Package ‘DepLogo’

March 13, 2019

Type Package
Title Dependency Logo
Version 1.0
Maintainer Jan Grau <grau@informatik.uni-halle.de>
Description Plots dependency logos from a set of input sequences.
License GPL-3
NeedsCompilation no
RoxygenNote 6.0.1
VignetteBuilder knitr
Suggests knitr, testthat
Author Jan Grau [aut, cre],
     Jens Keilwagen [aut],
     Martin Nettling [aut]
Repository CRAN
Date/Publication 2019-03-13 16:23:27 UTC

R topics documented:

Alphabet ................................................. 2
alphabet.dna ........................................... 3
alphabet.dna.gap ...................................... 3
alphabet.protein ...................................... 3
alphabet.protein.gap .................................. 4
alphabet.rna .......................................... 4
alphabet.rna.gap ...................................... 4
colorchart ............................................. 5
deprects ............................................... 6
DLData .................................................. 7
filter.by.conservation .............................. 8
filter.by.dependencies ............................ 9
filter.by.gaps ....................................... 10
filterColumns ....................................... 10
Description

builds an object of class Alphabet from the given set of symbols and colors

Usage

Alphabet(chars, cols)

Arguments

chars            set of symbols
cols             set of colors; one for each symbol

Value

the Alphabet object

Author(s)

Martin Nettling

Examples

DNA <- Alphabet(c("A", "C", "G", "T"), c("green4", "blue", "orange", "red"))
**alphabet.dna**

**DNA alphabet**

**Description**

DNA alphabet

**Usage**

alphabet.dna

**Format**

An object of class list of length 2.

---

**alphabet.dna.gap**

**DNA alphabet with gaps**

**Description**

DNA alphabet with gaps

**Usage**

alphabet.dna.gap

**Format**

An object of class list of length 2.

---

**alphabet.protein**

**Amino acid alphabet**

**Description**

Amino acid alphabet

**Usage**

alphabet.protein

**Format**

An object of class list of length 2.
<table>
<thead>
<tr>
<th><strong>alphabet.protein.gap</strong></th>
<th><em>Amino acid alphabet with gaps</em></th>
</tr>
</thead>
</table>

**Description**

Amino acid alphabet with gaps

**Usage**

alphabet.protein.gap

**Format**

An object of class list of length 2.

<table>
<thead>
<tr>
<th><strong>alphabet.rna</strong></th>
<th><em>RNA alphabet</em></th>
</tr>
</thead>
</table>

**Description**

RNA alphabet

**Usage**

alphabet.rna

**Format**

An object of class list of length 2.

<table>
<thead>
<tr>
<th><strong>alphabet.rna.gap</strong></th>
<th><em>RNA alphabet with gaps</em></th>
</tr>
</thead>
</table>

**Description**

RNA alphabet with gaps

**Usage**

alphabet.rna.gap

**Format**

An object of class list of length 2.
colorchart

Plot a colorchart representation of a set of sequences.

Description

This function is a low-level plotting function (using image with add=TRUE, internally).

Usage

colorchart(part, yoff, ic.scale = TRUE)

Arguments

part the set of sequences as DLData object
yoff the offset in y-direction within the current plot
ic.scale ignored for colorcharts

Value

the vertical (y) offset after this plot

Author(s)

Jan Grau <grau@informatik.uni-halle.de>

Examples

# read data and create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
                   stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[,1], weights = log1p(seqs[, 2]) )

# create high-level plot
plot(NULL, xlim = c(1, ncol(data$data) - 1), ylim = c(0, nrow(data$data)),
     ylab = nrow(data$data), axes = FALSE)

# and add colorchart and axis
colorchart(data, yoff = nrow(data$data))
axis(1)
deprects

*Rectangles of averaged colors*

**Description**

Plot a representation of a set of sequences by rectangles of (scaled) averaged color values of the symbols at each position.

**Usage**

```r
deprects(part, yoff, ic.scale = TRUE)
```

**Arguments**

- `part` the set of sequences as `DLData` object
- `yoff` the offset in y-direction within the current plot
- `ic.scale` if TRUE, alpha values of colors will be assigned based on "information content" of the distribution at each position

**Details**

This function is a low-level plotting function (using `rect`, internally).

**Value**

the vertical (y) offset after this plot

**Author(s)**

Jan Grau <grau@informatik.uni-halle.de>

**Examples**

```r
# read data and create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
                   stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[, 1], weights = log1p(seqs[, 2]))

# create high-level plot
plot(NULL, xlim = c(1, ncol(data$data) - 1), ylim = c(0, nrow(data$data)),
     ylab = nrow(data$data), axes = FALSE)
# and add deprects and axis
deprects(data, yoff = nrow(data$data))
axis(1)
```
Create DLDData object

Description

Creates a new DLDData object from a set of input sequences.

Usage

DLDData(sequences, weights = NULL, symbols = alphabet.dna$alphabet,
    colors = alphabet.dna$colors, delim = "",
    sortByWeights = !is.null(weights), axis.labels = NULL)

Arguments

- **sequences**: the input sequences, may be provided as i) "character" vector or ii) a data.frame with sequences organized in rows and one symbol per column
- **weights**: weights associated with the sequences, numeric vector of the same length as sequences has sequences
- **symbols**: the symbols (alphabet) over which the sequences are defined
- **colors**: colors for each of the symbols, not necessarily unique
- **delim**: delimiter between the symbols in the input sequences, ignored if sequences as a data.frame
- **sortByWeights**: if TRUE, sequences will be ordered by their weight in decreasing order
- **axis.labels**: the labels of the individual sequence positions; if NULL, indexes from 1 to to total number of positions will be used

Details

Sequences may either be provided as a "character" vector or as a data.frame. All symbols occurring in these sequences need to be defined and assigned to colors, which are used for plotting later. Colors do not need to be unique, but symbols with identical colors may become indistinguishable in subsequent plots (which might even be desired, for instance, when visualizing protein properties instead of amino acids). Sequences may have an associated weight, which is used to order sequences, e.g., for creating chunks/blocks of sequences in subsequent plots (see chunks parameter of plotDeplogo).

Value

- the DLDData object

Author(s)

- Jan Grau <grau@informatik.uni-halle.de>
See Also

plotDeplogo

Examples

# creating a DLData object using default (DNA) alphabet and colors  
# from a character vector with two entries  
data <- DLData(c("ACGT", "ATTA"))

# creating a DLData object using a custom, binary alphabet and custom colors  
symbols = c("A", "B"), colors = c("red", "green"), delim = ",")

# creating a DLData object from a data frame  
# (created from a character vector, in this case)  
df <- as.data.frame(t(sapply(vec, function(a){strsplit(a, ",")[[1]]})))  
data.df <- DLData(df, symbols = c("A", "B"), colors = c("red", "green"))

# creating a DLData object from sequences and weights, read from a tabular file  
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),  
stringsAsFactors = FALSE)  
data3 <- DLData(sequences = seqs[, 1], weights = log1p(seqs[, 2]) )

---

filter.by.conservation

*Filter columns (sequence positions) by conservation*

**Description**

Filters columns based on the relative information content of each column which is the standard information content normalized to the interval [0,1], where 0 corresponds to uniform distribution and 1 to perfect conservation of one nucleotide or amino acid, respectively.

**Usage**

`filter.by.conservation(relative.ic)`

**Arguments**

- `relative.ic` the maximum relative information content allowed to retain a position

**Value**

function that, given a DLData object, returns TRUE for every column that does not exceed the specified relative information content
Author(s)
Jan Grau <grau@informatik.uni-halle.de>

Examples
fun <- filter.by.conservation(relative.ic = 0.9)

filter.by.dependencies

Filter columns (sequence positions) by dependency

Description
Filters columns based on the average or maximum mutual information of a column to all other columns. Mutual information is normalized to interval $[0,1]$, where 0 corresponds to independence and 1 to perfect dependence.

Usage
filter.by.dependencies(mi.threshold, use.max = FALSE)

Arguments
mi.threshold  the minimum average or maximum mutual information required
use.max  if TRUE, the maximum and otherwise the average mutual information will be considered

Value
function that, given a DLData object, returns TRUE for every column that does exceed the specified average mutual information

Author(s)
Jan Grau <grau@informatik.uni-halle.de>

Examples
fun <- filter.by.dependencies(mi.threshold = 0.3)
filter.by.gaps  
*Filter columns (sequence positions) by gaps*

**Description**
Filter columns (sequence positions) by gaps

**Usage**

```
filter.by.gaps(percent.gap)
```

**Arguments**

- `percent.gap` the maximum fraction of gaps allowed to retain a column

**Value**
function that, given a `DLData` object, returns `TRUE` for every column that does not exceed the specified number of gaps

**Author(s)**
Jan Grau <grau@informatik.uni-halle.de>

**Examples**

```
fun <- filter.by.gaps(percent.gap = 0.1)
```

---

filterColumns  
*Filter data columns by some filter function*

**Description**
Filters the columns of the input data, i.e., positions of input sequences, by a filter function that, given a `DLData` object, returns a list containing i) as element $\text{selected}$ a vector with entries `TRUE` for every column that should be retained in the filtered data and ii) as element $\text{range}$ the range of values obtained for the filtering criterion

**Usage**

```
filterColumns(data, filter.fun)
```

**Arguments**

- `data` the data as `DLData` object
- `filter.fun` the filter function
**getDeps**

**Value**

a DLData object containing the filtered columns and the indexes of the remaining in its axis.labels field

**Author(s)**

Jan Grau <grau@informatik.uni-halle.de>

**See Also**

filter.by.gaps
filter.by.dependencies
filter.by.conservation

**Examples**

```
# read data and create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
  stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[,1], weights = log1p(seqs[,2]))

# create a filter function based on the percentage of gap symbols (at most 10%)
fun <- filter.by.gaps(percent.gap = 0.1)
data2 <- filterColumns(data, fun)
```

---

**getDeps**  
*Compute dependencies between positions*

**Description**

Computes the dependencies (as measures by mutual information) between all positions (columns) of discrete data. Specifically, it returns for each pair of positions (i,j) the mutual information I(X_i,X_j) multiplied by the number N of sequences (rows), which may also be used for testing the statistical significance of mutual information values, as for large N, 2*N*I(X_i,X_j) is approximately chi squared.

**Usage**

getDeps(data, ...)

**Arguments**

data  
the data for computing mutual information. Either a DLData object or a data.frame. In the latter case, the symbols of the alphabet must be provided as a second parameter.

...  
the symbols of the alphabet as character vector, only if data is a data.frame
**getPWM**

**Position weight matrix from DLData object**

**Description**

Determines the position weight matrix from a DLData object as relative frequency of symbols in each column of the data slot.

**Usage**

```r
getPWM(part)
```

**Arguments**

- `part` the DLData object

**Value**

The position weight matrix, where columns correspond to positions (columns of the DLData$data slot) and rows to symbols.

**Author(s)**

Jan Grau <grau@informatik.uni-halle.de>

**Examples**

```r
data <- DLData(c("ACGT", "ATTA"))
deps <- getDeps(data)

data <- DLData(c("ACGT", "ATTA"))
getPWM(data)
```
Sequence logo

Description

Plot a representation of a set of sequences as a sequence logo.

Usage

logo(part, yoff, ic.scale = TRUE)

Arguments

- **part**: the set of sequences as DLData object
- **yoff**: the offset in y-direction within the current plot
- **ic.scale**: if TRUE, symbols are scaled by "information content" of the distribution at each position

Details

This function is a low-level plotting function (using polygon, internally).

Value

the vertical (y) offset after this plot

Author(s)

Jan Grau <grau@informatik.uni-halle.de>

Examples

```r
# read data and create DLData object
seqs <- read.table(system.file("extdata", "cjum.txt", package = "DepLogo"),
                   stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[, 1], weights = log1p(seqs[,2]) )

# create high-level plot
plot(NULL, xlim = c(1, ncol(data$data) - 1), ylim = c(0, nrow(data$data)),
ylab = nrow(data$data), axes = FALSE)
# and add sequence logo and axis
logo(data, yoff = nrow(data$data))
axis(1)
```
**partition**

*Partition data by most inter-dependent positions*

**Description**

Partitions data by the nucleotides at the most inter-dependent positions as measures by pairwise mutual information. Partitioning is performed recursively on the resulting subsets until i) the number of sequences in a partition is less then `minelements`, ii) the average pairwise dependency between the current position and `numBestForSorting` other positions with the largest mutual information value drops below `threshold`, or iii) `maxNum` recursive splits have already been performed. If splitting results in smaller partitions than `minelements`, these are added to the smallest partition with more than `minelements` sequences.

**Usage**

```r
partition(data, minelements = 10, threshold = 0.1, numBestForSorting = 3, maxNum = 6, sortByWeights = NULL)
```

**Arguments**

- `data` the data as `DLData` object
- `minelements` the minimum number of elements to perform a further split.
- `threshold` the threshold on the average mutual information value
- `numBestForSorting` the number of dependencies to other positions considered
- `maxNum` the maximum number of recursive splits
- `sortByWeights` if `TRUE`, partitions are ordered by their average weight value, if `false` by frequency of symbols at the partitioning position otherwise. If `NULL`, the `$sortByWeights` value of the `DLData` object is used.

**Value**

the partitions as list of `DLData` objects

**Author(s)**

Jan Grau <grau@informatik.uni-halle.de>

**Examples**

```r
# create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
                  stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[, 1], weights = log1p(seqs[,2]) )

# partition data using default parameters
partitions <- partition(data)
```
# partition data using a threshold of 0.3 on the mutual
# information value to the most dependent position,
# sorting the resulting partitions by weight
partitions2 <- partition(data = data, threshold = 0.3, numBestForSorting = 1, sortByWeights = TRUE)

---

**plotBlocks**  
Plots blocks of data

## Description

Plots the blocks of data in `data` by successive, vertically arranged sub-plots of the function provided as `block.fun`. If `data` is a single `DLData` object, one block is plotted. Further arguments are provided to `block.fun`

## Usage

```r
plotBlocks(data, show.number = TRUE, block.fun = deprects,
           ic.scale = TRUE, add = FALSE, ...)
```

## Arguments

- **data**: the data, a single `DLData` object or a list of `DLData` objects
- **show.number**: if true, the number of sequences (in total) in `data` is displayed on the left side of the plot
- **block.fun**: the function called for each of the blocks
- **ic.scale**: if TRUE, output of `block.fun` may be scaled by "information content"
- **add**: if TRUE, the plot is added to an existing plot
- **...**: if add=FALSE forwarded to the internal call to `plot`

## Author(s)

Jan Grau <grau@informatik.uni-halle.de>

## See Also

- `deprects`
- `logo`
- `colorchart`
Examples

```r
# read data and create DLDData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
   stringsAsFactors = FALSE)
data <- DLDData(sequences = seqs[, 1], weights = log1p(seqs[, 2]))

# plot all data
plotBlocks(data)

# partition data
partitions <- partition(data, threshold = 0.3)
# and plot partitions
plotBlocks(partitions)

# or plot partitions as sequence logos
plotBlocks(partitions, block.fun = logo)
```

`plotDeparcs`  
Plot a graph representation of dependency values.

Description

Plots a representation of dependency values as arcs between the sequence positions. Internally, dependency values are computed using `getDeps` on the `data` object.

Usage

```r
plotDeparcs(data, axis.at.bottom = TRUE, add.legend = TRUE,
   show.pvals = FALSE, axis.labels = NULL, threshold = 0.1)
```

Arguments

- `data`: the `DLDData` object containing the data
- `axis.at.bottom`: if TRUE, the x-axis is shown at the bottom (side=1) of the plot, and at the top (side=3) otherwise
- `add.legend`: if TRUE a legend of the color scale is added to the plot
- `show.pvals`: if TRUE, -log10 p-values (computed by `pchisq`) are shown instead of mutual information values
- `axis.labels`: the labels of the x-axis
- `threshold`: threshold in mutual information values, edges below this value are not shown; ignored in `show.pvals=TRUE`

Author(s)

Jan Grau <grau@informatik.uni-halle.de>
Examples

```r
# create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
                    stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[,1], weights = log1p(seqs[, 2]) )

# plot using default parameters
plotDeparcs(data)

# plot with axis at top, without a legend (color scale), and using p-values
plotDeparcs(data, axis.at.bottom = FALSE, add.legend = FALSE, show.pvals = TRUE)
```

Description

Plots a dependency logo.

Usage

```r
plotDeplogo(data, dep.fun = plotDeparcs, block.fun = deprects,
summary.fun = logo, weight.fun = NULL, chunks = NULL,
chunk.height = 800, summary.height = 100, minPercent = 0.03,
threshold = 0.1, numBestForSorting = 3, maxNum = 6,
sortByWeights = NULL, dep.fun.legend = TRUE,
show.dependency.pvals = FALSE, axis.labels = NULL, ...)
```

Arguments

data the data, currently implemented for DLData objects
dep.fun the function for plotting the representation of dependency values (as computed by getDeps)
block.fun the function for plotting a representation of the individual partitions of the data generated in dependency logos.
summary.fun the function for plotting a representation of the summary plot for (one chunk of) the data
weight.fun the function for plotting a representation of the weights values of the sequences within one partition
chunks the size of chunks the data is split into. The sum of the chunk sizes must not be greater than the number of data points in data. The default value of NULL corresponds to one chunk containing all data points.
chunk.height the (relative) height of the parts of the plot representing each of the chunks, one height for each chunk.
summary.height the (relative) height of the block summaries in the plot.
minPercent the minimum percentage of the (sub) data set that may constitute its own partition in the dependency logo.

threshold the threshold on the dependency value for further splits.

numBestForSorting the number of dependencies between position i and all other positions when computing the dependency value of position i.

maxNum the maximum number of splits allowed.

sortByWeights are partitions sorted by their average weight (descending).

dep.fun.legend if TRUE, a legend of the color scale used for plotting the dependency values in dep.fun is added to the plot.

show.dependency.pvals if TRUE, p-values are used for plotting dependency values in dep.fun instead of mutual information values.

axis.labels labels for the x-axis, vector of the same length as the individual sequences.

Details

The function dep.fun provided for plotting the representation of dependencies is currently implemented in plotDeparcs and plotDepmatrix. Custom implementations must have the same signature as these functions and create a single plot without using layout (or similar).

The functions block.fun and summary.fun provided for plotting the representation of individual partitions of the data generated in dependency logos are currently implemented in deprects, colorchart, and logo. Custom implementations must have the same signature as these functions and create a single plot without using layout (or similar).

The function weight.fun for plotting a representation of the weights values of the sequences within one partition is currently implemented in subLines and subBoxes. Custom implementations must have the same signature as these functions and create a single plot without using layout (or similar).

Value

a list of DLData objects with the partitions created for the dependency logo

Author(s)

Jan Grau <grau@informatik.uni-halle.de>

Examples

```r
# read data and create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"), stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[, 1], weights = log1p(seqs[, 2]) )

# plot default dependency logo
plotDeplogo(data)
```
# refine threshold for clearer picture
plotDeplogo(data, threshold = 0.3)

# customize different parts of the plot
plotDeplogo(data, threshold = 0.3, dep.fun = plotDepmatrix, block.fun = colorchart)

# add plots of the weights
plotDeplogo(data, weight.fun = subBoxes)

---

**plotDepmatrix**  
*Plot a matrix representation of dependency values.*

## Description

Plots a representation of dependency values as a triangular matrix rotated by 45 degrees. Internally, dependency values are computed using `getDeps` on the data object.

## Usage

```
plotDepmatrix(data, axis.at.bottom = TRUE, add.legend = TRUE, show.pvals = FALSE, axis.labels = NULL, threshold = 0.1)
```

## Arguments

- `data`  
  the `DLData` object containing the data

- `axis.at.bottom`  
  if `TRUE`, the x-axis is shown at the bottom (side=1) of the plot, and at the top (side=3) otherwise

- `add.legend`  
  if `TRUE` a legend of the color scale is added to the plot

- `show.pvals`  
  if `TRUE`, -log10 p-values (computed by `pchisq`) are shown instead of mutual information values

- `axis.labels`  
  the labels of the x-axis

- `threshold`  
  ignored

## Author(s)

Jan Grau <grau@informatik.uni-halle.de>

## Examples

```
# create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"), stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[, 1], weights = log1p(seqs[, 2]))

# plot using default parameters
plotDepmatrix(data)
```
replaceColors

Replace colors in DLData object

Description
Replace colors in DLData object

Usage
replaceColors(data, colors)

Arguments
data the data
colors the new colors

Value
the modified DLData object

Author(s)
Jan Grau <grau@informatik.uni-halle.de>

See Also
replaceColors

Examples
# read data and create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
  stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[, 1], weights = log1p(seqs[, 2]))

replaceColors(data, c("red", "green", "blue", "yellow"))
### revcom

**Reverse complement**

**Description**
Determine the reverse complementary `DLData` object. Only works for DNA or RNA. Data may include gap symbols.

**Usage**

```r
revcom(data)
```

**Arguments**

- `data` the data

**Value**

the reverse complement

**Author(s)**
Jan Grau <grau@informatik.uni-halle.de>

**Examples**

```r
data <- DLData(c("ACGT", "ATTA"))
revcom(data)
```

---

### subBoxes

**Plot weights as boxplots**

**Description**
Plots a representation of the weights of a list of `DLData` objects. Each entry of the list is shown as an independent boxplot.

**Usage**

```r
subBoxes(sub.parts, range, axis.above = TRUE, axis.below = TRUE)
```

**Arguments**

- `sub.parts` a list of `DLData` objects
- `range` the range of values shown in the plot (i.e., the `xlim` value of the call to `plot`)
- `axis.above` if `TRUE`, an axis at the top of the plot (side=3) is shown
- `axis.below` if `TRUE`, an axis at the bottom of the plot (side=1) is shown
Author(s)
Jan Grau <grau@informatik.uni-halle.de>

Examples

# read data and create DLDa object
seqs <- read.table(system.file("extdata", "nrsf.txt", package = "DepLogo"),
                   stringsAsFactors = FALSE)
data <- DLDa(sequences = seqs[, 1], weights = log1p(seqs[, 2])

# create dependency logo with plotted weights
plotDeplogo(data, threshold = 0.03, weight.fun = subBoxes)

subLines

Plot weights as lines

Description

Plots a representation of the weights of a list of DLDa objects. Each entry of the list is shown
as an independent line with the median value shown as a red vertical line. Plots of list entries are
separated by horizontal grey lines.

Usage

subLines(subParts, range, axisAbove = TRUE, axisBelow = TRUE)

Arguments

subParts a list of DLDa objects
range the range of values shown in the plot (i.e., the xlim value of the call to plot)
axisAbove if TRUE, an axis at the top of the plot (side=3) is shown
axisBelow if TRUE, an axis at the bottom of the plot (side=1) is shown

Author(s)
Jan Grau <grau@informatik.uni-halle.de>

Examples

# read data and create DLDa object
seqs <- read.table(system.file("extdata", "nrsf.txt", package = "DepLogo"),
                   stringsAsFactors = FALSE)
data <- DLDa(sequences = seqs[, 1], weights = log1p(seqs[, 2])

# create dependency logo with plotted weights
plotDeplogo(data, threshold = 0.03, weight.fun = subLines)
suggestColors

Suggest colors for symbols

Description

Suggests colors for the symbols in data based on the co-occurrence of symbols at common positions, weighted by the dependency values at those positions. The idea is to assign similar colors only to symbols that either mostly occur at different positions or that are present at positions with low inter-dependencies to other positions.

Usage

suggestColors(data)

Arguments

data the data

Value

the colors

Author(s)

Jan Grau <grau@informatik.uni-halle.de>

See Also

replaceColors

Examples

# read data and create DLData object
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[, 1], weights = log1p(seqs[, 2]))
suggestColors(data)
summary.DLData  

**Summarizing DLData objects**

**Description**

summary method for class "DLData". The summary includes the number of sequences, the consensus sequence and the number of sequences in object that match the consensus.

**Usage**

```r
## S3 method for class 'DLData'
summary(object, delete.gaps = FALSE, ...)
```

**Arguments**

- `object` an object of class "DLData"
- `delete.gaps` if gaps should be removed from the consensus
- `...` further arguments passed to or from other methods

**Value**

a list with elements `members` containing the number of sequences, `consensus` containing the consensus sequences, and `equal.consentus` containing the number of sequences in object that are identical to consensus

**Author(s)**

Jens Keilwagen, Jan Grau <grau@informatik.uni-halle.de>

**Examples**

```r
seqs <- read.table(system.file("extdata", "cjun.txt", package = "DepLogo"),
                  stringsAsFactors = FALSE)
data <- DLData(sequences = seqs[, 1], weights = log1p(seqs[, 2]) )
summary(data)
```
# Index

*Topic **datasets**

- alphabet.dna, 3
- alphabet.dna.gap, 3
- alphabet.protein, 3
- alphabet.protein.gap, 4
- alphabet.rna, 4
- alphabet.rna.gap, 4

- Alphabet, 2
  - alphabet.dna, 3
  - alphabet.dna.gap, 3
  - alphabet.protein, 3
  - alphabet.protein.gap, 4
  - alphabet.rna, 4
  - alphabet.rna.gap, 4

- colorchart, 5, 15, 18

- deprects, 6, 15, 18

- DLDData, 5, 6, 7, 8–11, 13–22

- filter.by.conservation, 8, 11
- filter.by.dependencies, 9, 11
- filter.by.gaps, 10, 11
- filterColumns, 10

- getDeps, 11, 16, 17, 19
- getPWM, 12

- image, 5

- layout, 18
- logo, 13, 15, 18

- partition, 14
- pchisq, 16, 19
- plot, 15, 21, 22
- plotBlocks, 15
- plotDeparcs, 16, 18
- plotDeplogo, 8, 17
- plotDepmatrix, 18, 19

- polygon, 13
- rect, 6
- replaceColors, 20, 20, 23
- revcom, 21

- subBoxes, 18, 21
- subLines, 18, 22
- suggestColors, 23
- summary.DLDData, 24