Package 'PTMods'

May 2, 2026

Type Package

Version 1.1.0

Title Managing Post-Translational Modifications in R

Description An interface to the community supported database for amino acid/protein modifications using mass spectrometry.

Depends R (>= 4.5.0), methods

Suggests xml2, testthat, knitr, BiocStyle, Biostrings

Enhances PSMatch

License GPL-3

VignetteBuilder knitr

biocViews Proteomics, MassSpectrometry

BugReports https://github.com/RforMassSpectrometry/PTMods/issues

URL https://github.com/RforMassSpectrometry/PTMods

RoxygenNote 7.3.3

Encoding UTF-8

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

git_branch devel

git_last_commit 08ea1e1

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-01

Author Laurent Gatto [aut] (ORCID: <https://orcid.org/0000-0002-1520-2268>), Sebastian Gibb [aut] (ORCID: <https://orcid.org/0000-0001-7406-4443>), Guillaume Deflandre [cre] (ORCID: <https://orcid.org/0009-0008-1257-2416>)

Maintainer Guillaume Deflandre <guillaume.deflandre@uclouvain.be>

Contents

`PTMods-package` *Managing amino acid modifications for mass spectrometry in R.*

Description

The 'PTMods' package focuses on handling post-translational modifications (PTMs). It distributes PTMs from the Unimod database and allows to convert PTM annotations between multiple formats.

Author(s)

Maintainer: Guillaume Deflandre <guillaume.deflandre@uclouvain.be> (ORCID)

Authors:

- Laurent Gatto <laurent.gatto@uclouvain.be> (ORCID)
- Sebastian Gibb <mail@sebastiangibb.de> (ORCID)

References

https://github.com/RforMassSpectrometry/PTMods/

See Also

Useful links:

- https://github.com/RforMassSpectrometry/PTMods
- Report bugs at https://github.com/RforMassSpectrometry/PTMods/issues

`addFixedModifications` *Add fixed modifications*

Description

Applies fixed modifications to peptide sequences.

The annotation style of modifications already present in 'sequences' is preserved as-is. The annotation style of newly applied modifications matches the style supplied in 'fixedModifications': numeric values produce deltaMass notation (e.g. '[+79.966331]'), while character values are used verbatim (e.g. '[Phospho]' or '[UNIMOD:21]').

Usage

```
addFixedModifications(sequences, fixedModifications = c(C = 57.021464))
```

Arguments

`sequences`	Character vector. Peptide sequences that may have modifications or not.
`fixedModifications`	Named 'numeric' or 'character'. If a 'character' is given, values must be in UniMod name or UniMod ID format (e.g. '"Phospho"', '"UNIMOD:21"'). The annotation style of the values is preserved in the output. Specifies which fixed modifications are applied to which amino acids. By default, carbamidomethylation is applied (+57.021464 on cysteines). Supplying multiple modifications for the same amino acid is not supported; call the function consecutively instead:

```
"ATK" |>
  addFixedModifications(c(T = "+57.024")) |>
  addFixedModifications(c(T = "Phospho"))
```

Value

A named list of character strings with all fixed modifications applied to the corresponding amino acids, one element per input sequence.

Author(s)

Guillaume Deflandre <guillaume.deflandre@uclouvain.be>

Examples

```
addFixedModifications("ATCK")
addFixedModifications("ATCK", fixedModifications = c(Nterm = 304, C = 57.02146))
addFixedModifications("ATSK", fixedModifications = c(A = -42, S = 79.966331))
addFixedModifications("ATK", fixedModifications = c(T = "Phospho", A = 42))
addFixedModifications("A[+42]TK", fixedModifications = c(T = "Phospho"))
addFixedModifications("AT[+79.966]AK", fixedModifications = c(A = 42))
addFixedModifications("ATA[+79.966]K", fixedModifications = c(A = 42))

## Apply two modifications to the same residue consecutively
"ATK" |>
    addFixedModifications(c(T = "+57.024")) |>
    addFixedModifications(c(T = "Phospho"))
```

`addModifications` *Add fixed and variable modifications to peptide sequences*

Description

A convenience wrapper around [addFixedModifications()] and [addVariableModifications()] that applies both fixed and variable modifications in a single call. Fixed modifications are applied first, followed by the enumeration of all possible combinations of variable modifications. Optionally, the annotation style of the output can be converted via [convertAnnotation()].

Usage

```
addModifications(
  sequences,
  fixedModifications = NULL,
  variableModifications = NULL,
  convertToStyle = NULL,
  ...
)
```

Arguments

`sequences` 'character'. Peptide sequences with or without existing modification annotations. Modifications must be enclosed in square brackets (e.g. '"AT[+79.966]K"', '"[+304]-ATK"').

`fixedModifications` Named 'numeric' or 'character', or 'NULL'. Fixed modifications to be applied unconditionally to the specified amino acids or termini. Names correspond to single-letter amino acid codes, '"Nterm"', or '"Cterm"'. If 'character', values must be in UniMod name or UniMod ID format (e.g. '"Phospho"', '"UNIMOD:21"'). See [addFixedModifications()] for full details. Set to 'NULL' to skip.

`variableModifications` Named 'numeric' or 'character', or 'NULL'. Variable modifications to enumerate over the specified amino acids. Names correspond to single-letter amino acid codes. If 'character', values must be in UniMod name or UniMod ID format. See [addVariableModifications()] for full details. Set to 'NULL' to skip.

`convertToStyle` 'character(1)'. Controls the annotation format of modifications in the returned sequences. One of:

'NULL' (default) Same as input modification style.

'"deltaMass"' Delta mass notation, e.g. '[+79.966]'.

'"unimodId"' UniMod ID notation, e.g. '[UNIMOD:21]'.

'"name"' UniMod name notation, e.g. '[Phospho]'.

Conversion is performed by [convertAnnotation()].

`...` Additional arguments passed to the underlying modification functions. Currently supported:

'maxMods' Forwarded to [addVariableModifications()]. A 'numeric(1)' specifying the maximum number of variable modifications that may be applied simultaneously. Defaults to 'Inf'.

Value

A ‘character‘ vector of peptidoforms. The length is greater than or equal to ‘length(sequences)‘, as each input sequence may expand into multiple peptidoforms corresponding to all combinations of variable modifications.

Author(s)

Guillaume Deflandre <guillaume.deflandre@uclouvain.be>

See Also

* addFixedModifications for applying fixed modifications.

* addVariableModifications for enumerating variable modification combinations.

* convertAnnotation for converting between modification annotation styles.

Examples

```
## Fixed modifications only
addModifications(
    c("ATCK", "PEPCK"),
    fixedModifications = c(C = 57.02146)
)

## Variable modifications only
addModifications(
    c("ATSK", "PEPTSK"),
    variableModifications = c(S = 79.966331, T = 79.966331),
    maxMods = 2
)

## Both fixed and variable modifications
addModifications(
    "ATCSK",
    fixedModifications = c(C = 57.02146),
    variableModifications = c(S = 79.966331, T = 79.966331),
    maxMods = 1
)

## Return annotations in UniMod name style
addModifications(
    "ATSK",
    variableModifications = c(S = 79.966331, T = 79.966331),
    convertToStyle = "name"
)

## N-terminal fixed modification with variable modifications
addModifications(
    "ATSK",
    fixedModifications = c(Nterm = 304),
    variableModifications = c(S = 79.966331)
)
```

addVariableModifications *Add variable modifications*

Description

Generates all possible combinations of variable modifications for peptide sequences.

The annotation style of modifications already present in 'sequences' is preserved as-is. The annotation style of newly applied modifications matches the style supplied in 'variableModifications': numeric values produce deltaMass notation (e.g. '[+79.966331]'), while character values are used verbatim (e.g. '[Phospho]' or '[UNIMOD:21]').

Usage

```
addVariableModifications(
  sequences,
  variableModifications = NULL,
  maxMods = Inf
)
```

Arguments

sequences	Character vector. Peptide sequences that may have modifications or not.
variableModifications	Named 'numeric' or 'character'. If a 'character' is given, values must be in UniMod name or UniMod ID format (e.g. '"Phospho"', '"UNIMOD:21"'). The annotation style of the values is preserved in the output. Specifies which variable modifications are used on which amino acids. Supplying multiple modifications for the same amino acid is not supported; call the function consecutively instead:

```
seq |>
  addVariableModifications(c(T = "+57.024")) |>
  addVariableModifications(c(T = "Phospho"))
```

maxMods	Numeric. Indicates how many modifications can be applied at once.

Value

A named list of character vectors with all possible combinations of modifications, one element per input sequence.

Author(s)

Guillaume Deflandre <guillaume.deflandre@uclouvain.be>

Examples

```
addVariableModifications(
    "APGHKA",
    variableModifications = c(A = 4, K = 5, S = 8),
    maxMods = 2
)
```

```
addVariableModifications(
    "APGHKA",
    variableModifications = c(A = 4, K = 5, S = 8),
    maxMods = 3
)
addVariableModifications(
    c("ATK", "PAK"),
    variableModifications = c(T = "Phospho", A = "UNIMOD:1")
)
addVariableModifications(
    "A[+10]KT[-5]",
    variableModifications = c(A = -20, T = "Phospho")
)

## Apply two modifications to the same residue consecutively
"ATK" |>
    addVariableModifications(c(T = "+57.024")) |>
    addVariableModifications(c(T = "Phospho"))
```

`aminoacids` *Amminoacids data set.*

Description

'data.frame' of aminoacids from the unimod database.

Usage

```
data("aminoacids")
```

Format

A 'data.frame' with 11 columns (OneLetter, ThreeLetter, FullName, AvgMass, MonoMass, H, C, N, O S, Se) for the aminoacids. The H/C/N/O/S/Se columns contain the number of elements that build the aminoacid.

Details

It was created as follows:

"' PTMods:::.createDataSets() "'

Source

Taken from the unimod database: http://www.unimod.org/xml/unimod.xml.

Examples

```
data(aminoacids)
head(aminoacids)
```

combineModifications *Combines consecutive modifications on amino acids in peptide sequences.*

Description

Combines consecutive modifications on amino acids in peptide sequences.

Usage

```
combineModifications(sequences)
```

Arguments

sequences	'character'. A character vector of peptide sequences with modifications in any format accepted by 'convertAnnotation'. Multiple modifications per amino acid are allowed.

Value

A character vector of the same length as 'sequences' with consecutive modifications on each amino acid summed into a single modification. This summed modifications is written in delta mass format.

Author(s)

Guillaume Deflandre <guillaume.deflandre@uclouvain.be>

Examples

```
combineModifications("AT[+10][-5]K")
combineModifications("[+10]-[-5]-ATK")
combineModifications("AT[Phospho][UNIMOD:1]K")
combineModifications("EM[+15.9949][-10]EK[+42][-8]")
combineModifications(c("AT[+10][-5]K", "EM[+15.9949][-10]EK"))
```

convertAnnotation *Convert sequences from one annotation style to another*

Description

Converts modifications between different annotation formats for multiple sequences at once. See the details and examples sections for more information. The annotation styles are inferred from the 'modifications' dataframe (see '?modifications').

Usage

```
convertAnnotation(
  x,
  convertToStyle = c("deltaMass", "unimodId", "name"),
  massTolerance = 0.01,
  verbose = TRUE
)
```

Arguments

x	Character vector with peptide sequences in ProForma format
convertToStyle	Character string specifying target format. Options: "deltaMass", "unimodId", "name"
massTolerance	Numeric mass tolerance in Daltons for matching modifications (default: 0.01). Used when converting from deltaMass.
verbose	'logical(1)'. If 'FALSE', warnings about unrecognised modifications are silenced (default: 'TRUE').

Details

The function handles three main conversion scenarios:

- Name to deltaMass: "M[Oxidation]PEPTIDE" -> "M[+15.994915]PEPTIDE"
- Name to unimodId: "M[Oxidation]PEPTIDE" -> "M[UNIMOD:35]PEPTIDE"
- deltaMass to name: "M[+15.995]PEPTIDE" -> "M[Oxidation]PEPTIDE"
- deltaMass to unimodId: "M[+15.995]PEPTIDE" -> "M[UNIMOD:35]PEPTIDE"
- unimodId to name: "M[UNIMOD:35]PEPTIDE" -> "M[Oxidation]PEPTIDE"
- unimodId to deltaMass: "M[UNIMOD:35]PEPTIDE" -> "M[+15.994915]PEPTIDE"

Value

Character vector with the sequences in the target annotation format

Author(s)

Guillaume Deflandre <guillaume.deflandre@uclouvain.be>

Examples

```
# Convert sequence from name to delta mass
convertAnnotation("M[Oxidation]PEPTIDE", convertToStyle = "deltaMass")
# Result: "M[+15.994915]PEPTIDE"

# Name to Unimod ID
convertAnnotation("M[Oxidation]PEPTIDE", convertToStyle = "unimodId")
# Result: "M[UNIMOD:35]PEPTIDE"

# Delta mass to name
convertAnnotation("M[+15.995]PEPTIDE", convertToStyle = "name")
# Result: "M[Oxidation]PEPTIDE"

# Multiple modifications
convertAnnotation("M[Oxidation]EVNES[Phospho]PEK", convertToStyle = "deltaMass")
# Result: "M[+15.994915]EVNES[+79.966331]PEK"

# Convert multiple sequences from name to delta mass
sequences <- c("M[Oxidation]PEPTIDE", "EVNES[Phospho]PEK", "PEPTIDE")
convertAnnotation(sequences, convertToStyle = "deltaMass")
# Result: c("M[+15.994915]PEPTIDE", "EVNES[+79.966331]PEK", "PEPTIDE")

# Convert from delta mass to name
```

```
sequences <- c("M[+15.995]PEPTIDE", "S[+79.966]EQUENCE")
convertAnnotation(sequences, convertToStyle = "name")
# Result: c("M[Oxidation]PEPTIDE", "S[Phospho]EQUENCE")

# Convert to Unimod IDs
sequences <- c("M[Oxidation]PEPTIDE", "C[Carbamidomethyl]PEPTIDE")
convertAnnotation(sequences, convertToStyle = "unimodId")
# Result: c("M[UNIMOD:35]PEPTIDE", "C[UNIMOD:4]PEPTIDE")
```

`elements` *Elements data set.*

Description

'data.frame' of chemical elements from the unimod database.

Usage

```
data("elements")
```

Format

A 'data.frame' with 4 columns (Name, FullName, AvgMass, MonoMass) for the chemical elements.

Details

It was created as follows:

``` PTMods:::.createDataSets() ```

Source

Taken from the unimod database: http://www.unimod.org/xml/unimod.xml.

Examples

```
data(elements)
head(elements)
```

`getCanonicalSequence` *Get canonical sequence from any peptidoform*

Description

Remove any modifications annotation on a sequence. Annotation from any type (so long as they are specified within brackets "[..]") are removed.

Usage

```
getCanonicalSequence(x)
```
```

Arguments

x	'character', ProForma sequence.

Value

'character', a 'character' vector with sequences cleaned of all modifications.

Examples

```
getCanonicalSequence(
    c(
        "EM[-15.9949]EVEES[+79.9663]PEK",
        "EK[Acetyl]EVEES[UNIMOD:21]PEK"
    )
)
```

---

getModificationsCounts

Extract modification counts from one or more peptide sequences

---

Description

Given a character vector of peptide sequences containing modifications in any annotation style (deltaMass, UniMod ID, or name), returns a named integer vector with the count of each unique modification. Regardless of the input annotation style, all modifications are converted to their UniMod name in the output (e.g. '[+79.966]' and '[UNIMOD:21]' both appear as 'Phospho').

Usage

```
getModificationsCounts(sequences)
```

Arguments

sequences	A 'character()' vector of peptide sequences in ProForma format.

Value

A named integer vector where names are UniMod modification names and values are their occurrence counts across all input sequences. Returns an empty integer vector if no modifications are found.

Author(s)

Guillaume Deflandre <guillaume.deflandre@uclouvain.be>

Examples

```
seqs <- c(
    "[+304]-AT[Phospho]K",
    "AC[Carbamidomethyl]T[+79.966]S[Phospho]K",
    "PEPTIDE"
)
getModificationsCounts(seqs)
```

`modifications` *Modifications data set.*

Description

'data.frame' of modifications from the unimod database.

Usage

```
data("modifications")
```

Format

A 'data.frame' with 15 columns (Id (created by Name:(Position-)Site(:NeutralLoss) because unimod id is not unique), UnimodId, Name, Description, AvgMass, MonoMass, Site, Position, Classification, SpecGroup, NeutralLoss, LastModification, Approved, Hidden) for the modifications.

Details

It was created as follows:

"' PTMods:::.createDataSets() "'

Source

Taken from the unimod database: http://www.unimod.org/xml/unimod.xml.

Examples

```
data(modifications)
head(modifications)
```

Index