Package ‘dagLogo’

May 29, 2024

Type Package

Title dagLogo: a Bioconductor package for visualizing conserved amino acid sequence pattern in groups based on probability theory

Version 1.42.0

Author Jianhong Ou, Haibo Liu, Alexey Stukalov, Niraj Nirala, Usha Acharya, Lihua Julie Zhu

Maintainer Jianhong Ou <jianhong.ou@duke.edu>

Description

Visualize significant conserved amino acid sequence pattern in groups based on probability theory.

License GPL (>=2)

Depends R (>= 3.0.1), methods, grid

Imports pheatmap, Biostrings, UniProt.ws, BiocGenerics, utils, biomaRt, motifStack, httr

Suggests XML, grImport, grImport2, BiocStyle, knitr, rmarkdown, testthat

biocViews SequenceMatching, Visualization

VignetteBuilder knitr

RoxygenNote 7.2.1

Encoding UTF-8

LazyLoad yes

LazyData true

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

git_branch RELEASE_3_19

git_last_commit 9abaa82

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29

1
Contents

dagLogo-package .......................................................... 2
addScheme ............................................................... 3
availableSchemes ......................................................... 4
buildBackgroundModel .................................................... 4
buildZTestBackgroundModel ............................................. 6
cleanPeptides ........................................................... 7
colorsets ................................................................. 8
colorsets2 ................................................................. 9
dagBackground-class ..................................................... 10
dagHeatmap ............................................................... 10
dagLogo ................................................................. 11
dagPeptides-class ......................................................... 13
ecoli.proteome ........................................................... 13
fetchSequence ........................................................... 14
formatSequence .......................................................... 17
getData ................................................................. 18
getGroupingSymbol ....................................................... 18
initiateBackgroundModel ............................................... 19
nameHash ................................................................. 20
prepareProteome ........................................................ 20
prepareProteomeByFTP .................................................. 21
prepareProteomeByUniProtWS ......................................... 22
Proteome-class .......................................................... 23
proteome.example ........................................................ 24
seq.example ............................................................. 25
testDAU ................................................................. 26
testDAUresults-class ..................................................... 27

Index 29

dagLogo-package

Visualize significant conserved peptide sequence pattern in groups
based on the probability theory

Description

dagLogo provides differential analysis of grouped/ungrouped amino acid usage between an input
set of aligned peptide sequences and a background set of aligned peptide sequences which can
be generated in different ways. Results of Fisher’s exact test and/or Z-test are visualized using a
heatmap or DAG Logo.
Details

DAG: Differential Amino acid Group

There are several differences between dagLogo from iceLogo:
1. The sequence patterns can be grouped by charge, chemistry, hydrophobicity and etc.
2. dagLogo accepts different length of unaligned amino acid sequences.
3. Except that random, regional (called anchored) and terminal (called restricted in dagLogo) background model built from the whole proteome, the background set could be generated using subsequences from regions of the protein sequences matching the input set and complementary set of the input set.

Author(s)

Jianhong Ou, Haibo Liu, Julie Lihua Zhu
Maintainer: Jianhong Ou <jianhong.ou@duke.edu>

Examples

```r
data("seq.example")
data("proteome.example")
bg <- buildBackgroundModel(seq.example, proteome=proteome.example, numSubsamples=10L)
t <- testDAU(seq.example, bg)
dagLogo(t)
```

addScheme

Add a custom coloring or grouping scheme.

Description

Add a custom coloring or grouping scheme for ungrouped or grouped amino acids as desired.

Usage

```r
addScheme(
  color = vector("character"),
  symbol = vector("character"),
  group = NULL
)
```

Arguments

- `color`: A named vector of character. This vector specifies different colors for visualizing the different amino acids or amino acid groups.
- `symbol`: A named vector of character. This vector specifies the different symbols for visualizing the different amino acids or amino acid groups.
- `group`: A list or NULL. If only coloring amino acids of similar property is desired, set group to NULL; otherwise group should be a list with same names as those of color and symbol.
Value

Add the custom coloring or grouping scheme to the environment cacheEnv.

Examples

```r
## Add a grouping scheme based on the BLOSUM50 level 3
color = c(LVIMC = "#33FF00", AGSTP = "#CCFF00",
FYW = '#00FF66', EDNQKRH = "#FF0066")
symbol = c(LVIMC = "L", AGSTP = "A", FYW = "F", EDNQKRH = "E")
group = list(
  LVIMC = c("L", "V", "I", "M", "C"),
  AGSTP = c("A", "G", "S", "T", "P"),
  FYW = c("F", "Y", "W"),
  EDNQKRH = c("E", "D", "N", "Q", "K", "R", "H")
):addScheme(color = color, symbol = symbol, group = group)
```

availableSchemes

Get all predefined coloring and grouping schemes

Description

List all predefined coloring and grouping schemes stored in the environment ‘cacheEnv’

Usage

`availableSchemes()`

Value

A vector of names of predefined coloring and grouping schemes stored in the environment ‘cacheEnv’.

Author(s)

Haibo Liu

buildBackgroundModel

Build background models for DAU tests

Description

A method used to build background models for testing differential amino acid usage.
Usage

buildBackgroundModel(
  dagPeptides,
  background = c("wholeProteome", "inputSet", "nonInputSet"),
  model = c("any", "anchored"),
  targetPosition = c("any", "Nterminus", "Cterminus"),
  uniqueSeq = FALSE,
  numSubsamples = 300L,
  rand.seed = 1,
  replacement = FALSE,
  testType = c("ztest", "fisher"),
  proteome
)

Arguments

dagPeptides  An object of `dagPeptides-class` containing peptide sequences as the input set.
background A character vector with options: "wholeProteome", "inputSet", and "nonInputSet", indicating what set of peptide sequences should be considered to generate a background model.
model A character vector with options: "any" and "anchored", indicating whether an anchoring position should be applied to generate a background model.
targetPosition A character vector with options: "any", "Nterminus" and "Cterminus", indicating which part of protein sequences of choice should be used to generate a background model.
uniqueSeq A logical vector indicating whether only unique peptide sequences are included in a background model for sampling.
numSubsamples An integer, the number of random sampling.
rand.seed An integer, the seed used to perform random sampling
replacement A logical vector of length 1, indicating whether replacement is allowed for random sampling.
testType A character vector of length 1. Available options are "ztest" and "fisher".
proteome An object of Proteome, output of `prepareProteome`

Details

The background could be generated from wholeProteome, inputSet or nonInputSet. Case 1: If background = "wholeProteome" and model = "any": The background set is composed of randomly selected subsequences from the wholeProteome with each subsequence of the same length as input sequences.

Case 2: If background = "wholeProteome and model = "anchored": The background set is composed of randomly selected subsequences from the wholeProteome with each subsequence of same length as input sequences. Additionally, the amino acids at the anchoring positions must be the same amino acid as that defined in the dagPeptides object, such as "K" for lysine.
Case 3: If background = "inputSet" and model = "any": similar to Case 1, but the full length protein sequences matching the protein sequence IDs in the inputSet are used for build background model after excluding the subsequences specified in the inputSet from the full length sequences.

Case 4: If background = "inputSet" and model = "anchored": similar to Case 2, but the full-length protein sequences matching the protein sequence IDs in the inputSet are used for build background model after excluding the subsequences specified in the inputSet from the full length sequences.

Case 5: If background = "nonInputSet" and model = "any": The background set is composed of randomly selected subsequences from the wholeProteome, not including the sequences corresponding to the inputSet sequences with each subsequence of same length as input sequences.

Case 6: If background = "nonInputSet" and model = "anchored": similar to Case 5, but the amino acids at the anchoring positions must be the same amino acid as that defined in the dagPeptides object, such as "K" for lysine.

Value

An object of `dagBackground-class`.

Author(s)

Jianhong Ou, Haibo Liu

Examples

```r
dat <- unlist(read.delim(system.file("extdata", "grB.txt", package = "dagLogo"),
    header = FALSE, as.is = TRUE))
##prepare an object of Proteome Class from a fasta file
proteome <- prepareProteome(fasta = system.file("extdata",
    "HUMAN.fasta",
    package = "dagLogo"),
    species = "Homo sapiens")
##prepare an object of dagPeptides Class
seq <- formatSequence(seq = dat, proteome = proteome, upstreamOffset = 14,
    downstreamOffset = 15)
bg_fisher <- buildBackgroundModel(seq, background = "wholeProteome",
    proteome = proteome, testType = "fisher")
bg_ztest <- buildBackgroundModel(seq, background = "wholeProteome",
    proteome = proteome, testType = "ztest")
```

buildZTestBackgroundModel

Build a background model for Z-test.

Description

Build a background model for Z-test.
Usage

```r
buildZTestBackgroundModel(
    dagPeptides,
    matches,
    numSubsamples = 30L,
    rand.seed = 1,
    testType = "ztest",
    replacement = FALSE
)
```

Arguments

- `dagPeptides`: An object of `dagPeptides-class` containing peptide sequences as the input set.
- `matches`: A character vector with the matched subsequences.
- `numSubsamples`: An integer, the number of random sampling.
- `rand.seed`: An integer, the seed used to perform random sampling.
- `replacement`: A logical vector of length 1, indicating whether replacement is allowed for random sampling.

Value

An object of `dagBackground-class`.

Author(s)

Jianhong Ou, Haibo Liu

---

**cleanPeptides**

*clean up peptides*

**Description**

clean up the input peptide subsequences. The function removes peptides which do NOT contain any anchoring amino acid. Adds peptide for each additional anchor in each peptide, and allows multiple anchoring amino acids.

**Usage**

cleanPeptides(dat, anchors)

**Arguments**

- `dat`: input data. The input data contains two columns `symbol`, protein ID, and `peptides`, peptide sequence. The anchoring amino acid must be in lower case.
- `anchors`: A vector of character, anchoring amino acid must be in lower case.
colorsets

Value

A data.frame with columns: `symbol`, `peptides` and `anchor`

Author(s)

Jianhong Ou, Julie Zhu

Examples

```r
dat <- read.csv(system.file("extdata", "peptides2filter.csv", package="dagLogo"))
dat
dat.new <- cleanPeptides(dat, anchors = c("s", "t"))
```

---

colorsets  

Color sets for visualization.

Description

Create color encoding for visualization of a peptide sequence logo.

Usage

```r
colorsets(colorScheme = ls(envir = cachedEnv))
```

Arguments

- `colorScheme`: A character vector of length 1, determining the color scheme based on amino acid classification methods. The available `colorScheme` are "no", "bulkiness_Zimmerman", "hydrophobicity_W"", "hydrophobicity_HW", "isoelectric_point_Zimmerman", "contact_potential_Maiorov", "chemistry_property_Mahler", "consensus_similarity_SF", "volume_Bigelow", "structure_alignments_Mirny", "polarity_Grantham", "sequence_alignment_Dayhoff", "bulkiness_Zimmerman_group", "hydrophobicity_KD_group", "hydrophobicity_HW_group", "charge_group", "contact_potential_Maiorov_group", "chemistry_property_Mahler_group", "consensus_similarity_SF_group", "volume_Bigelow_group", "structure_alignments_Mirny_group", "polarity_Grantham_group", "sequence_alignment_Dayhoff_group", "custom" and "custom_group". If "custom" or "custom_group" are used, users must define a grouping scheme using a list containing sublist named as "color", and "symbol" using the function addScheme, with group set as "NULL" or a list with same names as those of color and symbol. No grouping was applied for the first 12 schemes. It is used to color AAs based on similarities or group amino acids into groups of similarities.

Value

A named character vector of colors
Author(s)

Jianhong Ou, Haibo Liu

See Also

addScheme

Examples

colorsets("polarity_Grantham_group")

Description

retrieve prepared color setting for logo

Usage

colorsets2(
  colorScheme = c("null", "classic", "charge", "chemistry", "hydrophobicity")
)

Arguments

  colorScheme A vector of length 1, the option could be 'null', 'charge', 'chemistry', 'classic' or 'hydrophobicity'

Value

A character vector of color scheme

Author(s)

Jianhong Ou
dagBackground-class  

*Class* dagBackground.

**Description**

An S4 class to represent a background composed of a formatted, aligned peptides for dagLogo analysis.

**Slots**

background  A list of data frame, each of which represents one subset of the background set. Within each n-by-1 dataframe is a the aligned peptides of same length.

numSubsamples  An integer. That is the length of the background list

testType  An character. The type of statistic testing for dagLogo analysis of differential usage of amino acids.

**Author(s)**

Jianhong Ou, Haibo Liu

---

dagHeatmap  

*Visualize daglogo using a heatmap.*

**Description**

Using a heatmap to visualize results of testing differential amino acid usage.

**Usage**

```
dagHeatmap(testDAUresults, type = c("diff", "statistics"), ...)```

**Arguments**

- `testDAUresults`  An object of *testDAUresults-class*, which contains results of testing differential amino acid usage.

- `type`  A character vector of length 1, the type of metrics to display on y-axis. The available options are "diff" and "statistics", which are differences in amino acid usage at each position between the inputSet and the backgroundSet, and the Z-scores or odds ratios when Z-test or Fisher’s exact test is performed to test the differential usage of amino acid at each position between the two sets.

- `...`  other parameters passed to the *pheatmap* function.

**Value**

The output from the *pheatmap* function.
**Author(s)**

Jianhong Ou, Haibo Liu

**Examples**

```r
data("seq.example")
data("proteome.example")
bg <- buildBackgroundModel(seq.example, proteome=proteome.example,
                           numSubsamples=10)
t0 <- testDAU(seq.example, bg)
dagHeatmap(testDAUresults = t0, type = "diff")
```

---

**Description**

Create sequence logo for visualizing results of testing differential usage of amino acids.

**Usage**

```r
dagLogo(
  testDAUresults,
  type = c("diff", "zscore"),
  pvalueCutoff = 0.05,
  groupingSymbol = getGroupingSymbol(testDAUresults@group),
  font = "Helvetica",
  fontface = "bold",
  fontsize = 8,
  title = NULL,
  legend = FALSE,
  labelRelativeToAnchor = FALSE,
  labels = NULL,
  alpha = 1,
  markers = list()
)
```

**Arguments**

- `testDAUresults`: An object of `testDAUresults-class`, which contains results of testing differential amino acid usage.
- `type`: A character vector of length 1. Type of statistics to be displayed on y-axis. Available choices are "diff" or "zscore".
- `pvalueCutoff`: A numeric vector of length 1. A cutoff of p-values.
- `groupingSymbol`: A named character vector.
- `font`: A character vector of length 1. Font type for displaying sequence Logo.
fontface An integer, fontface of text for axis annotation and legends.

fontsize An integer, fontsize of text for axis annotation and legends.

title A character vector of length 1, main title for a plot.

legend A logical vector of length 1, indicating whether to show the legend.

labelRelativeToAnchor

A logical vector of length 1, indicating whether x-axis label should be adjusted relative to the anchoring position.

labels A character vector, x-axis labels.

alpha Alpha channel for transparency of low affinity letters.

markers A list of marker-class.

Value

A sequence Logo is plotted without returned values.

Author(s)

Jianhong Ou, Haibo Liu

Examples

data('seq.example')
data('proteome.example')
bg <- buildBackgroundModel(seq.example, proteome=proteome.example, numSubsamples=10, testType = "ztest")
t0 <- testDAU(seq.example, bg)
t1 <- testDAU(dagPeptides = seq.example, dagBackground = bg, groupingScheme = "hydrophobicity_KD")
t2 <- testDAU(dagPeptides = seq.example, dagBackground = bg, groupingScheme = "charge_group")
t3 <- testDAU(dagPeptides = seq.example, dagBackground = bg, groupingScheme = "chemistry_property_Mahler")
t4 <- testDAU(dagPeptides = seq.example, dagBackground = bg, groupingScheme = "hydrophobicity_KD_group")
dagLogo(t0, markers = list(new("marker", type="rect", start=c(5, 8), gp=gpar(lty=3, fill=NA)),
               new("marker", type="text", start=9, label="*",
                gp=gpar(col=3))))

dagLogo(t1, groupingSymbol = getGroupingSymbol(t1@group))
dagLogo(t2, groupingSymbol = getGroupingSymbol(t2@group))
dagLogo(t3, groupingSymbol = getGroupingSymbol(t3@group))
dagLogo(t4, groupingSymbol = getGroupingSymbol(t4@group))
Class `dagPeptides`. An S4 class to represent formatted, aligned peptides for dagLogo analysis.

Description

Class `dagPeptides`. An S4 class to represent formatted, aligned peptides for dagLogo analysis.

Slots

data  A data frame with column names: IDs, anchorAA, anchorPos, peptide and anchor.
peptides  A matrix of character, each element is a single-character symbol for a amino acid.
upstreamOffset  An integer, the upstream offset relative to the anchoring position.
downstreamOffset  An integer, the downstream offset relative to the anchoring position.
type  A character vector of length 1. Available options :"UniProt", and "fasta" if the `dagPeptides` object is generated using the function `formatSequence`, or "entrezgene" and "uniprotswissprot" if generated by the function `fetchSequence`.

Objects from the Class

Objects can be created by calls of the form

```
new("dagPeptides", data, peptides, upstreamOffset, downstreamOffset, type).
```

Author(s)

Jianhong Ou

dati.ome  An object of `Proteome-class` representing the Escherichia coli proteome.

Description

A dataset containing the *E. coli* proteome.

Usage

dati.ome
fetchSequence

*Fetch protein/peptide sequences and create a *dagPeptides-class* object.*

## Description

This function fetches protein/peptide sequences from a Biomart database or from a *Proteome-class* object based on protein/peptide IDs and create a *dagPeptides-class* object following restriction as specified by parameters: anchorAA or anchorPos, upstreamOffset and downstreamOffset.

## Usage

```r
call = fetchSequence(
  IDs,
  type = "entrezgene",
  anchorAA = NULL,
  anchorPos,
  upstreamOffset,
  downstreamOffset,
  mart,
  taxId = taxId,
  species = "Escherichia coli"
)
```
fetchSequence

proteome,
upstreamOffset,
downstreamOffset
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDs</td>
<td>A character vector containing protein/peptide IDs used to fetch sequences from a Biomart database or a Proteome-class object.</td>
</tr>
<tr>
<td>type</td>
<td>A character vector of length 1. The available options are &quot;entrezgene&quot; and &quot;uniprotswissprot&quot; if parameter mart is missing; otherwise it can be any type of IDs available in Biomart databases.</td>
</tr>
<tr>
<td>anchorAA</td>
<td>A character vector of length 1 or the same length as that of anchorPos, each element of which is a single letter symbol of amino acids, for example, &quot;K&quot; for lysine.</td>
</tr>
<tr>
<td>anchorPos</td>
<td>A character or numeric vector. Each element of which is (1) a single-letter symbol of amino acid followed by the position of the anchoring amino acid in the target peptide/protein sequence, for example, &quot;K123&quot; for lysine at position 123 or the position of the anchoring amino acid in the target peptide/protein sequence, for example, &quot;123&quot; for an amino acid at position 123; or (2) a vector of subsequences containing the anchoring AAs.</td>
</tr>
<tr>
<td>mart</td>
<td>A Biomart database name you want to connect to. Either of parameters mart or proteome should be provided.</td>
</tr>
<tr>
<td>proteome</td>
<td>An object of Proteome-class. Either of parameters mart or Proteome-class should be provided.</td>
</tr>
<tr>
<td>upstreamOffset</td>
<td>An integer, the upstream offset relative to the anchoring position.</td>
</tr>
<tr>
<td>downstreamOffset</td>
<td>An integer, the downstream offset relative to the anchoring position.</td>
</tr>
</tbody>
</table>

Value

An object of class dagPeptides-class

Examples

## Case 1: You have both positions of the anchoring AAs and the identifiers of their enclosing peptide/protein sequences for fetching sequences using the fetchSequence function via the Biomart.

if (interactive())
{
  try(
    mart <- useMart("ensembl")
    fly_mart <-
      useDataset(mart = mart, dataset = "dmelanogaster_gene_ensembl")
    dat <- read.csv(system.file("extdata", "dagLogoTestData.csv",
      package = "dagLogo")
    seq <- fetchSequence(}
function via the Biomart. In the following examples, the anchoring AAs are marked by asterisks.

```r
if (interactive()) {
  try({
    mart <- useMart("ensembl")
    fly_mart <-
      useDataset(mart = mart, dataset = "dmelanogaster_gene_ensembl")
    dat <- read.csv(system.file("extdata", "dagLogoTestData.csv", package = "dagLogo"))
    seq <- fetchSequence(
      IDs = as.character(dat$entrez_geneid),
      anchorAA = "*",
      anchorPos = as.character(dat$NCBI_site),
      mart = fly_mart,
      upstreamOffset = 7,
      downstreamOffset = 7
    )
    head(seq@peptides)
  })
}
```

## Case 2: You don’t have the exactly position information, but you have the interesting peptide subsequences and the identifiers of their enclosing peptide/protein sequences for fetching sequences using the `fetchSequence` function via the Biomart. In the following examples, the anchoring AAs are marked by asterisks.

```r
if (interactive()) {
  try({
    mart <- useMart("ensembl")
    fly_mart <-
      useDataset(mart = mart, dataset = "dmelanogaster_gene_ensembl")
    dat <- read.csv(system.file("extdata", "dagLogoTestData.csv", package = "dagLogo"))
    seq <- fetchSequence(
      IDs = as.character(dat$entrez_geneid),
      anchorAA = "*",
      anchorPos = as.character(dat$NCBI_site),
      mart = fly_mart,
      upstreamOffset = 7,
      downstreamOffset = 7
    )
    head(seq@peptides)
  })
}
```

## In following example, the anchoring AAs are lower-case "s" for amino acid serine.

```r
if (interactive()) {
  try({
    dat <- read.csv(system.file("extdata", "peptides4dagLogo.csv", package = "dagLogo"))
    mart <- useMart("ensembl")
    human_mart <-
      useDataset(mart = mart, dataset = "hsapiens_gene_ensembl")
    seq <- fetchSequence(IDs = toupper(as.character(dat$symbol)),
      anchorAA = "s",
      anchorPos = as.character(dat$peptide),
      mart = human_mart,
      upstreamOffset = 7,
      downstreamOffset = 7
    )
    head(seq@peptides)
  })
}
```
formatSequence

Format already aligned peptide sequences.

Description

Convert already aligned peptide sequences into an object of `dagPeptides-class`.

Usage

formatSequence(seq, proteome, upstreamOffset, downstreamOffset)

Arguments

- `seq`: A vector of aligned peptide sequences of the same length
- `proteome`: An object of `Proteome-class`.
- `upstreamOffset`: An integer, the upstream offset relative to the anchoring position.
- `downstreamOffset`: An integer, the downstream offset relative to the anchoring position.

Value

An object of `dagPeptides-class` Class

Author(s)

Jianhong Ou, Haibo Liu

Examples

```r
## Suppose you already have the aligned peptides sequences at hands. Then you can use
## the formatSequence function to prepare an object of dagPeptides. Befor doing
## that, you need prepare a Proteome object by the prepareProteome function.

dat <- unlist(read.delim(system.file("extdata", "grB.txt", package = "dagLogo"),
                      header = FALSE, as.is = TRUE))

## prepare an object of Proteome Class from a fasta file
proteome <- prepareProteome(fasta = system.file("extdata",
                       "HUMAN.fasta",
                       package = "dagLogo"),
                       species = "Homo sapiens")
```

## prepare an object of dagPeptides Class from a Proteome object

```r
seq <- formatSequence(seq = dat, proteome = proteome, upstreamOffset = 14,
                       downstreamOffset = 15)
```

---

### getData

*Get the data for visualization.*

**Description**

A helper function to get the data and the label for visualization.

**Usage**

```r
data(type, testDAUresults)
```

**Arguments**

- `type` A character vector of length 1, the type of metrics to display on y-axis. The available options are "diff" and "statistics", which are differences in amino acid usage at each position between the inputSet and the backgroundSet, and the Z-scores or odds ratios when Z-test or Fisher’s exact test is performed to test the differential usage of amino acid at each position between the two sets.
- `testDAUresults` An object of `testDAUresults-class`, which contains results of testing differential amino acid usage.

**Value**

A list containing the following components: label A character vector of length 1. The type of data for visualization. dat A matrix of numeric data for visualization.

**Author(s)**

Haibo Liu

---

### getGroupingSymbol

*Get character symbols for grouped amino acids*

**Description**

Get character symbols for grouped amino acids

**Usage**

```r
groupingSymbol(groupingScheme = ls(envir = cachedEnv))
```
Arguments

groupingScheme A character vector of length 1, determining the character symbols used to represent amino acids grouped by their physical and chemical properties. The available groupingScheme are "no", "bulkiness_Zimmerman", "hydrophobicity_KD", "hydrophobicity_HW", "isoelectric_point_Zimmerman", "contact_potential_Maiorov", "chemistry_property_Mahler", "consensus_similarity_SF", "volume_Bigelow", "structure_alignments_Mirny", "polarity_Grantham", "sequence_alignment_Dayhoff", "bulkiness_Zimmerman_group", "hydrophobicity_KD_group", "hydrophobicity_HW_group", "charge_group", "contact_potential_Maiorov_group", "chemistry_property_Mahler_group", "consensus_similarity_SF_group", "volume_Bigelow_group", "structure_alignments_Mirny_group", "polarity_Grantham_group", "sequence_alignment_Dayhoff_group", and "custom". If "custom" is used, users must define a grouping scheme using a list containing sublists named as "color", "symbol" and "group" using the function addScheme. No grouping was applied for the first 12 schemes.

Value

A named character vector of character symbols if grouping is applied; otherwise NULL.

Author(s)

Jianhong Ou, Haibo Liu

See Also

customScheme

Examples

getGroupingSymbol("polarity_Grantham_group")
NameHash

**Value**

An object of `dagBackground-class`.

**Author(s)**

Haibo Liu

---

**Description**

convert group name to a single character to shown in a logo

**Usage**

```r
nameHash(nameScheme = c("classic", "charge", "chemistry", "hydrophobicity"))
```

**Arguments**

- `nameScheme` could be "classic", "charge", "chemistry", "hydrophobicity"

**Value**

A character vector of name scheme

**Author(s)**

Jianhong Ou

---

**prepareProteome**

**Description**

prepare proteome from UniProt webserver or a fasta file

**Usage**

```r
prepareProteome(source, fasta, species = "unknown", ...)
```

**Arguments**

- `source` An object of `UniProt.ws` or a character "UniProt".
- `fasta` fasta file name or an object of `AAStringSet`
- `species` an character to assign the species of the proteome
- `...` parameters could be passed to `prepareProteomeByFTP`.
Value

an object of Proteome which contain protein sequence information.

Author(s)

Jianhong Ou

See Also

formatSequence, buildBackgroundModel

Examples

if(interactive()){
  library(UniProt.ws)
  availableUniprotSpecies("Drosophila melanogaster")
  UniProt.ws <- UniProt.ws(taxId=7227)
  proteome <- prepareProteome(UniProt.ws, species="Drosophila melanogaster")
}

Arguments

source A character vector of length 1 or NULL. A database source from which the proteome sequences are to be downloaded. By default, currently it is "UniProt". If it is NULL, then fastaFile has to be specified. The priority of source is higher than fastaFile.

taxonID NULL,  
species NULL,  
destDir = tempdir(check = TRUE),  
fastaFile,  
...}
**taxonID**
An integer(1), specifying Taxonomy ID for a species of interest. Check the NCBI taxonomy database: [https://www.ncbi.nlm.nih.gov/taxonomy](https://www.ncbi.nlm.nih.gov/taxonomy) or the UniProt database [http://www.uniprot.org/taxonomy/](http://www.uniprot.org/taxonomy/). At least one of the two parameters, taxonID and species, should be specified. If both are specified, taxonID will be used preferentially.

**species**
A character vector of length 1. The Latin name of a species confirming to the Linnaean taxonomy nomenclature system. CAUTION: for species with different strains, attention should be paid. You can interactively choose the right taxonID from an output list.

**destDir**
A character vector of length 1. A destination directory with writing permission for saving downloaded sequences. Default is a temporary directory in the system’s temporary directory.

**fastaFile**
A character vector of length 1. A fasta file name from which protein sequences are read in.

... other parameters passing to the function *download.file*.

**Value**
An object of Proteome

**Author(s)**
Haibo Liu

**Examples**
```r
## Not run:
## Prepare an object of Proteome Class for a proteome from the UniProt database
#' proteome <- prepareProteomeByFTP(source = "UniProt", species = "Homo sapiens")

## End(Not run)
## Prepare an object of Proteome Class from a fasta file
fasta <- system.file("extdata", "HUMAN.fasta", package="dagLogo")
proteome <- prepareProteomeByFTP(source = NULL, species = "Homo sapiens",
fastaFile=fasta)
```

**Description**
Create an object of Proteome Class by query the UniProt database of an organism of a given species’ scientific name, or by using peptide sequences in a fasta file or in an AAStringSet object.

**Usage**
```r
prepareProteomeByUniProtWS(UniProt.ws, fasta, species = "unknown")
```
**Proteome-class**

**Arguments**

- **UniProt.ws**  
  An object of `UniProt.ws`.  

- **fasta**  
  A fasta file name or an object of `AAStringSet`.  

- **species**  
  An character vector of length (1) to designate the species of the proteome.

**Value**

An object of `Proteome` which contain protein sequence information.

**Author(s)**

Jianhong Ou

**See Also**

`formatSequence`, `buildBackgroundModel`

**Examples**

```r
if(interactive()){  
  library(UniProt.ws)  
  availableUniprotSpecies("Drosophila melanogaster")  
  UniProt.ws <- UniProt.ws(taxId=7227)  
  proteome <- prepareProteomeByUniProtWS(UniProt.ws, species="Drosophila melanogaster")
}
```

**Description**

An S4 class to represent a whole proteome for dagLogo analysis.

**Slots**

- **proteome** A data frame.

- **type** A character vector of length 1. Available options :"UniProt", and "fasta".

- **species** A character vector of length 1, such as a conventional Latin name for a species.

**Objects from the Class**

Objects can be created by calls of the form

`new("Proteome", proteome, type, species)`.  

**Author(s)**

Jianhong Ou
proteome.example

An object of **Proteome-class** representing the subset of Drosophila melanogaster proteome.

**Description**

The subset **Proteome-class** of fruit fly.

**Usage**

proteome.example

**Format**

An object of **Proteome-class** for fly subset proteome. The format is: A list with one data frame and an character.

*‘proteome‘: ’data.frame‘: 1406 obs. of 4 variables *‘type‘: ’character‘: ”UniProt“ *‘species‘: ’character‘: ”Drosophila melanogaster”

The format of proteome is

*‘ENTREZ_GENE‘: a character vector, records entrez gene id *‘SEQUENCE‘: a character vector, peptide sequences *‘ID‘: a character vector, Uniprot ID *‘LEN‘: a character vector, length of peptides

**Details**

used as an example dataset

Annotation data obtained by:

```r
library(UniProt.ws)
taxId(UniProt.ws) <- 7227
proteome <- prepareProteome(UniProt.ws)
proteome@proteome <- proteome@proteome[sample(1:19902, 1406), ]
```

**Source**


**Examples**

data(proteome.example)
head(proteome.example@proteome)
proteome.example@type
An object of `dagPeptides-class` representing acetylated lysine-containing peptides.

Description

A dataset containing the acetylated lysine-containing peptides from *Drosophila melanogaster*.

Usage

`seq.example`

Format

An object of `dagPeptides-class` Class The format is: A list.

*`data'`: `data.frame`: 732 obs. of 7 variables *`peptides'`: `matrix`: amino acid in each position
*`upstreamOffset'`: an integer, upstream offset position *`downstreamOffset'`: an integer, downstream offset position *`type'`: "character", type of identifiers

The format of data is

*`IDs'`: a character vector, input identifiers *`anchorAA'`: a character vector, anchor amino acid provided in inputs *`anchorPos'`: a numeric vector, anchor position in the protein *`peptide'`: a character vector, peptide sequences *`anchor'`: a character vector, anchor amino acid in the protein *`upstream'`: a character vector, upstream peptides *`downstream'`: a character vector, downstream peptides

Details

used as an example dataset

seq obtained by:

```
mart <- useMart("ensembl", "dmelanogaster_gene_ensembl")
dat <- read.csv(system.file("extdata", "dagLogoTestIData.csv", package="dagLogo"))
seq <- fetchSequence(as.character(dat$entrez_geneid),
anchorPos=as.character(dat$NCBI_site),
mart=mart,
upstreamOffset=7,
downstreamOffset=7)
```

Examples

```
data(seq.example)
head(seq.example@peptides)
seq.example@upstreamOffset
seq.example@downstreamOffset
```
Differential usage test of amino acids or amino acid groups.

Description
Test differential usage of amino acids with or without grouping between experimental sets and background sets.

Usage
testDAU(
dagPeptides, 
dagBackground, 
groupingScheme = ls(envir = cachedEnv), 
bgNoise = NA, 
method = "none"
)

Arguments

dagPeptides An object of Class dagPeptides-class.
dagBackground An object of Class dagBackground-class.
groupingScheme A character vector of length 1. Available choices are "no", "bulkiness_Zimmerman", "hydrophobicity_KD", "hydrophobicity_HW", "isoelectric_point_Zimmerman", "contact_potential_Maiorov", "chemistry_property_Mahler", "consensus_similarity_SF", "volume_Bigelow", "structure_alignments_Mirny", "polarity_Grantham", "sequence_alignment_Dayhoff", "bulkiness_Zimmerman_group", "hydrophobicity_KD_group", "hydrophobicity_HW_group", "charge_group", "contact_potential_Maiorov_group", "chemistry_property_Mahler_group", "consensus_similarity_SF_group", "volume_Bigelow_group", "structure_alignments_Mirny_group", "polarity_Grantham_group", "sequence_alignment_Dayhoff_group", "custom" and "custom_group". If "custom" or "custom_group" are used, users must define a grouping scheme using a list containing sublist named as "color", and "symbol" using the function addScheme, with group set as "NULL" or a list with same names as those of color and symbol. No grouping was applied for the first 12 schemes. It is used to color AAs based on similarities or group amino acids into groups of similarities.

bgNoise A numeric vector of length 1 if not NA. It should be in the interval of (0, 1) when not NA.

method A character vector of length 1, specifying the method used for p-value adjustment to correct for multiple testing. It can be "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", or "none". For more details, see p.adjust.methods and p.adjust.

Value
An object of Class testDAUresults-class.
Author(s)
Jianhong Ou, Haibo Liu

Examples

dat <- unlist(read.delim(system.file("extdata", "grB.txt", package = "dagLogo"),
               header = FALSE, as.is = TRUE))

# prepare an object of Proteome Class from a fasta file
proteome <- prepareProteome(fasta = system.file("extdata",
                                        "HUMAN.fasta",
                                        package = "dagLogo"),
                           species = "Homo sapiens")

# prepare an object of dagPeptides Class
seq <- formatSequence(seq = dat, proteome = proteome, upstreamOffset = 14,
                       downstreamOffset = 15)
bg_fisher <- buildBackgroundModel(seq, background = "wholeProteome",
                                  proteome = proteome, testType = "fisher")
bg_ztest <- buildBackgroundModel(seq, background = "wholeProteome",
                                 proteome = proteome, testType = "ztest")

## no grouping and distinct coloring scheme, adjust p-values using the
## "BH" method.
t0 <- testDAU(seq, dagBackground = bg_ztest, method = "BY")

## grouped by polarity index (Grantham, 1974)
t1 <- testDAU(dagPeptides = seq, dagBackground = bg_ztest,
              groupingScheme = "polarity_Grantham_group")

## grouped by charge.
t2 <- testDAU(dagPeptides = seq, dagBackground = bg_ztest,
              groupingScheme = "charge_group")

## grouped on the basis of the chemical property of side chains.
t3 <- testDAU(dagPeptides = seq, dagBackground = bg_ztest,
              groupingScheme = "chemistry_property_Mahler_group")

## grouped on the basis of hydrophobicity (Kyte and Doolittle, 1982)
t4 <- testDAU(dagPeptides = seq, dagBackground = bg_ztest,
              groupingScheme = "hydrophobicity_KD_group")

desc<-class(testDAUresults, "testDAUresults")
desc
Slots

- `group` A character vector of length 1, the type of method for grouping amino acid.
- `testType` A character vector of length 1, the type of statistic testing. The available options are "fisher" and "z-test".
- `difference` A numeric matrix consisting of differences of amino acid proportions between the test set and the background set of aligned, formatted peptides at each position.
- `statistics` A numeric matrix consisting of Z-scores or odds ratios for Z-test and Fisher’s exact test, respectively.
- `pvalue` A numeric matrix consisting of p-values.
- `background` A numeric matrix consisting of amino acid proportions in the background set of aligned, formatted peptides at each position.
- `motif` A numeric matrix consisting of amino acid proportions at each position for visualization by dagLogo.
- `upstreamOffset` A positive integer, the upstream offset relative to the anchoring position.
- `downstreamOffset` A positive integer, the upstream offset relative to the anchoring position.

Author(s)

Jianhong Ou, Haibo Liu
Index

* datasets
  - ecoli.proteome, 13
  - proteome.example, 24
  - seq.example, 25
* figure
  - colorsets2, 9
  - nameHash, 20
* internal
  - buildZTestBackgroundModel, 6
  - colorsets, 8
  - dagLogo-package, 2
  - getData, 18
  - getGroupingSymbol, 18
  - initiateBackgroundModel, 19
* misc
  - availableSchemes, 4
  - cleanPeptides, 7
  - prepareProteome, 20
    - prepareProteomeByUniProtWS, 22
  - _PACKAGE (dagLogo-package), 2
  - AAStringSet, 23
  - addScheme, 3, 9, 19
  - availableSchemes, 4
  - buildBackgroundModel, 4, 21, 23
  - buildZTestBackgroundModel, 6
  - cleanPeptides, 7
  - colorsets, 8
  - colorsets2, 9
  - dagBackground (dagBackground-class), 10
  - dagBackground-class, 10, 19
  - dagHeatmap, 10
  - dagLogo, 11
  - dagLogo-package, 2
  - dagPeptides, 13
  - dagPeptides (dagPeptides-class), 13
  - dagPeptides-class, 13, 14, 25
  - download.file, 22
  - ecoli.proteome, 13
  - fetchSequence, 13, 14
  - formatSequence, 13, 17, 21, 23
  - getData, 18
  - getGroupingSymbol, 18
  - initiateBackgroundModel, 19
  - marker-class, 12
  - nameHash, 20
  - p.adjust, 26
  - p.adjust.methods, 26
  - pheatmap, 10
  - prepareProteome, 5, 20
  - prepareProteomeByFTP, 20, 21
  - prepareProteomeByUniProtWS, 22
  - Proteome, 21–23
  - Proteome (Proteome-class), 23
  - Proteome-class, 13, 23, 24
  - proteome.example, 24
  - seq.example, 25
  - testDAU, 19, 26
  - testDAUresults (testDAUresults-class), 27
  - testDAUresults-class, 27
  - UniProt.ws, 20, 23