Package ‘Spec2Annot’

September 28, 2026

Title Annotation of Mass Spectra

Version 1.3.5

Description Provides a comprehensive suite of functions to efficiently annotate mass spectra data. Motivated by the need for rapid and accurate chemical identification in high-resolution mass spectrometry, it integrates built-in chemical databases and high-performance C++ algorithms. Users can perform mass-to-charge (m/Z) and retention time searches, determine elemental compositions of molecules using heuristic rules, including specific isotopes, and annotate MS2 spectra with structural metrics using configurable chemistry rules.

License CeCILL

URL https://github.com/odisce/Spec2Annot

BugReports https://github.com/odisce/Spec2Annot/issues

Depends R (>= 4.0.0)

Encoding UTF-8

Imports data.table, magrittr, Rcpp, stringr

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

LinkingTo Rcpp

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Sylvain Dechaumet [aut, cre],
Etienne Thevenot [ctb],
Eric Venot [rev],
Annelaure Damont [ctb],
Anaïs Legrand [ctb]

Maintainer Sylvain Dechaumet <sylvain.dechaumet@cea.fr>

Repository CRAN

Date/Publication 2026-09-28 06:50:02 UTC

Contents

Adduct_db *adduct data*

Description

adduct data

Usage

```
Adduct_db
```

Format

A `data.table` with the common adducts in LC-ESI-MS

adduct Formula

charge charge

mz_query Exact mass value

add_formula_to_annot *Add formula to annotate_mz()*

Description

Add formula to annotate_mz()

Usage

```
add_formula_to_annot(mzannot_dt)
```

Arguments

`mzannot_dt`	A `data.table` containing the elemental composition of ions as returned by `annotate_mz()`.

Value

Return the mzannot_dt table with a new `formula` column containing a string with the elemental composition.

annotate_mz *Annotate a mass spectrum*

Description

This function annotate a full mass spectrum using a full brute force search optionnaly restricted by an elemental composition compo. Since generating the space to research can take some times, the parameter search_space can be pre-calculated in advance.

Usage

```
annotate_mz(
  input_spectrum,
  ppm = 5,
  polarity = NULL,
  compo = NULL,
  use_golden_ratio = TRUE
)
```

Arguments

input_spectrum	A mass spectrum as a data.table with mz and i columns
ppm	mass tolerance for the search in ppm.
polarity	Polarity is used to convert the mass spectrum to neutral mass before running the search.
compo	(optional) Elemental composition as a string ("C6H12O3") of the neutral form.
use_golden_ratio	Logical to return the maximum number of each element by using the ratios from the seven golden rules (TRUE) or not (FALSE).

Value

Return a data.table with the annotated spectrum.

Examples

```
annotate_mz(spectra_ms2, ppm = 3, polarity = 1, compo = "C10H12N5O6P1")
```

annotate_spectra *Annotate a matrix and return network edges*

Description

This function returns edges of networked ions

Usage

```
annotate_spectra(mass, debugL = FALSE)
```

Arguments

mass	A numeric vector
debugL	logisfggdcal to print debug message

Value

Return a numeric vector of network edges

annot_to_html *Convert annotation to html string*

Description

Convert annotation to html string

Usage

```
annot_to_html(annotation, compo = "", compo_replace = "X")
```

Arguments

annotation	String corresponding to the annotation
compo	Elemental composition to replace to in the annotation string
compo_replace	String to replace with compo in the annotation

Value

An html string of the annotation.

Examples

```
annot_to_html("[M+H]+_13C2", "C6H12O2", "M")
annot_to_html("[M+H-H2O]+_13C2", "C6H12O2", "M")
annot_to_html("[M+H-H2O]+_13C2_18O", "C6H12O2", "M")
annot_to_html("C6H13O2-H2O+_13C2_18O", "", "M")
annot_to_html("[M+H-(OH•)]+_13C2_18O", "C6H12O2", "M")
```

`brute_force_const` *Group m/Z from a vector based on tolerance*

Description

This function returns a matrix with m/Z groups based on a tolerance.

Usage

```
brute_force_const(
  mass = 120,
  ppm = 5,
  mass_vc = 0L,
  maxiter_vc_ = NULL,
  name_vc = 0L,
  debugl = 0L,
  debugit = 0L
)
```

Arguments

`mass`	Targeted mass
`ppm`	Mass tolerance in ppm to restrict results
`mass_vc`	Vector of elemnt masses (size = n)
`maxiter_vc_`	Vector of element maximum limits (optional) (size = n)
`name_vc`	Vector of elemnt names (size = n)
`debugl`	Integer (0: no message, 1: short, 2: verbose)
`debugit`	Integer for max iter to do

Value

A NumericMatrix containing the count of elements, exact mass `mass`, `ppm` deviation and `iteration` number of solutions found by the brute force algorithm.

`CJ1` *Cross join of two data.table*

Description

Cross join of two data.table

Usage

```
CJ1(DT1, DT2)
```

Arguments

`DT1`	A data.table
`DT2`	A data.table

Value

A data.table corresponding to the cross join of the two tables.

Examples

```
require(data.table)
CJ1(
 data.table(A = 1:10, B = "A"),
 data.table(
   C = 50:100,
   D = rep(c("C", "D", "E", "F"), length.out = 51)
 )
)
```

`db_monocharge` *Mono charged ions*

Description

Mono charged ions

Usage

```
db_monocharge
```

Format

A `data.table` with the common charges in LC-ESI-MS

Formula Formula

charge charge

lossL Does the resulting ion can produce in-source losses ?

mz_query Exact mass value

`electron_mass` *Return the mass of an electorn*

Description

Return the mass of an electorn

Usage

```
electron_mass()
```

Value

Return the mass of an electron as a numeric value

Examples

```
electron_mass()
```

`Element` *Periodic table as a data.table*

Description

Periodic table as a data.table

Usage

```
Element
```

Format

A `data.table` with the Periodic Table Elements fields

atomic_nb Atomic number

atomic_symb Atomic symbol

mass_nb Mass number

atomic_mass Exact atomic mass

isotopic_compo Relative isotopic abundance

References

https://physics.nist.gov/cgi-bin/Compositions/stand_alone.pl?ele=&ascii=ascii2&isotype=some

element_from_formula *Get element count from formula*

Description

Get element count from formula

Usage

```
element_from_formula(formula)
```

Arguments

formula	Formula as returned by gen_formula_from_compo

Value

Return a data.table with element count

Examples

```
# Exempl A
formula <- gen_formula_from_compo("C6H2O3Ca2K1")
element_from_formula(formula)

# Exempl B
formula <- gen_formula_from_compo("[C6H12O2+H]+_13C2")
element_from_formula(formula)
```

find_compo_from_mass *Find composition from mass*

Description

Find composition from mass

Usage

```
find_compo_from_mass(
  mass_target,
  ppm = 5,
  use_golden_ratio = TRUE,
  elements_vc = NULL,
  debugl = 0
)
```

Arguments

mass_target	mass to decompose
ppm	mass error to filter the results (in ppm)
use_golden_ratio	Logical to return the maximum number of each element by using the ratios from the seven golden rules (TRUE) or not (FALSE).
elements_vc	(optional) character vector containing the element to include
debugl	Integer (0: no message, 1: short, 2: verbose)

Value

A data.table with one proposition by line with it's elemental composition, the theoretical mass and the mass deviation with the query in ppm.

Examples

```
find_compo_from_mass(125.215, ppm = 10)
find_compo_from_mass(528.125, ppm = 3)
find_compo_from_mass(89.0476, ppm = 3, elements_vc = "C3H7NO2")
```

---

fun_generate_ions_from_mz

Calculate the ions database

Description

Calculate the ions database

Usage

```
fun_generate_ions_from_mz(
  mass = 252.2534,
  losses = Spec2Annot::Losses_db,
  db_iso = Spec2Annot::Isotopes_db,
  charges = Spec2Annot::db_monocharge,
  adducts = Spec2Annot::Adduct_db,
  polarity = NULL
)
```

Arguments

mass	numerical mass to use as base
losses	losses as data.table (see Spec2Annot::Losses_db) or NULL
db_iso	A data.table with isotopes informations (see Spec2Annot::Isotopes_db as example).

`charges`	charges as data.table (see Spec2Annot::db_monocharge)
`adducts`	adducts as data.table (see Spec2Annot::Adduct_db) or NULL
`polarity`	Polarity to subset the forms (1 for positive, 0 for negative mode or NULL for both)

Value

A data.table containing the full list of possible ions with the formula to calculate them

Examples

```
fun_get_ion_list()
```

`fun_get_ion_list` *Calculate the ions database*

Description

Calculate the ions database

Usage

```
fun_get_ion_list(
  losses = Spec2Annot::Losses_db,
  charges = Spec2Annot::db_monocharge,
  adducts = Spec2Annot::Adduct_db,
  polarity = NULL
)
```

Arguments

`losses`	losses as data.table (see Spec2Annot::Losses_db) or NULL
`charges`	charges as data.table (see Spec2Annot::db_monocharge)
`adducts`	adducts as data.table (see Spec2Annot::Adduct_db) or NULL
`polarity`	Polarity to subset the forms (1 for positive, 0 for negative mode or NULL for both)

Value

A data.table containing the full list of possible ions with the formula to calculate them

Examples

```
fun_get_ion_list()
```

gen_adduct *Generate adduct list from mz*

Description

Generate adduct list from mz

Usage

```
gen_adduct(
  mz,
  label = NULL,
  mz_type = c("neutral", "pos", "neg")[[1]],
  adduct_db = Spec2Annot::Adduct_db
)
```

Arguments

`mz`	Exact mass
`label`	(optional) a string to use in the ion labels if not set it will be just 'X'.
`mz_type`	Type of mass for targeted compound: "pos", "neg" or "neutral"
`adduct_db`	A data.table with adduct informations see (`Spec2Annot::Adduct_db`).

Value

A data.table with adducts mass and labels.

Examples

```
gen_adduct(125.53658, "Isovalerine", "neutral")
gen_adduct(116.0837, "C6H12O2", "neutral")
gen_adduct(116.0837, "C6H12O2", "pos")
```

gen_formula_from_compo *Generate formula from composition*

Description

Generate formula from composition

Usage

```
gen_formula_from_compo(compo)
```

Arguments

compo	Elemental composition as a string (ex.: "C6H12O3NH2")

Value

Return a string with an arithmetic formula used to calculate the final mass. This function can be used to check if the string parser behave correctly.

Examples

```
gen_formula_from_compo("C6H2O3NH4")
gen_formula_from_compo("-(C6H2O)-(H2O)")
gen_formula_from_compo("-C6H2O-H2O+Ca2-")
gen_formula_from_compo("[C6H12O2+H-H2O]+_13C")
gen_formula_from_compo("[2(C6H2O3)+NH4]+_13C3")
gen_formula_from_compo("[C6H12O2+H-H2O]++")
gen_formula_from_compo("[C6H12O2+H-H2O]--_13C1")
```

---

gen_ionlist — *Calculate list of targetd ions from neutral mz*

Description

Calculate list of targetd ions from neutral mz

Usage

```
gen_ionlist(
  neutral_mz = 153.5125,
  polarity = c(0, 1)[1],
  iso = c(TRUE, FALSE)[1],
  multi = 0,
  losses = c(TRUE, FALSE)[1],
  adducts = c(TRUE, FALSE)[1],
  mono_db = Spec2Annot::db_monocharge,
  loss_db = Spec2Annot::Losses_db,
  adduct_db = Spec2Annot::Adduct_db,
  db_iso = Spec2Annot::Isotopes_db
)
```

Arguments

neutral_mz	Neutral Mass
polarity	Polarity
iso	Logical
multi	Multimers

losses	Logical
adducts	Logical
mono_db	Database
loss_db	Database
adduct_db	Database
db_iso	Database

Value

a data.table

Examples

```
gen_ionlist(153.5125, 1, TRUE, 2, TRUE, TRUE)
```

`gen_isotopes` *Function to generate isotopes from an exact mass*

Description

Function to generate isotopes from an exact mass

Usage

```
gen_isotopes(mz, label = NULL, db_iso = Spec2Annot::Isotopes_db)
```

Arguments

mz	Exact mass
label	(optional) a string to use in the ion labels if not set it will be just 'X'.
db_iso	A data.table with isotopes informations (see `Spec2Annot::Isotopes_db` as example).

Value

Generate a data.table with isotopes mass and label based on the `mz` value. By default, the isotopes are chosen from `Spec2Annot::Isotopes_db` but it could be specified by the user.

Examples

```
gen_isotopes(125.53658, "Isovalerine")
```

gen_losses — *Generate losses list from mz*

Description

Generate losses list from mz

Usage

```
gen_losses(
  mz,
  label = NULL,
  mz_type = c("neutral", "pos", "neg")[[1]],
  loss_db = Spec2Annot::Losses_db
)
```

Arguments

`mz`	Exact mass
`label`	(optional) a string to use in the ion labels if not set it will be just 'X'.
`mz_type`	Type of mass for targeted compound: "pos", "neg" or "neutral"
`loss_db`	A data.table with losses information see `Spec2Annot::Losses_db`

Value

A data.table with losses mass and labels.

Examples

```
gen_losses(125.53658, "Isovalerine", "neutral")
gen_losses(116.0837, "C6H12O2", "neutral")
gen_losses(116.0837, "C6H12O2", "pos")
```

gen_monocharge — *Function to generate monocharges from an exact mass*

Description

Function to generate monocharges from an exact mass

Usage

```
gen_monocharge(
  mz,
  label = NULL,
  mz_type = c("neutral", "pos", "neg")[[1]],
  ion_mode = c("pos", "neg")[[1]],
  mono_db = Spec2Annot::db_monocharge
)
```

Arguments

mz	Exact mass
label	(optional) a string to use in the ion labels if not set it will be just 'X'.
mz_type	Type of mass for targeted compound: "pos", "neg" or "neutral"
ion_mode	Output mode wanted: "pos", "neg"
mono_db	A data.table with charges information (see `Spec2Annot::db_monocharge`)

Value

A data.table with ID, label and mz_query

Examples

```
gen_monocharge(125.53658, "Isovalerine", "neutral", "neg")
gen_monocharge(116.0837, "C6H12O2", "neutral", "neg")
gen_monocharge(116.0837, "C6H12O2", "neutral", "pos")
```

---

get_charge_from_compo *Get charge from composition*

Description

Get charge from composition

Usage

```
get_charge_from_compo(compo)
```

Arguments

compo	Elemental composition as a string (ex.: "C6H12O3NH2")

Value

Return the number of positive or negative charge

Examples

```
get_charge_from_compo("C6H2O3NH4")
get_charge_from_compo("C6H2O3NH4+")
get_charge_from_compo("C6H2O3NH4++")
get_charge_from_compo("C6H2O3NH4-")
get_charge_from_compo("C6H2O3NH4---")
get_charge_from_compo("[C6H12O2+H-H2O]+_13C")
get_charge_from_compo("[2(C6H2O3)+NH4]++_13C3")
get_charge_from_compo("[2(C6H2O3)-NH4]--_13C3_15N1")
```

`get_dbe` *Calculate DBE from an elemental composition*

Description

Calculation is performed as explained in http://ms-textbook.com/chapter-6/answer-6-3/

Usage

```
get_dbe(element_dt)
```

Arguments

`element_dt`	Either a data.table as returned by `Spec2Annot::element_from_formula()` or a string as "C18H12O3P".

Value

Return a string from an element table

Examples

```
get_dbe("C32H25N3O2S")
```

`get_element_from_mass` *Get Element from a mass*

Description

Get Element from a mass

Usage

```
get_element_from_mass(mass, use_golden_ratio = FALSE)
```

Arguments

`mass`	input mass
`use_golden_ratio`	Logical to return the maximum number of each element by using the ratios from the seven golden rules (`TRUE`) or not (`FALSE`).

Value

Return a string with the number maximum number of each element to expect from the mass. get_element_from_mass(125.2535, FALSE) get_element_from_mass(125.2535, TRUE)

`get_iso_from_annot` *Get isotopes from annotation*

Description

Get isotopes from annotation

Usage

```
get_iso_from_annot(annotation)
```

Arguments

`annotation`	Annotation in the form of `[M+H]+_13C1`

Value

Return a data.table with the following isotopes informations:

- `element`: element type (13C, 18O, ...)
- `text`: label to use for the element
- `elmt_nb`: Element count
- `isotope`: Isotope number (13, 18, ...)
- `mass`: Mass of the isotope
- `ID`: Unique identifier (`integer`)

Examples

```
get_iso_from_annot("[M+H]+_13C_18O")
get_iso_from_annot("[M+H]+_13C2")
```

get_nrule *Check Nitrogen rule*

Description

Calculation is performed as explained in https://en.wikipedia.org/wiki/Nitrogen_rule

Usage

```
get_nrule(n, mass)
```

Arguments

n	Nitrogen atom number
mass	Ion mass

Value

A logical corresponding to `TRUE` if the N rule is respected or `FALSE` if not.

Examples

```
get_nrule(5, 125.1235)
```

get_range *Get index in range*

Description

Get index in range

Usage

```
get_range(input, valA, valB)
```

Arguments

input	a numeric vector
valA	a numeric value for the lower bound
valB	a numeric value for the higher bound

Value

An `integer` vector of closest indexes found

get_senior — *Check Senior theorems*

Description

Check the Senior theorems as described in Morikawa and Newbold (2003) with the following: i) the sum of valencies is an even number, or the total number of atoms having odd valencies is even. ii) the sum of valencies is greater than or equal to twice the maximum valency. iii) the sum of valencies is greater than or equal to twice the number of atoms minus 1.

Usage

```
get_senior(element_dt, global = TRUE)
```

Arguments

`element_dt`	Either a data.table as returned by `Spec2Annot::element_from_formula()` or a string as "C18H12O3P".
`global`	Logical to return a unique value if all the theorem are valid (`TRUE`) or a vector with the result of each theorem (`FALSE`).

Value

If global is set to `TRUE`, return a unique logical value corresponding to `TRUE` if all the theorems pass or `FALSE` if any of them isn't. If global is set to `FALSE`, return a named logical vector with the result of each theorem.

References

1. Senior JK (1951) Partitions and Their Representative Graphs. Am J Math 73:663. doi: 10.2307/2372318
2. Morikawa T, Newbold BT (2003) Analogous odd-even parities in mathematics and chemistry. Chemistry 12:445–450

Examples

```
get_senior("C6H12O3")
```

`ions_database` *ions data*

Description

ions data

Usage

```
ions_database
```

Format

A `data.table` with the common ions found in LC-ESI-MS

Attribution Encoded formula

exact_mass Exact mass formula as a character string

charge charge

Type Molecular form (Monocharge, Loss, Adduct, ...)

`Isotopes_db` *isotopes data*

Description

isotopes data

Usage

```
Isotopes_db
```

Format

A `data.table` with the common isotopes found in LC-ESI-MS

isotope Isotope formula

mass Exact mass

abundance Natural abundance (in percent)

mass_diff Exact mass difference from non-isotopic element

Losses_db *losses data*

Description

losses data

Usage

```
Losses_db
```

Format

A `data.table` with the common losses found in LC-ESI-MS

loss loss formula

mz_query Exact mass

encod Character encoding

match_tables *Match two table*

Description

This function match two table, searching every entries in B which are contained in A mz +- ppmtol and A.rt +- rttol

Usage

```
match_tables(db_dt, exp_dt, ppmtol, rttol, debugL = FALSE)
```

Arguments

`db_dt`	a data.frame with at least mz and rt values (ref table)
`exp_dt`	a data.frame with at least mz and rt values (exp table)
`ppmtol`	a numeric value for the ppm tolerance
`rttol`	a numeric value for the rt tolerance
`debugL`	logisfggdcal to print debug message

Value

A `data.frame` with matched entries between db_dt and exp_dt.

mz_calc_ion *Calculate ion mass using form*

Description

Calculate ion mass using form

Usage

```
mz_calc_ion(mass, form = "-H")
```

Arguments

mass	numerical mass to use as base
form	chemical form to add or substract (on of `Spec2Annot::db_monocharge[, unique(Formula)]`)

Value

A numeric value corresponding to the mass form.

Examples

```
mz_calc_ion(142.5236, "-H")
```

mz_from_string *Calculate mz from a string with signs*

Description

Calculate mz from a string with signs

Usage

```
mz_from_string(string)
```

Arguments

string	Formula as a string of the form "C6H5O3+" or with isotopes `[C6H5O3+H]+_13C1`

Value

Return a numeric value corresponding to the mass of the `string` input

Examples

```
mz_from_string("C6H5O3+")
mz_from_string("C6H5O3++")
mz_from_string("[C6H4O3+H]+_13C1")
mz_from_string("[C6H4O3+H]+_13C2")
```

mz_ppm *Get mass deviation in ppm*

Description

This function returns a numeric value corresponding to the mass deviation between massa and massb in ppm.

Usage

```
mz_ppm(massa = 120.1253, massb = 120.1263)
```

Arguments

massa	First mass
massb	Second mass

Value

ppm as a numeric value between massa and massb.

mz_range *Get mass range with ppm*

Description

Get mass range with ppm

Usage

```
mz_range(mass = 120.1253, ppm = 10)
```

Arguments

mass	First mass
ppm	ppm tolerance mass +- ppm(mass)

Value

A numeric vector with the mass window.

`mz_vec_aggregate` *Group m/Z from a vector based on tolerance*

Description

This function returns a matrix with m/Z groups based on a tolerance.

Usage

```
mz_vec_aggregate(xx, tt)
```

Arguments

`xx`	A numeric and sorted vector of m/Z
`tt`	Tolerance in absolute m/Z to group peaks

Value

A `numeric vector` with group indexes

`search_db` *Search DB egaint EXP*

Description

Search DB egaint EXP

Usage

```
search_db(db_dt, exp_dt, ppmtol = 5, rttol = 5)
```

Arguments

`db_dt`	a data.table with id, mz, rt columns
`exp_dt`	a data.table with id, mz, rt columns
`ppmtol`	numeric value to set ppm search in m/Z dimension. Can be set to NULL to use only rt search.
`rttol`	numeric value to set rt tolerance in rt dimension. Can be set to NULL to use only mz search.

Value

Return the exp_dt data.table entries matching the criteria with 2 new columns: dbid (ids in db_dt) and expid (ids in exp_dt).

search_db_cpp *Search DB egaint EXP using cpp function*

Description

Search DB egaint EXP using cpp function

Usage

```
search_db_cpp(in_db, in_exp, ppmtol = 5, rttol = 5)
```

Arguments

`in_db`	a data.table with id, mz, rt columns
`in_exp`	a data.table with id, mz, rt columns
`ppmtol`	numeric value to set ppm search in m/Z dimension.
`rttol`	numeric value to set rt tolerance in rt dimension.

Value

Return the exp_dt data.table entries matching the criteria with 2 new columns: dbid (ids in db_dt) and expid (ids in exp_dt).

spectra_full *Spectra example*

Description

Spectra example

Usage

```
spectra_full
```

Format

A `data.table` with a centroided MS spectra used as example

mz m/Z value

intensity Measured intensity

ID Unique identifier

spectra_ms2 *Spectra example*

Description

MS2 spectrum of C10H12N5O6P1 Precursor mass: 330.0597 Mode: ESI + Collision mode: HCD Energy: 30

Usage

```
spectra_ms2
```

Format

A `data.table` with a centroided MS2 spectra used as example

mz m/Z value

i Measured intensity

string_from_element *Return a string from an element table*

Description

Return a string from an element table

Usage

```
string_from_element(element_dt)
```

Arguments

`element_dt`	A data.table as returned by `Spec2Annot::element_from_formula()`

Value

Return a string from an element table

Examples

```
"C6H12O3" %>%
  gen_formula_from_compo() %>%
  element_from_formula() %>%
  string_from_element()
```

`valence_db` *Valence data*

Description

Valence data

Usage

```
valence_db
```

Format

A `data.table` with element valence

atomic_symb Atomic symbol

valence valence value

Index