Package 'lcmsPlot'

August 4, 2026

Type Package

Title Comprehensive Liquid Chromatography-Mass Spectrometry (LC-MS) data visualisation package

Version 1.1.8

Description lcmsPlot is an R package designed for visualising Liquid Chromatography-Mass Spectrometry (LC-MS) data with publication-ready high-quality plots.
The package enables users to generate and customise chromatograms, mass traces, spectra, and more with fine-tuned aesthetics and annotation options.

License GPL-3

Encoding UTF-8

LazyData false

Depends R (>= 4.4.0)

Imports methods, rlang, dplyr, tibble, tidyr, BiocParallel, MSnbase, xcms, MsExperiment, mzR, Spectra, MsBackendMsp, S4Vectors, ggplot2, scales, patchwork, DBI, RSQLite, jsonlite

Suggests knitr, rmarkdown, BiocStyle, openxlsx, faahKO, rawrr, msPurity, ggnewscale, htmltools, shiny, shinytoastr, shinytest2, testthat (>= 3.0.0)

Collate 'lcmsPlot-package.R' 'zzz.R' 'helpers-data.R' 'helpers-s4.R' 'raw-file-reader.R' 'xcms-raw-list.R' 'options.R' 'processing-xcms.R' 'data-source.R' 'data-source-compound-discoverer.R' 'data-source-mzmine.R' 'data-source-ms-dial.R' 'data-source-compound-discoverer-node.R' 'data-source-lipid-search.R' 'creators-purity.R' 'constructors-chromatograms.R' 'constructors-spectra.R' 'data-validation.R' 'data-adapters.R' 'plot-units.R' 'plot-chrom-peak-rects.R' 'plot-chromatogram.R' 'plot-mass-trace.R' 'plot-spectrum.R' 'plot-total-ion-current.R' 'plot-intensity-map.R' 'plot-peak-density.R' 'plot-peak-count-image.R' 'plot-rt-diff.R' 'plot-purity.R' 'plot-variants.R' 'plot.R' 'lcmsPlotDataContainer-class.R' 'creators-chromatograms.R' 'creators-intensity-map.R' 'creators-peak-density.R' 'creators-peak-count-image.R' 'creators-rt-diff.R'

'creators-spectra.R' 'creators-total-ion-current.R'
'lcmsPlot-class.R' 'app-utils.R' 'app-options.R'
'app-module-chromatogram.R' 'app-module-peak-density.R'
'app-module-spectra.R' 'app-module-tic.R' 'app-interface.R'
'app-launcher.R'

VignetteBuilder knitr

URL https://github.com/computational-metabolomics/lcmsPlot

Roxygen list(markdown = TRUE)

biocViews Metabolomics, MassSpectrometry

BugReports https://github.com/computational-metabolomics/lcmsPlot/issues

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

git_url https://git.bioconductor.org/packages/lcmsPlot

git_branch devel

git_last_commit d45d6ac

git_last_commit_date 2026-08-02

Repository Bioconductor 3.24

Date/Publication 2026-08-03

Author Ossama Edbali [aut, cre] (ORCID:
<https://orcid.org/0000-0003-0132-8668>),
Ralf Johannes Maria Weber [aut] (ORCID:
<https://orcid.org/0000-0002-8796-4771>)

Maintainer Ossama Edbali <o.edbali@bham.ac.uk>

Contents

`lcmsPlot-package` *lcmsPlot: Comprehensive Liquid Chromatography-Mass Spectrometry (LC-MS) data visualisation package*

Description

lcmsPlot offers flexible and powerful visualisation of raw and processed LC-MS data. It supports chromatograms, mass spectra, and other plot types, combining high performance with broad customisation. Designed for large datasets, it facilitates assessment of signal quality, feature comparison across samples, and generation of publication-ready figures.

Details

Main features

- Unified, intuitive, and ggplot2-like interface.
- Plot from different types of sources, such as raw files (e.g. mzML), XCMS objects (e.g., `XCMSnExp`), or Compound Discoverer results.
- Plot chromatograms, mass traces, spectra, and more.
- Combine different types of LC-MS data plots (e.g. chromatograms with spectra).
- Arrange plots in different ways according to metadata factors.
- Large-scale plotting through iterative batching.

Author(s)

Maintainer: Ossama Edbali <o.edbali@bham.ac.uk> (ORCID)

Authors:

- Ossama Edbali <o.edbali@bham.ac.uk> (ORCID)
- Ralf Johannes Maria Weber <r.j.weber@bham.ac.uk> (ORCID)

See Also

Useful links:

- https://github.com/computational-metabolomics/lcmsPlot
- Report bugs at https://github.com/computational-metabolomics/lcmsPlot/issues

+,lcmsPlotClass,function-method

Apply a function to an `lcmsPlotClass` *object using the infix* `+` *operator*

Description

This provides a convenient infix style for applying transformations to `lcmsPlotClass` objects.

Usage

```
## S4 method for signature 'lcmsPlotClass,function'
e1 + e2
```

Arguments

`e1`	An instance of class `lcmsPlotClass`.
`e2`	A function that takes an `lcmsPlotClass` object and returns another.

Value

An instance of class `lcmsPlotClass`.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

p <- lcmsPlot(raw_files) +
  lp_chromatogram(aggregation_fun = "max") +
  lp_arrange(group_by = "sample_id") +
  lp_legend(position = "bottom") +
  lp_labels(legend = "Sample")
```

.apply_options

Apply Options-panel choices as `lp_*()` *layers to an lcmsPlotClass*

Description

Pure function (no Shiny). Given an `lcmsPlotClass` and a list as produced by `.options_server()`, appends the corresponding `lp_facets`, `lp_arrange`, `lp_legend`, and `lp_labels` layers when their inputs are non-`NULL`. Used by every plot module immediately before `lp_get_plot()`.

Usage

```
.apply_options(obj, opts)
```

Arguments

obj	An lcmsPlotClass.
opts	A named list of options (see .options_server()).

Value

The (possibly layered) lcmsPlotClass.

.app_content *Build the content area: hidden tabset + per-tab options panel*

Description

Each shiny::tabPanel() hosts a plot card on the left and the Options card on the right. Tab switching is driven from the left nav rail via shiny::updateTabsetPanel() — the tabsetPanel itself is type = "hidden" so no Bootstrap tab strip is shown.

Usage

```
.app_content()
```

Value

A shiny::tags$main.

.app_navrail *Build the left nav rail (file upload, samples, tab nav)*

Description

Build the left nav rail (file upload, samples, tab nav)

Usage

```
.app_navrail()
```

Value

A shiny::tags$aside.

.APP_TABS *Tabs displayed in the dashboard's left nav rail*

Description

Each entry is the value used by the hidden tabsetPanel; the label and optional FontAwesome icon (via shiny::icon()) are rendered into the nav buttons by .app_navrail().

Usage

```
.APP_TABS
```

.app_topbar *Build the sticky topbar of the dashboard*

Description

Build the sticky topbar of the dashboard

Usage

```
.app_topbar()
```

Value

A `shiny::tags$header`.

.build_server *Internal server builder for the Shiny app*

Description

Internal server builder for the Shiny app

Usage

```
.build_server()
```

Value

A function suitable for `shiny::shinyApp(server = ...)`.

.build_ui *Internal UI builder for the Shiny app*

Description

Lays out the dashboard: sticky topbar, left nav rail, main content with a hidden tabset. The Tailwind stylesheet is attached via `.lcmsPlot_dep()`.

Usage

```
.build_ui()
```

Value

A `shiny::tagList` wrapping the dashboard.

.cd_node_build_compounds

Build the consolidated per-compound, per-file compound table

Description

Associate each compound, in each study file, with a representative ion (preferring the molecular ion, otherwise the most abundant feature).

Usage

```
.cd_node_build_compounds(tabs)
```

Arguments

tabs	A list of classified tables from .cd_node_classify_tables().

Details

Dispatches on the export topology: when a per-file features table and its Compounds-per-File link are present the feature-level builder is used; otherwise the Compound-centric builder is used (m/z from the compound, per-file RT/area from the instance table).

Value

A tibble with one row per compound and study file, containing compound_id (the Compound Discoverer "Compounds ID" primary key), study_file_id, name, formula, adduct, mz, rt, rtmin, rtmax, into, maxo, and compound_rank (a dense rank of compounds by descending total area, for top-N selection).

.cd_node_build_compounds_compound

Compound-centric compound builder (CD "Unknown Compounds" topology)

Description

Builds the per-compound, per-file table when m/z lives on the consolidated compounds table and per-file RT/area live on the cpf (instance) table. The representative adduct label is taken from the molecular ion of the compound's features/ions via link_cmp_feat when available, otherwise NA.

Usage

```
.cd_node_build_compounds_compound(tabs)
```

.cd_node_build_compounds_feature

Feature-level compound builder (per-file feature topology)

Description

Feature-level compound builder (per-file feature topology)

Usage

```
.cd_node_build_compounds_feature(tabs)
```

.cd_node_build_metadata

Map study files to sample paths

Description

Builds a sample metadata table from the distinct study files referenced in the consolidated compound table and the user-supplied `sample_paths`. When a study-file table with file names is available, study files are matched to `sample_paths` by basename (ignoring extension); otherwise study files are matched positionally by sorted study-file ID.

Usage

```
.cd_node_build_metadata(compounds, study_files, sample_paths)
```

Arguments

`compounds`	A `tibble` from `.cd_node_build_compounds()`.
`study_files`	A `tibble` study-file table or `NULL`.
`sample_paths`	A `character` vector of raw sample (e.g. mzML) paths.

Value

A `tibble` with `study_file_id`, `sample_index`, `sample_id`, and `sample_path`.

.cd_node_classify_tables

Classify the Compound Discoverer scripting-node tables

Description

Identifies the Compounds, Compounds per File, Features, the two link tables, and (optionally) a study-file table from a list of exported tables, based on their columns rather than their position.

Usage

```
.cd_node_classify_tables(tables)
```

Arguments

tables	A list of tibbles as returned by .cd_node_read_tables().

Details

Two export topologies are supported:

- **Per-file feature** (targeted-style): each features row carries m/z, Ion and RT, joined to cpf via a Compounds-per-File ↔ Features link (link_cpf_feat).
- **Compound-centric** (CD "Unknown Compounds"): the consolidated compounds table carries m/z (compound_mz), features/ions link to the *compound* (link_cmp_feat), and per-file RT/area live on the cpf instance table. In this topology features/link_cpf_feat are optional.

Value

A named list with elements compounds, cpf, features, link_cmp_cpf, link_cpf_feat, link_cmp_feat, and study_files (any of which may be NULL).

.cd_node_compound_labels

Synthesize unique, non-empty compound labels

Description

Compound Discoverer "Unknown Compounds" have an empty Name; the plot keys compound identity on name, so a stable, unique, non-empty label is required. Uses Name when present, else Formula, else Compound <id>, and appends " [#<id>]" only where a label would otherwise collide across compound IDs.

Usage

```
.cd_node_compound_labels(ids, names, formulas)
```

.cd_node_read_tables *Read the Compound Discoverer scripting-node tables*

Description

Parses the node_args.json file produced by a Compound Discoverer Scripting Node (or accepts an already-parsed list) and reads each referenced tab-delimited table export into a tibble.

Usage

```
.cd_node_read_tables(node_args)
```

Arguments

node_args	A character path to a node_args.json file or an already-parsed list with the same structure.

Value

A list of tibbles, one per entry in Tables.

.cd_node_write_plot_column *Write a Compound Discoverer* node_response.json *adding a per-compound column*

Description

Writes a node_response.json that returns **only** the Compounds table (the one table modified) with one new column appended, matched on the Compounds table's "Compounds ID" primary key. Each row's value comes from id_to_value (rows without a mapping get an empty string). The column is displayed in Compound Discoverer via the supplied SpecialCellRenderer GUID - e.g. CD's filename renderer, which turns the cell into a clickable file link. Only the modified table is returned.

Usage

```
.cd_node_write_plot_column(
  node_args,
  id_to_value,
  column_name = "Plot",
  renderer = "EB29D794-4F2E-4785-8B80-A24D8C0FB3E4",
  position_after = "Name"
)
```

Arguments

node_args	A character path to the node_args.json file.
id_to_value	A data.frame with columns compound_id and value mapping each Compounds ID to its cell value (e.g. a plot file path).
column_name	The new column's header. Default "Plot".
renderer	The SpecialCellRenderer GUID string.
position_after	An existing Compounds column to place the new column after. Default "Name".

Value

The path to the written node_response.json (invisibly), or NULL if the node_args has no ExpectedResponsePath (e.g. a standalone run).

.chromatogram_server *Server for the BPC / TIC tab*

Description

Server for the BPC / TIC tab

Usage

```
.chromatogram_server(id, lcms_obj, selected_samples, metadata_cols)
```

Arguments

id	The module namespace id.
lcms_obj	A reactive returning an lcmsPlotClass object, or NULL if no data has been loaded yet.
selected_samples	A reactive returning a character vector of sample IDs to plot, or NULL for all samples.
metadata_cols	A reactive returning a character vector of metadata column names to feed the Options panel pickers.

Value

The module server's return value (currently NULL).

.chromatogram_ui *UI for the BPC / TIC tab*

Description

UI for the BPC / TIC tab

Usage

```
.chromatogram_ui(id)
```

Arguments

id	The module namespace id.

Value

A shiny::tagList.

.eic_features *Build a features matrix for an extracted ion chromatogram*

Description

Produces the single-row features matrix expected by lp_chromatogram() from a target mz, a ppm tolerance, and an optional retention-time window. When rt is NULL the RT columns are filled with NA so the full RT range is plotted.

Usage

```
.eic_features(mz, ppm, rt = NULL, rt_tol = NULL)
```

Arguments

mz	A numeric scalar — the target m/z.
ppm	A numeric scalar — the ppm tolerance.
rt	A numeric scalar or NULL — the target retention time (in the same unit as the data, typically seconds).
rt_tol	A numeric scalar — the symmetric RT tolerance. Ignored when rt is NULL.

Value

A 1x4 matrix with column names mzmin, mzmax, rtmin, rtmax.

.eic_server *Server for the EIC tab*

Description

Server for the EIC tab

Usage

```
.eic_server(id, lcms_obj, selected_samples, metadata_cols)
```

Arguments

id	The module namespace id.
lcms_obj	Reactive returning the base lcmsPlotClass.
selected_samples	Reactive returning the selected sample IDs.
metadata_cols	Reactive returning available metadata column names.

Value

NULL.

.eic_ui *UI for the EIC tab*

Description

UI for the EIC tab

Usage

```
.eic_ui(id)
```

Arguments

id	The module namespace id.

Value

A shiny::tagList.

.empty_to_null *Normalise an empty-string input to NULL*

Description

Normalise an empty-string input to NULL

Usage

```
.empty_to_null(x)
```

.file_ext *Lowercased file extension without the dot*

Description

Lowercased file extension without the dot

Usage

```
.file_ext(path)
```

Arguments

path	A character vector of file paths.

Value

A character vector of the same length, lowercased extensions.

.lcmsPlot_dep htmlDependency *for the bundled Tailwind stylesheet*

Description

Standard Shiny pattern for serving a CSS file from inst/www/. Uses htmltools::htmlDependency() so the stylesheet picks up a version stamp (bust the browser cache on package upgrades) and resolves the path via system.file() internally — works under devtools::load_all() too.

Usage

```
.lcmsPlot_dep()
```

.load_dataset *Translate a set of uploaded file paths to whatever* lcmsPlot() *expects*

Description

Inspects file extensions to decide how each upload should be handed off to lcmsPlot(). Pure function, no Shiny — easy to unit-test.

Usage

```
.load_dataset(paths)
```

Arguments

paths	A character vector of full file paths.

Details

Dispatch:

- mzML / CDF / raw paths → returned unchanged so lcmsPlot() can dispatch them through the existing character-vector path.
- A single .cdResult path → returned unchanged; lcmsPlot() opens the SQLite connection internally via is_cd_results_path().
- A single .rds → readRDS()'d and the resulting object returned.
- A single .RData / .Rdata / .rda → load()'d into a fresh environment; the first object inheriting from one of .SUPPORTED_DATA_CLASSES is returned. Errors with a clear message if no such object exists.

Empty input, unrecognised extensions, or mixed types raise a stop() so the caller can surface the message via .toast_error().

Value

Either a character vector of paths, a character scalar path, or an R object — whichever is appropriate for lcmsPlot().

.ls_adduct *Derive the adduct from a LipidSearch ion name*

Description

LipidSearch ion names carry the adduct as a suffix, e.g. AEA(18:2)+H or PC(16:0_18:1)+NH4.

Usage

```
.ls_adduct(lipid_ion)
```

Arguments

lipid_ion	A character vector of lipid ion names.

Value

A character vector of adducts, NA where none can be determined.

.ls_bracket_key *Extract the bracket key from a suffixed column name*

Description

"Area[s1-1]" -> "s1-1"; a name without a bracket suffix yields NA.

Usage

```
.ls_bracket_key(cols)
```

Arguments

cols	A character vector of column names.

Value

A character vector of bracket keys.

.ls_build_lipids *Reshape a LipidSearch table into one row per lipid and declared sample*

Description

Reshape a LipidSearch table into one row per lipid and declared sample

Usage

```
.ls_build_lipids(res, keep_rejected)
```

Arguments

res	The parsed result of .ls_read_results().
keep_rejected	A logical value indicating whether rows flagged as rejected should be retained.

Value

A tibble with one row per (lipid row, declared sample), carrying the lipid-level fields plus lipid_row (the source row index), sample_key and detected.

.ls_build_metadata *Build the sample metadata for a LipidSearch source*

Description

sample_paths is authoritative: every supplied path becomes a sample. Paths are matched to the result file's samples by .ls_match_samples() - by filename for 4.2, by sample key / position for 5.2. Paths that match nothing are kept as extra samples, plotted from each lipid's consensus m/z and RT window.

Usage

```
.ls_build_metadata(declared, sample_paths, metadata = NULL)
```

Arguments

declared	A tibble of sample declarations from .ls_read_results().
sample_paths	A character vector of raw sample file paths.
metadata	A data.frame (optional) of additional sample metadata with one row per sample_paths entry.

Value

A tibble with sample_index, sample_id, sample_path, sample_key, sample_group and in_results.

.ls_coerce *Coerce a LipidSearch column to the expected type*

Description

Cells for undetected lipids are left empty, which makes read.table() type otherwise-numeric columns as character.

Usage

```
.ls_coerce(values, numeric, n)
```

Arguments

values	The raw column values, or NULL when the column is absent.
numeric	A logical value indicating whether the field is numeric.
n	The number of rows to return when values is NULL.

Value

A numeric or character vector of length n.

.ls_consensus *Compute per-lipid consensus extraction values*

Description

These drive chromatogram extraction in samples that are absent from the result file, and act as a fallback for declared samples in which the lipid was not detected. Consensus values are taken over the samples where the lipid *was* detected, falling back to all declared samples, the reported `BaseRt` (5.2), and finally the theoretical m/z.

Usage

```
.ls_consensus(lipids)
```

Arguments

`lipids`	A `tibble` as returned by `.ls_build_lipids()`.

Value

A `tibble` with one row per `lipid_row` and columns `lipid_row`, `mz`, `rt`, `rtmin` and `rtmax` (retention times in minutes).

.ls_discover_samples *Discover the 5.2 sample declarations from the table columns*

Description

LipidSearch 5.2 files carry no `#[key]:file` declarations. The sample keys are taken from the group-level `OrgMeanArea[...]` columns (`s1`, `s2`, ...); each key's per-injection data lives in columns suffixed `<key>-<n>` (e.g. `Area[s1-1]`), so the first such injection key is resolved per sample.

Usage

```
.ls_discover_samples(cols)
```

Arguments

`cols`	A `character` vector of the table's column names.

Value

A `tibble` of `sample_key`, `sample_group`, `injection_key` and `file_name` (always `NA`), or `NULL` when no group columns are found.

.ls_escape *Escape a string for literal use inside a regular expression*

Description

Escape a string for literal use inside a regular expression

Usage

```
.ls_escape(x)
```

.ls_lipid_fields *Resolve the lipid-level columns of a LipidSearch table*

Description

Resolve the lipid-level columns of a LipidSearch table

Usage

```
.ls_lipid_fields(df)
```

Arguments

df	The lipid table from `.ls_read_results()`.

Value

A `tibble` with the standardised lipid-level columns. Fields absent from the export are filled with `NA`.

.ls_lipid_labels *Synthesize unique lipid labels*

Description

The plot keys compound identity on `name`, but `LipidIon` is not unique: the same ion is commonly reported at several retention times. Duplicated ions are disambiguated with their consensus RT, falling back to the source row index.

Usage

```
.ls_lipid_labels(lipid_ion, rt, lipid_row)
```

Arguments

lipid_ion	A `character` vector of lipid ion names.
rt	A `numeric` vector of consensus retention times, in minutes.
lipid_row	An `integer` vector of source row indices.

Value

A `character` vector of unique labels.

`.ls_match_samples` *Match sample paths to the file's declared samples*

Description

Returns, for each `sample_paths` entry, the index into `declared` it maps to (`NA` for an extra sample). LipidSearch 4.2 files name their raw files, so paths are matched by basename without extension. LipidSearch 5.2 files carry no filenames: a named `sample_paths` vector is matched by its names against the sample keys (`s1`, `s2`, ...), and an unnamed vector is matched positionally.

Usage

```
.ls_match_samples(declared, sample_paths)
```

Arguments

`declared`	A `tibble` of sample declarations from `.ls_read_results()`.
`sample_paths`	A `character` vector of raw sample file paths.

Value

An `integer` vector of `nrow == length(sample_paths)` of row indices into `declared`, or `NA` for extra samples.

`.ls_parse_rejected` *Parse the LipidSearch rejection flag as a logical*

Description

The flag is `0`/`1` in LipidSearch 4.2 and `false`/`true` in 5.2. Absent or unparseable values are treated as *not* rejected.

Usage

```
.ls_parse_rejected(values, n)
```

Arguments

`values`	The raw column values, or `NULL` when the column is absent.
`n`	The number of rows to return when `values` is `NULL`.

Value

A `logical` vector of length `n`, never `NA`.

.ls_read_results *Read a LipidSearch result file*

Description

Reads a LipidSearch result file and detects its version. LipidSearch 4.2 files begin with a block of `#[key]:file` sample declarations and `#key:value` settings, followed by the tab-delimited lipid table. LipidSearch 5.2 files carry no declarations; the table starts on the first line and the samples are discovered from the `OrgMeanArea[...]` columns.

Usage

```
.ls_read_results(path)
```

Arguments

`path`	A `character` path to a LipidSearch result file.

Value

A named `list` with elements `version` (`"4.2"` or `"5.2"`), `samples` (a `tibble` of `sample_key`, `sample_group`, `injection_key` and `file_name`), `settings` (a named `list`), and `data` (a `tibble` of the lipid table).

.ls_sample_fields *Extract one sample's columns from a LipidSearch table*

Description

Extract one sample's columns from a LipidSearch table

Usage

```
.ls_sample_fields(df, injection_key)
```

Arguments

`df`	The lipid table from `.ls_read_results()`.
`injection_key`	A `character` value giving the per-injection column key, e.g. `"c-1"` (4.2) or `"s1-1"` (5.2).

Value

A `tibble` with one row per lipid and the standardised per-sample columns. Fields absent from the export are filled with `NA`. In 5.2 the undetected marker `"NP"` becomes `NA` for numeric fields.

.mz_window *Compute an m/z window from a target m/z and a ppm tolerance*

Description

Internal helper used by the Shiny app to translate a user-supplied centre mass and ppm tolerance into the mzmin/mzmax pair expected by lp_chromatogram().

Usage

```
.mz_window(mz, ppm)
```

Arguments

mz	A numeric scalar — the target m/z.
ppm	A numeric scalar — the symmetric tolerance in parts-per-million.

Value

A length-2 numeric vector c(mzmin, mzmax).

.nav_active_js *Tiny JS handler that toggles the* lp-nav-active *class on a nav link*

Description

Injected once into the page head so the R .build_server() can switch the highlighted nav button via session$sendCustomMessage().

Usage

```
.nav_active_js
```

.na_to_null *Normalise an NA numeric input to NULL*

Description

Normalise an NA numeric input to NULL

Usage

```
.na_to_null(x)
```

.options_server *Server-side counterpart for the Options panel*

Description

Collects the input values into a single reactive list with one entry per option. Empty strings and NA are normalised to NULL so that .apply_options() can simply test for non-NULL values when deciding which lp_*() layer to append.

Usage

```
.options_server(id, metadata_cols = NULL)
```

Arguments

id	The module namespace id.
metadata_cols	Optional reactive returning a character vector of metadata column names to populate the facet/group dropdowns.

Details

If metadata_cols is supplied, the Facet by and Group by dropdowns are kept in sync with the loaded data's metadata columns.

Value

A reactive returning a named list with members facets, facet_ncol, free_x, free_y, arrange_by, legend_position, title, legend_title.

.options_ui *Build the per-tab Options panel UI*

Description

Renders a card with form controls for lp_facets, lp_arrange, lp_legend, and lp_labels. Inputs are placed inside the module namespace given by id.

Usage

```
.options_ui(id, metadata_cols = character(0))
```

Arguments

id	The module namespace id.
metadata_cols	A character vector of metadata column names to offer as choices for the facet and grouping pickers. Pass character(0) before any data is loaded — the dropdowns will simply be empty.

Value

A shiny::tagList containing the option controls wrapped in a Tailwind lp-card.

.peak_density_server *Server for the peak density tab*

Description

Server for the peak density tab

Usage

```
.peak_density_server(id, lcms_obj, metadata_cols)
```

Arguments

id	The module namespace id.
lcms_obj	Reactive returning the base lcmsPlotClass.
metadata_cols	Reactive returning available metadata column names.

Value

NULL.

.peak_density_ui *UI for the peak density tab*

Description

UI for the peak density tab

Usage

```
.peak_density_ui(id)
```

Arguments

id	The module namespace id.

Value

A shiny::tagList.

.spectra_server *Server for the spectra tab*

Description

Server for the spectra tab

Usage

```
.spectra_server(id, lcms_obj, metadata_cols)
```

Arguments

id	The module namespace id.
lcms_obj	Reactive returning the base lcmsPlotClass.
metadata_cols	Reactive returning available metadata column names.

Value

NULL.

.spectra_ui *UI for the spectra tab*

Description

UI for the spectra tab

Usage

```
.spectra_ui(id)
```

Arguments

id	The module namespace id.

Value

A shiny::tagList.

.SUPPORTED_DATA_CLASSES *Classes the Shiny app accepts when loading a* .rds *or* .RData *payload*

Description

These match the classes `lcmsPlot()` understands as `data_obj` (see `R/lcmsPlotDataContainer-class.R`). Kept as a module-level constant so it's easy to extend in one place.

Usage

```
.SUPPORTED_DATA_CLASSES
```

.tic_from_raw_files *Build the total ion current dataset from raw MS files*

Description

Reads the per-scan total ion current (TIC) from the headers of the raw files referenced in `metadata` and returns it in the layout expected by the `total_ion_current` slot of an `lcmsPlotDataContainer`.

Usage

```
.tic_from_raw_files(metadata, options)
```

Arguments

`metadata`	A `data.frame` of sample metadata with at least the columns `sample_id`, `sample_index`, and `sample_path`.
`options`	A `list` of plot options. `options$total_ion_current$sample_ids` selects which samples to read; `options$parallel_param`, when set, is used to read files in parallel.

Value

A `tibble` with columns `intensity`, `metadata_index`, and `feature_metadata_id` (one row per MS1 scan).

.tic_server *Server for the TIC summary tab*

Description

Server for the TIC summary tab

Usage

```
.tic_server(id, lcms_obj, selected_samples, metadata_cols)
```

Arguments

id	The module namespace id.
lcms_obj	Reactive returning the base lcmsPlotClass.
selected_samples	Reactive returning the selected sample IDs.
metadata_cols	Reactive returning available metadata column names.

Value

NULL.

.tic_ui *UI for the TIC summary tab*

Description

UI for the TIC summary tab

Usage

```
.tic_ui(id)
```

Arguments

id	The module namespace id.

Value

A shiny::tagList.

.toast_error *Show an error toast and return invisibly*

Description

Thin wrapper around `shinytoastr::toastr_error()` used by module servers to surface `lp_*()` errors without crashing the session.

Usage

```
.toast_error(msg, title = "Error")
```

Arguments

`msg`	A `character` message to display.
`title`	A `character` title for the toast.

Value

`NULL` invisibly.

.uploaded_files_to_paths *Copy uploaded Shiny files to a session temp dir under their original names*

Description

`shiny::fileInput()` writes uploaded files to randomly-named tempfiles (`/tmp/.../0.mzML`) and reports the original names in a separate `name` column. The `lcmsPlot()` constructor derives sample IDs from file basenames, so we copy each upload to a session-scoped directory under its original name to keep sample IDs meaningful.

Usage

```
.uploaded_files_to_paths(upload, dir)
```

Arguments

`upload`	A `data.frame` as produced by `shiny::fileInput()` with at least the columns `datapath` and `name`. May be `NULL`.
`dir`	A `character` directory path to copy files into. The directory is created if it does not exist.

Value

A `character` vector of full paths to the renamed files, in the same order as `upload`. Returns `character(0)` if `upload` is `NULL` or empty.

add_compound_rank *Add a dense compound rank (by descending total area) for top-N selection*

Description

Shared by the compound-centric data sources (Compound Discoverer scripting node, LipidSearch). Compounds are ranked by the total area summed across samples, so `rank == 1` is the most abundant compound.

Usage

```
add_compound_rank(rows, rank_column = "compound_rank")
```

Arguments

`rows`	A `data.frame` with `name` and `into` columns.
`rank_column`	A `character` value giving the name of the rank column to add.

Value

`rows` with the rank column appended.

bin_coordinate *Snap a coordinate onto a grid of the given bin width*

Description

Widths that are a negative power of ten are handled with `round(x, digits)`, which is more accurate than scaling by hand and is what the previously hard-coded 0.1 binning used, so default output is unchanged.

Usage

```
bin_coordinate(x, width)
```

Arguments

`x`	A `numeric` vector of coordinates.
`width`	A `numeric` value giving the bin width.

Value

A `numeric` vector of bin centres.

`chrom_peak_rects_layer` *Draw chromatographic peak boundaries on an rt / m/z panel*

Description

Draws one rectangle per detected chromatographic peak, spanning `rtmin`-`rtmax` by `mzmin`-`mzmax`. This is the primitive behind `xcms::plotChromPeaks()` and `xcms::plot(type = "XIC")`: the former draws the rectangles on an empty frame, the latter over the ion map.

Usage

```
chrom_peak_rects_layer(
  p,
  detected_peaks,
  options,
  x_dim = "rt",
  rt_range = NULL,
  mz_range = NULL
)
```

Arguments

`p`	A `ggplot` object whose axes are retention time and m/z.
`detected_peaks`	A `data.frame` of detected peaks, requiring columns `rtmin`, `rtmax`, `mzmin`, `mzmax`, and `sample_id`.
`options`	A `list` of plot options (reads `options$chrom_peak_rects`).
`x_dim`	A `character` naming the dimension on the x axis, `"rt"` (default) or `"mz"`. The other axis carries the remaining dimension.
`rt_range`	An optional length-2 `numeric` clamping the rectangles to the host panel's retention-time window.
`mz_range`	An optional length-2 `numeric` clamping the rectangles to the host panel's m/z window.

Details

The caller passes its own axis assignment, so the rectangles follow when the host map is flipped to m/z on x. Used both by `plot_chrom_peak_rects()` for the standalone panel and by the mass trace and intensity map renderers when the layer decorates them.

Value

The modified `ggplot` object.

`CompoundDiscovererNodeSource` *Create a Compound Discoverer data source from a Scripting Node export*

Description

Reads the `node_args.json` file produced by a Compound Discoverer Scripting Node together with the tab-delimited table exports it references, and builds a data source that can be plotted with `lcmsPlot()`. Chromatograms are later extracted from the raw (mzML) files supplied via `sample_paths`.

Usage

```
CompoundDiscovererNodeSource(node_args, sample_paths, metadata = NULL)
```

Arguments

`node_args`	A `character` path to the `node_args.json` file passed by Compound Discoverer or an already-parsed `list`.
`sample_paths`	A `character` vector of paths to the raw sample files (e.g. mzML) used to extract chromatograms. Study files are matched to these paths by file basename when a study-file table is available, otherwise positionally by study-file ID.
`metadata`	A `data.frame` (optional) of additional sample metadata. It must contain one row per sample, aligned with `sample_paths`. When supplied, its columns are column-bound onto the generated metadata.

Details

The Scripting Node must be configured to request the Compounds, Compounds per File, and Features tables (with their link tables). For each compound and study file a representative ion is selected, preferring a molecular ion and otherwise the most abundant feature.

Value

An object of class `CompoundDiscovererNodeSource`.

See Also

`lp_compound_discoverer()`

Examples

```
## Inside a real Compound Discoverer Scripting Node the JSON path arrives as
## the 6th command-line argument, i.e. node_args <- commandArgs()[6]. The
## export below stands in for one so the example can run standalone.
node_dir <- tempfile("cd_node")
dir.create(node_dir)

write_tab <- function(df, name) {
    path <- file.path(node_dir, name)
    utils::write.table(
```

```
        df, path, sep = "\t", row.names = FALSE, quote = FALSE)
    path
}

## Two compounds, one detected in each study file.
compounds <- data.frame(
    "Compounds ID" = c(1, 2),
    "Name" = c("L-Proline", "L-Kynurenine"),
    "Formula" = c("C5 H9 N O2", "C10 H12 N2 O3"),
    check.names = FALSE)

compounds_per_file <- data.frame(
    "Compounds per File ID" = c(1, 2),
    "StudyFileID" = c(1, 2),
    "Area" = c(24050, 18700),
    "Intensity" = c(24050, 18700),
    check.names = FALSE)

## Compound Discoverer reports retention times in minutes.
features <- data.frame(
    "Features ID" = c(1, 2),
    "mz" = c(116.0704, 209.0918),
    "Ion" = c("[M+H]+1", "[M+H]+1"),
    "Area" = c(24050, 18700),
    "RT [min]" = c(7.10, 6.10),
    "Left RT [min]" = c(6.95, 5.95),
    "Right RT [min]" = c(7.25, 6.25),
    check.names = FALSE)

## Study files are matched to `sample_paths` by basename.
study_files <- data.frame(
    "StudyFileID" = c(1, 2),
    "File Name" = c("l-proline-MS1.mzML", "l-kynurenine-MS1.mzML"),
    check.names = FALSE)

node_args <- list(Tables = list(
    list(DataFile = write_tab(compounds, "compounds.txt")),
    list(DataFile = write_tab(compounds_per_file, "cpf.txt")),
    list(DataFile = write_tab(features, "features.txt")),
    list(DataFile = write_tab(
        data.frame(
            "Compounds ID" = c(1, 2),
            "Compounds per File ID" = c(1, 2),
            check.names = FALSE),
        "link_cmp_cpf.txt")),
    list(DataFile = write_tab(
        data.frame(
            "Compounds per File ID" = c(1, 2),
            "Features ID" = c(1, 2),
            check.names = FALSE),
        "link_cpf_feat.txt")),
    list(DataFile = write_tab(study_files, "study_files.txt"))))

json_path <- file.path(node_dir, "node_args.json")
writeLines(jsonlite::toJSON(node_args, auto_unbox = TRUE), json_path)

## Chromatograms are extracted from the raw files, not from the export.
```

```
mzml_dir <- file.path(tempdir(), "cd-node-example")
utils::unzip(
    system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
    exdir = mzml_dir)

sample_paths <- file.path(
    mzml_dir, "mzml",
    c("l-proline-MS1.mzML", "l-kynurenine-MS1.mzML"))

ds <- CompoundDiscovererNodeSource(
    node_args = json_path,
    sample_paths = sample_paths)

ds

p <- lcmsPlot(ds) +
    lp_compound_discoverer(rt_extend = 15) +
    lp_chromatogram(highlight_peaks = TRUE) +
    lp_grid(rows = "sample_id", cols = "name", free_x = TRUE)
p
```

CompoundDiscovererNodeSource-class *Compound Discoverer scripting-node data source*

Description

An S4 class wrapping the compound and peak data exported by a Compound Discoverer Scripting Node.

Slots

`name` A `character` value identifying the data source.

`metadata` A `data.frame` of sample metadata (one row per study file), including a `sample_path` column with the raw (e.g. mzML) file paths.

`peaks` A `data.frame` of detected peaks (one row per compound and study file) with the standard `mz`, `rt`, `rtmin`, `rtmax`, `into`, `maxo`, and `sample_index` columns, plus `name`, `formula`, and `adduct` annotations.

`compounds` A `data.frame` of the consolidated per-compound, per-file table used to drive chromatogram extraction.

---

convert_rt_to_seconds *Convert retention times to seconds*

Description

Convert retention time columns from minutes to seconds.

Usage

```
convert_rt_to_seconds(peaks)
```

Arguments

peaks	A tibble containing rt, rtmin, and rtmax columns in minutes.

Value

peaks with retention time columns converted to seconds.

---

create_bpc_tic *Create a base peak or total ion current chromatogram*

Description

Create a base peak or total ion current chromatogram

Usage

```
create_bpc_tic(raw_data, aggregation_fun, rt_adjusted = NULL)
```

Arguments

raw_data	An instance of class MsRawReader.
aggregation_fun	A character value indicating the aggregation method. One of "max" (base peak chromatogram), "sum" (total ion current), or "mean" (averaged ion chromatogram).
rt_adjusted	A numeric vector representing the adjusted RT values. If NULL it will use the raw RT values.

Value

A list with one tibble containing the chromatograms with columns rt and intensity.

---

create_chromatogram *Create an extracted ion chromatogram*

Description

Create an extracted ion chromatogram

Usage

```
create_chromatogram(
  raw_data,
  mz_range,
  rt_range,
  ms_level = 1,
  fill_gaps = FALSE,
  adjusted_rt = NULL
)
```

Arguments

raw_data	An instance of class MsRawReader.
mz_range	A numeric vector indicating the m/z range.
rt_range	A numeric vector indicating the RT range.
ms_level	A numeric value indicating the MS level of the scans to consider.
fill_gaps	A logical indicating whether to fill gaps between scans with zeros.
adjusted_rt	A tibble containing the raw and adjusted RTs.

Value

A list with two data frames (chromatograms and mass_traces) containing the chromatograms with columns rt and intensity and mass traces with columns rt and mz.

---

create_chromatograms *Create chromatogram data from a data object*

Description

Dispatches to the appropriate implementation based on the type of data_obj and features. Returns a named list with slots to be assigned onto lcmsPlotDataContainer.

Usage

```
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'DBIConnection,data.frame,list,NULL'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'CompoundDiscovererNodeSource,data.frame,list,NULL'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'LipidSearchSource,data.frame,list,NULL'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'XcmsRawList,data.frame,list,NULL'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'XcmsRawList,data.frame,list,ANY'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'MChromatograms,data.frame,list,NULL'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'XChromatogram,data.frame,list,NULL'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'XChromatograms,data.frame,list,NULL'
create_chromatograms(data_obj, metadata, options, features)
```

```
## S4 method for signature 'ANY,data.frame,list,NULL'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'ANY,data.frame,list,character'
create_chromatograms(data_obj, metadata, options, features)

## S4 method for signature 'ANY,data.frame,list,ANY'
create_chromatograms(data_obj, metadata, options, features)
```

Arguments

data_obj	The data object (e.g. XCMSnExp, DBIConnection, character).
metadata	A tibble of sample metadata.
options	A list of plot options.
features	NULL, a character vector of feature IDs, or a matrix/tibble of feature ranges.

Value

A named list with elements chromatograms, mass_traces, feature_metadata, and detected_peaks.

---

create_compound_chromatograms

Extract one chromatogram per compound and sample from the raw files

---

Description

Shared by the compound-centric data sources (Compound Discoverer scripting node, LipidSearch). Each row of all_compounds describes where to extract a compound in one sample: an mz, an rtmin/rtmax window, and the sample_index it belongs to.

Usage

```
create_compound_chromatograms(all_compounds, metadata, options, rt_extend)
```

Arguments

all_compounds	A tibble with at least name, formula, adduct, mz, rt, rtmin, rtmax, into, maxo, and sample_index columns. Optional class and detected columns are used when present.
metadata	A tibble of sample metadata.
options	A list of plot options.
rt_extend	A numeric value indicating how much (in seconds) to extend the RT window on each side of the compound peak.

Details

Compounds with no mz or no RT window are skipped, as are compounds absent from a given sample. When all_compounds carries a detected column, a detected_peaks row - the one the highlight_peaks layer draws - is emitted only where it is TRUE, so compounds that were looked for but not found are still plotted without a degenerate highlight.

Value

A named `list` with elements `chromatograms`, `mass_traces`, `feature_metadata`, and `detected_peaks`.

`create_data_container_from_obj` *Create an instance of class* `lcmsPlotDataContainer` *from a data object*

Description

The `create_data_container_from_obj` function creates an instance of class `lcmsPlotDataContainer` given a data object. See `lcmsPlotDataContainer` for more information about the supported data objects.

Usage

```
create_data_container_from_obj(data_obj, sample_id_column, metadata)
```

Arguments

`data_obj`	The data object (see `lcmsPlotDataContainer`).
`sample_id_column`	A `character` value indicating the sample ID column.
`metadata`	A `data.frame` containing the samples metadata in case it is not provided in the dataset object.

Value

An instance of class `lcmsPlotDataContainer`. The object contains the input data and the standardised metadata.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

data_container <- create_data_container_from_obj(
  data_obj = raw_files,
  sample_id_column = NULL,
  metadata = NULL
)
```

create_intensity_map *Create an instance of class* lcmsPlotDataContainer *from an intensity map*

Description

This function creates an lcmsPlotDataContainer object from an intensity map which is a point-cloud of m/z and RT points.

Usage

```
create_intensity_map(obj, options)

## S4 method for signature 'lcmsPlotDataContainer,list'
create_intensity_map(obj, options)
```

Arguments

obj	An instance of class lcmsPlotDataContainer.
options	A list representing the plot object's options.

Value

An instance of class lcmsPlotDataContainer with the created intensity map, a tibble with columns mz and rt.

create_isolation_window *Populate the* spectra *slot with an MS1 isolation window spectrum*

Description

Extracts the MS1 survey scan nearest to a fragmentation event from purityA@puritydf, reads the spectrum from the raw file, and stores it in the spectra slot. Isolation-window boundaries and the inPurity score are stored in feature_metadata for downstream annotation by plot_isolation_window().

Usage

```
create_isolation_window(obj, options, pid, sample_id)
```

Arguments

obj	An instance of class lcmsPlotDataContainer whose data_obj is a purityA object.
options	A list of plot options (reads options$isolation_window).
pid	An integer pid from puritydf selecting which fragmentation event(s) to visualise. NULL selects all events.
sample_id	A character sample ID restricting which file to read. Ignored when pid is provided. NULL selects all samples.

Value

A modified `lcmsPlotDataContainer` with `spectra` and `feature_metadata` slots populated.

`create_peak_count_image` *Create an instance of class* `lcmsPlotDataContainer` *from peak counts per retention-time bin*

Description

This function creates an `lcmsPlotDataContainer` object holding the number of chromatographic peaks each sample yielded in each retention-time bin, mirroring `xcms::plotChromPeakImage()`.

Usage

```
create_peak_count_image(obj, options)

## S4 method for signature 'lcmsPlotDataContainer,list'
create_peak_count_image(obj, options)
```

Arguments

`obj`	An instance of class `lcmsPlotDataContainer`.
`options`	A `list` representing the plot object's options.

Details

Bins with no peaks are kept with a count of zero rather than dropped, so a sample that stopped producing peaks part-way through a run shows up as an empty stretch instead of quietly disappearing from the plot.

Value

An instance of class `lcmsPlotDataContainer` with the created peak count image. The `peak_count_image` slot is a `tibble` with columns `rt` (the bin centre), `n_peaks`, `metadata_index`, and `feature_metadata_id`.

`create_peak_density` *Create an instance of class* `lcmsPlotDataContainer` *from a peak density dataset.*

Description

This function creates an `lcmsPlotDataContainer` object from a dataset that contains peak density data for each feature (i.e., the kernel density of peak apex RTs across samples for each m/z bin), mirroring the algorithm used by `xcms::plotChromPeakDensity()`.

Usage

```
create_peak_density(obj, options)

## S4 method for signature 'lcmsPlotDataContainer,list'
create_peak_density(obj, options)
```

Arguments

obj	An instance of class lcmsPlotDataContainer.
options	A list representing the plot object's options.

Value

An instance of class lcmsPlotDataContainer with the created peak density dataset. The peak_density slot is a tibble with columns rt, density, rtmin, rtmax, data_type ("density" or "rect"), mzmin, mzmax, metadata_index, and feature_metadata_id.

---

create_purity_scores *Populate the* purity_scores *slot from a* purityA *object*

Description

Extracts per-scan precursor ion purity scores from purityA@puritydf and stores them in the purity_scores slot of the data container.

Usage

```
create_purity_scores(obj, options)
```

Arguments

obj	An instance of class lcmsPlotDataContainer whose data_obj slot is a purityA object.
options	A list of plot options.

Value

A modified lcmsPlotDataContainer with the purity_scores slot populated.

create_rt_diff *Create an instance of class* lcmsPlotDataContainer *from an RT adjustment dataset*

Description

This function creates an lcmsPlotDataContainer object from a dataset that contains the data to generate RT raw vs adjusted plots.

Usage

```
create_rt_diff(obj, options)

## S4 method for signature 'lcmsPlotDataContainer,list'
create_rt_diff(obj, options)
```

Arguments

obj	An instance of class lcmsPlotDataContainer.
options	A list representing the plot object's options.

Value

An instance of class lcmsPlotDataContainer with the created RT adjustment dataset, a tibble with columns rt_raw, rt_adj, and diff.

create_spectra *Create an instance of class* lcmsPlotDataContainer *from spectra*

Description

Create an instance of class lcmsPlotDataContainer from spectra

Usage

```
create_spectra(obj, options)

## S4 method for signature 'lcmsPlotDataContainer,list'
create_spectra(obj, options)
```

Arguments

obj	An instance of class lcmsPlotDataContainer.
options	A list representing the plot object's options.

Value

An lcmsPlotDataContainer object with the created spectra.

create_spectra_for_sample

Create spectra for a single sample

Description

Create spectra for a single sample

Usage

```
create_spectra_for_sample(
  raw_obj,
  detected_peaks,
  sample_metadata,
  options,
  rt_range = NULL
)
```

Arguments

raw_obj	An instance of class mzR.
detected_peaks	A tibble containing the detected peaks to consider. Only applicable for modes "closest_apex" and "across_peak".
sample_metadata	A tibble containing the sample's metadata.
options	A list representing the plot object's options.
rt_range	A numeric value indicating the RT range to apply to the detected peaks.

Value

A tibble representing spectra with columns mz, intensity, and rt.

create_spectrum_from_closest_scan_to_rt

Create a spectrum of the closest scan to the specified RT

Description

Create a spectrum of the closest scan to the specified RT

Usage

```
create_spectrum_from_closest_scan_to_rt(raw_data, rt, ms_level)
```

Arguments

raw_data	An instance of class MsRawReader.
rt	A numeric value indicating the RT to consider.
ms_level	A numeric value indicating the MS level of the scans.

Value

A `tibble` representing a spectrum with columns `mz`, `intensity`, and `rt`.

`create_spectrum_from_scan_index` *Create a spectrum of the specified scan*

Description

Create a spectrum of the specified scan

Usage

```
create_spectrum_from_scan_index(raw_data, sample_metadata, scan_index)
```

Arguments

`raw_data`	An instance of class `MsRawReader`.
`sample_metadata`	A `tibble` indicating the sample metadata.
`scan_index`	A `numeric` value indicating the scan index.

Value

A `tibble` representing a spectrum with columns `mz`, `intensity`, and `rt`.

`create_total_ion_current` *Create an instance of class* `lcmsPlotDataContainer` *from total ion current (TIC) dataset*

Description

This function creates an `lcmsPlotDataContainer` object from a dataset that contains TIC values for each sample.

Usage

```
create_total_ion_current(obj, options)

## S4 method for signature 'lcmsPlotDataContainer,list'
create_total_ion_current(obj, options)
```

Arguments

`obj`	An instance of class `lcmsPlotDataContainer`.
`options`	A `list` representing the plot object's options.

Value

An instance of class `lcmsPlotDataContainer` with the created RT adjustment dataset, a `tibble` with columns `sample_id` and `intensity`.

`create_xcms_raw_list` *Create an* `XcmsRawList` *from raw LC-MS files*

Description

Reads one or more raw MS files using `xcms::xcmsRaw()` and wraps the results in an `XcmsRawList`. Optionally runs the reads in parallel via `BiocParallel`.

Usage

```
create_xcms_raw_list(
  paths,
  profstep = 1,
  mslevel = NULL,
  scanrange = NULL,
  BPPARAM = NULL
)
```

Arguments

`paths`	A `character` vector of file paths to raw MS files (e.g. `.mzML`, `.mzXML`, `.CDF`).
`profstep`	A `numeric` value passed to `xcms::xcmsRaw()` controlling the mass bin size used to build the profile matrix. Defaults to `0`, which skips profile matrix generation. Only set this to a positive value if you need the profile matrix for peak detection.
`mslevel`	A `numeric` value passed to `xcms::xcmsRaw()` indicating which MS level to load. `NULL` loads all levels. Defaults to `NULL`.
`scanrange`	A length-2 `integer` vector passed to `xcms::xcmsRaw()` restricting the range of scans to read. `NULL` reads all scans. Defaults to `NULL`.
`BPPARAM`	A `BiocParallelParam` object controlling parallel execution. When `NULL` (default) files are read sequentially.

Value

An `XcmsRawList` object with one `xcmsRaw` element per file.

Examples

```
paths <- dir(
    system.file("cdf", package = "faahKO"),
    full.names = TRUE,
    recursive = TRUE
)[1:3]
xl <- create_xcms_raw_list(paths)
```

default_options *Get the default options*

Description

`default_options()` returns a list of default settings used throughout the package. These options control aspects such as sample metadata handling, plotting, chromatogram and spectra display, and general visualization parameters.

Usage

```
default_options()
```

Value

A named `list` containing all default options.

detect_separator *Detect field separator from file extension*

Description

Determines the column separator to use when reading a delimited text file based on its file extension. Files ending in `.tsv` (case-insensitive) are assumed to be tab-delimited; all others default to comma-delimited.

Usage

```
detect_separator(path)
```

Arguments

`path`	A `character` value indicating the path to the file whose separator should be detected.

Value

A `character` value indicating the field separator: `"\t"` for TSV files, otherwise `","`.

`ExternalDataSource-class` *External data source wrapper*

Description

An S4 class representing an external data source such as MZmine. This class acts as an adapter for external data sources by converting data to a common format (XCMS).

Slots

`name` A `character` value indicating the name of the data source.

`metadata` A `data.frame` representing the sample metadata.

`peaks` A `data.frame` representing the exported peaks.

`field` *Construct a field specification for data frame validation*

Description

Construct a field specification for data frame validation

Usage

```
field(name, type = NULL, required = TRUE)
```

Arguments

`name`	A `character` value giving the column name.
`type`	A function used to check the column's type (e.g. `is.numeric`). If `NULL`, no type check is performed.
`required`	A `logical` value indicating whether the column must be present. Defaults to `TRUE`.

Value

A named `list` with elements `name`, `type`, and `required`.

first_matching_column *Resolve the first matching column name in a data frame*

Description

Vendor exports name the same logical field differently between versions, so each field is resolved against a list of candidate column names.

Usage

```
first_matching_column(df, candidates)
```

Arguments

df	A data.frame.
candidates	A character vector of candidate column names.

Value

The first candidate present in df, or NA_character_ if none.

get_adjusted_rts *Retrieve raw and adjusted retention times from an xcms object*

Description

Extracts raw and adjusted retention times from an xcms-processed object when retention time correction has been performed.

Usage

```
get_adjusted_rts(obj)
```

Arguments

obj	An object potentially containing xcms-processed LC-MS data.

Details

If the object is not an xcms processed data object, or if retention time adjustment has not been applied, the function returns NULL.

Value

A tibble with one row per scan, containing:

file_index Index of the originating raw data file.

raw_rt Original (unadjusted) retention time.

adj_rt Adjusted retention time after RT correction.

If no adjusted retention times are available, NULL is returned.

`get_detected_peaks` *Get the detected peaks from the data object (e.g. XCMSnExp)*

Description

`get_detected_peaks()` is an internal helper that standardises extraction of detected chromatographic peaks across different object types commonly used in LC-MS workflows.

Usage

```
get_detected_peaks(obj)

## S3 method for class 'character'
get_detected_peaks(obj)

## S3 method for class 'XCMSnExp'
get_detected_peaks(obj)

## S3 method for class 'MsExperiment'
get_detected_peaks(obj)

## S3 method for class 'MChromatograms'
get_detected_peaks(obj)

## S3 method for class 'XChromatograms'
get_detected_peaks(obj)

## S3 method for class 'XChromatogram'
get_detected_peaks(obj)

## S3 method for class 'XcmsRawList'
get_detected_peaks(obj)

## S3 method for class 'ExternalDataSource'
get_detected_peaks(obj)

## S3 method for class 'CompoundDiscovererNodeSource'
get_detected_peaks(obj)

## S3 method for class 'LipidSearchSource'
get_detected_peaks(obj)

## S3 method for class 'purityA'
get_detected_peaks(obj)
```

Arguments

`obj`	A data object containing or representing samples.

Details

Supported inputs behave as follows:

- `character` – Assumed to represent sample paths; no peak detection information is available. Always returns `NULL`.
- `XCMSnExp` and `MsExperiment` – If the object is processed and contains chromatographic peaks, extracts `xcms::chromPeaks(obj)` and returns it as a data frame. The column `sample` is renamed to `sample_index`.

When peaks are not found or the object is not processed, `NULL` is returned.

Value

A `tibble` of detected peaks (one row per peak), or `NULL` if no peaks are available.

`get_features` *Get the feature data (m/z and RT ranges) for a set of features*

Description

Get the feature data (m/z and RT ranges) for a set of features

Usage

```
get_features(
  options,
  sample_metadata,
  grouped_peaks = NULL,
  full_rt_range = NULL
)
```

Arguments

`options`	The plot object's options. This contains the input features.
`sample_metadata`	The sample's metadata.
`grouped_peaks`	The grouped peaks to use as input features.
`full_rt_range`	The full RT range if an RT range is not given.

Value

A list of feature data (see get_feature_data).

get_feature_data *Get a feature's m/z and RT ranges given different feature specifications*

Description

Get a feature's m/z and RT ranges given different feature specifications

Usage

```
get_feature_data(feature, options, full_rt_range)
```

Arguments

feature	The input feature which can be a vector or a data frame row. The accepted columns combinations are: • mz, rt (optional) • mzmin, mzmax, rtmin (optional), rtmax (optional)
options	The plot object's options.
full_rt_range	The full RT range if an RT range is not given.

Value

A named list defining a feature with names: feature_id, mzr, rtr.

get_feature_definitions *Get the stored feature definitions from the data object*

Description

`get_feature_definitions()` retrieves the correspondence results already stored on an object, as opposed to `get_grouped_peaks()` which reshapes them for feature labelling. Used by `lp_peak_density()` when `simulate = FALSE` to draw the feature groups the object actually holds rather than re-deriving candidates from the density curve.

Usage

```
get_feature_definitions(obj)

## Default S3 method:
get_feature_definitions(obj)

## S3 method for class 'XCMSnExp'
get_feature_definitions(obj)

## S3 method for class 'XcmsExperiment'
get_feature_definitions(obj)
```

```
## S3 method for class 'MsExperiment'
get_feature_definitions(obj)

## S3 method for class 'XChromatograms'
get_feature_definitions(obj)
```

Arguments

obj	A data object containing or representing samples.

Value

A tibble of feature definitions with at least rtmin and rtmax, plus mzmin/mzmax and row where the object provides them; or NULL when no correspondence results are stored.

---

get_feature_windows *Get the m/z windows a data object already defines*

Description

get_feature_windows() returns the m/z (and, where known, retention time) windows that the object itself defines, so a layer does not have to ask the caller for features that the data already carries. Chromatogram objects define one window per extracted ion chromatogram.

Usage

```
get_feature_windows(obj)

## Default S3 method:
get_feature_windows(obj)

## S3 method for class 'XChromatograms'
get_feature_windows(obj)

## S3 method for class 'XChromatogram'
get_feature_windows(obj)
```

Arguments

obj	A data object containing or representing samples.

Value

A tibble with columns row, mzmin, mzmax, rtmin and rtmax, row identifying the originating EIC; or NULL when the object defines no windows.

get_grouped_peaks *Get the grouped peaks across samples (features) from the data object*

Description

get_grouped_peaks() is an internal helper that retrieves feature-level grouped peaks, i.e., chromatographic peaks aligned across samples.

Usage

```
get_grouped_peaks(obj)

## Default S3 method:
get_grouped_peaks(obj)

## S3 method for class 'XCMSnExp'
get_grouped_peaks(obj)

## S3 method for class 'XcmsExperiment'
get_grouped_peaks(obj)

## S3 method for class 'MsExperiment'
get_grouped_peaks(obj)

## S3 method for class 'purityA'
get_grouped_peaks(obj)
```

Arguments

obj	A data object containing or representing samples.

Value

A tibble of grouped (feature-level) peaks, or NULL if not available.

get_grouping_variables *Get the grouping variables for a plot*

Description

get_grouping_variables() extracts the variables used to group data in a plot based on the provided plot options. It first checks for facet specifications, and if absent, falls back to grid row/column settings.

Usage

```
get_grouping_variables(opts)
```

Arguments

opts	A list of plot options.

Value

A character vector containing the names of the grouping variables used for faceting or grid layout. May contain NULL values if no grouping is specified.

---

get_metadata *Get the metadata associated with the input object*

Description

get_metadata() is a generic helper used internally to standardise metadata extraction across different classes of input objects.

Usage

```
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'character'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'XCMSnExp'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'MsExperiment'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'MChromatograms'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'XChromatograms'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'XChromatogram'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'XcmsRawList'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'ExternalDataSource'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'CompoundDiscovererNodeSource'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'LipidSearchSource'
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'DBIConnection'
```

```
get_metadata(obj, sample_id_column, metadata)

## S3 method for class 'purityA'
get_metadata(obj, sample_id_column, metadata)
```

Arguments

obj	A data object containing or representing samples.
sample_id_column	A character value indicating the column that should be used as the sample ID.
metadata	Optional metadata tibble used to replace or augment sample metadata when not already embedded in the object.

Details

Depending on the class of obj, metadata may be:

- **constructed** (e.g., from character vectors of file paths), or
- **extracted and optionally replaced** (e.g., from XCMSnExp or MsExperiment objects).

Across all methods, the returned metadata is enriched with:

- sample_index - a sequential index of samples
- sample_id - an identifier column selected via sample_id_column
- sample_path - a file path associated with each sample (if applicable)

Value

A tibble containing standardised metadata with at least sample_index, sample_id, and sample_path.

Character vector input (character)

A character vector is treated as a list of sample file paths (e.g., .mzML, .mzXML, .cdf).

If metadata **is NULL:** Metadata is *constructed automatically*:

- sample_path: full paths given in obj
- sample_index: row number
- sample_id: basename of each file without extension

If metadata **is provided:** The supplied metadata is used and the following columns are added:

- sample_index: row number
- sample_id: extracted from the sample_id_column
- sample_path: the input paths from obj

XCMSnExp input

Metadata is taken from xcms::phenoData(obj).

If metadata **is provided:** It replaces the existing phenoData.

The returned metadata always includes:

- sample_index: row number
- sample_id: extracted using sample_id_column
- sample_path: values from xcms::fileNames(obj)

`MsExperiment` **input**

Metadata is taken from `MsExperiment::sampleData(obj)`.

If `metadata` **is provided:** It replaces existing `sampleData`.

The returned metadata includes:

- `sample_index`: row number
- `sample_id`: extracted using `sample_id_column`
- `sample_path`: values from `xcms::fileNames(obj)`

`MChromatograms` **and** `XChromatograms` **input**

Metadata is taken from `MSnbase::phenoData(obj)`.

`sample_id` is resolved in the following order:

1. `sample_id_column` (if provided and present in `phenoData`)
2. Basename without extension of the `spectraOrigin` column
3. `"sample1"`, `"sample2"`, ... (fallback)

`sample_path` is set to `spectraOrigin` if present, otherwise `NA`.

If `metadata` **is provided:** It is column-bound onto the extracted `phenoData`.

The returned metadata always includes:

- `sample_index`: row number
- `sample_id`: resolved as above
- `sample_path`: from `spectraOrigin` or `NA`

`XChromatogram` **input**

Represents a single chromatogram (one sample). `sample_path` is always `NA`.

If `metadata` **is NULL:** Returns a single-row tibble with `sample_index = 1`, `sample_id = "sample1"`, and `sample_path = NA`.

If `metadata` **is provided:** Must have exactly one row. The supplied metadata is used with:

- `sample_index`: `1`
- `sample_id`: from `sample_id_column` if present, otherwise `"sample1"`
- `sample_path`: `NA`

`XcmsRawList` **input**

Each element of the list is an `xcmsRaw` object. `sample_path` is extracted from the `@filepath` slot of each `xcmsRaw`.

If `metadata` **is NULL:** Metadata is *constructed automatically*:

- `sample_index`: sequential index
- `sample_id`: basename without extension of `sample_path`
- `sample_path`: from `xcmsRaw@filepath`

If `metadata` **is provided:** Must have one row per `xcmsRaw` object. The supplied metadata is used with:

- `sample_index`: row number
- `sample_id`: from `sample_id_column` if present, otherwise basename of path
- `sample_path`: from `xcmsRaw@filepath`

`ExternalDataSource` **input**

Metadata is taken from the `@metadata` slot of the `ExternalDataSource` object. The `metadata` parameter is ignored.

The returned metadata always includes:

- `sample_index`: row number
- `sample_id`: extracted using `sample_id_column`

`LipidSearchSource` **input**

Metadata is taken from the `@metadata` slot of the `LipidSearchSource` object, which already has one row per supplied sample path. The `metadata` parameter is ignored.

The returned metadata always includes:

- `sample_index`: row number
- `sample_id`: from `sample_id_column` if present, otherwise the existing `sample_id` column

`DBIConnection` **input**

Metadata is queried from a Compound Discoverer SQLite database via `get_workflow_input_files()`.

If `metadata` **is NULL:** Returns the query result with:

- `sample_index`: row number
- `sample_id`: from `StudyFileID`
- `sample_path`: from `PhysicalFileName`

If `metadata` **is provided:** It is joined onto the query result using `sample_id_column`.

`get_mz_range` *Compute an m/z range given a ppm tolerance*

Description

Compute an m/z range given a ppm tolerance

Usage

```
get_mz_range(mz, ppm = 5)
```

Arguments

`mz`	A `numeric` value indicating the m/z value to calculate the range for.
`ppm`	A `numeric` value indicating the tolerance in parts per million (ppm). Default is 5.

Value

A `numeric` vector with two values indicating the m/z range.

get_workflow_input_files *Retrieve workflow input files from a Compound Discoverer database*

Description

Extracts the contents of the `WorkflowInputFiles` table from a Compound Discoverer results database.

Usage

```
get_workflow_input_files(conn)
```

Arguments

`conn`	A `DBIConnection` to a Compound Discoverer results database.

Value

A `tibble` containing information about the workflow input files.

get_XCMSnExp_object_example *Get an* `XCMSnExp` *example object from the* `faahKO` *dataset*

Description

Get an `XCMSnExp` example object from the `faahKO` dataset

Usage

```
get_XCMSnExp_object_example(indices = c(1, 2, 3), should_group_peaks = FALSE)
```

Arguments

`indices`	A `numeric` vector of sample indices to select from the `faahKO` CDF files. Defaults to the first three samples.
`should_group_peaks`	A `logical` value indicating whether to group the detected peaks.

Value

An `XCMSnExp` object containing raw data, detected chromatographic peaks, and, if requested, grouped features.

Examples

```
get_XCMSnExp_object_example(indices = 1:5)
```

get_xic_traces_from_compounds *Extract XIC traces associated with selected compounds*

Description

Queries a Compound Discoverer results database to retrieve extracted ion chromatogram (XIC) traces associated with consolidated compounds matching a user-supplied filter expression.

Usage

```
get_xic_traces_from_compounds(conn, compounds_query_str)
```

Arguments

conn	A DBIConnection to a Compound Discoverer results database.
compounds_query_str	A character value giving a filtering expression evaluated on the resulting compound table (e.g. using compound name, formula, retention time, or m/z).

Details

The query joins multiple internal Compound Discoverer tables to link consolidated compounds to reference ions, chromatogram peaks, and XIC trace data.

Value

A tibble containing XIC trace metadata and binary trace data for the selected compounds.

io_close_raw_data *Close open* MsRawReader *connections*

Description

Close open MsRawReader connections

Usage

```
io_close_raw_data(raw_data)
```

Arguments

raw_data	A list of MsRawReader objects (as returned by io_get_raw_data()).

Value

NULL

io_get_raw_data *Get* MsRawReader *objects for a set of sample paths*

Description

Get MsRawReader objects for a set of sample paths

Usage

```
io_get_raw_data(sample_paths)
```

Arguments

sample_paths	A character vector of file paths to the raw MS data files (e.g., .mzML, .mzXML, .CDF, .raw).

Value

A named list of MsRawReader objects, where each element corresponds to a file in sample_paths.

is_cd_node_source *Check whether an object is a Compound Discoverer scripting-node data source*

Description

Check whether an object is a Compound Discoverer scripting-node data source

Usage

```
is_cd_node_source(obj)
```

Arguments

obj	An object to test.

Value

A logical value indicating whether the object is a CompoundDiscovererNodeSource.

is_cd_result — *Check whether an object is a Compound Discoverer database connection*

Description

The object must inherit from `DBIConnection`, and its `dbname` slot must reference a path ending in `.cdResult`.

Usage

```
is_cd_result(obj)
```

Arguments

`obj`	An object to test.

Value

A `logical` value indicating whether the object is a Compound Discoverer database connection.

is_cd_results_path — *Check whether a path corresponds to a Compound Discoverer results file*

Description

Check whether a path corresponds to a Compound Discoverer results file

Usage

```
is_cd_results_path(path)
```

Arguments

`path`	A `character` value giving a file system path.

Value

A `logical` value indicating whether the path corresponds to a Compound Discoverer results directory.

is_lipid_search_source *Check whether an object is a LipidSearch data source*

Description

Check whether an object is a LipidSearch data source

Usage

```
is_lipid_search_source(obj)
```

Arguments

obj	An object to test.

Value

A `logical` value indicating whether the object is a `LipidSearchSource`.

is_mspurity_data *Check whether an object is an msPurity purityA result*

Description

Check whether an object is an msPurity purityA result

Usage

```
is_mspurity_data(obj)
```

Arguments

obj	The input object to check.

Value

A `logical` value.

is_xcms_data *Check whether an object is from XCMS*

Description

Check whether an object is from XCMS

Usage

```
is_xcms_data(obj)
```

Arguments

obj	The input object to check.

Value

A `logical` value indicating whether the object is an XCMS one, i.e. `XCMSnExp` or `MsExperiment`.

is_xcms_processed_data *Check whether an object is an XCMS data container*

Description

Check whether an object is an XCMS data container

Usage

```
is_xcms_processed_data(obj)
```

Arguments

obj	The input object to check.

Value

A `logical` value indicating whether `obj` is an XCMS experiment object.

`iterate_plot_batches` *Iterate on the batches of plots*

Description

`iterate_plot_batches` iterates over batches of plots defined by the `batch_size` parameter passed to the `lcmsPlot` constructor function.

Usage

```
iterate_plot_batches(object, iter_fn)

## S4 method for signature 'lcmsPlotClass,function'
iterate_plot_batches(object, iter_fn)
```

Arguments

`object`	An instance of class `lcmsPlotClass`.
`iter_fn`	The function to apply to each item being iterated on.

Value

`NULL` (called for its side effect).

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

p <- lcmsPlot(raw_files, batch_size = 2) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
  lp_arrange(group_by = "sample_id") +
  lp_legend(position = "bottom") +
  lp_labels(legend = "Sample")

pdf(tempfile(fileext = ".pdf"))
iterate_plot_batches(p, function(plot_obj) {
  print(plot_obj)
})
dev.off()
```

lcmsPlot *Create an* lcmsPlotClass *object*

Description

The lcmsPlotClass class allows a unified approach for the management of LC-MS data for the purpose of visualisation. It includes the options for customising the plot, the LC-MS data, and the underlying plot object. The lcmsPlot function is the main entry point and the preferred approach to creating lcmsPlotClass objects.

Usage

```
lcmsPlot(
  dataset,
  sample_id_column = "sample_id",
  metadata = NULL,
  batch_size = NULL,
  BPPARAM = NULL
)
```

Arguments

dataset	An object of type XCMSnExp, MsExperiment, MZmineSource, or character. If a character vector is supplied, it will be interpreted as a list of mzML paths.
sample_id_column	A character value indicating which column should be used as the sample ID. By default it is "sample_id".
metadata	A data.frame containing the samples metadata in case it is not provided in the dataset object.
batch_size	A numeric value indicating the number of samples per batch. This parameter is necessary when plotting multiple batches using the iterate_plot_batches or next_plot functions.
BPPARAM	A BiocParallelParam object for enabling parallelism. See BiocParallelParam for more information.

Value

An instance of lcmsPlotClass. It will create the necessary internal structures related to the data (data slot) and options (options slot).

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

p <- lcmsPlot(raw_files)
```

lcmsPlotApp *Launch the lcmsPlot interactive Shiny app*

Description

Returns a shiny::shinyApp object that lets users upload one or more raw files (mzML / CDF) and explore them through the existing `lp_*()` plotting API: base-peak / total-ion chromatograms, extracted ion chromatograms, peak density, and spectra by scan index. Each plot tab carries a side Options panel for `lp_facets`, `lp_arrange`, `lp_legend`, and `lp_labels`.

Usage

```
lcmsPlotApp(max_upload_size = 2 * 1024^3)
```

Arguments

`max_upload_size`
: A `numeric` value in bytes setting the maximum per-file upload size. Defaults to `2 * 1024^3` (2 GB) since raw mzML files routinely exceed Shiny's 5 MB default. Set with care if running the app on shared infrastructure.

Details

Per Bioconductor's Shiny guidelines the function does *not* call `shiny::runApp()` itself — callers run the returned app with `shiny::runApp(lcmsPlotApp())` or by printing it at the R prompt.

Requires the optional packages `shiny` and `shinytoastr`; both are declared in `Suggests`. An informative error is raised if either is missing.

Value

A shiny::shinyApp object.

Examples

```
if (interactive()) {
    shiny::runApp(lcmsPlotApp())
}
```

lcmsPlotClass-class *Managing LC-MS data for visualisation*

Description

The `lcmsPlotClass` class allows a unified approach for the management of LC-MS data for the purpose of visualisation. It includes the options for customising the plot, the LC-MS data, and the underlying plot object.

Slots

`options` A `list` to store the plot options.

`data` An instance of class `lcmsPlotDataContainer`.

`history` A `list` to store the applied layers to generate a plot; for internal use.

`plot` A `patchwork` object representing the underlying plot object.

General information

The `lcmsPlotClass` class has been designed to be the entry point for all data and outputs related to the `lcmsPlot` package. The class abstracts away the data handling, making it easier to use `lcmsPlot` with existing data wrappers like `MsExperiment` or `XCMSnExp`.

Preferred usage

The `lcmsPlotClass` class can be used directly to instantiate an object, however the preferred approach is to use the `lcmsPlot` function.

`lcmsPlotDataContainer-class`

A unified storing mechanism for LC-MS data

Description

The `lcmsPlotDataContainer` class allows the storage of different types of LC-MS data. This class can be used independently from the plotting utilities, however the preferred approach is to use it with the `lcmsPlotClass` class.

Slots

`data_obj` The data object. One of: `XCMSnExp`, `MsExperiment`, `MChromatograms`, `XChromatograms`, `XChromatogram`, `XcmsRawList`, `purityA`, or `character` representing mzML paths.

`metadata` A `data.frame` containing the sample metadata.

`chromatograms` A `data.frame` containing the chromatograms.

`mass_traces` A `data.frame` containing the mass traces.

`spectra` A `data.frame` containing the spectra.

`peak_density` A `data.frame` containing peak density curve data and optional feature-group rectangles, as produced by `lp_peak_density()`.

`peak_count_image` A `data.frame` containing the number of chromatographic peaks per sample per retention-time bin, as produced by `lp_peak_count_image()`.

`total_ion_current` A `data.frame` containing the total ion current.

`intensity_maps` A `data.frame` containing the 2D intensity maps representing the distribution of detected peaks across m/z and RT.

`rt_diff` A `data.frame` containing the raw and adjusted RT values.

`feature_metadata` A `data.frame` containing feature/compound annotations attached to datasets through a column called `feature_metadata_id`.

`detected_peaks` A `data.frame` containing the detected peaks from an `XCMSnExp`, `MsExperiment`, or `purityA` object.

`purity_scores` A `data.frame` containing per-scan precursor ion purity scores from a `purityA` object, populated by the msPurity layer functions.

`LipidSearchSource` *Create a data source from a LipidSearch result file*

Description

Reads a LipidSearch result file and builds a data source that can be plotted with `lcmsPlot()`. Both **LipidSearch 4.2** and **5.2** exports are supported; the version is auto-detected. Chromatograms are later extracted from the raw files supplied via `sample_paths`.

Usage

```
LipidSearchSource(
  results_path,
  sample_paths,
  metadata = NULL,
  keep_rejected = FALSE
)
```

Arguments

`results_path`	A `character` path to a LipidSearch result file (4.2 or 5.2).
`sample_paths`	A `character` vector of paths to the raw sample files (e.g. `.raw` or `.mzML`) used to extract chromatograms. For 5.2 files this may be named by sample key (see *Sample-to-file mapping*).
`metadata`	A `data.frame` (optional) of additional sample metadata. It must contain one row per sample, aligned with `sample_paths`. When supplied, its columns are column-bound onto the generated metadata.
`keep_rejected`	A `logical` value indicating whether lipids flagged as rejected (`Rej./Rej`) should be retained. Defaults to `FALSE`.

Value

An object of class `LipidSearchSource`.

Sample-to-file mapping

A LipidSearch 4.2 file names the analysed raw files in its header (`#[c-1]:sample.raw`), so paths are matched to samples by file basename without extension - a result file that lists ThermoFisher `.raw` files works with converted `.mzML` files. A LipidSearch 5.2 file carries **no** filenames; its samples are keyed `s1`, `s2`, ... (from the `OrgMeanArea[...]` columns). For 5.2, supply either a **named** `sample_paths` vector whose names are those keys (`c("s1" = "a.mzML", "s2" = "b.mzML")`, order-independent, any subset), or an **unnamed** vector matched positionally (`s1`, `s2`, ... in order).

Samples beyond the result file

`sample_paths` determines which samples are plotted. A path that matches no sample (an unnamed extra for 5.2, or an unmatched basename for 4.2) is still a full sample: for every queried lipid its chromatogram is extracted using the lipid's consensus m/z and retention-time window (the median across the samples in which LipidSearch did detect it, falling back to the reported `BaseRt` for 5.2), it simply has no reported peak to highlight.

See Also

lp_lipid_search()

Examples

```
## Build a minimal LipidSearch 4.2 export. The raw files are declared in the
## header block, and every row - including the column header - ends with a
## tab, as in a real export.
results_path <- file.path(tempdir(), "lipids_pos.txt")

columns <- c(
    "Rej.", "LipidIon", "LipidGroup", "Class", "FattyAcid", "CalcMz",
    "IonFormula",
    "Area[c-1]", "Area[c-2]", "Height[c-1]", "Height[c-2]",
    "Rt[c-1]", "Rt[c-2]", "ObsMz[c-1]", "ObsMz[c-2]",
    "Grade[c-1]", "Grade[c-2]")

## One lipid detected in each of the two samples.
rows <- list(
    c(0, "PRO(0:0)+H", "PRO(0:0)", "PRO", "(0:0)", 116.0706,
      "C5 H10 O2 N1", 24050, 0, 24050, 0, 7.10, 0,
      "116.0704", "", "A", ""),
    c(0, "KYN(0:0)+H", "KYN(0:0)", "KYN", "(0:0)", 209.0921,
      "C10 H13 O3 N2", 0, 18700, 0, 18700, 0, 6.10,
      "", "209.0918", "", "A"))

writeLines(
    c("#[c-1]:l-proline-MS1.raw",
      "#[c-2]:l-kynurenine-MS1.raw",
      "#mScoreThreshold:5.0",
      "",
      vapply(
          c(list(columns), rows),
          function(x) paste0(paste(x, collapse = "\t"), "\t"),
          character(1))),
    results_path)

## Chromatograms are extracted from the raw files, not from the result file.
mzml_dir <- file.path(tempdir(), "lipid-search-example")
utils::unzip(
    system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
    exdir = mzml_dir)

sample_paths <- file.path(
    mzml_dir, "mzml",
    c("l-proline-MS1.mzML", "l-kynurenine-MS1.mzML"))

## 4.2: sample files are matched to the header declarations by basename.
ds <- LipidSearchSource(
    results_path = results_path,
    sample_paths = sample_paths)

ds

p <- lcmsPlot(ds) +
    lp_lipid_search(rt_extend = 30) +
```

```
    lp_chromatogram(highlight_peaks = TRUE) +
    lp_grid(rows = "sample_id", cols = "name", free_x = TRUE)
p
```

LipidSearchSource-class *LipidSearch data source*

Description

An S4 class wrapping the lipid annotations exported by LipidSearch.

Slots

`name` A `character` value identifying the data source.

`version` A `character` value giving the detected LipidSearch format, `"4.2"` or `"5.2"`.

`metadata` A `data.frame` of sample metadata (one row per supplied sample path), including a `sample_path` column with the raw file paths and an `in_results` flag marking samples declared in the result file.

`peaks` A `data.frame` of detected peaks (one row per lipid and sample in which LipidSearch integrated a peak) with the standard `mz`, `rt`, `rtmin`, `rtmax`, `into`, `maxo` and `sample_index` columns, plus `name`, `formula`, `adduct`, `class` and `grade` annotations.

`compounds` A `data.frame` with one row per lipid and sample - including samples absent from the result file - used to drive chromatogram extraction.

---

lp_arrange *Define the arrangement of chromatograms*

Description

The `lp_arrange` function specifies how chromatograms should be arranged when visualised. It determines the grouping metadata factor through the `group_by` parameter.

Usage

```
lp_arrange(group_by)
```

Arguments

`group_by`	A `character` value determining the column to group by in the samples metadata.

Value

A function that takes an `lcmsPlot` object and returns a modified version with the specified arrangement options stored in `options$arrangement`. It is intended for use with the `+` operator, which incrementally layers new data or visual components onto the `lcmsPlot` object.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

## Plots chromatograms overlayed without specifying a grouping factor
p <- lcmsPlot(raw_files) +
  lp_chromatogram(aggregation_fun = "max")
p

## Plots chromatograms overlayed specifying a grouping factor
## (e.g., sample_id)
p <- p + lp_arrange(group_by = "sample_id")
p
```

`lp_chromatogram` *Define the chromatograms to plot*

Description

The `lp_chromatogram` function allows the generation of different types of chromatograms.

Usage

```
lp_chromatogram(
  features = NULL,
  sample_ids = NULL,
  ppm = 10,
  rt_tol = 10,
  line_type = "solid",
  highlight_peaks = FALSE,
  highlight_peaks_color = NULL,
  highlight_peaks_mode = "polygon",
  highlight_peaks_factor = "sample_id",
  aggregation_fun = "max",
  rt_type = "uncorrected",
  rt_unit = "second",
  intensity_unit = "absolute",
  fill_gaps = FALSE,
  na.rm = FALSE,
  highlight_apices = list(column = NULL, top_n = NULL),
  stacked = 0,
  transform = identity
)
```

Arguments

`features`	Specifies which features to generates the chromatogram for. This can be either: a `matrix` with columns `mz` and `rt` (optional); a `matrix` with columns `mzmin`, `mzmax`, `rtmin` (optional), `rtmax` (optional); a `data.frame` with columns `sample_id`, `mz` and `rt` (optional); a `data.frame` with columns `sample_id`,

mzmin, mzmax, rtmin (optional), rtmax (optional); a character vector representing the grouped peaks (feature) names as returned by xcms::groupnames - requires the data to be an XCMSnExp or MsExperiment object with grouped peaks.

sample_ids A character vector specifying the sample IDs to include in the plot. If NULL, the function uses the sample IDs specified in the lcmsPlot object.

ppm A numeric value specifying the mass accuracy (in ppm) used when generating chromatograms. Ignored when the features parameter specifies both mzmin and mzmax.

rt_tol A numeric value specifying the RT tolerance used when generating chromatograms. Ignored when the features parameter specifies both rtmin and rtmax.

line_type A character value specifying the line type (from ggplot2). One of: "solid", "dashed", "dotted", "dotdash", "longdash", "twodash".

highlight_peaks
A logical value indicating whether to highlight the detected peaks; the input data must be an XCMSnExp or MsExperiment object.

highlight_peaks_color
A character value indicating the color of the highlighted peaks.

highlight_peaks_mode
A character value indicating how the peaks should be highlighted. One of "polygon" (fills the area under the curve), "rectangle" (draws a bounding box from rtmin to rtmax up to the peak apex), or "point" (marks the peak apex with a point). Defaults to "polygon".

highlight_peaks_factor
A character value indicating the factor from the metadata that determines the color. By default it colors by sample_id.

aggregation_fun
A character value indicating which aggregation function to use for the spectra intensities; one of max (base peak chromatogram), sum (total ion current), or mean (averaged ion chromatogram). Only applicable to summary chromatograms.

rt_type A chracter value indicating what type of RT to use for the chromatograms. One of uncorrected (default), corrected, or both; the input data must be an XCMSnExp or MsExperiment object. If both is chosen, this will give access to a metadata column called rt_adjusted that can be used to differentiate the two RT types (e.g., through faceting).

rt_unit A character value indicating the unit to use for the RT axis; one of "minute" or "second".

intensity_unit A character value indicating the unit to use for the intensity axis; one of "absolute" or "relative".

fill_gaps A logical value indicating whether to fill gaps in RT with 0 intensity.

na.rm A logical value. When TRUE, data points whose intensity is NA are removed before plotting. Defaults to FALSE.

highlight_apices
A logical value indicating whether to highlight apices with the corresponding RT values in a chromatogram.

stacked A numeric value in [0, 1]. When non-zero, each series in a panel is offset up the y axis in m/z order by that fraction of the intensity range, so co-eluting traces stop occluding each other. Defaults to 0 (no stacking). Which series get offset follows lp_arrange(group_by =); without it, samples are stacked.

`transform` A `function` applied to the intensities before plotting, such as `log10`, to compress the dynamic range. It is applied to the peak highlight geometry as well, so shaded peaks stay attached to their traces. Defaults to `identity`. Note that `log10` maps the exact zeros injected by `fill_gaps = TRUE` to `-Inf`.

Value

This function returns another function that takes an `lcmsPlot` object and produces a modified version containing the generated chromatograms in its `data` slot. It is designed to be used with the `+` operator, which serves as a layering mechanism. Each use of `+` incrementally enriches the `lcmsPlot` object by adding new data or visual components.

Summary chromatograms

In this type of chromatogram, the intensities of the spectra from each scan in an LC–MS dataset are aggregated into a single value per scan. To create such chromatograms do not specify the `features` parameter as that will create the chromatograms for the selected features. In this context, the main parameter is `aggregation_fun`, which can take `max` (base peak chromatogram, BPC), `sum` (total ion current, TIC), or `mean` (averaged ion chromatogram).

Feature chromatograms

A feature is a combination of retention time (RT) and m/z. Feature chromatograms can be created by specifiying the `features` parameter.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

p <- lcmsPlot(raw_files) +
  lp_chromatogram(aggregation_fun = "max") +
  lp_arrange(group_by = "sample_id")

p
```

---

lp_chrom_peak_rects *Plot chromatographic peak boundaries in the rt / m/z plane*

Description

`lp_chrom_peak_rects()` draws one rectangle per detected chromatographic peak, spanning `rtmin`-`rtmax` by `mzmin`-`mzmax`. It works in two ways, matching the two xcms methods that draw the same primitive.

Usage

```
lp_chrom_peak_rects(
  sample_ids = NULL,
  border = "#c0392b",
  fill = NA,
  alpha = 0.25,
  linewidth = 0.3,
  rt_range = NULL,
  mz_range = NULL
)
```

Arguments

sample_ids	A character vector of sample IDs to include. NULL (default) follows the host layer when there is one, so the overlay never introduces panels the host does not draw, and otherwise uses every sample.
border	A character colour for the rectangle outline.
fill	A character colour for the rectangle interior, or NA (default) for unfilled boxes.
alpha	A numeric value in [0, 1] controlling opacity.
linewidth	A numeric value controlling the outline width.
rt_range	An optional length-2 numeric restricting the retention-time range, equivalent to xlim in xcms::plotChromPeaks(). Ignored when the layer decorates an lp_intensity_map() panel, which supplies its own window.
mz_range	An optional length-2 numeric restricting the m/z range, equivalent to ylim in xcms::plotChromPeaks(). Ignored when the layer decorates an lp_intensity_map() panel.

Details

Called on its own it owns a panel, drawing the rectangles on an otherwise empty retention time / m/z frame as xcms::plotChromPeaks() does, with one facet per sample. Called after lp_mass_trace() or lp_intensity_map() it instead decorates that panel, as xcms::plot(type = "XIC") does, and follows the host's axes, window and samples. On an intensity-versus-rt panel the equivalent is lp_chromatogram(highlight_peaks = TRUE, highlight_peaks_mode = "rectangle").

The data object has to report chromatographic peaks. On m/z-binned data mzmin equals mzmax, so each box is given a minimum height and reads as a horizontal segment rather than vanishing.

Value

A layer function for use with the + operator.

Examples

```
data_obj <- get_XCMSnExp_object_example(
  indices = 1:2,
  should_group_peaks = TRUE)

## On its own the layer owns the panel, one facet per sample.
p <- lcmsPlot(data_obj, sample_id_column = "sample_name") +
  lp_chrom_peak_rects(rt_range = c(2500, 3500), mz_range = c(300, 320))
p
```

```
## After a host layer it decorates that panel instead.
p <- lcmsPlot(data_obj, sample_id_column = "sample_name") +
  lp_intensity_map(mz_range = c(300, 320), rt_range = c(2500, 3500)) +
  lp_chrom_peak_rects()
p
```

`lp_compound_discoverer` *Define the options to use when plotting LC-MS data coming from Compound Discoverer results.*

Description

Define the options to use when plotting LC-MS data coming from Compound Discoverer results.

Usage

```
lp_compound_discoverer(compounds_query = NULL, rt_extend = 10)
```

Arguments

`compounds_query`
: A `character` value indicating the expression used to filter compounds from the Compound Discoverer results. The expression is evaluated on the compound table and can reference the following columns:

  name Compound name.

  formula Chemical formula of the compound.

  adduct Ion adduct (e.g. `[M+H]+`, `[M-H]-`).

  rt Retention time of the compound (in seconds).

  rtmin Minimum retention time of the compound peak.

  rtmax Maximum retention time of the compound peak.

  mz Mass-to-charge ratio (m/z) of the detected ion.

  maxo Maximum observed peak intensity.

  into Integrated peak area reported by Compound Discoverer.

`rt_extend`
: A `numeric` value indicating how much (in seconds) the retention time window should be extended on each side of the compound peak when extracting and plotting chromatograms.

Value

A function that takes an `lcmsPlot` object and returns a modified version with the specified Compound Discoverer options stored in `options$compound_discoverer`. It is intended for use with the + operator, which incrementally layers new data or visual components onto the `lcmsPlot` object.

Examples

```
## A minimal stand-in for a Compound Discoverer Scripting Node export.
## See [CompoundDiscovererNodeSource()] for the table layout.
node_dir <- tempfile("cd_node")
dir.create(node_dir)

write_tab <- function(df, name) {
    path <- file.path(node_dir, name)
    utils::write.table(
        df, path, sep = "\t", row.names = FALSE, quote = FALSE)
    path
}

node_args <- list(Tables = list(
    list(DataFile = write_tab(
        data.frame(
            "Compounds ID" = c(1, 2),
            "Name" = c("L-Proline", "L-Kynurenine"),
            "Formula" = c("C5 H9 N O2", "C10 H12 N2 O3"),
            check.names = FALSE),
        "compounds.txt")),
    list(DataFile = write_tab(
        data.frame(
            "Compounds per File ID" = c(1, 2),
            "StudyFileID" = c(1, 2),
            "Area" = c(24050, 18700),
            "Intensity" = c(24050, 18700),
            check.names = FALSE),
        "cpf.txt")),
    list(DataFile = write_tab(
        data.frame(
            "Features ID" = c(1, 2),
            "mz" = c(116.0704, 209.0918),
            "Ion" = c("[M+H]+1", "[M+H]+1"),
            "Area" = c(24050, 18700),
            "RT [min]" = c(7.10, 6.10),
            "Left RT [min]" = c(6.95, 5.95),
            "Right RT [min]" = c(7.25, 6.25),
            check.names = FALSE),
        "features.txt")),
    list(DataFile = write_tab(
        data.frame(
            "Compounds ID" = c(1, 2),
            "Compounds per File ID" = c(1, 2),
            check.names = FALSE),
        "link_cmp_cpf.txt")),
    list(DataFile = write_tab(
        data.frame(
            "Compounds per File ID" = c(1, 2),
            "Features ID" = c(1, 2),
            check.names = FALSE),
        "link_cpf_feat.txt")),
    list(DataFile = write_tab(
        data.frame(
            "StudyFileID" = c(1, 2),
            "File Name" = c("l-proline-MS1.mzML", "l-kynurenine-MS1.mzML"),
```

```
            check.names = FALSE),
        "study_files.txt"))))

json_path <- file.path(node_dir, "node_args.json")
writeLines(jsonlite::toJSON(node_args, auto_unbox = TRUE), json_path)

mzml_dir <- file.path(tempdir(), "cd-node-example")
utils::unzip(
    system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
    exdir = mzml_dir)

ds <- CompoundDiscovererNodeSource(
    node_args = json_path,
    sample_paths = file.path(
        mzml_dir, "mzml",
        c("l-proline-MS1.mzML", "l-kynurenine-MS1.mzML")))

## `compounds_query` is evaluated on the compound table.
p <- lcmsPlot(ds) +
  lp_compound_discoverer(
    compounds_query = 'name %in% c("L-Proline", "L-Kynurenine")',
    rt_extend = 15
  ) +
  lp_chromatogram(highlight_peaks = TRUE) +
  lp_grid(rows = "sample_id", cols = "name", free_x = TRUE) +
  lp_labels(title = "Compound Discoverer example", legend = "Sample") +
  lp_legend(position = "bottom")
p
```

`lp_facets` *Define the plot's faceting*

Description

The `lp_facets` function arranges plots into a grid based on a metadata factor, creating a series of smaller plots (facets).

Usage

```
lp_facets(facets, ncol = NULL, nrow = NULL, free_x = FALSE, free_y = FALSE)
```

Arguments

`facets`	A `character` vector of factors from the sample metadata to use for faceting.
`ncol`	A `numeric` value indicating the number of columns in the layout.
`nrow`	A `numeric` value indicating the number of rows in the layout.
`free_x`	A `logical` value indicating whether the x-axis scales are allowed to vary across panels.
`free_y`	A `logical` value indicating whether the y-axis scales are allowed to vary across panels.

Value

A function that takes an `lcmsPlot` object and returns a modified version with the specified faceting options stored in `options$facets`. It is intended for use with the + operator, which incrementally layers new data or visual components onto the `lcmsPlot` object.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

## Plots chromatograms overlayed
p <- lcmsPlot(raw_files) +
  lp_chromatogram(aggregation_fun = "max")
p

## Using lp_facets we create facets for each sample_id
p <- p + lp_facets(facets = "sample_id")
p
```

---

lp_get_plot *Get the underlying plot object.*

Description

Get the underlying plot object.

Usage

```
lp_get_plot()
```

Value

A function that takes an `lcmsPlot` object and returns a modified version with the rendered plot stored in the `plot` slot. It is intended for use with the + operator, which incrementally layers new data or visual components onto the `lcmsPlot` object.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:4]

## Create faceted chromatogram plots with a reference RT line
p <- lcmsPlot(raw_files) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
  lp_facets(facets = 'sample_id', ncol = 4) +
```

```
  lp_rt_line(intercept = 2800, line_type = 'solid', color = 'red')
p

## Extract the ggplot object and apply a theme
p <- p +
  lp_get_plot() +
  ggplot2::theme_bw()
p
```

---

lp_grid *Define a gridded plot*

Description

The lp_grid function arranges plots into a matrix of panels defined by row and column faceting metadata factors.

Usage

```
lp_grid(rows, cols, free_x = FALSE, free_y = FALSE)
```

Arguments

rows	A character value indicating the factors that represent rows.
cols	A character value indicating the factors that represent columns.
free_x	A logical value indicating whether the x-axis scales are allowed to vary across panels.
free_y	A logical value indicating whether the y-axis scales are allowed to vary across panels.

Value

A function that takes an lcmsPlot object and returns a modified version with the specified grid options stored in options$grid. It is intended for use with the + operator, which incrementally layers new data or visual components onto the lcmsPlot object.

Examples

```
raw_files <- dir(
  system.file("cdf", package = "faahKO"),
  full.names = TRUE,
  recursive = TRUE
)[1:4]

## Create metadata for the samples
metadata <- data.frame(
  sample_id = sub("\\.CDF", "", basename(raw_files)),
  factor1 = c("S", "S", "C", "C"),
  factor2 = c("T", "U", "T", "U")
)

## Create feature chromatograms for the specified samples
```

```
p <- lcmsPlot(raw_files, metadata = metadata) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900)))
p

## Arrange chromatograms in a grid split by experimental factors
## Rows correspond to `factor1` and columns correspond to `factor2`
p <- p + lp_grid(rows = "factor1", cols = "factor2")
p
```

`lp_intensity_map` *Define a 2D intensity map*

Description

The `lp_intensity_map` function produces an intensity map in which signal intensity is represented at each m/z / RT coordinate.

Usage

```
lp_intensity_map(
  mz_range,
  rt_range,
  sample_ids = NULL,
  x_dim = "rt",
  y_dim = "mz",
  fill_scale = NULL,
  geom = "tile",
  point_size = 0.5,
  bin_rt = 0.1,
  bin_mz = 0.1,
  colour_scale = NULL
)
```

Arguments

`mz_range`	A `numeric` value indicating the m/z range of the map.
`rt_range`	A `numeric` value indicating the RT range of the map.
`sample_ids`	A `character` vector specifying the sample IDs to include in the plot. If `NULL`, the function uses the sample IDs specified in the `lcmsPlot` object.
`x_dim`	A `character` value indicating which dimension to place on the x-axis. One of `"rt"` (default) or `"mz"`.
`y_dim`	A `character` value indicating which dimension to place on the y-axis. One of `"mz"` (default) or `"rt"`.
`fill_scale`	A ggplot2 scale object to use for the fill aesthetic (e.g. `scale_fill_gradient(low = "white", high = "red")`). When `NULL` (default), uses `scale_fill_viridis_c` for intensity and `scale_fill_viridis_d` for density plots. Applies to `geom = "tile"` and `geom = "density"`.

geom	A character value selecting how the map is drawn. One of "tile" (default) for a binned heatmap, "point" for a scatter of the individual centroids, or "density" for a 2D kernel density estimate. "point" skips the binning entirely, so gaps in the mass traces stay visible rather than being implied away.
point_size	A numeric value controlling the point size when geom = "point".
bin_rt	A numeric value giving the retention-time bin width in seconds used when geom = "tile". Defaults to 0.1, which is too fine for slow-scanning data and too coarse for fast-scanning data.
bin_mz	A numeric value giving the m/z bin width used when geom = "tile". Defaults to 0.1; high-resolution data usually wants less.
colour_scale	A ggplot2 scale object to use for the colour aesthetic when geom = "point". When NULL (default), uses scale_colour_viridis_c.

Value

This function returns another function that takes an lcmsPlot object and produces a modified version containing the generated 2D intensity map in its data slot. It is designed to be used with the + operator, which serves as a layering mechanism. Each use of + incrementally enriches the lcmsPlot object by adding new data or visual components.

Examples

```
raw_files <- dir(
  system.file("cdf", package = "faahKO"),
  full.names = TRUE,
  recursive = TRUE)[1]

p <- lcmsPlot(raw_files) +
  lp_intensity_map(
    mz_range = c(200, 600),
    rt_range = c(4200, 4500),
    geom = "density")
p
```

---

lp_isolation_window *Visualise the isolation window for a precursor fragmentation event*

Description

lp_isolation_window() extracts the MS1 survey spectrum at the scan immediately preceding a chosen MS2 event from a purityA object and annotates it with a semi-transparent rectangle spanning the isolation window and a dashed line at the precursor m/z. The inPurity score is shown as a plot subtitle.

Usage

```
lp_isolation_window(
  pid = NULL,
  sample_id = NULL,
  half_width = 0.5,
  zoom_factor = 3
)
```

Arguments

pid	An integer or integer vector of precursor IDs (the pid column in purityA@puritydf) selecting which event(s) to display. NULL shows all events (use with care on large datasets).
sample_id	A character sample ID to filter by when pid is NULL.
half_width	A numeric value (in Da) for the isolation window half-width on each side of the precursor m/z. Default is 0.5.
zoom_factor	A numeric multiplier controlling how far beyond the isolation window the x-axis extends. The visible range is precursor_mz ± zoom_factor * half_width. Default is 3.

Value

A layer function for use with the + operator.

Examples

```
mzml_dir <- file.path(tempdir(), "purity-example")
utils::unzip(
    system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
    exdir = mzml_dir)

ms2_files <- file.path(
    mzml_dir, "mzml",
    c("l-proline-MS2.mzML", "l-kynurenine-MS2.mzML"))

pa <- msPurity::purityA(ms2_files)

## Inspect the fragmentation events available to plot.
slot(pa, "puritydf")[, c("pid", "precursorMZ", "precursorRT", "inPurity")]

## `pid = 4` is a co-isolated precursor (inPurity ~ 0.51). These files
## record an isolation width of 0.6 Da on each side of the precursor.
p <- lcmsPlot(pa) +
    lp_isolation_window(pid = 4, half_width = 0.6)
p
```

---

lp_labels *Define the labels of the plot*

Description

The lp_labels function allows the specification of the plot title and the legend title.

Usage

```
lp_labels(title = NULL, legend = NULL)
```

Arguments

title	A character value indicating the plot title.
legend	A character value indicating the legend's title.

Value

A function that takes an `lcmsPlot` object and returns a modified version with the specified label options stored in `options$labels`. It is intended for use with the + operator, which incrementally layers new data or visual components onto the `lcmsPlot` object.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

## Create a chromatogram plot by grouping samples into batches
## By default, the legend is derived from the grouping variable
p <- lcmsPlot(raw_files, batch_size = 2) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
  lp_arrange(group_by = "sample_id")
p

## Customise the legend label
p <- p + lp_labels(legend = "Sample")
p
```

`lp_layout` *Define the plot layout*

Description

Define the plot layout

Usage

```
lp_layout(design = NULL)
```

Arguments

`design`	Specification of the location of areas in the layout See https://patchwork.data-imaginist.com/reference/wrap_plots.html

Value

A function that takes an `lcmsPlot` object and returns a modified version with the specified layout options stored in `options$layout`. It is intended for use with the + operator, which incrementally layers new data or visual components onto the `lcmsPlot` object.

Examples

```
data_obj <- get_XCMSnExp_object_example(indices = 1)

## Plot chromatograms and spectra for selected samples and features
p <- lcmsPlot(data_obj, sample_id_column = 'sample_name') +
  lp_chromatogram(
    features = rbind(
      c(mzmin = 334.9, mzmax = 335.1, rtmin = 2700, rtmax = 2900),
      c(mzmin = 278.99721, mzmax = 279.00279, rtmin = 2740, rtmax = 2840)
    ),
    sample_ids = 'ko15',
    highlight_peaks = TRUE
  ) +
  lp_spectra(mode = "closest_apex", ms_level = 1) +
  lp_facets(facets = "feature_id", ncol = 2)

## Customise panel layout to place chromatogram above spectra
p <- p + lp_layout(design = "C\nS\nS")
```

---

lp_legend *Define the legend layout*

Description

Define the legend layout

Usage

```
lp_legend(position = NULL)
```

Arguments

position	A character value indicating the legend's position. One of "top", "right", "bottom", "left", or "inside".

Value

A function that takes an lcmsPlot object and returns a modified version with the specified legend options stored in options$legend. It is intended for use with the + operator, which incrementally layers new data or visual components onto the lcmsPlot object.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

## Create a chromatogram plot grouped by sample with a custom legend label
p <- lcmsPlot(raw_files, batch_size = 2) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
```

```
    rtmin = 2700,
    rtmax = 2900))) +
  lp_arrange(group_by = "sample_id") +
  lp_labels(legend = "Sample")
p

## Move the legend below the plot
p <- p + lp_legend(position = "bottom")
p
```

`lp_lipid_search` *Define the options to use when plotting LC-MS data coming from a LipidSearch result file.*

Description

Define the options to use when plotting LC-MS data coming from a LipidSearch result file.

Usage

```
lp_lipid_search(lipids_query = NULL, rt_extend = 30)
```

Arguments

`lipids_query` A `character` value indicating the expression used to select which lipids to plot. The expression is evaluated on the lipid table and can reference the following columns:

name Plot label: the lipid ion, disambiguated with its retention time when the same ion is reported more than once.

lipid_ion Lipid ion as reported by LipidSearch, e.g. `AEA(18:2)+H` (4.2 `LipidIon` / 5.2 `LipidID`).

lipid_group LipidSearch lipid group.

class Lipid class, e.g. `PC`, `AcCa`.

sub_class Lipid subclass (5.2 only; `NA` for 4.2).

fatty_acid Fatty acid composition (4.2 only; `NA` for 5.2).

formula Ion formula.

adduct Adduct: the explicit `AdductIon` column for 5.2 (e.g. `M+H`), otherwise derived from the ion name for 4.2 (e.g. `+H`).

calc_mz Theoretical m/z.

rej LipidSearch rejection flag as a `logical` (`TRUE` if rejected).

mz m/z used for extraction (observed where available).

rt, rtmin, rtmax Retention time and peak bounds, in seconds.

into Integrated peak area; `NA` where nothing was detected.

maxo Peak height; `NA` where nothing was detected.

grade LipidSearch identification grade (`A` to `D`).

mscore, sn LipidSearch m-score / ID score and signal-to-noise (`sn` is 4.2 only).

detected Whether LipidSearch integrated a peak in this sample.

lipid_rank Dense rank of lipids by descending total area, for top-N selection.

The query selects *lipids*, not lipid/sample rows: a predicate such as `grade == "A"` keeps the lipids graded A in at least one sample, and each of them is then extracted from **every** sample - including samples that the result file does not cover, which use the lipid's consensus m/z and retention-time window.

`rt_extend` A `numeric` value indicating how much (in seconds) the retention time window should be extended on each side of the lipid peak when extracting and plotting chromatograms.

Value

A function that takes an `lcmsPlot` object and returns a modified version with the specified LipidSearch options stored in `options$lipid_search`. It is intended for use with the `+` operator, which incrementally layers new data or visual components onto the `lcmsPlot` object.

See Also

`LipidSearchSource()`

Examples

```
## A minimal LipidSearch 4.2 export; see [LipidSearchSource()] for the
## format details.
results_path <- file.path(tempdir(), "lipids_pos.txt")

columns <- c(
    "Rej.", "LipidIon", "LipidGroup", "Class", "FattyAcid", "CalcMz",
    "IonFormula",
    "Area[c-1]", "Area[c-2]", "Height[c-1]", "Height[c-2]",
    "Rt[c-1]", "Rt[c-2]", "ObsMz[c-1]", "ObsMz[c-2]",
    "Grade[c-1]", "Grade[c-2]")

rows <- list(
    c(0, "PRO(0:0)+H", "PRO(0:0)", "PRO", "(0:0)", 116.0706,
      "C5 H10 O2 N1", 24050, 0, 24050, 0, 7.10, 0,
      "116.0704", "", "A", ""),
    c(0, "KYN(0:0)+H", "KYN(0:0)", "KYN", "(0:0)", 209.0921,
      "C10 H13 O3 N2", 0, 18700, 0, 18700, 0, 6.10,
      "", "209.0918", "", "A"))

writeLines(
    c("#[c-1]:l-proline-MS1.raw",
      "#[c-2]:l-kynurenine-MS1.raw",
      "#mScoreThreshold:5.0",
      "",
      vapply(
          c(list(columns), rows),
          function(x) paste0(paste(x, collapse = "\t"), "\t"),
          character(1))),
    results_path)

mzml_dir <- file.path(tempdir(), "lipid-search-example")
utils::unzip(
    system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
    exdir = mzml_dir)

ds <- LipidSearchSource(
```

```
    results_path = results_path,
    sample_paths = file.path(
        mzml_dir, "mzml",
        c("l-proline-MS1.mzML", "l-kynurenine-MS1.mzML")))

## `lipids_query` selects lipids, which are then extracted from every sample.
p <- lcmsPlot(ds) +
  lp_lipid_search(
    lipids_query = 'grade == "A"',
    rt_extend = 30
  ) +
  lp_chromatogram(highlight_peaks = TRUE) +
  lp_grid(rows = "sample_id", cols = "name", free_x = TRUE) +
  lp_labels(title = "LipidSearch example", legend = "Sample") +
  lp_legend(position = "bottom")
p
```

`lp_mass_trace` *Define the mass trace to plot*

Description

The `lp_mass_trace` function enables the generation of mass traces, which are graphical representations commonly used in mass spectrometry data analysis. A mass trace plots individual data points defined by their retention time and corresponding mass-to-charge ratio (m/z), making it easier to visualise how specific ions behave over the course of a chromatographic run.

Usage

```
lp_mass_trace()
```

Value

This function returns another function that takes an `lcmsPlot` object and produces a modified version containing the generated mass traces in its `data` slot. It is designed to be used with the `+` operator, which serves as a layering mechanism. Each use of `+` incrementally enriches the `lcmsPlot` object by adding new data or visual components.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

## Create chromatograms of a specific feature
p <- lcmsPlot(raw_files) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
  lp_arrange(group_by = "sample_id")
```

```
## Add mass traces
p <- p + lp_mass_trace()

p
```

`lp_peak_count_image` *Plot chromatographic peak counts per retention-time bin*

Description

`lp_peak_count_image()` produces a common quality-control view: retention-time bins on the x axis, samples on the y axis, and fill showing how many chromatographic peaks each sample yielded in each bin.

Usage

```
lp_peak_count_image(
  bin_size = 30,
  log = FALSE,
  sample_ids = NULL,
  rt_range = NULL,
  fill_scale = NULL
)
```

Arguments

`bin_size`	A `numeric` value giving the retention-time bin width in seconds. Defaults to `30`.
`log`	A `logical` value. When `TRUE`, counts are shown on a `log2` scale, which is the usual way to read the plot when counts are skewed. Empty bins are left blank rather than being drawn as `-Inf`.
`sample_ids`	A `character` vector of sample IDs to include. `NULL` uses all samples.
`rt_range`	An optional length-2 `numeric` limiting the binned retention-time range. When `NULL` (default) the full acquisition range of the data object is used, so a short run shows as empty bins at the edge.
`fill_scale`	A ggplot2 scale object to use for the fill aesthetic. When `NULL` (default), uses `scale_fill_viridis_c`.

Details

Samples are ordered by injection order, and bins containing no peaks are drawn as zero rather than dropped.

Value

This function returns another function that takes an `lcmsPlot` object and produces a modified version containing the generated peak counts in its `data` slot. It is designed to be used with the `+` operator, which serves as a layering mechanism.

Examples

```
data_obj <- get_XCMSnExp_object_example(
  indices = 1:3,
  should_group_peaks = TRUE)

p <- lcmsPlot(data_obj, sample_id_column = "sample_name") +
  lp_peak_count_image(bin_size = 30)
p
```

`lp_peak_density` *Plot the peak density for one or more m/z features*

Description

The `lp_peak_density` function produces a peak density plot. For each supplied feature (m/z bin):

- The x axis shows retention time.
- The y axis shows sample indices (1 to n), positioned within the density range.
- Detected peaks are plotted as coloured points at each sample's position.
- The kernel density estimate of peak apex RTs is drawn as a line.
- When `min_fraction` is supplied, the density-descent grouping algorithm is simulated and feature groups that pass the threshold are highlighted with semi-transparent rectangles.

Usage

```
lp_peak_density(
  features = NULL,
  bw = 30,
  min_fraction = NULL,
  min_samples = 1L,
  sample_groups = NULL,
  max_features = 50L,
  rt_unit = "second",
  simulate = NULL
)
```

Arguments

`features`	A `matrix` or `data.frame` with columns `mzmin` and `mzmax` (required) and `rtmin`, `rtmax` (optional). Each row defines one m/z bin. When omitted, the features are taken from `lp_chromatogram` if it has already been called on the same object. For `XChromatograms` and `XChromatogram` objects it is redundant: each extracted ion chromatogram already carries its own m/z and retention-time window, which is used instead.
`bw`	A `numeric` value specifying the kernel density bandwidth in seconds. Passed to `stats::density()`. Defaults to `30`.
`min_fraction`	A `numeric` value in `[0, 1]`. When supplied, simulates the `PeakDensityParam` grouping algorithm and draws semi-transparent rectangles for feature groups where at least this fraction of samples (per group) contain a peak. Defaults to `NULL` (no simulation).

min_samples	An integer specifying the minimum absolute number of samples per group required to define a feature group. Only used when min_fraction is set. Defaults to 1L.
sample_groups	A vector of length equal to the number of samples assigning each sample to a group (as in PeakDensityParam). Defaults to NULL, which treats all samples as a single group.
max_features	An integer specifying the maximum number of feature group rectangles to draw per m/z bin. Defaults to 50L.
rt_unit	A character value indicating the unit for the RT axis; one of "second" (default) or "minute".
simulate	A logical value mirroring xcms::plotChromPeakDensity(). TRUE descends the density curve to derive the feature groups the supplied parameters *would* produce, using min_fraction (defaulting to 0.5 when unset). FALSE instead draws the feature definitions the object already stores, ignoring min_fraction and min_samples. Defaults to NULL, which simulates whenever min_fraction is given.

Value

This function returns another function that takes an lcmsPlot object and produces a modified version containing the generated peak density data in its data slot. It is designed to be used with the + operator, which serves as a layering mechanism.

Examples

```
data_obj <- get_XCMSnExp_object_example(
  indices = 1:3,
  should_group_peaks = TRUE)
p <- lcmsPlot(data_obj, sample_id_column = "sample_name") +
  lp_peak_density(
    features = rbind(c(mzmin = 334.9, mzmax = 335.1,
                       rtmin = 2700,  rtmax = 2900)),
    bw = 30,
    min_fraction = 0.5)
p
```

---

lp_purity_distribution

Plot the distribution of precursor ion purity scores per sample

---

Description

lp_purity_distribution() generates a violin (or box/jitter) plot of inPurity scores grouped by sample. An optional horizontal dashed line marks a purity acceptance threshold. Requires the data object to be a purityA result from the msPurity package.

Usage

```
lp_purity_distribution(sample_ids = NULL, threshold = NULL, type = "violin")
```

Arguments

sample_ids	A character vector of sample IDs to include. NULL uses all samples.
threshold	A numeric value in [0, 1] drawn as a horizontal reference line. NULL suppresses the line.
type	A character value; one of "violin", "boxplot", or "jitter". Defaults to "violin".

Value

A layer function for use with the + operator.

Examples

```
mzml_dir <- file.path(tempdir(), "purity-example")
utils::unzip(
    system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
    exdir = mzml_dir)

ms2_files <- file.path(
    mzml_dir, "mzml",
    c("l-proline-MS2.mzML", "l-kynurenine-MS2.mzML"))

pa <- msPurity::purityA(ms2_files)

## Compare the spread of purity scores between samples.
p <- lcmsPlot(pa) +
    lp_purity_distribution(threshold = 0.7, type = "boxplot")
p
```

---

lp_purity_overlay *Overlay precursor ion purity scores on a chromatogram*

Description

lp_purity_overlay() adds a geom_point layer to the chromatogram panel where each triangle marks the retention time of an MS/MS fragmentation event, coloured by the interpolated precursor ion purity (inPurity). Requires the data object to be a purityA result from the msPurity package and lp_chromatogram() to be called first.

Usage

```
lp_purity_overlay(sample_ids = NULL, threshold = NULL, point_size = 2)
```

Arguments

sample_ids	A character vector of sample IDs to include. NULL uses all samples.
threshold	A numeric value in [0, 1] drawn as a reference on the colour scale midpoint. NULL defaults to 0.5.
point_size	A numeric value controlling the point size.

Value

A layer function for use with the + operator.

Examples

```
## lcmsPlot ships two DDA files carrying real precursor isolation metadata.
mzml_dir <- file.path(tempdir(), "purity-example")
utils::unzip(
    system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
    exdir = mzml_dir)

ms2_files <- file.path(
    mzml_dir, "mzml",
    c("l-proline-MS2.mzML", "l-kynurenine-MS2.mzML"))

pa <- msPurity::purityA(ms2_files)

## The overlay annotates an existing chromatogram, so the retention time
## window has to span the fragmentation events to show anything.
features <- data.frame(
    sample_id = c("l-proline-MS2", "l-kynurenine-MS2"),
    mz = c(116.0703793, 209.091824),
    rt = c(425.74, 365.72))

p <- lcmsPlot(pa) +
    lp_chromatogram(features = features, ppm = 10, rt_tol = 60) +
    lp_purity_overlay(threshold = 0.7)
p
```

---

lp_purity_timeline *Plot precursor ion purity scores as a timeline*

Description

`lp_purity_timeline()` generates a scatter plot of `inPurity` (y-axis) versus retention time (x-axis), one point per MS/MS acquisition event, coloured by sample. An optional horizontal dashed line marks a purity threshold. Requires the data object to be a `purityA` result.

Usage

```
lp_purity_timeline(sample_ids = NULL, threshold = NULL)
```

Arguments

`sample_ids`	A `character` vector of sample IDs to include. `NULL` uses all samples.
`threshold`	A `numeric` value in `[0, 1]` drawn as a horizontal reference line. `NULL` suppresses the line.

Value

A layer function for use with the + operator.

Examples

```
mzml_dir <- file.path(tempdir(), "purity-example")
utils::unzip(
    system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
    exdir = mzml_dir)

ms2_files <- file.path(
    mzml_dir, "mzml",
    c("l-proline-MS2.mzML", "l-kynurenine-MS2.mzML"))

pa <- msPurity::purityA(ms2_files)

## One point per MS/MS event, coloured by sample.
p <- lcmsPlot(pa) +
    lp_purity_timeline(threshold = 0.7)
p
```

---

lp_rt_diff_plot *Generate the retention time difference plot between raw and adjusted datasets*

Description

The `lp_rt_diff_plot` function generates the data necessary to plot the difference between the raw and retention time adjusted datasets. Only applicable to `XCMSnExp` and `MsExperiment` objects.

Usage

```
lp_rt_diff_plot()
```

Value

This function returns another function that takes an `lcmsPlot` object and produces a modified version containing the generated retention time differences, between raw and adjusted, in its `data` slot. It is designed to be used with the + operator, which serves as a layering mechanism. Each use of + incrementally enriches the `lcmsPlot` object by adding new data or visual components.

Examples

```
data_obj <- get_XCMSnExp_object_example(
  indices = 1:3,
  should_group_peaks = TRUE)
p <- lcmsPlot(data_obj, sample_id_column = "sample_name") +
  lp_rt_diff_plot()
p
```

`lp_rt_line` *Define a vertical line on a retention time value*

Description

Define a vertical line on a retention time value

Usage

```
lp_rt_line(intercept, line_type = "dashed", color = "black")
```

Arguments

`intercept`	A `numeric` value indicating the retention time axis (x-axis) intercept.
`line_type`	A `character` value indicating the line type. One of `"solid"`, `"dashed"`, `"dotted"`, `"dotdash"`, `"longdash"`, `"twodash"`.
`color`	A `character` value indicating the line color.

Value

A function that takes an `lcmsPlot` object and returns a modified version with the specified RT line options stored in `options$rt_lines`. It is intended for use with the + operator, which incrementally layers new data or visual components onto the `lcmsPlot` object.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:4]

## Create chromatogram plots faceted by sample
p <- lcmsPlot(raw_files) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
  lp_facets(facets = 'sample_id', ncol = 4)
p

## Add a vertical retention time reference line
p <- p + lp_rt_line(intercept = 2800, line_type = 'solid', color = 'red')
p
```

lp_spectra *Define the spectra to plot*

Description

The lp_spectra function enables the generation of spectra, which are graphical representations of ions detected at each mass-to-charge ratio (m/z) with their corresponding absolute or relative intensities.

Usage

```
lp_spectra(
  sample_ids = NULL,
  mode = "closest_apex",
  ms_level = 1,
  rt = NULL,
  scan_index = NULL,
  interval = 3,
  spectral_match_db = NULL,
  match_target_index = NULL,
  peak_label_size = 3,
  intensity_breaks_by = 20,
  auto_facet = TRUE
)
```

Arguments

sample_ids	A character vector specifying the sample IDs to include in the plot. If NULL, the function uses the sample IDs specified in the lcmsPlot object or the lp_chromatogram function.
mode	The method to choose the scan from which to extract the spectra. One of: closest, the closest scan to the specified RT - rt parameter); closest_apex, the closest scan to a detected peak; across_peak, selects scans across a detected peak at a certain interval specified in the interval parameter. mode is not applicable to standalone spectra.
ms_level	The MS level to consider for the scan.
rt	When mode = "closest", the RT to consider.
scan_index	The exact scan index to consider for extracting a spectrum. scan_index and mode are mutually exclusive.
interval	When mode = "across_peak." The RT interval to consider.
spectral_match_db	The database containing reference spectra used for matching and comparison with the input spectra.
match_target_index	The index, ranked by descending match score, identifying which reference spectrum to display in the mirror plot.
peak_label_size	A numeric value controlling the font size of m/z labels annotated on spectral peaks. Defaults to 3.

intensity_breaks_by
: A numeric value specifying the step size (in percent) between y-axis intensity breaks. Defaults to 20.

auto_facet
: A logical value. When TRUE (default), a facet is automatically added to separate spectra from different samples or scans. Set to FALSE to suppress automatic faceting.

Value

This function returns another function that takes an lcmsPlot object and produces a modified version containing the generated spectra in its data slot. It is designed to be used with the + operator, which serves as a layering mechanism. Each use of + incrementally enriches the lcmsPlot object by adding new data or visual components.

Spectra associated with chromatograms

A spectrum is obtained from a scan at a specific retention time (RT). Therefore, when plotting a chromatogram together with its associated spectra, it is common to mark the RT with a vertical line on the chromatogram to indicate where the spectra were acquired. See the example below on how to generate these types of spectra.

Standalone spectra

Standalone spectra can also be generated, provided no chromatograms are present (i.e., lp_chromatogram has not been used).

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1]

p <- lcmsPlot(raw_files) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
  lp_spectra(mode = "closest", rt = 2785)
p
```

---

lp_total_ion_current *Define the total ion current (TIC)*

Description

The lp_total_ion_current generates summary data for the total ion current (TIC) of the selected samples. It works both with XCMS objects (XCMSnExp, MsExperiment) and with raw files passed directly to lcmsPlot() (a character vector of .mzML, .mzXML, .CDF, or .raw paths).

Usage

```
lp_total_ion_current(sample_ids = NULL, type = "boxplot")
```

Arguments

sample_ids	A character vector specifying the sample IDs to include in the plot. If NULL, the function uses the sample IDs specified in the lcmsPlot object.
type	A character value indicating the type of plot; one of "boxplot", "violin", "jitter".

Value

This function returns another function that takes an lcmsPlot object and produces a modified version containing the generated total ion current (TIC) in its data slot. It is designed to be used with the + operator, which serves as a layering mechanism. Each use of + incrementally enriches the lcmsPlot object by adding new data or visual components.

Examples

```
data_obj <- get_XCMSnExp_object_example()

p <- lcmsPlot(data_obj, sample_id_column = "sample_name") +
  lp_total_ion_current(type = "violin") +
  lp_arrange(group_by = "sample_id")
p
```

---

merge_by_index *Merge two data frames using a row-index match*

Description

Merge two data frames using a row-index match

Usage

```
merge_by_index(a, b, index_col)
```

Arguments

a	The left data frame.
b	The right data frame whose row order defines the index.
index_col	The name of the column in a that contains row indices referring to b.

Value

A tibble resulting from a left join of a and the indexed b.

`MsDialPeaksSource` *Create an MS-DIAL data source from peak lists*

Description

This function reads MS-DIAL peak list files and sample metadata and converts them into a format compatible with xcms-style peak tables.

Usage

```
MsDialPeaksSource(peaks_paths, sample_paths, metadata_path = NULL)
```

Arguments

`peaks_paths`	A `character` vector of paths to MS-DIAL peak list files (CSV or TSV). The ordering should match the one in the metadata (i.e., the samples).
`sample_paths`	A `character` vector of paths to raw sample files (e.g., `mzML`).
`metadata_path`	A `character` value (optional) indicating the path to a metadata file (CSV or TSV). If `NULL`, a metadata `data.frame` is created from `sample_paths`.

Value

An object of class `ExternalDataSource`.

Examples

```
## Create temporary example files
tmp_dir <- tempdir()

## Fake raw sample paths
sample_paths <- file.path(
  tmp_dir,
  c("sample1.mzML", "sample2.mzML")
)

## Create minimal MS-DIAL peak list files (one per sample)
peak_list_paths <- file.path(
  tmp_dir,
  c("sample1_peaks.csv", "sample2_peaks.csv")
)

msdial_peaks_1 <- tibble::tibble(
  "Precursor m/z" = c(100.1, 200.2),
  "RT (min)" = c(5.0, 10.0),
  "Area" = c(10000, 20000),
  "Height" = c(500, 800),
  "RT left(min)" = c(4.8, 9.8),
  "RT right (min)" = c(5.2, 10.2)
)

msdial_peaks_2 <- tibble::tibble(
  "Precursor m/z" = c(150.3, 250.4),
  "RT (min)" = c(6.0, 12.0),
```

```
  "Area" = c(15000, 25000),
  "Height" = c(600, 900),
  "RT left(min)" = c(5.8, 11.8),
  "RT right (min)" = c(6.2, 12.2)
)

utils::write.csv(msdial_peaks_1, peak_list_paths[1], row.names = FALSE)
utils::write.csv(msdial_peaks_2, peak_list_paths[2], row.names = FALSE)

## Create the data source
ds <- MsDialPeaksSource(
  peaks_paths = peak_list_paths,
  sample_paths = sample_paths
)
```

MsRawReader-class *Virtual base class for raw MS file readers*

Description

MsRawReader is an abstract S4 class that defines the interface for reading raw mass spectrometry data files. Concrete subclasses implement the interface for specific backends (e.g., mzR, rawrr).

---

ms_chromatogram *Extract an extracted ion chromatogram (XIC) using the native backend*

Description

ms_chromatogram() uses the backend's own XIC extraction mechanism rather than reconstructing chromatograms scan-by-scan from ms_peaks(). Currently only implemented for RawrrReader (ThermoFisher .raw files), which delegates to rawrr::readChromatogram().

Usage

```
ms_chromatogram(reader, mz, ppm, rt_range)

## S4 method for signature 'MzrReader'
ms_chromatogram(reader, mz, ppm, rt_range)

## S4 method for signature 'RawrrReader'
ms_chromatogram(reader, mz, ppm, rt_range)

## S4 method for signature 'XcmsRawReader'
ms_chromatogram(reader, mz, ppm, rt_range)
```

Arguments

reader	An instance of MsRawReader.
mz	A numeric scalar — the target m/z.
ppm	A numeric scalar — the mass tolerance in ppm.
rt_range	A length-2 numeric vector — the RT window in seconds.

Value

A `list` with two `tibbles`: `chromatograms` (columns `rt` and `intensity`) and `mass_traces` (empty for backends that do not expose per-scan m/z data via this interface).

---

ms_close *Close a raw MS file reader connection*

Description

Close a raw MS file reader connection

Usage

```
ms_close(reader)

## S4 method for signature 'MzrReader'
ms_close(reader)

## S4 method for signature 'RawrrReader'
ms_close(reader)

## S4 method for signature 'XcmsRawReader'
ms_close(reader)
```

Arguments

`reader`	An instance of `MsRawReader`.

Value

`invisible(NULL)`

---

ms_header *Get scan header information from a raw MS file reader*

Description

Get scan header information from a raw MS file reader

Usage

```
ms_header(reader)

## S4 method for signature 'MzrReader'
ms_header(reader)

## S4 method for signature 'RawrrReader'
ms_header(reader)

## S4 method for signature 'XcmsRawReader'
ms_header(reader)
```

Arguments

reader	An instance of MsRawReader.

Value

A tibble with at least the columns seqNum, retentionTime, msLevel, basePeakIntensity, totIonCurrent, peaksCount, and meanIntensity.

meanIntensity is the per-scan average intensity backing aggregation_fun = "mean". Each backend defines it in the way that is faithful to its own data model: XcmsRawReader takes the profile-matrix column mean when a profile matrix is available, matching xcms::plotChrom(base = FALSE) exactly, while the file-based backends average over the measured peaks of each scan. Backends that cannot supply it return NA_real_.

---

ms_peaks *Get peak matrices for selected scans from a raw MS file reader*

Description

Get peak matrices for selected scans from a raw MS file reader

Usage

```
ms_peaks(reader, scans)

## S4 method for signature 'MzrReader'
ms_peaks(reader, scans)

## S4 method for signature 'RawrrReader'
ms_peaks(reader, scans)

## S4 method for signature 'XcmsRawReader'
ms_peaks(reader, scans)
```

Arguments

reader	An instance of MsRawReader.
scans	An integer vector of scan sequence numbers to retrieve.

Value

A list of two-column matrices with columns mz and intensity, one element per scan (always a list, even for a single scan).

`MZmineFeatureListsSource` *Create an MZmine data source from feature lists*

Description

This function reads MZmine feature list files (supports version 2 and above) and sample metadata and converts them into a format compatible with xcms-style peak tables.

Usage

```
MZmineFeatureListsSource(
  feature_lists_paths,
  sample_paths,
  metadata_path = NULL
)
```

Arguments

`feature_lists_paths`	A `character` vector of paths to MZmine feature list files (CSV or TSV).
`sample_paths`	A `character` vector of paths to raw sample files (e.g., `mzML`).
`metadata_path`	A `character` value (optional) indicating the path to a metadata file (CSV or TSV). If `NULL`, a metadata `data.frame` is created from `sample_paths`.

Value

An object of class `ExternalDataSource`.

Examples

```
## Create temporary example files
tmp_dir <- tempdir()

## Fake sample paths
sample_paths <- file.path(
  tmp_dir,
  c("sample1.mzML", "sample2.mzML")
)

## Create a minimal MZmine 2 feature list CSV
feature_list_path <- file.path(tmp_dir, "mzmine_features.csv")

mzmine_features <- tibble::tibble(
  "row m/z" = c(100.1, 200.2),
  "sample1.mzML Feature status" = c("DETECTED", "DETECTED"),
  "sample1.mzML Feature m/z" = c(100.1, 200.2),
  "sample1.mzML Feature RT" = c(300, 600),
  "sample1.mzML Peak area" = c(10000, 20000),
  "sample1.mzML Peak height" = c(500, 800),
  "sample1.mzML Feature m/z min" = c(99.9, 199.9),
  "sample1.mzML Feature m/z max" = c(100.3, 200.4),
  "sample1.mzML Feature RT start" = c(290, 590),
```

```
  "sample1.mzML Feature RT end" = c(310, 610),
  check.names = FALSE
)

utils::write.csv(mzmine_features, feature_list_path, row.names = FALSE)

## Create the data source
ds <- MZmineFeatureListsSource(
  feature_lists_paths = feature_list_path,
  sample_paths = sample_paths
)
```

MzrReader-class *An* mzR*-backed raw MS file reader*

Description

Wraps an open mzR connection (as returned by mzR::openMSfile()).

Slots

connection An mzR connection object.

---

next_plot *Move to the next plot object (batch-mode)*

Description

next_plot progresses an lcmsPlotClass object to the next plot in a batch-processing sequence. This is typically used when multiple plots are generated and inspected iteratively, such as when navigating large LC–MS datasets in a batched workflow. The batch size is defined in the lcmsPlot function's argument batch_size.

Usage

```
next_plot(object)

## S4 method for signature 'lcmsPlotClass'
next_plot(object)
```

Arguments

object	An instance of class lcmsPlotClass.

Value

An instance of class lcmsPlotClass.

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

p <- lcmsPlot(raw_files, batch_size = 2) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
  lp_arrange(group_by = "sample_id") +
  lp_legend(position = "bottom") +
  lp_labels(legend = "Sample")

p <- next_plot(p)
p
```

---

open_cd_result_connection

Open a connection to a Compound Discoverer results database

Description

Establishes a read-only SQLite database connection to a Compound Discoverer results directory.

Usage

```
open_cd_result_connection(cd_result_path)
```

Arguments

`cd_result_path` A `character` value giving the path to a Compound Discoverer results file.

Value

A `DBIConnection` object connected to the Compound Discoverer SQLite database.

---

open_raw_reader

Open a raw MS file as an `MsRawReader`

Description

Dispatches to `MzrReader` for standard open formats (`.mzML`, `.mzXML`, `.CDF`) or `RawrrReader` for ThermoFisher `.raw` files.

Usage

```
open_raw_reader(path)
```

Arguments

path	A character value giving the file path.

Value

An instance of MsRawReader (either MzrReader or RawrrReader).

---

parse_trace *Parse a binary XIC trace from Compound Discoverer*

Description

Decodes a binary XIC trace blob stored in a Compound Discoverer results database and converts it into a retention time-intensity data frame.

Usage

```
parse_trace(data)
```

Arguments

data	A raw vector or binary object containing the encoded XIC trace.

Value

A tibble with columns:

rt Retention time in seconds.

intensity Signal intensity.

If the trace cannot be parsed, NULL is returned.

---

plot_chromatogram *Plot chromatograms from one or more datasets*

Description

plot_chromatogram() generates a ggplot2 chromatogram plot from processed datasets. It can handle multiple datasets, optionally highlight detected peaks or apices, and apply faceting or grid layouts based on plot options.

Usage

```
plot_chromatogram(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

datasets	A named list of data frames containing the primary datasets to plot. Typically includes chromatograms and optionally spectra.
supporting_datasets	A list of supporting data frames, such as detected peaks, used for highlighting features.
options	A list of plot options, controlling units, faceting, highlighting, and other visual parameters.
single	A logical value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is FALSE.

Value

A ggplot object representing the chromatogram plot.

---

plot_chrom_peak_rects *Plot chromatographic peak boundaries on their own panel*

Description

plot_chrom_peak_rects() draws the peak rectangles on an otherwise empty retention time / m/z frame, which is what xcms::plotChromPeaks() produces. It is used when lp_chrom_peak_rects() is called without a host layer; with lp_mass_trace() or lp_intensity_map() present the rectangles are drawn as an overlay by those renderers instead.

Usage

```
plot_chrom_peak_rects(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

datasets	A named list of data frames containing the primary datasets to plot. The used key is chrom_peak_rects.
supporting_datasets	A list of supporting data frames. Unused; present for API consistency.
options	A list of plot options, controlling units, faceting, highlighting, and other visual parameters.
single	A logical value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is FALSE.

Value

A ggplot object representing the peak boundary panel.

plot_data *Plot the specified LC-MS data*

Description

Plot the specified LC-MS data

Usage

```
plot_data(datasets, obj)
```

Arguments

datasets	A list of data frames, each representing a dataset to be visualised.
obj	The lcmsPlot object.

Value

A patchwork plot object representing the final plot.

plot_intensity_map *Plot an LC-MS intensity map*

Description

plot_intensity_map() generates a ggplot2 from an intensity map which is a point-cloud of m/z and RT points.

Usage

```
plot_intensity_map(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

datasets	A named list of data frames containing the primary datasets to plot. For a 2D intensity map the used key is intensity_maps.
supporting_datasets	A list of supporting data frames, such as detected peaks, used for highlighting features.
options	A list of plot options, controlling units, faceting, highlighting, and other visual parameters.
single	A logical value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is FALSE.

Value

A ggplot object representing the 2D intensity map plot.

plot_isolation_window *Plot an MS1 spectrum with isolation window annotation*

Description

Extends the standard spectrum plot by adding a semi-transparent rectangle spanning the isolation window and a dashed vertical line at the precursor m/z. Requires `in_purity`, `precursor_mz`, and `isolation_window_half_width` columns in the dataset (merged from `feature_metadata` by the render pipeline).

Usage

```
plot_isolation_window(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

`datasets`	A named `list` containing a `spectra` element.
`supporting_datasets`	Unused; present for API consistency.
`options`	A `list` of plot options.
`single`	A `logical` value passed through to the base spectrum plot.

Value

A `ggplot` object.

plot_mass_trace *Plot a mass trace*

Description

`plot_mass_trace()` generates a ggplot2 from a mass trace

Usage

```
plot_mass_trace(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

`datasets`	A named `list` of data frames containing the primary datasets to plot. For a mass trace plot the used key is `mass_traces`.
`supporting_datasets`	A `list` of supporting data frames, such as detected peaks, used for highlighting features.
`options`	A list of plot options, controlling units, faceting, highlighting, and other visual parameters.
`single`	A `logical` value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is `FALSE`.

Value

A ggplot object representing the mass trace plot.

`plot_multiple_datasets` *Plot multiple dataset types*

Description

`plot_multiple_datasets()` iterates over the provided datasets, dispatches each one to its corresponding plotting function as defined in `plot_config`, and combines the resulting plots into a single patchwork object.

Usage

```
plot_multiple_datasets(datasets, obj, plot_config)
```

Arguments

`datasets`	A named `list` of data frames, where each element represents a dataset to be plotted. The names must match the supported dataset types defined in `plot_config`.
`obj`	An instance of class `lcmsPlotClass`.
`plot_config`	A named `list` defining the plotting functions to use for each dataset type. List names correspond to dataset types, and values are functions that return ggplot objects.

Value

A patchwork object combining all generated plots.

`plot_multiple_faceted_datasets` *Plot multiple faceted dataset types*

Description

`plot_multiple_faceted_datasets()` generates plots for multiple dataset types and arranges them into a faceted grid. Datasets are split according to the faceting variables defined in `obj@options$facets`, with each facet panel containing a vertically stacked set of dataset-specific plots.

Usage

```
plot_multiple_faceted_datasets(datasets, obj, plot_config)
```

Arguments

datasets	A named list of data frames, where each element represents a dataset to be plotted. The names must match the supported dataset types defined in plot_config.
obj	An instance of class lcmsPlotClass.
plot_config	A named list defining the plotting functions to use for each dataset type. List names correspond to dataset types, and values are functions that return ggplot objects.

Value

A patchwork object combining all generated plots.

---

plot_peak_count_image *Plot chromatographic peak counts per retention-time bin*

Description

plot_peak_count_image() draws retention-time bins on x and samples on y, filled by how many chromatographic peaks each sample yielded in each bin. It is the ggplot2 counterpart of xcms::plotChromPeakImage(), and is read as a quality-control view: an empty stretch marks a sample that stopped producing peaks, a pale column a retention-time region where detection collapsed.

Usage

```
plot_peak_count_image(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

datasets	A named list of data frames containing the primary datasets to plot. The used key is peak_count_image.
supporting_datasets	A list of supporting data frames. Unused; present for API consistency.
options	A list of plot options, controlling units, faceting, highlighting, and other visual parameters.
single	A logical value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is FALSE.

Value

A ggplot object representing the peak count image.

plot_peak_density *Plot a peak density plot*

Description

`plot_peak_density()` generates a ggplot2 peak density plot that mirrors `xcms::plotChromPeakDensity()`. For each m/z feature:

- The x axis shows retention time.
- The y axis shows sample indices (1 to n), positioned within the density range so that sample rows are evenly spaced from 0 to `max(density)`.
- Detected peaks are plotted as coloured points at each sample's y position.
- The kernel density estimate of peak apex RTs is drawn as a line.
- When feature group simulation is enabled, semi-transparent rectangles mark feature groups that pass the `min_fraction` / `min_samples` threshold.

Usage

```
plot_peak_density(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

`datasets`	A named `list` of data frames. The used key is `peak_density`.
`supporting_datasets`	A `list` of supporting data frames. The used key is `detected_peaks`, which provides per-sample peak positions.
`options`	A list of plot options.
`single`	A `logical` value indicating whether to treat the plot as a single dataset variant. Default is `FALSE`.

Value

A `ggplot` object representing the peak density plot.

plot_purity_distribution *Plot the distribution of precursor ion purity scores per sample*

Description

Generates a violin or boxplot of `inPurity` scores grouped by sample, with an optional threshold line.

Usage

```
plot_purity_distribution(
  datasets,
  supporting_datasets,
  options,
  single = FALSE
)
```

Arguments

datasets	A named list containing a purity_distribution element.
supporting_datasets	Unused; present for API consistency.
options	A list of plot options.
single	Unused; present for API consistency.

Value

A ggplot object.

---

plot_purity_timeline *Plot precursor ion purity scores over retention time*

Description

Generates a scatter plot of inPurity (y-axis) versus retention time (x-axis) for each sample, with an optional horizontal threshold line.

Usage

```
plot_purity_timeline(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

datasets	A named list containing a purity_timeline element.
supporting_datasets	Unused; present for API consistency.
options	A list of plot options.
single	Unused; present for API consistency.

Value

A ggplot object.

plot_rt_diff *Plot an RT alignment difference plot*

Description

plot_rt_diff() generates a ggplot2 from a dataset containing the differences between raw and adjusted retention times.

Usage

```
plot_rt_diff(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

datasets	A named list of data frames containing the primary datasets to plot. For an RT difference plot the used key is rt_diff.
supporting_datasets	A list of supporting data frames, such as detected peaks, used for highlighting features.
options	A list of plot options, controlling units, faceting, highlighting, and other visual parameters.
single	A logical value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is FALSE.

Value

A ggplot object representing the RT difference plot.

plot_single_dataset *Plot a single dataset type*

Description

plot_single_dataset() allows the plotting of a single dataset type (e.g., only chromatograms).

Usage

```
plot_single_dataset(datasets, obj, plot_config)
```

Arguments

datasets	A named list of data frames, where each element represents a dataset to be plotted. The names must match the supported dataset types defined in plot_config.
obj	An instance of class lcmsPlotClass.
plot_config	A list that defines the plot configuration. This list should use the supported dataset types as keys (e.g., chromatograms), with each value providing the corresponding function used to plot that dataset type.

Value

A patchwork object combining all generated plots. In this case the patchwork object contains a single ggplot2 object.

---

plot_spectrum *Plot spectra from one or more datasets*

Description

plot_spectrum() generates a ggplot2 MS spectra plot.

Usage

```
plot_spectrum(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

datasets	A named list of data frames containing the primary datasets to plot. For a spectra plot the used key is spectra.
supporting_datasets	A list of supporting data frames, such as detected peaks, used for highlighting features.
options	A list of plot options, controlling units, faceting, highlighting, and other visual parameters.
single	A logical value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is FALSE.

Value

A ggplot object representing the RT difference plot.

---

plot_total_ion_current *Plot total ion current for one or more samples*

Description

plot_total_ion_current() generates a ggplot2 of the total ion current of the samples within the dataset.

Usage

```
plot_total_ion_current(datasets, supporting_datasets, options, single = FALSE)
```

Arguments

datasets	A named list of data frames containing the primary datasets to plot. For a TIC plot the used key is total_ion_current.
supporting_datasets	A list of supporting data frames, such as detected peaks, used for highlighting features.
options	A list of plot options, controlling units, faceting, highlighting, and other visual parameters.
single	A logical value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is FALSE.

Value

A ggplot object representing the RT difference plot.

---

process_metadata *Process sample metadata*

Description

Construct a metadata data frame aligned with a set of sample paths. If a metadata file is provided, it is read from disk and combined with the sample paths. Otherwise, a minimal metadata table is created.

Usage

```
process_metadata(sample_paths, metadata_path = NULL)
```

Arguments

sample_paths	A character vector of sample file paths.
metadata_path	A character value indicating the path to a delimited metadata file with a header.

Value

A tibble containing a sample_path column and any additional metadata.

purity_overlay_layer *Add purity score overlay to a chromatogram plot*

Description

Adds a `geom_point` layer to a chromatogram ggplot where each point marks the retention time of an MS/MS acquisition event, coloured by interpolated precursor ion purity (`inPurity`).

Usage

```
purity_overlay_layer(p, purity_scores, options)
```

Arguments

`p`	A `ggplot` object (a chromatogram).
`purity_scores`	A `data.frame` with columns `rt`, `in_purity`, `metadata_index`, and any additional sample columns merged in by the render pipeline.
`options`	A `list` of plot options (reads `options$purity_overlay`).

Value

The modified `ggplot` object.

RawrrReader-class *A* `rawrr`*-backed raw MS file reader*

Description

Stores the path to a ThermoFisher `.raw` file. `rawrr` does not maintain persistent connections, so the path is re-used for each read operation.

Slots

`path` A `character` value giving the path to the `.raw` file.

remove_null_elements *Remove* NULL *elements from a list*

Description

Remove NULL elements from a list

Usage

```
remove_null_elements(lst)
```

Arguments

lst	A list from which NULL entries should be removed.

Value

A list containing only the non-NULL elements of lst.

run_matching_plot_variant *Selects and executes the appropriate plot variant*

Description

Determines which plotting variant is applicable for the given datasets and plotting options, then executes the first matching variant. Plot variants are evaluated in order, and the first whose condition evaluates to TRUE is used to generate the plot.

Usage

```
run_matching_plot_variant(datasets, obj)
```

Arguments

datasets	A list of data frames, each representing a dataset to be visualised.
obj	An instance of class lcmsPlotClass.

Value

A patchwork plot object generated by the selected plot variant.

series_offset *Look up the stacking offset for every row of a data frame*

Description

Look up the stacking offset for every row of a data frame

Usage

```
series_offset(df, offsets, stack_col)
```

Arguments

df	A data.frame carrying stack_col.
offsets	A named numeric vector as returned by stacked_offsets(), or NULL when the plot is not stacked.
stack_col	A character naming the series column, or NULL.

Value

A numeric vector of offsets, zero where no offset applies.

show,CompoundDiscovererNodeSource-method *Show a summary of a* CompoundDiscovererNodeSource *object*

Description

Show a summary of a CompoundDiscovererNodeSource object

Usage

```
## S4 method for signature 'CompoundDiscovererNodeSource'
show(object)
```

Arguments

object	An instance of class CompoundDiscovererNodeSource.

Value

Invisible NULL

show,ExternalDataSource-method
Show a summary of an instance of class ExternalDataSource

Description

Show a summary of an instance of class ExternalDataSource

Usage

```
## S4 method for signature 'ExternalDataSource'
show(object)
```

Arguments

object	An instance of class ExternalDataSource.

Value

Invisible NULL

Examples

```
## Create dummy metadata
metadata <- data.frame(
  sample_id = c("S1", "S2"),
  sample_path = c("sample1.mzML", "sample2.mzML")
)

## Create dummy peaks
peaks <- data.frame(
  mz = c(100.1, 150.2),
  rt = c(300, 450),
  rtmin = c(290, 440),
  rtmax = c(310, 460),
  into = c(10000, 15000),
  maxo = c(2000, 2500),
  sample_index = c(1, 2)
)

## Create ExternalDataSource object
eds <- new(
  "ExternalDataSource",
  name = "Example data source",
  metadata = metadata,
  peaks = peaks
)

## Show summary
eds
```

show,lcmsPlotClass-method

Plot the lcmsPlotClass *object*

Description

Display an instance of lcmsPlotClass class to the selected device.

Usage

```
## S4 method for signature 'lcmsPlotClass'
show(object)
```

Arguments

object	An instance of class lcmsPlotClass.

Value

Invisible NULL

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

## Shows summary information as the plot has not been built yet
p <- lcmsPlot(raw_files)
p

## Shows the actual plot
p <- lcmsPlot(raw_files) +
  lp_chromatogram(aggregation_fun = "max") +
  lp_arrange(group_by = "sample_id") +
  lp_legend(position = "bottom") +
  lp_labels(legend = "Sample")

p
```

show,lcmsPlotDataContainer-method

Show a summary of an instance of class lcmsPlotDataContainer

Description

Show a summary of an instance of class lcmsPlotDataContainer

Usage

```
## S4 method for signature 'lcmsPlotDataContainer'
show(object)
```

Arguments

object	An instance of class lcmsPlotDataContainer.

Value

Invisible NULL

Examples

```
raw_files <- dir(
   system.file("cdf", package = "faahKO"),
   full.names = TRUE,
   recursive = TRUE)[1:5]

data_obj <- new("lcmsPlotDataContainer",
    data_obj = raw_files,
    metadata = tibble::tibble(),
    chromatograms = tibble::tibble(),
    mass_traces = tibble::tibble(),
    spectra = tibble::tibble(),
    peak_density = tibble::tibble(),
    peak_count_image = tibble::tibble(),
    total_ion_current = tibble::tibble(),
    intensity_maps = tibble::tibble(),
    rt_diff = tibble::tibble(),
    feature_metadata = tibble::tibble(),
    detected_peaks = tibble::tibble(),
    purity_scores = tibble::tibble())
data_obj
```

---

show,LipidSearchSource-method

Show a summary of a LipidSearchSource *object*

Description

Show a summary of a LipidSearchSource object

Usage

```
## S4 method for signature 'LipidSearchSource'
show(object)
```

Arguments

object	An instance of class LipidSearchSource.

Value

Invisible NULL

---

show,XcmsRawList-method

Show a summary of an XcmsRawList *object*

Description

Show a summary of an XcmsRawList object

Usage

```
## S4 method for signature 'XcmsRawList'
show(object)
```

Arguments

object	An instance of XcmsRawList.

Value

Invisible NULL.

---

stacked_offsets

Offsets that stack the series of a chromatogram panel up the y axis

---

Description

Reproduces the offsets computed by xcms::plotChromatogramsOverlay(): the series are ordered by m/z and mapped linearly onto stacked * y_limits, so the largest m/z sits at the top of the band.

Usage

```
stacked_offsets(dataset, stack_col, stacked, y_limits)
```

Arguments

dataset	A data.frame of chromatogram rows carrying stack_col, and feature_mz when the input provides per-series m/z values.
stack_col	A character naming the column that identifies a series.
stacked	A numeric fraction of the intensity range to spread the series over.
y_limits	A length-2 numeric giving the transformed intensity range.

Value

A named numeric vector of offsets, one per series key.

validate_data_frame *Validate a data frame against a list of field specifications*

Description

Returns TRUE on success or a character string describing the first failure. NULL and empty data frames always pass.

Usage

```
validate_data_frame(df, fields, exact = FALSE)
```

Arguments

df	A data.frame or tibble to validate.
fields	A list of field specifications created with field().
exact	A logical value. If TRUE, the column names of df must be identical to the names in fields (same set, same order). Defaults to FALSE.

Value

TRUE if validation passes, or a character string describing the first failure.

validate_object *Validate the data frame slots of an S4 object*

Description

Iterates over a named list of validator functions, applying each to the corresponding slot of object.

Usage

```
validate_object(object, validators)
```

Arguments

object	An S4 object whose slots are to be validated.
validators	A named list of functions, where each name matches a slot of object and each function accepts a data.frame/tibble and returns TRUE or a character error string (as produced by validate_data_frame()).

Value

TRUE if all validators pass, or a character string describing the first failure in the form "@slot_name: <message>".

XcmsRawList *Create an* XcmsRawList *object*

Description

Wraps one or more xcmsRaw objects into an XcmsRawList container for use as the dataset argument of lcmsPlot.

Usage

```
XcmsRawList(...)
```

Arguments

...	One or more xcmsRaw objects, or a single list of xcmsRaw objects.

Value

An instance of XcmsRawList.

Examples

```
paths <- dir(
    system.file("cdf", package = "faahKO"),
    full.names = TRUE,
    recursive = TRUE
)[c(1, 2)]
raw1 <- xcms::xcmsRaw(paths[1])
raw2 <- xcms::xcmsRaw(paths[2])
xl <- XcmsRawList(raw1, raw2)
```

XcmsRawList-class *A list of* xcmsRaw *objects representing multiple samples*

Description

XcmsRawList is a container for one or more xcmsRaw objects, where each element corresponds to a single sample. It is the recommended way to pass in-memory raw LC-MS data to lcmsPlot.

Slots

data A list of xcmsRaw objects, one per sample.

XcmsRawReader-class *An* xcmsRaw*-backed raw MS file reader*

Description

Wraps an in-memory xcmsRaw object. No file connection is opened or closed.

Slots

obj An xcmsRaw object.

xcmsraw_to_readers *Convert an* XcmsRawList *to a named list of* XcmsRawReader*s*

Description

Each element is keyed by the file path stored in @filepath of the corresponding xcmsRaw object.

Usage

```
xcmsraw_to_readers(xcmsraw_list)
```

Arguments

xcmsraw_list An XcmsRawList object.

Value

A named list of XcmsRawReader objects.

Index