

# Package ‘lcmsPlot’

April 22, 2026

**Type** Package

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

**Version** 0.99.21

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

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

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

**VignetteBuilder** knitr

**URL** <https://github.com/computational-metabolomics/lcmsPlot>

**RoxygenNote** 7.3.3

**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** 6295896

**git\_last\_commit\_date** 2026-04-20

**Repository** Bioconductor 3.24

**Date/Publication** 2026-04-21

**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 . . . . .	4
+,lcmsPlotClass,function-method . . . . .	5
convert_rt_to_seconds . . . . .	5
create_bpc_tic . . . . .	6
create_chromatogram . . . . .	6
create_chromatograms . . . . .	7
create_data_container_from_obj . . . . .	8
create_intensity_map . . . . .	9
create_peak_density . . . . .	9
create_rt_diff . . . . .	10
create_spectra . . . . .	10
create_spectra_for_sample . . . . .	11
create_spectrum_from_closest_scan_to_rt . . . . .	11
create_spectrum_from_scan_index . . . . .	12
create_total_ion_current . . . . .	12
create_xcms_raw_list . . . . .	13
default_options . . . . .	14
detect_separator . . . . .	14
ExternalDataSource-class . . . . .	15
field . . . . .	15
get_adjusted_rts . . . . .	16
get_detected_peaks . . . . .	16
get_features . . . . .	17
get_feature_data . . . . .	18
get_grouped_peaks . . . . .	18
get_grouping_variables . . . . .	19
get_metadata . . . . .	20
get_mz_range . . . . .	23
get_workflow_input_files . . . . .	23
get_XCMSnExp_object_example . . . . .	24

get_xic_traces_from_compounds . . . . .	24
io_close_raw_data . . . . .	25
io_get_raw_data . . . . .	25
is_cd_result . . . . .	26
is_cd_results_path . . . . .	26
is_xcms_data . . . . .	27
is_xcms_processed_data . . . . .	27
iterate_plot_batches . . . . .	28
lcmsPlot . . . . .	29
lcmsPlotClass-class . . . . .	30
lcmsPlotDataContainer-class . . . . .	30
lp_arrange . . . . .	31
lp_chromatogram . . . . .	32
lp_compound_discoverer . . . . .	34
lp_facets . . . . .	35
lp_get_plot . . . . .	36
lp_grid . . . . .	37
lp_intensity_map . . . . .	38
lp_labels . . . . .	39
lp_layout . . . . .	40
lp_legend . . . . .	41
lp_mass_trace . . . . .	42
lp_peak_density . . . . .	42
lp_rt_diff_plot . . . . .	44
lp_rt_line . . . . .	44
lp_spectra . . . . .	45
lp_total_ion_current . . . . .	47
merge_by_index . . . . .	48
MsDialPeaksSource . . . . .	48
MsRawReader-class . . . . .	49
ms_chromatogram . . . . .	50
ms_close . . . . .	50
ms_header . . . . .	51
ms_peaks . . . . .	52
MZmineFeatureListsSource . . . . .	52
MzrReader-class . . . . .	53
next_plot . . . . .	54
open_cd_result_connection . . . . .	55
open_raw_reader . . . . .	55
parse_trace . . . . .	56
plot_chromatogram . . . . .	56
plot_data . . . . .	57
plot_intensity_map . . . . .	57
plot_mass_trace . . . . .	58
plot_multiple_datasets . . . . .	58
plot_multiple_faceted_datasets . . . . .	59
plot_peak_density . . . . .	59
plot_rt_diff . . . . .	60
plot_single_dataset . . . . .	61
plot_spectrum . . . . .	61
plot_total_ion_current . . . . .	62
process_metadata . . . . .	62

RawrrReader-class	63
remove_null_elements	63
run_matching_plot_variant	63
show,ExternalDataSource-method	64
show,lcmsPlotClass-method	65
show,lcmsPlotDataContainer-method	65
show,XcmsRawList-method	66
validate_data_frame	67
validate_object	67
XcmsRawList	68
XcmsRawList-class	68
XcmsRawReader-class	69
xcmsraw_to_readers	69

<b>Index</b>	<b>70</b>
--------------	-----------

---

lcmsPlot-package	<i>lcmsPlot: Comprehensive Liquid Chromatography-Mass Spectrometry (LC-MS) data visualisation package</i>
------------------	---

---

## 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 = "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")
```

---

convert\_rt\_to\_seconds *Convert retention times to seconds*


---

### Description

Convert retention time columns from minutes to seconds.

### Usage

```
convert_rt_to_seconds(peaks)
```

### Arguments

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

**Value**

peaks with retention time columns converted to seconds.

---

create_bpc_tic	<i>Create a base peak or total ion current chromatogram</i>
----------------	---

---

**Description**

Create a base peak or total ion current chromatogram

**Usage**

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

**Arguments**

raw_data	An instance of class MsRawReader.
aggregation_fun	A 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 tibble containing the chromatograms with columns rt and intensity.

---

create_chromatogram	<i>Create an extracted ion chromatogram</i>
---------------------	---

---

**Description**

Create an extracted ion chromatogram

**Usage**

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

**Arguments**

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

**Value**

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

---

create\_chromatograms *Create chromatogram data from a data object*

---

**Description**

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

**Usage**

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

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

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

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

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

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

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

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

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

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

**Arguments**

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

**Value**

A named list with elements chromatograms, mass\_traces, feature\_metadata, and detected\_peaks.

---

```
create_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 = "faahK0"),
  full.names = TRUE,
  recursive = TRUE)[1:5]

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

---

create\_intensity\_map    *Create an instance of class lcmsPlotDataContainer from an intensity map*

---

### Description

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

### Usage

```
create_intensity_map(obj, options)

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

### Arguments

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

### Value

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

---

create\_peak\_density    *Create an instance of class lcmsPlotDataContainer from a peak density dataset.*

---

### Description

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

### Usage

```
create_peak_density(obj, options)

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

### Arguments

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

**Value**

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

---

<code>create_rt_diff</code>	<i>Create an instance of class <code>lcmsPlotDataContainer</code> from an RT adjustment dataset</i>
-----------------------------	---

---

**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**

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

**Value**

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

---

<code>create_spectra</code>	<i>Create an instance of class <code>lcmsPlotDataContainer</code> from spectra</i>
-----------------------------	--

---

**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**

<code>obj</code>	An instance of class <code>lcmsPlotDataContainer</code> .
<code>options</code>	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**

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

**Value**

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

---

create\_spectrum\_from\_closest\_scan\_to\_rt

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

---

**Description**

Create a spectrum of the closest scan to the specified RT

**Usage**

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

**Arguments**

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

**Value**

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

---

create\_spectrum\_from\_scan\_index  
*Create a spectrum of the specified scan*

---

**Description**

Create a spectrum of the specified scan

**Usage**

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

**Arguments**

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

**Value**

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

---

create\_total\_ion\_current  
*Create an instance of class lcmsPlotDataContainer from total ion current (TIC) dataset*

---

**Description**

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

**Usage**

```
create_total_ion_current(obj, options)

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

**Arguments**

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

**Value**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Examples**

```
paths <- dir(
  system.file("cdf", package = "faahK0"),
  full.names = TRUE,
  recursive = TRUE
)[1:3]
xl <- create_xcms_raw_list(paths)
```

---

default_options	<i>Get the default options</i>
-----------------	--------------------------------

---

**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	<i>Detect field separator from file extension</i>
------------------	---

---

**Description**

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

**Usage**

```
detect_separator(path)
```

**Arguments**

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

**Value**

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

---

ExternalDataSource-class

*External data source wrapper*

---

### Description

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

### Slots

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

metadata A data.frame representing the sample metadata.

peaks A data.frame representing the exported peaks.

---

field

*Construct a field specification for data frame validation*

---

### Description

Construct a field specification for data frame validation

### Usage

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

### Arguments

name A character value giving the column name.

type A function used to check the column's type (e.g. `is.numeric`). If `NULL`, no type check is performed.

required A logical value indicating whether the column must be present. Defaults to `TRUE`.

### Value

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

---

get_adjusted_rts	<i>Retrieve raw and adjusted retention times from an xcms object</i>
------------------	--

---

### Description

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

### Usage

```
get_adjusted_rts(obj)
```

### Arguments

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

### Details

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

### Value

A tibble with one row per scan, containing:

**file\_index** Index of the originating raw data file.

**raw\_rt** Original (unadjusted) retention time.

**adj\_rt** Adjusted retention time after RT correction.

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

---

get_detected_peaks	<i>Get the detected peaks from the data object (e.g. XCMSnExp)</i>
--------------------	--

---

### Description

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

### Usage

```
get_detected_peaks(obj)
```

```
## S3 method for class 'character'
```

```
get_detected_peaks(obj)
```

```
## S3 method for class 'XCMSnExp'
```

```
get_detected_peaks(obj)
```

```
## S3 method for class 'MsExperiment'
```

```
get_detected_peaks(obj)

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

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

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

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

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

### 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 re-named to `sample_index`.

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

### Value

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

---

get_features	<i>Get the feature data (m/z and RT ranges) for a set of features</i>
--------------	---

---

### 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`).

---

<code>get_feature_data</code>	<i>Get a feature's m/z and RT ranges given different feature specifications</i>
-------------------------------	---

---

**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: <ul style="list-style-type: none"> <li>• m/z, rt (optional)</li> <li>• mzmin, mzmax, rtmin (optional), rtmax (optional)</li> </ul>
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`.

---

<code>get_grouped_peaks</code>	<i>Get the grouped peaks across samples (features) from the data object</i>
--------------------------------	---

---

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

---

get\_grouping\_variables

*Get the grouping variables for a plot*

---

**Description**

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

**Usage**

```
get_grouping_variables(opts)
```

**Arguments**

opts                    A list of plot options.

**Value**

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

---

`get_metadata`*Get the metadata associated with the input object*

---

### Description

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

### Usage

```
get_metadata(obj, sample_id_column, metadata)

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

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

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

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

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

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

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

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

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

### Arguments

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

### Details

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

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

Across all methods, the returned metadata is enriched with:

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

### Value

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

### Character vector input (character)

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

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

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

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

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

### XCMSnExp input

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

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

The returned metadata always includes:

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

### MsExperiment input

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

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

The returned metadata includes:

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

**MChromatograms and XChromatograms input**

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

`sample_id` is resolved in the following order:

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

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

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

The returned metadata always includes:

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

**XChromatogram input**

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

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

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

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

**XcmsRawList input**

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

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

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

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

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

**ExternalDataSource input**

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

The returned metadata always includes:

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

**DBIConnection input**

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

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

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

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

---

<code>get_mz_range</code>	<i>Compute an m/z range given a ppm tolerance</i>
---------------------------	---

---

**Description**

Compute an m/z range given a ppm tolerance

**Usage**

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

**Arguments**

<code>mz</code>	A numeric value indicating the m/z value to calculate the range for.
<code>ppm</code>	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.

---

<code>get_workflow_input_files</code>	<i>Retrieve workflow input files from a Compound Discoverer database</i>
---------------------------------------	--

---

**Description**

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

**Usage**

```
get_workflow_input_files(conn)
```

**Arguments**

<code>conn</code>	A <code>DBIConnection</code> to a Compound Discoverer results database.
-------------------	---

**Value**

A tibble containing information about the workflow input files.

---

```
get_XCMSnExp_object_example
```

*Get an XCMSnExp example object from the faahKO dataset*

---

### Description

Get an XCMSnExp example object from the faahKO dataset

### Usage

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

### Arguments

`indices` A numeric vector of sample indices to select from the faahKO CDF files. Defaults to the first three samples.

`should_group_peaks` A logical value indicating whether to group the detected peaks.

### Value

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

### Examples

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

---

```
get_xic_traces_from_compounds
```

*Extract XIC traces associated with selected compounds*

---

### Description

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

### Usage

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

### Arguments

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

`compounds_query_str` A character value giving a filtering expression evaluated on the resulting compound table (e.g. using compound name, formula, retention time, or m/z).

**Details**

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

**Value**

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

---

io_close_raw_data	<i>Close open MsRawReader connections</i>
-------------------	---

---

**Description**

Close open MsRawReader connections

**Usage**

```
io_close_raw_data(raw_data)
```

**Arguments**

raw\_data      A list of MsRawReader objects (as returned by io\_get\_raw\_data()).

**Value**

NULL

---

io_get_raw_data	<i>Get MsRawReader objects for a set of sample paths</i>
-----------------	--

---

**Description**

Get MsRawReader objects for a set of sample paths

**Usage**

```
io_get_raw_data(sample_paths)
```

**Arguments**

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

**Value**

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

---

is_cd_result	<i>Check whether an object is a Compound Discoverer database connection</i>
--------------	---

---

**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	<i>Check whether a path corresponds to a Compound Discoverer results file</i>
--------------------	---

---

**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	<i>Check whether an object is from XCMS</i>
--------------	---

---

**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	<i>Check whether an object is an XCMS data container</i>
------------------------	--

---

**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 = "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	<i>Create an lcmsPlotClass object</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. 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

<code>dataset</code>	An object of type <code>XCMSnExp</code> , <code>MsExperiment</code> , <code>MZmineSource</code> , or character. If a character vector is supplied, it will be interpreted as a list of mzML paths.
<code>sample_id_column</code>	A character value indicating which column should be used as the sample ID. By default it is "sample_id".
<code>metadata</code>	A data.frame containing the samples metadata in case it is not provided in the dataset object.
<code>batch_size</code>	A numeric value indicating the number of samples per batch. This parameter is necessary when plotting multiple batches using the <code>iterate_plot_batches</code> or <code>next_plot</code> functions.
<code>BPPARAM</code>	A <code>BiocParallelParam</code> object for enabling parallelism. See <a href="#">BiocParallelParam</a> 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`, `MChromatograms`, `XChromatograms`, `XChromatogram`, `XcmsRawList`, or character representing mzML paths.  
`metadata` A data.frame containing the sample metadata.  
`chromatograms` A data.frame containing the chromatograms.  
`mass_traces` A data.frame containing the mass traces.  
`spectra` A data.frame containing the spectra.  
`peak_density` A data.frame containing peak density curve data and optional feature-group rectangles, as produced by `lp_peak_density()`.

total\_ion\_current A data.frame containing the total ion current.

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

rt\_diff A data.frame containing the raw and adjusted RT values.

feature\_metadata A data.frame containing feature/compound annotations attached to datasets through a column called feature\_metadata\_id.

detected\_peaks A data.frame containing the detected peaks from an XCMSnExp 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 overlaid without specifying a grouping factor
p <- lcmsPlot(raw_files) +
  lp_chromatogram(aggregation_fun = "max")
p

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

---

lp\_chromatogram      *Define the chromatograms to plot*

---

### Description

The lp\_chromatogram function allows the generation of different types of chromatograms.

### Usage

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

### Arguments

features	Specifies which features to generate the chromatogram for. This can be either: a matrix with columns mz and rt (optional); a matrix with columns mzmin, mzmax, rtmin (optional), rtmax (optional); a data.frame with columns sample_id, mz and rt (optional); a data.frame with columns sample_id, mzmin, mzmax, rtmin (optional), rtmax (optional); a character vector representing the grouped peaks (feature) names as returned by xcms::groupnames - requires the data to be an XCMSnExp or MsExperiment object with grouped peaks.
sample_ids	A character vector specifying the sample IDs to include in the plot. If NULL, the function uses the sample IDs specified in the lcmsPlot object.
ppm	A numeric value specifying the mass accuracy (in ppm) used when generating chromatograms. Ignored when the features parameter specifies both mzmin and mzmax.
rt_tol	A numeric value specifying the RT tolerance used when generating chromatograms. Ignored when the features parameter specifies both rtmin and rtmax.
line_type	A character value specifying the line type (from ggplot2). One of: "solid", "dashed", "dotted", "dotdash", "longdash", "twodash".
highlight_peaks	A logical value indicating whether to highlight the detected peaks; the input data must be an XCMSnExp or MsExperiment object.

highlight_peaks_color	A character value indicating the color of the highlighted peaks.
highlight_peaks_mode	A character value indicating how the peaks should be highlighted. One of "polygon" (fills the area under the curve), "rectangle" (draws a bounding box from rtmin to rtmax up to the peak apex), or "point" (marks the peak apex with a point). Defaults to "polygon".
highlight_peaks_factor	A character value indicating the factor from the metadata that determines the color. By default it colors by sample_id.
aggregation_fun	A character value indicating which aggregation function to use for the spectra intensities; one of max or sum. Only applicable to summary chromatograms.
rt_type	A character value indicating what type of RT to use for the chromatograms. One of uncorrected (default), corrected, or both; the input data must be an XCMSnExp or MsExperiment object. If both is chosen, this will give access to a metadata column called rt_adjusted that can be used to differentiate the two RT types (e.g., through faceting).
rt_unit	A character value indicating the unit to use for the RT axis; one of "minute" or "second".
intensity_unit	A character value indicating the unit to use for the intensity axis; one of "absolute" or "relative".
fill_gaps	A logical value indicating whether to fill gaps in RT with 0 intensity.
na.rm	A logical value. When TRUE, data points whose intensity is NA are removed before plotting. Defaults to FALSE.
highlight_apices	A logical value indicating whether to highlight apices with the corresponding RT values in a chromatogram.

## Value

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

## Summary chromatograms

In this type of chromatogram, the intensities of the spectra from each scan in an LC-MS dataset are either summed to produce the total ion current (TIC) chromatogram or the most intense peak is selected to produce the base peak chromatogram (BPC). To create such chromatograms do not specify the `features` parameter as that will create the chromatograms for the selected features. In this context, the main parameter is `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 specifying 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]<sup>+</sup>, [M-H]<sup>-</sup>).

**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 overlaid
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 = "faahKO"),
  full.names = TRUE,
  recursive = TRUE
)[1:4]

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

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

## Arrange chromatograms in a grid split by experimental factors
```

```
## Rows correspond to `factor1` and columns correspond to `factor2`
p <- p + lp_grid(rows = "factor1", cols = "factor2")
p
```

---

lp\_intensity\_map      *Define a 2D intensity map*

---

## Description

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

## Usage

```
lp_intensity_map(
  mz_range,
  rt_range,
  sample_ids = NULL,
  density = FALSE,
  x_dim = "rt",
  y_dim = "mz",
  fill_scale = NULL
)
```

## Arguments

<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 NULL, the function uses the sample IDs specified in the <code>lcmsPlot</code> object.
<code>density</code>	A logical value indicating whether to show a density plot.
<code>x_dim</code>	A character value indicating which dimension to place on the x-axis. One of "rt" (default) or "mz".
<code>y_dim</code>	A character value indicating which dimension to place on the y-axis. One of "mz" (default) or "rt".
<code>fill_scale</code>	A <code>ggplot2</code> 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.

## 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

<code>title</code>	A character value indicating the plot title.
<code>legend</code>	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 <a href="https://patchwork.data-imaginist.com/reference/wrap_plots.html">https://patchwork.data-imaginist.com/reference/wrap_plots.html</a>
--------	--

## 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 = "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	<i>Define the mass trace to plot</i>
---------------	--------------------------------------

---

### 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 = "faahK0"),
  full.names = TRUE,
  recursive = TRUE)[1:5]

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

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

p
```

---

lp_peak_density	<i>Plot the peak density for one or more <math>m/z</math> 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"
)
```

## Arguments

<code>features</code>	A matrix or data.frame 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.
<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.
<code>sample_groups</code>	A vector of length equal to the number of samples assigning each sample to a group (as in <code>PeakDensityParam</code> ). Defaults to <code>NULL</code> , which treats all samples as a single group.
<code>max_features</code>	An integer specifying the maximum number of feature group rectangles to draw per m/z bin. Defaults to 50L.
<code>rt_unit</code>	A character value indicating the unit for the RT axis; one of "second" (default) or "minute".

## Value

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

**Examples**

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

---

lp_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	<i>Define a vertical line on a retention time value</i>
------------	---

---

**Description**

Define a vertical line on a retention time value

**Usage**

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

## Arguments

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

## Value

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

## Examples

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

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

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

---

lp\_spectra

*Define the spectra to plot*

---

## Description

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

## Usage

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

```

    match_target_index = NULL,
    peak_label_size = 3,
    intensity_breaks_by = 20,
    auto_facet = TRUE
  )

```

### Arguments

sample_ids	A character vector specifying the sample IDs to include in the plot. If NULL, the function uses the sample IDs specified in the <code>lcmsPlot</code> object or the <code>lp_chromatogram</code> function.
mode	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>mode</code> is not applicable to standalone spectra.
ms_level	The MS level to consider for the scan.
rt	When <code>mode = "closest"</code> , the RT to consider.
scan_index	The exact scan index to consider for extracting a spectrum. <code>scan_index</code> and <code>mode</code> are mutually exclusive.
interval	When <code>mode = "across_peak."</code> The RT interval to consider.
spectral_match_db	The database containing reference spectra used for matching and comparison with the input spectra.
match_target_index	The index, ranked by descending match score, identifying which reference spectrum to display in the mirror plot.
peak_label_size	A numeric value controlling the font size of m/z labels annotated on spectral peaks. Defaults to 3.
intensity_breaks_by	A numeric value specifying the step size (in percent) between y-axis intensity breaks. Defaults to 20.
auto_facet	A logical value. When TRUE (default), a facet is automatically added to separate spectra from different samples or scans. Set to FALSE to suppress automatic faceting.

### Value

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

### Spectra associated with chromatograms

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

## Standalone spectra

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

## Examples

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

---

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	<i>Merge two data frames using a row-index match</i>
----------------	--

---

**Description**

Merge two data frames using a row-index match

**Usage**

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

**Arguments**

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

**Value**

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

---

MsDialPeaksSource	<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 data.frame is created from sample_paths.

**Value**

An object of class ExternalDataSource.

## Examples

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

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

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

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

msdial_peaks_2 <- tibble::tibble(
  "Precursor m/z" = c(150.3, 250.4),
  "RT (min)" = c(6.0, 12.0),
  "Area" = c(15000, 25000),
  "Height" = c(600, 900),
  "RT left(min)" = c(5.8, 11.8),
  "RT right (min)" = c(6.2, 12.2)
)

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

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

---

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

---

## Description

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

---

ms_chromatogram	<i>Extract an extracted ion chromatogram (XIC) using the native backend</i>
-----------------	---

---

### Description

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

### Usage

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

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

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

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

### Arguments

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

### Value

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

---

ms_close	<i>Close a raw MS file reader connection</i>
----------	--

---

### Description

Close a raw MS file reader connection

### Usage

```
ms_close(reader)

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

## S4 method for signature 'RawrrReader'
```

```
ms_close(reader)

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

**Arguments**

reader            An instance of MsRawReader.

**Value**

invisible(NULL)

---

ms_header	<i>Get scan header information from a raw MS file reader</i>
-----------	--

---

**Description**

Get scan header information from a raw MS file reader

**Usage**

```
ms_header(reader)

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

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

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

**Arguments**

reader            An instance of MsRawReader.

**Value**

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

---

<code>ms_peaks</code>	<i>Get peak matrices for selected scans from a raw MS file reader</i>
-----------------------	---

---

**Description**

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

**Usage**

```
ms_peaks(reader, scans)

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

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

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

**Arguments**

<code>reader</code>	An instance of MsRawReader.
<code>scans</code>	An integer vector of scan sequence numbers to retrieve.

**Value**

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

---

<code>MZmineFeatureListsSource</code>	<i>Create an MZmine data source from feature lists</i>
---------------------------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

An object of class `ExternalDataSource`.

**Examples**

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

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

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

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

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

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

---

MzrReader-class

*An mzR-backed raw MS file reader*


---

**Description**

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

**Slots**

connection An mzR connection object.

---

next_plot	<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
```

---

open\_cd\_result\_connection  
*Open a connection to a Compound Discoverer results database*

---

**Description**

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

**Usage**

```
open_cd_result_connection(cd_result_path)
```

**Arguments**

cd\_result\_path A character value giving the path to a Compound Discoverer results file.

**Value**

A DBIConnection object connected to the Compound Discoverer SQLite database.

---

open\_raw\_reader      *Open a raw MS file as an MsRawReader*

---

**Description**

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

**Usage**

```
open_raw_reader(path)
```

**Arguments**

path                  A character value giving the file path.

**Value**

An instance of MsRawReader (either MzrReader or RawrrReader).

---

parse_trace	<i>Parse a binary XIC trace from Compound Discoverer</i>
-------------	--

---

**Description**

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

**Usage**

```
parse_trace(data)
```

**Arguments**

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

**Value**

A tibble with columns:

**rt** Retention time in seconds.

**intensity** Signal intensity.

If the trace cannot be parsed, NULL is returned.

---

plot_chromatogram	<i>Plot chromatograms from one or more datasets</i>
-------------------	---

---

**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	<i>Plot the specified LC-MS data</i>
-----------	--------------------------------------

---

**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	<i>Plot an LC-MS intensity map</i>
--------------------	------------------------------------

---

**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\_peak\_density

*Plot a peak density plot*

---

**Description**

plot\_peak\_density() generates a ggplot2 peak density plot that mirrors xcms::plotChromPeakDensity(). For each m/z feature:

- The x axis shows retention time.
- The y axis shows sample indices (1 to n), positioned within the density range so that sample rows are evenly spaced from 0 to max(density).
- Detected peaks are plotted as coloured points at each sample's y position.
- The kernel density estimate of peak apex RTs is drawn as a line.
- When feature group simulation is enabled, semi-transparent rectangles mark feature groups that pass the min\_fraction / min\_samples threshold.

**Usage**

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

**Arguments**

`datasets` A named list of data frames. The used key is `peak_density`.

`supporting_datasets` A list of supporting data frames. The used key is `detected_peaks`, which provides per-sample peak positions.

`options` A list of plot options.

`single` A logical value indicating whether to treat the plot as a single dataset variant. Default is `FALSE`.

**Value**

A ggplot object representing the peak density plot.

---

<code>plot_rt_diff</code>	<i>Plot an RT alignment difference plot</i>
---------------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

`datasets` A named list of data frames containing the primary datasets to plot. For an RT difference plot the used key is `rt_diff`.

`supporting_datasets` A list of supporting data frames, such as detected peaks, used for highlighting features.

`options` A list of plot options, controlling units, faceting, highlighting, and other visual parameters.

`single` A logical value that indicates whether it should treat the plot as a single dataset variant, which can affect faceting and layout behavior. Default is `FALSE`.

**Value**

A ggplot object representing the RT difference plot.

---

plot\_single\_dataset     *Plot a single dataset type*

---

**Description**

plot\_single\_dataset() allows the plotting of a single dataset type (e.g., only chromatograms).

**Usage**

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

**Arguments**

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

**Value**

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

---

plot\_spectrum     *Plot spectra from one or more datasets*

---

**Description**

plot\_spectrum() generates a ggplot2 MS spectra plot.

**Usage**

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

**Arguments**

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

**Value**

A ggplot object representing the RT difference plot.

---

plot\_total\_ion\_current

*Plot total ion current for one or more samples*

---

### Description

plot\_total\_ion\_current() generates a ggplot2 of the total ion current of the samples within the dataset.

### Usage

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

### Arguments

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

### Value

A ggplot object representing the RT difference plot.

---

process\_metadata

*Process sample metadata*

---

### Description

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

### Usage

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

### Arguments

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

### Value

A tibble containing a sample\_path column and any additional metadata.

---

RawrrReader-class     *A rawrr-backed raw MS file reader*

---

**Description**

Stores the path to a ThermoFisher .raw file. rawrr does not maintain persistent connections, so the path is re-used for each read operation.

**Slots**

path A character value giving the path to the .raw file.

---

remove\_null\_elements     *Remove NULL elements from a list*

---

**Description**

Remove NULL elements from a list

**Usage**

```
remove_null_elements(lst)
```

**Arguments**

lst                     A list from which NULL entries should be removed.

**Value**

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

---

run\_matching\_plot\_variant  
                               *Selects and executes the appropriate plot variant*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

A patchwork plot object generated by the selected plot variant.

---

show,ExternalDataSource-method

*Show a summary of an instance of class ExternalDataSource*

---

**Description**

Show a summary of an instance of class ExternalDataSource

**Usage**

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

**Arguments**

object            An instance of class ExternalDataSource.

**Value**

Invisible NULL

**Examples**

```
## Create dummy metadata  
metadata <- data.frame(  
  sample_id = c("S1", "S2"),  
  sample_path = c("sample1.mzML", "sample2.mzML")  
)  
  
## Create dummy peaks  
peaks <- data.frame(  
  mz = c(100.1, 150.2),  
  rt = c(300, 450),  
  rtmin = c(290, 440),  
  rtmax = c(310, 460),  
  into = c(10000, 15000),  
  maxo = c(2000, 2500),  
  sample_index = c(1, 2)  
)  
  
## Create ExternalDataSource object  
eds <- new(  
  "ExternalDataSource",  
  name = "Example data source",  
  metadata = metadata,  
  peaks = peaks  
)  
  
## Show summary  
eds
```

---

show,lcmsPlotClass-method

*Plot the lcmsPlotClass object*

---

### Description

Display an instance of lcmsPlotClass class to the selected device.

### Usage

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

### Arguments

object            An instance of class lcmsPlotClass.

### Value

Invisible NULL

### Examples

```
raw_files <- dir(  
  system.file("cdf", package = "faahKO"),  
  full.names = TRUE,  
  recursive = TRUE)[1:5]  
  
## Shows summary information as the plot has not been built yet  
p <- lcmsPlot(raw_files)  
p  
  
## Shows the actual plot  
p <- lcmsPlot(raw_files) +  
  lp_chromatogram(aggregation_fun = "max") +  
  lp_arrange(group_by = "sample_id") +  
  lp_legend(position = "bottom") +  
  lp_labels(legend = "Sample")  
  
p
```

---

show,lcmsPlotDataContainer-method

*Show a summary of an instance of class lcmsPlotDataContainer*

---

### Description

Show a summary of an instance of class lcmsPlotDataContainer

**Usage**

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

**Arguments**

object            An instance of class lcmsPlotDataContainer.

**Value**

Invisible NULL

**Examples**

```
raw_files <- dir(  
  system.file("cdf", package = "faahK0"),  
  full.names = TRUE,  
  recursive = TRUE)[1:5]  
  
data_obj <- new("lcmsPlotDataContainer",  
  data_obj = raw_files,  
  metadata = tibble::tibble(),  
  chromatograms = tibble::tibble(),  
  mass_traces = tibble::tibble(),  
  spectra = tibble::tibble(),  
  peak_density = tibble::tibble(),  
  total_ion_current = tibble::tibble(),  
  intensity_maps = tibble::tibble(),  
  rt_diff = tibble::tibble(),  
  feature_metadata = tibble::tibble(),  
  detected_peaks = tibble::tibble())  
data_obj
```

---

show,XcmsRawList-method

*Show a summary of an XcmsRawList object*

---

**Description**

Show a summary of an XcmsRawList object

**Usage**

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

**Arguments**

object            An instance of XcmsRawList.

**Value**

Invisible NULL.

---

validate\_data\_frame     *Validate a data frame against a list of field specifications*

---

### Description

Returns TRUE on success or a character string describing the first failure. NULL and empty data frames always pass.

### Usage

```
validate_data_frame(df, fields, exact = FALSE)
```

### Arguments

df	A data.frame or tibble to validate.
fields	A list of field specifications created with field().
exact	A logical value. If TRUE, the column names of df must be identical to the names in fields (same set, same order). Defaults to FALSE.

### Value

TRUE if validation passes, or a character string describing the first failure.

---

validate\_object     *Validate the data frame slots of an S4 object*

---

### Description

Iterates over a named list of validator functions, applying each to the corresponding slot of object.

### Usage

```
validate_object(object, validators)
```

### Arguments

object	An S4 object whose slots are to be validated.
validators	A named list of functions, where each name matches a slot of object and each function accepts a data.frame/tibble and returns TRUE or a character error string (as produced by validate_data_frame()).

### Value

TRUE if all validators pass, or a character string describing the first failure in the form "@slot\_name: <message>".

---

XcmsRawList	<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.

---

XcmsRawReader-class    *An xcmsRaw-backed raw MS file reader*

---

**Description**

Wraps an in-memory xcmsRaw object. No file connection is opened or closed.

**Slots**

obj    An xcmsRaw object.

---

xcmsraw\_to\_readers    *Convert an XcmsRawList to a named list of XcmsRawReaders*

---

**Description**

Each element is keyed by the file path stored in @filepath of the corresponding xcmsRaw object.

**Usage**

```
xcmsraw_to_readers(xcmsraw_list)
```

**Arguments**

xcmsraw\_list    An XcmsRawList object.

**Value**

A named list of XcmsRawReader objects.

# Index

## \* internal

convert\_rt\_to\_seconds, 5  
create\_bpc\_tic, 6  
create\_chromatogram, 6  
create\_chromatograms, 7  
create\_data\_container\_from\_obj, 8  
create\_intensity\_map, 9  
create\_peak\_density, 9  
create\_rt\_diff, 10  
create\_spectra, 10  
create\_spectra\_for\_sample, 11  
create\_spectrum\_from\_closest\_scan\_to\_rt,  
11  
create\_spectrum\_from\_scan\_index,  
12  
create\_total\_ion\_current, 12  
default\_options, 14  
detect\_separator, 14  
field, 15  
get\_adjusted\_rts, 16  
get\_detected\_peaks, 16  
get\_feature\_data, 18  
get\_features, 17  
get\_grouped\_peaks, 18  
get\_grouping\_variables, 19  
get\_metadata, 20  
get\_mz\_range, 23  
get\_workflow\_input\_files, 23  
get\_xic\_traces\_from\_compounds, 24  
io\_close\_raw\_data, 25  
io\_get\_raw\_data, 25  
is\_cd\_result, 26  
is\_cd\_results\_path, 26  
is\_xcms\_data, 27  
is\_xcms\_processed\_data, 27  
merge\_by\_index, 48  
ms\_chromatogram, 50  
ms\_close, 50  
ms\_header, 51  
ms\_peaks, 52  
MsRawReader-class, 49  
MzrReader-class, 53  
open\_cd\_result\_connection, 55

open\_raw\_reader, 55  
parse\_trace, 56  
plot\_chromatogram, 56  
plot\_data, 57  
plot\_intensity\_map, 57  
plot\_mass\_trace, 58  
plot\_multiple\_datasets, 58  
plot\_multiple\_faceted\_datasets, 59  
plot\_peak\_density, 59  
plot\_rt\_diff, 60  
plot\_single\_dataset, 61  
plot\_spectrum, 61  
plot\_total\_ion\_current, 62  
process\_metadata, 62  
RawrrReader-class, 63  
remove\_null\_elements, 63  
run\_matching\_plot\_variant, 63  
validate\_data\_frame, 67  
validate\_object, 67  
xcmsraw\_to\_readers, 69  
XcmsRawReader-class, 69  
+,lcmsPlotClass, function-method, 5

## BiocParallelParam, 29

convert\_rt\_to\_seconds, 5  
create\_bpc\_tic, 6  
create\_chromatogram, 6  
create\_chromatograms, 7  
create\_chromatograms, ANY, data.frame, list, ANY-method  
(create\_chromatograms), 7  
create\_chromatograms, ANY, data.frame, list, character-meth  
(create\_chromatograms), 7  
create\_chromatograms, ANY, data.frame, list, NULL-method  
(create\_chromatograms), 7  
create\_chromatograms, DBIConnection, data.frame, list, NULL  
(create\_chromatograms), 7  
create\_chromatograms, MChromatograms, data.frame, list, NULL  
(create\_chromatograms), 7  
create\_chromatograms, XChromatogram, data.frame, list, NULL  
(create\_chromatograms), 7  
create\_chromatograms, XChromatograms, data.frame, list, NULL  
(create\_chromatograms), 7

- create\_chromatograms, XcmsRawList, data.frame, list-method (create\_chromatograms), 7
- create\_chromatograms, XcmsRawList, data.frame, list-method (create\_chromatograms), 7
- create\_data\_container\_from\_obj, 8
- create\_intensity\_map, 9
- create\_intensity\_map, lcmsPlotDataContainer, list-method (create\_intensity\_map), 9
- create\_peak\_density, 9
- create\_peak\_density, lcmsPlotDataContainer, list-method (create\_peak\_density), 9
- create\_rt\_diff, 10
- create\_rt\_diff, lcmsPlotDataContainer, list-method (create\_rt\_diff), 10
- create\_spectra, 10
- create\_spectra, lcmsPlotDataContainer, list-method (create\_spectra), 10
- create\_spectra\_for\_sample, 11
- create\_spectrum\_from\_closest\_scan\_to\_rt, 11
- create\_spectrum\_from\_scan\_index, 12
- create\_total\_ion\_current, 12
- create\_total\_ion\_current, lcmsPlotDataContainer, list-method (create\_total\_ion\_current), 12
- create\_xcms\_raw\_list, 13
- default\_options, 14
- detect\_separator, 14
- ExternalDataSource-class, 15
- field, 15
- get\_adjusted\_rts, 16
- get\_detected\_peaks, 16
- get\_feature\_data, 18
- get\_features, 17
- get\_grouped\_peaks, 18
- get\_grouping\_variables, 19
- get\_metadata, 20
- get\_mz\_range, 23
- get\_workflow\_input\_files, 23
- get\_XCMSnExp\_object\_example, 24
- get\_xic\_traces\_from\_compounds, 24
- io\_close\_raw\_data, 25
- io\_get\_raw\_data, 25
- is\_cd\_result, 26
- is\_cd\_results\_path, 26
- is\_xcms\_data, 27
- is\_xcms\_processed\_data, 27
- iterate\_plot\_batches, 28
- iterate\_plot\_batches, lcmsPlotClass, function-method (iterate\_plot\_batches), 28
- list-method, 29
- lcmsPlot-package, 4
- lcmsPlot-class, 30
- lcmsPlotDataContainer-class, 30
- lp\_arrange, 31
- lp\_chromatogram, 32
- lp\_compound\_discoverer, 34
- lp\_facets, 35
- lp\_get\_plot, 36
- lp\_intensity\_map, 38
- lp\_labels, 39
- lp\_layout, 40
- lp\_legend, 41
- lp\_mass\_trace, 42
- lp\_peak\_density, 42
- lp\_rt\_diff\_plot, 44
- lp\_rt\_line, 44
- lp\_spectra, 45
- lp\_total\_ion\_current, 47
- merge\_by\_index, 48
- ms\_chromatogram, 50
- ms\_chromatogram, MzrReader-method (ms\_chromatogram), 50
- ms\_chromatogram, RawrrReader-method (ms\_chromatogram), 50
- ms\_chromatogram, XcmsRawReader-method (ms\_chromatogram), 50
- ms\_close, 50
- ms\_close, MzrReader-method (ms\_close), 50
- ms\_close, RawrrReader-method (ms\_close), 50
- ms\_close, XcmsRawReader-method (ms\_close), 50
- ms\_header, 51
- ms\_header, MzrReader-method (ms\_header), 51
- ms\_header, RawrrReader-method (ms\_header), 51
- ms\_header, XcmsRawReader-method (ms\_header), 51
- ms\_peaks, 52
- ms\_peaks, MzrReader-method (ms\_peaks), 52
- ms\_peaks, RawrrReader-method (ms\_peaks), 52
- ms\_peaks, XcmsRawReader-method (ms\_peaks), 52
- MsDialPeaksSource, 48
- MsRawReader-class, 49
- MzrReader-class, 53
- MzrReader-class, 53

`next_plot`, [54](#)  
`next_plot`, `lcmsPlotClass`-method  
    (`next_plot`), [54](#)

`open_cd_result_connection`, [55](#)  
`open_raw_reader`, [55](#)

`parse_trace`, [56](#)  
`plot_chromatogram`, [56](#)  
`plot_data`, [57](#)  
`plot_intensity_map`, [57](#)  
`plot_mass_trace`, [58](#)  
`plot_multiple_datasets`, [58](#)  
`plot_multiple_faceted_datasets`, [59](#)  
`plot_peak_density`, [59](#)  
`plot_rt_diff`, [60](#)  
`plot_single_dataset`, [61](#)  
`plot_spectrum`, [61](#)  
`plot_total_ion_current`, [62](#)  
`process_metadata`, [62](#)

`RawrrReader`-class, [63](#)  
`remove_null_elements`, [63](#)  
`run_matching_plot_variant`, [63](#)

`show`, `ExternalDataSource`-method, [64](#)  
`show`, `lcmsPlotClass`-method, [65](#)  
`show`, `lcmsPlotDataContainer`-method, [65](#)  
`show`, `XcmsRawList`-method, [66](#)  
`stats::density()`, [43](#)

`validate_data_frame`, [67](#)  
`validate_object`, [67](#)

`xcmsraw_to_readers`, [69](#)  
`XcmsRawList`, [68](#)  
`XcmsRawList`-class, [68](#)  
`XcmsRawReader`-class, [69](#)