Package ‘MetaboCoreUtils’

January 26, 2024

Title Core Utils for Metabolomics Data
Version 1.10.0
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 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 7.2.3
git_url https://git.bioconductor.org/packages/MetaboCoreUtils
git_branch RELEASE_3_18
git_last_commit 4c181cc
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-01-26

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],
**addElements**

Liesa Salzer [ctb] (<https://orcid.org/0000-0003-0761-0656>),
Sebastian Gibb [aut] (<https://orcid.org/0000-0001-7406-4443>),
Michael Stravs [ctb] (<https://orcid.org/0000-0002-1426-8572>),
Roger Gine [aut] (<https://orcid.org/0000-0003-0288-9619>),
Philippine Louail [aut] (<https://orcid.org/0009-0007-5429-6846>)

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

### R topics documented:

- addElements
- adductFormula
- adductNames
- calculateKendrickMass
- calculateMass
- containsElements
- convertMtime
- correctRindex
- countElements
- formula2mz
- indexRtime
- internalStandardMixNames
- internalStandards
- isotopicSubstitutionMatrix
- isotopologues
- mass2mz
- mclosest
- multiplyElements
- mz2mass
- pasteElements
- standardizeFormula
- subtractElements

**Index**

```
Index

addElements Combine chemical formulae

Description

addElements Add one chemical formula to another.

Usage

addElements(x, y)
```
adductFormula

Arguments

\(x\) character strings with chemical formula
\(y\) character strings with chemical formula that should be added from \(x\)

Value

character Resulting formula

Author(s)

Michael Witting and Sebastian Gibb

Examples

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

adductFormula  Calculate a table of adduct (ionic) formulas

Description

adductFormula calculates the chemical formulas for the specified adducts of provided chemical formulas.

Usage

adductFormula(formulas, adduct = "[M+H]+", standardize = TRUE)

Arguments

formulas character with molecular formulas for which adduct formulas should be calculated.
adduct character or data.frame of valid adduct. to be used. Custom adduct definitions can be provided via a data.frame but its format must follow adducts()
standardize logical(1) whether to standardize the molecular formulas to the Hill notation system before calculating their mass.

Value

character matrix with formula rows and adducts columns containing all ion formulas. In case an ion can’t be generated (eg. [M-NH3+H]+ in a molecule that doesn’t have nitrogen), a NA is returned instead.

Author(s)

Roger Gine
adductNames

See Also

adductNames() for a list of all available predefined adducts and adducts() for the adduct data.frame definition style.

Examples

# Calculate the ion formulas of glucose with adducts [M+H]+, [M+Na]+ and [M+K]+


# Use a custom set of adduct definitions (For instance, a iron (Fe2+) adduct)
custom_ads <- data.frame(name = "[M+Fe]2+", mass_multi = 0.5, charge = 2,
                         formula_add = "Fe", formula_sub = "C0",
                         positive = "TRUE")
adductFormula("C6H12O6", custom_ads)

adductNames

Retrieves 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

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

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

**Description**

Kendrick mass defect analysis is a way to analyze high-resolution MS data in order to identify homologous series. The Kendrick mass (KM) is calculated by choosing a specific molecular fragment (e.g., CH2) and setting its mass to an integer mass. In case of CH2 the mass of 14.01565 would be set to 14. The Kendrick mass defect (KMD) is defined as the difference between the KM and the nominal (integer) KM. All molecules of homologous series, e.g., only differing in the number of CH2, will have an identical KMD. In an additional step the KMD can be referenced to the mass defect of specific lipid backbone and by this normalize values to the referenced KMD (RKMD). This leads to values of 0 for saturated species or -1, -2, -3, etc for unsaturated species.

Available functions are:

- `calculateKm`: calculates the Kendrick mass from an exact mass for a specific molecular fragment, e.g., "CH2".
- `calculateKmd`: calculates the Kendrick mass defect from an exact mass for a specific molecular fragment, e.g., "CH2".
- `calculateRkmd`: calculates the referenced Kendrick mass defect from an exact mass for a specific molecular fragment, e.g., "CH2", and a reference KMD.
- `isRkmd`: Checks if a calculated RKMD falls within a specific error range around a negative integer corresponding the number of double bonds, in case of CH2 as fragment.

**Usage**

```r
calculateKm(x, fragment = 14/14.01565)
calculateKmd(x, fragment = 14/14.01565)
calculateRkmd(x, fragment = 14/14.01565, rkmd = 0.749206)
```

**Arguments**

- `x` numeric with exact masses or calculated RKMDs in case of `isRkmd`.
- `fragment` numeric(1) or character(1) corresponding factor or molecular formula of molecular fragment, e.g. 14 / 14.01565 or "CH2" for CH2.
calculateMass

rkmd numeric(1) KMD used for referencing of KMDs.

rkmdTolerance numeric(1) Tolerance to check if RKMD fall around a negative integer corresponding to the number of double bonds

Value
numeric or boolean. All functions, except isRkmd return a numeric with same length as the input corresponding to the KM, KMD or RMKD. isRkmd returns a logical with TRUE or FALSE indicating if the RKMD falls within a specific range around a negative integer corresponding to the number of double bonds.

Author(s)
Michael Witting

Examples

calculateKm(760.5851)
calculateKmd(760.5851)
calculateRkmd(760.5851, rkmd = 0.749206)
isRkmd(calculateRkmd(760.5851, rkmd = 0.749206))

---

calculateMass  Calculate exact mass

Description

calculateMass calculates the exact mass from a formula.

Usage

calculateMass(x)

Arguments

x character representing chemical formula(s) or a list of numeric with element counts such as returned by countElements().

Value
numeric Resulting exact mass.

Author(s)
Michael Witting
containsElements

Examples

```r
calculateMass("C6H12O6")
calculateMass("NH3")
calculateMass(c("C6H12O6", "NH3"))
calculateMass(c("C6H12O6", "[13C3]C3H12O6"))
```

containsElements checks if one formula is contained in another.

**Usage**

`containsElements(x, y)`

**Arguments**

- `x` character strings with a chemical formula
- `y` character strings with a chemical formula that shall be contained in `x`

**Value**

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

**Author(s)**

Michael Witting and Sebastian Gibb

**Examples**

```r
containsElements("C6H12O6", "H2O")
containsElements("C6H12O6", "NH3")
```
**Description**

convertMtime performs effective mobility scale transformation of CE(-MS) data, which is used to overcome variations of the migration times, caused by differences in the Electroosmotic Flow (EOF) between different runs. In order to monitor the EOF and perform the transformation, neutral or charged EOF markers are spiked into the sample before analysis. The information of the EOF markers (migration time and effective mobility) will be then used to perform the effective mobility transformation of the migration time scale.

**Usage**

```r
convertMtime(
  x = numeric(),
  rtime = numeric(),
  mobility = numeric(),
  tR = 0,
  U = numeric(),
  L = numeric()
)
```

**Arguments**

- `x` numeric vector with migration times in minutes.
- `rtime` numeric vector that holds the migration times (in minutes) of either one or two EOF markers in the same run of which the migration time is going to be transformed.
- `mobility` numeric vector containing the respective effective mobility (in mm² / (kV * min)) of the EOF markers. If two markers are used, one is expected to be the neutral marker, i.e. having a mobility of 0.
- `tR` numeric a single value that defines the time (in minutes) of the electrical field ramp. The default is 0.
- `U` numeric a single value that defines the voltage (in kV) applied. Note that for reversed polarity CE mode a negative value is needed. Is only used if the transformation is performed based on a single marker.
- `L` numeric a single value that defines the total length (in mm) of the capillary that was used for CE(-MS) analysis. Is only used if the transformation is performed based on a single marker.

**Value**

numeric vector of same length as `x` with effective mobility values.
correctRindex

Author(s)
Liesa Salzer

Examples

```r
rtime <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
marker_rt <- c(20, 80)
mobility <- c(0, 2000)
convertMtime(rtime, marker_rt, mobility)
```

---

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

```r
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 strings representing a chemical formula into a named vector of element counts.

Usage
countElements(x)

Arguments
x character() representing a chemical formula.

Value
list of integer with the element counts (names being elements).

Author(s)
Michael Witting and Sebastian Gibb

See Also
pasteElements()

Examples
countElements(c("C6H12O6", "C11H12N2O2"))

formula2mz  Calculate mass-to-charge ratio from a formula

Description
formula2mz calculates the m/z values from a list of molecular formulas and adduct definitions. 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
formula2mz(formula, adduct = "[M+H]+", standardize = TRUE)
Arguments

- **formula** character with one or more valid molecular formulas for which their 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.
- **standardize** logical whether to standardize the molecular formulas to the Hill notation system before calculating their mass.

Value

Numeric matrix with same number of rows than elements in `formula` 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 `formula` for each defined adduct.

Author(s)

Roger Gine

Examples

```r
## Calculate m/z values of adducts of a list of formulas
formulas <- c("C6H12O6", "C9H11NO3", "C16H13ClN2O")
formula2mz(formulas, ads)
formula2mz(formulas, adductNames()) #All available adducts

## Use custom-defined adducts as input
custom_ads <- data.frame(mass_add = c(1, 2, 3), mass_multi = c(1, 2, 0.5))
formula2mz(formulas, custom_ads)

## Use standardize = FALSE to keep formula unaltered
formula2mz("H12C6O6",)
formula2mz("H12C6O6", standardize = FALSE)
```

---

**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.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
internalStandards

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.
isotopicSubstitutionMatrix

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 is associated with an isotopic substitution (which can involve isotopes of several elements or different isotopes of the same element). In general for each isotopic substitution multiple rows are present in the data.frame. Each row provides parameters to compute bounds (for the ratio between the isotopologue peak and the monoisotopic one) on a certain mass range. 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:

- "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.
- "maxmass": the maximal mass of a compound for which the isotopic substitution was found. Peaks with a mass higher 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.
- "leftend": left endpoint of the mass interval.
- "rightend": right endpoint of the mass interval.
- "LBint": intercept of the lower bound line on the mass interval whose endpoints are "leftend" and "rightend".
- "LBslope": slope of the lower bound line on the mass interval.
- "UBint": intercept of the upper bound line on the mass interval.
- "UBslope": slope of the upper bound line on the mass interval.

Usage

isotopicSubstitutionMatrix(source = c("HMDB_NEUTRAL"))
isotopologues

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

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

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

```r
isotopologues(
x,
substDefinition = isotopicSubstitutionMatrix(),
tolerance = 0,
ppm = 20,
```

Identifying isotopologue peaks in MS data

isotopologues
isotopologues

```r
seedMz = numeric(),
charge = 1,
.check = TRUE
)
```

**Arguments**

- `x` matrix or data.frame with spectrum data. The first column is expected to contain \textit{m/z} and the second column intensity values. The \textit{m/z} values in that matrix are expected to be increasingly ordered and no NA values should be present.

- `substDefinition` matrix or data.frame with definition of isotopic substitutions (columns "name" and "md" are among the mandatory columns). The rows in this matrix have to be ordered by column \textit{md} in increasing order. See \texttt{isotopicSubstitutionMatrix()} for more information on the format and content.

- `tolerance` numeric(1) representing the absolute tolerance for the relaxed matching of \textit{m/z} values of peaks. See \texttt{MsCoreUtils::closest()} for details.

- `ppm` numeric(1) representing a relative, value-specific parts-per-million (PPM) tolerance for the relaxed matching of \textit{m/z} values of peaks. See \texttt{MsCoreUtils::closest()} for details.

- `seedMz` numeric vector of ordered \textit{m/z} values. If provided, the function checks if there are peaks in \textit{x} which \textit{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.

- `check` logical(1) to disable input argument check. Should only be set to FALSE if provided \textit{m/z} values are guaranteed to be increasingly ordered and don’t contain NA values.

**Details**

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

For matching peaks, the function next evaluates whether their intensity is within the expected (pre-defined) intensity range. Using "LBInt", "LBslope", "UBInt", "UBslope" of the previously matched isotopic substitution in \texttt{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 indexes of the rows in \textit{x} with potential isotopologues of the same compound.
mass2mz

Author(s)

Andrea Vicini

Examples

```r
## 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
```r
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)
```

mcclosest

Extract closest values in a pairwise manner between two matrices

Description
The mclosest function calculates the closest rows between two matrices (or data frames) considering pairwise differences between values in columns of x and table. It returns the index of the closest row in table for each row in x.

Usage
```r
mcclosest(x, table, ppm = 0, tolerance = Inf)
```

Arguments
- **x**: numeric matrix or data frame representing the query data. Each row in x will be compared to every row in table. Both x and table are expected to have the same number of columns, and the columns are expected to be in the same order.
### multiplyElements

Multiply the number of atoms of each element by a constant, positive, integer

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>table</code></td>
<td>numeric matrix or data frame containing the reference data to be matched with each row of <code>x</code>. Each row in <code>table</code> will be compared to every row in <code>x</code>. Both <code>table</code> and <code>x</code> are expected to have the same number of columns, and the columns are expected to be in the same order.</td>
</tr>
<tr>
<td><code>ppm</code></td>
<td>numeric representing a relative, value-specific parts-per-million (PPM) tolerance that is added to tolerance (default is 0).</td>
</tr>
<tr>
<td><code>tolerance</code></td>
<td>numeric accepted tolerance. Defaults to <code>tolerance = Inf</code>, thus for each row in <code>x</code> the closest row in <code>table</code> is reported, regardless of the magnitude of the (absolute) difference.</td>
</tr>
</tbody>
</table>

### Details

If, for a row of `x`, two rows of `table` are closest only the index of first row will be returned.

For both the `tolerance` and `ppm` arguments, if their length is different to the number of columns of `x` and `table`, the input argument will be replicated to match it.

### Value

integer vector of indices indicating the closest row of `table` for each row of `x`. If no suitable match is found for a row in `x` based on the specified `tolerance` and `ppm`, the corresponding index is set to NA.

### Author(s)

Philippine Louail

### Examples

```r
x <- data.frame(a = 1:5, b = 3:7)
table <- data.frame(c = c(11, 23, 3, 5, 1), d = c(32:35, 45))

## Get for each row of `x` the index of the row in `table` with the smallest difference of values (per column)
mclosest(x, table)

## If the absolute difference is larger than `tolerance`, return `NA`. Note that the tolerance value of `25` is used for difference for each pairwise column in `x` and `table`.
mclosest(x, table, tolerance = 25)
```

---

### multiplyElements

Multiply chemical formulas by a scalar

Description

multiplyElements Multiply the number of atoms of each element by a constant, positive, integer
Usage

multiplyElements(x, k)

Arguments

x character strings with chemical formula
k numeric(1) positive integer by which each formula will be multiplied.

Value

character strings with the standardized chemical formula.

Author(s)

Roger Gine

Examples

multiplyElements("H2O", 3)
multiplyElements(c("C6H12O6", "Na", "CH4O"), 2)

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**

```r
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` list/integer with element counts, names being individual elements.
standardizeFormula

Standardize a chemical formula

Description

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

Usage

standardizeFormula(x)

Arguments

x character, strings with the chemical formula to standardize.

Value

character strings with the standardized chemical formula.

Author(s)

Michael Witting and Sebastian Gibb

See Also

countElements()
pasteElements()

Examples

standardizeFormula("C6O6H12")
**subtractElements**  
subtract two chemical formula

**Description**

subtractElements subtracts one chemical formula from another.

**Usage**

subtractElements(x, y)

**Arguments**

- **x**  
  character strings with chemical formula
- **y**  
  character strings with chemical formula that should be subtracted from x

**Value**

character Resulting formula

**Author(s)**

Michael Witting and Sebastian Gibb

**Examples**

subtractElements("C6H12O6", "H2O")

subtractElements("C6H12O6", "NH3")
Index

addElements, 2
adductFormula, 3
adductNames, 4
adductNames(), 4, 11, 17, 18, 20, 21
adducts (adductNames), 4
adducts(), 3, 4, 10, 17, 20

calculateKendrickMass, 5
calculateKm (calculateKendrickMass), 5
calculateKmd (calculateKendrickMass), 5
calculateMass, 6
calculateRkmd (calculateKendrickMass), 5
containsElements, 7
correctRtime, 8
correctRindex, 9
countElements, 10
countElements(), 6, 21, 22

formula2mz, 10

indexRtime, 11
internalStandardMixNames, 12
internalStandardMixNames(), 13
internalStandards, 13
isotopicSubstitutionMatrix, 14
isotopicSubstitutionMatrix(), 16
isotopologues, 15
isotopologues(), 14
isRkmd (calculateKendrickMass), 5

mass2mz, 17
mass2mz(), 4, 21
mclosest, 18
MsCoreUtils::closest(), 16
multiplyElements, 19
mz2mass, 20
mz2mass(), 4, 18

pasteElements, 21
pasteElements(), 10, 22

standardizeFormula, 22
subtractElements, 23