OCplus

April 19, 2010

`average.fdr` *Average a two-dimensional local false discovery rate*

Description

This function averages two-dimensional local false discovery rates as computed by `fdr2d` for binned values of the first test statistic and across the values of the second test statistic. The result can easily be plotted and should be comparable to the one-dimensional fdr as provided by `fdr1d`, provided that the smoothing parameters were chosen suitably.

Usage

```
average.fdr(x, breaks)
```

Arguments

`x`	an object returned by `fdr2d`.
`breaks`	breaks defining intervals into which the first test statistic is binned; by default the same values that were used by `fdr2d`.

Details

Assuming that we have smoothed the estimate suitably and have chosen the proportion of non-dffierentially expressed genes suitably, we should get very much the same results from `fdr2d` as from `fdr1d` when we average across the logarithmized standard errors, see Examples.

The averaging is done across the estimated values for the actual genes; this corresponds to a weighted mean of the smoothed estimates on a grid, where the weight is proportional to cell frequencies.

Note that it is usuually easier to get a good match in the tails of the curves than in the center, which is okay, as this is where we want to estimate the fdr reliably.

Value

A matrix with two columns `tstat` and `fdr.local`.

Author(s)

A. Ploner

References

Ploner A, Calza S, Gusnanto A, Pawitan Y (2005) Multidimensional local false discovery rate for micorarray studies. *Submitted Manuscript*.

See Also

fdr2d, fdr1d

Examples

```
# Create res1d
example(fdr1d)

# Compute fdr2d using the p0
res2d = fdr2d(xdat, grp, p0=p0(res1d))

# Show it
par(mfrow=c(2,1))
plot(res1d, main="fdr1d and averaged fdr2d")
lines(average.fdr(res2d), col="red")
plot(res2d, grid=TRUE, main="fdr2d is averaged across columns")
```

DrawContourlines *Draw and label a set of pre-calculated isolines*

Description

This function draws and labels isolines computed by `contourLines`, though the labelling is done very clumsily and with a specialized application in mind.

Usage

```
DrawContourlines(x, label = FALSE, cex = 0.7, vfont = c("sans serif", "bold"), .
```

Arguments

`x`	a list of isolines as produced by `contourLines`.
`label`	a logical value indicating whether to label the isolines.
`cex`	size of labels
`vfont`	a vector font specification for the labels as in `contour`.
`...`	extra arguments to `lines`

Details

This routine is used by `Tornadoplot` and `Volcanoplot` to draw and label isolines that were computed via contourLines and afterwards transformed. The problem is that all the nice options that `contour` has for labelling isolines are not avaiable independently, so this function uses the following crude procedure that kind of works for the intended applications:

- isoline completely left of zerolabel the leftmost point;
- isoline completely right of zerolabel the rightmost point;

- isoline crosses zero horizontallylabel the topmost point.

Hopefully, one of these days someone will come up with a nice general-purpose function for doing all the nifty stuff that `contour` offers.

Author(s)

A. Ploner

See Also

`contour`, `contourLines`, `Tornadoplot`

`EOC` *Estimated or empirical FDR, sensitivity, etc as a function of cutoff level*

Description

`EOC` computes and optionally plots the estimated operating characteristics for data from a microarray experiment with two groups of subjects. The false discovery rate (FDR) is estimated based on random permutations of the data and plotted against the cutoff level on the t-statistic; a curve for the classical sensitivity can be added. Different curves for different proportions of non-differentially expressed genes can be compared in the same plot, and the sample size per group can be varied between plots.

`FDRp` is the function that does the underlying hard work and requires package `multtest`.

Usage

```
EOC(xdat, grp, p0, paired = FALSE, nperm = 25, seed = NULL, plot = TRUE, ...)

FDRp(xdat, grp, test = "t.equalvar", p0, nperm, seed)
```

Arguments

`xdat`	the matrix of expression values, with genes as rows and samples as columns
`grp`	a grouping variable giving the class membership of each sample, i.e. each column in `xdat`; for `EOC`, this can be any type of variable, as long as it has exactly two distinct values, whereas `FDRp` expects to see only 0s and 1s, see Details.
`p0`	if supplied, an estimate for the proportion of non-differentially expressed genes; if not supplied, the routine will estimate it, see Details.
`paired`	logical value indicating whether this is independent sample situation (default) or a paired sample situation. Note that paired samples need to follow each other in the data matrix (as in 010101...when `paired=TRUE`.
`nperm`	number of permutations for establishing the null distribution of the t-statistic
`test`	the type of test to use, see `mt.teststat`; when called from `EOC`, this is always the default.
`seed`	the random seed from which the permutations are started
`plot`	logical value indicating whether to do the plot
`...`	graphical parameters, passed to `plot.FDR.result`

Details

EOC is the empirical counterpart of the function TOC. It estimates the FDR and sensitivity for a given data set of expression values measured on subjects in two groups. The FDR is estimated locally based on the empirical Bayes approach outlined by Efron et al., see References. FDRp implements the details of this method; this requires among other things the permutation distribution of the t-statistic, which is calculated via a call to function mt.teststat of package multtest. This explains why both functions barf at missing values in the expression data.

Note that p0 is by default estimated from the data, as originally suggested by Efron et al. so as to make ratio between the densities of the observed distribution of t-statistics and the permutation distribution smaller than 1; alternatively, the user can supply his own guesstimate of the proportion of non-differentially expressed genes in the data.

Note also that FDRp keeps all permuations in the memory during compuations. For a large number of genes, this will limit the number of possible permuations.

Value

For EOC, an object of class FDR.result, which inherits from class data.frame. The three columns list for each gene its t-statistic, the estimated FDR (two-sided), and the estimated sensitivity. Additionally, the object carries an attribute param, which is a list with four entries: p0, the assumed proportion of non-differentially expressed genes used in calculating the FDR; p0.est, a logical value indicating whether p0 was estimated or user-supplied; statistic indicates how the t-statistic was computed, i.e. how its sign should be interpreted in terms of relative over- or under expression, and a logical flag paired to indicate whether a paired t-statistic was used.

FDRp returns a list with essentially the same elements, plus additionally the values of the observed and permuted distribution of the t-statistics for each gene.

Note

Both the curve labels and the legend may be squashed if the plotting device is too small. Increasing the size of the device and re-plotting should improve readability.

Author(s)

Y. Pawitan and A. Ploner

References

Pawitan Y, Michiels S, Koscielny S, Gusnanto A, Ploner A (2005) False Discovery Rate, Sensitivity and Sample Size for Microarray Studies. *Bioinformatics*, 21, 3017-3024.

Efron B, Tibshirani R, Storey JD, Tusher V. (2001) Empirical Bayes Analysis of a Microarray Experiment. *JASA*, 96(456), p. 1151-60.

See Also

plot.FDR.result, OCshow, mt.teststat

Examples

```
# We simulate a small example with 5 percent regulated genes and
# a rather large effect size
set.seed(2003)
xdat = matrix(rnorm(50000), nrow=1000)
```

```
xdat[1:25, 1:25] = xdat[1:25, 1:25] - 2
xdat[26:50, 1:25] = xdat[26:50, 1:25] + 2
grp = rep(c("Sample A","Sample B"), c(25,25))

# The default, with legend
ret = EOC(xdat, grp, legend=TRUE)
# Look at the results: yes
ret[1:10,]
which(ret$FDR<0.05)
# Extra information
attr(ret,"param")

# Run the same data with different permutations: fairly stable, but with
# different p0
ret = EOC(xdat, grp, seed=2000)
which(ret$FDR<0.07)

# Misspecify the p0: not too bad here
ret = EOC(xdat, grp, p0=0.99)
which(ret$FDR<0.01)

# We simulate data in a paired setting
# Note the arrangement of the columns
set.seed(2004)
xdat = matrix(rnorm(50000), nrow=1000)
ndx1 = seq(1,50, by=2)
xdat[1:25, ndx1] = xdat[1:25, ndx1] - 2
xdat[26:50, ndx1] = xdat[26:50, ndx1] + 2
grp = rep(c("Sample A","Sample B"), 25)

ret = EOC(xdat, grp, paired=TRUE)
which(ret$FDR<0.05)
```

`fdr1d` *Compute classical local false discovery rate*

Description

Calculates the classical local false discovery rate for multiple parallel t-statistics.

Usage

```
fdr1d(xdat, grp, test, p0, nperm = 100, nr = 50, seed = NULL, null = NULL,
      zlim = 1, sv2 = 0.01, err = 1e-04, verb = TRUE, ...)
```

Arguments

`xdat`	the matrix of expression values, with genes as rows and samples as columns
`grp`	a grouping variable giving the class membership of each sample, i.e. each column in `xdat`
`test`	a function that takes `xdat` and `grp` as the first two arguments and returns the test statistic; by default, two-sample t-statistics are calculated.

p0	if supplied, an estimate for the proportion of non-differentially expressed genes; if not supplied, the routine will estimate it, see Details.
nperm	number of permutations for establishing the null distribution of the t-statistic
nr	the number of equidistant breaks into which the range of test statistics is broken for calculating the fdr.
seed	if specified, the random seed from which the permuations are started
null	optional argument for passing in a pre-calculated null distribution, see Details.
zlim	if no p0 is specified, the ratio of densities in the range of test statistics between -zlim and zlim will be used to estimate the proportion of non-differentially expressed genes; ignored if p0 is specified.
sv2	positive number controlling the initial degree of smoothing for the densities involved, with smaller values indicating more smoothing; see Details.
err	positive number controlling the convergence of the smoothing procedure, with smaller values implying more iterations; see Details.
verb	logical value indicating whether provide extra information.
...	extra arguments to function test.

Details

This function calculates the local false discovery rate (fdr, as opposed to global FDR) introduced by Efron et al., 2001. The underlying model assumes that for a given grouping of samples, we study a mixture of differentially expressed (DE) and non-DE genes, and that consequently, the observed distribution of test statistics is a mixture of test statistics under the alternative and the null statistic, respectively. The densities involved are estimated nonparametrically and smoothed, using a permutation argument for the null distribution.

By default, the null distribution is generated and stored only within the function, and is not available outside. If someone wants to study the null distribution, or wants to re-use the same null distribution efficiently while e.g. varying the smoothing parameter, the argument null allows the use of an externally generated null distribution, created e.g. using the PermNull function.

Theoretically, the function should support any kind of function along the lines of tstatistics, however, this has not been tested, and the current setup is very much geared towards t-tests.

We use non-Gaussian mixed model smoothing for Possion counts for smoothing the density estimates, see Pawitan, 2001, and smooth1d.

Value

Basically, a data frame with one row per gene and two columns: tstat, the test statistic, and fdr.local, the local false discovery rate. This data frame has the additional class attributes fdr1d.result and fdr.result, see Examples. This is the bad old S3 class mechanism employed to provide plot and summary functions.

Additional information is provided by a param attribute, which is a list with the following entries:

p0	the proportion of non-differentially expressed genes used when calculating the fdr.
p0.est	a logical value indicating whether p0 was estimated from the data or supplied by the user.
fdr	the smoothed fdr values calculated for the original intervals.
xbreaks	vector of breaks for the test statistic defining the interval for calculation.

Author(s)

A. Ploner

References

Efron B, Tibshirani R, Storey JD, Tusher V (2001) Empirical Bayes Analysis of a Microarray Experiment. *JASA*, 96(456), p. 1151-60.

Pawitan Y.(2001) *In All Likelihood*, Oxford University Press, ch. 18.11

See Also

`plot.fdr1d.result`, `summary.fdr.result`, `OCshow`, `tstatistics`, `smooth1d`, `fdr2d`, `PermNull`

Examples

```
# We simulate a small example with 5 percent regulated genes and
# a rather large effect size
set.seed(2000)
xdat = matrix(rnorm(50000), nrow=1000)
xdat[1:25, 1:25] = xdat[1:25, 1:25] - 1
xdat[26:50, 1:25] = xdat[26:50, 1:25] + 1
grp = rep(c("Sample A","Sample B"), c(25,25))

# A default run
res1d = fdr1d(xdat, grp)
res1d[1:20,]

# Looking at the results
summary(res1d)
plot(res1d)
res1d[res1d$fdr<0.05, ]

# Averaging estimates the global FDR for a set of genes
ndx = abs(res1d$tstat) > 3
mean(res1d$fdr[ndx])

# Extra information
class(res1d)
attr(res1d,"param")
```

`fdr2d` *Compute two-dimensional local false discovery rate*

Description

This function calculates the local false discovery rate for a two-sample problem using a bivariate test statistic, consisting of classical t-statistics and the corresponding logarithmized standard error.

Usage

```
fdr2d(xdat, grp, test, p0, nperm = 100, nr = 15, seed = NULL, null = NULL,
  constrain = TRUE, smooth = 0.2, verb = TRUE, ...)
```

Arguments

xdat	the matrix of expression values, with genes as rows and samples as columns
grp	a grouping variable giving the class membership of each sample, i.e. each column in xdat
test	a function that takes xdat and grp as the first two arguments and returns the bivariate test statistics as two-column matrix; by default, two-sample t-statistics and logrithmized standard errors are calculated.
p0	if supplied, an estimate for the proportion of non-differentially expressed genes; if not supplied, the routine will estimate it, see Details.
nperm	number of permutations for establishing the null distribution of the t-statistic
nr	the number of equidistant breaks for the range of each test statistic; fdr values are calculated on the resulting (nr-1) x (nr-1) grid of cells.
seed	if specified, the random seed from which the permuations are started
null	optional argument for passing in a pre-calculated null distribution, see Examples.
constrain	logical value indicating whether the estimated fdr should be constrained to be monotonously decreasing with the absolute size of the t-statistic (more generally, the first test statistic).
smooth	a numerical value between 0.01 and 0.99, indicating which percentage of the available degrees of freedom are used for smoothing the fdr estimate; larger values indicate more smoothing.
verb	logical value indicating whether provide extra information.
...	extra arguments to function test.

Details

This routine computes a bivariate extension of the classical local false discovery rate as available through function fdr1d. Consequently, many arguments have identical or similar meaning. Specifically for fdr2d, nr specifies the number of equidistant breaks defining a two-dimensional grid of cells on which the bivariate test statistics are counted; argument constrain can be set to ensure that the estimated fdr is decreasing with increasing absolute value of the t-statistic; and argument smooth specifies the degree of smoothing when estimating the fdr.

Note that while fdr2d might be used for any suitable pair of test statistics, it has only been tested for the default pair, and the smoothing procedure specifically is optimized for this situation.

Note also that the estimation of the proportion p0 directly from the data may be quite unstable and dependant on the degree of smoothing; too heavy smoothing may even lead to estimates greater than 1. It is usually more stable use an estimate of p0 provided by fdr1d.

Note that fdr1d can also be used to check the degree of smoothing, see average.fdr.

Value

Basically, a data frame with one row per gene and three columns: tstat, the test statistic, logse, the corresponding logarithmized standard error, and fdr.local, the local false discovery rate. This data frame has the additional class attributes fdr2d.result and fdr.result, see Examples. This is the bad old S3 class mechanism employed to provide plot and summary functions.

Additional information is provided by a param attribute, which is a list with the following entries:

p0	the proportion of non-differentially expressed genes used when calculating the fdr.

p0.est	a logical value indicating whether p0 was estimated from the data or supplied by the user.
fdr	the matrix of smoothed fdr values calculated on the original grid.
xbreaks	vector of breaks for the first test statistic.
ybreaks	vector of breaks for the second test statistic.

Author(s)

A Ploner and Y Pawitan

References

Ploner A, Calza S, Gusnanto A, Pawitan Y (2005) Multidimensional local false discovery rate for micorarray studies. *Submitted Manuscript*.

See Also

plot.fdr2d.result, summary.fdr.result, OCshow, fdr1d, average.fdr

Examples

```
# We simulate a small example with 5 percent regulated genes and
# a rather large effect size
set.seed(2000)
xdat = matrix(rnorm(50000), nrow=1000)
xdat[1:25, 1:25] = xdat[1:25, 1:25] - 1
xdat[26:50, 1:25] = xdat[26:50, 1:25] + 1
grp = rep(c("Sample A","Sample B"), c(25,25))

# A default run
res2d = fdr2d(xdat, grp)
res2d[1:20,]

# Looking at the results
summary(res2d)
plot(res2d)
res2d[res2d$fdr<0.05, ]

# Extra information
class(res2d)
attr(res2d,"param")
```

FDR — *Compute FDR for general scenarios*

Description

FDR computes the false discovery rate for comparing gene expression between two groups of subjects when the distribution of the test statistic under the null and alternative hypothesis are both mixtures of t-distributions. CDF and CDFmix calculate these mixtures.

Usage

```
FDR(x, n1, n2, pmix, D0, p0, D1, p1, sigma)

CDF(x, n1, n2, D, p, sigma)
CDFmix(x, n1, n2, pmix, D0, p0, D1, p1, sigma)

FDR.paired(x, n, pmix, D0, p0, D1, p1, sigma)

CDF.paired(x, n, D, p, sigma)
CDFmix.paired(x, n, pmix, D0, p0, D1, p1, sigma)
```

Arguments

x	vector of quantiles (two-sample t-statistics)
n, n1, n2	vector of sample sizes (as subjects per group)
pmix	the proportion of non-differentially expressed genes
D0	vector of effect sizes for the null distribution
p0	vector of mixing proportions for D0; must be the same length as D0 and sum to one
D1	vector of effect sizes for the alternative distribution
p1	vector of mixing proportions for D1, same as p0
D, p	generic vectors of effect sizes and mixing proportions as above
sigma	the standard deviation

Details

These functions are designed for a simple experimental setup, where we wish to compare gene expression between two groups of subjects of size n1 and n2 for an unspecified number of genes, using an equal-variance t-statistic.

100pmix% of the genes are assumed to be not differentially expressed. The corresponding t-statistics follow a mixture of t-distributions; this is more general than the usual central t-distribution, because we may want to include genes with biologically small effects under the null hypothesis (Pawitan et al., 2005). The other 100(1-pmix)% genes are assumed to be differentially expressed; their t-statistics are also mixtures of t-distributions.

The mixture proportions of t-distributions under the null and alternative hypothesis are specified via p0 and p1, respectively. The individual t-distributions are specified via the means D0 and D1 and the standard deviation sigma of the underlying data (instead of the mathematically more obvious, but less intuitive non centrality parameters). As the underlying data are the logarithmized expression values, D0 and D1 can be interpreted as average log-fold change between conditions, measured in units of sigma. See Examples.

CDF computes the cumulative distribution function for a mixture of t-distributions based on means D and standard deviation sigma with mixture proportions p. This function is the work horse for CDFmix.

Note that the base functions (FDR, CDFmix, CDF) assume two groups of experimental units; the .paired functions provide the same functionality for one group of paired observations.

The distribution functions call pt for computation; correspondingly, the quantiles x and all arguments that define degrees of freedom and non centrality parameters (n1, n2, D0, D1, sigma) can be vectors, and will be recycled as necessary.

Value

The appropriate vector of FDRs or probabilities.

Author(s)

Y. Pawitan and A. Ploner

References

Pawitan Y, Michiels S, Koscielny S, Gusnanto A, Ploner A. (2005) False Discovery Rate, Sensitivity and Sample Size for Microarray Studies. *Bioinformatics*, 21, 3017-3024.

See Also

TOC, samplesize

Examples

```
# FDR for H0: 'log fold change is zero'
#     vs. H1: 'log fold change is -1 or 1'
#             (ie two-fold up- or down regulation)
FDR(1:6, n1=10, n2=10, pmix=0.90, D0=0, p0=1,
    D1=c(-1,1), p1=c(0.5, 0.5), sigma=1)

# Include small log fold changes in the H0
# Naturally, this increases the FDR
FDR(1:6, n1=10, n2=10, pmix=0.90, D0=c(-0.25,0, 0.25), p0=c(1/3,1/3,1/3),
    D1=c(-1,1), p1=c(0.5, 0.5), sigma=1)

# Consider an asymmetric alternative
# 10 percent of the regulated genes are assumed to be four-fold upregulated
FDR(1:6, n1=10, n2=10, pmix=0.90, D0=0, p0=1,
    D1=c(-1,1,2), p1=c(0.45, 0.45, 0.1), sigma=1)
```

MAsim.smyth *Simulate two-sample microarray data*

Description

These functions simulate two-sample microarray data from various different models.

Usage

```
MAsim(ng = 10000, n = 10, n1 = n, n2 = n, D = 1, p0 = 0.9, sigma = 1)


MAsim.var(ng = 10000, n = 10, n1 = n, n2 = n, D = 1, p0 = 0.9)

MAsim.smyth(ng = 10000, n = 10, n1 = n, n2 = n, p0 = 0.9, d0 = 4,
            s2_0 = 4, v0 = 2)

MAsim.real(xdat, grp, n, n1, n2, D = 1, p0 = 0.9, replace = TRUE)
```

Arguments

ng	number of genes
n, n1, n2	number of samples per group; by default balanced, except for MAsim.real.
p0	proportion of differentially expressed genes
D	effect size for differentially expressed genes, in units of the gene-specific standard deviation (sigma in MAsim).
sigma	standard deviation, constant for all genes
d0, s2_0, v0	prior parameters for effect size and variability across genes in Smyth's model, see Details.
xdat, grp	expression data and grouping variable for an existing microarray data set, as specified in EOC.
replace	logical switch indicating whether to sub-sample (replace=FALSE) or bootstrap (replace=TRUE) from the existing data. Note that the specified group-sizes have to be smaller than the real group sizes in case of sub-sampling.

Details

MAsim simulates normal data with constant standard deviation sigma across genes and fixed effect size D; the sign of the effect is equally and randomly split between up- and down-regulation, and effects are added to the second group. MAsim.var does the same, but instead of relying on a fixed variance across genes, it simulates gene-specific variances from a standard exponential distribution.

MAsim.smyth simulates from the model suggested in Smyth (2004), using a normal error distribution. The variances are assumed to follow an inverse chisquared distribution with d0 degrees of freedom and are scaled by s2_0; consequently, large values of d0 lead to similar gene-wise variances across genes, whereas small values lead to very different variances between genes. The effect sizes for differentially expressed genes are assumed to follow a normal distribution with mean zero and variance v0 times the previously simulated gene-specific variance; consequently, large values of v0 lead to large effects in the model.

MAsim.real finally uses existing real or simulated existing data sets to generate simulated data with fixed effect sizes: for each group, the specified number of samples is sampled either with or without replacement from the columns of xdat; for each gene, the group means are subtracted from the resampled data, so that the groupwise and overall mean for each gene is zero. Then, noise from an appropriate t-distribution is added to each group to break the sum-to-zero constraint in a consistent manner. The specified effect (evenly split between up- and down-regulation) for the differentially expressed genes is again added to the second group.

Value

The functions all return a matrix with ng rows and n1+n2 columns, except for MAsim.real, where the default is to return a matrix of the same dimensions as xdat. The group membership of each column is given by its column name. The matrix has additionally the attribute DE, which is a logical vector specifying for each gene whether or not it was assumed to be differentially expressed in the simulation.

References

Smyth G (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. Statistical Applications in Genetics and Molecular Biology 3, No. 1, Article 3

See Also

EOC

Examples

```
# Small examples only
sim1 = MAsim(ng=1000, n=10, p0=0.8)
sim2 = MAsim.var(ng=1000, n1=15, n2=5, p0=0.8)
sim3 = MAsim.smyth(ng=1000, n=10, p0=0.8)

# Assess FDR
eoc1 = EOC(sim1, colnames(sim1), plot=FALSE)
eoc2 = EOC(sim2, colnames(sim2), plot=FALSE)
eoc3 = EOC(sim3, colnames(sim3), plot=FALSE)

# Show
par(mfrow=c(2,2))
plot(eoc1)
plot(eoc2)
plot(eoc3)
OCshow(eoc1, eoc2, eoc3)

# The truth will make you fret
table(eoc1$FDR<0.1, attr(sim1, "DE"))
```

OCshow — *Show one ore several OC curves graphically*

Description

Plots empirical OC curves for one or several data sets simultaneously, showing the local or global false discovery rate among the top regulated genes. This is the preferred way of comparing the OC of different analyses.

Usage

```
OCshow(x, ..., global = TRUE, percentage = TRUE, top = 0.1, legend, lty, col, ma
```

Arguments

x, ...	one or several objects created by either EOC, fdr1d, or fdr2d
global	logical value indicating whether to show the global or the local false discovery rates; note that if any of the objects to be plotted was created by EOC, only global Fdr is available.
percentage	logical switch indicating whether to show the percentage of top regulated genes or the actual numbers; note that the cutoff top is always a percentage
top	a value between 0 and 1 specifying the percentage of top-regulated genes that is to be shown in the plot
legend	a character vector giving names for each of the objects to be plotted for a legend in the left upper corner
lty, col	line styles and colors for the different OC curves

main	a plot title
xlab, ylab	axis labels

Details

Each object generated by EOC, fdr1d, and fdr2d lists for each gene a t-statistic and either a local or a global false discovery rate. The OC curves are constrcuted by ordering the genes according to the false discovery rates, and counting how many fall under a given threshold. These counts are plotted (either directly or as percentage of all genes) on the horzontal axis, while the thresholds are plotted on the vertical axis. Where appropriate, local false discovery rates are converted to global rates by simple averaging.

Author(s)

A. Ploner

See Also

EOC, fdr1d, fdr2d

Examples

```
# We simulate a small example with 5 percent regulated genes and
# a rather large effect size
set.seed(2003)
xdat = matrix(rnorm(50000), nrow=1000)
xdat[1:25, 1:25] = xdat[1:25, 1:25] - 2
xdat[26:50, 1:25] = xdat[26:50, 1:25] + 2
grp = rep(c("Sample A","Sample B"), c(25,25))

# Compute the different false discovery rates
# p0 is fixed
global = EOC(xdat, grp, plot=FALSE, p0=0.95)
local1d = fdr1d(xdat, grp, p0=0.95)
local2d = fdr2d(xdat, grp, p0=0.95)

# Some possible arrangements
leg = c("global","local1d","local2d")
par(mfrow=c(2,2))
OCshow(global, local1d, local2d, legend=leg, main="Default")
OCshow(global, local1d, local2d, legend=leg, percentage=FALSE,
       main="Number of genes")
OCshow(global, local1d, local2d, legend=leg, top=1, main="All genes")
OCshow(local1d, local2d, legend=leg[2:3], global=FALSE, main="Local fdr")
```

---

plot.fdr1d.result *Plot univariate local false discovery output*

Description

A plotting function for fdr1d.

Usage

```
plot.fdr1d.result(x, add = FALSE, grid = FALSE, rug = TRUE,
                  xlab = "t-Statistic", ylab = "fdr", lcol = "black", ...)
```

Arguments

x	output from fdr1d
add	logical value indicating whether to create a new plot or add to an existing one
grid	logical value indicating whether to show the intervals used for calculating the fdr.
rug	logical value indicating whether to add a 1D scatterplot showing the observed test statistics
xlab, ylab	the usual axis labels
lcol	the color of the lines
...	extra options passed to plot.default.

Author(s)

A Ploner

See Also

fdr1d

Examples

```
example(fdr1d)
plot(res1d, grid=TRUE, rug=FALSE)
```

---

plot.fdr2d.result *Plotting the bivariate local false discovery results*

Description

These functions provide different ways of plotting the output fdr2d.

Usage

```
plot.fdr2d.result(x, levels, nr.plot = 20, add = FALSE, grid = FALSE,
  pch = ".",xlab, ylab, vfont = c("sans serif", "plain"), lcol = "black", ...)

Tornadoplot(x, levels, nr.plot = 20, label = FALSE, constrain = FALSE,
  pch = ".", xlab, ylab, vfont = c("sans serif", "plain"), lcol = "black", ...)

Volcanoplot(x, df, levels, nr.plot = 20, label = FALSE, constrain = FALSE,
  pch = ".", xlab, ylab, vfont = c("sans serif", "plain"), lcol = "black", ...)
```

Arguments

x	an object created by fdr2d.
df	the appropriate degrees of freedom for a two-sample t-test with equal variances (in order to provide p-values for the volcano plot).
levels	vector of levels for drawing fdr isolines
nr.plot	number of equidistant breaks defining a two-dimensional grid for smoothing isolines, see Details.
add	logical value indicating whether to create a new plot, or to add to an existing plot.
grid	logical value indicating whether the original grid used for estimating the local fdr should be shown.
label	logical value indicating whether to draw labels on the isolines.
constrain	logical flag indicating whether transformed fdr values should be made monotnously decreasing with the absolute size of the first test statistic, see fdr2d.
pch, xlab, ylab	the usual graphical parameters
vfont	vector font specification for labelling the isolines, see contour.
lcol	colour used for drawing the isolines
...	extra graphical parameters passed to plot.default.

Details

The plot format is basically a scatter plot of the observed test statistics, overlayed with isolines showing the estimated fdr. The generic plot function displays the original test statistics that are used to estimate the fdr, i.e. the two-sample t-statistics and the logarithmized standard errors; the other plots use different, but mathematically equivalent test statistics:

- mean difference and logarithmized standard error for the tornado plot,
- mean difference and $-\log 10(p)$ for the volcano plot, where p is the p-value from the standard two-sample t-test.

By default, the estimated fdr isolines are smoothed and cropped to the convex hull of the observed test statistics by using **akima**. This is entirely a graphical pre-processing step which produces smoother isolines and enforces sanity at the edges of the observed distribution; it does not change the estimated fdr at all. This graphical smoothing is controlled via the argument nr.plot, which specifies the grid size, with lower values resulting in stronger smoothing. In order to suppress graphical smoothing, set nr.plot to zero.

Note that the test statistics and the fdr for the volcano- and tornado plots are not computed from scratch, but rather through transformation of the original results. Specifically, the isolines in these plots are also transformed; this has the unfortunate side effect that the labelling of isolines in these plots is not nearly as pretty as the standard provided by contour. This functionality is currently not available outside of contour, and our implementation in DrawContourlines frankly leaves a lot to be desired. We apologize for the inconvenience.

Value

The original x, invisibly.

Author(s)

A. Ploner

References

Ploner A, Calza S, Gusnanto A, Pawitan Y (2005) Multidimensional local false discovery rate for micorarray studies. *Submitted Manuscript*.

See Also

fdr2d, DrawContourlines

Examples

```
# Create res2d
example(fdr2d)

par(mfrow=c(2,2))
plot(res2d, main="Generic plotting")
Volcanoplot(res2d, df=length(grp)-2, main="Volcano plot", label=TRUE)
Tornadoplot(res2d, main="Tornado plot", label=TRUE)
# This is without graphical smoothing
plot(res2d, main="Generic plotting, raw", nr.plot=0)
```

---

plot.FDR.result *Plot the empirical FDR as a function of the cutoff level*

Description

Plots the output from EOC. The resulting graph is the empirical counterpart to those produced by TOC, i.e. the estimated FDR as a function of the cutoff-level on the t-statistic.

Usage

```
plot.FDR.result(x, add=FALSE, sensitivity.show = TRUE, legend.show = FALSE,
     xlim, ylim = c(0, 1), xlab, ylab, main, ...)
```

Arguments

x	an object created by EOC
add	logical value indicating whether to add to an existing plot or start a new one
sensitivity.show	logical value indicating whether to show the classical sensitivity for testing one hypothesis as a function of the cutoff level.
legend.show	logical value indicating whether to add a legend to the plot
xlim, ylim	limits for the horizontal and vertical axis
xlab, ylab	axis labels
main	plot title
...	the usual graphical parameters, passed to plot

Author(s)

A. Ploner

See Also

EOC

Examples

```
# We simulate a small example with 5 percent regulated genes and
# a rather large effect size
set.seed(2003)
xdat = matrix(rnorm(50000), nrow=1000)
xdat[1:25, 1:25] = xdat[1:25, 1:25] - 2
xdat[26:50, 1:25] = xdat[26:50, 1:25] + 2
grp = rep(c("Sample A","Sample B"), c(25,25))

# Compute the EOC without plotting
ret = EOC(xdat, grp, plot=FALSE)

# Some possible arrangements
par(mfrow=c(2,2))
plot(ret)
plot(ret, legend=TRUE)
plot(ret, sensitivity=FALSE)
```

---

samplesize *FDR as a function of sample size*

Description

This function tabulates the false discovery rate (FDR) for selecting differentially expressed genes as a function of sample size and cutoff level. Additionally, the same information can be displayed through an attractive plot.

Usage

```
samplesize(n = seq(5, 50, by = 5), p0 = 0.99, sigma = 1, D, F0, F1,
           paired = FALSE, crit, crit.style = c("top percentage", "cutoff"),
   plot =TRUE, local.show=FALSE, nplot = 100, ylim = c(0, 1), main,
   legend.show = FALSE, grid.show = FALSE, ...)
```

Arguments

n	sample size (as subjects per group)
p0	the proportion of non-differentially expressed genes
sigma	the standard deviation for the log expression values
D	assumed average log fold change (in units of sigma), by default 1; this is a shortcut for specifying a simple symmetrical alternative hypothesis through F1.

F0	the distribution of the log2 expression values under the null hypothesis; by default, this is normal with mean zero and standard deviation sigma, but mixtures of normals can be specified, see Details and Examples.
F1	the distribution of the log2 expression values under the alternative hypothesis; by default, this is an equal mixture of two normals with means D and -D and standard deviation sigma; mixture of normals are again possible, see Details and Examples.
paired	logical value indicating whether this is the independent sample case (default) or the paired sample case.
crit	a vector of cutoff values for selecting differentially expressed genes; the interpretation depends on crit.style.
crit.style	indicates how differentially expressed genes are selected: either by a fixed cutoff level for the absolute value of the t-statistic or as a fixed percentage of the absolute largest t-statistics.
plot	logical value indicating whether to do the plotting business
local.show	logical value indicating whether to show local or global false discovery rate (default: global).
nplot	number of points that are evaluated for the curves
ylim	the usual limits on the vertical axis
main	the main title of the plot
legend.show	logical value indicating whether to show a legend for the types of gene selection in the plot
grid.show	logical value indicating whether to draw grid lines showing the sample sizes n to be tabulated in the plot
...	the usual graphical parameters, passed to plot

Details

This function plots the FDR as a function of the sample size when comparing the expression of multiple genes between two groups of subjects. This is based on a model assuming that a proportion p0 of genes is not differentially expressed (regulated) between groups, and that 1-p0 genes are. The logarithmized gene expression values of regulated and non regulated genes are assumed to be generated by mixtures of normal distributions; these mixtures can be specified through the parameters F0, F1 or D, and sigma; please see TOC for details on the model and the specification of the mixtures. By default, the null distribution of the log expression values is a normal centered on zero, and the alternative an equal mixture of normals centered at +D and −D.

The list of nominally differentially expressed genes can be selected in two ways:

- all genes with absolute t-statistic larger than the specified critical cutoff values (cutoff),
- all genes that represent the specified critical top percentage of the absolutely largest t-statistics (top percentage).

  Multiple critical values correspond to multiple curves, each labeled by the critical value, but only one value can be specified for the proportion of non-regulated genes p0 and the standard deviation sigma.

Value

A matrix with rows corresponding to elements of n and columns corresponding to the specified critical values is returned. The matrix has the attribute param that contains the specified arguments, see Examples.

Note

Both the curve labels and the legend may be squashed if the plotting device is too small. Increasing the size of the device and re-plotting should improve readability.

Author(s)

Y. Pawitan and A. Ploner

References

Pawitan Y, Michiels S, Koscielny S, Gusnanto A, Ploner A (2005) False Discovery Rate, Sensitivity and Sample Size for Microarray Studies. *Bioinformatics*, 21, 3017-3024.

Jung SH (2005) Sample size for FDR-control in microarray data analysis. *Bioinformatics*, 21, 3097-104.

See Also

FDR, TOC, EOC

Examples

```
# Default assumes a proportion of 0.01 regulated genes equally split
# between two-fold up- and down-regulated
# We select the top 1, 2, 3 percent absolute largest t-statistics
samplesize(crit=c(0.03,0.02, 0.01))

# Same model, but using a hard cutoff for the t-statistics
samplesize(crit=2:4, crit.style="cutoff")

# Paired test of the same size has slightly better FDR (as expected)
samplesize(paired=TRUE)

# Compare the effect of p0 and effect size
par(mfrow=c(2,2))
samplesize(crit=c(0.03,0.02, 0.01), p0=0.95, D=1)
samplesize(crit=c(0.03,0.02, 0.01), p0=0.99, D=1)
samplesize(crit=c(0.03,0.02, 0.01), p0=0.95, D=2)
samplesize(crit=c(0.03,0.02, 0.01), p0=0.99, D=2)

# An asymmetric alternative distribution: 20 percent of the regulated genes
# are expected to be (at least) four-fold up regulated
# NB, no graphical output
ret = samplesize(F1=list(D=c(-1,1,2), p=c(2,2,1)), p0=0.95, crit=0.05, plot=FALSE)
ret
# Look at the parameters
attr(ret, "param")

# A wide null distribution that allows to disregard genes with small effect
# Here: |log2 fold change| < 0.25, i.e. fold change of less than 19 percent
samplesize(F0=list(D=c(-0.25,0,0.25)), grid=TRUE)

# This is close to Example 3 in Jung's paper (see References):
# p0=0.99 and sensitivity=0.6, so we want a rejection rate of
# around 0.006 from the top list.
# Here we require around 40 arrays/group, compared to
```

```
# around 37 in Jung's paper, most likely because we use
# the t-distribution instead of normal. Jung's alternative
# is only one-sided, so the exact correspondence is
#
samplesize(p0=0.99,crit.style="top", crit=0.006, F1=list(D=1, p=1), grid=TRUE)
abline(h=0.01)

#The result is very close to the symmetric alternatives:
samplesize(p0=0.99,crit=0.006, D=1, grid=TRUE, ylim=c(0,0.9))
```

`smooth1d` *Smoothing a vector of counts*

Description

This function takes a vector of counts and uses a mixed model approach to smooth it. A common use of this is smoothing binned counts of an observed quantity prior to estimating its density nonparametrically through the relative frequencies.

Usage

```
smooth1d(y, sv2 = 0.1, err = 0.01, verb = TRUE)
```

Arguments

`y`	the vector of counts
`sv2`	the user-specified starting value for the variance of the random effects, see Details.
`err`	Tolerance for convergence, see Details
`verb`	logical value indicating whether to print diagnostics.

Details

The smoothing assumes that the counts are Poisson from a generalized linear mixed model, where the second differences are normally distributed. Using the extended likelihood approach described in Pawitan (2001) and the initial estimate `sv2` for the variance of the random effects, the routine iteratetively optimizes the fixed and random contributions to the extended likelihood, until the estimate for the variance convergences with tolerance `err`. The result is quite stable within a reasonable range of starting values and tolerances, and the function can be used for fairly automatic smoothing ((i.e. withou fixing a bandwidth parameter).

Value

A list with three components:

`fit`	the smoothed counts
`df`	the degrees of freedom used for smoothing at convergence
`sv2`	the estimated variance at convergence, equivalent to `df`.

Author(s)

Y. Pawitan and A. Ploner

References

Pawitan Y.(2001) *In All Likelihood*, Oxford University Press, ch. 18.11

See Also

fdr1d

Examples

```
# Stupid dummies, obviously
smooth1d(1:10)
smooth1d(1:10, sv2=1)
```

---

summary.fdr.result *Display functions for local fdr output*

Description

Display functions for output from fdr1d and fdr2d, summarizing the output, displaying the proportion of non-differentially expressed genes and extracting the list of top-regulated genes.

Usage

```
summary.fdr.result(object, ...)

p0(x, how = FALSE)

topDE(x, co = 0.1)
```

Arguments

x, object	an object of class fdr.result created by fdr1d or fdr2d.
how	a logical value indicating whether to return only the numerical value of the proportion of non-differentially expressed genes, or a list whose second element indicates whether the proportion was estimated from the data or supplied by the user.
...	extra arguments, currently unused
co	cutoff for either FDR or fdr

Value

For summary.fdr.result, a list with the summary items.

For p0, either a numerical value or a list with two elements, depending on the value of parameter how.

For topDE, the genes that have FDR (EOC) or fdr (fdr1d, fdr2d) less or equal than co, sorted by FDR or fdr respectively.

Author(s)

A. Ploner

See Also

fdr1d, fdr2d, EOC

Examples

```
# Create object res1d
example(fdr1d)

summary(res1d)
p0(res1d)
p0(res1d, how=TRUE)
topDE(res1d)
```

---

tMixture *Fit a mixture of t-distributions*

Description

For a vector of individual genewise t-statistics, this functions fits a distribution of central and non-central t-distributions, with the primary goal of estimating the proportion `p0` of non-differentially expressed genes.

Usage

```
tMixture(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 inifinity.
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). tMixture will in general make a valiant effort to deduce valid starting values from any combination of nq, p0, p1, D, and delta specified by the user, and will complain if that is not possible.

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

Value

A list with the following components:

p0.est	the estimated proportion of non-differentially expressed genes, after collapsing components with estimated non-centrality sizes below threshold.delta.
p0.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.
AIC	the AIC value for the maximum likelihood fit.
opt	The output from optim, giving details about the optimization process.

Author(s)

Y. Pawitan and A. Ploner

References

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

See Also

tstatistics, EOC, optim

Examples

```
# We simulate a small example with 5 percent regulated genes and
# a rather large effect size
set.seed(2011)
xdat = matrix(rnorm(50000), nrow=1000)
xdat[1:25, 1:25] = xdat[1:25, 1:25] - 2
xdat[26:50, 1:25] = xdat[26:50, 1:25] + 2
grp = rep(c("Sample A","Sample B"), c(25,25))
# Use a helper function for the test statistics
myt = tstatistics(xdat, grp)$tstat
r1 = tMixture(myt, n1=25, nq=3)
r1

# Equivalently, we could have specified the same set of starting values
# as follows:
# r1 = tMixture(myt, n1=25, p0=1/3, p1=c(1/3, 1/3), delta=c(-1,1))

# Alternative starting value for p0, other starting values are filled in
r2 = tMixture(myt, n1=25, nq=3, p0=0.80)
r2

# Specification of too many components usually leads to spurious
# noncentral components like here - note the difference between
# p0.est and p0.raw!
r3 = tMixture(myt, n1=25, nq=5)
r3

# We simulate a data in a paired setting
# Note the arrangement of the columns
set.seed(2012)
xdat = matrix(rnorm(50000), nrow=1000)
ndx1 = seq(1,50, by=2)
xdat[1:25, ndx1] = xdat[1:25, ndx1] - 2
xdat[26:50, ndx1] = xdat[26:50, ndx1] + 2
grp = rep(c("Sample A","Sample B"), 25)
# Use a helper function for the test statistics
myt = tstatistics(xdat, grp, paired=TRUE)$tstat
r1p = tMixture(myt, n1=25, nq=3)
r1p
```

Description

Computes and plots the operating characteristics for a two group microarray experiment based on a theoretical model. The false discovery rate (FDR) is plotted against the cutoff level on the t-statistic. Optionally, curves for the the classical significance level and sensitivity can be added. Different curves for different proportions of non-differentially expressed genes can be compared in the same plot, and the sample size per group can be varied between plots.

Usage

```
TOC(n = 10, p0 = 0.95, sigma = 1, D, F0, F1, n1 = n, n2 = n, paired = FALSE,
    plot = TRUE, local.show=FALSE, alpha.show = TRUE, sensitivity.show = TRUE,
nplot = 100, xlim, ylim = c(0, 1), main, legend.show = FALSE, ...)
```

Arguments

`n, n1, n2`	number of samples per group, by default equal and specified via `n`, but can be set to different values via `n1` and `n2`.
`p0`	the proportion of not differentially expressed genes, may be vector valued
`sigma`	the standard deviation for the log expression values
`D`	assumed average log fold change (in units of `sigma`), by default 1; this is a shortcut for specifying a simple symmetrical alternative hypothesis through `F1`.
`F0`	the distribution of the log2 expression values under the null hypothesis; by default, this is normal with mean zero and standard deviation `sigma`, but mixtures of normals can be specified, see Details and Examples.
`F1`	the distribution of the log2 expression values under the alternative hypothesis; by default, this is an equal mixture of two normals with means `D` and `-D` and standard deviation `sigma`; mixture of normals are again possible, see Details and Examples.
`paired`	logical value indicating whether two distinct groups of observations or one group of paired observations are studied.
`plot`	logical value indicating whether the results should be plotted.
`local.show`	logical value indicating whether to show local or global false discovery rate (default: global).
`alpha.show`	logical value indicating whether to show the classical significance level for testing one hypothesis as a function of the cutoff level.
`sensitivity.show`	logical value indicating whether to show the classical sensitivity for testing one hypothesis as a function of the cutoff level.
`nplot`	number of points that are evaluated for the curves
`xlim`	the usual limits on the horizontal axis
`ylim`	the usual limits on the vertical axis
`main`	the main title of the plot
`legend.show`	logical value indicating whether to show a legend for the different types of curves in the plot.
`...`	the usual graphical parameters, passed to `plot`

Details

This function plots the FDR as a function of the cutoff level when comparing the expression of multiple genes between two groups of subjects. We study a gene selection mechanism that declares all genes to be differentially expressed whose t-statistics have an absolute value greater than a specified cutoff value. The comparison is based on a two-sample t-statistic for equal variances, for either paired or unpaired observations.

The underlying model assumes that a proportion `p0` of genes are not differentially expressed between groups, and that 1-`p0` are. The logarithmized gene expression values are assumed to be generated by mixtures of normal distributions. Both null and alternative hypothesis are specified through the means of the respective mixture components; these means can be interpreted as average log2 fold changes in units of the standard deviation `sigma`.

Note that the model does *not* assume that all genes have the same standard deviation `sigma`, only that the mean log2 fold change for all regulated genes is proportional to their individual variability (standard deviation). `sigma` generally does not need to be specified explicitly and can be left at its default value of one, so that `D` can be interpreted straightforward as log2 fold change between groups.

The default null distribution of the log2 expression values is a single normal distribution with mean zero (and standard deviation `sigma`); the default alternative distribution is is an equal mixture of two normals with means `D` and `-D` (and again standard deviation `sigma`). However, general mixtures of normals can be specified for both null and alternative distribution through `F0` and `F1`, respectively: both are lists with two elements:

- `D` is the vector of means (i.e. log2 fold changes),
- `p` is the vector of mixing proportions for the means.

If present, `p` must be the same length as `D`; its elements do not need to be normalized, i.e. sum to one; if absent, equal mixing is assumed, see Examples. A wide (mixture) null hypothesis, or an empirical null hypothesis as outlined by Efron (2004), can be used if genes with log fold changes close to zero are thought to be of no biological interest, and are counted as effectively not regulated. Similarly, the alternative hypothesis can be any mixture of large and small effects, symmetric or non-symmetric, depending on the expected regulation patterns, see Examples.

As a consequence, both the null distribution of the t-statistics (for the unregulated genes) and their alternative distribution (for the regulated genes) are mixtures of (generally non-central) t-distributions, see `FDR`.

Sample size `n` and standard deviation `sigma` are atomic values, but multiple `p0` can be specified, resulting in multiple curves. Additionally, the usual significance level and sensitivity for a classical one-hypothesis can be displayed.

Value

This function returns invisibly a data frame with `nplot` rows whose columns contain the information for the individual curves. The number of columns and their names will depend on the number and value of the `p0` specified, and whether alpha and sensitivity are displayed. Additionally, the returned data frame has an attribute `param`, which is a list with all the non-plotting arguments to the function.

Note

Both the curve labels and the legend may be squashed if the plotting device is too small. Increasing the size of the device and re-plotting should improve readability.

Author(s)

Y. Pawitan and A. Ploner

References

Pawitan Y, Michiels S, Koscielny S, Gusnanto A, Ploner A. (2005) False Discovery Rate, Sensitivity and Sample Size for Microarray Studies. *Bioinformatics*, 21, 3017-3024.

Efron, B. (2004) Large-Scale Simultaneous Hypothesis Testing: The Choice of a Null Hypothesis. *JASA*, 99, 96-104.

See Also

FDR, samplesize, EOC

Examples

```
# Default null and alternative distributions, assuming different proportions
# of regulated genes
TOC(p0=c(0.90, 0.95, 0.99), legend.show=TRUE)

# The effect of sample size and effect size
par(mfrow=c(2,2))
TOC(p0=c(0.90, 0.95, 0.99), n=5, D=1)
TOC(p0=c(0.90, 0.95, 0.99), n=30, D=1)
TOC(p0=c(0.90, 0.95, 0.99), n=5, D=2)
TOC(p0=c(0.90, 0.95, 0.99), n=30, D=2)

# A wide null distribution that allows to disregard genes of small effect
# unspecified p means equal mixing proportions
ret = TOC(F0=list(D=c(-0.25,0,0.25)), main="Wide F0")
attr(ret,"param")$F0 # the null hypothesis

# An extended (and unsymmetric) alternative
ret = TOC(F1=list(D=c(-2,-1,1), p=c(1,2,2)), p0=0.95, main="Unsymmetric F1")
attr(ret,"param")$F1 # F1$p is normalized

# Unequal sample sizes
TOC(n1=10, n2=30)

# Curves for a paired t-test
TOC(paired=TRUE)

# The output contains all the x- and y-coordinates
ret = TOC(p0=c(0.90, 0.95, 0.99), main="Default settings")
dim(ret)
colnames(ret)
ret[1:10,]
# Additionally, the list of arguments that determine the experiment
attr(ret,"param")
```

`tstatistics` *Compute multiple parallel t-statistics*

Description

`tstatistics` computes either two-sample or paired t-statistics for a bunch of variables measured on the same objects, e.g. genewise t-statistics for a microarray experiment. `PermNull` uses `tstatistics` to generate a permutation distribution.

Usage

```
tstatistics(xdat, grp, logse = FALSE, paired = FALSE)

PermNull(xdat, grp, nperm = 100, seed = NULL, logse = FALSE, paired=FALSE)
```

Arguments

`xdat`	the matrix of expression values, with genes (or variables) as rows and samples as columns.
`grp`	a grouping variable giving the class membership of each sample, i.e. each column in `xdat`, see Details.
`nperm`	number of permutations for establishing the null distribution of the t-statistic
`seed`	random number generator seed for initializing the permutations from a known starting point.
`logse`	logical flag indicating whether to return the logarithmized standard errors, too.
`paired`	indicates whether to use two-sample or paired t-statistic.

Details

`tstatistics` is a fairly fast replacement for function `mt.teststat` in package **multtest**, which is written exlusively in R and does not require loading half the Bioconductor infrastructure packages before doing anything. As such, it is used for computing the default test statistics by `fdr1d` and `fdr2d`.

Note that for the paired test, `tstatistics` requires the same data structure as `mt.teststat`: columns belonging to the same pair must be consecutive (though not necessarily in the same order throughout, as '`grp`' will indicate the order). The function checks for this and barfs if it does not hold.

`PermNull` returns the t-statistics and optionally the logarithmized standard errors of the mean for a specified number of permutations.

Both functions are not especially economic in using memory, and collecting the whole set of permutations like `PermNull` does instead of binning and counting them directly as they come is inherently wasteful.

Value

A data frame with first column `tstat` and optionally (if `logse=TRUE`) a second column `logse`. `tstat` returns the same number of test statistics as rows in `xdat` and in the same order, `PermNull` does the same for consecutive permuations of the grouping variable `grp`.

If the argument `seed` is specified, `PermNull` adds an attribute of the same name to the returned data frame.

Author(s)

A. Ploner

See Also

fdr1d, fdr2d, examples in tMixture

Index