

Package: lcmsPlot (via r-universe)

October 26, 2026

Type Package

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

Version 1.1.21

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 (>= 1.1.0), tibble, tidyr, BiocParallel, MSnbase, xcms, MsExperiment, mzR, Spectra, Chromatograms (>= 1.2.0), MsCoreUtils, ProtGenerics, MsBackendMsp, S4Vectors, IRanges, data.table, ggplot2, scales, patchwork, DBI, RSQLite, jsonlite

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

Collate 'lcmsPlot-package.R' 'zzz.R' 'helpers-data.R' 'helpers-s4.R' 'xcms-raw-list.R' 'raw-data.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' 'data-source-mztab.R' 'data-validation.R' 'data-adapters.R' 'lcmsPlotDataContainer-class.R' 'helpers-peaks.R' 'raw-extract.R' 'dataset-chromatograms.R' 'dataset-mass-traces.R' 'dataset-spectra.R' 'dataset-intensity-map.R' 'dataset-total-ion-current.R' 'dataset-rt-diff.R' 'dataset-peak-density.R'

'dataset-peak-count-image.R' 'dataset-purity.R' 'plot-units.R'
 'plot-theme.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'
 'plot-data.R' 'lcmsPlot-class.R' 'layers-data.R'
 'layers-sources.R' 'layers-purity.R' 'layers-layout.R'
 'subset-features.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 8.1.1

Config/pak/sysreqs

cmake libgmp-dev make libicu-dev libuv1-dev libxml2-dev libnetcdf-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-10-26 12:06:24 UTC

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

RemoteRef HEAD

RemoteSha 26fe7cd3348a5da367d1833182fab05527ccb012

Contents

lcmsPlot-package	4
+,lcmsPlotClass,function-method	5
CompoundDiscovererNodeSource	5
CompoundDiscovererNodeSource-class	8
create_xcms_raw_list	8
ExternalDataSource-class	9
get_XCMSnExp_object_example	10
iterate_plot_batches	10
lcmsPlot	11
lcmsPlotApp	13
lcmsPlotClass-class	14
lcmsPlotDataContainer	14
lcmsPlotDataContainer-accessors	15
lcmsPlotDataContainer-class	17

LipidSearchSource	18
LipidSearchSource-class	20
lp_arrange	20
lp_chrom_peak_rects	21
lp_chromatogram	23
lp_compound_discoverer	25
lp_facets	28
lp_get_plot	29
lp_grid	30
lp_intensity_map	31
lp_isolation_window	32
lp_labels	34
lp_layout	35
lp_legend	36
lp_lipid_search	37
lp_mass_trace	39
lp_mztab	40
lp_peak_count_image	41
lp_peak_density	43
lp_purity_distribution	44
lp_purity_overlay	45
lp_purity_timeline	46
lp_rt_diff_plot	47
lp_rt_line	48
lp_spectra	49
lp_theme	52
lp_total_ion_current	53
MsDialPeaksSource	54
MZmineFeatureListsSource	55
MzTabMSource	57
MzTabMSource-class	59
next_plot	59
show,CompoundDiscovererNodeSource-method	60
show,ExternalDataSource-method	61
show,lcmsPlotClass-method	62
show,lcmsPlotDataContainer-method	63
show,LipidSearchSource-method	63
show,MzTabMSource-method	64
show,XcmsRawList-method	64
subset_features	65
XcmsRawList	66
XcmsRawList-class	66

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 = "faahK0"),
  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")
```

`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

<code>node_args</code>	A character path to the <code>node_args.json</code> file passed by Compound Discoverer or an already-parsed list.
<code>sample_paths</code>	A character vector of paths to the raw sample files (e.g. <code>mzML</code>) 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.
<code>metadata</code>	A <code>data.frame</code> (optional) of additional sample metadata. It must contain one row per sample, aligned with <code>sample_paths</code> . 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.

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 <code>xcms::xcmsRaw()</code> 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 <code>xcms::xcmsRaw()</code> indicating which MS level to load. NULL loads all levels. Defaults to NULL.
scanrange	A length-2 integer vector passed to <code>xcms::xcmsRaw()</code> restricting the range of scans to read. NULL reads all scans. Defaults to NULL.
BPPARAM	A <code>BiocParallelParam</code> 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 = "faahK0"),
  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 = "faahK0"),
  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	The data to plot: an XcmsExperiment, MsExperiment, XCMSnExp, MChromatograms, XChromatograms, XChromatogram, XcmsRawList, purityA, a data source (see MZmineFeatureListsSource(), MsDialPeaksSource(), CompoundDiscovererNodeSource(), LipidSearchSource()), a Compound Discoverer .cdResult path, or a character vector of raw file paths (.mzML, .mzXML, .CDF, .raw).
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 = "faahK0"),
  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	<i>Managing LC-MS data for visualisation</i>
---------------------	--

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.

<code>lcmsPlotDataContainer</code>	<i>Create an instance of class <code>lcmsPlotDataContainer</code> from a data object</i>
------------------------------------	--

Description

The `lcmsPlotDataContainer` function standardises the input data object into an `lcmsPlotDataContainer`. See `lcmsPlotDataContainer-class` for the supported data objects.

Usage

```
lcmsPlotDataContainer(
  data_obj,
  sample_id_column = "sample_id",
  metadata = NULL
)
```

Arguments

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

Details

Creating the container is cheap: raw spectra are only read once a layer needs them.

Value

An instance of class `lcmsPlotDataContainer`.

Examples

```
raw_files <- dir(
  system.file("cdf", package = "faahK0"),
  full.names = TRUE,
  recursive = TRUE)[1:5]

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

`lcmsPlotDataContainer-accessors`*Access the data held by an `lcmsPlotDataContainer`*

Description

Accessors for the data structures held by an `lcmsPlotDataContainer`, or by the container of an `lcmsPlotClass` object.

Usage

```
sampleMetadata(object, ...)
```

```
detectedPeaks(object, ...)
```

```
summaries(object, ...)
```

```
## S4 method for signature 'lcmsPlotDataContainer'
```

```
sampleMetadata(object, ...)

## S4 method for signature 'lcmsPlotDataContainer'
spectra(object, ...)

## S4 method for signature 'lcmsPlotDataContainer'
chromatograms(object, ...)

## S4 method for signature 'lcmsPlotDataContainer'
detectedPeaks(object, ...)

## S4 method for signature 'lcmsPlotDataContainer'
summaries(object, ...)

## S4 method for signature 'lcmsPlotClass'
sampleMetadata(object, ...)

## S4 method for signature 'lcmsPlotClass'
spectra(object, ...)

## S4 method for signature 'lcmsPlotClass'
chromatograms(object, ...)

## S4 method for signature 'lcmsPlotClass'
detectedPeaks(object, ...)

## S4 method for signature 'lcmsPlotClass'
summaries(object, ...)
```

Arguments

<code>object</code>	An <code>lcmsPlotDataContainer</code> or <code>lcmsPlotClass</code> object.
<code>...</code>	Ignored.

Details

- `sampleMetadata()` returns the sample metadata as a tibble, with `sample_index`, `sample_id`, and `sample_path` columns. It is the `sampleData()` of the underlying `MsExperiment`.
- `chromatograms()` returns the extracted chromatograms as a `Chromatograms` object.
- `spectra()` returns the extracted spectra as a `Spectra` object.
- `detectedPeaks()` returns the detected peaks selected by the chromatogram layer as a `data.frame`.
- `summaries()` returns the derived datasets as a named list of `data.frames`.

Value

See the description.

Examples

```
raw_files <- dir(
  system.file("cdf", package = "faahK0"),
  full.names = TRUE,
  recursive = TRUE)[1:2]

p <- lcmsPlot(raw_files) + lp_chromatogram()
sampleMetadata(p)
chromatograms(p)
```

lcmsPlotDataContainer-class

A unified storing mechanism for LC-MS data

Description

The `lcmsPlotDataContainer` class holds the input data and every dataset derived from it for plotting, using the `RforMassSpectrometry` data structures where one exists.

Details

Every dataset is keyed by `sample_index`, the row of the sample metadata (`sampleMetadata()`) it belongs to. Feature annotations (such as `feature_id`, `name`, or `formula`) are carried by the rows they describe: `chromData()` columns of the chromatograms, spectra variables of the spectra, or columns of the summaries. Sample metadata is joined in once, when the plot is rendered.

Slots

experiment An `MsExperiment` whose `sampleData()` holds the sample metadata, with columns `sample_index` (the row number), `sample_id`, and `sample_path`. Its spectra are only loaded once a layer needs the raw data.

source The input object, kept for the information that has no `RforMassSpectrometry` counterpart (detected peaks, feature definitions, compound tables, or purity scores). One of: `XcmsExperiment`, `MsExperiment`, `MChromatograms`, `XChromatograms`, `XChromatogram`, `XcmsRawList`, `ExternalDataSource`, `CompoundDiscovererNodeSource`, `LipidSearchSource`, `DBIConnection`, `purityA`, or character (raw file paths). `XCMSnExp` input is stored as an `XcmsExperiment`.

chromatograms A `Chromatograms` object with the extracted chromatograms, held in memory.

spectra A `Spectra` object with the extracted spectra, held in memory. Spectra variables `sample_index`, `feature_id`, and reference link each spectrum to its sample, feature, and library match.

detected_peaks A `data.frame` of the detected peaks selected by the chromatogram layer, with columns `mz`, `mzmin`, `mzmax`, `rt`, `rtmin`, `rtmax`, `into`, `maxo`, and `sample_index`, plus `feature_id` when the peak was matched to a feature.

summaries A named list of `data.frames` holding the derived datasets that have no `RforMassSpectrometry` counterpart: `mass_traces`, `intensity_maps`, `total_ion_current`, `rt_diff`, `peak_density`, `peak_count_image`, and `purity_scores`.

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

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

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 meta-data.

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 = "faahK0"),
  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_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 <code>xcms::plotChromPeaks()</code> . Ignored when the layer decorates an <code>lp_intensity_map()</code> panel, which supplies its own window.
mz_range	An optional length-2 numeric restricting the m/z range, equivalent to ylim in <code>xcms::plotChromPeaks()</code> . Ignored when the layer decorates an <code>lp_intensity_map()</code> 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_chromatogram	<i>Define the chromatograms to plot</i>
-----------------	---

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 = NULL,
  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 of feature identifiers, i.e. the row names of xcms::featureDefinitions() - whatever they have been set to, or the "FT0001"-style defaults assigned during correspondence. This requires the data to be an XCMSnExp or MsExperiment object with grouped peaks; identifiers matching no feature are skipped with a warning. For the matrix and data.frame forms, each feature's identifier (used for labelling and as the feature_id faceting column) is taken from a feature_id column, or else from the row names (e.g. rbind(my_feature = c(mz = 335, rt
----------	--

	= 2800))). Features without one get an <code>xcms::groupnames()</code> -style "M<mz>T<rt>" identifier.
<code>sample_ids</code>	A character vector specifying the sample IDs to include in the plot. If NULL, the function uses the sample IDs specified in the <code>lcmsPlot</code> object.
<code>ppm</code>	A numeric value specifying the mass accuracy (in ppm) used when generating chromatograms. Ignored when the <code>features</code> parameter specifies both <code>mzmin</code> and <code>mzmax</code> .
<code>rt_tol</code>	A numeric value specifying the RT tolerance used when generating chromatograms. Ignored when the <code>features</code> parameter specifies both <code>rtmin</code> and <code>rtmax</code> .
<code>line_type</code>	A character value specifying the line type (from <code>ggplot2</code>). One of: "solid", "dashed", "dotted", "dotdash", "longdash", "twodash".
<code>highlight_peaks</code>	A logical value indicating whether to highlight the detected peaks; the input data must be an <code>XCMSnExp</code> or <code>MsExperiment</code> object.
<code>highlight_peaks_color</code>	A character value indicating the color of the highlighted peaks.
<code>highlight_peaks_mode</code>	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 <code>rtmin</code> to <code>rtmax</code> up to the peak apex), or "point" (marks the peak apex with a point). Defaults to "polygon".
<code>highlight_peaks_factor</code>	A character value indicating the factor from the metadata that determines the color. By default (NULL) the peaks take the colour of their trace: the <code>group_by</code> column of <code>lp_arrange()</code> , else <code>sample_id</code> .
<code>aggregation_fun</code>	A character value indicating which aggregation function to use for the spectra intensities; one of <code>max</code> (base peak chromatogram), <code>sum</code> (total ion current), or <code>mean</code> (averaged ion chromatogram). Only applicable to summary chromatograms.
<code>rt_type</code>	A character value indicating what type of RT to use for the chromatograms. One of <code>uncorrected</code> (default), <code>corrected</code> , or <code>both</code> ; the input data must be an <code>XCMSnExp</code> or <code>MsExperiment</code> object. If both is chosen, this will give access to a metadata column called <code>rt_adjusted</code> that can be used to differentiate the two RT types (e.g., through faceting).
<code>rt_unit</code>	A character value indicating the unit to use for the RT axis; one of "minute" or "second".
<code>intensity_unit</code>	A character value indicating the unit to use for the intensity axis; one of "absolute" or "relative".
<code>fill_gaps</code>	A logical value indicating whether to fill gaps in RT with 0 intensity.
<code>na.rm</code>	A logical value. When TRUE, data points whose intensity is NA are removed before plotting. Defaults to FALSE.
<code>highlight_apices</code>	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 <code>lp_arrange(group_by =)</code> ; without it, samples are stacked.
transform	A function applied to the intensities before plotting, such as <code>log10</code> , 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 <code>log10</code> maps the exact zeros injected by <code>fill_gaps = TRUE</code> to $-\text{Inf}$.

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 specifying the `features` parameter.

Examples

```
raw_files <- dir(
  system.file("cdf", package = "faahK0"),
  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. checked Whether the compound is checked in the Compounds table. Available only once compounds have actually been checked in the Compound Discoverer application: .cdResult files store no such column until then, and the scripting node exports none. Referencing checked in that case raises an error rather than matching nothing.
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

<code>facets</code>	A character vector of factors from the sample metadata to use for faceting.
<code>ncol</code>	A numeric value indicating the number of columns in the layout.
<code>nrow</code>	A numeric value indicating the number of rows in the layout.
<code>free_x</code>	A logical value indicating whether the x-axis scales are allowed to vary across panels.
<code>free_y</code>	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 = "faahK0"),
  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 = "faahK0"),
  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

<code>rows</code>	A character value indicating the factors that represent rows.
<code>cols</code>	A character value indicating the factors that represent columns.
<code>free_x</code>	A logical value indicating whether the x-axis scales are allowed to vary across panels.
<code>free_y</code>	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 = "faahK0"),
  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	<i>Define a 2D intensity map</i>
------------------	----------------------------------

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

<code>mz_range</code>	A numeric value indicating the m/z range of the map.
<code>rt_range</code>	A numeric value indicating the RT range of the map.
<code>sample_ids</code>	A character vector specifying the sample IDs to include in the plot. If <code>NULL</code> , the function uses the sample IDs specified in the <code>lcmsPlot</code> object.
<code>x_dim</code>	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. <code>scale_fill_gradient(low = "white", high = "red")</code>). When NULL (default), uses <code>scale_fill_viridis_c</code> for intensity and <code>scale_fill_viridis_d</code> for density plots. Applies to <code>geom = "tile"</code> and <code>geom = "density"</code> .
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 <code>geom = "point"</code> .
bin_rt	A numeric value giving the retention-time bin width in seconds used when <code>geom = "tile"</code> . 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 <code>geom = "tile"</code> . Defaults to 0.1; high-resolution data usually wants less.
colour_scale	A ggplot2 scale object to use for the colour aesthetic when <code>geom = "point"</code> . When NULL (default), uses <code>scale_colour_viridis_c</code> .

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 = "faahK0"),
  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
```

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

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

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	<i>Define the legend layout</i>
-----------	---------------------------------

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 = "faahK0"),
  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	<i>Define the options to use when plotting LC-MS data coming from a LipidSearch result file.</i>
-----------------	--

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 <i>lipids</i>, 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 Lipid-Search 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_mztab	<i>Define the options to use when plotting LC-MS data coming from an mzTab-M file.</i>
----------	--

Description

Define the options to use when plotting LC-MS data coming from an mzTab-M file.

Usage

```
lp_mztab(features_query = NULL, rt_extend = 10)
```

Arguments

features_query A character value indicating the expression used to select which features to plot. The expression is evaluated on the feature table and can reference the following columns:

feature_id The feature's SMF_ID.

name Compound name (chemical_name of the SML row referencing the feature); NA for unannotated features.

formula Chemical formula (chemical_formula).

database_identifier Database identifier of the compound.

reliability Identification reliability, e.g. 1 to 4.

sml_id The SML_ID of the annotation.

ambiguous TRUE when several SML rows reference the feature.

adduct_ion, charge Adduct ion and charge of the feature.

mz Mass-to-charge ratio (exp_mass_to_charge).

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

into Abundance in the sample; NA where the feature has none.

detected Whether the feature has an abundance in the sample.

The query selects *features*, not feature/sample rows: a predicate such as `into > 1e6` keeps the features above that abundance in at least one sample, and each of them is then extracted from **every** sample.

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

Value

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

See Also[MzTabMSource\(\)](#)**Examples**

```
## A minimal mzTab-M file; see [MzTabMSource()] for the format details.
mzml_dir <- file.path(tempdir(), "mztab-layer-example")
utils::unzip(
  system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
  exdir = mzml_dir)

mztab_path <- file.path(tempdir(), "standards-layer.mzTab")
writeLines(
  c(
    "MTD\tms_run[1]-location\tfile:///data/l-proline-MS1.mzML",
    "MTD\tms_run[2]-location\tfile:///data/l-kynurenine-MS1.mzML",
    "MTD\tassay[1]-ms_run_ref\tms_run[1]",
    "MTD\tassay[2]-ms_run_ref\tms_run[2]",
    paste(
      "SMH", "SML_ID", "SMF_ID_REFS", "chemical_name",
      "reliability", sep = "\t"),
    paste("SML", 1, 1, "L-Proline", 1, sep = "\t"),
    paste("SML", 2, 2, "L-Kynurenine", 3, sep = "\t"),
    paste(
      "SFH", "SMF_ID", "exp_mass_to_charge",
      "retention_time_in_seconds", "retention_time_in_seconds_start",
      "retention_time_in_seconds_end", "abundance_assay[1]",
      "abundance_assay[2]", sep = "\t"),
    paste("SMF", 1, 116.0706, 426, 417, 435, 24050, "null", sep = "\t"),
    paste("SMF", 2, 209.0921, 366, 357, 375, "null", 18700, sep = "\t")),
  mztab_path)

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

## Only the confidently identified features.
p <- lcmsPlot(ds) +
  lp_mztab(features_query = "reliability <= 2", rt_extend = 30) +
  lp_chromatogram(highlight_peaks = TRUE) +
  lp_grid(rows = "sample_id", cols = "name", free_x = TRUE) +
  lp_labels(title = "mzTab-M example", legend = "Sample") +
  lp_legend(position = "bottom")
p
```

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	<i>Plot the peak density for one or more m/z features</i>
-----------------	---

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

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

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	<i>Generate the retention time difference plot between raw and adjusted datasets</i>
-----------------	--

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 = "faahK0"),
  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,
  ppm = 10,
  select = "most_intense",
  consensus_ppm = 20,
  consensus_min_prop = 0.5,
  precursor_marker_shape = 23,
  precursor_marker_position = "bottom",
  precursor_marker_color = "black",
  spectral_match_db = NULL,
  match_target_index = NULL,
  peak_label_size = 3,
  intensity_breaks_by = 20,
  mz_breaks_n = 6,
  auto_facet = TRUE
)
```

Arguments

- | | |
|-------------------------|--|
| <code>sample_ids</code> | A character vector specifying the sample IDs to include in the plot. If <code>NULL</code> , the function uses the sample IDs specified in the <code>lcmsPlot</code> object or the <code>lp_chromatogram</code> function. |
| <code>mode</code> | The method to choose the scan from which to extract the spectra. One of: <code>closest</code> , the closest scan to the specified RT - <code>rt</code> parameter); <code>closest_apex</code> , the closest scan to a detected peak; <code>across_peak</code> , selects scans across a detected peak at a certain interval specified in the <code>interval</code> parameter; <code>precursor</code> , the |

	MS2 scans fragmenting the precursor of each feature (see the section above). mode is not applicable to standalone spectra.
ms_level	The MS level to consider for the scan. Ignored by mode = "precursor", which always reads MS2 scans.
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.
ppm	When mode = "precursor", the tolerance (in ppm) between a scan's precursor m/z and the m/z of the feature's detected peak. Defaults to 10.
select	When mode = "precursor", which of a feature's matching MS2 scans to plot in each sample. One of "most_intense" (default), the scan with the most intense precursor, or with the highest total ion current when the file does not report precursor intensities; "closest_apex", the scan closest to the peak apex; "all", every matching scan; or "consensus", one spectrum combining the matching scans.
consensus_ppm	When select = "consensus", the tolerance (in ppm) within which fragments of different scans are combined. Defaults to 20.
consensus_min_prop	When select = "consensus", the minimum proportion of the combined scans a fragment must be found in to be kept. Defaults to 0.5.
precursor_marker_shape	The shape of the marker drawn at the precursor m/z of MS2 spectra, as a ggplot2 point shape: a number from 0 to 25 or a name such as "triangle down filled". Defaults to 23 (a filled diamond).
precursor_marker_position	Where the precursor marker is drawn: on the m/z axis ("bottom", the default) or on top of the precursor's peak ("top"), the highest peak within ppm of the precursor m/z. A spectrum in which the precursor left no peak has its marker on the m/z axis.
precursor_marker_color	A character value giving the color of the precursor marker, used for both its outline and its fill. Defaults to "black".
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.

mz_breaks_n	A numeric value specifying the approximate number of m/z axis breaks, passed to <code>scales::pretty_breaks(n = ...)</code> . Defaults to 6.
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).

MS2 spectra of a feature

With data-dependent acquisition (DDA), `mode = "precursor"` plots the fragmentation spectra of each feature: the MS2 scans whose precursor m/z lies within ppm of the m/z of the feature's detected peak and whose RT lies within the peak's `[rtmin, rtmax]`. In a sample where the feature has no detected peak, the extraction window given to `lp_chromatogram(features = )` is used instead. `select` decides which of the matching scans are plotted. Each spectrum keeps the `feature_id` of its feature, and its precursor m/z is marked, by default with a diamond on the m/z axis (see the `precursor_marker_*` parameters). Features with no matching MS2 scan are skipped with a warning. Data-independent acquisition (MS2 scans without a defined precursor) is not supported.

Examples

```
raw_files <- dir(
  system.file("cdf", package = "faahK0"),
  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

## The MS2 spectrum of a feature, from data-dependent acquisition
mzml_dir <- file.path(tempdir(), "precursor-example")
utils::unzip(
  system.file("extdata", "standards-mzml.zip", package = "lcmsPlot"),
  exdir = mzml_dir)

ms2_file <- file.path(mzml_dir, "mzml", "l-kynurenine-MS2.mzML")

p <- lcmsPlot(ms2_file) +
  lp_chromatogram(
    features = rbind(kynurenine = c(mz = 209.0918, rt = 365)),
    rt_tol = 15) +
  lp_spectra(mode = "precursor")
p

```

lp_theme

*Define the theme and colour palette of the plot***Description**

The `lp_theme` function styles every panel of the plot with a theme preset and colours the series (the `group_by` column of `lp_arrange()`, and the highlighted peaks) with a discrete palette. Continuous colour scales, such as those of intensity maps or purity scores, are left unchanged.

Usage

```

lp_theme(
  preset = "minimal",
  palette = "okabe_ito",
  base_size = NULL,
  line_width = NULL
)

```

Arguments

<code>preset</code>	The theme preset. One of "minimal" (default), "classic", "publication" (white background, axis lines, no minor grid, and an 8 pt base size for print), or any ggplot2 theme object.
<code>palette</code>	The discrete palette. One of "okabe_ito" (default; colourblind-safe, falling back to viridis beyond 8 groups), "viridis", "ggplot2" (the default ggplot2 hues), or a character vector of colours. A named vector maps each group to its colour; an unnamed one is used in order.
<code>base_size</code>	A numeric value indicating the base font size in points. NULL uses the preset's default (11, or 8 for "publication").
<code>line_width</code>	A numeric value indicating the width of the plotted lines (chromatogram traces, spectrum sticks) and of the theme's lines. NULL scales it with <code>base_size</code> . <code>base_size</code> and <code>line_width</code> only affect the theme when <code>preset</code> is one of the named presets.

Details

A theme added after `lp_get_plot()` (e.g. `lp_get_plot() + theme(...)`) still overrides the preset.

Value

A function that takes an `lcmsPlot` object and returns a modified version with the specified theme options stored in `options$theme`. 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 = "faahK0"),
  full.names = TRUE,
  recursive = TRUE)[1:4]

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")

## Style the plot for print
p + lp_theme(preset = "publication", palette = "okabe_ito", base_size = 8)

## Colour each sample explicitly
p + lp_theme(palette = c(
  ko15 = "#0072B2", ko16 = "#56B4E9",
  ko18 = "#D55E00", ko19 = "#E69F00"))
```

`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: one value per MS1 scan. It works both with XCMS objects (`XcmsExperiment`, `MsExperiment`, `XCMSnExp`), `XcmsRawList` objects, 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", initial = 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 <code>lcmsPlot</code> object.
type	A character value indicating the type of plot; one of "boxplot", "violin", "jitter".
initial	A logical value, as in <code>Spectra::tic()</code> . With TRUE (the default) each scan's TIC is the one recorded in the file's scan header; with FALSE it is the sum of the scan's peak intensities. The two can differ widely: netCDF headers report the instrument's total ion current, which includes signal that is not among the stored peaks. TIC chromatograms (<code>lp_chromatogram(aggregation_fun = "sum")</code>) sum the peaks of netCDF files, so on such files they match <code>initial = FALSE</code> . Scans whose header reports no TIC are always summed from their peaks.

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	<i>Create an MS-DIAL data source from peak lists</i>
-------------------	--

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 <code>data.frame</code> is created from <code>sample_paths</code> .

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
)
```

MzTabMSource*Create a data source from an mzTab-M file*

Description

Reads the small molecule features (SMF) of an mzTab-M 2.0 file - as exported by MS-DIAL, MZmine or OpenMS, among others - and pairs them with the raw files they were computed from. Chromatograms are later extracted from the raw files supplied via `sample_paths`, and the features can be selected with `lp_mztab()`.

Usage

```
MzTabMSource(mztab_path, sample_paths, metadata_path = NULL)
```

Arguments

<code>mztab_path</code>	A character path to an mzTab-M 2.0 file.
<code>sample_paths</code>	A character vector of paths to raw sample files (e.g., mzML).
<code>metadata_path</code>	A character value (optional) indicating the path to a metadata file (CSV or TSV), with one row per sample path. If NULL, a metadata data.frame is created from <code>sample_paths</code> . A <code>sample_id</code> column (the file name without extension) is added when the metadata has none.

Value

An object of class `MzTabMSource`, which extends `ExternalDataSource`.

Sample-to-file mapping

Each `assay[n]-ms_run_ref` of the metadata section names an `ms_run[m]`, whose `ms_run[m]-location` is the URI of the raw file. The location's basename is matched to the basenames of `sample_paths`, ignoring the file extension when there is no exact match, so a file listing `.raw` files works with converted `.mzML` files. Every run must match a sample path. Sample paths that no run matches are still plotted, with no peaks to highlight.

Feature columns

Each SMF row is one feature across all assays. It gives one peak per assay with an abundance:

feature_id SMF_ID.

mz exp_mass_to_charge.

rt, rtmin, rtmax retention_time_in_seconds and its _start/_end bounds, shared by every sample. When the bounds are null they default to rt, so the extraction window is $rt \pm rt_extend$.

into abundance_assay[n].

maxo NA: mzTab-M reports no peak height.

adduct_ion, charge As reported.

Features are annotated from the small molecule summary (SML) rows that list them in SMF_ID_REFS: name (chemical_name), formula (chemical_formula), database_identifier, reliability and sml_id. A feature referenced by several SML rows keeps the first, and its ambiguous column is TRUE. Unannotated features have NA annotations.

See Also

[lp_mztab\(\)](#)

Examples

```
mzml_dir <- file.path(tempdir(), "mztab-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"))

## A minimal mzTab-M file: two runs, one feature found in each.
mztab_path <- file.path(tempdir(), "standards.mzTab")
writeLines(
  c(
    "MTD\tmzTab-version\t2.0.0-M",
    "MTD\tms_run[1]-location\tfile:///data/l-proline-MS1.mzML",
    "MTD\tms_run[2]-location\tfile:///data/l-kynurenine-MS1.mzML",
    "MTD\tassay[1]-ms_run_ref\tms_run[1]",
    "MTD\tassay[2]-ms_run_ref\tms_run[2]",
    """,
    paste(
      "SMH", "SML_ID", "SMF_ID_REFS", "chemical_name",
      "chemical_formula", "reliability", sep = "\t"),
    paste("SML", 1, 1, "L-Proline", "C5H9NO2", 1, sep = "\t"),
    paste("SML", 2, 2, "L-Kynurenine", "C10H12N2O3", 2, sep = "\t"),
    """,
    paste(
      "SFH", "SMF_ID", "adduct_ion", "exp_mass_to_charge", "charge",
      "retention_time_in_seconds", "retention_time_in_seconds_start",
```

```

      "retention_time_in_seconds_end", "abundance_assay[1]",
      "abundance_assay[2]", sep = "\t"),
  paste(
    "SMF", 1, "[M+H]1+", 116.0706, 1, 426, 417, 435, 24050, "null",
    sep = "\t"),
  paste(
    "SMF", 2, "[M+H]1+", 209.0921, 1, 366, 357, 375, "null", 18700,
    sep = "\t")),
  mztab_path)

ds <- MzTabMSource(mztab_path, sample_paths)
ds

p <- lcmsPlot(ds) +
  lp_mztab(rt_extend = 30) +
  lp_chromatogram(highlight_peaks = TRUE) +
  lp_grid(rows = "sample_id", cols = "name", free_x = TRUE)
p

```

MzTabMSource-class	<i>mzTab-M data source</i>
--------------------	----------------------------

Description

An S4 class wrapping the small molecule features of an mzTab-M 2.0 file. It extends ExternalDataSource, so its peaks hold the features detected in each sample.

Slots

compounds A data.frame with one row per feature and sample - including samples in which the feature has no abundance - used to drive chromatogram extraction.

See Also

[MzTabMSource\(\)](#), [lp_mztab\(\)](#)

next_plot	<i>Move to the next plot object (batch-mode)</i>
-----------	--

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 = "faahK0"),
  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,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 = "faahK0"),
  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 <code>lcmsPlotDataContainer</code> .
--------	---

Value

Invisible NULL

Examples

```
raw_files <- dir(
  system.file("cdf", package = "faahK0"),
  full.names = TRUE,
  recursive = TRUE)[1:5]

lcmsPlotDataContainer(raw_files)
```

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,MzTabMSource-method
<i>Show a summary of a MzTabMSource object</i>

Description

Show a summary of a MzTabMSource object

Usage

```
## S4 method for signature 'MzTabMSource'
show(object)
```

Arguments

object An instance of class MzTabMSource.

Value

Invisible NULL

show,XcmsRawList-method
<i>Show a summary of an XcmsRawList object</i>

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.

subset_features	<i>Keep the data matching a condition on its annotations</i>
-----------------	--

Description

subset_features() narrows the data already extracted for an lcmsPlotClass object, without reading the raw data again. Each condition is applied to every dataset that carries at least one of the columns it refers to: the chromatograms (through their chromData() columns), the spectra (through their spectra variables), the detected peaks, and the summaries. Datasets that carry none of them are kept whole.

Usage

```
subset_features(obj, ...)
```

Arguments

obj	An instance of class lcmsPlotClass.
...	Conditions on the dataset columns, as in <code>dplyr::filter()</code> .

Details

This is useful to draw one plot per compound or feature out of a single extraction.

Value

The lcmsPlotClass object with the matching data only.

Examples

```
raw_files <- dir(
  system.file("cdf", package = "faahK0"),
  full.names = TRUE,
  recursive = TRUE)[1:2]

features <- data.frame(
  feature_id = c("A", "B"),
  mzmin = c(334.9, 344.0),
  mzmax = c(335.1, 344.2),
  rtmin = c(2700, 3400),
  rtmax = c(2900, 3500))

p <- lcmsPlot(raw_files) + lp_chromatogram(features = features)
p_a <- subset_features(p, feature_id == "A")
```

XcmsRawList	<i>Create an XcmsRawList object</i>
-------------	-------------------------------------

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 = "faahK0"),
  full.names = TRUE,
  recursive = TRUE
)[c(1, 2)]
raw1 <- xcms::xcmsRaw(paths[1])
raw2 <- xcms::xcmsRaw(paths[2])
x1 <- XcmsRawList(raw1, raw2)
```

XcmsRawList-class	<i>A list of xcmsRaw objects representing multiple samples</i>
-------------------	--

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

`+`, `lcmsPlotClass`, function-method, 5

`BiocParallelParam`, 12

`chromatograms`, `lcmsPlotClass`-method
(`lcmsPlotDataContainer`-accessors), 15

`chromatograms`, `lcmsPlotDataContainer`-method
(`lcmsPlotDataContainer`-accessors), 15

`CompoundDiscovererNodeSource`, 5

`CompoundDiscovererNodeSource`-class, 8

`create_xcms_raw_list`, 8

`detectedPeaks`
(`lcmsPlotDataContainer`-accessors), 15

`detectedPeaks`, `lcmsPlotClass`-method
(`lcmsPlotDataContainer`-accessors), 15

`detectedPeaks`, `lcmsPlotDataContainer`-method
(`lcmsPlotDataContainer`-accessors), 15

`dplyr::filter()`, 65

`ExternalDataSource`-class, 9

`get_XCMSnExp_object_example`, 10

`iterate_plot_batches`, 10

`iterate_plot_batches`, `lcmsPlotClass`, function-method
(`iterate_plot_batches`), 10

`lcmsPlot`, 11

`lcmsPlot`-package, 4

`lcmsPlotApp`, 13

`lcmsPlotClass`-class, 14

`lcmsPlotDataContainer`, 14

`lcmsPlotDataContainer`-accessors, 15

`lcmsPlotDataContainer`-class, 17

`LipidSearchSource`, 18

`LipidSearchSource()`, 38

`LipidSearchSource`-class, 20

`lp_arrange`, 20

`lp_arrange()`, 24, 52

`lp_chrom_peak_rects`, 21

`lp_chromatogram`, 23

`lp_compound_discoverer`, 25

`lp_compound_discoverer()`, 6

`lp_facets`, 28

`lp_get_plot`, 29

`lp_get_plot()`, 53

`lp_grid`, 30

`lp_intensity_map`, 31

`lp_isolation_window`, 32

`lp_labels`, 34

`lp_layout`, 35

`lp_legend`, 36

`lp_lipid_search`, 37

`lp_lipid_search()`, 19

`lp_mass_trace`, 39

`lp_mztab`, 40

`lp_mztab()`, 57–59

`lp_peak_count_image`, 41

`lp_peak_density`, 43

`lp_purity_distribution`, 44

`lp_purity_overlay`, 45

`lp_purity_timeline`, 46

`lp_rt_diff_plot`, 47

`lp_rt_line`, 48

`lp_spectra`, 49

`lp_theme`, 52

`lp_total_ion_current`, 53

`MsDialPeaksSource`, 54

`MZmineFeatureListsSource`, 55

`MzTabMSource`, 57

`MzTabMSource()`, 41, 59

`MzTabMSource`-class, 59

`next_plot`, 59

`next_plot`, `lcmsPlotClass`-method
 (`next_plot`), [59](#)

`sampleMetadata`
 (`lcmsPlotDataContainer`-accessors),
 [15](#)

`sampleMetadata`, `lcmsPlotClass`-method
 (`lcmsPlotDataContainer`-accessors),
 [15](#)

`sampleMetadata`, `lcmsPlotDataContainer`-method
 (`lcmsPlotDataContainer`-accessors),
 [15](#)

`shiny::runApp()`, [13](#)

`shiny::shinyApp`, [13](#)

`show`, `CompoundDiscovererNodeSource`-method,
 [60](#)

`show`, `ExternalDataSource`-method, [61](#)

`show`, `lcmsPlotClass`-method, [62](#)

`show`, `lcmsPlotDataContainer`-method, [63](#)

`show`, `LipidSearchSource`-method, [63](#)

`show`, `MzTabMSource`-method, [64](#)

`show`, `XcmsRawList`-method, [64](#)

`spectra`, `lcmsPlotClass`-method
 (`lcmsPlotDataContainer`-accessors),
 [15](#)

`spectra`, `lcmsPlotDataContainer`-method
 (`lcmsPlotDataContainer`-accessors),
 [15](#)

`stats::density()`, [43](#)

`subset_features`, [65](#)

`summaries`
 (`lcmsPlotDataContainer`-accessors),
 [15](#)

`summaries`, `lcmsPlotClass`-method
 (`lcmsPlotDataContainer`-accessors),
 [15](#)

`summaries`, `lcmsPlotDataContainer`-method
 (`lcmsPlotDataContainer`-accessors),
 [15](#)

`XcmsRawList`, [66](#)

`XcmsRawList`-class, [66](#)