Package 'dagLogo'

April 22, 2019

Type Package

Title dagLogo

Version 1.21.3

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, biomaRt, grImport2, grid, motifStack

Imports pheatmap, Biostrings, UniProt.ws, BiocGenerics

Suggests XML, BiocStyle, knitr, rmarkdown, testthat

biocViews SequenceMatching, Visualization

VignetteBuilder knitr

RoxygenNote 6.1.1

Encoding UTF-8

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

git_branch master

git_last_commit 425d66d

git_last_commit_date 2019-04-08

Date/Publication 2019-04-21

R topics documented:

---

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

```
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

```
## 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)
```

`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` Class containing peptide sequences as the input set.
`background`	A character vector with defaults: "wholeProteome" and "inputSet", "nonInputSet", indicating what set of peptide sequences should be considered to generate a background model.
`model`	A character vector with defaults: "any" and "anchored", indicating whether an anchoring position should be applied to generate a background model.
`targetPosition`	A character vector with defaults: "any", "Nterminus" and "Cterminus", indicating whether 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 wholeGenome, inputSet or nonInputSet. whole genome: randomly select subsequences from the whole genome with each subsequence containing amino acids with same width of input sequences. anchored whole genome: randomly select subsequences from the whole genome with each subsequence containing amino acids with same width of input sequences where the middle amino acids must contain anchor amino acid, e.g., K, which is specified by user. input set: same to whole genome, but only use protein sequence from input id and not including the site specified in input sequences anchored input set: same to anchored whole genome, but only use protein sequences from input id, and not including the site specified in input sequences. non-input set: whole genome - input set. anchored non-input set: whole genome - input set and the middle amino acids must contain anchor amino acid.

Value

An object of dagBackground-class 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")
```

---

cleanPeptides *clean up peptides*

Description

clean the input file. The function removes peptides without any anchors, adds peptide for each additional anchor in each peptide, and allows multiple anchors as input.

Usage

```
cleanPeptides(dat, anchors)
```

Arguments

dat	input data. The input dat contains two columns symbol and peptides. the anchor AA must be in lower case.
anchors	a vector of character, anchor Amino Acid, must be lower case.

Value

an data.frame with columns: ‘symbol‘, ‘peptides‘ and ‘anchor‘

Author(s)

Jianhong Ou, Julie Zhu

Examples

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

`colorsets2` *retrieve color setting for logo*

Description

retrieve prepared color setting for logo

Usage

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

Arguments

`colorScheme`	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 of a formatted, aligned peptides for dagLogo analysis.

Slots

`background` A list of data frame. 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. Test type.

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 (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 thepheatmap function.

Value

The output from the pheatmap function.

Author(s)

Jianhong Ou, Haibo Liu

Examples

```
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")
```

dagLogo *Create sequence logos.*

Description

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

Usage

```
dagLogo(testDAUresults, type = c("diff", "zscore"),
  pvalueCutoff = 0.05,
  groupingSymbol = getGroupingSymbol(testDAUresults@group),
  font = "Helvetica-Bold", fontface = "bold", fontsize = 5,
  title = NULL, legend = FALSE, labelRelativeToAnchor = FALSE,
  labels = NULL, alpha = 0.5)
```

Arguments

testDAUresults	An object of testDAUresults-class (results of testing differential amino acid usage).
type	A character vector of length 1. Type of statistics to display 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.

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)
```

```
dagLogo(t1, groupingSymbol = getGroupingSymbol(t1@group))
dagLogo(t2, groupingSymbol = getGroupingSymbol(t2@group))
dagLogo(t3, groupingSymbol = getGroupingSymbol(t3@group))
dagLogo(t4, groupingSymbol = getGroupingSymbol(t4@group))
```

`dagPeptides-class` *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 columns: 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 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

`ecoli.proteome` *An object of* `Proteome-class` *representing the* Escherichia coli *proteome.*

Description

A dataset containing the *E. coli* proteome.

Usage

```
ecoli.proteome
```

Format

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

*‘proteome‘: ’data.frame’: 13780 obs. of 4 variables *‘type‘: ’character’: "UniProt" *‘species‘: ’character’: "Escherichia coli"

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 in the examples

Annotation data obtained by:

library(UniProt.ws)

taxId(UniProt.ws) <- 562

proteome <- prepareProteome(UniProt.ws, species="Escherichia coli")

Source

http://www.uniprot.org/

Examples

```
data(ecoli.proteome)
head(ecoli.proteome@proteome)
ecoli.proteome@type
```

`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

```
fetchSequence(IDs, type = "entrezgene", anchorAA = NULL, anchorPos,
  mart, proteome, upstreamOffset, downstreamOffset)
```

Arguments

IDs	A character vector containing protein/peptide IDs used to fetch sequences from a Biomart database or a Proteome-class object.
type	A character vector of length 1. The available options are "entrezgene" and "uniprotswissprot" if parameter mart is missing; otherwise it can be any type of IDs available in Biomart databases.

anchorAA — 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, "K" for lysine.

anchorPos — 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, "K123" for lysine at position 123 or the position of the anchoring amino acid in the target peptide/protein sequence, for example, "123" for an amino acid at position 123; or (2) a vector of subsequences containing the anchoring AAs.

mart — A Biomart database name you want to connect to. One of parameters mart and proteome should be provided.

proteome — An object of Proteome-class class. One of parameters mart and Proteome-class should be provided.

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 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(
       IDs = as.character(dat$entrez_geneid),
       anchorPos = as.character(dat$NCBI_site),
       mart = fly_mart,
       upstreamOffset = 7,
       downstreamOffset = 7)
   head(seq@peptides)
   })
}


## Case 2: You don't have the exactly postion infor, 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.
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$entrez_geneid),
            anchorAA = "*",
            anchorPos = as.character(dat$peptide),
            mart = fly_mart,
            upstreamOffset = 7,
            downstreamOffset = 7
        )
        head(seq@peptides)
    })
}

## In following example, the anchoring AAs are lower case "s" for amino acid
## serine.
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$symbol)),
                             type = "hgnc_symbol",
                             anchorAA = "s",
                             anchorPos = as.character(dat$peptides),
                             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` 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 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

```
## You already have the aligned peptides sequences in hand. 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
seq <- formatSequence(seq = dat, proteome = proteome, upstreamOffset = 14,
                      downstreamOffset = 15)
```

---

nameHash *convert group name to a single character*

Description

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

Usage

```
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` *prepare proteome for background building*

Description

prepare proteome from UniProt webserver or a fasta file

Usage

```
prepareProteome(UniProt.ws, fasta, species = "unknown")
```

Arguments

`UniProt.ws`	an object of UniProt.ws
`fasta`	fasta file name or an object of AAStringSet
`species`	an character to assign the species of the proteome

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")
 }
```

`Proteome-class` *Class* `Proteome`.

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, 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 in the examples

Annotation data obtained by:

library(UniProt.ws)

taxId(UniProt.ws) <- 7227

proteome <- prepareProteome(UniProt.ws)

proteome@proteome <- proteome@proteome[sample(1:19902, 1406),]

Source

http://www.uniprot.org/

Examples

```
data(proteome.example)
head(proteome.example@proteome)
proteome.example@type
```

`seq.example` *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’: amnio 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 in the examples

seq obtained by:

mart <- useMart("ensembl", "dmelanogaster_gene_ensembl")

dat <- read.csv(system.file("extdata", "dagLogoTestData.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
```

`testDAU` *Differential usage test of amino acids or amino acid groups.*

Description

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

Usage

```
testDAU(dagPeptides, dagBackground, groupingScheme = ls(envir =
  cachedEnv), bgNoise = NA)
```

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_ "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_ "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.

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
t0 <- testDAU(seq, dagBackground = bg_ztest)

## grouped by polarity index (Granthm, 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")
```

---

testDAUresults-class *Class* testDAUresults.

Description

An S4 class to represent a DAU statistical test result from dagLogo analysis.

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 amino acid proportions in the background set of aligned, formatted peptides at each position.

`motif` A numeric matrix consisting of proportions for 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