Package 'les'

November 12, 2026

Type Package

Title Identifying Differential Effects in Tiling Microarray Data

Version 1.62.0

Author Julian Gehring, Clemens Kreutz, Jens Timmer

Maintainer Julian Gehring <jg-bioc@gmx.com>

Imports boot, gplots, RColorBrewer

Depends R (>= 2.13.2), methods, graphics, fdrtool

Suggests Biobase, limma

Enhances parallel

Description The 'les' package estimates Loci of Enhanced Significance (LES) in tiling microarray data. These are regions of regulation such as found in differential transcription, CHiP-chip, or DNA modification analysis. The package provides a universal framework suitable for identifying differential effects in tiling microarray data sets, and is independent of the underlying statistics at the level of single probes.

License GPL-3

LazyLoad yes

biocViews Microarray, DifferentialExpression, ChIPchip, DNAMethylation, Transcription

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

git_branch RELEASE_3_23

git_last_commit d7d9b97

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

les-package 2
chi2,Les-method 3
ci,Les-method 5
estimate,Les-method 7
export,Les-method 9
Les,ANY,numeric-method 11
Les-class 12
les-internal 14
plot,Les-method 15
regions,Les-method 16
spikeInData 18
spikeInStat 19
threshold,Les-method 20
weighting 21

Index **23**

`les-package` *Identifying Differential Effects in Tiling Microarray Data*

Description

The 'les' package estimates Loci of Enhanced Significance (LES) in tiling microarray data. These are regions of regulation such as found in differential transcription, CHiP-chip, or DNA modification analysis. The package provides a universal framework suitable for identifying differential effects in tiling microarray data sets, and is independent of the underlying statistics at the level of single probes.

Details

The 'les' package provides a universal framework for detecting differential effects in tiling microarray experiments.

It is universal in the sense that one is free to choose any statistical test to estimate the effect of differential effect for each probe on the tiling microarray. Provided with the p-values for each probe and the corresponding positions of the probes, 'les' uses a sliding window approach to estimate the fraction of regulated probes in the local surrounding of each probe. The approach is related to computing a spatially resolved and weighted false discovery rate, and yields a interpretable statistical feature Λ.

Resulting regions can be scored according to their overall effect. Methods for high-level plotting and export of the results to other software and genome browsers are provided.

The 'les' package is published under the GPL-3 license.

Author(s)

Julian Gehring, Clemens Kreutz, Jens Timmer

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

References

in preparation

This package is based on:

Kilian Bartholome, Clemens Kreutz, and Jens Timmer: Estimation of gene induction enables a relevance-based ranking of gene sets, Journal of Computational Biology: A Journal of Computational Molecular Cell Biology 16, no. 7 (July 2009): 959-967. http://www.liebertonline.com/doi/abs/10.1089/cmb.2008.0226

See Also

Class: Les

Methods and functions: Les estimate threshold regions ci chi2 export plot

Examples

```
data(spikeInStat)

x <- Les(pos, pval)
x <- estimate(x, 200)
x <- threshold(x)
x <- regions(x)

subset <- pos >= 5232300 & pos <= 5233200
x <- ci(x, subset, conf=0.90, nBoot=50)

## plot data
plot(x, region=TRUE)
plot(x, region=TRUE, error="ci")

## Not run:
## export data of chromosome 1
export(x, file="les_out.bed", chr=1)
export(x, file="les_out.wig", format="wig", chr=1)

## End(Not run)
```

---

chi2,Les-method *chi2*

Description

The 'chi2' method can be used to optimize the window size for defined regions of interest. It computes $\chi^2(w)$ for window sizes w based on the estimates of Λ and the false-discovery rate.

Usage

```
chi2(object, winSize, regions, offset, fdr = "lfdr", method,
scaling = les:::scaleNorm, nCores = NULL,  verbose = FALSE, ...)

## S4 method for signature 'Les'
chi2(object, winSize, regions, offset, fdr = "lfdr",
method, scaling = les:::scaleNorm, nCores = NULL,  verbose = FALSE, ...)
```

Arguments

object	Object of class 'Les' as returned by 'estimate' or a subsequent step.
winSize	Integer vector specifying the window sizes w for which $\chi^2(w)$ should be computed. For each value of 'winSize' and each region a computation as for 'estimate' is run. For details please see the 'win' argument in 'estimate'.
regions	Data frame containing the regions of interest. It has to contain the columns 'start', 'end' and 'chr' specifying the start and end position for each region, as well the chromosome if more than one chromosome is present. The structure is related to the 'regions' output of the 'regions' method. If missing the data frame from the 'regions' slot will be used if available. For details please see the 'regions' method.
offset	Integer or vector of integers specifying the offset for the regions given by the 'regions' input argument. If missing the regions will be taken as specified. If present start and end of each regions will be taken as 'start - offset' and 'end + offset'.
fdr	Character string specifying the fdr method to use for χ^2 computation (default: 'lfdr'). Possible values are 'lfdr' for local fdr and 'qval' for q-values. For details see the 'Details' section below and the 'fdrtool' package.
method	Character string specifying the method used for linear regression. It is equivalent to the 'method' argument in the 'estimate' method. If missing the value set in the 'estimate' method will be used.
scaling	Function specifying the scaling of Lambda and fdr (default: les:::scaleNorm). By default both will be scaled to the range [0,1].
nCores	Integer indicating the number of cores to use for computation. This feature requires the 'parallel' package which is only available for certain platforms. The package is used only if 'library(parallel)' has been called manually by the user before and if 'nCores' is an integer unequal NULL specifying the number of cores to use. The value is passed directly to 'mclapply' as argument 'n.cores'. For details and benefits please see the 'Details' section.
verbose	Logical indicating whether the progress of the computation should be printed on screen (default: FALSE).
...	Further arguments passed to subsequent functions.

Details

The 'chi2' method can be used to optimize the window size for defined regions of interest. It computes $\chi^2(w)$ for each window size w based on the estimates of false-discovery rate (fdr) and

$\Lambda(w)$ with a Leave-One-Out Cross Validation (LOOCV). The shape of the $\chi^2(w)$ landscape can constrain suitable values for w.

Value

Object of class 'Les' with additionally filled slots: winSize, chi2

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: les-package fdrtool

Class: Les

Methods and functions: Les estimate threshold regions ci chi2 export plot weighting

Examples

```
data(spikeInStat)

x <- Les(pos, pval)
x <- estimate(x, 200, weighting=rectangWeight)
x <- threshold(x)
x <- regions(x)

regions <- x["regions"]
winsize <- seq(100, 300, by=20)
x <- chi2(x, winsize, regions, offset=2500)

plot(winsize, x["chi2"], type="b")
```

---

ci,Les-method *ci*

Description

Computes confidence intervals (CI) for Λ using bootstrapping.

Usage

```
ci(object, subset, nBoot = 100, conf = 0.95, nCores = NULL, ...)

## S4 method for signature 'Les'
ci(object, subset, nBoot = 100, conf = 0.95, nCores =
NULL, ...)
```

Arguments

object	Object of class 'Les' as returned by 'estimate' or 'regions'.
subset	Vector of logical specifying the probes for which the CIs should be computed. If missing CIs will be computed for all probes.
nBoot	Integer specifying the number of bootstrap samples (default: 100). For details see 'boot' from the 'boot' package.
conf	Numeric specifying the confidence level (default: 0.95). For details see 'boot' from the 'boot' package.
nCores	Integer indicating the number of cores to use for computation. This feature requires the 'multicore' package which is only available for certain platforms. The package is used only if 'library(multicore)' has been called manually by the user before and if 'nCores' is an integer unequal NULL specifying the number of cores to use. The value is passed directly to 'mclapply' as argument 'n.cores'. For details and benefits please see the 'Details' section.
...	Further arguments passed to subsequent functions.

Details

The 'ci' method computes confidence intervals (CI) by bootstrapping probes in each window. Since based on percentiles the resulting CIs are asymmetrical.

All arguments for computation are taken from 'object' and thereby kept the same as for the results from 'estimation'.

Since bootstrapping is computational demanding and CIs are often only wanted for certain regions of interest it may be useful to restrict computation with the 'subset' argument.

The 'multicore' package can be used to spread the computation over several cores in a simple way. This can be useful on multi-core machines for large datasets or computation of confidence intervals for many probes. The 'multicore' package is not available on all platforms. To use multicore processing 'library(multicore)' has to be called beforehand and a number of cores to use has to be specified in 'nCores'. For details see the documentation of the 'multicore' package.

Value

Object of class 'Les' with additionally filled slots: ci, subset, nBoot, conf

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: les-package boot

Class: Les

Methods and functions: Les estimate threshold regions ci chi2 export plot

Examples

```
data(spikeInStat)

x <- Les(pos, pval)
x <- estimate(x, win=200, grenander=FALSE)

subset <- pos >= 5232300 & pos <= 5233200
x <- ci(x, subset, conf=0.90, nBoot=50)

plot(x, error="ci")

## Not run:
## multicore computation
## only available on certain platforms
library(multicore)
x <- ci(x, subset=150:200, conf=0.90, nBoot=50, nCores=2)

## End(Not run)
```

`estimate,Les-method` *estimate*

Description

The 'estimate' method computes the fraction Λ of probes with significant effect in the local surrounding of the genome.

Usage

```
estimate(object, win, weighting = triangWeight, grenander = TRUE,
se = FALSE, minProbes = 3, method = "la", nCores = NULL, verbose =
FALSE, ...)

## S4 method for signature 'Les'
estimate(object, win, weighting = triangWeight,
grenander = TRUE, se = FALSE, minProbes = 3, method = "la", nCores =
NULL, verbose = FALSE, ...)
```

Arguments

`object`	Object of class 'Les' containing experimental data, as returned by 'Les'.
`win`	Integer specifying window size for the weighting function. This value is directly passed to the function specified by 'weighting'. For details see the description for the used window function 'weighting'.
`weighting`	Function specifying the shape of the weighting window. If not specified the supplied 'triangWeight' function with a triangular window will be used. For details on other window functions and how to specify own functions please see the 'Details' section.

grenander	Logical specifying if the Grenander correction for the cumulative density should be used (default: TRUE). For details see the 'Details' section.
se	Logical indicating whether the standard error (SE) from the final linear model should be computed and stored (default: FALSE). The standard error displays the goodness of fit for every probe, but is not needed for further computation. If computation time is a critical factor computation of the SE can be omitted to save some time.
minProbes	Integer specifying the minimal number of unqiue p-values that must be present for each fit (default: 3). For very small number of p-values the cumulative density is not well defined and therefore estimation has a high uncertainty. If the number of unique p-values is smaller than 'minProbes' no estimation is performed for this probe and Λ =NA is returned.
method	Character string specifying the method used for linear regression (default: 'la'). Possible options are 'la' for a method based on linear algebra or 'qr' for a method based on qr decomposition. 'la' will be faster for few probes, 'qr' for many probes in a window. The best choice varies between data sets, parameters and machines. However this option only influences computation time but not the results.
nCores	Integer indicating the number of cores to use for computation. This feature requires the 'multicore' package which is only available for certain platforms. The package is used only if 'library(multicore)' has been called manually by the user before and if 'nCores' is an integer unequal NULL specifying the number of cores to use. The value is passed directly to 'mclapply' as argument 'n.cores'. For details and benefits please see the 'Details' section.
verbose	Logical indicating whether the progress of the computation should be printed on screen (default: FALSE).
...	Further arguments passed to subsequent functions.

Details

This function estimates Λ_i for all probes i. This is normally the first step in the analysis after storing the experimental data with 'Les'.

The 'win' argument influences the number of neighboring probes taken into account by the weighting function. The value is passed to the function specified in 'weighting'. Larger values result in a smoother features. Details on a reasonable choice for this value can be found in the references.

With the 'weighting' argument the applied weighting function can be specified from a predefined set or a custom function can be used. In the 'les' package the four functions 'triangWeight', 'rectangWeight', 'epWeight' and 'gaussWeight' are already supplied and offer a triangular, rectangular, Epanechnikov and Gaussian weighting function respectively. For details on the functions itself and how to use custom ones please see the documentation of the single functions or the vignette of this package.

The Grenander correction for the cumulative density includes the general knowledge about the concave shape of the cumulative density. This reduces the variance of the estimates and leads to a conservative estimation. Please note that using this feature may significantly increase computation time.

The 'multicore' package can be used to spread the computation over several cores in a simple way. This can be useful on multi-core machines for large datasets. The 'multicore' package is not available on all platforms. To use multicore processing 'library(multicore)' has to be called beforehand and a number of cores to use has to be specified in 'nCores'. For details see the documentation of the 'multicore package.

Please note that calling 'estimate' with an object returned by the methods 'ci' and 'regions' will overwrite data stored by these two methods. This ensures that no inconsistent data is stored.

Value

Object of class 'Les' with additionally filled slots: lambda, win, weighting, grenander, nProbes, se (se only if computed)

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: les-package

Class: Les

Methods and functions: Les estimate threshold regions ci chi2 export plot weighting

Examples

```
data(spikeInStat)

x <- Les(pos, pval)
x <- estimate(x, win=200)
x
```

---

export,Les-method *export*

Description

Exports the results into files for interaction with other software. Estimated regions can be exported in 'bed' and 'gff' format, Λ in 'wig' format.

Usage

```
export(object, file, format="bed", chr, range, description = "Lambda",
strand=".", group="les", precision=4, ...)

## S4 method for signature 'Les'
export(object, file, format="bed", chr, range,
description = "Lambda", strand=".", group="les", precision=4, ...)
```

Arguments

object	Object of class 'Les' containing experimental data, as returned by 'estimate' or 'regions'.
file	Character string specifying the file path and name for export.
format	Character string with the export format (default: 'bed'). Possible values are 'bed' and 'gff' for export of the estimated regions and 'wig' for export of Λ for the probes. If set to 'bed' or 'gff' the method 'regions' has to be run beforehand. Partial matches are also possible. For details on the formats please see the 'Details' section.
chr	Character string specifying the chromosome from which results should be exported. This value must be set if exporting to 'wig' format, for other formats is optional. 'chr' must have exactly one match in the 'chr' argument specified in 'Les'.
range	Numeric vector specifying the range of probe positions which should be exported. If missing all probes of the chromosome will be exported. This value has only an effect if 'format' is set to 'wig'.
description	Character string with description for the exported track (default: 'Lambda'). This will be used as description by several programs and genome browsers.
strand	Character string with strand specification for 'gff' export (default: '.'). Possible values are '+', '-' or '.'.
group	Character string with group specifications of the resulting regions in 'gff' format (default: 'les').
precision	Integer specifying the number of digits Λ should be rounded to for export to 'wig' format (default: 4).
...	Further arguments passed to subsequent functions.

Details

This function is useful to export the estimated Lambda to external programs and genome browsers.

The 'bed', 'gff' and 'wig' format provide widely used standard formats and are compatible with most genome browsers and related software. For details about the file formats see http://genome.ucsc.edu/FAQ/FAQformat.html.

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: les-package

Class: Les

Methods and functions: Les estimate threshold regions ci chi2 export plot

Examples

```
## Not run:
data(spikeInStat)

x <- Les(pos, pval)
x <- estimate(x, 200)
x <- threshold(x)
x <- regions(x)

export(x, file="test.bed")
export(x, file="test.gff", format="gff")
export(x, file="test.wig", format="wig", chr=0)

## End(Not run)
```

`Les,ANY,numeric-method`
Les

Description

Constructs an object of class 'Les' and stores experimental tiling microarray data. This is the initial step for all further analysis with the 'les' package.

Usage

```
Les(pos, pval, chr)
```

Arguments

`pos`	Integer vector containing the probe positions.
`pval`	Numeric vector containing the p-values corresponding to 'pos'. It must be of the same length as 'pos'.
`chr`	Vector specifying the chromosome each probe in located on. If missing all probes are located on one chromosome (default: 'chr0'). If specified it must be of the same length as 'pos'. Internally it will be stored as a factor.

Details

This method gathers all data necessary for subsequent analysis, checks for valid inputs and stores it in an object of class 'Les'.

The data is checked for the following criteria: 'pos' and 'chr' must not contain any NAs. 'pval' may contain NAs but such probes (including corresponding 'pos' and 'chr') are discarded for subsequent computation since they contain no usable information. Please note that in such a case fewer probes are stored in the resulting object then were passed to 'Les'. In case of duplicate probe positions on one chromosome a warning is printed. This normally indicates that probes from both strands are present in the data.

After storing the experimental data with 'Les' in an object the variables containing the original data can be deleted from the workspace. All further steps of 'les' will get their data from this object. This can be useful in cases when memory usage is a critical factor.

Value

Object of class 'Les' with filled slots: pos, pval, chr

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: `les-package`

Class: `Les`

Methods and functions: `Les` `estimate` `threshold` `regions` `ci` `chi2` `export` `plot`

Examples

```
data(spikeInStat)

x <- Les(pos, pval)
x
```

Les-class *Class 'Les'*

Description

The class 'Les' is used by the package 'les'.

Objects from the Class

Objects of class 'Les' are constructed by calling the function 'Les'.

Slots

`pos`: Integer vector with probe positions. Provided by the user through the 'Les' function.

`chr`: Factor with chromosomes for each probe. Provided by the user through the 'Les' function.

`lambda`: Numeric vector with estimates Lambda for each probe.

`lambda0`: Numeric vector with estimates Lambda0 without the Grenander estimator for each probe. This is used if 'grenander' is set to TRUE.

`se`: Numeric vector with standard error from linear fitting for each probe.

`nProbes`: Integer with number of probes used in fit for each probe

`pval`: Numeric vector with p-values for each probe. Provided by the user through the 'Les' function.

`win`: Integer with the window size. Set by the user in the 'estimate' function.

`weighting`: Function with the weighting function used for probe weighting according to position. Set by the user in the 'estimate' function.

`grenander`: Logical indicating usage of the Grenander estimator. Set by the user in the 'estimate' function.

`ci`: Data frame with the confidence intervals for each probe specified in 'subset'.

`nBoot`: Integer with the number of bootstraps to be drawn. Set by the user in the 'ci' function.

`conf`: Numeric with the confidence level. Set by the user in the 'ci' function.

`subset`: Integer vector with indices of bootstrap subset. Set by the user in the 'ci' function.

`theta`: Numeric with the threshold value Theta.

`nSigProbes`: Integer with the number of estimated significant probes.xs

`regions`: Data frame with estimated Loci of Enhanced Significance.

`limit`: Numeric with the threshold value for estimation of 'regions'. Set by the user in the 'regions' function.

`nChr`: Integer with the number of chromosomes.

`maxGap`: Numeric specifying the largest gap allowed in one region. Set by the user in the 'regions' function.

`minLength`: Integer specifying the minimal number of probes in one region. Set by the user in the 'regions' function.

`minProbes`: Integer specifying the minimal number of unique p-values allowed for each fit. Set by the user in the 'estimate' function.

`method`: Character specifying the method used for linear regression.

`winSize`: Integer vector specifying the window sizes used for chi2 computation.

`chi2`: Matrix containing the chi2 values for different window sizes.

`state`: Character vector containing the analysis steps applied on the data object. Used for internal consistency checks.

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: `les-package`

Class: `Les`

Methods and functions: `Les` `estimate` `threshold` `regions` `ci` `chi2` `export` `plot`

Examples

```
showClass("Les")
```

`les-internal` *Internal functions of the 'les' package.*

Description

Internal les functions

Usage

```
calcSingle(ind0, pos, pval, win, weighting, grenander, se, nBoot, conf,
minProbes, custom)
fitGsri(pval, index, cweight, nValidProbes, grenander, se, custom)
itLinReg(x, y, cweight, nValidProbes, se, custom, noBoot)
gsri(pval, grenander=FALSE, se=TRUE, custom=FALSE)
wcdf2(pval, weight)
mcsapply(X, FUN, ..., mc.cores=NULL, simplify=TRUE)
seFast(x, y, b)
log2ind(log)
ind2log(ind, n)
slopeWeight(x, y, c)
diagSquare(w, n)
qrSlope(x, y, w)
cdfDuplicates(pvalSort, weightSort)
seSlopeWeight(fit, n=length(fit$weights))
setState(state, flag)
checkState(state, flag, text=as.character(flag))
modGrenander(x, y, eta0, unique, check)
getArgs(name, first=NULL, last=NULL, ...)
```

Details

Internal functions of the 'les' package. These should not be called by the user.

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: `les-package`

Class: `Les`

Methods and functions: `Les` `estimate` `threshold` `regions` `ci` `chi2` `export` `plot`

plot,Les-method *plot method for class Les*

Description

The 'plot' method plots the estimates of the 'les' package along the genome. This includeds Λ with confidence intervals and estimated regions.

Usage

```
plot(x, y, ...)

## S4 method for signature 'Les'
plot(x, y, chr, error="none", region=FALSE, limit=TRUE,
rug=FALSE, xlim, ylim=c(0, 1),  ...)
```

Arguments

x	Object of class 'Les', as returned by 'estimate', 'threshold' or 'ci'.
y	Annotation object, currently not used.
chr	Character or numeric specifying which chromosome to plot. Must have a match in 'chr' passed to 'Les'. A value is required if the probes are located on more than one chromosome.
error	Character string specifying if error estimates for Λ should be plotted (default: "none"). Valid values are "none" for no error estimates and "ci" for confidence intervals computed by the 'ci' method.
region	Logical indicating whether the estimated regions should be included in the plot. The 'regions' method must have been called beforehand.
limit	Logical specifying whether the estimated threshold Θ should be indicated on the y-axis if estimated (default: TRUE).
rug	Logical whether the positions of the probes should be indicated along the x-axis (default: FALSE). For details see 'rug'.
xlim	Numeric vector with two elements specifying the range on the x-axis.
ylim	Numeric vector with two elements specifying the range on the y-axis.
...	Optional arguments used in order to customize the plot. See the 'details' section.

Details

This method provides high-level plotting for the 'Les' class.

The `plot` method uses a special system in order to customize the graphical elements of the figure. It allows to refer to the different components with the name of the additional input argument; its value is a list containing named graphical parameters for the underlying plot function. The following list describes the possible names and their contribution.

`plotArgs` Arguments for the axis and the labeling, passed to the `plot` function.

`borderArgs` Arguments for the border lines at Λ equal to 0 and 1, passed to the `abline` function.

`errorArgs` Arguments for the confidence interval of Λ, passed to the `plotCI` function of the **gplots** package.

`probeArgs` Arguments for the representation of probes, passed to the `points` function.

`limitArgs` Arguments for the horizontal line representing the threshold Θ, passed to the `abline` function.

`sigArgs` Arguments for the representation of significant probes with $\Lambda >= \Theta$, passed to the `points` function.

`rugArgs` Arguments for the representation of the probe coverage along the genome, passed to the `rug` function.

`regionArgs` Arguments for the representation of estimated LES, passed to the `rect` function. If 'col' is specified as a function it determines the color of each region depending on its input Λ (default: gray()). If 'col' is a vector its elements are used to color the regions with recycling.

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: `les-package`

Class: `Les`

Methods and functions: `Les` `estimate` `threshold` `regions` `ci` `chi2` `export` `plot`

Examples

```
data(spikeInStat)

x <- Les(pos, pval)
x <- estimate(x, 200)
x <- threshold(x)
x <- regions(x)

plot(x, region=TRUE)
```

`regions,Les-method` *regions*

Description

Estimates distinct regions of continuously elevated Λ_i above a threshold Θ.

Usage

```
regions(object, limit=NULL, minLength=2, maxGap=Inf, verbose = FALSE,
...)

## S4 method for signature 'Les'
regions(object, limit=NULL, minLength=2, maxGap=Inf,
verbose = FALSE, ...)
```

Arguments

object	Object of class 'Les' as returned by 'threshold'.
limit	Numeric specifying the threshold Θ for Λ with regions are defined as $\Lambda_i \geq \Theta$ (default: NULL). If 'NULL' Θ will be derived from the estimated number of regulated probes as computed by 'threshold'.
minLength	Integer specifying the minimum number of probes in a region (default: 2).
maxGap	Integer specifying maximum gap in base pairs between two neighboring probes in a region (default: Inf). If this value is exceeded the region will be split into smaller ones such that one region does not have neighboring probes with larger distance than 'maxGap'. If not specified arbitrary large gap sizes are allowed.
verbose	Logical indicating whether a summary of the estimated regions should be printed on screen (default: FALSE).
...	Further arguments passed to subsequent functions.

Details

This method finds distinct regions in Λ by thresholding with Θ. The regions have to meet the following criteria:

(1) For all probes in the region $\Lambda_i \geq \Theta$ has to hold. (2) Each region has to contain at least as many probes as specified in 'minLength'. (3) The gap between to neighboring probes has to be smaller or equal to 'maxGap'.

Along with the boundaries of the regions the number of regulated probes within each region is estimated. This is used to sort the regions in order to get a list of top regions. The resulting data frame containing the regions can be accessed with the '[' method.

Value

Object of class 'Les' with additionally filled slots: regions, limit

'regions' is a data frame with variables:

chr	Chromosome the regions is located on.
start	Position of the beginning of the region.
end	Position of the end of the region.
size	Extend of the region in base pairs (measured in the same units as input 'pos', normally base pairs).
nProbes	Number of probes in the region.

ri	Regulation Index of the region indicating the fraction of regulated probes in the region.
se	Standard error of the estimation of 'ri'.
rs	Regulation score defined as 'ri'/'se'. The data frame is sorted according to this variable.

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: les-package

Class: Les

Methods and functions: Les estimate threshold regions ci chi2 export plot

Examples

```
data(spikeInStat)

x <- Les(pos, pval)
x <- estimate(x, win=200)
x <- threshold(x, grenander=TRUE, verbose=TRUE)
x <- regions(x, verbose=TRUE)

print(x["regions"])
```

---

spikeInData *Spike-in ChIP-chip data set*

Description

This data set is part of a quality control study for tiling microarrays (Johnson et al., 2008), in which spike-ins were used to assess the influence of microarray platforms, preparation procedures, and analysis algorithms on the accuracy and reproducibility of ChIP-chip experiments. Here, the expression intensities of one region from the 'undiluted' data set investigated with Affymetrix arrays is selected, consisting of 452 probes and two conditions with three replicates each. The data has been normalized using quantile normalization and probe positions remapped to a common reference.

Usage

```
data(spikeInData)
```

Format

exprs Matrix with expression intensities, with rows representing probes and columns arrays. The names of the rows and columns contain the probe position and the treatment, respectively.

pval Vector with p-values assessing the differential effect between the conditions with a modified t-test. For details, please see the vignette of this package.

reference Data frame containing the location of the spike-in.

Source

GEO accession IDs: GSM248996, ..., GSM249001

References

Johnson, D. S. et al. (2008). Systematic evaluation of variability in ChIP-chip experiments using predefined DNA targets. Genome Research, 18(3):393-403.

---

spikeInStat *Spike-in ChIP-chip data set*

Description

This data set is part of a quality control study for tiling microarrays (Johnson et al., 2008), in which spike-ins were used to assess the influence of microarray platforms, preparation procedures, and analysis algorithms on the accuracy and reproducibility of ChIP-chip experiments. Here, the expression intensities of one region from the 'undiluted' data set investigated with Affymetrix arrays is selected, consisting of 452 probes and two conditions with three replicates each. The data has been normalized using quantile normalization and probe positions remapped to a common reference.

Usage

```
data(spikeInStat)
```

Format

pos Vector with probe positions.

pval Vector with p-values assessing the differential effect between the conditions with a modified t-test. For details, please see the vignette of this package.

Source

GEO accession IDs: GSM248996, ..., GSM249001

References

Johnson, D. S. et al. (2008). Systematic evaluation of variability in ChIP-chip experiments using predefined DNA targets. Genome Research, 18(3):393-403.

`threshold,Les-method` *threshold*

Description

The 'threshold' method estimates a suitable threshold Θ from the data. Thresholding provides the ability the define distinct Loci of Enhanced Significance (LES) with the 'regions' method in a later step.

Usage

```
threshold(object, grenander = FALSE, verbose = FALSE, ...)

## S4 method for signature 'Les'
threshold(object, grenander = FALSE, verbose = FALSE,
...)
```

Arguments

`object`	Object of class 'Les' as returned by 'estimate' or 'Les'.
`grenander`	Logical indicating whether the Grenander estimator for the cumulative density should be used (default: FALSE). For details see below and at the 'GSRI' package.
`verbose`	Logical indicating whether a summary of the estimated number of regulated probes should be printed on screen (default: FALSE).
`...`	Further arguments passed to subsequent functions.

Details

This method estimates the number of probes with a significant effect R. The estimation is based on the p-value distribution. The analysis is based on the 'Gene Set Regulation Index' by Bartholome et al., 2009.

Estimation of the threshold is independent of the computation performed by the 'estimate' method.

A reasonable estimate for the cutoff value Θ can be chosen such that $\mid \Lambda_i \geq \Theta \mid = R$.

The Grenander estimator for the cumulative density results in a conservative estimate for the number of significant probes with decreased variance.

A reasonable subsequent step is to call 'regions' to find distinct Loci of Enhanced Significance.

Value

Object of class 'Les' with additionally filled slots:

`nSigProbes`	Estimated number of significant probes.
`theta`	Estimated threshold Θ.

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

References

Kilian Bartholome, Clemens Kreutz, and Jens Timmer: Estimation of gene induction enables a relevance-based ranking of gene sets, Journal of Computational Biology: A Journal of Computational Molecular Cell Biology 16, no. 7 (July 2009): 959-967. http://www.liebertonline.com/doi/abs/10.1089/cmb.2008.0226

See Also

Package: les-package

Class: Les

Methods and functions: Les estimate threshold regions ci export plot

Examples

```
data(spikeInStat)

x <- Les(pos, pval)
x <- estimate(x, 200)
x <- threshold(x, verbose=TRUE)
```

---

weighting *Weighting functions*

Description

Set of functions to compute spatial weights between probes.

Usage

```
triangWeight(distance, win)
rectangWeight(distance, win)
gaussWeight(distance, win)
epWeight(distance, win)
```

Arguments

distance	Numeric vector specifying the distance of probes from the central probe. Negative values refer to probes upstream, positive values to probes downstream.
win	Integer specifying maximum size of window.

Details

The functions 'triangWeight', 'rectangWeight', 'epWeight' and 'gaussWeight' provide a triangular, rectangular, Epanechnikov and Gaussian weighting window, respectively. The weighting function can be specified by the 'weightingFunction' argument in the 'estimate' method.

This way it is also possible to use custom weighting functions. In general they have to be called the same way as the functions mentioned before and have to return a vector of weights of the same length as the argument 'distance'. For more details on how to use own weighting functions please refer to the vignette of this package.

Please note that the returned weights do not have to be normalized since this is done at the computation of the weighted cumulative density.

Value

A numeric vector with weights for each probe in the window.

Author(s)

Julian Gehring

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Package: les-package

Class: Les

Methods and functions: Les estimate threshold regions ci chi2 export plot

Examples

```
distance <- seq(-50, 50)
win <- 50

weight <- triangWeight(distance, win)
plot(distance, weight, type="l", main="triangWeight")

weight <- rectangWeight(distance, win)
plot(distance, weight, type="l", main="rectangWeight")

weight <- gaussWeight(distance, win)
plot(distance, weight, type="l", main="gaussWeight")

weight <- epWeight(distance, win)
plot(distance, weight, type="l", main="epWeight")

## simple example for a custom weighting function
ownWeighting <- function(distance, win)  {
    weight <- as.integer(abs(distance) < win)
    return(weight)
}
```

Index

∗ **IO**
  export,Les-method, 9
∗ **classes**
  Les-class, 12
∗ **datasets**
  spikeInData, 18
  spikeInStat, 19
∗ **hplot**
  plot,Les-method, 15
∗ **htest**
  chi2,Les-method, 3
  ci,Les-method, 5
  estimate,Les-method, 7
  les-package, 2
  regions,Les-method, 16
  threshold,Les-method, 20
  weighting, 21
∗ **internal**
  les-internal, 14
∗ **methods**
  chi2,Les-method, 3
  ci,Les-method, 5
  estimate,Les-method, 7
  export,Les-method, 9
  Les,ANY,numeric-method, 11
  Les-class, 12
  plot,Les-method, 15
  regions,Les-method, 16
  threshold,Les-method, 20
∗ **package**
  les-package, 2
∗ **smooth**
  weighting, 21
∗ **utilities**
  export,Les-method, 9
  Les,ANY,numeric-method, 11
  plot,Les-method, 15
[,Les-method *(*Les-class*)*, 12
[<-,Les-method *(*Les-class*)*, 12

boot, *6*

calcSingle *(*les-internal*)*, 14
cdfDuplicates *(*les-internal*)*, 14
checkState *(*les-internal*)*, 14
chi2, *3*, *5*, *6*, *9*, *10*, *12–14*, *16*, *18*, *22*
chi2 *(*chi2,Les-method*)*, 3
chi2,Les-method, 3
chi2-methods *(*chi2,Les-method*)*, 3
ci, *3*, *5*, *6*, *9*, *10*, *12–14*, *16*, *18*, *21*, *22*
ci *(*ci,Les-method*)*, 5
ci,Les *(*ci,Les-method*)*, 5
ci,Les-method, 5
ci-methods *(*ci,Les-method*)*, 5

epWeight *(*weighting*)*, 21
estimate, *3*, *5*, *6*, *9*, *10*, *12–14*, *16*, *18*, *21*, *22*
estimate *(*estimate,Les-method*)*, 7
estimate,Les-method, 7
estimate-methods *(*estimate,Les-method*)*, 7
export, *3*, *5*, *6*, *9*, *10*, *12–14*, *16*, *18*, *21*, *22*
export *(*export,Les-method*)*, 9
export,Les *(*export,Les-method*)*, 9
export,Les-method, 9
export-methods *(*export,Les-method*)*, 9
exprs *(*spikeInData*)*, 18

fdrtool, *5*
fitGsri *(*les-internal*)*, 14

gaussWeight *(*weighting*)*, 21

ind2log *(*les-internal*)*, 14
itLinReg *(*les-internal*)*, 14

Les, *3*, *5*, *6*, *9*, *10*, *12–14*, *16*, *18*, *21*, *22*
Les *(*Les,ANY,numeric-method*)*, 11
les *(*les-package*)*, 2
Les,ANY,numeric-method, 11
Les-class, 12

les-internal, 14
Les-methods *(*Les,ANY,numeric-method*)*, 11
les-package, 2
log2ind *(*les-internal*)*, 14

mcsapply *(*les-internal*)*, 14
modGrenander *(*les-internal*)*, 14

plot, *3*, *5, 6*, *9, 10*, *12–14*, *16*, *18*, *21, 22*
plot *(*plot,Les-method*)*, 15
plot,Les *(*plot,Les-method*)*, 15
plot,Les-method, 15
plot-methods *(*plot,Les-method*)*, 15
pos *(*spikeInStat*)*, 19
pval *(*spikeInStat*)*, 19

qrSlope *(*les-internal*)*, 14

rectangWeight *(*weighting*)*, 21
reference *(*spikeInData*)*, 18
regions, *3*, *5, 6*, *9, 10*, *12–14*, *16*, *18*, *21, 22*
regions *(*regions,Les-method*)*, 16
regions,Les *(*regions,Les-method*)*, 16
regions,Les-method, 16
regions-methods *(*regions,Les-method*)*, 16

seFast *(*les-internal*)*, 14
seSlopeWeight *(*les-internal*)*, 14
setState *(*les-internal*)*, 14
show,Les-method *(*Les-class*)*, 12
slopeWeight *(*les-internal*)*, 14
spikeInData, 18
spikeInStat, 19
summary,Les-method *(*Les-class*)*, 12

threshold, *3*, *5, 6*, *9, 10*, *12–14*, *16*, *18*, *21, 22*
threshold *(*threshold,Les-method*)*, 20
threshold,Les *(*threshold,Les-method*)*, 20
threshold,Les-method, 20
threshold-methods
*(*threshold,Les-method*)*, 20
triangWeight *(*weighting*)*, 21

wcdf2 *(*les-internal*)*, 14
weighting, *5*, *9*, 21