Package 'missRows'

November 12, 2026

Type Package

Title Handling Missing Individuals in Multi-Omics Data Integration

Version 1.32.0

Date 2022-01-14

Author Ignacio Gonzalez and Valentin Voillet

Maintainer Gonzalez Ignacio <ignacio.gonzalez@bbox.fr>

Description The missRows package implements the MI-MFA method to deal with missing individuals ('biological units') in multi-omics data integration. The MI-MFA method generates multiple imputed datasets from a Multiple Factor Analysis model, then the yield results are combined in a single consensus solution. The package provides functions for estimating coordinates of individuals and variables, imputing missing individuals, and various diagnostic plots to inspect the pattern of missingness and visualize the uncertainty due to missing values.

License Artistic-2.0

Depends R (>= 3.5), methods, ggplot2, grDevices, MultiAssayExperiment

Imports plyr, stats, gtools, S4Vectors

VignetteBuilder knitr

Suggests BiocStyle, knitr, testthat

biocViews Software, StatisticalMethod, DimensionReduction, PrincipalComponent, MathematicalBiology, Visualization

RoxygenNote 6.0.1

git_url https://git.bioconductor.org/packages/missRows

git_branch RELEASE_3_23

git_last_commit e2bb57e

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

missRows-package . . . 2
eigenvalue . . . 3
estimNC . . . 4
imputeDataMFA . . . 5
imputedData . . . 6
MFA . . . 7
MIDTList-class . . . 8
MIMFA . . . 10
missPattern . . . 12
NCI60 . . . 13
plotInd . . . 15
plotVar . . . 16
RVcoeff . . . 18
searchsComb . . . 19
STATIS . . . 20
tuneM . . . 20
wrapperSVD . . . 22

Index **23**

`missRows-package` *Handling Missing Individuals in Multi-Omics Data Integration*

Description

The missRows package implements the MI-MFA method to deal with missing individuals ('biological units') in multi-omics data integration. The MI-MFA method generates multiple imputed datasets from a Multiple Factor Analysis model, then the yield results are combined in a single consensus solution. The package provides functions for estimating coordinates of individuals and variables, imputing missing individuals, and various diagnostic plots to inspect the pattern of missingness and visualize the uncertainty due to missing values.

Details

Package:	missRows
Type:	Package
Version:	1.0
Date:	2018-03-19
License:	Artistic-2.0
Depends:	R (>= 3.4)
Imports:	methods, gtools, plyr, ggplot2, stats, grDevices, S4Vectors, MultiAssayExperiment

Author(s)

Ignacio González and Valentin Voillet

Maintainer: Ignacio González <ignacio.gonzalez@somewhere.net>

References

Voillet V., Besse P., Liaubet L., San Cristobal M., González I. (2016). Handling missing rows in multi-omics data integration: Multiple Imputation in Multiple Factor Analysis framework. *BMC Bioinformatics*, 17(40).

Examples

```
## A typical MI-MFA session might look like the following.
## Here we assume there are two data tables with missing rows,
## "table1" and "table2", and the stratum for each individual
## is stored in a data frame "df".

## Not run:

#-- Data preparation
midt <- newMIDTList(table1, table2, colData=df)

#-- Performing MI
midt <- MIMFA(midt, ncomp=2, M=30)

#-- Analysis of the results
plotInd(midt)
plotVar(midt)

## End(Not run)
```

`eigenvalue` *Scaled Firts Singular Value of the SVD of a Matrix*

Description

Obtain the scaled first singular value of the singular-value decomposition of a rectangular matrix `X` as computed by `svd`. Scaling is done by dividing the firts singular value by the root square of the number of rows in `X`. This function is internally called by `MFA` and is not usually called directly by a user.

Usage

```
eigenvalue(X)
```

Arguments

`X`	a numeric matrix whose SVD decomposition can be computed.

Value

The scaled first singular value of `svd(X)`.

`estimNC` *Estimate the Number of Components for Data Imputation*

Description

`estimNC` estimates the number of MFA components for data imputation. This function is called internally by `MIMFA` and is not usually called directly by a user.

Usage

```
estimNC(X, minNC=0, maxNC)
```

Arguments

`X`	a numeric matrix.
`minNC`	minimum number of components to consider, by default 0.
`maxNC`	maximum number of components to test.

Details

Partially borrowed from the `estim_npc` function in the **FactoMineR** package, `estimNC` estimates the number of MFA components for data imputation using the generalized cross-validation approximation method.

Value

Return the number of MFA components to use in data imputation.

References

Josse, J. and Husson, F. (2012). Selecting the number of components in PCA using cross-validation approximations. *Computational Statistics and Data Analysis*, **56**, 1869-1879.

`imputeDataMFA` *Impute Missing Rows and Estimates MFA Axes*

Description

Impute the missing rows of data tables using the alternating least squares algorithm used in PCA. This function is internally called by `MIMFA` and is not usually called directly by a user.

Usage

```
imputeDataMFA(datasets, U, missRows, comp, maxIter=500, tol=1e-10)
```

Arguments

`datasets`	a list containing the data tables with missing rows. Tables in the list should be arranged in samples $\times$ variables, with samples order matching in all data tables.
`U`	the compromise configuration, a matrix with the individuals coordinates as returned by `STATIS` function.
`missRows`	a list containing character vectors with the name of the missing individuals (rows) per table.
`comp`	a number of components kept for imputation.
`maxIter`	integer, maximum number of iterations for the iterative algorithm.
`tol`	positive value, the threshold for assessing convergence.

Details

Since the core of MFA is a PCA of the merged data tables K, the algorithm suggested to estimate MFA axes and impute missing values is inspired from the alternating least squares algorithm used in PCA. This consists in finding matrices F and U which minimize the following criterion:

$$||K - M - FU||^2 = \sum_i \sum_k \left(K_{ik} - M_{ik} - \sum_{d=1}^{D} F_{id} U_{kd} \right)^2 ,$$

where M is a matrix with each row equal to a vector of the mean of each variable and D is the kept dimensions in PCA. The solution is obtained by alternating two multiple regressions until convergence, one for estimating axes (loadings $\hat{U}$) and one for components (scores $\hat{F}$):

$$\hat{U}' = (\hat{F}'\hat{F})^{-1}\hat{F}'(K - \hat{M})$$
$$\hat{F} = (K - \hat{M})\hat{U}(\hat{U}'\hat{U})^{-1}.$$

The `imputeDataMFA` algorithm first consists in imputing missing values in K with initial values (the column means on the non-missing entries), then $\hat{M}$ is computed. The second step of the iterative algorithm is to calculate $\hat{F} = (K - \hat{M})U(U'U)^{-1}$ on the completed dataset by using D components of U. Missing values are estimated as $\hat{K} = \hat{M} + \hat{F}U'$. The new imputed data set K is

obtained by replacing the missing values of the original K matrix with the corresponding elements of $\hat{K}$, whilst keeping the observed values unaltered. These steps of estimation of the parameters and imputation of the missing values are iterate until convergence. The number D of components used in the algorithm can be estimated setting the `estim.ncp` argument to `TRUE` in the function `MIMFA`.

Value

A list containing components with the imputed rows for each data table.

Author(s)

Ignacio González

`imputedData` *Get the Imputed Data Tables*

Description

`imputedData` function exports the list of data tables with imputed data.

Usage

```
imputedData(object)
```

Arguments

`object`	an object of class `MIDTList` as returned by `MIMFA` function.

Value

A list of length the data tables number, each component containing a completed data table.

Author(s)

Ignacio González

See Also

`MIMFA`

Examples

```
#-- load data and create MIDTList object
data(NCI60)
midt <- MIDTList(NCI60$mae)

#-- performs MIMFA
midt <- MIMFA(midt, ncomp=2, M=5)

#-- exports the imputed data tables
completeData <- imputedData(midt)
```

MFA *Multiple Factor Analysis (MFA)*

Description

Perform Multiple Factor Analysis on quantitative variables. This function is internally called by MIMFA and is not usually called directly by a user.

Usage

```
MFA(dataTables, ncomp, nbRows, nbTables, ncTables)
```

Arguments

dataTables	a list containing data tables without missing data. Tables in the list should be arranged in samples $\times$ variables, with samples order matching in all data tables.
ncomp	a number of components to include in MFA.
nbRows	a number of rows, equal for all data tables.
nbTables	a number of data tables in dataTables.
ncTables	a vector containing the number of columns per data table.

Details

MFA function performs Multiple Factor Analysis in the sense of Escofier-Pages on data tables of quantitative variables.

Value

MFA returns a matrix of individuals coordinates.

Author(s)

Ignacio González, Valentin Voillet

References

Escofier, B. and Pages, J. (1994) Multiple Factor Analysis (AFMULT package). *Computational Statistics and Data Analysis*, **18**, 121-140.

See Also

MFA in FactoMineR

MIDTList-class *Infrastructure for Multiple Imputation and Multi-omics Experiments*

Description

MIDTList is an S4 class that extends the class MultiAssayExperiment by providing the infrastructure (slots) to store the input data, intermediate calculations and results of a multiple imputation approach.

Objects from the Class

Objects can be created by calls of the form:

```
MIDTList(..., colData=NULL, strata=NULL, assayNames=NULL)
new("MIDTList", ..., colData=NULL, strata=NULL, assayNames=NULL)
```

Arguments

`...` arguments passed to '...' can be:

1) data tables with missing individuals. Two or more objects which can be interpreted as matrices (or data frames). Data tables passed as arguments in ... must be arranged in variables (rows) $\times$ individuals (columns), with individual names matching row names of colData.

2) a list containing two or more data tables with missing individuals. Data tables (matrices or data frames) within list must be arranged in variables (rows) $\times$ individuals (columns), with individual names matching row names of colData.

3) an object of class MultiAssayExperiment. In this case colData and assayNames arguments are ignored.

`colData` a DataFrame giving the characteristics for all individuals (biological units). The row names of colData must contain individual identifiers.

`assayNames` optional. A character vector giving the name for each table.

`strata` a character indicating the column of colData to be used as strata in the construction of MIDTList.

Details

To facilitate programming pipelines, NULL values are input for compromise, configurations, imputedIndv and MIparam slots, in which case the default value is used as if the argument had been missing. These slots will be updated after multiple imputation (MIMFA) approach.

Slots

`ExperimentList` an `ExperimentList` class object for each assay dataset.

`colData` a `DataFrame` of all clinical/specimen data available across experiments.

`sampleMap` a `DataFrame` of translatable identifiers of samples (individuals) and participants.

`metadata` additional data describing the `MultiAssayExperiment` object.

`drops` a metadata `list` of dropped information.

`strata`**:** a numeric value or character. The column of `colData` to be used as strata in `MIMFA`.

`missingIndv`**:** a list containing character vectors with the name of the missing individuals per table.

`compromise`**:** the compromise configuration, a matrix with the individuals coordinates as returned by `STATIS` function.

`configurations`**:** a list containing the individuals coordinates for each imputed dataset as returned by `MIMFA` function.

`imputedIndv`**:** a list containing the imputed individuals for each data table as returned by `imputeDataMFA` function.

`MIparam`**:** a list containing the parameters used in the `MIMFA` function.

Extends

Class `MultiAssayExperiment`, directly.

Methods

initialize `signature(.Object = "MIDTList")`: See 'Objects from the Class' section for description.

Class-specific methods return the corresponding objects:

strata `signature(object="MIDTList")`: Return factor of strata giving the stratum for each individual.

missingIndv `signature(object="MIDTList")`: Return list containing character vectors with the name of the missing individuals per table.

compromise `signature(object="MIDTList")`: Return matrix with the individuals coordinates as returned by `STATIS` function.

configurations `signature(object="MIDTList", M="all")`: Get all configurations. If `M` is a positive integer, the `M`th configuration is returned.

imputedIndv `signature(object="MIDTList")`: Return list containing the imputed individuals for each data table.

MIparam `signature(object="MIDTList")`: Return list containing the parameters used in the `MIMFA` function.

Standard generic methods:

show `signature(object="MIDTList")`: Informatively display object contents.

See MultiAssayExperiment-class for generic methods associated to the `MultiAssayExperiment` class.

Author(s)

Ignacio González

See Also

MultiAssayExperiment-class, MultiAssayExperiment-methods

Examples

```
#-- load data
data(NCI60)

#-- MIDTList object from separate data tables
table1 <- NCI60$dataTables$trans
table2 <- NCI60$dataTables$prote
colData <- NCI60$dataTables$cell.line

midt <- MIDTList(table1, table2, colData=colData,
                 assayNames=c("transcrip", "proteome"))
midt

#-- MIDTList object from a list
tablesList <- NCI60$dataTables[1:2]
colData <- NCI60$dataTables$cell.line

midt <- MIDTList(tablesList, colData=colData)
midt

#-- MIDTList object directly from a 'MultiAssayExperiment'
midt <- MIDTList(NCI60$mae)
midt
```

MIMFA *Handling Missing Individuals in MFA*

Description

The MIMFA function estimates coordinates of individuals and variables on the MFA components by implementing a multiple imputation (MI) approach in order to deal with multiple tables in presence of missing individuals.

Usage

```
MIMFA(object, ncomp=2, M=NULL, estimeNC=FALSE, maxIter=500, tol=1e-10)
```

Arguments

object	an object of class MIDTList.
ncomp	a number of components to include in MFA when estimeNC=FALSE (default to 2). If estimeNC=TRUE, then ncomp correspond to the maximum number of components to test.
M	integer, number of imputations. Default to min(30, Mtotal), where Mtotal is the total number of possible imputations.
estimeNC	logical. If TRUE the number of MFA components for data imputation is estimated. Default is FALSE.
maxIter	integer, maximum number of iterations for the imputeDataMFA function.
tol	positive value, the threshold for assessing convergence in the imputeDataMFA algorithm.

Details

According to the MI methodology, missing individuals are filled in by several sets of plausible values, resulting in M completed data. MFA is then applied to each completed data leading to M different configurations. Finally, the M configurations are combined using the STATIS method to yield one consensus solution.

If estimeNC=TRUE, the number of MFA components for data imputation is estimated using the generalized cross-validation approximation method. In this case, ncomp corresponds to the maximum number of components to test.

Value

A MIDTList object containing additional slots for:

compromise

configurations

imputedIndv

MIparam

See MIDTList for description.

Author(s)

Ignacio González and Valentin Voillet

References

Voillet V., Besse P., Liaubet L., San Cristobal M., González I. (2016). Handling missing rows in multi-omics data integration: Multiple Imputation in Multiple Factor Analysis framework. *BMC Bioinformatics*, 17(40).

Lavit C., Escoufier Y., Sabatier R., Traissac P. (1994). The ACT (STATIS method). *Computational Statistics & Data Analysis*, 18(1), 97–119.

Josse J., Husson F. (2012). Selecting the number of components in PCA using cross-validation approximations. *Computational Statistics and Data Analysis*, 56, 1869–1879.

See Also

plotInd, plotVar, tuneM

Examples

```
#-- load data and create MIDTList object
data(NCI60)
midt <- MIDTList(NCI60$mae)
midt

#-- performs MIMFA
midt <- MIMFA(midt, ncomp=3, M=10)
midt

#-- estimates the number of MFA components for data imputation
#-- ncomp is chosen to being enough large
## Not run:
midt <- MIMFA(midt, ncomp=50, M=10, estimeNC=TRUE)
midt
## End(Not run)
```

---

missPattern *Inspects Pattern of Missingness*

Description

This function inspects and plots the structure of missing individuals in data tables.

Usage

```
missPattern(object, colStrata=NULL, colMissing="grey70", cexTitles=12,
            legTitle="Strata", missLab="miss", showPlot=TRUE)

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

Arguments

object	an object of class MIDTList.
x	an object of class inheriting from missPattern
colStrata	a character vector of the same length than the number of strata, containing the color names to be used to annotate the individuals per stratum.
colMissing	the fill color for missing individuals.
cexTitles	a positive number. The amount by which table titles should be magnified.
legTitle	character. The legend title.
missLab	character. The label legend for missing individuals.
showPlot	logical. Whether the plot will be displayed. Default is TRUE.
...	not used currently.

Details

missPattern calculates the amount of missing/available individuals in each stratum per data table and plots a missingness map showing where missingness occurs. For plotting, tables are arranged in individuals (rows) $\times$ features (columns). Data tables are plotted separately on a same device showing the pattern of missingness. The individuals are colored according to their stratum whereas missing individuals (rows) are specific colored (see colMissing).

Value

A list with the following components:

nbMissing	a data.frame containing the amount of missing/available rows in each stratum per data table.
isMissing	a data.frame containing the indicator matrix for the missing rows.
ggp	an object of class ggplot.

Author(s)

Ignacio González

Examples

```
#-- load data and create MIDTList object
data(NCI60)
midt <- MIDTList(NCI60$mae)

#-- inspects pattern of missingness
patt <- missPattern(midt)
patt
```

NCI60 *Data of the NCI-60 Cell Lines with Missing Individuals*

Description

The NCI60 data contain both transcriptomic and proteomic expression for a collection of 60 cell lines from the National Cancer Institute (NCI-60). Data tables with missing individuals have been generated for illustration purposes.

Usage

```
data(NCI60)
```

Format

A list with two components, `dataTables` and `mae`:

`dataTables` contain a list with the following components:

`trans` a matrix containing 300 rows and 48 columns. The mRNA transcription levels of the NCI60 cell lines. There are 12 missing individuals.

`prote` a matrix containing 162 rows and 52 columns. The protein abundance levels of the NCI60 cell lines. There are 8 missing individuals.

`cell.line` a `DataFrame` of cancer types: colon (CO), renal (RE), ovarian (OV), breast (BR), prostate (PR), lung (LC), central nervous system (CNS), leukemia (LE) and melanoma (ME).

`mae` contain a 'MultiAssayExperiment' instance from NCI60 data with transcriptome and proteomic experiments as described in `dataTables`.

Details

The transcriptome data was retrieved directly from the `NCI60_4arrays` package. This data table contains gene expression profiles generated by the Agilent platform with only few hundreds of genes randomly selected to keep the size of the Bioconductor package small. However, the full dataset is available in Reinhold *et al.* (2012).

The proteomic data was retrieved directly from the `rcellminerData` package. Protein abundance levels were available for 162 proteins.

The scripts used to generate this data are contained within the inst/extdata folder of the `missRows` package.

Source

`NCI60_4arrays` package.

`rcellminerData` package.

References

Reinhold W.C., Sunshine M., Liu H., Varma S., Kohn K.W., Morris J., Doroshow J., Pommier Y. (2012). Cellminer: A web-based suite of genomic and pharmacologic tools to explore transcript and drug patterns in the NCI-60 cell line set. *Cancer Research*, 72(14):3499-511.

The CellMiner project website: http://discover.nci.nih.gov/cellminer

plotInd *Plot of Individuals (Experimental Units)*

Description

This function provides scatter plots for individuals (experimental units) representation from MIMFA results.

Usage

```
plotInd(object, comp=c(1, 2), colStrata=NULL, colMissing="white",
        confAreas=c("none", "ellipse", "convex.hull"), confLevel=0.95,
        ellipseType=c("norm", "t"), alpha=0.1, lwd=0.3, cex=3,
        legTitle="Strata")
```

Arguments

object	an object of class MIDTList as returned by MIMFA function.
comp	an integer vector of length two. The components that will be used on the horizontal and the vertical axes respectively to project the individuals.
colStrata	a character vector of the same length than the number of strata containing the color names to be used to annotate the individuals per stratum.
colMissing	the fill color for imputated individuals.
confAreas	a character string indicating whether to plot "none", "ellipse" or "convex.hull" confidence areas.
confLevel	a numerical value indicating the confidence level of ellipses being plotted when confAreas = "ellipse". The default is set to 0.95, for a 95% confidence region.
ellipseType	the type of ellipse. The default "norm" assumes a multivariate normal distribution, and "t" assumes a multivariate t-distribution.
alpha	the alpha transparency for filled color of the confidence areas, values are any numbers from 0 (transparent) to 1 (opaque).
lwd	a positive number. The border line width of the confidence areas.
cex	a positive number. The amount by which plotting symbols should be magnified.
legTitle	character. The legend title.

Details

plotInd function makes scatter plot for individuals representation from MIMFA results. Each point corresponds to an individual. The individuals are colored with rapport to their stratum, whereas imputed individuals are colored according to the colMissing argument.

Multiple imputation generates M imputed datasets and the variance between-imputations reflects the uncertainty associated to the estimation of the missing values. The plotInd function proposes two approaches to visualize the uncertainty due to missing data: confidence ellipses and convex hulls.

The idea is to project all the multiple imputed datasets onto the compromise configuration. Each individual is represented by M points, each corresponding to one of the M configurations. Confidence ellipses and convex hulls can then be constructed for the M configurations for each individual. For ease of understanding, not all individuals for the M configurations obtained are plotted.

Confidence ellipses can be created by setting the confAreas argument to "ellipse". The 95% confidence ellipses are showed by default. Convex hulls are plotted by setting the confAreas argument to "convex.hull". The computed convex hull results in a polygon containing all M solutions.

Value

An object of class ggplot.

Author(s)

Ignacio González and Valentin Voillet

See Also

plotVar

Examples

```
#-- load data and create MIDTList object
data(NCI60)
midt <- MIDTList(NCI60$mae)

#-- performs MIMFA
midt <- MIMFA(midt, ncomp=2, M=10)

#-- default plot
plotInd(midt)

#-- with confidence ellipses
plotInd(midt, confAreas="ellipse")

#-- with convex hull areas
plotInd(midt, confAreas="convex.hull")
```

---

plotVar *Plot of Variables: Correlation Circle*

Description

This function provides "correlation circle", scatter plots for variables representation from MIMFA results.

Usage

```
plotVar(object, comp=c(1, 2), col=NULL, varNames=FALSE, cex=3,
        pch=19, alpha=0.7, spty=TRUE, cutoff=0, radIn=0.5,
        overlap=TRUE, ncols=2, legTitle="Tables")
```

Arguments

object	an object of class MIDTList as returned by MIMFA function.
comp	an integer vector of length two. The components that will be used on the horizontal and the vertical axis respectively to project the variables.
col	a character or integer vector of colors for plotted character and symbols, it must be of length the total number of data tables (see Details).
varNames	either a character vector with data table names, or FALSE for no plotting variable labels. If TRUE, the variable names in the data tables are used as variable labels in the plot. See Details.
cex	a numeric vector of character expansion sizes for the plotted character and symbols, can be of length one or of length the total number of data tables. See Details.
pch	plotting 'character'. A vector of single characters or integers, can be of length one or of length the total number of data tables (see Details). See points for all alternatives.
alpha	the alpha transparency for plotting color of the symbols, values are any numbers from 0 (transparent) to 1 (opaque).
spty	logical, specifying the type of plot region to be used. If TRUE (the default), a square plotting region is generated. If not, a maximal plotting region is produced.
cutoff	a numeric between 0 and 1. Variables with correlations below this cutoff in absolute value are not plotted (see Details).
radIn	a numeric between 0 and 1, the radius of the inner circle. Defaults to 0.5.
overlap	logical. Whether the variables in data tables should be plotted in one single panel. Default is TRUE.
ncols	numeric. When overlap = FALSE subsequent figures will be drawn in a multi-panel on the device with ncols columns.
legTitle	character. The legend title.

Details

plotVar produces a "correlation circle", *i.e.* the correlations between each variable and the selected components are plotted as scatter plot, with concentric circles of radius one and radius given by radIn. Each point corresponds to a variable.

The varNames argument can be used in order to select a part of the data table variable labels that are drawn. For example if you have data tables named "table1" and "table2", you can use varNames = "table1" and then the variable names in the "table1" are drawn.

The arguments cex and pch can be either vectors of length one or of length the total number of data tables. In the first case, the single value determine the graphics attributes for all data table variables. Otherwise, multiple argument values can be specified so that each data table variable can be given its own graphic attributes. In this case, each component of the vector corresponds to the attributes of the each data table variable.

Value

A list containing the following components:

df	a data frame used to generate the ggplot.
ggp	an object of class ggplot.

Author(s)

Ignacio González

See Also

plotInd

Examples

```
#-- load data and create MIDTList object
data(NCI60)
midt <- MIDTList(NCI60$mae)

#-- performs MIMFA
midt <- MIMFA(midt, ncomp=2, M=5)

#-- default plot
plotVar(midt)

#-- select data table variables to draw and cutoff
plotVar(midt, varNames="trans", cutoff=0.55)
plotVar(midt, varNames=TRUE, cutoff=0.55)
```

RVcoeff *Calculate the RV Coefficient*

Description

Calculate the RV coefficient between two matrices X and Y. This function is internally called by tuneM and is not usually called directly by a user.

Usage

```
RVcoeff(X, Y)
```

Arguments

X, Y	a matrix with n rows and p columns.

Value

The RV coefficient between the two matrices.

References

Robert, P., Escoufier, Y. (1976). A unifying tool for linear multivariate statistical methods: The RV coefficient. *Journal of the Royal Statistical Society*. 25(3), 257–265.

---

searchsComb *Search the Element of a Combination*

Description

Search the element of a combination from all combinations of the supplied vectors as created by `expand.grid` without creating the combinations. This function is internally called by `MIMFA` and is not usually called directly by a user.

Usage

```
searchsComb(args, idx)
```

Arguments

args	a list containing the vector sources for combinations.
idx	the index of the searched combination.

Value

The combination corresponding to `idx`.

Author(s)

Ignacio González

STATIS *STATIS Analysis of Multiple Data Tables*

Description

Partially borrowed from the `statis` function implemented within the **ade4** package, `STATIS` performs a STATIS analysis of multiple data tables. This function is internally called by `MIMFA` and is not usually called directly by a user.

Usage

```
STATIS(Ktab, nf=3, tol=1e-07)
```

Arguments

`Ktab`	a list containing the data tables. Tables in the list should be arranged in samples $\times$ variables, with samples order matching in all data tables.
`nf`	an integer indicating the number of kept axes for the compromise configuration.
`tol`	a tolerance threshold to test whether an eigenvalue is positive.

Value

A data frame with the row coordinates.

References

Lavit C., Escoufier Y., Sabatier R., Traissac P. (1994) The ACT (STATIS method). *Computational Statistics & Data Analysis*, 18(1), 97–119.

See Also

`statis`

tuneM *Tune the Number of Imputations in MI-MFA*

Description

`tuneM` can be used to determine the appropriate number of imputed datasets needed to obtain satisfactory results with MI-MFA.

Usage

```
tuneM(object, ncomp=2, Mmax=30, inc=5, N=10, tol=1e-06, showPlot=TRUE)

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

Arguments

object	an object of class MIDTList.
x	an object of class inheriting from tuneM.
ncomp	a number of components to include in MFA.
Mmax	an integer corresponding to the maximum number of imputed datasets. See Details.
inc	integer. The increment of the sequence for the number M of imputations considered. See Details.
N	integer. Collections of size N are generated for each number of imputations M. See Details.
tol	a positive value, the tolerance used for assessing stabilization.
showPlot	logical. If TRUE (the default) a plot showing the stability of the estimated MFA configurations is displayed.
...	not currently used.

Details

The appropriate number of imputations can be informally determined by carrying out MI-MFA on N replicate sets of M_l imputations for $l = 0, 1, 2, \ldots$, with $M_0 < M_1 < M_2 < \cdots < M_{max}$, until the estimate compromise configurations are stabilized.

tuneM function implements such a procedure. Collections of size N are generated for each number of imputations M, with M = seq(inc, Mmax, by = inc). The stability of the estimated MI-MFA configurations is then determined by calculating the RV coefficient between the configurations obtained using M_l and M_{l+1} imputations.

If showPlot = TRUE a plot showing the stability of the estimated MFA configurations is displayed. The values shown are the mean RV coefficients for the N configurations as a function of the number of imputations. Error bars represent the standard deviation of the RV coefficients.

Value

A list with the following components:

stats	a data.frame containing the information used to generate the plot.
ggp	an object of class ggplot.

Author(s)

Ignacio González, Valentin Voillet

References

Voillet V., Besse P., Liaubet L., Cristobal M.S., González I. (2016). Handling missing rows in multi-omics data integration: Multiple Imputation in Multiple Factor Analysis framework. *BMC Bioinformatics*, 17(40).

See Also

MIMFA

Examples

```
#-- load data and create MIDTList object
data(NCI60)
midt <- MIDTList(NCI60$mae)

#-- tune the number of imputations
## Not run:
tune <- tuneM(midt, ncomp=2, Mmax=100, inc=10, N=10)
tune
## End(Not run)
```

---

wrapperSVD *Singular Value Decomposition of a Matrix*

Description

Wrapper function from the **FactoMineR** package to perform Singular Value Decomposition of a matrix. This function is called internally by MFA and is not usually called directly by a user.

Usage

```
wrapperSVD(X, rWeights=NULL, cWeights=NULL, ncp=Inf)
```

Arguments

X	a numeric matrix.
rWeights	vector with the weights of each row (NULL by default and the weights are uniform).
cWeights	vector with the weights of each column (NULL by default and the weights are uniform).
ncp	the number of components kept for the outputs.

Value

A list that contains the following components:

vs	a vector containing the singular values of X.
U	a matrix whose columns contain the left singular vectors of X.
V	a matrix whose columns contain the right singular vectors of X.

See Also

svd, svd.triplet

Index

∗ **algebra**
  eigenvalue, 3
  wrapperSVD, 22
∗ **classes**
  MIDTList-class, 8
∗ **datagen**
  MIMFA, 10
∗ **datasets**
  NCI60, 13
∗ **dplot**
  missPattern, 12
  plotInd, 15
  plotVar, 16
  tuneM, 20
∗ **hplot**
  missPattern, 12
  plotInd, 15
  plotVar, 16
∗ **multivariate**
  estimNC, 4
  imputeDataMFA, 5
  MFA, 7
  MIMFA, 10
  RVcoeff, 18
  STATIS, 20
  tuneM, 20
∗ **package**
  missRows-package, 2
∗ **utilities**
  imputedData, 6
  searchsComb, 19

compromise *(*MIDTList-class*)*, 8
compromise,MIDTList-method
        *(*MIDTList-class*)*, 8
configurations *(*MIDTList-class*)*, 8
configurations,MIDTList-method
        *(*MIDTList-class*)*, 8

eigenvalue, 3
estimNC, 4
expand.grid, *19*
ExperimentList, *9*

imputeDataMFA, 5, *9*, *11*
imputedData, 6
imputedIndv *(*MIDTList-class*)*, 8
imputedIndv,MIDTList-method
        *(*MIDTList-class*)*, 8
initialize,MIDTList-method
        *(*MIDTList-class*)*, 8

MFA, *3*, 7, *8*, *22*
MIDTList, *11*
MIDTList *(*MIDTList-class*)*, 8
MIDTList-class, 8
MIMFA, *4–9*, 10, *15–17*, *19*, *20*, *22*
MIparam *(*MIDTList-class*)*, 8
MIparam,MIDTList-method
        *(*MIDTList-class*)*, 8
missingIndv *(*MIDTList-class*)*, 8
missingIndv,MIDTList-method
        *(*MIDTList-class*)*, 8
missPattern, 12
missRows *(*missRows-package*)*, 2
missRows-package, 2
MultiAssayExperiment, *8*, *9*
MultiAssayExperiment-class, *9*, *10*
MultiAssayExperiment-methods, *10*

NCI60, 13

plotInd, *12*, 15, *18*
plotVar, *12*, *16*, 16
points, *17*
print.missPattern *(*missPattern*)*, 12
print.tuneM *(*tuneM*)*, 20

RVcoeff, 18

searchsComb, 19

STATIS, *5*, *9*, 20
statis, *20*
strata *(*MIDTList-class*)*, 8
strata,MIDTList-method
*(*MIDTList-class*)*, 8
svd, *3*, *22*
svd.triplet, *22*

tuneM, *12*, *18*, 20

wrapperSVD, 22