Package: ptm (via r-universe)

November 14, 2026

Type Package

Title Analyses of Protein Post-Translational Modifications

Version 1.0.1

Description Contains utilities for the analysis of post-translational modifications (PTMs) in proteins, with particular emphasis on the sulfoxidation of methionine residues. Features include the ability to download, filter and analyze data from the sulfoxidation database 'MetOSite'. Utilities to search and characterize S-aromatic motifs in proteins are also provided. In addition, functions to analyze sequence environments around modifiable residues in proteins can be found. For instance, 'ptm' allows to search for amino acids either overrepresented or avoided around the modifiable residues from the proteins of interest. Functions tailored to test statistical hypothesis related to these differential sequence environments are also implemented. Further and detailed information regarding the methods in this package can be found in (Aledo (2020) <https://metositeptm.com>).

License GPL (>= 2)

URL https://bitbucket.org/jcaledo/ptm, https://github.com/jcaledo/Rptm, https://metositeptm.com

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Depends R (>= 4.0.0)

Imports bio3d (>= 2.3-4), curl, graphics, httr, jsonlite, stats

Suggests knitr, markdown, testthat

NeedsCompilation no

Author Juan Carlos Aledo [aut, cre]

Maintainer Juan Carlos Aledo <caledo@uma.es>

Config/pak/sysreqs libssl-dev

Repository https://cranhaven.r-universe.dev

Date/Publication 2026-11-14 03:02:00 UTC

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/ptm

RemoteSha 6d2c60e1c6090f23f125f8dd8a1e37190c73e84c

RemoteSubdir ptm

Contents

.get.exepath 2
.get.url 3
aa.at 3
aa.comp 4
get.seq 5
gracefully_fail 6
hmeto 7
is.at 8
meto.list 9
meto.scan 10
meto.search 11
pairwise.dist 12
saro.dist 13
saro.geometry 14
saro.motif 15
xprod 16

Index **17**

`.get.exepath` *Find Full Paths to Executables*

Description

Finds the path to an executable.

Usage

```
.get.exepath(prg)
```

Arguments

`prg`	name of the executable.

Value

Returns the absolute path.

.get.url *Get Web Resource*

Description

Gets a web resource.

Usage

```
.get.url(url, n_tries = 3)
```

Arguments

url	url to be reached.
n_tries	number of tries.

Value

Returns the response or an error message.

aa.at *Residue Found at the Requested Position*

Description

Returns the residue found at the requested position.

Usage

```
aa.at(at, target, uniprot = TRUE)
```

Arguments

at	the position in the primary structure of the protein.
target	a character string specifying the UniProt ID of the protein of interest or, alternatively, the sequence of that protein.
uniprot	logical, if TRUE the argument 'target' should be an ID.

Details

Please, note that when uniprot is set to FALSE, target can be the string returned by a suitable function, such as get.seq or other.

Value

Returns a single character representing the residue found at the indicated position in the indicated protein.

Author(s)

Juan Carlos Aledo

See Also

is.at(), aa.comp()

Examples

```
## Not run: aa.at(28, 'P01009')
```

`aa.comp` *Amino Acid Composition*

Description

Returns a table with the amino acid composition of the target protein.

Usage

```
aa.comp(target, uniprot = TRUE, reference = 'human', init = FALSE)
```

Arguments

`target`	a character string specifying the UniProt ID of the protein of interest or, alternatively, the sequence of that protein.
`uniprot`	logical, if TRUE the argument 'target' should be an ID.
`reference`	amino acid frequencies (in percent) of the proteinogenic amino acids to be used as reference. It should be either 'human', 'up' (composition of proteins in UniProt in 2019). Alternatively, the user can pass as argument any vector with 20 values to be used as reference.
`init`	logical, whether remove or not the first residue (initiation methionine) from the sequence.

Value

Returns a list where the first element is a dataframe with the observed and expected frequencies for each amino acid, the second element is the result of the Chi-squared test. In addition, a plot to reflect potential deviations from the reference standard composition is shown.

Author(s)

Juan Carlos Aledo

See Also

is.at(), renum.pdb(), renum.meto(), renum(), aa.at()

Examples

```
aa.comp('MPSSVSWGILLLAGLCCLVPVSLAEDPQGDAAQK', uniprot = FALSE)
```

---

get.seq *Import a Protein Sequence from a Database*

Description

Imports a protein sequence from a selected database.

Usage

```
get.seq(id, db = 'uniprot', as.string = TRUE)
```

Arguments

id	the identifier of the protein of interest.
db	a character string specifying the desired database; it must be one of 'uniprot' or 'metosite'.
as.string	logical, if TRUE the imported sequence will be returned as a character string.

Details

MetOSite uses the same type of protein ID than UniProt.

Value

Returns a protein sequence either as a character vector or a as a character string.

Examples

```
get.seq('P01009')
```

`gracefully_fail` *Check that Internet Resource Work Properly and Fail Gracefully When Not*

Description

Checks that internet resource works properly and fail gracefully when not.

Usage

```
gracefully_fail(call, timeout = 10,  ...)
```

Arguments

`call`	url of the resource.
`timeout`	set maximum request time in seconds.
`...`	further named parameters, such as query, headers, etc.

Details

To be used as an ancillary function.

Value

The response object or NULL when the server does not respond properly.

Author(s)

thefactmachine

References

https://gist.github.com/thefactmachine/18279b7796c0836d9188

Examples

```
gracefully_fail("http://httpbin.org/delay/2")
```

`hmeto` *Human MetO sites oxidized by hydrogen peroxide treatment.*

Description

A dataset containing data regarding human MetO sites oxidized by H2O2.

Usage

```
hmeto
```

Format

A data frame with 4472 rows and 15 variables:

prot_id UniProt ID of the oxidized protein

prot_name the protein's name

met_pos the position of the MetO site in the primary structure

met_vivo_vitro conditions under which the oxidation experiment was carried out

MetOsites array with all the sites oxidized in that protein

site_id primary key identifying the site

positive sequence environment of the MetO site

control sequence environment of a non oxidized Met from the same protein

IDP Intrinsically Disordered Proteins, 0: the protein is not found in DisProt; 1: the protein contains disordered regions; 2: the protein may contain disordered regions but the experimental evidences are ambiguous

IDR Intrinsically Disordered Region, TRUE: the MetO site belong to the IDR, FALSE: the MetO site doesn't belong to the IDR

abundance protein abundance, in ppm

N protein length, in number of residues

met number of methionine residues

fmet relative frequency of Met in that protein

prot_vivo_vitro whether the protein has been described to be oxidized in vivo, in vitro or under both conditions

Source

https://metosite.uma.es/

`is.at` *Check Residue a Fixed Position*

Description

Checks if a given amino acid is at a given position.

Usage

```
is.at(at, target, aa = 'M', uniprot = TRUE)
```

Arguments

`at`	the position in the primary structure of the protein.
`target`	a character string specifying the UniProt ID of the protein of interest or, alternatively, the sequence of that protein.
`aa`	the amino acid of interest.
`uniprot`	logical, if TRUE the argument 'target' should be an ID.

Details

Please, note that when uniprot is set to FALSE, target can be the string returned by a suitable function, such as get.seq or other.

Value

Returns a boolean. Either the residue is present at that position or not.

Author(s)

Juan Carlos Aledo

See Also

aa.at(), aa.comp()

Examples

```
## Not run: is.at(28, 'P01009', 'Q')
```

`meto.list` *List Proteins Found in MetOSite Matching a Keyword*

Description

Lists proteins found in MetOSite with names matching the keyword.

Usage

```
meto.list(keyword)
```

Arguments

`keyword`	a character string corresponding to the keyword

Value

This function returns a dataframe with the uniprot id, the protein name and the species, for those proteins present into MetOSite whose name contains the keyword.

Author(s)

Juan Carlos Aledo

References

Valverde et al. 2019. Bioinformatics 35:4849-4850 (PMID: 31197322)

See Also

meto.search(), meto.scan()

Examples

```
meto.list('inhibitor')
```

`meto.scan` *Scans a Protein in Search of MetO Sites*

Description

Scans a given protein in search of MetO sites.

Usage

```
meto.scan(up_id, report = 1)
```

Arguments

`up_id`	a character string corresponding to the UniProt ID.
`report`	it should be a natural number between 1 and 3.

Details

When the 'report' parameter has been set to 1, this function returns a brief report providing the position, the function category and literature references concerning the residues detected as MetO, if any. If we wish to obtain a more detailed report, the option should be: report = 2. Finally, If we want a detailed and printable report (saved in the current directory), we should set report = 3

Value

This function returns a report regarding the MetO sites found, if any, in the protein of interest.

Author(s)

Juan Carlos Aledo

References

Valverde et al. 2019. Bioinformatics 35:4849-4850 (PMID: 31197322)

See Also

meto.search(), meto.list()

Examples

```
meto.scan('P01009')
```

`meto.search` *Search for Specific MetO Sites*

Description

Searches for specific MetO sites filtering MetOSite according to the selected criteria.

Usage

```
meto.search(highthroughput.group = TRUE,
                    bodyguard.group = TRUE,
                    regulatory.group = TRUE,
                    gain.activity = 2, loss.activity = 2, gain.ppi = 2,
                    loss.ppi = 2, change.stability = 2, change.location = 2,
                    organism = -1, oxidant = -1)
```

Arguments

`highthroughput.group`
: logical, when FALSE the sites described in a high-throughput study (unknown effect) are filtered out.

`bodyguard.group`
: logical, when FALSE the sites postulated to function as ROS sink (because when oxidized no apparent effect can be detected) are filtered out.

`regulatory.group`
: logical, when FALSE the sites whose oxidation affect the properties of the protein (and therefore may be involved in regulation) are filtered out.

`gain.activity`
: introduce 1 or 0 to indicate whether the oxidation of the selected sites implies a gain of activity or not, respectively. If we do not wish to use this property to filter, introduce 2.

`loss.activity`
: introduce 1 or 0 to indicate whether or not the oxidation of the selected sites implies a loss of activity or not, respectively. If we do not wish to use this property to filter, introduce 2.

`gain.ppi`
: introduce 1 or 0 to indicate whether the oxidation of the selected sites implies a gain of protein-protein interaction or not, respectively. If we do not wish to use this property to filter, introduce 2.

`loss.ppi`
: introduce 1 or 0 to indicate whether or not the oxidation of the selected sites implies a loss of protein-protein interaction or not, respectively. If we do not wish to use this property to filter, introduce 2.

`change.stability`
: introduce 1 or 0 to indicate whether the oxidation of the selected sites leads to a change in the protein stability or not, respectively. If we do not wish to use this property to filter, introduce 2.

`change.location`
: introduce 1 or 0 to indicate whether or not the oxidation of the selected sites implies a change of localization or not, respectively. If we do not wish to use this property to filter, introduce 2.

`organism`
: a character string indicating the scientific name of the species of interest, or -1 if we do not wish to filter by species.

`oxidant`
: a character string indicating the oxidant, or -1 if we do not wish to filter by oxidants.

Details

Note that all the arguments of this function are optional. We only pass an argument to the function when we want to use that parameter to filter. Thus, meto.search() will return all the MetO sites found in the database MetOSite.

Value

This function returns a dataframe with a line per MetO site.

Author(s)

Juan Carlos Aledo

References

Valverde et al. 2019. Bioinformatics 35:4849-4850 (PMID: 31197322)

See Also

meto.scan(), meto.list()

Examples

```
meto.search(organism = 'Homo sapiens', oxidant = 'HClO')
```

---

pairwise.dist *Compute Euclidean Distances*

Description

Computes the pairwise distance matrix between two sets of points

Usage

```
pairwise.dist(a, b, squared = TRUE)
```

Arguments

a, b	matrices (NxD) and (MxD), respectively, where each row represents a D-dimensional point.
squared	return containing squared Euclidean distance

Value

Euclidean distance matrix (NxM). An attribute "squared" set to the value of param squared is provided.

Examples

```
pairwise.dist(matrix(1:9, ncol = 3), matrix(9:1, ncol = 3))
```

---

saro.dist *Compute Distances to the Closest Aromatic Residues*

Description

Computes distances to the closest aromatic residues.

Usage

```
saro.dist(pdb, threshold = 7, rawdata = FALSE)
```

Arguments

pdb	either the path to the PDB file of interest or the 4-letters identifier.
threshold	distance in ångströms, between the S atom and the aromatic ring centroid, used as threshold.
rawdata	logical to indicate whether we also want the raw distance matrix between delta S and aromatic ring centroids.

Details

For each methionyl residue this function computes the distances to the closest aromatic ring from Y, F and W. When that distance is equal or lower to the threshold, it will be computed as a S-aromatic motif.

Value

The function returns a dataframe with as many rows as methionyl residues are found in the protein. The distances in ångströms to the closest tyrosine, phenylalanine and triptophan are given in the columns, as well as the number of S-aromatic motifs detected with each of these amino acids. Also a raw distance matrix can be provided.

Author(s)

Juan Carlos Aledo

References

Reid, Lindley & Thornton, FEBS Lett. 1985, 190:209-213.

See Also

saro.motif(), saro.geometry()

Examples

```
## Not run: saro.dist('1CLL')
```

---

saro.geometry *Compute Geometric Parameters of S-Aromatic Motifs*

Description

Computes distances and angles of S-aromatic motifs.

Usage

```
saro.geometry(pdb, rA, chainA = 'A', rB, chainB = 'A')
```

Arguments

pdb	either the path to the PDB file of interest or the 4-letters identifier.
rA	numeric position of one of the two residues involved in the motif.
chainA	a character indicating the chain to which belong the first residue.
rB	numeric position of the second residue involved in the motif.
chainB	a character indicating the chain to which belong the second residue.

Details

The distance between the delta sulfur atom and the centroid of the aromatic ring is computed, as well as the angle between this vector and the one perpendicular to the plane containing the aromatic ring. Based on the distance (d) and the angle (theta) the user decide whether the two residues are considered to be S-bonded or not (usually when d < 7 and theta < 60º).

Value

The function returns a dataframe providing the coordinates of the sulfur atom and the centroid (centroids when the aromatic residue is tryptophan), as well as the distance (ångströms) and the angle (degrees) mentioned above.

Author(s)

Juan Carlos Aledo

References

Reid, Lindley & Thornton, FEBS Lett. 1985, 190, 209-213.

See Also

saro.motif(), saro.dist()

Examples

```
## Not run: saro.geometry('1CLL', rA = 141, rB = 145)
```

---

saro.motif *Search for S-Aromatic Motifs*

Description

Searches for S-aromatic motifs in proteins.

Usage

```
saro.motif(pdb, threshold = 7, onlySaro = TRUE)
```

Arguments

pdb	either the path to the PDB file of interest or the 4-letters identifier.
threshold	distance in ångströms, between the S atom and the aromatic ring centroid, used as threshold.
onlySaro	logical, if FALSE the output includes information about Met residues that are not involved in S-aromatic motifs.

Details

For each methionyl residue taking place in a S-aromatic motif, this function computes the aromatic residues involved, the distance between the delta sulfur and the aromatic ring's centroid, as well as the angle between the sulfur-aromatic vector and the normal vector of the plane containing the aromatic ring.

Value

The function returns a dataframe reporting the S-aromatic motifs found for the protein of interest.

Author(s)

Juan Carlos Aledo

References

Reid, Lindley & Thornton, FEBS Lett. 1985, 190, 209-213.

See Also

saro.dist(), saro.geometry()

Examples

```
## Not run: saro.motif('1CLL')
```

`xprod` *Compute Cross Product*

Description

Computes the cross product of two vectors in three-dimensional euclidean space.

Usage

```
xprod(...)
```

Arguments

...	vectors involved in the cross product.

Details

For each methionyl residue taking place in a S-aromatic motif, this function computes the aromatic residue involved, the distance between the delta sulfur and the aromatic ring's centroid, as well as the angle between the sulfur-aromatic vector and the normal vector of the plane containing the aromatic ring.

Value

This function returns a vector that is orthogonal to the plane containing the two vector used as arguments.

Author(s)

Juan Carlos Aledo

Examples

```
xprod(c(1,1,1), c(1,2,1))
```

Index

∗ **datasets**
    hmeto, 7
∗ **internal.**
    .get.exepath, 2
    .get.url, 3
.get.exepath, 2
.get.url, 3

aa.at, 3
aa.comp, 4

get.seq, 5
gracefully_fail, 6

hmeto, 7

is.at, 8

meto.list, 9
meto.scan, 10
meto.search, 11

pairwise.dist, 12

saro.dist, 13
saro.geometry, 14
saro.motif, 15

xprod, 16