tilingArray

November 11, 2009

R topics documented:

`breakpointsPretend` *Accessor methods for breakpointsPretend objects - not to be called by the user.*

Description

Accessor methods for breakpointsPretend objects - not to be called by the user.

These functions are used in the interface between the `segmentation` class and the `confint.breakpointsfull` method of the strucchange package. This method calls `breakpoints` and `residuals` methods for its first argument, and since we pass an argument of S3 class `breakpointsPretend`, we can avoid the overhead of the corresponding methods for `breakpointsfull` objects. These functions are of no interest to the user.

Usage

```
residuals.breakpointsPretend(object, breaks, ...)
breakpoints.breakpointsPretend(obj, breaks, ...)
```

Arguments

`object`	a breakpointsPretend object.
`obj`	a breakpointsPretend object.
`breaks`	dummy argument, is ignored.
`...`	futher arguments.

Value

residuals and breakpoints.

Author(s)

W. Huber ⟨huber@ebi.ac.uk⟩

`segmentation` *The class segmentation represents a segmentation result.*

Description

This class represents the result of a segmentation, usually a call to the function `segment`.

Objects from the Class

Objects can be created by calls of the function `segment` or by calls of the form `new("segmentation", ...)`.

Slots

`y`: A matrix with the data (the dependent variable(s)), see `segment`.

`x`: A numeric vector with the regressor variable. The length of this vector must be either the same as `nrow(y)`, or 0. The latter case is equivalent to `x=1:nrow(y)`.

`flag`: An integer vector, whose length must be either the same as `nrow(y)`, or 0. This can be used to *flag* certain probes for special treatment, for example by `plotAlongChrom`.

`breakpoints`: List of segmentations. The element `breakpoints[[j]]` corresponds to a segmentation fit of `j` segments, i.e. with `j-1` breakpoints. It is a matrix with `(j-1)` rows and 1 or 3 columns. It always contains a column named `estimate` with the point estimates. Optionally, it may contain columns `lower` and `upper` with the confidence intervals. The point estimates are the row indices in `y` where new segments start, for example: let `z=breakpoints[[j]]`, then the first segment is from row `1` to `z[1, "estimate"]-1`, the second from row `z[1, "estimate"]` to `z[2, "estimate"]-1`, and so on.

`negloglik`: Numeric vector of the same length as `breakpoints`. The negative log-likelihood of the piecewise constant models under the data `y`.

hasConfint: Logical vector of the same length as `breakpoints`. TRUE if the confidence interval estimates are present, i.e. if the matrix `breakpoints[[j]]` has columns `lower` and `upper`.

nrSegments: A scalar integer, value must be either `NA` or between `1` and `length(breakpoints)`. Can be used to select one of the fits in `breakpoints` for special treatment, for example by `plotAlongChrom`.

Methods

confint The method `confint(object, parm, level=0.95, het.reg=FALSE, het.err=FALSE, ...)` computes confidence intervals for the change point estimates of the segmentation. Typically, these were obtained from a previous call to the function `segment` that created the object. This is just a wrapper for the function `confint.breakpointsfull` from the `strucchange` package, which does all the hard computations. Parameters: `object` an object of class `segmentation`, `parm` an integer vector, it determines for which of the segmentation fits confidence intervals are computed. See also `segment`. The other parameters are directly passed on to `confint.breakpointsfull`.

logLik The method `logLik(object, penalty="none", ...)` returns the log-likelihoods of fitted models. Valid values for the argument `penalty` are `none`, `AIC` and `BIC`.

plot The method `plot(x, y, xlim, xlab="x", ylab="y", bpcol="black", bplty=1, pch=16, ...)` provides a simple visualization of the result of a segmentation. Parameters: `x` an object of class `segmentation`, `y` an integer between `1` and `length(x@breakpoints)`, selecting which of the fits contained in `x` to plot, `bpcol` and `bplty` color and line type of breakpoints. The plot shows the numeric data along with breakpoints and if available their confidence intervals.

show summary.

Author(s)

Wolfgang Huber ⟨huber@ebi.ac.uk⟩

See Also

`segment`

Examples

```
## generate random data with 5 segments:
y = unlist(lapply(c(0,3,0.5,1.5,5), function(m) rnorm(10, mean=m)))

seg = segment(y, maxseg=10, maxk=15)
seg = confint(seg, parm=c(3,4,5))

if(interactive())
  plot(seg, 5)

show(seg)
```

`comparisonPlot` *Plot a vertical layout of panels for the comparison of different along-chromosome profiles.*

Description

This function is used for Figure 5 in the David et al. (PNAS 2006) paper and in the Huber et al. methods paper.

Usage

```
comparisonPlot(x, y, xscale=range(x), yscale, anno, ticks, pch=20, cex=1, bgcol=
```

Arguments

`x`	numeric vector.
`y`	list of numeric vector, each of same length as `x`.
`xscale`	numeric vector of length 2.
`yscale`	matrix with 2 rows and columns corresponding to the elements of `x`.
`anno`	dataframe with columns `start`, `end`, and `name`, each row corresponds to one gene CDS to be plotted at the bottom.
`ticks`	numeric vector, where to plot the ticks.
`pch`	A numeric or character vector indicating what sort of plotting symbol to use, see `grid.points`.
`cex`	Multiplier applied to fontsize, see `gpar`.
`bgcol`	Color to use as background for some of the plot panels.

Value

Function is called for its side-effect.

Author(s)

W. Huber ⟨huber@ebi.ac.uk⟩

References

...

Examples

```
##
```

costMatrix *Segmentation cost matrix*

Description

This function calculates the cost matrix for the segmentation model

Usage

```
costMatrix(x, maxk)
```

Arguments

x	Numeric vector of length n or matrix with n rows and d columns, where n is the number of sample points and d the number of replicate measurements (e.g. from multiple arrays).
maxk	Positive integer.

Details

See the package vignette *Calculation of the cost matrix*.

Value

Matrix with maxk rows and length(x) columns.

Author(s)

W. Huber

Examples

```
d = 4
x = apply(matrix(rnorm(200), ncol=d), 2, cumsum)
maxk = 50

G = costMatrix(x, maxk=maxk)

G.pedestrian = matrix(NA, nrow=nrow(G), ncol=ncol(G))
for(i in 1:(ncol(G)))
  for(k in 1:min(nrow(G), nrow(x)-i+1))
    G.pedestrian[k, i] = (k*d-1)*var(as.vector(x[i:(i+k-1), ]))

stopifnot(identical(is.na(G), is.na(G.pedestrian)))
stopifnot(max(abs(G-G.pedestrian), na.rm=TRUE) <= 1e-6)
```

`findSegments` *Fit a piecewise constant curve to a sequence of numbers – OBSOLETE, please use function segment instead.*

Description

This function is only here for backward compatibility - please use `segment`.

The function fits a piecewise constant curve to a sequence of numbers using a simple least squares cost function and the dynamic programming algorithm described by Picard et al. (see reference).

Usage

```
findSegments(x, maxcp, maxk, verbose=TRUE)
```

Arguments

`x`	Numeric (real) vector.
`maxcp`	Integer (length 1): maximum number of segments (= 1 + maximum number of change points).
`maxk`	Integer (length 1): maximum length of a segment.
`verbose`	Logical: if this parameter has a positive value, various diagnostic output is printed.

Details

The complexity of the algorithm is `length(x)*maxk` in memory and `length(x)*maxk*maxcp` in time.

Value

An object of class `"segmentation"` A list with elements

`J`	likelihood criterion
`th`	matrix of segment start points
`dat`	the data used for the segmentation
`call`	the function call

.

See the vignette, and the paper cited below for details.

Note

This function is depracated and replaced by function `segment`, but still included for backward compability.

Author(s)

W. Huber ⟨huber@ebi.ac.uk⟩, Joern Toedling ⟨toedling@ebi.ac.uk⟩

References

A statistical approach for CGH microarray data analysis. Franck Picard, Stephane Robin, Marc Lavielle, Christian Vaisse, Gilles Celeux, Jean-Jacques Daudin, Rapport de recherche No. 5139, Mars 2004, Institut National de Recherche en Informatique et en Automatique (INRIA), ISSN 0249-6399. The code of this function is based on the Matlab implementation presented at http://www.inapg.fr/ens_rech/mathinfo/recherche/mathematique/outil.html, but it has evolved.

Examples

```
x = rep( sin((0:4)/2*pi), each=3) + rnorm(3*5, sd=0.1)
res = findSegments(x, maxcp=6, maxk=15)
```

`normalizeByReference`

Probe-specific normalization of hybridization intensities from an oligonucleotide microarray

---

Description

Adjust the hybridization intensities from an oligonucleotide microarray for probe-specific response effect by using one or several reference hybridizations. If `x` contains more than one array, `vsnMatrix` from the `vsn` package is called for between array normalization.

Usage

```
normalizeByReference(x, reference, pm, background, refSig, nrStrata=10,
  cutoffQuantile=0.05, plotFileNames, verbose=FALSE)
```

Arguments

`x`	ExpressionSet containing the data to be normalized.
`reference`	ExpressionSet with the same number of features as `x`, containing the reference signal, on the raw scale (non-logarithmic). This argument can be used to directly input the data from a set of replicate DNA hybridizations. Alternatively, the argument `refSig` can be specified.
`pm`	Indices specifying the perfect match features in `reference` (see Details). This can be either an integer vector with values between 1 and `nrow(exprs(reference))` or a logical vector.
`background`	Indices specifying a set of background features in `x` (see Details). This can be either an integer vector with values between 1 and `nrow(exprs(x))` or a logical vector.
`refSig`	A numeric vector of the same length as `pm` with estimates of probe response effects, on a logarithm-like scale. This argument can be specified alternatively to `reference`.
`nrStrata`	Integer (length 1), number of strata for the estimation of the background function.
`cutoffQuantile`	Numeric (length 1), the probes whose reference signal is below this quantile are thrown out.

plotFileNames
: Character vector whose length is the same as the number of arrays in x. Optional, if missing, no plots are produced.

verbose
: Logical of length 1, if TRUE, some messages about progress are printed.

Details

The intensities in x are adjusted according to the reference values. Typically, the reference values are obtained by hybridizing a DNA sample to the array, so that the abundance of target is the same for all reference probes, and their signal can be used to estimate the probe sequence effect. A reference probe is a probe that perfectly matches the target genome exactly once. Usually, not all probes on a chip are reference probes, hence the subset of those that are is specified by the argument pm.

The background signal is estimated from the probes indicated by the argument background. They need to be a strict subset of the reference probes. I.e., they need to uniquely match the target organism's DNA, but are not expected to match any of its transcripts. A robust estimation method is used, so a small fraction of background probes that do hit transcripts is not harmful.

A limitation of this normalization method is that it only makes sense for the data from reference probes, NA values are returned for all other probes.

The functions PMindex and BGindex can be used to produce the pm and background arguments from a probeAnno environment such as provided in the davidTiling package.

To summarize, a reference probe (indicated by argument pm) is a probe that perfectly matches the target genome exactly once, a background probe (indicated by argument background) is a reference probe which we expect not to be transcribed. These should not be confused with what is called 'perfect match' and 'mismatch' probes in Affymetrix annotation.

Value

A copy of x with the normalized intensities.

Author(s)

W. Huber ⟨huber@ebi.ac.uk⟩

References

Huber W, Toedling J, Steinmetz, L. Transcript mapping with high-density oligonucleotide tiling arrays. Bioinformatics 22, 1963-1970 (2006).

See Also

PMindex, BGindex

Examples

```
## see vignette assessNorm.Rnw in inst/scripts directory
```

otherStrand *Return the name of the opposite strand*

Description

Return the name of the opposite strand

Usage

```
otherStrand(x)
```

Arguments

x	Character vector whose elements are "+" or "-".

Details

This is a rather trivial convenience function.

An alternative would be to code strands with integers `-1` and `+1`, in which case the inversion would be a trivial builtin operation. However, many genomic databases and input data files use the character string / factor notation.

Value

Character vector of same length as `x`, with strands reversed.

Author(s)

W. Huber <huber@ebi.ac.uk>

Examples

```
otherStrand(c("+", "-"))
```

plotAlongChromLegend *Plot a legend for genomic features*

Description

Plot a legend for genomic features

Usage

```
plotAlongChromLegend(vpr, nr=2,
    featureColorScheme=1,
    featureExclude=c("chromosome", "nucleotide_match", "insertion"),
    mainLegend, cexLegend=0.35, cexMain=1)
```

Arguments

vpr	vector specifying where to place the legend in figure (set up by using the viewport function from the grid package. When this function is called directly by the user this argument should be left missing.
nr	numeric scalar, specifying the number of rows to plot legend over (default value is 2).
featureColorScheme	numeric scalar, used to select a color scheme for the boxes representing genomic features such as coding sequences, ncRNAs etc. Currently the only value supported is 1.
featureExclude	character vector of names of feature types (in gff) that should not be plotted. Default is "chromosome", "nucleotide_match" and "insertion". Additional possible candidates include: "ARS", "repeat_region", "repeat_family" and "nc_primary_transcript".
mainLegend	character vector specifying legend title.
cexLegend	numeric scalar specifying the magnification to be used for the legend text relative to the current text size.
cexMain	numeric scalar specifying the magnification to be used for the legend title relative to the current text size.

Details

This function is usually called by plotAlongChrom when doLegend is TRUE. It can also be called directly by the user to produce a separate legend.

The following features are included in the legend (unless excluded using the featuredExclude option): "chromosome", "nucleotide_match", "pseudogene", "uORF", "nc_primary_transcript" "region", "repeat_family", "repeat_region", "transposable_element", "transposable_ele "ARS", "centromere", "telomere", "insertion", "CDS", "CDS_dubious", "ncRNA", "tRNA", "snRNA", "rRNA", "snoRNA", "binding_site" and "TF_binding_site".

Author(s)

Wolfgang Huber <huber@ebi.ac.uk>

Examples

```
## plotAlongChromLegend(mainLegend="Legend")
```

---

plotAlongChrom *Plot signals and segmentation for a region of a chromosome*

Description

Plot signals and segmentation for a region of a chromosome

Usage

```
plotAlongChrom(segObj, y, probeAnno, gff,
    isDirectHybe=FALSE,
    what = c("dots"), ## "heatmap"
    chr, coord, highlight,
    colors, doLegend=FALSE,
    featureExclude=c("chromosome", "nucleotide_match", "insertion"),
    featureColorScheme=1, extras,
    rowNamesHeatmap, rowNamesExtras, ylab, ylabExtras, main,
    colHeatmap=colorRamp(brewer.pal(9, "YlGnBu")),
    colExtras=colorRamp(brewer.pal(9, "Reds")),
    sepPlots=FALSE, reOrder=TRUE, ...)
```

Arguments

segObj	Either an environment or an object of S4 class segmentation. See *Details*.
y	a numeric vector or matrix containing the signal to be plotted. See *Details*.
probeAnno	environment with probe annotations. See *Details*, and package davidTiling for an example.
gff	data frame with genome annotation from the GFF file.
isDirectHybe	logical scalar: if TRUE, the mapping of probes to genomic strands is reversed with respect to the default. This is appropriate for data from a direct RNA hybridization that used no reverse transcription.
what	character scalar indicating which signal visualization to plot. Can be either dots to plot each probe intensity with a point, or heatmap to produce a colorscale representation of the intesities.
chr	integer of length 1 indicating the chromosome number to plot.
coord	integer vector of length 2 containing the start and end coordinates (in bases) for the plot.
highlight	(optional) list with two elements: a single numeric value coord and a character strand. If present, this position is marked by a vertical red bar on the coordinate axis. The color can be changed using the colors argument below.
colors	(optional) named character vector. If missing, a default color scheme is used: c("+"="#00441b", "-"="#081d58", "duplicated"="grey", "cp"="#101010" "highlight"="red", "threshold"="grey"), where the first three elements refer to the colors of data points and the last three to the colors of lines in the plot.
doLegend	logical: should the plot contain a legend?
featureExclude	character vector of names of feature types (in gff) that should not be plotted. Default is "chromosome", "nucleotide_match" and "insertion". Additional possibilities include: "ARS", "repeat_region", "repeat_family" and "nc_primary_transcript".
featureColorScheme	numeric scalar, used to select a color scheme for the boxes representing genomic features such as coding sequences, ncRNAs etc. Currently the only value supported is 1 (see plotAlongChromLegend or plotFeatures for further information).

extras	a matrix containing additional values to be plotted along the chromosome in a separate panel (such as p-values). This option is only available when y is specified. These values should be on the scale [0,1].
rowNamesHeatmap	character vector of row names for the main heatmap.
rowNamesExtras	character vector of row names for the extra heatmap.
ylab	character label for y-axis of main plot.
ylabExtras	character label for y-axis on extras panel (if specified).
main	character: plot title.
colHeatmap	function describing color scheme for the main heatmap plot (defaults to YlGnBu from RColorBrewer package).
colExtras	function describing color scheme for the extra heatmap plot (if specified) (defaults to Reds from RColorBrewer package).
sepPlots	logical scalar. If TRUE, each column of intensities in segObj or y is plotted separately (maximum of 3) in the same figure. When FALSE, the average is plotted. This argument is only used when what is set to dots.
reOrder	logical scalar (only used when sepPlots is TRUE). If TRUE, the first column of intensities is printed at the bottom of each plot, and the subsequent columns are plotted above. If FALSE, the first appears at the top, and the subsequent columns are plotted below.
...	further arguments that can be passed to the functions that implement the what option above (see plotSegmentationDots and plotSegmentationHeatmap) or gff plotting (see plotFeatures and plotAlongChromLegend).

Details

Intensities: There are two alternative, mutually exclusive ways of providing the intensities that are to be plotted to this function.

1. Via the parameters y and probeAnno. In this case, y is a matrix of intensities, whose rows correspond to probes on the array, and its columns to different conditions, time points, etc. It is also acceptable that y is provided as a vector, in which case it is converted to an nrow(y) x 1 matrix. probeAnno is an environment whose elements correspond to target sequences (e.g. chromosome strands) and that contain integer vectors of length nrow(y) with information about the probes: start and end positions of their alignment to the target sequence, their row indices in y, the type of alignment (is it perfect? is is unique?). For example, the start positions and indices of probes for the + strand of chromosome 1 would be described by environment elements "1.+.start" and "1.+.index".
2. Via the parameter segObj.

segObj: This can be either an object of S4 class segmentation or an environment that by convention contains a certain set of objects. Future work on this package will focus on the S4 class segmentation. The environment option is provided for backward compatibility.

Explanation of the environment: the intended workflow is as follows: Use the script segment.R (in the inst/scripts directory of this package) to generate segmentations. This can be run in parallel on several processors, separately for each chromosome and strand. The results of this are stored in files of the name 1.+.rda, 1.-.rda, 2.+.rda, and so forth, typically within a dedicated directory. Then use the script readSegments.R to collect the R objects in these .rda files into the environment. It contains three types of data:

- microarray intensities in along-chromosome order.
- the segmentation objects (output of findSegments).
- a dataframe named `segScore` with segment scores; it can be missing iff `nrBasesPerSeg` is present,
- a numeric scalar names `theThreshold`, which is used to draw a horizontal "threshold" line in the plot.

... and the different signal visualization methods (`what` option): If `what=="dots"`, the argument `showConfidenceIntervals` can be a logical scalar to choose whether vertical dashed lines are drawn for the confidence interval. In any case, these are only drawn if they are present in the `segmentation` object in `segObj`.

Author(s)

Wolfgang Huber <huber@ebi.ac.uk>

Examples

```
## 1. see viewSegmentation.R script in the inst/scripts directory
## 2. (newer): segmentation.Rnw
```

---

plotFeatures *Plot genomic features for a region along a chromosome*

Description

Plot genomic features for a region along a chromosome

Usage

```
plotFeatures(gff, chr, xlim, strand, vpr, featureColorScheme=1,
             featureExclude=c("chromosome", "nucleotide_match", "insertion"),
             featureNoLabel=c("uORF", "CDS"), ...)
```

Arguments

`gff`	data frame with genome annotation from the GFF file.
`chr`	integer of length 1 specifying the chromosome to plot the features for.
`xlim`	integer of length 2 with start and end coordinates (in bases) for plotting.
`strand`	character scalar which should be set to either + or − to indicate which strand of DNA to plot the features from.
`vpr`	which viewport to plot the features in.
`featureColorScheme`	numeric scalar, used to select a color scheme for the boxes representing genomic features such as coding sequences, ncRNAs etc. Currently the only value supported is 1.

`featureExclude`
: character vector of names of feature types (in gff) that should not be plotted. Default is `"chromosome"`, `"nucleotide_match"` and `"insertion"`. Additional possible candidates include: `"ARS"`, `"repeat_region"`, `"repeat_family"` and `"nc_primary_transcript"`.

`featureNoLabel`
: character vector, names of feature types (in gff) that should not be labelled with their names (if they are plotted).

`...`
: additional arguments.

Details

This function is called by `plotAlongChrom` when the `gff` argument has been specified. It should not be called directly by the user.

Author(s)

Wolfgang Huber <huber@ebi.ac.uk>

---

plotPenLL *Plot the log-likelihood and penalized log-likelihoods (AIC, BIC)*

Description

Plot the log-likelihood and two versions of penalized log-likelihoods (AIC, BIC) for a segmentation object.

Usage

```
plotPenLL(seg, extrabar=numeric(0), type="b", lty=1, pch=16, lwd=2, ...)
```

Arguments

`seg`
: A `segmentation` object.

`extrabar`
: In addition to the location of maximmal BIC, vertical bars are drawn at these x-positions as well.

`type, pch, lty, lwd, ...`
: Get passed on to `matplot`.

Details

This function is used in the vignette: *How to use the segment function to fit a piecewise constant curve.*

Value

The function is called for its side effect, which is creating a plot in the current graphics device.

Author(s)

Wolfgang Huber <huber@ebi.ac.uk>

Examples

```
x = rep( sin((0:4)/2*pi), each=3) + rnorm(3*5, sd=0.1)
res = segment(x, maxseg=8, maxk=15)
plotPenLL(res)
```

`plotSegmentationDots` *Plot points for a region along a chromosome*

Description

Plot points for a region along a chromosome

Usage

```
plotSegmentationDots(dat, xlim, ylim, ylab, threshold=NA,
                     chr=1, strand="+", vpr, colors, main,
                     pointSize=unit(0.6, "mm"), showConfidenceIntervals=TRUE,
                     sepPlots=FALSE, cexAxisLabel=1, cexAxis=1,...)
```

Arguments

`dat`	list containing data to be plotted (see *Details* section below for particulars).
`xlim`	integer vector of length 2 with start and end coordinates (in bases) for plotting.
`ylim`	numeric vector containing the y limits of the plot.
`ylab`	character scalar (if `sepPlots=FALSE`) or vector containing y-axis label(s).
`threshold`	numeric scalar indicating the threshold of expression (default value is NA, for no threshold. If a value is supplied, it is subtracted from the intensity measures in `dat$y`.
`chr`	integer of length 1 indicating the chromosome to be plot (defaults to 1).
`strand`	character scalar which should be set to either + or − to indicate which strand of DNA to plot the intensity values from (defaults to "+").
`vpr`	which viewport to plot the figure in. If this function is called directly by the user this argument should be left missing.
`colors`	named character vector, optional. If missing, a default color scheme is used: `c("+"="#00441b", "-"="#081d58", "duplicated"="grey", "cp"="#101010" "highlight"="red", "threshold"="grey")`, where the first three elements refer to the colors of data points and the last three to the colors of lines in the plot.
`main`	character vector specifying plot title.
`pointSize`	an object of class unit which specifies the size of each point. Default value is `unit(0.6, "mm")`.
`showConfidenceIntervals`	logical scalar indicating whether confidence intervals for each change-point are to be plotted (only available once segmentation has occurred).

sepPlots	logical scalar indicating whether the intensities are plotted separately for each array (if dat$y has multiple columns). Defaults to FALSE, in which case the average intensity for each probe is plotted. When TRUE, up to 3 arrays can be plotted separately (more than 3 gets crowded).
cexAxisLabel	numeric scalar specifying the magnification to be used for the y-axis label relative to the current test size.
cexAxis	numeric scalar specifying the magnification to be used for the y-axis annotation relative to the current text size.
...	additional arguments.

Details

This function is called by plotAlongChrom when the argument what is set to dots. Although this function can be called directly by the user, this is not recommended.

The dat list contains the following items: items x: x-coordinates (in bases) along chromosome,

1. y: intensity matrix of probes along chromosome,
2. flag: indicates probe uniqueness in the genome. Possibilities are 3: multiple perfect matches, 2: has no PM but one or more near-matches, 1: has exactly one PM and some near-matches in the genome, 0: has exactly one PM and no near-matches.
3. extras: (optional) matrix of additional values (such as test-statistics/p-values) to be plotted.

Author(s)

Wolfgang Huber <huber@ebi.ac.uk>

---

plotSegmentationHeatmap

Plot a heatmap diagram for a region along a chromosome

Description

Plot a heatmap diagram for a region along a chromosome

Usage

```
plotSegmentationHeatmap(dat, xlim, ylab, rowNames,
                        chr=1, strand="+", vpr, colors,
                        colHeatmap=colorRamp(brewer.pal(9, "YlGnBu")),
                        showConfidenceIntervals=TRUE,
                        just=c("left","centre"),
                        main, ...)
```

Arguments

dat	list containing data to be plotted (see *Details* section below for particulars).
xlim	integer vector of length 2 with start and end coordinates (in bases) for plotting.
ylab	character scalar specifying y-axis label.
rowNames	character vector specifying a name for each row in the heatmap plot.

chr	integer of length 1 indicating the chromosome to plot (defaults to 1).
strand	character scalar which should be set to either + or − to indicate which strand of DNA to plot the intensity values from (defaults to "+").
vpr	which viewport to plot the figure in. If this function is called directly by the user this argument should be left missing.
colors	named character vector, optional. If missing, a default color scheme is used: `c("+"="#00441b", "-"="#081d58", "duplicated"="grey", "cp"="#101010" "highlight"="red", "threshold"="grey")`, where the first three elements refer to colors of data points and the last three to those of lines in the plot.
colHeatmap	function describing color scheme for the heatmap plot (defaults to `YlGnBu` from RColorBrewer package).
showConfidenceIntervals	logical scalar indicating whether confidence intervals for each change-point are to be plotted (only available once segmentation has occurred).
just	character vector specifying the justification of the supplied values to the given coordinates; setting the first entry to "left" indicates that the supplied x-coordinates are the start positions of the probes, change this to "centre" if the x-coordinates are the probe middle positions. Usually the second entry should be "centre" (see `grid.rect`)
main	character vector specifying plot title.
...	additional arguments.

Details

This function is called by `plotAlongChrom` if the argument `what` is set to `heatmap`. Although this function can be called directly by the user, this is not recommended. The `dat` list contains the following items:

`x` x-coordinates (in bases) along chromosome

`y` intensity matrix of probes along chromosome

`flag` indicates probe uniqueness in the genome. Possibilities are 3: multiple perfect matches, 2: has no PM but one or more near-matches, 1: has exactly one PM and some near-matches in the genome, 0: has exactly one PM and no near-matches.

`extras` (optional) matrix of additional values (such as test-statistics/p-values) to be plotted

Author(s)

Wolfgang Huber <huber@ebi.ac.uk>

`PMindex` *Find the index of the exact match (PM) or background probes from a probeAnno environment*

Description

Find the index of the exact match (PM) or background probes from a probeAnno environment

Usage

```
PMindex(probeAnno)
BGindex(probeAnno)
```

Arguments

probeAnno	environment with probe annotations. See package davidTiling for an example (?probeAnno).

Details

These functions extract the exact match probes (PM) or background probes (from intergenic regions outside of known annotations) indices from probeAnno. These indices can be used to select the relevant rows of intensity data from the ExpressionSet object for plotting and normalization.

Value

Numeric vector of indices.

Author(s)

Matt Ritchie <ritchie@ebi.ac.uk>

Examples

```
## library(davidTiling)
## data(davidTiling)
## data(probeAnno)
## pmind <- PMindex(probeAnno)
## mmind <- MMindex(probeAnno)
## bgind <- BGindex(probeAnno)
## boxplot(as.data.frame(log2(exprs(davidTiling))[pmind,]), outline=FALSE)
```

---

posMin *Find the smallest positive number in a vector*

Description

Find the smallest positive number in a vector

Usage

```
posMin(x, ...)
```

Arguments

x	Numeric vector.
...	Further arguments that get passed on to min.

Details

This is a rather trivial convenience function.

Value

Numeric of length 1.

Author(s)

W. Huber <huber@ebi.ac.uk>

Examples

```
x = runif(5)
posMin(x-0.5)
posMin(x-2)
```

---

qcPlots *Generate simple diagnostic plots for Affymetrix tiling array data*

Description

Generate simple diagnostic plots for Affymetrix tiling array data

Usage

```
qcPlots(x, html=TRUE, plotdir=NULL, probeAnno, gff,
                    chr=4, coord=c(230000,245000),
                    nr = 2560, nc = 2560,
                    ylimchrom=c(5,16), nucleicAcid, pmindex, pgm=TRUE,
                    ext=".cel", ranks=FALSE, ...)
```

Arguments

x	ExpressionSet containing the data to be plotted.
html	logical scalar. If TRUE an html summary page 'qcsummary.htm' is generated. If FALSE, no summary page is generated.
plotdir	optional character string specifying the filepath where the plots will be saved. Defaults to current working directory.
probeAnno	environment with probe annotations. See package davidTiling for an example (?probeAnno).
gff	data frame with genome annotation from the GFF file.
chr	integer of length 1 indicating the chromosome number to plot.
coord	integer vector of length 2 containing the start and end coordinates (in bases) for the along chromosome intensity plot.
nr	integer, indicating the number of probes in each row on the array (2560 for yeast tiling arrays).
nc	integer, indicating the number of probes in each column on the array (2560 for yeast tiling arrays).
ylimchrom	numeric vector containing the y limits of the along chromosome intensity plot.
nucleicAcid	character vector or factor indicating what sample has been hybridised to each array. Used to color the boxplots and smoothed histograms of intensities.

pmindex	integer vector of indices of PM probes in x. If missing, this information is extracted from probeAnno.
pgm	logical scalar. If TRUE, image plots will be saved as .pgm files. Otherwise (FALSE), they are converted to jpegs. On windows machines, this argument should be set to TRUE.
ext	character string indicating the file extension.
ranks	logical scalar. If TRUE, imageplots will show ranks of standardised probe intensities. Otherwise (FALSE, default), the standardised probe intensities are plotted.
...	further arguments that can be passed to the plotting function plotSegmentationDots.

Details

This function creates boxplots, smoothed histogram (density) plots, imageplots and along chromosome plots of the raw (log base 2) probe intensity data.

An html page called 'qcsummary.htm' which displays the results, is created when html=TRUE.

Imageplots of standardised intensities (i.e. (probe intensity - minimum probe intensity) divided by the difference between the maximum and minimum probe intensities, all on log base 2 scale) or the ranks of these standardised intensities are plotted depending on the ranks argument.

The individual plots are named by replacing the file extension (specified by ext) of each 'celfile.ext', with 'density.png' for smoothed histogram plots, 'gencoord.jpg', for along chromosome plots and either 'log.pgm' ('log.jpg' if pgm=FALSE) or 'rank.pgm' ('rank.jpg' if pgm=FALSE) for the imageplots, depending on the ranks argument.

Author(s)

Matt Ritchie <ritchie@ebi.ac.uk> and Wolfgang Huber <huber@ebi.ac.uk>

Examples

```
## library(davidTiling)
## data(davidTiling)
## data(probeAnno)
## qcPlots(davidTiling, probeAnno)
```

---

readCel2eSet *Read celfiles into an ExpressionSet object.*

Description

This is a wrapper for ReadAffy that returns an ExpressionSet object rather than an AffyBatch. This is particularly usefiles for arrays for which we have or need no CDF environment.

Usage

```
readCel2eSet(filename, adf, path=".", rotated=FALSE, ...)
```

Arguments

filename	Character vector with CEL file names. Either filename or adf need to be specified, but not both.
adf	Object of class AnnotatedDataFrame.
path	Character scalar with path to CEL files.
rotated	Logical scalar, see details.
...	Further arguments that are passed on to new("ExpressionSet").

Details

The rotate options allows to deal with different versions of the scanner software. Older versions rotated the image by 90 degrees, newer ones do not. Use the default rotated=FALSE for CEL files produced by the newer version.

Value

ExpressionSet object.

Author(s)

W. Huber

Examples

```
## To test the rotation, look at the scatterplot between two DNA hybes
## that were measured with scanner software that rotated (041120) and did
## not rotate (060125)
##
## cp /ebi/research/huber/Projects/tilingArray/Celfiles/041120_S96genDNA_re-hybe.cel.gz ~
## cp /ebi/research/huber/Projects/allelicTranscription/celfiles_allelictrans/060125_S96_
## cd ~/p/tmp
## gunzip 041120_S96genDNA_re-hybe.cel.gz
## unzip 060125_S96_genomicDNA.zip
##
## Not run:
library("affy")
options(error=recover)

e1 = readCel2eSet("041120_S96genDNA_re-hybe.cel", rotated=TRUE)
e2 = readCel2eSet("060125_S96_genomicDNA.CEL")

smoothScatter(log(exprs(e1)), log(exprs(e2)), nrpoints=0)
## End(Not run)
```

---

sampleStep *Sampling of ascending numbers to ensure minimal spacing.*

Description

Given a vector of ascending numbers and a step width, sample the numbers such that the difference between consecutive numbers is greater than or equal to step.

Usage

```
sampleStep(x, step)
```

Arguments

x	Numeric or integer vector.
step	Numeric scalar.

Details

The simple algorithm works greedily from `x[1]` to `x[length(x)]`. First, `x[1]` is selected. Then, if `x[i]` is selected, all numbers `x[j]` with `j>i` and `x[j]-x[i]<step` are dropped. Then, `i` is set to the smallest j with `x[j]-x[i]>=step`.

Value

A logical vector of the same length as `x`, representing the selected subsample.

Author(s)

W. Huber <huber@ebi.ac.uk>

Examples

```
x   = sort(as.integer(runif(20)*100))
sel = sampleStep(x, step=10)
x
x[sel]
```

---

segChrom — *Fit a piecewise constant curve to along chromosome data (wrapper function)*

Description

Wrapper around the `segment` function for each strand of one or more chromosomes specified by the user. It does some typical preprocessing and I/O.

Usage

```
segChrom(y, probeAnno, chr=1:17, strands=c("+", "-"),
  nrBasesPerSegment = 1500, maxk = 3000, step = 7, confint = FALSE,
  confintLevel = 0.95, useLocks=TRUE, verbose=TRUE, savedir)
```

Arguments

y	ExpressionSet or matrix containing the data to be segmented.
probeAnno	environment with probe annotations. See package davidTiling for an example (?probeAnno).
chr	integer scalar or vector specifying which chromosome(s) to segment.
strands	character scalar or vector specifying which strands to segment.
nrBasesPerSegment	integer (length 1): the parameter maxseg of the segment function is calculated as the length of the chromosome divided by nrBasesPerSegment. Thus, it determines the average segment length in the finest segmentation.
maxk	passed on to the function segment.
step	integer scalar, indicating the minimum distance between consecutive probes. In cases when probes are offset by less than step bases, the probes are sampled to achieve the desired spacing.
confint	logical scalar. If TRUE, confidence intervals for each change-point are calculated.
confintLevel	numeric scalar between 0 and 1 indicating the probability level for the confidence intervals that are calculated for each change-point.
useLocks	logical scalar. Should a file locking mechanism be used that allows for a simple-minded parallelization of this function.
verbose	logical scalar. Should we be chatty about our progress?
savedir	character scalar. If specified, resulting segmentation objects are saved (with save) to this directory.

Details

This function is a wrapper for the segment function. Refer to its help page for further details.

Value

An environment containing S4 objects of class "segmentation" called "1.+", "1.-", etc. (depending on the values in chr and strands), where "+" and "-" indicate the strand and the preceding number refers to the chromosome. If savedir is specified, there is also the side-effect that a series of files "1.+.rda", "1.-.rda", etc. is saved in that directory.

Author(s)

Matt Ritchie <ritchie@ebi.ac.uk> and Wolfgang Huber <huber@ebi.ac.uk>

Examples

```
## Not run:
  library("davidTiling")
  data("davidTiling")
  data("probeAnno")
  isDNA = seq(1:3)
  yn = normalizeByReference(davidTiling[,-isDNA],davidTiling[,isDNA], probeAnno=probeAnno
  seg = segChrom(yn, probeAnno) ## this will take a while to run!
## End(Not run)
```

`segment` *Fit a piecewise constant curve: segmentation by dynamic programming*

Description

The function fits a piecewise constant curve to one or multiple sequences of measurements, using a least squares cost function and an O(n) dynamic programming algorithm (see references).

Usage

```
segment(y, maxseg, maxk)
```

Arguments

`y`	Numeric matrix. Rows correspond to the `x`-variable, columns to replicate measurements at the same value of `x`. Breakpoints are fitted along the `x`-axis. For example, the `x`-variable can be genomic coordinates or time. The segmentation will be along the rows of `y`.
`maxseg`	integer of length 1, maximum number of segments (= 1 + maximum number of change points).
`maxk`	integer of length 1, maximum length of a single segment.

Details

The complexity of the algorithm is `length(x)*maxk` in memory and `length(x)*maxk*maxseg` in time.

Value

An object of class `segmentation`.

Author(s)

W. Huber ⟨huber@ebi.ac.uk⟩

References

[1] Transcript mapping with high-density oligonucleotide tiling arrays. Huber W, Toedling J, Steinmetz, L. Bioinformatics 22, 1963-1970 (2006).

[2] A statistical approach for CGH microarray data analysis. Franck Picard, Stephane Robin, Marc Lavielle, Christian Vaisse, Gilles Celeux, Jean-Jacques Daudin. BMC Bioinformatics. 2005 Feb 11; 6:27.

Examples

```
x = rep( sin((0:4)/2*pi), each=3) + rnorm(3*5, sd=0.1)
res = segment(x, maxseg=6, maxk=15)
```

tilingArray-package *tilingArray package overview*

Description

tilingArray package overview

Details

The package provides some functionalities that can be useful for the analysis of high-density tiling microarray data (such as Affymetrix genechips) for measuring transcript abundance and architecture. The main functionalities of the package are:

- The segmentation class for representing partitionings of a linear series of data (such as microarray intensity readings along a chromosome strand).
- The function `segment` for fitting piecewise constant models using a dynamic programming algorithm that is both fast and exact, and `confint` for calculating confidence intervals using the strucchange package. Please see the vignette *Segmentation demo* in the file inst/doc/segmentation.pdf (source file inst/scripts/segmentation.Rnw).
- The function `plotAlongChrom` for generating pretty plots of segmentations along with genomic features. Please also see the vignette *Segmentation demo*.
- The function `normalizeByReference` for probe-sequence dependent response adjustment from a (set of) reference hybridizations. Please see the vignette *Assessing signal/noise ratio before and after normalization* in the file inst/doc/assessNorm.pdf (source file inst/scripts/assessNorm.Rnw).

Author(s)

W. Huber <huber@ebi.ac.uk>

Index