Package ‘epistack’

April 12, 2022

Title Heatmaps of Stack Profiles from Epigenetic Signals

Version 1.0.0

Description The epistack package main objective is the visualizations of stacks of genomic tracks (such as, but not restricted to, ChIP-seq, ATAC-seq, DNA methyation or genomic conservation data) centered at genomic regions of interest.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Imports GenomicRanges, BiocGenerics, S4Vectors, IRanges, viridisLite, graphics, plotrix, grDevices, stats

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Depends R (>= 4.1)

Suggests testthat (>= 3.0.0), BiocStyle, knitr, rmarkdown, EnrichedHeatmap, biomaRt, rtracklayer, covr, vdiffr, magick

Config/testthat/edition 3

VignetteBuilder knitr

biocViews RNASeq, Preprocessing, ChIPSeq, GeneExpression

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

git_branch RELEASE_3_14

git_last_commit fe0a920

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

Author SACI Safia [aut],
DEVAILLY Guillaume [cre]

Maintainer DEVAILLY Guillaume `<gdevailly@hotmail.com>`

R topics documented:

---

addBins *addBins()*

Description

Add an optional `bin` metadata column to `gr`, to serve as annotations for the epistack plots.

Usage

```
addBins(gr, nbins = 5L, bin = NULL)
```

Arguments

`gr`	a GRanges object.
`nbins`	an integer number, the number of bins.
`bin`	a vector containing pre-determined bins, in the same order as `gr`.

Details

`nbins` is taken into account only if `bin` is `NULL`. `gr` should be sorted first, usually with the `addMetricAndArrangeGRanges()` function. `addBin(gr,bin = vec)` is equivalent to `gr$bin <-vec`, while `addBin(gr,nbins = 5)` will create 5 bins of equal size based on `gr` order.

Value

the `gr` GRanges object with a new `bin` metadata column

See Also

`addMetricAndArrangeGRanges`

Examples

```
data("stackepi")
addBins(stackepi)

# 3 bins instead of 5
addBins(stackepi, nbins = 3)

# assign bins using a vector
addBins(stackepi, bin = rep(c("a", "b", "c"),
 length.out = length(stackepi)))
```

`addMetricAndArrangeGRanges` *addMetricAndArrangeGRanges()*

Description

Perform an inner join between a GRanges object and a data.frame. Sort the resulting GRanges based on a metric column.

Usage

```
addMetricAndArrangeGRanges(
  gr,
  order,
  gr_key = "name",
  order_key = "name",
  order_value = "exp",
  shuffle_tie = TRUE
)
```

Arguments

`gr`	a GRanges object.
`order`	a data.frame with at least two columns: keys and values.
`gr_key`	name of the gr metadata column containing unique names for each genomic region in `gr`. Usually gene names/id or peak id.
`order_key`	name of the `order` column that will be used as key for the inner join.
`order_value`	name of the `order` column that contain value used for sorting.
`shuffle_tie`	a boolean Value (TRUE / FALSE). When TRUE, shuffle the GRanges before sorting, mixing the ties.

Details

This utility function allow the addition of a metric column to genomic regions of interest. One of its common use case is to add gene expression values on a set of transcription start sites. The resulting GRanges object will only contain regions presents in both `gr` and `order`.

Value

a GRanges sorted in descending order.

Examples

```
data("stackepi")
ramdomOrder <- data.frame(gene_id = stackepi$gene_id,
   value = rnorm(length(stackepi)))
addMetricAndArrangeGRanges(stackepi,
   ramdomOrder, gr_key = "gene_id",
   order_key = "gene_id", order_value = "value" )
```

`meanColor` *meanColor*

Description

Return the average color of a vector of colors, computed in the RGB space.

Usage

```
meanColor(colors)
```

Arguments

`colors`	a vector of colors

Details

Input colors can be either in html or color name formats. The alpha channel is supported but optional.

Value

a single color value

See Also

`redimMatrix`

Examples

```
meanColor(c("#000000FF", "#FFFFFF00", "#FFFF00FF", "#FF0000FF"))

# works with color names
meanColor(c("blue", "red"))

# Mix color names and HTML codes
meanColor(c("blue", "red", "#FFFF00FF"))

# works without alpha channel in inputs (but outputs an alpha channel):
meanColor(c("#000000", "#FFFFFF", "#FFFF00", "#FF0000"))
```

`plotAverageProfile` *plotAverageProfile()*

Description

Plot the average stack profiles +/- error (sd or sem). If a `bin` column is present in `gr`, one average profile is drawn for each bin.

Usage

```
plotAverageProfile(
  gr,
  pattern = "^window_",
  x_labels = c("Before", "Anchor", "After"),
  palette = colorRampPalette(c("magenta", "black", "green")),
  alpha_for_se = 0.25,
  error_type = c("sd", "sem"),
  reversed_z_order = FALSE,
  ylim = NULL
)
```

Arguments

`gr`	a GRanges input
`pattern`	a single character that should match metadata of `gr` (can be a regular expression).
`x_labels`	x-axis labels.
`palette`	a color palette function, by default: `colorRampPalette(c("magenta","black","green"))`
`alpha_for_se`	the transparency (alpha) value for the error band.
`error_type,`	can be either `sd` (standard deviation) or `sem` (standard error of the mean). Default: `sem`.
`reversed_z_order`	should the z-order of the curves be reversed (i.e. first or last bin on top?)
`ylim`	a vector of two numbers corresponding to the y-limits of the plot

Value

Display a plot.

Examples

```
data("stackepi")
plotAverageProfile(stackepi)
```

---

plotBinning *plotBinning()*

Description

Plot a vertical color bar of the `bin` column.

Usage

```
plotBinning(
  gr,
  target_height = 650,
  palette = colorRampPalette(c("magenta", "black", "green"))
)
```

Arguments

`gr`	a GRanges object containing a `bin` metadata column
`target_height`	an integer, the approximate height (in pixels) of the final plot. Used to avoid overplotting artefacts.
`palette`	a function taking a number as a first argument, and returning a vector of colors.

Value

Display a plot.

Examples

```
data("stackepi")
gr <- stackepi
gr <- addBins(gr, nbins = 3)
plot_bin <- plotBinning(gr)

gr2 <- data.frame(bin = rep(c(1,2,3,4), each = 5))
plotBinning(gr2, palette = colorRampPalette(c("blue4", "forestgreen", "coral3", "goldenrod")))
```

plotBoxMetric *plotBoxMetric()*

Description

Plot distribution of a metric values as boxplots depending of bins. If the `bin` is absent from `gr`, a single boxplot is drawn.

Usage

```
plotBoxMetric(
  gr,
  metric = "expr",
  title = "Metric",
  trans_func = function(x) x,
  ylim = NULL,
  ylab = "metric",
  palette = colorRampPalette(c("magenta", "black", "green"))
)
```

Arguments

`gr`	a GRanges input
`metric`	name of the column in `gr` metadata containing scores.
`title`	title of the plot.
`trans_func`	A function to transform value of x before ploting. Useful to apply log10 transformation (i.e. with `trans_func = function(x) log10(x+1)`).
`ylim`	limit of the y axis; format: `ylim = c(min,max)`
`ylab`	y-axis title
`palette`	a function that returns a palette of `n` colors.

Value

Display a plot.

Examples

```
data("stackepi")
plotBoxMetric(
      stackepi,
      trans_func = function(x) x,
      metric = "exp",
      title = "Metric"
  )
```

`plotEpistack` *plotEpistack()*

Description

Given a list of genomic regions, epigenetic signals surrounding these regions, and a score for each regions, plot epigenetic stacks depending on the score. An optional `bin` column allow the grouping of several genomic regions to produce average profiles per bins.

Usage

```
plotEpistack(
  gr,
  patterns = "^window_",
  tints = "gray",
  titles = "",
  legends = "",
  x_labels = c("Before", "Anchor", "After"),
  zlim = c(0, 1),
  ylim = NULL,
  metric_col = "expr",
  metric_title = "Metric",
  metric_label = "metric",
  metric_transfunc = function(x) x,
  bin_palette = colorRampPalette(c("magenta", "black", "green")),
  npix_height = 650,
  n_core = 1,
  high_mar = c(2.5, 0.6, 4, 0.6),
  low_mar = c(2.5, 0.6, 0.3, 0.6),
  error_type = c("sd", "sem"),
  ...
)
```

Arguments

`gr`	a GRanges input.
`patterns`	a character vector of column prefixes (can be regular expressions) that should match columns of `gr`.
`tints`	a vector of colors to tint the heatmaps.
`titles`	titles of each heatmap.
`legends`	legend names for the epistacks.
`x_labels`	a character vector of length 3 used as x-axis labels.
`zlim`	the minimum and maximum z values the heatmap. Format: `zlim = c (min,max)`
`ylim`	limits of the y axis for bottom plots. Format: `ylim = c (min,max)`

`metric_col`	a character, name of a column in `gr` such as expression value, peak height, pvalue, fold change, etc.
`metric_title`	title to be display on the leftmost plots.
`metric_label`	label of the leftmost plots.
`metric_transfunc`	a function to transform value of `metric_col` before plotting. Useful to apply log10 transformation (i.e. with `trans_func = function(x) log10(x+1)`).
`bin_palette`	a palette of color, (i.e. a function of parameter n that should retrun n colors), used to color average profiles per bin in the bottom plots.
`npix_height`	The matrix height is reduced to this number of rows before plotting. Useful to limit overplotting artefacts. It should roughly be set to the pixel height in the final heatmaps
`n_core`	number of core used to speedup the martrix resizing.
`high_mar`	a vector of numerical values corresponding to the margins of the top figures. c(bottom, left, top, right)
`low_mar`	a vector of numerical values corresponding to the margins of the bottom figures. c(bottom, left, top, right)
`error_type,`	can be either `sd` (standard deviation) or `sem` (standard error of the mean). Default: `sem`.
`...`	Arguments to be passed to `par` such as `cex`

Details

This function produce a comprehensive figure including epigenetic heatmaps and average epigenetic profiles from a well formated `GRanges` object with expected metadata columns. It scales resonably well up to hundreds of thousands of genomic regions.

The visualisation is centered on an anchor, a set of genomic coordinated that can be transcription start sites or peak center for example. Anchor coordinates are those of the `GRanges` used as an input (hereafter `gr`).

Anchors are plotted from top to bottom in the same order as in `gr`. One should sort `gr` before plotting if needed.

`gr` should have a metric column that is used in the leftmost plots. The name of the metric column must be specified to `metric_col`. The metric can be transformed before plotting if needed using the `metric_transfunc` parameter.

The matrix or matrices used to display the heatmap(s) should be passed as additional metadata columns of `gr`. Such matrix can be obtained using `EnrichedHeatmap::normalizeToMatrix()` for example. The matrix columns names are then specified through `patterns` using prefixes, suffixes or regular expressions.

If an optionnal `bin` column is present in `gr`, it will be used to group genomic regions to performed average profile per bins in the bottom plots.

Epistack are multipanel plots build using `layout()`. Margins for the panels can be specified using `high_mar` and `low_mar` parameters if needed, especially to avoid text overlaps. The default value should be appropriate in most situations. Individual component can be plotted using severa `epistack` functions such has `plotStackProfile()` or `plotAverageProfile()`.

Plotting more than > 1000 regions can lead to overplotting issued as well as some plotting artefacts (such as horizontal white strips). Both issues can be resolved with fidling with the `npix_height` parameter. `npix_height` should be smaller than the number of regions, and in the same order of magnitude of the final heatmap height in pixels. Last minutes call to the `redimMatrix()` function will hapen before plotting using `npix_height` as target height. Parameter `n_core` is passed to `redimMatrix()` to speed up the down-scaling.

Value

Display a plot.

See Also

`plotStackProfile`, `plotAverageProfile`, `redimMatrix`, `normalizeToMatrix`, `addMetricAndArrangeGRanges`, `addBins`

Examples

```
data("stackepi")
plotEpistack(stackepi,
    metric_col = "exp",
    ylim = c(0, 1),
    metric_transfunc = function(x) log10(x+1))
```

`plotMetric` *plotMetric()*

Description

Plot a vertical line chart of the metric column, in the same order as the input.

Usage

```
plotMetric(
  x,
  trans_func = function(x) x,
  title = "Metric",
  ylim = NULL,
  xlab = NULL
)
```

Arguments

`x`	a numeric vector.
`trans_func`	a function to transform `x` values before ploting. Useful to apply log10 transformation (i.e. with `trans_func = function(x) log10(x+1)`).
`title`	Title of the plot.

`ylim`	limit of the y axis; format: `ylim = c(min,max)`
`xlab`	x-axis title

Value

Display a plot.

See Also

`plotEpistack`, `plotBoxMetric`

Examples

```
data("stackepi")
plotMetric(stackepi$exp)
```

`plotStackProfile` *plotStackProfile()*

Description

Display a heatmap of an epigenetic track centered at genomic anchors such as Transcription Start Sites or peak center.

Usage

```
plotStackProfile(
  gr,
  pattern = "^window_",
  x_labels = c("Before", "Anchor", "After"),
  title = "",
  zlim = NULL,
  palette = function(n) viridisLite::viridis(n, direction = -1),
  target_height = 650,
  summary_func = function(x) mean(x, na.rm = TRUE),
  n_core = 1
)
```

Arguments

`gr`	a GRanges input
`pattern`	a character vector of length 1 of a column prefixe (can be regular expressions) that should match columns of `gr`.
`x_labels`	a character vectors of length 3 used to label the x-axis.
`title`	The title of the heatmap

zlim	The minimum and maximum z values to match color to values. Format: zlim = c (min, max)
palette	a palette of color, (i.e. a function of parameter n that should retrun n colors).
target_height	The matrix height is reduced to this number of rows before plotting. Useful to limit overplotting artefacts. It should roughly be set to the pixel height in the final heatmap.
summary_func	function passed to `redimMatrix()`. Usualy `mean`, but can be set to `median` or `max` for sparse matrices.
n_core	multicore option, passed to `redimMatrix()`.

Details

The visualisation is centered on an anchor, a set of genomic coordinated that can be transcription start sites or peak center for example. Anchor coordinates are those of the `GRanges` used as an input (hereafter `gr`).

Anchors are plotted from top to bottom in the same order as in `gr`. One should sort `gr` before plotting if needed.

The matrix used to display the heatmap should be passed as additional metadata columns of `gr`. Such matrix can be obtained using `EnrichedHeatmap::normalizeToMatrix()` for example. The matrix columns names are then specified through `what_pattern` using a prefix, a suffix or a regular expressions.

This function scale reasonnably wells up to hundred thousands of regions. Overplotting issues are solved by last-minute reduction of the matrix size using `redimMatrix()`.

Value

Display a plot.

See Also

`plotAverageProfile`, `plotEpistack`, `normalizeToMatrix`, `plotStackProfileLegend`

Examples

```
data("stackepi")
plotStackProfile(stackepi,
                 target_height = 650,
                 zlim = c(0, 1),
                 palette = colorRampPalette(c("white", "dodgerblue", "black")),
                 title = "DNA methylation")
```

`plotStackProfileLegend` *plotStackProfileLegend()*

Description

Utility function to plot the corresponding legend key of `plotStackProfile()`'s plots.

Usage

```
plotStackProfileLegend(
  zlim,
  palette = colorRampPalette(c("white", "grey", "black")),
  title = NA
)
```

Arguments

`zlim`	the limits of the values to be displayed. Format: `c(min,max)`
`palette`	a palette of color, (i.e. a function of parameter n that should retrun n colors).
`title`	an optionnal title to be display bellow the color legend.

Value

Display a plot.

See Also

`plotStackProfile`

Examples

```
plotStackProfileLegend(zlim = c(0, 2),
    palette = colorRampPalette(c("white", "grey", "black")))
```

`redimMatrix` *redimMatrix()*

Description

Reduce the input matrix size by applying a summary function on cells to be fused.

Usage

```
redimMatrix(
  mat,
  target_height = 100,
  target_width = 100,
  summary_func = function(x) mean(x, na.rm = TRUE),
  output_type = 0,
  n_core = 1
)
```

Arguments

`mat`	the input matrix.
`target_height`	height of the output matrix (should be smaller than or equal to `nrow(mat)`).
`target_width`	width of the output matrix (should be smaller than or equal to `ncol(mat)`).
`summary_func`	how to summerize cells? A function such has `mean`, `median`, `max`, or `meanColors`.
`output_type`	Type of the output, to be passed to `vapply`'s FUN.VALUE.
`n_core`	number of core to use for parallel processing.

Details

This function is used to reduce matrix right before plotting them in order to avoid overplotting issues as well as other plotting artefacts.

Value

a resized matrix of size `target_width` x `target_height` where the `summary_fun` was apply to adjacent cells.

See Also

`meanColor`

Examples

```
data("stackepi")
mat <- S4Vectors::mcols(stackepi)
whichCols <- grepl("^window_", colnames(mat))
mat <- as.matrix(mat[, whichCols])
dim(mat)
smallMat <- redimMatrix(mat, target_height = 10, target_width = ncol(mat))
dim(smallMat)

mat <- matrix(sample(1:40,100,replace=TRUE),nrow=10,ncol=10)
dim(mat)
smallMat <- redimMatrix(mat, target_height = 5, target_width = ncol(mat),
   summary_func = function(x) max(x, na.rm = TRUE))
dim(smallMat)
```

`stackepi` *epistack example and test dataset*

Description

DNA methylation profiles (from MBD-seq data) arround transcription start sites of the 693 chr18 genes annotated on the pig genome (Sscrofa11.1), as well as gene expression levels in Transcript Per Million (TPM) measured by RNA-seq in the same duodenum sample.

Usage

```
data("stackepi")
```

Format

A `GRanges` of the 693 rows and 54 metadata columns

Source

This dataset was generated from ENSEMBL annotation data and data generated by our lab (publicly available soon).

Index