1. . Consider the MovieLense data that is available in the recommenderlab package >data(MovieLense) >?MovieLense. The data was collected through the MovieLens web site during a seven-month, and contains about 100,000 ratings (1-5) from 943 users on 1664 movies. See the help file on the data to understand how to best manipulate the object. Design and evaluate a user-based recommender system. Create the system so that outputs a user's top ten recommendations. Demo it on 3

```
In [ ]: library(recommenderlab)
```

```
Loading required package: Matrix

Loading required package: arules


Attaching package: ‘arules’


The following objects are masked from ‘package:base’:

    abbreviate, write


Loading required package: proxy


Attaching package: ‘proxy’


The following object is masked from ‘package:Matrix’:

    as.matrix


The following objects are masked from ‘package:stats’:

    as.dist, dist


The following object is masked from ‘package:base’:

    as.matrix


Registered S3 methods overwritten by 'registry':
  method               from 
  print.registry_field proxy
  print.registry_entry proxy
```

```
In [ ]: data(MovieLense)
```

```
In [ ]: ?MovieLense
```

```
In [ ]: d <- MovieLense
```

```
In [ ]: head(d)
```

```
6 x 1664 rating matrix of class ‘realRatingMatrix’ with 789 ratings.
```

```
In [ ]: dim(d)
```

```
943  1664
```

```
In [ ]:  head(as(d[10,], "list")[[1]])
```

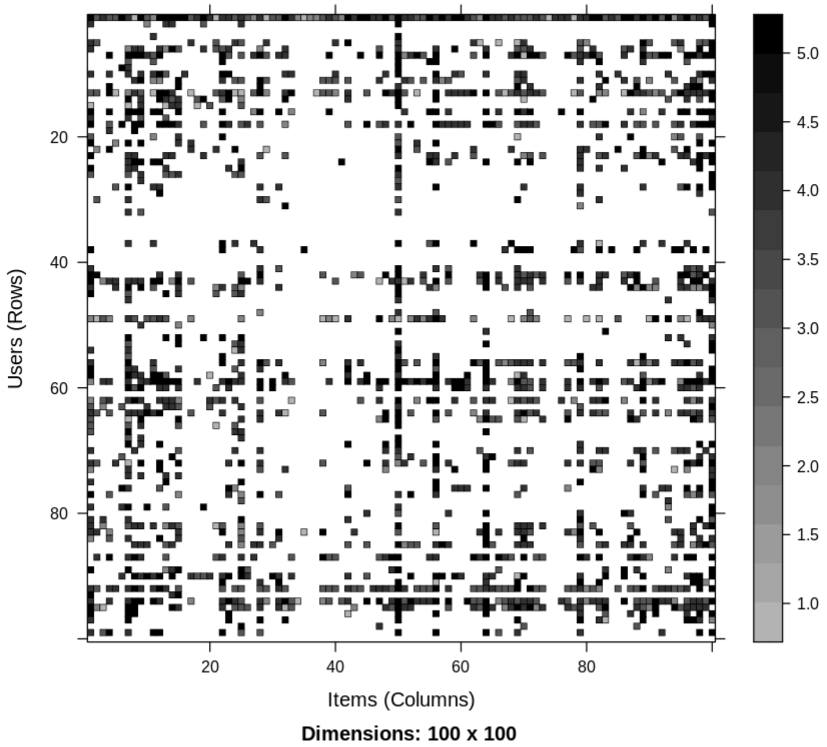
Toy Story (1995)
4
Get Shorty (1995)
4
Twelve Monkeys (1995)
4
Dead Man Walking (1995)
4
Seven (Se7en) (1995)
4
Usual Suspects, The (1995)
5

Exploratory Data Analysis

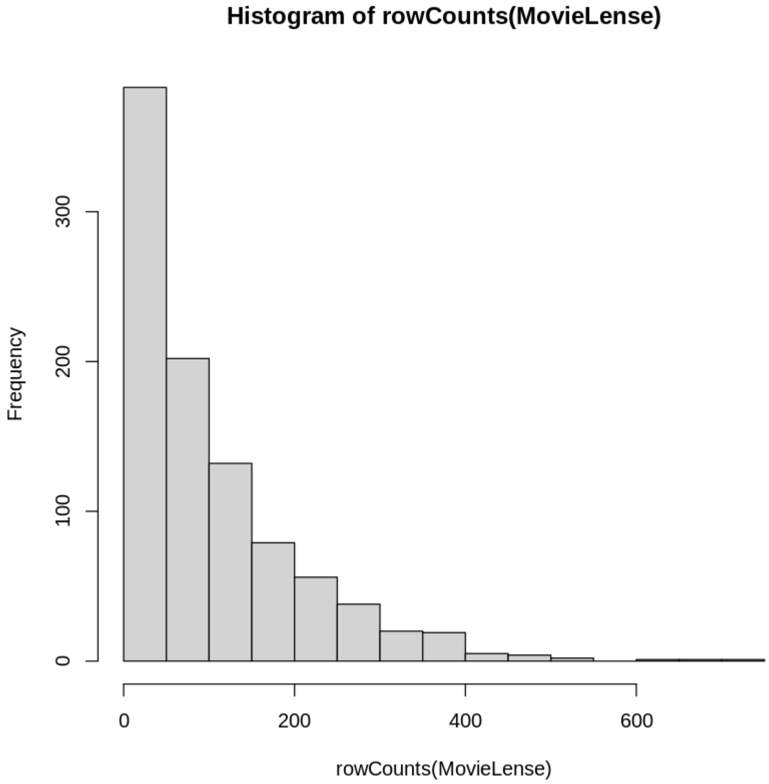
```
In [ ]: rownames(d)
```

```
'1' '2' '3' '4' '5' '6' '7' '8' '9' '10' '11' '12' '13' '14' '15' '16' '17' '18' '19' '20' '21'
'22' '23' '24' '25' '26' '27' '28' '29' '30' '31' '32' '33' '34' '35' '36' '37' '38' '39' '40'
'41' '42' '43' '44' '45' '46' '47' '48' '49' '50' '51' '52' '53' '54' '55' '56' '57' '58' '59'
'60' '61' '62' '63' '64' '65' '66' '67' '68' '69' '70' '71' '72' '73' '74' '75' '76' '77' '78'
'79' '80' '81' '82' '83' '84' '85' '86' '87' '88' '89' '90' '91' '92' '93' '94' '95' '96' '97'
'98' '99' '100' '101' '102' '103' '104' '105' '106' '107' '108' '109' '110' '111' '112' '113'
'114' '115' '116' '117' '118' '119' '120' '121' '122' '123' '124' '125' '126' '127' '128'
'129' '130' '131' '132' '133' '134' '135' '136' '137' '138' '139' '140' '141' '142' '143'
'144' '145' '146' '147' '148' '149' '150' '151' '152' '153' '154' '155' '156' '157' '158'
'159' '160' '161' '162' '163' '164' '165' '166' '167' '168' '169' '170' '171' '172' '173'
'174' '175' '176' '177' '178' '179' '180' '181' '182' '183' '184' '185' '186' '187' '188'
'189' '190' '191' '192' '193' '194' '195' '196' '197' '198' '199' '200' ⋯ '744' '745'
'746' '747' '748' '749' '750' '751' '752' '753' '754' '755' '756' '757' '758' '759' '760'
'761' '762' '763' '764' '765' '766' '767' '768' '769' '770' '771' '772' '773' '774' '775'
'776' '777' '778' '779' '780' '781' '782' '783' '784' '785' '786' '787' '788' '789' '790'
'791' '792' '793' '794' '795' '796' '797' '798' '799' '800' '801' '802' '803' '804' '805'
'806' '807' '808' '809' '810' '811' '812' '813' '814' '815' '816' '817' '818' '819' '820'
'821' '822' '823' '824' '825' '826' '827' '828' '829' '830' '831' '832' '833' '834' '835'
'836' '837' '838' '839' '840' '841' '842' '843' '844' '845' '846' '847' '848' '849' '850'
'851' '852' '853' '854' '855' '856' '857' '858' '859' '860' '861' '862' '863' '864' '865'
'866' '867' '868' '869' '870' '871' '872' '873' '874' '875' '876' '877' '878' '879' '880'
'881' '882' '883' '884' '885' '886' '887' '888' '889' '890' '891' '892' '893' '894' '895'
'896' '897' '898' '899' '900' '901' '902' '903' '904' '905' '906' '907' '908' '909' '910'
'911' '912' '913' '914' '915' '916' '917' '918' '919' '920' '921' '922' '923' '924' '925'
'926' '927' '928' '929' '930' '931' '932' '933' '934' '935' '936' '937' '938' '939' '940'
'941' '942' '943'
```

In [ ]:
```
image(d[1:100,1:100])
```

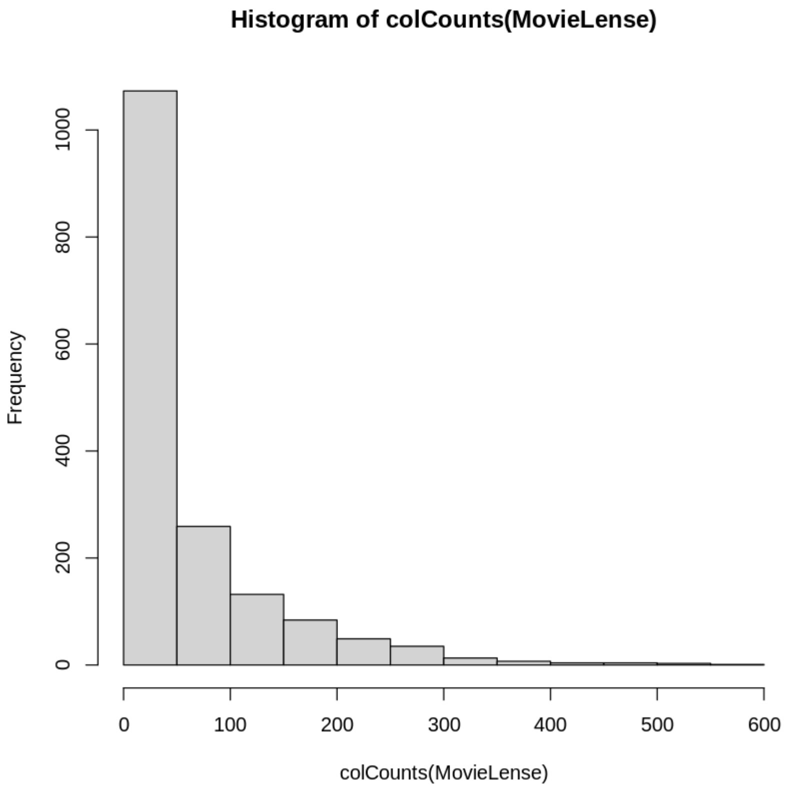


In [ ]:
```
hist(rowCounts(d))
```



In [ ]:

```
hist(colCounts(d))
```



In [ ]:

```
mean(rowMeans(d))
```

```
3.58756455155972
```

In [ ]:
```
head(MovieLenseMeta)
```

A data.frame: 6 × 22

	title	year	url	unknown	Action	Adventure	Animation
	<chr>	<dbl>	<chr>	<int>	<int>	<int>	<int>
1	Toy Story (1995)	1995	http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)	0	0	0	1
2	GoldenEye (1995)	1995	http://us.imdb.com/M/title-exact?GoldenEye%20(1995)	0	1	1	0
3	Four Rooms (1995)	1995	http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)	0	0	0	0
4	Get Shorty (1995)	1995	http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)	0	1	0	0
5	Copycat (1995)	1995	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0
6	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)	1995	http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995)	0	0	0	0

In [ ]:
```
head(MovieLenseUser)
```

A data.frame: 6 × 5

	id	age	sex	occupation	zipcode
	<int>	<int>	<fct>	<fct>	<fct>
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101

Rating Matrix

In [ ]:

```
dim(getRatingMatrix(d))
getRatingMatrix(d)[1:10, 1:10]
```

```
943  1664

  [[ suppressing 10 column names ‘Toy Story (1995)’, ‘GoldenEye (1995)’, ‘Fou
r Rooms (1995)’ ... ]]


10 x 10 sparse Matrix of class "dgCMatrix"

1  5 3 4 3 3 5 4 1 5 3
2  4 . . . . . . . . 2
3  . . . . . . . . . .
4  . . . . . . . . . .
5  4 3 . . . . . . . .
6  4 . . . . . 2 4 4 .
7  . . . 5 . . 5 5 5 4
8  . . . . . . 3 . . .
9  . . . . . 5 4 . . .
10 4 . . 4 . . 4 . 4 .
```

In [ ]:

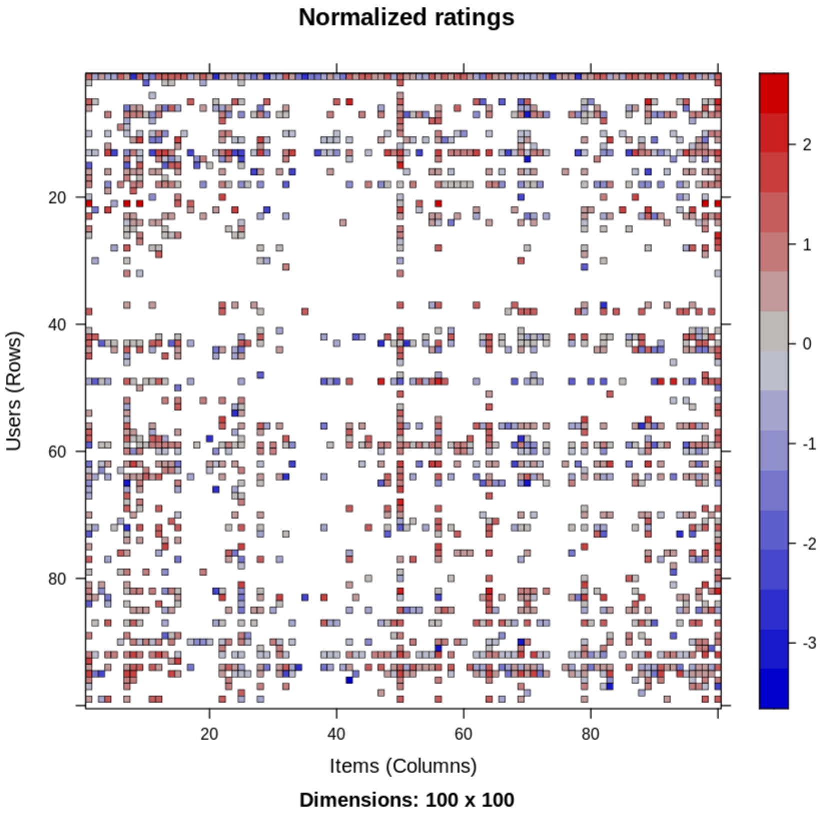
```
### Normalizing ###
```

In [ ]:

```
d_Normalize <- normalize(d)
d_Normalize
```

```
943 x 1664 rating matrix of class ‘realRatingMatrix’ with 99392 ratings.
Normalized using center on rows.
```

```
In [ ]: image(d_Normalize[1:100,1:100], main = "Normalized ratings")
```



In [ ]:
```
getRatingMatrix(d_Normalize)[1:10, 1:10]
```

```
  [[ suppressing 10 column names ‘Toy Story (1995)’, ‘GoldenEye (1995)’, ‘Fou
r Rooms (1995)’ ... ]]


10 x 10 sparse Matrix of class "dgCMatrix"

1   1.3948339 -0.6051661 0.3948339 -0.6051661 -0.6051661 1.3948339  0.3948339
2   0.2950820  .         .          .          .         .          .
3   .          .         .          .          .         .          .
4   .          .         .          .          .         .          .
5   1.1257143  0.1257143 .          .          .         .          .
6   0.3605769  .         .          .          .         .        -1.6394231
7   .          .         .         1.0350000  .         .          1.0350000
8   .          .         .          .          .         .        -0.7966102
9   .          .         .          .          .         0.7272727 -0.2727273
10 -0.2065217  .         .         -0.2065217  .         .        -0.2065217

1  -2.6051661  1.3948339 -0.6051661
2   .          .         -1.7049180
3   .          .          .
4   .          .          .
5   .          .          .
6   0.3605769  0.3605769  .
7   1.0350000  1.0350000  0.0350000
8   .          .          .
9   .          .          .
10  .         -0.2065217  .
```

In [ ]:
```
### DENORMALIZE ###
```

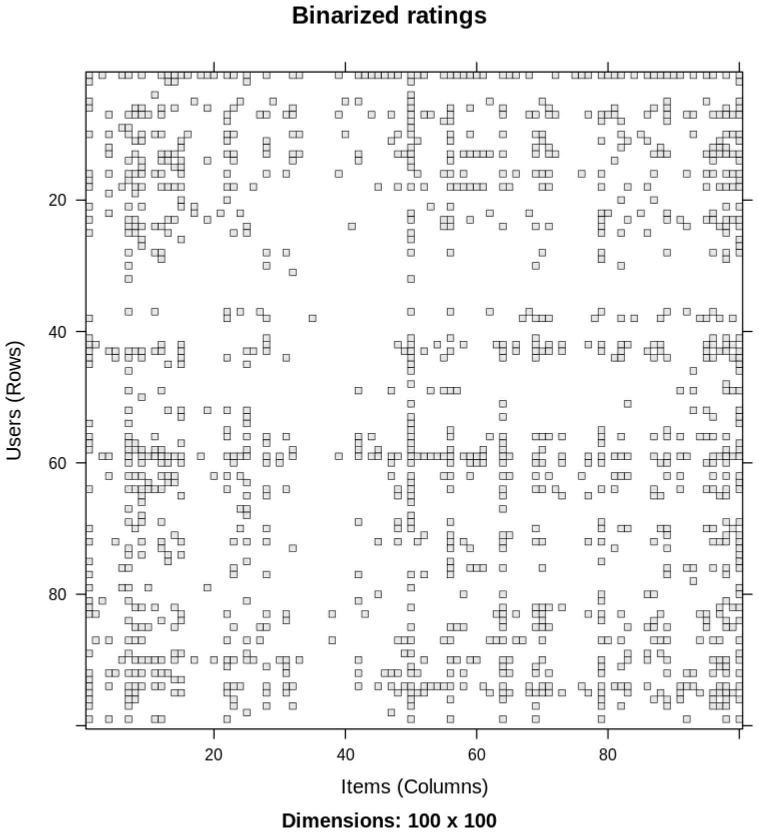
In [ ]:
```
d_denormalize <- denormalize(d_Normalize)
```

In [ ]:
```
### BINARY MATRIX ###
```

In [ ]:
```
d_binarize <- binarize(d_denormalize, minRating = 4)
getRatingMatrix(d_binarize)
```

```
itemMatrix in sparse format with
 943 rows (elements/transactions) and
 1664 columns (items)
```

In [ ]:
```
image(d_binarize[1:100,1:100], main = "Binarized ratings")
```



In [ ]:
```
### CREATING RECOMMENDER SYSTEM ###
```

In [ ]:
```


recommender_popularity <- Recommender(d[943:1], method = "POPULAR")
getModel(recommender_popularity)$topN
```

```
Recommendations as ‘topNList’ with n = 1664 for 1 users.
```

In [ ]:

```
# Create top 10 recommendations for 3 users
recom <- predict(recommender_popularity, d[50:52], n=10)
recom
as(recom, "list")
```

```
Recommendations as ‘topNList’ with n = 10 for 3 users.
```

$`50`
'Star Wars (1977)' 'Godfather, The (1972)' 'Raiders of the Lost Ark (1981)'
'Silence of the Lambs, The (1991)' 'Titanic (1997)' 'Schindler\'s List (1993)'
'Shawshank Redemption, The (1994)' 'Empire Strikes Back, The (1980)'
'Return of the Jedi (1983)' 'Usual Suspects, The (1995)'

$`51`
'Godfather, The (1972)' 'Fargo (1996)' 'Raiders of the Lost Ark (1981)'
'Silence of the Lambs, The (1991)' 'Titanic (1997)' 'Schindler\'s List (1993)'
'Usual Suspects, The (1995)' 'L.A. Confidential (1997)' 'Casablanca (1942)'
'Pulp Fiction (1994)'

$`52`
'Star Wars (1977)' 'Godfather, The (1972)' 'Raiders of the Lost Ark (1981)'
'Silence of the Lambs, The (1991)' 'Titanic (1997)' 'Shawshank Redemption, The (1994)'
'Empire Strikes Back, The (1980)' 'Return of the Jedi (1983)' 'Usual Suspects, The (1995)'
'Casablanca (1942)'

In [ ]:

```
# extract sublists
Recom3 <- bestN(recom, n = 3)
Recom3
as(Recom3, "list")
```

```
Recommendations as ‘topNList’ with n = 3 for 3 users.
```

$`50`
'Star Wars (1977)' 'Godfather, The (1972)' 'Raiders of the Lost Ark (1981)'

$`51`
'Godfather, The (1972)' 'Fargo (1996)' 'Raiders of the Lost Ark (1981)'

$`52`
'Star Wars (1977)' 'Godfather, The (1972)' 'Raiders of the Lost Ark (1981)'

In [ ]:
```
# Predict the ratings for three users
user_ratings <- predict(recommender_popularity, d[50:52], type = "ratings")
user_ratings
as(user_ratings, "matrix")[,1:10]
```

```
3 x 1664 rating matrix of class ‘realRatingMatrix’ with 4890 ratings.
```

A matrix: 3 × 10 of type dbl

	Toy Story (1995)	GoldenEye (1995)	Four Rooms (1995)	Get Shorty (1995)	Copycat (1995)	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)	Twelve Monkeys (1995)	Babe (1995)	Dead Man Walking (1995)
50	3.821541	3.268644	3.116264	3.492302	3.316419	3.624702	3.764166	3.891878	NA
51	3.865019	3.312122	3.159742	3.535780	3.359897	3.668180	3.807644	3.935356	3.882002
52	4.567659	4.014762	3.862382	4.238420	4.062537	4.370820	NA	4.637996	4.584642

In [ ]:
```
predict_ratings <- predict(recommender_popularity, d[50:52], type = "ratingMat
predict_ratings
as(predict_ratings, "matrix")[,1:10]
```

```
3 x 1664 rating matrix of class ‘realRatingMatrix’ with 4992 ratings.
```

A matrix: 3 × 10 of type dbl

	Toy Story (1995)	GoldenEye (1995)	Four Rooms (1995)	Get Shorty (1995)	Copycat (1995)	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)	Twelve Monkeys (1995)	Babe (1995)	Dead Man Walking (1995)
50	3.821541	3.268644	3.116264	3.492302	3.316419	3.624702	3.764166	3.891878	3.838524
51	3.865019	3.312122	3.159742	3.535780	3.359897	3.668180	3.807644	3.935356	3.882002
52	4.567659	4.014762	3.862382	4.238420	4.062537	4.370820	4.510284	4.637996	4.584642

This is the predicted ratings for 3 users

In [ ]:
```
### EVALUATION ###
```

In [ ]:

```
dim(d)
new <- sample(d)
dim(new)
```

```
943  1664

943  1664
```

In [ ]:

```
eval<- evaluationScheme(new,method = "split", given = 15, train=0.5, goodRatin
eval
```

```
as(<dgCMatrix>, "dgTMatrix") is deprecated since Matrix 1.5-0; do as(., "Tspa
rseMatrix") instead


Evaluation scheme with 15 items given
Method: ‘split’ with 1 run(s).
Training set proportion: 0.500
Good ratings: >=4.000000
Data set: 943 x 1664 rating matrix of class ‘realRatingMatrix’ with 99392 rat
ings.
```

In [ ]:

```
### COLLABRATIVE FILTERING ###
```

In [ ]:

```
umodel<- Recommender(getData(eval,"train"), "UBCF")
umodel
```

```
Recommender of type ‘UBCF’ for ‘realRatingMatrix’
learned using 471 users.
```

In [ ]:

```
### PREDICT RATIGNS USING UBCF MODEL ###
```

In [ ]:

```
P<- predict(umodel, getData(eval, "known"), type="ratings")
```

In [ ]:

```
error <- rbind(UBCF = calcPredictionAccuracy(P, getData(eval,"unknown")))
error
```

A matrix: 1 × 3 of type dbl

	RMSE	MSE	MAE
UBCF	1.21445	1.474889	0.9511854

In [ ]:

```
# Evaluating top-N recommender
```

In [ ]:
```
s<- evaluationScheme(new, method="cross",k=4, given=3, goodRating=4) ##?
s
```

```
Evaluation scheme with 3 items given
Method: ‘cross-validation’ with 4 run(s).
Good ratings: >=4.000000
Data set: 943 x 1664 rating matrix of class ‘realRatingMatrix’ with 99392 rat
ings.
```

In [ ]:
```
results<- evaluate(s, method = "POPULAR", type="topNList", n=c(1,3,5,10,15,20)
results
getConfusionMatrix(results)[[1]]
```

```
POPULAR run fold/sample [model time/prediction time]
	 1  [0.011sec/0.86sec] 
	 2  [0.007sec/0.993sec] 
	 3  [0.008sec/0.716sec] 
	 4  [0.007sec/0.67sec] 

Evaluation results for 4 folds/samples using method ‘POPULAR’.
```

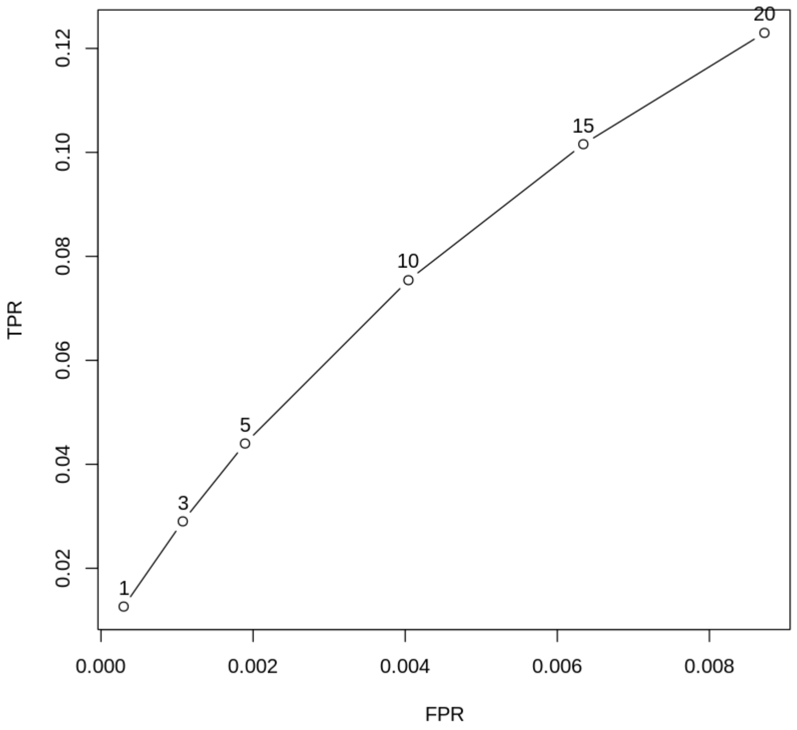
A matrix: 6 × 10 of type dbl

TP	FP	FN	TN	N	precision	recall	TPR	FPI
0.5084034	0.4915966	57.13866	1602.861	1661	0.5084034	0.01155535	0.01155535	0.000301802
1.1890756	1.8109244	56.45798	1601.542	1661	0.3963585	0.02562286	0.02562286	0.001115434
1.9579832	3.0420168	55.68908	1600.311	1661	0.3915966	0.04143687	0.04143687	0.001873306
3.3529412	6.6470588	54.29412	1596.706	1661	0.3352941	0.06746386	0.06746386	0.004100271
4.7983193	10.2016807	52.84874	1593.151	1661	0.3198880	0.09737269	0.09737269	0.006298090
5.9285714	14.0714286	51.71849	1589.282	1661	0.2964286	0.11810206	0.11810206	0.008693729

In [ ]:
```
## PLOTTING THE RESULTS ###
```

In [ ]:

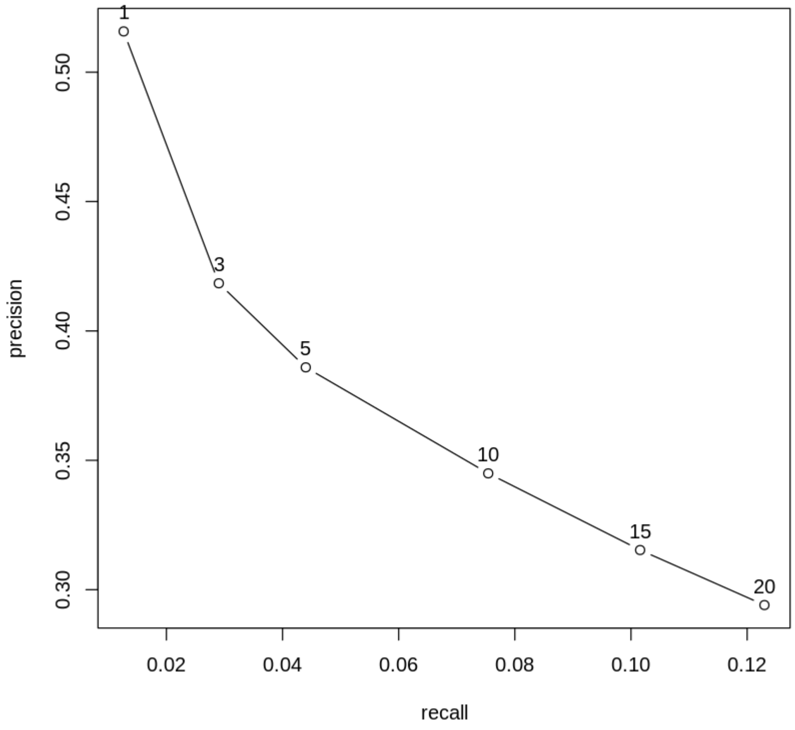
```
plot(results, annotate=TRUE)
graphics.off()
```



In [ ]:

```
### PRECISION AND RECALL PLOT ###

plot(results, "prec/rec", annotate=TRUE)
graphics.off()
```



In [ ]: