Package 'futile.matrix'

April 20, 2018

Type Package

Title Random Matrix Generation and Manipulation

Version 1.2.7

Date 2018-04-20

Author Brian Lee Yung Rowe

Maintainer Brian Lee Yung Rowe <r@zatonovo.com>

Depends R (>= 3.0.0)

Imports utils, lambda.r (>= 1.1.6), lambda.tools, futile.logger (>= 1.3.0), RMTstat

Suggests testit

Description A collection of functions for manipulating matrices and generating ensembles of random matrices. Used primarily to identify the cutoff point for the noise portion of the eigenvalue spectrum.

License LGPL-3

LazyLoad yes

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-20 17:06:01 UTC

R topics documented:

`futile.matrix-package` *A collection of matrix manipulation functions*

Description

This package provides functions for working with random matrices. It also provides various convenience functions for examining data within matrices as well as some optimized functions for reading matrices in various formats.

Details

Package:	futile.matrix
Type:	Package
Version:	1.2.7
Date:	2018-04-20
License:	LGPL-3
LazyLoad:	yes

Random matrix ensembles can be created using this package. It's also possible to fit the Marcenko-Pastur distribution to Wishart matrices, enabling you to isolate the noise portion of the eigenvalue spectrum.

Author(s)

Brian Lee Yung Rowe <r@zatonovo.com>

References

The Distribution Functions of Random Matrix Theory, Craig A. Tracy, UC Davis http://www.math.ucsc.edu/research/rmtg.html

Introduction to the Random Matrix Theory: Gaussian Unitary Ensemble and Beyond http://arxiv.org/abs/math-ph/0412017v2

Tyler's M-Estimator, Random Matrix Theory, and Generalized Elliptical Distributions with Applications to Finance http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1287683

See Also

select, expand, read.matrix

Examples

```
# Generate a random ensemble
m <- rmatrix(WishartModel(100,400))

# Select sub-matrices
library(datasets)
select(swiss, "Rive")
select(swiss, col.pat='^E')
select(swiss, "Rive", '^E') <- -1
dimnames <- list( c(rownames(swiss), 'Zermat', 'Zurich', 'Geneva'),
 c(colnames(swiss), 'Age','Hair.Color') )
my.swiss <- expand(swiss, dimnames)
```

---

arrange *Order matrix elements based on rownames and/or colnames*

Description

This is a convenience function for ordering rows and columns within a matrix.

Usage

```
arrange(m, order.rows = TRUE, order.cols = TRUE, comparator = NULL)
```

Arguments

m	A matrix whose rows and/or columns are to be arranged
order.rows	Whether rows are to be ordered. Defaults to TRUE
order.cols	Whether columns are to be ordered. Defaults to TRUE
comparator	A function to define the ordering. Currently unused

Details

To ensure proper operations are performed, the ordering of rows and columns within matrix operands must be consistent. Arrange conveniently performs this ordering.

In the future, a comparator will be added so that custom orderings can be applied to the function.

Value

A matrix whose rows and/or columns have been ordered. By default, both rows and columns are ordered.

Examples

```
library(datasets)
arrange(state.x77)
```

`assign_matrix_dense` *Create a matrix from a triplet representation*

Description

A somewhat optimized constructor of dense matrices from triplets

Usage

```
assign_matrix_dense(raw, row.ids, col.ids)
```

Arguments

`raw`	data.frame in triplet form where each row represents a triplet
`row.ids`	The row ids
`col.ids`	The col ids

Value

A matrix

`ct` *Perform the conjugate transpose of a matrix*

Description

Convenience function

Arguments

`m`	A matrix

Value

THe conjugate transpose of the original matrix

Usage

ct(m) %::% matrix : matrix

ct(m)

Details

This is a convenience function to compute the conjugate transpose. For real-valued matrices, ct(m) = t(m).

Examples

```
x <- matrix(rcomp(16), nrow=4)
ct(x)
```

---

cutoff *Calculate the upper bound of the noise spectrum*

Description

Finds the cutoff point between the noise portion of the eigenvalue spectrum and the "signal" part.

Arguments

p	A random matrix

Value

The eigenvalue associated with the upper bound of the noise spectrum

Usage

```
cutoff(p)
cutoff(p, es, estimator)
```

Examples

```
cutoff(rmatrix(WignerModel(100)))
```

---

dmatrix *Get the density for the given random matrix model*

Description

This is the analog to dnorm except for a random matrix

Arguments

x	A numeric vector
model	The random matrix model

Value

A vector of corresponding densities

Usage

dmatrix(x, model)

dmatrix(x, model)

dmatrix(x, model)

Examples

```
m <- WignerModel(100)
dmatrix(seq(-1,1,by=0.02), m)
```

---

domain *Get the bounds of the eigenvalues for the given model*

Description

Currently only supports Wishart matrices

Arguments

model	The random matrix model

Value

A vector containing the lower and upper bounds

Usage

domain(model)

---

expand *Expand a matrix to larger dimensions, filling in new entries*

Description

Provides a convenient mechanism for expanding the dimensions of a matrix with a specified default value. This is particularly useful if the matrix needs to match dimensions with another matrix. Expand can take either another matrix or a set of dimnames to grow the matrix. In either case, the expanded matrix will have dimensions that match the explicitly or implicitly specified dimnames.

Usage

```
expand(m, target, default = 0)
```

Arguments

m	A matrix to expand
target	A list containing the dimnames or a matrix that contains dimnames
default	The default value to use for the new entries

Details

To properly expand m to target, the rownames and colnames of m must be a strict subset of target's rownames and colnames. If this requirement is not satisfied, the behavior is currently undefined (although most likely an error will result). In the future, the behavior might be configurable to drop those rows/columns that are not in target's rownames/colnames.

In general, expand tries to err on the side of accomodation, although the implementation is incomplete. If target is a list, then the format is the same as when constructing a matrix and passing dimnames as an argument. Currently, only a list or a matrix are supported. If a list, target[[0]] represent the row names and target[[1]] are the column names. This could be relaxed in the future to any object that has a rownames and colnames.

Note that a current limitation/feature in expand is that it orders the resulting matrix by rows and columns. More precise control needs to be provided here, with the default being the ordering of the rows and columns conforming to target.

TODO: Consistency check to ensure all rownames/colnames of m are a subset of target

Value

The expanded matrix with rows and columns corresponding to the target dimnames

Examples

```
rows.m <- c('row.1', 'row.2', 'row.3')
cols.m <- c('col.1', 'col.2')
rows.n <- c(rows.m, 'row.4')
cols.n <- c(cols.m, 'col.3')
m <- matrix(c(1,4,7,2,5,8), ncol=2, dimnames=list(rows.m,cols.m))
n <- matrix(c(1,4,7,10,2,5,8,11,3,6,9,12), ncol=3,
  dimnames=list(rows.n,cols.n))
expand(m, n)
```

---

fit.density *Fit the eigenvalue spectrum to model*

Description

This only works for fitting the Marcenko-Pastur distribution. It's also not designed for external use.

Arguments

lambda	Eigenvalues
fitter	A fit function

Value

Optimal parameters

Examples

```
## Not run:
m <- rmatrix(WishartModel(50, 200))
es <- eigen(m)
fit.density(es, MaximumLikelihoodFit(hint=c(1,1)))

## End(Not run)
```

---

peek *Peek inside a matrix or vector*

Description

Peek is a simple utility to conveniently look at a portion of a matrix. This is similar to head and tail but provides a 2-dimensional slice instead of a complete row. This is useful for debugging and inspecting matrices.

Usage

```
peek(x, upper = 5, lower = 1)
```

Arguments

x	Any object that supports subsetting
upper	The upper bound in the subsetting
lower	The lower bound in the subsetting

Value

A subset of the original matrix, data.frame, etc.

Examples

```
m <- matrix(c(1,3,4,2, 5,10,11,2, 3,42,8,22, 23,15,3,8), ncol=4)
peek(m, 2)
```

print.Ensemble *Print a random matrix ensemble*

Description

Pretty print the ensemble instead of dumping a bunch of matrices

Usage

```
## S3 method for class 'Ensemble'
print(x, ...)
```

Arguments

x	An ensemble of random matrices
...	Reserved for later

Value

Used for side-effects

See Also

Ensemble

RandomMatrixModel *Type constructors for random matrices and ensembles of random matrices*

Description

Provides type constructors for creating random matrices. Various studies can be initiated afterward.

Arguments

real	Whether the matrix has real components or not
n	Number of rows
m	Number of columns
m1	Number of columns
m2	Number of columns
sd	Standard deviation of the sample population
count	Number of matrices in the ensemble
model	The random matrix model to use

Value

Returns a model type. Use with `rmatrix` or `Ensemble` to generate actual matrices.

Usage

RandomMatrixModel(real=TRUE, ...)

WignerMatrix(x, model)

WishartModel(n, m, sd=1, ...)

JacobiModel(n, m1, m2, ...)

Ensemble(count, model)

Examples

```
model <- WignerModel(10)
m <- rmatrix(model)
e <- Ensemble(20, model)
```

---

rcomp *Generate random complex numbers*

Description

Generate random complex numbers using the specified distribution. By default `rnorm` is used.

Arguments

`n`	Length of the output vector
`dist`	The distribution for the random number genertor

Value

A vector of random numbers

Usage

rcomp(n, dist)

rcomp(n, dist=rnorm)

Details

This function is used primarily to generate random matrices.

Examples

```
rcomp(10)
rcomp(10, runif)
```

read.matrix *Read a sparse matrix from a file and return a matrix*

Description

Reading matrices from files can be time consuming depending on the size of the matrix. read.matrix implements a fairly efficient routine to read in sparse matrices and return dense matrix counterparts.

Usage

```
read.matrix(file, header = FALSE, skip = 1, row.ids = NULL,
  col.ids = NULL, colClasses = c("character", "character", "numeric"),
  assign.fn = assign_matrix_dense, filter.fn = NULL, ...)
```

Arguments

file	A file or connection to read from
header	Whether header lines exist defining all possible rows an columns. If this is false, then the defined triplet elements will produce the complete set of rows and columns.
skip	The number of rows to skip. This assumes there is a single header line, which is skipped.
row.ids	If header is TRUE, the row number that defines the row.ids If header == FALSE, the row.ids to use for the matrix
col.ids	If header is TRUE, the col number that defines the col.ids If header == FALSE, the col.ids to use for the matrix
colClasses	The classes to use for the columns in the triplet file
assign.fn	The function to use to construct the sparse representation that is then converted to a dense matrix
filter.fn	An optional function used to filter/clean the input data and/or row/column ids. The signature of filter.fn must have arguments for data, row.ids, and col.ids
...	Additional arguments to pass to the construction portion of the implementation

Details

Matrices that have dimensions on the order of thousands can be slow to load into R. 'read.matrix' provides an efficient implementation for reading sparse matrices in triplet form from a file or other connection. This version removes dependencies from other packages and shows a speed improvement over those methods.

The primary benefit of this function is that named rows and columns can be used as opposed to integer indexes, as compared to the slam package. The other main motivation is that if the memory is available, dense matrix calculations can be faster than their sparse counterparts, not to mention having a wider range of operators available.

When header == TRUE, the row names and/or column names are read from the file. The names are expected to be comma separated in a single line.

Various methods can be used to construct a sparse matrix representation that is used as the basis for constructing the dense matrix. Currently only the `assign_matrix_dense` function is available, which works well for matrices in triplet form.

Value

A matrix object generated from sparse triplet data

Author(s)

Brian Lee Yung Rowe

Examples

```
## Not run:
  path <- system.file('sample-data/triplet.csv', package='futile.matrix')
  m <- read.matrix(path)

  rows <- paste('row', 1:10000, sep='.')
  cols <- paste('col', 1:10000, sep='.')
  n <- read.matrix(path, row.ids=rows, col.ids=cols)

## End(Not run)
```

`rmatrix` *Generation of random matrices*

Description

Generate various types of random matrices

Arguments

`model`	The matrix model to use, which includes the size of the matrix. The model argument must be of type RandomMatrixModel. Numerous sub-types (e.g. WignerModel, WishartModel) are supported generating the appropriate type of random matrix.

Usage

rmatrix(model)

rmatrix(model)

rmatrix(model)

rmatrix(model)

Details

Used to generate a random matrix from various families. The idea is to specify a model, which is then used to generate random realizations and also to compute other properties of the matrix.

See Also

dmatrix

Examples

```
model <- WignerModel(10)
m <- rmatrix(model)

## Not run:
e <- Ensemble(20, model)
hist(max_eigen(e), freq=FALSE)

## End(Not run)
```

---

select *Select a portion of a matrix based on a regular expression of the row and/or column names.*

Description

Extract a subset of a matrix based on regex patterns on either the rownames, the colnames or both. Once this subset has been selected, assignments can be made following standard consistency rules.

Usage

```
select(m, row.pat = NULL, col.pat = NULL, ...)
```

Arguments

m	A matrix from which to select a subset
row.pat	A regular expression to use for rownames
col.pat	A regular expression to use for colnames
...	Additional arguments to pass to grep

Details

Oftentimes it is useful to get at a specific subset of data within a matrix. In large matrices, it can be cumbersome to access specific rows and/or columns using traditional subsetting methods, particularly if it is a complex set that is to be extracted. select provides regex searching on named matrices to access portions of a matrix that satisfy the regex. Note that select will work for data.frames as well.

It is possible to assign values to the selected subset as a means to modify the original matrix. Standard consistency rules must be satisfied for any assignment operations.

Value

A matrix containing all rows and columns that satisfy the patterns given. If no values match, then an empty matrix will be returned.

Examples

```
library(datasets)
select(swiss, "Rive")

select(swiss, col.pat="E", fixed=TRUE)

select(swiss, row.pat='^[A-T]', col.pat="^E")

select(swiss, "Rive", "Ed") <- min(select(swiss, "^[^R]", "Ed"))
```

Index