Package ‘pi0’

July 9, 2017

Type Package
Title Estimating the Proportion of True Null Hypotheses for FDR
Version 1.4-1
Date 2017-07-09
Depends R (>= 3.0.0)
Imports Matrix (>= 1.0-0), numDeriv, limSolve (>= 1.5.2),
scatterplot3d, qvalue, Iso(>= 0.0-5), quadprog (>= 1.5-3),
kernlab, LowRankQP, graphics, grDevices, methods, stats
Suggests rgl, limma, OCplus, multtest
Description Methods for estimating the proportion of true null hypotheses, i.e., the pi0, 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 non-centrality parameters from a large number of parametric tests.
License GPL (>= 2)
URL https://github.com/gitlongor/pi0
BugReports https://github.com/gitlongor/pi0/issues
NeedsCompilation yes
Author Long Qu [aut, cre, ctb],
Kun Liang [ctb],
Yudi Pawitan [aut] (The author(s) of OCplus::tMixture, which was modified slightly leading to discTMix in this package.),
Alexander Ploner [aut] (The author(s) of OCplus::tMixture, which was modified slightly leading to discTMix in this package.),
Egil Ferkingstad [aut] (The author(s) of limma::convest, which was modified slightly leading to convest in this package.),
Mette Langaas [aut] (The author(s) of limma::convest, which was modified slightly leading to convest in this package.),
Marcus Davy [aut] (The author(s) of limma::convest, which was modified slightly leading to convest in this package.)
Maintainer Long Qu <rtistician@gmail.com>
Repository  CRAN
Date/Publication  2017-07-09 06:14:22 UTC

R topics documented:

pi0-package .................................................. 3
agejack.pi0 ................................................. 5
CBUM ............................................................... 6
clean.cdf, save.cdf and load.cdf .......................... 7
draini.pi0 ..................................................... 8
combn2R .......................................................... 9
cond.cdf .......................................................... 11
convex ............................................................ 12
discTMix .......................................................... 14
dncp ............................................................... 16
dt.int2 ........................................................... 17
dt.lap .............................................................. 18
dt.sad and pt.sad .............................................. 19
dtn.mix ........................................................... 20
textr.pi0 ......................................................... 21
fdr ............................................................... 23
fitted ............................................................. 25
fitted.dicTMix .................................................. 26
geomean and harmean ........................................ 27
gjack ............................................................. 27
grid.search ...................................................... 28
histf1 ............................................................. 29
lastbin ............................................................ 30
lfdr and ppee ................................................... 31
logLik.ncpest .................................................. 32
marginal.dt ..................................................... 33
matrix.t.test ................................................... 34
mTruncNorm ..................................................... 35
nparncpp ......................................................... 37
nparncpp.iter .................................................. 38
nparncpt ........................................................ 39
parncpt .......................................................... 42
pavaf1 ............................................................ 45
pdf.dist ........................................................ 45
plot.extrpi0 .................................................... 46
plot.nparncpt ................................................... 47
plot.parcpt ..................................................... 48
plot.parcpt ..................................................... 48
plot.spncpt ..................................................... 50
plot.subex ...................................................... 51
plot.subt ........................................................ 52
print.extrpi0 .................................................. 53
print.subex ..................................................... 54
Estimating the proportion of true null hypotheses and False Discovery Rates

Description

This package implements method(s) to (approximately unbiasedly) estimate the proportion of true null hypotheses, i.e., the pi0, 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

- **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.

<table>
<thead>
<tr>
<th>pi0-package</th>
<th>Estimating the proportion of true null hypotheses and False Discovery Rates</th>
</tr>
</thead>
</table>

Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>print.subt</td>
<td>55</td>
</tr>
<tr>
<td>reflect and fold</td>
<td>57</td>
</tr>
<tr>
<td>sim.dat</td>
<td>58</td>
</tr>
<tr>
<td>simulatedDat</td>
<td>59</td>
</tr>
<tr>
<td>simulatedExtrpi0</td>
<td>60</td>
</tr>
<tr>
<td>simulatedSubex</td>
<td>61</td>
</tr>
<tr>
<td>simulatedSubt</td>
<td>62</td>
</tr>
<tr>
<td>simulatedTstat</td>
<td>63</td>
</tr>
<tr>
<td>sparncpt</td>
<td>63</td>
</tr>
<tr>
<td>subt</td>
<td>65</td>
</tr>
<tr>
<td>subex</td>
<td>65</td>
</tr>
<tr>
<td>summary.nparncpt</td>
<td>69</td>
</tr>
<tr>
<td>summary.parncpt</td>
<td>70</td>
</tr>
<tr>
<td>summary.sparncpt</td>
<td>71</td>
</tr>
<tr>
<td>varB</td>
<td>72</td>
</tr>
<tr>
<td>vcov.ncpest</td>
<td>72</td>
</tr>
<tr>
<td>znormix</td>
<td>73</td>
</tr>
</tbody>
</table>

License: GPL version 2 or newer

Package: pi0
Type: Package
Version: 1.3-354
Date: 2014-08-22
• **subex** A wrapper that automates `subt`, `extrp.pi0`, and `fdr`.
• **combn2R** Generating a sample of combinations by choosing \( m_1 \) out of \( n_1 \) and \( m_2 \) out of \( n_2 \) 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 <long.qu@wright.edu>

**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, 68, 1178–1187.


**See Also**

`subex, subt, extrp.pi0, fdr, combn2R, nparncpt, parncpt, sparncpt, nparncpp`

**Examples**

```r
## 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)
```
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

- **p**: a numeric vector the p-values
- **start.pi0**: numeric scalar, starting value of pi0 for EM algorithm.
- **thresh.censor**: numeric scalar, the threshold of censoring. If isTRUE(lambda<min(p)), this is equivalent to the BUM method of Pounds and Morris (2003).
- **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

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 $\pi_0$ of true null hypotheses. The result has a class 'CBUM', with the following attributes:

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

Author(s)

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

References


See Also

`qvalue`, `histfQ`

Examples

```r
set.seed(99722)
p = c(runif(3500), rbeta(1500, .8, 2.2))
CBUM(p)
```

---

clean.cdf, save.cdf and load.cdf

**Description**

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

**Usage**

```r
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
Characer: 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


See Also

cdf.cdf

---

calc.ncest Extract estimated parameters

Description

Extract estimated parameters from object of ncpest class

Usage

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

Arguments

object
an object of ncpest 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 n1 and n2 elements, respectively. Each time, m1 elements will be randomly chosen from group 1; and m2 elements will be randomly chosen from group 2. These m1+m2 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

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

try.rest logical, together with sample.method, specifying how samples should be taken, if not all combinations are generated; see Details.

... optionally, further arguments to FUN.

Details

This function enhances \texttt{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 \( x \) nor \( m \) is missing and neither \( m=0 \) nor \( m=0 \), the function works in two-group mode. In this situation,

(A) if \( \text{sample.method} = \text{"diff2"} \), \texttt{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 \( \text{try.rest} = \text{FALSE} \), 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 \( \text{try.rest} = \text{TRUE} \), then \( \text{sample.method} \) will be reset to "all" and the function will try to get \( r \) samples from all \( \text{choose}(n_1,m_1)\times\text{choose}(n_2,m_2) \) combinations (see below).

(B) if \( \text{sample.method} = \text{"all"} \), \texttt{combn2R} will try to sample \( R \) samples from all \( \text{choose}(n_1,m_1)\times\text{choose}(n_2,m_2) \) 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 \( \text{sample.method} = \text{"diff2"} \). However, this will guarantee any two samples of size \( m+m_2 \) will differ in at least one element.

(C) if \( \text{sample.method} = \text{"replace"} \), \texttt{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 \( \text{try.rest} = \text{TRUE} \), then \( \text{sample.method} \) 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 \( n_2 \) is zero, the function works in one-group mode. In this situation, \( \text{sample.method} = \text{"diff2"} \) and \( \text{sample.method} = \text{"all"} \) will be treated the same as \( \text{sample.method} = \text{"noReplace"} \), and \texttt{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, \( \text{sample.method} \) will be reset to "replace" and no guarantee will be made to ensure the \( R \) combinations are different from each other.
cond.cdf

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

```r
combn2R(4, 2) ### the same as combin(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 combinR(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.
### no common samples, even considering only one group
combn2R(1:4, 2, 2, 5, 1, R=5, sample.method="diff2")
combn2R(1:3, 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

```r
cond.cdf(p.eval, ncp, test=c("t","z"), alternative=c("two.sided","less", "greater"), df=if(test="z")Inf else df, keep.cdf=NULL,
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**: Either NULL or an environment. If this is non-null, the computed CDF will also be stored in `keep.cdf` environment to allow later use. As of version 1.4-0, `keep.cdf=TRUE`, is no longer supported to comply with new CRAN policies; `keep.cdf=FALSE` will be treated the same as `keep.cdf=NULL`.
- **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


**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 \( \pi_0 \).

Value

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

Author(s)

Long Qu slightly modified and \code{convest} function by Egil Ferkingstad and Mette Langaas in \pkg{limma} package.

References


See Also

See \code{08.Tests} for other functions for producing or interpreting p-values.

Examples

```r
set.seed(9992722)
pvals = runif(5e3)^1.5
convest(pvals, niter=50)[1]
```
**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 slightly modified code of `tmixture` in OCplus package.

**Usage**

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

**Arguments**

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

- \( p0\_est \): the estimated proportion of non-differentially expressed genes, after collapsing components with estimated non-centrality sizes below \texttt{threshold.delta}.
- \( p0\_raw \) and \( p1\_raw \): the estimated proportion before collapsing the components.
- \( p1 \): the estimated proportions of differentially expressed genes corresponding to the effect sizes, relating to \( p0\_raw \).
- \( D \): effect sizes of the differentially expressed genes in multiples of the gene-by-gene standard deviation.
- \( \delta \): effect sizes of the differentially expressed genes expressed as the noncentrality parameter of the corresponding noncentral t-distribution.
- \( \text{AIC} \): the AIC value for the maximum likelihood fit.
- \( \text{opt} \): The output from \texttt{optim}, giving details about the optimization process.
- \( \text{data} \): A list of \texttt{tstat} and \texttt{df}.

Author(s)

Long Qu slightly modified the \texttt{tmixture} by Y. Pawitan and A. Ploner in \texttt{OCplus} package.

References

Density of noncentrality parameters

dncp

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 'parncpt2'
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 ncpest, 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


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. 68. 1178-1187.

See Also

parncpt, parncpt2, nparncpt, sparcncpt, nparncpp

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
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 integrates to 1 (not implemented yet). If none, no normalization is performed.
- ... currently not used.

Value
numeric vector of densities

Author(s)
Long Qu

References

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


See Also

dt.int2, dt.lap, dt
**dtm.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

```r
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. `int2` computes exact density for integer `df` and polynomially interpolate to non-integer degrees of freedom. `saddlepoint` computes the saddle point approximation of the noncentral t-density. `laplace` computes the laplacian approximation of the noncentral t-density. `none` uses the (sort of) true noncentral t-density `dt` function. However, if all degrees of freedom are integers, `int2` will be used even if `none` is specified, both of which being exact.
- `...`: 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


Qu L, Nettleton D, Dekkers JCM. (2012) Improved Estimation of the Noncentrality Parameter Distribution from a Large Number of St$\$-statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. Biometrics. 68. 1178-1187.

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., \( \pi_0 \), 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 \( \pi_0 \).

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 \( \pi_0 \) 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 \( \gamma^2 \). This regression function has two parameters,

\[
f_1 = (1 - \pi_0) \sqrt{\frac{n_1 + n_2}{n_1 + n_2 + n_1n_2\gamma^2}} + \pi_0
\]

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:

\[
f_1 = a \left( \frac{n_1 + n_2}{n_1 + n_2 + n_1n_2\gamma^2} \right)^c + \pi_0
\]

subject to additional constraints of \( a > 0 \) and \( c > 0 \).

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

Value

an object of class \texttt{extrpi0}, which is a numeric vector of length 1, named "\( \pi_0 \)". giving the estimated \( \pi_0 \), with the following attributes:

\begin{itemize}
  \item \texttt{attr('fitted.obj')}  a list, which is the object returned by \texttt{FUN}.
  \item \texttt{attr('nparm')}  the same as the first element of \texttt{nparm}.
  \item \texttt{attr('extrpFUN')}  the same as \texttt{FUN}.
  \item \texttt{attr('start.val')}  the first \texttt{nparm} elements of \texttt{starts}.
  \item \texttt{attr('subt.data')}  the same as \texttt{dat}.
\end{itemize}
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


See Also

`subt`, `subex`, `constrOptim`, `optimize`, `lsei`, `print.extrpi0`, `plot.extrpi0`.

Examples

```r
## 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

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

- **p**: a numeric vector of p-values
- **pi0**: numeric, a given estimate of the proportion of true null hypotheses, π₀, truncated to [0,1]. The default is the conservative Benjamini and Hochberg (1995) version.

Details

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

\[
q_i = \min_{i \leq j \leq G} G \cdot \pi_0 \cdot p(j)/j
\]

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


See Also

qvalue, mt.rawp2adjp

Examples

```r
set.seed(9992722)
pvals=runif(5e4)*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
```
Density evaluated at observed statistics for ncpest class

Description

Density evaluated at observed statistics for ncpest class

Usage

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

## S3 method for class 'parncpt'
fitted(object, ...)
## S3 method for class 'parncpt2'
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. 68. 1178-1187.
fitted.discTMix

See Also

sparncpt, parncpt, parncpt2, nparncpt, nparncpp

fitted.discTMix  

*Density evaluated at observed statistics for discTMix class*

Description

Density evaluated at observed statistics for discTMix class

Usage

```r
## 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


See Also

sparncpt, parncpt, nparncpt, nparncpp, discTMix
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

Generalized jackknife

Description

Generalized jackknife bias correction

Usage

gjack(theta1, theta2, R)

Arguments

theta1 Numeric
theta2 Numeric
R Numeric
grid.search

Details
Computes $(\theta_1 - R \theta_2)/(1-R)$

Value
Numeric

Author(s)
Long Qu

References

grid.search
Performs a grid search to minimize the objective function

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, rightBoundary = FALSE, 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
rightBoundary Logical; if TRUE, then the tail mean is computed from the right boundary of the chosen bin.
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

A numeric scalar value of estimated p-value density at 1.

Author(s)

Long Qu, Kun Liang
References


See Also

lastbin, qvalue

Examples

```r
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

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

Arguments

- `p`: a numeric vector the p-values
- `bw`: numeric, the bin width of histogram of p-values
- `trunc`: 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:

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

Value

a single numeric value as the estimate.

Author(s)

Long Qu

See Also

qvalue, histf1

Examples

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

Description

These functions return \( \pi_0 \) *(null test statistic density)/(marginal test statistic density).

Usage

```R
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'
```
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

logLik.ncpest

log likelihood from an object of class ncpest

Description

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

Usage

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

Arguments

object the object of class ncpest

... 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

marginaldt

Estimated marginal density of t-statistics

Description

Estimated marginal density of t-statistics from ncpest class

Usage

marginaldt(obj,...)
## S3 method for class 'parncpt'
marginaldt(obj,...)
## S3 method for class 'nparncpt'
marginaldt(obj,...)
## S3 method for class 'sparncpt'
marginaldt(obj,...)

Arguments

obj an object of ncpest (nparncpt or parncpt)
... Other argument passed to dtn_mix, most notably, the approximaiton 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

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 tOnly=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 pool is TRUE, this vector is of length 1, i.e. n1+n2-2; if pool is FALSE, this vector is of the same length as stat, 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

```r
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

Moments of truncated normal distribution and the integral in the noncentral t-distribution

Description

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

Usage

```r
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. int2 is exact for integer $r$ and interpolate to non-integer $r$. laplace uses laplacian approximation. numerical uses numerical integration.
- integral.only: logical. If TRUE, only the integral in noncentral t-distribution is returned. Otherwise, it is normalized to be the $r$th 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 mTruncNorm.int2

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
**Description**

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

**Usage**

```r	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=NULL,
    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
nparncpp.iter

keep.cdf Either NULL or an environment. If non-null, the computed conditional CDF will be saved keep.cdf. 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

See Also
nparncpt, sparncpt, parncpt, dncp

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=NULL,...)
Nonparametric estimation of noncentrality parameters

The functions use Gaussian basis functions to estimate the noncentrality parameters (ncp) from a large number of t-statistics.

Usage

nparncpt(tstat, df, ...)  
nparncpt.sqp(tstat, df, penalty=3L, lambdas=10^seq(-1,5,by=1), starts,  
IC=c('BIC','CAIC','HQIC','AIC'), K=100,  
bounds=quantile(tstat,c(.01,.99)),  
solver=c('solve.QP','lsei','ipop','LowRankQP'),  
plotit=FALSE, verbose=FALSE, approx.hess=TRUE, ... )
Arguments

- **tstat**: Numeric vector of noncentrality parameters
- **df**: Numeric vector of degrees of freedom
- **penalty**: An integer scalar among 1 through 5, indicating the order of derivatives of the estimated density function of ncp. The integral of square of such derivatives is the penalty to the log likelihood function. A character value among c('1st.deriv','2nd.deriv','3rd.deriv','4th.deriv','5th.deriv') is also accepted but deprecated.
- **lambdas**: Numeric vector of smoothness tuning parameter lambda to be tried. The one that minimizes NIC will be chosen.
- **starts**: Optional numeric vector of starting values. If missing, parncpt will be called with zeromean set to FALSE to get an initial estimate of pi0. And the starting values (theta) will be set all equal to each other and sum to 1-pi0. Note that this is the starting value for the largest lambdas only. For smaller lambdas, the estimates from larger lambdas will be used as starting values (i.e., warm start).
- **IC**: Character; one of AIC, BIC, CAIC, HQIC, specifying the factor multiplied to the ENP in computing Information Criterion (IC).
- **K**: The number of basis Gaussian density functions.
- **bounds**: A numeric vector of length 2, giving the approximate bounds where most of the probability of ncp lies.
- **solver**: Character. The name of the function for solving quadratic programming problems. Note that ipop and kernlab are not very reliable. solve.QP is faster but lsei is more stable.
- **plotit**: logical; indicating if plot.parncpt should be called after estimation. This is always recommended before accepting the results.
- **verbose**: logical; if TRUE, extensive messages will be printed.
- **approx.hess**: either logical or a number between 0 and 1. This helps in reducing time in evaluating the hessian matrix. If it is set to TRUE, for the kth Gaussian basis function and the gth tstat, the marginal t-statistic density evaluated at this tstat will be set to zero if it is below the average of all K*length(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 parameters passed to dtm.mix. Usually, the approximation argument.

Details

parncpt is a wrapper for parncpt.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")

- \( \pi_0 \) estimated proportion of true nulls
- \( \mu_{\text{ncp}} \) mean of ncp
- \( \text{sd}_{\text{ncp}} \) SD of ncp
- \( \text{logLik} \) an object of class \( \text{loglik} \). The associated df is the estimated effective number of parameters (enp). The log likelihood is also penalized likelihood. See also \( \text{loglik\.ncpest} \) and \( \text{AIC} \).
- \( \text{enp} \) estimated ENP
- \( \text{par} \) estimated parameters \( \theta \)
- \( \lambda \) the lambda that minimizes NIC
- \( \text{gradient} \) analytic gradient at the estimate
- \( \text{hessian} \) analytic hessian at the estimate
- \( \beta \) estimated mixing proportions for the NCP distribution
- \( \text{IC} \) the information criterion specified by the user
- \( \text{all\.mus} \) mean of each basis Gaussian density
- \( \text{all\.sigs} \) SD of each basis Gaussian density
- \( \text{data} \) a list of \text{tstat} and \text{df}
- \( \text{i.final} \) the index of \( \lambda \) that minimizes NIC
- \( \text{all\.pi0s} \) estimated \( \pi_0 \) for each lambda
- \( \text{all\.enps} \) ENP for each lambda
- \( \text{all\.thetas} \) parameter estimates for each lambda
- \( \text{all\.nics} \) Network information criterion (NIC) for each lambda
- \( \text{all\.nic.sd} \) SD of NIC for each lambda
- \( \text{all\.lambdas} \) the \( \lambda \) argument itself
- \( \text{nobs} \) the number of test statistics

Note

df could be \( \text{Inf} \) 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 \( \text{St} \)-statistics, with Applications to False Discovery Rate Estimation in Microarray Data Analysis. Biometrics, 68, 1178–1187.
parncpt

Parametric estimation of noncentrality parameter distribution

Description

Assuming normality of noncentrality parameters (parncpt) or a mixture of two normal distributions (parncpt2), the MLE of its standard deviation(s) (and possibly mean(s) also) is estimated from observed t-statistics.

Usage

parncpt(tstat, df, zeromean = TRUE, ...)  
parncpt.bfgs.omean(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)  
parncpt2(tstat, df, common = c("mean", "sd"), ...)

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 not estimated.  
common character vector. Allowed values are 'mean', 'sd', 'none'. If 'none' is present, common must be a scalar, and an unrestricted 2-component normal mixture is fit to ncp distribution. NULL is treated the same as 'none'. If mean is present, the means of the two normal components of the ncp distribution are assumed to be negative of each other. If sd is present, the standard deviations of the two normal components of the ncp distribution are assumed to be common.  
... Other arguments to optim.
parncpt

**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 pi0, 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.

parncpt2 calls parncpt2.constrOptim to find the maximum likelihood estimates of parameters when the noncentrality parameter distribution is assumed to be a mixture of two normals. The parameterization being used is such that pi0 is the proportion of true nulls and pi1 is the proportion of non-nulls of which the noncentrality parameters come from the normal component with smaller mean. Therefore, for the noncentrality parameter distribution, tau=pi1/(1-pi0) is the mixing proportion for the normal component with smaller mean.

**Value**

Except for parncpt2, the result is a list with class attribute being c('parncpt', 'ncpest').

pi0
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 logLik. Call logLik.ncpest to extract. Similarly, AIC is callable.

enp
the (effective) number of parameters in the model
parncpt

par estimated parameters. Call coef.ncpest to extract.
obj the negative loglikelihood function that is minimized
gradient analytic gradient at the estimate
hessian numeric hessian at the estimate
nobs the number of test statistics

For parncpt2, the result is a list with class attribute being c('parncpt2', 'parncpt', 'ncpest'), which is a list with the following additional components:

pi1 proportion of non-nulls of which the noncentrality parameters come from the normal component with smaller mean.
tau.ncp the mixing proportion of the normal component of the ncp distribution with smaller mean.
mu1.ncp the mean of the normal component of the ncp distribution with smaller mean.
sd1.ncp the SD of the normal component of the ncp distribution with smaller mean.
mu2.ncp the mean of the normal component of the ncp distribution with larger mean.
sd2.ncp the SD of the normal component of the ncp distribution with larger mean.

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 ncpest class.

Author(s)
Long Qu

References

See Also
sparcpt, nparcpt, fitted.parcpt, plot.parcpt, summary.parcpt, coef.ncpest, logLik.ncpest, vcov.ncpest, AIC, dncp

Examples
## Not run:
data(simulatedTstat)
(pfit=parncpt(tstat=simulatedTstat, df=8, zeromean=FALSE)); plot(pfit)
(pfit0=parncpt(tstat=simulatedTstat, df=8, zeromean=TRUE)); plot(pfit0)
(pfit2=parncpt2(tstat=simulatedTstat, df=8)); plot(pfit2)
## 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**

```r
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

**Distance between densities**

**Description**

Compute the distance between two density functions

**Usage**

```r
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 $\infty$. 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**

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

# absolute difference between standard normal and standard cauchy
f2=function(x)dt(x,1)
df.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**

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

**Arguments**

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

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


See Also

rgl, extrp.pi0

Examples

## Not run:

```r
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 (pi0) of true null hypotheses for different choices of tuning parameters; also plot the estimated density of noncentrality parameters.

Usage

```r
## 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


See Also

`nparncpt`, `sparncpt`, `parncpt`

Examples

```r
## 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.nparncpt**

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

```r
## S3 method for class 'parncpt'
plot(x, ...)  
## S3 method for class 'parncpt2'
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


See Also

`parncpt`, `nparncpt`, `sparncpt`

Examples

```r
## 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

plot an object of class sparncpt, i.e., semiparametric estimate of non-centrality 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

```r
## 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


See Also

`parncpt`, `nparncpt`, `sparncpt`
Examples

```r
## Not run:
data(simulatedTstat)
(npfite=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=sparcpt(npfite,pfit)); plot(spfit)

## End(Not run)
```

plot.subex  

---

### 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

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

### Arguments

- `x`  
  the subex object

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

- `rgl`  
  logical, specifying whether or not the `rgl` package is used for making better 3D interactive graphs.

- `...`  
  other arguments to be passed to `persp3d`, or `persp` 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 \( \pi_0 \), 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 \( \pi_0 \). 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

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,TRUE)

## 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


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)
Arguments

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

Value

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

Author(s)

Long Qu

References


See Also

eextrp.pi0

Examples

```r
## 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)
```
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


See Also

subex, extrpi0, 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)
Usage

```r
## 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


See Also

`subt`

Examples

```r
## 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

**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**

```r
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**

```r
## 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(ddexp)

## folded normal pdf
dfnorm=fold(dnorm)
```
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 $\pi_0$ 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

```r
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 $G_1$ standardized effect sizes, where $G_1$ 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 $G \times N$ 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 $G_1 = \text{round}(G \times \pi_0)$. 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.
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

```r
set.seed(9992722)
simulatedDat = sim.dat(G=5000)
```

References


See Also

`sim.dat`
Examples

data(simulatedDat)

A Simulated 'extrpi0' Object

Description

This is the result from calling extr.pi0 on simulatedSubt.

Usage

data(simulatedExtrpi0)

Format

an object of class extrpi0.

Details

This is the result from calling

data(simulatedSubt)
simulatedExtrpi0=extr.pi0(simulatedSubt)

References


See Also

extr.pi0

Examples

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

# End(Not run)
simulatedSubex

A Simulated 'subex' Object

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


See Also

subex

Examples

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

## End(Not run)
simulatedSubt

A Simulated 'subt' Object

Description

This is the result from calling \code{subt} on \code{simulatedDat}.

Usage

data(simulatedSubt)

Format

an object of class \code{subt} with \code{balanced=FALSE} with \code{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

\code{subt}

Examples

data(simulatedSubt)
print(simulatedSubt)

## Not run:
plot(simulatedSubt)

## End(Not run)
**simulatedTstat**

```
  t-Statistics and p-Values from a Simulated Microarray Data Set
```

**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(simulatedDat)
```

**References**


**See Also**

`sim.dat`, `simulatedDat`

**Examples**

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

**sparncpt**

```
Semiparametric density estimation for noncentrality parameters
```

**Description**

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

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 of 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', 'ncpest'):

- `pi0` estimated proportion of true nulls
- `mu.ncp` mean of ncp
- `sd.ncp` SD of ncp
- `logLik` an object of class `logLik`. 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
- `nobs` the number of test statistics

Author(s)

Long Qu
## References


## See Also

parncpt, nparncpt, fitted.sparncpt, plot.sparncpt, summary.sparncpt, coef.ncpest, loglik.ncpest, vcov.ncpest, AIC, dncp

## Examples

```r
## 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)
```

### 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

```r
subex(dat, n1 = round(ncol(dat)/2), n2 = ncol(dat) - n1,
      f1method = 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 ot 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 extrpNpiP 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 \( \pi_0 \) as well as the false discovery rates (FDR) based on the estimated \( \pi_0 \).

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

- \( \pi_0 \)  a numeric value, giving the estimated \( \pi_0 \)
- extrp.fit  an object of class extrpi0, the results from calling extrp.pi0.
pvalues  a numeric vector of length the same as nrow(dat), the p-values for each gene.
qvalues  a numeric vector of length the same as nrow(dat), 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
subt

Subsampling a Microarray Data Set for Estimating Proportion of True Null Hypotheses

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 \( n_1 \) columns correspond to treatment group 1 and the rest \( n_2 \) 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 \( \infty \), then all possible subsamples will be tried. However, see Notes and the \( R \) argument of `combnRr`.

See Also

```
subt, extrp.pi0, matrix.t.test, fdr, plot.subex, print.subex
```

Examples

```r
## 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)
```
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 $M=3,4,...,N$, where $N=n1+n2$, do the following,

- 1For each treatment 1 subsample size $m1=1,2,...,n1$, let $m2=M-m1$. If $1\leq m2\leq n2$ 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 $m1$ subjects from treatment group 1 and $m2$ subjects from treatment group 2, by using the function `combn2R` with `sample.method="diff2"` and `try.rest=TRUE`. 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 group1 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

**summary.nparncpt**

Print summary for an object of class nparncp

**Description**

Print summary for an object of class nparncp. This includes the estimated proportion of true null hypotheses (\(\pi_p\)), 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 (\(\lambda\)) that controls the amount of smoothing.

**Usage**

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

**Arguments**

- `object, x` an object of class nparncp
- `...` currently not used

**Value**

the `invisible` object itself

**Author(s)**

Long Qu

---

**See Also**

`print.subt, plot.subt, extrp.pi0, matrix.t.test, combn2R, subex, lastbin, qvalue`

**Examples**

```r
## 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)
```
References


See Also

parncpt, nparncpt, sparcpt

summary.parncpt

Print summary for an object of class parncpt

Description

Print summary for an object of class parncpt. This includes the estimated proportion of true null hypotheses ($\pi_0$), 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,...)
## S3 method for class 'parncpt2'
summary(object,...)
## S3 method for class 'parncpt2'
print(x,...)

Arguments

object,x  
an object of class parncpt

...  
currently not used

Value

the invisible object itself

Author(s)

Long Qu

References

summary.sparncpt

See Also

parncpt, nparncpt, sparncpt

summary.sparncpt

Print summary for an object of class sparncpt

Description

Print summary for an object of class sparncpt. This includes the estimated proportion of true null hypotheses (\(\pi_0\)), estimated mean of noncentrality parameters (\(\mu_{ncp}\)), estimated standard deviation of noncentrality parameters (\(sd_{ncp}\)), and effective number of parameters (\(enp\)).

Usage

```r
## 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, 68, 1178–1187.

See Also

parncpt, nparncpt, sparncpt
**varB**  
*Variance of the reflected Bsplines*

**Description**  
2nd order raw moment of the Bsplines

**Usage**  
```r
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**  
*extract inverse Hessian matrix from ncpest class*

**Description**
return inverse Hessian matrix from ncpest class

**Usage**
```r
## S3 method for class 'ncpest'
vcov(object, ...)
```

**Arguments**
- `object` an object of class ncpest
- `...` 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 pi0

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 pi0 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 \( z = \text{qnorm}(1-p) \).
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 implemented 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


See Also

qvalue, histfQ

Examples

set.seed(99722)
p=1-pnorm(c(rnorm(7000),rnorm(3000,1)))
znormix(p)[`pi0`]
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, 16
  dt.int2, 17
  dt.lap, 18
  dt.sad and pt.sad, 19
  dtn.mix, 20
  marginal.dt, 33
  mTruncNorm, 35
  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.parncpt, 48
  plot.parnncpt, 50
  plot.subex, 51
  plot.subt, 52
*Topic htest
  CBUM, 6
  coef.ncpest, 8
  convest, 12
  extrp.pi0, 21
  fdr, 23
  lastbin, 30
  logLik.ncpest, 32
  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
  lfdrr and ppee, 31
  plot.extrpi0, 46
  plot.nparncpt, 47
  plot.parncpt, 48
  plot.parnncpt, 50
  plot.subex, 51
  plot.subt, 52
  print.extrpi0, 53
  print.subex, 54
  print.subt, 55
  summary.nparncpt, 69
  summary.parncpt, 70
  summary.parnncpt, 71
*Topic models
  discTMix, 14
  nparncpt, 39
  parncpt, 42
  pi0-package, 3
  sparncpt, 63
*Topic multivariate
CBUM, 6
extrp.pi0, 21
fdr, 23
lastbin, 30
pi0-package, 3
sim.dat, 58
subex, 65
subt, 67
znormix, 73

*Topic **nonlinear**
extrp.pi0, 21
pi0-package, 3
subex, 65

*Topic **nonparametric**
agjack, pi0, 5
dncp, 16
extrp.pi0, 21
fdr, 23
pi0-package, 3
subex, 65
subt, 67

*Topic **optimize**
extrp.pi0, 21
gird.search, 28
nparncpt, 39
parncpt, 42
pi0-package, 3
subex, 65

*Topic **package**
pi0-package, 3

*Topic **print**
print.extrp, 53
print.subex, 54
print.subt, 55

*Topic **regression**
extrp.pi0, 21
subex, 65

*Topic **smooth**
fitted, 25
fitted.discTMix, 26
nparncpp, 37
nparncpp.iter, 38
nparncpt, 39
pi0-package, 3
sparncpt, 63
varB, 72

*Topic **univar**
discTMix, 14

geomean and harmean, 27
gjack, 27
histfl, 29
marginal.dt, 33
mTruncNorm, 35
pavafl, 45
pdf.dist, 45
reflect and fold, 57

*Topic **utilities**
clean.cdf, save.cdf and load.cdf, 7
combn2R, 9

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), 27
gjack (gjack), 27
harmean (geomean and harmean), 27
histfl (histfl), 29
load.cdf (clean.cdf, save.cdf and load.cdf), 7
nparncpp (nparncpp), 37
nparncpp.iter (nparncpp.iter), 38
pavafl (pavafl), 45
save.cdf (clean.cdf, save.cdf and load.cdf), 7

varB (varB), 72

agjack.pi0, 5
AIC, 33, 41–44, 64, 65
apply, 34, 35
array, 11
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, 39
constrOptim, 23
convest, 12, 13, 32

discTMix, 14, 26, 32
dncp, 16, 38, 42, 44, 65
dt, 18–20, 36, 43
dt.int2, 17, 18–21, 36
dt.lap, 18, 18, 19, 21
dt.sad, 18, 20, 21
dt.sad (dt.sad and pt.sad), 19
dt.sad and pt.sad, 19
dtn.mix, 18, 20, 33, 40, 64

EOC, 16
extrp.pi0, 3, 4, 21, 47, 53–55, 60, 65–67, 69

fdr, 3, 4, 23, 32, 55, 65–67
fitted, 25
fitted.discMix, 16, 26
fitted.nparncpt, 42
fitted.parncpt, 44
fitted.sparcnt, 65
fold (reflect and fold), 57

geomean and harmean, 27
gjack, 5, 6, 27
grid.search, 28

histf1, 7, 29, 31, 74

invisible, 49, 50, 69–71

lastbin, 4, 30, 30, 69
lfdr (lfdr and ppee), 31
lfdr and ppee, 31
list, 11
logLik, 33
logLik.ncpest, 32, 41–44, 64, 65
LowRankQP, 38
lsei, 23, 38

marginal.dt, 33
matrix.t.test, 4, 34, 66, 67, 69
mt.rawp2adjp, 24
mTruncNorm, 18, 35

NBsplines, 72
nparsn cpp, 4, 9, 17, 26, 32, 37, 38
nparsn cpp.iter, 38

parncpt, 4, 9, 17, 26, 32–34, 38, 42, 65, 48–50, 63–65, 70, 71, 73

optim, 16, 29, 42, 43
optimize, 23

parncpt, 4, 9, 17, 26, 32–34, 38, 42, 42, 48–50, 63–65, 70, 71, 73

parncpt2, 17, 26
parncpt2 (parncpt), 42
pavafl, 45
pdf.dist, 45
persp, 46, 51
persp3d, 46, 51
pi0 (pi0-package), 3
pi0-package, 3
plot.extrpi0, 23, 46, 51, 52, 66
plot.nparncpt, 40, 42, 47
plot.parncpt, 44, 48
plot.parncpt2 (plot.parncpt), 48
plot.sparcnt, 50, 65
plot.subex, 51, 67
plot.subt, 52, 69
plot3d, 52
ppee (lfdr and ppee), 31
print.extrpi0, 23, 53, 55
print.nparncpt (summary.nparncpt), 69
print.parncpt (summary.parncpt), 70
print.parncpt2 (summary.parncpt), 70
print.sparcnt (summary.sparcnt), 71
print.subex, 54, 67
print.subt, 55, 69
pt, 12, 36
pt.sad (dt.sad and pt.sad), 19
qvalue, 7, 23, 24, 30, 31, 69, 73, 74

reflect (reflect and fold), 57
reflect and fold, 57

scatterplot3d, 52
sim.dat, 58, 59, 63
simulatedDat, 59, 61–63
simulatedExtrpi0, 60
simulatedPval (simulatedTstat), 63
simulatedSubex, 61
simulatedSubt, 60, 62
simulatedTstat, 63
sparcnt, 4, 9, 17, 26, 32–34, 38, 42, 44, 48–50, 63, 70, 71
subex, 4, 23, 51, 52, 55, 61, 65, 69
summary.extrpi0 (print.extrpi0), 53
summary.nparncpt, 42, 69
summary.parncpt, 44, 70
summary.parncpt2 (summary.parncpt), 70
summary.sparcnt, 65, 71
summary.subex (print.subex), 54
summary.subt (print.subt), 55

t.test, 12, 35, 37
tMixture, 14, 15
tstatistics, 16

varB, 72
vcov.ncpest, 42, 44, 65, 72

znormix, 4, 32, 73