Package 'flip'

October 31, 2015

Type Package

Title Multivariate Permutation Tests

Version 2.4.3

Date 2015-10-08

Author Livio Finos, with contributions by Florian Klinglmueller, Dario Basso, Aldo Solari, Lucia Benetazzo, Jelle Goeman and Marco Rinaldo. Special thanks are due to Ivan Marin-Franch and Fredrik Nilsson for the debugging and the good questions.

Maintainer Livio Finos <livio@stat.unipd.it>

Depends methods, e1071, someMTP, cherry

Imports Rcpp,

LinkingTo Rcpp, RcppArmadillo

Description It implements many univariate and multivariate permutation (and rotation) tests. Allowed tests: the t one and two samples, ANOVA, linear models, Chi Squared test, rank tests (i.e. Wilcoxon, Mann-Whitney, Kruskal-Wallis), Sign test and McNemar. Test on Linear Models are performed also in presence of covariates (i.e. nuisance parameters). The permutation and the rotation methods to get the null distribution of the test statistics are available. It also implements methods for multiplicity control such as Westfall-Young minP procedure and Closed Testing (Marcus, 1976) and k-FWER. Moreover, it allows to test for fixed effects in mixed effects models.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-04-14 19:04:45

R topics documented:

`flip-package` *The library is devoted to permutation-based inferential methods.*

Description

It implements many univariate and multivariate permutation (and rotation) tests.

The tests comprised are: the one and two samples, ANOVA, linear models, Chi Squared test, rank tests (i.e. Wilcoxon, Mann-Whitney, Kruskal-Wallis), Kolmogorov-Smirnov and Anderson-Darling.

Test on Linear Models are performed also in presence of covariates (i.e. nuisance parameters).

The permutation and the rotation method to get the null distribution of the test statistic(s) are available.

It also implements methods for multiplicity control such as Westfall-Young min-p procedure and Closed Testing (Marcus, 1976).

Details

Package:	flip
Type:	Package
Version:	1.1
Date:	2012-02-05
License:	GPL <=2
LazyLoad:	yes
Depends:	methods, e1071, someMTP, cherry

Author(s)

Livio Finos, with contributions by Florian Klinglmueller, Dario Basso, Aldo Solari, Lucia Benetazzo, Jelle Goeman and Marco Rinaldo. Special thanks are due to Ivan Marin-Franch and Fredrik Nilsson for the debugging and the good questions.

Maintainer: livio finos <livio@stat.unipd.it>

References

For the general framework of univariate and multivariate permutation tests see: Pesarin, F. (2001) Multivariate Permutation Tests with Applications in Biostatistics. Wiley, New York.

For analysis of mixed-models see: L. Finos and D. Basso (2014) Permutation Tests for Between-Unit Fixed Effects in Multivariate Generalized Linear Mixed Models. Statistics and Computing. Vo- lume 24, Issue 6, pp 941-952. DOI: 10.1007/s11222-013-9412-6 J. J. Goeman and

D. Basso, L. Finos (2011) Exact Multivariate Permutation Tests for Fixed Effec- ts in Mixed-Models. Communications in Statistics - Theory and Methods. DOI 10.1080/03610926.2011.627103

For Rotation tests see: Langsrud, O. (2005) Rotation tests, Statistics and Computing, 15, 1, 53-60

A. Solari, L. Finos, J.J. Goeman (2014) Rotation-based multiple testing in the multivariate linear model. Biometrics. Accepted

The structure of `flip` is widely borrowed from library `globaltest` by J. Goeman and J. Oosting in bioconductor.org.

Examples

```
Y=data.frame(matrix(rnorm(50),10,5))
names(Y)=LETTERS[1:5]
Y[,1:2]=Y[,1:2]
x=rep(0:1,5)
data=data.frame(x=x, Z=rnorm(10))
res = flip(Y+matrix(x*2,10,5),~x,~Z,data=data)
res

plot(res)

p2=npc(res,"fisher",subsets=list(c1=c("A","B"),c2=names(Y)))
p2
```

`flip` *The main function for univariate and multivariate testing under a permutation (and rotation) framework + some utilities.*

Description

`flip` is the main function for permutation (or rotation) test.

It allows for multivariate one sample, C>=2 samples and any regression tests. Also the use of covariates (to be fitted in the model but) not under test is allowed.

Usage

```
flip(Y, X=NULL, Z=NULL, data=NULL, tail = 0, perms = 1000, statTest=NULL,
     Strata=NULL, flipReturn, testType=NULL, ...)

flip.statTest
#    c("t", "F", "ANOVA",
#  "Wilcoxon","Kruskal-Wallis", "kruskal", "rank", "Mann-Whitney",
# "chisq","chisq.separated", "Fisher",
#  "McNemar", "Sign","sum","coeff","cor","cor.Spearman","cor.rank","NA")
```

```
orthoZ(Y, X=NULL, Z=NULL, returnGamma=FALSE)
```

Arguments

Y	The response vector of the regression model. May be supplied as a vector or as a formula object. In the latter case, the right hand side of Y is passed on to alternative if that argument is missing, or otherwise to null.
X	The part of the design matrix corresponding to the alternative hypothesis. The covariates of the null model do not have to be supplied again here. May be given as a half formula object (e.g. ~a+b). In that case the intercept is always suppressed.
Z	The part of the design matrix corresponding to the null hypothesis. May be given as a design matrix or as a half formula object (e.g. ~a+b). The default for Z is ~1, i.e. only an intercept. This intercept may be suppressed, if desired, with Z = ~0.
data	Only used when Y, X, or Z is given in formula form. An optional data frame, list or environment containing the variables used in the formula. If the variables in a formula are not found in data, the variables are taken from environment(formula), typically the environment from which gt is called.
tail	Vector of values -1, 0 or 1 indicating the tail to be used in the test for each column of Y. tail=1 (-1) means that greater (smaller) values bring more evidence to the alternative hypothesis. tail=0 indicates a two sided alternative. If the length of tail is smaller than number of columns of Y, the values are recycled.
perms	The number of permutations to use. The default is perms = 1000. Alternatively it can be a matrix (i.e. the permutation space) or a list with elements number and seed.
Strata	A vector, which unique values identifies strata. This option is used only with testType="permutation"; parameter Z is not considered in this case. Also note that when only two levels with one observation per each level are present in each stratum, the problem becomes a paired two-samples problem and hence simplified to a one-sample test.
statTest	Choose a test statistic from flip.statTest. See also Details section.
flipReturn	list of objects indicating what will be included in the output. e.g. list(permP=TRUE,permT=TRUE,data=TRUE).
testType	by default testType="permutation". The use of option "combination" is more efficient when X is indicator of groups (i.e. C>1 samples testing). When the total number of possible combinations exceeds 10 thousand, "permutation" is performed. As an alternative, if you choose "rotation", resampling is performed through random linear combinations (i.e. a rotation test is performed). This option is useful when only few permutations are available, that is, minimum reachable significance is hight. See also the details section for the algorithm used. The old syntax rotationTest=TRUE is maintained for compatibility but is deprecated, use testType="rotation" instead.
returnGamma	logical. Should be the eigenvectors (with corresponding non-null eigenvalues) of the anti-projection matrix of Z (i.e. I- Z(Z'Z)^-1 Z') returned?

... Further parameters. The followings are still valid but deprecated:
permT.return = TRUE, permP.return = FALSE,
permSpace.return = FALSE, permY.return = FALSE. Use flipReturn instead.
dummyfy a named list of logical values (eg. list(X=TRUE,Y=TRUE))
rotationTest= TRUE. Deprecated, use testType='rotation' instead.

Details

statTest="t" is the t statistic derived from the correlation among each Xs and each Ys (i.e. a linear model for each couples of Xs and Ys). This is different from the fit of a multiple (multivariate) linear models, since the correlation does not consider the other covariates). The test t is valid only under the assumption that each variable in X is independent of each variable in Y. To get adequate test while adjusting for covariates, use Z (see example below) The test statistic "sum" is the sum of values (or frequencies) of the given sample centered on the expected (i.e. computed on the overall sample). "coeff" is the statistic based on the estimated coefficient of an lm. It produces a test for every possible combination of (columns of) X and Y (p-values can be combined using npc). "cor" is the correlation (i.e. not partial correlation) between each column of X and each of Y. "cor.Spearman" (or "cor.rank") is the analogous for Spearman's rank correlation coefficient.

"ANOVA" is synonyms of "F". Only valid for dependence tests (i.e. non constant X). "Mann-Whitney" is synonyms of "Wilcoxon". "rank" choose among "Wilcoxon" and "Kruskal-Wallis" depending if the samples are two or more (respectively).

The "Wilcoxon" statistic is based on the 'sum of ranks of second sample minus n1*(n+1)/2' instead of 'sum of ranks of smallest sample minus nSmallest*(n+1)/2'. Therefore the statistic is centered on 0 and allow for two sided alternatives. Despite the p-value are ok, it requires the X to be a two-levels factor in order to compute the right test statistic. When the X is not a two-levels factor, it measures the codeviance among X and ranks of Y.

For paired samples (see also the argument Strata and the example below) the Signed Rank test is performed. To perform the Sign Test use option Sign (i.e. same as Signed Rank but without using magnitude of ranks).

The "Fisher" test is allowed only with dichotomous Ys. The reported statistic is the bottom-right cell of the 2 by 2 frequencies table. The "chisq.separated" test perform cell-wise chi squared (see also Finos and Salmaso (2004) Communications in Statistics - Theory and methods).

The "McNemar" test is based on the signs of the differences, hence it can be used also with ordinal or continuous responses. Only valid for symmetry tests (i.e. X is constant or NULL). The reported statistic for "McNemar" test is the signed squared root of the McNemar statistic. Hence it allows for tailed alternatives.

For ordered X, a stochastic ordering test can be performed using "t","Wilcoxon","sum" and then combining the separated test using npc.

When statTest is a function, the first argument must be Y. This same function is ran to observed data Y and to a number of permuted rows of Y. The returned value must be a vector of test statistics. Please note that argument tail must be defined accordingly. The default way the rows of Y are rearranged is through permutation (without strata). More complex permutation strategies can be defined through proper definition of argument perm (see also permutationSpace).

For testType="rotation": As long as the number of orthogonalized residuals (i.e. the number of observations minus the number of columns in Z) is lower than 50, the function rom is used. The the

number is larger, the faster version `romFast` is used instead. Although the latter is less accurate, for such a big sample size, it is not expected to affect the control of the type I error.

Value

An object of class `flip.object`. Several operations and plots can be made from this object. See also `flip.object-class`.

Author(s)

livio finos (livioATstatDOTunipdDOTit)

References

For the general framework of univariate and multivariate permutation tests see: Pesarin, F. (2001) Multivariate Permutation Tests with Applications in Biostatistics. Wiley, New York.

For Rotation tests see: Langsrud, O. (2005) Rotation tests, Statistics and Computing, 15, 1, 53-60

A. Solari, L. Finos, J.J. Goeman (2014) Rotation-based multiple testing in the multivariate linear model. Biometrics. Accepted

See Also

The permutation spaces on which the test is based: `permutationSpace` function and useful functions associated with that object.

Multiplicity correction: `flip.adjust` and Global test: `npc`.

Examples

```
Y=matrix(rnorm(50),10,5)
colnames(Y)=LETTERS[1:5]
Y[,1:2]=Y[,1:2] +2
res = flip(Y)
res
plot(res)

X=rep(0:1,5)
Y=Y+matrix(X*2,10,5)

data=data.frame(Y,X=X, Z=rnorm(10))
#testing dependence among Y's and X
(res = flip(Y,~X,data=data))
#same as:
#res = flip(A+B+C+D+E~X,data=data)


#testing dependence among Y's and X, also using covariates
res = flip(Y,~X,~Z,data=data)
res
#Note that
#flip(Y,X=~X,Z=~1,data=data)
#is different from
```

```
#flip(Y,~X,data=data)
#since the former is based on orthogonalized residuals of Y and X by Z.

## Not run:
#Rotation tests:
rot=flip(Y,X,Z=~1,testType="rotation")
# note the use Z=~1.

## End(Not run)

#Using rank tests:
res = flip(Y,~X,data=data,statTest="Wilcoxon")
res

#testing symmetry of Y around 0
Y[,1:2]=Y[,1:2] +2
res = flip(Y)
res
plot(res)


#use of strata (in this case equal to paired samples)
data$S=rep(1:5,rep(2,5))
#paired t
flip(A+B+C+D+E~X,data=data,statTest="t",Strata=~S)
#signed Rank test
flip(A+B+C+D+E~X,data=data,statTest="Wilcox",Strata=~S)

# tests for categorical data
data=data.frame(X=rep(0:2,10))
data=data.frame(X=factor(data$X),Y=factor(rbinom(30,2,.2+.2*data$X)))
flip(~Y,~X,data=data,statTest="chisq")
# separated chisq (Finos and Salmaso, 2004. Nonparametric multi-focus analysis
# for categorical variables. CommStat - T.M.)
(res.sep=flip(~Y,~X,data=data,statTest="chisq.separated"))
npc(res.sep,"sumT2") #note that combined test statistic is the same as chisq

## Not run:
# User-defined test statistic:
my.fun <- function(Y){
  summary(lm(Y~X))$coeff[1,"Pr(>|t|)"]
}
X<- matrix(rep(0:2,10))
Y <- matrix(rnorm(30))
flip(Y=Y,X=X,statTest=my.fun)

## End(Not run)
```

`flip.object-class` *Class "flip.object" (and related functions) for storing the result of the function flip and flipMix*

Description

The class flip.object is the output of a call to flip, flipMix, npc and flip.adjust. It stores the information needed for various diagnostic plots. Specific functions to deal with these objects are also documented here.

Usage

```
getFlip(obj, element)

cFlip(...)
```

Arguments

obj	Any flip-object
element	Character string of either slot names of obj (i.e. "res","call","permP","permT", "permSpace","permY","tail","data","call.env"), elements of slot obj@data (e.g. "Y","X","Z","Strata") or columns of obj@res (usually "Test","Stat", "tail","p-value"). Specific uses: "Adjust:" returns all columns of adjusted p-values in obj@res. Any among "nperms","perms","B" return the number of permutation used.
...	flip-objects to be concatenated.

Objects from the Class

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

Slots

res: Object of class flip.object

call: Object of class "call"

permP: A matrix p-values of size B x p (number of permutations x number of variables), first line representing the p-value computed on observed data.

permT: A matrix test statistics of size B x p (number of permutations x number of variables), first line representing the test statistic computed on observed data.

permSpace: A list that may contain B,n,rotFunct,permID. B is the number of permutations/rotations, n number of observations (possibly after orthogonalization of the residuals), rotFunct is the function used to resample the elements. permID is a matrix permutation of size B x n (number of permutations x sampel size), first line representing the p-value computed on observed data. This matrix is usually generated by make.permSpace or make.signSpace

permY: Not implemented yet

tail: tail of the alternative. 1 means right tail, -1 left tail and 0 is bidirectional alternative

Methods

[Select tests from the `flip.object`. Tests can be selected by position of name. It also allow for grep-like selection.

[[Same as `[`

hist Produces an histogram for each element to visualize the distribution based on permutation of test statistics (same as `globaltest:gt.object`). The colors used are mostly taken from `wes.palette(5, "Darjeeling")` of `library(wesanderson)`.

length Number of tests

names `signature(x = "flip.object")`: ...

names<- `signature(x = "flip.object")`: ...

p.value Extracts p-values from the object.

p.adjust Performs multiple testing correction and produces multiplicity-corrected p-values. See `flip.adjust` for details.

plot `signature(x = "flip.object", y = "missing")`. It plots the permutation space and the observed test statistic. The plot is an histogram (see `hist.flip.object`) if there is only one test, while it is a biplot when there are more variables. In this case the arrows of the rotations are colored in red when the associated `p-value(res)<=.05`. The colors used are mostly taken from `wes.palette(5, "Darjeeling")` of `library(wesanderson)`.

result report table of results

show report table of results

size Returns the size of permT (i.e. number of permutations X number of tests)

sort Sorts the tests to increasing p-values.

summary Provides the call, the number of permutations. Option `star.signif`: `=TRUE` (default) set stars depending on the last column of p-values; `=FALSE` inhibits stars, `="p-value"` or any other names indicates the column to which the stars refer.

Author(s)

livio finos

Examples

```
showClass("flip.object")

y=matrix(rnorm(50),10,5)
colnames(y)=c("X1","X2","Y1","Y2","Y3")
res=flip(y)

## Sort by p-values
sort(res)

## Selecting tests
res[1:2]
#same as
res["X"]
```

```
#different from (it selects tests having "1" or "2" in the name)
res[c("1","2")]

## Concatenates two flip-objects
cFlip(res[1:2],res[5])


#plotting results
plot(flip(y))

#Get any slot of the flip-object. eg the permutation space:
head(getFlip(res,"permT"))
#Get any element of the list obj@data. eg Y:
getFlip(res,"Y")
#Get any columns of the results table: obj@res. eg Statistic (choose among colnames(obj@res) ):
getFlip(res,"Stat")
```

`flipMix` *The main function for testing mixed models under a permutation (and rotation) framework*

Description

It allows to test fixed effect in mixed models. You can test within-unit effects, between-unit and interactions of the two. The response can be uni- or multi-variate. See also examples below.

Usage

```
flipMix(modelWithin, X = NULL, Z = NULL, units, perms = 1000, data = NULL,
          tail = 0, statTest = NULL, flipReturn, testType = "permutation",
        Su = NULL, equal.se = FALSE, se = NA, replaceNA.coeffWithin = "coeffMeans",
          replaceNA.coeffWithin.se = replaceNA.coeffWithin, ...)

flipMixWithin(modelWithin, units, X = ~1, perms=1000, data=NULL, tail=0,
                          statTest=NULL, flipReturn,
                       Su=NULL, equal.se=FALSE, se=NA, replaceNA.coeffWithin=0,
                          replaceNA.coeffWithin.se= Inf, ...)

obs2coeffWithin(modelWithin, units, X = NULL, Z = NULL, data = NULL, equal.se = FALSE,
        se = NA, replaceNA.coeffWithin = NA, replaceNA.coeffWithin.se = Inf, ...)
```

Arguments

`modelWithin` When it is a `formula` object, a (possibly multivariate) multiple linear model is fitted. Responses are on the left, while the right part contains ONLY within-unit variables. In this case `data` must be supplied. Alternatively, it can be a `glm`, a `lm` or `vgam` (`library(VGAM)`) (i.e. `vglm`) object. The `modelWithin` have to be performed using only variables within-unit, without using `units` indicator (in

this case the argument data is not used). It can be also a list of models. It can be null if data is provided in the right format (see below).

X — The part of the design matrix corresponding to the between-unit effect that are not null under the alternative hypothesis. If it is a matrix or a data.frame it must have a number of rows equal to the number of units or equal to the total number of observations (in the latest case all elements of the same units must have the same values since they are between-unit effects). The non-null between-unit covariates of null model are defined in Z (see argument below) and do not have to be supplied again here. See also the function flip

NOTE: When called from flipMixWithin, W is used only if statTest="TBTWest".

Z — The part of the design matrix corresponding to the non-null between-unit covariates of the model under the null hypothesis. May be given as a design matrix or as a half formula object (e.g. ~a+b). See also the function flip. If it is a matrix or a data.frame it must have a number of rows equal to the number of units or equal to the total number of observations (in the latest case all elements of the same units must have the same values since they are between-unit effects).

units — Vector of units IDs. May be given as a vector or as a half formula object (e.g. ~subj).

perms — The number of permutations to use. The default is perms = 1000. Alternatively it can be a matrix (i.e. the permutation space) or a list with elements number and seed. See also the function flip.

data — Same as in the function flip. If can also be the results of obs2coeffWithin.

tail — Same as in the function flip.

statTest — For function flipMix choose among "t" and "F" (very similar to statTest in function flip.\ For function flipMixWithin choose among "Tnaive" (i.e. no estimate of the variance), "TH0est" (Default, i.e. estimate of the variance under H0), "TH1est" (i.e. estimate of the variance under H1 for each permutation. Slower but some time more powerful) and "TBTWest" (i.e. estimate of the variance using ILS algorithm at each permutation; it allows for Z different from a constant term. This is the same algorithm used for flipMix. MUCH slower but some time even more powerful).

Both functions allow for vector arguments.

flipReturn — Same as in the function flip.

testType — See also the function flip. Note that this option used only with function flipMix.

Su — Usually NULL. It is the covariance matrix of the random effects. If not supplied, it is estimated by iterative least square algorithm.

equal.se — Logical. If TRUE it force the unit to have the same variance of errors (like it is usually assumed in the lmer methods).

se — Usually NULL. It is a matrix of unit-specific standard errors. If not supplied it is estimated by the algorithm.

replaceNA.coeffWithin — deafult is NA i.e. no replacement. You can provide a specific value (or a vector of values). You can also choose among "coeffMeans" and "unitMeans" (i.e. mean along columns or along rows of Y).

replaceNA.coeffWithin.se
: deafult is Inf. Use the same options of replaceNA.coeffWithin (but means are over the variances and then rooted).

...
: Further parameters. test.coeffWithin Vector of names or IDs of within-unit variables that have to be tested (and reported). Note that variables not in the list are used in the model (i.e. the they play the role of nuisance parameters). fastSumCombination,onlyMANOVA and linComb are used in flipMix to deal with combination of variables/coefficents.

See also the function flip for other parameters.

Value

flipMix and flipMixWithin return an object of class flip.object. Several operations and plots can be made from this object. See also flip.object-class.

Note that function flipMix with statTest="t" or "F" provides tests for each effect between (and interaction) and also provides the overall test PC1 and sum (i.e. all effects ar null, same as npc does).

Use npc with any comb.funct=c("data.sum","data.linComb","data.pc","data.trace") to combine results.

obs2coeffWithin return a list of objects that can be used as argument of data in the function flipMix and flipMixWithin.

Author(s)

Livio Finos and Dario Basso

References

L. Finos and D. Basso (2013) Permutation Tests for Between-Unit Fixed Effects in Multivariate Generalized Linear Mixed Models. Statistics and Computing.

D. Basso, L. Finos (2011) Exact Multivariate Permutation Tests for Fixed Effects in Mixed-Models. Communications in Statistics - Theory and Methods.

See Also

flip, npc

Examples

```
N=10
toyData= data.frame(subj=rep(1:N,rep(4,N)), Within=rep(1:2,N*2),
          XBetween= rep(1:2,rep(N/2*4,2)),ZBetween= rep(rnorm(N/2),rep(8,N/2)))
toyData= cbind(Y1=rnorm(n=N*4,mean=toyData$subj+toyData$ZBetween+toyData$XBetween),
            Y2=rnorm(n=N*4,mean=toyData$subj+toyData$ZBetween+toyData$Within*2),toyData)
(toyData)

#####################
###Testing Between-unit effects
(res=flipMix(modelWithin=as.matrix(toyData[,c("Y1","Y2")])~Within,data=toyData,
     X=~XBetween,Z=~ZBetween,units=~subj,perms=1000,testType="permutation",statTest="t"))
```

```
#same as:
modelWithin <- lm(as.matrix(toyData[,c("Y1","Y2")])~Within,data=toyData)
(flipMix(modelWithin=modelWithin,data=toyData, X=~XBetween,Z=~ZBetween,units= ~subj,
        perms=1000,testType="permutation",statTest="t"))

### Note that this is different from:
modelWithin <- list(Y1=lm(Y1~Within,data=toyData),Y2=lm(Y2~Within,data=toyData))
(flipMix(modelWithin=modelWithin,data=toyData, X=~XBetween,Z=~ZBetween,units= ~subj,
        perms=1000,testType="permutation",statTest="t"))

### combining results
(npc(res,"data.pc"))
(npc(res,"data.trace"))
################################
###Testing Within-unit effects
## The resulting test is approximated. The estimate of the variance within units
## takes in account the presence of effects between units.
(flipMix(modelWithin=as.matrix(toyData[,c("Y1","Y2")])~Within,data=toyData,
        units= ~subj, perms=1000,testType="permutation",statTest="t"))

###The resulting tests are exact. If effects between are presents,
## statTest="Tnaive" or "TBTWest" are more suitable:
(res=flipMixWithin(modelWithin=as.matrix(toyData[,c("Y1","Y2")])~Within,data=toyData,
        units= ~subj, perms=1000,statTest=c("TH1est")))
npc(res)
```

---

npc *Functions for multiplicity corrections*

Description

npc provides overall tests (i.e. weak FWER control), while `flip.adjust` provides adjusted p-values (i.e. strong FWER control).

Usage

```
npc(permTP, comb.funct = c(flip.npc.methods, p.adjust.methods),
          subsets=NULL,weights=NULL, stdSpace=FALSE, ...)
flip.adjust(permTP, method = flip.npc.methods, maxalpha = 1,
          weights = NULL, stdSpace=FALSE, ...)

flip.npc.methods
#    c("Fisher", "Liptak", "Tippett", "MahalanobisT", "MahalanobisP",
#      "minP", "maxT", "maxTstd", "sumT", "Direct", "sumTstd", "sumT2","kfwer",
#      "data.sum","data.linComb","data.pc","data.trace")
# use methods in the last row only for outputs of function flipMix()
```

Arguments

permTP	A permutation space (B times m matrix) or an flip.object as produced by flip. Alternatively it can be a flip.object-class resulting, for example from a call of function flip.
comb.funct	A combining function flip.npc.methods (all but "kfwer"): "Fisher", "Liptak", "MahalanobisT" , "MahalanobisP" (i.e. related to Hotelling T2), "minP" (i.e. Tippet), "maxT", "sumT" (i.e. direct) , "sumT2" (sum of T^2). "Fisher" combining function is the default. See also the section Details.
method	A method among flip.npc.methods or p.adjust.methods. By default "maxT" is used. See also the section Details.
maxalpha	Adjusted p-values greater than maxalpha are forced to 1. It saves computational time when there are many hypotheses under test.
weights	Optional argument that can be used to give certain variables greater weight in the combined test. Can be a vector or a list of vectors. In the latter case, a separate test will be performed for each weight vector. If both subsets and weights are specified as a list, they must have the same length. In that case, weights vectors may have either the same length as the number of covariates in alternative, or the same length as the corresponding subset vector. Weights can be negative; the sign has no effect unless directional is TRUE. It works for npc and flip.adjust with method= "maxT", "maxTstd" or "minP"
subsets	Optional argument that can be used to test one or more subsets of variables. Can be a vector of column names or indices of a flip.object-class (names(flipObject)), or a list of such vectors. In the latter case, a separate test will be performed for each subset. Only for comb.funct
stdSpace	Ask if the permutation distribution of the test statistic should be standardized or not. The default is FALSE. The option is applied only if comb.funct or method is equal to "maxT" or "sumT", it becomes useful when test statistics are of different nature (e.g. chisquare and t-test).
...	further arguments. Among them, tail can be used to set the tail of the alternative for the permTP (see also flip). The arguments statTest, fastSumCombination and linComb are used in objects flipMix and comb.funct= "data.sum","data.linComb","data.pc" or "data.trace".

Details

npc combines the p-values using the combining functions (and the method) described in Pesarin (2001). It makes use of the join space of the permutations. This is usually derived from a call of flip function or flipMixWithin.

Very shortly:\ "Fisher" =-sum log(p-values) \ "Liptak" =sum qnorm(p-values) \ "MahalanobisT" = Mahalanobis distance of centered matrix permTP (or permTP@permT) \ "MahalanobisP" = same as above, but using scores defined by qnorm(p-values) (tails are forced to be one-sided) \ "minP" = "Tippett" = min(p-values) \ "maxT" = max(test statistics) \ "maxTstd" = max(standardized test statistics) \ "sumT" = sum (test statistics) \ "sumTstd" = sum (standardized test statistics) \ "sumT2" = sum (test statistics)^2\ The followings have to be used carefully and only with objects from function flipMix()\ "data.sum" = sum of all columns of Y\ "data.linComb" = sum of all columns of Y (include a vector or matrix linComb among the arguments)\ "data.pc" = extract the first Principal

component from the covariance matrix (you may also include a vector whichPCs indicating which PCs you want to consider)\ "data.trace" = Extends the Pillai Trace, use parametric bootstrap to asses the significance.\ "kfwer" = can be only used with flip.adjust (not in npc). It requires an extra parameter k (k=11 by default).

flip.adjust adjusts the p-value for multiplicity (FamilyWise Error Rate -FWER- and kFWER). When method is equal to "maxT", "maxTstd" (i.e. max T on scale(permTP)) or "minP" (i.e. Tippett) it performs the step-down method of Westfall and Young (1993). For any other element of flip.npc.methods (i.e. "Fisher", "Liptak", "sumT" (i.e. direct) or "sumT2" (sum of T^2)) a call to npc together with a closed testing procedure is used (it make use of cherry:closed). When method is any among p.adjust.methods the function stats:p.adjust or -if weights are provided- someMTP:p.adjust.w is used. To perform control of the kFWER use flip.adjust with method="kfwer" and extra parameter k.

Value

The function returns an object of class flip.object-class (and the use of getFlip(obj,"Adjust").

Author(s)

livio finos (livioATstatDOTunipdDOTit) and Aldo Solari.

References

Pesarin (2001) Multivariate Permutation Tests with Applications in Biostatistics. Wiley, New York.

P. H. Westfall and S. S. Young (1993). Resampling-based multiple testing: Examples and methods for p-value adjustment. John Wiley & Sons.

Examples

```
Y=data.frame(matrix(rnorm(50),10,5))
names(Y)=LETTERS[1:5]
Y[,1:2]=Y[,1:2]+1.5
res = flip(Y,perms=10000)

########npc
p2=npc(res) # same as p2=npc(res,"Fisher")
summary(p2)
p2=npc(res,"minP")
summary(p2)
p2=npc(res,"Fisher",subsets=list(c1=c("A","B"),c2=names(Y)))
summary(p2)
p2=npc(res,"Fisher",subsets=list(c1=c("A","B"),c2=names(Y)),weights=1:5)
summary(p2)



res=flip.adjust(res,"maxT")

#res=flip.adjust(res,"BH")
##same as
```

```
#p.adjust(res,"BH")

## now try
#getFlip(res,"Adjust")
```

`permutationSpace` *These functions handle the orbit of permutation/rotation tests (i.e. permutation/rotation space).*

Description

`make.permSpace` computes the `perms` x n matrix of ids used for test of dependence. `make.signSpace` computes the `perms` x n vector of +1 and -1 used for symmetry test.

`rom` computes a Random Orthogonal Matrix of size nXn (C-compiled function, very fast)

`romFast` computes a Random Orthogonal Matrix of size nXn using the `qr.Q` decomposition. `romFast` is faster than `rom` but can be inaccurate (i.e. providing inaccurate type I error control when used in testing), specially for very small `n` (i.e. sample size).

`allpermutations` computes all permutations of a vector `Y`. Is is based on the function `permutations` of the `library(e1071)`.

`t2p` computes the (possibily multivariate) space of p-values from the space of test statistic.

Usage

```
make.permSpace(IDs,perms,return.permIDs=FALSE,testType="permutation",Strata=NULL, X=NULL,...)
make.signSpace(N, perms)
allpermutations(Y)
npermutations(Y)
rom(N)
romFast(N)
t2p(T, obs.only = NULL, tail = NULL)
```

Arguments

`IDs`	vector of IDs to be permuted. If `IDs` is a scalar, it is replaced with `1:IDs`.
`return.permIDs`	logical. If `TRUE`, the matrix of permuted IDs is stored and returned. Only used with `testType="permutaiton"`
`N`	number of elements of the sample. It is also the dimention of the random orthogonal matrix in `rom`.
`Y`	a vector of data. It can also be a vector 1:N referring to the IDs of observations.
`perms`	number of random permutations. If it is a list, it has two elements `number` (the number of random permutation requested) and `seed` (the seed to be set when start generating. it is useful for reproducibility) If perms > number of all possible flips, then compute the complete space.
`T`	the (possibly multivariate) permutation space as returned, for example by `flip`

obs.only	logical. If TRUE only the p-value for observed test statistic is returned, otherwise the whole space is computed. Defaults: TRUE if T is a flip-object, FALSE otherwise.
tail	Tail of the distribution being significant for H1. See also argument tail in flip. Defaults: 1 if T is NOT a flip-object, it is taken from T otherwise.
testType	See argument testType in flip
Strata	See argument testType in flip
X	A vector of length N with a different value for each group. Only used together with testType="combination".
...	other parameters

Details

rom implements the algorithm of Stewart (1980). The function is compiled in C++.

Author(s)

Livio Finos, Aldo Solari, Marco Rinaldo and Lucia Benetazzo

References

Pesarin (2001) Multivariate Permutation Tests with Applications in Biostatistics. Wiley, New York.

Stewart, G. W. (1980). The efficient generation of random orthogonal matrices with an application to condition estimators. SIAM Journal on Numerical Analysis 17, 403-409.

See Also

flip

Examples

```
#10 random elements of the orbit of a one-sample test
make.signSpace(5, 10)

#All elements of the orbit of a one-sample test (the size of the space is 2^5 < 1000)
make.signSpace(5, 1000)

## Not run:
#A random rotation matrix of size 3
(r=rom(3))
#verify that it is orthogonal:
r

## End(Not run)
```

seeds *Seeds data*

Description

Famous seeds growing data from Pesarin, F. (2001) Multivariate Permutation Tests with Applications in Biostatistics. Wiley, New York.

Usage

```
seeds
```

Format

the data.frame contains the three columns: grs, x, y

Index