Package ‘seqPattern’

April 24, 2024

Title Visualising oligonucleotide patterns and motif occurrences across a set of sorted sequences

Version 1.35.0

Date 08-05-2015

Author Vanja Haberle <vanja.haberle@gmail.com>

Maintainer Vanja Haberle <vanja.haberle@gmail.com>

Imports Biostrings, GenomicRanges, IRanges, KernSmooth, plotrix

Depends methods, R (>= 2.15.0)

Suggests BSgenome.Drerio.UCSC.danRer7, CAGEr, RUnit, BiocGenerics, BiocStyle

Enhances parallel

Description Visualising oligonucleotide patterns and sequence motifs occurrences across a large set of sequences centred at a common reference point and sorted by a user defined feature.

License GPL-3

biocViews Visualization, SequenceMatching

Collate MotifScanningFunctions.R MotifScanningMethods.R PatternOccurrenceFunctions.R PatternOccurrenceMethods.R PlottingFunctions.R PlottingMethods.R

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

git_branch devel

git_last_commit aee1f02

git_last_commit_date 2023-10-24

Repository Bioconductor 3.19

Date/Publication 2024-04-23

Contents

seqPattern-package 2
getPatternOccurrenceList 3

motifScanHits 4
motifScanScores 6
plotMotifDensityMap 7
plotMotifOccurrenceAverage 10
plotMotifScanScores 12
plotPatternDensityMap 14
plotPatternOccurrenceAverage 17
TBPpwm 19
zebrafishPromoters 19
zebrafishPromoters24h 20

Index **21**

`seqPattern-package` *Visualising oligonucleotide patterns and motif occurences across a set of ordered sequences*

Description

Visualising oligonucleotide patterns and sequence motifs occurrences across a large set of sequences centred at a common reference point and sorted by a user defined feature.

Details

Package:	SeqPattern
Type:	Package
Version:	1.0
Date:	2014-12-05
License:	GPL-3
Depends:	R (>= 3.0.1), methods

Author(s)

Vanja Haberle

Maintainer: Vanja Haberle <vanja.haberle@gmail.com>

getPatternOccurrenceList

Occurrence of sequence patterns in a set of ordered sequences

Description

Finds positions of specified sequence patterns in a list of sequences of the same length ordered by a provided index. Sequence patterns can be consensus sequences of variable length and can contain IUPAC ambiguity code. Position of each pattern occurrence is specified in two-dimensional matrix, *i.e.* the first coordinate provides the ordinal number of the sequence and the second coordinate gives the position within the sequence where the pattern occurs.

Usage

```
getPatternOccurrenceList(regionsSeq, patterns, seqOrder =
        c(1:length(regionsSeq)), useMulticore = FALSE, nrCores = NULL)
```

Arguments

`regionsSeq`	A `DNAStringSet` object. Set of sequences of the same length in which to search for the patterns.
`patterns`	Character vector specifying one or more DNA sequence patterns (oligonucleotides). IUPAC ambiguity codes can be used and will match any letter in the subject that is associated with the code.
`seqOrder`	Integer vector specifying the order of the provided input sequences. Must have the same length as the number of sequences in the `regionSeq`. The default value will order the sequences as they are ordered in the `regionSeq` object.
`useMulticore`	Logical, should multicore be used. `useMulticore = TRUE` is supported only on Unix-like platforms.
`nrCores`	Number of cores to use when `useMulticore = TRUE`. Default value `NULL` uses all detected cores.

Details

This function uses the `matchPattern` function to find occurrences of given sequence patterns in a set of input sequences. Input sequences must all be of the same length and are ordered according to the index provided in the `seqOrder` argument, creating a `n * m` matrix, where `n` is the number of sequences and `m` is the length of the sequences. Positions of pattern matches in the resulting matrix are returned as two-dimensional coordinates.

Value

The function returns a named list with one element for each sequence pattern specified in the `patterns` argument. Each element of the list is a `data.frame` with positions of the corresponding pattern in the set of input sequences. The input sequences of the same length are sorted according

to the index in seqOrder argument and the positions of pattern matches in the resulting n * m matrix (where n is the number of sequences and m is the length of the sequence) are provided. The sequence column in the data.frame provides the ordinal number of the sequence in the ordered list of sequences and the position column provides the start position of the pattern match within that sequence.

Author(s)

Vanja Haberle

See Also

plotPatternDensityMap
motifScanHits

Examples

```
library(GenomicRanges)
load(system.file("data", "zebrafishPromoters.RData", package="seqPattern"))
promoterWidth <- elementMetadata(zebrafishPromoters)$interquantileWidth

# dinucleotide patterns
patternsOccurrence <- getPatternOccurrenceList(regionsSeq = zebrafishPromoters,
                    patterns = c("TA", "GC"), seqOrder = order(promoterWidth))
names(patternsOccurrence)
head(patternsOccurrence[["GC"]])

# motif consensus sequence
patternsOccurrence <- getPatternOccurrenceList(regionsSeq = zebrafishPromoters,
                    patterns = "TATAWAWR", seqOrder = order(promoterWidth))
names(patternsOccurrence)
head(patternsOccurrence[["TATAWAWR"]])
```

---

motifScanHits *Occurrence of motifs in a set of ordered sequences*

Description

Finds positions of sequence motif hits above a specified threshold in a list of sequences of the same length ordered by a provided index. Motif is specified by a position weight matrix (PWM) that contains estimated probability of base b at position i and is usually constructed via call to PWM function. Position of each motif hit is specified in two-dimensional matrix, *i.e.* the first coordinate provides the ordinal number of the sequence and the second coordinate gives the position within the sequence where the motif occurs.

Usage

```
motifScanHits(regionsSeq, motifPWM, minScore = "80%",
    seqOrder = c(1:length(regionsSeq)))
```

Arguments

`regionsSeq` A `DNAStringSet` object. Set of sequences of the same length in which to search for the motif hits.

`motifPWM` A numeric matrix representing the Position Weight Matrix (PWM), such as returned by `PWM` function. Can contain either probabilities or log2 probability ratio of base b at position i.

`minScore` The minimum score for counting a motif hit. Can be given as a character string containing a percentage (*e.g.* `"85%"`) of the PWM score or a single number specifying score threshold. If a percentage is given, it is converted to a score value taking into account both minimal and maximal possible PWM scores as follows: `minPWMscore + percThreshold/100 * (maxPWMscore - minPWMscore)` This differs from the formula in the `matchPWM` function from the `Biostrings` package which takes into account only the maximal possible PWM score and considers the given percentage as the percentage of that maximal score: `percThreshold/100 * maxPWMscore`

`seqOrder` Integer vector specifying the order of the provided input sequences. Must have the same length as the number of sequences in the `regionSeq`. The default value will order the sequences as they are ordered in the input `regionSeq` object.

Details

This function uses the `matchPWM` function to find matches to given motif in a set of input sequences. Only matches above specified `minScore` are considered as hits. Input sequences must all be of the same length and are ordered according to the index provided in the `seqOrder` argument, creating a `n * m` matrix, where `n` is the number of sequences and `m` is the length of the sequences. Positions of motif hits in the resulting matrix are returned as two-dimensional coordinates.

Value

The function returns a `data.frame` with positions of the motif hits in the set of input sequences. The input sequences of the same length are sorted according to the index in `seqOrder` argument and the positions of motif hits in the resulting `n * m` matrix (where `n` is the number of sequences and `m` is the length of the sequence) are provided. The `sequence` column in the data.frame provides the ordinal number of the sequence in the ordered list of sequences and the `position` column provides the start position of the motif hit within that sequence.

Author(s)

Vanja Haberle

See Also

`plotMotifDensityMap`
`getPatternOccurrenceList`

Examples

```
library(GenomicRanges)
load(system.file("data", "zebrafishPromoters.RData", package="seqPattern"))
promoterWidth <- elementMetadata(zebrafishPromoters)$interquantileWidth

load(system.file("data", "TBPpwm.RData", package="seqPattern"))

motifOccurrence <- motifScanHits(regionsSeq = zebrafishPromoters,
                                 motifPWM = TBPpwm, minScore = "85%",
                                 seqOrder = order(promoterWidth))
head(motifOccurrence)
```

`motifScanScores` *Motif scanning scores for a set of ordered sequences*

Description

Provides motif scanning scores along the full length of a sequence for a list of sequences of the same length ordered by a provided index. Motif is specified by a position weight matrix (PWM) that contains estimated probability of base b at position i and is usually constructed via call to `PWM` function. Scanning scores are returned in the form of a two-dimensional matrix, where the rows are sequences ordered by the specified index and the columns are relative positions within the sequence. Each cell in the matrix contains the score of the specified motif in the given sequence starting at the given position. The resulting matrix can be used to visualise motif occurrences and their strength in an ordered set of sequences centered at a common reference point.

Usage

```
motifScanScores(regionsSeq, motifPWM, seqOrder = c(1:length(regionsSeq)),
        asPercentage = TRUE)
```

Arguments

`regionsSeq`	A `DNAStringSet` object. Set of sequences of the same length to be scanned with the motif.
`motifPWM`	A numeric matrix representing the Position Weight Matrix (PWM), such as returned by `PWM` function. Can contain either probabilities or log2 probability ratio of base b at position i.
`seqOrder`	Integer vector specifying the order of the provided input sequences. Must have the same length as the number of sequences in the `regionSeq`. The default value will order the sequences as they are ordered in the input `regionSeq` object.
`asPercentage`	Logical, should the scores represent percentage of the maximal motif PWM score (`TRUE`) or raw scores (`FALSE`).

Details

This function uses the PWMscoreStartingAt function to get scores for a given motif starting at each position (nucleotide) in a set of input sequences. Input sequences must all be of the same length and are ordered according to the index provided in the seqOrder argument, creating an n * m matrix, where n is the number of sequences and m is the length of the sequences. Each cell in the matrix contains the score of the specified motif in the given sequence starting at the given position.

Value

The function returns a matrix with motif scanning scores for each position in the set of input sequences.

Author(s)

Vanja Haberle

See Also

plotMotifScanScores
motifScanHits

Examples

```
library(GenomicRanges)
load(system.file("data", "zebrafishPromoters.RData", package="seqPattern"))
promoterWidth <- elementMetadata(zebrafishPromoters)$interquantileWidth

load(system.file("data", "TBPpwm.RData", package="seqPattern"))

motifScores <- motifScanScores(regionsSeq = zebrafishPromoters,
                            motifPWM = TBPpwm, seqOrder = order(promoterWidth),
                            asPercentage = TRUE)
dim(motifScores)
motifScores[1:10,1:10]
```

---

plotMotifDensityMap *Plotting density maps of motif occurrence*

Description

Plots density of motif occurrences in an ordered set of sequences of the same length in the form of a two dimensional map centered at a common reference position. Motif is specified by a position weight matrix (PWM) that contains estimated probability of base b at position i, and only motif hits above specified threshold are taken into account and plotted.

Usage

```
plotMotifDensityMap(regionsSeq, motifPWM, minScore = "80%",
    seqOrder = c(1:length(regionsSeq)), flankUp = NULL, flankDown = NULL,
    nBin = NULL, bandWidth = NULL, color = "blue", transf = NULL, xTicks = NULL,
    xTicksAt = NULL, xLabel = "", yTicks = NULL, yTicksAt = NULL, yLabel = "",
    cexAxis = 8, plotScale = TRUE, scaleLength = NULL, scaleWidth = 15,
    addReferenceLine = TRUE, plotColorLegend = TRUE, outFile = "DensityMap",
    plotWidth = 2000, plotHeight = 2000)
```

Arguments

`regionsSeq` A `DNAStringSet` object. Set of sequences of the same length for which the motif occurrence density should be visualised.

`motifPWM` A numeric matrix representing the Position Weight Matrix (PWM), such as returned by `PWM` function. Can contain either probabilities or log2 probability ratio of base b at position i.

`minScore` The minimum score for counting a motif hit. Can be given as a character string containing a percentage (*e.g.* `"85%"`) of the PWM score or a single number specifying score threshold. If a percentage is given, it is converted to a score value taking into account both minimal and maximal possible PWM scores as follows: `minPWMscore + percThreshold/100 * (maxPWMscore - minPWMscore)` This differs from the formula in the `matchPWM` function from the `Biostrings` package which takes into account only the maximal possible PWM score and considers the given percentage as the percentage of that maximal score: `percThreshold/100 * maxPWMscore`

`seqOrder` Integer vector specifying the order of the provided input sequences. Must have the same length as the number of sequences in the `regionSeq`. Input sequences will be sorted according to this index in an ascending order form top to the bottom of the plot, *i.e.* the sequence labeled with the lowest number will appear at the top of the plot. The default value will order the sequences as they are ordered in the input `regionSeq` object.

`flankUp`, `flankDown` The number of base-pairs upstream and downstream of the reference position in the provided sequences, respectively. `flankUp + flankDown` must sum up to the length of the sequences. If no values are provided both `flankUp` and `flankDown` are set to be half of the length of the input sequences, *i.e.* the reference position is assumed to be in the middle of the sequences.

`nBin` Numeric vector with two values containing the number of equally spaced points in each direction over which the density is to be estimated. The first value specifies number of bins along x-axis, *i.e.* along the nucleotides in the sequence, and the second value specifies the number of bins along y-axis, *i.e.* across ordered input sequences. The values are passed on to the `gridsize` argument of the `bkde2D` function to compute a 2D binned kernel density estimate. If `nBin` is not specified it will default to `c(n, m)`, where `n` is the number of input sequences and `m` is the length of sequences.

`bandWidth` Numeric vector of length 2, containing the bandwidth to be used in each coordinate direction. The first value specifies the bandwidth along the x-axis, *i.e.*

along the nucleotides in the sequence, and the second value specifies the bandwidth along y-axis, *i.e.* across ordered input sequences. The values are passed on to the `bandwidth` argument of the `bkde2D` function to compute a 2D binned kernel density estimate and are used as standard deviation of the bivariate Gaussian kernel. If `bandWidth` is not specified it will default to `c(3,3)`.

`color` Character specifying the color palette for the density plot. One of the following color palettes can be specified: `"blue"`, `"brown"`, `"cyan"`, `"gold"`, `"gray"`, `"green"`, `"pink"`, `"purple"`, `"red"`. Please refer to the vignette for the appearance of these palettes.

`transf` The function mapping the density scale to the color scale. See Details.

`xTicks` Character vector of labels to be placed at the tick-marks on x-axis. The default `NULL` value produces five tick-marks: one at the reference point and two equally spaced tick-marks both upstream and downstream of the reference point.

`xTicksAt` Numeric vector of positions of the tick-marks on the x-axis. The values can range from 1 (the position of the first base-pair in the sequence) to input sequence length. The default `NULL` value produces five tick-marks: one at the reference point and two equally spaced tick-marks both upstream and downstream of the reference point.

`xLabel` The label for the x-axis. The default is no label, *i.e.* empty string.

`yTicks` Character vector of labels to be placed at the tick-marks on y-axis. The default `NULL` value produces no tick-marks and labels.

`yTicksAt` Numeric vector of positions of the tick-marks on the y-axis. The values can range from 1 (the position of the last sequence on the bottom of the plot) to input sequence length (the position of the first sequence on the top of the plot). The default `NULL` value produces no tick-marks.

`yLabel` The label for the y-axis. The default is no label, *i.e.* empty string.

`cexAxis` The magnification to be used for axis annotation.

`plotScale` Logical, should the scale bar be plotted in the lower left corner of the plot.

`scaleLength` The length of the scale bar to be plotted. Used only when `plotScale = TRUE`. If no value is provided, it defaults to one fifth of the input sequence length.

`scaleWidth` The width of the line for the scale bar. Used only when `plotScale = TRUE`.

`addReferenceLine` Logical, should the vertical dashed line be drawn at the reference point.

`plotColorLegend` Logical, should the color legend for the pattern density be plotted. If `TRUE` a separate .png file named `outFile`."ColorLegend.png" will be created, showing mapping of pattern density values to colours.

`outFile` Character vector specifying the base name of the output plot file. The final name of the plot file for each pattern will be `outFile`."pattern.jpg".

`plotWidth, plotHeight` Width and height of the density plot in pixels.

Value

The function produces a PNG file in the working directory, visualising density of the motif occurrence above specified threshold in the set of ordered input sequences.

Author(s)

Vanja Haberle

References

Haberle *et al.* (2014) Two independent transcription initiation codes overlap on vertebrate core promoters, *Nature* **507**:381-385.

See Also

motifScanHits
plotPatternDensityMap

Examples

```
library(GenomicRanges)
load(system.file("data", "zebrafishPromoters.RData", package="seqPattern"))
promoterWidth <- elementMetadata(zebrafishPromoters)$interquantileWidth

load(system.file("data", "TBPpwm.RData", package="seqPattern"))

plotMotifDensityMap(regionsSeq = zebrafishPromoters, motifPWM = TBPpwm,
                    minScore = "85%", seqOrder = order(promoterWidth),
                    flankUp = 400, flankDown = 600, color = "red")
```

---

plotMotifOccurrenceAverage

Plotting average profile of motif occurrence

Description

Plots average profile of motif occurrence for a set of input sequences of the same length. Motif is specified by a position weight matrix (PWM) that contains estimated probability of base b at position i, and only motif hits above specified threshold are taken into account.

Usage

```
plotMotifOccurrenceAverage(regionsSeq, motifPWM, minScore = "80%", flankUp =
    NULL, flankDown = NULL, smoothingWindow = 1, color = "black", xLabel =
    "Distance to reference point (bp)", yLabel = "Relative frequency", cexAxis =
    1, addReferenceLine = TRUE, plotLegend = FALSE, cexLegend = 1, add = FALSE,
    ...)
```

Arguments

`regionsSeq` A `DNAStringSet` object. Set of sequences of the same length for which the patterns occurrence profile should be visualised.

`motifPWM` A numeric matrix representing the Position Weight Matrix (PWM), such as returned by `PWM` function. Can contain either probabilities or log2 probability ratio of base b at position i.

`minScore` The minimum score for counting a motif hit. Can be given as a character string containing a percentage (*e.g.* `"85%"`) of the PWM score or a single number specifying score threshold. If a percentage is given, it is converted to a score value taking into account both minimal and maximal possible PWM scores as follows: `minPWMscore + percThreshold/100 * (maxPWMscore - minPWMscore)` This differs from the formula in the `matchPWM` function from the `Biostrings` package which takes into account only the maximal possible PWM score and considers the given percentage as the percentage of that maximal score: `percThreshold/100 * maxPWMscore`

`flankUp, flankDown` The number of base-pairs upstream and downstream of the reference position in the provided sequences, respectively. `flankUp + flankDown` must sum up to the length of the sequences. If no values are provided both `flankUp` and `flankDown` are set to be half of the length of the input sequences, *i.e.* the reference position is assumed to be in the middle of the sequences.

`smoothingWindow` Integer specifying the size of a window (in base-pairs) to be used for smoothing the signal. Default value of 1 corresponds to no smoothing.

`color` Value specifying the color for plotting.

`xLabel, yLabel` Character strings for x and y axis labels, respectively.

`cexAxis` The magnification to be used for axis annotation relative to the current setting of `cex`.

`addReferenceLine` Logical, should the vertical dashed line be drawn at the reference point.

`plotLegend` Logical, should the legend be plotted at the top.

`cexLegend` The magnification to be used for legend relative to the current setting of `cex`.

`add` Logical, should the pattern occurrence profiles be added to the existing plot.

`...` Further arguments to be passed to `plot()` or `lines()` methods, such as `lty`, *etc*.

Value

The function finds all hits matching the motif above the specified score threshold in the set of input sequences and plots an average profile reflecting the occurrence of the motif across input sequences.

Author(s)

Vanja Haberle

See Also

motifScanHits
plotMotifDensityMap

Examples

```
load(system.file("data", "zebrafishPromoters.RData", package="seqPattern"))
load(system.file("data", "TBPpwm.RData", package="seqPattern"))

plotMotifOccurrenceAverage(regionsSeq = zebrafishPromoters, motifPWM = TBPpwm,
                    minScore = "85%", flankUp = 400, flankDown = 600,
                    smoothingWindow = 3)
```

---

plotMotifScanScores *Plotting heatmap of motif scanning scores*

Description

Plots heatmap of motif scanning scores for a set of sorted sequences of the same length in the form of a two dimensional map centered at a common reference position. Motif is specified by a position weight matrix (PWM) that contains estimated probability of base b at position i, and the percentage of the maximal PWM score is plotted for each position (nucleotide) in each sequence.

Usage

```
plotMotifScanScores(regionsSeq, motifPWM, seqOrder = c(1:length(regionsSeq)),
    flankUp = NULL, flankDown = NULL, xTicks = NULL, xTicksAt = NULL,
    xLabel = "", yTicks = NULL, yTicksAt = NULL, yLabel = "", cexAxis = 8,
    plotScale = TRUE, scaleLength = NULL, scaleWidth = 15,
    addReferenceLine = TRUE, plotColorLegend = TRUE, outFile =
    "MotifScanningScores.png", plotWidth = 2000, plotHeight = 2000)
```

Arguments

regionsSeq	A DNAStringSet object. Set of sequences of the same length for which the motif occurrence density should be visualised.
motifPWM	A numeric matrix representing the Position Weight Matrix (PWM), such as returned by PWM function. Can contain either probabilities or log2 probability ratio of base b at position i.
seqOrder	Integer vector specifying the order of the provided input sequences. Must have the same length as the number of sequences in the regionSeq. Input sequences will be sorted according to this index in an ascending order form top to the bottom of the plot, *i.e.* the sequence labeled with the lowest number will appear at the top of the plot. The default value will order the sequences as they are ordered in the input regionSeq object.

`flankUp`, `flankDown`
: The number of base-pairs upstream and downstream of the reference position in the provided sequences, respectively. `flankUp` + `flankDown` must sum up to the length of the sequences.

`xTicks`
: Character vector of labels to be placed at the tick-marks on x-axis. The default `NULL` value produces five tick-marks: one at the reference point and two equally spaced tick-marks both upstream and downstream of the reference point.

`xTicksAt`
: Numeric vector of positions of the tick-marks on the x-axis. The values can range from 1 (the position of the first base-pair in the sequence) to input sequence length. The default `NULL` value produces five tick-marks: one at the reference point and two equally spaced tick-marks both upstream and downstream of the reference point.

`xLabel`
: The label for the x-axis. The default is no label, *i.e.* empty string.

`yTicks`
: Character vector of labels to be placed at the tick-marks on y-axis. The default `NULL` value produces no tick-marks and labels.

`yTicksAt`
: Numeric vector of positions of the tick-marks on the y-axis. The values can range from 1 (the position of the last sequence on the bottom of the plot) to input sequence length (the position of the first sequence on the top of the plot). The default `NULL` value produces no tick-marks.

`yLabel`
: The label for the y-axis. The default is no label, *i.e.* empty string.

`cexAxis`
: The magnification to be used for axis annotation.

`plotScale`
: Logical, should the scale bar be plotted in the lower left corner of the plot.

`scaleLength`
: The length of the scale bar to be plotted. Used only when `plotScale` = `TRUE`. If no value is provided, it defaults to one fifth of the input sequence length.

`scaleWidth`
: The width of the line for the scale bar. Used only when `plotScale` = `TRUE`.

`addReferenceLine`
: Logical, should the vertical dashed line be drawn at the reference point.

`plotColorLegend`
: Logical, should the color legend for the scanning score be plotted on the right side of the plot.

`outFile`
: Character vector specifying the base name of the output plot file. The final name of the plot file for each pattern will be `outFile`."pattern.jpg".

`plotWidth`, `plotHeight`
: Width and height of the density plot in pixels.

Value

The function produces a PNG file in the working directory, visualising motif scanning scores in the set of ordered input sequences.

Author(s)

Vanja Haberle

See Also

motifScanScores
plotPatternDensityMap

Examples

```
library(GenomicRanges)
load(system.file("data", "zebrafishPromoters.RData", package="seqPattern"))
promoterWidth <- elementMetadata(zebrafishPromoters)$interquantileWidth

load(system.file("data", "TBPpwm.RData", package="seqPattern"))

plotMotifScanScores(regionsSeq=zebrafishPromoters, motifPWM = TBPpwm,
        seqOrder=order(promoterWidth), flankUp = 400, flankDown = 600)
```

---

plotPatternDensityMap *Plotting density maps of sequence pattern occurrence*

Description

Plots density of sequence pattern occurrences in an ordered set of sequences of the same length in the form of a two dimensional map centered at a common reference position. Multiple sequence patterns can be processed at once and one plot per pattern will be created with the same color scale across all plots, allowing visual density comparison across different patterns.

Usage

```
plotPatternDensityMap(regionsSeq, patterns, seqOrder = c(1:length(regionsSeq)),
    flankUp = NULL, flankDown = NULL, nBin = NULL, bandWidth = NULL,
    color = "blue", transf = NULL, xTicks = NULL, xTicksAt = NULL, xLabel = "",
    yTicks = NULL, yTicksAt = NULL, yLabel = "", cexAxis = 8, plotScale = TRUE,
    scaleLength = NULL, scaleWidth = 15, addPatternLabel = TRUE, cexLabel = 8,
    labelCol = "black", addReferenceLine = TRUE, plotColorLegend = TRUE,
    outFile = "PatternDensityMap", plotWidth = 2000, plotHeight = 2000,
    useMulticore = FALSE, nrCores = NULL)
```

Arguments

`regionsSeq` A DNAStringSet object. Set of sequences of the same length for which the patterns occurrence density should be visualised.

`patterns` Character vector specifying one or more DNA sequence patterns (oligonucleotides). IUPAC ambiguity codes can be used and will match any letter in the subject that is associated with the code.

`seqOrder` Integer vector specifying the order of the provided input sequences. Must have the same length as the number of sequences in the `regionSeq`. Input sequences will be sorted according to this index in an ascending order form top to the bottom of the plot, *i.e.* the sequence labeled with the lowest number will appear

at the top of the plot. The default value will order the sequences as they are ordered in the input `regionSeq` object.

`flankUp`, `flankDown`
: The number of base-pairs upstream and downstream of the reference position in the provided sequences, respectively. `flankUp` + `flankDown` must sum up to the length of the sequences. If no values are provided both `flankUp` and `flankDown` are set to be half of the length of the input sequences, *i.e.* the reference position is assumed to be in the middle of the sequences.

`nBin`
: Numeric vector with two values containing the number of equally spaced points in each direction over which the density is to be estimated. The first value specifies number of bins along x-axis, i.e. along the nucleotides in the sequence, and the second value specifies the number of bins along y-axis, i.e. across ordered input sequences. The values are passed on to the `gridsize` argument of the `bkde2D` function to compute a 2D binned kernel density estimate. If `nBin` is not specified it will default to `c(n, m)`, where `n` is the number of input sequences and `m` is the length of sequences.

`bandWidth`
: Numeric vector of length 2, containing the bandwidth to be used in each coordinate direction. The first value specifies the bandwidth along the x-axis, *i.e.* along the nucleotides in the sequence, and the second value specifies the bandwidth along y-axis, *i.e.* across ordered input sequences. The values are passed on to the `bandwidth` argument of the `bkde2D` function to compute a 2D binned kernel density estimate and are used as standard deviation of the bivariate Gaussian kernel. If `bandWidth` is not specified it will default to `c(3,3)`.

`color`
: Character specifying the color palette for the density plot. One of the following color palettes can be specified: `"blue"`, `"brown"`, `"cyan"`, `"gold"`, `"gray"`, `"green"`, `"pink"`, `"purple"`, `"red"`. Please refer to the vignette for the appearance of these palettes.

`transf`
: The function mapping the density scale to the color scale. See Details.

`xTicks`
: Character vector of labels to be placed at the tick-marks on x-axis. The default `NULL` value produces five tick-marks: one at the reference point and two equally spaced tick-marks both upstream and downstream of the reference point.

`xTicksAt`
: Numeric vector of positions of the tick-marks on the x-axis. The values can range from 1 (the position of the first base-pair in the sequence) to input sequence length. The default `NULL` value produces five tick-marks: one at the reference point and two equally spaced tick-marks both upstream and downstream of the reference point.

`xLabel`
: The label for the x-axis. The default is no label, *i.e.* empty string.

`yTicks`
: Character vector of labels to be placed at the tick-marks on y-axis. The default `NULL` value produces no tick-marks and labels.

`yTicksAt`
: Numeric vector of positions of the tick-marks on the y-axis. The values can range from 1 (the position of the last sequence on the bottom of the plot) to input sequence length (the position of the first sequence on the top of the plot). The default `NULL` value produces no tick-marks.

`yLabel`
: The label for the y-axis. The default is no label, *i.e.* empty string.

`cexAxis`
: The magnification to be used for axis annotation.

`plotScale` Logical, should the scale bar be plotted in the lower left corner of the plot.

`scaleLength` The length of the scale bar to be plotted. Used only when `plotScale = TRUE`. If no value is provided, it defaults to one fifth of the input sequence length.

`scaleWidth` The width of the line for the scale bar. Used only when `plotScale = TRUE`.

`addPatternLabel` Logical, should the pattern label be written in the upper left corner of the plot.

`cexLabel` The magnification to be used for pattern label.

`labelCol` The color to be used for pattern label and scale bar.

`addReferenceLine` Logical, should the vertical dashed line be drawn at the reference point.

`plotColorLegend` Logical, should the color legend for the pattern density be plotted. If `TRUE` a separate .png file named `outFile`."ColorLegend.png" will be created, showing mapping of pattern density values to colours.

`outFile` Character vector specifying the base name of the output plot file. The final name of the plot file for each pattern will be `outFile`."pattern.png".

`plotWidth, plotHeight` Width and height of the density plot(s) in pixels.

`useMulticore` Logical, should multicore be used. `useMulticore = TRUE` is supported only on Unix-like platforms.

`nrCores` Number of cores to use when `useMulticore = TRUE`. Default value `NULL` uses all detected cores.

Value

The function produces PNG files in the working directory, visualising density of patterns occurrence in the set of ordered input sequences. One file/plot per specified pattern is created.

Author(s)

Vanja Haberle

References

Haberle *et al.* (2014) Two independent transcription initiation codes overlap on vertebrate core promoters, *Nature* **507**:381-385.

See Also

`getPatternOccurrenceList`
`plotMotifDensityMap`

Examples

```
library(GenomicRanges)
load(system.file("data", "zebrafishPromoters.RData", package="seqPattern"))

promoterWidth <- elementMetadata(zebrafishPromoters)$interquantileWidth

# dinucleotide patterns
plotPatternDensityMap(regionsSeq = zebrafishPromoters, patterns = c("TA", "GC"),
            seqOrder = order(promoterWidth), flankUp = 400, flankDown = 600,
            color = "blue")

# motif consensus sequence
plotPatternDensityMap(regionsSeq = zebrafishPromoters, patterns = "TATAWAWR",
            seqOrder = order(promoterWidth), flankUp = 400, flankDown = 600,
            color = "cyan")
```

---

plotPatternOccurrenceAverage

Plotting average profile of sequence pattern occurrence

Description

Plots average profile of sequence pattern occurrence for a set of input sequences of the same length. Multiple sequence patterns can be processed at once and visualised in the same plot, allowing comparison across different patterns.

Usage

```
plotPatternOccurrenceAverage(regionsSeq, patterns, flankUp = NULL,
    flankDown = NULL, smoothingWindow = 1, color = rainbow(length(patterns)),
    xLabel = "Distance to reference point (bp)", yLabel = "Relative frequency",
    cexAxis = 1, addReferenceLine = TRUE, plotLegend = TRUE, cexLegend = 1,
    useMulticore = FALSE, nrCores = NULL, add = FALSE, ...)
```

Arguments

`regionsSeq` A `DNAStringSet` object. Set of sequences of the same length for which the patterns occurrence profile should be visualised.

`patterns` Character vector specifying one or more DNA sequence patterns (oligonucleotides). IUPAC ambiguity codes can be used and will match any letter in the subject that is associated with the code.

`flankUp`, `flankDown` The number of base-pairs upstream and downstream of the reference position in the provided sequences, respectively. `flankUp` + `flankDown` must sum up to the length of the sequences. If no values are provided both `flankUp` and `flankDown` are set to be half of the length of the input sequences, *i.e.* the reference position is assumed to be in the middle of the sequences.

`smoothingWindow`
: Integer specifying the size of a window (in base-pairs) to be used for smoothing the signal. Default value of 1 corresponds to no smoothing.

`color`
: A vector of values specifying the colors for plotting. Number of colors must match the number of patterns that should be plotted.

`xLabel, yLabel`
: Character strings for x and y axis labels, respectively.

`cexAxis`
: The magnification to be used for axis annotation relative to the current setting of `cex`.

`addReferenceLine`
: Logical, should the vertical dashed line be drawn at the reference point.

`plotLegend`
: Logical, should the legend be plotted at the top.

`cexLegend`
: The magnification to be used for legend relative to the current setting of `cex`.

`useMulticore`
: Logical, should multicore be used. `useMulticore = TRUE` is supported only on Unix-like platforms.

`nrCores`
: Number of cores to use when `useMulticore = TRUE`. Default value `NULL` uses all detected cores.

`add`
: Logical, should the pattern occurrence profiles be added to the existing plot.

`...`
: Further arguments to be passed to `plot()` or `lines()` methods, such as `lty`, *etc*.

Value

The function finds all hits matching the specified patterns in the set of input sequences and plots one average profile per pattern reflecting the occurrence of patterns across sequences.

Author(s)

Vanja Haberle

See Also

getPatternOccurrenceList
plotPatternDensityMap

Examples

```
load(system.file("data", "zebrafishPromoters.RData", package="seqPattern"))

# dinucleotide patterns
plotPatternOccurrenceAverage(regionsSeq = zebrafishPromoters, patterns = c("AT",
            "TA", "CG", "GC"), flankUp = 400, flankDown = 600, smoothingWindow =
            3, color = c("gold", "darkred", "forestgreen", "navy"))

# motif consensus sequence
plotPatternOccurrenceAverage(regionsSeq = zebrafishPromoters, patterns =
            "TATAWAWR", flankUp = 400, flankDown = 600, smoothingWindow = 3,
            color = "gray")
```

TBPpwm — *Position-weight matrix for TATA-box binding protein motif*

Description

This is a `matrix` object representing a position-weight matrix (PWM) for TATA-box binding protein motif. The scores in the matrix correspond to log2 ratio between probability of observing base b (rows) at position i (columns) and background probability of base b. This PWM has been derived using position frequency matrix from Jaspar database. It is intended to be used as an input data in running examples from `seqPattern` package help pages.

Usage

```
data(TBPpwm)
```

Format

A `matrix` object

Value

A `matrix` object

zebrafishPromoters — *Zebrafish promoters sequences*

Description

This is a `DNAStringSet` object that contains sequence of 1000 randomly selected promoters active in 24 hpf zebrafish *Danio rerio* embryos. The data is taken from Nepal *et al.* Genome Research 2013, and represents regions flanking 400 bp upstream and 600 bp downstream of the dominant TSS detected by Cap analysis of gene expression (CAGE). It is intended to be used as an input data in running examples from `seqPattern` package help pages.

Usage

```
data(zebrafishPromoters)
```

Format

A `DNAStringSet` object

Value

A `DNAStringSet` object

References

Nepal *et al.* (2013) Dynamic regulation of the transcription initiation landscape at single nucleotide resolution during vertebrate embryogenesis, *Genome Research* **23**(11):1938-1950.

`zebrafishPromoters24h` *Zebrafish promoters genomic coordinates*

Description

This is a `data.frame` object that contains genomic coordinates of 12078 promoters active in zebrafish *Danio rerio* embryos at 24 hours past fertilisation. For each promoter additional information is provided, including position of the dominant (most frequently used) TSS position, number of CAGE tags per million supporting that promoter and the interquantile width of the promoter (width of the central region containing >= 80% of the CAGE tags). The data is taken from Nepal *et al.* Genome Research 2013, and it is intended to be used for running examples from `seqPattern` package vignette.

Usage

```
data(zebrafishPromoters24h)
```

Format

A `data.frame` object

Value

A `data.frame` object

References

Nepal *et al.* (2013) Dynamic regulation of the transcription initiation landscape at single nucleotide resolution during vertebrate embryogenesis, *Genome Research* **23**(11):1938-1950.

Index

∗ **datasets**
TBPpwm, 19
zebrafishPromoters, 19
zebrafishPromoters24h, 20
∗ **package**
seqPattern-package, 2

bkde2D, *8*, *9*, *15*

data.frame, *20*
DNAStringSet, *3*, *5*, *6*, *8*, *11*, *12*, *14*, *17*, *19*

getPatternOccurrenceList, 3, *5*, *16*, *18*
getPatternOccurrenceList,DNAStringSet-method (getPatternOccurrenceList), 3

matchPattern, *3*
matchPWM, *5*, *8*, *11*
matrix, *19*
motifScanHits, *4*, 4, *7*, *10*, *12*
motifScanHits,DNAStringSet,matrix-method (motifScanHits), 4
motifScanScores, 6, *14*
motifScanScores,DNAStringSet,matrix-method (motifScanScores), 6

plotMotifDensityMap, *5*, 7, *12*, *16*
plotMotifDensityMap,DNAStringSet,matrix-method (plotMotifDensityMap), 7
plotMotifOccurrenceAverage, 10
plotMotifOccurrenceAverage,DNAStringSet,matrix-method (plotMotifOccurrenceAverage), 10
plotMotifScanScores, *7*, 12
plotMotifScanScores,DNAStringSet,matrix-method (plotMotifScanScores), 12
plotPatternDensityMap, *4*, *10*, *14*, 14, *18*
plotPatternDensityMap,DNAStringSet-method (plotPatternDensityMap), 14
plotPatternOccurrenceAverage, 17
plotPatternOccurrenceAverage,DNAStringSet-method (plotPatternOccurrenceAverage), 17
PWM, *4*–*6*, *8*, *11*, *12*
PWMscoreStartingAt, *7*

seqPattern, *19*, *20*
seqPattern (seqPattern-package), 2
seqPattern-package, 2

TBPpwm, 19

zebrafishPromoters, 19
zebrafishPromoters24h, 20