# Package 'reconsi'

April 12, 2022

**Type** Package

**Title** Resampling Collapsed Null Distributions for Simultaneous
Inference

**Version** 1.6.0

**Author** Stijn Hawinkel <stijn.hawinkel@ugent.be>

**Maintainer** Joris Meys <joris.meys@ugent.be>

**Description** Improves simultaneous inference under dependence of tests by
estimating a collapsed null distribution through resampling. Accounting for
the dependence between tests increases the power while reducing the
variability of the false discovery proportion. This dependence is common in
genomics applications, e.g. when combining flow cytometry measurements with
microbiome sequence counts.

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.0

**Imports** phyloseq, KernSmooth, reshape2, ggplot2, stats, methods,
graphics, grDevices, matrixStats

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**biocViews** Metagenomics, Microbiome, MultipleComparison, FlowCytometry

**BugReports** https://github.com/CenterForStatistics-UGent/reconsi/issues

**git_url** https://git.bioconductor.org/packages/reconsi

**git_branch** RELEASE_3_14

**git_last_commit** 353b8ef

**git_last_commit_date** 2021-10-26

**Date/Publication** 2022-04-12

# R topics documented:

| calcWeights | *Obtain weights as posterior probabilities to calculate the consensus null* |
|---|---|

## Description

Obtain weights as posterior probabilities to calculate the consensus null

## Usage

```
calcWeights(logDensPerm, fdr)
```

## Arguments

| logDensPerm | A matrix with B rows of logged density estimates of the B permutation distributions, and p columns for the p observed test statistics |
|---|---|
| fdr | A vector of local false discovery rates for the observed tests statistics of length p |

## Value

A vector of weights of length B

---

| estNormal | *Fast estimation of mean and standard deviation of a normal distrbution, optionally with weights* |
|---|---|

---

## Description

Fast estimation of mean and standard deviation of a normal distrbution, optionally with weights

## Usage

```
estNormal(y, w = NULL, p = length(y))
```

## Arguments

| | |
|---|---|
| y | vector of observations |
| w | optional weight vector |
| p | The number of features |

## Value

A vector of length 2 with mean and standard deviation

---

| estP0 | *Estimate the fraction of true null hypotheses.* |
|---|---|

---

## Description

Estimate the fraction of true null hypotheses.

## Usage

```
estP0(statObs, fitAll, z0quantRange, smooth.df)
```

## Arguments

| | |
|---|---|
| statObs | A vector of observed z-values |
| fitAll | the estimated normal null |
| z0quantRange | a number of quantiles between 0 and 0.5 |
| smooth.df | degrees of freedom for the spline smoother |

## Details

A natural spline is used over a range of intervals. Based on the qvalue::qvalue() function and Storey and Tibshirani, 2003

**Value**

The estimated null fraction, the value of the spline evaluated at the first element of z0quantRange

---

getApproxCovar *Obtain a null covariance matrix of binned test statistics*

---

**Description**

Obtain a null covariance matrix of binned test statistics

**Usage**

```
getApproxCovar(statsPerm, nBins = 82L, binEdges = c(-4.1, 4.1))
```

**Arguments**

| | |
|---|---|
| statsPerm | The pxB matrix of permutation z-values in the columns |
| nBins | an integer, the number of bins |
| binEdges | A vector of length 2 with the outer bin edges |

**Value**

The covariance matrix of binned z-values

**Note**

This is not the covariance matrix of the p test statistic, nor of the data! It is an approximate covariance matrix of binned test statistics for visualization purposes.

**Examples**

```
p = 200; n = 50; B = 5e1
x = rep(c(0,1), each = n/2)
mat = cbind(
matrix(rnorm(n*p/10, mean = 5+x),n,p/10), #DA
matrix(rnorm(n*p*9/10, mean = 5),n,p*9/10) #Non DA
)
mat = mat = mat + rnorm(n, sd = 0.3) #Introduce some dependence
fdrRes = reconsi(mat, x, B = B)
corMat = getApproxCovar(fdrRes$statsPerm)
```

---

| getFdr | *Calculate tail-area (Fdr) and local (fdr) false discovery rates, based on a certain null distribution* |
|---|---|

---

## Description

Calculate tail-area (Fdr) and local (fdr) false discovery rates, based on a certain null distribution

## Usage

```
getFdr(statObs, fitAll, fdr, zSeq, p, p0, zValsDensObs, smoothObs, ...)
```

## Arguments

| | |
|---|---|
| statObs | Vector of observed z-values |
| fitAll | The parameters of the estimated random null |
| fdr | local false discovery rate, already estimated |
| zSeq | Support of the density estimation |
| p | the number of hypotheses |
| p0 | The estimated fraction of null hypotheses |
| zValsDensObs | estimated densities of observed test statistics |
| smoothObs | A boolean, should estimated observed densities of the test statistics be used in estimating the Fdr |
| ... | more arguments, ignored |

## Value

A list with components

| | |
|---|---|
| Fdr | Tail are false discovery rate |
| fdr | Local false discovery rate |

---

| getG0 | *Obtain the consensus null* |
|---|---|

---

## Description

Obtain the consensus null

## Usage

```
getG0(
  statObs,
  statsPerm,
  z0Quant,
  gridsize,
  maxIter,
  tol,
  estP0args,
  testPargs,
  B,
  p,
  pi0
)
```

## Arguments

| | |
|---|---|
| statObs | A vector of lenght p with observed test statistics |
| statsPerm | A pxB matrix with permuation z-values |
| z0Quant | a vector of length of quantiles defining the central part R of the distribution. If a single number is supplied, then (z0quant, 1-z0quant) will be used |
| gridsize | An integer, the gridsize for the density estimation |
| maxIter | An integer, the maximum number of iterations in determining R |
| tol | The convergence tolerance. |
| estP0args | A list of arguments passed on to the estP0args() function |
| testPargs | A list of arguments passed on to quantileFun |
| B | an integer, the number of permutations |
| p | an integer, the number of hypotheses |
| pi0 | A known fraction of true null hypotheses. |

## Value

A list with following entries

| | |
|---|---|
| PermDensFits | The permutation density fits |
| zSeq | The support of the kernel for density estimation |
| zValsDensObs | The estimated densities of the observed z-values |
| convergence | A boolean, has the algorithm converged? |
| weights | Vector of length B with weights for the permutation distributions |
| fdr | Estimated local false discovery rate along the support of the kernel |
| p0 | The estimated fraction of true null hypotheses |
| iter | The number of iterations |
| fitAll | The consensus null fit |

---

| getTestStats | *A function to calculate observed and permuation z-statistics on a n-by-p matrix of observations* |
|---|---|

---

## Description

A function to calculate observed and permuation z-statistics on a n-by-p matrix of observations

## Usage

```
getTestStats(
  Y,
  center,
  test = "wilcox.test",
  x,
  B,
  argList,
  tieBreakRan,
  replace,
  scale
)
```

## Arguments

| | |
|---|---|
| Y | The nxp data matrix |
| center | a boolean, should data be centered prior to permuation |
| test | A function name, possibly user defined. See details. |
| x | A vector defining the groups. Will be coerced to factor. |
| B | an integer, the number of permuations |
| argList | A list of further arguments passed on to the test function |
| tieBreakRan | A boolean, should ties of permutation test statistics be broken randomly? If not, midranks are used |
| replace | A boolean. If FALSE, samples are permuted (resampled without replacement), if TRUE the samples are bootstrapped (resampled with replacement) |
| scale | a boolean, should data be scaled prior to resampling |

## Details

For test "wilcox.test" and "t.test", fast custom implementations are used. Other functions can be supplied but must accept a y outcome variable, a x as grouping variable, and possibly a list of other arguments. It must return all arguments needed to evaluate its quantile function if z-values are to be used.

## Value

A list with components

| | |
|---|---|
| statObs | A vector of length p of observed test statistics |
| statsPerm | A p-by-B matrix of permutation test statistics |
| resamDesign | The resampling design |

---

| getTstat | *A function to obtain a t-test statistic efficiently. For internal use only* |
|---|---|

---

## Description

A function to obtain a t-test statistic efficiently. For internal use only

## Usage

```
getTstat(y1, y2, mm, nn)
```

## Arguments

| | |
|---|---|
| y1, y2 | vectors of obsereved values in the two groups |
| mm, nn | number of observations in the corresponding groups |

## Value

A list with items

| | |
|---|---|
| tstat | The t-test statistic |
| df | The degrees of freedom (Welch approximation) |

---

| plotApproxCovar | *Plot an approximatio of the correlation structure of the test statistics* |
|---|---|

---

## Description

Plot an approximatio of the correlation structure of the test statistics

## Usage

```
plotApproxCovar(
  reconsiFit,
  col = colorRampPalette(c("yellow", "blue"))(12),
  x = seq(-4.2, 4.2, 0.1),
  y = seq(-4.2, 4.2, 0.1),
  xlab = "Z-values",
  ylab = "Z-values",
  nBins = 82L,
  binEdges = c(-4.1, 4.1),
  ...
)
```

## Arguments

reconsiFit        The reconsi fit

col, x, y, xlab, ylab, ...

              A list of arguments for the image() function.

nBins, binEdges

              passed on to the getApproxCovar function

## Details

By default, yellow indicates negative correlaton between bin counts, blue positive correlation

## Value

invisible()

## Note

This is not the covariance matrix of the p test statistic, nor of the data! It is an approximate covariance matrix of binned test statistics for visualization purposes.

## Examples

```
p = 200; n = 50; B = 5e1
x = rep(c(0,1), each = n/2)
mat = cbind(
matrix(rnorm(n*p/10, mean = 5+x),n,p/10), #DA
matrix(rnorm(n*p*9/10, mean = 5),n,p*9/10) #Non DA
)
mat = mat = mat + rnorm(n, sd = 0.3) #Introduce some dependence
fdrRes = reconsi(mat, x, B = B)
plotApproxCovar(fdrRes)
```

---

| plotNull | *Plot the obtained null distribution along with a histogram of observed test statistics* |
|---|---|

---

## Description

Plot the obtained null distribution along with a histogram of observed test statistics

## Usage

```
plotNull(
  fit,
  lowColor = "yellow",
  highColor = "blue",
  idDA = NULL,
  nResampleCurves = length(fit$Weights),
  hSize = 0.5
)
```

## Arguments

| | |
|---|---|
| `fit` | an object returned by the reconsi() (or testDAA()) function |
| `lowColor, highColor` | |
| | The low and high ends of the colour scale |
| `idDA` | indices of known null taxa |
| `nResampleCurves` | |
| | The number of resampling null distributions to plot |
| `hSize` | A double, the size of the line of the collapsed null estimate |

## Value

a ggplot2 plot object

## Examples

```
p = 175; n = 50; B = 1e2
#Low number of resamples keeps computation time down
x = rep(c(0,1), each = n/2)
mat = cbind(
matrix(rnorm(n*p/10, mean = 5+x),n,p/10), #DA
matrix(rnorm(n*p*9/10, mean = 5),n,p*9/10) #Non DA
)
fdrRes = reconsi(mat, x, B = B)
plotNull(fdrRes)
```

---

| ptEdit | *A custom function to calculate the distribution function of the t-test statistic. For internal use only* |
|---|---|

---

## Description

A custom function to calculate the distribution function of the t-test statistic. For internal use only

## Usage

```
ptEdit(q)
```

## Arguments

q               a vector with t-statistic and degrees of freedom

## Value

A value between 0 and 1, the evaluation of the cdf

---

| qtEdit | *A custom function to calculate the quantile function of the t-test statistic. For internal use only* |
|---|---|

---

## Description

A custom function to calculate the quantile function of the t-test statistic. For internal use only

## Usage

```
qtEdit(p)
```

## Arguments

p               a vector with quantile and degrees of freedom

## Value

the corresponding quantile

---

quantCorrect                    *Correct quantiles by not returning 0 or 1*

---

### Description

Correct quantiles by not returning 0 or 1

### Usage

```
quantCorrect(quants)
```

### Arguments

quants            A vector of quantiles

### Value

The same vector of quantiles but without 0 or 1 values

---

reconsi                         *Perform simultaneous inference through collapsed resampling null distributions*

---

### Description

Perform simultaneous inference through collapsed resampling null distributions

### Usage

```
reconsi(
  Y,
  x = NULL,
  B = 1000L,
  test = "wilcox.test",
  argList = list(),
  distFun = "pnorm",
  zValues = TRUE,
  testPargs = list(),
  z0Quant = pnorm(c(-1, 1)),
  gridsize = 801L,
  maxIter = 1000L,
  tol = 1e-08,
  center = FALSE,
  replace = is.null(x),
  zVals = NULL,
  estP0args = list(z0quantRange = seq(0.05, 0.45, 0.0125), smooth.df = 3),
```

```
    resamZvals = FALSE,
    smoothObs = TRUE,
    scale = FALSE,
    tieBreakRan = FALSE,
    pi0 = NULL
)
```

## Arguments

| | |
|---|---|
| Y | the matrix of sequencing counts |
| x | a grouping factor. If provided, this grouping factor is permuted. Otherwise a bootstrap procedure is performed |
| B | the number of resampling instances |
| test | Character string, giving the name of the function to test for differential absolute abundance. Must accept the formula interface. Features with tests resulting in observed NA test statistics will be discarded |
| argList | A list of arguments, passed on to the testing function |
| distFun | the distribution function of the test statistic, or its name. Must at least accept an argument named 'q', 'p' and 'x' respectively. |
| zValues | A boolean, should test statistics be converted to z-values. See details |
| testPargs | A list of arguments passed on to distFun |
| z0Quant | A vector of length 2 of quantiles of the null distribution, in between which only null values are expected |
| gridsize | The number of bins for the kernel density estimates |
| maxIter | An integer, the maximum number of iterations in the estimation of the null distribution |
| tol | The tolerance for the infinity norm of the central borders in the iterative procedure |
| center | A boolean, should observations be centered in each group prior to permuations? See details. |
| replace | A boolean. Should resampling occur with replacement (boostrap) or without replacement (permutation) |
| zVals | An optional list of observed (statObs) and resampling (statsPerm) z-values. If supplied, the calculation of the observed and resampling test statistics is skipped and the algorithm proceeds with calculation of the consensus null distribution |
| estP0args | A list of arguments passed on to the estP0 function |
| resamZvals | A boolean, should resampling rather than theoretical null distributions be used? |
| smoothObs | A boolean, should the fitted rather than estimated observed distribution be used in the Fdr calculation? |
| scale | a boolean, should data be scaled prior to resampling |
| tieBreakRan | A boolean, should ties of resampling test statistics be broken randomly? If not, midranks are used |
| pi0 | A known fraction of true null hypotheses. If provided, the fraction of true null hypotheses will not be estimated. Mainly for oracle purposes. |

**Details**

Efron (2007) centers the observations in each group prior to permutation. As permutations will remove any genuine group differences anyway, we skip this step by default. If zValues = FALSE, the density is fitted on the original test statistics rather than converted to z-values. This unlocks the procedure for test statistics with unknown distributions, but may be numerically less stable.

**Value**

A list with entries

| | |
|---|---|
| statsPerm | Resampling Z-values |
| statObs | Observed Z-values |
| distFun | Density, distribution and quantile function as given |
| testPargs | Same as given |
| zValues | A boolean, were z-values used? |
| resamZvals | A boolean, were the resampling null distribution used? |
| cdfValObs | Cumulative distribution function evaluation of observed test statistics |
| p0estimated | A boolean, was the fraction of true null hypotheses estimated from the data? |
| Fdr,fdr | Estimates of tail-area and local false discovery rates |
| p0 | Estimated or supplied fraction of true null hypotheses |
| iter | Number of iterations executed |
| fitAll | Mean and standard deviation estimated collapsed null |
| PermDensFits | Mean and standard deviations of resamples |
| convergence | A boolean, did the iterative algorithm converge? |
| zSeq | Basis for the evaluation of the densities |
| weights | weights of the resampling distributions |
| zValsDensObs | Estimated overall densities, evaluated in zSeq |

**Note**

Ideally, it would be better to only use unique resamples, to avoid unnecesarry replicated calculations of the same test statistics. Yet this issue is almost alwyas ignored in practice; as the sample size grows it also becomes irrelevant. Notice also that this would require to place weights in case of the bootstrap, as some bootstrap samples are more likely than others.

**Examples**

```
#Important notice: low number of resamples B necessary to keep
# computation time down, but not recommended. Pray set B at 200 or higher.
p = 50; n = 20; B = 5e1
x = rep(c(0,1), each = n/2)
mat = cbind(
matrix(rnorm(n*p/10, mean = 5+x), n, p/10), #DA
matrix(rnorm(n*p*9/10, mean = 5), n, p*9/10) #Non DA
)
```

```
fdrRes = reconsi(mat, x, B = B)
fdrRes$p0
#Indeed close to 0.9
estFdr = fdrRes$Fdr
#The estimated tail-area false discovery rates.

#With another type of test. Need to supply quantile function in this case
fdrResLm = reconsi(mat, x, B = B,
test = function(x, y){
fit = lm(y~x)
c(summary(fit)$coef["x","t value"], fit$df.residual)},
distFun = function(q){pt(q = q[1], df = q[2])})

#With a test statistic without known null distribution(for small samples)
fdrResKruskal = reconsi(mat, x, B = B,
test = function(x, y){
kruskal.test(y~x)$statistic}, zValues = FALSE)

#Provide an additional covariate through the 'argList' argument
z = rpois(n , lambda = 2)
fdrResLmZ = reconsi(mat, x, B = B,
test = function(x, y, z){
fit = lm(y~x+z)
c(summary(fit)$coef["x","t value"], fit$df.residual)},
distFun = function(q){pt(q = q[1], df = q[2])},
argList = list(z = z))

#When nog grouping variable is provided, a bootstrap is performed
matBoot = cbind(
matrix(rnorm(n*p/10, mean = 1), n, p/10), #DA
matrix(rnorm(n*p*9/10, mean = 0), n, p*9/10) #Non DA
)
fdrResBoot = reconsi(matBoot, B = B,
test = function(y, x){testRes = t.test(y, mu = 0, var.equal = TRUE);
c(testRes$statistic, testRes$parameter)},
distFun = function(q){pt(q = q[1], df = q[2])},
center = TRUE, replace = TRUE)
```

---

| rowMultiply | *A function to efficiently row multiply a a-by-b matrix by a vector of length b. More memory intensive but that does not matter with given matrix sizes* |
|---|---|

---

## Description

A function to efficiently row multiply a a-by-b matrix by a vector of length b. More memory intensive but that does not matter with given matrix sizes

## Usage

```
rowMultiply(matrix, vector)
```

## Arguments

| | |
|---|---|
| `matrix` | a numeric matrix of dimension a-by-b |
| `vector` | a numeric vector of length b |

## Details

t(t(matrix)*vector) but then faster

## Value

a matrix, row multplied by the vector

---

| `stabExp` | *A function to numerically stabilize an exponentiation. For internal use only* |
|---|---|

---

## Description

A function to numerically stabilize an exponentiation. For internal use only

## Usage

```
stabExp(exps)
```

## Arguments

| | |
|---|---|
| `exps` | the vector to be exponentiated |

## Value

the vector with the maximum subtracted

---

| `testDAA` | *A function to test for differential absolute abundance on a phyloseq object* |
|---|---|

---

## Description

A function to test for differential absolute abundance on a phyloseq object

## Usage

```
testDAA(Y, ...)

## S4 method for signature 'phyloseq'
testDAA(Y, groupName, FCname, ...)

## S4 method for signature 'matrix'
testDAA(Y, FC, x, S = rowSums(Y), tieBreakRan = TRUE, ...)
```

## Arguments

| | |
|---|---|
| Y | A phyloseq object, or a data matrix with samples in the rows and OTUs in the columns |
| ... | passed on to the reconsi() function |
| groupName | A character string, the name of a variable in physeq indicating the grouping factor |
| FCname | A character string, the name of a variable in physeq containing the total cell count |
| FC | a vector of length n with total flow cytometry cell counts |
| x | a grouping factor of length n |
| S | a vector of library sizes. Will be calculated if not provided |
| tieBreakRan | A boolean, should ties be broken at random. |

## Value

See the reconsi() function

## Examples

```
#Test for phyloseq object
library(phyloseq)
VandeputtePruned = prune_samples(Vandeputte,
samples = sample_names(Vandeputte)[20:40])
testVanDePutte = testDAA(VandeputtePruned, "Health.status", "absCountFrozen",
B = 15)
#Test for matrix
testMat = testDAA(as(otu_table(VandeputtePruned), "matrix"),
get_variable(VandeputtePruned, "Health.status"),
get_variable(VandeputtePruned,"absCountFrozen"), B = 15)
```

---

Vandeputte                            *Microbiomes of Crohn's disease patients and healthy controls*

---

## Description

Microbiome sequencing data of Crohn's disease patients, and healthy controls, together with other baseline covariates. Both sequencing and flow cytometry data are available.

## Usage

```
Vandeputte
```

## Format

A phyloseq object with an OTU-table and sample data

**otu_table**  Count data matrix of 234 taxa in 135 samples

**sample_data**  Data frame of patient covariates

## Source

https://www.ncbi.nlm.nih.gov/pubmed/29143816

# Index