Package: lcmsPlot (via r-universe)

June 4, 2026

Type Package

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

Version 1.1.3

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

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' 'creators-purity.R' 'constructors-chromatograms.R' 'constructors-spectra.R' 'data-validation.R' 'data-adapters.R' 'plot-units.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-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-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

Config/pak/sysreqs
cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev libnetcdf-dev libssl-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-06-03 08:47:30 UTC

RemoteUrl https://github.com/bioc/lcmsPlot

RemoteRef HEAD

RemoteSha 1a326fb0e7f6da05b1aad2855d9c6d7c7a35dab8

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")
```

`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)
```

`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.

`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)
```

`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().

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.

---

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)
)
```

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 or sum. 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.

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 either summed to produce the total ion current (TIC) chromatogram or the most intense peak is selected to produce the base peak chromatogram (BPC). 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 if `aggregation_fun` which can take either `sum` (TIC) or `max` (BPC).

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_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

```
## Not run:
lcmsPlot("cd_example.cdResult") +
  lp_compound_discoverer(
    compounds_query = 'name %in% c("Proline", "Betaine")',
    rt_extend = 5
  ) +
  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")

## End(Not run)
```

`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,
  density = FALSE,
  x_dim = "rt",
  y_dim = "mz",
  fill_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.
density	A logical value indicating whether to show a density plot.
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.

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),
    density = TRUE)
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.

---

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_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_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"
)
```

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.
`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".

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.

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.

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.

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.

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
```

`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
)
```

`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
)
```

---

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
```

---

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(),
    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,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.

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.

Index