

Package ‘pi0’

July 2, 2014

Type Package

Title Estimating the proportion of true null hypotheses for FDR

Version 1.3-250

Date 2012-09-06

Author Long Qu

Maintainer Long Qu <longorognol@hotmail.com>

Depends R (>= 2.12.0), LowRankQP

Imports Matrix (>= 1.0-0), numDeriv, limSolve (>= 1.5.2), rgl,scatterplot3d, qvalue, Iso(>= 0.0-5), quadprog (>= 1.5-3),kernlab

Enhances stats, graphics, limma, OCplus

Description This package implements method(s) to (approximately unbiasedly) estimate the proportion of true null hypotheses,i.e., the π_0 , when a very large number of hypotheses are simultaneously tested, especially for the purpose of (local) false discovery rate control for microarray data. It also contains functions to estimate the distribution of noncentrality parameters from a large number of parametric tests.

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-09-06 11:18:38

NeedsCompilation yes

R topics documented:

pi0-package	3
agjack.pi0	5
CBUM	6
clean.cdf, save.cdf and load.cdf	7
coef.ncpest	8
combn2R	9
cond.cdf	11
convest	12
discTMix	13
dncp	15
dt.int2	16
dt.lap	17
dt.sad and pt.sad	18
dtm.mix	19
extrp.pi0	20
fdr	23
fitted	24
fitted.discTMix	25
geomean and harmean	26
gjack	27
grid.search	27
histf1	28
lastbin	30
lfdr and ppee	31
loadOrInstall	32
logLik.ncpest	33
marginal.dt	33
matrix.t.test	34
mTruncNorm	36
npnrcpp	37
npnrcpp.iter	39
npnrcpt	40
parncpt	42
pavaf1	44
pdf.dist	45
plot.extrpi0	46
plot.npnrncpt	47
plot.pnrncpt	48
plot.sparncpt	50
plot.subex	51
plot.subt	52
print.extrpi0	53
print.subex	54
print.subt	55
reflect and fold	57
sim.dat	58

simulatedDat	59
simulatedExtrpi0	60
simulatedSubex	61
simulatedSubt	62
simulatedTstat	63
sparncpt	63
subex	65
subt	67
summary.nparncpt	69
summary.parncpt	70
summary.sparncpt	71
varB	72
vcov.ncpest	72
znormix	73

Index	75
--------------	-----------

pi0-package	<i>Estimating the proportion of true null hypotheses and False Discovery Rates</i>
-------------	--

Description

This package implements method(s) to (approximately unbiasedly) estimate the proportion of true null hypotheses, i.e., the π_0 , when a very large number of hypotheses are simultaneously tested, especially for the purpose of (local) false discovery rate control for microarray data. It also contains functions to estimate the distribution of noncentrality parameters from a large number of parametric tests.

Details

Package: pi0
 Type: Package
 Version: 1.3-250
 Date: 2012-09-06
 License: GPL version 2 or newer

- [subt](#) Subsampling a microarray data set, do t-test for each gene, and estimate p-value density at 1 for each subsample.
- [extrp.pi0](#) Extrapolate the p-value density at 1 over subsample sizes to estimate the proportion of true null hypotheses.
- [fdr](#) Estimate false discovery rate based on p-values and a given estimate of the proportion of true null hypotheses.
- [subex](#) A wrapper that automates [subt](#), [extrp.pi0](#), and [fdr](#).

- [combn2R](#) Generating a sample of combinations by choosing m1 out of n1 and m2 out of n2 simultaneously.
- [matrix.t.test](#) Apply a t-test to each row or column of a matrix.
- [lastbin](#) Estimate p-value density at 1 based on a histogram.
- [parncpt](#) Parametrically estimate the distribution of noncentrality parameters.
- [nparncpt](#) Nonparametrically estimate the distribution of noncentrality parameters.
- [sparncpt](#) Semiparametrically estimate the distribution of noncentrality parameters.
- [nparncpp](#) Nonparametric estimate of the distribution of absolute noncentrality parameters from a large number of p-values.
- [CBUM](#) (Censored) Beta-Uniform mixture model for p-values.
- [znormix](#) Normal mixture model for z-scores.

Author(s)

Long Qu

Maintainer: Long Qu <longorognol@hotmail.com>

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

Ruppert D, Nettleton D, Hwang JT. (2007) Exploring the Information in p -values for the Analysis and Planning of Multiple-test Experiments. *Biometrics*. 63. 483-495.

G.J. McLachlan, R.W. Bean and L. Ben-Tovim Jones. (2006) A Simple implementation of a normal mixture approach to differential gene expression in multiclass microarrays. *Bioinformatics*, 22(13):1608-1615.

Anastasios Markitsis and Yinglei Lai (2010) A censored beta mixture model for the estimation of the proportion of non-differentially expressed genes. *Bioinformatics* 26(5):640-646.

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[subex](#), [subt](#), [extrp.pi0](#), [fdr](#), [combn2R](#), [nparncpt](#), [parncpt](#), [sparncpt](#), [nparncpp](#)

Examples

```
## Not run:
set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
## this is how the 'simulatedSubex' object in this package generated
simulatedSubex=subex(simulatedDat,balanced=FALSE,max.reps=Inf,plotit=FALSE)
plot(simulatedSubex)
```

```
## End(Not run)
data(simulatedSubex); print(simulatedSubex)
## parametric, nonparametric, semiparametric estimate of
## noncentrality parameter distribution from t-statistics
data(simulatedTstat)
(npfit=npnrcpt(tstat=simulatedTstat, df=8, plotit=FALSE));
(pfit=parncpt(tstat=simulatedTstat, df=8, zeromean=FALSE));
(pfit0=parncpt(tstat=simulatedTstat, df=8, zeromean=TRUE));
(spfit=spnrcpt(npfit,pfit));
```

agjack.pi0

Averaged generalized jackknife estimate of pi0

Description

Averaged generalized jackknife estimate of pi0

Usage

```
agjack.pi0(subtobj,mean.n=c('mean','harmean','geomean') ,
  pointpair=FALSE, trunc=TRUE,tol=1e-5)
```

Arguments

subtobj	A subt object
mean.n	A character: name of the function to compute the average sample sizes
pointpair	logical: if TRUE, then gjack is called for each pair of rows in subtobj; otherwise, gjack is called for each pair of unique average sample sizes.
trunc	logical, indicating if each gjack estimate is truncated to [0,1].
tol	A small tolerance number.

Details

When pointpair is FALSE, the rows in subtobj are first grouped by combination of sample sizes and the estimates are averaged for each group. Then [gjack](#) is called for each pair of groups.

Value

A numeric scalar of estimated pi0.

Author(s)

Long Qu

See Also

[gjack](#)

Examples

```
data(simulatedSubt)
agjack.pi0(simulatedSubt) ##
```

 CBUM

(Censored) Beta-Uniform mixture for p-values

Description

This function implements the method of Markitsis and Lai (2010).

Usage

```
CBUM(p, start.pi0=0.5, thresh.censor=0.05, eps=1e-5, niter=Inf, verbose=FALSE)
```

Arguments

<code>p</code>	a numeric vector the p-values
<code>start.pi0</code>	numeric scalar, starting value of pi0 for EM algorithm.
<code>thresh.censor</code>	numeric scalar, the threshold of censoring. If <code>isTRUE(lambda < min(p))</code> , this is equivalent to the BUM method of Pounds and Morris (2003).
<code>eps</code>	numeric scalar, maximum tolerable absolute difference of parameter estimates for successive iterations in the EM algorithm.
<code>niter</code>	numeric scalar, maximum number of EM iterations.
<code>verbose</code>	logical scalar, indicating whether excessive outputs will be printed during EM algorithm.

Details

This function is an improved version of the `CBpi0` function available at <http://home.gwu.edu/~ylai/research/CBpi0/CBpi0.txt>, which implements the censored (1-parameter beta)-uniform mixture model to a large number of p-values.

Value

A numeric scalar, being the proportion pi0 of true null hypotheses. The result has a class 'CBUM', with the following attributes:

<code>converged</code>	logical, convergence status.
<code>iter</code>	numeric, number of iterations.
<code>call</code>	the <code>match.call()</code> result.
<code>alpha</code>	estimated alpha parameter for the beta component.
<code>lfdr</code>	numeric vector estimated local false discovery rates, if <code>thresh.censor < min(p)</code> ; NULL, otherwise
<code>thresh.censor</code>	the censoring threshold for p-values.

Author(s)

Long Qu modified the code from Markitsis and Lai (2010).

References

Anastasios Markitsis and Yinglei Lai (2010) A censored beta mixture model for the estimation of the proportion of non-differentially expressed genes. *Bioinformatics* 26(5):640-646.

See Also

[qvalue](#), [histf1](#)

Examples

```
set.seed(9992722)
p=c(runif(7000), rbeta(3000, .8, 2.2))
CBUM(p, eps=1e-5)
```

clean.cdf, save.cdf and load.cdf

*Saving, loading or removing pre-computed conditional cdf objects
from memory or disk*

Description

Saving, loading or removing pre-computed conditional cdf objects from memory or disk

Usage

```
save.cdf(path=getwd())
load.cdf(path=getwd())
clean.cdf(where=c("memory", "disk", "both"), path=getwd())
```

Arguments

where	Clean either memory or disk or both cdf objects
path	Character: path where pre-computed conditional cdf RData files are stored

Details

The filename on the disk is `.pi0cdfp.RData`; the environment name is `.pi0cdfp` which contains the pre-computed conditional cdf objects. The environment itself is stored in global environment.

Value

invisible logical status for `save.cdf` and `clean.cdf`. For `load.cdf`, a character string from `load` or a try-error object.

Author(s)

Long Qu

References

Ruppert, Nettleton, Hwang. 2007. Exploring the Information in p -values for the Analysis and Planning of Multiple-test Experiments. *Biometrics*. 63. 483-495.

See Also

[cond.cdf](#)

coef.ncest

Extract estimated parameters

Description

Extract estimated parameters from object of ncest class

Usage

```
## S3 method for class 'ncest'  
coef(object, ...)
```

Arguments

object	an object of ncest class
...	currently not used

Value

a numeric vector of the estimated parameters

Author(s)

Long Qu

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#), [nparncpp](#)

combn2R *Randomly Choosing R Combinations of Two Groups of n and n2 Elements Taken m and m2 at a Time*

Description

This is the enhanced version of `combn`. There are two groups, with n_1 and n_2 elements, respectively. Each time, m_1 elements will be randomly chosen from group 1; and m_2 elements will be randomly chosen from group 2. These m_1+m_2 elements form one combination. This function generate either all `choose(n1, m1)*choose(n2, m2)` such combinations or a subset of R of them. A function, possibly identity, is then applied to each selected combination.

Usage

```
combn2R(x, m, x2, m2, R, FUN = c, simplify = TRUE,
        sample.method="all", try.rest=TRUE, ...)
```

Arguments

<code>x</code>	group 1 vector for combinations, or positive integer n for <code>x <- seq(n)</code> .
<code>m</code>	number of elements to choose from <code>x</code> , i.e., group 1.
<code>x2</code>	group 2 vector for combinations, or positive integer n_2 for <code>x2 <- seq(n)</code> . If missing, it reduces to get R combinations from <code>x</code> taken m at a time. See details.
<code>m2</code>	number of elements to choose from <code>x2</code> , i.e., group 2. If missing, it reduces to get R combinations from <code>x</code> taken m at a time. See details.
<code>R</code>	the number of combinations to be randomly chosen from all <code>choose(n, m)*choose(n2, m2)</code> combinations. If missing or larger than all possible combinations, results from all combinations will be <i>tried</i> , but <i>not</i> guaranteed when the total number of possible combinations is too large. See details.
<code>FUN</code>	a function to be applied to each chosen combination. When neither <code>x2</code> nor <code>m2</code> is missing, and neither <code>m=0</code> nor <code>m2=0</code> , this function needs to accept at least two arguments, the first of which is a vector of length m , which is a subset of <code>x</code> ; and the second argument of <code>FUN</code> is a vector of length m_2 , which is a subset of <code>x2</code> . Additional arguments are supplied with dots. When <code>combn2R</code> is only used for one group situation (similar to <code>combn</code>), the second argument of <code>FUN</code> is not required.
<code>simplify</code>	logical, indicating if the result should be simplified to an array (typically a matrix); see <code>combn</code> .
<code>sample.method</code>	character, specifying how samples should be taken, if not all combinations are generated; possible choices are "diff2", "all", and "replace" for two-group situation, and "replace" and "noReplace" in one-group situation. See Details.
<code>try.rest</code>	logical, together with <code>sample.method</code> , specifying how samples should be taken, if not all combinations are generated; see Details.
<code>...</code>	optionally, further arguments to <code>FUN</code> .

Details

This function enhances `combn` in two ways. One is to deal with two-group situation, which is commonly seen in real designs; the other is to choose only a random sample of size R from all possible combinations to avoid unnecessary computation.

When neither x_2 nor m_2 is missing and neither $m=0$ nor $m_2=0$, the function works in two-group mode. In this situation,

(A) if `sample.method="diff2"`, `combn2R` will try to sample R combinations from each group separately. That is, first sample R combinations from x taken m at a time, and then sample R combinations from x_2 taken m_2 at time. The results are then combined to give R combinations from the two groups. This sampling method will make the samples as different from each other as possible. But when R is larger than $\min(\text{choose}(n_1, m_1), \text{choose}(n_2, m_2))$, it is not possible to get R samples from each group separately. If this happens and `try.rest=FALSE`, then R will be reset to $\min(\text{choose}(n, m), \text{choose}(n_2, m_2))$ and the function works as before; otherwise, if `try.rest=TRUE`, then `sample.method` will be reset to "all" and the function will try to get R samples from all $\text{choose}(n_1, m_1) * \text{choose}(n_2, m_2)$ combinations (see below).

(B) if `sample.method="all"`, `combn2R` will try to sample R samples from all $\text{choose}(n_1, m_1) * \text{choose}(n_2, m_2)$ combinations directly. This means two samples of size $m+m_2$ may have the same sample of size m (or m_2) which comes from x (or x_2). For example, if $x=1:3$, $m=1$, $x_2=4:6$, $m_2=2$ and $R=2$, then it is possible to get one sample to be 1 and 4,5, but the other sample is 1 and 5,6. That is, the same sample from x is used in both results. This will not happen when `sample.method='diff2'`. However, this will guarantee any two samples of size $m+m_2$ will differ in at least one element.

(C) if `sample.method='replace'`, `combn2R` will not guarantee the uniqueness of the R combinations in any way. It is possible to have two exactly the same samples of size $m+m_2$.

Because the number of possible combinations grows very fast, computational limitations may be reached. In this case, if `try.rest=TRUE`, then `sample.method` will be reset to "replace", which uses the least computational resources; otherwise, an error will be generated.

When either x_2 or m_2 is missing, or one of m and m_2 is zero, the function works in one-group mode. In this situation, `sample.method="diff2"` and `sample.method="all"` will be treated the same as `sample.method="noReplace"`, and `combn2R` will try to obtain R different combinations from all possible combinations for the non-missing group. Again, if this fails due to computational limitations, `sample.method` will be reset to "replace" and no guarantee will be made to ensure the R combinations are different from each other.

Value

a `list` or `array` (in nondegenerate cases), similar to `combn`. An attribute "sample.method" will be added to the `list` or `array`, which stores the *actual* sampling method that has been used, which may or may not be the same as specified in the argument.

Note

Note that the results are *not* necessarily in order, which is a difference from [combn](#).

Author(s)

Long Qu

See Also

[combn](#) in **utils** or **combinat**,

Examples

```
combn2R(4,2) ### the same as combn(4,2), except an additional attribute
combn2R(1:4*2,2)
combn2R(4,2,5,1)
combn2R(4,2,5,1,FUN=sum)
set.seed(992722)
combn2R(4,2,R=3) ### the same as combnR(4,2,3), except an additional attribute
combn2R(4,2,R=10) ### only 6 combinations possible
combn2R(4,2,5,1,R=8)
combn2R(1:4*2,2,5,1,R=50) ### only 30 combinations possible
combn2R(1:4*2,2,5,1,R=5) ### when considering only one group, there are several common samples.
combn2R(1:4*2,2,5,1,R=5, sample.method="diff2") ### no common samples, even considering only one group
combn2R(1:3*3,1,3,1,R=5, sample.method="replace") ### two pairs of exactly common samples
combn2R(100,3,100,3,R=5, sample.method="all") ### 'all' combinations not feasible (~3e10)
```

cond.cdf

conditional cdf of p-values given noncentrality parameters

Description

conditional cdf of p-values given noncentrality parameters (ncp)

Usage

```
cond.cdf(p.eval,ncp,test=c("t","z"),alternative=c("two.sided","less",
"greater"), df=if(test=="z")Inf else df,keep.cdf=TRUE,
suppressWarnings=TRUE)
```

Arguments

p.eval	numeric vector, at which the conditional CDF is evaluated.
ncp	Numeric vector of noncentrality parameters
test	Either t-test or z-test.
alternative	The same as in t.test .
df	The degree of freedom.

keep.cdf Logical. If TRUE, the computed CDF will be stored in a hidden environment `.pi0cdfp` within the global environment for later re-use.

suppressWarnings Logical, indicating if warnings are suppressed

Value

A numeric matrix, with each row corresponding to `p.eval` and each column corresponding to `ncp`.

Note

Warnings might be produced when full precision is not achieved in `pt`, but this is rarely very problematic.

Author(s)

Long Qu

References

Ruppert, Nettleton, Hwang. 2007. Exploring the Information in p -values for the Analysis and Planning of Multiple-test Experiments. *Biometrics*. 63. 483-495.

convest

Estimate Proportion of True Null Hypotheses

Description

Returns an estimate of the proportion of true null hypotheses using a convex decreasing density estimate on a vector of p -values.

Usage

```
convest(p, niter = 100, doplot = FALSE, doreport = FALSE)
```

Arguments

`p` numeric vector of p -values, calculated using any test of your choice. Missing values are not allowed

`niter` number of iterations to be used in fitting the convex, decreasing density for the p -values. Default is 100.

`doplot` logical, should updated plots of fitted convex decreasing p -value density be produced at each iteration? Default is FALSE.

`doreport` logical, should the estimated proportion be printed at each iteration? Default is FALSE.

Details

The proportion of true null hypotheses is often denoted π_0 .

Value

Numeric value in the interval [0,1] representing the estimated proportion of true null hypotheses, with `class` being `convest` and the `lfdr` attribute containing estimated local false discovery rates.

Author(s)

Long Qu slightly modified and `convest` function by Egil Ferkingstad and Mette Langaas in `limma` package.

References

Langaas, M., Ferkingstad, E., and Lindqvist, B. (2005). Estimating the proportion of true null hypotheses, with application to DNA microarray data. *Journal of the Royal Statistical Society Series B*, 67, 555-572. Preprint at <http://www.math.ntnu.no/~mettela/>

See Also

See [08.Tests](#) for other functions for producing or interpreting p-values.

Examples

```
set.seed(9992722)
convest(runif(1e4)^1.5, niter=100, doplot=interactive())
```

discTMix

Fit a discrete mixture of (noncentral) t-distributions

Description

Discrete mixture on central t and a specified number of noncentral t-distributions. This is essentially the same as `tMixture` in `OCplus` package.

Usage

```
discTMix(tstat, n1 = 10, n2 = n1, nq, p0, p1, D, delta, paired = FALSE,
         tbreak, ext = TRUE, threshold.delta=0.75, ...)
```

Arguments

<code>tstat</code>	the vector of genewise t-statistics
<code>n1</code>	number of samples in the first group
<code>n2</code>	number of samples in the second group
<code>nq</code>	the number of components in the mixture that is fitted
<code>p0</code>	a starting value for the proportion of non-differentially expressed genes.
<code>p1</code>	a vector with starting values for the proportions of genes that are differentially expressed with effect size D .
<code>D</code>	a vector of starting values for the effect sizes of the differentially expressed genes, corresponding to the proportions <code>p1</code> .
<code>delta</code>	a vector of starting values for the effect sizes of the differentially expressed genes, expressed as non-centrality parameters; this is just a different way of specifying D , though if both are given, <code>delta</code> will get priority.
<code>paired</code>	a logical value indicating whether the t-statistics are two-sample or paired.
<code>tbreak</code>	either the number of equally spaced bins for tabulating <code>tstat</code> , or the explicit break points for the bins, very much like the argument <code>breaks</code> to function <code>cut</code> ; the default value is the square root of the number of genes.
<code>ext</code>	a logical value indicating whether to extend the bins, i.e. to set the lowest bin limit to $-\infty$ and the largest bin limit to ∞ .
<code>threshold.delta</code>	mixture components with an estimated absolute non-centrality parameter <code>delta</code> below this value are considered to be too small for independent estimation; these components and their corresponding <code>p1</code> are pooled with the null-component and <code>p0</code> , see Details .
<code>...</code>	additional arguments that are passed to <code>optim</code> to control the optimization.

Details

The minimum parameter that needs to be specified is `nq` - if nothing else is given, the proportions are equally distributed between `p0` and the `p1`, and the noncentrality parameters are set up symmetrically around zero, e.g. `nq=5` leads to equal proportions of 0.2 and noncentrality parameters -2 , -1 , 1 , and 2 . If any of `p1`, `D`, or `delta` is specified, `nq` is redundant and will be ignored (with a warning). `discTMix` will in general make a valiant effort to deduce valid starting values from any combination of `nq`, `p0`, `p1`, `D`, and `delta` specified by the user, and will complain if that is not possible.

The fitting problem that this function tries to solve is badly conditioned, and will in general depend on the precise set of starting values. Multiple runs from different starting values are usually a good idea. We have found however, that the model seems fairly robust towards misspecification of the number of components, at least when estimating `p0`. What happens when too many components are specified is that some of the nominally noncentral t-distributions describing the behaviour of differentially expressed genes are fitted with noncentrality parameters very close to zero, and the true `p0` gets spread out between the nominal `p0` and the almost-central components. Adding up these different contributions usually gives a similar solution to re-fitting the model with fewer components. The cutoff for the size of non-centrality parameters that can be estimated realistically is specified via `threshold.delta`, whose default value is based on a small simulation study reported in Pawitan et al. (2005); see [Examples](#). (Note that the AIC can also be helpful in determining the number of components.)

Value

A list with class `discTMix`, with the following components:

<code>p0.est</code>	the estimated proportion of non-differentially expressed genes, after collapsing components with estimated non-centrality sizes below <code>threshold.delta</code> .
<code>p0.raw</code> and <code>pi0</code>	the estimated proportion before collapsing the components.
<code>p1</code>	the estimated proportions of differentially expressed genes corresponding to the effect sizes, relating to <code>p0.raw</code> .
<code>D</code>	effect sizes of the differentially expressed genes in multiples of the gene-by-gene standard deviation.
<code>delta</code>	effect sizes of the differentially expressed genes expressed as the noncentrality parameter of the corresponding noncentral t-distribution.
<code>AIC</code>	the AIC value for the maximum likelihood fit.
<code>opt</code>	The output from <code>optim</code> , giving details about the optimization process.
<code>data</code>	A list of <code>tstat</code> and <code>df</code> .

Author(s)

Long Qu modified the `tMixture` by Y. Pawitan and A. Ploner in `OCplus` package.

References

Pawitan Y, Krishna Murthy KR, Michiels S, Ploner A (2005) Bias in the estimation of false discovery rate in microarray studies, *Bioinformatics*.

See Also

[tstatistics](#), [EOC](#), [optim](#), [fitted.discTMix](#)

dncp

Density of noncentrality parameters

Description

These functions return the density function of noncentrality parameters, from `ncpest` objects

Usage

```
dncp(obj, ...)
## S3 method for class 'parncpt'
dncp(obj, fold=FALSE, ...)
## S3 method for class 'nparncpt'
dncp(obj, fold=FALSE, ...)
## S3 method for class 'sparncpt'
dncp(obj, fold=FALSE, ...)
## S3 method for class 'nparncpp'
dncp(obj, reflect=TRUE, ...)
```

Arguments

obj	an object of class <code>ncest</code> , from which noncentrality parameter density to be extracted
fold	Logical: if TRUE, then the density of noncentrality parameters is folded about zero to give the density of absolute noncentrality parameters.
reflect	Logical: if TRUE, then the density of absolute noncentrality parameters is reflected about zero.
...	Further arguments.

Value

A function of one argument

Note

`dncp.nparncpp` is not yet implemented.

Author(s)

Long Qu

References

Ruppert D, Nettleton D, Hwang JT. (2007) Exploring the Information in p -values for the Analysis and Planning of Multiple-test Experiments. *Biometrics*. 63. 483-495.

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#), [nparncpp](#)

dt.int2

INTerpolation of INTeger degrees of freedom noncentral t-density

Description

This function evaluates the noncentral t -density using an iterative procedure for integer degrees of freedom. This is much faster than two calls to the `pt` approach. For non-integer degrees of freedom, the polynomial interpolation is used to approximate the density.

Usage

```
dt.int2(x, df, ncp, log = FALSE, ndiv = 8)
```


Arguments

x	A numeric vector of quantiles
df	A numeric vector degrees of freedom
ncp	A numeric vector of noncentrality parameters
log	logical; if TRUE, log densities are returned.
ndiv	numeric; the number of points used for polynomial interpolation

Details

This function uses the iterative relation for the integral in the noncentral t-density. It starts with $df=0$ and $df=1$, and then iteratively computes the integral for larger df . For non-integer df , it uses `ndiv` nearest points to perform a divided difference polynomial interpolation approximation. For integer df , this function is about 2 to 3 times faster than `dt` function and is exact.

Value

A numeric vector of densities.

Author(s)

Long Qu

See Also

[dtn.mix](#), [dt](#), [dt.lap](#), [dt.sad](#), [mTruncNorm](#)

dt.lap

Laplace approximation to the noncentral t-density

Description

Laplace approximation to noncentral t-density

Usage

```
dt.lap(x, df, ncp = 0, log = FALSE, normalize = c("central",
"integral", "none"), ...)
```

Arguments

x	A numeric vector of quantiles
df	A numeric vector of degrees of freedom
ncp	A numeric vector of noncentrality parameters
log	logical; if TRUE, log densities are returned

normalize	character. If central, the normalization is such that the approximation is exact when ncp=0. If integral, numerical integration is performed such that the density integrate to 1 (not implemented yet). If none, no normalization is performed.
...	currently not used.

Value

numeric vector of densities

Author(s)

Long Qu

References

Young, G.A. and Smith R.L. 2005. Essentials of statistical inference. Cambridge University Press. Cambridge, UK.

See Also

[dt.int2](#), [dt.sad](#), [dt](#)

dt.sad and pt.sad *Saddle Point Approximation of noncentral t-distribution*

Description

Density and cumulative distribution function of noncentral t-distribution

Usage

```
dt.sad(x, df, ncp = 0, log = FALSE, normalize = c("approximate", "derivative", "integral", "none"), epsilon = 1e-04)
pt.sad(q, df, ncp = 0, log = FALSE, epsilon = 1e-04)
```

Arguments

x, q	numeric vector of quantiles
df	numeric vector of degrees of freedom
ncp	numeric vector of noncentrality parameter
log	logical; whether log should be taken.
epsilon	a small numeric scalar; if the difference between q and ncp is closer than this, results will be computed differently.
normalize	the way to normalize the approximate density so that it is closer to a true density.

Value

dt.sad returns density; pt.sad returns the probability.

Author(s)

Long Qu

References

Broda, Simon and Paolella, Marc S. 2007. Saddlepoint approximations for the doubly noncentral t distribution, Computational Statistics & Data Analysis, 51,6, 2907-2918.

See Also

[dt.int2](#), [dt.lap](#), [dt](#)

dtn.mix

Density of noncentral t-normal mixture

Description

Density of noncentral t-distribution, with noncentrality parameter (NCP) being normally distributed. This is a scaled noncentral t-density.

Usage

```
dtn.mix(t, df, mu.ncp, sd.ncp, log = FALSE, approximation = c("int2",
  "saddlepoint", "laplace", "none"), ...)
```

Arguments

t	A numeric vector of quantiles
df	A numeric vector of degrees of freedom
mu.ncp	A numeric vector of normal mean of NCP
sd.ncp	A numeric vector of normal SD of NCP
log	logical; if TRUE, log density is returned.
approximation	character; Method of approximation. <code>int2</code> computes exact density for integer <code>df</code> and polynomially interpolate to non-integer degrees of freedom. <code>saddlepoint</code> computes the saddle point approximation of the noncentral t-density. <code>laplace</code> computes the laplacian approximation of the noncentral t-density. <code>none</code> uses the (sort of) true noncentral t-density dt function.
...	other arguments passed to dt.int2 or dt.sad .

Details

Mathematically, this is equivalent to $dt(t/s, df, \mu.ncp/s)/s$ where $s=\sqrt{1+sd.ncp*sd.ncp}$. But the various approximations are usually sufficient for large problems where speed is more important than precision.

Value

numeric vector of densities

Note

For normal-normal mixture, set $df=Inf$. When this is the case, approximation is ignored.

Author(s)

Long Qu

References

Broda, Simon and Paoletta, Marc S. (2007) Saddlepoint approximations for the doubly noncentral t distribution, *Computational Statistics & Data Analysis*, 51,6, 2907-2918.

Young, G.A. and Smith R.L. (2005) *Essentials of statistical inference*. Cambridge University Press. Cambridge, UK.

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[dt.sad](#), [dt.int2](#), [dt.lap](#)

extrp.pi0

Extrapolate the Estimates of P-value Density at 1 from Subsamples to Estimate the Proportion of True Nulls

Description

This is the second step of the Subsampling-Extrapolation (SubEx) procedure for estimating the proportion of **TRUE** null hypotheses, i.e., π_0 , when a large number of two-sample t-tests are simultaneously performed. It regresses the p-value density estimates at 1 from subsamples over various subsample sizes and extrapolates the curve/plane to infinite sample sizes in each treatment group. This estimated limit is used to estimate π_0 .

Usage

```

extrp.pi0(dat,slope.constraint=TRUE, gamma2.range=2^c(-4,3),
           rate.margin=c(0.5,0.5), plotit=TRUE)
extrp.pi0.only(n1,n2,y,gam2)
extrp.pi0.slope(n1,n2,y,gam2,eps=1e-5)
extrp.pi0.rate(n1,n2,y,gam2,rate.interval=c(.3,2),eps=1e-5)
extrp.pi0.both(n1,n2,y,gam2,rate.interval=c(.3,2),eps=1e-5)
extrp.pi0.gam2(n1,n2,y,gam2.interval=c(1e-3,6))
extrp.pi0.slope.gam2(n1,n2,y,gam2.interval=c(1e-3,6),eps=1e-5)
extrp.pi0.rate.gam2(n1,n2,y,gam2.interval=c(1e-3,6),rate.interval=c(.3,2),eps=1e-5)
extrp.pi0.both.gam2(n1,n2,y,gam2.interval=c(1e-3,6),rate.interval=c(.3,2),eps=1e-5)

```

Arguments

`dat` an object of class `subt`; typically resulting from calling the function `subt`.

`slope.constraint` logical: whether slope a should be constrained to be positive

`gamma2.range`, `gam2.interval` a numeric vector of length 2, defining the appropriate range of the gamma square parameter. When they are equal, it is assumed as known.

`rate.margin` a numeric vector of length 2, defining the margin of c parameter. When they are equal, it is assumed as known.

`plotit` logical: whether plot should be produced

`n1,n2` subsample size vectors for each of the two treatment groups.

`y` a numeric vector of estimated π_0 at the corresponding subsample sizes.

`gam2` gamma square value, assumed to be known.

`rate.interval` a numeric vector of length 2 defining the appropriate range of rate parameter.

`eps` a small number of tolerance.

Details

Two regression functions may be used, as specified by `nparm`. One is assuming the nonzero standardized effect sizes have a marginal distribution of zero mean normal distribution with variance γ^2 . This regression function has two parameters,

$$f_1 = (1 - \pi_0) \sqrt{\frac{n_1 + n_2}{n_1 + n_2 + n_1 n_2 \gamma^2}} + \pi_0$$

where f_1 is the density estimates at 1 for subsamples, n_1 and n_2 are the corresponding subsampling sizes, $0 \leq \pi_0 \leq 1$, and $\gamma^2 > 0$.

The other regression function is more flexible by replacing the square root and the $1 - \pi_0$ term with another two parameters:

$$f_1 = a \left(\frac{n_1 + n_2}{n_1 + n_2 + n_1 n_2 \gamma^2} \right)^c + \pi_0$$

subject to additional constraints of $a > 0$ and $c > 0$.

It is highly recommended to have **rgl** package available to display the estimated regression surface and possibly rotate it with the mouse.

Value

an object of class `extrpi0`, which is a numeric vector of length 1, named "`pi0`", giving the estimated π_0 , with the following attributes:

```
attr(,'fitted.obj')
      a list, which is the object returned by FUN.
attr(,'nparm')
      the same as the first element of nparm.
attr(,'extrpFUN')
      the same as FUN.
attr(,'start.val')
      the first nparm elements of starts.
attr(,'subt.data')
      the same as dat.
```

Note

Only `extrap.pi0` is expected to be called by a user. Other functions are called within this master function. But if problem occurs, the user may call each individual function to perform the extrapolation. These functions differ in the free parameters (shown in the function names) to be estimated.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[subt](#), [subex](#), [constrOptim](#), [optimize](#), [lsei](#), [print.extrpi0](#), [plot.extrpi0](#).

Examples

```
## Not run:
set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
## this is how the 'simulatedSubt' object in this package generated
simulatedSubt=subt(simulatedDat,balanced=FALSE,max.reps=Inf)
## this is how the 'simulatedExtrpi0' data set in this package generated
simulatedExtrpi0=extrp.pi0(simulatedSubt)
```

```
## End(Not run)
data(simulatedExtrpi0)
summary(simulatedExtrpi0)
```

fdr *False Discovery Rate (FDR) Estimation Based on a Given Estimate of Pi0*

Description

This function estimate the [qvalue](#) based on p-values and an estimate of the proportion of true null hypotheses, pi0.

Usage

```
fdr(p, pi0 = 1)
```

Arguments

p a numeric vector of p-values

pi0 numeric, a given estimate of the proportion of true null hypotheses, π_0 , truncated to [0,1]. The default pi0=1 is the conservative *Benjamini and Hochberg (1995)* version.

Details

The estimation of q-value/FDR is the simple and quick plug-in method:

$$q_i = \min_{i \leq j \leq G} G * \pi_0 * p_{(j)} / j$$

Value

a numeric vector of the same length as p, giving the estimated q-values corresponding to each p-value.

Note

This implementation avoids explicit loops and is much faster when the number of p-values are very large.

Author(s)

Long Qu

References

- Benjamini, Y., Hochberg, Y. (1995) Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society, Series B (Methodological)* 57, 289-300
- Storey, J.D., Tibshirani, R. (2003) Statistical significance for genomewide studies. *The Proceedings of the National Academy of Sciences* 100, 9440-9445

See Also

[qvalue](#), [mt.rawp2adjp](#)

Examples

```
set.seed(9992722)
pvals=runif(5e5)^1.5 ## simulate some fake 'p-values'
library(qvalue)
qvalObj=qvalue(pvals) ## warning: this may be slow!
fdrObj=fdr(pvals,qvalObj$pi0)
all.equal(fdrObj,qvalObj$qval) ## should be TRUE
```

fitted

Density evaluated at observed statistics for npest class

Description

Density evaluated at observed statistics for npest class

Usage

```
## S3 method for class 'nparncpt'
fitted(object, ...)

## S3 method for class 'parncpt'
fitted(object, ...)

## S3 method for class 'sparncpt'
fitted(object, ...)

## S3 method for class 'nparncpp'
fitted(object, ...)
```

Arguments

object object of class nparncpt, parncpt, sparncpt or nparncpp
 ... other arguments passed to dtn.mix

Value

numeric vector of densities at each observed statistic of object

Note

Functions for nparncpp are not yet implemented.

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[sparncpt](#), [parncpt](#), [nparncpt](#), [nparncpp](#)

fitted.discTMix

Density evaluated at observed statistics for discTMix class

Description

Density evaluated at observed statistics for discTMix class

Usage

```
## S3 method for class 'discTMix'  
fitted(object, ...)
```

Arguments

object	object of class discTMix
...	Not used currently.

Value

numeric vector of densities at each observed statistic of object

Author(s)

Long Qu

References

Pawitan Y, Krishna Murthy KR, Michiels S, Ploner A (2005) Bias in the estimation of false discovery rate in microarray studies, *Bioinformatics*.

See Also

[sparncpt](#), [parncpt](#), [nparncpt](#), [nparncpp](#), [discTMix](#)

geomean and harmean *Geometric mean and harmonic mean functions*

Description

Geometric mean and harmonic mean functions

Usage

```
geomean(x)
harmean(x)
```

Arguments

x numeric vector

Value

numeric scalar. For geomean, this is the geometric mean of x; for harmean, this is the harmonic mean of x.

Author(s)

Long Qu

Examples

```
geomean(10^(1:10)) #[1] 316227.8
harmean(10^(1:10)) #[1] 90
```

gjack	<i>Generalized jackknife</i>
-------	------------------------------

Description

Generalized jackknife bias correction

Usage

```
gjack(theta1, theta2, R)
```

Arguments

theta1	Numeric
theta2	Numeric
R	Numeric

Details

Computes $(\text{theta1}-R*\text{theta2})/(1-R)$

Value

Numeric

Author(s)

Long Qu

References

Henry L. Gray, W. R. Schucany. 1972. The generalized jackknife statistic. New York, M. Dekker.

grid.search	<i>Performs a grid search to minimize the objective function</i>
-------------	--

Description

Performs a grid search to minimize the objective function

Usage

```
grid.search(obj, lower, upper, ngrid, ...)
```

Arguments

obj	objective function to be minimized
lower	numeric vector giving the lower bound of grid for each dimension
upper	numeric vector giving the upper bound of grid for each dimension
ngrid	numeric vector giving the number of points each dimension
...	other arguments passed to obj

Details

This function first call `expand.grid` then evaluate `obj` to find a minimum. The number of calls to `obj` is `prod(ngrid)`. This is useful for finding a good starting values for many optimization routines.

Value

a numeric vector of the parameter that minimizes `obj`

Author(s)

Long Qu

See Also

[optim](#)

histf1

Histogram estimator of p-value density evaluated at 1

Description

Histogram estimator of p-value density evaluated at 1. See references.

Usage

```
histf1(p,max.bins=20,bin.method=c("max","nmse","bootstrap",  
  "Sturges","Scott","FD"),discrete=FALSE,seq.perm=FALSE,  
  nboots=200,plotit=FALSE,perm.n,perm.h,...)
```

Arguments

p	Vector of p-values
max.bins	maximum number of bins
bin.method	binning method
discrete	Whether p-values are discrete
seq.perm	Whether p-values come from sequential permutation tests
nboots	bootstrap sample size
plotit	Whether to plot the histogram
perm.n	n for sequential permutation tests
perm.h	h for sequential permutation tests
...	Other arguments passed to hist

Value

Numeric scalar

Author(s)

Long Qu

References

Nettleton, Hwang, Caldo, Wise. 2006. Estimating the number of true null hypotheses from a histogram of p values. *Journal of Agricultural, Biological, and Environmental Statistics*. 11. 337-356.

Bancroft and Nettleton. 2009. Estimation of False Discovery Rate Using Permutation P-values with Different Discrete Null Distributions. Iowa State University Department of Statistics Preprint Series, #2009-05.

Bancroft and Nettleton. 2009. Computationally Efficient Estimation of False Discovery Rate Using Sequential Permutation P-values. Iowa State University Department of Statistics Preprint Series, #2009-04.

See Also

[lastbin](#), [qvalue](#)

Examples

```
set.seed(9992722)
histf1(runif(5e5)^1.5) ## [1] 0.6762
```

`lastbin`*Histogram estimator of p-value density evaluated at 1*

Description

This function reports the density estimate of the right most bin of histogram of p-values.

Usage

```
lastbin(p, bw = 0.2, trunc = TRUE)
```

Arguments

<code>p</code>	a numeric vector the p-values
<code>bw</code>	numeric, the bin width of histogram of p-values
<code>trunc</code>	logical, indicating if the resulting estimate should be truncated to within [0,1].

Details

This is a very fast and cheap estimate of p-value density at one, with a slight positive bias, because it is only an unbiased estimate of density at the bin center, not the right bin edge. But this is usually ignorable. The function is defined as:

```
function(p,bw=.2,trunc=TRUE)  if(trunc)max(min(1,mean(p>=1-bw)/bw),0) else mean(p>=1-bw)/bw
.
```

Value

a single numeric value as the estimate.

Author(s)

Long Qu

See Also

[qvalue](#), [histf1](#)

Examples

```
set.seed(9992722)
lastbin(runif(5e5)^1.5) ## [1] 0.69511
```

lfdr and ppee	<i>Local False Discovery Rates (lfdr), i.e., Posterior Probability of Equivalent Expression (ppee) for gene expression profiling data.</i>
---------------	--

Description

These functions return $\pi_0 * (\text{null test statistic density}) / (\text{marginal test statistic density})$.

Usage

```
lfdr(object, ...)  
ppee(object, ...)  
## Default S3 method:  
lfdr(object, ...)  
  
## S3 method for class 'parncpt'  
lfdr(object, ...)  
  
## S3 method for class 'sparncpt'  
lfdr(object, ...)  
  
## S3 method for class 'nparncpt'  
lfdr(object, ...)  
  
## S3 method for class 'nparncpp'  
lfdr(object, ...)  
  
## S3 method for class 'CBUM'  
lfdr(object, ...)  
  
## S3 method for class 'znormix'  
lfdr(object, ...)  
  
## S3 method for class 'convest'  
lfdr(object, ...)  
  
## S3 method for class 'discTMix'  
lfdr(object, ...)
```

Arguments

object	an object of corresponding classes.
...	Other arguments currently not used.

Value

A numeric vector of lfdr.

Note

lfdr and ppee are equivalent.

Author(s)

Long Qu

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#), [nparncpp](#), [fdr](#), [CBUM](#), [znormix](#), [convest](#), [discTMix](#)

loadOrInstall

Load packages or install and load packages

Description

This functino loads packages if they are installed already. If failed, it tries to install the package automatically and re-load the package after installation.

Usage

```
loadOrInstall(package, dependencies="Depends",...)
```

Arguments

package	A character vector of the names of packages to be loaded.
dependencies	The same as in install.packages .
...	Additional arguments passed to require , install.packages .

Details

The function calls [require](#) to see if the package can be loaded. If not, it calls [install.packages](#). Then [require](#) is called again for the installed package.

Value

A logical scalar. TRUE if all packages are loaded sucessfully. If FALSE, its "failed" attribute contains a vector of package names that failed to be loaded.

Author(s)

Long Qu

See Also

[library](#), [require](#), [install.packages](#)

logLik.ncest	<i>log likelihood from an object of class ncest</i>
--------------	---

Description

log likelihood from an object of class ncest. This could be penalized likelihood in the case of a nparncpt object.

Usage

```
## S3 method for class 'ncest'  
logLik(object,...)
```

Arguments

object	the object of class ncest
...	currently not used

Details

Extract the logLik component. This is used by [AIC](#). The df is the estimated effective number of parameters.

Value

an object of class [logLik](#)

Author(s)

Long Qu

See Also

[logLik](#), [AIC](#), [nparncpt](#), [parncpt](#), [sparncpt](#)

marginal.dt	<i>Estimated arginal density of t-statistics</i>
-------------	--

Description

Estimated arginal density of t-statistics from ncest class

Usage

```
marginal.dt(obj,...)
## S3 method for class 'parncpt'
marginal.dt(obj,...)
## S3 method for class 'nparncpt'
marginal.dt(obj, ...)
## S3 method for class 'sparncpt'
marginal.dt(obj, ...)
```

Arguments

obj an object of `ncest` ([nparncpt](#) or [parncpt](#))
... Other argument passed to `dtn.mix`, most notably, the approximation argument

Details

When `obj$data$df` are all equal to each other, a single marginal density is clearly defined for all `obj$data$tstat`. Otherwise, the marginal density is defined as a discrete mixture of densities, one for each distinct degree of freedom, with mixing proportion based on that of `obj$data$df`.

Value

A function of one argument (x), i.e., the marginal density function.

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#)

matrix.t.test

Apply a Two Sample T-test to Each Row or Column of a Matrix

Description

This function applies the two sample t-test to each row or column of a matrix.

Usage

```
matrix.t.test(x, MARGIN = 1, n1 = if (MARGIN == 1) floor(ncol(x)/2)
             else floor(nrow(x)/2), n2 = if (MARGIN == 1) ncol(x) - n1 else
             nrow(x) - n1, pool = TRUE, pOnly=TRUE, tOnly = FALSE)
```

Arguments

x	a numeric matrix to which the t-test will be applied to, by row or by column.
MARGIN	either 1 or 2. If MARGIN=1, apply the t-test to each row of x; otherwise, if MARGIN=2, apply the t-test to each column of x. See also apply .
n1	sample size of the first group. It should be smaller than the appropriate dim of x.
n2	sample size of the second group. It should be smaller than the appropriate dim of x.
pool	logical, indicating if the variance estimate should be pooled. If FALSE, Welch (i.e. Satterthwaite) approximation to the degrees of freedom is used.
pOnly	logical, indicating if only a vector of p-values should be returned.
tOnly	logical, indicating if only a vector of t-statistics should be returned. This argument overwrites pOnly, if they are conflicting.

Details

This is a much faster function for "almost" the same purpose of apply each MARGIN of x a `t.test`, i.e., the mean of the first n1 elements is compared with the mean of the rest n2 elements, for each row or column depending on the MARGIN. See the Value section for differences.

Value

If `pOnly=TRUE` (the default situation), a numeric vector of p-values is returned, the length of which is determined by MARGIN.

If `tOnly=TRUE`, a numeric vector of t-statistics is returned, the length of which is determined by MARGIN.

If `tOnly=TRUE` and `pOnly=TRUE`, a numeric vector of t-statistics is returned, the length of which is determined by MARGIN, as `tOnly` overwrites `pOnly`.

If `pOnly=FALSE` and `tOnly=FALSE`, a list of three components is returned:

stat	a numeric vector of the t-statistics, one for each row or column, depending on MARGIN.
df	a numeric vector of degrees of freedom. If <code>pool</code> is TRUE, this vector is of length 1, i.e. $n1+n2-2$; if <code>pool</code> is FALSE, this vector is of the same length as <code>stat</code> , depending on MARGIN.
p.value	a numeric vector of p-values, one for each row or column, depending on MARGIN.

Author(s)

Long Qu

See Also

[apply](#), [t.test](#)

Examples

```
set.seed(9992722)
dat=matrix(rnorm(30),3,10)
(pvals=matrix(t.test(dat,1,5,5)) # [1] 0.2112825 0.8366920 0.2891014
(pvals2=apply(dat,1,function(xx)t.test(xx[1:5],xx[6:10],var.equal=TRUE)$p.val))
all.equal(pvals,pvals2) ## TRUE
```

mTruncNorm	<i>Moments of truncated normal distribution and the integral in the non-central t-distribution</i>
------------	--

Description

Compute the moments of truncated normal distribution and the integral that appears in the noncentral t-distribution

Usage

```
mTruncNorm(r = 1, mu = 0, sd = 1, lower = -Inf, upper = Inf,
           approximation = c("int2", "laplace", "numerical"),
           integral.only = FALSE, ...)
mTruncNorm.int2(r = as.integer(1), mu = 0, sd = 1, lower = -Inf,
               upper = Inf, takeLog = TRUE, ndiv = 8)
```

Arguments

r	the order of moments to be computed. It could be noninteger, but has to be non-negative. This is also the degrees of freedom for the noncentral t-distribution.
mu	mean of the normal distribution, before truncating.
sd	SD of the normal distribution, before truncating.
lower	lower truncation point
upper	upper truncation point
approximation	Method of approximation. <i>int2</i> is exact for <i>integer</i> r and <i>interpolate</i> to non-integer r. <i>laplace</i> uses laplacian approximation. <i>numerical</i> uses numerical integration.
integral.only	logical. If TRUE, only the integral in noncentral t-distribution is returned. Otherwise, it is normalized to be the rth moments of truncated normal distribution.
takeLog	logical. If TRUE and r is not an integer, the polynomial interpolation will be on the log scale. But final result is on the original scale.
ndiv	number of points with closes integer r to be used in polynomial interpolation.
...	other arguments passed to <code>mTruncNorm.int2</code>

Details

mTruncNorm.int2 uses iterative relation over r to compute the integral iteratively starting from $r=0$ and $r=1$ whose analytic results are available. If r is not an integer, the nearest `ndiv` nonnegative integer r will be used to do divided difference polynomial interpolation.

Value

numeric vector. If `integral.only` is TRUE, this is the integral in the noncentral t-density; otherwise this is the r th moments of truncated normal distribution.

Author(s)

Long Qu

See Also

[dt](#), [pt](#), [dt.int2](#)

nparncpp

Estimation of the density of absolute noncentrality parameters

Description

Estimation of the density of absolute noncentrality parameters, using linear B-spline model.

Usage

```
nparncpp(p,
  breaks=min(2000,round(length(p)/5)),
  test=c("t","z"),
  df,
  alternative=c("two.sided", "less", "greater"),
  compromise.n=1,
  lambdas=#if(penalty_type==1)10^seq(-2,6,length=6) else
          10^seq(-4,6,length=11),
  deltamax='auto',
  nknots,
  ndelta=500,
  solver=c("lsei", "LowRankQP", "solve.QP", "ipop"),
  weights=1,
  keep.cdf=TRUE,
  LowRankQP.method=c('LU', 'CHOL'),
  lsei.method=c('chol', 'svd', 'eigen'),
  debugging=FALSE,
  ...)
```

Arguments

p	p-value vector
breaks	break points to bin the p-values
test	either t-test or z-test
df	degrees of freedom for the test
alternative	Same as in t.test
compromise.n	Number of components in the compromised estimate
lambdas	Candidate tuning parameters
deltamax	Assumed maximum noncentrality parameters
nknots	Number of knots
ndelta	Number of points to evaluate the noncentrality parameters
solver	Quadratic programming solver function
weights	Bin weights
keep.cdf	Logical: whether computed conditional CDF is saved in global environment. See cond.cdf .
LowRankQP.method	Method for LowRankQP
lsei.method	Method for lsei
debugging	Logical: print excessive messages
...	Additional arguments to solver

Value

An object of class `c('nparncpp', 'ncpest')`.

Note

The code right now is not completely compatible with the `ncpest` class and is subject to change in future versions.

Author(s)

Long Qu translated and enhanced the original MATLAB code from Dr. David Ruppert.

References

Ruppert, Nettleton, Hwang. (2007) Exploring the Information in p -values for the Analysis and Planning of Multiple-test Experiments. *Biometrics*. 63. 483-495.

See Also

[nparncpt](#), [sparncpt](#), [parncpt](#), [dnpc](#)

nparncpp.iter	<i>iterative call to the nparncpp function</i>
---------------	--

Description

A wrapper to iteratively call the `nparncpp` function

Usage

```
nparncpp.iter(p, estimates=c("all", "compromise", "pi0", "f1"), iter=2,  
             weights, eps=1e-6, keep.cdf=TRUE, ...)
```

Arguments

<code>p</code>	p-value vector
<code>estimates</code>	Character: what to estimate
<code>iter</code>	max number of iterations
<code>weights</code>	bin weights
<code>eps</code>	Small tolerance number
<code>keep.cdf</code>	Logical, as in <code>nparncpp</code>
<code>...</code>	other arguments to <code>nparncpp</code>

Value

An object of class `c('nparncpp', 'ncest')`

Author(s)

Long Qu

References

Ruppert, Nettleton, Hwang. 2007. Exploring the Information in p -values for the Analysis and Planning of Multiple-test Experiments. *Biometrics*. 63. 483-495.

be set to zero if it is below the average of all $K \times \text{length}(\text{tstat})$ such values. If it is set to FALSE or 0, then none of the density will be treated as zero, no matter how small they are. If it is set to a number between 0 and 1, values below this quantile will be treated as zero. Note that this approximation only affects the computation of hessian matrix, which does not need to be exact in an optimization routine. Hence, a reasonable sparseness speeds up computation of a hessian matrix but might increase the number of iterations to converge. Set this to TRUE seems a reasonable trade-off between the two effects and usually saves computing time.

... other paramters passed to `dtn.mix`. Usually, the approximation argument.

Details

nparncpt is a wrapper for `nparncpt.sqp`, the latter of which uses a sequential quadratic programming algorithm to find the mixing proportions of the basis Gaussian density functions.

Value

A list with class attribute `c("nparncpt", "ncpest")`

<code>pi0</code>	estimated proportion of true nulls
<code>mu.ncp</code>	mean of ncp
<code>sd.ncp</code>	SD of ncp
<code>logLik</code>	an object of class <code>logLik</code> . The associated <code>df</code> is the estimated effective number of parameters (<code>enp</code>). The log likelihood is also penalized likelihood. See also logLik.ncpest and AIC .
<code>enp</code>	estimated ENP
<code>par</code>	estimated parameters theta
<code>lambda</code>	the lambda that minimizes NIC
<code>gradient</code>	analytic gradient at the estimate
<code>hessian</code>	analytic hessian at the estimate
<code>beta</code>	estimated mixing proportions for the NCP distribution
<code>IC</code>	the information criterion specified by the user
<code>all.mus</code>	mean of each basis Gaussian density
<code>all.sigs</code>	SD of each basis Gaussian density
<code>data</code>	a list of <code>tstat</code> and <code>df</code>
<code>i.final</code>	the index of lambdas that minimizes NIC
<code>all.pi0s</code>	estimated <code>pi0</code> for each lambda
<code>all.enps</code>	ENP for each lambda
<code>all.thetas</code>	parameter estimates for each lambda
<code>all.nics</code>	Network information criterion (NIC) for each lambda
<code>all.nic.sd</code>	SD of NIC for each lambda
<code>all.lambdas</code>	the lambdas argument itself

Note

df could be Inf for z-tests. When this is the case, approximation is ignored.

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [sparncpt](#), [fitted.nparncpt](#), [plot.nparncpt](#), [summary.nparncpt](#), [coef.ncpest](#), [logLik.ncpest](#), [vcov.ncpest](#), [AIC](#), [dncp](#)

Examples

```
## Not run:
data(simulatedTstat)
(npfit=parncpt(tstat=simulatedTstat, df=8));
(pfit=parncpt(tstat=simulatedTstat, df=8, zeromean=FALSE)); plot(pfit)
(pfit0=parncpt(tstat=simulatedTstat, df=8, zeromean=TRUE)); plot(pfit0)
(spfit=sparncpt(npfit,pfit)); plot(spfit)

## End(Not run)
```

parncpt

Parametric estimation of noncentrality parameter distribution

Description

Assuming normality of noncentrality parameters, the MLE of its standard deviation (and possibly mean also) is estimated from observed t -statistics

Usage

```
parncpt(tstat, df, zeromean = TRUE, ...)
parncpt.bfgs.0mean(tstat, df, starts, grids, approximation = "int2", ...)
parncpt.bfgs.non0mean(tstat, df, starts, grids, approximation = "int2", ...)
parncpt.momeff(tstat, n1, n2=n1, zeromean, gamma2, lower.df=6.1, upper.df=100, approx=TRUE)
```

Arguments

tstat	numeric vector of t-statistics
df	numeric vector of degrees of freedom
zeromean	logical; if TRUE, then mean of noncentrality parameters is assumed to be zero and is <i>not</i> estimated.
...	Other arguments to optim
starts	An optional vector of starting values. If missing, a grid search will be performed to get a good starting value.
grids	A list of three components (lower, upper, ngrid) defining the grids to be searched in find a good starting value. Each component is a numeric vector of the same length as the number of parameters. lower and upper give the bounds, and ngrid specifies the number of points for each dimension.
approximation	Methods of approximating the noncentral t-density. int2 is exact for integer df, but interpolate to fractional df. 'laplace' is the laplacian approximation; 'saddlepoint' is the saddlepoint approximation; 'none' computes the (sort of) exact density using the default dt function.
n1	Treatment 1 sample size
n2	Treatment 2 sample size
gamma2	Gamma square parameter, i.e., variance of effect sizes.
lower.df	lower bound of degrees of freedom, in case of n1 is missing
upper.df	upper bound of degrees of freedom, in case of n1 is missing
approx	logical, indicating if no exact solutions are available, whether approx. solutions are returned.

Details

parncpt calls either `parncpt.bfgs.0mean` or `parncpt.bfgs.non0mean`, depending whether `zeromean` is TRUE or FALSE. Both `parncpt.bfgs.0mean` and `parncpt.bfgs.non0mean` use the 'L-BFGS-B' algorithm by calling [optim](#). All gradients are analytical, but the Hessian is only numerical approximation. The first parameter is always π_0 , i.e., the proportion of true null hypotheses; the last parameter is always the standard deviation of noncentrality parameters; for `parncpt.bfgs.non0mean` the middle parameter is the mean of noncentrality parameters, whereas for `parncpt.bfgs.0mean` the mean is set to 0 a priori.

Value

a list with class attribute being `c('parncpt', 'ncpest')`.

π_0	proportion of true nulls
mu.ncp	mean of ncp
sd.ncp	SD of ncp
data	a list of tstat and df
logLik	an object of class <code>logLik</code> . Call <code>logLik.ncpest</code> to extract. Similarly, <code>AIC</code> is callable.

enp	the (effective) number of parameters in the model
par	estimated parameters. Call <code>coef.ncpest</code> to extract.
obj	the negative loglikelihood function that is minimized
gradient	analytic gradient at the estimate
hessian	numeric hessian at the estimate

Note

df could be Inf for z-tests. When this is the case, approximation is ignored.

parncpt.momeff is the old code using method of moments estimates. It is outdated, depreciated, and not completely compatible with current ncest class.

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[sparncpt](#), [nparncpt](#), [fitted.parncpt](#), [plot.parncpt](#), [summary.parncpt](#), [coef.ncpest](#), [logLik.ncpest](#), [vcov.ncpest](#), [AIC](#), [dnpc](#)

Examples

```
## Not run:
data(simulatedTstat)
(npfit=nparncpt(tstat=simulatedTstat, df=8));
(pfit=parncpt(tstat=simulatedTstat, df=8, zeromean=FALSE)); plot(pfit)
(pfit0=parncpt(tstat=simulatedTstat, df=8, zeromean=TRUE)); plot(pfit0)
(spfit=sparncpt(npfit,pfit)); plot(spfit)

## End(Not run)
```

pavaf1

pooling adjacent violator algorithm estimate of p-value density at 1

Description

pooling adjacent violator algorithm estimate of p-value density at 1

Usage

```
pavaf1(p,max.bins=20,bin.method=c("max","Sturges","Scott","FD"),
       discrete=FALSE,plotit=FALSE,...)
```

Arguments

p	p-value vector
max.bins	max number of bins
bin.method	binning method
discrete	logical: whether p-values are discrete.
plotit	logical: whether results are plotted.
...	Other arguments to hist

Details

This function bin the p-values and then run PAVA to estimate the minimum of its density.

Value

Numeric scalar

Author(s)

Long Qu

pdf.dist	<i>Distance between densities</i>
----------	-----------------------------------

Description

Compute the distance between two density functions

Usage

```
pdf.dist(f1, f2, method = c("Hellinger", "abdif"))
```

Arguments

f1, f2	Two functions of one argument, both of which are densities defined over the whole real line.
method	character; specifying the definition of distance. Current choices are Hellinger and abdif, the latter of which is the integrated absolute differences between the two function.

Details

Numerical integration is performed from $-\infty$ to ∞ . Hence, the two functions must be able to accept arguments over the whole real line.

Value

a numeric scalar which is the computed distance, or `NA_real_` if any problem occurs.

Author(s)

Long Qu

Examples

```
# Hellinger distance between standard normal and log-normal
pdf.dist(dnorm, dlnorm, 'Hell') # 0.5981035

# absolute difference between standard normal and standard cauchy
f2=function(x)dt(x,1)
pdf.dist(dnorm, f2, 'abd') #[1] 0.5023312
```

plot.extrpi0

Plotting the Estimated Regression Surface

Description

This function plots the regression surface, overlaid with 3D scatter plot, for objects of class `extrpi0`, typically resulting from calling the function `extrp.pi0`.

Usage

```
## S3 method for class 'extrpi0'
plot(x,y,rgl=TRUE,...)
```

Arguments

<code>x</code>	the <code>extrpi0</code> object.
<code>y</code>	the same as <code>rgl</code> . If not missing, it overrides <code>rgl</code> .
<code>rgl</code>	logical, specifying whether or not the rgl package is used for making better 3D interactive graphs.
<code>...</code>	other arguments to be passed to either <code>persp3d</code> , or <code>persp</code> if rgl is not available.

Value

an `invisible(NULL)`. Used for side effects only.

Note

When **rgl** is not available, a warning is always generated.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[rgl.extrp.pi0](#)

Examples

```
## Not run:
set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
## this is how the 'simulatedSubt' object in this package generated
simulatedSubt=subt(simulatedDat,balanced=FALSE,max.reps=Inf)
## this is how the 'simulatedExtrpi0' data set in this package generated
simulatedExtrpi0=extrp.pi0(simulatedSubt,plotit=FALSE)
plot(simulatedExtrpi0)
plot(simulatedExtrpi0,FALSE)

## End(Not run)
```

plot.nparncpt

plot an object of class nparncpt, i.e., nonparametric estimate of non-centrality parameters

Description

Plot the Network information criterion (NIC), effective number of parameters (ENP), and estimated proportion (π_0) of true null hypotheses for different choices of tuning parameters; also plot the estimated density of noncentrality parameters

Usage

```
## S3 method for class 'nparncpt'
plot(x, ...)
```

Arguments

x an object of class nparncpt
 . . . currently not used.

Details

For NIC, only values within 2 s.e.'s of the minimum are shown. The solid line on NIC, ENP and pi0 shows the final tuning parameter, i.e., the one that minimizes NIC.

Value

Invisible par.

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[nparncpt](#), [sparncpt](#), [parncpt](#)

Examples

```
## Not run:
data(simulatedTstat)
(npfit=nparncpt(tstat=simulatedTstat, df=8, plotit=FALSE)); plot(npfit)
(pfit=parncpt(tstat=simulatedTstat, df=8, zeromean=FALSE)); plot(pfit)
(pfit0=parncpt(tstat=simulatedTstat, df=8, zeromean=TRUE)); plot(pfit0)
(spfit=sparncpt(npfit,pfit)); plot(spfit)

## End(Not run)
```

plot.parncpt

plot an object of class parncpt, i.e., parametric estimate of noncentrality parameters

Description

Plot the histogram of observed t -statistics together with its fitted density estimate; also plotted is the estimated density of noncentrality parameters.

Usage

```
## S3 method for class 'parncpt'  
plot(x,...)
```

Arguments

x	an object of class parncpt
...	currently not used

Details

Left panel shows the density estimate of observed t-statistics, overlapped with a histogram; right panel shows the estimated density of noncentrality parameters. Solid line is the actual mean of the estimate; dashed line is located at zero.

Value

the [invisible](#) x itself

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#)

Examples

```
## Not run:  
data(simulatedTstat)  
(npfit=nparncpt(tstat=simulatedTstat, df=8));  
(pfit=parncpt(tstat=simulatedTstat, df=8, zeromean=FALSE)); plot(pfit)  
(pfit0=parncpt(tstat=simulatedTstat, df=8, zeromean=TRUE)); plot(pfit0)  
(spfit=sparncpt(npfit,pfit)); plot(spfit)  
  
## End(Not run)
```

plot.sparncpt	<i>plot an object of class sparncpt, i.e., semiparametric estimate of non-centrality parameters</i>
---------------	---

Description

Plot the histogram of observed t-statistics together with its fitted density estimate; also plotted is the estimated density of noncentrality parameters.

Usage

```
## S3 method for class 'sparncpt'  
plot(x, ...)
```

Arguments

x	an object of class sparncpt
...	currently not used

Details

Left panel shows the density estimate of observed t-statistics, overlapped with a histogram; right panel shows the estimated density of noncentrality parameters. Solid line is the actual mean of the estimate; dashed line is located at zero.

Value

the `invisible` x itself

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#)

Examples

```
## Not run:
data(simulatedTstat)
(npfit=mparncpt(tstat=simulatedTstat, df=8));
(pfit=parncpt(tstat=simulatedTstat, df=8, zeromean=FALSE)); plot(pfit)
(pfit0=parncpt(tstat=simulatedTstat, df=8, zeromean=TRUE)); plot(pfit0)
(spfit=sparncpt(npfit,pfit)); plot(spfit)

## End(Not run)
```

plot.subex

*Plotting the P-value, Q-values, and the Regression Surface***Description**

This function plots the p-value, q-values, and the regression surface for an object of class `subex`, typically from calling the function `subex`.

Usage

```
## S3 method for class 'subex'
plot(x,y,rgl = TRUE,...)
```

Arguments

<code>x</code>	the <code>subex</code> object
<code>y</code>	the same as <code>rgl</code> . If not missing, it overrides <code>rgl</code> .
<code>rgl</code>	logical, specifying whether or not the rgl package is used for making better 3D interactive graphs.
<code>...</code>	other arguments to be passed to <code>persp3d</code> , or <code>persp</code> if rgl package is not available.

Details

Two plots will be generated. The first one is a histogram of p-values in the blue color. A horizontal blue line is added indicating the height of π_0 , i.e., the proportion of true null hypotheses. This histogram is overlaid with a red line of FDRs, indicating the corresponding q-value for each p-value. The right most end, i.e., the q-value corresponding to a p-value of 1, is the also of height π_0 . The other plot is the same as the `plot.extrpi0`.

Value

an `invisible(NULL)`, used for side effects only.

Note

Because this function will call `plot.extrpi0`, a warning will be generated when **rgl** package is not available.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[plot.extrpi0](#), [rgl](#), [subex](#)

Examples

```
## Not run:
set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
## this is how the 'simulatedSubex' object in this package generated
simulatedSubex=subex(simulatedDat,balanced=FALSE,max.reps=Inf,plotit=FALSE)
plot(simulatedSubex)
plot(simulatedSubex,FALSE)

## End(Not run)
```

plot.subt

3D Scatter Plot of Subsample Sizes and P-value Density at One.

Description

This function generates a 3d scatter plot for objects of class `subt`, typically resulting from calling the function `subt`.

Usage

```
## S3 method for class 'subt'
plot(x,y,rgl=TRUE,...)
```

Arguments

`x` the `subt` object.

`y` the same as `rgl`. If not missing, it overrides `rgl`.

`rgl` logical, specifying whether or not the `plot3d` in the `rgl` package is used. If `FALSE`, `scatterplot3d` will be used.

`...` other arguments to be passed to either `plot3d`, or `scatterplot3d`.

Value

an invisible(NULL). Used for side effects only.

Note

When **rgl** is not available, a warning is always generated.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[rgl.subt](#)

Examples

```
## Not run:
set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
## this is how the 'simulatedSubt' object in this package generated
simulatedSubt=subt(simulatedDat,balanced=FALSE,max.reps=Inf)
plot(simulatedSubt)
plot(simulatedSubt,FALSE)

## End(Not run)
```

print.extrpi0

Print the Summary of the Extrapolation

Description

This function print out the summary of an extrpi0 object, typically from calling the function [extrp.pi0](#).

Usage

```
## S3 method for class 'extrpi0'
print(x,...)
## S3 method for class 'extrpi0'
summary(object, ...)
```

Arguments

`x, object` the `extrpi0` object, for which to print summaries.
`...` ignored.

Value

an invisible(NULL), used for side effects only.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[extrp.pi0](#)

Examples

```
## Not run:
set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
## this is how the 'simulatedSubt' object in this package generated
simulatedSubt=subt(simulatedDat,balanced=FALSE,max.reps=Inf)
## this is how the 'simulatedExtrpi0' data set in this package generated
simulatedExtrpi0=extrp.pi0(simulatedSubt)

## End(Not run)
data(simulatedExtrpi0)
print(simulatedExtrpi0)
```

`print.subex`

Printing a Summary of Subsampling-Extrapolation Results

Description

This function prints out a summary of a `subt` object.

Usage

```
## S3 method for class 'subex'
print(x,...)
## S3 method for class 'subex'
summary(object,...)
```

Arguments

x, object the subex object, for which to print summaries.
... ignored.

Details

This function will first print a summary of the corresponding `extrpi0` object. Then several quantiles of the p-values and q-values are printed.

Value

an invisible(NULL), used only for its side effects.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[subex](#), [extrp.pi0](#), [print.extrpi0](#), [fdr](#)

Examples

```
## Not run:  
set.seed(9992722)  
## this is how the 'simulatedDat' data set in this package generated  
simulatedDat=sim.dat(G=5000)  
## this is how the 'simulatedSubex' data set in this package generated  
simulatedSubex=subex(simulatedDat,balanced=FALSE,max.reps=Inf)  
  
## End(Not run)  
data(simulatedSubex)  
summary(simulatedSubex)
```

print.subt

Print the Summary of the Subsampling T-tests

Description

This function print out the summary of an `subt` object, typically from calling the function [subt](#).

Usage

```
## S3 method for class 'subt'  
print(x,...)  
## S3 method for class 'subt'  
summary(object,...)
```

Arguments

x, object	the subt object, for which to print summaries.
...	ignored.

Value

an invisible(NULL), used for side effects only.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[subt](#)

Examples

```
## Not run:  
set.seed(9992722)  
## this is how the 'simulatedDat' data set in this package generated  
simulatedDat=sim.dat(G=5000)  
## this is how the 'simulatedSubt' object in this package generated  
simulatedSubt=subt(simulatedDat,balanced=FALSE,max.reps=Inf)  
  
## End(Not run)  
data(simulatedSubt)  
print(simulatedSubt)
```

reflect and fold	<i>Reflection and folding of a function about zero</i>
------------------	--

Description

reflect reflects the function defined on the nonnegative real line about zero to get a function defined on the whole real line, and then divide it by 2. fold folds a function defined on the whole real line at zero to get a function defined only on the non-negative real line.

Usage

```
reflect(f, ...)  
fold(f, ...)
```

Arguments

f	the function to be reflected or folded
...	other arguments passed to f

Details

See examples.

Value

the new function

Author(s)

Long Qu

Examples

```
## reflect function is currently defined as  
function(x,...) ifelse(x>0, f(x,...), f(-x,...))/2  
  
## fold function is currently defined as  
function(x,...) ifelse(x>=0, f(x,...)+f(-x,...), 0)  
  
## double exponential pdf  
ddexp=reflect(dexp)  
  
## folded normal pdf  
dfnorm=fold(dnorm)
```

sim.dat

*Simulating a Microarray Data Set***Description**

This function simulates a two-group comparison microarray data set according to a hierarchical model, where the standardized effect sizes across all genes are assumed to be independently and identically distributed. This distribution is a two-component mixture. It has probability π_0 of being zero; and probability $1 - \pi_0$ of being from another distribution. The observed values are simulated independently conditional on the standardized effect sizes.

Usage

```
sim.dat(G = 10000, pi0 = 0.75, gamma2 = 1, n1 = 5, n2 = n1,
        errdist = rnorm, effdist = function(g, gamma2)
        rnorm(g, , sqrt(gamma2)), ErrArgs, EffArgs)
```

Arguments

G	a numeric positive integer, the number of genes
pi0	a numeric value between 0 and 1, the proportion of non-differentially expressed genes.
gamma2	a positive value, which is always the second argument passed to effdist. If the nonzero standardized effect sizes have a zero normal distribution, this is the variance of this distribution. The larger it is, the larger the mean absolute effects are.
n1	a positive integer, the sample size in treatment group 1.
n2	a positive integer, the sample size in treatment group 2.
errdist	a function, which simulate K random errors, where K is the first argument of errdist. The second argument is always ErrArgs, if it is not missing.
effdist	a function, which simulate G1 standardized effect sizes, where G1 is the first argument of effdist. The second argument is always gamma2. The third argument is always EffArgs, if it is not missing.
ErrArgs	a list of additional arguments used by errdist.
EffArgs	a list of additional arguments used by effdist.

Details

The function simulates $G * N$ errors according to errdist, where $N = n_1 + n_2$. The results are organized into a G-by-N matrix. The G_1 standardized effect sizes are simulated according to effdist, controlled by the parameter gamma2, where $G_1 = \text{round}(G * \pi_0)$. Then, each column of the upper-left G_1 -by- n_1 submatrix were added by the simulated effect sizes.

Value

a G-by- (n_1+n_2) matrix.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

Examples

```
set.seed(54457704)
## an unusually small data set of 20 genes and 3 samples in each of the two treatment groups.
dat=sim.dat(G=20, n1=3,n2=3)

set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
```

simulatedDat

A Simulated Microarray Data Set

Description

This is the result from calling [sim.dat](#)

Usage

```
data(simulatedDat)
```

Format

a matrix with 5000 rows and 10 columns. The first 5 columns correspond to treatment one, and the rest 5 columns correspond to treatment 2. Each row corresponds to a gene.

Details

```
This is the result from calling
set.seed(9992722)
simulatedDat=sim.dat(G=5000)
```

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[sim.dat](#)

Examples

```
data(simulatedDat)
```

simulatedExtrpi0	<i>A Simulated 'extrpi0' Object</i>
------------------	-------------------------------------

Description

This is the result from calling `extrp.pi0` on `simulatedSubt`.

Usage

```
data(simulatedExtrpi0)
```

Format

an object of class `extrpi0`.

Details

This is the result from calling
`data(simulatedSubt)`
`simulatedExtrpi0=extrp.pi0(simulatedSubt)`

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

`extrp.pi0`

Examples

```
data(simulatedExtrpi0)
print(simulatedExtrpi0)
## Not run:
plot(simulatedExtrpi0)

## End(Not run)
```

simulatedSubex	<i>A Simulated 'subex' Object</i>
----------------	-----------------------------------

Description

This is the result from calling `subex` on `simulatedDat`.

Usage

```
data(simulatedSubex)
```

Format

An object of class `subex`

Details

This is the result from calling
`data(simulatedDat)`
`simulatedSubex=subex(simulatedDat,balanced=FALSE,max.reps=Inf,plotit=FALSE)`

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

`subex`

Examples

```
data(simulatedSubex)
print(simulatedSubex)
## Not run:
plot(simulatedSubex)

## End(Not run)
```

simulatedSubt	<i>A Simulated 'subt' Object</i>
---------------	----------------------------------

Description

This is the result from calling `subt` on `simulatedDat`

Usage

```
data(simulatedSubt)
```

Format

an object of class `subt` with `balanced=FALSE` with `max.reps=Inf` attributes.

Details

```
This is the result from calling
data(simulatedDat)
simulatedSubt=subt(simulatedDat,balanced=FALSE,max.reps=Inf)
```

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

`subt`

Examples

```
data(simulatedSubt)
print(simulatedSubt)
## Not run:
plot(simulatedSubt)

## End(Not run)
```

simulatedTstat	<i>t-Statistics and p-Values from a Simulated Microarray Data Set</i>
----------------	---

Description

These are the results from applying t-tests to the simulated data set [simulatedDat](#).

Usage

```
data(simulatedTstat)
data(simulatedPval)
```

Format

A numeric vector of two-sample t-statistics of length 5000.

Details

This is the result from calling

```
data(simulatedDat) simulatedTstat=matrix.t.test(simulatedDat,tOnly=TRUE) simulatedPval=matrix.t.test
```

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[sim.dat](#), [simulatedDat](#)

Examples

```
data(simulatedTstat)
data(simulatedPval)
```

sparncpt	<i>Semiparametric density estimation for noncentrality parameters</i>
----------	---

Description

Semiparametric density estimation for noncentrality parameters using the combination method of Olkin and Spiegelman (1987), based on fits from both [parncpt](#) and [nparncpt](#).

Usage

```

sparncpt(obj1, obj2, ...)
## S3 method for class 'parncpt'
sparncpt(obj1, obj2, ...)
## S3 method for class 'nparncpt'
sparncpt(obj1, obj2, ...)
## S3 method for class 'numeric'
sparncpt(obj1, obj2, ...)

```

Arguments

obj1, obj2	Case 1: obj1 and obj2 are of class parncpt and nparncpt respectively; or vice versa; Case 2: obj1 is a numeric vector of t-statistics and obj2 is a vector degrees of freedom
...	other arguments passed to dtn.mix , most notably the approximation argument.

Details

This is a two-component mixture of a parametric fit from [parncpt](#) and a nonparametric fit from [nparncpt](#), with mixing proportion rho. If obj1 and obj2 are t-statistics and degrees of freedom respectively, calls to each of [parncpt](#) and [nparncpt](#) are made and their results are used in combination.

Value

a list with class `c('sparncpt', 'npest')`:

pi0	estimated proportion of true nulls
mu.ncp	mean of ncp
sd.ncp	SD of ncp
logLik	an object of class <code>logLik</code> . The associated df is the estimated effective number of parameters (enp). The log likelihood is also penalized likelihood. See also logLik.ncpest and AIC .
enp	estimated ENP
par	estimated mixing proportion rho
gradient	analytic gradient at the estimate (not implemented)
hessian	analytic hessian at the estimate (not implemented)
parfit	the fitted parncpt object
nparfit	the fitted nparncpt object

Author(s)

Long Qu

References

I. Olkin and C. H. Spiegelman. (1987) A Semiparametric Approach to Density Estimation. *Journal of the American Statistical Association*. 82,399,858–865

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [nparncpt](#), [fitted.sparncpt](#), [plot.sparncpt](#), [summary.sparncpt](#), [coef.ncpest](#), [logLik.ncpest](#), [vcov.ncpest](#), [AIC](#), [dncp](#)

Examples

```
## Not run:
data(simulatedTstat)
(npfit=nparncpt(tstat=simulatedTstat, df=8));
(pfit=parncpt(tstat=simulatedTstat, df=8, zeromean=FALSE)); plot(pfit)
(pfit0=parncpt(tstat=simulatedTstat, df=8, zeromean=TRUE)); plot(pfit0)
(spfit=sparncpt(npfit,pfit)); plot(spfit)

## End(Not run)
```

subex

Subsampling-Extrapolation Based Estimation of Proportion of True Null Hypotheses and False Discovery Rates for Microarray Data

Description

This function is a wrapper of [subt](#), [extrp.pi0](#) and [fdr](#), and is a ready to use directly on a matrix of microarray data.

Usage

```
subex(dat, n1 = round(ncol(dat)/2), n2 = ncol(dat) - n1,
      flmethod = c("lastbin", "qvalue"),
      max.reps = 20, balanced = FALSE, nparm = c(2, 4),
      extrpFUN = c("constrOptim", "genoud"),
      starts = c(pi0 = 0.75, gam2 = 1, a = 0.5, c = 0.5), plotit = TRUE)
```

Arguments

dat	a numeric matrix, which is the microarray data. Each row represent a gene, and each column represent a subject. The first n1 columns correspond to the first treatment group; and the rest n2 columns correspond of the second treatment group.
n1	a positive integer, the sample size in treatment group 1.

n2	a positive integer, the sample size in treatment group 2.
f1method	character, the name of the function used to estimate the p-value density at one. See subt for details.
max.reps	a positive integer, the maximum number of subsamples "per subsample size configuration". See subt for details.
balanced	logical, indicating if only balanced subsamples are generated. See subt for details.
nparm	either 2 or 4, indicating the number of parameters used in extrapolation. See extrp.pi0 for details.
extrpFUN	character, specifying the name of the optimization function for nonlinear regression. See the FUN argument of extrp.pi0 for details.
starts	a numeric vector of length nparm, specifying the starting values of optimization. See extrp.pi0 for details.
plotit	logical, indicating if the extrapolation plot will be produced. See extrp.pi0 for details.

Details

This function calls [subt](#), [extrp.pi0](#), [matrix.t.test](#) and [fdr](#) sequentially to estimate the proportion of true null hypotheses π_0 as well as the false discovery rates (FDR) based on the estimated π_0 .

Value

an object of class `subex`, which is a list 4 components:

pi0	a numeric value, giving the estimated π_0
.	.
extrp.fit	an object of class <code>extrpi0</code> , the results from calling extrp.pi0 .
pvalues	a numeric vector of length the same as <code>nrow(dat)</code> , the p-values for each gene.
qvalues	a numeric vector of length the same as <code>nrow(dat)</code> , the q-values for each gene.

Note

Plotting using package **rgl** will be tried. If not available, a warning will be generated. See [plot.extrpi0](#) for details.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[subt](#), [extrp.pi0](#), [matrix.t.test](#), [fdr](#), [plot.subex](#), [print.subex](#)

Examples

```
## Not run:
set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
## this is how the 'simulatedSubex' object in this package generated
simulatedSubex=subex(simulatedDat,balanced=FALSE,max.reps=Inf,plotit=FALSE)
plot(simulatedSubex)

## End(Not run)
data(simulatedSubex)
print(simulatedSubex)
```

subt	<i>Subsampling a Microarray Data Set for Estimating Proportion of True Null Hypotheses</i>
------	--

Description

This function subsamples the columns (arrays) of a microarray data set and do two-sample t-tests. Subsamples from each treatment group are obtained and combined. A t-test is conducted for each row (gene) of the subsampled data set and the p-value density at one is estimated for each combined subsample.

Usage

```
subt(dat, n1 = round(ncol(dat)/2), n2 = ncol(dat) - n1,
      f1method = c("lastbin", "qvalue"),
      max.reps = if(balanced)20 else 5, balanced = FALSE, ...)
```

Arguments

dat	a numeric matrix, the microarray data set with each row being a gene, and each column being a subject. The first n1 columns correspond to treatment group 1 and the rest n2 columns correspond to treatment group 2.
n1	a positive integer, the original sample size in treatment group 1.
n2	a positive integer, the original sample size in treatment group 2.
f1method	character, the name of the function to be used to estimate the p-value density at 1. The first argument of the function needs to be a vector of values.
max.reps	a positive integer, the maximum number of subsamples to obtain per subsample size configuration. If this is set to Inf, then all possible subsamples will be tried. However, see Notes and the R argument of combn2R .

balanced logical, indicating whether only balanced subsamples are obtained. This is computationally faster and is good for initial exploration purposes.

... additional arguments used by `f1method`.

Details

This function tries to get possible subsamples through `combn2R`.

For each total subsample size $M=3,4,\dots,N$, where $N=n_1+n_2$, do the following,

- 1 For each treatment 1 subsample size $m_1=1,2,\dots,n_1$, let $m_2=M-m_1$. If $1\leq m_2\leq n_2$ and at least one of `balanced` and `m1=m2` is true, then do the following,
 - 1.1 Randomly choose `max.reps` subsamples among all possible subsamples by choosing m_1 subjects from treatment group 1 and m_2 subjects from treatment group 2, by using the function `combn2R` with `sample.method="diff2"` and `try.rest=TURE`. Note that this may *not* be always possible due to some practical computational limitations. See `combn2R` for details.
 - 1.2 For each subsample obtained in 1.1, (1) do a t-test for each gene (i.e., each row of the subsample), and (2) estimate the p-value density at one.

Value

an object of class `c("subt", "matrix")`, which is a G -by-3 numeric matrix, where G is `nrow{dat}`, with column names `'f1'`, `'n1'`, and `'n2'`, corresponding to the p-value density at 1 and subsample size in each treatment group. This object also has the following `attributes`,

`n1` the same as the argument `n1`.

`n2` the same as the argument `n2`.

`f1method` the same as the argument `f1method`.

`max.reps` the same as the argument `max.reps`.

`balanced` the same as the argument `balanced`.

Note

`max.reps` applies to each subsample size configuration. For example, 2 subjects subsampled from treatment group 1 and 3 subjects subsampled from treatment group 2 will be considered as a different subsample size configuration than 3 subjects subsampled from treatment group 1 and 2 subjects subsampled from treatment group 2. For the small sample sizes commonly seen in microarray data, a large `max.reps` is rarely a big computational burden. But be careful when you do have a very large sample size, as the number of all possible subsamples grows very fast.

Author(s)

Long Qu

References

Qu, L., Nettleton, D., Dekkers, J.C.M. Subsampling Based Bias Reduction in Estimating the Proportion of Differentially Expressed Genes from Microarray Data. Unpublished manuscript.

See Also

[print.subt](#), [plot.subt](#), [extrp.pi0](#), [matrix.t.test](#), [combn2R](#), [subex](#), [lastbin](#), [qvalue](#)

Examples

```
## Not run:
set.seed(9992722)
## this is how the 'simulatedDat' data set in this package generated
simulatedDat=sim.dat(G=5000)
## this is how the 'simulatedSubt' object in this package generated
simulatedSubt=subt(simulatedDat,balanced=FALSE,max.reps=Inf)

## End(Not run)
data(simulatedSubt)
print(simulatedSubt)
```

summary.nparncpt	<i>Print summary for an object of class nparncpt</i>
------------------	--

Description

Print summary for an object of class nparncpt. This includes the estimated proportion of true null hypotheses (π_0), estimated mean of noncentrality parameters ($\mu.ncp$), estimated standard deviation of noncentrality parameters ($sd.ncp$), the estimated effective number of parameters (enp), and the tuning parameter (λ) that controls the amount of smoothing.

Usage

```
## S3 method for class 'nparncpt'
summary(object, ...)
## S3 method for class 'nparncpt'
print(x, ...)
```

Arguments

object, x	an object of class nparncpt
...	currently not used

Value

the [invisible](#) object itself

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#)

summary.parncpt	<i>Print summary for an object of class parncp</i>
-----------------	--

Description

Print summary for an object of class parncpt. This includes the estimated proportion of true null hypotheses (π_0), estimated mean of noncentrality parameters ($\mu.ncp$), estimated standard deviation of noncentrality parameters ($sd.ncp$).

Usage

```
## S3 method for class 'parncpt'
summary(object,...)
## S3 method for class 'parncpt'
print(x,...)
```

Arguments

object,x	an object of class parncpt
...	currently not used

Value

the [invisible](#) object itself

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#)

summary.sparncpt	<i>Print summary for an object of class sparncp</i>
------------------	---

Description

Print summary for an object of class sparncpt. This includes the estimated proportion of true null hypotheses (π_0), estimated mean of noncentrality parameters ($\mu.ncp$), estimated standard deviation of noncentrality parameters ($sd.ncp$), and effective number of parameters (enp).

Usage

```
## S3 method for class 'sparncpt'  
summary(object, ...)  
## S3 method for class 'sparncpt'  
print(x, ...)
```

Arguments

object, x	an object of class sparncpt
...	currently not used

Value

the [invisible](#) object itself

Author(s)

Long Qu

References

Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of t -statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. *Biometrics* (in press).

See Also

[parncpt](#), [nparncpt](#), [sparncpt](#)

varB	<i>Variance of the reflected Bsplines</i>
------	---

Description

2nd order raw moment of the Bsplines

Usage

```
varB(m,deltamax,K)
```

Arguments

m	The mth spline function to which variance is to be computed
deltamax	maximum noncentrality parameter
K	Number of splines

Value

Numeric scalar

Author(s)

Long Qu

See Also

[NBsplines](#)

vcov.ncpest	<i>extract inverse Hessian matrix from napest class</i>
-------------	---

Description

return inverse Hessian matrix from napest class

Usage

```
## S3 method for class 'napest'
vcov(object,...)
```

Arguments

object	an object of class napest
...	currently not used

Value

a numeric inverse Hessian matrix

Author(s)

Long Qu

See Also

[parncpt](#), [nparncpt](#)

znormix

Normal-mixture based estimation of LFDR and π_0

Description

This function implements the method of McLachLan, Bean and Jones (2006).

Usage

```
znormix(p, theoretical.null=TRUE, start.pi0, eps=1e-5, niter=Inf, verbose=FALSE)
```

Arguments

p	a numeric vector the p-values
theoretical.null	logical scalar, indicating whether theoretical $N(0,1)$ null distribution is assumed for z-scores.
start.pi0	optional numeric scalar, starting value of π_0 for EM algorithm; if missing, qvalue will be called with default arguments to get this starting value.
eps	numeric scalar, maximum tolerable absolute difference of parameter estimates for successive iterations in the EM algorithm.
niter	numeric scalar, maximum number of EM iterations.
verbose	logical scalar, indicating whether excessive outputs will be printed during EM algorithm.

Details

A two-component normal mixture model is fit thru EM algorithm on the z-scores, where $z = qnorm(1-p)$.

Value

A length 5 numeric named vector of estimated parameters, with class 'znormix' and attributes

`theoretical.null`

the same as input.

`converged`

logical, convergence status.

`iter`

numeric, number of iterations.

`call`

the `match.call()` result.

`lfdr`

numeric vector of local false discovery rates, with order being the same as the input p-values.

`fdr`

numeric vector of false discovery rates, with order being the same as the input p-values.

Note

There are two small differences with McLachlan, Bean and Jones (2006):

- If `start.pi0` is missing, it is estimated by the q-value smoother method implimented in [qvalue](#).
- For the empirical null case, a call to `quantile(z, start.pi0)` is used as the threshold to determine the initial component assignment, when choosing starting values.

Author(s)

Long Qu

References

G.J. McLachlan, R.W. Bean and L. Ben-Tovim Jones. (2006) A Simple implementation of a normal mixture approach to differential gene expression in multiclass microarrays. *Bioinformatics*, 22(13):1608-1615.

See Also

[qvalue](#), [histf1](#)

Examples

```
set.seed(9992722)
p=1-pnorm(c(rnorm(7000),rnorm(3000,1)))
znormix(p, eps=1e-5)['pi0'] ## 0.6759555
```

Index

- *Topic **aplot**
 - plot.extrpi0, 46
 - plot.subex, 51
 - plot.subt, 52
- *Topic **datagen**
 - sim.dat, 58
- *Topic **datasets**
 - simulatedDat, 59
 - simulatedExtrpi0, 60
 - simulatedSubex, 61
 - simulatedSubt, 62
 - simulatedTstat, 63
- *Topic **distribution**
 - cond.cdf, 11
 - dncp, 15
 - dt.int2, 16
 - dt.lap, 17
 - dt.sad and pt.sad, 18
 - dtm.mix, 19
 - marginal.dt, 33
 - mTruncNorm, 36
 - pdf.dist, 45
 - pi0-package, 3
 - reflect and fold, 57
- *Topic **dynamic**
 - plot.extrpi0, 46
 - plot.subex, 51
 - plot.subt, 52
- *Topic **hplot**
 - plot.extrpi0, 46
 - plot.nparncpt, 47
 - plot.parcncpt, 48
 - plot.sparncpt, 50
 - plot.subex, 51
 - plot.subt, 52
- *Topic **htest**
 - CBUM, 6
 - coef.ncpest, 8
 - convest, 12
 - extrp.pi0, 20
 - fdr, 23
 - lastbin, 30
 - logLik.ncpest, 33
 - matrix.t.test, 34
 - pi0-package, 3
 - subex, 65
 - subt, 67
 - vcov.ncpest, 72
 - znormix, 73
- *Topic **iplot**
 - plot.extrpi0, 46
 - plot.subex, 51
 - plot.subt, 52
- *Topic **iteration**
 - combn2R, 9
 - subex, 65
 - subt, 67
- *Topic **methods**
 - lfdr and ppee, 31
 - plot.extrpi0, 46
 - plot.nparncpt, 47
 - plot.parcncpt, 48
 - plot.sparncpt, 50
 - plot.subex, 51
 - plot.subt, 52
 - print.extrpi0, 53
 - print.subex, 54
 - print.subt, 55
 - summary.nparncpt, 69
 - summary.parcncpt, 70
 - summary.sparncpt, 71
- *Topic **models**
 - discTMix, 13
 - nparncpt, 40
 - parncpt, 42
 - pi0-package, 3
 - sparncpt, 63
- *Topic **multivariate**

- CBUM, 6
- extrp.pi0, 20
- fdr, 23
- lastbin, 30
- pi0-package, 3
- sim.dat, 58
- subex, 65
- subt, 67
- znormix, 73
- *Topic **nonlinear**
 - extrp.pi0, 20
 - pi0-package, 3
 - subex, 65
- *Topic **nonparametric**
 - agjack.pi0, 5
 - dncp, 15
 - extrp.pi0, 20
 - fdr, 23
 - pi0-package, 3
 - subex, 65
 - subt, 67
- *Topic **optimize**
 - extrp.pi0, 20
 - grid.search, 27
 - nparncpt, 40
 - parncpt, 42
 - pi0-package, 3
 - subex, 65
- *Topic **package**
 - pi0-package, 3
- *Topic **print**
 - print.extrpi0, 53
 - print.subex, 54
 - print.subt, 55
- *Topic **regression**
 - extrp.pi0, 20
 - subex, 65
- *Topic **smooth**
 - fitted, 24
 - fitted.discTMix, 25
 - nparncpp, 37
 - nparncpp.iter, 39
 - nparncpt, 40
 - pi0-package, 3
 - sparncpt, 63
 - varB, 72
- *Topic **univar**
 - discTMix, 13
 - geomean and harmean, 26
 - gjack, 27
 - histf1, 28
 - marginal.dt, 33
 - mTruncNorm, 36
 - pavaf1, 44
 - pdf.dist, 45
 - reflect and fold, 57
- *Topic **utilities**
 - clean.cdf, save.cdf and load.cdf, 7
 - combn2R, 9
 - loadOrInstall, 32
- 08.Tests, 13
 - agjack.pi0 (agjack.pi0), 5
 - clean.cdf (clean.cdf, save.cdf and load.cdf), 7
 - cond.cdf (cond.cdf), 11
 - geomean (geomean and harmean), 26
 - gjack (gjack), 27
 - harmean (geomean and harmean), 26
 - histf1 (histf1), 28
 - load.cdf (clean.cdf, save.cdf and load.cdf), 7
 - loadOrInstall (loadOrInstall), 32
 - nparncpp (nparncpp), 37
 - nparncpp.iter (nparncpp.iter), 39
 - pavaf1 (pavaf1), 44
 - save.cdf (clean.cdf, save.cdf and load.cdf), 7
 - varB (varB), 72
- agjack.pi0, 5
- AIC, 33, 41–44, 64, 65
- apply, 35, 36
- array, 10
- attributes, 68
- CBUM, 4, 6, 32
- clean.cdf, save.cdf and load.cdf, 7
- coef.ncpest, 8, 42, 44, 65
- combn, 9–11
- combn2R, 4, 9, 67–69
- cond.cdf, 8, 11, 38
- constrOptim, 22
- convest, 12, 13, 32
- discTMix, 13, 26, 32
- dncp, 15, 38, 42, 44, 65

- dt, [17–19](#), [37](#), [43](#)
- dt.int2, [16](#), [18–20](#), [37](#)
- dt.lap, [17](#), [17](#), [19](#), [20](#)
- dt.sad, [17–20](#)
- dt.sad (dt.sad and pt.sad), [18](#)
- dt.sad and pt.sad, [18](#)
- dtm.mix, [17](#), [19](#), [34](#), [41](#), [64](#)
- EOC, [15](#)
- extrp.pi0, [3](#), [4](#), [20](#), [47](#), [53–55](#), [60](#), [65–67](#), [69](#)
- fdr, [3](#), [4](#), [23](#), [32](#), [55](#), [65–67](#)
- fitted, [24](#)
- fitted.discTMix, [15](#), [25](#)
- fitted.nparncpt, [42](#)
- fitted.parcncpt, [44](#)
- fitted.sparncpt, [65](#)
- fold (reflect and fold), [57](#)
- geomean and harmean, [26](#)
- gjack, [5](#), [27](#)
- grid.search, [27](#)
- histf1, [7](#), [28](#), [30](#), [74](#)
- install.packages, [32](#)
- invisible, [49](#), [50](#), [69–71](#)
- lastbin, [4](#), [29](#), [30](#), [69](#)
- lfdr (lfdr and ppee), [31](#)
- lfdr and ppee, [31](#)
- library, [32](#)
- list, [10](#)
- loadOrInstall, [32](#)
- logLik, [33](#)
- logLik.ncpest, [33](#), [41–44](#), [64](#), [65](#)
- LowRankQP, [38](#)
- lsei, [22](#), [38](#)
- marginal.dt, [33](#)
- matrix.t.test, [4](#), [34](#), [66](#), [67](#), [69](#)
- mt.rawp2adjp, [24](#)
- mTruncNorm, [17](#), [36](#)
- NBsplines, [72](#)
- nparncpp, [4](#), [8](#), [16](#), [25](#), [26](#), [32](#), [37](#), [39](#)
- nparncpp.iter, [39](#)
- nparncpt, [4](#), [8](#), [16](#), [25](#), [26](#), [32–34](#), [38](#), [40](#), [44](#), [48–50](#), [63–65](#), [70](#), [71](#), [73](#)
- optim, [15](#), [28](#), [43](#)
- optimize, [22](#)
- parncpt, [4](#), [8](#), [16](#), [25](#), [26](#), [32–34](#), [38](#), [42](#), [42](#), [48–50](#), [63–65](#), [70](#), [71](#), [73](#)
- pavaf1, [44](#)
- pdf.dist, [45](#)
- persp, [46](#), [51](#)
- persp3d, [46](#), [51](#)
- pi0 (pi0-package), [3](#)
- pi0-package, [3](#)
- plot.extrpi0, [22](#), [46](#), [51](#), [52](#), [66](#)
- plot.nparncpt, [40](#), [42](#), [47](#)
- plot.parcncpt, [44](#), [48](#)
- plot.sparncpt, [50](#), [65](#)
- plot.subex, [51](#), [67](#)
- plot.subt, [52](#), [69](#)
- plot3d, [52](#)
- ppee (lfdr and ppee), [31](#)
- print.extrpi0, [22](#), [53](#), [55](#)
- print.nparncpt (summary.nparncpt), [69](#)
- print.parcncpt (summary.parcncpt), [70](#)
- print.sparncpt (summary.sparncpt), [71](#)
- print.subex, [54](#), [67](#)
- print.subt, [55](#), [69](#)
- pt, [12](#), [37](#)
- pt.sad (dt.sad and pt.sad), [18](#)
- qvalue, [7](#), [23](#), [24](#), [29](#), [30](#), [69](#), [73](#), [74](#)
- reflect (reflect and fold), [57](#)
- reflect and fold, [57](#)
- require, [32](#)
- scatterplot3d, [52](#)
- sim.dat, [58](#), [59](#), [63](#)
- simulatedDat, [59](#), [61–63](#)
- simulatedExtrpi0, [60](#)
- simulatedPval (simulatedTstat), [63](#)
- simulatedSubex, [61](#)
- simulatedSubt, [60](#), [62](#)
- simulatedTstat, [63](#)
- sparncpt, [4](#), [8](#), [16](#), [25](#), [26](#), [32–34](#), [38](#), [42](#), [44](#), [48–50](#), [63](#), [70](#), [71](#)
- subex, [3](#), [4](#), [22](#), [51](#), [52](#), [55](#), [61](#), [65](#), [69](#)
- subt, [3–5](#), [21](#), [22](#), [53–56](#), [62](#), [65–67](#), [67](#)
- summary.extrpi0 (print.extrpi0), [53](#)
- summary.nparncpt, [42](#), [69](#)
- summary.parcncpt, [44](#), [70](#)
- summary.sparncpt, [65](#), [71](#)

summary.subex (print.subex), [54](#)
summary.subt (print.subt), [55](#)

t.test, [11](#), [36](#), [38](#)
tMixture, [13](#), [15](#)
tstatistics, [15](#)

varB, [72](#)
vcov.ncest, [42](#), [44](#), [65](#), [72](#)

znormix, [4](#), [32](#), [73](#)