Package ‘InPosition’

April 15, 2025

Type Package

Title Inference Tests for ExPosition

Version 1.0.0

Date 2025-04-08

Description Non-parametric resampling-based inference tests for ExPosition.

License GPL-2

Encoding UTF-8

Depends prettyGraphs (>= 2.2.0), ExPosition (>= 2.11.0)

BugReports `https://github.com/derekbeaton/ExPosition1/issues`

RoxygenNote 7.3.2

NeedsCompilation no

Author Derek Beaton [aut, cre],
Joseph Dunlop [aut],
Herve Abdi [aut]

Maintainer Derek Beaton <`exposition.software@gmail.com`>

Repository CRAN

Date/Publication 2025-04-15 04:20:05 UTC

Contents

InPosition-package 2
boot.compute.fj 3
boot.ratio.test 4
boot.samples 5
caChiTest 6
contingency.data.break 7
continueResampling 8
epCA.inference.battery 9
epMCA.inference.battery 10
epPCA.inference.battery 12
inGraphs 14

print.epCA.inference.battery . . . 16
print.epMCA.inference.battery . . . 16
print.epPCA.inference.battery . . . 17
print.inpoBoot . . . 17
print.inpoBootTests . . . 18
print.inpoComponents . . . 18
print.inpoOmni . . . 19
print.inpoOutput . . . 19
rebuildMCAtable . . . 20

Index **21**

`InPosition-package` *InPosition: Inference Tests for* Exploratory *Analysis with the Singular Value Decom*Position *(*`ExPosition`*).*

Description

InPosition provides multiple forms of inference tests for the `ExPosition` package.

Author(s)

Questions, comments, compliments, and complaints go to Derek Beaton `<exposition.software@gmail.com>`.
Also see the bug-tracking and live update website for ExPosition: https://github.com/derekbeaton/ExPosition1

Primary authors and contributors are: Derek Beaton, Joseph Dunlop, and Hervé Abdi

References

Permutation:
Berry, K. J., Johnston, J. E., & Mielke, P. W. (2011). Permutation methods. *Wiley Interdisciplinary Reviews: Computational Statistics*,*3*, 527–542.
Peres-Neto, P. R., Jackson, D. A., & Somers, K. M. (2005). How many principal components? Stopping rules for determining the number of non-trivial axes revisited. *Computational Statistics & Data Analysis*, *49*(4), 974-997.

Bootstrap:
Chernick, M. R. (2008). *Bootstrap methods: A guide for practitioners and researchers* (Vol. 619). Wiley-Interscience.
Hesterberg, T. (2011). Bootstrap. *Wiley Interdisciplinary Reviews: Computational Statistics*, *3*, 497–526.

See Also

`epPCA.inference.battery`, `epCA.inference.battery`, `epMCA.inference.battery`. There are no inference tests for MDS at this time. We recommend PCA for inference instead of MDS (some MDS inference tests require the rectangluar table, not the distances, so it is easier to just use PCA).

See also `inGraphs` for graphing and `caChiTest` for an alternate to resampling methods for Correspondence Analysis.

`boot.compute.fj` *Compute bootstrap resampled* `fj` *as supplemental elements.*

Description

This function computes a bootstrap resampled set of data and projects `fj` as supplemental elements.

Usage

```
boot.compute.fj(DATA, res, DESIGN = NULL, constrained = FALSE)
```

Arguments

`DATA`	The original data matrix to be bootstrapped. Rows will be bootstrapped and are assumed to be observations.
`res`	of class `expoOutput`. Results from one of the `ExPosition` methods (e.g., `epPCA`, `epMCA`),
`DESIGN`	A design matrix (in disjunctive coding). Only used if `constrained` is TRUE.
`constrained`	a boolean. If TRUE, bootstrap resampling will occur within groups as designated by the `DESIGN` matrix.

Value

`fjj`	a set of factor scores of the measures (columns, `fj`) for the bootstrapped data.

Author(s)

Derek Beaton

References

Chernick, M. R. (2008). *Bootstrap methods: A guide for practitioners and researchers* (Vol. 619). Wiley-Interscience.
Hesterberg, T. (2011). Bootstrap. *Wiley Interdisciplinary Reviews: Computational Statistics*, *3*, 497–526.

See Also

See the functions `supplementaryCols` and `link{boot.samples}`

Examples

```
##the following code generates 100 bootstrap resampled
##projections of the measures from the Iris data set.
data(ep.iris)
data <- ep.iris$data
design <- ep.iris$design
iris.pca <- epPCA(data,scale="SS1",DESIGN=design,make_design_nominal=FALSE)
boot.fjs.unconstrained <- array(0,dim=c(dim(iris.pca$ExPosition.Data$fj),100))
boot.fjs.constrained <- array(0,dim=c(dim(iris.pca$ExPosition.Data$fj),100))
for(i in 1:100){
#unconstrained means we resample any of the 150 flowers
boot.fjs.unconstrained[,,i] <- boot.compute.fj(ep.iris$data,iris.pca)
#constrained resamples within each of the 3 groups
boot.fjs.constrained[,,i] <- boot.compute.fj(data,iris.pca,design,TRUE)
}
```

`boot.ratio.test` *Performs bootstrap ratio test.*

Description

Performs bootstrap ratio test which is analogous to a *t*- or *z*-score.

Usage

```
boot.ratio.test(boot.cube, critical.value = 2)
```

Arguments

`boot.cube`	an `array`. This is the bootstrap resampled data. dim 1 (rows) are the items to be tested (e.g., `fj`, see `boot.compute.fj`). dim 2 (columns) are the components from the supplemental projection. dim 3 (depth) are each bootstrap sample.
`critical.value`	numeric. This is the value that would be used as a cutoff in a *t*- or *z*-test. Default is 2 (i.e., 1.96 rounded up). The higher the number, the more difficult to reject the null.

Value

A list with the following items:
return(list(sig.boot.ratios=significant.boot.ratios,boot.ratios=boot.ratios,critical.value=critical.value))

`sig.boot.ratios`	This is a matrix with the same number of rows and columns as `boot.cube`. If TRUE, the bootstrap ratio was larger than `critical.value`. If FALSE, it was smaller.
`boot.ratios`	This is a matrix with bootstrap ratio values that has the same number of rows and columns as `boot.cube`.
`critical.value`	the critical value input is also returned.

Author(s)

Derek Beaton and Hervé Abdi

References

The name bootstrap ratio comes from the Partial Least Squares in Neuroimaging literature. See:
McIntosh, A. R., & Lobaugh, N. J. (2004). Partial least squares analysis of neuroimaging data: applications and advances. *Neuroimage*, *23*, S250–S263.

The bootstrap ratio is related to other tests of values with respect to the bootstrap distribution, such as the Interval-*t*. See:
Chernick, M. R. (2008). *Bootstrap methods: A guide for practitioners and researchers* (Vol. 619). Wiley-Interscience.
Hesterberg, T. (2011). Bootstrap. *Wiley Interdisciplinary Reviews: Computational Statistics*, *3*, 497–526.

See Also

`boot.compute.fj`

Examples

```
##the following code generates 100 bootstrap resampled
##projections of the measures from the Iris data set.
data(ep.iris)
data <- ep.iris$data
design <- ep.iris$design
iris.pca <- epPCA(data,scale="SS1",DESIGN=design,make_design_nominal=FALSE)
boot.fjs.unconstrained <- array(0,dim=c(dim(iris.pca$ExPosition.Data$fj),100))
boot.fjs.constrained <- array(0,dim=c(dim(iris.pca$ExPosition.Data$fj),100))
for(i in 1:100){
#unconstrained means we resample any of the 150 flowers
boot.fjs.unconstrained[,,i] <- boot.compute.fj(ep.iris$data,iris.pca)
#constrained resamples within each of the 3 groups
boot.fjs.constrained[,,i] <- boot.compute.fj(data,iris.pca,design,TRUE)
}
#now compute the bootstrap ratios:
ratios.unconstrained <- boot.ratio.test(boot.fjs.unconstrained)
ratios.constrained <- boot.ratio.test(boot.fjs.constrained)
```

`boot.samples` *Compute indicies for bootstrap resampling.*

Description

This function computes a set of indicies for bootstrap resampling. It can be unconstrained or bootstrap within a group design.

Usage

```
boot.samples(DATA, DESIGN = NULL, constrained = FALSE)
```

Arguments

`DATA`	The original data matrix to be bootstrapped. Rows will be bootstrapped and are assumed to be observations.
`DESIGN`	A design matrix (in disjunctive coding). Only used if `constrained` is TRUE.
`constrained`	a boolean. If TRUE, bootstrap resampling will occur within groups as designated by the `DESIGN` matrix.

Value

a set of indicies to be used to be used as the bootstrap resampled indices.

Author(s)

Derek Beaton

See Also

`boot.compute.fj` and `boot.ratio.test`

Examples

```
data(ep.iris)
unconstrained.indices <- boot.samples(ep.iris$data)
#ep.iris$data[unconstrained.indices,]
constrained.indices <- boot.samples(ep.iris$data,DESIGN=ep.iris$design,constrained=TRUE)
#ep.iris$data[constrained.indices,]
```

`caChiTest` *caChiTest: correspondence analysis tests without resampling.*

Description

caChiTest performs 3 sets of chi-square tests along the lines of Lebart's v-tests. These tests are designed to be conservative estimates of chi-square tests on contingency data. The tests treat this data in a standard chi-square framework, but are helpful to understand correspondence analysis data when permutation and bootstrap become unfeasible.

Usage

```
caChiTest(DATA, res, critical.value = 2)
```

Arguments

`DATA`	Data as would be entered for Correspondence Analysis (see `link{epCA}`)
`res`	Results from correspondence analysis (e.g., output from `link{epCA}`).
`critical.value`	numeric. A value, analogous to a z- or t-score to be used to determine significance (via bootstrap ratio).

Value

a list with the following values:

`j.sig.vals`	boolean matrix. Identifies which column items are significant (based on `critical.value`).
`j.signed.vals`	chi-square values associated to column items, multiplied by the sign of their component scores ($fj).
`j.p.vals`	p values associated to column items in a chi-square test.
`i.sig.vals`	boolean matrix. Identifies which row items are significant (based on `critical.value`).
`i.signed.vals`	chi-square values associated to row items, multiplied by the sign of their component scores ($fi).
`i.p.vals`	p values associated to row items in a chi-square test.
`omni.val`	chi-square value associated to the table.
`omni.p`	p value associated to a chi-square tests of the table.

Author(s)

Derek Beaton

See Also

`epCA.inference.battery`

`contingency.data.break`

Bootstrap or permutation resampling for contingency tables

Description

Bootstrap or permutation resampling for contingency tables. More specifically, for correspondence analysis (`epCA`).

Usage

```
contingency.data.break(DATA, boot = FALSE)
```

Arguments

`DATA`	A contingency table to resample.
`boot`	a boolean. If TRUE, use bootstrap (resample with replacement) resampling. If FALSE, use permutation (resample with no replacement).

Value

A resampled contingency table.

Author(s)

Joseph Dunlop and Derek Beaton

See Also

`epCA`, `epCA.inference.battery`

Examples

```
data(authors)
boot.authors <- contingency.data.break(authors$ca$data,boot=TRUE)
perm.authors <- contingency.data.break(authors$ca$data)
```

`continueResampling` *A stopping mechanism if resampling will take too long.*

Description

This function asks the user if they want to continue with resampling if the total time for resampling takes more than 10 minutes. It also provides an estimate of how long resampling takes. This function is required for `InPosition` and `TInPosition` and we do not recommend others use it.

Usage

```
continueResampling(cycle.time)
```

Arguments

`cycle.time`	Is the subtraction of two calls to `proc.time`.

Note

If computation time is expected to take more than 10 minutes and `interactive()` is TRUE, this asks the user if they would like to continue. If 'Y', looping continues. If 'N', it stops.

If computation time is expected to takre more than 10 minutes and `interactive()` is FALSE, the function will proceed as is and perform inference tests.

A progress bar is provided so the user can see how long the tests will take.

See inference battery functions for details.

Author(s)

Derek Beaton

`epCA.inference.battery` *epCA.inference.battery: Inference tests for Correspondence Analysis (CA) via InPosition.*

Description

Correspondence Analysis (CA) and a battery of inference tests via InPosition. The battery includes permutation and bootstrap tests.

Usage

```
epCA.inference.battery(
  DATA,
  DESIGN = NULL,
  make_design_nominal = TRUE,
  masses = NULL,
  weights = NULL,
  hellinger = FALSE,
  symmetric = TRUE,
  graphs = TRUE,
  k = 0,
  test.iters = 100,
  critical.value = 2
)
```

Arguments

`DATA`	original data to perform a CA on.
`DESIGN`	a design matrix to indicate if rows belong to groups.
`make_design_nominal`	a boolean. If TRUE (default), DESIGN is a vector that indicates groups (and will be dummy-coded). If FALSE, DESIGN is a dummy-coded matrix.
`masses`	a diagonal matrix or column-vector of masses for the row items.
`weights`	a diagonal matrix or column-vector of weights for the column it
`hellinger`	a boolean. If FALSE (default), Chi-square distance will be used. If TRUE, Hellinger distance will be used.
`symmetric`	a boolean. If TRUE (default) symmetric factor scores for rows and columns are computed. If FALSE, the simplex (column-based) will be returned.
`graphs`	a boolean. If TRUE (default), graphs and plots are provided (via `epGraphs`)
`k`	number of components to return.

`test.iters`	number of iterations
`critical.value`	numeric. A value, analogous to a z- or t-score to be used to determine significance (via bootstrap ratio).

Details

`epCA.inference.battery` performs correspondence analysis and inference tests on a data matrix.

If the expected time to compute the results (based on `test.iters`) exceeds 1 minute, you will be asked (via command line) if you want to continue.

Value

Returns two lists ($Fixed.Data and $Inference.Data). For $Fixed.Data, see `epCA`, `coreCA` for details on the descriptive (fixed-effects) results.

$Inference.Data returns:

`components`	Permutation tests of components. p-values ($p.vals) and distributions of eigenvalues ($eigs.perm) for each component
`fj.boots`	Bootstrap tests of measures (columns). See `boot.ratio.test` output details.
`omni`	Permutation tests of components. p-values ($p.val) and distributions of total inertia ($inertia.perm)

Author(s)

Derek Beaton, Joseph Dunlop, and Hervé Abdi.

See Also

`epCA`, `epMCA`, `epMCA.inference.battery`, `caChiTest`

Examples

```
##warning: this example takes a while to compute. This is why it is reduced.
data(authors)
ca.authors.res <- epCA.inference.battery(authors$ca$data/100)
```

`epMCA.inference.battery`

epMCA.inference.battery: Inference tests for Multiple Correspondence Analysis (CA) via InPosition.

Description

Multiple Correspondence Analysis (CA) and a battery of inference tests via InPosition. The battery includes permutation and bootstrap tests.

Usage

```
epMCA.inference.battery(
  DATA,
  make_data_nominal = TRUE,
  DESIGN = NULL,
  make_design_nominal = TRUE,
  masses = NULL,
  weights = NULL,
  hellinger = FALSE,
  symmetric = TRUE,
  correction = c(”b”),
  graphs = TRUE,
  k = 0,
  test.iters = 100,
  constrained = FALSE,
  critical.value = 2
)
```

Arguments

`DATA`	original data to perform a MCA on. This data can be in original formatting (qualitative levels) or in dummy-coded variables.
`make_data_nominal`	a boolean. If TRUE (default), DATA is recoded as a dummy-coded matrix. If FALSE, DATA is a dummy-coded matrix.
`DESIGN`	a design matrix to indicate if rows belong to groups.
`make_design_nominal`	a boolean. If TRUE (default), DESIGN is a vector that indicates groups (and will be dummy-coded). If FALSE, DESIGN is a dummy-coded matrix.
`masses`	a diagonal matrix or column-vector of masses for the row items.
`weights`	a diagonal matrix or column-vector of weights for the column it
`hellinger`	a boolean. If FALSE (default), Chi-square distance will be used. If TRUE, Hellinger distance will be used.
`symmetric`	a boolean. If TRUE symmetric factor scores for rows.
`correction`	which corrections should be applied? "b" = Benzécri correction, "bg" = Greenacre adjustment to Benzécri correction.
`graphs`	a boolean. If TRUE (default), graphs and plots are provided (via `epGraphs`)
`k`	number of components to return.
`test.iters`	number of iterations
`constrained`	a boolean. If a DESIGN matrix is used, this will constrain bootstrap resampling to be within groups.
`critical.value`	numeric. A value, analogous to a z- or t-score to be used to determine significance (via bootstrap ratio).

Details

`epMCA.inference.battery` performs multiple correspondence analysis and inference tests on a data matrix.

If the expected time to compute the results (based on `test.iters`) exceeds 1 minute, you will be asked (via command line) if you want to continue.

Value

Returns two lists ($Fixed.Data and $Inference.Data). For $Fixed.Data, see `epMCA`, `coreCA` for details on the descriptive (fixed-effects) results.

$Inference.Data returns:

`components`	Permutation tests of components. p-values ($p.vals) and distributions of eigenvalues ($eigs.perm) for each component
`fj.boots`	Bootstrap tests of measures (columns). See `boot.ratio.test` output details.
`omni`	Permutation tests of components. p-values ($p.val) and distributions of total inertia ($inertia.perm). This is only useful if `corrections` are performed. Total inertia is constant for permutation with no corrections in MCA.

Author(s)

Derek Beaton, Joseph Dunlop, and Hervé Abdi.

See Also

`epMCA`, `epCA`, `epCA.inference.battery`

Examples

```
data(mca.wine)
mca.wine.res <- epMCA.inference.battery(mca.wine$data)
```

`epPCA.inference.battery`

epPCA.inference.battery: Inference tests for Principal Component Analysis (PCA) via InPosition.

Description

Principal Component Analysis (PCA) and a battery of inference tests via InPosition. The battery includes permutation and bootstrap tests.

Usage

```
epPCA.inference.battery(
  DATA,
  scale = TRUE,
  center = TRUE,
  DESIGN = NULL,
  make_design_nominal = TRUE,
  graphs = TRUE,
  k = 0,
  test.iters = 100,
  constrained = FALSE,
  critical.value = 2
)
```

Arguments

`DATA`	original data to perform a PCA on.
`scale`	a boolean, vector, or string. See `expo.scale` for details.
`center`	a boolean, vector, or string. See `expo.scale` for details.
`DESIGN`	a design matrix to indicate if rows belong to groups.
`make_design_nominal`	a boolean. If TRUE (default), DESIGN is a vector that indicates groups (and will be dummy-coded). If FALSE, DESIGN is a dummy-coded matrix.
`graphs`	a boolean. If TRUE (default), graphs and plots are provided (via `epGraphs`)
`k`	number of components to return.
`test.iters`	number of iterations
`constrained`	a boolean. If a DESIGN matrix is used, this will constrain bootstrap resampling to be within groups.
`critical.value`	numeric. A value, analogous to a z- or t-score to be used to determine significance (via bootstrap ratio).

Details

`epPCA.inference.battery` performs principal components analysis and inference tests on a data matrix.

If the expected time to compute the results (based on `test.iters`) exceeds 1 minute, you will be asked (via command line) if you want to continue.

Value

Returns two lists ($Fixed.Data and $Inference.Data). For $Fixed.Data, see `epPCA`, `corePCA` for details on the descriptive (fixed-effects) results.

$Inference.Data returns:

`components`	Permutation tests of components. p-values ($p.vals) and distributions of eigenvalues ($eigs.perm) for each component
`fj.boots`	Bootstrap tests of measures (columns). See `boot.ratio.test` output details.

Author(s)

Derek Beaton and Hervé Abdi.

See Also

`epPCA`

Examples

```
data(words)
pca.words.res <- epPCA.inference.battery(words$data)
```

`inGraphs` *inGraphs: InPosition plotting function*

Description

InPosition plotting function which is an interface to `prettyGraphs`.

Usage

```
inGraphs(
  res,
  DESIGN = NULL,
  x_axis = 1,
  y_axis = 2,
  inference.info = NULL,
  color.by.boots = TRUE,
  boot.cols = c("plum4", "darkseagreen", "firebrick3"),
  fi.col = NULL,
  fi.pch = NULL,
  fj.col = NULL,
  fj.pch = NULL,
  col.offset = NULL,
  constraints = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  bootstrapBars = TRUE,
  correlationPlotter = TRUE
)
```

Arguments

`res`	results from InPosition or ExPosition. If results are from ExPosition, `inference.info` must be included.
`DESIGN`	A design matrix to apply colors (by pallete selection) to row items
`x_axis`	which component should be on the x axis?
`y_axis`	which component should be on the y axis?
`inference.info`	Inference data as output by InPosition (of class inpoOutput).
`color.by.boots`	a boolean. If TRUE, items are colored by bootstrap ratio test. Items larger than `critical.value` are colored 'plum4' on the horizontal component, 'darkseagreen' on the vertical component, or 'firebrick3' if the item is significant on both components (to be visualized). If FALSE, the color of the items will be used.
`boot.cols`	vector of colors: `c(horizontal component color,vertical component color, color when item is significant on both)`.
`fi.col`	A matrix of colors for the row items. If NULL, colors will be selected.
`fi.pch`	A matrix of pch values for the row items. If NULL, pch values are all 21.
`fj.col`	A matrix of colors for the column items. If NULL, colors will be selected.
`fj.pch`	A matrix of pch values for the column items. If NULL, pch values are all 21.
`col.offset`	A numeric offset value. Is passed to `createColorVectorsByDesign`.
`constraints`	Plot constraints as returned from `prettyPlot`. If NULL, constraints are selected.
`xlab`	x axis label
`ylab`	y axis label
`main`	main label for the graph window
`bootstrapBars`	a boolean. If TRUE (default), bootstrap ratio bar plots will be created.
`correlationPlotter`	a boolean. If TRUE (default), a correlation circle plot will be created. Applies to PCA family of methods (CA is excluded for now).

Value

Currently, nothing is returned. This function, for now, works as a visualizer for inference tests. Colors and constraints come from the descriptive (fixed effects) analysis.

Author(s)

Derek Beaton

See Also

`epGraphs`

Examples

```
data(ep.iris)
data<-ep.iris$data
design<-ep.iris$design
pca.iris.res <- epPCA.inference.battery(data,DESIGN=design,make_design_nominal=FALSE)
inGraphs(pca.iris.res,y_axis=3)
```

`print.epCA.inference.battery` *Print Correspondence Analysis (CA) Inference results*

Description

Print Correspondence Analysis (CA) Inference results.

Usage

```
## S3 method for class 'epCA.inference.battery'
print(x, ...)
```

Arguments

`x`	an list that contains items to make into the epCA.inference.battery class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.epMCA.inference.battery` *Print Multiple Correspondence Analysis (MCA) Inference results*

Description

Print Multiple Correspondence Analysis (MCA) Inference results.

Usage

```
## S3 method for class 'epMCA.inference.battery'
print(x, ...)
```

Arguments

`x`	an list that contains items to make into the epMCA.inference.battery class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.epPCA.inference.battery` *Print Principal Components Analysis (PCA) Inference results*

Description

Print Principal Components Analysis (PCA) Inference results.

Usage

```
## S3 method for class 'epPCA.inference.battery'
print(x, ...)
```

Arguments

`x`	an list that contains items to make into the epPCA.inference.battery class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.inpoBoot` *Print results from InPosition Bootstraps*

Description

Print bootstrap results from the InPosition.

Usage

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

Arguments

`x`	an list that contains items to make into the inpoBoot class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.inpoBootTests` *Print results from InPosition Bootstrap Ratio Tests*

Description

Print bootstrap ratio tests results from the InPosition.

Usage

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

Arguments

`x`	an list that contains items to make into the inpoBootTests class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.inpoComponents` *Print results from InPosition Components Permutation Test*

Description

Print Components permutation test results from the inposition.

Usage

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

Arguments

`x`	an list that contains items to make into the inpoComponents class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.inpoOmni` *Print results from InPosition Omnibus Permutation Test*

Description

Print Omnibus permutation test results from the inposition.

Usage

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

Arguments

`x`	an list that contains items to make into the inpoOmni class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.inpoOutput` *Print results from InPosition*

Description

Print results from the InPosition.

Usage

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

Arguments

`x`	an list that contains items to make into the inpoOutput class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

See Also

`epPCA.inference.battery`, `inGraphs`

`rebuildMCAtable` *rebuildMCAtable: rebuild categorical table from the disjunctive table.*

Description

rebuildMCAtable takes the disjunctive table used in MCA and rebuilds a categorical form of it. This function is used for permutation tests when only a disjunctive table is available.

Usage

```
rebuildMCAtable(DATA)
```

Arguments

`DATA`	Disjunctive coded data table

Value

A categorical data table is returned. It has the same structure as the disjunctive table in a format that can be permuted.

Author(s)

Derek Beaton

Index

∗ **analysis**
  caChiTest, 6
∗ **bootstrap**
  boot.compute.fj, 3
  boot.ratio.test, 4
  boot.samples, 5
  contingency.data.break, 7
  epCA.inference.battery, 9
  epMCA.inference.battery, 10
  epPCA.inference.battery, 12
  inGraphs, 14
∗ **correspondence**
  caChiTest, 6
∗ **graphs**
  inGraphs, 14
∗ **inference**
  boot.compute.fj, 3
  boot.ratio.test, 4
  caChiTest, 6
∗ **misc**
  caChiTest, 6
  inGraphs, 14
  rebuildMCAtable, 20
∗ **multivariate**
  boot.compute.fj, 3
  boot.ratio.test, 4
  caChiTest, 6
  epCA.inference.battery, 9
  epMCA.inference.battery, 10
  epPCA.inference.battery, 12
  inGraphs, 14
  InPosition-package, 2
  rebuildMCAtable, 20
∗ **package**
  InPosition-package, 2
∗ **permutation**
  contingency.data.break, 7
  epCA.inference.battery, 9
  epMCA.inference.battery, 10
  epPCA.inference.battery, 12
  inGraphs, 14
∗ **print**
  print.epCA.inference.battery, 16
  print.epMCA.inference.battery, 16
  print.epPCA.inference.battery, 17
  print.inpoBoot, 17
  print.inpoBootTests, 18
  print.inpoComponents, 18
  print.inpoOmni, 19
  print.inpoOutput, 19

array, *4*

boot.compute.fj, 3, *4–6*
boot.ratio.test, 4, *6*, *10*, *12*, *14*
boot.samples, 5

caChiTest, *3*, 6, *10*
contingency.data.break, 7
continueResampling, 8
coreCA, *10*, *12*
corePCA, *13*
createColorVectorsByDesign, *15*

epCA, *7, 8*, *10*, *12*
epCA.inference.battery, *2*, *7, 8*, 9, *12*
epGraphs, *9*, *11*, *13*, *15*
epMCA, *3*, *10*, *12*
epMCA.inference.battery, *2*, *10*, 10
epPCA, *3*, *13*, *14*
epPCA.inference.battery, *2*, 12, *19*
expo.scale, *13*
ExPosition, *2, 3*

inGraphs, *3*, 14, *19*
InPosition *(*InPosition-package*)*, 2
InPosition-package, 2

prettyGraphs, *14*
prettyPlot, *15*

`print.epCA.inference.battery`, 16
`print.epMCA.inference.battery`, 16
`print.epPCA.inference.battery`, 17
`print.inpoBoot`, 17
`print.inpoBootTests`, 18
`print.inpoComponents`, 18
`print.inpoOmni`, 19
`print.inpoOutput`, 19

`rebuildMCAtable`, 20

`supplementaryCols`, *3*