Package ‘KCsmart’

April 6, 2026

Type Package

Title Multi sample aCGH analysis package using kernel convolution

Version 2.68.0

Date 2009-11-26

Author Jorma de Ronde, Christiaan Klijn, Arno Velds

Maintainer Jorma de Ronde <j.d.ronde@nki.nl>

Description Multi sample aCGH analysis package using kernel convolution

License GPL-3

Depends siggenes, multtest, KernSmooth

Imports methods, BiocGenerics

Enhances Biobase, CGHbase

Collate 'AllClasses.R' 'access-methods.R' 'length-methods.R' 'plot-methods.R' 'show-methods.R' 'sort-methods.R' 'unlist-methods.R' 'write.table-methods.R' 'KC.R'

LazyLoad yes

biocViews CopyNumberVariation, Visualization, aCGH, Microarray

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

git_branch RELEASE_3_22

git_last_commit 2e7c515

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-04-05

Contents

KCsmart-package 2
calcSpm 3
calcSpmCollection 5
compareSpmCollection 6
compKc-class 8
compKcSigRegions-class 8
findSigLevelFdr 9

findSigLevelTrad 10
getSigRegionsCompKC 11
getSigSegments 12
hsMirrorLocs 13
hsSampleData 14
idPoints 14
KCData-class 15
KcghData-class 16
KcghDataMirror-class 16
KcghDataSplit-class 17
KcghDataSum-class 18
mmMirrorLocs 18
plot 19
plotScaleSpace 20
probeAnnotation-class 21
samplePointMatrix-class 22
sigSegments-class 23
spmCollection-class 23
write.table 24

Index **26**

`KCsmart-package` *KCsmart*

Description

Multiple sample aCGH analysis using kernel convolution

Details

Package:	KCsmart
Type:	Package
Version:	2.9.1
Date:	2011-02-21
License:	GPL

Use the wrapper function 'calcSpm' to calculate the sample point matrix. Use 'findSigLevelTrad' to find a significance threshold using permutation based testing. Use 'plot' to plot the sample point matrix or 'plotScaleSpace' to plot the significant regions over multiple scales (sigmas). Use 'getSigSegments' to retrieve the significantly gained and lost regions using specific cutoffs. To use the comparative version of KCsmart, use the 'calcSpmCollection', 'compareSpmCollection' and 'getSigRegionsCompKC' functions. See the documentation of those function for details on how to use these.

Author(s)

Jorma de Ronde, Christiaan Klijn

Maintainer: Jorma de Ronde <j.d.ronde@nki.nl>

References

Identification of cancer genes using a statistical framework for multiexperiment analysis of nondiscretized array CGH data. Nucleic Acids Res. 2008 Feb;36(2):e13.

See Also

calcSpm, findSigLevelTrad, findSigLevelFdr, plot, plotScaleSpace, getSigSegments

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)
spm4mb <- calcSpm(hsSampleData, hsMirrorLocs, sigma=4000000)

plot(spm1mb)
plot(spm1mb, chromosomes=c(1,5,6,'X'))

siglevel1mb <- findSigLevelTrad(hsSampleData, spm1mb, n=3)
siglevel4mb <- findSigLevelTrad(hsSampleData, spm4mb, n=3)

plot(spm1mb, sigLevel=siglevel1mb)

plotScaleSpace(list(spm1mb, spm4mb), list(siglevel1mb, siglevel4mb), type='g')

sigSegments1mb <- getSigSegments(spm1mb, siglevel1mb)


spmc1mb <- calcSpmCollection(hsSampleData, hsMirrorLocs, cl=c(rep(0,10),rep(1,10)))
spmcc1mb <- compareSpmCollection(spmc1mb, nperms=3)
spmcc1mbSigRegions <- getSigRegionsCompKC(spmcc1mb)

plot(spmcc1mb, sigRegions=spmcc1mbSigRegions)
```

---

calcSpm *KCsmart wrapper*

Description

Wrapper function that calculates the sample point matrix from the aCGH data

Usage

```
calcSpm(data, mirrorLocs, sigma = 1e+06, sampleDensity = 50000, maxmem = 1000, verbose=T, old=F)
```

Arguments

data	The aCGH data. Can either be in DNAcopy format or as a data.frame described in the details section
mirrorLocs	List containing the chromosome start, centromere and end positions

`sigma`	The kernel width
`sampleDensity`	The sample point matrix resolution
`maxmem`	This parameter controls memory usage, set to lower value to lower memory consumption
`verbose`	If set to false, no progress information is displayed
`old`	If set to true the old implementation of KCsmart will be used to calculate the spm

Details

'data' can be in cghRaw (CGHbase), DNAcopy or in data.frame format. When using the latter, the data.frame must have the following two columns: 'chrom' stating the chromosome the probe is located on, 'maploc' describing the position on the chromosome of the probe. The remainder of the data.frame will be interpreted as sample data points. The row names of that data will be used as probe names (when available). Important note: the data can not contain any missing values. If your data includes missing values you will need to preprocess (for example impute) it using other software solutions.

The mirror locations for Homo Sapiens and Mus Musculus are provided in the package. These can be loaded using data(hsMirrorLocs) and data(mmMirrorLocs) respectively. The 'mirrorLocs' object is a list with vectors containing the start, centromere (optional) and end of each chromosome as the list elements. Additionally it should contain an attribute 'chromNames' listing the chromosome names of each respective list element.

'sigma' defines the kernel width of the kernel used to convolute the data.

'sampleDensity' defines the resolution of the sample point matrix to be calculated. A sampleDensity of 50000 would correspond to a sample point every 50k base pairs.

'old' can be used if you want to reproduce data that was generated with old (pre 2.9.0) versions of KCsmart, for any new analyses we recommend this flag to be set to false

Value

Returns a sample point matrix object. The object has several slots of which the 'data' slot contains a list where each list item represents a chromosome. Each list item in turn contains the sample point matrix for the gains and the losses separately and an attribute specifying the corresponding chromosome. The sample point matrix contains the following additional slots: totalLength: Total length of the sample point matrix maxy and miny: Maximal and minimal score attained

The other slots just represent the parameters used to calculate the sample point matrix.

Use 'plot' to plot the sample point matrix and 'findSigLevelTrad' to find a significance threshold. 'plotScaleSpace' can be used to plot the significant regions of multiple sample point matrices (using different sigmas).

Author(s)

Jorma de Ronde

See Also

`plot`, `findSigLevelTrad`, `plotScaleSpace`

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)
spm4mb <- calcSpm(hsSampleData, hsMirrorLocs, sigma=4000000)

plot(spm1mb)
plot(spm1mb, chromosomes=c(1,5,6,'X'))
```

---

calcSpmCollection *KCsmart Comparative wrapper*

Description

Wrapper function that calculates the sample point matrix collection from the aCGH data. The sample point matrix collection is used in the comparative version of KCsmart.

Usage

```
calcSpmCollection(data, mirrorLocs, cl=NULL, data2=NULL, sigma=1000000, sampleDensity=50000, maxme
```

Arguments

data	The aCGH data. Can either be in DNAcopy format or as a data.frame described in the details section
mirrorLocs	List containing the chromosome start, centromere and end positions
cl	A class vector indicating which samples belong to which class
data2	Instead of a class vector a second data set can be provided which will be combined with the first data set into one sample point matrix collection
sigma	The kernel width
sampleDensity	The sample point matrix resolution
maxmem	This parameter controls memory usage, set to lower value to lower memory consumption
verbose	If set to false, no progress information is displayed
doChecks	If set to false, the data will not be checked for consistency
old	If set to true the old implementation of KCsmart will be used to calculate the spm

Details

The input can either consist of a single data set and a class vector or two separate datasets. In the latter case a class vector will be created assigning each data set to its own class. 'data' can be in cghRaw (CGHbase), DNAcopy or in data.frame format. When using the latter, the data.frame must have the following two columns: 'chrom' stating the chromosome the probe is located on, 'maploc' describing the position on the chromosome of the probe. The remainder of the data.frame will be interpreted as sample data points. The row names of that data will be used as probe names (when available). Important note: the data can not contain any missing values. If your data includes missing values you will need to preprocess (for example impute) it using other software solutions.

The mirror locations for Homo Sapiens and Mus Musculus are provided in the package. These can be loaded using data(hsMirrorLocs) and data(mmMirrorLocs) respectively. The 'mirrorLocs' object is a list with vectors containing the start, centromere (optional) and end of each chromosome as the list elements. Additionally it should contain an attribute 'chromNames' listing the chromosome names of each respective list element.

'sigma' defines the kernel width of the kernel used to convolute the data.

'sampleDensity' defines the resolution of the sample point matrix to be calculated. A sampleDensity of 50000 would correspond to a sample point every 50k base pairs.

'old' can be used if you want to reproduce data that was generated with old (pre 2.9.0) versions of KCsmart, for any new analyses we recommend this flag to be set to false

Value

Returns a sample point matrix collection object. The object has several slots of which the 'data' slot contains a matrix with the kernel smoothed estimates of all samples. The sample point matrix collection contains the following additional slots: cl: A class vector indicating which samples belong to which class. annotation: The annotation (containing the chromosome and position on the chromosome) for the sample points in the 'data' slot

The other slots just represent the parameters used to calculate the sample point matrix collection.

Use 'compareSpmCollection' to get a 'compKc' object for which the significant regions can be calculated using 'getSigRegionsCompKC'.

Author(s)

Jorma de Ronde

See Also

compareSpmCollection, getSigRegionsCompKC

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spmc1mb <- calcSpmCollection(hsSampleData, hsMirrorLocs, cl=c(rep(0,10),rep(1,10)))
spmcc1mb <- compareSpmCollection(spmc1mb, nperms=3)
spmcc1mbSigRegions <- getSigRegionsCompKC(spmcc1mb)

plot(spmcc1mb, sigRegions=spmcc1mbSigRegions)
```

---

compareSpmCollection *KCsmart Comparative calculate null distribution*

Description

Compare the samples of one class in the sample point matrix collection to the samples in the other class and calculate the null distribution

Usage

```
compareSpmCollection(spmCollection, nperms=20, method=c("siggenes", "perm"), siggenes.args=NULL, a
```

Arguments

spmCollection	An spmCollection object as created by the 'calcSpmCollection' function
nperms	The number of permutations to be used to calculate the null distribution
altcl	Instead of using the class vector from the spmCollection object an alternative vector can be used
method	The method to be used to calculate the null distribution
siggenes.args	Optional additional arguments to the siggenes function

Details

The method to be used to determine significant regions can either be the SAM methodology from the siggenes package or a signal-to-noise/permutation based method. For more information regarding the siggenes method please check the corresponding package.

Value

Returns a compKc object which returns the original data and, depending on the method used, the permuted data or the fdr-delta value combinations as calculated by the siggenes package.

Author(s)

Jorma de Ronde

See Also

compareSpmCollection, getSigRegionsCompKC

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spmc1mb <- calcSpmCollection(hsSampleData, hsMirrorLocs, cl=c(rep(0,10),rep(1,10)))
spmcc1mb <- compareSpmCollection(spmc1mb, nperms=3)
spmcc1mbSigRegions <- getSigRegionsCompKC(spmcc1mb)

plot(spmcc1mb, sigRegions=spmcc1mbSigRegions)
```

compKc-class *KC smart comparative*

Description

A matrix containing the results from a call to compareSpmCollection

Objects from the Class

Objects can not be created by the user directly but rather through compareSpmCollection.

Slots

`spmCollection`: The original spmCollection used to compare the samples

`method`: The method used to create the null distribution

`siggenesResult`: In case of the siggenes method being used, a siggenes object containing the fdr-cutoff table

`snrResult`: In case of the signal-noise/permutation based method being used, the signal-to-noise data and a matrix with the (class based) permutations

Methods

initialize `signature(.Object = "compKc")`: Internal use only

plot `signature(x = "compKc")`: ...

show `signature(object = "compKc")`: ...

Examples

```
showClass("compKc")
```

compKcSigRegions-class *KC smart comparative*

Description

A matrix containing the results the significant regions for a given compKc object and FDR.

Objects from the Class

Objects can not be created by the user directly but rather through getSigRegionsCompKC.

Slots

`regionTable`: The significant regions

`method`: The method used to create the null distribution

`cutoff`: The cutoff for the given false discovery rate which was used to determine the significant regions

`fdr`: The false discovery rate used to determine the significant regions

Methods

show `signature(object = "compKcSigRegions")`: ...

write.table `signature(object = "compKcSigRegions")`: ...

Examples

```
showClass("compKcSigRegions")
```

---

findSigLevelFdr *This function has not been properly implemented yet*

Description

Method to find the cutoff at which gains and losses are considered significant using permutations

Usage

```
findSigLevelFdr(data, observedSpm, n = 1, fdrTarget=0.05, maxmem=1000)
```

Arguments

`data`	aCGH data in the same format as used for 'calcSpm'
`observedSpm`	A sample point matrix as produced by 'calcSpm'
`n`	Number of permutations
`fdrTarget`	Target False Discovery Rate (FDR)
`maxmem`	This parameter controls memory usage, set to lower value to lower memory consumption

Details

The number of permutations needed for reliable results depends on the data and can not be determined beforehand. As a general rule-of-thumb around 100 permutations should be used for 'quick checks' and around 2000 permutations for more rigorous testing. The FDR method is less conservatie than the p-value based approach since instead of controlling the family wise error rate (FWER, P(false positive > 1)) it controls the false discovery rate (FDR) (false positives / total number of called data points).

Value

A list with the cutoffs corresponding to the given FDR

`pos`	The cutoff for the gains
`neg`	The cutoff for the losses'

Author(s)

Jorma de Ronde

See Also

`plotScaleSpace`

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)

sigLevel1mb <- findSigLevelTrad(hsSampleData, spm1mb, n=3)

plot(spm1mb, sigLevels=sigLevel1mb)
plotScaleSpace(list(spm1mb), list(sigLevel1mb), type='g')
```

---

findSigLevelTrad *Find significance level*

Description

Method to find the cutoff at which gains and losses are considered significant using permutations

Usage

```
findSigLevelTrad(data, observedSpm, n = 1, p = 0.05, maxmem = 1000)
```

Arguments

data	aCGH data in the same format as used for 'calcSpm'
observedSpm	A sample point matrix as produced by 'calcSpm'
n	Number of permutations
p	Alpha level for significance
maxmem	This parameter controls memory usage, set to lower value to lower memory consumption

Details

The number of permutations needed for reliable results depends on the data and can not be determined beforehand. As a general rule-of-thumb around 100 permutations should be used for 'quick checks' and around 2000 permutations for more rigorous testing.

p is the uncorrected alpha level, the method corrects for multiple testing internally using simple Bonferroni correction. See the referenced publication for more details.

Value

A list with the cutoffs corresponding to the given alpha level

pos	The cutoff for the gains
neg	The cutoff for the losses'

Author(s)

Jorma de Ronde

See Also

plotScaleSpace

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)

sigLevel1mb <- findSigLevelTrad(hsSampleData, spm1mb, n=3)

plot(spm1mb, sigLevels=sigLevel1mb)
plotScaleSpace(list(spm1mb), list(sigLevel1mb), type='g')
```

---

getSigRegionsCompKC *KCsmart Comparative calculate the signficant regions*

Description

Extract the significant regions from a compKC object for a given false discovery rate (FDR).

Usage

```
getSigRegionsCompKC(compKc, fdr=.01, maxRegionGap=10)
```

Arguments

compKc	A compKc object as created by the 'compareSpmCollection' function
fdr	The false discovery rate to be used to calculate the significantly different regions from the compKc object
maxRegionGap	The maximum number of sample points that is allowed to fall under the threshold in a continuous significant region

Details

The false discovery rate that is set is used to determine the significant regions. When the compKc object was created by the siggenes method the corresponding cutoff is looked up in the siggenes results table, otherwise it is calculated from the permuted data. The maxRegionGap determines how many sample points can be under this threshold in a continuous significant region.

Value

Returns a compKcSigRegions object that contains the significant regions for the given FDR in the 'regionTable' slot. The method used to determine the cutoff, the fdr and the cutoff itself are stored in their corresponding slots. Use 'plot' to visualize the results.

Author(s)

Jorma de Ronde

See Also

compareSpmCollection, getSigRegionsCompKC

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spmc1mb <- calcSpmCollection(hsSampleData, hsMirrorLocs, cl=c(rep(0,10),rep(1,10)))
spmcc1mb <- compareSpmCollection(spmc1mb, nperms=3)
spmcc1mbSigRegions <- getSigRegionsCompKC(spmcc1mb)

plot(spmcc1mb, sigRegions=spmcc1mbSigRegions)
```

---

getSigSegments — *Retrieve the significantly gained and lost regions including the corresponding, original probes*

Description

Retrieve the significantly gained and lost regions including the corresponding, original probes. A significance level must be selected by the user.

Usage

```
getSigSegments(spm, sigLevels, chromosomes=NULL)
```

Arguments

spm	The sample point matrix to be plotted
sigLevels	The significance thresholds to be used
chromosomes	Takes a vector of chromosomes to be plotted. Defaults to all chromosomes.

Details

'sigLevels' should contain the significance thresholds in a list with the positive (gains) threshold in the 'pos' element and the negative (losses) threshold in the 'neg' element. This is the format as returned by 'findSigLevelTrad' and 'findSigLevelFdr'.

Value

Returns a sigSegments object containing the chromosome, start position, end position, average KC score and the mode of the KC score in that region of all segments passing the thresholds as set in 'sigLevels'. Additionally, returns the IDs and indices of the probes and the positions in the sample point matrix within the significant regions. The results are stored in two separate slots: 'gains' for gains and 'losses' for losses. Use 'write.table' to save the results to file.

Author(s)

Jorma de Ronde

References

~put references to the literature/web site here ~

See Also

findSigLevelTrad, findSigLevelTrad, write.table

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)

siglevel1mb <- findSigLevelTrad(hsSampleData, spm1mb, n=3)

sigSegments1mb <- getSigSegments(spm1mb, siglevel1mb)
write.table(sigSegments1mb, file=file.path(tempdir(),'sigSegments1mb.txt'))
```

---

hsMirrorLocs *Mirror locations of the human genome*

Description

Mirror locations of the human genome, based on the NCBI 36 assembly of the human genome, for use with the KCsmart package.

Usage

```
hsMirrorLocs
```

Format

A list containing for each chromosome the start and end position and the centromere location (if a centromere is present).

Source

Ensembl

References

http://www.ensembl.org

hsSampleData — *Homo Sapiens artificial cgh data set*

Description

An artificial cgh data set, created by permuting a BAC data set consisting of 20 samples and introducing an artificial gain on 1p. To be used with the KCsmart package.

Usage

```
hsSampleData
```

Format

A data.frame containing 3268 rows and 22 columns

Source

Artificial data set

idPoints — *Identify points in sample point matrix plot*

Description

Identify points in sample point matrix plot

Usage

```
idPoints(spm, mode='pos', dev=2, chromosomes=NULL)
```

Arguments

spm	The sample point matrix object of which points are to be identified
mode	Determines which points will be identified: mode='pos' will identify points in gained regions, mode='neg' will identify points in lost regions
dev	The device on which the sample point matrix was plotted
chromosomes	If not all chromosomes contained in the sample point matrix were plotted (using the 'chromosomes' argument in the 'plot' command), the same chromosomes must be entered here as an argument

Details

Using the mouse pointer points in a sample point matrix plot can be identified by left-clicking on the to-be-identified points. Right-clicking exits the selection and returns the selected points.

Value

Returns a data.frame listing the the position and the KC score for each identified point.

`KCscore`	KCscore of the identified point
`chromosome`	Chromosome on which the identified point is located
`chromPosition`	Position on the chromosome of the identified point
`colin`	Co-linear location of the identified point (given the selected chromosomes)

Author(s)

Jorma de Ronde

See Also

`plot`

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

#spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)

#plot(spm1mb, type=1)
#idPoints(spm1mb)

#x11()
#plot(spm1mb, chromosomes=c(1,2,5))
#idPoints(spm1mb, mode='neg', dev=3, chromosomes=c(1,2,5))
```

`KCData-class` *Internal class "KCData"*

Description

Internal superclass "KCData"

Objects from the Class

The user is not meant to create instances of this class

Slots

`data`**:** Internal data

Methods

[[<- `signature(x = "KCData")`: ...

[[`signature(x = "KCData")`: ...

length `signature(x = "KCData")`: ...

unlist `signature(x = "KCData")`: ...

Warning

This class is meant for internal use only

Note

Internal class

Author(s)

Jorma de Ronde

KcghData-class *Class "KcghData"*

Description

Internal class

Slots

`probeAnnotation`: Object of class `"probeAnnotation"`
`data`: Holds aCGH data

Methods

initialize `signature(.Object = "KcghData")`: Internal use only
sort `signature(x = "KcghData")`: Internal use only

Note

For internal use only

Author(s)

Jorma de Ronde

KcghDataMirror-class *Class "KcghDataMirror"*

Description

Internal class

Slots

`mirrorLocs`: Holds mirrorLocs object
`probeAnnotation`: Object of class `"probeAnnotation"`
`pos`: Holds aCGH data for losses
`neg`: Holds aCGH data for gains
`nrSamples`: The number of samples in this analysis

Extends

Class "KcghDataSum", directly.

Methods

initialize signature(.Object = "KcghDataMirror"): For internal use only

Note

For internal use only

Author(s)

Jorma de Ronde

KcghDataSplit-class *Class "KcghDataSplit"*

Description

Internal class

Slots

probeAnnotation: Object of class "probeAnnotation"

pos: Holds aCGH data for losses

neg: Holds aCGH data for gains

Methods

initialize signature(.Object = "KcghDataSplit"): Internal use only

Note

For internal use only

Author(s)

Jorma de Ronde

KcghDataSum-class *Class "KcghDataSum"*

Description

Internal class

Slots

`probeAnnotation`: Object of class `"probeAnnotation"`

`pos`: Holds aCGH data for losses

`neg`: Holds aCGH data for gains

`nrSamples`: The number of samples in this analysis

Methods

initialize `signature(.Object = "KcghDataSum")`: For internal use only

sort `signature(x = "KcghDataSum")`: For internal use only

Note

For internal use only

Author(s)

Jorma de Ronde

mmMirrorLocs *Mirror locations of the mouse genome*

Description

Mirror locations of the mouse genome, based on the NCBI m37 mouse assembly, for use with the KCsmart package.

Usage

```
mmMirrorLocs
```

Format

A list containing for each chromosome the start and end position.

Source

Ensembl

References

http://www.ensembl.org

plot *Plot a sample point matrix*

Description

Plot the sample point matrix or parts of it

Usage

```
plot(x,y, ...)
## S4 method for signature 'scaleSpace,missing'
plot(x, y, spm, type='b', ...)
## S4 method for signature 'samplePointMatrix,missing'
plot(x, y, type="b", sigLevels=NULL, chromosomes=NULL, colinAxis=NULL, fillColor=NULL, maploc=NULL
## S4 method for signature 'compKc,missing'
plot(x, sigRegions=NULL, type="1", chromosomes=NULL, colinAxis=NULL, maploc=NULL, interpolation=1,
```

Arguments

x	either an object of class samplePointMatrix, scaleSpace or compKc
y	object of class missing
type	Determines which data is plotted. 'g' for gains only, 'l' for losses only and 'b' and '1' for both in one plot device
spm	add stuff here
sigRegions	The significant regions as calculated by the compKcSigRegions function
sigLevels	If given, the cutoffs will be drawn as lines in the plots. Optional
chromosomes	Takes a vector of chromosomes to be plotted. Defaults to all chromosomes.
colinAxis	Allows you to override default behaviour of axis labeling. Choose False for genomic position labeling for each individual chromosome, True for colinear labeling.
fillColor	Allows you to choose the colors used to fill the significant areas under the curve. Takes a list with the 'pos' element giving the color for the gains and the 'neg' element the color for the losses.
maploc	Currently not in use
interpolation	Determines which points from the sample point matrix will actually be plotted. If the value of 'interpolation' is n, then every n-th point will be plotted. The default value of 1 will results in all points being plotted. This can be useful when a high density sample point matrix results in big file size when exporting the image (especially to pdf or eps format).
main	Set the title of the plot
col	Set the color of the plotted lines
col1	Set the color of the plotted lines
col2	Set the color of the plotted lines
ylim	Set the y-axis limits
add	When set to true the plot is added to the current plot device
...	Any other parameters you would like to pass to 'plot'. See 'par' for more details.

Value

Plots the sample point matrix. The gains and the losses are plotted separately. The KC normalized score is plotted on the y-axis, the genomic position on the x-axis. If centromeres are present these are represented by dotted, lightblue lines. Setting type to 'b' or to '1' will both make the plot appear in one plot device, '1' will plot the gains and the losses in one plot, 'b' will plot the gains and losses separately. Using the 'add' flag it is possible to add a plot to the current plot device. The 'col' and 'ylim' arguments can be used to set the color of each plot and the plot regions. The function 'idPoints' can be used to identify points in the sample point matrix plot. See the corresponding documentation for details.

In case of plotting a compKc object, col1 and col2 can be used to set the colors of the group 1 and group 2 mean values respectively.

Author(s)

Jorma de Ronde

See Also

calcSpm, plotScaleSpace, idPoints

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)

plot(spm1mb)
plot(spm1mb, interpolation=10)
plot(spm1mb, chromosomes=c(1,4,'X'))

siglevel1mb <- findSigLevelTrad(hsSampleData, spm1mb, n=3)
plot(spm1mb, chromosomes=c(1,4,'X'), sigLevels=siglevel1mb)
plot(spm1mb, chromosomes=c(1,4,'X'), sigLevels=siglevel1mb, fillColor=list(pos='darkred',neg='darkgreen'))
```

---

plotScaleSpace *Plot multiple significant regions in one figure*

Description

Plots significant regions in different scale spaces in one figure

Usage

```
plotScaleSpace(spms, sigLevels, chromosomes=NULL, type='b')
```

Arguments

spms	List of sample point matrices
sigLevels	List of significance levels
chromosomes	Takes a vector of chromosomes to be plotted. Defaults to all chromosomes.
type	Determines which data is plotted. 'g' for gains only, 'l' for losses only and 'b' for both. When type='b' is used, two devices (x11) will be opened.

Details

Takes sample point matrices that were calculated using (different) kernel widths (sigma), then calculates the significant regions given the cutoffs as defined by 'sigLevels' and plots these in one figure.

Value

Depending on the 'type' parameter, produces one or two plots, one for the gains and one for the losses. The heatmap color indicates the level of the gain or loss.

Author(s)

Jorma de Ronde

See Also

plot

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)
spm4mb <- calcSpm(hsSampleData, hsMirrorLocs, sigma=4000000)

siglevel1mb <- findSigLevelTrad(hsSampleData, spm1mb, n=3)
siglevel4mb <- findSigLevelTrad(hsSampleData, spm4mb, n=3)

plotScaleSpace(list(spm1mb, spm4mb), list(siglevel1mb, siglevel4mb), type='g')
```

---

probeAnnotation-class *Class "probeAnnotation"*

Description

Holds the probe annotation

Objects from the Class

Instances of this class are not meant to be created by the user

Slots

`chromosome`: Chromosome on which the probe is located

`maploc`: Location of the probe on the chromosome

`name`: Probe name

Methods

[`signature(x = "probeAnnotation")`: Access information about a probe

initialize `signature(.Object = "probeAnnotation")`: Internal use only

Author(s)

Jorma de Ronde

---

samplePointMatrix-class

Sample point matrix

Description

A sample point matrix resulting from a call to calcSpm

Objects from the Class

Objects can not be created by the user directly but rather through calcSpm.

Slots

`totalLength`: The total length of the sample point matrix, measures in sample points

`maxy`: The maximum KC score attained over the sample point matrix

`miny`: The minimum KC score attained over the sample point matrix

`sampleDensity`: The sample density used to calculate the sample point matrix. ie the distance between two points in the sample point matrix, measured in base pairs.

`sigma`: The sigma used for the kernel to calculate the sample point matrix.

`mirrorLocs`: The mirror locations list used to calculate the sample point matrix

`probeAnnotation`: The original probe annotation from the input data.

`data`: The sample point matrix data points in the form of a list where each list element represents a chromosome.

Methods

plot `signature(x = "samplePointMatrix")`: ...

show `signature(object = "samplePointMatrix")`: ...

Examples

```
showClass("samplePointMatrix")
```

sigSegments-class *Significant segments*

Description

Lists the significant segments found in a given sample point matrix using a given significance level

Objects from the Class

Objects can not be created by the user directly but rather through getSigSegments.

Slots

`gains`: Gained segments

`losses`: Lost segments

`sigma`: The sigma used for the kernel to calculate the sample point matrix.

`sigLevels`: The significance levels at which significant segments are calculated

Methods

show `signature(object = "sigSegments")`: ...

write.table `signature(x = "sigSegments")`: ...

Examples

```
showClass("sigSegments")
```

spmCollection-class *Sample point matrix collection*

Description

A sample point matrix collection resulting from a call to calcSpmCollection

Objects from the Class

Objects can not be created by the user directly but rather through calcSpmCollection.

Slots

`annotation`: The annotation (containing the chromosome and position on the chromosome) for the sample points in the 'data' slot

`data`: A matrix with the kernel smoothed estimates of all samples

`cl`: A class vector indicating which samples belong to which class

`sampleDensity`: The sample density used to calculate the sample point matrix. ie the distance between two points in the sample point matrix, measured in base pairs.

`sigma`: The sigma used for the kernel to calculate the sample point matrix.

`mirrorLocs`: The mirror locations list used to calculate the sample point matrix

Methods

show `signature(object = "spmCollection")`: ...

Examples

```
showClass("spmCollection")
```

`write.table` *Write summary of the significant regions to a table*

Description

Write summary of the significant regions to a table

Usage

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
                 eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"), fileEncoding = "")
## S4 method for signature 'sigSegments'
write.table(x, file="", append = FALSE, quote = 7, sep = "\t", eol = "\n", na = "NA", dec = ".", row.na
## S4 method for signature 'compKcSigRegions'
write.table(x, file="", append = FALSE, quote = 7, sep = "\t", eol = "\n", na = "NA", dec = ".", row.na
```

Arguments

`x`	The sigSegments object to be summarized
`file`	either a character string naming a file or a connection open for writing. '""' indicates output to the console.
`append`	logical. Only relevant if 'file' is a character string. If 'TRUE', the output is appended to the file. If 'FALSE', any existing file of the name is destroyed.
`quote`	a logical value ('TRUE' or 'FALSE') or a numeric vector. If 'TRUE', any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of columns to quote. In both cases, row and column names are quoted if they are written. If 'FALSE', nothing is quoted.
`sep`	the field separator string. Values within each row of 'x' are separated by this string.
`eol`	the character(s) to print at the end of each line (row).
`na`	the string to use for missing values in the data.
`dec`	the string to use for decimal points in numeric or complex columns: must be a single character.
`row.names`	either a logical value indicating whether the row names of 'x' are to be written along with 'x', or a character vector of row names to be written.
`col.names`	either a logical value indicating whether the column names of 'x' are to be written along with 'x', or a character vector of column names to be written. See the section on 'CSV files' for the meaning of 'col.names = NA'.

`qmethod` a character string specifying how to deal with embedded double quote characters when quoting strings. Must be one of '"escape"' (default), in which case the quote character is escaped in C style by a backslash, or '"double"', in which case it is doubled. You can specify just the initial letter.

`fileEncoding` character string: if non-empty declares the encoding to be used on a file (not a connection) so the character data can be re-encoded as they are written. See `file`.

Details

Writes a summary of the sigSegments object to file. The resulting table contains 7 columns. The interpretation of the columns is as follows:

- Status Either 'L' for loss or 'G' for gain
- Chromosome The chromosome on which this segment is located
- Start The start position (in base pairs) of the segment on the chromosome
- End The end position of the segment on the chromosome
- Average KC scoreThe average KCsmart score over all base pairs in this segment
- Mode KC scoreThe highest (for gains) or lowest (for losses) KCsmart score over all base pairs in this segment
- Probes All probes from the original data that fall into this segment

Author(s)

Jorma de Ronde

See Also

`calcSpm`, `getSigSegments`

Examples

```
data(hsSampleData)
data(hsMirrorLocs)

spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)

siglevel1mb <- findSigLevelTrad(hsSampleData, spm1mb, n=3)

sigSegments1mb <- getSigSegments(spm1mb, siglevel1mb)
write.table(sigSegments1mb, file=file.path(tempdir(),'sigSegments1mb.txt'))
```

Index

∗ **IO**
write.table, 24
∗ **classes**
compKc-class, 8
compKcSigRegions-class, 8
KCData-class, 15
KcghData-class, 16
KcghDataMirror-class, 16
KcghDataSplit-class, 17
KcghDataSum-class, 18
probeAnnotation-class, 21
samplePointMatrix-class, 22
sigSegments-class, 23
spmCollection-class, 23
∗ **datasets**
hsMirrorLocs, 13
hsSampleData, 14
mmMirrorLocs, 18
∗ **file**
write.table, 24
∗ **hplot**
plot, 19
plotScaleSpace, 20
∗ **iplot**
idPoints, 14
∗ **manip**
calcSpm, 3
calcSpmCollection, 5
compareSpmCollection, 6
findSigLevelFdr, 9
findSigLevelTrad, 10
getSigRegionsCompKC, 11
getSigSegments, 12
∗ **package**
KCsmart-package, 2
[,probeAnnotation-method (probeAnnotation-class), 21
[[,KCData-method (KCData-class), 15
[[<-,KCData-method (KCData-class), 15

calcSpm, *3*, 3, *20*, *25*
calcSpmCollection, 5
compareSpmCollection, *6*, 6, *7*, *12*
compKc-class, 8
compKcSigRegions-class, 8

file, *25*
findSigLevelFdr, *3*, 9
findSigLevelTrad, *3*, *4*, 10, *13*

getSigRegionsCompKC, *6*, *7*, 11, *12*
getSigSegments, *3*, 12, *25*

hsMirrorLocs, 13
hsSampleData, 14

idPoints, 14, *20*
initialize,compKc-method (compKc-class), 8
initialize,KcghData-method (KcghData-class), 16
initialize,KcghDataMirror-method (KcghDataMirror-class), 16
initialize,KcghDataSplit-method (KcghDataSplit-class), 17
initialize,KcghDataSum-method (KcghDataSum-class), 18
initialize,probeAnnotation-method (probeAnnotation-class), 21

KCData-class, 15
KcghData-class, 16
KcghDataMirror-class, 16
KcghDataSplit-class, 17
KcghDataSum, *17*
KcghDataSum-class, 18
KCsmart (KCsmart-package), 2
KCsmart-package, 2

length,KCData-method (KCData-class), 15

mmMirrorLocs, 18

plot, *3*, *4*, *15*, 19, *21*
plot,compKc,missing-method (plot), 19
plot,compKc-method (compKc-class), 8
plot,samplePointMatrix,missing-method (plot), 19

plot,scaleSpace,missing-method (plot), 19
plotScaleSpace, *3*, *4*, *9*, *11*, *20*, 20
probeAnnotation-class, 21

samplePointMatrix-class, 22
show,compKc-method (compKc-class), 8
show,compKcSigRegions-method (compKcSigRegions-class), 8
show,samplePointMatrix-method (samplePointMatrix-class), 22
show,sigSegments-method (sigSegments-class), 23
show,spmCollection-method (spmCollection-class), 23
sigSegments-class, 23
sort,KcghData-method (KcghData-class), 16
sort,KcghDataSum-method (KcghDataSum-class), 18
spmCollection-class, 23

unlist,KCData-method (KCData-class), 15

write.table, *13*, 24
write.table,compKcSigRegions-method (write.table), 24
write.table,sigSegments-method (write.table), 24