Package ‘recommenderlab’

February 20, 2015

Version 0.1-5
Date 2014-08-18
Title Lab for Developing and Testing Recommender Algorithms
Description Provides a research infrastructure to test and develop recommender algorithms.
Classification/ACM G.4, H.2.8
Depends R (>= 2.10.0), Matrix, registry, arules (>= 1.0-1), proxy
Imports methods
Suggests ROCR
License GPL-2
Copyright (C) Michael Hahsler (PCA and SVD implementation (C) Saurabh Bathnagar)
Author Michael Hahsler [aut, cre, cph]
Maintainer Michael Hahsler <mhahsler@lyle.smu.edu>
NeedsCompilation no
Repository CRAN
Date/Publication 2014-08-18 20:55:29

R topics documented:

- .get_parameters ......................................................... 2
- binaryRatingMatrix-class ........................................... 3
- calcPredictionAccuracy ................................................ 4
- dissimilarity ........................................................... 6
- dropNA ................................................................. 7
- evaluate ............................................................... 8
- evaluationResultList-class ......................................... 10
- evaluationResults-class ............................................. 11
Helper Functions

Description

Internal helper functions.

Usage

.gee.parameters(p, parameter)

Arguments

p default parameters.

parameter user specified parameters.

Value

Returns a user specified parameter list completed with default parameters.
Class "binaryRatingMatrix": A Binary Rating Matrix

Description

A matrix to represent binary rating data. 1 codes for a positive rating and 0 codes for either no or a negative rating. This coding is common for market basket data where products are either bought or not.

Objects from the Class

Objects can be created by calls of the form `new("binaryRatingMatrix", data = im)`, where `im` is an `itemMatrix` as defined in package `arules`, by coercion from a matrix (all non-zero values will be a 1), or by using `binarize` for an object of class "realRatingMatrix".

Slots

data: Object of class "itemMatrix" (see package `arules`)

Extends

Class "ratingMatrix", directly.

Methods

- `coerce` signature(from = "matrix", to = "binaryRatingMatrix")
- `coerce` signature(from = "itemMatrix", to = "binaryRatingMatrix")
- `coerce` signature(from = "data.frame", to = "binaryRatingMatrix")
- `coerce` signature(from = "binaryRatingMatrix", to = "matrix")
- `coerce` signature(from = "binaryRatingMatrix", to = "dgTMatrix")
- `coerce` signature(from = "binaryRatingMatrix", to = "ngCMatrix")
- `coerce` signature(from = "binaryRatingMatrix", to = "dgCMatrix")
- `coerce` signature(from = "binaryRatingMatrix", to = "itemMatrix")
- `coerce` signature(from = "binaryRatingMatrix", to = "list")

See Also

`itemMatrix` in `arules.getList`.
Examples

```r
## create a 0-1 matrix
m <- matrix(sample(c(0, 1), 50, replace=TRUE), nrow=5, ncol=10,
             dimnames=list(users=paste("u", 1:5, sep=''),
                           items=paste("i", 1:10, sep=''))

## coerce it into a binaryRatingMatrix
b <- as(m, "binaryRatingMatrix")

## coerce it back to see if it worked
as(b, "matrix")

## use some methods defined in ratingMatrix
dim(b)
dimnames(b)

## counts
rowCounts(b)
colCounts(b)

## plot
image(b)

## sample and subset
sample(b, 2)
b[1:2, 1:5]

## coercion
as(b, "list")
head(as(b, "data.frame"))
head(getData.frame(b, ratings=FALSE))

## creation from user/item tuples
df <- data.frame(user=c(1,1,2,2,2,3), items=c(1,4,1,2,3,5))
df
b2 <- as(df, "binaryRatingMatrix")
b2
as(b2, "matrix")
```

calcPredictionAccuracy

*Calculate the Prediction Error for a Recommendation*

Description

Calculate prediction accuracy. For predicted ratings MAE (mean average error), MSE (means squared error) and RMSE (root means squared error) are calculated. For topNLLists various binary classification metrics are returned.
calcPredictionAccuracy

Usage

calcPredictionAccuracy(x, data, ...)  
## S4 method for signature 'realRatingMatrix,realRatingMatrix'
calcPredictionAccuracy(x, data, byUser=FALSE,...)  
## S4 method for signature 'topNList,realRatingMatrix'
calcPredictionAccuracy(x, data, byUser=FALSE, given=NULL, goodRating=NA,...)  
## S4 method for signature 'topNList,binaryRatingMatrix'
calcPredictionAccuracy(x, data, byUser=FALSE, given=NULL,...)

Arguments

x Predicted items in a "topNList" or predicted ratings as a "realRatingMatrix"
data Actual ratings by the user as a "RatingMatrix"
byUser logical; Should the errors be averaged by user or over all recommendations?
given how many items were given to create the predictions.
goodRating threshold for determining what rating is a good rating. Used only if x is a topN-List and data is a "realRatingMatrix".
...

Value

Returns a vector with the average measures for byUser=TRUE. Otherwise, a matrix with a row for each user is returned.

References


See Also
topNList, binaryRatingMatrix, realRatingMatrix.

Examples

### real valued recommender
data(Jester5k)

e <- evaluationScheme(Jester5k[1:500,], method="split", train=0.9, k=1, given=15)
e

## create a user-based CF recommender using training data
r <- Recommender(getData(e, "train"), "UBCF")

## create predictions for the test data using known ratings (see given above)
p <- predict(r, getData(e, "known"), type="ratings")
p
## dissimilarity

### Description

Calculate dissimilarities/similarities between ratings by users and for items.

### Usage

```r
## S4 method for signature 'binaryRatingMatrix'
dissimilarity(x, y = NULL, method = NULL, args = NULL, which="users")
```
```
## S4 method for signature 'realRatingMatrix'
dissimilarity(x, y = NULL, method = NULL, args = NULL, which="users")
```
```
similarity(x, y = NULL, method = NULL, args = NULL, ...)
```
```
## S4 method for signature 'ratingMatrix'
similarity(x, y = NULL, method = NULL, args = NULL, which="users")
```

### Arguments

- `x`: a ratingMatrix.
y
method
args
which
...
### Arguments

- **x**  
  A matrix (for `dropNA()`) or a `dgCMatrix` (for `dropNA2matrix()`)

### Details

The representation is based on `dgCMatrix` in `Matrix` but instead of zeros, NAs are dropped. Be careful when working with the `dgCMatrix` directly since all dropped values are NA and not 0!

### Value

Returns a `dgCMatrix` or a matrix.

### See Also

- `dgCMematrix` in `Matrix`.

### Examples

```r
m <- matrix(sample(c(NA,0:5),50, replace=TRUE, prob=c(.5,rep(.5/6,6))), nrow=5, ncol=10, dimnames = list(users=paste('u', 1:5, sep=''), items=paste('i', 1:10, sep='')))

# drop all NAs in the representation
sparse <- dropNA(m)
sparse

# convert back to matrix
dropNA2matrix(sparse)

# Using regular coercion to `dgCMatrix`. Note that regular coercion
# to `dgCMatrix` drops 0s and not NAs!
as(m, "dgCMatrix")
```

---

### Description

Evaluates a single or a list of recommender model given an evaluation scheme.

### Usage

```r
evaluate(x, method, ...)
```

- **S4 method for signature** 'evaluationScheme,character'
  ```r
evaluate(x, method, type="topNList", n=1:10, parameter=NULL, progress = TRUE, keepModel=FALSE)
  ```

- **S4 method for signature** 'evaluationScheme,list'
  ```r
evaluate(x, method, type="topNList", n=1:10, parameter=NULL, progress = TRUE, keepModel=FALSE)
  ```
Arguments

- **x**: an evaluation scheme (class "evaluationScheme").
- **method**: a character string or a list. If a single character string is given it defines the recommender method used for evaluation. If several recommender methods need to be compared, method contains a nested list. Each element describes a recommender method and consists of a list with two elements: a character string named "method" containing the method and a list names "parameters" containing the parameters used for this recommender method. See Recommender for available methods.
- **type**: evaluate "topNList" or "ratings"?
- **n**: N (number of recommendations) of the top-N lists generated (only if type="topNList").
- **parameter**: a list with parameters for the recommender algorithm (only used when method is a single method).
- **progress**: logical; report progress?
- **keepModel**: logical; store used recommender models?
- **...**: further arguments.

Value

Returns an object of class "evaluationResults" or if method is a list an object of class "evaluationResultList".

See Also

- `evaluationScheme`, `evaluationResults`, `evaluationResultList`.

Examples

```r
### evaluate top-N list recommendations on a binary data set
data("MSWeb")
MSWeb10 <- sample(MSWeb[rowCounts(MSWeb) >10], 20)

### create an evaluation scheme
es <- evaluationScheme(MSWeb10, method="cross-validation", k=4, given=3)

### run evaluation
ev <- evaluate(es, "POPULAR", n=c(1,3,5,10))
ev

### look at the results (by the length of the topNList)
avg(ev)
plot(ev, annotate = TRUE)

### now run evaluate with a list
algorithms <- list(
  RANDOM = list(name = "RANDOM", param = NULL),
  POPULAR = list(name = "POPULAR", param = NULL),
  UBCF = list(name = "UBCF", param = NULL)
)```
Class "evaluationResultList": Results of the Evaluation of a Multiple Recommender Methods

Description

Contains the evaluation results for several runs using multiple recommender methods in form of confusion matrices. For each run the used models might be available.

Objects from the Class

Objects are created by evaluate.
Slots

.Data: Object of class "list": a list of "evaluationResults".

Extends

Class "list", from data part.

Methods

avg signature(x = "evaluationResultList"): returns a list of average confusion matrices.

signature(x = "evaluationResultList", i = "ANY", j = "missing", drop = "missing")

coerce signature(from = "list", to = "evaluationResultList")

show signature(object = "evaluationResultList")

See Also

evaluate, evaluationResults.

evaluationResults-class

Class "evaluationResults": Results of the Evaluation of a Single Recommender Method

Description

Contains the evaluation results for several runs using the same recommender method in form of
confusion matrices. For each run the used model might be available.

Objects from the Class

Objects are created by evaluate.

Slots

results: Object of class "list": contains objects of class "ConfusionMatrix", one for each run
specified in the used evaluation scheme.

Methods

avg signature(x = "evaluationResults"): returns an averaged confusion matrix.

getConfusionMatrix signature(x = "evaluationResults"): returns a list of confusion matrices.

getModel signature(x = "evaluationResults"): returns a list of used recommender models
(if available).

getRuns signature(x = "evaluationResults"): returns the number of runs/number of confusion matrices.

show signature(object = "evaluationResults")
See Also

*evaluate*

evaluationScheme  
*Creator Function for evaluationScheme*

**Description**

Creates an evaluationScheme object from a data set. The scheme can be a simple split into training and test data, k-fold cross-evaluation or using k independent bootstrap samples.

**Usage**

```r
evaluationScheme(data, ...)  

## S4 method for signature 'ratingMatrix'
evaluationScheme(data, method="split",
                 train=0.9, k=NULL, given, goodRating = NA)
```

**Arguments**

- `data`: data set as a ratingMatrix.
- `method`: a character string defining the evaluation method to use (see details).
- `train`: fraction of the data set used for training.
- `k`: number of folds/times to run the evaluation (defaults to 10 for cross-validation and bootstrap and 1 for split).
- `given`: single number of items given for evaluation or a vector of length of data giving the number of items given for each observation.
- `goodRating`: numeric; threshold at which ratings are considered good for evaluation. E.g., with `goodRating=3` all items with actual user rating of greater or equal 3 are considered positives in the evaluation process. Note that this argument is only used if the ratingMatrix is a of subclass realRatingMatrix!
- `...`: further arguments.

**Details**

`evaluationScheme` creates an evaluation scheme (training and test data) with k runs and one of the given methods:

- "split" randomly assigns the proportion of objects given by `train` to the training set and the rest is used for the test set.
- "cross-validation" creates a k-fold cross-validation scheme. The data is randomly split into k parts and in each run k-1 parts are used for training and the remaining part is used for testing. After all k runs each part was used as the test set exactly once.
"bootstrap" creates the training set by taking a bootstrap sample (sampling with replacement) of size \textit{train} times number of users in the data set. All objects not in the training set are used for testing.

For evaluation, the scheme chooses from the test data for each user given items are randomly as "known" items, the remaining items are "unknown." The known items are used to create a prediction and it is evaluated how well the algorithm predicts the unknown items.

**Value**

Returns an object of class "evaluationScheme".

**References**


**See Also**

\texttt{getData, evaluationScheme, ratingMatrix}.

**Examples**

```r
data("MSWeb")
MSWeb10 <- sample(MSWeb[rowCounts(MSWeb) > 10, ], 50)
MSWeb10

esSplit <- evaluationScheme(MSWeb10, method="split",
   train = 0.9, k=1, given=3)
esSplit

esCross <- evaluationScheme(MSWeb10, method="cross-validation",
   k=4, given=3)
esCross
```

---

**evaluationScheme-class**

*Class "evaluationScheme": Evaluation Scheme*

**Description**

An evaluation scheme created from a data set. The scheme can be a simple split into training and test data, k-fold cross-evaluation or using k bootstrap samples.

**Objects from the Class**

Objects can be created by \texttt{evaluationScheme(data, method="split", train=0.9, k=NULL, given=3)}. 
Slots

data: Object of class "ratingMatrix"; the data set.
given: Object of class "integer"; items given for evaluation.
goodRating: Object of class "numeric"; Rating at which an item is considered a positive for evaluation.
k: Object of class "integer"; number of runs for evaluation. Default is 1 for method "split" and 10 for "cross-validation" and "bootstrap".
knownData: Object of class "ratingMatrix"; data set with only known (given) items.
method: Object of class "character"; evaluation method. Available methods are: "split", "cross-validation" and "bootstrap".
runstrain: Object of class "list"; internal representation for the split in training and test data for the evaluation runs.
train: Object of class "numeric"; portion of data used for training for "split" and "bootstrap".
unknownData: Object of class "ratingMatrix"; data set with only unknown items.

Methods

dataGet signature(x = "evaluationscheme"): access data. Parameters are type ("train", "known" or "unknown") and run (1...k).
show signature(object = "evaluationscheme")

See Also

ratingMatrix and the creator function evaluationScheme.

Description

Create a list or data.frame representation for various objects used in recommenderlab. These functions are used in addition to available coercion to allow for parameters like decode.

Usage

gelist(from, ...)
## S4 method for signature 'realRatingMatrix'
gelist(from, decode = TRUE, ratings = TRUE, ...)
## S4 method for signature 'binaryRatingMatrix'
gelist(from, decode = TRUE, ...)
## S4 method for signature 'topNLList'
gelist(from, decode = TRUE, ...)

getDataFrame(from, ...)
## S4 method for signature 'ratingMatrix'
getDataFrame(from, decode = TRUE, ratings = TRUE, ...)

getList
Arguments

from object to be represented as a list.
decode use item names or item IDs (column numbers) for items?
ratings include ratings in the list or data.frame?
... further arguments (currently unused).

Details

Lists have one vector with items (and ratings) per user. The data.frame has one row per rating with
the user in the first column, the item as the second and the rating as the third.

Value

Returns a list or a data.frame.

See Also

binaryRatingMatrix, realRatingMatrix, topNList.

Examples

```r
data(Jester5k)
getList(Jester5k[1,])
getData.frame(Jester5k[1,])
```

---

**Jester5k**

**Jester dataset (5k sample)**

Description

The data set contains a sample of 5000 users from the anonymous ratings data from the Jester Online
Joke Recommender System collected between April 1999 and May 2003.

Usage

```r
data(Jester5k)
```

Format

The format is: Formal class ‘realRatingMatrix’ [package “recommenderlab”]

Details

5000 x 100 rating matrix (5000 users and 100 jokes) with ratings between -10.00 and +10.00. All
selected users have rated 36 or more jokes.
References


Examples

data(Jester5k)
Jester5k

nratings(Jester5k)
hist(getRatings(Jester5k), main="Distribution of ratings")

---

MovieLense Dataset (100k)

Description

The 100k MovieLense ratings data set. The data was collected through the MovieLens web site (movielens.umn.edu) during the seven-month period from September 19th, 1997 through April 22nd, 1998. The data set contains about 100,000 ratings (1-5) from 943 users on 1664 movies.

Usage

data(MovieLense)

Format

The format is an object of class "realRatingMatrix"

Source

GroupLens Research, http://www.grouplens.org/node/73

References


Examples

data(MovieLense)
MovieLense

## look at the first few ratings of the first user
as(MovieLense[1,], "list")[[1]][1:10]

## visualize part of the matrix
image(MovieLense[1:100,1:100])

## number of ratings per user
hist(rowCounts(MovieLense))

## number of ratings per movie
hist(colCounts(MovieLense))

## mean rating (averaged over users)
mean(rowMeans(MovieLense))

---

### MSWeb

**Anonymous web data from www.microsoft.com**

#### Description

Vroots visited by users in a one week timeframe.

#### Usage

data(MSWeb)

#### Format

The format is: Formal class "binaryRatingMatrix".

#### Details

The data was created by sampling and processing the www.microsoft.com logs. The data records the use of www.microsoft.com by 38000 anonymous, randomly-selected users. For each user, the data lists all the areas of the web site (Vroots) that user visited in a one week timeframe in February 1998.

This dataset contains 32710 valid users and 285 Vroots.

#### Source


#### References

Examples

```r
data(MSWeb)
MSWeb

nratings(MSWeb)

## look at first two users
as(MSWeb[1:2,], "list")

## items per user
hist(rowCounts(MSWeb), main="Distribution of Vroots visited per user")
```

**normalize**

*Normalize the ratings*

Description

Provides the generic for normalize/denormalize and a method to normalize/denormalize the ratings in a realRatingMatrix.

Usage

```r
normalize(x, ...)  
## S4 method for signature 'realRatingMatrix'
normalize(x, method="center", row=TRUE)
```

```r
denormalize(x, ...)  
## S4 method for signature 'realRatingMatrix'
denormalize(x, method=NULL, row=NULL, 
               factors=NULL)
```

Arguments

- **x**: a realRatingMatrix.
- **method**: normalization method. Currently "center" or "Z-score".
- **row**: logical; normalize rows (or the columns)?
- **factors**: a list with the factors to be used for denormalizing (elements are "mean" and "sds"). Usually these are not specified and the values stored in **x** are used.
- **...**: further arguments (currently unused).

Details

Normalization tries to reduce the individual rating bias by row centering the data, i.e., by subtracting from each available rating the mean of the ratings of that user (row). Z-score in addition divides by the standard deviation of the row/column. Normalization can also be done on columns.

Denormalization reverses normalization. It uses the normalization information stored in **x** unless the user specifies method, row and factors.
Value

A normalized realRatingMatrix.

Examples

## create a matrix with ratings
m <- matrix(sample(c(NA,0:5),50, replace=TRUE, prob=c(.5,rep(.5/6,6))), nrow=5, ncol=10, dimnames = list(users=paste('u', 1:5, sep=''), items=paste('i', 1:10, sep='')))## do normalization
r <- as(m, "realRatingMatrix")
r_n1 <- normalize(r)
r_n2 <- normalize(r, method="Z-score")

r
r_n1
r_n2

## show normalized data
image(r, main="Raw Data")
image(r_n1, main="Centered")
image(r_n2, main="Z-Score Normalization")

---

**Plot Evaluation Results**

Description

Creates precision-recall or ROC plots for recommender evaluation results.

Usage

```r
## S4 method for signature 'evaluationResults'
plot(x, y,
    avg = TRUE, add=FALSE, type = "b", annotate = FALSE, ...)
## S4 method for signature 'evaluationResultList'
pplot(x, y,
    xlim=NULL, ylim=NULL, col = NULL, pch = NULL, lty = 1,
    avg = TRUE, type = "b", annotate= 0, legend="bottomright", ...)
```

Arguments

- **x**: the object to be plotted.
- **y**: a character string indicating the type of plot (e.g., "ROC" or "prec/rec").
- **avg**: plot average of runs?
- **add**: add to a plot?
predict  

Predict Recommendations

Description

Creates recommendations using a recommender model and data about new users.

Usage

```r
## S4 method for signature 'Recommender'
predict(object, newdata, n = 10, data=NULL,
        type="topNlist", ...)
```

Arguments

- **object**: a recommender model (class "Recommender").
- **newdata**: data for active users (class "ratingMatrix") or the index of users in the training data to create recommendations for. If an index is used then some recommender algorithms need to be passed the training data as argument data. Some algorithms may only support user indices.
- **n**: number of recommendations in the top-N list.
- **data**: training data needed by some recommender algorithms if newdata is a user index and not user data.
- **type**: type of recommendation. The default type is "topNlist" which creates a top-N recommendation list with recommendations. Some recommenders can also create other results (e.g., type "ratings" returns only predicted ratings with known ratings represented by NA, or type "ratingMatrix" which returns a completed rating matrix).
- **...**: further arguments.

See Also

evaluationResults, evaluationResultList. See evaluate for examples.
Value

Returns an object of class "topNList" or of other appropriate classes.

See Also

Recommender, ratingMatrix.

Examples

data("MovieLense")
MovieLense100 <- MovieLense[rowCounts(MovieLense) >100,]
train <- MovieLense100[1:50]

rec <- Recommender(train, method = "POPULAR")
rec

## create top-N recommendations for new users
pre <- predict(rec, MovieLense100[101:102], n = 10)
pre
as(pre, "list")

## predict ratings for new users
pre <- predict(rec, MovieLense100[101:102], type="ratings")
pre
as(pre, "matrix")[,1:10]

## create recommendations using user ids with ids 1..10 in the
## training data
pre <- predict(rec, 1:10 , data = train, n = 10)
pre
as(pre, "list")
realRatingMatrix-class

Methods

[ signature(x = "ratingMatrix", i = "ANY", j = "ANY", drop = "ANY"): subset the
rating matrix (drop is ignored).

coerce signature(from = "ratingMatrix", to = "list")

coerce signature(from = "ratingMatrix", to = "data.frame"): a data.frame with three
columns. Col 1 contains user ids, col 2 contains item ids and col 3 contains ratings.

colCounts signature(x = "ratingMatrix"): number of ratings per column.

rowCounts signature(x = "ratingMatrix"): number of ratings per row.

colMeans signature(x = "ratingMatrix"): column-wise rating means.

rowMeans signature(x = "ratingMatrix"): row-wise rating means.

dim signature(x = "ratingMatrix"): dimensions of the rating matrix.

dimnames<- signature(x = "ratingMatrix", value = "list"): replace dimnames.

dimnames signature(x = "ratingMatrix"): retrieve dimnames.

getNormalize signature(x = "ratingMatrix"): returns a list with normalization information
for the matrix (NULL if data is not normalized).

image signature(x = "ratingMatrix"): plot the matrix.

nratings signature(x = "ratingMatrix"): number of ratings in the matrix.

sample signature(x = "ratingMatrix"): sample from users (rows).

show signature(object = "ratingMatrix")

See Also

See implementing classes realRatingMatrix and binaryRatingMatrix. See getList, getData.frame,
similarity, dissimilarity and dissimilarity.

realRatingMatrix-class

Class "realRatingMatrix": Real-valued Rating Matrix

Description

A matrix containing ratings (typically 1-5 stars, etc.).

Objects from the Class

Objects can be created by calls of the form new("realRatingMatrix", data = m), where m is
sparse matrix of class dgCMatrix (see package Matrix), or by coercion from a regular matrix.

Slots

data: Object of class "dgCMatrix", a sparse matrix defined in package Matrix.
realRatingMatrix-class

Extends

Class "ratingMatrix", directly.

Methods

coerce signature(from = "matrix", to = "realRatingMatrix")
coerce signature(from = "realRatingMatrix", to = "matrix")
coerce signature(from = "data.frame", to = "realRatingMatrix"): coercion from a
data.frame with three columns. Col 1 contains user ids, col 2 contains item ids and col 3
contains ratings.
coerce signature(from = "realRatingMatrix", to = "dTMatrix")
coerce signature(from = "realRatingMatrix", to = "ngCMatrix")
coerce signature(from = "realRatingMatrix", to = "dgCMatrix")
binarize signature(x = "realRatingMatrix"): create a "binaryRatingMatrix" by setting all
ratings larger or equal to the argument minRating as 1 and all others to 0.
getRatings signature(x = "realRatingMatrix"): returns all ratings in x as a numeric vector.
removeKnownRatings signature(x = "realRatingMatrix"): removes all ratings in x for
which ratings are available in the realRatingMatrix (of same dimensions as x) passed as the
argument known.
rowSds signature(x = "realRatingMatrix"): calculate the standard deviation of ratings for
rows (users).
colSds signature(x = "realRatingMatrix"): calculate the standard deviation of ratings for
columns (items).

See Also

See ratingMatrix inherited methods, binaryRatingMatrix, topNList, getList and getData.frame.
Also see dgCMatrix, dTMatrix and ngCMatrix in Matrix.

Examples

```r
## create a matrix with ratings
m <- matrix(sample(c(NA,0:5),100, replace=TRUE, prob=c(.7,rep(.3/6,6))),
            nrow=10, ncol=10, dimnames = list(
                user=paste('u', 1:10, sep=''),
                item=paste('i', 1:10, sep='')
            ))

m

## coerce into a realRatingMatrix
r <- as(m, "realRatingMatrix")
r

## get some information
dimnames(r)
rowCounts(r)
colCounts(r)
```
Recommender

Description

Learns a recommender model from given data.

Usage

Recommender(data, ...)

## S4 method for signature 'ratingMatrix'
Recommender(data, method, parameter=NULL)

Arguments

data training data.

method a character string defining the recommender method to use (see details).

parameter parameters for the recommender algorithm.

... further arguments.

Details

Recommender uses the registry mechanism from package registry to manage methods. This let's the user easily specify and add new methods. The registry is called recommenderRegistry. See examples section.
Recommender-class

Value

An object of class 'Recommender'.

See Also

recommender, ratingMatrix.

Examples

data("MSWeb")
MSWeb10 <- sample(MSWeb[rowCounts(MSWeb) >10,], 100)

rec <- Recommender(MSWeb10, method = "POPULAR")
rec
str(getModel(rec))

## look at registry
recommenderRegistry$get_entry_names()
recommenderRegistry$get_entry("POPULAR", dataType = "binaryRatingMatrix")

Recommender-class

Class "Recommender": A Recommender Model

Description

Represents a recommender model learned for a given data set (a rating matrix).

Objects from the Class

Objects are created by the creator function Recommender(data, method, parameter = NULL)

Slots

method: Object of class "character"; used recommendation method.
datatype: Object of class "character"; concrete class of the input data.
ntrain: Object of class "integer"; size of training set.
model: Object of class "list"; the model.
predict: Object of class "function"; code to compute a recommendation using the model.

Methods

getModel signature(x = "Recommender"): retrieve the model.
predict signature(object = "Recommender"): create recommendations for new data (argument newdata).
show signature(object = "Recommender")
topNList-class

Class "topNList": Top-N List

Description

Recommendations a Top-N list.

Objects from the Class

Objects can be created by predict with a recommender model and new data.

Slots

items: Object of class "list". Each element in the list represents a top-N recommendation (an integer vector) with item IDs (column numbers in the rating matrix). The items are ordered in each vector.

itemLabels: Object of class "character"

n: Object of class "integer" specifying the number of items in each recommendation. Note that the actual number on recommended items can be less depending on the data and the used algorithm.

Methods

coerce signature(from = "topNList", to = "dgTMatrix")
coerce signature(from = "topNList", to = "dgCMatrix")
coerce signature(from = "topNList", to = "ngCMatrix")
coerce signature(from = "topNList", to = "matrix")
coerce signature(from = "topNList", to = "list")

bestN signature(x = "topNList"): returns only the best n recommendations (second argument is n which defaults to 10).

getTopNLists signature(x = "realRatingMatrix"): create top-N lists from the ratings in x. Arguments are n and minRating. Items with a rating below minRating will not be part of the top-N list.

length signature(x = "topNList"): for how many users does this object contain a top-N list?

removeKnownItems signature(x = "topNList"): remove items from the top-N list which are known (have a rating) for the user given as a ratingMatrix passed on as argument known.

colCounts signature(x = "topNList"): in how many top-N does each item occur?

rowCounts signature(x = "topNList"): number of recommendations per user.

show signature(object = "topNList")

See Also

evaluate, realRatingMatrix
Index

*Topic classes
  binaryRatingMatrix-class, 3
evaluationResultList-class, 10
evaluationResults-class, 11
evaluationScheme-class, 13
ratingMatrix-class, 21
realRatingMatrix-class, 22
Recommender-class, 25
topNList-class, 26

*Topic datasets
  JesterSk, 15
  MovieLens, 16
  MSWeb, 17

*Topic manipulation
  normalize, 18
  get_parameters, 2
  [., evaluationResultList, ANY, missing, missing-method]
  (evaluationResultList-class), 10
  [., ratingMatrix, ANY, ANY, ANY-method]
  (ratingMatrix-class), 21
  avg (evaluationResults-class), 11
  avg, evaluationResultList-method
  (evaluationResultList-class), 10
  avg, evaluationResults-method
  (evaluationResults-class), 11

  bestN (topNList-class), 26
  bestN, topNList-method (topNList-class), 26
  binarize (realRatingMatrix-class), 22
  binarize, realRatingMatrix-method
  (realRatingMatrix-class), 22
  binaryRatingMatrix, 5, 15, 22, 23
  binaryRatingMatrix-class, 3
calcPredictionAccuracy, 4
calcPredictionAccuracy, realRatingMatrix, realRatingMatrix-method
  (calcPredictionAccuracy), 4
calcPredictionAccuracy, topNList, binaryRatingMatrix-method
  (calcPredictionAccuracy), 4
calcPredictionAccuracy, topNList, realRatingMatrix-method
  (calcPredictionAccuracy), 4
coerce, binaryRatingMatrix, dgCMatrix-method
  (binaryRatingMatrix-class), 3
coerce, binaryRatingMatrix, dgTMatrix-method
  (binaryRatingMatrix-class), 3
coerce, binaryRatingMatrix, itemMatrix-method
  (binaryRatingMatrix-class), 3
coerce, binaryRatingMatrix, list-method
  (binaryRatingMatrix-class), 3
coerce, binaryRatingMatrix, method
  (binaryRatingMatrix-class), 3
  coerce, binaryRatingMatrix, ngCMatrix-method
  (binaryRatingMatrix-class), 3
coerce, data.frame, binaryRatingMatrix-method
  (binaryRatingMatrix-class), 3
coerce, data.frame, realRatingMatrix-method
  (realRatingMatrix-class), 22
coerce, itemMatrix, binaryRatingMatrix-method
  (binaryRatingMatrix-class), 3
coerce, list, evaluationResultList-method
  (evaluationResultList-class), 10
coerce, matrix, binaryRatingMatrix-method
  (binaryRatingMatrix-class), 3
coerce, matrix, realRatingMatrix-method
  (realRatingMatrix-class), 22
coerce, ratingMatrix, data.frame-method
  (ratingMatrix-class), 21
coerce, ratingMatrix, list-method
  (ratingMatrix-class), 21
coerce, realRatingMatrix, dgCMatrix-method
  (realRatingMatrix-class), 22
coerce, realRatingMatrix, dgTMatrix-method
  (realRatingMatrix-class), 22

27
coerce, realRatingMatrix, matrix-method
    (realRatingMatrix-class), 22
coerce, realRatingMatrix, ngCMatrix-method
    (realRatingMatrix-class), 22
coerce, topNList, dgCMatrix-method
    (topNList-class), 26
calculate.topnlist, dgCMatrix-method
    (topNList-class), 26
calculate.topnlist, list-method
    (topNList-class), 26
calculate.topnlist, matrix-method
    (topNList-class), 26
calculate.topnlist, matrix-method
    (topNList-class), 26
calculate.topnlist, ngCMatrix-method
    (topNList-class), 26
calculate.topnlist, ratingMatrix-method
    (topNList-class), 26
calculate.topnlist, ratingMatrix-method
    (topNList-class), 26
calculate.topnlist, ratingMatrix-method
    (topNList-class), 26
calculate.topnlist, topnlist-method
    (topNList-class), 26
calculate.topnlist, topnlist-method
    (topNList-class), 26
calculate.topnlist, list-method
    (topNList-class), 26
calculate.topnlist, matrix-method
    (topNList-class), 26
calculate.topnlist, matrix-method
    (topNList-class), 26
calculate.topnlist, ngCMatrix-method
    (topNList-class), 26
calculate.topnlist, ratingMatrix-method
    (topNList-class), 26
calculate.topnlist, ratingMatrix-method
    (topNList-class), 26
confusionMatrix-class
    (evaluationResults-class), 11
denormalize (normalize), 18
denormalize, realRatingMatrix-method
    (normalize), 18
dgCMatrix, 8, 23
dgTMatrix, 23
dim, ratingMatrix-method
    (ratingMatrix-class), 21
dimnames, ratingMatrix-method
    (ratingMatrix-class), 21
dimnames<-, ratingMatrix, list-method
    (ratingMatrix-class), 21
dissimilarity, 6, 7, 22
dissimilarity, binaryRatingMatrix-method
    (dissimilarity), 6
dissimilarity, realRatingMatrix-method
    (dissimilarity), 6
dropNA, 7
dropNA2matrix (dropNA), 7
evaluate, 8, 11, 12, 20, 26
evaluate, evaluationScheme, character-method
    (evaluate), 8
INDEX

itemMatrix, 3
Jester5k, 15
length, topNList-method (topNList-class), 26
list, 11
MovieLense, 16
MSWeb, 17
ngCMatrix, 23
normalize, 18
normalize, realRatingMatrix-method (normalize), 18
nratings (ratingMatrix-class), 21
nratings, ratingMatrix-method (ratingMatrix-class), 21
plot, 19
plot, evaluationResultList-method (plot), 19
plot, evaluationResults-method (plot), 19
predict, 20
predict, Recommender-method (predict), 20
ratingMatrix, 3, 7, 13, 14, 21, 23, 25
realRatingMatrix, 5, 15, 22, 26
realRatingMatrix-class, 22
Recommender, 21, 24, 25, 26
Recommender, ratingMatrix-method (Recommender), 24
Recommender-class, 25
recommenderRegistry (Recommender), 24
removeKnownItems (topNList-class), 26
removeKnownItems, topNList-method (topNList-class), 26
removeKnownRatings
(realRatingMatrix-class), 22
removeKnownRatings, realRatingMatrix-method (realRatingMatrix-class), 22
rowCounts (ratingMatrix-class), 21
rowCounts, ratingMatrix-method (ratingMatrix-class), 21
rowCounts, topNList-method (topNList-class), 26
rowMeans, ratingMatrix-method (ratingMatrix-class), 21
rowSds (realRatingMatrix-class), 22
rowSds, realRatingMatrix-method (realRatingMatrix-class), 22
sample, ratingMatrix-method (ratingMatrix-class), 21
show, evaluationResultList-method (evaluationResultList-class), 10
show, evaluationResults-method (evaluationResults-class), 11
show, evaluationScheme-method (evaluationScheme-class), 13
show, ratingMatrix-method (ratingMatrix-class), 21
show, Recommender-method (Recommender-class), 25
show, topNList-method (topNList-class), 26
similarity, 22
similarity (dissimilarity), 6
similarity, ratingMatrix-method (dissimilarity), 6
topNList, 5, 15, 23
topNList-class, 26