Package ‘rOCEAN’

November 12, 2024

Type Package

Title Two-Way Feature Set Testing for Multi-Omics

Version 1.0

Maintainer Mitra Ebrahimpoor <mitra.ebrahimpoor@gmail.com>

Description For any two way feature-set from a pair of pre-processed omics data, 3 different true discovery proportions (TDP), namely pairwise-TDP, column-TDP and row-TDP are calculated. Due to embedded closed testing procedure, the choice of feature-sets can be changed infinite times and even after seeing the data without any change in type I error rate. For more details refer to Ebrahimpoor et al., (2024) <doi:10.48550/arXiv.2410.19523>.

License GPL (>= 2)

Date 2024-10-31

Encoding UTF-8

RoxygenNote 7.3.2

Imports ff

NeedsCompilation no

Author Mitra Ebrahimpoor [aut, cre] (<https://orcid.org/0000-0002-2299-876X>)

Repository CRAN

Date/Publication 2024-11-12 14:10:02 UTC

Contents

corPs *Calculate pairwise p-value*

Description

Calculates pairwise matrix of p-values based on Pearson's correlation test for two matrices. To gain speed and manage RAM usage, the matrices are split into several smaller chunks.

Usage

```
corPs(pm1, pm2, type = c("Mat", "Vec"), pthresh = 0.05)
```

Arguments

pm1, pm2	Subsets of two omics data sets where rows are the features and columns are samples. The rows of the two matrices would define the two-way feature set of interest.
type	Two options are available. Mat: Calculate the correlation of subsets and return a matrix; Vec: calculate the correlation matrix, subset by the given threshold and return a vector of p-values.
pthresh	Only relevant for type="Vec". The threshold by which the p-values are filtered (p>pthresh is removed). Default value is 0.05.

Value

Either a matrix or vector of pairwise p-values, as indicated by `type` parameter.

Examples

```
#number of subjects
n<-30
#number of features from omic1 in pathway
n_rows<-20
#number of features from omic2 in pathway
n_cols<-30

#random datasets
set.seed(1258)
pm1<-matrix(runif(n_rows*n, min=0, max=1)^8, nrow=n_rows, ncol=n)
pm2<-matrix(runif(n_rows*n, min=0, max=1)^8, nrow=n_rows, ncol=n)

#calculate correlation matrix
pmat<-corPs(pm1, pm2, type="Mat")

pmat
```

getCat *Calculate cumulative p-categories*

Description

Calculates cumulative p-categories for a given matrix of p-values.

Usage

```
getCat(mps, gCT, scale = c("col", "row"))
```

Arguments

mps	Matrix of p-values, representing pairwise associations between two feature sets.
gCT	Parameters of the global closed testing, which is the output of simesCT function.
scale	Scale of the quantification, a character string. Possible choices are "col" and "row".

Value

Matrix of p-categories.

See Also

simesCT

Examples

```
#number of features per omic data set
n_cols<-100
n_rows<-120

#random matrix of p-values
set.seed(1258)
pvalmat<-matrix(runif(n_rows*n_cols, min=0, max=1)^5, nrow=n_rows, ncol=n_cols)

#calculate CT parameters
gCT<-simesCT(mps=pvalmat, m=nrow(pvalmat)*ncol(pvalmat))

#define the two-way feature set
subpmat<-pvalmat[61:75,81:100]

#calculate p-categories matrix for feature set by rows
rCat<-getCat(mps=subpmat, gCT, scale="row")

#calculate p-categories matrix for feature set by columns
cCat<-getCat(mps=subpmat, gCT, scale="col")
```

ocean *OCEAN algorithm*

Description

Calculates heuristic and lower bound for the true discovery proportion (TDP) in 3 scales for a specified two-way feature set (Algorithm 1 in the reference). The input is either two omics data sub-matrices or the pre-calculated matrix of p-values for pairwise associations. In case the result is not exact, the function adopts branch and bound (Algorithm 2 in the reference), if `nMax` allows.

Usage

```
ocean(
  pm1,
  pm2,
  gCT,
  scale = c("pair", "row", "col"),
  mps,
  nMax = 100,
  verbose = TRUE
)
```

Arguments

`pm1, pm2`	Matrix; Subsets of two omics data sets where rows are the features and columns are samples. The rows of the two matrices would define the two-way feature set of interest.
`gCT`	Vector; Parameters of the global closed testing, output of simesCT function.
`scale`	Optional character vector; It specifies the scale of TDP quantification. Possible choices are "pair" (pair-TDP), "col" (col-TDP) and "row" (for row-TDP'). If not specified, all three scales are returned.
`mps`	Optional matrix of p-values; A sub-matrix of pairwise associations, representing the two-way feature set of interest. If provided, `pm1` and `pm2` are not required. If not provided, matrix of pairwise associations will be derived from `pm1` and `pm2` based on Pearson's correlation.
`nMax`	Maximum number of steps for branch and bound algorithm, if set to 1 branch and bound is skipped even if the result is not exact. The default value is a 100. The algorithm may stop before the `nMax` is reached if it converges sooner.
`verbose`	Logical; if `TRUE`, progress messages will be displayed during the function's execution. Default is `TRUE`.

Value

TDP is returned for the specified scales, along with number of steps taken and convergence status for branch and bound algorithm.

See Also

simesCT pairTDP runbab

Examples

```
#number of features per omic data set
n_cols<-100
n_rows<-120

#random matrix of p-values
set.seed(1258)
pvalmat<-matrix(runif(n_rows*n_cols, min=0, max=1)^6, nrow=n_rows, ncol=n_cols)

#calculate CT parameters
gCT<-simesCT(mps=pvalmat, m=nrow(pvalmat)*ncol(pvalmat))

#calculate TDPs for a random feature set
subpmat<-pvalmat[1:40,10:75]

out<-ocean(mps=subpmat, gCT=gCT, nMax=2)
out
```

pairTDP *pairwise true discoveries proportion*

Description

Calculates the TDP over pairs; based on SEA algorithm

Usage

```
pairTDP(mps, n, gCT)
```

Arguments

`mps` Matrix or vector of pairwise associations.

`n` Number of pairs; may not be the size of p if a threshold is used to remove large p-values.

`gCT` Parameters of the global closed testing, output of simesCT function.

Value

Proportion of true discoveries out of n pairs of features.

See Also

SEA, simesCT

Examples

```
#number of features per omic data set
n_cols<-100
n_rows<-120

#random matrix of p-values
set.seed(1258)
pvalmat<-matrix(runif(n_rows*n_cols, min=0, max=1)^5, nrow=n_rows, ncol=n_cols)

#calculate CT parameters
gCT<-simesCT(mps=pvalmat, m=nrow(pvalmat)*ncol(pvalmat))

#define the two-way feature set
subpmat<-pvalmat[61:80,26:50]

#calculate pairwise TDP for feature set
pairTDP(subpmat, n=nrow(subpmat)*ncol(subpmat), gCT)
```

runbab *Branch and bound algorithm implementation*

Description

Performs B&B when the bound are not exact

Usage

```
runbab(sCat, ssh, ssb, nMax = 100)
```

Arguments

sCat	Category matrix, output of getCat function
ssh	current Heuristic as provided by SingleStep function
ssb	current Bound as provided by SingleStep function
nMax	Maximum number of steps for the algorithm, the algorithm may stop sooner if it converges.

Value

A list, including the heuristic and the bound for the number of true discoveries, along with number of steps taken and convergence status.

See Also

getCat singleStep

Examples

```
#number of features per omic data set
n_cols<-100
n_rows<-120

#random matrix of p-values
set.seed(1258)
pvalmat<-matrix(runif(n_rows*n_cols, min=0, max=1)^4, nrow=n_rows, ncol=n_cols)

#calculate CT parameters
gCT<-simesCT(mps=pvalmat, m=nrow(pvalmat)*ncol(pvalmat))

#define the two-way feature set
subpmat<-pvalmat[1:10,31:40]

#calculate p-categories matrix for feature set by rows
rCat<-getCat(mps=subpmat, gCT, scale=”row”)

#calculate the heuristic and bound
SSout<-singleStep(rCat)

#run branch nd bound
runbab(rCat, SSout$heuristic, SSout$bound, nMax=800)
```

simesCT *Closed testing with Simes*

Description

Calculates five parameters from closed testing with Simes local tests based on raw data. These parameter are unique per data/alpha-level combination and do not depend on feature sets. Calculation may be somewhat long depending on the size of data sets and PC configurations.

Usage

```
simesCT(om1, om2, mps, m, alpha = 0.05)
```

Arguments

`om1, om2` Two omics data sets where rows are features and columns are samples.

`mps, m` Optional, pre-calculated matrix/vector of pairwise associations and the size. To save time in calculation of parameters, a threshold such as the type I error may be applies to remove larger p-values. If a threshold is used, size of matrix and `m` will not match. `m` should always be the size of the matrix of associations (number of features in `om1` X number of features in `om2`).

`alpha` type I error rate, default value is 0.05.

Value

Vector of integers: grand H value, concentration p-value, size of concentration set z, size of the original pair-wise associations matrix and the type I error level used in calculations.

References

See more details in "Hommel's procedure in linear time" doi:10.1002/bimj.201700316.

Examples

```
#number of feature per omic data set
n_cols<-100
n_rows<-120

#random matrix of p-values
set.seed(1258)
pvalmat<-matrix(runif(n_rows*n_cols, min=0, max=1)^6, nrow=n_rows, ncol=n_cols)

#calculate CT parameters
gCT<-simesCT(mps=pvalmat, m=nrow(pvalmat)*ncol(pvalmat))
```

`singleStep` *Single step algorithm*

Description

Calculates heuristic and upper-bound for the number of true discoveries based on the Algorithm 1 introduced in paper.

Usage

```
singleStep(sCat, B)
```

Arguments

sCat	p-categories matrix, output of getCat function.
B	Optional, to identify rows to be fixed (1) or removed (0) while splitting the search space.

Value

A list of two objects, the lower bound and a heuristic for the number of true discoveries

See Also

getCat

Examples

```
#number of features per omic data set
n_cols<-100
n_rows<-120

#random matrix of p-values
set.seed(1258)
pvalmat<-matrix(runif(n_rows*n_cols, min=0, max=1)^5, nrow=n_rows, ncol=n_cols)

#calculate CT parameters
gCT<-simesCT(mps=pvalmat, m=nrow(pvalmat)*ncol(pvalmat))

#define the two-way feature set
subpmat<-pvalmat[61:75,81:100]

#calculate p-categories matrix for feature set by rows
rCat<-getCat(mps=subpmat, gCT, scale=”row”)

#get the bounds based on algorithm 1
singleStep(rCat)

#calculate p-categories matrix for feature set by columns
cCat<-getCat(mps=subpmat, gCT, scale=”col”)

#get the bounds based on algorithm 1 while removing column 1 and forcing column 2
singleStep(cCat, B=c(0,1))
```

Index