

# Package ‘DepLogo’

October 12, 2022

**Type** Package

**Title** Dependency Logo

**Version** 1.2

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

**Description** Plots dependency logos from a set of aligned input sequences.

**License** GPL-3

**NeedsCompilation** no

**RoxygenNote** 7.1.0

**VignetteBuilder** knitr

**Suggests** knitr, rmarkdown, testthat

**Author** Jan Grau [aut, cre],  
Jens Keilwagen [aut],  
Martin Nettling [aut]

**Repository** CRAN

**Date/Publication** 2021-04-16 20:10:02 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

getDeps . . . . .	11
getPWM . . . . .	12
logo . . . . .	13
partition . . . . .	14
plotBlocks . . . . .	15
plotDeparcs . . . . .	16
plotDeplogo . . . . .	17
plotDepmatrix . . . . .	19
replaceColors . . . . .	20
revcom . . . . .	21
subBoxes . . . . .	22
subLines . . . . .	23
suggestColors . . . . .	23
summary.DLData . . . . .	24

<b>Index</b>	<b>26</b>
--------------	-----------

---

Alphabet	<i>built alphabet</i>
----------	-----------------------

---

## 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"))
```

---

<code>alphabet.dna</code>	<i>DNA alphabet</i>
---------------------------	---------------------

---

**Description**

DNA alphabet

**Usage**

`alphabet.dna`

**Format**

An object of class `list` of length 2.

---

<code>alphabet.dna.gap</code>	<i>DNA alphabet with gaps</i>
-------------------------------	-------------------------------

---

**Description**

DNA alphabet with gaps

**Usage**

`alphabet.dna.gap`

**Format**

An object of class `list` of length 2.

---

<code>alphabet.protein</code>	<i>Amino acid alphabet</i>
-------------------------------	----------------------------

---

**Description**

Amino acid alphabet

**Usage**

`alphabet.protein`

**Format**

An object of class `list` of length 2.

alphabet.protein.gap    *Amino acid alphabet with gaps*

---

**Description**

Amino acid alphabet with gaps

**Usage**

alphabet.protein.gap

**Format**

An object of class `list` of length 2.

---

alphabet.rna            *RNA alphabet*

---

**Description**

RNA alphabet

**Usage**

alphabet.rna

**Format**

An object of class `list` of length 2.

---

alphabet.rna.gap        *RNA alphabet with gaps*

---

**Description**

RNA alphabet with gaps

**Usage**

alphabet.rna.gap

**Format**

An object of class `list` of length 2.

---

colorchart	<i>Plots a colorchart representation of a set of sequences</i>
------------	--

---

### Description

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

### Usage

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

### Arguments

<code>part</code>	the set of sequences as <a href="#">DLData</a> object
<code>yoff</code>	the offset in y-direction within the current plot
<code>ic.scale</code>	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

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

### Usage

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

### Arguments

part	the set of sequences as <a href="#">DLData</a> 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

```
# 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)
```

---

DLData                      *Create DLData object*

---

### Description

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

### Usage

```
DLData(  
  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 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 DLData 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
data2 <- DLData(c("A,B,B,A,B", "A,B,B,A,A", "A,B,A,A,B"),
  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)
vec <- c("A,B,B,A,B", "A,B,B,A,A", "A,B,A,A,B")
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

*Filters 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
```

*Filters 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**

<code>mi.threshold</code>	the minimum average or maximum mutual information required
<code>use.max</code>	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	<i>Filters columns (sequence positions) by gaps</i>
----------------	---

---

**Description**

Filters 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	<i>Filters data columns by some filter function</i>
---------------	---

---

**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 \$selected a vector with entries TRUE for every column that should be retained in the filtered data and ii) as element \$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

**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

**Value**

a matrix of the mutual information values, where the diagonal is fixed to zero

**Author(s)**

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

**Examples**

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

---

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**

```
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**

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

---

logo

*Sequence logo*

---

## Description

Plots 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 <a href="#">DLData</a> 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

```
# 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$data), axes = FALSE)
# and add sequence logo and axis
logo(data, yoff = nrow(data$data))
axis(1)
```

---

partition

*Partitions 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 than `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

```
partition(  
  data,  
  minElements = 10,  
  threshold = 0.1,  
  numBestForSorting = 3,  
  maxNum = 6,  
  sortByWeights = NULL,  
  partition.by = NULL  
)
```

### Arguments

<code>data</code>	the data as <a href="#">DLData</a> object
<code>minElements</code>	the minimum number of elements to perform a further split
<code>threshold</code>	the threshold on the average mutual information value
<code>numBestForSorting</code>	the number of dependencies to other positions considered
<code>maxNum</code>	the maximum number of recursive splits
<code>sortByWeights</code>	if TRUE, partitions are ordered by their average weight value, if false by frequency of symbols at the partitioning position otherwise. If NULL, the <code>sortByWeights</code> value of the <a href="#">DLData</a> object is used
<code>partition.by</code>	specify fixed positions to partition by

### Value

the partitions as list of [DLData](#) objects

### 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]) )

# 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**

```

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

```

**Arguments**

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

**Author(s)**

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

**See Also**

[deprects](#)  
[logo](#)  
[colorchart](#)

**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]) )

# 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

*Plots 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 [getDepts](#) on the data object.

**Usage**

```
plotDeparcs(
  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, $-\log_{10}$ p-values (computed by <code>pchisq</code> ) 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 <code>show.pvals=TRUE</code>

**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
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)
```

---

plotDeplogo	<i>Plot a dependency logo</i>
-------------	-------------------------------

---

**Description**

Plots a dependency logo

**Usage**

```
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,
weight.ratio = 5,
partition.by = NULL,
...
)

```

## Arguments

<code>data</code>	the data, currently implemented for <a href="#">DLData</a> objects
<code>dep.fun</code>	the function for plotting the representation of dependency values (as computed by <a href="#">getDeps</a> )
<code>block.fun</code>	the function for plotting a representation of the individual partitions of the data generated in dependency logos.
<code>summary.fun</code>	the function for plotting a representation of the summary plot for (one chunk of) the data
<code>weight.fun</code>	the function for plotting a representation of the weights values of the sequences within one partition
<code>chunks</code>	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 <code>data</code> ; The default value of <code>NULL</code> corresponds to one chunk containing all data points
<code>chunk.height</code>	the (relative) height of the parts of the plot representing each of the chunks, one height for each chunk
<code>summary.height</code>	the (relative) height of the block summaries in the plot
<code>minPercent</code>	the minimum percentage of the (sub) data set that may constitute its own partition in the dependency logo
<code>threshold</code>	the threshold on the dependency value for further splits
<code>numBestForSorting</code>	the number of dependencies between position <code>i</code> and all other positions when computing the dependency value of position <code>i</code>
<code>maxNum</code>	the maximum number of splits allowed
<code>sortByWeights</code>	are partitions sorted by their average weight (descending)
<code>dep.fun.legend</code>	if <code>TRUE</code> , a legend of the color scale used for plotting the dependency values in <code>dep.fun</code> is added to the plot
<code>show.dependency.pvals</code>	is <code>TRUE</code> , p-values are used for plotting dependency values in <code>dep.fun</code> instead of mutual information values
<code>axis.labels</code>	labels for the x-axis, vector of the same length as the individual sequences
<code>weight.ratio</code>	the factor by which the plotting width for the main plot is larger than for <code>weight.fun</code>
<code>partition.by</code>	specify fixed positions to partition by
<code>...</code>	forwarded to the high-level plot that contains the blocks plotted by <code>block.fun</code>

## 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

```
# 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

*Plots 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 <a href="#">DLData</a> 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 <a href="#">pchisq</a> ) 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)  
  
# plot with axis at top, without a legend (color scale), and using p-values  
plotDepmatrix(data, axis.at.bottom = FALSE, add.legend = FALSE, show.pvals = TRUE)
```

---

replaceColors

*Replaces colors in [DLData](#) object*

---

**Description**

Replaces 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**

```
revcom(data)
```

**Arguments**

data	the data
------	----------

**Value**

the reverse complement

**Author(s)**

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

**Examples**

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

---

subBoxes

*Plots 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**

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

**Arguments**

sub.parts	a list of <a href="#">DLData</a> objects
range	the range of values shown in the plot (i.e., the <code>xlim</code> value of the call to <a href="#">plot</a> )
axis.above	if TRUE, an axis at the top of the plot ( <code>side=3</code> ) is shown
axis.below	if TRUE, an axis at the bottom of the plot ( <code>side=1</code> ) is shown

**Author(s)**

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

**Examples**

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

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

---

subLines	<i>Plots weights as lines</i>
----------	-------------------------------

---

**Description**

Plots a representation of the weights of a list of [DLData](#) 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(sub.parts, range, axis.above = TRUE, axis.below = TRUE)
```

**Arguments**

sub.parts	a list of <a href="#">DLData</a> objects
range	the range of values shown in the plot (i.e., the <code>xlim</code> value of the call to <a href="#">plot</a> )
axis.above	if TRUE, an axis at the top of the plot ( <code>side=3</code> ) is shown
axis.below	if TRUE, an axis at the bottom of the plot ( <code>side=1</code> ) is shown

**Author(s)**

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

**Examples**

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

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

---

suggestColors	<i>Suggests colors for symbols</i>
---------------	------------------------------------

---

**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**

```
## 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.consensus` containing the number of sequences in object that are identical to consensus

**Author(s)**

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

**Examples**

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

# Index

## \* 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, 16, 19

deprects, 6, 16, 19

DLData, 5, 6, 7, 8–11, 13–16, 18–23

filter.by.conservation, 8, 11

filter.by.dependencies, 9, 11

filter.by.gaps, 10, 11

filterColumns, 10

getDeps, 11, 16, 18, 19

getPWM, 12

image, 5

layout, 19

logo, 13, 16, 19

partition, 14

pchisq, 17, 20

plot, 15, 22, 23

plotBlocks, 15

plotDeparcs, 16, 19

plotDeplogo, 8, 17

plotDepmatrix, 19, 19

polygon, 13

rect, 6

replaceColors, 20, 21, 24

revcom, 21

subBoxes, 19, 22

subLines, 19, 23

suggestColors, 23

summary.DLData, 24