Package ‘isoorbi’

October 3, 2025

Type Package

Title Process Orbitrap Isotopocule Data

Version 1.5.2

URL https://isoorbi.isoverse.org/, https://github.com/isoverse/isoorbi

BugReports https://github.com/isoverse/isoorbi/issues

Depends R (>= 4.4.0)

Imports utils (>= 4.1.0), stats (>= 4.1.0), methods, rlang (>= 1.0.0), cli (>= 3.6.0), glue, withr (>= 3.0.0), lifecycle (>= 1.0.0), tidyr (>= 1.2.0), tibble, tidyselect (>= 1.2.0), dplyr (>= 1.1.1), ggplot2 (>= 3.4.0), scales (>= 1.2.1), readr (>= 2.1.0), readxl, openxlsx, purrr, prettyunits (>= 1.2.0), arrow (>= 21.0.0.0), knitr(>= 1.5.0)

Suggests devtools, fansi, rmarkdown, testthat (>= 3.0.0), vdiffr, forcats

Description Read and process isotopocule data from an Orbitrap Isotope Solutions mass spectrometer. Citation: Kantnerova et al. (Nature Protocols, 2024).

License MIT + file LICENSE

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Caj Neubauer [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-5348-5609>), Sebastian Kopf [aut] (ORCID: <https://orcid.org/0000-0002-2044-0201>), Kristýna Kantnerová [aut] (ORCID: <https://orcid.org/0000-0001-6259-3225>)

Maintainer Caj Neubauer `<caj.neubauer@colorado.edu>`

Repository CRAN

Date/Publication 2025-10-03 21:10:02 UTC

Contents

`isoorbi-package` *isoorbi: Process Orbitrap Isotopocule Data*

Description

Read and process isotopocule data from an Orbitrap Isotope Solutions mass spectrometer. Citation: Kantnerova et al. (Nature Protocols, 2024).

Details

Resources:

- Website for the isoorbi package: `https://isoorbi.isoverse.org`
- Package options: orbi_options

Author(s)

Maintainer: Caj Neubauer <`caj.neubauer@colorado.edu`> (ORCID) [copyright holder]

Authors:

- Sebastian Kopf <`sebastian.kopf@colorado.edu`> (ORCID)
- Kristýna Kantnerová <`kristyna.kantnerova@colorado.edu`> (ORCID)

See Also

Useful links:

- `https://isoorbi.isoverse.org/`
- `https://github.com/isoverse/isoorbi`
- Report bugs at `https://github.com/isoverse/isoorbi/issues`

`orbi_add_blocks_to_plot` *Plot blocks background*

Description

This function can be used to add colored background to a plot of dual-inlet data where different colors signify different data types (data, startup time, changeover time, unused). Note that this function only works with continuous and pseudo-log y axis, not with log y axes.

Usage

```
orbi_add_blocks_to_plot(
  plot,
  x = c("guess", "scan.no", "time.min"),
  data_only = FALSE,
  fill = .data$data_type,
 fill_colors = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#66A61E", "#E6AB02",
    "#A6761D", "#666666"),
  fill_scale = scale_fill_manual("blocks", values = fill_colors),
  alpha = 0.5,
  show.legend = !data_only
)
```

Arguments

`plot`	object with a dataset that has defined blocks
`x`	which x-axis to use (time vs. scan number). If set to "guess" (the default), the function will try to figure it out from the plot.
`data_only`	if set to TRUE, only the blocks flagged as "data" (`orbi_get_option("data_type_data")`) are highlighted
`fill`	what to use for the fill aesthetic, default is the block `data_type`
`fill_colors`	which colors to use, by default a color-blind friendly color palettes (RColorBrewer, dark2)
`fill_scale`	use this parameter to replace the entire fill scale rather than just the `fill_colors`
`alpha`	opacity settings for the background
`show.legend`	whether to include the background information in the legend

---

`orbi_adjust_block` *Manually adjust block delimiters*

Description

This function can be used to manually adjust where certain `block` starts or ends after it's been defined with `orbi_define_block_for_flow_injection()` or `orbi_define_blocks_for_dual_inlet()` using either time or scan number. Note that adjusting blocks removes all block segmentation. Make sure to call `orbi_segment_blocks()` **after** adjusting block delimiters.

Usage

```
orbi_adjust_block(
  dataset,
  block,
  filename = NULL,
  shift_start_time.min = NULL,
```

```
  shift_end_time.min = NULL,
  shift_start_scan.no = NULL,
  shift_end_scan.no = NULL,
  set_start_time.min = NULL,
  set_end_time.min = NULL,
  set_start_scan.no = NULL,
  set_end_scan.no = NULL
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`block`	the block for which to adjust the start and/or end
`filename`	needs to be specified only if the `dataset` has more than one `filename`
`shift_start_time.min`	if provided, the start time of the block will be shifted by this many minutes (use negative numbers to shift back)
`shift_end_time.min`	if provided, the end time of the block will be shifted by this many minutes (use negative numbers to shift back)
`shift_start_scan.no`	if provided, the start of the block will be shifted by this many scans (use negative numbers to shift back)
`shift_end_scan.no`	if provided, the end of the block will be shifted by this many scans (use negative numbers to shift back)
`set_start_time.min`	if provided, sets the start time of the block as close as possible to this time
`set_end_time.min`	if provided, sets the end time of the block as close as possible to this time
`set_start_scan.no`	if provided, sets the start of the block to this scan number (scan must exist in the `dataset`)
`set_end_scan.no`	if provided, sets the end of the block to this scan number (scan must exist in the `dataset`)

Value

A data frame (tibble) with block limits altered according to the provided start/end change parameters. Any data that is no longer part of the original block will be marked with the value of `orbi_get_option("data_type_unused")`. Any previously applied segmentation will be discarded (`segment` column set to `NA`) to avoid unintended side effects.

`orbi_aggregate_raw` *Aggregate data from raw files*

Description

This function allows dynamic aggregation and validation of data read by `orbi_read_raw()`. Like `orbi_read_raw()`, it is designed to be fail save by safely catching errors and reporting back on them (see `orbi_get_problems()`). This function should work out of the box for most files without additional modification of the `aggregator`.

Usage

```
orbi_aggregate_raw(
  files_data,
  aggregator = "standard",
  show_progress = rlang::is_interactive(),
  show_problems = TRUE
)
```

Arguments

`files_data`	the files read in by `orbi_read_raw()`
`aggregator`	typically the name of a registered aggregator (see all with `orbi_get_option("aggregators")`), default is the "standard" aggregator included in the package (`orbi_get_aggregator("standard")`). Other options are "minimal" (`orbi_get_aggregator("minimal")`) and "extended" (`orbi_get_aggregator("extended")`). The `aggregator` parameter can can also directly be an aggregator tibble (created/modified with `orbi_start_aggregator()` and/or `orbi_add_to_aggregator()`) that defines which data should be aggregated and how.
`show_progress`	whether to show a progress bar, by default always enabled when running interactively e.g. inside RStudio (and disabled in a notebook), turn off with `show_progress = FALSE`
`show_problems`	whether to show problems encountered along the way (rather than just keeping track of them with `orbi_get_problems()`). Set to `show_problems = FALSE` to turn off the live printout. Either way, all encountered problems can be retrieved with running `orbi_get_problems()` for the returned list

Value

a list of merged dataframes collected from the `files_data` based on the `aggregator` definitions

---

`orbi_analyze_shot_noise` *Shot noise calculation*

Description

This function computes the shot noise calculation.

Usage

```
orbi_analyze_shot_noise(dataset, include_flagged_data = FALSE)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules()` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`include_flagged_data`	whether to include flagged data in the shot noise calculation (FALSE by default)

Details

Analyze shot noise

will calculate for all combinations of `filename`, `compound`, and `isotopocule` in the provided `dataset`

Value

The processed data frame with new columns: `n_effective_ions`, `ratio`, `ratio_rel_se.permil`, `shot_noise.permil`

`orbi_calculate_ions` *Calculate ions from intensities*

Description

This functions calculates ions (`ions.incremental`) from intensities based on the equation

$$N_{ions} = S/N \cdot C_N/z \cdot \sqrt{R_N/R} \cdot \sqrt{N_{MS}}$$

where S is the reported signal (`intensity`) of the isotopocule, N is the noise associated with the signal (`peakNoise`), measured at the resolution setting R (`resolution`), the noise factor C_N (`CN`) is the number of charges corresponding to the Orbitrap noise band at some reference resolution R_N (`RN`), N_{MS} is the number of microscans, and z is the charge per ion (`charge`) of the isotopocule. See Makarov and Denisov (2009) and Eiler et al. (2017) for details about this equation. The default values for `CN` and `RN` are from the Orbitrap Exploris Isotope Solutions Getting Started Guide (BRE0032999, Revision A, October 2022). Note that the exact values of these factors are only critical if the number of ions are interpreted outside of ratio calculations (in ratio calculations, these factors cancel).

Usage

```
orbi_calculate_ions(dataset, CN = 3, RN = 240000)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`CN`	noise factor
`RN`	reference resolution of the noise factor

Details

If using a dataset read from isox files you might have to add a `charge` column if it does not yet exist that indicates the charge of the isotopocule. If using data from raw files, `orbi_identify_isotopocules()` will automatically call this function with the default `CN` and `RN` so you don't need to call it explicitly unless you want to change these parameters.

Value

same object as provided in `dataset` with new column `ions.incremental`

`orbi_calculate_ratios` *Calculate direct isotopocule ratios*

Description

This function calculates isotopocule/base peak ratios for all isotopocules. It does not summarize or average the ratios in any way. For a summarizing version of this function, see `orbi_summarize_results()`.

Usage

```
orbi_calculate_ratios(dataset)
```

Arguments

`dataset`	A data frame output after running `orbi_define_basepeak()`

Value

Returns a mutated dataset with `ratio` column added.

`orbi_calculate_summarized_ratio` *Calculate isotopocule ratio*

Description

This function calculates the ratio of two isotopocules (the `numerator` and `denominator`). This function averages multiple measurements of each using the `ratio_method` and returns a single value. Normally this function is not called directly by the user, but via the function `orbi_summarize_results()`, which calculates isotopocule ratios and other results for an entire dataset.

Usage

```
orbi_calculate_summarized_ratio(
  numerator,
  denominator,
  ratio_method = c("direct", "mean", "sum", "median", "geometric_mean", "slope",
    "weighted_sum")
)
```

Arguments

`numerator`	Column(s) used as numerator; contains ion counts
`denominator`	Column used as denominator; contains ion counts
`ratio_method`	Method for computing the ratio. **Please note well**: the formula used to calculate ion ratios matters! Do not simply use arithmetic mean. The best option may depend on the type of data you are processing (e.g., MS1 versus M+1 fragmentation). `ratio_method` can be one of the following: • `mean`: arithmetic mean of ratios from individual scans. • `sum`: sum of all ions of the numerator across all scans divided by the sum of all ions observed for the denominator across all scans. • `geometric_mean`: geometric mean of ratios from individual scans. • `slope`: The ratio is calculated using the slope obtained from a linear regression model that is weighted by the `numerator x`, using `stats::lm(x ~ y + 0, weights = x)`. • `weighted_sum`: A derivative of the `sum` option. The weighing function ensures that each scan contributes equal weight to the ratio calculation, i.e. scans with more ions in the Orbitrap do not contribute disproportionately to the total `sum` of `x` and `y` that is used to calculate `x/y`.

Value

Single value ratio between the isotopocules defined as `numerator` and `denominator` calculated using the `ratio_method`.

Examples

```
df <-
  system.file("extdata", "testfile_flow.isox", package = "isoorbi") |>
  orbi_read_isox()

ions_18O <- dplyr::filter(df, isotopocule == "18O")$ions.incremental
ions_M0 <- dplyr::filter(df, isotopocule == "M0")$ions.incremental

orbi_calculate_summarized_ratio(
  numerator = ions_18O, denominator = ions_M0, ratio_method = "sum"
)

orbi_calculate_summarized_ratio(
  numerator = ions_18O, denominator = ions_M0, ratio_method = "slope"
)
```

`orbi_check_isoraw` *Check for the isoorbi raw file reader*

Description

By default, this will install the isoraw reader if it is missing or outdated, and will ask the user to agree to Thermo's license agreement for the Thermo RawFileReader before proceeding. This function runs automatically during a raw file read and does not usually need to be called directly by the user.

Usage

```
orbi_check_isoraw(
  install_if_missing = !on_cran(),
  reinstall_if_outdated = !on_cran(),
  reinstall_always = FALSE,
  min_version = "0.2.2",
 source = paste0("https://github.com/isoverse/isoorbi/releases/download/isoraw-v",
    min_version),
  accept_license = FALSE,
  ...
)
```

Arguments

`install_if_missing`	install the reader if it's missing
`reinstall_if_outdated`	install the reader if it's outdated (i.e. not at least `min_version`)
`reinstall_always`	whether to (re-)install no matter what

`min_version`	the minimum version number required
`source`	the URL (or local path) where to find the raw file reader, by default this is the latests release of the executables on github
`accept_license`	explicitly accept Thermo's license agreement (if this is FALSE and the license has not previously been accepted, you will be asked about it)
`...`	passed on to `download.file` if (re-) installing the reader

`orbi_default_theme` *Default isoorbi plotting theme*

Description

Default isoorbi plotting theme

Usage

```
orbi_default_theme(text_size = 16, facet_text_size = 20)
```

Arguments

`text_size`	a font size for text
`facet_text_size`	a font size for facet text

Value

ggplot theme object

`orbi_define_basepeak` *Define the denominator for ratio calculation*

Description

`orbi_define_basepeak()` sets one isotopocule in the data frame as the base peak (ratio denominator) and calculates the instantaneous isotope ratios against it.

Usage

```
orbi_define_basepeak(dataset, basepeak_def)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`basepeak_def`	The isotopocule that gets defined as base peak, i.e. the denominator to calculate ratios

Value

same object as provided in `dataset` without the rows of the basepeak isotopocule and instead three new columns called `basepeak`, `basepeak_ions`, and `ratio` holding the basepeak information and the isotope ratios vs. the base peak

Examples

```
fpath <- system.file("extdata", "testfile_flow.isox", package = "isoorbi")
df <- orbi_read_isox(file = fpath) |>
  orbi_simplify_isox() |>
  orbi_define_basepeak(basepeak_def = "M0")
```

`orbi_define_blocks_for_dual_inlet`

Binning raw data into blocks for dual inlet analyses

Description

This function sorts out (bins) data into indivual blocks of reference, sample, changeover time, and startup time.

Usage

```
orbi_define_blocks_for_dual_inlet(
  dataset,
  ref_block_time.min,
  change_over_time.min,
  sample_block_time.min = ref_block_time.min,
  startup_time.min = 0,
  ref_block_name = orbi_get_option("di_ref_name"),
  sample_block_name = orbi_get_option("di_sample_name")
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`ref_block_time.min`	time where the signal is stable when reference is analyzed
`change_over_time.min`	time where the signal is unstable after switching from reference to sample or back
`sample_block_time.min`	time where the signal is stable when sample is analyzed

`startup_time.min`	initial time to stabilize spray
`ref_block_name`	the name of the reference being measured
`sample_block_name`	the name of the sample being measured

Value

A data frame (tibble) with block annotations in the form of the additional columns described below:

- `data_group` is an integer that numbers each data group (whether that's startup, a sample block, a segment, etc.) in each file sequentially to uniquely identify groups of data that belong together - this columns is NOT static (i.e. functions like `orbi_adjust_block()` and `orbi_segment_blocks()` will lead to renumbering) and should be used purely for grouping purposes in calculations and visualization
- `block` is an integer counting the data blocks in each file (0 is the startup block)
- `sample_name` is the name of the material being measured as defined by the `ref_block_name` and `sample_block_name` parameters
- `segment` is an integer defines segments within individual blocks - this will be `NA` until the optional `orbi_segment_blocks()` is called
- `data_type` is a text value describing the type of data in each `data_group` - for a list of the main categories, call `orbi_get_options("data_type")`

`orbi_define_block_for_flow_injection`

Define data block for flow injection

Description

Define a data block by either start and end time or start and end scan number. If you want to make segments in the blocks (optional), note that this function - manually defining blocks - removes all block segmentation. Make sure to call `orbi_segment_blocks()` **only after** finishing block definitions.

Usage

```
orbi_define_block_for_flow_injection(
  dataset,
  start_time.min = NULL,
  end_time.min = NULL,
  start_scan.no = NULL,
  end_scan.no = NULL,
  sample_name = NULL
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`start_time.min`	set the start time of the block
`end_time.min`	set the end time of the block
`start_scan.no`	set the start scan of the block
`end_scan.no`	set the end scan of the block
`sample_name`	if provided, will be used as the `sample_name` for the block

Value

A data frame (tibble) with block definition added. Any data that is not part of a block will be marked with the value of `orbi_get_option("data_type_unused")`. Any previously applied segmentation will be discarded (`segment` column set to `NA`) to avoid unintended side effects.

`orbi_export_data_to_excel`

Export data to excel

Description

This functions exports the `dataset` into an Excel file. If the `dataset` is aggregated data, use the `include` parameter to decide which part of the data to export.

Usage

```
orbi_export_data_to_excel(
  dataset,
  file,
  dbl_digits = 7,
  int_format = "0",
  dbl_format = sprintf(sprintf("%%.%sf", dbl_digits), 0),
  include = c("file_info", "summary", "scans", "peaks", "problems"),
  show_progress = rlang::is_interactive()
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`file`	file path to export the file - recursively creates the directory if non-existent
`dbl_digits`	how many digits to show for dbls (all are exported)

`int_format`	the excel formatting style for integers
`dbl_format`	the excel formatting style for doubles (created automatically from the dbl_digits parameter)
`include`	which tibbles to include if `dataset` is aggregated data. By default includes all but spectra
`show_progress`	whether to show a progress bar, by default always enabled when running interactively e.g. inside RStudio (and disabled in a notebook), turn off with `show_progress = FALSE`

Value

returns dataset invisibly for use in pipes

`orbi_filter_files` *Basic generic data files filter*

Description

This is a basic filter function for file names, compounds and time ranges. For filtering isotopocules, this function calls `orbi_filter_isotopocules()` internally (as of isoorbi version 1.5.0 `orbi_filter_isotopocules()` can also be used directly instead of via this function). Default value for all parameters is NULL, i.e. no filter is applied.

Usage

```
orbi_filter_files(
  dataset,
  filenames = NULL,
  compounds = NULL,
  isotopocules = NULL,
  time_min = NULL,
  time_max = NULL
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`filenames`	Vector of file names to keep, keeps all if set to `NULL` (the default)
`compounds`	Vector of compounds to keep, keeps all if set to `NULL` (the default)
`isotopocules`	Vector of isotopocules to keep, keeps all if set to `NULL` (the default)
`time_min`	Minimum retention time in minutes (`time.min`), no minimum if set to `NULL` (the default)
`time_max`	Maximum retention time in minutes (`time.min`), no maximum if set to `NULL` (the default)

Value

Filtered tibble

Examples

```
fpath <- system.file("extdata", "testfile_flow.isox", package = "isoorbi")
df <-
  orbi_read_isox(file = fpath) |>
  orbi_simplify_isox() |>
  orbi_filter_files(
    filenames = c("s3744"),
    compounds = "HSO4-",
    isotopocules = c("M0", "34S", "18O")
  )
```

`orbi_filter_flagged_data`
Filter out flagged data

Description

This function filters out data that have been previously flagged using functions `orbi_flag_satellite_peaks()`, `orbi_flag_weak_isotopocules()`, and/or `orbi_flag_outliers()`. Note that this function is no longer necessary to call explicitly as `orbi_analyze_shot_noise()` and `orbi_summarize_results()` automatically exclude flagged data.

Usage

```
orbi_filter_flagged_data(dataset)
```

Arguments

`dataset`	a tibble with previously flagged data from `orbi_flag_satellite_peaks()`, `orbi_flag_weak_isotopocules()`, and/or `orbi_flag_outliers()`

Value

a dataset with the flagged data filtered out

`orbi_filter_isotopocules` *Filter isotopocules*

Description

This function helps filter out missing isotopcules, unidentified peaks, or select for specific isotopocule. It can be called any time after `orbi_identify_isotopocules()` or after reading from an isox file. By default (i.e. if run without setting any parameters), it removes unidentified peaks and missing isotopcules and keeps all others.

Usage

```
orbi_filter_isotopocules(
  dataset,
  isotopocules = c(),
  keep_missing = FALSE,
  keep_unidentified = FALSE
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`isotopocules`	if provided, only these isotopocules will be kept
`keep_missing`	whether to keep missing isotopocules in the peaks list (i.e. those that should be there but are not), default is not to keep them
`keep_unidentified`	whether to keep unidentified isotopocules in the peaks list (i.e. peaks that have not been identified as a specificic isotopocule), default is not to keep them

Value

the `dataset` but filtered for these isotopocules

`orbi_filter_isox` *Filter isox files*

Description

[Deprecated] `orbi_filter_isox()` was renamed `orbi_filter_files()` to incorporate its wider scope for filtering both isox and raw files data.

Usage

```
orbi_filter_isox(...)
```

Arguments

`...`	arguments passed on to `orbi_filter_files()`

`orbi_filter_satellite_peaks` *Function replaced by* `orbi_flag_satellite_peaks()`

Description

Function replaced by `orbi_flag_satellite_peaks()`

Usage

```
orbi_filter_satellite_peaks(...)
```

Arguments

`...`	parameters passed on to the new function `orbi_flag_satellite_peaks()`.

`orbi_filter_scan_intensity` *Function replaced by* `orbi_flag_outliers()`

Description

Function replaced by `orbi_flag_outliers()`

Usage

```
orbi_filter_scan_intensity(..., outlier_percent)
```

Arguments

`...`	parameters passed on to the new function `orbi_flag_outliers()`.
`outlier_percent`	outlier_percent needs to be between 0 and 10, flags extreme scans based on TIC x injection time (i.e., ion intensity)

`orbi_filter_weak_isotopocules` *Function replaced by* `orbi_flag_weak_isotopocules()`

Description

Function replaced by `orbi_flag_weak_isotopocules()`

Usage

```
orbi_filter_weak_isotopocules(...)
```

Arguments

`...`	parameters passed on to the new function orbi_flag_weak_isotopocules().

`orbi_find_isox` *Find isox files*

Description

Finds all .isox files in a folder.

Usage

```
orbi_find_isox(folder, recursive = TRUE)
```

Arguments

`folder`	path to a folder with isox files
`recursive`	whether to find files recursively

Examples

```
# all .isox files provided with the isoorbi package
orbi_find_isox(system.file("extdata", package = "isoorbi"))
```

`orbi_find_raw` *Find raw files*

Description

Finds all .raw files in a folder.

Usage

```
orbi_find_raw(folder, pattern = NULL, include_cache = TRUE, recursive = TRUE)
```

Arguments

`folder`	path to a folder with raw files
`pattern`	provide a name pattern to find only specific raw files
`include_cache`	whether to include .raw.cache.zip folders in the absence of the corresponding .raw file so that copies of the cache are read even in the absence of the original raw files
`recursive`	whether to find files recursively

Examples

```
# all .raw files provided with the isoorbi package
orbi_find_raw(system.file("extdata", package = "isoorbi"))
```

`orbi_flag_outliers` *Flag outlier scans*

Description

This function flags outliers using one of the different methods provided by the parameters (to use multiple, please call this function several times sequentially). Note that this function evaluates outliers within each "uidx", "filename", and "injection" (for those of the columns that exist), and additionally within each "block" and "segment" if `by_block = TRUE`. in addition to any groupings already defined before calling this function using dplyr's `group_by()`. It restores the original groupings in the returned datasert.

Usage

```
orbi_flag_outliers(
  dataset,
  agc_fold_cutoff = NA_real_,
  agc_window = c(),
  agc_absolute_cutoff = c(),
  by_block = TRUE
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`agc_fold_cutoff`	flags scans with a fold cutoff based on the average number of ions in the Orbitrap analyzer. For example, `agc_fold_cutoff` = 2 flags scans that have more than 2 times, or less than 1/2 times the average. TIC multiplied by injection time serves as an estimate for the number of ions in the Orbitrap.
`agc_window`	flags scans with a critically low or high number of ions in the Orbitrap analyzer. Provide a vector with 2 numbers `c(x,y)` flagging the lowest x percent and highest y percent. TIC multiplied by injection time serves as an estimate for the number of ions in the Orbitrap.
`agc_absolute_cutoff`	flags scans with a number of ions in the Orbitrap analyzer outside of an absolute range. Provide a vector with 2 numbers `c(x,y)` flagging data below x and above y of the TIC multiplied by injection time (which serves as an estimate for the number of ions in the Orbitrap).
`by_block`	if the `dataset` has block and segment definitions, should the outlier flag be evaluated within each block+segment or globally? default is within each block+segment, switch to globally by turning `by_block = FALSE`

Value

same object as provided in `dataset` with new columns `is_outlier` and `outlier_type` (if they don't already exist) that flags outliers identified by the method and provides the type of outlier (e.g. "2 fold agc cutoff"), respectively.

Examples

```
fpath <- system.file("extdata", "testfile_flow.isox", package = "isoorbi")
df <-
  orbi_read_isox(file = fpath) |>
  orbi_simplify_isox() |>
  orbi_flag_outliers(agc_window = c(1,99))
```

`orbi_flag_satellite_peaks`

Flag minor satellite peaks Flags minor signals for an isotopocule that matches multiple peaks within its exact mass +/- tolerance interval in the same scan. These are often small satellite peaks generated by the Fourier transform. However, if there are satelite peaks of high intensity or very many satellite peaks, it can indicate that the m/z and tolerance setting used for identifying isotopcules need to be revisited. Visualize the flagged satellite peaks with `orbi_plot_satellite_peaks()`.

Description

Flag minor satellite peaks Flags minor signals for an isotopocule that matches multiple peaks within its exact mass +/- tolerance interval in the same scan. These are often small satellite peaks generated by the Fourier transform. However, if there are satelite peaks of high intensity or very many satellite peaks, it can indicate that the m/z and tolerance setting used for identifying isotopcules need to be revisited. Visualize the flagged satellite peaks with `orbi_plot_satellite_peaks()`.

Usage

```
orbi_flag_satellite_peaks(dataset)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)

Value

same object as provided in `dataset` with new column `is_satellite_peak` that flags satellite peaks

Examples

```
fpath <- system.file("extdata", "testfile_flow.isox", package = "isoorbi")
df <-
  orbi_read_isox(file = fpath) |>
  orbi_simplify_isox() |>
  orbi_flag_satellite_peaks()
```

`orbi_flag_weak_isotopocules`

Flag weak isotopocules

Description

This function flags isotopocules that are not detected in a minimum of `min_percent` of scans that then can be easily visualized with `orbi_plot_isotopocule_coverage()`. It evaluates weak isotopocules within each "uidx", "filename", "block", "segment" and "injection" (for those of the columns that exist), in addition to any groupings already defined before calling this function using dplyr's `group_by()`. It restores the original groupings in the returned data.

Usage

```
orbi_flag_weak_isotopocules(dataset, min_percent = 100)
```

Arguments

`dataset`	A simplified IsoX data frame to be processed
`min_percent`	A number between 0 and 100 (inclusive). Isotopocule must be observed in at least this percentage of scans (please note: the percentage is defined relative to the most commonly observed isotopocule of each compound). The default is 100, the most stringent condition to ensure reliable isotpocule coverage and ratio calculations across data blocks. If you lower the default, be mindful of potential misinterprations from using isotopotcules that are very close to their detection limit within a datablock. For continuous flow operations it may be necessary to make data blocks smaller using `orbi_define_block_for_flow_injection()` and `orbi_adjust_block()`.

Value

same object as provided in `dataset` with new column `is_weak_isotopocule` that flags weak isotopocules.

Examples

```
fpath <- system.file("extdata", "testfile_flow.isox", package = "isoorbi")
df <- orbi_read_isox(file = fpath) |>
      orbi_simplify_isox() |>
      orbi_flag_weak_isotopocules(min_percent = 100)
```

`orbi_get_blocks_info` *Summarize blocks info*

Description

This function provides an overview table `blocks_info` which shows information on blocks in the dataset (block number, sample name, data type, scan number and start time where a block starts, and scan number and end time where a block ends).

Usage

```
orbi_get_blocks_info(
  dataset,
  .by = c("uidx", "filename", "injection", "data_group", "block", "sample_name",
    "data_type", "segment")
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`.by`	grouping columns for block info (akin to dplyr's `.by` parameter e.g. in `dplyr::summarize()`). If not set by the user, all columns in the parameter's default values are used, if present in the dataset.

Value

a block summary or if no blocks defined yet, an empty tibble (with warning)

`orbi_get_data` *Get data frame from aggregated data*

Description

Retrieve a specific subset of the aggregated data into a single data frame by specifying which columns to take from each dataset (file_info, scans, peaks, etc.) using `dplyr::select()` syntax. If data from more than one dataset is selected (e.g. some columns from scans AND some from peaks), the datasets are combined with an `dplyr::inner_join()` using the columns listed in by (only the ones actually in the datasets). Joins that would lead to duplicated data entries (i.e. many-to-many joins) are not allowed and will throw an error to avoid unexpected replications of individual datapoints. If you really want to do such a join, you'll have to do it manually.

Usage

```
orbi_get_data(
  aggregated_data,
  file_info = c("filename"),
  scans = NULL,
  peaks = NULL,
  spectra = NULL,
  problems = NULL,
  summary = NULL,
  by = c("uidx", "scan.no")
)
```

Arguments

`aggregated_data`	datasets aggregated from `orbi_aggregate_raw()`
`file_info`	columns to get from the aggregated `file_info`, all `dplyr::select()` syntax is supported
`scans`	columns to get from the aggregated `scans`, all `dplyr::select()` syntax is supported

`peaks`	columns to get from the aggregated `peaks`, all `dplyr::select()` syntax is supported
`spectra`	columns to get from the aggregated `spectra`, all `dplyr::select()` syntax is supported
`problems`	columns to get from the aggregated `problems`, all `dplyr::select()` syntax is supported
`summary`	columns to get from the `summary` calculated via `orbi_summarize_results()`, all `dplyr::select()` syntax is supported. Warning: it is not advisable to combine columns from `summary` with anything other than `file_info` as it will lead to duplicated datasets given that `summary` integrates across multiple scans.
`by`	which columns to look for when joining datasets together. Make sure to include the relevant `by` columns in the selections of the individual datasets so they are joined correctly. The default is usually sufficient

Value

a tibble

`orbi_get_example_files` *Get available example file(s)*

Description

This function will provide the path(s) to example file(s). If a requested file is not yet available locally but is available on https://github.com/isoverse/isodata, it will download it from there into local storage. By default, it will download only cache files (.raw.cache.zip) instead of the original .raw files because the cache files are significantely smaller. Todownload the original raw files instead, use `download_raw_files = TRUE`.

Usage

```
orbi_get_example_files(
  filenames,
  download_raw_files = FALSE,
  download_always = FALSE
)
```

Arguments

`filenames`	names of the example files
`download_raw_files`	should the original raw files be downloaded? By default only cache files (raw.cache.zip) are downloaded as they are usually much smaller. However, they will not work for retrieving additional spectra. To download the original spectra, switch to `download_raw_files = TRUE`
`download_always`	whether to download files anew even if they already exist locally

Value

file path(s) that can be passed directly to `orbi_read_raw()`

`orbi_get_problems` *Retrieve parsing problems*

Description

This function retrieves parsing problems encountered during the reading and processing of files.

This function prints out parsing problems encountered during the reading and processing of files.

Usage

```
orbi_get_problems(obj, strip_ansi = TRUE)

orbi_show_problems(obj)
```

Arguments

`obj`	data object that holds problems information
`strip_ansi`	whether to remove ansi characters from the message, yes by default

Value

tibble data frame with a list of problems encountered during processing

---

`orbi_get_settings` *Get all isoorbi package settings*

Description

[Deprecated]

`orbi_get_settings()` was renamed `orbi_get_options()` as part of `isoorbi` switching from 'settings' to 'options' to be consistent with base R naming conventions

Usage

```
orbi_get_settings(pattern = NULL)
```

Arguments

`pattern`	passed on to `orbi_get_options()`

`orbi_identify_isotopocules` *Identify isotopocules*

Description

Map the mass spectral peaks to specific isotopocules based on their mass.

Usage

```
orbi_identify_isotopocules(
  aggregated_data,
  isotopocules,
  default_tolerance = 1,
  default_charge = 1
)
```

Arguments

`aggregated_data`	either data aggregated from `orbi_aggregate_raw()` or a straight-up tibble data frame of the peaks (e.g. retrieved via `orbi_get_data(peaks = everything())`).
`isotopocules`	list of isotopocules to map, can be a data frame/tibble, a named vector such as `c("M0" = 61.9878, "15N" = 62.9850)`, or the name of a file to read from (.csv/.tsv/.xlsx are all supported). If provided as a tibble/file, the required columns are `isotopocule/isotopolog` and `mz/mass` (these alternative names for the columns, including uppercase versions, are recognized automatically). In addition, `tolerance/tolerance [mmu]/tolerance [mDa]`, `charge/z`, `#compound/compound`, and `fragment` are recognized, as well as any other (arbitrarily named) columns with additional information. Character columns in the `isotopocules` table (including `isotopocule` and `compound`) are turned into factors with levels that preserve the order of isotopocules. That means that to change the order of isotopocules in downstream plotting functions, make sure to list them in the order you'd like them presented in. Note that if `tolerance/tolerance [mmu]/tolerance [mDa]` or `charge/z` are not provided, the values in the parameters `default_tolerance` and `default_charge` are used, respectively.
`default_tolerance`	tolerance (in mmu) to be used for isotopocule identification if a `tolerance/tolerance [mmu]/tolerance [mDa]` column is not included in `isotopocules`
`default_charge`	charge to be used for any unidentified peak, and if a `charge/z` column is not included in `isotopocules`

Value

same object as provided in `aggregated_data` with added columns compound (if provided), `itc_uidx` (introduced unique isotopocule index), `isotopocule`, `mzExact`, `charge`, and `ions.incremental` (via `orbi_calculate_ions()`), as as well as any other additional information columns provided in `isotopocules`. Note that if the default `CN` and `RN` values of `orbi_calculate_ions()` are not the ones that should be used, simply run `orbi_calculate_ions()` explicitly afterwards. Also note that the information about columns that were NOT aggregated in previous steps is purposefully not preserved at this step.

`orbi_options` *Package options*

Description

These options are best set via `orbi_options()` and queried via `orbi_get_option()`. However, the base functions `options()` and `getOption()` work as well but require an `isoorbi.` prefix (the package name and a dot) for the option name. Setting an option to a value of `NULL` means that the default is used. `orbi_get_options()` is available as an additional convenience function to retrieve a subset of options with a regular expression pattern.

Usage

```
orbi_options(...)

orbi_get_options(pattern = NULL)

orbi_get_option(x)
```

Arguments

`...`	set package options, syntax identical to `options()`
`pattern`	to retrieve multiple options (as a list) with a shared pattern
`x`	name of the specific option to retrieve

Functions

- `orbi_options()`: set/get option values
- `orbi_get_options()`: get a subset of option values that fit a pattern
- `orbi_get_option()`: retrieve the current value of one option (option must be defined for the package)

Options for the isoorbi package

- `di_ref_name`: the text label for dual inlet reference blocks
- `di_sample_name`: the text label for dual inlet sample blocks
- `data_type_data`: the text used to flag raw data as actually being data
- `data_type_startup`: the text used to flag raw data as being part of the startup
- `data_type_changeover`: the text used to flag raw data as being part of a changeover

- `data_type_unused`: the text used to flag raw data as being unused
- `aggregators`: data aggregators for pulling data out of raw files. The list of available aggregators is accessible via `orbi_get_option("aggregators")`. Individiual aggregators are available via the shortcut helper function `orbi_get_aggregator("standard")`. Register new/overwrite existing aggregators via `orbi_register_aggregator()`.
- debug: turn on debug mode
- `auto_use_ansi`: whether to automatically enable correct rendering of stylized (ansi) output in HTML reports from notebooks that call `library(isoorbi)`. Can be turned off by calling `isoorbi::orbi_options(auto_use_ansi = FALSE)` **before** call `library(isoorbi)`.

Examples

```
# All default options
orbi_get_options()

# Options that contain 'data' in the name
orbi_get_options("data")

# Specific option
orbi_get_option("data_type_unused")

# Change an option
orbi_options(data_type_unused = "flagged")
orbi_get_option("data_type_unused")

# Change back to default
orbi_options(data_type_unused = NULL)
orbi_get_option("data_type_unused")
```

`orbi_plot_isotopocule_coverage`

Isotopocule coverage

Description

The coverage of each isotopcule across scans/time is an important indicator for data completeness. These functions provide ways to summarize and visualize the isotopocule coverage in a dataset.

Usage

```
orbi_plot_isotopocule_coverage(
  dataset,
  isotopocules = c(),
  x = c("scan.no", "time.min"),
  x_breaks = scales::breaks_pretty(5),
  add_data_blocks = TRUE
)
```

```
orbi_get_isotopocule_coverage(dataset)
```

Arguments

`dataset`	a data frame or aggregated dataset with satellite peaks already identified (i.e. after `orbi_flag_satellite_peaks()`)
`isotopocules`	which isotopocules to visualize, if none provided will visualize all (this may take a long time or even crash your R session if there are too many isotopocules in the data set)
`x`	x-axis column for the plot, either "time.min" or "scan.no", default is "scan.no"
`x_breaks`	what breaks to use for the x axis, change to make more specifid tickmarks
`add_data_blocks`	add highlight for data blocks if there are any block definitions in the dataset (uses `orbi_add_blocks_to_plot()`). To add blocks manually, set `add_data_blocks = FALSE` and manually call the `orbi_add_blocks_to_plot()` function afterwards.

Value

a ggplot object

summary data frame

Functions

- `orbi_plot_isotopocule_coverage()`: visualizes isotope coverage. Weak isotopocules (if previously defined by `orbi_flag_weak_isotopocules()`) are highlighted in red.
- `orbi_get_isotopocule_coverage()`: calculates which stretches of the data have data for which isotopocules. This function is usually used indicrectly by `orbi_plot_isotopocule_coverage()` but can be called directly to investigate isotopocule coverage.

`orbi_plot_raw_data` *Visualize data*

Description

Call this function to visualize orbitrap data vs. time or scan number. The most common uses are `orbi_plot_raw_data(y = intensity)`, `orbi_plot_raw_data(y = ratio)`, and `orbi_plot_raw_data(y = tic * it.ms)`. If the selected `y` is peak-specific data (rather than scan-specific data like `tic * it.ms`), the `isotopocules` argument can be used to narrow down which isotopocules will be plotted. By default includes all isotopcules that have not been previously identified by `orbi_flag_weak_isotopcules()` (if already called on dataset).

Usage

```
orbi_plot_raw_data(
  dataset,
  isotopocules = c(),
  x = c("time.min", "scan.no"),
  x_breaks = scales::breaks_pretty(5),
  y,
  y_scale = c("raw", "linear", "pseudo-log", "log"),
  y_scale_sci_labels = TRUE,
  color = .data$isotopocule,
 colors = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#66A61E", "#E6AB02", "#A6761D",
    "#666666", "#BBBBBB"),
  color_scale = scale_color_manual(values = colors),
  add_data_blocks = TRUE,
  add_all_blocks = FALSE,
  show_outliers = TRUE
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`isotopocules`	which isotopocules to visualize, if none provided will visualize all (this may take a long time or even crash your R session if there are too many isotopocules in the data set)
`x`	x-axis column for the plot, either "time.min" or "scan.no", default is "scan.no"
`x_breaks`	what breaks to use for the x axis, change to make more specifid tickmarks
`y`	expression for what to plot on the y-axis, e.g. `intensity`, `tic * it.ms` (pick one `isotopocules` as this is identical for different istopocules), `ratio`. Depending on the variable, you may want to adjust the `y_scale` and potentially `y_scale_sci_labels` argument.
`y_scale`	what type of y scale to use: "log" scale, "pseudo-log" scale (smoothly transitions to linear scale around 0), "linear" scale, or "raw" (if you want to add a y scale to the plot manually instead)
`y_scale_sci_labels`	whether to render numbers with scientific exponential notation
`color`	expression for what to use for the color aesthetic, default is isotopocule
`colors`	which colors to use, by default a color-blind friendly color palettes (RColorBrewer, dark2)
`color_scale`	use this parameter to replace the entire color scale rather than just the `colors`
`add_data_blocks`	add highlight for data blocks if there are any block definitions in the dataset (uses `orbi_add_blocks_to_plot()`). To add blocks manually, set `add_data_blocks = FALSE` and manually call the `orbi_add_blocks_to_plot()` function afterwards.

`add_all_blocks`	add highlight for all blocks, not just data blocks (equivalent to the `data_only = FALSE` argument in `orbi_add_blocks_to_plot()`)
`show_outliers`	whether to highlight data previously flagged as outliers by `orbi_flag_outliers()`

Value

a ggplot object

`orbi_plot_satellite_peaks`

Visualize satellite peaks

Description

Call this function any time after flagging the satellite peaks to see where they are. Use the `isotopocules` argument to focus on the specific isotopocules of interest.

Usage

```
orbi_plot_satellite_peaks(
  dataset,
  isotopocules = c(),
  x = c("scan.no", "time.min"),
  y = c("ions.incremental", "intensity"),
  x_breaks = scales::breaks_pretty(5),
  y_scale = c("log", "pseudo-log", "linear", "raw"),
  y_scale_sci_labels = TRUE,
 colors = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#66A61E", "#E6AB02", "#A6761D",
    "#666666", "#BBBBBB"),
  color_scale = scale_color_manual(values = colors)
)
```

Arguments

`dataset`	a data frame or aggregated dataset with satellite peaks already identified (i.e. after `orbi_flag_satellite_peaks()`)
`isotopocules`	which isotopocules to visualize, if none provided will visualize all (this may take a long time or even crash your R session if there are too many isotopocules in the data set)
`x`	x-axis column for the plot, either "time.min" or "scan.no", default is "scan.no"
`y`	y-axis column for the plot, typially either "ions.incremental" or "intensity", default is "ions.incremental" (falls back to "intensity" if "ions.incremental" has not been calculated yet for the provided dataset)
`x_breaks`	what breaks to use for the x axis, change to make more specifid tickmarks

`y_scale`	what type of y scale to use: "log" scale, "pseudo-log" scale (smoothly transitions to linear scale around 0), "linear" scale, or "raw" (if you want to add a y scale to the plot manually instead)
`y_scale_sci_labels`	whether to render numbers with scientific exponential notation
`colors`	which colors to use, by default a color-blind friendly color palettes (RColorBrewer, dark2)
`color_scale`	use this parameter to replace the entire color scale rather than just the `colors`

Value

a ggplot object

`orbi_plot_shot_noise` *Make a shot noise plot*

Description

This function creates a shot noise plot using a `shotnoise` data frame created by the `orbi_analyze_shot_noise()` function.

Usage

```
orbi_plot_shot_noise(
  shotnoise,
  x = c("time.min", "n_effective_ions"),
  permil_target = NA_real_,
  color = "ratio_label",
  colors = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#66A61E", "#E6AB02", "#A6761D",
    "#666666")
)
```

Arguments

`shotnoise`	a `shotnoise` data frame
`x`	x-axis for the shot noise plot, either "time.min" or "n_effective_ions"
`permil_target`	highlight the target permil in the shotnoise plot
`color`	which column to use for the color aesthetic (must be a factor)
`colors`	which colors to use, by default a color-blind friendly color palettes (RColorBrewer, dark2)

Details

plot shot noise

Value

a ggplot object

`orbi_plot_spectra` *Plot mass spectra*

Description

This function visualizes mass spectra from aggregated raw file data. The spectra have to be be previously read in with `include_spectra = c(1, 10, 100)` in `orbi_read_raw()`. By default, this function tries to visualize different isotopcule ranges (monoisotopic peak, M+1, M+2, M+3). To focus only on isotopcules of interest, run `orbi_identify_isotopocules()` and `orbi_filter_isotopocules()` first.

Usage

```
orbi_plot_spectra(
  aggregated_data,
  mz_min = 0,
  mz_max = Inf,
  mz_base_peak = NULL,
  mz_focus_nominal_offsets = 0:4,
  max_scans = 6,
  max_files = 4,
  label_peaks = TRUE,
  show_filenames = TRUE,
  show_ref_and_lock_peaks = TRUE,
  show_focus_backgrounds = TRUE,
 background_colors = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#66A61E", "#E6AB02",
    "#A6761D", "#666666", "#BBBBBB")
)
```

Arguments

`aggregated_data`	data aggregated by `orbi_aggregate_raw()` and, optionally, with isotopocules already identified by `orbi_identify_isotopocules()`, and (also optionally), alreadty filtered with `orbi_filter_isotopocules()`
`mz_min`	which mz to start the main plot window at. By default include all.
`mz_max`	which mz to end the main plot window at. By default include all.
`mz_base_peak`	where is the base peak at (approximately)?. If not specified (the default), takes the largest peak in the `mz_min` to `mz_max` window.
`mz_focus_nominal_offsets`	which panels to visualize? 0 = whole spectrum, 1 = spectrum around monoisotopic peak + 1 mu (M+1), 2 = M+2, etc. By default includes the whole spectrum and up to +1, +2, +3, and +4 peaks (if they exist). To visualize only the whole spectrum, use `mz_focus_nominal_offsets = 0`. Likewise, to visualize only the area around the monoisotopic peak +1, provide `mz_focus_monimal_offsets = 1` (or = `c(1, 2)` for both +1 and +2 windows).

`max_scans`	spectra from how many scans to show at most. By default up to 6 (the number of available linetypes). To show only the spectrum from a single scan, set `max_scans = 1`. If more than 6 scan spectra are allowed (and there are more than 6 loaded in the `aggregated_data`), turns of the linetype aesthetic.
`max_files`	spectra from how many files to show at most. Each file is shown as an additional line of panels.
`label_peaks`	whether to label the peaks in the M+1/2/3 panels. If isotopcules are already identified from `orbi_identify_isotopocules()`, uses the isotopcule names, otherwise the m/z values. Peaks that are missing (identified by `orbi_identify_isotopocules()`) in all spectra are highlighted in red. To avoid labeling unidentified/missing peaks, run `orbi_filter_isotopocules()` first.
`show_filenames`	whether to show the filename in the first panel of reach row (usually the full spectrum panel)
`show_ref_and_lock_peaks`	whether to show reference and lock mass peaks in the spectrum
`show_focus_backgrounds`	whether to highlight the M+x panels with specific background colors that match them with the mass bands highlighted in the first panel
`background_colors`	the colors to use for the background highlighting

`orbi_read_isox` *Read IsoX file*

Description

Read an IsoX dataput file (`.isox`) into a tibble data frame.

Usage

```
orbi_read_isox(file)
```

Arguments

`file`	Path to the `.isox` file(s), single value or vector of paths

Details

Additional information on the columns:

- `filename`: name of the original Thermo `.raw` file processed by IsoX
- `scan.no`: scan number
- `time.min`: acquisition or retention time in minutes
- `compound`: name of the compound (e.g., NO3-)
- `isotopocule`: name of the isotopocule (e.g., 15N); called `isotopolog` in `.isox`

- `ions.incremental`: estimated number of ions, in increments since it is a calculated number
- `tic`: total ion current (TIC) of the scan
- `it.ms`: scan injection time (IT) in millisecond (ms)

Value

A tibble containing at minimum the columns `filename`, `scan.no`, `time.min`, `compound`, `isotopocule`, `ions.incremental`, `tic`, `it.ms`

Examples

```
fpath <- system.file("extdata", "testfile_dual_inlet.isox", package = "isoorbi")
df <- orbi_read_isox(file = fpath)
```

`orbi_read_raw` *Read RAW files*

Description

Read raw data files (`.raw`) from Orbitrap IRMS runs directly. This function extracts all available information and thus can be relatively slow (~1s / Mb on a typical personal computer) but with the caching this is only true the first time. The results can be used directly or, more typically, are aggregated with `orbi_aggregate_raw()` to safely extract the relevant information for downstream processing. This function is designed to be fail save by safely catching errors and reporting back on them (see `orbi_get_problems()`).

Usage

```
orbi_read_raw(
  file_paths,
  show_progress = rlang::is_interactive(),
  show_problems = TRUE,
  include_spectra = FALSE,
  read_cache = TRUE,
  cache = TRUE,
  cache_spectra = cache,
  keep_cached_spectra = cache
)
```

Arguments

`file_paths`	paths to the `.raw` file(s), single value or vector of paths. Use `orbi_find_raw()` to get all raw files in a folder.
`show_progress`	whether to show a progress bar, by default always enabled when running interactively e.g. inside RStudio (and disabled in a notebook), turn off with `show_progress = FALSE`

`show_problems`	whether to show problems encountered along the way (rather than just keeping track of them with `orbi_get_problems()`). Set to `show_problems = FALSE` to turn off the live printout. Either way, all encountered problems can be retrieved with running `orbi_get_problems()` for the returned list
`include_spectra`	whether to include the spectral data from specific scans (e.g. `include_spectra = c(5, 100, 200)` reads out the spectra from scans 5, 100, and 200 for each file if they exist) or from all scans (`include_spectra = TRUE`). Including many or all scan spectra makes the read process slower (especially if `cache_spectra = FALSE`) and the returned data frame tibble significantely larger. The default is `FALSE` (i.e. scan spectra are not returned).
`read_cache`	whether to read the file from cached .parquet files (if they exist) or anew
`cache`	whether to automatically cache the read raw files (writes highly efficient .parquet files in a folder with the same name as the file .cache appended)
`cache_spectra`	whether to automatically cache requested scan spectra (this can take up significant disc space), by default the same as `cache`
`keep_cached_spectra`	whether to keep the spectra from a raw file that were previously cached whenever `include_spectra` changes and requires reading the file anew. Having this TRUE (the default) makes it faster to iterate on code that changes which spectra to read but leads to larger cache files.

Value

a tibble data frame where each row holds the file path and nested tibbles of datasets extracted from the raw file (typically `file_info`, `scans`, `peaks`, and `spectra`). This is the safest way to extract the data without needing to make assumptions about compatibility across files. Extract your data of interest from the tibble columns or use `orbi_aggregate_raw()` to extract safely across files.

`orbi_segment_blocks` *Segment data blocks*

Description

This step is optional and is intended to make it easy to explore the data within a sample or ref data block. Note that any raw data not identified with `data_type` set to "data" (`orbi_get_option("data_type_data")`) will stay unsegmented. This includes raw data flagged as "startup", "changeover", and "unused".

Usage

```
orbi_segment_blocks(
  dataset,
  into_segments = NULL,
  by_scans = NULL,
  by_time_interval = NULL
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`into_segments`	segment each data block into this many segments. The result will have exactly this number of segments for each data block except for if there are more segments requested than observations in a group (in which case each observation will be one segment)
`by_scans`	segment each data block into segments spanning this number of scans. The result will be approximately the requested number of scans per segment, depending on what is the most sensible distribution of the data. For example, in a hypothetical data block with 31 scans, if by_scans = 10, this function will create 3 segments with 11, 10 and 10 scans each (most evenly distributed), instead of 4 segments with 10, 10, 10, 1 (less evenly distributed).
`by_time_interval`	segment each data block into segments spanning this time interval. The result will have the requested time interval for all segments except usually the last one which is almost always shorter than the requested interval.

`orbi_set_settings` *Set package settings*

Description

[Deprecated]

`orbi_set_settings()` was renamed `orbi_options()` as part of `isoorbi` switching from 'settings' to 'options' to be consistent with base R naming conventions

Usage

```
orbi_set_settings(...)
```

Arguments

`...`	named arguments to set specific options, passed on to `orbi_options()`

`orbi_simplify_isox` *Simplify IsoX data*

Description

Keep only columns that are directly relevant for isotopocule ratio analysis. This function is optional and does not affect any downstream function calls.

Usage

```
orbi_simplify_isox(dataset, add = c())
```

Arguments

`dataset`	IsoX data that is to be simplified
`add`	additional columns to keep

Value

A tibble containing only the 9 columns: `filepath`, `filename`, `scan.no`, `time.min`, `compound`, `isotopocule`, `ions.incremental`, `tic`, `it.ms`, plus any additional columns defined in the `add` argument

Examples

```
fpath <- system.file("extdata", "testfile_flow.isox", package="isoorbi")
df <- orbi_read_isox(file = fpath) |> orbi_simplify_isox()
```

`orbi_start_aggregator` *Dynamic data agreggator*

Description

These functions allow definition of custom data aggregators for processing data extracted from raw files. An aggregator is run on each imported file and pulls together the relevant data users are interested in while making sure data formats are correct so that the aggregated data can be merged across several imported files for fast downstream processing.

Usage

```
orbi_start_aggregator(name)

orbi_add_to_aggregator(
  aggregator,
  dataset,
  column,
  source = column,
  default = NA,
  cast = "as.character",
  regexp = FALSE,
  func = NULL,
  args = NULL
)

orbi_register_aggregator(aggregator, name = attr(aggregator, "name"))

orbi_get_aggregator(name)
```

Arguments

`name`	a descriptive name for the aggregator. This name is automatically used as the default name when registering the aggregator via `orbi_register_aggregator()`.
`aggregator`	the aggregator table generated by `orbi_start_aggregator()` or passed from a previous call to `orbi_add_to_aggregator()` for constructing the entire aggregator by piping
`dataset`	the name of the dataset to aggregate from (`file_info`, `scans`, `peaks`, `spectra`)
`column`	the name of the column in which data should be stored
`source`	single character column name or vector of column names (if alternatives could be the source) where in the `dataset` to find data for the `column`. If a vector of multiple column names is provided (e.g. `source = c("a1", "a2")`), the first column name that's found during processing of a dataset will be used and passed to the function defined in `func` (if any) and then the one defined in `cast`. To provide multiple parameters from the data to `func`, define a list instead of a vector `source = list("a", "b", "c")` or if multiple alternative columns can be the source for any of the arguments, define as `source = list(c("a1", "a2"), "b", c("c1", "c2", "c3"))`
`default`	the default value if no `source` columns can be found or another error is encountered during aggregatio. Note that the `default` value will also be processed with the function in `cast` to make sure it has the correct data type.
`cast`	what to cast the values of the resulting column to, most commonly `"as.character"`, `"as.integer"`, `"as.numeric"`, or `"as.factor"`. This is required to ensure all aggregated values have the correct data type.
`regexp`	whether source columm names should be interpreted as a regular expressions for the purpose of finding the relevant column(s). Note if `regexp = TRUE`, the search for the source column always becomes case-insensitive so this can also

	be used for a direct match of a source column whose upper/lower casing can be unreliable. If a column is matched by a regexp and also by a direct aggregator rule, the direct aggregator rule takes precedence.
`func`	name of a processing function to apply before casting the value with the `cast` function. This is optional and can be used to conduct more elaborate preprocessing of a data or combining data from multiple source columns in the correct way (e.g. pasting together from multiple columns).
`args`	an optional list of arguments to pass to the `func` in addition to the values coming from the source colummn(s)

Value

an orbi aggregator tibble

Functions

- `orbi_start_aggregator()`: starts the aggregator
- `orbi_add_to_aggregator()`: add additional column to aggregate data for. Overwrites an existing aggregator entry for the same dataset and column if it already exists.
- `orbi_register_aggregator()`: register an aggregator in the isoorbi options so it can be retrieved with `orbi_get_aggregator()`
- `orbi_get_aggregator()`: retrieve a registered aggregator (get all aggregators with `orbi_get_option("aggregators`

`orbi_summarize_results`

Generate the results table

Description

Contains the logic to generate the results table. It passes the `ratio_method` parameter to the `orbi_calculate_summarized_ratio()` function for ratio calculations.

Usage

```
orbi_summarize_results(
  dataset,
  ratio_method = c("mean", "sum", "median", "geometric_mean", "slope", "weighted_sum"),
  .by = c("block", "sample_name", "segment", "data_group", "data_type", "injection"),
  include_flagged_data = FALSE,
  include_unused_data = FALSE
)
```

Arguments

`dataset`	An aggregated dataset or a data frame of peaks (i.e. works directly after `orbi_identify_isotopocules(` as well as with a tibble from orbi_get_data(peaks = everything()) or when reading from an IsoX file)
`ratio_method`	Method for computing the ratio. **Please note well**: the formula used to calculate ion ratios matters! Do not simply use arithmetic mean. The best option may depend on the type of data you are processing (e.g., MS1 versus M+1 fragmentation). `ratio_method` can be one of the following: - `mean`: arithmetic mean of ratios from individual scans. - `sum`: sum of all ions of the numerator across all scans divided by the sum of all ions observed for the denominator across all scans. - `geometric_mean`: geometric mean of ratios from individual scans. - `slope`: The ratio is calculated using the slope obtained from a linear regression model that is weighted by the `numerator x`, using `stats::lm(x ~ y + 0, weights = x)`. - `weighted_sum`: A derivative of the `sum` option. The weighing function ensures that each scan contributes equal weight to the ratio calculation, i.e. scans with more ions in the Orbitrap do not contribute disproportionately to the total `sum` of `x` and `y` that is used to calculate `x/y`.
`.by`	additional grouping columns for the results summary (akin to dplyr's `.by` parameter e.g. in `dplyr::summarize()`). If not set by the user, all columns in the parameter's default values are used, if present in the dataset. Note that the order of these is also used to arrange the summary.
`include_flagged_data`	whether to include flagged data in the calculations (FALSE by default)
`include_unused_data`	whether to include unused data in the calculations (FALSE by default), in addition to peaks actually flagged as orbi_get_option("data_type_data")

Value

Returns a results summary table (as tibble if `dataset` is a tibble, as `dataset$tibble` if `dataset` is aggregated raw data) with the columns `filename`, `compound`, `isotopocule` and `basepeak` as well as the grouping columns from the `.by` parameter that are part of the input `dataset`. Additionally this function adds the following results columns: `start_scan.no`, `end_scan.no`, `start_time.min`, `mean_time.min`, `end_time.min`, `ratio`, `ratio_sem`, `ratio_relative_sem_permil`, `shot_noise_permil`, `No.of.Scans`, `minutes_to_1e6_ions`

- `ratio`: The isotope ratio between the `isotopocule` and the `basepeak`, calculated using the `ratio_method`
- `ratio_sem`: Standard error of the mean for the ratio
- `number_of_scans`: Number of scans used for the final ratio calculation
- `minutes_to_1e6_ions`: Time in minutes it would take to observe 1 million ions of the `isotopocule` used as numerator of the ratio calculation.
- `shot_noise_permil`: Estimate of the shot noise (more correctly thermal noise) of the reported ratio in permil.
- `ratio_relative_sem_permil`: Relative standard error of the reported ratio in permil

Examples

```
fpath <- system.file("extdata", "testfile_flow.isox", package = "isoorbi")
df <- orbi_read_isox(file = fpath) |>
      orbi_simplify_isox() |>
      orbi_define_basepeak("M0")  |>
      orbi_summarize_results(ratio_method = "sum")
```

Index