Package: OrgMassSpecR (via r-universe)

August 23, 2024

Type Package

Title Organic Mass Spectrometry

Version 0.5-3

Date 2017-08-12

Depends R (>= 3.0.0), grid

Suggests lattice, knitr, rmarkdown

Description Organic/biological mass spectrometry data analysis.

License BSD_2_clause + file LICENSE

URL http://OrgMassSpec.github.io/

VignetteBuilder knitr

LazyData yes

NeedsCompilation no

Author Nathan Dodder [cre, aut, cph], Katharine Mullen [ctb]

Maintainer Nathan Dodder <ndodder@mail.sdsu.edu>

Repository CRAN

Date/Publication 2017-08-13 09:18:08 UTC

Contents

OrgMassSpecR-package 2
ConvertConcentration 2
ConvertPeptide 3
DeadVolume 4
Digest 5
DrawChromatogram 7
example.chromatogram.single 9
example.sequence 9
example.spectrum.labeled 10
example.spectrum.peptide 10
example.spectrum.unknown 11

ExchangeableAmides 11
FlowTime 12
FragmentPeptide 13
IsotopicDistribution 15
IsotopicDistributionHDX 16
IsotopicDistributionN 18
ListFormula 20
MolecularWeight 21
MonoisotopicMass 22
PeptideSpectrum 24
ReadMspDirectory 25
ReadMspFile 26
RetentionIndex 27
SpectrumSimilarity 28
WriteMspFile 30

Index **33**

`OrgMassSpecR-package` *Organic Mass Spectrometry*

Description

Organic/biological mass spectrometry data analysis.

http://OrgMassSpec.github.io/

`ConvertConcentration` *Convert Concentration Basis*

Description

Change the unit basis for a sample concentration, such as ng/g wet weight to ng/g dry weight, or pg/g lipid weight to pg/g wet weight.

Usage

```
ConvertConcentration(x, convert, percent)
```

Arguments

`x`	numeric value. The concentration to be converted.
`convert`	character string. One of `wet.to.dry`, `dry.to.wet`, `wet.to.lipid`, or `lipid.to.wet`. See details.
`percent`	numeric value. Either the percent moisture or percent lipid. See details.

Details

The convert argument specifies the type of conversion, for example wet.to.dry is wet weight basis to dry weight basis.

Conversion between wet weight basis and dry weight basis requires percent to equal the percent moisture in the sample. Conversion between wet weight basis and lipid weight basis requires percent to equal the percent lipid in the sample. The following definitions for percent moisture and percent lipid are used.

$$percentmoisture = \frac{wetweight - dryweight}{wetweight} * 100$$

$$percentlipid = \frac{lipidweight}{wetweight} * 100$$

The unit prefixes (for example ng/g or pg/g) are not changed by the conversion.

Value

Numeric vector of length 1. The unit prefixes are the same as for x, only the concentration basis is converted.

Author(s)

Nathan G. Dodder

Examples

```
## Convert a concentration of 15.3 ng/g wet weight,
## 5 percent lipid, to ng/g lipid weight.

ConvertConcentration(15.3, "wet.to.lipid", 5)
```

ConvertPeptide *Convert peptide sequence.*

Description

Convert single amino acid codes to an elemental formula or three letter codes.

Usage

```
ConvertPeptide(sequence, output = "elements", IAA = TRUE)
```

Arguments

sequence	a character string representing the amino acid sequence.
output	a character string specifying the output. Options are "elements" (default) for the elemental formula or "3letter" for the three letter codes.
IAA	logical. TRUE specifies iodoacetylated cysteine and FALSE specifies unmodified cystine. Used only in determining the elemental formula, not the three letter codes.

Details

The amino acid residues must be specified by the one letter codes defined in the help for Digest.

The argument IAA specifies treatment of the protein with iodoacetamide to break the disulfide bonds. This treatment produces iodoacetylated cysteine residues (elemental formula C5H8N2O2S).

Value

If output = "elements", the value is a list specifying the number of each element. This list can be used as input to other functions, see the examples. If output = "3letter", the value is a vector of length 1, containing the amino acid sequence in three letter codes.

Author(s)

Nathan G. Dodder

Examples

```
ConvertPeptide("SEQENCE", output = "3letter")

# as input to MonoisotopicMass
MonoisotopicMass(formula = ConvertPeptide("SEQENCE"), charge = 1)

# as input to MolecularWeight
MolecularWeight(formula = ConvertPeptide("SEQENCE"))
```

DeadVolume *Internal volume of tubing.*

Description

Calculate the internal volume of a defined length of tubing.

Usage

```
DeadVolume(internalDiameterMicrometers = 24, tubeLengthCentimeters = 45)
```

Arguments

`internalDiameterMicrometers`
: the internal diameter of the tubing, in micrometers; numeric value.

`tubeLengthCentimeters`
: the length of tubing, in centimeters; numeric value.

Value

Vector of the dead volume in microliters.

Author(s)

Nathan G. Dodder

See Also

`FlowTime`

Examples

```
DeadVolume(internalDiameterMicrometers = 47, tubeLengthCentimeters = 33)
```

`Digest` *Predict Peptides Resulting from Enzymatic Digest*

Description

Cleave an amino acid sequence (a protein or peptide) according to enzyme specific rules and calculate the precursor ion *m/z* values.

Usage

```
Digest(sequence, enzyme = "trypsin", missed = 0, IAA = TRUE,
       N15 = FALSE, custom = list())
```

Arguments

`sequence`
: a character string representing the amino acid sequence to be cleaved by the enzyme.

`enzyme`
: a character string specifying the rules for cleavage. Options are `"trypsin"` (default), `"trypsin.strict"` (see details), or `"pepsin"`.

`missed`
: the maximum number of missed cleavages. Must be an integer of 0 (default) or greater. An error will result if the specified number of missed cleavages is greater than the maximum possible number of missed cleavages.

`IAA`
: logical. `TRUE` specifies iodoacetylated cysteine and `FALSE` specifies unmodified cystine.

N15	logical indicating if the nitrogen-15 isotope should be used in place of the default nitrogen-14 isotope.
custom	a list specifying user defined residues as custom = list(code, mass), where code is a vector of one letter characters and mass is a vector of the respective monoisotopic masses. See Details and Examples.

Details

The amino acid residues must be specified by one letter codes. The predefined residues are:

A = alanine	L = leucine
R = arginine	K = lysine
N = asparagine	M = methionine
D = aspartic acid	F = phenylalanine
C = cysteine	P = proline
E = glutamic acid	S = serine
Q = glutamine	T = threonine
G = glycine	W = tryptophan
H = histidine	Y = tyrosine
I = isoleucine	V = valine

If "trypsin" is specified, the sequence is cleaved on the c-terminal side of K and R residues, except if K or R is followed by P. If "trypsin.strict" is specified, the sequence is cleaved on the c-terminal side of K and R residues. If "pepsin" is specified, the sequence is cleaved on the c-terminal side of F, L, W, Y, A, E, and Q residues. This rule is specific to pepsin at pH > 2, as used in hydrogen-deuterium exchange experiments.

When "trypsin" is specified, KP and RP are not considered missed cleavages when missed is $>$ 0.

The argument IAA specifies treatment of the protein with iodoacetamide. This treatment produces iodoacetylated cysteine residues (elemental formula C5H8N2O2S).

If TRUE, the argument N15 specifies 100% nitrogen-15 incorporation. It is intended for proteins grown with a nitrogen-15 labeled food source. (Although the experiment itself may grow a protein with less than 100% nitrogen-15 incorporation). Setting N15 = TRUE does not modify the mass of a custom residue, or the mass of the nitrogen(s) added if IAA = TRUE.

If a custom residue code is identical to a predefined residue code, the custom residue mass will be used in place of the predefined mass.

The error message "object "mass" not found" indicates the input sequence contains an undefined residue(s).

Value

A data frame with the following column names.

peptide	resulting peptides.
start	beginning residue positions in the the original sequence.
end	ending residue positions in the the original sequence.

mc	number of missed cleavages.
mz1	monoisotopic *m/z* values for the $[M + H]^{1+}$ ions (where M is the precursor mass).
mz2	monoisotopic *m/z* values for the $[M + 2H]^{2+}$ ions.
mz3	monoisotopic *m/z* values for the $[M + 3H]^{3+}$ ions.

Author(s)

Nathan G. Dodder

References

The relative atomic masses of the isotopes are from the NIST Physical Reference Data Website http://physics.nist.gov/PhysRefData/Compositions/. The molar mass of a proton (H+) is from the NIST CODATA Website http://physics.nist.gov/cuu/Constants/index.html.

See Also

MonoisotopicMass, FragmentPeptide

Examples

```
## digest human serum albumin with 0 and 1 missed cleavages
Digest(example.sequence, missed = 1)

## digest human serum albumin with a phosphoserine at position 58
## and all methionines oxidized
modifiedHsaSequence <- strsplit(example.sequence, split = "")[[1]]
modifiedHsaSequence[58] <- "s"   # insert code for phosphoserine
modifiedHsaSequence <- paste(modifiedHsaSequence, collapse = "")
Digest(modifiedHsaSequence, custom = list(code = c("s","M"),
       mass = c(MonoisotopicMass(list(C=3, H=6, N=1, O=5, P=1)),
                MonoisotopicMass(list(C=5, H=9, N=1, O=2, S=1)))))

## digest human serum albumin with strict rules
Digest(example.sequence, enzyme = "trypsin.strict")
```

DrawChromatogram *Plot a Chromatogram*

Description

Plot a chromatogram, color the area under specified peak(s), and calculate the peak area(s).

Usage

```
DrawChromatogram(time, intensity, range = list(start, stop), color = "blue",
                 xlab = "retention time", ylab = "intensity",
                 ylim = c(0, max(intensity) * 1.1), las = 1, ...)
```

Arguments

time	numeric vector containing the time points (the x-axis).
intensity	numeric vector containing the respective signal intensities at each time point (the y-axis).
range	list describing the start and stop time points for each peak, defined as range = list(start, stop), where start is the numeric vector of starting time points and stop is the numeric vector of the respective ending time points.
color	vector of character strings specifying the color for each peak given in range. If a single color is specified, it is applied to all peaks.
xlab	character string specifying the x-axis label.
ylab	character string specifying the y-axis label.
ylim	numeric vector of length 2 specifying the range of the y axis.
las	numeric value specifying the rotation of the axis labels, see par for options.
...	additional parameters to be passed to plot().

Details

The area under the peak(s) is rendered using the polygon function. The area calculation assumes that the polygon does not self-intersect.

Value

A data frame with the following column names.

retention.time	retention times of the peaks specified in range
area	the respective peak areas
apex.intensity	the respective intensities at each peak apex

Author(s)

Nathan G. Dodder

Examples

```
## single peak
x <- DrawChromatogram(example.chromatogram.single$time,
                      example.chromatogram.single$intensity/100,
                      range = list(start = 25.4, stop = 26.1),
                      main = "example chromatogram 1",
                      ylab = "intensity x 100")
# label peak with retention time and area
text(x$retentionTime, x$apexIntensity + 1500,
     labels = paste("RT = ", round(x$retentionTime, digits = 1),
                    ", Area = ", round(x$peakArea), sep = ""), cex = 0.9)

## multiple peaks
y <- DrawChromatogram(example.chromatogram.multiple$time,
```

```
                        example.chromatogram.multiple$intensity / 1000,
                        range = list(start = c(21.5, 21.925, 23.1, 25.5, 27.35),
                          stop = c(21.925, 22.4, 23.6, 26.2, 28.0)),
                        color = c("blue", "red", "green", "yellow", "orange"),
                        xlab = "retention time (min)",
                        ylab = "intensity x 1000 (cps)",
                        main = "Example Chromatogram")
    ## label peaks
    text(y$retentionTime, y$apexIntensity + 50, labels = c("a", "b", "c", "d", "e"))
```

---

example.chromatogram.single *Example Chromatograms*

Description

Examples for DrawChromatogram.

Usage

```
example.chromatogram.single
example.chromatogram.multiple
```

Format

A data frame consisting of time (minutes) in the first column and intensity (counts per second) in the second.

Author(s)

Nathan G. Dodder

---

example.sequence *Human Serum Albumin Amino Acid Sequence*

Description

Used in examples.

Usage

```
example.sequence
```

Format

Character sequence.

Source

Downloaded from the ExPASy Proteomics Server (P02768) on 2009-02-07. Residues 25-609.

`example.spectrum.labeled`

Example MALDI-TOF Spectrum of Nitrogen-15 Labeled Peptide

Description

Example used in Vinegette.

Usage

```
example.spectrum.labeled
```

Format

A data frame consisting of *m/z* values in the first column and intensity in the second.

Author(s)

Protein expressed by Wei-Li Liao.

Source

Precursor ion spectrum of peptide YEVQGEVFTKPQLWP acquired on a MALDI-TOF/TOF. The peptide is labeled with approximately 99% nitrogen-15.

`example.spectrum.peptide`

Example Peptide Fragmentation Spectrum

Description

Example for `PeptideSpectrum`.

Usage

```
example.spectrum.peptide
```

Format

A data frame consisting of (centroid) *m/z* values in the first column and peak intensities in the second.

Source

Peptide sequence NIDALSGMEGR. Acquired on a hybrid triple quadruploe/linear ion trap mass spectrometer in collision induced dissociation (CID) mode.

`example.spectrum.unknown` *Example EI Fragmentation Spectra*

Description

Examples for `SpectrumSimilarity`.

Usage

```
example.spectrum.unknown
example.spectrum.authentic
```

Format

A data frame consisting of (centroid) *m/z* values in the first column and peak intensities in the second.

Author(s)

Spectra acquired by Gauthier Eppe.

Source

Mass spectra of alanine derivatized with N-Methyl-N-[tert-butyldimethyl-silyl]trifluoroacetimide (MTBSTFA) collected in electron impact (EI) mode on a 2 dimensional gas chromatography time of flight (GCxGC-TOF) mass spectrometer. example.spectrum.unknown was acquired from a human plasma extract. example.spectrum.authentic was acquired from an authentic material.

`ExchangeableAmides` *Determine the Number of Backbone Amide Hydrogens*

Description

Determine the number of backbone amide hydrogens given a protein/peptide sequence. Used in hydrogen-deuterium exchange experiments.

Usage

```
ExchangeableAmides(sequence)
```

Arguments

sequence	character vector containing one or more amino acid sequences.

Details

The number of backbone amide hydrogens in an amino acid sequence is the number of residues, minus the number of prolines, minus 1.

Value

A numeric vector containing the number of exchangeable hydrogens.

Author(s)

Nathan G. Dodder

See Also

IsotopicDistributionHDX

Examples

```
ExchangeableAmides(c("VDVMCTA", "VSTPTL"))

## find the number of exchangeable amides for
## each peptide in a digest of human serum albumin
x <- Digest(example.sequence, enzyme = "pepsin", IAA = FALSE, missed = 4)
transform(x, exchange = ExchangeableAmides(x$peptide))
```

FlowTime *Solvent transit time.*

Description

Calculate the time required for a liquid to flow through a defined length of tubing. Intended for conventional-flow or nano-flow liquid chromatography.

Usage

```
FlowTime(internalDiameterMicrometers = 24, tubingLengthCentimeters = 45,
                flowRateMicrolitersPerMinute = 0.3)
```

Arguments

internalDiameterMicrometers	the internal diameter of the tubing, in micrometers; numeric value.
tubingLengthCentimeters	the length of tubing, in centimeters; numeric value.
flowRateMicrolitersPerMinute	the flow rate of the solvent in microliters per minute; numeric value

Details

The calculation assumes the viscosity of the liquid is negligible.

Value

Vector of the flow time in seconds.

Author(s)

Nathan G. Dodder

See Also

DeadVolume

Examples

```
FlowTime(internalDiameterMicrometers = 47, tubingLengthCentimeters = 33,
                  flowRateMicrolitersPerMinute = 5)
```

FragmentPeptide *Predict Peptide Fragment Ions*

Description

Determine the b- and y-ions or c- and z-ions produced by the fragmentation of a peptide by tandem mass spectrometry.

Usage

```
FragmentPeptide(sequence, fragments = "by", IAA = TRUE,
                N15 = FALSE, custom = list())
```

Arguments

sequence	a vector of character strings representing the amino acid sequences to be fragmented by the mass spectrometer.
fragments	character string specifying the fragmentation rules. Options are "by" (default) for the b- and y-ions, or "cz" for the c- and z-ions.
IAA	logical. TRUE specifies iodoacetylated cysteine and FALSE specifies unmodified cystine.
N15	logical indicating if the nitrogen-15 isotope should be used in place of the default nitrogen-14 isotope.
custom	a list specifying user defined residues as custom = list(code, mass), where code is a vector of one letter characters and mass is a vector of the respective monoisotopic masses. See Details and Examples.

Details

The amino acid residues must be specified by the one letter codes defined in the help for Digest.

The fragmentation rules can be set for collision induced dissociation (b- and y-ions) or electron transfer dissociation (c- and z-ions).

The argument IAA specifies treatment of the protein with iodoacetamide to break the disulfide bonds. This treatment produces iodoacetylated cysteine residues (elemental formula C5H8N2O2S).

If TRUE, the argument N15 specifies 100% nitrogen-15 incorporation. It is intended for proteins grown with a nitrogen-15 labeled food source. (Although the experiment itself may grow a protein with less than 100% nitrogen-15 incorporation). Setting N15 = TRUE does not modify the mass of a custom residue, or the mass of the nitrogen(s) added if IAA = TRUE.

If a custom residue code is identical to a predefined residue code, the custom residue mass will be used in place of the predefined mass.

Value

A data frame with the following column names. The data frame is arranged this way to facilitate selection of product-precursor ion pairs.

ms1seq	precursor ion sequence.
ms1z1	monoisotopic *m/z* value for the $[M + H]^{1+}$ precursor ions (where M is the precursor mass).
ms1z2	monoisotopic *m/z* value for the $[M + 2H]^{2+}$ precursor ions.
ms1z3	monoisotopic *m/z* value for the $[M + 3H]^{3+}$ precursor ions.
ms2seq	product ion sequence.
ms2type	the type and charge state of the product ions.
ms2mz	monoisotopic *m/z* values for the product ions.

Author(s)

Nathan G. Dodder and Katharine M. Mullen

References

The relative atomic masses of the isotopes are from the NIST Physical Reference Data Website http://physics.nist.gov/PhysRefData/Compositions/. The molar mass of a proton (H+) is from the NIST CODATA Website http://physics.nist.gov/cgi-bin/cuu/Value?mmp.

See Also

MonoisotopicMass, Digest, PeptideSpectrum

Examples

```
## fragment unlabeled peptide
FragmentPeptide("NECFLQHK")

## fragment peptide with carbon-13 labeled lysine
k.mass <- MonoisotopicMass(formula = list(C = 6, H = 12, N = 2, O = 1),
                        isotopes = list(C = 13.0033548378))
FragmentPeptide("NECFLQHk", custom = list(code = "k", mass = k.mass))

## fragment peptide with two modifications
m.mass <- MonoisotopicMass(formula = list(C=5, H=9, N=1, O=2, S=1))
FragmentPeptide("NDmELWk", custom = list(code = c("m", "k"), mass = c(m.mass, k.mass)))

## fragment a vector of peptides produced by Digest
x <- Digest(example.sequence)
y <- subset(x, nchar(x$peptide) > 5 & nchar(x$peptide) < 12)
FragmentPeptide(y$peptide)
```

IsotopicDistribution *Isotopic Distribution of an Organic Molecule.*

Description

Simulate the isotopic distribution of an organic molecule using the natural abundances of the isotopes.

Usage

```
IsotopicDistribution(formula = list(), charge = 1)
```

Arguments

formula	a list describing the charged elemental formula. The allowed elements are C, H, N, O, S, P, Br, Cl, F, Si.
charge	an integer specifying the number of positive or negative charges. A charge of zero is not allowed.

Details

The elemental formula should be that of the charged molecule; i.e, the charge argument does not change the elemental formula.

The algorithm used in this function is based on sample, and will give a slightly different result each time it is run.

This function is intended for simulating the isotopic distributions of small molecules (< approximately 3000 amu and <= 3 charges) as measured on low resolution mass spectrometers.

Value

A data frame with the following column names.

`mz`	the *m/z* value
`intensity`	the number of counts at each *m/z* value. The total is 10000.
`percent`	the intensity at each *m/z* value, expressed as a percent of the maximum intensity.

Author(s)

Nathan G. Dodder

References

The relative atomic masses of the isotopes are from the NIST Physical Reference Data Website http://physics.nist.gov/PhysRefData/Compositions/.

See Also

IsotopicDistributionN, IsotopicDistributionHDX, sample

Examples

```
x <- IsotopicDistribution(formula = list(C = 9, H = 4, Br = 5, Cl = 1, N = 2))
library(lattice)
xyplot(percent ~ mz, data = x,
  type = "h",
  xlab = "m/z",
  ylab = "intensity (%)",
  main = "Isotopic Distribution, C9H4Br5ClN2")
```

IsotopicDistributionHDX

Isotopic Distribution of a Peptide Undergoing H-D Exchange

Description

Simulates the isotopic distribution of a peptide undergoing hydrogen-deuterium exchange. Only the peptide backbone amides are labeled with deuterium.

Usage

```
IsotopicDistributionHDX(sequence, incorp, charge = 1,
                        custom = list(code = NULL, elements = NULL))
```

Arguments

sequence	character vector specifying the amino acid sequence.
incorp	numeric value from 0 to 1, specifying the fraction of deuterium incorporation in the backbone amides.
charge	numeric value specifying the number of positive charges ($[M + nH]^{n+}$)
custom	a list specifying user defined residues as `custom = list(code, mass)`, where `code` is a vector of one letter characters and `mass` is a vector of the respective monoisotopic masses.

Details

The amino acid residues must be specified by the one letter codes defined in the help for `Digest`. If a custom residue code is identical to a predefined residue code, the custom residue mass will be used in place of the predefined mass.

The natural incorporation of deuterium is 0.000115.

The algorithm used in this function is based on `sample`, and will give a slightly different result each time it is run. `IsotopicDistributionHDX` has not been tested for sequences over approximately 3000 amu and charges greater that 3+.

Value

A data frame with the following column names.

mz	the *m/z* value
intensity	the number of counts at each *m/z* value. The total is 10000.
percent	the intensity at each *m/z* value, expressed as a percent of the maximum intensity.

Author(s)

Nathan G. Dodder

References

The relative atomic masses of the isotopes are from the NIST Physical Reference Data Website http://physics.nist.gov/PhysRefData/Compositions/. The molar mass of a proton (H+) is from the NIST CODATA Website http://physics.nist.gov/cuu/Constants/index.html.

See Also

`IsotopicDistribution`, `IsotopicDistributionN`, `sample`

Examples

```
## simulate a peptide with 0.0115 and 100 percent incorporation and plot
x <- IsotopicDistributionHDX("NECFLQHK", incorp = 0.000115)
x$t <- "incorp = 0.0115%"
y <- IsotopicDistributionHDX("NECFLQHK", incorp = 1)
y$t <- "incorp = 100%"
```

```
z <- rbind(x, y)
library(lattice)
print(xyplot(percent ~ mz | t, data = z,
       type = "h", main = "H-D exchange simulation",
       xlab = "m/z", ylab = "intensity (%)"))
```

IsotopicDistributionN *Isotopic Distribution of a Nitrogen-15 Labeled Peptide.*

Description

Simulates the isotopic distribution of a nitrogen-15 labeled peptide. Intended for peptides from proteins grown in media with, for example, nitrogen-15 labeled ammonium chloride as the only nitrogen source.

Usage

```
IsotopicDistributionN(sequence, incorp, IAA = TRUE, charge = 1,
                      custom = list(code = NULL, elements = NULL))
```

Arguments

sequence	character vector specifying the amino acid sequence.
incorp	numeric value from 0 to 1, specifying the fraction of nitrogen-15.
IAA	logical. TRUE specifies iodoacetylated cysteine and FALSE specifies unmodified cystine.
charge	numeric value specifying the number of positive charges. Must be 1, 2, or 3. One proton per charge is added to the amino acid sequence ($[M + nH]^{n+}$).
custom	a list specifying a user defined residue as custom = list(code, elements), where code is a one letter character vector representing the custom residue and elements is a vector of the number of elements in the character vector, with the indicies equal to C,H,N,O,S,or P.

Details

The amino acid residues must be specified by the one letter codes defined in the help for Digest. If a custom residue code is identical to a predefined residue code, the custom residue will NOT be used in place of the predefined residue; doing so will cause errors.

The natural nitrogen-15 incorporation is 0.00368.

The argument IAA specifies treatment of the protein with iodoacetamide. This treatment produces iodoacetylated cysteine residues (elemental formula C5H8N2O2S). The nitrogen(s) added is not labeled.

The algorithm used in this function is based on sample, and will give a slightly different result each time it is run. It is similar to the algorithm used in reference 1. IsotopicDistributionN has not been tested for sequences over approximately 3000 amu and charges greater that 3+.

Value

A data frame with the following column names.

`mz`	the *m/z* value
`intensity`	the number of counts at each *m/z* value. The total is 10000.
`percent`	the intensity at each *m/z* value, expressed as a percent of the maximum intensity.

Author(s)

Nathan G. Dodder

References

1. "Method for Estimating the Isotopic Distributions of Metabolically Labeled Proteins by MALDI-TOFMS: Application to NMR Samples." Choudhary K, Spicer VL, Donald LJ, Duckworth HW, Ens W, Loewen PC, Standing KG, *Analytical Chemistry*, 2006, 78, 5419-5423.

2. "Perturbation and Interpretation of Nitrogen Isotope Distribution Patterns in Proteomics" Snijders APL, de Koning B, Wright PC, *Journal of Proteome Research*, 2005, 4, 2185-2191.

3. "Measurement of the Isotope Enrichment of Stable Isotope-Labeled Proteins Using High-Resolution Mass Spectra of Peptides" MacCoss MJ, Wu CC, Matthews DE, and Yates III JR, *Analytical Chemistry*, 2005, 77, 7646-7653.

The relative atomic masses of the isotopes are from the NIST Physical Reference Data Website http://physics.nist.gov/PhysRefData/Compositions/. The molar mass of a proton (H+) is from the NIST CODATA Website http://physics.nist.gov/cuu/Constants/index.html.

Note that in reference 3 "a biologically relevant 13C/12C isotope ratio of 1.096% was used instead of the more carbonate specific ratio of 1.112%." The 13C/12C ratio used by `IsotopicDistributionN` is 1.081%.

See Also

`IsotopicDistribution`, `IsotopicDistributionHDX`, `sample`

Examples

```
## simulate a peptide with 0.368 to 100 percent incorporation and plot
dn <- IsotopicDistributionN("NECFLQHK", incorp = 0.00368)
dn$t <- "incorp = 0.368%"
d50 <- IsotopicDistributionN("NECFLQHK", incorp = 0.5)
d50$t <- "incorp = 50%"
d99 <- IsotopicDistributionN("NECFLQHK", incorp = 0.99)
d99$t <- "incorp = 99%"
x <- rbind(dn, d50, d99)
library(lattice)
xyplot(percent ~ mz | t, data = x,
    type = "h",
    xlab = "m/z",
    ylab = "intensity (%)",
    main = "Isotopic distribution of peptide NECFLQHK with
```

```
            varying nitrogen-15 incorporation")
## Not run:
## simulate peptide DVFLGMFLYPYAR with oxidized methionine
IsotopicDistributionN("DVFLGmFLYPYAR", incorp = 0.5,
                      custom = list(code = "m",
                                    elements = c(C=5, H=9, N=1, O=2, S=1)))

## End(Not run)
```

`ListFormula` *Convert an Elemental Formula to a List*

Description

Convert a character string representing an elemental formula to a list representing the elemental formula. The list can be used as input to other functions.

Usage

```
ListFormula(elemental.formula)
```

Arguments

`elemental.formula`
: character string representing the elemental formula.

Details

To maintain compatibility with `MolecularWeight`, `MonoisotopicMass`, and `IsotopicDistribution` the elemental formula can contain only C,H,N,O,S,P,Br,Cl,F,I,Si, and Sn (I and Sn are default elements in `MolecularWeight` only). Elements not in this set will be ignored and a warning will be generated. The function can handle repeated elements, but not element multiplication indicated by parenthesis.

Known issue: Lower case letters after numbers, such as the "o" in "C12oBr5H5" will not be caught with a warning and the resulting list will contain the incorrect number of elements.

Author(s)

Nathan G. Dodder

See Also

`ConvertPeptide`

Examples

```
ListFormula("C14H8Cl4")
ListFormula("C6H5OH")
ListFormula("C15H12I3NO4")
```

`MolecularWeight` *Calculate the molecular weight of an organic molecule.*

Description

Given an elemental formula and the average relative atomic masses of the elements, determine the molecular weight.

Usage

```
MolecularWeight(formula = list(), amu = list())
```

Arguments

`formula`	a list describing the elemental formula. The allowed elements are C, H, N, O, S, P, Br, Cl, F, I, Si, Sn, and a user defined "x". See Examples.
`amu`	a list specifing user defined standard atomic masses of the elements (in atomic mass units).

Details

The user defined `x` in the argument lists can be used to define an additional element, such as a metal. See example.

Due to apparent rounding differences at the elemental level, other sources of molecular weight information may vary in the second or third decimal place.

This function will accept values that do not correspond to known physical reality, such as a fractional number of elements or the wrong standard atomic mass for an element.

Value

The molecular weight of the molecule.

Author(s)

Nathan G. Dodder

References

The relative atomic masses of the elements are from the NIST Physical Reference Data Website https://www.nist.gov/pml/atomic-weights-and-isotopic-compositions-relative-atomic-masses.

See Also

MonoisotopicMass

Examples

```
MolecularWeight(formula = list(C=2, H=4))

## Molecular weight of cyanocobalamin (C63H88CoN14O14P) with user defined cobalt
MolecularWeight(formula = list(C=63, H=88, N=14, O=14, P=1, x=1),
                amu = list(x = 58.933200))

## Molecular weight of triiodothyronine (C15H12I3NO4) using output from ListFormula
MolecularWeight(formula = ListFormula("C15H12I3NO4"))
```

`MonoisotopicMass` *Calculate the monoisotopic mass or monoisotopic m/z value of an organic molecule.*

Description

Given an elemental formula, the relative atomic masses of the isotopes, and the charge state, determine the monoisotopic mass or monoisotopic m/z value.

Usage

```
MonoisotopicMass(formula = list(), isotopes = list(), charge = 0)
```

Arguments

`formula`	a list describing the uncharged elemental formula. The allowed elements are C, H, N, O, S, P, Br, Cl, F, Si, and a user defined "x". See Examples.
`isotopes`	a list specifing the relative atomic masses of the isotopes.
`charge`	an integer specifying the number of positive or negative charges. If a charge is specified, the m/z of the molecule is calculated by adding or subtracting the specified number of protons. If `charge = 0`, the monoisotopic mass is returned. See details.

Details

The elemental formula describes the uncharged molecule; the charge argument will add or remove hydrogens as needed. This assumes an electrospray or MALDI type ionization. In electron impact ionization a positive charge is generated by loss of an electron, not additon of H+, therefore to calculate the correct m/z, the charge should be set to zero as a workaround.

The relative atomic masses of the most abundant isotopes (carbon-12, hydrogen-1, nitrogen-14, and oxygen-16) are set as the default values. For reference, the relative atomic masses of some common isotopic labels are:

carbon-13	13.0033548378
hydrogen-2	2.0141017780
nitrogen-15	15.0001088984
oxygen-18	17.9991604

The user defined element x can be used to define an additional element, such as a metal, or to specify a certain number of isotopically labeled atoms in a molecule. See Examples.

This function will accept values that do not correspond to known physical reality, such as a fractional number of elements, a fractional charge, or the wrong relative atomic mass value for an isotope.

Value

The monoisotopic mass of the unchaged molecule or the monoisotopic m/z value of the charged molecule.

Author(s)

Nathan G. Dodder and Katharine M. Mullen

References

The relative atomic masses of the isotopes are from the NIST Physical Reference Data Website http://physics.nist.gov/PhysRefData/Compositions/. The molar mass of a proton (H+) is from the NIST CODATA Website http://physics.nist.gov/cuu/Constants/index.html.

See Also

Digest, FragmentPeptide, MolecularWeight

Examples

```
## monoisotopic m/z of creatinine (C4H7N3O), +1 charge
## unlabeled
MonoisotopicMass(formula = list(C=4, H=7, N=3, O=1), charge = 1)
## with all carbon-13 atoms
MonoisotopicMass(formula = list(C=4, H=7, N=3, O=1),
                 isotopes = list(C = 13.0033548378),
                 charge = 1)
## with 2 carbon-12 atoms and 2 carbon-13 atoms
MonoisotopicMass(formula = list(C=2, H=7, N=3, O=1, x=2),
                 isotopes = list(x = 13.0033548378),
                 charge = 1)

## monoisotopic mass of cyanocobalamin (C63H88CoN14O14P)
MonoisotopicMass(formula = list(C=63, H=88, N=14, O=14, P=1, x=1),
                 isotopes = list(x = 58.9332002))
```

PeptideSpectrum *Plot Annotated Peptide Fragmentation Mass Spectrum*

Description

Identify the b- and y-ions or c- and z-ions in a peptide fragmentation mass spectrum given the sequence. Generate a plot and table identifying the fragment ions.

Usage

```
PeptideSpectrum(expt, theory, t = 0.4, b = 5, label = "", xlim = c(100, 1500),
                supress = FALSE)
```

Arguments

expt	data frame containing the experimental MS/MS peak list with the m/z values in the first column and corresponding intensities in the second
theory	data frame of theoretical fragment ions generated by FragmentPeptide
t	numeric value specifying the m/z tolerance for matching theoretical fragment ions to experimental peaks
b	numeric value specifying the baseline threshold for peak identification. Expressed as a percent of the maximum intensity.
label	character string to label the spectrum.
xlim	numeric vector of length 2, defining the beginning and ending values of the x-axis.
supress	logical specifying whether or not to supress the peak annotations.

Details

A graphics window of width = 10 and height = 5.5 seems to work well. A crude attempt is made to prevent overlapping peak annotations. If this is unsuccessful, use supress = TRUE and then add the annotations using the text function. If two or more peak identifications are made within the set tolerance, all identifications will be printed, overlapping, on the plot.

Value

Generates an annotated spectrum and a data frame showing the matches between the experimental peaks and the theoretical fragment ions. The column names refer to the experimental *m/z* value (expt_mz), the experimental peak intensity (expt_int) and the error between the experimental and theoretical *m/z* values (error). The other column names are the same as for FragmentPeptide.

Author(s)

Nathan G. Dodder

See Also

FragmentPeptide

Examples

```
## plot spectrum of peptide NIDALSGMEGR
t <- FragmentPeptide("NIDALSGMEGR")   # generate theoretical fragment ions
PeptideSpectrum(example.spectrum.peptide, t, label = "CID", xlim = c(100, 1200))
mtext("Peptide fragmentation mass spectrum", line = 1)
```

ReadMspDirectory *Read in MSP Formatted Mass Spectra*

Description

Reads in all `.msp` files within a directory and makes a single concatenated data frame of *m/z* values and intensities.

Usage

```
ReadMspDirectory(directory, skip = 2, comment.char = "",
                 remove.placeholders = TRUE)
```

Arguments

`directory`	character string. The path to the directory containing the `.msp` files. Can be "./" for the current directory on unix-like systems.
`skip`	integer. The number of lines at the top of the input file to skip before reading. Passed internally to read.table.
`comment.char`	a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether. Passed internally to read.table.
`remove.placeholders`	TRUE or FALSE specifying if zero intensity peaks in the MSP file should be removed from the resulting data frame.

Details

MSP is a National Institute of Standards and Technology (NIST) text format for centroid spectra. This format is exported by some instrument software.

The spectral files do not need to have the extension `.msp`. But an attempt will be made to process all files in the directory, regardless of the extension. The presense of other file types may cause errors.

`ReadMspDirectory` is a wrapper for `ReadMspFile`.

Value

A data frame with the following column names:

`mz`	the centroid *m/z* values.
`intensity`	the respective intensities.
`filename`	the corresponding filename (used to identify the spectra within the data frame).

Author(s)

Nathan G. Dodder

See Also

`ReadMspFile`

Examples

```
## The package directory msp contains two .msp files
ReadMspDirectory(paste(system.file(package = "OrgMassSpecR"),
  "/extdata/msp", sep = ""))
```

`ReadMspFile` *Read in a MSP Formatted Mass Spectrum*

Description

Reads in a `.msp` file and makes a data frame of *m/z* values and intensities.

Usage

```
ReadMspFile(file, skip = 2, comment.char = "",
            remove.placeholders = TRUE)
```

Arguments

`file`	character string. The `.msp` filename. Include the path to the filename if necessary.
`skip`	integer. The number of lines at the top of the input file to skip before reading. Passed internally to read.table.
`comment.char`	a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether. Passed internally to read.table.
`remove.placeholders`	TRUE or FALSE specifying if zero intensity peaks in the MSP file should be removed from the resulting data frame.

Details

MSP is a National Institute of Standards and Technology (NIST) ASCII format for centroid spectra. This format is exported by some instrument software.

The spectral file does not need to have the extension `.msp`.

Value

A data frame with the following column names:

`mz`	the centroid *m/z* values.
`intensity`	the respective intensities.

Author(s)

Nathan G. Dodder

See Also

`ReadMspDirectory`

Examples

```
ReadMspFile(paste(system.file(package = "OrgMassSpecR"), "/extdata/msp/pentaBDE.msp", sep = ""))
```

`RetentionIndex` *Non-Isothermal Retention Index Calculation*

Description

Determine the non-isothermal (temperature-programmed) gas chromatographic retention index of a target compound based on the retention times of adjacent n-alkanes.

Usage

```
RetentionIndex(n, target, preceding, following)
```

Arguments

`n`	the number of carbon atoms in the n-alkane eluting prior to the target compound.
`target`	the retention time of the target compound
`preceding`	the retention time of the n-alkane eluting prior to the target compound.
`following`	the retention time of the n-alkane eluting after the target compound.

Details

The retention time units can be either seconds or minutes, but should be consistent. The n-alkanes eluting before and after the target compound should differ by one carbon atom. This equation does not apply to isothermal separations or when the n-alkane series contains only odd or even carbon atoms.

Value

A vector giving the retention index.

Author(s)

Nathan G. Dodder

References

Van Den Dool H, Kratz PD, *Journal of Chromatography*, 1963, 11, 463-471.

Examples

```
RetentionIndex(8, 30, 28, 33)
```

SpectrumSimilarity *Similarity Between Two Mass Spectra*

Description

Generate a head-to-tail plot of the two mass spectra and calculate a similarity score.

Usage

```
SpectrumSimilarity(spec.top, spec.bottom, t = 0.25, b = 10,
                   top.label = NULL, bottom.label = NULL,
                   xlim = c(50, 1200), x.threshold = 0,
                   print.alignment = FALSE, print.graphic = TRUE,
                   output.list = FALSE)
```

Arguments

spec.top	data frame containing the experimental spectrum's peak list with the *m/z* values in the first column and corresponding intensities in the second
spec.bottom	data frame containing the reference spectrum's peak list with the *m/z* values in the first column and corresponding intensities in the second
t	numeric value specifying the tolerance used to align the *m/z* values of the two spectra.
b	numeric value specifying the baseline threshold for peak identification. Expressed as a percent of the maximum intensity.

`top.label` character string to label the top spectrum.

`bottom.label` character string to label the bottom spectrum.

`xlim` numeric vector of length 2, defining the beginning and ending values of the x-axis.

`x.threshold` numeric value of length 1 specifying the *m/z* threshold used for the similarity score calculation. Only peaks with *m/z* values above the threshold are used in the calculation. This can be used to exclude noise and/or non-specific ions at the low end of the spectrum. By default all ions are used.

`print.alignment` TRUE or FALSE value specifying if a data frame showing the aligned *m/z* values should be printed. Default is FALSE. The data frame contains aligned peak intensities from the top and bottom spectra, and is used to check the results. It contains all peaks, not only those subsetted by the `b` and `x.threshold` arguments.

`print.graphic` TRUE or FALSE value specifying if the head-to-tail plot should be printed. The defaut value is TRUE. The unaligned spectra are shown in the plot.

`output.list` TRUE or FALSE. If TRUE, will return a list with the similarity score, alignment dataframe, and plot. The plot is a gTree (grid graphics) object.

Details

The mass spectral similarity score is calculated as (where $\cdot$ is the dot product)

$$\cos\theta = \frac{u \cdot v}{\sqrt{u \cdot u}\sqrt{v \cdot v}}$$

where u and v are the aligned intensity vectors of the two spectra, as subsetted by the `b` and `x.threshold` arguments. The `t` argument is used to align the intensities. The bottom spectum is used as the reference spectrum, and the *m/z* values of peaks in the top spectrum that are within `t` of a reference *m/z* value are paired with that reference peak. Ideally, a single peak from the top spectrum should be paired with a single peak from the reference spectrum. Peaks without a match are paired with an intensity of zero.

Note that, although both are based on the cosine of the two intensity vectors, the spectral similarity score given by `SpectrumSimilarity` is not the same as that given by the NIST MS Search program, described in the reference below.

Value

A vector containing the similarity score. By default a base graphics head-to-tail plot of the two mass spectra is printed, and optionally the data frame showing the peak alignment is printed. Alternatively, if `output.list = TRUE` the function will return a list with the similarity score, alignment dataframe, and plot. In this case the plot is a gTree (grid graphics) object.

Author(s)

Nathan G. Dodder

References

"Optimization and Testing of Mass Spectral Library Search Algorithms for Compound Identification," Stein SE, Scott DR, *Journal of the American Society for Mass Spectrometry*, 1994, 5, 859-866.

See Also

PeptideSpectrum

Examples

```
## With output.list = FALSE (default)
SpectrumSimilarity(example.spectrum.unknown, example.spectrum.authentic,
                   top.label = "unknown, electron impact",
                   bottom.label = "derivatized alanine, electron impact",
                   xlim = c(25, 350))
# label peaks
plot.window(xlim = c(25,350), ylim = c(-125, 125))
text(c(73, 147, 158, 232, 260), c(100, 23, 44, 22, 15) + 10,
     c(73, 147, 158, 232, 260), cex = 0.75)
text(c(73, 147, 158, 232, 260), -c(100, 47, 74, 33, 20) - 10,
     c(73, 147, 158, 232, 260), cex = 0.75)
mtext("Spectrum similarity", line = 1)

## With output.list = TRUE
x <- SpectrumSimilarity(example.spectrum.unknown, example.spectrum.authentic,
                   top.label = "unknown, electron impact",
                   bottom.label = "derivatized alanine, electron impact",
                   xlim = c(25, 350), output.list = TRUE)

print(x)
grid.newpage()
grid.draw(x$plot) # draw to device
```

WriteMspFile *Write Spectra to the NIST MSP Text Format*

Description

Writes all spectra in the OrgMassSpecR data frame format to a single text file in NIST MSP format. The MSP files can be imported into the NIST MS Search program to make a custom seachable library.

Usage

```
WriteMspFile(spectra, metadata, filename = "library.msp", comment = "")
```

Arguments

spectra	data frame containing the mass spectra.
metadata	data frame containing the metadata.
filename	character string. Name (including path if needed) of the MSP text file.
comment	charater string applied to the COMMENT field of the MSP file. The comment is applied to all spectra.

Details

The spectra and metadata data frame format is specified in the help files for the LibraryReport functions in the spectral library packages, for example see SpecLibExample::LibraryReport. The spectra data frame must contain at least columns: filename, mz, and intensity. The metadata data frame must contain at least columns: filename and compound.

The NAME field for each spectrum in the MSP file is assigned based on compound in the metadata data frame.

Newlines are CR+LF since the NIST MS Search program is Windows based.

Value

An MSP file is written using a file connection.

Author(s)

Nathan G. Dodder

Examples

```
## Simple example to illustrate the input data formats.

spectra <- data.frame(filename = c(rep("Spectrum A", 2), rep("Spectrum B", 2)),
                      mz = c(50, 51, 100, 101),
                      intensity = c(70, 71, 90, 91))

metadata <- data.frame(filename = c("Spectrum A", "Spectrum B"),
                       compound = c("Compound A", "Compound B"))

## Not run:
WriteMspFile(spectra = spectra,
             metadata = metadata,
             filename = "Test.txt",
             comment = "Test Comment")

## End(Not run)

## Example using SpecLibExample package located at http://OrgMassSpec.github.io/libraries.html.
## Not run:
library(SpecLibExample)
WriteMspFile(spectra = spec,
             metadata = meta,
```

```
                comment = "example output")

## End(Not run)
```

Index

∗ **datasets**
    example.chromatogram.single, 9
    example.sequence, 9
    example.spectrum.labeled, 10
    example.spectrum.peptide, 10
    example.spectrum.unknown, 11
∗ **package**
    OrgMassSpecR-package, 2

ConvertConcentration, 2
ConvertPeptide, 3, *20*

DeadVolume, 4, *13*
Digest, *4*, 5, *14*, *17, 18*, *23*
DrawChromatogram, 7, *9*

example.chromatogram.multiple
        *(*example.chromatogram.single*)*, 9
example.chromatogram.single, 9
example.sequence, 9
example.spectrum.authentic
        *(*example.spectrum.unknown*)*, 11
example.spectrum.labeled, 10
example.spectrum.peptide, 10
example.spectrum.unknown, 11
ExchangeableAmides, 11

file, *31*
FlowTime, *5*, 12
FragmentPeptide, *7*, 13, *23–25*

IsotopicDistribution, 15, *17*, *19, 20*
IsotopicDistributionHDX, *12*, *16*, 16, *19*
IsotopicDistributionN, *16, 17*, 18

ListFormula, 20

MolecularWeight, *20*, 21, *23*
MonoisotopicMass, *7*, *14*, *20*, *22*, 22

OrgMassSpecR *(*OrgMassSpecR-package*)*, 2
OrgMassSpecR-package, 2

PeptideSpectrum, *10*, *14*, 24, *30*
polygon, *8*

ReadMspDirectory, 25, *27*
ReadMspFile, *25, 26*, 26
RetentionIndex, 27

sample, *15–19*
SpectrumSimilarity, *11*, 28

text, *24*

WriteMspFile, 30