Package ‘MetaboCoreUtils’

April 12, 2022

Title Core Utils for Metabolomics Data

Version 1.2.0

Description MetaboCoreUtils defines metabolomics-related core functionality provided as low-level functions to allow a data structure-independent usage across various R packages. This includes functions to calculate between ion (adduct) and compound mass-to-charge ratios and masses or functions to work with chemical formulas. The package provides also a set of adduct definitions and information on some commercially available internal standard mixes commonly used in MS experiments.

Depends R (>= 4.0)

Imports stringr, utils, MsCoreUtils

Suggests BiocStyle, testthat, knitr, rmarkdown

License Artistic-2.0

LazyData no

VignetteBuilder knitr

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

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

biocViews Infrastructure, Metabolomics, MassSpectrometry

Roxygen list(markdown=TRUE)

RoxygenNote 7.1.1

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

git_branch RELEASE_3_14

git_last_commit cec6b89

git_last_commit_date 2021-11-03

Date/Publication 2022-04-12

Author Johannes Rainer [aut, cre] (<https://orcid.org/0000-0002-6977-7147>), Michael Witting [aut] (<https://orcid.org/0000-0002-1462-4426>), Andrea Vicini [aut]

Maintainer Johannes Rainer <Johannes.Rainer@eurac.edu>

R topics documented:

addElements 2
adductNames 3
containsElements 3
correctRindex 4
countElements 5
indexRtime 6
internalStandardMixNames 7
internalStandards 7
isotopicSubstitutionMatrix 8
isotopologues 10
mass2mz 11
mz2mass 13
pasteElements 14
standardizeFormula 15
subtractElements 15

Index **17**

`addElements` *Combine chemical formulae*

Description

`addElements` Add one chemical formula to another.

Usage

```
addElements(x, y = NA_character_)
```

Arguments

x	`character` Vector with 1 or more chemical formulae to be added
y	`character` Vector with 1 or more chemical formulae to be added

Value

`character` Resulting formula

Author(s)

Michael Witting

Examples

```
addElements("C6H12O6", "Na")

addElements("C6H12O6", c("Na", "H2O"))
```

adductNames — *Retrieve names of supported adducts*

Description

`adductNames` returns all supported adduct definitions that can be used by `mass2mz()` and `mz2mass()`.

`adducts` returns a `data.frame` with the adduct definitions.

Usage

```
adductNames(polarity = c("positive", "negative"))

adducts(polarity = c("positive", "negative"))
```

Arguments

`polarity`	`character(1)` defining the ion mode, either `"positive"` or `"negative"`.

Value

for `adductNames`: `character` vector with all valid adduct names for the selected ion mode. For `adducts`: `data.frame` with the adduct definitions.

Author(s)

Michael Witting, Johannes Rainer

Examples

```
## retrieve names of adduct names in positive ion mode
adductNames(polarity = "positive")

## retrieve names of adduct names in negative ion mode
adductNames(polarity = "negative")
```

containsElements — *Check if one formula is contained in another*

Description

`containsElements` checks if one sum formula is contained in another.

Usage

```
containsElements(x, y)
```

Arguments

x	`character` Single string with a chemical formula
y	`character` Single string with a chemical formula that shall be contained in `x`

Value

`logical` TRUE if `y` is contained in `x`

Author(s)

Michael Witting

Examples

```
containsElements("C6H12O6", "H2O")
containsElements("C6H12O6", "NH3")
```

`correctRindex` *2-point correction of RIs*

Description

`correctRindex` performs correction of retention indices (RIs) based on reference substances. Even after conversion of RTs to RIs slight deviations might exist. These deviations can be further normalized, if they are linear, by using two metabolites for which the RIs are known (e.g. internal standards).

Usage

```
correctRindex(x, y)
```

Arguments

x	`numeric` vector with retention indices, calculated by `indexRtime`
y	`data.frame` containing two columns. The first is expected to contain the measured RIs of the reference substances and the second the reference RIs.

Value

`numeric` vector of same length than `x` with corrected retention indices. Values are floating point decimals. If integer values shall be used conversion has to be performed manually.

Author(s)

Michael Witting

Examples

```
ref <- data.frame(rindex = c(110, 210),
refindex = c(100, 200))
rindex <- c(110, 210)
correctRindex(rindex, ref)
```

---

countElements *Count elements in a chemical formula*

Description

`countElements` parses a string representing a chemical formula into a named vector of element counts.

Usage

```
countElements(x)
```

Arguments

x	`character(1)` representing a chemical formula.

Value

`integer` with the element counts (names being elements).

Author(s)

Michael Witting

See Also

`pasteElements()`

Examples

```
countElements("C6H12O6")
countElements("C11H12N2O2")
```

indexRtime *Convert retention times to retention indices*

Description

indexRtime uses a list of known substances to convert retention times (RTs) to retention indices (RIs). By this retention information is normalized for differences in experimental settings, such as gradient delay volume, dead volume or flow rate. By default linear interpolation is performed, other ways of calculation can supplied as function.

Usage

```
indexRtime(x, y, FUN = rtiLinear, ...)
```

Arguments

x	numeric vector with retention times
y	data.framedata.frame containing two columns, where the first holds the retention times of the indexing substances and the second the actual index value
FUN	function function defining how the conversion is performed, default is linear interpolation
...	additional parameter used by FUN

Value

numeric vector of same length as x with retention indices. Values floating point decimals. If integer values shall be used conversion has to be performed manually

Author(s)

Michael Witting

Examples

```
rti <- data.frame(rtime = c(1,2,3),
rindex = c(100,200,300))
rtime <- c(1.5, 2.5)
indexRtime(rtime, rti)
```

`internalStandardMixNames` *Get names of internal standard mixes provided by the package*

Description

`internalStandardMixNames` returns available names of internal standard mixes provided by the `MetaboCoreUtils` package.

Usage

```
internalStandardMixNames()
```

Value

`character` names of available IS mixes

Author(s)

Michael Witting

Examples

```
internalStandardMixNames()
```

`internalStandards` *Get definitions for internal standards*

Description

`internalStandards` returns a table with metabolite standards available in commercial internal standard mixes. The returned data frame contains the following columns:

- `"name"`: the name of the standard
- `"formula_salt"`: chemical formula of the salt that was used to produce the standard mix
- `"formula_metabolite"`: chemical formula of the metabolite in free form
- `"smiles_salt"`: SMILES of the salt that was used to produced the standard mix
- `"smiles_metabolite"`: SMILES of the metabolite in free form
- `"mol_weight_salt"`: molecular (average) weight of the salt (can be used for calculation of molar concentration, etc.)
- `"exact_mass_metabolite"`: exact mass of free metabolites
- `"conc"`: concentration of the metabolite in ug/mL (of salt form)
- `"mix"`: name of internal standard mix

Usage

```
internalStandards(mix = "QReSS")
```

Arguments

`mix`	`character(1)` Name of the internal standard mix that shall be returned. One of `internalStandardMixNames()`.

Value

`data.frame` data on internal standards

Author(s)

Michael Witting

See Also

`internalStandardMixNames()` for provided internal standard mixes.

Examples

```
internalStandards(mix = "QReSS")
internalStandards(mix = "UltimateSplashOne")
```

---

isotopicSubstitutionMatrix — *Definitions of isotopic substitutions*

Description

In order to identify potential isotopologues based on only m/z and intensity values with the `isotopologues()` function, sets of pre-calculated parameters are required. This function returns such parameter sets estimated on different sources/databases. The nomenclature used to describe isotopes follows the following convention: the number of neutrons is provided in `[` as a prefix to the element and the number of atoms of the element as suffix. [13]C2[37]Cl3 describes thus an isotopic substitution containing 2 [13]C isotopes and 3 [37]Cl isotopes.

Each row in the returned `data.frame` characterizes an isotopic substitution (which can involve isotopes of several elements or different isotopes of the same element). The provided isotopic substitutions are in general the most frequently observed substitutions in the database (e.g. HMDB) on which they were defined. Parameters (columns) defined for each isotopic substitution are:

- `"degree"`: the *degree* of the isotopic substitution. Isotopic substitutions with a single element (such as [15]N1 or [13]C1) are of degree 1 while isotopic substitutions with more isotopes are of a higher degree ([37]Cl5 and [34]S1[37]Cl4 are e.g. both of degree 5).
- `"minmass"`: the minimal mass of a compound for which the isotopic substitution was found. Peaks with a mass lower than this will most likely not have the respective isotopic substitution.

- "md": the mass difference between the monoisotopic peak and a peak of an isotopologue characterized by the respective isotopic substitution.
- "min_slope": used to calculate the lower expected bound for the ratio between the probabilities of isotopologues associated to a given substitution and the corresponding monoisotopic isotopologues. If a linear relationship between the number of each element in the substitution and the monoisotopic mass of compounds having those elements can be assumed then the previously mentioned ratio has a polynomial trend with degree equal to the *degree* of the isotopic substitution. The ratios for each compound and for a given substitution can be transformed by taking the root corresponding to *degree* of the substitution. In that way a linear trend in the monoisotopic mass can be obtained for the ratios. "min_slope" represent the slope of a lower bound line for this trend. In other words, for a given substitution we expect the ratio between the intensity of an isotopologue of a given compound corresponding to that substitution and the intensity of the monoisotopic isotopologue to be >= (monoisotopic mass * min_slope)^degree.
- "max_slope": used to calculate the expected upper intensity ratio bound.

Usage

```
isotopicSubstitutionMatrix(source = c("HMDB"))
```

Arguments

source	character(1) defining the set of predefined parameters and isotopologue definitions to return.

Value

data.frame with parameters to detect the defined isotopic substitutions

Available pre-calculated substitution matrices

- source = "HMDB": most common isotopic substitutions and parameters for these have been calculated for all compounds from the Human Metabolome Database (HMDB, July 2021). Note that the substitutions were calculated on the **neutral masses** (i.e. the chemical formulas of the compounds, not considering any adducts).

Author(s)

Andrea Vicini

Examples

```
## Get the substitution matrix calculated on HMDB
isotopicSubstitutionMatrix("HMDB")
```

isotopologues *Identfying isotopologue peaks in MS data*

Description

Given a spectrum (i.e. a peak matrix with m/z and intensity values) the function identifies groups of potential isotopologue peaks based on pre-defined mass differences and intensity (probability) ratios that need to be passed to the function with the `substDefinition` parameter. Each isotopic substitution in a compound determines a certain isotopologue and it is associated with a certain mass difference of that with respect to the monoisotopic isotopologue. Also each substitution in a compound is linked to a certain ratio between the intensities of the peaks of the corresponding isotopologue and the monoisotopic one. This ratio isn't the same for isotopologues corresponding to the same isotopic substitution but to different compounds. Through the `substDefinition` parameter we provide upper and lower values to compute bounds for each isotopic substitution dependent on the peak's mass.

Usage

```
isotopologues(
  x,
  substDefinition = isotopicSubstitutionMatrix(),
  tolerance = 0,
  ppm = 20,
  seedMz = numeric(),
  charge = 1
)
```

Arguments

`x`	`matrix` with spectrum data (columns `mz` and `intensity`).
`substDefinition`	`matrix` or `data.frame` with definition of isotopic substitutions (columns `"name"`, `"degree"`, `"md"`, `"min_slope"`, `"max_slope"`). The rows in this matrix have to be ordered by column `md` in increasing order. See `isotopicSubstitutionMatrix()` for more information on the format and content.
`tolerance`	`numeric(1)` representing the absolute tolerance for the relaxed matching of m/z values of peaks. See `MsCoreUtils::closest()` for details.
`ppm`	`numeric(1)` representing a relative, value-specific parts-per-million (PPM) tolerance for the relaxed matching of m/z values of peaks. See `MsCoreUtils::closest()` for details.
`seedMz`	`numeric` vector of **ordered** m/z values. If provided, the function checks if there are peaks in `x` which m/z match them. If any, it looks for groups where the first peak is one of the matched ones.
`charge`	`numeric(1)` representing the expected charge of the ionized compounds.

Details

The function iterates over the peaks (rows) in `x`. For each peak (which is assumed to be the monoisotopic peak) it searches other peaks in `x` with a difference in mass matching (given `ppm` and `tolerance`) any of the pre-defined mass differences in `substDefinitions` (column `"md"`). The mass is obtained by multiplying the m/z of the peaks for the `charge` expected for the ionized compounds.

For matching peaks, the function next evaluates whether the intensity is within the expected (pre-defined) intensity range. Using `"min_slope"` and `"max_slope"` for the respective potentially matching isotopic substitution in `substDefinition`, the function estimates a (mass dependent) lower and upper intensity ratio limit based on the peak's mass.

When some peaks are grouped together their indexes are excluded from the set of indexes that are searched for further groups (i.e. peaks already assigned to an isotopologue group are not considered/tested again thus each peak can only be part of one isotopologue group).

Value

`list` of `integer` vectors. Each `integer` vector contains the indixes of the rows in `x` with potential isotopologues of the same compound.

Author(s)

Andrea Vicini

Examples

```
## Read theoretical isotope pattern (high resolution) from example file
x <- read.table(system.file("exampleSpectra",
    "serine-alpha-lactose-caffeine.txt", package = "MetaboCoreUtils"),
    header = TRUE)
x <- x[order(x$mz), ]
plot(x$mz, x$intensity, type = "h")

isos <- isotopologues(x, ppm = 5)
isos

## highlight them in the plot
for (i in seq_along(isos)) {
    z <- isos[[i]]
    points(x$mz[z], x$intensity[z], col = i + 1)
}
```

`mass2mz` *Calculate mass-to-charge ratio*

Description

`mass2mz` calculates the m/z value from a neutral mass and an adduct definition.

Custom adduct definitions can be passed to the `adduct` parameter in form of a `data.frame`. This `data.frame` is expected to have columns `"mass_add"` and `"mass_multi"` defining the *additive* and *multiplicative* part of the calculation. See `adducts()` for examples.

Usage

```
mass2mz(x, adduct = "[M+H]+")
```

Arguments

`x`	`numeric` neutral mass for which the adduct m/z shall be calculated.
`adduct`	either a `character` specifying the name(s) of the adduct(s) for which the m/z should be calculated or a `data.frame` with the adduct definition. See `adductNames()` for supported adduct names and the description for more information on the expected format if a `data.frame` is provided.

Value

numeric `matrix` with same number of rows than elements in `x` and number of columns being equal to the length of `adduct` (adduct names are used as column names). Each column thus represents the m/z of `x` for each defined `adduct`.

Author(s)

Michael Witting, Johannes Rainer

See Also

`mz2mass()` for the reverse calculation, `adductNames()` for supported adduct definitions.

Examples

```
exact_mass <- c(100, 200, 250)
adduct <- "[M+H]+"

## Calculate m/z of [M+H]+ adduct from neutral mass
mass2mz(exact_mass, adduct)

exact_mass <- 100
adduct <- "[M+Na]+"

## Calculate m/z of [M+Na]+ adduct from neutral mass
mass2mz(exact_mass, adduct)

## Calculate m/z of multiple adducts from neutral mass
mass2mz(exact_mass, adduct = adductNames())

## Provide a custom adduct definition.
```

```
adds <- data.frame(mass_add = c(1, 2, 3), mass_multi = c(1, 2, 0.5))
rownames(adds) <- c("a", "b", "c")
mass2mz(c(100, 200), adds)
```

---

mz2mass *Calculate neutral mass*

Description

mz2mass calculates the neutral mass from a given m/z value and adduct definition.

Custom adduct definitions can be passed to the adduct parameter in form of a data.frame. This data.frame is expected to have columns "mass_add" and "mass_multi" defining the *additive* and *multiplicative* part of the calculation. See adducts() for examples.

Usage

```
mz2mass(x, adduct = "[M+H]+")
```

Arguments

x	numeric m/z value for which the neutral mass shall be calculated.
adduct	either a character specifying the name(s) of the adduct(s) for which the m/z should be calculated or a data.frame with the adduct definition. See adductNames() for supported adduct names and the description for more information on the expected format if a data.frame is provided.

Value

numeric matrix with same number of rows than elements in x and number of columns being equal to the length of adduct (adduct names are used as column names. Each column thus represents the neutral mass of x for each defined adduct.

Author(s)

Michael Witting, Johannes Rainer

See Also

mass2mz() for the reverse calculation, adductNames() for supported adduct definitions.

Examples

```
ion_mass <- c(100, 200, 300)
adduct <- "[M+H]+"

## Calculate m/z of [M+H]+ adduct from neutral mass
mz2mass(ion_mass, adduct)
```

```
ion_mass <- 100
adduct <- "[M+Na]+"

## Calculate m/z of [M+Na]+ adduct from neutral mass
mz2mass(ion_mass, adduct)

## Provide a custom adduct definition.
adds <- data.frame(mass_add = c(1, 2, 3), mass_multi = c(1, 2, 0.5))
rownames(adds) <- c("a", "b", "c")
mz2mass(c(100, 200), adds)
```

`pasteElements` *Create chemical formula from a named vector*

Description

`pasteElements` creates a chemical formula from element counts (such as returned by `countElements()`).

Usage

```
pasteElements(x)
```

Arguments

x	`integer` with element counts, names being individual elements.

Value

`character(1)` with the chemical formula.

Author(s)

Michael Witting

See Also

`countElements()`

Examples

```
elements <- c("C" = 6, "H" = 12, "O" = 6)
pasteElements(elements)
```

`standardizeFormula` *Standardize a chemical formula*

Description

`standardizeFormula` standardizes a supplied chemical chem_formula according to the Hill notation system.

Usage

```
standardizeFormula(x)
```

Arguments

`x`	`character` Single string with the chemical formula to standardize.

Value

`character` Single string with the standardized chemical formula.

Author(s)

Michael Witting

See Also

`pasteElements()` `countElements()`

Examples

```
standardizeFormula("C6O6H12")
```

`subtractElements` *subtract two chemical formula*

Description

`subtractElements` subtracts one chemical formula from another.

Usage

```
subtractElements(x, y)
```

Arguments

x	character Single string with chemical formula
y	character Single or multiple strings with chemical formula that should be subtracted from x

Value

character Resulting formula

Author(s)

Michael Witting

Examples

```
subtractElements("C6H12O6", "H2O")

subtractElements("C6H12O6", "NH3")
```

Index

addElements, 2
adductNames, 3
adductNames(), *12, 13*
adducts *(*adductNames*)*, 3
adducts(), *12, 13*

containsElements, 3
correctRindex, 4
countElements, 5
countElements(), *14, 15*

indexRtime, 6
internalStandardMixNames, 7
internalStandardMixNames(), *8*
internalStandards, 7
isotopicSubstitutionMatrix, 8
isotopicSubstitutionMatrix(), *10*
isotopologues, 10
isotopologues(), *8*

mass2mz, 11
mass2mz(), *3*, *13*
MsCoreUtils::closest(), *10*
mz2mass, 13
mz2mass(), *3*, *12*

pasteElements, 14
pasteElements(), *5*, *15*

standardizeFormula, 15
subtractElements, 15