Package 'protr'

January 27, 2015

Version 0.5-1

Date 2014-12-22

Title Generating Various Numerical Representation Schemes of Protein Sequence

Description Comprehensive toolkit for generating various numerical representation schemes of protein sequence. The descriptors and similarity scores included are extensively utilized in bioinformatics and chemogenomics research. For full functionality, ncbi-blast+ is required.

Author Nan Xiao <road2stat@gmail.com>, Qingsong Xu <dasongxu@gmail.com>, Dongsheng Cao <oriental-cds@163.com>

Maintainer Nan Xiao <road2stat@gmail.com>

License BSD 3-clause License + file LICENSE

URL https://github.com/road2stat/protr

BugReports https://github.com/road2stat/protr/issues

SystemRequirements ncbi-blast+

LazyData yes

Suggests Biostrings, GOSemSim, foreach, doParallel, org.Hs.eg.db

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-24 01:23:01

R topics documented:

`protr-package` *Generating Various Numerical Representation Schemes of Protein Sequence*

Description

The protr package is a comprehensive toolkit for generating various numerical representation schemes of protein sequence. The descriptors are extensively utilized in bioinformatics and chemogenomics research. The commonly used descriptors include amino acid composition, autocorrelation, CTD, conjoint traid, quasi-sequence order, pseudo amino acid composition, and profile-based descriptors derived by Position-Specific Scoring Matrix (PSSM). The descriptors for proteochemometric (PCM) modeling include the scales-based descriptors derived by principal components analysis, factor analysis, multidimensional scaling, amino acid properties (AAindex), 20+ classes of 2D and 3D molecular descriptors (Topological, WHIM, VHSE, etc.), and BLOSUM/PAM matrix-derived descriptors. The protr package also integrates the function of parallelized similarity computation derived by pairwise protein sequence alignment and Gene Ontology (GO) semantic similarity measures.

Details

Package:	protr
Type:	Package
Version:	0.5-1
License:	BSD 3-clause License

Note

The comprehensive user guide could be opened with `vignette('protr')`, which explains each descriptor included in this package and corresponding usage.

The web server for this package, `ProtrWeb` is located at: http://cbdd.csu.edu.cn:8080/protrweb/.

Bug reports and feature requests should be sent to https://github.com/road2stat/protr/issues.

Author(s)

Nan Xiao <<road2stat@gmail.com>> Qing-Song Xu <<dasongxu@gmail.com>> Dong-Sheng Cao <<oriental-cds@163.com>>

References

(to appear)

Examples

```
NULL
```

`AA2DACOR` *2D Autocorrelations Descriptors for 20 Amino Acids calculated by Dragon*

Description

2D Autocorrelations Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AA2DACOR)
```

Details

This dataset includes the 2D autocorrelations descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AA2DACOR)
```

AA3DMoRSE *3D-MoRSE Descriptors for 20 Amino Acids calculated by Dragon*

Description

3D-MoRSE Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AA3DMoRSE)
```

Details

This dataset includes the 3D-MoRSE descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AA3DMoRSE)
```

AAACF *Atom-Centred Fragments Descriptors for 20 Amino Acids calculated by Dragon*

Description

Atom-Centred Fragments Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAACF)
```

Details

This dataset includes the atom-centred fragments descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAACF)
```

AABLOSUM100 *BLOSUM100 Matrix for 20 Amino Acids*

Description

BLOSUM100 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM100)
```

Details

BLOSUM100 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AABLOSUM100)
```

AABLOSUM45 *BLOSUM45 Matrix for 20 Amino Acids*

Description

BLOSUM45 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM45)
```

Details

BLOSUM45 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AABLOSUM45)
```

AABLOSUM50 *BLOSUM50 Matrix for 20 Amino Acids*

Description

BLOSUM50 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM50)
```

Details

BLOSUM50 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AABLOSUM50)
```

AABLOSUM62 *BLOSUM62 Matrix for 20 Amino Acids*

Description

BLOSUM62 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM62)
```

Details

BLOSUM62 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AABLOSUM62)
```

AABLOSUM80 *BLOSUM80 Matrix for 20 Amino Acids*

Description

BLOSUM80 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM80)
```

Details

BLOSUM80 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AABLOSUM80)
```

AABurden *Burden Eigenvalues Descriptors for 20 Amino Acids calculated by Dragon*

Description

Burden Eigenvalues Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AABurden)
```

Details

This dataset includes the Burden eigenvalues descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AABurden)
```

AAConn — *Connectivity Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Connectivity Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAConn)
```

Details

This dataset includes the connectivity indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAConn)
```

AAConst — *Constitutional Descriptors for 20 Amino Acids calculated by Dragon*

Description

Constitutional Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAConst)
```

Details

This dataset includes the constitutional descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAConst)
```

AACPSA *CPSA Descriptors for 20 Amino Acids calculated by Discovery Studio*

Description

CPSA Descriptors for 20 Amino Acids calculated by Discovery Studio

Usage

```
data(AACPSA)
```

Details

This dataset includes the CPSA descriptors of the 20 amino acids calculated by Discovery Studio (version 2.5) used for scales extraction in this package. All amino acid molecules had also been optimized with MOE 2011.10 (semiempirical AM1) before calculating these CPSA descriptors. The SDF file containing the information of the optimized amino acid molecules is included in this package. See OptAA3d for more information.

Examples

```
data(AACPSA)
```

AADescAll *All 2D Descriptors for 20 Amino Acids calculated by Dragon*

Description

All 2D Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AADescAll)
```

Details

This dataset includes all the 2D descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AADescAll)
```

AAEdgeAdj *Edge Adjacency Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Edge Adjacency Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAEdgeAdj)
```

Details

This dataset includes the edge adjacency indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAEdgeAdj)
```

AAEigIdx *Eigenvalue-Based Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Eigenvalue-Based Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAEigIdx)
```

Details

This dataset includes the eigenvalue-based indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAEigIdx)
```

AAFGC *Functional Group Counts Descriptors for 20 Amino Acids calculated by Dragon*

Description

Functional Group Counts Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAFGC)
```

Details

This dataset includes the functional group counts descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAFGC)
```

AAGeom *Geometrical Descriptors for 20 Amino Acids calculated by Dragon*

Description

Geometrical Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAGeom)
```

Details

This dataset includes the geometrical descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAGeom)
```

AAGETAWAY *GETAWAY Descriptors for 20 Amino Acids calculated by Dragon*

Description

GETAWAY Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAGETAWAY)
```

Details

This dataset includes the GETAWAY descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAGETAWAY)
```

AAindex *AAindex Data of 544 Physicochemical and Biological Properties for 20 Amino Acids*

Description

AAindex Data of 544 Physicochemical and Biological Properties for 20 Amino Acids

Usage

```
data(AAindex)
```

Details

The data was extracted from the AAindex1 database ver 9.1 (ftp://ftp.genome.jp/pub/db/community/aaindex/aaindex1) as of Nov. 2012 (Data Last Modified 2006-08-14).

With this data, users could investigate each property's accession number and other details. Visit http://www.genome.jp/dbget/aaindex.html for more information.

Examples

```
data(AAindex)
```

AAInfo — *Information Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Information Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAInfo)
```

Details

This dataset includes the information indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAInfo)
```

AAMetaInfo — *Meta Information for the 20 Amino Acids*

Description

Meta Information for the 20 Amino Acids

Usage

```
data(AAMetaInfo)
```

Details

This dataset includes the meta information of the 20 amino acids used for the 2D and 3D descriptor calculation in this package. Each column represents:

- `AAName` Amino Acid Name
- `Short` One-Letter Representation
- `Abbreviation` Three-Letter Representation
- `mol` SMILE Representation
- `PUBCHEM_COMPOUND_CID` PubChem CID for the Amino Acid
- `PUBCHEM_LINK` PubChem Link for the Amino Acid

Examples

```
data(AAMetaInfo)
```

AAMOE2D *2D Descriptors for 20 Amino Acids calculated by MOE 2011.10*

Description

2D Descriptors for 20 Amino Acids calculated by MOE 2011.10

Usage

```
data(AAMOE2D)
```

Details

This dataset includes the 2D descriptors of the 20 amino acids calculated by MOE 2011.10 used for scales extraction in this package.

Examples

```
data(AAMOE2D)
```

AAMOE3D *3D Descriptors for 20 Amino Acids calculated by MOE 2011.10*

Description

3D Descriptors for 20 Amino Acids calculated by MOE 2011.10

Usage

```
data(AAMOE3D)
```

Details

This dataset includes the 3D descriptors of the 20 amino acids calculated by MOE 2011.10 used for scales extraction in this package. All amino acid molecules had also been optimized with MOE (semiempirical AM1) before calculating these 3D descriptors. The SDF file containing the information of the optimized amino acid molecules is included in this package. See OptAA3d for more information.

Examples

```
data(AAMOE3D)
```

AAMolProp *Molecular Properties Descriptors for 20 Amino Acids calculated by Dragon*

Description

Molecular Properties Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAMolProp)
```

Details

This dataset includes the molecular properties descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAMolProp)
```

AAPAM120 *PAM120 Matrix for 20 Amino Acids*

Description

PAM120 Matrix for 20 Amino Acids

Usage

```
data(AAPAM120)
```

Details

PAM120 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM120)
```

AAPAM250 *PAM250 Matrix for 20 Amino Acids*

Description

PAM250 Matrix for 20 Amino Acids

Usage

```
data(AAPAM250)
```

Details

PAM250 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM250)
```

AAPAM30 *PAM30 Matrix for 20 Amino Acids*

Description

PAM30 Matrix for 20 Amino Acids

Usage

```
data(AAPAM30)
```

Details

PAM30 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM30)
```

AAPAM40 *PAM40 Matrix for 20 Amino Acids*

Description

PAM40 Matrix for 20 Amino Acids

Usage

```
data(AAPAM40)
```

Details

PAM40 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM40)
```

AAPAM70 *PAM70 Matrix for 20 Amino Acids*

Description

PAM70 Matrix for 20 Amino Acids

Usage

```
data(AAPAM70)
```

Details

PAM70 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM70)
```

AARandic — *Randic Molecular Profiles Descriptors for 20 Amino Acids calculated by Dragon*

Description

Randic Molecular Profiles Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AARandic)
```

Details

This dataset includes the Randic molecular profiles descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AARandic)
```

AARDF — *RDF Descriptors for 20 Amino Acids calculated by Dragon*

Description

RDF Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AARDF)
```

Details

This dataset includes the RDF descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AARDF)
```

AATopo — *Topological Descriptors for 20 Amino Acids calculated by Dragon*

Description

Topological Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AATopo)
```

Details

This dataset includes the topological descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AATopo)
```

AATopoChg — *Topological Charge Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Topological Charge Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AATopoChg)
```

Details

This dataset includes the topological charge indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AATopoChg)
```

AAWalk *Walk and Path Counts Descriptors for 20 Amino Acids calculated by Dragon*

Description

Walk and Path Counts Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAWalk)
```

Details

This dataset includes the walk and path counts descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAWalk)
```

AAWHIM *WHIM Descriptors for 20 Amino Acids calculated by Dragon*

Description

WHIM Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAWHIM)
```

Details

This dataset includes the WHIM descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAWHIM)
```

acc *Auto Cross Covariance (ACC) for Generating Scales-Based Descriptors of the Same Length*

Description

Auto Cross Covariance (ACC) for Generating Scales-Based Descriptors of the Same Length

Usage

```
acc(mat, lag)
```

Arguments

`mat`	A `p * n` matrix. Each row represents one scale (total `p` scales), each column represents one amino acid position (total `n` amino acids).
`lag`	The lag parameter. Must be less than the amino acids.

Details

This function calculates the auto covariance and auto cross covariance for generating scale-based descriptors of the same length.

Value

A length `lag * p^2` named vector, the element names are constructed by: the scales index (crossed scales index) and lag index.

Note

To know more details about auto cross covariance, see the references.

Author(s)

Nan Xiao <http://r2s.name>

References

Wold, S., Jonsson, J., Sj\"orstr\"om, M., Sandberg, M., & R\"annar, S. (1993). DNA and peptide sequences and chemical processes multivariately modelled by principal component analysis and partial least-squares projections to latent structures. *Analytica chimica acta*, 277(2), 239–253.

Sj\"ostr\"om, M., R\"annar, S., & Wieslander, A. (1995). Polypeptide sequence property relationships in *Escherichia coli* based on auto cross covariances. *Chemometrics and intelligent laboratory systems*, 29(2), 295–305.

See Also

See extractScales for scales-based descriptors. For more details, see extractDescScales and extractProtFP.

Examples

```
p = 8    # p is the scales number
n = 200  # n is the amino acid number
lag = 7  # the lag paramter
mat = matrix(rnorm(p * n), nrow = p, ncol = n)
acc(mat, lag)
```

---

extractAAC *Amino Acid Composition Descriptor*

Description

Amino Acid Composition Descriptor

Usage

```
extractAAC(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Amino Acid Composition descriptor (Dim: 20).

Value

A length 20 named vector

Author(s)

Nan Xiao <http://r2s.name>

References

M. Bhasin, G. P. S. Raghava. Classification of Nuclear Receptors Based on Amino Acid Composition and Dipeptide Composition. *Journal of Biological Chemistry*, 2004, 279, 23262.

See Also

See extractDC and extractTC for Dipeptide Composition and Tripeptide Composition descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractAAC(x)
```

`extractAPAAC` *Amphiphilic Pseudo Amino Acid Composition Descriptor*

Description

Amphiphilic Pseudo Amino Acid Composition Descriptor

Usage

```
extractAPAAC(x, props = c("Hydrophobicity", "Hydrophilicity"), lambda = 30,
  w = 0.05, customprops = NULL)
```

Arguments

`x`	A character vector, as the input protein sequence.
`props`	A character vector, specifying the properties used. 2 properties are used by default, as listed below: `'Hydrophobicity'` Hydrophobicity value of the 20 amino acids `'Hydrophilicity'` Hydrophilicity value of the 20 amino acids
`lambda`	The lambda parameter for the APAAC descriptors, default is 30.
`w`	The weighting factor, default is 0.05.
`customprops`	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the Amphiphilic Pseudo Amino Acid Composition (APAAC) descriptor (Dim: `20 + (n * lambda)`, `n` is the number of properties selected, default is 80).

Value

A length `20 + n * lambda` named vector, `n` is the number of properties selected.

Note

Note the default `20 * 2` `prop` values have been already independently given in the function. Users could also specify other (up to 544) properties with the Accession Number in the `AAindex` data, with or without the default three properties, which means users should explicitly specify the properties to use. For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

Kuo-Chen Chou. Prediction of Protein Cellular Attributes Using Pseudo-Amino Acid Composition. *PROTEINS: Structure, Function, and Genetics*, 2001, 43: 246-255.

Type 2 pseudo amino acid composition. http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type2.htm

Kuo-Chen Chou. Using Amphiphilic Pseudo Amino Acid Composition to Predict Enzyme Subfamily Classes. *Bioinformatics*, 2005, 21, 10-19.

JACS, 1962, 84: 4240-4246. (C. Tanford). (The hydrophobicity data)

PNAS, 1981, 78:3824-3828 (T.P.Hopp & K.R.Woods). (The hydrophilicity data)

See Also

See `extractPAAC` for pseudo amino acid composition descriptor.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractAPAAC(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))

# Use 2 default properties, 4 properties in the AAindex database,
# and 3 cutomized properties
extractAPAAC(x, customprops = myprops,
             props = c('Hydrophobicity', 'Hydrophilicity',
```

```
'CIDH920105', 'BHAR880101',
'CHAM820101', 'CHAM820102',
'MyProp1', 'MyProp2', 'MyProp3'))
```

`extractBLOSUM` *BLOSUM and PAM Matrix-Derived Descriptors*

Description

BLOSUM and PAM Matrix-Derived Descriptors

Usage

```
extractBLOSUM(x, submat = "AABLOSUM62", k, lag, scale = TRUE,
  silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence.
`submat`	Substitution matrix for the 20 amino acids. Should be one of `AABLOSUM45`, `AABLOSUM50`, `AABLOSUM62`, `AABLOSUM80`, `AABLOSUM100`, `AAPAM30`, `AAPAM40`, `AAPAM70`, `AAPAM120`, `AAPAM250`. Default is `'AABLOSUM62'`.
`k`	Integer. The number of selected scales (i.e. the first `k` scales) derived by the substitution matrix. This could be selected according to the printed relative importance values.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the substitution matrix (`submat`) before doing eigen decomposition? Default is `TRUE`.
`silent`	Logical. Whether we print the relative importance of each scales (diagnal value of the eigen decomposition result matrix B) or not. Default is `TRUE`.

Details

This function calculates BLOSUM matrix-derived descriptors. For users' convenience, `protr` provides the BLOSUM45, BLOSUM50, BLOSUM62, BLOSUM80, BLOSUM100, PAM30, PAM40, PAM70, PAM120, and PAM250 matrices for the 20 amino acids to select.

Value

A length `lag * p^2` named vector, `p` is the number of scales selected.

Author(s)

Nan Xiao <http://r2s.name>

References

Georgiev, A. G. (2009). Interpretable numerical descriptors of amino acid space. Journal of Computational Biology, 16(5), 703–723.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
blosum = extractBLOSUM(x, submat = 'AABLOSUM62', k = 5, lag = 7, scale = TRUE, silent = FALSE)
```

`extractCTDC` *CTD Descriptors - Composition*

Description

CTD Descriptors - Composition

Usage

```
extractCTDC(x)
```

Arguments

`x`	A character vector, as the input protein sequence.

Details

This function calculates the Composition descriptor of the CTD descriptors (Dim: 21).

Value

A length 21 named vector

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extractCTDT` and `extractCTDD` for Transition and Distribution of the CTD descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractCTDC(x)
```

`extractCTDCClass` *CTD Descriptors - Composition (with Customized Amino Acid Classification Support)*

Description

CTD Descriptors - Composition (with Customized Amino Acid Classification Support)

Usage

```
extractCTDCClass(x, aagroup1, aagroup2, aagroup3)
```

Arguments

`x`	A character vector, as the input protein sequence.
`aagroup1`	A named list which contains the first group of customized amino acid classification. See example below.
`aagroup2`	A named list which contains the second group of customized amino acid classification. See example below.
`aagroup3`	A named list which contains the third group of customized amino acid classification. See example below.

Details

This function calculates the Composition descriptor of the CTD descriptors, with customized amino acid classification support.

Value

A length `k * 3` named vector, `k` is the number of amino acid properties used.

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extractCTDTClass` and `extractCTDDClass` for Transition and Distribution of the CTD descriptors with customized amino acid classification support.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]

# using five customized amino acid property classification
group1 = list(hydrophobicity  = c('R', 'K', 'E', 'D', 'Q', 'N'),
              normwaalsvolume = c('G', 'A', 'S', 'T', 'P', 'D', 'C'),
              polarizability  = c('G', 'A', 'S', 'D', 'T'),
              secondarystruct = c('E', 'A', 'L', 'M', 'Q', 'K', 'R', 'H'),
              solventaccess   = c('A', 'L', 'F', 'C', 'G', 'I', 'V', 'W'))

group2 = list(hydrophobicity  = c('G', 'A', 'S', 'T', 'P', 'H', 'Y'),
              normwaalsvolume = c('N', 'V', 'E', 'Q', 'I', 'L'),
              polarizability  = c('C', 'P', 'N', 'V', 'E', 'Q', 'I', 'L'),
              secondarystruct = c('V', 'I', 'Y', 'C', 'W', 'F', 'T'),
              solventaccess   = c('R', 'K', 'Q', 'E', 'N', 'D'))

group3 = list(hydrophobicity  = c('C', 'L', 'V', 'I', 'M', 'F', 'W'),
              normwaalsvolume = c('M', 'H', 'K', 'F', 'R', 'Y', 'W'),
              polarizability  = c('K', 'M', 'H', 'F', 'R', 'Y', 'W'),
              secondarystruct = c('G', 'N', 'P', 'S', 'D'),
              solventaccess   = c('M', 'S', 'P', 'T', 'H', 'Y'))

extractCTDCClass(x, aagroup1 = group1, aagroup2 = group2, aagroup3 = group3)
```

`extractCTDD` *CTD Descriptors - Distribution*

Description

CTD Descriptors - Distribution

Usage

```
extractCTDD(x)
```

Arguments

`x`	A character vector, as the input protein sequence.

Details

This function calculates the Distribution descriptor of the CTD descriptors (Dim: 105).

Value

A length 105 named vector

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extractCTDC` and `extractCTDT` for Composition and Transition of the CTD descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractCTDD(x)
```

`extractCTDDClass` *CTD Descriptors - Distribution (with Customized Amino Acid Classification Support)*

Description

CTD Descriptors - Distribution (with Customized Amino Acid Classification Support)

Usage

```
extractCTDDClass(x, aagroup1, aagroup2, aagroup3)
```

Arguments

`x`	A character vector, as the input protein sequence.
`aagroup1`	A named list which contains the first group of customized amino acid classification. See example below.
`aagroup2`	A named list which contains the second group of customized amino acid classification. See example below.
`aagroup3`	A named list which contains the third group of customized amino acid classification. See example below.

Details

This function calculates the Distribution descriptor of the CTD descriptors, with customized amino acid classification support.

Value

A length `k * 15` named vector, `k` is the number of amino acid properties used.

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extractCTDCClass` and `extractCTDTClass` for Composition and Transition of the CTD descriptors with customized amino acid classification support.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]

# using five customized amino acid property classification
group1 = list(hydrophobicity  = c('R', 'K', 'E', 'D', 'Q', 'N'),
              normwaalsvolume = c('G', 'A', 'S', 'T', 'P', 'D', 'C'),
              polarizability  = c('G', 'A', 'S', 'D', 'T'),
              secondarystruct = c('E', 'A', 'L', 'M', 'Q', 'K', 'R', 'H'),
              solventaccess   = c('A', 'L', 'F', 'C', 'G', 'I', 'V', 'W'))

group2 = list(hydrophobicity  = c('G', 'A', 'S', 'T', 'P', 'H', 'Y'),
              normwaalsvolume = c('N', 'V', 'E', 'Q', 'I', 'L'),
              polarizability  = c('C', 'P', 'N', 'V', 'E', 'Q', 'I', 'L'),
              secondarystruct = c('V', 'I', 'Y', 'C', 'W', 'F', 'T'),
              solventaccess   = c('R', 'K', 'Q', 'E', 'N', 'D'))

group3 = list(hydrophobicity  = c('C', 'L', 'V', 'I', 'M', 'F', 'W'),
              normwaalsvolume = c('M', 'H', 'K', 'F', 'R', 'Y', 'W'),
              polarizability  = c('K', 'M', 'H', 'F', 'R', 'Y', 'W'),
              secondarystruct = c('G', 'N', 'P', 'S', 'D'),
              solventaccess   = c('M', 'S', 'P', 'T', 'H', 'Y'))

extractCTDDClass(x, aagroup1 = group1, aagroup2 = group2, aagroup3 = group3)
```

`extractCTDT` *CTD Descriptors - Transition*

Description

CTD Descriptors - Transition

Usage

```
extractCTDT(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Transition descriptor of the CTD descriptors (Dim: 21).

Value

A length 21 named vector

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extractCTDC` and `extractCTDD` for Composition and Distribution of the CTD descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractCTDT(x)
```

`extractCTDTClass` *CTD Descriptors - Transition (with Customized Amino Acid Classification Support)*

Description

CTD Descriptors - Transition (with Customized Amino Acid Classification Support)

Usage

```
extractCTDTClass(x, aagroup1, aagroup2, aagroup3)
```

Arguments

`x`	A character vector, as the input protein sequence.
`aagroup1`	A named list which contains the first group of customized amino acid classification. See example below.
`aagroup2`	A named list which contains the second group of customized amino acid classification. See example below.
`aagroup3`	A named list which contains the third group of customized amino acid classification. See example below.

Details

This function calculates the Transition descriptor of the CTD descriptors, with customized amino acid classification support.

Value

A length `k * 3` named vector, `k` is the number of amino acid properties used.

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extractCTDCClass` and `extractCTDDClass` for Composition and Distribution of the CTD descriptors with customized amino acid classification support.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]

# using five customized amino acid property classification
group1 = list(hydrophobicity  = c('R', 'K', 'E', 'D', 'Q', 'N'),
              normwaalsvolume = c('G', 'A', 'S', 'T', 'P', 'D', 'C'),
              polarizability  = c('G', 'A', 'S', 'D', 'T'),
              secondarystruct = c('E', 'A', 'L', 'M', 'Q', 'K', 'R', 'H'),
              solventaccess   = c('A', 'L', 'F', 'C', 'G', 'I', 'V', 'W'))

group2 = list(hydrophobicity  = c('G', 'A', 'S', 'T', 'P', 'H', 'Y'),
              normwaalsvolume = c('N', 'V', 'E', 'Q', 'I', 'L'),
              polarizability  = c('C', 'P', 'N', 'V', 'E', 'Q', 'I', 'L'),
              secondarystruct = c('V', 'I', 'Y', 'C', 'W', 'F', 'T'),
              solventaccess   = c('R', 'K', 'Q', 'E', 'N', 'D'))

group3 = list(hydrophobicity  = c('C', 'L', 'V', 'I', 'M', 'F', 'W'),
              normwaalsvolume = c('M', 'H', 'K', 'F', 'R', 'Y', 'W'),
              polarizability  = c('K', 'M', 'H', 'F', 'R', 'Y', 'W'),
              secondarystruct = c('G', 'N', 'P', 'S', 'D'),
              solventaccess   = c('M', 'S', 'P', 'T', 'H', 'Y'))

extractCTDTClass(x, aagroup1 = group1, aagroup2 = group2, aagroup3 = group3)
```

`extractCTriad` *Conjoint Triad Descriptor*

Description

Conjoint Triad Descriptor

Usage

```
extractCTriad(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Conjoint Triad descriptor (Dim: 343).

Value

A length 343 named vector

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

J.W. Shen, J. Zhang, X.M. Luo, W.L. Zhu, K.Q. Yu, K.X. Chen, Y.X. Li, H.L. Jiang. Predicting Protein-protein Interactions Based Only on Sequences Information. *Proceedings of the National Academy of Sciences*. 007, 104, 4337–4341.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractCTriad(x)
```

---

extractCTriadClass *Conjoint Triad Descriptor (with Customized Amino Acid Classification Support)*

Description

Conjoint Triad Descriptor (with Customized Amino Acid Classification Support)

Usage

```
extractCTriadClass(x, aaclass)
```

Arguments

x	A character vector, as the input protein sequence.
aaclass	A list containing the customized amino acid classification. See example below.

Details

This function calculates the Conjoint Triad descriptor, with customized amino acid classification support.

Value

A length `k^3` named vector, where `k` is the number of customized classes of the amino acids.

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

J.W. Shen, J. Zhang, X.M. Luo, W.L. Zhu, K.Q. Yu, K.X. Chen, Y.X. Li, H.L. Jiang. Predicting Protein-protein Interactions Based Only on Sequences Information. *Proceedings of the National Academy of Sciences*. 007, 104, 4337–4341.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]

# using customized amino acid classification (normalized van der Waals volume)
newclass = list(c('G', 'A', 'S', 'T', 'P', 'D', 'C'),
                c('N', 'V', 'E', 'Q', 'I', 'L'),
                c('M', 'H', 'K', 'F', 'R', 'Y', 'W'))

extractCTriadClass(x, aaclass = newclass)
```

---

extractDC *Dipeptide Composition Descriptor*

Description

Dipeptide Composition Descriptor

Usage

```
extractDC(x)
```

Arguments

`x`	A character vector, as the input protein sequence.

Details

This function calculates the Dipeptide Composition descriptor (Dim: 400).

Value

A length 400 named vector

Author(s)

Nan Xiao <http://r2s.name>

References

M. Bhasin, G. P. S. Raghava. Classification of Nuclear Receptors Based on Amino Acid Composition and Dipeptide Composition. *Journal of Biological Chemistry*, 2004, 279, 23262.

See Also

See `extractAAC` and `extractTC` for Amino Acid Composition and Tripeptide Composition descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractDC(x)
```

---

extractDescScales *Scales-Based Descriptors with 20+ classes of Molecular Descriptors*

Description

Scales-Based Descriptors with 20+ classes of Molecular Descriptors

Usage

```
extractDescScales(x, propmat, index = NULL, pc, lag, scale = TRUE,
  silent = TRUE)
```

Arguments

x	A character vector, as the input protein sequence.
propmat	The matrix containing the descriptor set for the amino acids, which could be chosen from `AAMOE2D`, `AAMOE3D`, `AACPSA`, `AADescAll`, `AA2DACOR`, `AA3DMoRSE`, `AAACF`, `AABurden`, `AAConn`, `AAConst`, `AAEdgeAdj`, `AAEigIdx`, `AAFGC`, `AAGeom`, `AAGETAWAY`, `AAInfo`, `AAMolProp`, `AARandic`, `AARDF`, `AATopo`, `AATopoChg`, `AAWalk`, `AAWHIM`.

index	Integer vector or character vector. Specify which molecular descriptors to select from one of these deseriptor sets by specify the numerical or character index of the molecular descriptors in the descriptor set. Default is NULL, means selecting all the molecular descriptors in this descriptor set.
pc	Integer. The maximum dimension of the space which the data are to be represented in. Must be no greater than the number of AA properties provided.
lag	The lag parameter. Must be less than the amino acids.
scale	Logical. Should we auto-scale the property matrix (propmat) before doing MDS? Default is TRUE.
silent	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is TRUE.

Details

This function calculates the scales-based descriptors with molecular descriptors sets calculated by Dragon, Discovery Studio and MOE. Users could specify which molecular descriptors to select from one of these deseriptor sets by specify the numerical or character index of the molecular descriptors in the descriptor set.

Value

A length `lag * p^2` named vector, p is the number of scales selected.

Author(s)

Nan Xiao <http://r2s.name>

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
descscales = extractDescScales(x, propmat = 'AATopo', index = c(37:41, 43:47),
                               pc = 5, lag = 7, silent = FALSE)
```

---

extractFAScales *Scales-Based Descriptors derived by Factor Analysis*

Description

Scales-Based Descriptors derived by Factor Analysis

Usage

```
extractFAScales(x, propmat, factors, scores = "regression", lag,
  scale = TRUE, silent = TRUE)
```

Arguments

x	A character vector, as the input protein sequence.
propmat	A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
factors	Integer. The number of factors to be fitted. Must be no greater than the number of AA properties provided.
scores	Type of scores to produce. The default is "regression", which gives Thompson's scores, "Bartlett" given Bartlett's weighted least-squares scores.
lag	The lag parameter. Must be less than the amino acids number in the protein sequence.
scale	Logical. Should we auto-scale the property matrix (propmat) before doing Factor Analysis? Default is TRUE.
silent	Logical. Whether we print the SS loadings, proportion of variance and the cumulative proportion of the selected factors or not. Default is TRUE.

Details

This function calculates scales-based descriptors derived by Factor Analysis (FA). Users could provide customized amino acid property matrices.

Value

A length lag * p^2 named vector, p is the number of scales (factors) selected.

Author(s)

Nan Xiao <http://r2s.name>

References

Atchley, W. R., Zhao, J., Fernandes, A. D., & Druke, T. (2005). Solving the protein sequence metric problem. Proceedings of the National Academy of Sciences of the United States of America, 102(18), 6395-6400.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
data(AATopo)
tprops = AATopo[, c(37:41, 43:47)]  # select a set of topological descriptors
fa = extractFAScales(x, propmat = tprops, factors = 5, lag = 7, silent = FALSE)
```

`extractGeary` *Geary Autocorrelation Descriptor*

Description

Geary Autocorrelation Descriptor

Usage

```
extractGeary(x, props = c("CIDH920105", "BHAR880101", "CHAM820101",
  "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"),
  nlag = 30L, customprops = NULL)
```

Arguments

`x`	A character vector, as the input protein sequence.
`props`	A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below: **AccNo. CIDH920105** Normalized average hydrophobicity scales (Cid et al., 1992) **AccNo. BHAR880101** Average flexibility indices (Bhaskaran-Ponnuswamy, 1988) **AccNo. CHAM820101** Polarizability parameter (Charton-Charton, 1982) **AccNo. CHAM820102** Free energy of solution in water, kcal/mole (Charton-Charton, 1982) **AccNo. CHOC760101** Residue accessible surface area in tripeptide (Chothia, 1976) **AccNo. BIGC670101** Residue volume (Bigelow, 1967) **AccNo. CHAM810101** Steric parameter (Charton, 1981) **AccNo. DAYM780201** Relative mutability (Dayhoff et al., 1978b)
`nlag`	Maximum value of the lag parameter. Default is `30`.
`customprops`	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the Geary autocorrelation descriptor (Dim: `length(props) * nlag`).

Value

A length `nlag` named vector

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

AAindex: Amino acid index database. http://www.genome.ad.jp/dbget/aaindex.html

Feng, Z.P. and Zhang, C.T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. *Journal of Protein Chemistry*, 19, 269-275.

Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, 27, 451-477.

Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an Usage from an Amerindian tribal population. *American Journal of Physical Anthropology*, 129, 121-131.

See Also

See extractMoreauBroto and extractMoran for Moreau-Broto autocorrelation descriptors and Moran autocorrelation descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractGeary(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))

# Use 4 properties in the AAindex database, and 3 cutomized properties
extractGeary(x, customprops = myprops,
             props = c('CIDH920105', 'BHAR880101',
                       'CHAM820101', 'CHAM820102',
                       'MyProp1', 'MyProp2', 'MyProp3'))
```

`extractMDSScales` *Scales-Based Descriptors derived by Multidimensional Scaling*

Description

Scales-Based Descriptors derived by Multidimensional Scaling

Usage

```
extractMDSScales(x, propmat, k, lag, scale = TRUE, silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence.
`propmat`	A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
`k`	Integer. The maximum dimension of the space which the data are to be represented in. Must be no greater than the number of AA properties provided.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the property matrix (`propmat`) before doing MDS? Default is `TRUE`.
`silent`	Logical. Whether we print the `k` eigenvalues computed during the scaling process or not. Default is `TRUE`.

Details

This function calculates scales-based descriptors derived by Multidimensional Scaling (MDS). Users could provide customized amino acid property matrices.

Value

A length `lag * p^2` named vector, `p` is the number of scales (dimensionality) selected.

Author(s)

Nan Xiao <http://r2s.name>

References

Venkatarajan, M. S., & Braun, W. (2001). New quantitative descriptors of amino acids based on multidimensional scaling of a large number of physical-chemical properties. Molecular modeling annual, 7(12), 445–453.

See Also

See `extractScales` for scales-based descriptors derived by Principal Components Analysis.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
data(AATopo)
tprops = AATopo[, c(37:41, 43:47)]  # select a set of topological descriptors
mds = extractMDSScales(x, propmat = tprops, k = 5, lag = 7, silent = FALSE)
```

`extractMoran` *Moran Autocorrelation Descriptor*

Description

Moran Autocorrelation Descriptor

Usage

```
extractMoran(x, props = c("CIDH920105", "BHAR880101", "CHAM820101",
  "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"),
  nlag = 30L, customprops = NULL)
```

Arguments

`x`	A character vector, as the input protein sequence.
`props`	A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below: **AccNo. CIDH920105** Normalized average hydrophobicity scales (Cid et al., 1992) **AccNo. BHAR880101** Average flexibility indices (Bhaskaran-Ponnuswamy, 1988) **AccNo. CHAM820101** Polarizability parameter (Charton-Charton, 1982) **AccNo. CHAM820102** Free energy of solution in water, kcal/mole (Charton-Charton, 1982) **AccNo. CHOC760101** Residue accessible surface area in tripeptide (Chothia, 1976) **AccNo. BIGC670101** Residue volume (Bigelow, 1967) **AccNo. CHAM810101** Steric parameter (Charton, 1981) **AccNo. DAYM780201** Relative mutability (Dayhoff et al., 1978b)
`nlag`	Maximum value of the lag parameter. Default is `30`.
`customprops`	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the Moran autocorrelation descriptor (Dim: `length(props) * nlag`).

Value

A length `nlag` named vector

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

AAindex: Amino acid index database. http://www.genome.ad.jp/dbget/aaindex.html

Feng, Z.P. and Zhang, C.T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. *Journal of Protein Chemistry*, 19, 269-275.

Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, 27, 451-477.

Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an Usage from an Amerindian tribal population. *American Journal of Physical Anthropology*, 129, 121-131.

See Also

See `extractMoreauBroto` and `extractGeary` for Moreau-Broto autocorrelation descriptors and Geary autocorrelation descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractMoran(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))
```

```
# Use 4 properties in the AAindex database, and 3 cutomized properties
extractMoran(x, customprops = myprops,
             props = c('CIDH920105', 'BHAR880101',
                       'CHAM820101', 'CHAM820102',
                       'MyProp1', 'MyProp2', 'MyProp3'))
```

`extractMoreauBroto` *Normalized Moreau-Broto Autocorrelation Descriptor*

Description

Normalized Moreau-Broto Autocorrelation Descriptor

Usage

```
extractMoreauBroto(x, props = c("CIDH920105", "BHAR880101", "CHAM820101",
  "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"),
  nlag = 30L, customprops = NULL)
```

Arguments

`x`	A character vector, as the input protein sequence.
`props`	A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below: **AccNo. CIDH920105** Normalized average hydrophobicity scales (Cid et al., 1992) **AccNo. BHAR880101** Average flexibility indices (Bhaskaran-Ponnuswamy, 1988) **AccNo. CHAM820101** Polarizability parameter (Charton-Charton, 1982) **AccNo. CHAM820102** Free energy of solution in water, kcal/mole (Charton-Charton, 1982) **AccNo. CHOC760101** Residue accessible surface area in tripeptide (Chothia, 1976) **AccNo. BIGC670101** Residue volume (Bigelow, 1967) **AccNo. CHAM810101** Steric parameter (Charton, 1981) **AccNo. DAYM780201** Relative mutability (Dayhoff et al., 1978b)
`nlag`	Maximum value of the lag parameter. Default is `30`.
`customprops`	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the normalized Moreau-Broto autocorrelation descriptor (Dim: `length(props) * nlag`).

Value

A length `nlag` named vector

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

AAindex: Amino acid index database. http://www.genome.ad.jp/dbget/aaindex.html

Feng, Z.P. and Zhang, C.T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. *Journal of Protein Chemistry*, 19, 269-275.

Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, 27, 451-477.

Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an Usage from an Amerindian tribal population. *American Journal of Physical Anthropology*, 129, 121-131.

See Also

See `extractMoran` and `extractGeary` for Moran autocorrelation descriptors and Geary autocorrelation descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractMoreauBroto(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))
```

```
# Use 4 properties in the AAindex database, and 3 cutomized properties
extractMoreauBroto(x, customprops = myprops,
                   props = c('CIDH920105', 'BHAR880101',
                             'CHAM820101', 'CHAM820102',
                             'MyProp1', 'MyProp2', 'MyProp3'))
```

`extractPAAC` *Pseudo Amino Acid Composition Descriptor*

Description

Pseudo Amino Acid Composition Descriptor

Usage

```
extractPAAC(x, props = c("Hydrophobicity", "Hydrophilicity", "SideChainMass"),
  lambda = 30, w = 0.05, customprops = NULL)
```

Arguments

`x`	A character vector, as the input protein sequence.
`props`	A character vector, specifying the properties used. 3 properties are used by default, as listed below: `'Hydrophobicity'` Hydrophobicity value of the 20 amino acids `'Hydrophilicity'` Hydrophilicity value of the 20 amino acids `'SideChainMass'` Side-chain mass of the 20 amino acids
`lambda`	The lambda parameter for the PAAC descriptors, default is 30.
`w`	The weighting factor, default is 0.05.
`customprops`	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the Pseudo Amino Acid Composition (PAAC) descriptor (Dim: `20 + lambda`, default is 50).

Value

A length `20 + lambda` named vector

Note

Note the default `20 * 3` `prop` values have been already independently given in the function. Users could also specify other (up to 544) properties with the Accession Number in the `AAindex` data, with or without the default three properties, which means users should explicitly specify the properties to use. For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Nan Xiao <http://r2s.name>

References

Kuo-Chen Chou. Prediction of Protein Cellular Attributes Using Pseudo-Amino Acid Composition. *PROTEINS: Structure, Function, and Genetics*, 2001, 43: 246-255.

Type 1 pseudo amino acid composition. http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type1.htm

Kuo-Chen Chou. Using Amphiphilic Pseudo Amino Acid Composition to Predict Enzyme Subfamily Classes. *Bioinformatics*, 2005, 21, 10-19.

JACS, 1962, 84: 4240-4246. (C. Tanford). (The hydrophobicity data)

PNAS, 1981, 78:3824-3828 (T.P.Hopp & K.R.Woods). (The hydrophilicity data)

CRC Handbook of Chemistry and Physics, 66th ed., CRC Press, Boca Raton, Florida (1985). (The side-chain mass data)

R.M.C. Dawson, D.C. Elliott, W.H. Elliott, K.M. Jones, Data for Biochemical Research 3rd ed., Clarendon Press Oxford (1986). (The side-chain mass data)

See Also

See `extractAPAAC` for amphiphilic pseudo amino acid composition descriptor.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractPAAC(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))
```

```
# Use 3 default properties, 4 properties in the AAindex database,
# and 3 cutomized properties
extractPAAC(x, customprops = myprops,
            props = c('Hydrophobicity', 'Hydrophilicity', 'SideChainMass',
                      'CIDH920105', 'BHAR880101',
                      'CHAM820101', 'CHAM820102',
                      'MyProp1', 'MyProp2', 'MyProp3'))
```

`extractProtFP` *Amino Acid Properties Based Scales Descriptors (Protein Fingerprint)*

Description

Amino Acid Properties Based Scales Descriptors (Protein Fingerprint)

Usage

```
extractProtFP(x, index = NULL, pc, lag, scale = TRUE, silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence.
`index`	Integer vector or character vector. Specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database. Default is `NULL`, means selecting all the AA properties in the AAindex database.
`pc`	Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the property matrix before PCA? Default is `TRUE`.
`silent`	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is `TRUE`.

Details

This function calculates amino acid properties based scales descriptors (protein fingerprint). Users could specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database.

Value

A length `lag * p^2` named vector, `p` is the number of scales (principal components) selected.

Author(s)

Nan Xiao <http://r2s.name>

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
fp = extractProtFP(x, index = c(160:165, 258:296), pc = 5, lag = 7, silent = FALSE)
```

---

extractProtFPGap *Amino Acid Properties Based Scales Descriptors (Protein Fingerprint) with Gap Support*

Description

Amino Acid Properties Based Scales Descriptors (Protein Fingerprint) with Gap Support

Usage

```
extractProtFPGap(x, index = NULL, pc, lag, scale = TRUE, silent = TRUE)
```

Arguments

x	A character vector, as the input protein sequence. Use '-' to represent gaps in the sequence.
index	Integer vector or character vector. Specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database. Default is NULL, means selecting all the AA properties in the AAindex database.
pc	Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.
lag	The lag parameter. Must be less than the amino acids.
scale	Logical. Should we auto-scale the property matrix before PCA? Default is TRUE.
silent	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is TRUE.

Details

This function calculates amino acid properties based scales descriptors (protein fingerprint) with gap support. Users could specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database.

Value

A length `lag * p^2` named vector, p is the number of scales (principal components) selected.

Author(s)

Nan Xiao <http://r2s.name>

Examples

```
# amino acid sequence with gaps
x = readFASTA(system.file('protseq/align.fasta', package = 'protr'))$`IXI_235`
fp = extractProtFPGap(x, index = c(160:165, 258:296), pc = 5, lag = 7, silent = FALSE)
```

`extractPSSM` *Compute PSSM (Position-Specific Scoring Matrix) for given protein sequence*

Description

Compute PSSM (Position-Specific Scoring Matrix) for given protein sequence

Usage

```
extractPSSM(seq, start.pos = 1L, end.pos = nchar(seq),
  psiblast.path = NULL, makeblastdb.path = NULL, database.path = NULL,
  iter = 5, silent = TRUE, evalue = 10L, word.size = NULL,
  gapopen = NULL, gapextend = NULL, matrix = "BLOSUM62",
  threshold = NULL, seg = "no", soft.masking = FALSE,
  culling.limit = NULL, best.hit.overhang = NULL,
  best.hit.score.edge = NULL, xdrop.ungap = NULL, xdrop.gap = NULL,
  xdrop.gap.final = NULL, window.size = NULL, gap.trigger = 22L,
  num.threads = 1L, pseudocount = 0L, inclusion.ethresh = 0.002)
```

Arguments

`seq`	Character vector, as the input protein sequence.
`start.pos`	Optional integer denoting the start position of the fragment window. Default is 1, i.e. the first amino acid of the given sequence.
`end.pos`	Optional integer denoting the end position of the fragment window. Default is `nchar(seq)`, i.e. the last amino acid of the given sequence.
`psiblast.path`	Character string indicating the path of the `psiblast` program. If NCBI Blast+ was previously installed in the operation system, the path will be automatically detected.
`makeblastdb.path`	Character string indicating the path of the `makeblastdb` program. If NCBI Blast+ was previously installed in the system, the path will be automatically detected.
`database.path`	Character string indicating the path of a reference database (a FASTA file).
`iter`	Number of iterations to perform for PSI-Blast.

silent	Logical. Whether the PSI-Blast running output should be shown or not (May not work on some Windows versions and PSI-Blast versions), default is TRUE.
evalue	Expectation value (E) threshold for saving hits. Default is 10.
word.size	Word size for wordfinder algorithm. An integer >= 2.
gapopen	Integer. Cost to open a gap.
gapextend	Integer. Cost to extend a gap.
matrix	Character string. The scoring matrix name (default is 'BLOSUM62').
threshold	Minimum word score such that the word is added to the BLAST lookup table. A real value >= 0.
seg	Character string. Filter query sequence with SEG ('yes', 'window locut hicut', or 'no' to disable) Default is 'no'.
soft.masking	Logical. Apply filtering locations as soft masks? Default is FALSE.
culling.limit	An integer >= 0. If the query range of a hit is enveloped by that of at least this many higher-scoring hits, delete the hit. Incompatible with best.hit.overhang and best_hit_score_edge.
best.hit.overhang	Best Hit algorithm overhang value (A real value >= 0 and =< 0.5, recommended value: 0.1). Incompatible with culling_limit.
best.hit.score.edge	Best Hit algorithm score edge value (A real value >=0 and =< 0.5, recommended value: 0.1). Incompatible with culling_limit.
xdrop.ungap	X-dropoff value (in bits) for ungapped extensions.
xdrop.gap	X-dropoff value (in bits) for preliminary gapped extensions.
xdrop.gap.final	X-dropoff value (in bits) for final gapped alignment.
window.size	An integer >= 0. Multiple hits window size, To specify 1-hit algorithm, use 0.
gap.trigger	Number of bits to trigger gapping. Default is 22.
num.threads	Integer. Number of threads (CPUs) to use in the BLAST search. Default is 1.
pseudocount	Integer. Pseudo-count value used when constructing PSSM. Default is 0.
inclusion.ethresh	E-value inclusion threshold for pairwise alignments. Default is 0.002.

Details

This function calculates the PSSM (Position-Specific Scoring Matrix) derived by PSI-Blast for given protein sequence or peptides. For given protein sequences or peptides, PSSM represents the log-likelihood of the substitution of the 20 types of amino acids at that position in the sequence. Note that the output value is not normalized.

Value

The original PSSM, a numeric matrix which has end.pos - start.pos + 1 columns and 20 named rows.

Note

The function requires the `makeblastdb` and `psiblast` programs to be properly installed in the operation system or their paths provided.

The two command-line programs are included in the NCBI-BLAST+ software package. To install NCBI Blast+, just open the NCBI FTP site using web browser or FTP software: `ftp://anonymous@ftp.ncbi.nlm.nih.gov:21/blast/executables/blast+/LATEST/` then download the executable version of BLAST+ according to your operation system, and compile or install the downloaded source code or executable program.

Ubuntu/Debian users can directly use the command `sudo apt-get install ncbi-blast+` to install NCBI Blast+. For OS X users, download `ncbi-blast- ... .dmg` then install. For Windows users, download `ncbi-blast- ... .exe` then install.

Author(s)

Nan Xiao <http://r2s.name>

References

Altschul, Stephen F., et al. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." *Nucleic acids research* 25.17 (1997): 3389–3402.

Ye, Xugang, Guoli Wang, and Stephen F. Altschul. "An assessment of substitution scores for protein profile-profile comparison." *Bioinformatics* 27.24 (2011): 3356–3363.

Rangwala, Huzefa, and George Karypis. "Profile-based direct kernels for remote homology detection and fold recognition." *Bioinformatics* 21.23 (2005): 4239–4247.

See Also

extractPSSMFeature extractPSSMAcc

Examples

```
if (Sys.which('makeblastdb') == '' | Sys.which('psiblast') == '') {
  cat('Could not find makeblastdb or psiblast. Please install NCBI Blast+ first.')
} else {
  x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
  dbpath = tempfile('tempdb', fileext = '.fasta')
  invisible(file.copy(from = system.file('protseq/Plasminogen.fasta',
                                         package = 'protr'), to = dbpath))
  pssmmat = extractPSSM(seq = x, database.path = dbpath)
  dim(pssmmat)  # 20 x 562 (P00750: length 562, 20 Amino Acids)
}
```

`extractPSSMAcc` *Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix) and auto cross covariance*

Description

Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix) and auto cross covariance

Usage

```
extractPSSMAcc(pssmmat, lag)
```

Arguments

`pssmmat`	The PSSM computed by `extractPSSM`.
`lag`	The lag parameter. Must be less than the number of amino acids in the sequence (i.e. the number of columns in the PSSM matrix).

Details

This function calculates the feature vector based on the PSSM by running PSI-Blast and auto cross covariance tranformation.

Value

A length `lag * 20^2` named numeric vector, the element names are derived by the amino acid name abbreviation (crossed amino acid name abbreviation) and lag index.

Author(s)

Nan Xiao <http://r2s.name>

References

Wold, S., Jonsson, J., Sj\"orstr\"om, M., Sandberg, M., & R\"annar, S. (1993). DNA and peptide sequences and chemical processes multivariately modelled by principal component analysis and partial least-squares projections to latent structures. *Analytica chimica acta*, 277(2), 239–253.

See Also

extractPSSM extractPSSMFeature

Examples

```
if (Sys.which('makeblastdb') == '' | Sys.which('psiblast') == '') {
  cat('Could not find makeblastdb or psiblast. Please install NCBI Blast+ first.')
} else {
  x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
  dbpath = tempfile('tempdb', fileext = '.fasta')
  invisible(file.copy(from = system.file('protseq/Plasminogen.fasta',
                                         package = 'protr'), to = dbpath))
  pssmmat = extractPSSM(seq = x, database.path = dbpath)
  pssmacc = extractPSSMAcc(pssmmat, lag = 3)
  tail(pssmacc)
}
```

`extractPSSMFeature` *Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix)*

Description

Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix)

Usage

```
extractPSSMFeature(pssmmat)
```

Arguments

`pssmmat`	The PSSM computed by `extractPSSM`.

Details

This function calculates the profile-based protein representation derived by PSSM. The feature vector is based on the PSSM computed by `extractPSSM`. For a given sequence, The PSSM feature represents the log-likelihood of the substitution of the 20 types of amino acids at that position in the sequence. Each PSSM feature value in the vector represents the degree of conservation of a given amino acid type. The value is normalized to interval (0, 1) by the transformation 1/(1+e^(-x)).

Value

A numeric vector which has `20 x N` named elements, where `N` is the size of the window (number of rows of the PSSM).

Author(s)

Nan Xiao <http://r2s.name>

References

Ye, Xugang, Guoli Wang, and Stephen F. Altschul. "An assessment of substitution scores for protein profile-profile comparison." *Bioinformatics* 27.24 (2011): 3356–3363.

Rangwala, Huzefa, and George Karypis. "Profile-based direct kernels for remote homology detection and fold recognition." *Bioinformatics* 21.23 (2005): 4239–4247.

See Also

extractPSSM extractPSSMAcc

Examples

```
if (Sys.which('makeblastdb') == '' | Sys.which('psiblast') == '') {
  cat('Could not find makeblastdb or psiblast. Please install NCBI Blast+ first.')
} else {
  x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
  dbpath = tempfile('tempdb', fileext = '.fasta')
  invisible(file.copy(from = system.file('protseq/Plasminogen.fasta',
                                         package = 'protr'), to = dbpath))
  pssmmat = extractPSSM(seq = x, database.path = dbpath)
  pssmfeature = extractPSSMFeature(pssmmat)
  head(pssmfeature)
}
```

---

extractQSO *Quasi-Sequence-Order (QSO) Descriptor*

Description

Quasi-Sequence-Order (QSO) Descriptor

Usage

```
extractQSO(x, nlag = 30, w = 0.1)
```

Arguments

x	A character vector, as the input protein sequence.
nlag	The maximum lag, defualt is 30.
w	The weighting factor, default is 0.1.

Details

This function calculates the Quasi-Sequence-Order (QSO) descriptor (Dim: `20 + 20 + (2 * nlag)`, default is 100).

Value

A length `20 + 20 + (2 * nlag)` named vector

Author(s)

Nan Xiao <http://r2s.name>

References

Kuo-Chen Chou. Prediction of Protein Subcellar Locations by Incorporating Quasi-Sequence-Order Effect. *Biochemical and Biophysical Research Communications*, 2000, 278, 477-483.

Kuo-Chen Chou and Yu-Dong Cai. Prediction of Protein Sucellular Locations by GO-FunD-PseAA Predictor. *Biochemical and Biophysical Research Communications*, 2004, 320, 1236-1239.

Gisbert Schneider and Paul Wrede. The Rational Design of Amino Acid Sequences by Artifical Neural Networks and Simulated Molecular Evolution: Do Novo Design of an Idealized Leader Cleavge Site. *Biophys Journal*, 1994, 66, 335-344.

See Also

See `extractSOCN` for sequence-order-coupling numbers.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractQSO(x)
```

`extractScales` *Scales-Based Descriptors derived by Principal Components Analysis*

Description

Scales-Based Descriptors derived by Principal Components Analysis

Usage

```
extractScales(x, propmat, pc, lag, scale = TRUE, silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence.
`propmat`	A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
`pc`	Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.

lag	The lag parameter. Must be less than the amino acids.
scale	Logical. Should we auto-scale the property matrix (propmat) before PCA? Default is TRUE.
silent	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is TRUE.

Details

This function calculates scales-based descriptors derived by Principal Components Analysis (PCA). Users could provide customized amino acid property matrices. This function implements the core computation procedure needed for the scales-based descriptors derived by AA-Properties (AAindex) and scales-based descriptors derived by 20+ classes of 2D and 3D molecular descriptors (Topological, WHIM, VHSE, etc.) in the protr package.

Value

A length `lag * p^2` named vector, p is the number of scales (principal components) selected.

Author(s)

Nan Xiao <http://r2s.name>

See Also

See extractDescScales scales descriptors based on 20+ classes of molecular descriptors, and extractProtFP for amino acid property based scales descriptors (protein fingerprint).

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
data(AAindex)
AAidxmat = t(na.omit(as.matrix(AAindex[, 7:26])))
scales = extractScales(x, propmat = AAidxmat, pc = 5, lag = 7, silent = FALSE)
```

---

extractScalesGap *Scales-Based Descriptors derived by Principal Components Analysis (with Gap Support)*

Description

Scales-Based Descriptors derived by Principal Components Analysis (with Gap Support)

Usage

```
extractScalesGap(x, propmat, pc, lag, scale = TRUE, silent = TRUE)
```

Arguments

x	A character vector, as the input protein sequence. Use '-' to represent gaps in the sequence.
propmat	A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
pc	Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.
lag	The lag parameter. Must be less than the amino acids.
scale	Logical. Should we auto-scale the property matrix (propmat) before PCA? Default is TRUE.
silent	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is TRUE.

Details

This function calculates scales-based descriptors derived by Principal Components Analysis (PCA), with gap support. Users could provide customized amino acid property matrices. This function implements the core computation procedure needed for the scales-based descriptors derived by AA-Properties (AAindex) and scales-based descriptors derived by 20+ classes of 2D and 3D molecular descriptors (Topological, WHIM, VHSE, etc.) in the protr package.

Value

A length `lag * p^2` named vector, p is the number of scales (principal components) selected.

Author(s)

Nan Xiao <http://r2s.name>

See Also

See extractProtFPGap for amino acid property based scales descriptors (protein fingerprint) with gap support.

Examples

```
# amino acid sequence with gaps
x = readFASTA(system.file('protseq/align.fasta', package = 'protr'))$`IXI_235`
data(AAindex)
AAidxmat = t(na.omit(as.matrix(AAindex[, 7:26])))
scales = extractScalesGap(x, propmat = AAidxmat, pc = 5, lag = 7, silent = FALSE)
```

`extractSOCN` *Sequence-Order-Coupling Numbers*

Description

Sequence-Order-Coupling Numbers

Usage

```
extractSOCN(x, nlag = 30)
```

Arguments

`x`	A character vector, as the input protein sequence.
`nlag`	The maximum lag, defualt is 30.

Details

This function calculates the Sequence-Order-Coupling Numbers (Dim: `nlag * 2`, default is 60).

Value

A length `nlag * 2` named vector

Author(s)

Nan Xiao <http://r2s.name>

References

Kuo-Chen Chou. Prediction of Protein Subcellar Locations by Incorporating Quasi-Sequence-Order Effect. *Biochemical and Biophysical Research Communications*, 2000, 278, 477-483.

Kuo-Chen Chou and Yu-Dong Cai. Prediction of Protein Sucellular Locations by GO-FunD-PseAA Predictor. *Biochemical and Biophysical Research Communications*, 2004, 320, 1236-1239.

Gisbert Schneider and Paul Wrede. The Rational Design of Amino Acid Sequences by Artifical Neural Networks and Simulated Molecular Evolution: Do Novo Design of an Idealized Leader Cleavge Site. *Biophys Journal*, 1994, 66, 335-344.

See Also

See `extractQSO` for quasi-sequence-order descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractSOCN(x)
```

`extractTC` *Tripeptide Composition Descriptor*

Description

Tripeptide Composition Descriptor

Usage

```
extractTC(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Tripeptide Composition descriptor (Dim: 8000).

Value

A length 8000 named vector

Author(s)

Nan Xiao <http://r2s.name>

References

M. Bhasin, G. P. S. Raghava. Classification of Nuclear Receptors Based on Amino Acid Composition and Dipeptide Composition. *Journal of Biological Chemistry*, 2004, 279, 23262.

See Also

See `extractAAC` and `extractDC` for Amino Acid Composition and Dipeptide Composition descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
extractTC(x)
```

getUniProt *Get Protein Sequences from UniProt by Protein ID*

Description

Get Protein Sequences from UniProt by Protein ID

Usage

```
getUniProt(id)
```

Arguments

id	A character vector, as the protein ID(s).

Details

This function get protein sequences from uniprot.org by protein ID(s).

Value

A list, each component contains one of the protein sequences.

Author(s)

Nan Xiao <http://r2s.name>

References

UniProt. http://www.uniprot.org/

See Also

See readFASTA for reading FASTA format files.

Examples

```
# Network latency may slow down this example
# Only test this when your connection is fast enough
ids = c('P00750', 'P00751', 'P00752')
getUniProt(ids)
```

OptAA3d — *OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)*

Description

OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)

Details

OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)

Examples

```
# This operation requires the rcdk package
# require(rcdk)
# optaa3d = load.molecules(system.file('sysdata/OptAA3d.sdf', package = 'protr'))
# view.molecule.2d(optaa3d[[1]])  # view the first AA
```

---

parGOSim — *Protein Sequence Similarity Calculation based on Gene Ontology (GO) Similarity*

Description

Protein Sequence Similarity Calculation based on Gene Ontology (GO) Similarity

Usage

```
parGOSim(golist, type = c("go", "gene"), ont = "MF", organism = "human",
  measure = "Resnik", combine = "BMA")
```

Arguments

golist	A character vector, each component contains a character vector of GO terms or one Entrez Gene ID.
type	Input type of golist, 'go' for GO Terms, 'gene' for gene ID.
ont	Default is 'MF', could be one of 'MF', 'BP', or 'CC' subontologies.
organism	Default is 'human', could be one of 'anopheles', 'arabidopsis', 'bovine', 'canine', 'chicken', 'chimp', 'coelicolor', 'ecolik12', 'ecsakai', 'fly', 'human', 'malaria', 'mouse', 'pig', 'rat', 'rhesus', 'worm', 'xenopus', 'yeast' or 'zebrafish'.
measure	Default is 'Resnik', could be one of 'Resnik', 'Lin', 'Rel', 'Jiang' or 'Wang'.
combine	Default is 'BMA', could be one of 'max', 'average', 'rcmax' or 'BMA' for combining semantic similarity scores of multiple GO terms associated with protein.

Details

This function calculates protein sequence similarity based on Gene Ontology (GO) similarity.

Value

A n x n similarity matrix.

Author(s)

Nan Xiao <http://r2s.name>

See Also

See twoGOSim for calculating the GO semantic similarity between two groups of GO terms or two Entrez gene IDs. See parSeqSim for paralleled protein similarity calculation based on Smith-Waterman local alignment.

Examples

```
# Be careful when testing this since it involves GO similarity computation
# and might produce unpredictable results in some environments

require(GOSemSim)
require(org.Hs.eg.db)

# by GO Terms
go1 = c('GO:0005215', 'GO:0005488', 'GO:0005515', 'GO:0005625', 'GO:0005802', 'GO:0005905')  # AP4B1
go2 = c('GO:0005515', 'GO:0005634', 'GO:0005681', 'GO:0008380', 'GO:0031202')  # BCAS2
go3 = c('GO:0003735', 'GO:0005622', 'GO:0005840', 'GO:0006412')  # PDE4DIP
glist = list(go1, go2, go3)
gsimmat1 = parGOSim(glist, type = 'go', ont = 'CC')
print(gsimmat1)

# by Entrez gene id
genelist = list(c('150', '151', '152', '1814', '1815', '1816'))
gsimmat2 = parGOSim(genelist, type = 'gene')
print(gsimmat2)
```

---

parSeqSim *Parallellized Protein Sequence Similarity Calculation based on Sequence Alignment*

Description

Parallellized Protein Sequence Similarity Calculation based on Sequence Alignment

Usage

```
parSeqSim(protlist, cores = 2, type = "local", submat = "BLOSUM62")
```

Arguments

protlist	A length n list containing n protein sequences, each component of the list is a character string, storing one protein sequence. Unknown sequences should be represented as ''.
cores	Integer. The number of CPU cores to use for parallel execution, default is 2. Users could use the detectCores() function in the parallel package to see how many cores they could use.
type	Type of alignment, default is 'local', could be 'global' or 'local', where 'global' represents Needleman-Wunsch global alignment; 'local' represents Smith-Waterman local alignment.
submat	Substitution matrix, default is 'BLOSUM62', could be one of 'BLOSUM45', 'BLOSUM50', 'BLOSUM62', 'BLOSUM80', 'BLOSUM100', 'PAM30', 'PAM40', 'PAM70', 'PAM120', 'PAM250'.

Details

This function implemented the parallellized version for calculating protein sequence similarity based on sequence alignment.

Value

A n x n similarity matrix.

Author(s)

Nan Xiao <http://r2s.name>

See Also

See twoSeqSim for protein sequence alignment for two protein sequences. See parGOSim for protein similarity calculation based on Gene Ontology (GO) semantic similarity.

Examples

```
# Be careful when testing this since it involves parallelisation
# and might produce unpredictable results in some environments

require(Biostrings)
require(foreach)
require(doParallel)

s1 = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
s2 = readFASTA(system.file('protseq/P08218.fasta', package = 'protr'))[[1]]
s3 = readFASTA(system.file('protseq/P10323.fasta', package = 'protr'))[[1]]
```

```
s4 = readFASTA(system.file('protseq/P20160.fasta', package = 'protr'))[[1]]
s5 = readFASTA(system.file('protseq/Q9NZP8.fasta', package = 'protr'))[[1]]
plist = list(s1, s2, s3, s4, s5)
psimmat = parSeqSim(plist, cores = 2, type = 'local', submat = 'BLOSUM62')
print(psimmat)
```

`protcheck` *Check if the protein sequence's amino acid types are in the 20 default types*

Description

Check if the protein sequence's amino acid types are in the 20 default types

Usage

```
protcheck(x)
```

Arguments

`x`	A character vector, as the input protein sequence.

Details

This function checks if the protein sequence's amino acid types are in the 20.

Value

Logical. `TRUE` if all of the amino acid types of the sequence are within the 20 default types.

Author(s)

Nan Xiao <http://r2s.name>

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
protcheck(x) # TRUE
protcheck(paste(x, 'Z', sep = '')) # FALSE
```

protseg *Protein Sequence Segmentation*

Description

Protein Sequence Segmentation

Usage

```
protseg(x, aa = c("A", "R", "N", "D", "C", "E", "Q", "G", "H", "I", "L", "K",
  "M", "F", "P", "S", "T", "W", "Y", "V"), k = 7)
```

Arguments

x	A character vector, as the input protein sequence.
aa	A character, the amino acid type. One of 'A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V'.
k	A positive integer, specifys the window size (half of the window), default is 7.

Details

This function extracts the segmentations from the protein sequence.

Value

A named list, each component contains one of the segmentations (a character string), names of the list components are the positions of the specified amino acid in the sequence.

Author(s)

Nan Xiao <http://r2s.name>

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
protseg(x, aa = 'R', k = 5)
```

readFASTA *Read Protein Sequences in FASTA Format*

Description

Read Protein Sequences in FASTA Format

Usage

```
readFASTA(file = system.file("protseq/P00750.fasta", package = "protr"),
  legacy.mode = TRUE, seqonly = FALSE)
```

Arguments

file	The name of the file which the sequences in fasta format are to be read from. If it does not contain an absolute or relative path, the file name is relative to the current working directory, getwd. The default here is to read the P00750.fasta file which is present in the protseq directory of the protr package.
legacy.mode	If set to TRUE, lines starting with a semicolon ';' are ignored. Default value is TRUE.
seqonly	If set to TRUE, only sequences as returned without attempt to modify them or to get their names and annotations (execution time is divided approximately by a factor 3). Default value is FALSE.

Details

This function reads protein sequences in FASTA format.

Value

The result character vector

The three returned argument are just different forms of the same output. If one is interested in a Mahalanobis metric over the original data space, the first argument is all she/he needs. If a transformation into another space (where one can use the Euclidean metric) is preferred, the second returned argument is sufficient. Using A and B is equivalent in the following sense.

Note

Note that any different sets of instances (chunklets), e.g. 1, 3, 7 and 4, 6, might belong to the same class and might belong to different classes.

Author(s)

Nan Xiao <http://r2s.name>

References

Pearson, W.R. and Lipman, D.J. (1988) Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, **85**: 2444-2448

See Also

See `getUniProt` for retrieving protein sequences from uniprot.org

Examples

```
P00750 = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))
```

---

readPDB *Read Protein Sequences in PDB Format*

Description

Read Protein Sequences in PDB Format

Usage

```
readPDB(file = system.file("protseq/4HHB.pdb", package = "Rcpi"))
```

Arguments

`file`	The name of the file which the sequences in PDB format are to be read from. If it does not contain an absolute or relative path, the file name is relative to the current working directory, `getwd`. The default here is to read the `4HHB.PDB` file which is present in the `protseq` directory of the protr package.

Details

This function reads protein sequences in PDB (Protein Data Bank) format, and return the amino acid sequences represented by single-letter code.

Value

A character vector, representing the amino acid sequence of the single-letter code.

Author(s)

Nan Xiao <http://r2s.name>

References

Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description, Version 3.30. Accessed 2013-06-26. ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v33_Letter.pdf

See Also

See readFASTA for reading protein sequences in FASTA format.

Examples

```
Seq4HHB = readPDB(system.file('protseq/4HHB.pdb', package = 'protr'))
```

---

twoGOSim *Protein Similarity Calculation based on Gene Ontology (GO) Similarity*

Description

Protein Similarity Calculation based on Gene Ontology (GO) Similarity

Usage

```
twoGOSim(id1, id2, type = c("go", "gene"), ont = "MF", organism = "human",
  measure = "Resnik", combine = "BMA")
```

Arguments

id1	A character vector. length > 1: each element is a GO term; length = 1: the Entrez Gene ID.
id2	A character vector. length > 1: each element is a GO term; length = 1: the Entrez Gene ID.
type	Input type of id1 and id2, 'go' for GO Terms, 'gene' for gene ID.
ont	Default is 'MF', could be one of 'MF', 'BP', or 'CC' subontologies.
organism	Default is 'human', could be one of 'anopheles', 'arabidopsis', 'bovine', 'canine', 'chicken', 'chimp', 'coelicolor', 'ecolik12', 'ecsakai', 'fly', 'human', 'malaria', 'mouse', 'pig', 'rat', 'rhesus', 'worm', 'xenopus', 'yeast' or 'zebrafish'.
measure	Default is 'Resnik', could be one of 'Resnik', 'Lin', 'Rel', 'Jiang' or 'Wang'.
combine	Default is 'BMA', could be one of 'max', 'average', 'rcmax' or 'BMA' for combining semantic similarity scores of multiple GO terms associated with protein.

Details

This function calculates the Gene Ontology (GO) similarity between two groups of GO terms or two Entrez gene IDs.

Value

A n x n matrix.

Author(s)

Nan Xiao <http://r2s.name>

See Also

See parGOSim for protein similarity calculation based on Gene Ontology (GO) semantic similarity. See parSeqSim for paralleled protein similarity calculation based on Smith-Waterman local alignment.

Examples

```
# Be careful when testing this since it involves GO similarity computation
# and might produce unpredictable results in some environments

require(GOSemSim)
require(org.Hs.eg.db)

# by GO terms
go1 = c("GO:0004022", "GO:0004024", "GO:0004023")
go2 = c("GO:0009055", "GO:0020037")
gsim1 = twoGOSim(go1, go2, type = 'go', ont = 'MF', measure = 'Wang')
print(gsim1)

# by Entrez gene id
gene1 = '241'
gene2 = '251'
gsim2 = twoGOSim(gene1, gene2, type = 'gene', ont = 'BP', measure = 'Lin')
print(gsim2)
```

---

twoSeqSim *Protein Sequence Alignment for Two Protein Sequences*

Description

Protein Sequence Alignment for Two Protein Sequences

Usage

```
twoSeqSim(seq1, seq2, type = "local", submat = "BLOSUM62")
```

Arguments

seq1	A character string, containing one protein sequence.
seq2	A character string, containing another protein sequence.
type	Type of alignment, default is 'local', could be 'global' or 'local', where 'global' represents Needleman-Wunsch global alignment; 'local' represents Smith-Waterman local alignment.

submat — Substitution matrix, default is 'BLOSUM62', could be one of 'BLOSUM45', 'BLOSUM50', 'BLOSUM62', 'BLOSUM80', 'BLOSUM100', 'PAM30', 'PAM40', 'PAM70', 'PAM120', 'PAM250'.

Details

This function implements the sequence alignment between two protein sequences.

Value

An Biostrings object containing the scores and other alignment information.

Author(s)

Nan Xiao <http://r2s.name>

See Also

See parSeqSim for paralleled pairwise protein similarity calculation based on sequence alignment. See twoGOSim for calculating the GO semantic similarity between two groups of GO terms or two Entrez gene IDs.

Examples

```
# Be careful when testing this since it involves sequence alignment
# and might produce unpredictable results in some environments

require(Biostrings)

s1 = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
s2 = readFASTA(system.file('protseq/P10323.fasta', package = 'protr'))[[1]]
seqalign = twoSeqSim(s1, s2)
summary(seqalign)
print(seqalign@score)
```

Index