Package 'lcmsPlot'

February 20, 2026

Type Package

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

Version 0.99.16

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, tidyr, BiocParallel, MSnbase, xcms, MsExperiment, mzR, Spectra, MsBackendMsp, S4Vectors, ggplot2, scales, patchwork, DBI, RSQLite

Suggests knitr, rmarkdown, BiocStyle, openxlsx, faahKO, testthat (>= 3.0.0)

Collate 'lcmsPlot-package.R' 'zzz.R' 'helpers-data.R' 'helpers-s4.R' 'io-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' 'constructors-chromatograms.R' 'constructors-spectra.R' 'data-adapters.R' 'plot-units.R' 'plot-chromatogram.R' 'plot-mass-trace.R' 'plot-spectrum.R' 'plot-total-ion-current.R' 'plot-intensity-map.R' 'plot-rt-diff.R' 'plot-variants.R' 'plot.R' 'lcmsPlotDataContainer-class.R' 'creators-chromatograms.R' 'creators-intensity-map.R' 'creators-rt-diff.R' 'creators-spectra.R' 'creators-total-ion-current.R' 'lcmsPlot-class.R'

VignetteBuilder knitr

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

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

biocViews Metabolomics, MassSpectrometry

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

Config/testthat/edition 3

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

git_branch devel

git_last_commit d956f01

git_last_commit_date 2026-02-07

Repository Bioconductor 3.23

Date/Publication 2026-02-19

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

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

Contents

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

Description

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

Details

Main features

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

Author(s)

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

Authors:

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

See Also

Useful links:

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

+,lcmsPlotClass,function-method

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

Description

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

Usage

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

Arguments

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

Value

An instance of class `lcmsPlotClass`.

Examples

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

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

convert_rt_to_seconds *Convert retention times to seconds*

Description

Convert retention time columns from minutes to seconds.

Usage

```
convert_rt_to_seconds(peaks)
```

Arguments

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

Value

peaks with retention time columns converted to seconds.

---

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

Description

Create a base peak or total ion current chromatogram

Usage

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

Arguments

raw_data	An instance of class mzR.
aggregation_fun	A function indicating the aggregation method. One of "sum" or "max".
rt_adjusted	A numeric vector representing the adjusted RT values. If NULL it will use the raw RT values.

Value

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

---

create_chromatogram *Create an extracted ion chromatogram*

Description

Create an extracted ion chromatogram

Usage

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

Arguments

raw_data	An instance of class mzR.
mz_range	A numeric vector indicating the m/z range.
rt_range	A numeric vector indicating the RT range.
ms_level	A numeric value indicating the MS level of the scans to consider.
fill_gaps	A logical indicating whether to fill gaps between scans with zeros.
adjusted_rt	A data.frame

Value

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

---

create_chromatograms_from_compound_discoverer

Creates an instance of class lcmsPlotDataContainer *from a Compound Discoverer results object.*

Description

Creates an instance of class lcmsPlotDataContainer from a Compound Discoverer results object.

Usage

```
create_chromatograms_from_compound_discoverer(obj, options)

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

Arguments

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

Value

An instance of class lcmsPlotDataContainer with chromatograms of the specified Compound Discoverer compounds.

`create_chromatograms_from_features`
Creates an instance of class `lcmsPlotDataContainer` *from a features* `matrix` *or* `data.frame`

Description

The input features are specified in the `options` list under `chromatograms$features`.

Usage

```
create_chromatograms_from_features(obj, options)

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

Arguments

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

Value

An instance of class `lcmsPlotDataContainer` with chromatograms of the specified features.

`create_chromatograms_from_feature_ids`
Create an instance of class `lcmsPlotDataContainer` *from feature IDs*

Description

The input features are specified in the `options` list under `chromatograms$features`.

Usage

```
create_chromatograms_from_feature_ids(obj, options)

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

Arguments

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

Value

An instance of class `lcmsPlotDataContainer` with chromatograms of the specified feature IDs.

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

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

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

`create_full_rt_chromatograms` *Creates an instance of class* `lcmsPlotDataContainer` *from base peak chromatograms (BPC) or total ion chromatograms (TIC)*

Description

The type of chromatograms is defined in `options$chromatograms$aggregation_fun`.

Usage

```
create_full_rt_chromatograms(obj, options)

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

Arguments

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

Value

An instance of class `lcmsPlotDataContainer` with BPC or TIC.

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

Description

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

Usage

```
create_intensity_map(obj, options)

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

Arguments

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

Value

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

---

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

Description

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

Usage

```
create_rt_diff(obj, options)

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

Arguments

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

Value

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

---

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

Description

Create an instance of class `lcmsPlotDataContainer` from spectra

Usage

```
create_spectra(obj, options)

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

Arguments

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

Value

An lcmsPlotDataContainer object with the created spectra.

---

create_spectra_for_sample

Create spectra for a single sample

Description

Create spectra for a single sample

Usage

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

Arguments

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

Value

A data.frame representing spectra with columns mz, intensity, and rt.

`create_spectrum_from_closest_scan_to_rt` *Create a spectrum of the closest scan to the specified RT*

Description

Create a spectrum of the closest scan to the specified RT

Usage

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

Arguments

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

Value

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

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

Description

Create a spectrum of the specified scan

Usage

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

Arguments

`raw_data`	An instance of class `mzR`.
`sample_metadata`	A `data.frame` indicating the sample metadata.
`scan_index`	A `numeric` value indicating the scan index.

Value

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

create_total_ion_current

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

Description

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

Usage

```
create_total_ion_current(obj, options)

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

Arguments

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

Value

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

default_options

Get the default options

Description

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

Usage

```
default_options()
```

Value

A named `list` containing all default options.

detect_separator *Detect field separator from file extension*

Description

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

Usage

```
detect_separator(path)
```

Arguments

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

Value

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

ExternalDataSource-class *External data source wrapper*

Description

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

Slots

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

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

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

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

Description

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

Usage

```
get_adjusted_rts(obj)
```

Arguments

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

Details

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

Value

A data.frame with one row per detected feature, containing:

file_index Index of the originating raw data file.

raw_rt Original (unadjusted) retention time.

adj_rt Adjusted retention time after RT correction.

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

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

Description

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

Usage

```
get_detected_peaks(obj)

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

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

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

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

Arguments

obj	A data object containing or representing samples.

Details

Supported inputs behave as follows:

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

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

Value

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

---

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

Description

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

Usage

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

Arguments

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

Value

A list of feature data (see get_feature_data).

---

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

Description

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

Usage

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

Arguments

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

Value

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

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

Description

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

Usage

```
get_grouped_peaks(obj)

## Default S3 method:
get_grouped_peaks(obj)

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

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

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

Arguments

obj	A data object containing or representing samples.

Value

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

---

get_grouping_variables *Get the grouping variables for a plot*

Description

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

Usage

```
get_grouping_variables(opts)
```

Arguments

opts	A list of plot options.

Value

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

---

get_metadata *Get the metadata associated with the input object*

Description

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

Usage

```
get_metadata(obj, sample_id_column, metadata)

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

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

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

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

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

Arguments

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

Details

Depending on the class of `obj`, metadata may be:

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

Across all methods, the returned metadata is enriched with:

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

Value

A `data.frame` containing standardised metadata with at least `sample_index`, `sample_id`, and `sample_path`.

Character vector input (`character`)

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

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

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

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

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

`XCMSnExp` input

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

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

The returned metadata always includes:

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

`MsExperiment` **input**

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

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

The returned metadata includes:

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

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

Description

Compute an m/z range given a ppm tolerance

Usage

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

Arguments

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

Value

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

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

---

Description

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

Usage

```
get_workflow_input_files(conn)
```

Arguments

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

Value

A `data.frame` containing information about the workflow input files.

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

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

`get_xic_traces_from_compounds`
Extract XIC traces associated with selected compounds

Description

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

Usage

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

Arguments

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

Details

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

Value

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

---

io_close_raw_data *Close open* mzR *object connections*

Description

Close open mzR object connections

Usage

```
io_close_raw_data(raw_data)
```

Arguments

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

Value

NULL

io_get_raw_data — *Get the* mzR *objects for a set of sample paths*

Description

Get the mzR objects for a set of sample paths

Usage

```
io_get_raw_data(sample_paths)
```

Arguments

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

Value

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

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

Description

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

Usage

```
is_cd_result(obj)
```

Arguments

obj	An object to test.

Value

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

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

Description

Check whether a path corresponds to a Compound Discoverer results file

Usage

```
is_cd_results_path(path)
```

Arguments

path	A character value giving a file system path.

Value

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

is_xcms_data *Check whether an object is from XCMS*

Description

Check whether an object is from XCMS

Usage

```
is_xcms_data(obj)
```

Arguments

obj	The input object to check.

Value

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

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

Description

Check whether an object is an XCMS data container

Usage

```
is_xcms_processed_data(obj)
```

Arguments

obj	The input object to check.

Value

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

iterate_plot_batches *Iterate on the batches of plots*

Description

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

Usage

```
iterate_plot_batches(object, iter_fn)

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

Arguments

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

Value

NULL (called for its side effect).

Examples

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

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

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

---

lcmsPlot *Create an* lcmsPlotClass *object*

Description

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

Usage

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

Arguments

dataset	An object of type XCMSnExp, MsExperiment, MZmineSource, or character. If a character vector is supplied, it will be interpreted as a list of mzML paths.
sample_id_column	A character value indicating which column should be used as the sample ID. By default it is "sample_id".

metadata	A data.frame containing the samples metadata in case it is not provided in the dataset object.
batch_size	A numeric value indicating the number of samples per batch. This parameter is necessary when plotting multiple batches using the iterate_plot_batches or next_plot functions.
BPPARAM	A BiocParallelParam object for enabling parallelism. See BiocParallelParam for more information.

Value

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

Examples

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

p <- lcmsPlot(raw_files)
```

---

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

Description

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

Slots

options A list to store the plot options.

data An instance of class lcmsPlotDataContainer.

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

plot A patchwork object representing the underlying plot object.

General information

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

Preferred usage

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

lcmsPlotDataContainer-class

A unified storing mechanism for LC-MS data

Description

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

Slots

data_obj The data object. One of: XCMSnExp, MsExperiment or character representing mzML paths.

metadata A data.frame containing the sample metadata.

chromatograms A data.frame containing the chromatograms.

mass_traces A data.frame containing the mass traces.

spectra A data.frame containing the spectra.

total_ion_current A data.frame containing the total ion current.

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

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

additional_metadata A data.frame containing additional information attached to datasets through a column called additional_metadata_index.

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

lp_arrange *Define the arrangement of chromatograms*

Description

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

Usage

```
lp_arrange(group_by)
```

Arguments

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

Value

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

Examples

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

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

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

---

lp_chromatogram *Define the chromatograms to plot*

Description

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

Usage

```
lp_chromatogram(
  features = NULL,
  sample_ids = NULL,
  ppm = 10,
  rt_tol = 10,
  highlight_peaks = FALSE,
  highlight_peaks_color = NULL,
  highlight_peaks_factor = "sample_id",
  aggregation_fun = "max",
  rt_type = "uncorrected",
  rt_unit = "second",
  intensity_unit = "absolute",
  fill_gaps = FALSE,
  highlight_apices = list(column = NULL, top_n = NULL)
)
```

Arguments

features	Specifies which features to generates the chromatogram for. This can be either: a matrix with columns mz and rt (optional); a matrix with columns mzmin, mzmax, rtmin (optional), rtmax (optional); a data.frame with columns sample_id, mz and rt (optional); a data.frame with columns sample_id, mzmin, mzmax, rtmin (optional), rtmax (optional); a character vector representing the grouped peaks (feature) names as returned by xcms::groupnames - requires the data to be an XCMSnExp or MsExperiment object with grouped peaks.
sample_ids	A character vector specifying the sample IDs to include in the plot. If NULL, the function uses the sample IDs specified in the lcmsPlot object.
ppm	A numeric value specifying the mass accuracy (in ppm) used when generating chromatograms. Ignored when the features parameter specifies both mzmin and mzmax.
rt_tol	A numeric value specifying the RT tolerance used when generating chromatograms. Ignored when the features parameter specifies both rtmin and rtmax.
highlight_peaks	A logical value indicating whether to highlight the detected peaks; the input data must be an XCMSnExp or MsExperiment object.
highlight_peaks_color	A character value indicating the color of the highlighted peaks.
highlight_peaks_factor	A character value indicating the factor from the metadata that determines the color. By default it colors by sample_id.
aggregation_fun	A character value indicating which aggregation function to use for the spectra intensities; one of max or sum. Only applicable to summary chromatograms.
rt_type	A chracter value indicating what type of RT to use for the chromatograms. One of uncorrected (default), corrected, or both; the input data must be an XCMSnExp or MsExperiment object. If both is chosen, this will give access to a metadata column called rt_adjusted that can be used to differentiate the two RT types (e.g., through faceting).
rt_unit	A character value indicating the unit to use for the RT axis; one of "minute" or "second".
intensity_unit	A character value indicating the unit to use for the intensity axis; one of "absolute" or "relative".
fill_gaps	A logical value indicating whether to fill gaps in RT with 0 intensity.
highlight_apices	A logical value indicating whether to highlight apices with the corresponding RT values in a chromatogram.

Value

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

Summary chromatograms

In this type of chromatogram, the intensities of the spectra from each scan in an LC–MS dataset are either summed to produce the total ion current (TIC) chromatogram or the most intense peak is selected to produce the base peak chromatogram (BPC). To create such chromatograms do not specify the `features` parameter as that will create the chromatograms for the selected features. In this context, the main parameter if `aggregation_fun` which can take either `sum` (TIC) or `max` (BPC).

Feature chromatograms

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

Examples

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

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

p
```

---

lp_compound_discoverer

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

Description

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

Usage

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

Arguments

`compounds_query`

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

name Compound name.

formula Chemical formula of the compound.

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

rt Retention time of the compound (in seconds).

rtmin Minimum retention time of the compound peak.

rtmax Maximum retention time of the compound peak.

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

maxo Maximum observed peak intensity.

into Integrated peak area reported by Compound Discoverer.

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

Value

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

Examples

```
## Not run:
lcmsPlot("cd_example.cdResult") +
  lp_compound_discoverer(
    compounds_query = 'name %in% c("Proline", "Betaine")',
    rt_extend = 5
  ) +
  lp_chromatogram(highlight_peaks = TRUE) +
  lp_grid(rows = "sample_id", cols = "name", free_x = TRUE) +
  lp_labels(title = "Compound Discoverer example", legend = "Sample") +
  lp_legend(position = "bottom")

## End(Not run)
```

`lp_facets` *Define the plot's faceting*

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

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

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

---

lp_get_plot *Get the underlying plot object.*

Description

Get the underlying plot object.

Usage

```
lp_get_plot()
```

Value

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

Examples

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

## Create faceted chromatogram plots with a reference RT line
p <- lcmsPlot(raw_files) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
  lp_facets(facets = 'sample_id', ncol = 4) +
  lp_rt_line(intercept = 2800, line_type = 'solid', color = 'red')
p

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

---

lp_grid *Define a gridded plot*

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

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

## Create feature chromatograms for the specified samples
p <- lcmsPlot(raw_files, metadata = metadata) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900)))
p

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

---

lp_intensity_map *Define a 2D intensity map*

Description

The `lp_intensity_map` function produces an intensity map in which retention time is shown on the x-axis, m/z on the y-axis, and signal intensity is represented at each corresponding coordinate.

Usage

```
lp_intensity_map(mz_range, rt_range, sample_ids = NULL, density = FALSE)
```

Arguments

`mz_range`	A `numeric` value indicating the m/z range of the map.
`rt_range`	A `numeric` value indicating the RT range of the map.
`sample_ids`	A `character` vector specifying the sample IDs to include in the plot. If `NULL`, the function uses the sample IDs specified in the `lcmsPlot` object.
`density`	A `logical` value indicating whether to show a density plot.

Value

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

Examples

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

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

---

lp_labels *Define the labels of the plot*

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

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

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

---

lp_layout *Define the plot layout*

Description

Define the plot layout

Usage

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

Arguments

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

Value

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

Examples

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

## Plot chromatograms and spectra for selected samples and features
p <- lcmsPlot(data_obj, sample_id_column = 'sample_name') +
```

```
  lp_chromatogram(
    features = rbind(
      c(mzmin = 334.9, mzmax = 335.1, rtmin = 2700, rtmax = 2900),
      c(mzmin = 278.99721, mzmax = 279.00279, rtmin = 2740, rtmax = 2840)
    ),
    sample_ids = 'ko15',
    highlight_peaks = TRUE
  ) +
  lp_spectra(mode = "closest_apex", ms_level = 1) +
  lp_facets(facets = "feature_id", ncol = 2)

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

`lp_legend` *Define the legend layout*

Description

Define the legend layout

Usage

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

Arguments

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

Value

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

Examples

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

## Create a chromatogram plot grouped by sample with a custom legend label
p <- lcmsPlot(raw_files, batch_size = 2) +
  lp_chromatogram(features = rbind(c(
    mzmin = 334.9,
    mzmax = 335.1,
    rtmin = 2700,
    rtmax = 2900))) +
```

```
  lp_arrange(group_by = "sample_id") +
  lp_labels(legend = "Sample")
p

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

`lp_mass_trace` *Define the mass trace to plot*

Description

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

Usage

```
lp_mass_trace()
```

Value

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

Examples

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

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

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

p
```

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

Description

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

Usage

```
lp_rt_diff_plot()
```

Value

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

Examples

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

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

Description

Define a vertical line on a retention time value

Usage

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

Arguments

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

Value

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

Examples

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

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

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

---

lp_spectra *Define the spectra to plot*

Description

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

Usage

```
lp_spectra(
  sample_ids = NULL,
  mode = "closest_apex",
  ms_level = 1,
  rt = NULL,
  scan_index = NULL,
  interval = 3,
  spectral_match_db = NULL,
  match_target_index = NULL
)
```

Arguments

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

Value

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

Spectra associated with chromatograms

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

Standalone spectra

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

Examples

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

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

`lp_total_ion_current` *Define the total ion current (TIC)*

Description

The `lp_total_ion_current` generates summary data for the total ion current (TIC) of the selected samples.

Usage

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

Arguments

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

Value

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

Examples

```
data_obj <- get_XCMSnExp_object_example()

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

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

Description

Merge two data frames using a row-index match

Usage

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

Arguments

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

Value

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

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

Description

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

Usage

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

Arguments

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

Value

An object of class ExternalDataSource.

Examples

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

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

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

msdial_peaks_1 <- data.frame(
  "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),
  check.names = FALSE
)

msdial_peaks_2 <- data.frame(
  "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),
  check.names = FALSE
)

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 <- data.frame(
  "row m/z" = c(100.1, 200.2),
  "sample1.mzML Feature status" = c("DETECTED", "DETECTED"),
  "sample1.mzML Feature m/z" = c(100.1, 200.2),
  "sample1.mzML Feature RT" = c(300, 600),
  "sample1.mzML Peak area" = c(10000, 20000),
  "sample1.mzML Peak height" = c(500, 800),
  "sample1.mzML Feature m/z min" = c(99.9, 199.9),
  "sample1.mzML Feature m/z max" = c(100.3, 200.4),
  "sample1.mzML Feature RT start" = c(290, 590),
  "sample1.mzML Feature RT end" = c(310, 610),
  check.names = FALSE
```

```
)

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

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

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

Description

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

Usage

```
next_plot(object)

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

Arguments

`object`	An instance of class `lcmsPlotClass`.

Value

An instance of class `lcmsPlotClass`.

Examples

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

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

```
  lp_legend(position = "bottom") +
  lp_labels(legend = "Sample")

p <- next_plot(p)
p
```

`open_cd_result_connection` *Open a connection to a Compound Discoverer results database*

Description

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

Usage

```
open_cd_result_connection(cd_result_path)
```

Arguments

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

Value

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

`parse_trace` *Parse a binary XIC trace from Compound Discoverer*

Description

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

Usage

```
parse_trace(data)
```

Arguments

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

Value

A `data.frame` with columns:

rt Retention time in seconds.

intensity Signal intensity.

If the trace cannot be parsed, `NULL` is returned.

plot_chromatogram *Plot chromatograms from one or more datasets*

Description

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

Usage

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

Arguments

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

Value

A ggplot object representing the chromatogram plot.

plot_data *Plot the specified LC-MS data*

Description

Plot the specified LC-MS data

Usage

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

Arguments

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

Value

A patchwork plot object representing the final plot.

`plot_intensity_map` *Plot an LC-MS intensity map*

Description

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

Usage

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

Arguments

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

Value

A ggplot object representing the 2D intensity map plot.

`plot_mass_trace` *Plot a mass trace*

Description

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

Usage

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

Arguments

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

Value

A ggplot object representing the mass trace plot.

---

plot_multiple_datasets

Plot multiple dataset types

Description

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

Usage

```
plot_multiple_datasets(datasets, obj, plot_config)
```

Arguments

datasets	A named list of data frames, where each element represents a dataset to be plotted. The names must match the supported dataset types defined in plot_config.
obj	An instance of class lcmsPlotClass.
plot_config	A named list defining the plotting functions to use for each dataset type. List names correspond to dataset types, and values are functions that return ggplot objects.

Value

A patchwork object combining all generated plots.

plot_multiple_faceted_datasets *Plot multiple faceted dataset types*

Description

`plot_multiple_faceted_datasets()` generates plots for multiple dataset types and arranges them into a faceted grid. Datasets are split according to the faceting variables defined in `obj@options$facets`, with each facet panel containing a vertically stacked set of dataset-specific plots.

Usage

```
plot_multiple_faceted_datasets(datasets, obj, plot_config)
```

Arguments

`datasets`	A named `list` of data frames, where each element represents a dataset to be plotted. The names must match the supported dataset types defined in `plot_config`.
`obj`	An instance of class `lcmsPlotClass`.
`plot_config`	A named `list` defining the plotting functions to use for each dataset type. List names correspond to dataset types, and values are functions that return ggplot objects.

Value

A `patchwork` object combining all generated plots.

plot_rt_diff *Plot an RT alignment difference plot*

Description

`plot_rt_diff()` generates a ggplot2 from a dataset containing the differences between raw and adjusted retention times.

Usage

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

Arguments

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

Value

A ggplot object representing the RT difference plot.

---

plot_single_dataset *Plot a single dataset type*

Description

plot_single_dataset() allows the plotting of a single dataset type (e.g., only chromatograms).

Usage

```
plot_single_dataset(datasets, obj, plot_config)
```

Arguments

datasets	A named list of data frames, where each element represents a dataset to be plotted. The names must match the supported dataset types defined in plot_config.
obj	An instance of class lcmsPlotClass.
plot_config	A list that defines the plot configuration. This list should use the supported dataset types as keys (e.g., chromatograms), with each value providing the corresponding function used to plot that dataset type.

Value

A patchwork object combining all generated plots. In this case the patchwork object contains a single ggplot2 object.

plot_spectrum *Plot spectra from one or more datasets*

Description

plot_spectrum() generates a ggplot2 MS spectra plot.

Usage

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

Arguments

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

Value

A ggplot object representing the RT difference plot.

plot_total_ion_current *Plot total ion current for one or more samples*

Description

plot_total_ion_current() generates a ggplot2 of the total ion current of the samples within the dataset.

Usage

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

Arguments

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

Value

A ggplot object representing the RT difference plot.

---

process_metadata *Process sample metadata*

Description

Construct a metadata data frame aligned with a set of sample paths. If a metadata file is provided, it is read from disk and combined with the sample paths. Otherwise, a minimal metadata table is created.

Usage

```
process_metadata(sample_paths, metadata_path = NULL)
```

Arguments

sample_paths	A character vector of sample file paths.
metadata_path	A character value indicating the path to a delimited metadata file with a header.

Value

A data.frame containing a sample_path column and any additional metadata.

remove_null_elements *Remove* NULL *elements from a list*

Description

Remove NULL elements from a list

Usage

```
remove_null_elements(lst)
```

Arguments

lst	A list from which NULL entries should be removed.

Value

A list containing only the non-NULL elements of lst.

run_matching_plot_variant *Selects and executes the appropriate plot variant*

Description

Determines which plotting variant is applicable for the given datasets and plotting options, then executes the first matching variant. Plot variants are evaluated in order, and the first whose condition evaluates to TRUE is used to generate the plot.

Usage

```
run_matching_plot_variant(datasets, obj)
```

Arguments

datasets	A list of data frames, each representing a dataset to be visualised.
obj	An instance of class lcmsPlotClass.

Value

A patchwork plot object generated by the selected plot variant.

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"),
  stringsAsFactors = FALSE
)

## Create dummy peaks
peaks <- data.frame(
  mz = c(100.1, 150.2),
  rt = c(300, 450),
  rtmin = c(290, 440),
  rtmax = c(310, 460),
  into = c(10000, 15000),
  maxo = c(2000, 2500),
  sample_index = c(1, 2)
)

## Create ExternalDataSource object
eds <- new(
  "ExternalDataSource",
  name = "Example data source",
  metadata = metadata,
  peaks = peaks
)

## Show summary
```

eds

---

show,lcmsPlotClass-method
Plot the lcmsPlotClass *object*

Description

Display an instance of lcmsPlotClass class to the selected device.

Usage

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

Arguments

object	An instance of class lcmsPlotClass.

Value

Invisible NULL

Examples

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

## Shows summary information as the plot has not been built yet
p <- lcmsPlot(raw_files)
p

## Shows the actual plot
p <- lcmsPlot(raw_files) +
  lp_chromatogram(aggregation_fun = "max") +
  lp_arrange(group_by = "sample_id") +
  lp_legend(position = "bottom") +
  lp_labels(legend = "Sample")

p
```

show,lcmsPlotDataContainer-method
Show a summary of an instance of class lcmsPlotDataContainer

Description

Show a summary of an instance of class lcmsPlotDataContainer

Usage

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

Arguments

object	An instance of class lcmsPlotDataContainer.

Value

Invisible NULL

Examples

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

data_obj <- new("lcmsPlotDataContainer",
    data_obj = raw_files,
    metadata = data.frame(),
    chromatograms = data.frame(),
    mass_traces = data.frame(),
    spectra = data.frame(),
    total_ion_current = data.frame(),
    intensity_maps = data.frame(),
    rt_diff = data.frame(),
    additional_metadata = data.frame(),
    detected_peaks = data.frame())
data_obj
```

Index