Package 'SOUP'

April 4, 2015

Version 1.1

Date 2015-04-03

Title Stochastic Ordering Using Permutations (and Pairwise Comparisons)

Author Federico Mattiello [aut, cre]

Maintainer Federico Mattiello <federico.mattiello@gmail.com>

Depends R (>= 3.0.0), methods, tensor

Suggests flip

Description Construct a ranking of a set of treatments/groups by gathering together information coming from several response variables. It can be used with both balanced and unbalanced experiments (with almost all test statistics) as well as in presence of either continuous covariates or a stratifying (categorical) variable.

License GPL (>= 2)

Collate 'NPC.R' 'PermSpace.R' 'RankResults.R' 'SOUP.R' 'multiplicity.R' 'simpleAD.R' 'simpleHotelling.R' 'simpleMeanDiff.R' 'simpleTtest.R' 'strataAD.R' 'strataLmCoef.R' 'strataMeanDiff.R' 'strataTtest.R' 't2p.R' 'utilities.R' 'PValueMat.R' 'SoupObject.R' 'rankingRule.R' 'iterNPC.r'

NeedsCompilation no

Repository CRAN

Date/Publication 2015-04-04 16:14:05

R topics documented:

`.DesM` *Design Matrix For Pairwise Differences*

Description

Construct Design Matrix that pre-multiplied to the dataset gives the pairwise mean differences (wrt the groups)

Usage

```
.DesM(N)
```

Arguments

`N`	number of replication of the experiment for each group, either an `integer` vector or a `table` of length `G` *i.e.* number of groups/treatments

Value

matrix of

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

.dummyze *From Factor To Dummy Variables*

Description

Transform a `factor` or a factor-like vector of `character` into a matrix of dichotomous dummy variables

Usage

```
.dummyze(x)
```

Arguments

`x`	`factor`, `character` or `integer` to be *dummyzed*

Value

a matrix of {-1, 0, 1} of dimension `length(x)` $\times$ `G` where `G` is the number of levels (distinct values) of the factor (vector)

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

.makePermSpaceID *Constructs Row Indices Permutation Space*

Description

Constructs the permutation space of row indices depending on the type of analysis

Usage

```
.makePermSpaceID(nObs, analysisType, strata, seed,
  nPerms)
```

Arguments

`nObs`	`integer` total number of observations
`analysisType`	`character` type of the analysis to be performed
`strata`	`character` vector or `factor` containing the covarite used for stratification; if `analysisType` is `"strata"` then this parameter is provided
`seed`	optional `integer` seed for the `RNG`
`nPerms`	`integer` number of permutations to be performed

Value

either a `matrix` in the case of `"simple"` `analysisType` or a 3-ways `array`, containing the permuted row indices

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

`.makePValueMat` *Construct Univariate* p-*Values Matrix*

Description

Take as input the 3-ways array of *p*-values and return the $G \times G \times V$ matrix of observed *p*-values adjusted for multiplicity; G, V are, respectively, the number of groups/treatments and the number of variables.

Usage

```
.makePValueMat(P, multAdjMethod, groupsLabs)
```

Arguments

`P`	the 3-ways `array` containing the *p*-values for each pairwise comparison in each variable
`multAdjMethod`	the `character` string indicating which multiplicity correction must be used
`groupsLabs`	`character` vector containing the groups' labels

Value

an object of the class `PValueMat`

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

.orthoX — *Residualises the response variables* w.r.t. *the matrix of covariates like in the linear models, therefore projecting on the space $I - H$ where H is the* hat *matrix of X*

Description

Residualises the response variables *w.r.t.* the matrix of covariates like in the linear models, therefore projecting on the space $I - H$ where H is the *hat* matrix of X

Usage

```
.orthoX(Y, X)
```

Arguments

`Y`	the `matrix` or `data.frame` of response variables
`X`	the `matrix` of covariates

Value

the residualised `Y` `matrix`

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

as.list — *Methods for function* `as.list` *in package* **SOUP**

Description

Methods for function `as.list` in package **SOUP**

Methods

`signature(object = "RankResults")` conversion of the object (`RankResults`) into a `list`

BHS *Multiplicity adjustment by Bonferroni-Holm-Shaffer's rule*

Description

Multiplicity adjustment by Bonferroni-Holm-Shaffer's rule

Usage

```
BHS(pValues)
```

Arguments

pValues	numeric vector of *p*-vaules

Value

numeric vector of corrected *p*-vaules

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

References

Shaffer J.P. (1986) Modified Sequentially Rejective Multiple Test Procedures, *Journal of the American Statistical Association*, **81**, 826–831.

See Also

p.adjust, p.adjust.methods

Examples

```
set.seed(123)
p.raw <- runif(10, max = 0.2)
rbind(p.raw, p.adj = SOUP::BHS(p.raw))
```

FWEminP *FWE Adjustment Using Permutation*

Description

FWE Adjustment Using Permutation and NPC

Usage

```
FWEminP(Pmat)
```

Arguments

Pmat	matrix of *p*-values where comparisons are on the columns

Details

Multiplicity correction controlling the Family-Wise Error using the permutation *p*-values and Non-Parametric Combination with *minP* as combining function.

Value

numeric vector of corrected p.values

Author(s)

Dario Basso and Federico Mattiello <federico.mattiello@gmail.com>

References

Pesarin, F. and Salmaso, L. (2010) *Permutation Tests for Complex Data*. Wiley: United Kingdom

Finos, L. and Pesarin, F. and Salmaso, L. (2003) Test combinati per il controllo della molteplicit\'a mediante procedure di closed testing, *Statistica Applicata*, **15**, 301–329.

See Also

p.adjust, p.adjust.methods

Examples

```
set.seed(123)
P <- matrix(runif(1010), nrow = 101, ncol = 10,
  dimnames = list(c("p-obs", paste("p-*", 1L:100)), LETTERS[1L:10]))
P[1L, 1L:4] <- 1/100
FWEminP(P)
```

initialize *Methods for function* initialize *in package* **SOUP***: constructor*

Description

Methods for function initialize in package **SOUP**: constructor

Methods

signature(object = "PermSpace") constructor of the object (initialize)

iterNPC *Iterated NonParametric Combination*

Description

Iterated NonParametric Combination of the test statistic matrix, mostly for internal use.

Usage

```
iterNPC(P, tol = 1, maxIter = 10, plotIt = TRUE,
  combFun1 = function(x) {
      -2 * sum(log(x), na.rm = TRUE) },
  combFun2 = function(x) {
      sum(qnorm(1 - x), na.rm = TRUE) },
  combFun3 = function(x) {     -min(x, na.rm = TRUE) },
  test = c("SSQ", "ABS", "NORM2", "EDF"), Pmat = FALSE,
  onlyCombined = FALSE)
```

Arguments

P	input matrix containing the test statistic in the form of p.values (permutation or asymptotic)
tol	integer representing the desired tolerance, the actual one being $\frac{tol}{B}$ where B is the number of permutations
maxIter	integer maximum number of iterations to be performed, default 10
plotIt	logical, if TRUE (default) plots the diagnostic grahp of p-values obtained with each combining function *vs.* iteration index
combFun1	first combining function needed for the algorithm, default is *Fisher*'s: $-2\sum_i \log p_i$
combFun2	second combining function, default is *Liptak*: $\sum_i \Phi^{-1}(1-p_i)$
combFun3	third combining function, default is *Tippett*: $-\min_i p_i$

`test` `character`, it is the stopping rule used to check for convergence, each one of the 4 kinds currently implemented takes as input the vector with the result of the combination with the different combining functions for one permutation. There are 4 choices for this argument:

- `"SSQ"` Sum of SQuares, the algorithm stops when $\sqrt{(n-1)(s^2)}$ is smaller than the actual tolerance; here where s is the sample variance of the vector.
- `"ABS"` The algorithm stops if not all pairwise absolute differences between the elements are smaller than the actual tolerance
- `"NORM2"` The algorithm stops if the euclidean distance between two consecutive iterations is smaller than the actual tolerance.
- `"EDF"` It is based on the Empirical Distribution Function of the p.values. The algorithm stops if the standardized absolute difference between the average of two consecutive iterations is smaller than the actual tolerance. The standardization involves the variance of the numerator, it is a sort of t-test.

`Pmat` `logical`, if `TRUE` returns the final matrix of combined p-values, default is `FALSE`

`onlyCombined` `logical`, if `TRUE` returns only the first column of the final matrix, in case only the distribution of combined p-values is needed

Details

It takes as input a matrix whose columns have to be combined with the iterated NPC procedure, default combining functions are 3: Fisher, Liptak (normal version), and Tippett (minP)

Value

The output is conditioned on some of the input argument. The default is a `list` containing only the element `"P.iter"`: a `matrix` with 3 columns containing the observed p.values across iterations and for all combining functions (to manually check convergence). If `Pmat` is `TRUE` than the list contains also the element `"P.final"` that is the final permutation space of p.values obtained with all combining function. If `onlyCombined` is `TRUE` than the resulting output is just the vector containing the first column of `"P.final"`.

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

`NPC` *NonParametric Combination of Test Statistic*

Description

NonParametric Combination of the test statistic matrix, mostly for internal use.

Usage

```
NPC(rawStats, combFun = "Fisher", seed, p.values = FALSE,
  tails = NULL, subsets = NULL, weights = NULL,
  iteratedNPC = FALSE)
```

Arguments

rawStats	3-ways array containing the test statistic computed on all pairwise comparisons (both directions) in all variables, with dimensions $B + 1 \times p \times K$, where B is the number of permutations, p is the number of variables and K is the number of pairwise comparisons.
combFun	character string indicating the combining function to be used, can take as input p-values or the raw test statistic depending on the choice, can be one of "Fisher", "Liptak", "minP", "Tippett", "maxT", "sumT" and "direct"; note that "direct" is equivalent to "sumT" and "Tippett" is equivalent to "minP". "Liptak" use the normal quantile function. See references for details.
seed	integer seed for the RNG (random number generator), taken from the input for reproducibility purposes
p.values	logical, if TRUE means that the input matrix rawStats contains p-values rather than raw test statistics and so has to be treated differently, *i.e.* the first passage from test statistics to p-values is omitted (function t2p)
tails	integer vector of ± 1 containing the alternatives for response variables: +1 means "the higher the better", -1 means "the lower the better" (direction of preference), if NULL (default) all variables are considered to be of the type "the higher the better'
subsets	integer, character or list where each element contains the subset of column indices that need to be treated separately, if NULL (default) all input variables are considered
weights	integer, character or list where each element contains the weights of the variables, one for each subset, if NULL (default) variables are treated equally *i.e.* all have the same weight
iteratedNPC	logical, if TRUE it performs the iterated combination procedure running function iterNPC and the choice of comb.funct becomes thus irrelevant; otherwise just perform the requested NPC.

Details

It takes as input the 3-ways matrix containing the raw test statistics and perform the NPC with the possibility to add sets of weights for weighting variables differently, and/or to select subsets of variables to which NPC has to be applied separately. Note that weights and subsets are placed in the "..." argument of the **SOUP** function call hence they are not documented in the **SOUP** help.

Value

an object of class PermSpace or a list containing PermSpace objects, in the case of multiple weights and/or subsets.

Note

This function is mainly taken from the function npc in the package flip.

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

References

Pesarin, F. and Salmaso, L. (2010) *Permutation Tests for Complex Data*. Wiley: United Kingdom

Pesarin F. (2001) *Multivariate Permutation Tests with Applications in Biostatistics*. Wiley: New York.

See Also

t2p

PermSpace *Class* PermSpace

Description

Permutation Space

Methods for function plot in package **SOUP**

Details

Contains the permutation space of the test statistic, useful for reproducibility of the analyses

Objects from the Class

Objects can be created by calls of the form new("PermSpace", ...). It contains information of permutation spaces used in the analysis, the combined test statistics and p-values, IDs (row indexes) and the seed for the RNG, the rawStats (non-combined test statistics) and comb.funct (the nonparametric combining function). But objects of the class are principally supposed to be created and used internally for storing results of SOUP.

Slots

seed: integer seed for the Random Number Generator

T.H0Low: matrix containing the permutation space of *combined* test statistics with null hypothesis $H_0 : x_i \geq x_h, i < h, i, h = 1, \ldots, G$

T.H0Gre: matrix containing the permutation space of *combined* test statistics with null hypothesis $H_0 : x_i \leq x_h, i < h, i, h = 1, \ldots, G$

`P.H0Low`: `matrix` containing the permutation space of *combined* p-values with null hypothesis $H_0 : x_i \geq x_h, i < h, i, h = 1, \ldots, G$

`P.H0Gre`: `matrix` containing the permutation space of *combined* p-values with null hypothesis $H_0 : x_i \leq x_h, i < h, i, h = 1, \ldots, G$

`IDs`: `matrix` permutation space of row indexes

`rawStats`: 3-way `array` containing the permutation space of *non-combined* test statistics

`comb.funct`: nonparametric combining function used for `NPC` of `rawStats`

Methods

initialize constructor used when calling `new(PermSpace, ...)`

show `signature(object = "PermSpace")`: shows only the main information (on screen) for the object

print `signature(x = "PermSpace")`: It prints the whole object on screen (mostly useful for external saving)

`signature(x = "PermSpace")` Plots a bivariate representation of the permutation space, when there are more than 2 (original) variables then a Principal Component Analysis is performed and the first 2 variables in the transformed space are shown

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

Examples

```
showClass("PermSpace")
```

`print`

`signature(x = "PermSpace")` *It prints the whole object on screen (mostly useful for external saving)*

Description

`signature(x = "PermSpace")` It prints the whole object on screen (mostly useful for external saving)

`signature(x = "PValueMat")` It prints the whole object on screen (mostly useful for external saving)

`signature(x = "SoupObject")` It prints the whole object on screen (mostly useful for external saving)

PValueMat *Class* PValueMat

Description

Univariate P-Value Matrix

Details

Contains the raw *p*-values and the *p*-values adjusted for multiplicity for each pairwise comparison in each variable separately. This allows to see the contribution of each variable and each comparison to the final result

Objects from the Class

Objects can be created by calls of the form `new("PValueMat", ...)` but objects of the class are principally supposed to be created and used internally for storing results of NPC.

Slots

`raw.p.values`**:** 3-ways `array` containing the raw p.values

`adj.p.values`**:** 3-ways `array` containing the p.values adjusted for multiplicity

`p.adj.method`**:** multiplicity adjustment method employed

Prototype

`prototype` class has a $0 \times 0 \times 0$ element for the first two slots and a 0-length `character` string

Methods

show `signature(object = "PValueMat")`: shows only the main information (on screen) for the object

print `signature(x = "PValueMat")`: It prints the whole object on screen (mostly useful for external saving)

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

Examples

```
showClass("PValueMat")
```

rankingRule *Performs the ranking*

Description

Performs the ranking

Usage

```
rankingRule(permSpace, alpha = 0.05, multAdjMethod,
  groupsLabs)
```

Arguments

permSpace	object of the class PermSpace containing the permutation space of the test statistic
alpha	numeric significance level to be employed, in case it is a vector, a ranking is computed for each value of alpha
multAdjMethod	multiplicity adjustment method to be used for the pairwise hypotheses *p*-values
groupsLabs	character vector containing the groups' labels

Value

object of the class RankResults containing results of the ranking procedure and other information

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

RankResults *Ranking Results*

Description

Contains results of the algorithm, if n values of alpha are provided for to the **SOUP** function then there will be n (possibly different) rankings, one for each value of alpha

Objects from the Class

Objects can be created by calls of the form new("RankResults", ...).

Slots

`alpha`: `numeric` vector containing the values of `alpha` used for rejecting each pairwise hypothesis

`ranks`: `matrix` containing the rankings obtained from the ranking algorithm

`p.values`: `matrix` containing the p-values used for the rankings

`p.adj.method`: `character` string indicating which multiplicity adjustment method was used for the pairwise p-values

Prototype

`prototype` class has a $0 \times 0 \times 0$ element for the first two slots and a 0-length `character` string

Methods

show `signature(object = "RankResults")`: shows only the main information (on screen) for the object

print `signature(x = "RankResults")`: It prints the whole object on screen (mostly useful for external saving)

as.list Coerce the `RankResults` object to a list of `RankResults` objects

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

Examples

```
showClass("RankResults")
```

---

show

`signature(object = "PermSpace")` *Shows only the main information (on screen) for the object*

Description

`signature(object = "PermSpace")` Shows only the main information (on screen) for the object

Method show

`signature(object = "PValueMat")` Shows only the main information (on screen) for the object

`signature(object = "SoupObject")` Shows only the main information (on screen) for the object

SOUP *SOUP Main Function*

Description

Main function of the package, interface for every analysis. The dataset can be balanced or not for almost all possible choices of the input parameters. The function allows also for the presence of one or more continuous covariates or for stratified analysis.

Usage

```
SOUP(Y, covars, data = NULL, analysisType, p.adj.method,
  p.valuesType, testStatistic, combFunct,
  univ.p.values = TRUE, tails = NULL,
  linearInter = FALSE, returnPermSpace = TRUE,
  nPerms = 999L, alpha = 0.05, seed, iteratedNPC, ...)
```

Arguments

`Y`	input `matrix` where each column is a response variables.
`covars`	it can be a `matrix`, a `data.frame` or a `formula`, in the first two cases it must contains at least the labels of groups, in the latter case it has to be a right-sided `formula` (*e.g.* `~ v1 + v2`) specifying the model to extract from the `data` input.
`data`	optional `data.frame` containing covariates requested by `covars`, if `covars` is not a formula this input is useless.
`analysisType`	`character`, type of the analysis to be performed: it can be `"simple"` if the only covariate is the labels of groups, `"strata"` if there is also a stratifying (categorical) covariate, `"regres"` if there is one or more (numerical or not) covariate(s) besides labels of groups. In the latter case the linear effect of the covariates is removed from the response variables are residualised by the matrix $V^{-1/2}$ obtained from $V = I - H$ (where I is the identity matrix and H is the "hat" matrix of the OLS, by means of a spectral decomposition.
`p.adj.method`	`character` string containing the type of required p-value adjustment
`p.valuesType`	`character` string indicating the type of p-value to be used, it can be `"permutation"` or `"asymptotic"`
`testStatistic`	`character` string indicating the test statistic to be used, it depends on both `analysisType` and on `p.valuesType` and the alternatives are: `AD, meanDiff` for all `analysisType` but only using `permutation` p-values `Ttest` for all `analysisType` but only using `asymptotic` p-values `Hotelling` with both `permutation` and `asymptotic` emphp-values, with `"simple"` and `"regres"` but not with `"strata"` `analysisType` `lmCoef` only with `"strata"` `analysisType` and with `"asymptotic"` p-values
`combFunct`	`character` string containing the desired combining function to be used, choices are:

`Fisher` the famous Fisher's *p*-values combining function

`Liptak` it uses the quantile function of the Normal distribution to combine *p*-values

`minP, tippett` combine *p*-values by taking the minimum across the set

`maxT` combines directly the test statistics by taking the maximum across the set

`direct, sumT` combine the test statistics by summing them

`sumT2` combines the test statistics by squaring and summing them

See the references for more details about their properties.

`univ.p.values` `logical`, if `TRUE` (default) *p*-values are returned for each variable separately in a 3-ways `array`, the chosen multiplicity correction is performed independently for each variable

`tails` `integer` vector of ± 1 containing the alternatives for response variables: `+1` means "the higher the better", `-1` means "the lower the better" (direction of preference), if `NULL` (default) all variables are considered to be of the type "the higher the better"

`linearInter` `logical`, if `TRUE` the presence of linear interaction is assumed between levels of the stratifying covariate and response variables, this affects only the `"lmCoef"` test statistic in the (in the `"strata"` `analysisType`), basically the contrasts matrix of groups is multiplied by the levels of the stratifying factor.

`returnPermSpace`

`logical` if `TRUE` (default) the whole permutation space is returned, class `PermSpace`, otherwise it is an empty instance of the class.

`nPerms` `integer` number of permutation to be performed

`alpha` `numeric` desired significance level, *i.e.* type-I error

`seed` `integer` seed for the Random Number Generator

`iteratedNPC` `logical`, single or iterated Non-Parametric Combination, see \ codeiterNPC for details.

`...` put here the optional `weights` and `subsets` for the `NPC` function and the permutation space of rows indexes `permSpaceID`. The latter allows to exactly reproduce a previous analysis, if all other inputs are kept equal, or to see what happens changing for example only the `testStatistic`.

Details

Depending on the chosen p-values type and on the analysis type, only some options can be selected:

- with `"simple"` or `"regres"` analysis and `"asymptotic"` *p*-values, `"Hotelling"` and `"Ttest"`; with `permutation` *p*-values `"AD"`, `"Hotelling"` and `"meanDiff"` can be selected.
- With `"strata"` analysis and `"asymptotic"` *p*-values, `"lmCoef"` and `"Ttest"`; with `"permutation"` *p*-values `"AD"` and `"meanDiff"` can be selected.

Value

an object of class `SoupObject`.

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

References

Pesarin, F. and Salmaso, L. (2010) *Permutation Tests for Complex Data*. Wiley: United Kingdom.

Pesarin F. (2001) *Multivariate Permutation Tests with Applications in Biostatistics* Wiley: New York.

Federico Mattiello (2010) *Some resampling-based procedures for ranking of multivariate populations*, Master's Thesis, Faculty of Statistical Sciences: Padova.

Examples

```
###
### testing SOUP
###
rm(list = ls()); gc(reset = TRUE)

require(SOUP)
n <- 5L         # replication of the experiment
G <- 4L         # number of groups
nVar <- 10L     # number of variables
shift <- 1.5    # shift to be added to group 3
alpha <- c(0.01, 0.05, 0.1)        # significance levels

## groups factor
groups <- gl(G, n, labels = paste("gr", seq_len(n), sep = "_"))

set.seed(12345)
Y <- matrix(rnorm(n * G * nVar), nrow = n * G, ncol = nVar)
colnames(Y) <- paste("var", seq_len(nVar), sep = "_")
ind1 <- groups == unique(groups)[3L]
Y[ind1, ] <- Y[ind1, ] + shift

res <- SOUP(Y = Y, covars = as.matrix(groups), analysisType = "simple",
        testStatistic = "meanDiff", combFunct = "Fisher",
        alpha = alpha,
        subsets = list("first" = 1:5, "second" = 6:10),
        weights = list(
                "firstW" = c(.1, .2, .1, .5, .1),
                "secondW" = rep.int(1, 5)),
        p.valuesType = "permutation", p.adj.method = "FWEminP")
res
```

SoupObject *Class* SoupObject

Description

SOUP Main Object

Details

Contains the main results of the **SOUP** analysis

Objects from the Class

Objects can be created by calls of the form `new("SoupObject", ...)`. It contains all results from the analysis performed by the SOUP function.

Slots

`call`: a `call` object that contains the call of the SOUP main function

`pValueMat`: PValueMat object containing univariate *p*-values

`rankResults`: RankResults object containing the results of the ranking

`permSpace`: PermSpace object containing the permutation spaces

Methods

show `signature(object = "SoupObject")`: shows only the main information (on screen) for the object

print `signature(x = "SoupObject")`: It prints the whole object on screen (mostly useful for external saving)

Author(s)

Federico Mattiello <federico.mattiello@gmail.com>

Examples

```
showClass("SoupObject")
```

t2p *From Test Statistics To* p*-Values*

Description

Transforming Test Statistics to Permutation *p*-Values

Usage

```
t2p(Tmat, obsOnly = FALSE)
```

Arguments

Tmat	3-dimensional (2-dimensional) array containing the test statistics where the first horizontal slice (first row) contains the observed value
obsOnly	logical, if FALSE (default) the whole permutation distribution of the computed *p*-values is returned, if TRUE only the observed ones are returned.

Value

if obsOnly is FALSE an array of the same dimension of the input matrix Tmat, otherwise only the first row

Author(s)

Livio Finos, Aldo Solari and Federico Mattiello <federico.mattiello@gmail.com>

See Also

permutationSpace, NPC

Index