Package 'isoorbi'

November 9, 2023

Type Package

Title Process Orbitrap Isotopocule Data

Version 1.3.0

Date 2023-11-07

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

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

Depends R (>= 4.1.0)

Imports utils (>= 4.1.0), stats (>= 4.1.0), rlang (>= 1.0.0), lifecycle (>= 1.0.0), tidyr (>= 1.2.0), dplyr (>= 1.1.1), ggplot2 (>= 3.4.0), scales (>= 1.2.1), readr (>= 2.1.0), tidyselect (>= 1.2.0), openxlsx, purrr, methods

Suggests devtools, knitr, rmarkdown, testthat, forcats

Description Read and process isotopocule data from an Orbitrap Isotope Solutions mass spectrometer. Citation: Kantnerova et al. (in review).

License MIT + file LICENSE

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2023-11-09 08:10:02 UTC

R topics documented:

`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" (setting("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 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`	tibble produced by `orbi_define_blocks_for_dual_inlet()`
`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_settings("data_type_unused")`. Any previously applied segmentation will be discarded (`segment` column set to `NA`) to avoid unintended side effects.

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	a data frame output after running orbi_define_basepeak()
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_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_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`	A tibble from a `IsoX` output. Needs to contain columns for `filename`, `compound`, `scan.no`, `isotopocule`, and `ions.incremental`.
`basepeak_def`	The isotopocule that gets defined as base peak, i.e. the denominator to calculate ratios

Value

Input data frame 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 = setting("di_ref_name"),
  sample_block_name = setting("di_sample_name")
)
```

Arguments

`dataset`	A data frame or tibble produced from IsoX data by `orbi_simplify_isox()`
`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_settings("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`	tibble with Orbitrap data
`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_settings("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 frame to excel

Description

This functions exports the final `dataset` into an Excel file.

Usage

```
orbi_export_data_to_excel(
  dataset,
  file,
  dbl_digits = 7,
  int_format = "0",
  dbl_format = sprintf(sprintf("%%.%df", dbl_digits), 0)
)
```

Arguments

`dataset`	data frame
`file`	file path to export the file
`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)

Value

returns dataset invisibly for use in pipes

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_isox *Basic generic filter for IsoX data*

Description

A basic filter function orbi_filter_isox() for file names, isotopocules, compounds and time ranges. Default value for all parameters is NULL, i.e. no filter is applied.

Usage

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

Arguments

dataset	The IsoX data to be filtered
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_isox(
    filenames = c("s3744"),
    compounds = "HSO4-",
    isotopocules = c("M0", "34S", "18O")
  )
```

---

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_flag_outliers` *Flag outlier scans*

Description

The function `orbi_flag_outliers()` 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 "filename", "block", "segment" and "injection" (if these columns 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 frame.

Usage

```
orbi_flag_outliers(dataset, agc_fold_cutoff = NA_real_, agc_window = c())
```

Arguments

`dataset`	Simplified IsoX dataset to have outliers flagged
`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.

Details

Function is intended to flag scans that are outliers.

The input `dataset` is expected to have at least these 8 columns: `filename`, `scan.no`, `time.min`, `compound`, `isotopocule`, `ions.incremental`, `tic`, `it.ms`.

Value

A data frame 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*

Description

Flag minor signals (e.g., satellite peaks) that were reported by IsoX (filter them out with `orbi_filter_flagged_data()`).

Usage

```
orbi_flag_satellite_peaks(dataset)
```

Arguments

`dataset`	A data frame or tibble produced from IsoX data by `orbi_simplify_isox()`

Details

The `orbi_filter_satellite_peaks()` function removes minor signals for an isotopocule that have been reported by IsoX. These are often small `satellite peaks` generated by the Fourier transform.

If there are signal of high intensity or very many signals, this can indicate that the m/z and tolerance setting used for processing .raw files with IsoX were incorrect.

Value

A data frame 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

The function orbi_filter_weak_isotopocules() flags isotopocules that are not detected in a minimum of min_percent of scans. This function evaluates weak isotopocules within each "filename", "block", "segment" and "injection" (if these columns 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 frame.

Usage

```
orbi_flag_weak_isotopocules(dataset, min_percent)
```

Arguments

dataset	A simplified IsoX data frame to be processed
min_percent	A number between 0 and 90. 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)

Details

The input dataset is expected to have at least these 8 columns: filename, scan.no, time.min, compound, isotopocule, ions.incremental, tic, it.ms.

Value

A data frame 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 = 2)
```

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("filename", "injection", "data_group", "block", "sample_name", "data_type",
    "segment")
)
```

Arguments

`dataset`	tibble produced by `orbi_define_blocks_for_dual_inlet()`
`.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_isotopocule_coverage *Calculate isotopocule coverage*

Description

Calculate 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.

Usage

```
orbi_get_isotopocule_coverage(dataset)
```

Arguments

`dataset`	A data frame or tibble produced from IsoX data

Value

summary data frame

`orbi_get_settings` *Get all isoorbi package settings*

Description

Get all isoorbi package settings

Usage

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

Arguments

`pattern`	an optional parameter with a regular expression pattern by which to sub-select the returned settings

Value

list of all package settings and their values

Examples

```
orbi_get_settings()
```

`orbi_plot_isotopocule_coverage`
Plot isotopocule coverage

Description

Weak isotopocules (if previously defined by `orbi_flag_weak_isotopocules()`) are highlighted in the `weak_isotopocules_color`.

Usage

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

Arguments

dataset	isox data
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"
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

---

orbi_plot_raw_data *Visualize raw 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). By default includes all isotopcules that have not been previously identified by orbi_flag_weak_isotopcules() (if already called on dataset). To narrow down the isotopocules to show, use the isotopocule parameter.

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"),
  color_scale = scale_color_manual(values = colors),
  add_data_blocks = TRUE,
  add_all_blocks = FALSE,
  show_outliers = TRUE
)
```

Arguments

dataset	isox dataset
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"
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"),
  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"),
  color_scale = scale_color_manual(values = colors)
)
```

Arguments

dataset	isox dataset with satellite peaks identified (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"
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_read_isox *Read IsoX file*

Description

Read an IsoX output 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_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_settings("data_type")`) 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	tibble produced by orbi_define_blocks_for_dual_inlet()
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

Use this function to change the default package settings. When calling this function, only specify the settings you want to change, everything else will remain unchanged. The default value for each parameter is what the package uses by default for each setting.

Usage

```
orbi_set_settings(
  di_ref_name = "ref",
  di_sample_name = "sam",
  data_type_data = "data",
  data_type_startup = "startup",
  data_type_changeover = "changeover",
  data_type_unused = "unused",
  reset_all = FALSE
)
```

Arguments

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

`reset_all`
: if set to TRUE, will reset all settings back to their defaults

Details

FIXME: needs documentation completion FIXME: needs tests to change settings and then get the value back

Value

invisible list of all settings (see `orbi_get_settings()`)

---

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_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`	A tibble from `IsoX` output (`orbi_read_isox()`) and with a basepeak already defined (using `orbi_define_basepeak()`). Optionally, with block definitions (`orbi_define_blocks_for_dual_inlet()`) or even additional block segments (`orbi_segment_blocks()`).
`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 setting("data_type_data")

Value

Returns a results summary table retaining 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