Package ‘SSPA’

October 17, 2020

Type Package

Title General Sample Size and Power Analysis for Microarray and Next-Generation Sequencing Data

Version 2.28.0

Author Maarten van Iterson

Maintainer Maarten van Iterson <mviterson@gmail.com>

Description General Sample size and power analysis for microarray and next-generation sequencing data.

License GPL (>= 2)

LazyLoad yes

Imports graphics, stats, qvalue, lattice, limma

Depends R (>= 2.12), methods

Suggests BiocStyle, knitr, rmarkdown, genefilter, edgeR, DESeq

VignetteBuilder knitr

URL http://www.humgen.nl/MicroarrayAnalysisGroup.html

Collate 'zzz.R' 'numericalintegration.R' 'trimmingbinning.R' 'DistributionClass.R' 'PilotDataClass.R' 'SampleSizeClass.R' 'bitriangular.R' 'deconvolution.R' 'conjugategradient.R' 'Ferreira.R' 'tikhonov.R' 'powerandsamplesize.R'

biocViews ImmunoOncology, GeneExpression, RNASeq, Microarray, StatisticalMethod

RoxygenNote 7.0.2

git_url https://git.bioconductor.org/packages/SSPA

git_branch RELEASE_3_11

git_last_commit 6a3aa3a

git_last_commit_date 2020-04-27

Date/Publication 2020-10-16

R topics documented:

`cornerLcurve` *Find corner L-curve*

Description

Find corner L-curve

Usage

```
cornerLcurve(A, beta, b, lambda, z, verbose = FALSE)
```

Arguments

`A`	the A matrix of the system: Ax = b.
`beta`	regression coefficients.
`b`	the b vector of the system: Ax = b.
`lambda`	grid of penalty values.
`z`	intermediate matrix for L-curvature calculation.
`verbose`	Verbose TRUE/FALSE

Details

details follow.

Value

index for the corner of the L-curve.

Author(s)

Maarten van Iterson

`cornerScurve` *Find corner S-curve*

Description

Find corner S-curve

Usage

```
cornerScurve(A, beta, b, lambda, z, verbose = FALSE)
```

Arguments

`A`	the A matrix of the system: Ax = b.
`beta`	regression coefficients.
`b`	the b vector of the system: Ax = b.
`lambda`	grid of penalty values.
`z`	intermediate matrix for L-curvature calculation.
`verbose`	Verbose TRUE/FALSE

Details

details follow.

Value

index for the corner of the S-curve.

Author(s)

Maarten van Iterson

`dbitri` *Density function for a bi-triangular random variable.*

Description

Density function for a bi-triangular random variable.

Usage

```
dbitri(x, a = log2(1.2), b = log2(4), m = log2(2))
```

Arguments

`x`	vector
`a`	location of point ... Default a = log2(1.2).
`b`	location of point ... Default b = log2(4).
`m`	location of the midpoint of the triangle. Default m = log2(2).

Details

For more details see M. Langaas et al. JRSS B 2005.

Value

Gives the density function.

Author(s)

Maarten van Iterson

Examples

```
curve(dbitri, -4, 4)
```

`deconvolution` *deconvolution estimator using fft*

Description

TODO: suppress warning message in pt() : full precision was not achieved in 'pnt' TODO:replace massdist by trimbin? deconvolution estimator using fft

Usage

```
deconvolution(object)
```

Arguments

`object`	of class SampleSize

Details

details follow

Value

object of class SampleSize

Author(s)

Maarten van Iterson

deepSAGE *Test statistics derived from a deepSAGE experiment*

Description

follow

Usage

```
data(deepSAGE)
```

Format

A vector of 44882 test statistics.

Vector of test statistics obtained by performing a likelihood ratio test using edgeR

Details

follow

Source

’t Hoen, P.A.C. Ariyurek, Y. Thygesen, H.H. Vreugdenhil, E. Vossen, R.H.A.M. de Menezes, R.X. Boer, J.M. van Ommen, G.B. and den Dunnen, J.T., Deep Sequencing-based Expression analysis shows Major Advances in Robustness, Resolution and Inter-lab Portability over Five Microarray Platforms, Nucleic Acids Research, 2008.

Examples

```
data(deepSAGE)
str(deepSAGE)
```

defineEffectSizeRange *defineEffectSizeRange*

Description

defineEffectSizeRange

Usage

```
defineEffectSizeRange(object, from, to, resolution)
```

Arguments

`object`	pilotdata object
`from`	lower limit effect-sizes
`to`	upper limit effect-sizes
`resolution`	should be power of 2 default length statistics round to nearest power of 2

Value

effect-size range TODO use trimmingbinning function for this!

`findroot` *bisection method for finding ustar*

Description

bisection method for finding ustar

Usage

```
findroot(g, h, umax, verbose = FALSE, plot = FALSE)
```

Arguments

`g`	left-hand-side
`h`	right-hand side
`umax`	between [0, 1] usually alpha.
`verbose`	TRUE/FALSE
`plot`	TRUE/FALSE

Details

solve G(u) = u pi0 (1-alpha)/(aplha(1-pi0)), more details follow.

Value

ustar

Author(s)

Maarten van Iterson

`midpoint` *Midpoint rule for numerical integration.*

Description

Implementation of the midpoint rule for the numerical integration of uni- and bivariate functions.

Usage

```
midpoint(f, a, b, n, ...)
```

Arguments

`f` bivariate function.

`a` lower bound of the grid.

`b` upper bound of the grid.

`n` grid size.

`...` trick for evaluating the second parameter in case a bivariate functions was supplied.

Details

details follow

Value

vector or matrix of function evaluations use sum to obtain the integrand.

Author(s)

Maarten van Iterson

`nncg` *Non-negative conjugate gradient algorithm*

Description

Non-negative conjugate gradient algorithm

Usage

```
nncg(A, b, type = 1, trace = FALSE)
```

Arguments

`A` the A matrix of the system: Ax = b.

`b` the b vector of the system: Ax = b.

`type` for the conjugate-gradients method. Takes value ’1’ for the Fletcher-Reeves update, ’2’ for Polak-Ribiere and ’3’ for Beale-Sorenson.

`trace` tracing information on the progress of the optimization is produced.

Details

C-implementation, details follow.

Value

list containg regression coefficients and some additional information.

Author(s)

Maarten van Iterson

`Nutrigenomics` *Test statistics from a Nutrigenomics gene expression profiling experiment*

Description

There are five sets of test statistics each represents a different compound and exposure time. Test statistics were obtained by using an empirical Bayes linear model.

Usage

```
data(Nutrigenomics)
```

Format

A data frame with 16539 test statistics for five experiments.

First row indicates the effective sample size of the experiment. Column names refer to the compound and exposure time (see details).

Details

In this experiment the outcome of specific PPAR-alpha activation on murine small intestinal gene expression was examined using Affymetrix GeneChip Mouse 430 2.0 arrays. PPAR-alpha was activated by several PPAR-alpha-agonists that differed in activating potency. In this paper the data of three agonists were used, namely Wy14,643, fenofibrate and trilinolenin (C18:3). The first two compounds belong to the fibrate class of drugs that are widely prescribed to treat dyslipidemia, whereas trilinolenin is an agonist frequently found in the human diet. For intestinal PPAR-alpha, Wy14,643 is the most potent agonist followed by C18:3 and fenofibrate. Since time of exposure also affects the effect size, intestines were collected 6 hrs (all three agonists) or 5 days (Wy14,643 and fenofibrate only) after exposure.

Source

van Iterson, M. 't Hoen, P.A.C. Pedotti, P. Hooiveld, G.J.E.J. den Dunnen, J.T. van Ommen, G.J.B. Boer, J.M. Menezes, R.X., Relative power and sample size analysis on gene expression profiling data, BMC Genomics, (2009).

Examples

```
data(Nutrigenomics)
str(Nutrigenomics)
```

`pbitri` *Distribution function for a bi-triangular random variable.*

Description

Distribution function for a bi-triangular random variable.

Usage

```
pbitri(q, a = log2(1.2), b = log2(4), m = log2(2))
```

Arguments

`q`	vector of quantiles.
`a`	location of point, ... Default a = log2(1.2).
`b`	location of point, ... Default b = log2(4).
`m`	location of the midpoint of the triangle. Default m = log2(2).

Details

For more details see M. Langaas et al. JRSS B 2005.

Value

Gives the distribution function.

Author(s)

Maarten van Iterson

Examples

```
curve(pbitri, -4, 4)
```

`picardplot` *Picard-plot*

Description

Generates Picard-plot

Usage

```
picardplot(
  A,
  b,
  xlim = c(1, length(b)),
  ylim = NULL,
  main = "Picard-Plot",
  legend = TRUE
)
```

Arguments

`A`	the A matrix of the system: Ax = b.
`b`	the b vector of the system: Ax = b.
`xlim`	xlim of Picard-plot.
`ylim`	ylim of Picard-plot.
`main`	main
`legend`	legend

Details

details follow.

Value

generates Picard-plot.

Author(s)

Maarten van Iterson

`pilotData` *User friendly interface to class "PilotData"*

Description

User friendly interface to class "PilotData"

Usage

```
pilotData(
  statistics = NULL,
  samplesize = NULL,
  distribution = c("norm", "t", "f", "chisq"),
  ...
)
```

Arguments

`statistics`	vector of test statistics
`samplesize`	total sample size of the pilot-data or effective sample size in two-group case (see Details for more information).
`distribution`	type of the null/alternative distribution, one of ’norm’, ’t’, ’f’ or ’chisq’
`...`	additional arguments for the distribution like degrees of freedom

Details

details follow In the two-group case the effective sample size is defined as the square-root of the inverse of 1/n1 + 1/n2.

Value

object of class "PilotData"

Author(s)

Maarten van Iterson

Examples

```
pd <- pilotData(statistics=rnorm(100), samplesize=10, distribution="norm")
pd
plot(pd)
```

---

predictpower *Predict power for given vector of sample sizes*

Description

Predict power for given vector of sample sizes

Usage

```
predictpower(object, samplesizes, alpha = 0.1, verbose = FALSE, plot = FALSE)
```

Arguments

`object`	of class 'SampleSize'
`samplesizes`	vector of total sample sizes.
`alpha`	FDR.
`verbose`	TRUE/FALSE
`plot`	TRUE/FALSE

Details

details follow.

Value

predicted power.

Author(s)

Maarten van Iterson

qbitri *Quantile function for a bi-triangular random variable.*

Description

Quantile function for a bi-triangular random variable.

Usage

```
qbitri(p, a = log2(1.2), b = log2(4), m = log2(2))
```

Arguments

`p`	vector of probabilities.
`a`	location of point, ... Default a = log2(1.2).
`b`	location of point, ... Default b = log2(4).
`m`	location of the midpoint of the triangle. Default m = log2(2).

Details

For more details see M. Langaas et al. JRSS B 2005.

Value

Gives the quantile function.

Author(s)

Maarten van Iterson

Examples

```
curve(qbitri, 0, 1)
```

rbitri *Random generation of bitriangular distributed values.*

Description

Random generation of bitriangular distributed values.

Usage

```
rbitri(n, a = log2(1.2), b = log2(4), m = log2(2))
```

Arguments

`n`	number of observations.
`a`	location of point, ... Default a = log2(1.2).
`b`	location of point, ... Default b = log2(4).
`m`	location of the midpoint of the triangle. Default m = log2(2).

Details

For more details see M. Langaas et al. JRSS B 2005.

Value

Generates random deviates.

Author(s)

Maarten van Iterson

Examples

```
hist(rbitri(100), freq=FALSE)
curve(dbitri, add=TRUE)
```

---

regularization *Find optimal regularization parameter*

Description

Find optimal regularization parameter

Usage

```
regularization(
  b,
  A,
  beta,
  edf,
  lambda,
  z,
  method = c("lcurve", "gcv", "aic"),
  plot = TRUE,
  log = TRUE,
  verbose = FALSE
)
```

Arguments

b	the b vector of the system: Ax = b.
A	the A matrix of the system: Ax = b.
beta	regression coefficients.
edf	effective degrees of freedom.
lambda	grid of penalty values.
z	intermediate matrix for L-curvature calculation.
method	Either the L-curve, GCV or AIC.
plot	Plot TRUE/FALSE.
log	Plot on log-scale TRUE/FALSE.
verbose	Verbose TRUE/FALSE

Details

details follow.

Value

generates optionally figure and returns the index for the optimal penalty value.

Author(s)

Maarten van Iterson

---

sampleSize *User friendly interface to class 'SampleSize'*

Description

User friendly interface to class "SampleSize"

Usage

```
sampleSize(
  PilotData,
  method = c("deconv", "congrad", "tikhonov", "ferreira"),
  control = list(from = -6, to = 6, resolution = 2^9)
)
```

Arguments

`PilotData`	object of class 'PilotData'.
`method`	estimation method one of 'deconv', 'congrad', 'tikhonov' or 'ferreira'. See 'Details'.
`control`	A list of control parameters. See 'Details'.

Details

The default method is 'deconv' which is a kernel deconvolution density estimator implementated using fft. The 'nncg' is a nonnegative conjugate gradient algorithm based on R's implementation see optim. 'tikonov' implements ridge-regression with optimal penalty selection using the L-curve approach. Higher order penalties are possible as well using a transformation to standard form. The 'control' argument is a list that can supply any of the following components, however per method some logical checks are built-in:

- deconv:
  - method:'deconv', 'ferreira'
  - pi0Method:the pi0 estimation method one of 'Langaas', 'Storey', 'Ferreira', 'Userdefined'
  - pi0:if method = 'ferreira' grid pi0-value need to be suppled e.g. seq(0.1, 0.99, 0.01)
  - adjust:Default TRUE, adjust pi0 esitmate if density of effect size is somewhere negative.
  - a:Adjust pi0 better approach suggested by Efron. Symmetric range around zero of size 0.5.

  - bandwith:Default NULL uses 1/sqrt(log(length(statistics)))
  - kernel:Either 'fan', 'wand', 'sinc' kernels can be used.
  - from:Density of effect sizes should be estimated from = -6
  - to:6
  - resolution:Density of effect sizes should be estimated on 2^9 points.
  - verbose:Default FALSE if TRUE additional information is printed to the console.
- congrad:
  - integration:'midpoint', 'trapezoidal', 'simpson'
  - scale:'pdfstat', 'cdfstat', 'cdfpval'
  - trim:0.01, 0.99
  - symmetric:TRUE
  - bin:'epdf', 'ecdf'
  - from:-6
  - to:6
  - resolution:500
  - verbose:Default FALSE if TRUE additional information is printed to the console.
- tikhonov:
  - integration:'midpoint', 'trapezoidal', 'simpson'
  - scale:'pdfstat', 'cdfstat', 'cdfpval'
  - trim:0.01, 0.99
  - symmetric:TRUE
  - bin:'epdf', 'ecdf'
  - from:-6
  - to:6
  - resolution:500
  - modelselection:'lcurve', 'gcv', 'aic'
  - log:TRUE
  - penalty:0
  - lambda:10^seq(-10, 10, length=100)
  - verbose:Default FALSE if TRUE additional information is printed to the console.
- 'ferreira:'

Value

object of class SampleSize.

Author(s)

Maarten van Iterson

References

Langaas, Storey, Ferreira, Hansen, van Iterson

See Also

`optim`

Examples

```
m <- 5000 ##number of genes
J <- 10 ##sample size per group
pi0 <- 0.8 ##proportion of non-differentially expressed genes
m0 <- as.integer(m*pi0)
mu <- rbitri(m - m0, a = log2(1.2), b = log2(4), m = log2(2)) #effect size distribution
data <- simdat(mu, m=m, pi0=pi0, J=J, noise=NULL)
library(genefilter)
stat <- rowttests(data, factor(rep(c(0, 1), each=J)), tstatOnly=TRUE)$statistic
pd <- pilotData(statistics=stat, samplesize=sqrt(J/2), distribution='norm')
ss <- sampleSize(pd, method='deconv')
plot(ss)
```

show-methods *General show method for Classes PilotData and SampleSize*

Description

Methods for function show in package **SSPA**

Methods

signature(object = "PilotData") Show the content of a PilotData-object in a userfriendly way.

signature(object = "SampleSize") Show the content of a SampleSize-object in a userfriendly way.

simdat *Generate simulated microarray data using the bitriangular distribution.*

Description

Simulated microarray data.

Usage

```
simdat(
  mu,
  m,
  pi0,
  J,
  nullX = function(x) rnorm(x, 0, 1),
  nullY = function(x) rnorm(x, 0, 1),
  noise = 0.01
)
```

Arguments

`mu` vector of effect sizes drawn from the bitriangular distribution.

`m` number of features (genes, tags, ...).

`pi0` proportion of nondifferentially expressed features.

`J` number of samples per group.

`nullX` the distribution of nondifferentially expressed features.

`nullY` the distribution of nondifferentially expressed features.

`noise` standard deviation of the additive noise.

Details

details follow

Value

Matrix of size m x (2J), containing the simulated values.

Author(s)

Maarten van Iterson

Examples

```
##generate two-group microarray data
m <- 5000 ##number of genes
J <- 10 ##sample size per group
pi0 <- 0.8 ##proportion of non-differentially expressed genes
m0 <- as.integer(m*pi0)
mu <- rbitri(m - m0, a = log2(1.2), b = log2(4), m = log2(2)) #effect size distribution
data <- simdat(mu, m=m, pi0=pi0, J=J, noise=0.01)
```

`simpson` *Simpson's rule for numrical integration.*

Description

Implementation of Simpson's rule for the numerical integration of uni- and bivariate functions.

Usage

```
simpson(f, a, b, n = 5, ...)
```

Arguments

`f` bivariate function.

`a` lower bound of the grid.

`b` upper bound of the grid.

`n` grid size.

`...` trick for evaluating the second parameter in case a bivariate functions was supplied.

Details

details follow

Value

vector or matrix of function evaluations use sum to obtain the integrand.

Author(s)

Maarten van Iterson

`standardform` *Transform to standard from*

Description

Transforms an objective function containing non-diagonal penalty to standard form (diagonal penalty)

Usage

```
standardform(b, A, L)
```

Arguments

`b`	the b vector of the system: Ax = b.
`A`	the A matrix of the system: Ax = b.
`L`	non-diagonal penalty

Details

details follow.

Value

Transformed system ||Ax - b || + lambda ||L|| with L diagonal.

Author(s)

Maarten van Iterson

`tikhonov` *Tikhonov regularization*

Description

Tikhonov regularization.

Usage

```
tikhonov(b, A, lambda, penalty = 0)
```

Arguments

`b` the b vector of the system: Ax = b.

`A` the A matrix of the system: Ax = b.

`lambda` grid of values for the penalty.

`penalty` penalty either 0 = ridge, 1 = first order differences or 2 = second order differences.

Details

details follow.

Value

regression coefficients, effective degrees of freedom, intermediate matrix for L-curvature calculation.

Author(s)

Maarten van Iterson

`trapezoidal` *Trapezoidal rule for numrical integration.*

Description

Implementation of the trapezoidal rule for the numerical integration of uni- and bivariate functions.

Usage

```
trapezoidal(f, a, b, n, ...)
```

Arguments

`f` bivariate function.

`a` lower bound of the grid.

`b` upper bound of the grid.

`n` grid size.

`...` trick for evaluating the second parameter in case a bivariate functions was supplied.

Details

details follow

Value

vector or matrix of function evaluations use sum to obtain the integrand.

Author(s)

Maarten van Iterson

trimbin *Binning and trimming of the test statistics vector.*

Description

Binning and trimming of the test statistics vector.

Usage

```
trimbin(
  statistics,
  nbins = 100,
  trim = c(0.01, 0.99),
  bin = c("epdf", "ecdf"),
  symmetric = TRUE,
  plot = TRUE
)
```

Arguments

`statistics`	vector of test statistics.
`nbins`	number of bins.
`trim`	vector of size two with lower and upper limits for the trimming.
`bin`	"epdf" or "ecdf" binning using respectively, empirical density of cumulutive distribution function.
`symmetric`	symmetric trimming TRUE/FALSE.
`plot`	plot results TRUE/FALSE.

Details

details follow

Value

trimmed and binned vector of test statistics.

Author(s)

Maarten van Iterson

Index