red it is, the more red it is the stronger is the similarity between two people.

(Refer Slide Time: 7:15)

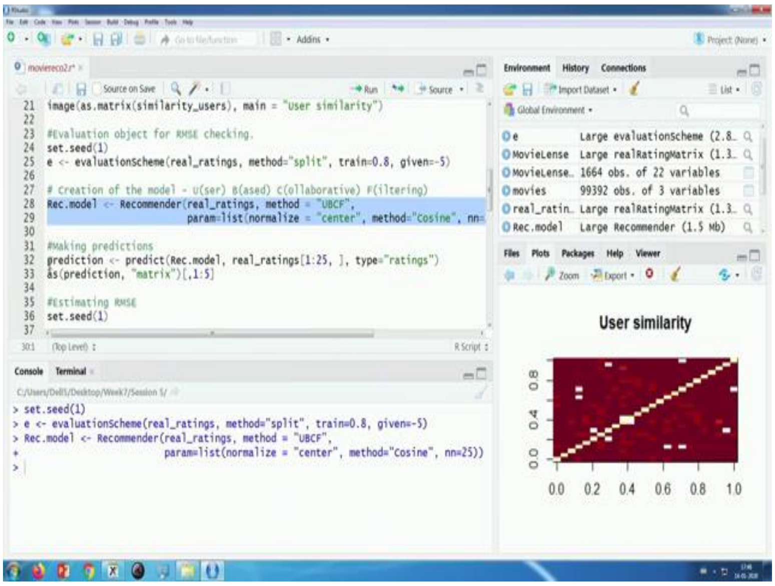


```
                                                        paste("m", 1:length(unique(movi

real_ratings <- new("realRatingMat
real_ratings

#User user similarity matrix
#similarity matrix
similarity_users <- similarity(rea
image(as.matrix(similarity_users),

#Evaluation object for RMSE checki
set.seed(1)
e <- evaluationScheme(real_ratings

# Creation of the model - U(ser) B
Rec.model <- Recommender(real_rati
```

```
Lense   Large realRatingMatrix (1.3…
Lense…  1664 obs. of 22 variables
s       99392 obs. of 3 variables
ratin…  Large realRatingMatrix (1.3…
e_rat…  Large dgCMatrix (1569152 el…
```

C:/Users/Dell5/Desktop/Week7/Session 5/

```
u6
u7
u8
u9  0.9938837
u10 0.9285927 1.0000000
> image(as.matrix(similarity_users), main = "User similarity")
> similarity_users <- similarity(real_ratings[1:25, ], method =  "cosine", which = "
users")
> image(as.matrix(similarity_users), main = "User similarity")
> 
```

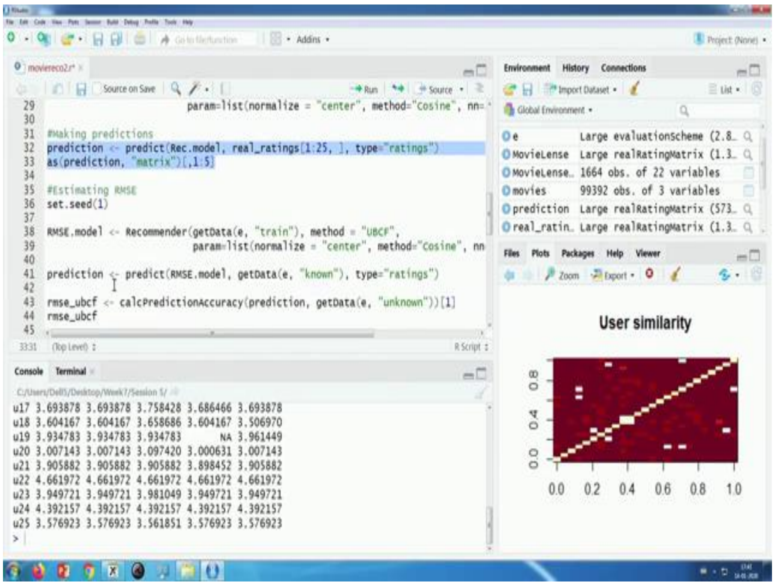


```
image(as.matrix(similarity_users), main = "User similarity")

#Evaluation object for RMSE checking.
set.seed(1)
e <- evaluationScheme(real_ratings, method="split", train=0.8, given=-5)

# Creation of the model - U(ser) B(ased) C(ollaborative) F(iltering)
Rec.model <- Recommender(real_ratings, method = "UBCF",
                         param=list(normalize = "center", method="Cosine", nn=

#Making predictions
prediction <- predict(Rec.model, real_ratings[1:25, ], type="ratings")
as(prediction, "matrix")[,1:5]

#Estimating RMSE
set.seed(1)
```

```
e            Large evaluationScheme (2.8…
MovieLense   Large realRatingMatrix (1.3…
MovieLense…  1664 obs. of 22 variables
movies       99392 obs. of 3 variables
real_ratin…  Large realRatingMatrix (1.3…
Rec.model    Large Recommender (1.5 Mb)
```

C:/Users/Dell5/Desktop/Week7/Session 5/

```
> set.seed(1)
> e <- evaluationScheme(real_ratings, method="split", train=0.8, given=-5)
> Rec.model <- Recommender(real_ratings, method = "UBCF",
+                          param=list(normalize = "center", method="Cosine", nn=25))
> 
```

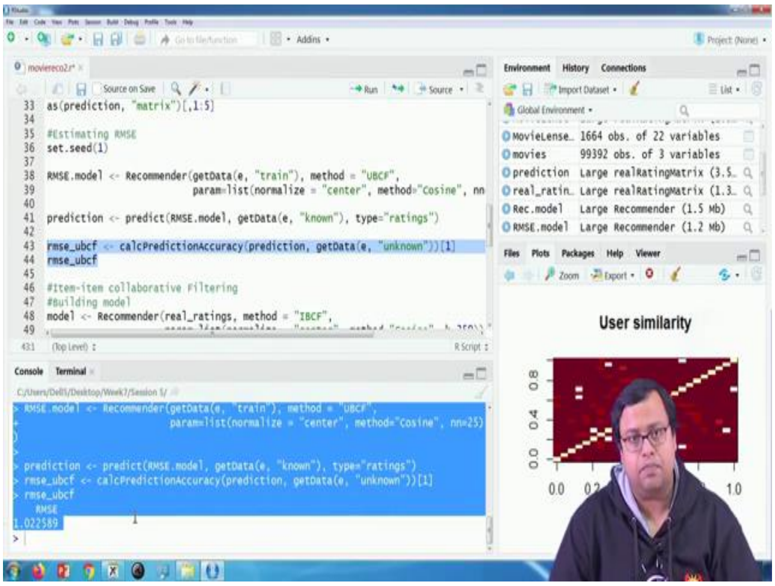


```
                         param=list(normalize = "center", method="Cosine", nn=

#Making predictions
prediction <- predict(Rec.model, real_ratings[1:25, ], type="ratings")
as(prediction, "matrix")[,1:5]

#Estimating RMSE
set.seed(1)

RMSE.model <- Recommender(getData(e, "train"), method = "UBCF",
                          param=list(normalize = "center", method="Cosine", nn

prediction <- predict(RMSE.model, getData(e, "known"), type="ratings")

rmse_ubcf <- calcPredictionAccuracy(prediction, getData(e, "unknown"))[1]
rmse_ubcf
```

```
e            Large evaluationScheme (2.8…
MovieLense   Large realRatingMatrix (1.3…
MovieLense…  1664 obs. of 22 variables
movies       99392 obs. of 3 variables
prediction   Large realRatingMatrix (573…
real_ratin…  Large realRatingMatrix (1.3…
```

C:/Users/Dell5/Desktop/Week7/Session 5/

```
u17 3.693878 3.693878 3.758428 3.686466 3.693878
u18 3.604167 3.604167 3.658686 3.604167 3.506970
u19 3.934783 3.934783 3.934783       NA 3.961449
u20 3.007143 3.007143 3.097420 3.000631 3.007143
u21 3.905882 3.905882 3.905882 3.898452 3.905882
u22 4.661972 4.661972 4.661972 4.661972 4.661972
u23 3.949721 3.949721 3.981049 3.949721 3.949721
u24 4.392157 4.392157 4.392157 4.392157 4.392157
u25 3.576923 3.576923 3.561851 3.576923 3.576923
> 
```



```
as(prediction, "matrix")[,1:5]

#Estimating RMSE
set.seed(1)

RMSE.model <- Recommender(getData(e, "train"), method = "UBCF",
                          param=list(normalize = "center", method="Cosine", nn

prediction <- predict(RMSE.model, getData(e, "known"), type="ratings")

rmse_ubcf <- calcPredictionAccuracy(prediction, getData(e, "unknown"))[1]
rmse_ubcf

#Item-item collaborative Filtering
#Building model
model <- Recommender(real_ratings, method = "IBCF",
```

```
C:/Users/Dell1/Desktop/Week7/Session 5/
> RMSE.model <- Recommender(getData(e, "train"), method = "UBCF",
+                           param=list(normalize = "center", method="Cosine", nn=25)
)
>
> prediction <- predict(RMSE.model, getData(e, "known"), type="ratings")
> rmse_ubcf <- calcPredictionAccuracy(prediction, getData(e, "unknown"))[1]
> rmse_ubcf
    RMSE
1.022589
>
```

```
MovieLense…  1664 obs. of 22 variables
movies       99392 obs. of 3 variables
prediction   Large realRatingMatrix (3.5…
real_ratin…  Large realRatingMatrix (1.3…
Rec.model    Large Recommender (1.5 Mb)
RMSE.model   Large Recommender (1.2 Mb)
```

So, 1 to 25 I have taken, initially I will come back to that 1 to 25 and then I will just run this once more. So, image of similarity comes up to be this, you see there are certain guys which are absolutely zero, the red ones are very similar and there are some small less red ones which are in between, so, something like that is there.

Now, I will find out with this similarity matrix how I can create a user based collaborative filtering. So, e is the evaluation where I am saying the evaluation parameter is the split method with training data 80 % rest data with 20 % and given =-5. So, I run that and then the recommender model is UBCF that means user based collaborative filtering and the parameter is normalized towards the center.

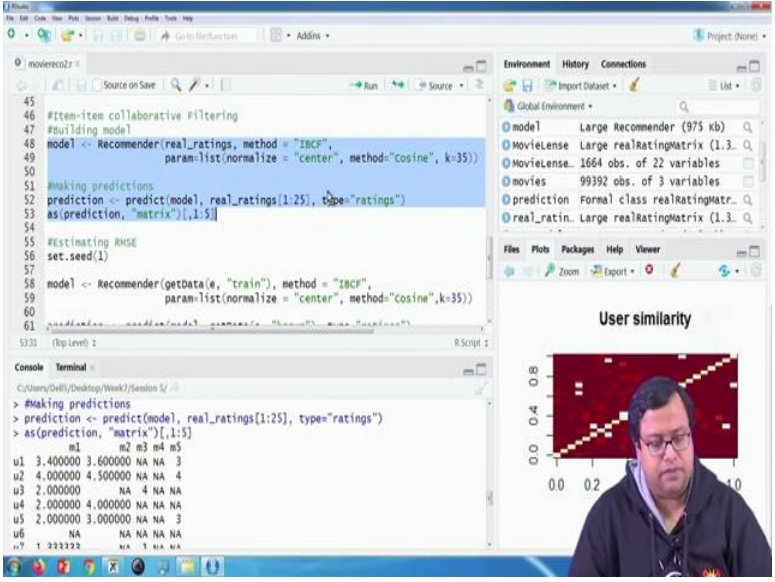
So, it is a center normalization, method is cosine and nearest neighbor is 25. So, take top first 25 similarity so, if I just run that, and based on that if I mark the predictions, I get these are my predictions for the first 5 m1, m2, m3, m4, m5 these are my corresponding user 1 to user 25 the predictions are given.

Now, based on that I will estimate the RMSE of this particular model so, the RMSE of this particular model is, see it is a training testing model. So, 80 % data has been taken so, rest of the 20 % we are checking that whatever recommendations I am giving and whatever this guy has actually seen are same or not. So, to do that I am creating the prediction here based on the same UBCF same model that I have created. And then I am calculating the RMSE which is the calculating the prediction accuracy, RMSE is coming 1.02. So, I will have used UBCF the same thing I will use by item based IBCF and then I will see that what is RMSE score.

So, whatever was there before UBCF, I have changed to IBCF, that is all and k = 350 means, nearest 350 movies you will be taken into account. So, instead of 25, because there are more number of movies and very less number of users so, user to user similarity 25 users is okay. But movie to movie similarity you have to take more movies.

So, that is what I am doing here and then predicting it so, it will take some time to do it 350 by 350, you might want to probably run it in the background because it might take some time, because I have taken 350 by 350 rather I will stop it probably.

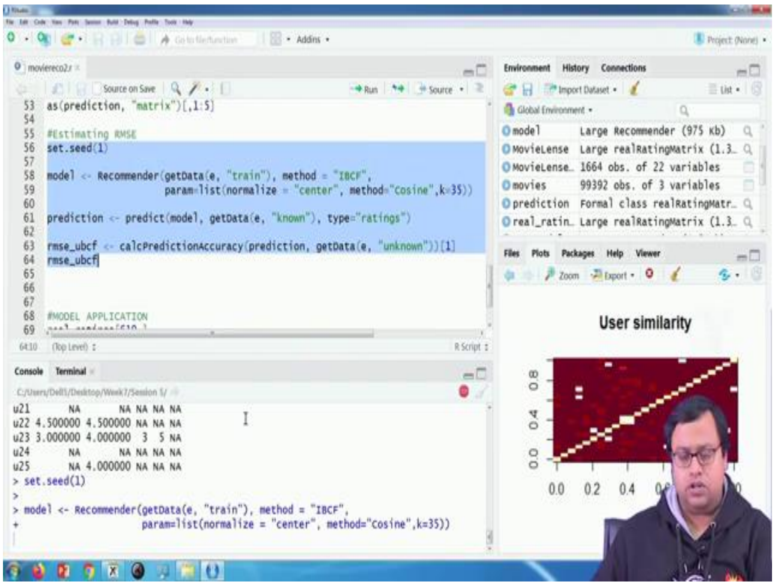
(Refer Slide Time: 10:17)



```
#Item-item collaborative Filtering
#Building model
model <- Recommender(real_ratings, method = "IBCF",
                     param=list(normalize = "center", method="Cosine", k=35))

#Making predictions
prediction <- predict(model, real_ratings[1:25], type="ratings")
as(prediction, "matrix")[,1:5]

#Estimating RMSE
set.seed(1)

model <- Recommender(getData(e, "train"), method = "IBCF",
                     param=list(normalize = "center", method="Cosine",k=35))
```

```
> #Making predictions
> prediction <- predict(model, real_ratings[1:25], type="ratings")
> as(prediction, "matrix")[,1:5]
         m1       m2 m3 m4 m5
u1 3.400000 3.600000 NA NA  3
u2 4.000000 4.500000 NA NA  4
u3 2.000000       NA  4 NA NA
u4 2.000000 4.000000 NA NA NA
u5 2.000000 3.000000 NA NA  3
u6       NA       NA NA NA NA
```

```
model        Large Recommender (975 Kb)
MovieLense   Large realRatingMatrix (1.3…
MovieLense…  1664 obs. of 22 variables
movies       99392 obs. of 3 variables
prediction   Formal class realRatingMatr…
real_ratin…  Large realRatingMatrix (1.3…
```

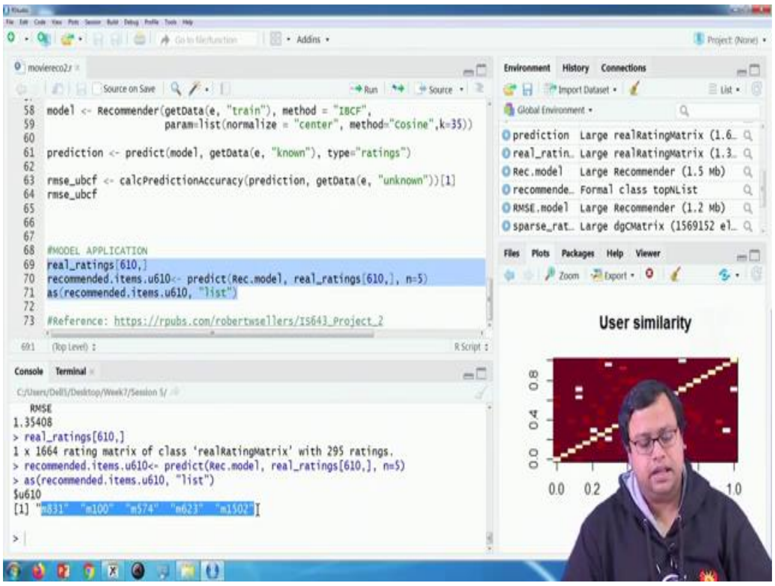


```
as(prediction, "matrix")[,1:5]

#Estimating RMSE
set.seed(1)

model <- Recommender(getData(e, "train"), method = "IBCF",
                     param=list(normalize = "center", method="Cosine",k=35))

prediction <- predict(model, getData(e, "known"), type="ratings")

rmse_ubcf <- calcPredictionAccuracy(prediction, getData(e, "unknown"))[1]
rmse_ubcf



#MODEL APPLICATION
```

```
model         Large Recommender (975 Kb)
MovieLense    Large realRatingMatrix (1.3...
MovieLense... 1664 obs. of 22 variables
movies        99392 obs. of 3 variables
prediction    Formal class realRatingMatr...
real_ratin... Large realRatingMatrix (1.3...
```

C:/Users/Dell1/Desktop/Week7/Session 5/

```
u21       NA       NA NA NA NA
u22 4.500000 4.500000 NA NA NA
u23 3.000000 4.000000  3  5 NA
u24       NA       NA NA NA NA
u25       NA 4.000000 NA NA NA
> set.seed(1)
>
> model <- Recommender(getData(e, "train"), method = "IBCF",
+                     param=list(normalize = "center", method="Cosine",k=35))
```

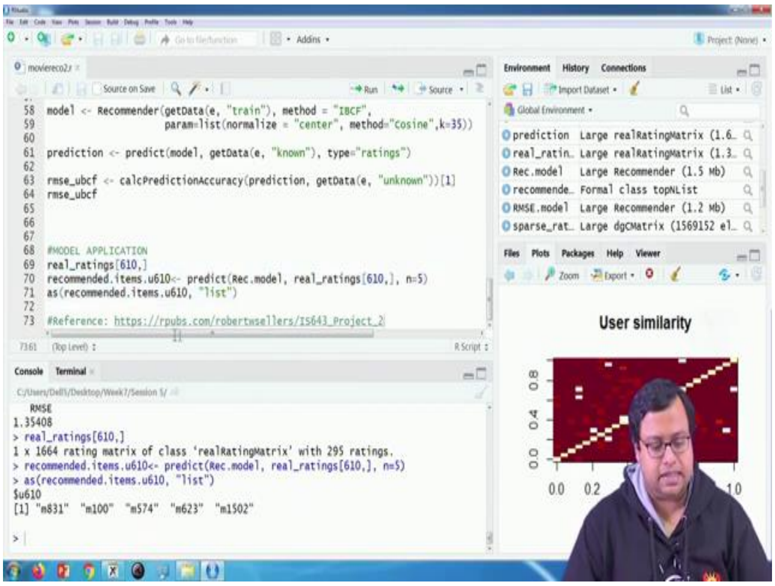


```
model <- Recommender(getData(e, "train"), method = "IBCF",
                     param=list(normalize = "center", method="Cosine",k=35))

prediction <- predict(model, getData(e, "known"), type="ratings")

rmse_ubcf <- calcPredictionAccuracy(prediction, getData(e, "unknown"))[1]
rmse_ubcf



#MODEL APPLICATION
real_ratings[610,]
recommended.items.u610<- predict(Rec.model, real_ratings[610,], n=5)
as(recommended.items.u610, "list")

#Reference: https://rpubs.com/robertwsellers/IS643_Project_2
```

```
prediction    Large realRatingMatrix (1.6...
real_ratin... Large realRatingMatrix (1.3...
Rec.model     Large Recommender (1.5 Mb)
recommende... Formal class topNList
RMSE.model    Large Recommender (1.2 Mb)
sparse_rat... Large dgCMatrix (1569152 el...
```

C:/Users/Dell1/Desktop/Week7/Session 5/

```
   RMSE
1.35408
> real_ratings[610,]
1 x 1664 rating matrix of class 'realRatingMatrix' with 295 ratings.
> recommended.items.u610<- predict(Rec.model, real_ratings[610,], n=5)
> as(recommended.items.u610, "list")
$u610
[1] "m831"  "m100"  "m574"  "m623"  "m1502"

>
```



```
model <- Recommender(getData(e, "train"), method = "IBCF",
                     param=list(normalize = "center", method="Cosine",k=35))

prediction <- predict(model, getData(e, "known"), type="ratings")

rmse_ubcf <- calcPredictionAccuracy(prediction, getData(e, "unknown"))[1]
rmse_ubcf



#MODEL APPLICATION
real_ratings[610,]
recommended.items.u610<- predict(Rec.model, real_ratings[610,], n=5)
as(recommended.items.u610, "list")

#Reference: https://rpubs.com/robertwsellers/IS643_Project_2
```

```
prediction    Large realRatingMatrix (1.6...
real_ratin... Large realRatingMatrix (1.3...
Rec.model     Large Recommender (1.5 Mb)
recommende... Formal class topNList
RMSE.model    Large Recommender (1.2 Mb)
sparse_rat... Large dgCMatrix (1569152 el...
```

C:/Users/Dell1/Desktop/Week7/Session 5/

```
   RMSE
1.35408
> real_ratings[610,]
1 x 1664 rating matrix of class 'realRatingMatrix' with 295 ratings.
> recommended.items.u610<- predict(Rec.model, real_ratings[610,], n=5)
> as(recommended.items.u610, "list")
$u610
[1] "m831"  "m100"  "m574"  "m623"  "m1502"

>
```

Let us do it for 35 then we can actually see the result so, 35 by 35 so, it has to be actually 350. It has to be big, but I am doing it because the time otherwise will be very high. So, 35 let us check the result. Still taking quite a bit of time. So, let it, I will give it one minute if it does not stop in one minute then we will see, we will probably reduce it. So, the real ratings, then what we will do is we will actually change it to probably a little bit so I have got 35 for m1, m2, m3, m4, m5. And that has been, that is something that has been plotted for the 25 observations and then if I do it for set seed1, the same thing k = 35 when I try to make the prediction, I will get a prediction something like this.

So it might take one minute again, because now it will be creating the model for the testing data and with the testing data, it will create the prediction and then match the prediction with the actual value. And it will give me RMSE score. So, it might take another one minute probably to give the RMSE score, if you remember the previous RMSE score was 1.02. So, I have to check the whether this RMSE code is similar to that or not.

So, you can do a…so, here it is 1.35 which is higher so, user to user is something that matches better than item to item so, this is a trial and error procedure. And the moral obligation is what? So, let us see for 6, item number 610 I will be creating that whether somebody should see it or not. So, 164 ratings of class and this class has been used for user 610 the best movies are m831, m100, m574, m623 and m1502.

So, again, the reference is this particular link you can go and read about from this link the details that the code has been taken from there. But the idea is that you can create a user if even if you have a huge rating matrix, you can change it to your sparse matrix and then use the recommender lab, recommender lab I am suggesting because recommender lab is fast, much faster than you running your loop. If you do not want to use recommender lab you can run your own loops also, it might take a little bit more time if you do column operations, you might be able to do a bit more faster.

But that is something that you can do. So, we have discussed quite a lot about recommendation engine for the last six videos. Please carefully see them, the code is sometimes are difficult, please listen to it carefully and try to do it on your hand, try to do it in a new dataset. Best practice is create a new dataset from some other sources from let us say a source that is available in Kaggle or you get your own dataset from somewhere or create a small matrix on your own, and then try to replicate whatever work, whatever data that we have done because these codes were readymade for that particular data.

That is why is coming very simple and smooth but if the data is bad then the code has to be changed and you have to actually fight a little bit to find out those codes. So I will strongly suggest try to do that and in the next week, we will meet with newer topics. Thank you very much for listening to me and following me up and I will see you in the next week. Thank you.