Package ‘bgafun’

November 20, 2016

Type Package
Title BGAfun A method to identify specificity determining residues in protein families
Version 1.36.0
Date 2007-08-03
Author Iain Wallace
Maintainer Iain Wallace <iain.wallace@ucd.ie>
Depends made4, seqinr, ade4
Description A method to identify specificity determining residues in protein families using Between Group Analysis
License Artistic-2.0
biocViews Classification
NeedsCompilation no

R topics documented:

convertAAP-package .................................................. 2
convert_amino-package ................................................. 2
add_pseudo_counts ................................................... 3
amino_counts .......................................................... 3
average_cols_aap ..................................................... 4
BGAfun ................................................................. 4
calculate_pseudo ..................................................... 5
Calculate_Row_Weights ................................................. 5
convert_aln_AAP ......................................................... 6
convert_aln_amino ..................................................... 7
convert_seq_amino ..................................................... 7
create_colnames_amino ................................................. 7
create_probab ........................................................ 7
create_profile ........................................................ 8
create_profile_strings ............................................... 8
Henikoff_weights ..................................................... 9
LDH ................................................................. 9
LDH.aap .............................................................. 9
LDH.aap.ave .......................................................... 9
LDH.amino ............................................................ 9
**convert_AAP-package**

Converts an alignment into a matrix using the AAP encoding

**Description**

Convert an alignment read in by seqinr into a matrix using the AAP encoding. This is suitable for BGA analysis using PCA

**Details**

- **Package**: convert_AAP
- **Type**: Package
- **Version**: 1.0
- **Date**: 2007-03-14
- **License**: Artistic License

**Author(s)**

Iain Wallace

**References**

BMC hopefully

---

**convert_amino-package**

The functions required to convert an alignment into a binary matrix suitable for BGA analysis

**Description**

The functions required to convert an alignment into a binary matrix suitable for BGA analysis
### add_pseudo_counts

**Details**
Read in the alignment, then convert into matrix

**Author(s)**
Iain Wallace

**References**
BMC hopefully

---

**add_pseudo_counts**  
*Add pseudo counts to amino acid matrix based on defined groups*

**Description**
This function will add pseudo counts to binary amino acid matrix based on the defined groups. It is used to minimise the effect of small sample size. The method of Henikoff and Henikoff is used to calculate the pseudocounts. An alternative method is to simply add 1 to the binary matrix.

**Usage**

```r
add_pseudo_counts(amino, groups)
```

**Arguments**
- `amino`  
  Matrix representation of alignment generated by `convert_aln_amino`
- `groups`  
  Vector or factor that shows the group representation for each sequence in the alignment

**Examples**

```r
library(bgafun)
data(LDH.amino.gapless)
data(LDH.groups)
LDH.pseudo=LDH.amino.gapless+1
# or use the function
LDH.pseudo=add_pseudo_counts(LDH.amino.gapless,LDH.groups)
```

---

**amino_counts**  
*Calculate count of amino acid types at each position*

**Description**
Internal Function Calculate the counts of amino acid types at each position in an alignment from a binary amino acid matrix.
average_cols_aap \hspace{1cm} \textit{Replaces gaps with the average of the column}

**Description**

This function will deal with gaps in the Amino Acid Property encoding scheme. It replaces gaps with the average in the column for each group, provided the column is highly occupied for that group. It will only average out over columns that have high percentage of gaps. It will remove all other columns containing gaps.

**Usage**

\begin{verbatim}
average_cols_aap(x,y)
\end{verbatim}

**Arguments**

\begin{itemize}
\item \textbf{x} Matrix representation of alignment generated by convert\_aln\_AAP
\item \textbf{y} Vector or factor that shows the group representation for each sequence in the alignment
\end{itemize}

**Examples**

\begin{verbatim}
library(bgafun)
data(LDH)
data(LDH.groups)
LDH.aap=convert_aln_AAP(LDH)
LDH.aap.ave=average_cols_aap(LDH.aap,LDH.groups)
dim(LDH.aap.ave)
\end{verbatim}

---

**BGAfun**

\textit{BGAfun A method to identify specificity determining residues in protein families}

**Description**

This Package combines between group analysis with sequence alignments to identify specificity determining residues in protein families.

**Author(s)**

Iain Wallace <iain.wallace@ucd.ie>

**References**

calculate_pseudo

Calculates pseudo count for each column in the amino acid matrix

Description

Internal function Calculates the pseudo count for each column in the amino acid matrix

Calculate_Row_Weights

Calculate the sequence weights for all the rows in my amino, using label as the grouping

Description

This will calculate the sequence weights for each group using the Heinkoff and Heinkoff method. Each residue in the sequence is assigned a weight depending on how unique it is in the column. The sequence weight is then the sum of these weights, and the total weight is the number of groups

Usage

Calculate_Row_Weights(my_amino, label)

Arguments

my_amino Matrix representation of alignment generated by convert_aln_amino
label Vector or factor that shows the group representation for each sequence in the alignment
References


Examples

```r
library("bgafun")
data(LDH.amino.gapless)
data(LDH.groups)
LDH.weights=Calculate_Row_Weights(LDH.amino.gapless,LDH.groups)
sum(LDH.weights)
```

convert_aln_AAP

Converts alignment into a matrix using the amino acid property encoding

Description

Each residue in the alignment is represented by a vector of five continuous variables as given by Atchley et al. They applied a multivariate statistic approach to reduce the information in 494 amino acid attributes into a set of five factors for each amino acid. Factor A is termed the polarity index. It correlates well with a large variety of descriptors including the number of hydrogen bond donors, polarity versus nonpolarity, and hydrophobicity versus hydrophilicity. Factor B is a secondary structure index. It represents the propensity of an amino acid to be in a particular type of secondary structure, such as a coil, turn or bend versus the frequency of it in an a-helix. Factor C is correlated with molecular size, volume and molecular weight. Factor D reflects the number of codons coding for an amino acid and amino acid composition. These attributes are related to various physical properties including refractivity and heat capacity. Factor E is related to the electrostatic charge. Gaps are represented by five zeros and should be either removed or replaced by the average of the column for a particular group.

Usage

```r
convert_aln_AAP(Alignment)
```

Arguments

Alignment

Alignment object read in using read.alignment function in seqinr

References


Examples

```r
library(bgafun)
data(LDH)
data(LDH.groups)
LDH.aap=convert_aln_AAP(LDH)
dim(LDH.aap)
LDH.aap.ave=average_cols_aap(LDH.aap,LDH.groups)
dim(LDH.aap.ave)
```
**convert_aln_amino**

Converts an alignment object into binary amino matrix

**Description**

Converts an alignment object, read in by the seqinr package, into a binary matrix. The binary matrix represents the absence or presence of amino acids at each position in the alignment.

**Usage**

```r
convert_aln_amino(Alignment)
```

**Arguments**

- **Alignment**
  - Alignment object read in using read.alignment function in seqinr.

**Examples**

```r
library(bgafun)
LDH <- read.alignment(file = system.file("sequences/LDH-MDH-PF00056.fasta", package = "bgafun"), format = "fasta")
LDH.amino <- convert_aln_amino(LDH)
dim(LDH.amino)
```

**convert_seq_amino**

Converts a single sequence into a binary string

**Description**

Internal Function Converts a single sequence from an alignment object into a binary string.

**create_colnames_amino**

Creates the column names for the binary matrix

**Description**

Internal Function Creates the column names for the matrix in the form "Position""Amino Acid Letter".

**create_probab**

Generates probability matrix for pseudocounts calculation

**Description**

Internal function. Generates an amino acid probability matrix which is based on BLOSUM 62, and is used to calculate how many pseudo counts should be added.
create_profile

Creates a sequence profile for an binary amino acid matrix

Description

Internal Function Returns a profile matrix, which show how many of each type of amino acids are in each position in an aligmnent It takes in a binary amino acid matrix

create_profile_strings

Create a profile string for each group in an alignment

Description

This function is used to analysis the amino acids at each position in the alignment. It can be used to analysis the columns that the bga analysis identified as interesting It creates a profile string, 1D vector which shows the number of amino acids at each position in an alignment for each group that has been defined

Usage

create_profile_strings(x,y)

Arguments

x

Matrix representation of alignment generated by convert\_aln\_amino

y

Vector or factor that shows the group representation for each sequence in the alignment

Examples

library(bgafun)
data(LDH.groups)
data(LDH.amino.gapless)
#run the analysis
LDH.binary.bga=bga(t(LDH.amino.gapless+1),LDH.groups)
#Get the important residues
top_res=top_residues_2_groups(LDH.binary.bga)
#To tidy up the results
names(top_res)=sub("X","",names(top_res))
# and now look at the amino acid content in the alignment
LDH.profiles=create_profile_strings(LDH.amino.gapless,LDH.groups)
# and now look at only those columns that are identified by BGA
#LDH.profiles[,,(colnames(LDH.profiles) %in% names(top_res))]
Henikoff_weights

Calculates Henikoff weights for each sequence in a binary amino acid matrix.

Description
Internal Function Calculates a sequence weight for each sequence in an alignment using the Henikoff method.

References

LDH
LDH alignment read in from a file

Description
Seqinr representation of the LDH example alignment.

LDH.aap
AAP matrix

Description
Amino Acid Properties representation of LDH alignment

LDH.aap.ave
AAP matrix

Description
Amino Acid Properties Matrix after averaging out gaps

LDH.amino
Binary amino acid matrix after converting the Lactate alignment

Description
Binary amino acid matrix after converting the Lactate alignment
LDH.amino.gapless  
*Amino acid matrix after removing gaps*

**Description**

The amino acid matrix for the lactate example, after removing gappy positions

LDH.amino.pseudo  
*Amino acid matrix after adding pseudo counts*

**Description**

Amino acid matrix after adding pseudo counts to the LDH.amino.gapless matrix

**Usage**

data(LDH.amino.pseudo)

LDH.groups  
*Groups in the LDH alignment*

**Description**

Factor assigning the sequences in the LDH alignment into one of two groups

pseudo_counts  
*Calculate pseudo counts for a profile*

**Description**

Internal function that is used to calculate pseudo counts for an amino acid profile. The Henikoff method is used.

remove_gaps  
*Removes gaps from a amino binary matrix*

**Description**

Internal Function This removes gappy positions from an alignment represented in a binary matrix.
**remove_gaps_groups**

**Description**

This function is used to deal with gaps in the binary amino acid encoding. It will remove positions from a binary amino matrix that contain more a certain fraction of gaps for any group in a column, in the alignment. The gap fraction should be between 0 and 1, and can be changed with the `gap_fraction` variable.

**Usage**

```r
remove_gaps_groups(x, z, gap_fraction = 0.6)
```

**Arguments**

- `x`: Matrix representation of alignment generated by `convert_aln_amino`
- `z`: Vector or factor that shows the group representation for each sequence in the alignment
- `gap_fraction`: Float between 0 and 1 indicating the fraction of gaps in a column before it should be removed

**Examples**

```r
library(bgafun)
data(LDH)
data(LDH.groups)
LDH.amino = convert_aln_amino(LDH)
dim(LDH.amino)
LDH.amino.gapless = remove_gaps_groups(LDH.amino, LDH.groups, gap_fraction = 0.6)
dim(LDH.amino.gapless)
```

---

**run_between_pca**

**Description**

This is a cover function that runs supervised PCA on a matrix that represents an alignment. The matrix can either be a binary matrix (with or without pseudocounts) or one that represents the properties at each position of the alignment.

**Usage**

```r
run_between_pca(x, z, y)
```
sum_aln

Arguments

x Matrix representation of alignment generated by convert\_aln\_amino

z Matrix representation of alignment generated by convert\_aln\_amino or convert\_aln\_AAP

y Vector or factor that shows the group representation for each sequence in the alignment

Examples

library(bgafun)
data(LDH)
data(LDH.groups)
#Used to calculate the sequence weights
data(LDH.amino.gapless)
data(LDH.aap.ave)
#Run the analysis
LDH.aap.ave.bga=run_between_pca(LDH.amino.gapless,LDH.aap.ave,LDH.groups)
class(LDH.aap.ave.bga)
#to visualise the results
plot(LDH.aap.ave.bga)

---

sum_20_aln

*Calculates number of amino acids in each group of 20 columns (1 column in an alignment)*

Description

Internal Function Calculates number of amino acids in each group of 20 columns which corresponds to 1 column in an alignment It takes in an binary amino acid matrix.

---

sum_20_cols

*Calculate number of amino acids in a column of an alignment*

Description

Internal Function Sum up 20 columns in an amino acid matrix which corresponds to one column in an alignment

---

sum_aln

*Calculate number of amino acids in each position in an alignment*

Description

Internal Function Calculates the total number of amino acids in each position. It is used to identify positions with a high percentage of gaps It works on an amino acid matrix
top_residues_2_groups

Return a list of the top residues at either end of the axis

Description
This will identify the residues that are most discriminating between the two groups, and as such are most likely to be specificity determining residues. It will return a list of the residues at the end of the axis in a bga analysis. It is used when there are two groups. The function create_profile_strings can be used to look at the amino acid content in the column that the analysis identifies.

Usage

top_residues_2_groups(bga_results, residue_number=20)

Arguments

- **bga_results**: Results of BGA analysis, either from BGA or run_between_pca function
- **residue_number**: Number of positions at each end of the axis to return

Examples

```r
library(bgafun)
data(LDH.groups)
data(LDH.amino.gapless)
LDH.binary.bga=bga(t(LDH.amino.gapless+1),LDH.groups)
top_res=top_residues_2_groups(LDH.binary.bga)
#To tidy up the results
names(top_res)=sub("X","",names(top_res))
# to look at the amino acid content in the alignment
LDH.profiles=create_profile_strings(LDH.amino.gapless,LDH.groups)
LDH.profiles[, colnames(LDH.profiles) %in% names(top_res)]
```

Weight_Amino

Calculates sequence weight for each sequence in an amino acid matrix

Description
Internal Function Calculates sequence weight for each sequence, and multiples the matrix by this weight. It returns a weighted amino acid matrix.
Index

*Topic IO
  average_cols_aap, 4
  convert_aln_AAP, 6
  convert_aln_amino, 7
  convert_seq_amino, 7
  LDH, 9
  remove_gaps, 10

*Topic datasets
  LDH.aap, 9
  LDH.aap.ave, 9
  LDH.amino, 9
  LDH.amino.gapless, 10
  LDH.amino.pseudo, 10
  LDH.groups, 10

*Topic manip
  add_pseudo_counts, 3
  amino_counts, 3
  calculate_pseudo, 5
  Calculate_Row_Weights, 5
  create_colnames_amino, 7
  create_probab, 7
  create_profile, 8
  create_profile_strings, 8
  Henikoff_weights, 9
  pseudo_counts, 10
  remove_gaps_groups, 11
  run_between_pca, 11
  sum_20_aln, 12
  sum_20_cols, 12
  sum_aln, 12
  top_residues_2_groups, 13
  Weight_Amino, 13

*Topic package
  BGAfun, 4
  convert_amino-package, 2
  convertAAP-package, 2
  calculate_pseudo, 5
  Calculate_Row_Weights, 5
  convert_aln_AAP, 6
  convert_aln_amino, 7
  convert_amino-package, 2
  convert_seq_amino, 7
  convertAAP (convertAAP-package), 2
  convertAAP-package, 2
  create_colnames_amino, 7
  create_probab, 7
  create_profile, 8
  create_profile_strings, 8
  Henikoff_weights, 9
  LDH, 9
  LDH.aap, 9
  LDH.aap.ave, 9
  LDH.amino, 9
  LDH.amino.gapless, 10
  LDH.amino.pseudo, 10
  LDH.groups, 10
  pseudo_counts, 10
  remove_gaps, 10
  remove_gaps_groups, 11
  run_between_pca, 11
  sum_20_aln, 12
  sum_20_cols, 12
  sum_aln, 12
  top_residues_2_groups, 13
  Weight_Amino, 13

BGAfun, 4
bgafun (BGAfun), 4