Package 'MetStaT'

July 2, 2014

Type Package

Title Statistical metabolomics tools

Version 1.0

Date 2012-12-10

Author Tim Dorscheidt

Maintainer Gooitzen Zwanenburg <g.zwanenburg@uva.nl>

Depends MASS, abind, pls

Description A diverse collection of metabolomics related statistical tools.

License Apache License (== 2.0)

NeedsCompilation no

Repository CRAN

Date/Publication 2013-11-18 10:09:08

R topics documented:

`ASCA.Calculate` *ASCA method (ANOVA-simultaneous component analysis)*

Description

ASCA does PCA on the averages of the treatment levels for an experimental design.

Usage

```
ASCA.Calculate(data, levels, equation.elements = "", scaling = FALSE,
only.means.matrix = FALSE, use.previous.asca = NULL)
```

Arguments

data	numeric data matrix that is to be analyzed with ASCA. Variables are represented by columns, observations by rows.
levels	numeric matrix describing the experimental design. Each factor is represented by a column. The elements of the columns give the treatment level the row belongs to.
equation.elements	a string value indicating which factors and interactions are to be part of the ASCA analysis. A factor is defined by writing its column number in the 'levels' matrix (eg. "1") and an interaction by combining the interacting factors' column numbers from the 'levels' matrix (eg. "123"). Multiple factors/interactions are seperated with comma's (eg. "1,2,12").
scaling	boolean; determines autoscaling of the data. Default is FALSE, data is not autoscaled.
only.means.matrix	boolean; if TRUE, only the matrix with averages for the treatment levels is returned. Default is FALSE. (Note: this is generally only used for performance optimization during many runs, such as permutation testing)
use.previous.asca	previous ASCA results can be used for some calculations that are independent of the data values. Useful for permutation testing. Default is NULL, do not use previous results.

Details

ASCA decomposes a data matrix X in effect matrices A, B, ... containing the level averages for each treatment level, interaction matrices U, V, ... between two or more factors and a residual matrix E with data that is not represented by the model: X = A + B + ... + U + V + ... + E. Principal component analysis is then used as a variable reduction method on each of the effect and interaction matrices. Scores, loadings and singular values for each factor and each interaction are returned.

Value

PerformAsca returns a list with the following components:

data	original data matrix.
levels	original matrix with treatment levels.
svd	an SVD performed on all elements in "equation.elements" using this package's custom PCA.Calculate.
remainder	residual matrix.
ee.names	string array containing the factors and interactions that were used in this ASCA (i.e. "equation.elements")

All remaining list elements (eg. "1", "12") correspond to a separate factor or interaction. Each is a list containing the ASCA results with the following elements:

`factors.evaluated`
: numerical array of the relevant factor (or multiple factors for an interactions)

`level.combinations`
: contains all information on which combinations of factor-levels occur in the data (row.patterns) and, for each combination, lists the row-indices where it occurs

`means.matrix`
: the matrix with means of the treatment levels

`reduced.matrix`
: values left after the (already reduced by previously calculated factors/interactions). The data matrix is reduced by this factor/interaction's means matrix

`svd`
: a SVD performed on this factor/interaction's means matrix using PCA.Calculate

Note

ASCA.Calculate uses the custom method "PCA.Calculate" (part of MetStaT package) for the principal component analysis.

Author(s)

Tim Dorscheidt, Gooitzen Zwanenburg

References

Smilde AK, Jansen JJ, Hoefsloot HCJ, Lamers R JAN, van der Greef J, Timmerman ME. *ANOVA simultaneous component analysis (ASCA): a new tool for analyzing designed metabolomics data*. Bioinformatics **21**, (2005), p. 3043 - 3048.

Gooitzen Zwanenburg, Huub C.J. Hoefsloot, Johan A. Westerhuis, Jeroen J. Jansen and Age K. Smilde, *ANOVA principal component analysis and ANOVA simultaneous component analysis: a comparison*. J Chemometrics, **25**, (2011), p. 561 - 567

See Also

`ASCA.DoPermutationTest`, `PCA.Calculate`

Examples

```
## use the data matrix, 'ASCAX', and an experimental design matrix, 'ASCAF'
data(ASCAdata)
ASCA <- ASCA.Calculate(ASCAX, ASCAF, equation.elements = "1,2,12", scaling = FALSE)

## plot the results
ASCA.Plot(ASCA)
```

ASCA.DoPermutationTest *Permutation test for ASCA*

Description

Does a permutation test on the results from ASCA.Calculate by repeating the ASCA analysis many times with permutated samples.

Usage

```
ASCA.DoPermutationTest(asca, perm = 1000)
```

Arguments

asca	a previously done ASCA analysis should be supplied.
perm	the number of permutations to be performed.

Details

The significance of treatment effects or of interactions between treatment effects can be evaluated by considering the p-values that are returned by ASCA.DoPermutationTest. The p-values are determined by the fraction permutations that have a larger value for the test statistic than the test statistic of the data matrix. The test statistic used is the sum of squares of the treatment level averages.

Value

An array is returned that contains the p-value per factor or interaction of the ASCA.

Note

Output of ASCA.Calculate is required.

Author(s)

Tim Dorscheidt, Gooitzen Zwanenburg

References

Gooitzen Zwanenburg, Huub C.J. Hoefsloot, Johan A. Westerhuis, Jeroen J. Jansen and Age K. Smilde, *ANOVA principal component analysis and ANOVA simultaneous component analysis: a comparison*. J Chemometrics, **25**, (2011), p. 561 - 567

MARTI J. ANDERSON, and CAJO J. F. TER BRAAK, *PERMUTATION TESTS FOR MULTI-FACTORIAL ANALYSIS OF VARIANCE*. Journal of Statistical Computation and Simulation, **73(2)**, (2003) p. 85 - 113

Examples

```
## Do ASCA on all (both) factors and the interaction between the two factors
data(ASCAdata)
ASCA <- ASCA.Calculate(ASCAX, ASCAF, equation.elements = "1,2,12", scaling = TRUE)

## Do a permutation test to evaluate the significance to the two factors and the interaction.
ASCA.DoPermutationTest(ASCA, perm=1000)
```

`ASCA.GetPowerSet` *Determines the power set of the input set.*

Description

Supplies a list that contains the power set of the input set, i.e. all the possible subsets of the input set.

Usage

```
ASCA.GetPowerSet(input.set, exclude.empty.set = FALSE, exclude.complete.set = FALSE)
```

Arguments

`input.set`	the input set for which the power set is to be determined
`exclude.empty.set`	whether the empty set should be included in the results
`exclude.complete.set`	whether the original complete set is to be included in the results

Author(s)

Tim Dorscheidt

`ASCA.GetRowRepeats` *Determination of all unique row value combinations*

Description

This function will determine the unique row-patterns in the input matrix. It will return a list which elements are named after the unique row-patterns, each containing the row-indices belonging to that pattern.

Usage

```
ASCA.GetRowRepeats(mat)
```

Arguments

mat	An input matrix

Value

A list with elements named after the unique row-pattern. Each element contains the row-indices for which this pattern occurs.

Author(s)

Tim Dorscheidt

ASCA.GetSummary *Summary method for ASCA analyses*

Description

Returns a matrix which contains a summary of the ASCA results. Each row in the matrix contains results for one factor or interaction studied in the performed ASCA. The columns contain the relevant principal components (PC's) found and the amount of variance explained per PC. A max of 10 PC's are shown, and only those PC's which explain more than 1% of the variance.

Usage

```
ASCA.GetSummary(asca, quietly = FALSE)
```

Arguments

asca	The result of an already performed ASCA by ASCA.Calculate
quietly	boolean. If TRUE the method will not print the table of results, but only return the matrix containing the results. Defaults to FALSE.

Value

summary	A matrix containing the variance explained per component per factor or interaction.

Note

Output of ASCA.Calculate is required.

Author(s)

Tim Dorscheidt

Examples

```
## load the example data
data(ASCAdata)

## Do ASCA on all (both) factors and the interaction between the two factors
ASCA <- ASCA.Calculate(ASCAX, ASCAF, equation.elements = "1,2,12", scaling = TRUE)

## Get a summary of the ASCA results
ASCA.GetSummary(ASCA)
```

`ASCA.Plot` *Plot ASCA results*

Description

This function generates several plots detailing the results of the performed ASCA

Usage

```
ASCA.Plot(asca)
```

Arguments

`asca`	A performed ASCA analysis

Details

ASCA.Plot takes the output of ASCA.Calculate as its input and generates scores and loading plots, including projections of the data on the first two principal components, for the complete data matrix and the effect matrices.

Value

Returns several plot windows in quick succession. Therefore, it is advised to capture by some other means (see examples below). The first two plots are a score plot and loadings plot of the performed principal component analysis (PCA) on the original data. Then, for each factor/interaction, the following plots follow: - a score plot of PC1 vs PC2 of the PCA performed on the means-matrix. - a loadings plot of PC1 and PC2 of the PCA performed on the means-matrix.

Note

Output of ASCA.Calculate is required as input.

Author(s)

Tim Dorscheidt

References

Gooitzen Zwanenburg, Huub C.J. Hoefsloot, Johan A. Westerhuis, Jeroen J. Jansen and Age K. Smilde, *ANOVA principal component analysis and ANOVA simultaneous component analysis: a comparison*. J Chemometrics, **25**, (2011), p. 561 - 567

Examples

```
##Plot the results after doing ASCA.Calculate
## use the data matrix, 'data', and an experimental design matrix, 'F'.
data(ASCAdata)
ASCA <- ASCA.Calculate(ASCAX, ASCAF, equation.elements = "1,2,12", scaling = TRUE)

## plot the results to a graphical output such as R's pdf writer
pdf("ASCA_Results.pdf")
ASCA.Plot(ASCA)
dev.off()
```

ASCA.PlotLoadings *Loadings plot for a specific factor/interaction of the ASCA*

Description

Allows the user to plot a single loadings plot for one factor or interaction (or for the SVD on the original data)

Usage

```
ASCA.PlotLoadings(asca, ee= "", pcs = c(1,2))
```

Arguments

asca	Results of a performed ASCA analysis
ee	Which factor or interaction to use (eg. "1", or "12", or leave empty to use the original data)
pcs	Which PCs (Principal Components) to use for plotting (eg. c1,2)

Value

Only the plot is returned.

Note

Output of ASCA.Calculate is required.

Author(s)

Tim Dorscheidt

Examples

```
##Plot selected loadings after doing PerformAsca
## use the data matrix, ASCAX, and an experimental design matrix, ASCAF.
data(ASCAdata)
ASCA <- ASCA.Calculate(ASCAX, ASCAF, equation.elements = "1,2,12", scaling = TRUE)

## plot the loadings of the first two principal components of the first factor
ASCA.PlotLoadings(ASCA, ee = "1", pcs="1,2")
```

`ASCA.PlotScores` *Score plot for a specific factor or interaction of the ASCA*

Description

Allows the user to plot a single score plot for one factor or interaction (or for the SVD on the original data)

Usage

```
ASCA.PlotScores(asca, ee = "", PCs = "1,2")
```

Arguments

`asca`	Results of a performed ASCA analysis
`ee`	Which factor or interaction to use (eg. "1", or "12", or leave empty to use original data)
`PCs`	Which PCs to use for plotting (eg. "1,2")

Value

Only the plot is returned.

Author(s)

Tim Dorscheidt

ASCA.PlotScoresPerLevel

ASCA scores plot with projected data.

Description

Plots the ASCA scores with projected data for a selected factor or interaction.

Usage

```
ASCA.PlotScoresPerLevel(asca, ee, pcs = "1,2")
```

Arguments

asca	results of a performed ASCA analysis
ee	which factor/interaction to use (eg. "1" or "12")
pcs	which PCs to use for plotting (eg. "1,2")

Value

Only the plot is returned

Note

Output of PerformAsca is required as input.

Author(s)

Tim Dorscheidt, Gooitzen Zwanenburg

References

Gooitzen Zwanenburg, Huub C.J. Hoefsloot, Johan A. Westerhuis, Jeroen J. Jansen and Age K. Smilde, *ANOVA principal component analysis and ANOVA simultaneous component analysis: a comparison*. J Chemometrics, **25**, (2011), p. 561 - 567

Examples

```
##Plot the results after doing PerformAsca
## use the data matrix, ASCAX, and an experimental design matrix, ASCAF.
data(ASCAdata)
ASCA <- ASCA.Calculate(ASCAX, ASCAF, equation.elements = "1,2,12", scaling = TRUE)

## plot the scores for the first two principal components and the projections of
## the data for the second factor
ASCA.PlotScoresPerLevel(ASCA, ee = "2", pcs = "1,2")
```

ASCAdata *Example data for ASCA.Calculate*

Description

The example contains data for a two factor experimental design. The first factor has two treatment levels, the second has three treatment levels. The design is balanced with 10 observations per factor/treatment level.

Usage

```
data(ASCAdata)
```

Format

The format is:
ASCAX numerical matrix of dimension 60 4
ASCAF numerical matrix of dimension 60 2

Details

The

ASCAF *Indicator matrix for ASCA example.*

Description

The matrix ASCAF gives the factor/treatment combinations for the rows of the data matrix ASCAX.

Usage

```
data(ASCAdata)
```

Format

The format is:
numerical matrix of dimension 60 2

Details

Each column in ASCAF represents a factor, the numbers in the columns correspond to the treatment level of that factor. In this example, the first factor has two treatment levels, indicated by 1 and 2 in the first column of ASCAF. The second factor has three treatment levels, indicated by the numbers 1, 2 and 3 in the second column. As an example, the two numbers 1 1 in the first row of ASCAF indicate that the first measurement was made on a sample in treatment level 1 for both factors.

ASCAX *Data matrix for ASCA example*

Description

The data matrix ASCAX contains the measurements of 4 variables in a two factor experimental design.

Usage

```
data(ASCAdata)
```

Format

The format is:
numerical matrix, dim(ASCAX) = 60 4

Details

The rows of the data matrix ASCAX contain measurements of 4 variables for an experimental design with two experimental factors. The first factor has 2 treatment levels, the second factor has 3 treatment levels. The assignments of the rows to the factor/treatment combinations is done trhough the matrix ASCAF. The design is balanced with 10 measurements per factor/treatment combination.

FoM.Calculate *Calculate best fitting Figure of Merit using method by Van Batenburg et al. (2011).*

Description

This function will accept metabolomics sample data for a single metabolite and includes sample identification. It will bin corresponding samples by intensity and calculate a measure for each bin's error. The intensity per bin versus this measure of error will be plotted, and a best fit will be found and plotted that consists of a additive and a multiplicative part. For more details see the references.

Usage

```
FoM.Calculate(data, cols.id = 1, col.value, ids.per.bin, quiet = FALSE,
repeats.per.id = -1, fit.type = 1, alpha.steps = 100)
```

Arguments

data	a matrix for which each row contains a sample's id and metabolomics data.
cols.id	indicate which column or columns in the matrix contain information pertaining to each sample's id, in which repeats of the same sample have the same id and can be binned together.
col.value	column in the data matrix that contains the metabolomics data to be plotted and fitted.
ids.per.bin	number of sample rows per bin.
quiet	should text output be given on the status of the calculations.
repeats.per.id	max number of repeats each sample may contain. If lower than the actual number repeats found, these extra repeats are ignored.
fit.type	which type of fitting should be used. Can be either "1" for use of the default R method "lm", or "2" for the more robust fitting method "rlm" from the 'MASS' package.
alpha.steps	how many points on the X axis in the final graph should be tried as a separation between the additive and the multiplicative fitting parts for finding the best fit.

Value

In addition to plotting the bins and showing the best fit, a list is returned containing the following values:

best.fit	a vector containing the three parameters that define the best fit: alpha value, additive coefficient, multiplicative coefficient, and the residual of the fit.
alphas	a vector containing all the alpha values used for fitting.
tot.res.ssq.per.alpha	a vector that gives the alpha values' corresponding residuals per fit.
ad.coeff.per.alpha	a vector that gives the alpha values' corresponding additive coefficient per fit.
mu.coeff.per.alpha	a vector that gives the alpha values' corresponding multiplicative coefficient per fit.

Author(s)

Marcel van Batenburg and Tim Dorscheidt

References

New Figures of Merit for Comprehensive Functional Genomics Data: The Metabolomics Case. Van Batenburg MF, Coulier L, van Eeuwijk F, Smilde AK, Westerhuis JA. Analytical Chemistry, Volume 83(9) (2011), pages 3267-3274

Examples

```
data(FoMData)
FoM.Calculate(FoMData, cols.id = c(1,2), 3, 5, quiet = FALSE, repeats.per.id = -1,
fit.type = 1, alpha.steps = 100)
```

FoM.FitBinnedSampleRepeatErrors

Fitting step in Figure of Merit calculation.

Description

This is the last step in obtaining a figure of merit. The binned data is fitted by iterating over all alpha values. For each alpha value the best additive fit left of the alpha value and the best multiplicative fit right of the alpha value is determined. The alpha value which then produces the best overall fit (smallest residual) is chosen for the overall fit.

Usage

```
FoM.FitBinnedSampleRepeatErrors(ordered.bins.by.mean, fit.type = 1, alpha.steps = 100)
```

Arguments

`ordered.bins.by.mean`	the result of the FoM.OrderAndBinIdByMeans method.
`fit.type`	which type of fitting approach should be used. This can be either "1" for use of the standard "lm" method in R. Or can be "2" in case the "rlm" method from the MASS package should be used. The latter is a more robust fitting method.
`alpha.steps`	the number of alpha values to be used (in essence it defines the number of points on the x axis for which different fits need to be tried to find the best fit).

Value

In addition to plotting the bins and showing the best fit, a list is returned containing the following values:

`best.fit`	A vector containing the three parameters that define the best fit: alpha value, additive coefficient, multiplicative coefficient, and the residual of the fit.
`alphas`	A vector containing all the alpha values used for fitting.
`tot.res.ssq.per.alpha`	A vector that gives the alpha values' corresponding residuals per fit.
`ad.coeff.per.alpha`	A vector that gives the alpha values' corresponding additive coefficient per fit.
`mu.coeff.per.alpha`	A vector that gives the alpha values' corresponding multiplicative coefficient per fit.

Note: this result is identical to the FoM.Calculate result, since this method is the last step in the overall FoM method.

Author(s)

Tim Dorscheidt

`FoM.GetIdOrderedMatrix`

Order the matrix of samples by their ID

Description

Finds the sample rows in the matrix 'data' that contain repeats of the same sample on the basis of the identifying columns, and assigns a new single unique ID to each of these rows. Also orders the matrix on the basis of these newly assigned IDs.

Usage

```
FoM.GetIdOrderedMatrix(data, cols.id = 1, cols.values = -1)
```

Arguments

`data`	the matrix which contains both the sample data and the identification information per sample.
`cols.id`	which columns contain the identity information per sample.
`cols.values`	which columns contain the relevant metabolomics sampling data.

Value

A list is returned containing the following values:

`ids`	A list that contains both the combinations of ID values found in the ID columns which are unique (row.patterns) and, for each of these combinations, a list with the row indices (indices.per.pattern) belonging to that unique ID combination.
`data`	A subset of the input data matrix which excludes all the identification columns (cols.id) and only contains columns defined by the argument cols.values.
`means.per.id`	Each sample can constitute of multiple repeats. This matrix contains the mean per sample over all of its repeats per value column requested.

Author(s)

Tim Dorscheidt

FoM.OrderAndBinIdByMeans

Bin sample repeats and samples with similar means

Description

Before the Figure of Merit method can be applied, sample values need to be binned in greater numbers in order to obtain a measure of error over multiple samples. Sample are binned in two ways. First, repeats of the same sample are binned together, which is based on identical values in the sample row's identity column. Second, samples are binned together that have a similar intensity according to the sample mean.

Usage

```
FoM.OrderAndBinIdByMeans(id.ordered.matrix, value.col, ids.per.bin, quiet = FALSE,
repeats.per.id = -1)
```

Arguments

id.ordered.matrix	the result of the FoM.GetIdOrderedMatrix function
value.col	only one value column can be chosen on which the binning is based. The FoM.GetIdOrderMatrix pre-processing step has already selected a subset of the original value columns. From this reduced matrix, the user should make a selection of only one value column (usually the first column).
ids.per.bin	how many sample rows should a bin contain.
quiet	boolean; if TRUE, intermediate results and feedback during calculation are shown.
repeats.per.id	maximum number of repeats each sample may contain. Any sample repeats above this value are treated as a new separate sample for binning purposes.

Value

A list is returned containing the following values:

mean.per.bin	A vector which contains the mean value per bin.
error.per.bin	A vector which contains the error value (mean of the variance per sample) per bin.
name.of.value.col	The column name of the value column in the original data matrix.

Author(s)

Tim Dorscheidt

FoMData *Example data for FoM.Calculate*

Description

The data set consists of a 151 by 3 data matrix. Each row represents a measurement, the first two colunms identify the samples by plot number and batch number. Plot could represent a plot in a field trial and batch a collection date. Likewise, plot could represent a tissue and batch a number of samples taken from the tissues.

Usage

```
data(FoMData)
```

Format

The format is:
matrix, dim(FoMData)=151, 3
colnames(FoMData)="Plot" "Batch" "P01"

Details

The program collects observations with unique number combinations of the identification columns, in this example the first two columns. The samples are then ordered according to the intensity of the measured value (third column in this example). After ordering the samples are binned and the binned values are fitted to the Rocke-Lorentzano model. The program returns the best fit parameters and a plot of the variance against the intensity of the binned values.

MetStaT.ConcatWithStringPars
Concatenates an array of strings

Description

Takes an array of string values and simply pastes them together to create one uninterrupted string value.

Usage

```
MetStaT.ConcatWithStringPars(string)
```

Arguments

string	the array containing the string values.

Value

A single string value.

Author(s)

Tim Dorscheidt

`MetStaT.ConvertToNumeric`

Converts a matrix of strings to numeric values

Description

This function takes a matrix containing string values and converts them to numeric values wherever possible. Strings of value "na" (case insensitive) are converted to NA without giving a warning. Uninterpretable strings also result in a NA value, but do generate a warning.

Usage

```
MetStaT.ConvertToNumeric(matrix)
```

Arguments

`matrix`	a matrix of string values.

Value

A numerical matrix is returned.

Author(s)

Tim Dorscheidt

`MetStaT.ConvertToNumericClasses`

Convert value-types in an array or matrix (per column) to pre-defined class-values

---

Description

This functions converts values in an array or matrix-column to pre-defined class-values. Each unique value in the array or matrix-column is assigned to a class-value in order of occurrence. For a matrix, this process is repeated per column (the user can define which columns). Default pre-defined class-values are -1 to 100.

Usage

```
MetStaT.ConvertToNumericClasses(data, cols = NULL, new.classes = NULL)
```

Arguments

data	an array or matrix containing the values to be converted to class values.
cols	which columns of the matrix need to be converted.
new.classes	user defined class values to be used (re-used in case class types needed exceeds class types defined)

Value

The same array or matrix as the input, but with each value replaced with numerical class values.

Author(s)

Tim Dorscheidt

MetStaT.CreateFileFromHeaderMatrix

Writes a matrix to file with column names and optionally row names as well.

Description

Essentially a wrapper function for R's write.table method. This function will write the input matrix to a file, always maintaining column name information from the input matrix. Row names in the matrix will be ignored by default, but can be included as well or supplied manually as a vector. The values in the output file will be separated by tabs.

Usage

```
MetStaT.CreateFileFromHeaderMatrix(file.to.create, header.matrix, rownames = FALSE)
```

Arguments

file.to.create	name of the file to be written.
header.matrix	the matrix containing the data which needs to be written.
rownames	set this to TRUE in case you wish the matrix' rownames to be stored as well. Or supply a vector of row names.

Value

A tab seperated output file containing the data from the matrix, and its column names (and row names if defined).

Author(s)

Tim Dorscheidt

`MetStaT.CreateFileFromHeaderRowMatrix`

Writes a matrix to file with row and column names, and optionally a description at the first (top left) data position in the file.

Description

This function will write the input matrix to a file, always including row names and column names. If no row or column names are present in the input matrix, default names will be generated. In addition, the first data position in the output file is reserved for a description.

Usage

```
MetStaT.CreateFileFromHeaderRowMatrix(file.to.create, row.header.matrix, description = "")
```

Arguments

`file.to.create`	name of the file to be written.
`row.header.matrix`	the matrix containing the data which needs to be written. Its row and column names will also be written to the output file (when not present in the matrix, default names are generated).
`description`	an optional data description can be supplied that will fill the first data position in the output file (the position that would normally be empty).

Value

A tab seperated output file containing the data from the matrix and its row and column names (default names are generated if where none were supplied). The first data position in the file (which is neither used by the matrix data, nor any of the row or column names) can contain an optional description.

Author(s)

Tim Dorscheidt

`MetStaT.ExportDataToFile` *Exports the results of supplied R expressions as text files in a single zip package.*

Description

This function will execute the expressions supplied, and will then attempt to write the results per expression to a text file using the method MetStaT.WriteDataObjectToFile. Finally, all text files are bundled as a zip file.

Usage

```
MetStaT.ExportDataToFile(zipfile.name, filename.no.ext, data.expressions, file.type)
```

Arguments

`zipfile.name`	the name of the final zip file.
`filename.no.ext`	the body of the filename that contains the data results for each of the supplied expressions. The final name will have a number added corresponding to the expression's index in the supplied expressions vector.
`data.expressions`	a vector of R expressions which need to be executed and whose results will be written to a text file.
`file.type`	the extension name for each of the text files.

Value

A single zip file which contains separate text files that each contain the results for one of the supplied R expressions.

Author(s)

Tim Dorscheidt

`MetStaT.GetFreqTable` *Tabulates the frequency of all unique values.*

Description

Very similar to R's table function in the base package. However, it returns the results in the order of occurrence, and not ordered by name orvalue. In addition, instead of tabulating over all unique value-occurrences in the data, an array can be supplied with pre-defined class-values over which to tabulate.

Usage

```
MetStaT.GetFreqTable(classes.to.check, class.types = NULL)
```

Arguments

classes.to.check	an array that contains the values with the unique class-values to be tabulated.
class.types	an optional array which contains pre-defined class values over which to tabulate.

Author(s)

Tim Dorscheidt

MetStaT.GetPcTuples *Obtain all possible pairs of principal components out of a defined set.*

Description

Supplied with the total number of principal components of interest, this method will return a list of each possible pairing between two principal components.

Usage

```
MetStaT.GetPcTuples(no.pc)
```

Arguments

no.pc	the number of principal components for which pairings need to found.

Value

Returns a list of pairs, each containing the two principal component numbers for that pairing.

Author(s)

Tim Dorscheidt

`MetStaT.KillAllDevices` *Kills all currently active plot windows.*

Description

All active plot windows are shut down.

Usage

```
MetStaT.KillAllDevices()
```

Value

returns nothing, except for the result of all plotting devices being shut down.

Author(s)

Tim Dorscheidt

`MetStaT.mldivide` *Matrix left hand division using a copy of the 'pracma' package 'mldivide' method.*

Description

This is a verbatim copy of the 'mldivide' function contained in the 'pracma' package by Hans W. Borchers (under GPL>=3 license), as found on CRAN on November 27, 2012. Together with the 'mrdivide' method, these are the only two 'pracma' package methods used in the 'MetStaT' package, which is why they were copied instead of depending on the entire 'pracma' package. For details on the mldivide method, please see the help file in the 'pracma' package.

Usage

```
MetStaT.mldivide(A, B)
```

Arguments

`A`

`B`

Author(s)

Original author: Hans W. Borchers. Copied and included in the MetStaT package by Tim Dorscheidt.

`MetStaT.mrdivide` *Matrix right hand division using a copy of the 'pracma' package 'mrdivide' method.*

Description

This is a verbatim copy of the 'mrdivide' function contained in the 'pracma' package by Hans W. Borchers (under GPL>=3 license), as found on CRAN on November 27, 2012. Together with the 'mldivide' method, these are the only two 'pracma' package methods used in the 'MetStaT' package, which is why they were copied instead of depending on the entire 'pracma' package. For details on the mrdivide method, please see the help file in the 'pracma' package.

Usage

```
MetStaT.mrdivide(A, B)
```

Arguments

`A`

`B`

Author(s)

Original author: Hans W. Borchers. Copied and included in the MetStaT package by Tim Dorscheidt.

`MetStaT.PlotToFile` *Save results of R plotting expressions to files in a zip package.*

Description

Executes supplied R plotting expressions, saves each plot as a separate graphical file, and bundles all files as a zip package.

Usage

```
MetStaT.PlotToFile(zipfile.name, filename.no.ext, plot.expressions, file.type, ...)
```

Arguments

`zipfile.name`	name of zip file that will contain all the plot outputs.
`filename.no.ext`	body of the file name for each of the plot results. A numbered index is added to this name.
`plot.expressions`	expressions that are to be executed in R and have a plot as a result.

file.type	which graphical output is to be used for writing the plots to a file. Can be either: "pdf", "bmp", "jpeg", "png", or "tiff", which will output the corresponding graphical format.
...	further arguments that are to be forwarded to the method which writes the plot to a file (eg. pdf (filename, output.width, output.height, ...)).

Value

A single zip file is returned, which will contain the seperate plots as individual files in the format chosen.

Author(s)

Tim Dorscheidt

MetStaT.ReadFileToHeaderMatrix

Reads a data file to a matrix.

Description

This function can read data files that have their data separated by a certain symbol (eg. comma's or tabs). In addition, the method can handle data files which contain row and column names, and maintains these in the resulting matrix.

Usage

```
MetStaT.ReadFileToHeaderMatrix(file.to.read, file.contains.header = TRUE,
file.contains.row.names = FALSE, rows = "", cols = "", separator = "\t",
force.numeric = FALSE)
```

Arguments

file.to.read	filename of file to be used as input.
file.contains.header	boolean. Does the file contain column names in its first row.
file.contains.row.names	boolean. Does the file contain row names in its first column.
rows	can be used to include or exclude rows during import. Only include a set of rows for import by supplying positive row numbers or ranges (eg. "1,2,3" or "1:3 will both limit the import to rows 1 to 3). Exclude a set of rows for import by supplying negative row numbers or ranges (eg. "-1,-2,-3" or "-1:-3 will both exclude rows 1 to 3 (and include the other rows)).
cols	can be used to include or exclude columns during import. Only include a set of columns for import by supplying positive row numbers or ranges (eg. "1,2,3" or "1:3 will both limit the import to columns 1 to 3). Exclude a set of columns for import by supplying negative columns numbers or ranges (eg. "-1,-2,-3" or "-1:-3 will both exclude columns 1 to 3 (and include the other columns)).

separator	which separator symbol is used in the file to separate the data values (eg. "\t" for tab separated values, or "," for comma separated values).
force.numeric	boolan. If TRUE, the values in the data range of the input file are forced to numerical values (non-numerical values will become NA).

Value

Returns a matrix containing the data values in the file, and optionally the row and column names.

Author(s)

Tim Dorscheidt

`MetStaT.RemoveNaColumns`

Removes all columns in a matrix that contain one or more NAs.

Description

This function will check per column whether any of its values are NA. If so, that column is removed from the resulting output matrix.

Usage

```
MetStaT.RemoveNaColumns(input.matrix, rows.to.ignore = NULL)
```

Arguments

input.matrix	the matrix to be checked for columns containing NA values.
rows.to.ignore	a vector of row-indices which will exclude those rows when checking for NA values.

Value

The same matrix as the input matrix, with the exception of the any columns that contain NA values (ignoring any rows defined by the user).

Author(s)

Tim Dorscheidt

`MetStaT.ScalePip` *Centering and scaling function*

Description

This function is an adapted version of the 'scale' function in R's base package. It allows the user to supply a matrix, which can then be scaled and centered. Returning the resulting centered and/or scaled matrix in a list that also contains the used scaling and centering vectors.

Usage

```
MetStaT.ScalePip(x.input, center = TRUE, scale = TRUE, quietly = FALSE)
```

Arguments

`x.input`	the data matrix that needs to be scaled.
`center`	boolean. If TRUE the data will also be centered per column (the mean of each column will become zero).
`scale`	this argument defines which type of scaling is to be applied. With the default value of TRUE, the data is autoscaled. When set to "pareto", pareto scaling is applied.
`quietly`	boolan. If TRUE, no intermediate text output concerning the centering and scaling methods is returned.

Value

A list is returned containing the following values:

`data`	The scaled and/or scaled data.
`description`	A string that contains a description on the centering and/or scaling methods used.
`center.vector`	The centering vector applied to the original data.
`scale.vector`	The scaling vector applied to the original data.

Author(s)

Adapted from 'Scale' method in R's 'base' package. Edited by Tim Dorscheidt.

MetStaT.TrimCustom *Limits an input string to a certain length*

Description

This function will return an adapted version of the input string and will guarantee a certain maximum length. If the string is shortened, it will end with the default symbols ".." or a used defined ending. Additionally, the user can define a certain number of characters that need to included from the end of the original string (useful if the input strings contain salient ending characters, such as numbers).

Usage

```
MetStaT.TrimCustom(text, max.length = 5, trim.ending = "..", include.ending.length = 0)
```

Arguments

`text`	the string which needs to be shortened.
`max.length`	the guaranteed maximum length that the resulting string may have.
`trim.ending`	the symbols to used when the string is shortened.
`include.ending.length`	the number of ending characters which must also be included in the resulting string.

Value

A shortened version of the input string.

Author(s)

Tim Dorscheidt

MetStaT.WriteDataObjectToFile *Write data contained within an R object as character output to a file.*

Description

Tries to write the supplied data object as character output to a file. Supports the following data objects: matrix, numerical and character. Will also write the contents of a list object to a file if it only contains objects of the supported types.

Usage

```
MetStaT.WriteDataObjectToFile(filename, data)
```

Arguments

filename	the filename of the text file the data needs to be written to.
data	the data object which needs to be written as character output to the file. Supports matrix, numerical and character objects. Also supports writing the complete contents of a list object to a single file, if the list only contains supported data object types.

Value

A single text file containing a character representation of the data object.

Author(s)

Tim Dorscheidt

PCA.Calculate *Adapted version of R's base Principal Component Analysis function (svd)*

Description

In addition to the svd function in R's base package, this principal component (pc) analysis function also adds the variance explained per pc, and the score matrix (t).

Usage

```
PCA.Calculate(data)
```

Arguments

data	the data matrix upon which the pca is to performed.

Value

A list of results is returned, containing:

d	a vector containing the singular values of x, of length min(n, p).
v	a matrix whose columns contain the right singular vectors of x, present if nv > 0. Dimension c(p, nv).
var.explained	percentage of variance explained per pc.
t	scores.

Author(s)

Tim Dorscheidt

PCA.PlotLoadings *Loadings plot for the results of PCA.Calculate*

Description

Allows the user to plot a loadings plot for two components.

Usage

```
PCA.PlotLoadings(pr.object, pcs = c(1, 2))
```

Arguments

pr.object	The result of PCA.Calculate.
pcs	Which principal components to use for plotting (eg. "1,2")

Value

Only the plot is returned.

Note

Output of ASCA.Calculate is required.

Author(s)

Tim Dorscheidt

Examples

```
##Plot selected loadings after doing PerformAsca
## use the data matrix, 'ASCAX', and an experimental design matrix, 'ASCAF'.
data(ASCAdata)
ASCA <- ASCA.Calculate(ASCAX, ASCAF, equation.elements = "1,2,12", scaling = TRUE)

## plot the loadings of the first two principal components of the first factor
ASCA.PlotLoadings(ASCA, ee = "1", pcs="1,2")
```

PCA.PlotScores *Score plot for the results of PCA.Calculate.*

Description

Allows the user to plot a score plot for two or more components. Also allows the user to set advanced labeling options.

Usage

```
PCA.PlotScores(pr.object, pcs = c(1, 2), labels = "none", custom.labels = NULL,
dot.class.vector = NULL, col.class.vector = NULL)
```

Arguments

pr.object	the result of PCA.Calculate
pcs	Which principal components to use for plotting (eg. "1,2" or "1:4")
labels	what type of labels to be used for the scores. Can either be: "none" or "dots" which will plot simple dots, "numerical" which will plot numbers, or "custom" which will plot user defined labels contained in the custom.labels argument.
custom.labels	in case the labels argument is set to "custom", this argument needs to contain a vector of character labels to be used for plotting.
dot.class.vector	NULL by default, in which case all dots are standard dots. A numerical vector can be supplied that defines each dot's symbol type (using R's default number to symbol type conversion).
col.class.vector	NULL by default, in which case all dots are black. A numerical vector can be supplied that defines each dot's color (using R's default number to color conversion).

Value

Only the plot is returned.

Note

When using colors to distinguish between points, keep in mind that only 8 different colors are available. When using symbols to distinguish between points, keep in mind that only 25 different symbols are available.

Author(s)

Tim Dorscheidt

`QStat.Calculate` *Global test for metabolic pathway differences between conditions*

Description

Calculates the Q statistic of Goeman's global test for metabolomic pathways. Also performs a permutation test, which can either be run for all possible permutations, or for a certain number of random permutations. See references for more details.

Usage

```
QStat.Calculate(X, y.boolean, permutations = "all")
```

Arguments

`X`	matrix with sampling data per row.
`y.boolean`	boolean. A vector of TRUE/FALSE values on whether each sample adheres to a certain condition.
`permutations`	argument to determine the type of permutation test. Can be either 'all', in which case all possible permutations are calculated (caution, this might take a long time), or a fixed number of random permutations. Set to 0 in case no permutation test is to be performed.

Value

A list will be returned with the following contents:

`y.boolean`	The original y.boolean input vector.
`X`	The original X input vector.
`Q`	The calculated Q statistic.
`p`	The p value found after the permutation test.

Author(s)

Diana Hendrickx and Tim Dorscheidt

References

Global test for metabolic pathway differences between conditions. Diana M. Hendrickx, Huub C.J. Hoefsloot, Margriet M.W.B. Hendriks, Andre. Canelas and Age K. Smilde. Analytica Chimica Acta, Volume 719 (2012), pages 8 - 15

Examples

```
data(QStat)
QStat.Calculate(QStatX, QStatY, 1000)
```

QStat.PermutationTest *Accompanying permutation test for QStat.Calculate*

Description

Performs a permutation test for the QStat.Calculate outcome, which is by default already done by that method.

Usage

```
QStat.PermutationTest(Qresult, no.permutations = "all", quietly = FALSE)
```

Arguments

Qresult	the result of the function QStat.Calculate.
no.permutations	either 'all', in which case all possible permutations are performed. Or a numerical value in case a certain number of randomly determined permutations should be performed.
quietly	boolean. Set to TRUE in case you wish to receive intermediate text feedback.

Value

The same list as the input argument Qresult is returned, but with a (re)calculated p value.

Author(s)

Diana Hendrickx and Tim Dorscheidt

References

Global test for metabolic pathway differences between conditions. Diana M. Hendrickx, Huub C.J. Hoefsloot, Margriet M.W.B. Hendriks, Andr?. Canelas and Age K. Smilde. Analytica Chimica Acta, Volume 719 (2012), pages 8 - 15

QStatX *Data matrix for QStat.Calculate example*

Description

This provides an example data matrix for the QStat.Calcualte method

Usage

```
data(QStat)
```

Format

The format is: numeric matrix dim(QStatX) = 18 6

Details

The matrix contains 18 rows, each row represents measurements of 6 variables. The first 10 rows are normally distributed numbers with zero mean and unit variance. The remaining 8 rows each have normally distributed numbers with mean 1 and variance 2.

Source

Global test for metabolic pathway differences between conditions. Diana M. Hendrickx, Huub C.J. Hoefsloot, Margriet M.W.B. Hendriks, Andre. Canelas and Age K. Smilde. Analytica Chimica Acta, Volume 719 (2012), pages 8 - 15

`QStatY` *Matrix for QStat.Calculate example*

Description

This provides the indicator matrix for the QStat.Calcualte method

Usage

```
data(QStat)
```

Format

The format is: numeric matrix dim(QStatX) = 18 1

Details

The matrix contains 18 rows, each row indicates whether the outcome belongs to the first condition, 0, or two the second condition, 1.

Source

Global test for metabolic pathway differences between conditions. Diana M. Hendrickx, Huub C.J. Hoefsloot, Margriet M.W.B. Hendriks, Andre. Canelas and Age K. Smilde. Analytica Chimica Acta, Volume 719 (2012), pages 8 - 15

RevNet.JacobianMethod *Reverse engineering of networks: Penalized Jacobian method*

Description

Network connections are estimated by calculating the Jacobian Matrix of the network. Details of algorithm and the theory behind the algorithm can be found in the references section. This method needs high frequency sampling data of small perturbations and steady state concentrations.

Usage

```
RevNet.JacobianMethod(data, delta.t, steady.state.concentrations, lamba.penalty.par,
kappa.penalty.par, jacobian.threshold)
```

Arguments

data	multi-dimensional sampling matrix: time x variables x experiments
delta.t	time between subsequent time points used to calculate the fourth order approximation to the Jacobian. It the data are not evenly sampled interpolation can be used to obtain evenly distributed data with time interval delta.t.
steady.state.concentrations	a vector indicating the steady concentrations of the variables (must match second dimension size of 'data')
lamba.penalty.par	lambda penalty parameter to ensure sparsity in the Jabocian matrix
kappa.penalty.par	kappa penalty parameter to ensure sparsity in the Jabocian matrix
jacobian.threshold	threshold above which values in the found Jacobian matrix indicate an edge in the network.

Value

A connectivity matrix is returned.

Author(s)

Diana Hendrickx and Tim Dorscheidt

References

Reverse engineering of metabolic networks, a critical assessment. Diana M. Hendrickx, Margriet M. W. B. Hendriks, Paul H. C. Eilers, Age K. Smilde and Huub C. J. Hoefsloot. Mol. BioSyst, Volume 7:2 (2011) pages 511-520

Examples

```
data(RevNetJacobian)
RevNet.JacobianMethod(RevNetJacobian, 0.01, SteadyState, 0.0002, 1, 0.0001)
```

`RevNet.TimeLaggedMethod`

Reverse engineering of networks: Time-lagged correlation method

---

Description

This method first finds the time-lagged correlation method, which it then converts to a connectivity matrix. Details of algorithm and the theory behind the algorithm can be found in the references section. This method needs sampling data of regular perturbations.

Usage

```
RevNet.TimeLaggedMethod(data, max.time.lag, threshold)
```

Arguments

`data`	matrix with sampling data: time x variables
`max.time.lag`	the maximum time lag for which correlations need to be taken into account
`threshold`	threshold to affect sparsity of connection matrix. Define as a percentage the quantile-cutoff of values in the correlation matrix to not be interpreted as edges in the network.

Value

A connectivity matrix is returned.

Author(s)

Diana Hendrickx and Tim Dorscheidt

References

Reverse engineering of metabolic networks, a critical assessment. Diana M. Hendrickx, Margriet M. W. B. Hendriks, Paul H. C. Eilers, Age K. Smilde and Huub C. J. Hoefsloot. Mol. BioSyst, Volume 7:2 (2011) pages 511-520

Examples

```
data(RevNetTimeLagged)
RevNet.TimeLaggedMethod(RevNetTimeLagged, 50, .55)
```

RevNet.ZeroSlopesMethod *Reverse engineering of networks: zero slopes method*

Description

Network connections are estimated by analyzing concentration profiles, whereby non-zero slopes possibly indicate a network connection. Details of algorithm and the theory behind the algorithm can be found in the references section. This method needs very high frequency sampling data of single variable perturbations.

Usage

```
RevNet.ZeroSlopesMethod(X, deltat, threshold)
```

Arguments

X	multi-dimensional sampling matrix: time x variables x experiments
deltat	single value indicating the time difference between sampling moments
threshold	threshold for determining whether an edge is assumed or not; affects network-sparsity.

Value

A connectivity matrix is returned.

Author(s)

Diana Hendrickx and Tim Dorscheidt

References

Reverse engineering of metabolic networks, a critical assessment. Diana M. Hendrickx, Margriet M. W. B. Hendriks, Paul H. C. Eilers, Age K. Smilde and Huub C. J. Hoefsloot. Mol. BioSyst, Volume 7:2 (2011) pages 511-520

Examples

```
data(RevNetZeroSlopes)
RevNet.ZeroSlopesMethod(RevNetZeroSlopes, 0.001, 0.00005)
```

RevNetJacobian *Example data for RevNetJacobianMethod*

Description

The data provide an example for the Jacobian method to reverse engineer a metabolomic network. The matrix dataset contains measured metabolite concentrations at different times, the list labels the names of the metabolites, `delta.t` is the time between measurements.

Usage

```
data(RevNetJacobian)
```

Format

The format is:
RevNetJacobian, numerical matrix with dim(dataset) = 51 4 4.
first dimension: time
second dimension: metabolites
third dimension: experiments
SteadyState, numerical matrix with steady state concentrations

Details

The dataset contains 4 experiments with the following initial conditions:
experiment 1, metabolite 1 is 2% increased from steady state
experiment 2, metabolite 2 is 2% increased from steady state
experiment 3, metabolite 3 is 2% increased from steady state
experiment 4, metabolite 4 is 2% increased from steady state

The time between measurements, `delta.t`, is used to calculate the Jacobian matrix. It is assumed that these times between measurements are all the same, with value `delta.t`. If not, interpolation can be used to obtain metabolite concentrations at regular time intervals `delta.t`.
Running the example produces the vertex-edge matrix:

1 1 0 0
1 1 1 0
0 1 1 0
0 0 1 1

Source

Chassagnole, C., D. A. Fell, et al. (2001). "Control of the threonine-synthesis pathway in Escherichia coli: a theoretical and experimental approach." Biochemical Journal 356: 433-444.

References

Reverse engineering of metabolic networks, a critical assessment. Diana M. Hendrickx, Margriet M. W. B. Hendriks, Paul H. C. Eilers, Age K. Smilde and Huub C. J. Hoefsloot. Mol. BioSyst, Volume 7:2 (2011) pages 511-520

`RevNetTimeLagged` *Example data for RevNetTimeLaggedMethod*

Description

The data provide an example for the Time Lagged method to reverse engineer a metabolomic network. The matrix dataset contains measured metabolite concentrations at different times.

Usage

```
data(RevNetTimeLagged)
```

Format

The format is:
RevNetTimeLagged, numerical matrix with dim(RevNetTimeLagged) = 100 4 first dimension: time second dimension: metabolites

Details

The dataset contains metabolite concentrations from the following experiment:
Every 9 seconds metabolite 1 is increased 20% from steady state. Running the example produces the vertex-edge matrix:

1 1 0 0
1 1 1 0
0 1 1 1
0 0 1 1

Source

Chassagnole, C., D. A. Fell, et al. (2001). "Control of the threonine-synthesis pathway in Escherichia coli: a theoretical and experimental approach." Biochemical Journal 356: 433-444.

References

Reverse engineering of metabolic networks, a critical assessment. Diana M. Hendrickx, Margriet M. W. B. Hendriks, Paul H. C. Eilers, Age K. Smilde and Huub C. J. Hoefsloot. Mol. BioSyst, Volume 7:2 (2011) pages 511-520

RevNetZeroSlopes *Example data for RevNetZeroSlopesMethod*

Description

The data provide an example for the Zero Slopes method to reverse engineer a metabolomic network. The matrix dataset contains measured metabolite concentrations at different times.

Usage

```
data(RevNetZeroSlopes)
```

Format

The format is:
RevNetZeroSlopes, numerical matrix with dim(dataset) = 51 4 4.
first dimension: time
second dimension: metabolites
third dimension: experiments

Details

The dataset contains 4 experiments with the following initial conditions:
experiment 1, metabolite 1 is 20% increased from steady state
experiment 2, metabolite 2 is 20% increased from steady state
experiment 3, metabolite 3 is 20% increased from steady state
experiment 4, metabolite 4 is 20% increased from steady state

Running the example produces the vertex-edge matrix:

-1 1 0 0
1 -1 1 0
0 1 -1 0
0 0 1 -1

Source

Chassagnole, C., D. A. Fell, et al. (2001). "Control of the threonine-synthesis pathway in Escherichia coli: a theoretical and experimental approach." Biochemical Journal 356: 433-444.

References

Reverse engineering of metabolic networks, a critical assessment. Diana M. Hendrickx, Margriet M. W. B. Hendriks, Paul H. C. Eilers, Age K. Smilde and Huub C. J. Hoefsloot. Mol. BioSyst, Volume 7:2 (2011) pages 511-520

`SteadyState` *Example data for RevNetJacobianMethod*

Description

Steady state concentrations used by the Jacobian method to reverse engineer a metabolomic network.

Usage

```
data(RevNetJacobian)
```

Format

SteadyState, numerical matrix with steady state concentrations
dim(SteadyState) = 1 4

Source

Chassagnole, C., D. A. Fell, et al. (2001). "Control of the threonine-synthesis pathway in Escherichia coli: a theoretical and experimental approach." Biochemical Journal 356: 433-444.

References

Reverse engineering of metabolic networks, a critical assessment. Diana M. Hendrickx, Margriet M. W. B. Hendriks, Paul H. C. Eilers, Age K. Smilde and Huub C. J. Hoefsloot. Mol. BioSyst, Volume 7:2 (2011) pages 511-520

Index