Package 'dclong.spt'

February 24, 2012

Type Package

Title Sequential Permutation Test

Version 1.1

Date 2012-02-06

Author Chuanlong Benjamin Du

Maintainer Chuanlong Benjamin Du <duchuanlong@gmail.com>

Description Permutation tests are commonly used. Usually it doesn't take much time to do a permutation test, but it can be time-consuming if you have to do many permution tests at the same time. This package allows you to do sequential permutation tests which save you time. Currently only sequential permutation test for no difference between two groups and sequential permutation test for no correlation between a reponse variable and a bunch of covariates are supported. Generally sequential permutation test (and parallel technic) will be supported soon.

Depends R (>= 2.10)

License GPL (>=2)

Collate 'spt.corr.r' 'spt.mean.r' 'zzz.r' 'est-m0.r' 'fdr.r''rbunif.R' 'barley.r' 'leukemia.r' 'marker.r' 'plot.spt.r'

R topics documented:

barley 2
fdr 2
leukemia 3
m0.storey 3
marker 5
plot.spt 5
rbunif 6
spt.corr 7
spt.mean 8

Index **10**

`barley` *Barley Gene Expression Data*

Description

The data produced by experiment "Genetic regulation of gene expression of barley in response to stem rust (Pgt isolate TTKS) and can be access from PLEXdb (BB64). There is a file called "BB64_RMA_tmt_medians.txt" on the download page contains RMA expressions. The rma expression for the 75 chips involve fungas infection is the dataset barley in this package. Note that this data set is used for illustration in Bancroft et al. (2012).

Format

The data set "barley" is a matrix with 22841 rows and 75 columns.

Source

http://www.plexdb.org/modules/PD_browse/experiment_browser.php.

References

Tim Bancroft, Chuanlong Du and Dan Nettleton (2012) Estimation of False Discovery Rate Using Sequential Permutation P -Values.

`fdr` *False Discover Rate (FDR)*

Description

Function `fdr` calculate FDR for specified cut-offs. If the pvalues is specified as the cutoffs, then you get the qvalues.

Usage

```
fdr(cutoff, p, m0.hat, delta)
```

Arguments

`cutoff`	a vector of cut-offs. For a given cut-off, the null hypotheses with pvalues less or equal to the cut-off are rejected.
`p`	the vector of pvalues.
`m0.hat`	an estimate of the number of true null hypotheses.

delta — the error tolerance in comparing pvalues with cut-offs. To specify an appropriate error tolerance is important if there are lots of pvalues that are very close to a cut-off. An extreme but happen-often (e.g. when calculating qvalues) case is that a cut-off is inside the support of sequential pvalues. A value no greater than `h/(100*n^2)` is recommended. Please refer to `spt.corr` for documentation of `h` and `n`.

---

leukemia — *Acute Lymphoma Leukemia Data*

Description

The Band T-cell Acute Lymphocyctic Leukemia (ALL) data set can be access via the Bioconductor ALL package at www.bioconductor.org. Measures of messenger ribonucleic acid (mRNA)–commonly referred to as expression levels–are available for 12,625 probesets in 128 ALL patients. Of these 128 patients, we focus on the 21 males who have been classified as having a translocation between chromosomes 9 and 22 (BCR/ABL) and the 5 males who have a translocation between chromosomes 4 and 11 (ALL1/AF4). This subset of data is the `leukemia` dataset in this package. Note that this dataset is used for illustrations in Bancroft et al. (2012).

Format

The data set `leukemia` is a matrix with the first 5 columns being gene expression for the 5 males who have a translocation between chromosome 4 and 22 (ALL1/AF4) and the last 21 columns being gene expression for the 21 males who have a translocation between chromosomes 9 and 22 (BCR/ABL).

References

Tim Bancroft, Chuanlong Du and Dan Nettleton (2012) Estimation of False Discovery Rate Using Sequential Permutation P -Values.

---

m0.storey — *Estimate the Number of True Null Hypotheses*

Description

Estimating the number of true null hypotheses (m0) is critical for estiamting the false discover rate (FDR). The functions listed here offers different methods for estimating m0 for both regular pvalues and sequential permutation test pvalues.

Usage

```
m0.storey(p, lambda = seq(0, 0.95, 0.05))

m0.nettleton(p, bins = 20, control)
```

Arguments

p	a vector of pvalues.
lambda	points chosen for fitting spline which can be considered as tuning parameters in estimating m0. These points must be in [0,1] and a defualt value seq(0,0.95,0.05) is used. For more information, please see Storey and Tibshirani's (PNAS, 2003).
bins	the number of bins in Nettleton's histogram based method for estimating m0 (see Nettleton et al. (2006) JABES 11, 337-356 and Bancroft et al. (2012)).
control	either NULL or a list. If control is NULL, then Nettleton's method for uniformlly distribution pvalues under null hypothesis is used. If control is a list (must contain at least two variables h which is the number of significant test statistics to hit before early termination in sequential permutation test) and n which is the number of permutations sampled), Nettleton's method for non-uniformly pvalues is used. You can include an extra variable delta in list control which is the error tolerance in counting pvalues. A default value h/(10*n^2) is used.

Value

the estimate of the number of true null hypotheses.

References

Storey JD and Tibshirani R. (2003) Statistical significance for genome-wide studies.

Dan Nettleton, J. T. Gene Hwang, Rico A. Caldo and Roger P. Wise (2006) Estimating the Number of True Null Hypotheses from a Histogram of p Values.

Tim Bancroft, Chuanlong Du and Dan Nettleton (2012) Estimation of False Discovery Rate Using Sequential Permutation P -Values.

See Also

fdr for estimating false discover rate and calculating qvalues.

Examples

```
## Not run:
  #simulate p vlaues
  p = rbunif(10000,beta=29)
  #estimate m0 using Storey's method
  m0.storey(p)
  #estimate m0 using Nettleton's method for regular pvalues
  m0.nettleton(p)
  #load data
  data(leukemia)
  #sequential permutation pvalues
  spt.mean(leukemia,5,10,1000)[,1] -> p
  #estimate m0 using Nettleton's method for sequential
  #permutation test pvalues
  m0.nettleton(p,control=list(h=10,n=1000))

## End(Not run)
```

marker *Barley Marker*

Description

Biologists genetically mutated/changed the genotypes of barley. They could not change everywhere, so they changed 378 positions on the chromosome of barley. In the map, "A" and "B" are two types (sort of open and close). Because they know where the mutations are, they called them "markers" (so that if a barley with a certain genotype has a higher expression level, then you may infer and say, oh that may be caused by the 145th marker, etc.). The map has 7 chromosomes of barley, 1H, 2H, ..., 7H. These numbers are locations of markers on the chromosomes, like coordinates. There are some missing values in the original map, a naive method was used to interpolate the missing values and produced this dataset "barley". Note that this data set is used for illustration in Bancoft et al. (2012).

Format

The dataset "barley" is a matrix with 22841 rows and 75 columns.

Source

http://www.plexdb.org/modules/PD_browse/experiment_browser.php.

References

Tim Bancroft, Chuanlong Du and Dan Nettleton (2012) Estimation of False Discovery Rate Using Sequential Permutation P -Values.

plot.spt *Plot Sequential Permutation Test Pvalues*

Description

The function `plot.spt` plot the observed sequential permutation pvalues together with the theoretical ones when all null hypotheses are true, so that you can have a general idea about whether there are lots of significant tests. It's analogous to the usual histogram of regular pvalues.

Usage

```
  ## S3 method for class 'spt'
plot(x,plim=1,...)
```

Arguments

`x`	on object of class "spt".
`plim`	the upper limit of pvalues to display.
`...`	some extra parameters than can be passed to function `plot`.

Details

If the plot shows a lack of big pvalues but overabundance of small pvalues, one expect there to be some siginificant tests.

Author(s)

Dan Nettleton and Chuanlong Du.

Examples

```
## Not run:
   #load data
   data(leukemia)
   spt.mean(leukemia,5,10,1000) -> spt.mean.out
   plot.spt(spt.mean.out,col="red")

## End(Not run)
```

---

rbunif *Mixture of Beta and Uniform Distribution*

Description

Generate random observations from a mixture of beta and uniform distribution. It's primarily used for demostrating use of other functions in this package.

Usage

```
rbunif(n, alpha, beta, gamma)
```

Arguments

n	number of observations to generate.
alpha	the first parameter of beta distribution.
beta	the second parameter of beta distribution.
gamma	probability of a observation coming from uniform distribution.

Details

The mixture distribution is gamma*U(0,1) + (1-gamma)*Beta(alpha,beta).

Examples

```
## Not run:
   rbunif(100,alpha=1,beta=29,gamma=0.7)

## End(Not run)
```

`spt.corr` *Sequential Permutation Test*

Description

Performs (multiple) sequential permutation tests for no correlations between response variables and covariates.

Usage

```
spt.corr(x, y, h, n)
```

Arguments

`x`	a matrix/vector with columns containing covariates.
`y`	a matrix/vector with columns containing response variables.
`h`	the number of significant test statistics to hit before early termination.
`n`	the maximum number of permutations to use, including the observed one, i.e., at most n - 1 permutations will be sampled.

Details

For each response varaible `y0` (a column in `y`), a sequential permutation test is done for H0: there's correlation between `y0` and covariates (columns in `x`) VS Ha: no correlation. The maximum absolute correlation is the test statistic used.

Value

an object of `class` "spt". An object of class "spt" is a list containing at least the following components:

`p`	pvalue of sequential permutation test for each gene/test.
`h`	see description in the Arguments section.
`n`	see description in the Arguments section.

The object return by function `spt.corr` also contains the following component(s):

`max.ac`	the maximum absolute correlation (which is the test statistic) for each gene.
`max.index`	the index of the covariate where the maximum absolute correlations is achieved for each gene/test.

Note that though `spt.corr` performs sequential permutation test, you can get results for regular permutation test by setting `h >= n`.

See Also

`spt.mean` sequential permutation tests for no difference between two treatment groups.

Examples

```
## Not run:
#load data
data(marker)
data(barley)
#sequntial permutation test for no correlation between gene expression
#and the markers (it might take a while)
spt.corr(t(marker),t(barley),10,1000)-> spt.corr.out
head(spt.corr.out)

## End(Not run)
```

`spt.mean` *Sequential Permutation Test*

Description

Performs (multiple) sequential permutation tests for no difference between two treatment groups.

Usage

```
spt.mean(data, size1, h, n)
```

Arguments

`data`	a matrix with each row being a data set for a test (gene).
`size1`	the size of the first group.
`h`	the number of significant test statistics to hit before early termination.
`n`	the maximum number of permutations to use, including the observed one, i.e., at most n - 1 permutations will be sampled.

Details

For each data set (a row in `data`) containing two groups, a sequential permutation test is done for H0: there's difference between the two groups VS no difference. The absolute difference between the means of the two groups is the test statistic used.

Value

an object of `class` "spt". An object of class "spt" is a list containing at least the following components:

`p`	pvalues of sequential permutation tests.
`h`	see description in the Arguments section.
`n`	see description in the Arguments section.

The object return by function `spt.corr` also contains the following component(s):

`n.perms` number of sequential permutations sampled for each test.

Note that though `spt.mean` performs sequential permutation tests, you can get results for regular permutation tests by setting `h >= n`.

See Also

`spt.corr` for sequential permutations test for no correlation between a response variable and a bunch of covariates.

Examples

```
## Not run:
   #load data
   data(leukemia)
   spt.mean(leukemia,5,10,1000) -> spt.mean.out
   head(spt.mean.out)

## End(Not run)
```

Index

∗Topic **datasets**
barley, 2
leukemia, 3
marker, 5
∗Topic **dataset**
barley, 2
∗Topic **data**
barley, 2
leukemia, 3
marker, 5
∗Topic **set**
barley, 2
leukemia, 3
marker, 5

barley, 2

class, *7, 8*

fdr, 2, *4*

leukemia, 3

m0.nettleton *(*m0.storey*)*, 3
m0.storey, 3
marker, 5

plot.spt, 5

rbunif, 6

spt.corr, *3*, 7, *9*
spt.mean, *7*, 8