Package 'RMThreshold'

January 20, 2025

Type Package

Title Signal-Noise Separation in Random Matrices by using Eigenvalue Spectrum Analysis

Version 1.1

Date 2016-06-03

Author Uwe Menzel

Maintainer Uwe Menzel <uwemenzel@gmail.com>

Description An algorithm which can be used to determine an objective threshold for signal-noise separation in large random matrices (correlation matrices, mutual information matrices, network adjacency matrices) is provided. The package makes use of the results of Random Matrix Theory (RMT). The algorithm increments a suppositional threshold monotonically, thereby recording the eigenvalue spacing distribution of the matrix. According to RMT, that distribution undergoes a characteristic change when the threshold properly separates signal from noise. By using the algorithm, the modular structure of a matrix - or of the corresponding network - can be unraveled.

License GPL

LazyLoad yes

Imports Matrix, png

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-23 19:57:40

Contents

RMThreshold-package 2
add.Gaussian.noise 3
create.rand.mat 4
rm.connections 5
rm.denoise.mat 6
rm.discard.zeros 7
rm.ev.density 8
rm.get.threshold 9

rm.matrix.validation 16
rm.show.plots 18
rm.spacing.distribution 19
RMThreshold-internal 20

Index **23**

`RMThreshold-package` *Signal-Noise Separation in Correlation Matrices by using Eigenvalue Spectrum Analysis*

Description

The package provides an algorithm that can be used to determine an objective threshold for signal-noise separation in large random matrices.

Details

The package provides an algorithm which can be used to determine an objective threshold for signal-noise separation in large random matrices (correlation matrices, mutual information matrices, network adjacency matrices). The package makes use of the results of Random Matrix Theory (RMT). The algorithm increments a suppositional threshold monotonically, thereby recording the eigenvalue spacing distribution of the matrix. According to RMT, that distribution undergoes a characteristic change when the threshold properly separates signal from noise. The modular structure of the matrix (or of the corresponding network) can be unraveled if such a threshold is found.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

References

https://en.wikipedia.org/wiki/Random_matrix

Wigner, E. P. , *Characteristic vectors of bordered matrices with infinite dimensions*, Ann. Math. 62, 548-564, 1955.

Mehta, M., *Random Matrices*, 3nd edition. Academic Press, 2004.

Furht, B. and Escalante, A. (eds.), *Handbook of Data Intensive Computing*, Springer Science and Business Media, 2011.

Luo, F. et al., *Constructing gene co-expression networks and predicting functions of unknown genes by random matrix theory.* BMC Bioinformatics, 2007.

Examples

```
## Not run:
  set.seed(777)
  random.mat <- create.rand.mat(size = 1000, distrib = "norm")$rand.matrix
  res <- rm.matrix.validation(random.mat) # ok
  res <- rm.get.threshold(random.mat) # threshold about 3.19
  rm.show.plots(res$comparison.plots)
  cleaned.matrix <- rm.denoise.mat(random.mat, threshold = 3.2)
  cleaned.matrix <- rm.discard.zeros(cleaned.matrix)

## End(Not run)
```

`add.Gaussian.noise` *Add Gaussian noise to a matrix*

Description

The function adds Gaussian (i.e. normally distributed) noise to a matrix.

Usage

```
add.Gaussian.noise(mat, mean = 0, stddev = 1, symm = TRUE)
```

Arguments

`mat`	Input matrix.
`mean`	Mean of the Gaussian noise to be added.
`stddev`	Standard deviation of the Gaussian noise to be added.
`symm`	A logical variable that determines if the matrix is to be symmetrized after adding the noise.

Details

The function uses the `rnorm` function to create the normally distributed noise and adds it to the input matrix. Optionally, the matrix is symmetrized by adding it's transpose and dividing by $\sqrt{2}$.

Value

The input matrix with noise added, optionally symmetrized.

Note

The matrix can not be symmetrized if it is not quadratic.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

References

https://en.wikipedia.org/wiki/Gaussian_noise

See Also

Random generation for the normal distribution: rnorm

Examples

```
## Not run:
N = 500
some.mat = matrix(rep(1, N*N), nrow = N)
some.mat[1:3, 1:10]
res <- rm.matrix.validation(some.mat) # not really a proper matrix for this approach.

## End(Not run)

## It can help to add Gaussian noise to an improper matrix
## Not run:
noisy.matrix <- add.Gaussian.noise(some.mat, mean = 0, stddev = 1, symm = TRUE)
noisy.matrix[1:3, 1:10]
res <- rm.matrix.validation(noisy.matrix) # better!
res <- rm.get.threshold(noisy.matrix) # about 4.3

## End(Not run)
```

`create.rand.mat` *Create a real-valued, symmetric random matrix*

Description

The function creates a real-valued, symmetric random matrix of desired dimension. Two alternatives for the probability distribution of the matrix elements are provided.

Usage

```
create.rand.mat(size = 1000, distrib = c("norm", "unif"), mean = 0, stddev = 1)
```

Arguments

`size`	Dimension of the (quadratic) matrix.
`distrib`	Desired probability distribution of the matrix elements. Can be `norm` or `unif`.
`mean`	Desired mean of the normal distribution. Only active if `distrib` = `'norm'` was chosen.
`stddev`	Desired standard deviation of the normal distribution. Only active if `distrib` = `'norm'` was chosen.

Details

The function creates a real-valued, symmetrical random matrix of desired dimension. Two alternatives for the probability distribution of the matrix elements are provided: normal and uniform. If `distrib = 'norm'`, the mean and the standard deviation can additionally be chosen. If `distrib = 'unif'`, the matrix elements will be uniformly distributed in the interval (-1,1).

Value

A list containing the following components:

`mean.diag`	The mean of the diagonal elements of the matrix.
`stddev.diag`	The standard deviation of the diagonal elements of the matrix.
`mean.triangle`	The mean of the upper triangle of the matrix (diagonal excluded).
`stddev.triangle`	The standard deviation of the upper triangle of the matrix (diagonal excluded).
`rand.matrix`	The random matrix created.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

Examples

```
set.seed(777)
random.matrix <- create.rand.mat(size = 1000, distrib = "norm")$rand.matrix
dim(random.matrix)
```

`rm.connections` *Create ordered list of largest matrix elements*

Description

The function creates a data frame which is sorted according to the (absolute) magnitude of the matrix elements.

Usage

```
rm.connections(mat, nr.list = 30, abs.val = TRUE, fn = NULL)
```

Arguments

`mat`	Input matrix.
`nr.list`	Number of matrix elements to show.
`abs.val`	Logical variable determining if absolute values should be used for sorting.
`fn`	A file name. If not `NULL`, the data frame is saved to that file.

Details

This function can for instance be useful if pairs of samples with the largest correlation/mutual information are to be identified. By default, the matrix elements are sorted according to their absolute values. The list will not be saved if no filename is invoked, otherwise it will be saved to a tab-separated text file.

Value

A data frame containing the values of the largest `nr.list` matrix elements, together with the respective row- and column numbers. If present, the referring row- and column names are also included.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

Examples

```
set.seed(777)
random.mat <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
dim(random.mat)

## After identification of a proper threshold:
cleaned.matrix <- rm.denoise.mat(random.mat, threshold = 3.2, keep.diag = TRUE)
cl2.matrix = rm.discard.zeros(cleaned.matrix)
df = rm.connections(cl2.matrix)
```

`rm.denoise.mat` *Remove noise from a random matrix by applying a threshold*

Description

Matrix elements with an absolute value below the given threshold are set to zero.

Usage

```
rm.denoise.mat(mat, threshold, keep.diag = TRUE)
```

Arguments

`mat`	The noisy input matrix.
`threshold`	Numerical value of the threshold.
`keep.diag`	A logical variable that determines if the diagonal of the matrix is thresholded or not. The default is `keep.diag = T`. In that case, diagonal matrix elements are not touched.

Details

The function outputs the number of non-zero matrix elements before and after thresholding.

Value

The thresholded matrix.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

See Also

Estimate an objective threshold for signal-noise separation in random matrices: `rm.matrix.validation`

Examples

```
set.seed(777)
random.matrix <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
dim(random.matrix)


## After identification of a proper candidate threshold:
cleaned.matrix <- rm.denoise.mat(random.matrix, threshold = 3.2, keep.diag = TRUE)
```

`rm.discard.zeros` *Discard rows and columns from a matrix that exclusively contain zero-valued off-diagonal matrix elements*

Description

The function removes those rows and columns from an input matrix that exclusively contain zero-valued off-diagonal elements.

Usage

```
rm.discard.zeros(mat, tol = 0, silent = FALSE)
```

Arguments

`mat`	Input matrix (typically after using `rm.denoise.mat`)
`tol`	A (small) real number specifying a thresholf for removal of matrix elements (see ’Details’).
`silent`	A logical variable that determines if the number of removed rows and columns is printed by the function or not.

Details

The diagonal of the matrix is not included when counting the zeros in a row/column, i.e. a row/column is actually removed if the diagonal element is the only non-zero element in that row/column. The tolerance `tol` specifies a threshold. Matrix elements below this threshold will be treated as if they were zero.

Value

A matrix with zero-valued rows/colums removed.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

See Also

Remove noise from a random matrix by applying a threshold: `rm.denoise.mat`

Examples

```
set.seed(777)
random.matrix <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
dim(random.matrix)

## After identification of a proper threshold:
cleaned.matrix <- rm.denoise.mat(random.matrix, threshold = 3.2, keep.diag = TRUE)
cl2.matrix = rm.discard.zeros(cleaned.matrix)
```

`rm.ev.density` *Create a density plot and a histogram of the eigenvalue distribution*

Description

The function creates a density plot of the empirical distribution of the eigenvalues, combined with a histogram. Optionally, a curve illustrating the Wigner semi-circle can be added. The plot can be saved or shown in a plot window. Marks on the x-axis can be added optionally.

Usage

```
 rm.ev.density(eigenvalues, nr.breaks = 51, min.bw = 0.01, wigner = TRUE,
mark.on.x = NULL, title = "Eigenvalue density distribution",
pop.up = TRUE, fn = NULL)
```

Arguments

`eigenvalues`	A numeric vector containing the eigenvalues.
`nr.breaks`	Number of bins used in the histogram.
`min.bw`	Minimum bandwidth for the calculation of the density curve. If the automatically calculated bandwidth gets too low, it is replaced by this value. That prevents the density curve from being too cliffy.
`wigner`	A logical variable that determines if the Wigner semi-circle is to be added to the plot.

`mark.on.x`	A numeric vector or NULL. If not NULL, marks will be added on the x-axis at the positions given by the vector.
`title`	String containing the title of the plot.
`pop.up`	A logical variable that determines if the plot is to be shown in a plot window.
`fn`	A string determining the filename for storage. Must have extension 'png' or 'pdf'.

Value

The name of the plot filename chosen, or NULL.

Note

This function plots the density of the eigenvalues. For illustration of their spacings, use `rm.spacing.distribution`.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

See Also

Plotting the eigenvalue spacing distribution: `rm.spacing.distribution`

Examples

```
## Plot histogram of the spacings of the unfolded eigenvalues of a random matrix:
set.seed(777)
random.matrix <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
res <- rm.ev.unfold(random.matrix)
rm.ev.density(res$eigenvalues, wigner = TRUE)
```

`rm.get.threshold` *Estimate an objective threshold for signal-noise separation in random matrices*

Description

This is the main function of the package. A suppositional threshold is incremented monotonically, thereby recording the eigenvalue spacing distribution (nearest neighbor spacing distribution, NNSD) of the thresholded matrix. According to Random Matrix Theory, that distribution undergoes a characteristic change when the threshold properly separates signal from noise. By subsequent removal of the below-threshold matrix elements, the modular structure of the matrix - or of a network characterized by this matrix - can be unraveled. The function works for real symmetric matrices.

Usage

```
rm.get.threshold(rand.mat, nr.thresholds = 51,
   unfold.method = "gaussian", bandwidth = "nrd0", nr.fit.points = 51,
   dist.method = "LL", nr.breaks = 51, discard.outliers = TRUE,
  discard.zeros = FALSE, min.mat.dim = 40, max.ev.spacing = 3, interval = NULL,
   interactive = TRUE, smooth.par = 0.5, plot.comp = TRUE,
   save.fit = FALSE, plot.spacing = FALSE, wait.seconds = 0)
```

Arguments

`rand.mat`	A random, real-valued, symmetric matrix.
`nr.thresholds`	Number of equidistant thresholds being probed and recorded.
`unfold.method`	A string variable that determines which type of unfolding algorithm is used. Must be one of 'gaussian' (Gaussian kernel density) or 'spline' (cubic spline interpolation on the cumulative distribution function).
`bandwidth`	Bandwidth used to calculate the Gaussian kernel density. Only active if `unfold.method = 'gaussian'` is used. See description of the `density` function.
`dist.method`	A string variable that determines which type of distance estimation to the limiting distributions is used. Must be one of 'LL' (Log Likelihood) or 'KLD' (Kullback-Leibler Distance).
`nr.fit.points`	Number of supporting points used for the cubic spline to the empirical cumulative distribution function. Only active if `unfold.method = 'spline'` is used.
`nr.breaks`	Number of bins used in the analysis to subdivide the empirical eigenvalue spacing distribution.
`smooth.par`	Parameter controlling the degree of smoothing in the loess regression curve presented in the final plot (distance vs. threshold).
`discard.outliers`	A logical variable that determines if outliers are to be discarded from the spectrum of eigenvalues (see 'Details').
`discard.zeros`	A logical variable specifying if rows and columns exclusively containing zeros outside the main diagonal are to be removed from the matrix after applying each threshold (see 'Details').
`min.mat.dim`	By thresholding a matrix, rows and columns exclusively containing zeros in the off-diagonal matrix elements likely emerge. The parameter `min.mat.dim` determines the minimum number of non-zero rows and columns of the probed matrix during the thresholding. The thresholding loop is stopped if the number of non-zero rows and columns is getting below `min.mat.dim`.
`max.ev.spacing`	A number determining the maximum eigenvalue spacing that is considered when plotting and analyzing the empirical eigenvalue spacing distribution (see 'Details').
`interval`	Interval of thresholds that is searched through. A numeric vector with two components (minimum and maximum threshold). The default is `interval = NULL` which means that the search interval ranges from the minimum absolute value of all matrix elements to the maximum absolute value of all matrix elements.

`interactive`	A logical variable that determines if the user wants to choose the candidate thresholds interactively, by mouse clicks.
`plot.comp`	A logical variable that determines if the empirical distribution of the eigenvalue spacings is displayed in a plot window during function execution.
`save.fit`	A logical variable that determines if a plot of the spline fit to the empirical cumulative distribution function is saved for each threshold. Can be used to check if fitting works well.
`plot.spacing`	A logical variable that determines if a scatterplot showing the eigenvalue spacing is saved at each suppositional threshold. Can be used to check if unfolding of the eigenvalue spacing works correctly (see 'Details').
`wait.seconds`	A numerical variable that, if set to non-zero values, enables viewing the plots with the eigenvalue spacing distribution during function execution for a predetermined duration. Useful on fast computers. Setting the variable to a specific number will cause the function to show the actual eigenvalue spacing distribution at least that number of seconds.

Details

The function `rm.get.threshold` is the main function of the package. It takes a random matrix as input and applies a sequence of suppositional thresholds on it by setting all matrix elements to zero whose absolute value is below the actual threshold. The eigenvalues of the modified matrix are calculated at each threshold by using the `eigen` function of the R `base` package. The eigenvalue spectrum is then unfolded, i.e. it is calibrated in such a way that the average global spacing between the eigenvalues is constant over the whole spectrum. The latter can be tracked by setting `plot.spacing = T`. Two methods are provided for unfolding: one method is based on calculation of the Gaussian kernel density of the eigenvalue spectrum; another method is based on fitting a cubic spline function to the cumulative empirical eigenvalue distribution. The latter is determined by the parameter `unfold.method`. See the references for details of the unfolding procedure. For each threshold, a distance between the empirical eigenvalue spacing distribution (NNSD) and both limiting distributions (Wigner-Dyson distribution and Exponential distribution, respectively) is estimated. Two methods of distance computation are implemented: a method based on computation of the log likelihood of the empirical eigenvalue spacing presupposing each of the limiting distributions, and a method based on calculation of the Kullback-Leibler divergence between empirical eigenvalue spacing distribution and these limiting distributions. If the assumed modular structure of the matrix is completely covered by noise, the empirical eigenvalue spacing distribution is close to the Wigner-Dyson distribution, a curve that approaches zero for small eigenvalue spacings. In the opposite case, when the modular structure of the matrix is prevailing, the empirical eigenvalue spacing distribution comes closer to an Exponential distribution (which represents the distribution of the intervals between two consecutive events in a Poisson process). If the matrix possesses a modular structure (hidden by noise), we expect that the NNSD changes gradually from the Wigner-Dyson case to the Exponential case when the threshold is increased stepwise. This change is monitored in a plot window if the `plot.comp` variable is left at it's default (`plot.comp = TRUE`). Two additional parameters are critical for proper functioning of the algorithm. For some types of input matrices, it might be necessary to remove the outliers of the eigenvalue distribution, in order to correctly investigate the bulk of the eigenvalue spectrum. This is achieved by the setting `discard.outliers = TRUE`, which is the default setting. In some other cases, it might be useful to retain the outliers during analysis. Another critical parameter is `discard.zeros`. If set to TRUE, rows and columns exclusively containing zeros outside the main diagonal are removed from the matrix at each threshold.

This causes the matrix to shrink during the loop over thresholds. Setting `discard.zeros = TRUE` can be especially useful when the NNSD piles up at the left tail of the histogram shown during program execution. For very fast computers, the argument `wait.seconds` can be set to a non-zero value, in order to enable the user to follow that change of the NNSD visually. The distance between NNSD and the limiting distributions is not calculated over the whole range of eigenvalue spacings but over the interval `(0, max.ev.spacing)`. At a spacing of zero, the difference between the Wigner-Dyson distribution and the Exponential distribution is mostly pronounced. The maximum spacing considered in the distance calculation is determined by the parameter `max.ev.spacing`. This parameter should not be lower than $\sqrt(2/\pi)$, where the peak of the Wigner-Dyson distribution lies. On the other hand, is does not make sense to choose too high values for `max.ev.spacing`, because both the Wigner-Dyson and the Exponential distribution assume rather low values in the right tail (which might cause numerical errors). If the algorithm works well, a relatively sharp transition from the Wigner-Dyson case to the Exponential case should become apparent. The latter is (hopefully) confirmed in a plot which is shown after completion of the loop over the sequence of suppositional thresholds. This plot shows the calculated distance between the NNSD and both limiting distributions versus applied threshold. The user can interactively choose candidate thresholds by clickung with the left mouse button on the points of the red-coloured curve. The selection is terminated by a right mouse-click somewhere on the plot. Likewise, candidate thresholds can be chosen in a plot showing the p-values for a Kolmogorov-Smirnov test (testing for exponentiality), and in a plot showing the Sum of Squared Errors between the empirical NNSD and the Exponential distribution versus threshold. The hereby chosen candidate thresholds are returned by the function. The analysis can now be refined by narrowing down the search interval for the thresholds by specifying the `interval` parameter.

Value

A list containing the following entries:

`unfold.method`	The method that was used for eigenvalue unfolding. Either 'gaussian' or 'spline'.
`dist.method`	The method that was chosen to estimate the distance to the limiting distributions. Either 'LL' (Log Likelihood) or 'KLD' (Kullback-Leibler Divergence).
`tested.thresholds`	A vector containing the probed thresholds.
`dist.Wigner`	A vector containing the numerical values of the estimated distance between empirical eigenvalue spacing distribution and Wigner-Dyson distribution.
`dist.Expon`	A vector containing the numerical values of the estimated distance between empirical eigenvalue spacing distribution and Exponential distribution.
`nr.zeros`	A vector containing the number of zero-valued matrix elements for each threshold.
`nr.uniq.ev`	An integer vector indicating the number of unique eigenvalues at each threshold (when degenerated eigenvalues are counted as one).
`max.ev.mult`	An integer vector indicating the maximum eigenvalue multiplicity (degeneracy) at each threshold.
`nr.spacings`	An integer vector containing the number of eigenvalue spacings at each threshold. This number is smaller than the matrix dimension if eigenvalues are degenerated.

`nr.small.spacings`	An integer vector containing the number of small spacings (< `max.ev.spacing/1000`) for each probed threshold.
`perc.small.spacings`	A vector containing the percentage of small spacings for each probed threshold.
`eff.dimension`	An integer number specifying the 'effective dimension' of the matrix after thresholding, i.e. the number of non-zero rows and columns.
`comparison.plots`	A character vector containing the names of the plots comparing the empirical eigenvalue spacing distribution with both limiting distributions (only if `plot.comp = T`).
`rm.dimension`	An integer vector indicating the dimension of the matrix after each thresholding (only if `discard.zeros = TRUE`).
`nr.outliers.removed`	An integer vector containing the number of outliers in the eigenvalue spectrum removed at each threshold. Only if `discard.outliers = TRUE`.
`p.ks`	A vector containing the p-values for the Kolmogorov-Smirnov test at each probed threshold.
`sse.exp`	A vector containing the Sum of Squared Errors (SSE) between observed NNSD and Exponential distribution for each probed threshold.
`number.zeros.plot`	A character string with the name of the plot depicting the number of zero-valued matrix elements versus threshold.
`number.uniq.ev.plot`	The name of the plot showing the number of unique eigenvalues vs. threshold.
`max.ev.mult.plot`	The name of the plot showing the maximum eigenvalue multiplicity vs. threshold.
`mat.dimension.plot`	The name of the plot showing the matrix dimension after each thresholding step (only if `discard.zeros = TRUE`).
`num.ev.spacings.plot`	The name of the plot showing the number of eigenvalue spacings vs. threshold.
`distance.plot`	The name of the (main) plot showing the distance of the empirical eigenvalue spacing distribution to the limiting distributions at each threshold.
`cumfit.plots`	A character vector containing the names of the plots showing the spline-function fitting of the cumulative eigenvalue spacing distribution (only if `save.fit = TRUE`).
`space.plots`	A character vector containing the names of the scatter plots with the eigenvalue spacing (only if `plot.spacing = TRUE`).
`chosen.thresholds`	A vector containing the potential thresholds chosen by the user from the distance plot.
`p.ks.plot`	The name of the plot showing the p-values for the Kolmogorov-Smirnow test versus probed thresholds.

p.ks.chosen	A vector containing the candidate thresholds chosen by the user based on the Kolmogorov-Smirnow test.
sse.plot	The name of the plot showing the SSE between observed NNSD and Exponential distribution versus probed thresholds.
sse.chosen	A vector containing the candidate thresholds chosen by the user based on the SSE test.

Note

It is recommended to check the input matrix using `rm.matrix.validation` before running this function. If the histogram of empirical eigenvalue spacings piles up at zero, one should set `discard.zeros = TRUE`

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

References

https://en.wikipedia.org/wiki/Random_matrix

Wigner, E. P. , *Characteristic vectors of bordered matrices with infinite dimensions*, Ann. Math. 62, 548-564, 1955.

Mehta, M., *Random Matrices*, 3nd edition. Academic Press, 2004.

Furht, B. and Escalante, A. (eds.), *Handbook of Data Intensive Computing*, Springer Science and Business Media, 2011.

Luo, F. et al., *Constructing gene co-expression networks and predicting functions of unknown genes by random matrix theory.* BMC Bioinformatics, 2007.

See Also

Creating a random matrix: create.rand.mat

Examples

```
## Run with pre-defined random matrix:
set.seed(777)
random.matrix <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
dim(random.matrix) # 1000 1000

## Not run:
res <- rm.get.threshold(random.matrix) # threshold might be 3.21
str(res) # List of 26
rm.show.plots(res$comparison.plots)   # watch sequence of plots once more

## End(Not run)

## Try other parameters:
## Not run:
res <- rm.get.threshold(random.matrix, unfold.method = "spline")
res <- rm.get.threshold(random.matrix, dist.method = "KLD")
res <- rm.get.threshold(random.matrix, discard.outliers = FALSE) # might cause problems
res <- rm.get.threshold(random.matrix, wait.seconds = 2)    # slow down
res <- rm.get.threshold(random.matrix, discard.zeros = TRUE)
res <- rm.get.threshold(random.matrix, discard.zeros = TRUE, dist.method = "KLD")

## End(Not run)

## Refine analysis by choosing narrower threshold range
## Not run:
res <- rm.get.threshold(random.matrix, interval = c(2.5, 3.5))

## End(Not run)

## Apply the identified threshold to the matrix
cleaned.matrix <- rm.denoise.mat(random.matrix, threshold = 3.21)
cleaned.matrix <- rm.discard.zeros(cleaned.matrix)
dim(cleaned.matrix) # smaller

## Find the clusters in the thresholded matrix:
## Not run:
  library(igraph)
  g  <- graph.adjacency(cleaned.matrix, mode = "undirected")
  clusters(g)

## End(Not run)


## Not run:

  ## Create modular matrix and validate:
  matlist = list()
  for (i in 1:4) matlist[[i]] = get.adjacency(erdos.renyi.game(250, 0.1))
  mat <- bdiag(matlist) # create block-diagonal matrix
  rm.matrix.validation(as.matrix(mat)) # Exponential case, modular matrix

  ## Add noise:
  mat1 = add.Gaussian.noise(as.matrix(mat), mean = 0, stddev = 0.1)

  ## Find threshold, denoise, reconstruct the modules:
  res <- rm.get.threshold(mat1) # threshold possibly about 0.46
  # a smaller interval had been ok as well:
  res <- rm.get.threshold(mat1, interval = c(0, 0.8))
  cleaned <- rm.denoise.mat(mat1, 0.5)
  matr <- cleaned != 0
  g  <- graph.adjacency(matr, mode = "undirected")
  clusters(g) # 4 clusters reconstructed


## End(Not run)
```

`rm.matrix.validation` *Validate input matrix prior to threshold computation*

Description

The function checks if the input matrix is well-conditioned for the algorithm used by `RMThreshold`. The matrix must be real-valued, symmetric, and large. Rank and sparseness of the matrix are calculated. Diagnostic plots are created.

Usage

```
 rm.matrix.validation(rand.mat, unfold.method = "gaussian",
bandwidth = "nrd0", nr.fit.points = 51, discard.outliers = TRUE)
```

Arguments

`rand.mat`	A random, real-valued, symmetric input matrix.
`unfold.method`	A string variable that determines which type of unfolding algorithm is used. Must be one of 'gaussian' (Gaussian kernel density) or 'spline' (cubic spline interpolation on the cumulative distribution function).
`bandwidth`	Bandwidth used to calculate the Gaussian kernel density. Only active if `unfold.method = 'gaussian'` is used. See the description of the `density` function.
`nr.fit.points`	Number of evenly spaced supporting points used for the cubic spline to the empirical cumulative distribution function.
`discard.outliers`	A logical variable that determines if outliers are to be discarded from the spectrum of eigenvalues.

Details

The input matrix must be real-valued and symmetric (a correlation or mutual information matrix self-evidently is). The matrix must not be too sparse (if so, you are probably done without thresholding). The rank of the matrix must not be too low in order to obtain a sufficient number of non-zero eigenvalues. Furthermore, the matrix must not be too small because Random Matrix Theory applies for large (theoretically infinite) matrices only. The function creates a diagnostic plot, showing the empirical eigenvalue distribution and the distribution of the spacings between them. The eigenvalue distribution of the input matrix should approximately resemble the Wigner semi-circle, while the spacings should resemble the Wigner-Dyson distribution (Wigner surmise).

Value

A list containing the following entries:

`sparseness`	The sparseness of the input matrix.
`rank`	The rank of the input matrix.

`validation.plot`	The name of the valdation plot.
`unfold.plot`	The name of the plot which can be used to check if eigenvalue unfolding worked correctly.
`nr.outliers.removed`	The number of eigenvalue outliers that have been removed. Only if `discard.outliers = TRUE` was used.

Note

It is highly recommended to check the input matrix using this function.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

References

`https://en.wikipedia.org/wiki/Random_matrix`

Wigner, E. P. , *Characteristic vectors of bordered matrices with infinite dimensions*, Ann. Math. 62, 548-564, 1955.

Mehta, M., *Random Matrices*, 3nd edition. Academic Press, 2004.

Furht, B. and Escalante, A. (eds.), *Handbook of Data Intensive Computing*, Springer Science and Business Media, 2011.

See Also

Creating a random matrix: `create.rand.mat`

Examples

```
## Run with self-created  random matrix:
set.seed(777)
random.matrix <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
dim(random.matrix) # 1000 1000   should be big enough

## Not run:
res <- rm.matrix.validation(random.matrix)
res <- rm.matrix.validation(random.matrix, discard.outliers = FALSE)
res <- rm.matrix.validation(random.matrix, unfold.method = "spline")
res <- rm.matrix.validation(random.matrix, unfold.method = "spline", discard.outliers = FALSE)

## End(Not run)

## Not run:
  library(igraph)

  ## Create noisy matrix and validate:
  g <- erdos.renyi.game(1000, 0.1)
  adj = as.matrix(get.adjacency(g))
  rm.matrix.validation(adj) # Wigner-Dyson case, unstructured matrix, noise

  ## Create modular (block-diagonal) matrix and validate:
  matlist = list()
  for (i in 1:4) matlist[[i]] = get.adjacency(erdos.renyi.game(250, 0.1))
  mat <- bdiag(matlist) # block-diagonal matrix
  rm.matrix.validation(as.matrix(mat)) # Exponential case, modular matrix


## End(Not run)
```

`rm.show.plots` *Display a sequence of plots on screen*

Description

The function displays a sequence of plots saved beforehand (e.g. by `rm.get.threshold`).

Usage

```
rm.show.plots(plotnames)
```

Arguments

`plotnames`	A character string or -vector containing the names of the plots to be viewed.

Details

Can be useful when the user wants to inspect the sequence of empirical eigenvalue spacing distributions (repeatedly).

Value

No return values are being created.

Note

Might cause problems in some environments, only tested on Linux.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

Examples

```
## Not run:
set.seed(777)
random.matrix <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
dim(random.matrix)
res <- rm.get.threshold(random.matrix)
rm.show.plots(res$comparison.plots) # watch sequence of plots once more

## End(Not run)
```

rm.spacing.distribution

Plot the empirical distribution of the eigenvalue spacings

Description

A histogram of the empirical distribution of the eigenvalue spacings is plotted. Optionally, curves illustrating the Wigner surmise and/or the Exponential distribution are added.

Usage

```
 rm.spacing.distribution(ev.spacing, nr.breaks = 51,
wigner = TRUE, expo = TRUE,
title = "Eigenvalue spacing distribution (NNSD)",
threshold = NA, dist.Wigner = NA,
dist.Expon = NA, pop.up = TRUE, fn = NULL)
```

Arguments

`ev.spacing`	A numeric vector containing the spacings of the eigenvalues.
`nr.breaks`	Number of bins used in the histogram.
`wigner`	A logical variable that determines if the Wigner-Dyson distribution (Wigner surmise) is to be added to a plot.
`expo`	A logical variable that determines if the Exponential distribution is to be added to the plot.
`title`	String containing the title of the plot.
`threshold`	If not NA, this value will be displayed in the plot, labeled 'threshold'.
`dist.Wigner`	If not NA, this value will be added to the plot, with a text indicating that it is the numerical value of the Kullback-Leibler distance between the empirical eigenvalue spacing distribution function and the Wigner-Dyson distribution function.
`dist.Expon`	If not NA, this value will be added to the plot, with a text indicating that it is the numerical value of the Kullback-Leibler distance between the empirical eigenvalue spacing distribution function and the Exponential distribution.
`pop.up`	A logical variable that determines if the plot is shown in a plot window.
`fn`	A string determining the filename for storage. Must have extension 'png' or 'pdf'.

Value

The name of the plot filename chosen, or NULL.

Author(s)

Uwe Menzel <uwemenzel@gmail.com>

See Also

Plotting the eigenvalue distribution: rm.ev.density

Examples

```
## Plot histogram of the spacings of the unfolded eigenvalues of a random matrix:
set.seed(777)
random.matrix <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
res <- rm.ev.unfold(random.matrix)
rm.spacing.distribution(res$ev.spacing)
```

RMThreshold-internal *Internal functions for the RMThreshold package*

Description

Internal functions for the RMThreshold package

Usage

```
kb.distance(histo)
rm.exp.distrib(x)
kld(observed, expected, plot)
rm.get.file.extension(plotnames)
rm.ev.unfold(rand.mat, unfold.method, bandwidth, nr.fit.points,
discard.outliers, fn, pop.up, silent)
rm.spacing.scatter(ev.spacing, title, pop.up, fn)
rm.trapez.int(x, y)
rm.reorder.ev(eigenvalues, eigenvec)
rm.get.sparseness(mat)
wigner.surmise(x)
wigner.semi.circle(x)
rm.get.distance(ev.spacing, dist.method, nr.breaks)
rm.likelihood.plot(thresholds, log.le, log.lw, smooth.par, fn, interactive)
rm.distance.plot(thresholds, dist.Expon, dist.Wigner, smooth.par, fn, interactive)
rm.unfold.gauss(eigenvalues, bandwidth, fn, pop.up, silent)
rm.unfold.spline(eigenvalues, nr.fit.points, fn, pop.up)
rm.sse(ev.spacing, bandwidth, nr.points, N)
rm.show.test(thresholds, p.values, main, fn, interactive)
rm.sse.plot(thresholds, sse.values, main, fn, interactive)
```

Arguments

`nr.fit.points`	Number of supporting points used for the cubic spline to the empirical cumulative distribution function.
`discard.outliers`	A logical variable that determines if outliers are discarded from the spectrum of eigenvalues.
`silent`	A logical variable that decides if a function outputs runtime messages or not.
`histo`	An R object of class 'histogram'. Output of the `hist` function.
`x`	A real-valued number.
`eigenvalues`	A numerical vector containing the eigenvalues of a matrix.
`wigner`	A logical variable that determines if the Wigner semi-circle or the Wigner surmise is added to a plot.
`title`	String variable containing the title of a plot.
`pop.up`	A logical variable that determines if a plot window is supposed tp pop up during function execution.
`fn`	A filename.
`observed`	A numerical vector with the observed frequency of eigenvalues in each bin of the distribution function.
`expected`	A numerical vector with the expected frequency of eigenvalues in each bin. The expected values refer to a limiting distribution, either Wigner-Dyson or Exponential distribution.
`plot`	A logical variable that determines if a plot should be created.
`plotnames`	A string or character vector of filenames for plots to be viewed.
`rand.mat`	A symmetric, real-valued random matrix.
`fit`	Name of the plot showing the cubic spline fit to the cumulative distribution. No plot will be made if `fit` = `NULL`
`ev.spacing`	A vector with the normalized eigenvalue spacings of a random matrix.
`y`	A numerical vector defining the y-values of a function.
`eigenvec`	A matrix containing the eigenvectors of a matrix (in columns).
`mat`	A real-valued matrix.
`nr.breaks`	Number of bins used in the histogram to subdivide the empirical eigenvalue spacing distribution.
`log.le`	Log likelihood of the observed eigenvalue distances when an exponential distribution is assumed.
`log.lw`	Log likelihood of the observed eigenvalue distances when a Wigner-Dyson distribution is assumed.
`smooth.par`	Parameter controlling the degree of smoothing in the loess regression curve presented in the final plot (distance vs. threshold).
`interactive`	A logical variable that determines if thresholds can be chosen by mouse clicks in the final plot (distance vs. threshold).

`bandwidth`	Bandwidth used to calculate the Gaussian kernel density. See description of the `density` function.
`unfold.method`	A string that decides which method is used for eigenvalue unfolding. One of 'gaussian' or 'spline'.
`dist.method`	A string that determines which method is used to estimate the distance to the limiting distributions. One of 'LL' (Log Likelihood) or 'KLD' (Kullback-Leibler Distance).
`thresholds`	A numerical vector containing the values of the thresholds probed in the main function (`rm.get.threshold`).
`dist.Expon`	A numerical vector containing the estimated distances to the Exponential distribution calculated in the main function (`rm.get.threshold`).
`dist.Wigner`	A numerical vector containing the estimated distances to the Exponential distribution calculated in the main function (`rm.get.threshold`).
`p.values`	A numerical vector containing the p-values for a significance test.
`sse.values`	A numerical vector containing the Sum of Squared Errors between observed NNSD and Exponential function.
`nr.points`	Number of supporting points used to approximate the density function.
`N`	Number of sections used to calculate the Sum of Squared Errors (SSE).
`main`	String variable containing the title of a plot.

Details

These functions are not intended to be called by the user.

Index

∗ **Correlation matrix**
rm.get.threshold, 9
rm.matrix.validation, 16
∗ **Gaussian noise**
add.Gaussian.noise, 3
∗ **Mutual information**
rm.matrix.validation, 16
∗ **Random Matrix Theory**
add.Gaussian.noise, 3
rm.get.threshold, 9
rm.matrix.validation, 16
RMThreshold-package, 2
∗ **Random matrix**
create.rand.mat, 4
∗ **Signal noise separation**
rm.get.threshold, 9
rm.matrix.validation, 16
∗ **models**
add.Gaussian.noise, 3
create.rand.mat, 4
rm.connections, 5
rm.get.threshold, 9
rm.matrix.validation, 16
RMThreshold-package, 2

add.Gaussian.noise, 3

create.rand.mat, 4, *14*, *17*

eigen, *11*

kb.distance *(*RMThreshold-internal*)*, 20
kld *(*RMThreshold-internal*)*, 20

rm.connections, 5
rm.denoise.mat, 6, *7*, *8*
rm.discard.zeros, 7
rm.distance.plot *(*RMThreshold-internal*)*, 20
rm.ev.density, 8, *20*
rm.ev.unfold *(*RMThreshold-internal*)*, 20
rm.exp.distrib *(*RMThreshold-internal*)*, 20
rm.get.distance *(*RMThreshold-internal*)*, 20
rm.get.file.extension *(*RMThreshold-internal*)*, 20
rm.get.sparseness *(*RMThreshold-internal*)*, 20
rm.get.threshold, 9, *18*
rm.likelihood.plot *(*RMThreshold-internal*)*, 20
rm.matrix.validation, *7*, 16
rm.reorder.ev *(*RMThreshold-internal*)*, 20
rm.show.plots, 18
rm.show.test *(*RMThreshold-internal*)*, 20
rm.spacing.distribution, *9*, 19
rm.spacing.scatter *(*RMThreshold-internal*)*, 20
rm.sse *(*RMThreshold-internal*)*, 20
rm.trapez.int *(*RMThreshold-internal*)*, 20
rm.unfold.gauss *(*RMThreshold-internal*)*, 20
rm.unfold.spline *(*RMThreshold-internal*)*, 20
RMThreshold *(*RMThreshold-package*)*, 2
RMThreshold-internal, 20
RMThreshold-package, 2
rnorm, *4*

wigner.semi.circle *(*RMThreshold-internal*)*, 20
wigner.surmise *(*RMThreshold-internal*)*, 20