

Package: mrmhub (via r-universe)

October 12, 2026

Title Small Molecule Mass Spectrometry Data Processing, Quality Control, and Reporting

Version 1.0.1

Description MRMhub-QUANT, the quantitation and reporting module of the MRMhub toolchain, is a tool for processing, quality control, and visualization of targeted metabolomics and lipidomics mass spectrometry data.

License AGPL-3

URL <https://github.com/SLINGhub/MRMhub>,
<https://slinghub.github.io/MRMhub/quant/>

BugReports <https://github.com/SLINGhub/MRMhub/issues>

Depends R (>= 4.1)

Imports assertr, methods, cli, dplyr (>= 1.1.0), tibble, forcats, fs, glue, lifecycle, lubridate, purrr (>= 1.1.0), readr, rlang, scales, stringr, tidyr, openxlsx2, ggh4x, ggplot2 (>= 3.5.0), ggrepel, ggnewscale

Suggests enviPat, qpdf, pdftools, ggvenn, patchwork, ggbeeswarm, ggpmisc, covr, ragg, svglite, fansi, lancer, rgoslin, knitr, ggExtra, MASS, mgcv, ranger, rmarkdown, sva, bibtex, here, testthat (>= 3.0.0), vdiff, withr, xml2, mirai, carrier, shiny, bslib, shinytest2, rmzTabM, DT, SummarizedExperiment, S4Vectors, lipidr

Remotes SLINGhub/lancer, github::lifs-tools/rmzTabM

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/markdown TRUE

LazyData true

LazyDataCompression xz

NeedsCompilation no

Collate 'build-workflow.R' 'calc-calibrations.R' 'calc-eda.R'
 'calc-istd-normalization.R' 'calc-ref-normalization.R'
 'calc-response.R' 'check-setup.R' 'mrmhub-global-definitions.R'
 'classes.R' 'correct-batch.R' 'correct-drift-batch.R'
 'correct-isotope.R' 'data-example.R' 'data-export.R'
 'data-import.R' 'data-managment.R' 'data-mztab.R'
 'data-outlier.R' 'data-summarize.R'
 'data-summarizedexperiment.R' 'datasets.R' 'functions-chem.R'
 'functions-math.R' 'helper.R' 'isotope-derive.R'
 'isotope-licar.R' 'lipidomics.R' 'metadata-import.R'
 'mrmhub-cli.R' 'mrmhub-package.R' 'plot-defaults.R'
 'plot-featureprofile.R' 'plot-qc-correlations.R'
 'plot-qc-matrixeffects.R' 'plot-runscluster.R' 'plot-save.R'
 'plots-calibcurves.R' 'plots-qc-cv.R' 'plots-qc-filtering.R'
 'plots-qc-identification.R' 'plots-qc-interference.R'
 'plots-qc-pca.R' 'plots-qc-responsecurves.R'
 'plots-qc-runorder.R' 'qc-filtering.R' 'tibble-classes.R'
 'utils-plot-style.R' 'utils.R'

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libx11-dev

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

Date/Publication 2026-10-12 02:44:51 UTC

RemoteUrl <https://github.com/SLINGhub/MRMhub>

RemoteRef HEAD

RemoteSha 46fd887f81ba7a0df73b00e8d8f138e6cdabc395

Contents

\$,MRMhubExperiment-method	5
add_metadata	5
build_workflow	6
calc_average_molweight	7
calc_calibration_results	8
calc_isotopic_interferences	10
calc_qc_metrics	12
calibrate_by_reference	15
check_setup	17
correct_batch_centering	18
correct_batch_combat	19
correct_batch_serrf	21
correct_custom_interferences	23
correct_drift	24
correct_drift_cubicspline	26
correct_drift_gam	29
correct_drift_gaussiankernel	31
correct_drift_loess	33

correct_interference_manual	36
correct_isotopic_interferences	37
cv	38
cv_log	39
data_load_example	39
data_sum_features	40
detect_outlier_pca	41
exclude_analyses	42
exclude_features	43
filter_features_qc	43
fun_cspline	47
fun_gam_smooth	48
fun_gauss.kernel.smooth	49
fun_loess	49
generate_workflow_qmd	50
get_analysis_end	51
get_analysis_start	52
get_analysis_breaks	52
get_analysis_count	53
get_analysis_duration	53
get_analyticaldata	54
get_batch_boundaries	54
get_calibration_metrics	55
get_feature_count	56
get_featurelist	57
get_iqr_tails	58
get_mad_tails	58
get_outlier_bounds	59
get_qc_bias_variability	60
get_response_curve_stats	62
get_runtime_median	63
import_data_csv_long	63
import_data_csv_wide	66
import_data_masshunter	68
import_data_mrmhub	69
import_data_mztab	70
import_data_skyline	72
import_metadata_analyses	73
import_metadata_features	75
import_metadata_from_data	76
import_metadata_istds	77
import_metadata_msorganiser	78
import_metadata_qcconcentrations	79
import_metadata_responsecurves	80
licar_pattern_choices	81
lipidomics_dataset	81
mrmhub_enable_cli_color	82
mrmhub_get_plot_defaults	83

mrmhub_reset_plot_defaults	83
mrmhub_set_plot_defaults	84
mrmhub_status	85
MRMhubExperiment	86
MRMhubExperiment-class	86
normalize_by_istd	87
parse_lipid_feature_names	88
parse_masshunter_csv	89
parse_mrmhub_result	90
parse_plain_long_csv	90
parse_plain_wide_csv	91
parse_skyline_result	92
plot_abundanceprofile	93
plot_calibrationcurves	96
plot_feature_correlations	100
plot_interference_correction	103
plot_matrixeffects	106
plot_normalization_qc	108
plot_pca	112
plot_pca_loading	115
plot_qc_interference_impact	118
plot_qc_summary_byclass	119
plot_qc_summary_overall	120
plot_qcmetrics_comparison	121
plot_responsecurves	125
plot_qla_boxplot	129
plot_rt_vs_chain	133
plot_runscatter	135
plot_runsequence	142
pooled_rsd	143
pooled_sd	144
quant_lcms_dataset	145
quantify_by_calibration	145
quantify_by_istd	147
read_dataset_rds	148
save_dataset_csv	149
save_dataset_mztab	151
save_dataset_rds	152
save_dataset_summarizedexperiment	154
save_feature_qc_metrics	156
save_metadata_msorganiser_template	156
save_metadata_templates	157
save_plot	157
save_report_xlsx	160
set_analysis_order	162
set_intensity_var	163
set_lipid_class	163
summarize_interferences	164

Index

165

\$.MRMhubExperiment-method

Access slots of a MRMhubExperiment object via \$ syntax

Description

\$ syntax can be used to as a shortcut for getting specific variables and results from a MRMhubExperiment object

Usage

```
## S4 method for signature 'MRMhubExperiment'
x$name
```

Arguments

x	MRMhubExperiment object
name	MRMhubExperiment slot

Value

Value with a variable or a tibble

Examples

```
mexp <- MRMhubExperiment(title = "Test Experiment", analysis_type = "lipidomics")
mexp$analysis_type
mexp$title
mexp$annot_analyses
```

add_metadata

Add metadata to an MRMhubExperiment object

Description

Metadata provided as a list of tibbles will be validated for consistency against the loaded analysis data of the provided MRMhubExperiment object and then transferred.

Usage

```
add_metadata(data = NULL, metadata, excl_unmatched_analyses = FALSE)
```

Arguments

`data` [MRMhubExperiment](#) object

`metadata` List of tibbles or data.frames containing analysis, feature, istd, response curve tables

`excl_unmatched_analyses` Exclude analyses (samples) that have no matching metadata

Value

An updated [MRMhubExperiment](#) object

See Also

The [metadata guide](#).

build_workflow	<i>Launch the MRMhub Workflow Builder</i>
----------------	---

Description

Opens an interactive Shiny application that helps you turn your data and metadata files into a runnable Quarto (.qmd) mrmhub workflow. Point the app at your files, tick the processing steps you need, and the app validates the inputs against each step (warning, for example, when `normalize_by_istd` is selected but no `istd_feature_id` is defined) while previewing the generated workflow document for download.

Usage

```
build_workflow()
```

Value

Invisible NULL. Launches the Shiny app in the default browser.

See Also

[generate_workflow_qmd\(\)](#) for the non-interactive generator.

Examples

```
if (interactive()) {  
  build_workflow()  
}
```

calc_average_molweight*Calculate average molecular weight from chemical formulas*

Description

Calculates the average molecular weight of one or more chemical formulas, based on the natural isotopic distribution of elements. The calculation uses the **enviPat** package to retrieve isotopic masses and abundances, and computes the weighted mean of the isotopic distribution.

Usage

```
calc_average_molweight(formula)
```

Arguments

formula A character vector of one or more chemical formulas to process.

Details

This function calculates the *average molecular weight* (not monoisotopic mass), which reflects the average of the isotopic distribution found in nature.

Isotopes can be specified explicitly in the formula. Atomic mass numbers for isotopes must be enclosed in square brackets (e.g., [13]C for carbon-13). Deuterium must be written as D instead of [2]H.

The function uses the **enviPat** package to validate and parse chemical formulas, calculate isotopic patterns, and determine the average molecular weight based on weighted means of isotopic abundances.

Value

A numeric vector of average molecular weights, one for each formula; NA for a missing formula.

References

Loos, M., Gerber, C., Corona, F., Hollender, J., & Singer, H. (2015). Accelerated Isotope Fine Structure Calculation Using Pruned Transition Trees. *Analytical Chemistry*, 87(11), 5738–5744. [doi:10.1021/acs.analchem.5b00941](https://doi.org/10.1021/acs.analchem.5b00941)

Examples

```
calc_average_molweight(c("C6H12O6", "[13]C6H12O6", "C8H10N4O2", "D2O"))
```

calc_calibration_results

Calculate external calibration curve results

Description

Calibration curves are calculated for each feature using ISTD-normalized intensities and the corresponding concentrations of calibration samples, as defined in the qc_concentrations metadata. The regression fit model (linear or quadratic) and the weighting method (either "none", "1/x", "1/x^2", or "1/sqrt(x)") can be defined globally via the arguments fit_model and fit_weighting for all features, if fit_overwrite is TRUE. Alternatively, the model and weighting can be defined individually for each feature in the feature metadata (columns curve_fit_model and curve_fit_weighting). If these details are missing in the metadata, the default values provided via fit_model and fit_weighting will be used.

Usage

```
calc_calibration_results(
  data = NULL,
  variable = "feature_norm_intensity",
  include_qualifier = TRUE,
  fit_overwrite,
  fit_model,
  fit_weighting,
  ignore_missing_annotation = FALSE,
  include_fit_object = FALSE,
  lod_sigma = c("residual", "intercept"),
  ignore_failed_calibration = FALSE
)
```

Arguments

data	A MRMhubExperiment object containing the data to be used for calibration.
variable	A character string specifying the variable for calibration. Use "feature_norm_intensity" for typical scenarios involving internal standardization. When performing only external standardization, without internal standardization, use "feature_intensity".
include_qualifier	A logical value. If TRUE, the function will include qualifier features in the calibration curve calculations.
fit_overwrite	If TRUE, the function will use the provided fit_model and fit_weighting values for all analytes and ignore any fit method and weighting settings defined in the metadata.
fit_model	A character string specifying the default regression fit method to use for the calibration curve. Must be one of "linear" or "quadratic". This method will be applied if no specific fit method is defined for a feature in the metadata, or when fit_overwrite = TRUE.

fit_weighting	A character string specifying the default weighting method for the regression points in the calibration curve. Must be one of "none", "1/x", "1/x^2", or "1/sqrt(x)". This method will be applied if no specific weighting method is defined for a feature in the metadata, or when fit_overwrite = TRUE.
ignore_missing_annotation	If FALSE, an error will be raised if calibration curve data is missing for any feature.
include_fit_object	If TRUE, the function will return the full regression fit objects for each feature in the metrics_calibration table.
lod_sigma	A character string selecting the response standard deviation (sigma) used in the ICH Q2 LoD/LoQ formulas. Must be one of "residual" (the residual standard error of the regression, Sy/x; the default) or "intercept" (the standard error of the intercept). No averaging of the two is performed.
ignore_failed_calibration	If FALSE, an error is raised if all quantifier calibration curve fits fail. If TRUE, a warning is shown instead and the failed fits are returned, so resulting concentrations are NA.

Details

A linear curve is also fitted with 2 calibrators, and a single calibrator gives a line through the origin ($\text{coef_a} = 0$). A quadratic curve needs at least 3 calibrators. A zero-concentration calibrator is dropped from weighted fits, so a blank plus one standard is then fitted through the origin. A curve that passes through all its calibrators (e.g. 2 points for a line, 3 for a quadratic) has no R^2 , sigma, LoD or LoQ (NA).

Additionally, the limit of detection (LoD) and limit of quantification (LoQ) are calculated for each feature based on the calibration curve, following the ICH Q2(R1/R2) approach ($\text{LoD} = 3.3 \text{ sigma} / S$, $\text{LoQ} = 10 \text{ sigma} / S$). Here S is the slope of the calibration curve, taken at zero concentration (the linear coefficient coef_b). For a quadratic fit the true slope is $b + 2cx$, which reduces to coef_b at zero concentration, so the quadratic term does not contribute to the slope used here. ICH Q2 specifies the slope formula for linear responses only; using the low-concentration tangent slope for a quadratic fit is an approximation beyond the guideline (adequate when the curvature near zero is small).

The response standard deviation sigma is selected via `lod_sigma`, following the ICH-acceptable choices: "residual" uses the residual standard error of the regression (Sy/x ; the default and prior behaviour), while "intercept" uses the standard error of the intercept. The sigma column reported in `metrics_calibration` is always the residual standard error, independent of this choice.

The results of the regression and the calculated LoD and LoQ values are stored in the `metrics_calibration` table of the returned `MRMhubExperiment` object.

Calibration coefficient columns:

The fitted curve is stored in **ascending** power order, following R's `lm()` / `poly()` convention. The model is

$\text{response} = \text{coef_a} + \text{coef_b} * x + \text{coef_c} * x^2$

- `coef_a` — intercept (0th-order term, x^0)
- `coef_b` — 1st-order (linear) coefficient (x^1); the slope of a linear fit

- coef_c — 2nd-order (quadratic) coefficient (x^2); NA for a linear fit

Only linear and quadratic models are fitted, so there are no higher-order terms. This ascending order is the **reverse** of the descending $a \cdot x^2 + b \cdot x + c$ form printed by some vendor software (e.g. Agilent MassHunter, where a is the quadratic term). When comparing against such an export, match coefficients by power (coef_a = intercept, coef_c = x^2), not by letter.

Value

A modified [MRMhubExperiment](#) object with an updated metrics_calibration table containing the calibration curve results, including concentrations, LoD, and LoQ values for each feature.

References

ICH Harmonised Tripartite Guideline. Validation of Analytical Procedures: Text and Methodology Q2(R1) (2005); Q2(R2) (2023). International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use.

See Also

[quantify_by_calibration\(\)](#) for calculating concentrations based on external calibration curves.

calc_isotopic_interferences

Derive isotopic interference relationships

Description

Automatically discovers isotopic ($M+2$) interference relationships between measured features and stores them in the annot_interferences slot, ready for [correct_isotopic_interferences\(\)](#). Two levels are supported:

- "MRM" (default): fragment-based front/back correction for class-based LC-MRM data, derived from the LICAR method (see the manual). Requires precursor **and** product m/z and an mrm_pattern per feature.
- "MS1": whole-molecule $M+2$ correction for **MS1 / full-scan** acquisitions. The factor is the $M+2$ relative abundance of each species' molecular formula; interfering pairs are matched by precursor m/z (~ 2 Da apart) within a feature class. Requires precursor m/z only.

Usage

```
calc_isotopic_interferences(
  data = NULL,
  level = c("MRM", "MS1"),
  mz_tol = 0.5,
  min_contribution = 0,
  check_coelution = FALSE
)
```

Arguments

data	A <code>MRMhubExperiment</code> object.
level	Correction level, "MRM" or "MS1". See description.
mz_tol	Precursor m/z tolerance (Da) for matching interfering pairs at the MS1 level (MRM uses a fixed 2 +/- 0.2 Da precursor window). Default 0.5.
min_contribution	Drop derived edges whose contribution factor K is below this value. Default 0 (keep all). Raise to skip negligible or borderline pairs (e.g. 0.002).
check_coelution	(<i>experimental</i>) If TRUE, an m/z-matched edge is kept only if the interferer and victim co-elute – the interferer’s peak apex falls within the victim’s integration window (imported borders <code>feature_int_start/feature_int_end</code> , else <code>feature_rt +/- FWHM</code>). Chromatographically resolved pairs are dropped. Default FALSE while the gate is validated.

Details

Important — MS1 is not a fallback for MRM. For MRM data the isotopic correction must be *fragment-based*: the contribution of heavy isotopes to a transition depends on the isotope’s location relative to the fragmentation (Gao et al. 2021). The whole-molecule ("MS1") level is only valid for genuine MS1/full-scan measurements and must **not** be used as a substitute for MRM data that happens to lack a product m/z.

Derived factors are sensitive to the **enviPat** version; version 2.8 is the reference. Existing manual interferences (source == "manual") are preserved; previously derived ("auto") rows are replaced.

How the contribution factor is computed:

The contribution K is the fraction of a lighter species’ signal that appears at M+2; it is subtracted downstream by `correct_isotopic_interferences()` as $value_{corrected} = value_{raw} - K \cdot value_{interferer}$.

- **MS1 (whole-molecule):** $K = \sum ab(\text{peaks within } \pm 0.5 \text{ Da of } M_0 + 2) / ab(M_0)$, computed from the species’ molecular formula with **enviPat** `isopattern` (see `mN_re1_abundance()`).
- **MRM (fragment-based, LICAR):** the fragment elemental formula is built from the precursor carbon/hydrogen count plus a per-class offset $c(C, H, O, N)$, with $H = 2C + c_H - 2H_{raw}$; K is that fragment’s M+2 isotopologue fraction from **enviPat** `isowrap` (centroided, algorithm 2 — chosen over `isopattern` for parity with published LICAR values). *Front* vs *back* distinguishes whether the heavy isotope lands on the retained product ion or the neutral loss; FA/LCB species carry a front + back pair, and LCB inverts the front/back assignment relative to FA.

Value

The `MRMhubExperiment` with a populated `annot_interferences` slot.

References

Gao L. et al. (2021). LICAR: An Application for Isotopic Correction of Targeted Lipidomic Data Acquired with Class-Based Chromatographic Separations Using Multiple Reaction Monitoring. *Analytical Chemistry*, 93(6), 3163-3171. doi:10.1021/acs.analchem.0c04565

calc_qc_metrics	<i>Calculate quality control (QC) metrics for features</i>
-----------------	--

Description

Computes various quality control (QC) metrics for each feature in a `MRMhubExperiment` object. Metrics are derived from different sample types and can be computed either across the full dataset or as medians of batch-wise calculations.

Usage

```
calc_qc_metrics(
  data = NULL,
  use_batch_medians = FALSE,
  use_robust_cv = FALSE,
  include_norm_intensity_stats = NA,
  include_conc_stats = NA,
  include_response_stats = NA,
  include_calibration_results = NA
)
```

Arguments

<code>data</code>	A <code>MRMhubExperiment</code> object containing data and metadata, whereby data needs to be normalized and quantitated for specific QC metrics, such as statistics based on normalized intensities and concentrations.
<code>use_batch_medians</code>	Logical, whether to compute QC metrics using the median of batch-wise derived values instead of the full dataset. Default is <code>FALSE</code> .
<code>use_robust_cv</code>	Logical, whether to use robust coefficient of variation (MAD/median) or the standard coefficient of variation (SD/mean) for intensity, norm_intensity and concentration metrics.
<code>include_norm_intensity_stats</code>	Logical. If <code>NA</code> (default), statistics on normalized intensity values are included if the data is available. If <code>TRUE</code> , they are always calculated, raising an error if data is missing.
<code>include_conc_stats</code>	Logical. If <code>NA</code> (default), concentration-related statistics are included if concentration data is available. If <code>TRUE</code> , they are always calculated, raising an error if data is missing.
<code>include_response_stats</code>	Logical. If <code>NA</code> (default), response curve statistics are included if the required data is available. If <code>TRUE</code> , they are always calculated, raising an error if data is missing.
<code>include_calibration_results</code>	Logical. If <code>NA</code> (default), external calibration results are incorporated into the QC metrics table if available. If <code>TRUE</code> , they are always incorporated.

Details

Batch-wise calculations: The function computes the following QC metrics for each feature and for different QC sample types (e.g., SPL, TQC, BQC, PBLK, NIST, LTR)

The format for the metrics is standardized as `metric_name_qc_type`, where `qc_type` refers to the specific QC sample type for which the metric is calculated. For example: `intensity_min_spl` refers to the minimum intensity Statistics of normalized intensities, external calibration, and response curves can be included by setting the relevant arguments (`include_norm_intensity_stats`, `include_conc_stats`, `include_response_stats`, `include_calibration_results`) to `TRUE`.

Note when corresponding underlying processed data is not available, the function will not raise an error but will return NA values for the respective metrics. This, however, does not apply for the optional metrics mentioned above. For these cases an error will be raised if the underlying data is missing.

If `use_batch_medians = TRUE`, batch-specific QC statistics are computed first, and then the median of these values is returned for each feature. However, response curve and calibration statistics are calculated per curve, irrespective of batches and `use_batch_medians` settings.

The calculated metrics are stored in the `metrics_qc` table of the `MRMhubExperiment` objects and comprises following details

- **Feature details:** Specific feature information extracted from the feature metadata table, such as feature class, associated ISTD, quantifier status.
- **Feature MS method information** (if available in the imported data): `precursor_mz`, `product_mz` and `collision_energy` per feature. A value that differs between analyses indicates inconsistent acquisition conditions; it is set to NA with a warning naming the features.
- **Missing Value Metrics:**
 - `missing_intensity_prop_spl`: Proportion of missing intensities for the SPL sample type.
 - `missing_norm_intensity_prop_spl`: Proportion of missing normalized intensities for SPL samples.
 - `missing_conc_prop_spl`: Proportion of missing concentration values for SPL samples.
 - `na_in_all`: Indicator if a feature has all missing intensities across all samples
- **Retention Time (RT) Metrics:** Requires that retention time data are available.
 - `rt_min_*`: Minimum retention time across different QC sample types (e.g., SPL, BQC, TQC).
 - `rt_max_*`: Maximum retention time across different QC sample types.
 - `rt_median_*`: Median retention time for specific QC sample types like PBLK, SPL, BQC, TQC, etc.
- **Intensity Metrics:**
 - `n_bqc`, `n_tqc`, `n_spl`: Number of analyses with a non-missing intensity per QC type, i.e. the replicates behind the %CV and D-ratio (the median of the per-batch counts with `use_batch_medians = TRUE`). %CV and D-ratio are NA below 3 replicates.
 - `intensity_min_*`: Minimum intensity value for features across different QC sample types such as SPL, TQC, BQC, etc.
 - `intensity_max_*`: Maximum intensity values across sample types.
 - `intensity_median_*`: Median intensity for various QC sample types.

- intensity_cv_*: Coefficient of variation (CV) of intensity values for specific QC types.
- sb_ratio_*: Signal-to-blank ratios such as the ratio of intensity values for SPL vs PBLK, UBLK, or SBLK. Blank medians count a blank analysis without detected signal (a missing value or no row for the feature) as zero, giving a ratio of Inf.
- intensity_q10_*: The 10th percentile of intensity values for the SPL sample type.
- **Normalized Intensity Metrics** (only if include_norm_intensity_stats = TRUE): Requires that raw intensities were normalized, see [normalize_by_istd\(\)](#) for details.
 - norm_intensity_cv_*: Coefficient of variation (CV) of normalized intensities for QC sample types like TQC, BQC, SPL, etc.
- **Concentration Metrics** (only if include_conc_stats = TRUE): Requires that concentration were calculated, see [quantify_by_istd\(\)](#) or [quantify_by_calibration\(\)](#) for details.
 - conc_median_*: Median concentration values for different QC sample types like TQC, BQC, SPL, NIST, and LTR.
 - conc_cv_*: Coefficient of variation (CV) for concentration values.
 - conc_dratio_sd_*: The ratio of standard deviations of concentration between BQC or TQC and SPL samples.
 - conc_dratio_mad_*: The ratio of median absolute deviations (MAD) between BQC or TQC and SPL concentrations.
- **Response Curve Metrics** (if include_response_stats = TRUE): Calculates response curve statistics for each feature and each curve (where # refers to the curve identifier). Requires that response curves are defined in the data. See [get_response_curve_stats\(\)](#) for additional details.
 - r2_rqc_#: R² value of the linear regression for the response curve, representing the goodness of fit.
 - slopenorm_rqc_#: Normalized slope of the linear regression for the response curve, indicating the relationship between the response and concentration.
 - y0norm_rqc_#: Normalized intercept of the linear regression for the response curve, representing the baseline or starting value.
- **External Calibration Results** Incorporates external calibration results, if include_calibration_results = TRUE and calibration curves are defined in the data:
 - fit_model: The regression model used for curve fitting.
 - fit_weighting: The weighting method applied during curve fitting.
 - lowest_cal: The lowest nonzero calibration concentration.
 - highest_cal: The highest calibration concentration.
 - r_squared: R² value indicating the goodness of fit.
 - coef_a: The intercept of the regression line (both **linear** and **quadratic** fits).
 - coef_b:
 - * For **linear fits**, the slope of the regression line.
 - * For **quadratic fits**, the coefficient of the linear term (x).
 - coef_c:
 - * For **quadratic fits**, the coefficient of the quadratic term (x²).
 - * Set to NA for linear fits.
 - sigma: The residual standard error of the regression model.
 - reg_failed: Boolean flag indicating if regression fitting failed.

Value

A [MRMhubExperiment](#) object with an updated metrics_qc table containing computed QC metrics for each feature.

`calibrate_by_reference`*Calibrate feature values using a reference sample*

Description

This function calibrates feature abundances based on a specified reference sample. Calibration can be applied to the entire dataset using one or more reference samples, or batch-wise using reference sample analyses present within each batch. For both approaches, multiple measurements of the same reference sample are summarized using either mean (default) or median (set by the `summarize_fun` argument).

Usage

```
calibrate_by_reference(  
  data,  
  variable,  
  reference_sample_id,  
  absolute_calibration,  
  batch_wise = FALSE,  
  summarize_fun = "mean",  
  store_conc_ratio = NULL,  
  undefined_conc_action = NULL,  
  store_normalized = FALSE  
)
```

Arguments

<code>data</code>	A MRMhubExperiment object containing the metabolomics data to be normalized
<code>variable</code>	Character string indicating which data type to calibrate. Must be one of: "intensity", "norm_intensity", or "conc"
<code>reference_sample_id</code>	Character vector specifying the sample ID(s) to use as reference(s) or standards. When more than one ID is given, all analyses whose <code>sample_id</code> matches any of them are pooled and summarized together (per feature, and per batch when <code>batch_wise = TRUE</code>).
<code>absolute_calibration</code>	Logical indicating whether to perform absolute calibration using known concentrations of the reference sample (TRUE) or relative calibration (FALSE).
<code>batch_wise</code>	Logical indicating whether to perform calibration for each batch separately (TRUE) or for all samples together (FALSE).

summarize_fun	Either "mean" or "median". If absolute_calibration = TRUE, this function is used to summarize the reference sample concentrations across analyses of specified reference_sample_id. Default is "mean".
store_conc_ratio	Logical. Whether to store the ratio of measured (non-calibrated) compared to the expected (known) concentrations. Only applied if absolute_calibration = TRUE. This ratio is stored under the feature variable feature_conc_ratio. By default it is TRUE when variable = 'conc', otherwise FALSE.
undefined_conc_action	Character string specifying how to handle features without defined concentrations in reference samples when absolute_calibration = TRUE. Must be one of "original" (keep original values), "na" (set to NA), or "error". Required when absolute_calibration = TRUE (no default).
store_normalized	Logical indicating whether to keep the normalized values in the dataset when absolute_calibration = TRUE. Default is FALSE. These values are then stored as [VARIABLE]_normalized, where [VARIABLE] is the input variable, e.g., conc.

Details

Calibration can be performed in two ways, either absolute, resulting in concentrations, or relative, resulting in ratios:

1. Absolute calibration (when absolute_calibration = TRUE)

Calibrates (or re-calibrates) feature abundances based on known concentrations of the corresponding features defined for a reference sample. The calibrated concentration for a given analyte is calculated as:

$$c_{\text{cal}}^{\text{Analyte}} = \frac{c_{\text{sample}}^{\text{Analyte}}}{c_{\text{ref}}^{\text{Analyte}}} \times c_{\text{known}}^{\text{Analyte}}$$

The input variable can be either conc, norm_intensity, or intensity, whereas the result is always stored under the variable conc (concentration), in the unit defined for the feature concentrations in the reference sample.

Metadata requirements:

- sample_id and analyte_id must be defined for the reference sample and features in the analysis and feature metadata, respectively.
- Known analyte concentrations must be defined in the QC concentration metadata for the reference sample
- An error will be raised if no concentrations are defined for any features

Missing analyte concentrations for the reference sample can be handled via undefined_conc_action with following options:

- original: Keep original feature values, i.e. the non-calibrated values will be returned. *Note:* this is only available when variable = conc. Use with caution to avoid mixing units.
- na: Set affected features values to NA

- error: The function stops with error in case of any undefined reference sample feature concentration.
- In case all feature concentrations are undefined, the function will stop with an error.

The re-calibrated feature concentrations are stored as `conc`, overwriting existing `conc` values. The original `conc` values are stored as `conc_beforecal`.

The ratio between the measured and expected (known) concentrations in the reference sample is available via the feature variable `feature_conc_ratio` and is calculated as follows:

$$C_{\text{ratio}}^{\text{Analyte}} = \frac{C_{\text{measured}}^{\text{Analyte}}}{C_{\text{expected}}^{\text{Analyte}}}$$

where C_{measured} is the measured (non-calibrated) concentration, and C_{expected} is the known or reference concentration for the same analyte. A ratio of 1 indicates perfect agreement; values above or below 1 indicate over- or underestimation.

To export the calibrated concentrations use `save_dataset_csv()` with `variable = "conc"`, or to export non-calibrated values with `variable = "conc_beforecal"`. When saving the MRMhub XLSX report, the calibrated concentrations will also be stored as `conc`.

2. Normalization (relative calibration, `absolute_calibration = FALSE`)

Normalizes feature abundances with corresponding feature abundances in a reference sample, resulting in ratios. Any available feature abundance variable (i.e., `conc`, `norm_intensity`, or `intensity`) can be used as input. The normalization is calculated for all present features. The resulting output will be stored as `[VARIABLE]_normalized`, whereby `[VARIABLE]` is the input variable, e.g., `conc_normalized`.

To export the normalized abundances, use `save_dataset_csv()` with `variable = "[VARIABLE]_normalized"`. When saving the MRMhub XLSX report via `save_report_xlsx()`, available unfiltered normalized feature abundances will be included by default. To include filtered normalized feature abundances, set `filtered_variable = "[VARIABLE]_normalized"`.

Value

A `MRMhubExperiment` object with calibrated data

See Also

`normalize_by_istd()`, `quantify_by_istd()`, `quantify_by_calibration()`

check_setup

Check MRMhub setup

Description

Validates that the user's R environment is correctly configured to use mrmhub. Checks R version, key dependencies, and optional packages.

Usage

```
check_setup(verbose = TRUE)
```

Arguments

verbose Logical. If TRUE (default), prints detailed results. If FALSE, returns results invisibly.

Value

A list (invisibly) with elements `r_version`, `required`, and `optional`, each containing pass/fail status.

Examples

```
check_setup()
```

```
correct_batch_centering
```

Batch centering correction

Description

This function performs batch centering correction on each feature. Optionally, the scale of the batches can be equalized across batches. The selected QC types (`ref_qc_types`) are used to calculate the medians, which are then used to align all other samples. The correction can be applied to one of three variables: "intensity", "norm_intensity", or "conc". The correction can either be applied on top of previous corrections or replace all prior batch corrections.

Usage

```
correct_batch_centering(
  data = NULL,
  variable,
  ref_qc_types,
  correct_scale = FALSE,
  ignore_istd = TRUE,
  replace_previous = TRUE,
  log_transform_internal = TRUE,
  feature_list = NULL,
  replace_exisiting_trendcurves = FALSE,
  ...
)
```

Arguments

data A [MRMhubExperiment](#) object containing the data to be corrected. This object must include information about QC types and measurements.

variable The variable to be corrected. Must be one of "intensity", "norm_intensity", or "conc".

ref_qc_types	A character vector specifying the QC types to be used as references for batch centering.
correct_scale	A logical value indicating whether to equalize the scale of the batches in addition to center them. Defaults to FALSE.
ignore_istd	Logical. Exclude internal standards (ISTDs) from correction if TRUE (the default). Their values are left unchanged.
replace_previous	A logical value indicating whether to replace any previous batch corrections or apply the new correction on top. Defaults to TRUE (replace).
log_transform_internal	A logical value indicating whether to log-transform the data internally during correction. Defaults to TRUE. This also sets the centering model: with TRUE the batch reference levels are aligned in log space, i.e. multiplicative (geometric) centering (the appropriate choice for multiplicatively-scaling MS intensities, and the reason it is the default); with FALSE they are aligned in raw space, i.e. additive centering (which can shift low values below zero). Either way the returned data are on the raw (untransformed) scale.
feature_list	Sets specific features for correction only. Can be character vector or a single string which is then interpreted as regular expression. Default is NULL which means all features are selected.
replace_exisiting_trendcurves	A logical value indicating whether to replace trend curves from previous corrections. This is only used for plotting using <code>plot_runscatter()</code> . Default is FALSE.
...	Additional arguments that can be passed to the batch correction function.

Value

A `MRMhubExperiment` object containing the corrected data.

See Also

`plot_runscatter()` for visualizing the correction before and after; the [drift-correction tutorial](#) for a worked example.

correct_batch_combat *ComBat batch correction*

Description**[Experimental]**

Adjusts batch effects with the empirical-Bayes ComBat method (Johnson et al. 2007), applied to one of "intensity", "norm_intensity", or "conc". ComBat models a location and scale batch effect per feature and shrinks those estimates across features, which can stabilise many small batches better than simple median centering.

Unlike `correct_batch_centering()` and `correct_batch_serrf()`, ComBat estimates batch effects from all study samples and routine QCs (optionally protecting biology via covariates), not only from the reference QCs. Blanks, response curves, calibrants and other analysis types are left out of the fit and keep their uncorrected values. On strongly unbalanced designs this can remove genuine biological signal, so supply covariates when the biological grouping is not balanced across batches. `ref_qc_types` is used only for the before/after QC-CV report and the plotting trend curves.

Batch correction is performed **after** normalization and drift correction in the recommended pipeline. Features with any missing or non-finite values in the selected variable are left uncorrected (ComBat requires complete data).

Usage

```
correct_batch_combat(
  data = NULL,
  variable,
  ref_qc_types,
  covariates = NULL,
  ref_batch = NULL,
  parametric = TRUE,
  ignore_istd = TRUE,
  replace_previous = TRUE,
  log_transform_internal = TRUE,
  feature_list = NULL,
  replace_exisiting_trendcurves = FALSE
)
```

Arguments

<code>data</code>	A <code>MRMhubExperiment</code> object.
<code>variable</code>	The variable to correct: one of "intensity", "norm_intensity", or "conc".
<code>ref_qc_types</code>	Character vector of QC types used for the QC-CV report and trend curves (not for the ComBat fit itself).
<code>covariates</code>	Optional model matrix of biological covariates to preserve (passed to <code>sva::ComBat()</code> as <code>mod</code>), with the analysis IDs as row names. Rows are matched to analyses by name; rows of analyses not used in the fit are ignored. Defaults to <code>NULL</code> (no covariates).
<code>ref_batch</code>	Optional reference batch to adjust the others towards (passed to <code>sva::ComBat()</code> as <code>ref.batch</code>). Defaults to <code>NULL</code> .
<code>parametric</code>	Use the parametric empirical-Bayes prior (<code>TRUE</code> , default) or the non-parametric prior (<code>FALSE</code>).
<code>ignore_istd</code>	Logical. Exclude internal standards (ISTDs) from correction if <code>TRUE</code> (the default). Their values are left unchanged, and they do not contribute to the empirical-Bayes prior.
<code>replace_previous</code>	Replace a previous batch correction (<code>TRUE</code> , default) or apply on top of it.

log_transform_internal
 Fit ComBat in log10 space (TRUE, default, appropriate for multiplicatively-scaling MS data). Returned data are always back-transformed to the raw scale.

feature_list
 Optional feature selection (character vector or a single regular expression); NULL (default) selects all features.

replace_exisiting_trendcurves
 Reseed the plotting trend curves. Default FALSE.

Value

A [MRMhubExperiment](#) with corrected data.

References

Johnson WE, Li C, Rabinovic A (2007). Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics*, 8(1), 118-127. doi:10.1093/biostatistics/kxj037

Applied through [sva::ComBat\(\)](#) from the sva package. See also Broadhurst D, et al. (2018), *Metabolomics*, 14, 72 (doi:10.1007/s1130601813673) on QC-based signal correction.

See Also

[correct_batch_centering\(\)](#), [correct_batch_serrf\(\)](#), [correct_drift_loess\(\)](#) and [plot_runscatter\(\)](#) for visualisation. The [drift and batch correction manual](#).

Examples

```
# mexp <- correct_batch_combat(mexp, variable = "conc", ref_qc_types = "BQC")
```

correct_batch_serrf *SERRF batch correction*

Description

[Experimental]

This is an independent re-implementation of SERRF in mrmhub, adapted from the reference code in the malbacR package (not the original authors' package); see Details. Validate results against the reference SERRF for your data.

Normalises systematic error with SERRF (Systematic Error Removal using Random Forest; Fan et al. 2019). For each feature and batch a random forest is trained on the reference QC samples, using the batch's most-correlated features as predictors, and the learned systematic error is removed from the study samples and routine QCs; blanks, response curves, calibrants and other analysis types keep their uncorrected values. Unlike ComBat, SERRF captures non-linear drift and batch effects jointly and is anchored on the QC samples, matching the QC-based design of the package; it is best suited to larger panels with dense QC coverage.

SERRF operates on the raw abundance scale (no log transform). Features with missing or non-positive values, and batches with fewer than two reference QCs, are left uncorrected. Batch correction is performed **after** normalization and drift correction in the recommended pipeline.

Usage

```
correct_batch_serrf(
  data = NULL,
  variable,
  ref_qc_types,
  n_correlated = 10,
  num_trees = 500,
  seed = 1L,
  num_threads = 1L,
  show_progress = TRUE,
  ignore_istd = TRUE,
  replace_previous = TRUE,
  feature_list = NULL,
  replace_exisiting_trendcurves = FALSE
)
```

Arguments

data	A MRMhubExperiment object.
variable	The variable to correct: one of "intensity", "norm_intensity", or "conc".
ref_qc_types	Character vector of QC types used as the SERRF training (reference) samples per batch.
n_correlated	Number of most-correlated features used as random-forest predictors per feature. Default 10.
num_trees	Number of trees per random forest. Default 500.
seed	Random seed for ranger::ranger() , for reproducibility. Default 1.
num_threads	Threads per random forest passed to ranger::ranger() . Default 1; kept low because batches are already corrected in parallel and each forest trains on a small QC set.
show_progress	Show a progress bar over batches. Default TRUE.
ignore_istd	Logical. Exclude internal standards (ISTDs) from correction if TRUE (the default). Their values are left unchanged, and they are not available as random-forest predictors.
replace_previous	Replace a previous batch correction (TRUE, default) or apply on top of it.
feature_list	Optional feature selection (character vector or a single regular expression); NULL (default) selects all features.
replace_exisiting_trendcurves	Reseed the plotting trend curves. Default FALSE.

Details

The implementation adapts the reference SERRF code in the `malbacR` package, keeping its normalization arithmetic but selecting each feature's predictors from the per-batch QC-sample Spearman correlation (a simplification of `malbacR`'s QC/sample correlation intersection). Random forests are

fit with `ranger::ranger()` and a fixed seed, so results are reproducible. Users are encouraged to validate results against the reference SERRF implementation for their data.

Batches are corrected independently and in parallel via `mirai` (through `purrr::in_parallel()`); set up workers with `mirai::daemons()` beforehand to use them, otherwise the batches are processed sequentially.

Value

A `MRMhubExperiment` with corrected data.

References

Fan S, Kind T, Cajka T, et al. (2019). Systematic Error Removal Using Random Forest for Normalizing Large-Scale Untargeted Lipidomics Data. *Analytical Chemistry*, 91(5), 3590-3596. doi:10.1021/acs.analchem.8b05592

Implementation adapted from the `malbacR` package (<https://github.com/pmartR/malbacR>); random forests via `ranger` (Wright MN, Ziegler A, 2017, *Journal of Statistical Software*, 77(1), doi:10.18637/jss.v077.i01).

See Also

`correct_batch_centering()`, `correct_batch_combat()`, `correct_drift_loess()` and `plot_runcscatter()` for visualisation. The [drift and batch correction manual](#).

Examples

```
# mexp <- correct_batch_serrf(mexp, variable = "conc", ref_qc_types = "BQC")
```

`correct_custom_interferences`

Correct declared (custom) interferences

Description

Applies the interference corrections you declared in the feature metadata (the legacy `interference_feature_id` / `interference_contribution` columns, and any `source == "manual"` rows in `annot_interferences`) to the raw feature intensities, using the same engine and formula as `correct_isotopic_interferences()`. Use this for in-source fragments, co-eluting isobars or other non-isotopic interferences you know about. Warns and returns the data unchanged if none are defined.

Usage

```
correct_custom_interferences(  
  data = NULL,  
  variable = "feature_intensity",  
  sequential_correction = TRUE,  
  neg_to_na = FALSE  
)
```

Arguments

data	MRMhubExperiment object.
variable	Name of the variable to correct. Only "feature_intensity" (the raw intensity) is supported. Default: "feature_intensity".
sequential_correction	Logical. If TRUE (default), a chain of interferences is corrected sequentially so each feature uses the already-corrected signal of its interferer (propagates along the chain). If FALSE, each feature is corrected from the raw interferer, without propagation.
neg_to_na	If TRUE, negative or zero values after correction are replaced with NA. Default: FALSE.

Value

[MRMhubExperiment](#) object with feature intensities corrected.

See Also

[correct_isotopic_interferences\(\)](#), [correct_interference_manual\(\)](#); the [interference-correction guide](#).

correct_drift	<i>Drift correction by custom function</i>
---------------	--

Description

Function to correct for run-order drifts within or across batches via a provided custom function

Usage

```
correct_drift(
  data = NULL,
  smooth_fun,
  variable,
  ref_qc_types,
  batch_wise,
  replace_previous = TRUE,
  log_transform_internal = TRUE,
  conditional_correction = FALSE,
  cv_diff_threshold = 0,
  use_original_if_fail = TRUE,
  ignore_istd = TRUE,
  feature_list = NULL,
  recalc_trend_after = FALSE,
  show_progress = TRUE,
  ...
)
```

Arguments

data	MRMhubExperiment object
smooth_fun	Function that performs the drift correction for a single feature within one batch. It must accept the arguments <code>tbl</code> (a data frame with the columns <code>qc_type</code> , <code>x</code> (run-order number), <code>y</code> (the variable to correct), and <code>analysis_id</code> , <code>feature_id</code> , <code>batch_id</code>), <code>ref_qc_types</code> (one or more QC-type strings used to fit the trend), <code>log_transform_internal</code> (logical), and ... (smoother-specific parameters such as <code>span</code> or <code>degree</code>). It must return a named list carrying <code>y_adj</code> (the drift-adjusted <code>y</code>) and <code>fit_error</code> (logical), and typically also <code>y_fit</code> , <code>fit_warning</code> , and the passed-through <code>analysis_id</code> , <code>feature_id</code> , <code>batch_id</code> , <code>qc_type</code> , and <code>x</code> . When the fit fails it must return that same structure with <code>y_adj = NA_real_</code> and <code>fit_error = TRUE</code> . See fun_loess() for a reference implementation.
variable	The variable to be corrected for drift effects. Must be one of "intensity", "norm_intensity", or "conc"
ref_qc_types	QC types used for drift correction
batch_wise	Apply to each batch separately if TRUE (the default)
replace_previous	Logical. Replace previous correction (TRUE), or adds on top of previous correction (FALSE). Default is TRUE.
log_transform_internal	Apply log transformation internally for smoothing if TRUE (default). This enhances robustness against outliers but does not affect the final data, which remains untransformed.
conditional_correction	Determines whether drift correction is applied to all features unconditionally (FALSE, the default) or, when TRUE, only to features whose difference of sample CV before vs after smoothing is below the threshold specified by <code>cv_diff_threshold</code> .
cv_diff_threshold	This parameter defines the maximum allowable change (difference) in the coefficient of variation (CV) of samples before and after smoothing for the correction to be applied. A value of 0 (the default) requires the CV to improve, while a value above 0 allows the CV to also become worse by a maximum of the defined difference.
use_original_if_fail	Determines the action when smoothing fails or results in invalid values for a feature. If TRUE (default), the original data is used; if FALSE, the result for each analysis is NA.
ignore_istd	Do not apply corrections to ISTDs
feature_list	Sets specific features for correction only. Can be character vector or a single string which is then interpreted as regular expression. Default is NULL which means all features are selected.
recalc_trend_after	Recalculate trend post-drift correction for <code>plot_runcscatter()</code> . This will double calculation time.
show_progress	Show progress bar. Default = TRUE.
...	Arguments specific for the smoothing function

Details

The drift correction function needs to be provided by the user. See `smooth_fun` for details.

Value

`MRMhubExperiment` object

`correct_drift_cubicspline`

Drift correction by cubic spline smoothing

Description

This function corrects for run-order drifts within or across batches using cubic spline smoothing. The correction is typically based on QC (Quality Control) samples that are measured at specific intervals throughout the run sequence. The smoothed curve derived from the QC samples is then used to adjust all other samples in the dataset. The correction can be applied to "intensity", "norm_intensity", or "conc" data.

The cubic spline smoothing approach, particularly when used with the regularization parameter `lambda`, is similar but not identical to previously described QC-based drift correction methods, such as **QC-RSC (Quality Control Regularized Spline Correction)**, described in Dunn et al. (Nat Protoc, 2011) and Kirwan et al. (Anal Bioanal Chem, 2013).

By default, the smoothing parameter is determined using cross-validation, which can lead to overfitting. To reduce overfitting the regularization parameter `lambda` may be defined, with a good starting point being `lambda = 0.01`. Additionally, the global smoothing parameter can be specified via `spar`.

It is recommended to visually inspect the correction using the `plot_runscatter()` function. Set the argument `recalc_trend_after = TRUE` so that the trends after correction are also available for plotting. For further details, refer to the description of `plot_runscatter()`.

The corrections can be applied on a batch-by-batch basis (`batch_wise = TRUE`, default) or across all batches (`batch_wise = FALSE`). Existing corrections are either replaced (`replace_previous = TRUE`) or added on top of them (`replace_previous = FALSE`).

Furthermore, drift correction can be applied unconditionally (`conditional_correction = FALSE`) or conditionally, based on whether the sample CV change before and after correction is below a defined threshold (`cv_diff_threshold`). This conditional correction is assessed independently for each batch if `batch_wise = TRUE`, where the median of the CV changes across the batch is compared with the threshold.

Note: The function outputs a message indicating the median CV change and the median absolute CV before and after correction for all samples. However, these metrics are experimental and should not be used as definitive criteria for correction (see Details below).

This cubic spline method is implemented using the base R function `stats::smooth.spline()`.

Usage

```
correct_drift_cubicspline(
  data = NULL,
  variable,
  ref_qc_types,
  batch_wise = TRUE,
  ignore_istd = TRUE,
  replace_previous = TRUE,
  cv = TRUE,
  spar = NULL,
  lambda = NULL,
  penalty = 1,
  conditional_correction = FALSE,
  recalc_trend_after = FALSE,
  log_transform_internal = TRUE,
  feature_list = NULL,
  cv_diff_threshold = 0,
  use_original_if_fail = FALSE,
  show_progress = TRUE
)
```

Arguments

data	MRMhubExperiment object
variable	The variable to be corrected for drift effects. Must be one of "intensity", "norm_intensity", or "conc"
ref_qc_types	QC types used for drift correction
batch_wise	Logical. Apply the correction to each batch separately (TRUE, default) or across all batches (FALSE).
ignore_istd	Logical. Exclude internal standards (ISTDs) from correction if TRUE.
replace_previous	Logical. Replace existing correction (TRUE, default) or layer on top of it (FALSE).
cv	Ordinary leave-one-out (TRUE) or 'generalized' cross-validation (GCV) when FALSE; is used for smoothing parameter computation only when spar is not specified
spar	Smoothing parameter for cubic spline smoothing. If not specified or NULL, the smoothing parameter is computed using the specified cv method. Typically (but not necessarily) (0,1].
lambda	Regularization parameter for cubic spline smoothing. Default is NULL, which means no regularization.
penalty	The coefficient of the penalty for degrees of freedom in the GCV criterion.
conditional_correction	Determines whether drift correction is applied to all features unconditionally (FALSE, the default) or, when TRUE, only conditionally based on sample CV change.

recalc_trend_after	Recalculate trend post-drift correction for <code>plot_runscatter()</code> . This will double calculation time.
log_transform_internal	Log transform the data for correction when TRUE (the default). Note: log transformation is solely applied internally for smoothing, results will not be log-transformed.
feature_list	Subset the features for correction whose names match the specified text using regular expression. Default is NULL.
cv_diff_threshold	Maximum allowable change (difference) in CV before and after smoothing for correction to be applied.
use_original_if_fail	Determines the action when smoothing fails or results in invalid values for a feature. If FALSE (default), the result for each feature will be NA for all batches, if TRUE, the original data is kept.
show_progress	Logical. Display progress bars if TRUE; disable for notebook rendering by setting to FALSE.

Details

In the output message, the median CV change is computed as the median of CV changes for all features in global correction or for features where the correction passed the defined CV difference threshold in case of conditional correction (`conditional_correction = TRUE`). For batch-wise correction, the change is calculated per batch, with the final median CV change being the median of these batch medians across features.

Value

`MRMhubExperiment` object

References

Dunn, W., Broadhurst, D., Begley, P. et al. Procedures for large-scale metabolic profiling of serum and plasma using gas chromatography and liquid chromatography coupled to mass spectrometry. *Nat Protoc* 6, 1060–1083 (2011). <https://doi.org/10.1038/nprot.2011.335>

Kirwan, J.A., Broadhurst, D.I., Davidson, R.L. et al. Characterising and correcting batch variation in an automated direct infusion mass spectrometry (DIMS) metabolomics workflow. *Anal Bioanal Chem* 405, 5147–5157 (2013). <https://doi.org/10.1007/s00216-013-6856-7>

See Also

`stats::smooth.spline()`; the [drift-correction tutorial](#) for a worked example.

correct_drift_gam	<i>Drift correction by generalized additive model (GAM) smoothing</i>
-------------------	---

Description

This function corrects for run-order drifts within or across batches using Generalized Additive Models (GAMs). The correction uses penalized splines, with automatic selection of smoothing parameters based on cross-validation or penalized likelihood. It is typically based on QC (Quality Control) samples measured at specific intervals throughout the run sequence. The correction can be applied to "intensity", "norm_intensity", or "conc" data.

It is recommended to visually inspect the correction using the `plot_runscatter()` function. Set the argument `recalc_trend_after = TRUE` so that the trends after correction are also available for plotting. For further details, refer to the description of `plot_runscatter()`.

The corrections can be applied on a batch-by-batch basis (`batch_wise = TRUE`, default) or across all batches (`batch_wise = FALSE`). Existing corrections are either replaced (`replace_previous = TRUE`) or added on top of them (`replace_previous = FALSE`).

Furthermore, drift correction can be applied unconditionally (`conditional_correction = FALSE`) or conditionally, based on whether the sample CV change before and after correction is below a defined threshold (`cv_diff_threshold`). This conditional correction is assessed independently for each batch if `batch_wise = TRUE`, where the median of the CV changes across the batch is compared with the threshold.

Note: The function outputs a message indicating the median CV change and the median absolute CV before and after correction for all samples. However, these metrics are experimental and should not be used as definitive criteria for correction (see Details below).

This method is implemented using penalized splines via the `mgcv::gam()` function.

Usage

```
correct_drift_gam(
  data = NULL,
  variable,
  ref_qc_types,
  batch_wise = TRUE,
  ignore_istd = TRUE,
  replace_previous = TRUE,
  bs = "ps",
  k = -1,
  sp = NULL,
  log_transform_internal = TRUE,
  conditional_correction = FALSE,
  recalc_trend_after = FALSE,
  feature_list = NULL,
  cv_diff_threshold = 0,
  use_original_if_fail = FALSE,
  show_progress = TRUE
)
```

Arguments

data	MRMhubExperiment object
variable	The variable to be corrected for drift effects. Must be one of "intensity", "norm_intensity", or "conc"
ref_qc_types	QC types used for drift correction
batch_wise	Logical. Apply the correction to each batch separately (TRUE, default) or across all batches (FALSE).
ignore_istd	Logical. Exclude internal standards (ISTDs) from correction if TRUE.
replace_previous	Logical. Replace existing correction (TRUE, default) or layer on top of it (FALSE).
bs	Basis type for the spline: "ps" (penalized spline, default) or "tp" (thin plate spline).
k	Number of basis functions (default: -1, automatically chosen by GAM).
sp	Smoothing parameter (NULL by default, estimated automatically).
log_transform_internal	Log transform the data for correction when TRUE (the default). Note: log transformation is solely applied internally for smoothing, results will not be log-transformed.
conditional_correction	Determines whether drift correction is applied to all features unconditionally (FALSE, the default) or, when TRUE, only conditionally based on sample CV change.
recalc_trend_after	Recalculate trend post-drift correction for plot_runcscatter(). This will double calculation time.
feature_list	Subset the features for correction whose names match the specified text using regular expression. Default is NULL.
cv_diff_threshold	Maximum allowable change (difference) in CV before and after smoothing for correction to be applied.
use_original_if_fail	Determines the action when smoothing fails or results in invalid values for a feature. If FALSE (default), the result for each feature will be NA for all batches, if TRUE, the original data is kept.
show_progress	Logical. Display progress bars if TRUE; disable for notebook rendering by setting to FALSE.

Details

In the output message, the median CV change is computed as the median of CV changes for all features in global correction or for features where the correction passed the defined CV difference threshold in case of conditional correction (`conditional_correction = TRUE`). For batch-wise correction, the change is calculated per batch, with the final median CV change being the median of these batch medians across features.

This smoothing is based on Generalized Additive Models (GAM) using penalized splines, implemented via `mgcv::gam()`.

Value

[MRMhubExperiment](#) object

See Also

[mgcv::gam\(\)](#); the [drift-correction tutorial](#) for a worked example.

correct_drift_gaussiankernel

Drift correction by Gaussian kernel smoothing

Description

Performs drift correction for run-order effects within or across batches using Gaussian kernel smoothing, as detailed in Teo et al. (2020). The Gaussian kernel estimates the local data trend, with bandwidth defined by the `kernel_size` parameter. This smoothing approach is mostly used with study samples and should only be applied to datasets with sufficiently randomized or stratified samples to avoid local biases and artifacts. The smoothing can be applied to concentration, norm_intensity, and intensity data.

Corrections can be applied on a batch-by-batch basis (`batch_wise = TRUE`, default) or across all batches (`batch_wise = FALSE`). The correction can either replace existing drift or batch corrections (`replace_previous = TRUE`, default) or applied on top of existing corrections (`replace_previous = FALSE`).

Drift correction can be applied to all features (`conditional_correction = FALSE`) or conditionally, based on whether the sample CV difference before and after correction is below a defined threshold (`cv_diff_threshold`). The conditional correction is applied separately for each batch if `batch_wise = TRUE`.

It is recommended to visually inspect the correction using the [plot_runscatter\(\)](#) function. Set the argument `recalc_trend_after = TRUE` so that the trends after correction are also available for plotting. For further details, refer to the description of [plot_runscatter\(\)](#). This will double the processing time.

Note: The function outputs a message indicating the median CV change and the median absolute CV before and after correction for all samples. However, these metrics are experimental and should not be used as definitive criteria for correction (see Details below).

Usage

```
correct_drift_gaussiankernel(  
  data = NULL,  
  variable,  
  ref_qc_types,  
  batch_wise = TRUE,  
  ignore_istd = TRUE,  
  replace_previous = TRUE,  
  kernel_size = 10,
```

```

outlier_filter = FALSE,
outlier_ksd = 5,
location_smooth = TRUE,
scale_smooth = FALSE,
log_transform_internal = TRUE,
conditional_correction = FALSE,
cv_diff_threshold = 0,
recalc_trend_after = FALSE,
feature_list = NULL,
use_original_if_fail = FALSE,
show_progress = TRUE
)

```

Arguments

data	A MRMhubExperiment object.
variable	The target variable for drift correction; options include "intensity", "norm_intensity", or "conc".
ref_qc_types	QC types used for drift correction, typically including study samples (SPL).
batch_wise	Logical. Apply the correction to each batch separately (TRUE, default) or across all batches (FALSE).
ignore_istd	Logical. Exclude internal standards (ISTDs) from correction if TRUE.
replace_previous	Logical. Replace existing correction (TRUE, default) or layer on top of it (FALSE).
kernel_size	Numeric. Defines the Gaussian kernel's bandwidth.
outlier_filter	Logical. Enable kernel outlier filtering if TRUE.
outlier_ksd	Numeric. Set the kernel's k times standard deviation for outlier detection.
location_smooth	Logical. Apply smoothing to the location parameter if TRUE.
scale_smooth	Logical. Apply smoothing to the scale parameter if TRUE.
log_transform_internal	Logical. Conduct log transformation internally for enhanced outlier robustness if TRUE (default); does not alter the output data.
conditional_correction	Determines whether drift correction is applied to all features unconditionally (FALSE, the default) or, when TRUE, only to features whose difference of sample CV before vs after smoothing is below the threshold specified by cv_diff_threshold.
cv_diff_threshold	This parameter defines the maximum allowable change (difference) in the coefficient of variation (CV) of samples before and after smoothing for the correction to be applied. A value of 0 (the default) requires the CV to improve, while a value above 0 allows the CV to also become worse by a maximum of the defined difference.
recalc_trend_after	Logical. Recalculate trends post-smoothing for visualization (e.g., <code>plot_runcscatter()</code>).

feature_list	Character vector. Regular expression pattern to select specific features for correction. Default is NULL for all features.
use_original_if_fail	Determines the action when smoothing fails or results in invalid values for a feature. If FALSE (default), the result for each feature will be NA for all batches, if TRUE, the original data is kept.
show_progress	Logical. Display progress bars if TRUE; disable for notebook rendering by setting to FALSE.

Details

In the output message, the median CV change is computed as the median of CV changes for all features in global correction or for features where the correction passed the defined CV difference threshold in case of conditional correction (`conditional_correction = TRUE`). For batch-wise correction, the change is calculated per batch, with the final median CV change being the median of these batch medians across features.

Value

Returns a `MRMhubExperiment` object.

References

Teo G., Chew WS, Burla B, Herr D, Tai ES, Wenk MR, Torta F, & Choi H (2020). MRMkit: Automated Data Processing for Large-Scale Targeted Metabolomics Analysis. *Analytical Chemistry*, 92(20), 13677–13682. <https://doi.org/10.1021/acs.analchem.0c03060>

correct_drift_loess	<i>Drift correction by LOESS smoothing</i>
---------------------	--

Description

This function corrects for run-order drifts within or across batches using LOESS (Locally Estimated Scatterplot Smoothing). The correction is typically based on QC (Quality Control) samples that were measured at specific intervals throughout the run sequence. The smoothed curve derived from the QC samples is then used to adjust all other samples in the dataset. The correction can be applied to "intensity", "norm_intensity", or "conc" data.

The degree of smoothing is controlled by the span parameter `span` (default is 0.75). Additionally, the degree parameter can be specified to control the degree of the polynomial used in the local regression (default is 2)

It is recommended to visually inspect the correction using the `plot_runscatter()` function. Set the argument `recalc_trend_after = TRUE` so that the trends after correction are also available for plotting. For further details, refer to the description of `plot_runscatter()`.

The LOESS correction only applies to samples that lie within the span of the QC samples used for smoothing. Extrapolation outside this range is not recommended, as it can lead to unreliable corrections or artefacts in the extrapolated regions. However, extrapolation can be activated by

setting `extrapolate = TRUE`. This may be useful in cases where specific drifts occur in segments of the analysis sequence that are not spanned by the QC samples, such as when the analysis was interrupted or the instrument had rapid changes in performance.

The corrections can be applied on a batch-by-batch basis (`batch_wise = TRUE`, default) or across all batches (`batch_wise = FALSE`). Existing corrections are either replaced (`replace_previous = TRUE`) or added on top of them (`replace_previous = FALSE`).

Furthermore, drift correction can be applied unconditionally (`conditional_correction = FALSE`) or conditionally, based on whether the sample CV change before and after correction is below a defined threshold (`cv_diff_threshold`). This conditional correction is assessed independently for each batch if `batch_wise = TRUE`, where the median of the CV changes across the batch is compared with the threshold.

Note: The function outputs a message indicating the median CV change and the median absolute CV before and after correction for all samples. However, these metrics are experimental and should not be used as definitive criteria for correction (see Details below).

This LOESS method is implemented using the base R function `stats::loess()`.

Usage

```
correct_drift_loess(
  data = NULL,
  variable,
  ref_qc_types,
  span = 0.75,
  degree = 2,
  batch_wise = TRUE,
  ignore_istd = TRUE,
  replace_previous = TRUE,
  conditional_correction = FALSE,
  recalc_trend_after = FALSE,
  log_transform_internal = TRUE,
  feature_list = NULL,
  cv_diff_threshold = 0,
  use_original_if_fail = FALSE,
  extrapolate = FALSE,
  show_progress = TRUE
)
```

Arguments

<code>data</code>	<code>MRMhubExperiment</code> object
<code>variable</code>	The variable to be corrected for drift effects. Must be one of "intensity", "norm_intensity", or "conc"
<code>ref_qc_types</code>	QC types used for drift correction
<code>span</code>	Loess span width (default is 0.75)
<code>degree</code>	Degree of the polynomial to be used in the loess smoothing, normally 1 or 2. Default is 2.

batch_wise	Logical. Apply the correction to each batch separately (TRUE, default) or across all batches (FALSE).
ignore_istd	Logical. Exclude internal standards (ISTDs) from correction if TRUE.
replace_previous	Logical. Replace existing correction (TRUE, default) or layer on top of it (FALSE).
conditional_correction	Determines whether drift correction is applied to all features unconditionally (FALSE, the default) or, when TRUE, only to features whose difference of sample CV before vs after smoothing is below the threshold specified by cv_diff_threshold.
recalc_trend_after	Recalculate trend post-drift correction for plot_runcscatter(). This will double calculation time.
log_transform_internal	Log transform the data for correction when TRUE (the default). Note: log transformation is solely applied internally for smoothing, results will not be log-transformed. Log transformation may result in more robust smoothing that is less sensitive to outliers.
feature_list	Subset the features for correction whose names matches the specified text using regular expression. Default is NULL which means all features are selected.
cv_diff_threshold	This parameter defines the maximum allowable change (difference) in the coefficient of variation (CV) of samples before and after smoothing for the correction to be applied. A value of 0 (the default) requires the CV to improve, while a value above 0 allows the CV to also become worse by a maximum of the defined difference.
use_original_if_fail	Determines the action when smoothing fails or results in invalid values for a feature. If FALSE (default), the result for each feature will be NA for all batches, if TRUE, the original data is kept.
extrapolate	Extrapolate loess smoothing. WARNING: It is generally not recommended to extrapolate outside of the range spanned by the QCs used for smoothing. See details below.
show_progress	Logical. Display progress bars if TRUE; disable for notebook rendering by setting to FALSE.

Details

In the output message, the median CV change is computed as the median of CV changes for all features in global correction or for features where the correction passed the defined CV difference threshold in case of conditional correction (`conditional_correction = TRUE`). For batch-wise correction, the change is calculated per batch, with the final median CV change being the median of these batch medians across features.

Value

[MRMhubExperiment](#) object

See Also

The [drift-correction tutorial](#) for a worked example.

correct_interference_manual

Manual isotopic interference correction

Description

Subtract a single, user-specified interference from one feature. Use this for a one-off correction or to validate a factor before trusting the automatic derivation; for the metadata-driven paths use [correct_isotopic_interferences\(\)](#) (automatic, isotopic) or [correct_custom_interferences\(\)](#) (declared). The interference is subtracted as:

$$Value_{Corrected} = Value_{Feature} - Factor_{Contribution} * Value_{InterferingFeature}$$

Usage

```
correct_interference_manual(
  data = NULL,
  variable,
  feature,
  interfering_feature,
  interference_contribution,
  neg_to_na = FALSE,
  updated_feature_id = NA
)
```

Arguments

data	MRMhubExperiment object
variable	Name of the variable to be corrected, e.g. feature_intensity.
feature	Name of feature to be corrected
interfering_feature	Name of feature that is interfering, i.e. contributing to the signal of feature
interference_contribution	Relative portion of the interfering feature contributing to the feature signal. Must be greater than 0; values are usually between 0 and 1, and values above 1 trigger a warning.
neg_to_na	If TRUE, negative or zero values after correction will be replaced with NA. Default: FALSE.
updated_feature_id	Optional. New name of corrected feature. If empty then feature name will not change.

Value

[MRMhubExperiment](#) object

correct_isotopic_interferences

Correct automatically derived isotopic interferences

Description

Applies the isotopic (M+2) interference corrections previously discovered by [calc_isotopic_interferences\(\)](#) (and any declared interferences, see [correct_custom_interferences\(\)](#)) to the raw feature intensities. Aborts with guidance if no interferences have been derived yet.

The subtraction is

$$value_{corrected} = value_{raw} - K \cdot value_{interferer}$$

applied on the raw feature_intensity. For auto-derived (source == "auto") edges the interferer is clamped at 0 before subtraction and the result clamped at 0 (LICAR parity); declared ("manual") edges are unclamped. See [calc_isotopic_interferences\(\)](#) for how K is computed.

A chain (e.g. PC 34:2 > PC 34:1 > PC 34:0) is corrected sequentially by default, each feature using its already-corrected interferer; set sequential_correction = FALSE to correct each from the raw interferer. The raw signal is preserved in feature_intensity_orig, and the correction is idempotent (re-running restores from raw first).

Usage

```
correct_isotopic_interferences(
  data = NULL,
  variable = "feature_intensity",
  sequential_correction = TRUE,
  neg_to_na = FALSE
)
```

Arguments

data	MRMhubExperiment object.
variable	Name of the variable to correct. Only "feature_intensity" (the raw intensity) is supported. Default: "feature_intensity".
sequential_correction	Logical. If TRUE (default), a chain of interferences is corrected sequentially so each feature uses the already-corrected signal of its interferer (propagates along the chain). If FALSE, each feature is corrected from the raw interferer, without propagation.
neg_to_na	If TRUE, negative or zero values after correction are replaced with NA. Default: FALSE.

Value

[MRMhubExperiment](#) object with feature intensities corrected.

References

Gao L., Ji S, Burla B, Wenk MR, Torta F, & Cazenave-Gassiot A (2021). LICAR: An Application for Isotopic Correction of Targeted Lipidomic Data Acquired with Class-Based Chromatographic Separations Using Multiple Reaction Monitoring. *Analytical Chemistry*, 93(6), 3163-3171. <https://doi.org/10.1021/acs.analchem.0c04565>

See Also

The [interference-correction guide](#).

cv	<i>Percent coefficient of variation (%CV)</i>
----	---

Description

This function computes the percent coefficient of variation of the values in x. If na.rm is TRUE then missing values are removed before computation proceeds.

Usage

```
cv(x, na.rm = FALSE, use_robust_cv = FALSE, min_n = 1L)
```

Arguments

x	a numeric vector with untransformed data
na.rm	logical, if TRUE then NA values are stripped from x before computation takes place
use_robust_cv	logical, if TRUE the robust coefficient of variation (scaled median absolute deviation / median) is computed instead of the standard CV (SD / mean)
min_n	integer, the minimum number of observations (non-missing ones when na.rm = TRUE) required to return a CV. Fewer than min_n values return NA_real_. The default (1) is a no-op, since sd()/mad() already return NA for fewer than two values.

Value

a numeric value. If the mean (or the median, for robust CV) is zero, or x is not numeric, NA_real_ is returned

Examples

```
cv(c(5, 6, 3, 4, 5, NA), na.rm = TRUE)
```

cv_log	<i>Percent coefficient of variation (%CV) based on log-transformation</i>
--------	---

Description

Computes the percent coefficient of variation (CV) based on the log-transformation of the input data. This can be more robust in certain cases, especially when the data is not normally distributed.

Usage

```
cv_log(x, na.rm = FALSE)
```

Arguments

x	a numeric vector with untransformed data
na.rm	logical, if TRUE then NA values are stripped from x before computation takes place

Value

a numeric value. If x contains a zero, then NaN is returned

References

Canchola et al. (2017) Correct use of percent coefficient of variation (% CV) formula for log-transformed data. MOJ Proteomics Bioinform. 2017;6(4):316–317. DOI: [10.15406/mojpb.2017.06.00200](https://doi.org/10.15406/mojpb.2017.06.00200)

Examples

```
cv_log(c(5, 6, 3, 4, 5, NA), na.rm = TRUE)
```

data_load_example	<i>Load an example MRMhubExperiment dataset</i>
-------------------	---

Description

Load an example MRMhubExperiment dataset: a small, preprocessed subset of a plasma lipidomics dataset (Tan et al., ATVB, 2022).

Usage

```
data_load_example(data = NULL, dataset = 1)
```

Arguments

data `MRMhubExperiment` object, optional. Data will be overwritten if provided.
dataset Which example dataset to load. Currently only 1 (the default) is available.

Value

`MRMhubExperiment` object

Examples

```
myexp <- MRMhubExperiment()  
myexp <- data_load_example(myexp)  
myexp
```

data_sum_features	<i>Sum up feature intensities per analyte</i>
-------------------	---

Description

This function sums up feature intensities per analyte_id.

This is useful when you have multiple features (e.g. adducts, isotopes, in-source fragments) or isomers that you want to combine into a single analyte intensity value, such as LPC sn1 and sn2 species.

Usage

```
data_sum_features(data, qualifier_action = "include")
```

Arguments

data `MRMhubExperiment` object
qualifier_action Character. How to handle qualifier features. To sum them up separately select "separate", to include them in the sum if quantifier select "include", to not sum them up select "exclude".

Details

Features are summed when they share an analyte_id (an empty one counts as missing): with `qualifier_action = "include"` all of them, with `"separate"` quantifiers and qualifiers each into their own feature (the qualifier sum is named `<analyte_id>_qual`), and with `"exclude"` only the quantifiers, while qualifiers are kept as they are. An analyte with a single feature in its group keeps its feature_id. Only features present in the dataset are summed: excluded features and features listed only in the metadata keep their feature_id and do not affect the sum.

Only raw signal variables are aggregated across the transitions of an analyte: `feature_intensity`, `feature_height` and `feature_area` are summed, and `feature_rt` is averaged. A sum is NA

in an analysis where a constituent is missing, since a partial sum would look like a valid value; a warning reports these analyses. A transition without a value in any analysis is left out of its sum. `feature_fwhm`, `feature_width`, `feature_int_start` and `feature_int_end` are set to NA for merged analytes: the constituents are separate chromatographic peaks, so no aggregate of their peak widths or borders describes the merged quantity.

Summing transitions redefines `feature_intensity`, so all values *derived* from the pre-merge intensities are invalidated and removed: normalized intensities, concentrations, drift/batch correction results and QC metrics. Re-run `normalize_by_istd()` and the quantitation/correction steps after merging. A message reports this when such values were present.

The summed features must be measured and quantified alike: an error is raised when they combine internal standards with analytes, or differ in `istd_feature_id`, `quant_istd_feature_id`, `response_factor` or `interference_feature_id`. An error is also raised when a summed id equals the `feature_id` of another feature. Transitions of one internal standard can be summed; references to summed features in the feature, ISTD and interference metadata are updated, and interferences between transitions summed into one feature are removed. Other metadata (`feature_class`, `feature_label`) comes from the first constituent, with a warning when the constituents disagree.

`is_quantifier` is not inherited but determined by the merge: the merged analyte is a quantifier if any of its constituents is one.

Value

`MRMhubExperiment` object

Experimental

This function is **experimental** and its behaviour may change. It overwrites the `feature_id` of features sharing an `analyte_id` in both the dataset and the feature metadata, and the original `feature_id` is not backed up anywhere. Run it after importing metadata and after any exclusions, `set_analysis_order()` or `set_intensity_var()`, and before normalization/quantitation: these steps rebuild the dataset from the imported data and stop with an error once features were summed. Running it on a processed object drops the derived variables (see Details).

<code>detect_outlier_pca</code>	<i>Get list of analyses classified as technical outliers</i>
---------------------------------	--

Description

Retrieves analysis IDs of data outliers based on the principal components (PCA) with SD or MAD fences

Usage

```
detect_outlier_pca(
  data = NULL,
  variable,
  filter_data,
```

```

    pca_component,
    qc_types = NA,
    fence_multiplier,
    summarize_fun = c("pca", "rma"),
    outlier_detection = c("sd", "mad"),
    log_transform = TRUE
  )

```

Arguments

<code>data</code>	<code>MRMhubExperiment</code> object
<code>variable</code>	Feature variable used for outlier detection
<code>filter_data</code>	Use all (default) or qc-filtered data
<code>pca_component</code>	PCA component to be used
<code>qc_types</code>	QC types included in the outlier detection
<code>fence_multiplier</code>	Multiplier for SD or MAD.
<code>summarize_fun</code>	Function used to summarize the features, either "pca" based on PCA, or "rma" based on Relative Mean Abundance (RMA) of all features
<code>outlier_detection</code>	Outlier detection method, either based on "sd" or "mad"
<code>log_transform</code>	Log-transform data for outlier detection

Value

`MRMhubExperiment` object

<code>exclude_analyses</code>	<i>Exclude analyses from the dataset</i>
-------------------------------	--

Description

This function excludes specified analyses from a `MRMhubExperiment` object, either by marking them as invalid for downstream processing. The function also allows to reset the exclusions.

Usage

```
exclude_analyses(data = NULL, analyses, clear_existing)
```

Arguments

<code>data</code>	A <code>MRMhubExperiment</code> object
<code>analyses</code>	A character vector of analysis IDs (case-sensitive) to be excluded from the dataset. If this is NA or an empty vector, the exclusion behavior will be handled as set via the <code>clear_existing</code> flag.
<code>clear_existing</code>	A logical value. If TRUE, existing <code>valid_analysis</code> flags will be overwritten. If FALSE, the exclusions will be appended, preserving any existing invalidated analyses.

Value

A modified `MRMhubExperiment` object with the specified analyses defined as excluded.

exclude_features	<i>Exclude features from the dataset</i>
------------------	--

Description

This function excludes specified features from a `MRMhubExperiment` object, either by marking them as invalid for downstream processing. The function also allows to reset the exclusions.

Usage

```
exclude_features(data = NULL, features, clear_existing)
```

Arguments

- | | |
|----------------|---|
| data | A <code>MRMhubExperiment</code> object |
| features | A character vector of feature IDs (case-sensitive) to be excluded from the dataset. If this is NA or an empty vector, the exclusion behavior will be handled as set via the <code>clear_existing</code> flag. |
| clear_existing | A logical value. If TRUE, existing <code>valid_feature</code> flags will be overwritten. If FALSE, the exclusions will be appended, preserving any existing invalidated features |

Value

A modified `MRMhubExperiment` object with the specified features defined as excluded.

filter_features_qc	<i>Feature filtering based on QC criteria</i>
--------------------	---

Description

Filters a dataset based on quality control (QC) criteria, including intensity, coefficient of variation (CV), signal-to-blank ratios, D-ratio, response curve properties, and proportion of missing values. Criteria apply to different QC types (BQC, TQC) and measurement variables (concentration, intensity, and normalized intensity).

To clear all existing filters, run `filter_features_qc()` with `clear_existing = TRUE` and without any filtering criteria.

Usage

```
filter_features_qc(  
  data = NULL,  
  clear_existing = TRUE,  
  recalc_metrics = FALSE,  
  use_batch_medians = FALSE,  
  use_robust_cv = FALSE,  
  include_qualifier = FALSE,  
  include_istd = FALSE,  
  features.to.keep = NA,  
  max.prop.missing.intensity.spl = NA,  
  max.prop.missing.normintensity.spl = NA,  
  max.prop.missing.conc.spl = NA,  
  min.intensity.lowest.bqc = NA,  
  min.intensity.lowest.tqc = NA,  
  min.intensity.lowest.spl = NA,  
  min.intensity.median.bqc = NA,  
  min.intensity.median.tqc = NA,  
  min.intensity.median.spl = NA,  
  min.intensity.highest.spl = NA,  
  min.signalblank.median.spl.pblk = NA,  
  min.signalblank.median.spl.ublk = NA,  
  min.signalblank.median.spl.sblk = NA,  
  max.cv.intensity.bqc = NA,  
  max.cv.intensity.tqc = NA,  
  max.cv.normintensity.bqc = NA,  
  max.cv.normintensity.tqc = NA,  
  max.cv.conc.bqc = NA,  
  max.cv.conc.tqc = NA,  
  response.curves.selection = NA,  
  response.curves.summary = NA,  
  min.rsquare.response = NA,  
  min.slope.response = NA,  
  max.slope.response = NA,  
  max.yintercept.response = NA,  
  max.dratio.sd.conc.bqc = NA,  
  max.dratio.sd.conc.tqc = NA,  
  max.dratio.mad.conc.bqc = NA,  
  max.dratio.mad.conc.tqc = NA,  
  max.dratio.sd.normint.bqc = NA,  
  max.dratio.sd.normint.tqc = NA,  
  max.dratio.mad.normint.bqc = NA,  
  max.dratio.mad.normint.tqc = NA  
)
```

Arguments

data [MRMhubExperiment](#) object.

clear_existing	Logical. If TRUE, replaces any existing filters; if FALSE, adds new filters on top of existing ones. Default is TRUE.
recalc_metrics	Logical. If TRUE, recalculates QC metrics before filtering. Default is FALSE.
use_batch_medians	Logical. If TRUE, uses batch-wise median QC values for filtering. Default is FALSE, or the setting of existing QC metrics; a different explicit value recalculates them.
use_robust_cv	Logical. If TRUE, uses robust coefficient of variation (MAD/median) instead of standard CV (SD/mean). Default is FALSE, or the setting of existing QC metrics; a different explicit value recalculates them.
include_qualifier	Logical. If TRUE, includes qualifier features in the filtering process.
include_istd	Logical. If TRUE, includes internal standards (ISTDs) in the filtering process.
features.to.keep	A vector of feature identifiers to retain, even if they do not meet the filtering criteria.
max.prop.missing.intensity.spl	Maximum proportion of missing intensity values among study samples (SPL). Default is NA.
max.prop.missing.normintensity.spl	Maximum proportion of missing normalized intensity values among study samples (SPL). Default is NA.
max.prop.missing.conc.spl	Maximum proportion of missing concentration values among study samples (SPL). Default is NA.
min.intensity.lowest.bqc	Minimum intensity of the lowest BQC sample. Default is NA.
min.intensity.lowest.tqc	Minimum intensity of the lowest TQC sample. Default is NA.
min.intensity.lowest.spl	Minimum intensity of the lowest study sample (SPL). Default is NA.
min.intensity.median.bqc	Minimum median intensity of BQC samples. Default is NA.
min.intensity.median.tqc	Minimum median intensity of TQC samples. Default is NA.
min.intensity.median.spl	Minimum median intensity of study samples (SPL). Default is NA.
min.intensity.highest.spl	Minimum intensity of the highest intensity study sample (SPL). Default is NA.
min.signalblank.median.spl.pblk	Minimum signal-to-blank ratio for SPL samples and PBLK. Default is NA.
min.signalblank.median.spl.ublk	Minimum signal-to-blank ratio for SPL samples and UBLK. Default is NA.

`min.signalblank.median.spl.sblk`
 Minimum signal-to-blank ratio for SPL samples and SBLK. Default is NA. For all signal-to-blank criteria, a feature not detected in a blank (missing or zero intensity) has a blank median of zero, i.e. a ratio of Inf, and passes; a feature not detected in the study samples fails. A criterion for a blank type without analyses in the dataset raises an error.

`max.cv.intensity.bqc`
 Maximum CV for intensity in BQC samples. Default is NA.

`max.cv.intensity.tqc`
 Maximum CV for intensity in TQC samples. Default is NA.

`max.cv.normintensity.bqc`
 Maximum CV for normalized intensity in BQC samples. Default is NA.

`max.cv.normintensity.tqc`
 Maximum CV for normalized intensity in TQC samples. Default is NA.

`max.cv.conc.bqc`
 Maximum CV for concentration in BQC samples. Default is NA.

`max.cv.conc.tqc`
 Maximum CV for concentration in TQC samples. Default is NA.

`response.curves.selection`
 Select specific response curves by ID. Default is NA.

`response.curves.summary`
 Define the method to summarize multiple response curves. Default is NA.

`min.rsquare.response`
 Minimum R² value for the response curves. Default is NA.

`min.slope.response`
 Minimum slope for the response curve. Default is NA.

`max.slope.response`
 Maximum slope for the response curve. Default is NA.

`max.yintercept.response`
 Maximum y-intercept of the response curve. Default is NA.

`max.dratio.sd.conc.bqc`
 Maximum allowed D-ratio (SD of BQC / SD of SPL) using standard deviation for BQC samples. Default is NA.

`max.dratio.sd.conc.tqc`
 Maximum allowed D-ratio (SD of TQC / SD of SPL) using standard deviation for TQC samples. Default is NA.

`max.dratio.mad.conc.bqc`
 Maximum allowed D-ratio (MAD of BQC / MAD of SPL) using median absolute deviation for BQC samples. Default is NA.

`max.dratio.mad.conc.tqc`
 Maximum allowed D-ratio (MAD of TQC / MAD of SPL) using median absolute deviation for TQC samples. Default is NA.

`max.dratio.sd.normint.bqc`
 Maximum allowed D-ratio (SD of normalized intensity in BQC / SD of SPL) using standard deviation. Default is NA.

max.dratio.sd.normint.tqc
Maximum allowed D-ratio (SD of normalized intensity in TQC / SD of SPL) using standard deviation. Default is NA.

max.dratio.mad.normint.bqc
Maximum allowed D-ratio (MAD of normalized intensity in BQC / MAD of SPL) using median absolute deviation. Default is NA.

max.dratio.mad.normint.tqc
Maximum allowed D-ratio (MAD of normalized intensity in TQC / MAD of SPL) using median absolute deviation. Default is NA.

Details

This function implements filtering criteria based on quality control (QC) samples and additional analytical parameters, following recommendations outlined by Broadhurst et al. (2018). The implemented criteria evaluate data quality through analysis of QC samples, blanks, and study samples.

Value

The input `MRMhubExperiment` object with the feature filtering criteria applied. Per-criterion verdicts are stored in `metrics_qc`: `pass_minint` (the `min.intensity.*` criteria), `pass_sb`, `pass_cva`, `pass_dratio`, `pass_linearity` and `pass_missingval`, combined in `all_filter_pass`. With a response-curve criterion, a feature without response-curve results fails `pass_linearity`; ISTDs without results are not failed.

References

Broadhurst, D., Goodacre, R., Reinke, S. N., Kuligowski, J., Wilson, I. D., Lewis, M. R., & Dunn, W. B. (2018). Guidelines and considerations for the use of system suitability and quality control samples in mass spectrometry assays applied in clinical studies. *Metabolomics*, 14(6), 72. doi:10.1007/s1130601813673

fun_cspline

Cubic spline smoothing helper function

Description

Function for cubic spline-based smoothing with optional cross-validation, for use by `correct_drift`

Usage

```
fun_cspline(tbl, ref_qc_types, log_transform_internal, ...)
```

Arguments

tbl	Table (tibble or data.frame) containing the fields qc_type, x (run order number), and y (variable)
ref_qc_types	QC types used for the smoothing (fit) by cubic spline
log_transform_internal	Apply log transformation internally for smoothing if TRUE (default). This does not affect the final data, which remains untransformed.
...	Additional arguments forwarded to smooth.spline

Value

List with a data.frame containing original x and the smoothed y values, and a boolean value indicating whether the fit failed or not.

fun_gam_smooth	<i>Generalized additive model (GAM) smoothing helper function</i>
----------------	---

Description

Function for penalized spline-based smoothing using GAM, for use by correct_drift

Usage

```
fun_gam_smooth(tbl, ref_qc_types, log_transform_internal = TRUE, ...)
```

Arguments

tbl	Table (tibble or data.frame) containing the fields qc_type, x (run order number), and y (variable)
ref_qc_types	QC types used for the smoothing (fit) by GAM
log_transform_internal	Apply log transformation internally for smoothing if TRUE (default). This does not affect the final data, which remains untransformed.
...	Additional arguments forwarded to mgcv::gam

Value

List with a data.frame containing original x and the smoothed y values, and a boolean value indicating whether the fit failed or not.

fun_gauss.kernel.smooth

Gaussian kernel smoothing helper function

Description

Function for Gaussian kernel-based smoothing, for use by correct_drift.

Usage

```
fun_gauss.kernel.smooth(tbl, ref_qc_types, log_transform_internal, ...)
```

Arguments

tbl	Table (tibble or data.frame) containing the fields qc_type, x (run order number), and y (variable)
ref_qc_types	QC types used for the smoothing (fit) by loess
log_transform_internal	Apply log transformation internally for smoothing if TRUE (default). This does not affect the final data, which remains untransformed.
...	Additional arguments

Value

List with a data.frame containing original x and the smoothed y values, and a boolean value indicting whether the fit failed or not.

Author(s)

Hyung Won Choi

fun_loess

Loess smoothing helper function

Description

Function for loess-based smoothing, for use by correct_drift

Usage

```
fun_loess(tbl, ref_qc_types, log_transform_internal, ...)
```

Arguments

<code>tbl</code>	Table (tibble or <code>data.frame</code>) containing the fields <code>qc_type</code> , <code>x</code> (run order number), and <code>y</code> (variable)
<code>ref_qc_types</code>	QC types used for the smoothing (fit) by loess
<code>log_transform_internal</code>	Apply log transformation internally for smoothing if TRUE (default). This does not affect the final data, which remains untransformed.
<code>...</code>	Additional arguments forwarded to Loess

Value

List with a `data.frame` containing original `x` and the smoothed `y` values, and a boolean value indicating whether the fit failed or not.

`generate_workflow_qmd` *Generate a runnable Quarto (.qmd) mrmhub workflow*

Description

Assembles a self-contained Quarto document that imports data and metadata and runs the selected processing steps, in canonical order, using the same function calls and defaults as the package tutorials. This is a pure function (no file I/O, no Shiny) so it can be tested and scripted directly; the interactive `build_workflow()` app calls it to power its live preview.

Usage

```
generate_workflow_qmd(spec)
```

Arguments

<code>spec</code>	A list describing the workflow. Recognised fields: <code>importer</code> One of "mrmhub", "masshunter", "skyline", "csv_long", "csv_wide". Default "mrmhub". <code>data_path</code> Path to the data file as it should appear in the generated document. <code>metadata_route</code> "embedded", "msorganiser", "tables", or "none". Default "embedded". <code>metadata_path</code> Path to the metadata file (when not embedded). <code>steps</code> Character vector of step ids to include (the ids defined in <code>workflow_steps()</code>). <code>drift_method</code> Drift model for the drift step: "gaussian" (default), "spline", or "loess". <code>ref_qc_types</code> , <code>reference_sample_id</code> Optional reference QC type(s) and reference sample id for drift/batch/calibration. The corrected feature variable is derived automatically as the highest processing level reached by the selected steps.
-------------------	---

column_mapping Named character vector for importer = "csv_long", mapping canonical names to file columns, e.g. c(analysis_id = "Sample", feature_id = "Compound", feature_area = "Area").

variable_name, analysis_id_col, first_feature_column Arguments for importer = "csv_wide".

output_xlsx Path for the exported report. Default "results.xlsx".

formats Character vector of Quarto output formats, any of "html", "docx", "pdf". "pdf" is rendered sans-serif. Default "html".

save_rds Logical; also save the object as .rds. Default FALSE.

title Document title. Default "MRMhub Workflow".

Value

A length-1 character string containing the .qmd source.

Examples

```
cat(generate_workflow_qmd(list(
  importer = "mrmhub",
  data_path = "MRMhub_demo.tsv",
  steps = c("normalize_istd", "quantify_istd")
)))
```

get_analys_end	<i>Get the end time of the analysis sequence</i>
----------------	--

Description

Returns the end time of the analysis, corresponding to the last acquisition_time_stamp from the dataset. Note: if estimate_sequence_end is set to FALSE, the function will return the timestamp of the last analysis in the dataset, corresponding to the start of the last analysis. Set estimate_sequence_end to TRUE to estimate the end time of the analysis sequence, based on the median runtime.

Usage

```
get_analys_end(data, estimate_sequence_end)
```

Arguments

data A [MRMhubExperiment](#) object

estimate_sequence_end If TRUE, the function will estimate the end time of the analysis sequence based on the median runtime. FALSE will return the start time of the last analysis in the sequence.

Value

A POSIXct timestamp, or NA_POSIXct_ if the dataset is empty.

get_analyis_start	<i>Get the start time of the analysis sequence</i>
-------------------	--

Description

Returns the start time of the analysis, corresponding to the earliest acquisition_time_stamp from the dataset.

Usage

```
get_analyis_start(data)
```

Arguments

data A [MRMhubExperiment](#) object

Value

A POSIXct timestamp, or NA_POSIXct_ if the dataset is empty.

get_analysis_breaks	<i>Get the number of analysis breaks in the analysis</i>
---------------------	--

Description

Counts the number of interruptions in the analysis, where an interruption is defined as a time gap between consecutive acquisition timestamps that exceeds a given threshold (break_duration_minutes).

Usage

```
get_analysis_breaks(data, break_duration_minutes)
```

Arguments

data A [MRMhubExperiment](#) object

break_duration_minutes

A numeric value specifying the minimum duration (in minutes) between two consecutive analyses that qualifies as an interruption.

Value

An integer with the number of interruptions, or NA_integer_ if the dataset is empty.

get_analysis_count	<i>Get the number of analyses in the dataset</i>
--------------------	--

Description

Returns the number of analyses in the dataset, with an optional filter based on qc_types.

Usage

```
get_analysis_count(data, qc_types = NULL)
```

Arguments

data	A MRMhubExperiment object
qc_types	Defines the qc_types to be counted. If NULL or NA, all analyses will be counted.

Value

An integer with the analysis count

get_analysis_duration	<i>Get the total duration of the analysis</i>
-----------------------	---

Description

This function returns the total duration of the analysis, which is the time difference between the timestamps of the first and last analyses in the sequence.

Usage

```
get_analysis_duration(data, estimate_sequence_end)
```

Arguments

data	A MRMhubExperiment object
estimate_sequence_end	If TRUE, the function will estimate the end time of the analysis sequence based on the median runtime, added to the timestamp of the last analysis. If FALSE, the function will calculate the time difference between the first and last analysis timestamps without any adjustment.

Details

If estimate_sequence_end is TRUE, the function will estimate the end time of the analysis sequence by adding the median runtime to the timestamp of the last analysis, instead of simply using the timestamp of the last analysis.

get_analyticaldata	<i>Get the annotated or the originally imported analytical data</i>
--------------------	---

Description

Get the annotated or the originally imported analytical data

Usage

```
get_analyticaldata(data = NULL, annotated)
```

Arguments

data	MRMhubExperiment object
annotated	Boolean indicating whether to return the annotated data (TRUE) or the original imported data (FALSE)

Value

A tibble with the analytical data in the long format

get_batch_boundaries	<i>Get the start and end analysis numbers of specified batches</i>
----------------------	--

Description

Returns the lower and upper analysis number (sequence position) spanned by the specified batch or range of batches.

Usage

```
get_batch_boundaries(data = NULL, batch_indices = NULL)
```

Arguments

data	MRMhubExperiment object
batch_indices	A numeric vector with one or two elements, representing the first and/or last batch index (i.e., sequential batch number). If NULL or invalid, the function will abort.

Value

A vector with two elements: the lower and upper analysis number for the specified batch(es).

```
get_calibration_metrics
```

Get calibration metrics

Description

Extracts calibration fit metrics from a `MRMhubExperiment` object.

Usage

```
get_calibration_metrics(  
  data = NULL,  
  include_qualifier = TRUE,  
  with_lod = TRUE,  
  with_loq = TRUE,  
  with_coefficients = TRUE,  
  with_sigma = TRUE,  
  summary_table = FALSE  
)
```

Arguments

<code>data</code>	A <code>MRMhubExperiment</code> object with QC metrics.
<code>include_qualifier</code>	Whether to include qualifier features. When FALSE, only quantifier features are returned. Default is TRUE.
<code>with_lod</code>	Whether to include LoD in output. Default is TRUE.
<code>with_loq</code>	Whether to include LoQ in output. Default is TRUE.
<code>with_coefficients</code>	Whether to include regression coefficients. Default is TRUE.
<code>with_sigma</code>	Whether to include sigma in output. Default is TRUE.
<code>summary_table</code>	When TRUE, return a compact, display-formatted summary tibble (analyte, fit_model, fit_weighting, r2, lod, loq) with r2 rounded to 5 decimals and lod/loq to 3 significant figures. This is a presentation view for reporting; the values are rounded , so do not use it for downstream computation. lod/loq follow with_lod/with_loq. The with_coefficients/with_sigma flags are ignored in this mode. Default is FALSE (full, unrounded metrics).

Details

Requires prior computation of regression results using `calc_calibration_results()`. See its documentation for details.

Returned Details and Metrics:

- `feature_id`: Feature identifier.

- `is_quantifier`: Logical, indicates if the feature is a quantifier.
- `fit_model`: Regression model used for fitting.
- `fit_weighting`: Weighting method used in fitting.
- `lowest_cal`: Lowest non-zero calibrator concentration with a response.
- `highest_cal`: Highest calibrator concentration with a response.
- `r2`: R^2 value, indicating goodness of fit. For a **weighted** fit this is the weighted coefficient of determination (computed from weighted sums of squares), matching the value reported by vendor software such as Agilent MassHunter for the same weighted curve.
- `coef_a`: **Intercept** (0th-order term, x^0) of the fitted curve.
- `coef_b`: **1st-order** (linear, x^1) coefficient — the slope in a **linear** fit, or the linear term in a **quadratic** fit.
- `coef_c`: **2nd-order** (quadratic, x^2) coefficient in **quadratic** fits. Returns NA for **linear** fits.
- `sigma`: Standard deviation of residuals.
- `reg_failed`: TRUE if regression fitting failed.
- `LoD` = $3.3 \times$ the sample standard error of residuals / slope of the regression (see Notes).
- `LoQ` = $10 \times$ the sample standard error of residuals / slope of the regression (see Notes).

The coefficients describe the curve in **ascending** power order (R's `lm()` / `poly()` convention): $\text{response} = \text{coef_a} + \text{coef_b} * x + \text{coef_c} * x^2$. This is the reverse of the descending $a * x^2 + b * x + c$ form used by some vendor software (e.g. Agilent MassHunter, where *a* is the x^2 term), so when comparing against such an export, match by power, not by letter.

Note: LoD/LoQ follow the ICH Q2(R1/R2) approach ($3.3 \text{ sigma} / S$ and $10 \text{ sigma} / S$). The slope *S* is the slope of the calibration curve at zero concentration (the linear coefficient `coef_b`); for a **quadratic** fit the quadratic term does not contribute to this slope. The response sigma is selectable in `calc_calibration_results()` via `lod_sigma` (residual standard error, the default, or the standard error of the intercept); the sigma column reported here is always the residual standard error.

For a **weighted** fit ($1/x$, $1/x^2$, $1/\sqrt{x}$) sigma is R's weighted residual standard error, which is not on the raw response scale that the ICH $3.3 \text{ sigma} / S$ formula assumes, so the reported LoD/LoQ are approximate (typically slightly optimistic for $1/x$). Use `fit_weighting = "none"` if you require the strict ICH response-scale Sy/x ; the back-calculated concentrations themselves are unaffected.

Value

A tibble with exported calibration metrics.

<code>get_feature_count</code>	<i>Get the number of features in the dataset</i>
--------------------------------	--

Description

Returns the number of features in the dataset, with optional filters whether counted features must be internal standard and/or quantifier.

Usage

```
get_feature_count(data, is_istd = NA, is_quantifier = NA)
```

Arguments

data	A MRMhubExperiment object
is_istd	If set, then defines whether to include or exclude internal standard features. Default is NA means no filter for internal standards is applied.
is_quantifier	If set, then defines whether to include or exclude quantifier features. Default is NA means no filter for quantifier features is applied.

Value

An integer with the feature count

get_featurelist	<i>Get feature IDs</i>
-----------------	------------------------

Description

Returns a vector of annotated feature IDs (feature_id) present in the dataset

Usage

```
get_featurelist(data, is_istd = NA, is_quantifier = NA)
```

Arguments

data	A MRMhubExperiment object
is_istd	If set, then defines whether to include or exclude internal standard features. Default is NA means no filter for internal standards is applied.
is_quantifier	If set, then defines whether to include or exclude quantifier features. Default is NA means no filter for quantifier features is applied.

Value

A character vector with feature_id values

get_iqr_tails	<i>Get Tukey's IQR fences</i>
---------------	-------------------------------

Description

Computes the lower and upper boundaries based on Tukey's Interquartile Range (IQR) fences from a numeric vector.

Usage

```
get_iqr_tails(x, k = 1.5, na.rm = FALSE)
```

Arguments

x	A numeric vector.
k	A numeric multiplier for the IQR. Default is 1.5.
na.rm	Logical. Should missing values be removed? Default is FALSE.

Value

A numeric vector of length 2: c(lower_tail, upper_tail).

get_mad_tails	<i>Get MAD-based tails</i>
---------------	----------------------------

Description

Computes the lower and upper boundaries based on Median Absolute Deviation (MAD) from a numeric vector. Returns c(NA_real_, NA_real_) if the input vector has less than 2 elements or if no values fall within the computed fences.

Usage

```
get_mad_tails(x, k, na.rm = FALSE)
```

Arguments

x	a numeric vector
k	multiplier for the MAD
na.rm	if TRUE then NA values are stripped from x before computation takes place.

Value

a numeric vector of length 2 with the lower and upper boundaries.

get_outlier_bounds	<i>Get outlier bounds via different methods</i>
--------------------	---

Description

Computes lower and upper bounds for a numeric vector using one of several methods:

Usage

```
get_outlier_bounds(
  x,
  method = c("iqr", "mad", "sd", "quantile", "z_normal", "z_robust", "fold_change"),
  k = NULL,
  outlier_log = FALSE,
  na.rm = FALSE
)
```

Arguments

x	A numeric vector.
method	Character string: one of "iqr", "mad", "sd", "quantile", "z_normal", "z_robust", or "fold_change".
k	Numeric multiplier or threshold. Defaults depend on method: • "iqr": 1.5 (multiplier of IQR) • "mad": 3 (multiplier of MAD) • "sd": 3 (multiplier of SD) • "z_normal": 3 (threshold in SD units) • "z_robust": 3.5 (threshold in robust Z units) • "fold_change": 2 (fold-change multiplier, assumes log-transformed data) • "quantile": 0.01 (lower/upper quantiles, e.g., 1% and 99%)
outlier_log	Logical. If TRUE, applies log10 transformation to x
na.rm	Logical. Should missing values be removed? Default is FALSE.

Details

- "iqr": Tukey's Interquartile Range fences
- "mad": Median Absolute Deviation
- "sd": Standard deviation from mean
- "quantile": Fixed percentile cutoffs
- "z_normal": Standard Z-score using mean & SD
- "z_robust": Modified Z-score using median & MAD
- "fold_change": Median $\pm$ log10(k), assumes log-transformed data

Value

A numeric vector of length 2: `c(lower_bound, upper_bound)` representing the smallest and largest observed values within the computed fences.

Examples

```
get_outlier_bounds(c(1, 2, 3, 4, 100), "iqr")
get_outlier_bounds(c(1, 2, 3, 4, 100), "mad")
get_outlier_bounds(c(1, 2, 3, 4, 100), "sd")
get_outlier_bounds(c(1, 2, 3, 4, 100), "quantile", k = 0.05)
get_outlier_bounds(c(1, 2, 3, 4, 100), "z_normal")
get_outlier_bounds(c(1, 2, 3, 4, 100), "z_robust")
get_outlier_bounds(log10(c(1, 2, 4, 8)), "fold_change") # default 2×
get_outlier_bounds(log10(c(1, 2, 4, 8)), "fold_change", k = 3) # 3× fold-change
```

```
get_qc_bias_variability
```

Retrieve QC bias and variability metrics

Description

This function retrieves quality control (QC) bias and variability metrics from a `MRMhubExperiment` object. It returns a summary of QC metrics for specified QC samples, including bias, absolute bias, and intra-assay coefficient of variation (CV). The standard deviation of the concentration ratio is also included unless it is NA for all analytes, i.e. when no replicates were measured.

Usage

```
get_qc_bias_variability(
  data,
  qc_types = NA,
  sample_ids = NA,
  wide_format = "none",
  include_qualifier = FALSE,
  with_conc = TRUE,
  with_conc_target = TRUE,
  with_bias = TRUE,
  with_bias_abs = FALSE,
  with_conc_ratio = FALSE,
  with_cv_intra = TRUE,
  with_conc_out_of_range = TRUE,
  summary_table = FALSE
)
```

Arguments

data	A MRMhubExperiment object containing the dataset and necessary annotations for calibration analysis.
qc_types	A character vector specifying the QC types to include in the results, in addition to CAL. If not specified, all applicable QC types are included by default.
sample_ids	A character vector specifying the sample IDs to include in the results. If not specified, all analyses regardless of their sample IDs are included by default.
wide_format	Format of the output table. Must be one of "none", "features", or "samples". If "none", the output is in long format. If "features", the output is in wide format with features as columns. If "samples", the output is in wide format with samples as columns.
include_qualifier	Logical. If TRUE, includes qualifier features in the results. Defaults to FALSE.
with_conc	Logical. If TRUE, includes target and measured mean concentrations in the results. Defaults to TRUE.
with_conc_target	Logical. If TRUE, includes target (known) concentration of the QC sample in the results. Defaults to TRUE.
with_bias	Logical. If TRUE, includes percentage bias in the results. Defaults to TRUE.
with_bias_abs	Logical. If TRUE, includes absolute bias in concentration units in the results. Defaults to FALSE.
with_conc_ratio	Logical. If TRUE, includes the ratio of measured to target concentration in the results. Defaults to FALSE.
with_cv_intra	Logical. If TRUE, includes intra-assay coefficient of variation (CV) in the results. Defaults to TRUE.
with_conc_out_of_range	Logical. If TRUE (and the dataset carries the <code>feature_conc_out_of_range</code> flag from quantify_by_calibration()), includes the fraction of replicate measurements whose concentration fell outside the calibrated range (<code>frac_conc_out_of_range</code>). Defaults to TRUE.
summary_table	Logical. If TRUE, round the numeric metric columns to 3 significant figures for display (the integer replicate count <code>n</code> is left intact). This is a presentation view for reporting; the values are rounded , so do not use it for downstream computation. Defaults to FALSE (full, unrounded metrics).

Details

The standard deviation of concentration is also included unless the number of replicates was 1.

The function uses data from the `MRMhubExperiment` object and filters it according to the specified QC types and other parameters. It then calculates summary statistics for each feature, such as bias and CV, and organizes the data into a user-specified format.

Value

A data frame containing the calibration results, including metrics such as bias, percentage bias, and intra-assay CV based on specified parameters.

```
get_response_curve_stats
```

Linear regression statistics of response curves

Description

This function calculates linear regression statistics (R^2 , slope, and intercept) for each response curve in the provided `MRMhubExperiment` object. Each curve is fitted on its points with a non-missing intensity, with a warning when some are missing; a curve with 2 such points has a slope and intercept but no R^2 (NA), and a curve with fewer gives NA throughout. Before fitting, the analyzed sample amount (x) and feature intensity (y) of these points are each scaled to their maximum (set to 1), so the returned `slopenorm` and `y0norm` are on this normalized scale. Optionally, it can include additional statistics from the `lancer` package (if installed) when `with_saturation_stats` is set to TRUE.

Usage

```
get_response_curve_stats(
  data = NULL,
  with_saturation_stats = FALSE,
  limit_to_rqc = FALSE,
  silent_invalid_data = FALSE
)
```

Arguments

<code>data</code>	A MRMhubExperiment object containing the dataset and response curve annotations.
<code>with_saturation_stats</code>	Logical, if TRUE, include additional statistics from the <code>lancer</code> package. Note: The <code>lancer</code> package must be installed when this argument is set to TRUE.
<code>limit_to_rqc</code>	Logical, if TRUE, only include rows with <code>qc_type == "RQC"</code> . Default is FALSE.
<code>silent_invalid_data</code>	Logical, if TRUE suppresses raising an error when required data or metadata are missing, or there is a mismatch between them.

Value

A tibble with linear regression statistics (`r2`, `slopenorm`, `y0norm`) for each curve, or NULL if no data matches the criteria.

get_runtime_median	<i>Get the median run time</i>
--------------------	--------------------------------

Description

Calculates the median run time (in seconds) based of the timestamps differences between consecutive analyses in the sequence.

Usage

```
get_runtime_median(data)
```

Arguments

data A [MRMhubExperiment](#) object

Value

A lubridate time period object, or NA if the dataset is empty.

import_data_csv_long	<i>Import analysis results from long-format CSV files</i>
----------------------	---

Description

This function imports analysis results from CSV files in long table format, where each row represents a unique observation of a feature-value pair for an analysis (sample), along with associated feature variables and other metadata. See "Details" below for more information on using this function.

Usage

```
import_data_csv_long(  
  data = NULL,  
  path,  
  import_metadata = TRUE,  
  column_mapping = NULL,  
  na_strings = "NA",  
  warn_unrecognized_columns = TRUE,  
  silent = FALSE  
)
```

Arguments

data	A <code>MRMhubExperiment</code> object to which the imported data will be added.
path	A single file path, multiple file paths, or a directory path. If a directory is provided, all *.csv files within will be imported.
import_metadata	Logical indicating whether to import additional metadata columns (e.g., batch ID, sample type) into the <code>MRMhubExperiment</code> object. Supported metadata column names include "qc_type", "batch_id", "is_quantifier", "is_istd", and "analysis_order".
column_mapping	A named character vector mapping internal column names to CSV column names. Should include keys such as "analysis_id", "feature_id", and feature variable names. If NULL (default), the function attempts automatic detection.
na_strings	Character vector of strings to interpret as missing values (NA). Blank fields are always treated as missing.
warn_unrecognized_columns	Logical indicating whether to issue a warning when unknown columns are encountered in the dataset.
silent	Logical indicating whether to suppress most notifications and messages.

Details

When no column mapping is provided via the `column_mapping` argument, the function will automatically detect and import columns with the following names:

CSV Column Name	MRMhub Internal Name	Required
analysis_id	analysis_id	Yes
feature_id	feature_id	Yes
qc_type	qc_type	No
batch_id	batch_id	No
istd_feature_id	istd_feature_id	No
feature_class	feature_class	No
precursor_mz	method_precursor_mz	No
product_mz	method_product_mz	No
area	feature_area	No
height	feature_height	No
intensity	feature_intensity	No
response	feature_response	No
conc	feature_conc	No
rt	feature_rt	No
fwhm	feature_fwhm	No
width	feature_width	No

Detection of these columns is case-insensitive. Additionally, if feature variable columns use the internal naming convention with prefixes "feature_" or "method_" (e.g. `feature_area` instead of `area`), the function will detect and import them automatically.

To import data with different column names, provide a named vector mapping CSV column names to the internal column names used by `MRMhubExperiment`. The mapping should be in the format: `c("analysis_id" = "[CSV column name for analysis]", "feature_id" = "[CSV column name for feature]", ...)`, where the right-hand side refers to the exact column name in the CSV file header. Columns matching internal names do not require mapping and will be imported automatically. The mapping is case-insensitive.

Note that the dataset must contain an analysis identifier, either as an `analysis_id` column or via a mapped column.

The function processes all CSV files in the specified directory or the given file(s), combining them into a single dataset. This supports datasets split across multiple files during preprocessing. Each feature and raw data file pair should appear only once to avoid duplication.

The `na_strings` parameter allows specifying character strings that should be interpreted as NA, ensuring proper handling of missing values.

Value

A `MRMhubExperiment` object containing the imported data.

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (`.mzML`, `.d`, `.raw`, `.wiff`, `.wiff2`, `.lcd`, `.chrom`, case-insensitive) are stripped from `analysis_id`.

The same normalization is applied to both the data and the metadata, which is what lets an `analysis_id` typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

Examples

```
file_path <- system.file("extdata", "plain_long_dataset.csv", package = "mrmhub")
mexp <- MRMhubExperiment()

# Define the column mapping; right side is the CSV column name
col_map <- c(
  "analysis_id" = "raw_data_filename",
  "qc_type" = "qc_type",
  "feature_id" = "feature_id",
  "feature_class" = "feature_class",
  "istd_feature_id" = "istd_feature_id",
  "feature_rt" = "rt",
  "feature_area" = "area"
)

mexp <- import_data_csv_long(
  data = mexp,
  path = file_path,
  column_mapping = col_map,
```

```

    import_metadata = TRUE
  )

  print(mexp)

```

import_data_csv_wide *Import analysis results from plain wide-format CSV files*

Description

Imports analysis result data from wide-format .csv files, where each row corresponds to a single analysis and each feature is stored in its own column. All feature columns hold the same variable type, given by variable_name.

Usage

```

import_data_csv_wide(
  data = NULL,
  path,
  variable_name,
  analysis_id_col = NA,
  import_metadata = TRUE,
  first_feature_column = NA,
  na_strings = "NA"
)

```

Arguments

data	A MRMhubExperiment object.
path	A file path or vector of file paths, or a directory path. If a directory is provided, all .csv files within it will be read.
variable_name	A character string specifying the variable type contained in the data. Must be one of "area", "height", "intensity", "norm_intensity", "response", "conc", "conc_raw", "rt", or "fwhm".
analysis_id_col	The column name or index to be used as analysis_id. Defaults to NA, in which case "analysis_id" is used if present; otherwise, the first column is used if it contains unique values.
import_metadata	Logical indicating whether to import additional metadata columns (e.g., batch ID, sample type) into the MRMhubExperiment object. Supported metadata columns are: "qc_type", "batch_id", "is_quantifier", "is_istd", and "analysis_order".
first_feature_column	Integer indicating the column number where feature value columns start.
na_strings	Character vector of strings to interpret as NA values. Blank fields are also treated as NA.

Details

Each row must be identified by an "analysis_id" column (or, if absent, the first column, provided it holds unique values). Every feature is stored in a separate column; the feature columns all contain the single variable type specified by variable_name. Use first_feature_column to mark where the feature columns begin so that leading metadata columns are not mistaken for features.

The variable_name argument specifies the data type represented in the table, which must be one of: "area", "height", "intensity", "norm_intensity", "response", "conc", "conc_raw", "rt", or "fwhm".

If there is no column named analysis_id, it will be inferred from the first column, provided it contains unique values.

When import_metadata is set to TRUE, the following metadata columns will be imported if present:

- analysis_order
- qc_type
- batch_id
- is_quantifier

To prevent additional non-metadata columns from being misinterpreted as features, use the first_feature_column parameter to specify the column where feature data starts.

If a directory path is provided to path, all .csv files in that directory will be processed and merged into a single dataset. This facilitates handling datasets split into multiple files during preprocessing. Ensure each feature and raw data file pair appears only once to avoid duplication errors.

The na_strings parameter allows specifying character strings to be interpreted as missing values (NA). Blank fields are also treated as missing.

Value

A [MRMhubExperiment](#) object containing the imported dataset.

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from analysis_id.

The same normalization is applied to both the data and the metadata, which is what lets an analysis_id typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

Examples

```
file_path <- system.file("extdata", "plain_wide_dataset.csv", package = "mrhub")
mexp <- MRMhubExperiment()
mexp <- import_data_csv_wide(
  data = mexp,
  path = file_path,
```

```

    variable_name = "conc",
    import_metadata = TRUE
)
print(mexp)

```

import_data_masshunter

Import Agilent MassHunter Quantitative Analysis CSV files

Description

Imports .csv files exported from Agilent MassHunter Quantitative Analysis software, containing peak integration results. The input files must have analyses (samples) in rows, features/compounds in columns, and either peak areas, peak heights, or response as the values. Additional columns, such as retention time (RT), full-width at half-maximum (FWHM), precursor m/z (PrecursorMZ), and collision energy (CE), will also be imported and made available in the MRMhubExperiment object for downstream analyses.

When a directory path is provided, all matching .csv files in that directory will be imported and merged into a single dataset. This is useful when importing datasets that were pre-processed in blocks, resulting in multiple files. Each unique combination of feature and raw data file must only occur once across all source data files. Duplicate combinations will result in an error.

Usage

```

import_data_masshunter(
  data = NULL,
  path,
  import_metadata = TRUE,
  expand_qualifier_names = TRUE,
  conc_column = "conc_final",
  silent = FALSE
)

```

Arguments

data	MRMhubExperiment object
path	One or more file paths, or a directory path (in which case all matching files will be imported)
import_metadata	Logical, whether to extract and add metadata from the analysis result file
expand_qualifier_names	Logical, whether to add the quantifier name in front of the qualifier name (the latter only has the m/z transition values)
conc_column	Which concentration field of the masshunter data to use, in case "Calc. Conc." and "Final Conc." are present. Default is "conc_final". Must be one of "conc_calc" or "conc_final" (default).
silent	Logical, whether to suppress most notifications

Value

`MRMhubExperiment` object with the imported data

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from `analysis_id`.

The same normalization is applied to both the data and the metadata, which is what lets an `analysis_id` typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

Examples

```
mexp <- MRMhubExperiment()
file_path = system.file("extdata", "MHQuant_demo.csv", package = "mrmhub")

mexp <- import_data_masshunter(
  data = mexp,
  path = file_path,
  import_metadata = TRUE,
  expand_qualifier_names = TRUE)

print(mexp)
```

<code>import_data_mrmhub</code>	<i>Import MRMhub peak integration results</i>
---------------------------------	---

Description

Imports tabular data files (*.tsv) generated from MRMhub containing peak integration results. The input files must be in a long format with columns for the raw data file name, feature ID, peak intensity, and other columns. Additional information, such as retention time, FWHM, precursor/product m/z, and CE will also be imported and made available in the `MRMhubExperiment` object for downstream analyses.

When a directory path is provided, all matching files in that directory will be imported and merged into a single dataset. This is useful when importing datasets that were pre-processed in blocks, resulting in multiple files. Each unique combination of feature and raw data file must only occur once across all source data files. Duplicate combinations will result in an error.

Usage

```
import_data_mrmhub(data = NULL, path, import_metadata = TRUE, silent = FALSE)
```

Arguments

data	MRMhubExperiment object
path	One or more file paths, or a directory path (in which case all matching files will be imported)
import_metadata	Logical, whether to import additional metadata columns (e.g., batch_id, qc_type)
silent	Logical, whether to suppress most notifications

Value

MRMhubExperiment object with the imported data

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from analysis_id.

The same normalization is applied to both the data and the metadata, which is what lets an analysis_id typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

Examples

```
mexp <- MRMhubExperiment()

file_path = system.file("extdata", "MRMhub_demo.tsv", package = "mrmhub")

mexp <- import_data_mrmhub(
  data = mexp,
  path = file_path,
  import_metadata = TRUE)
print(mexp)
```

import_data_mztab	<i>Import data from an mzTab-M file</i>
-------------------	---

Description

Imports quantitative results from an **mzTab-M** file (e.g. produced by Lipid Data Analyzer, MS-DIAL or MZmine) into an MRMhubExperiment. Each Small Molecule Feature (SMF) becomes an mrmhub feature and each assay an analysis; the per-assay abundances are imported as feature_intensity.

Usage

```
import_data_mztab(data = NULL, path, import_metadata = TRUE, silent = FALSE)
```

Arguments

data	An MRMhubExperiment object (e.g. from MRMhubExperiment()).
path	Path to a .mzTab file, or a directory of them.
import_metadata	If TRUE (default), derive analysis/feature metadata (incl. formula, neutral mass) from the imported data via import_metadata_from_data() .
silent	Suppress messages.

Details

mzTab-M is a quantification *report*, so an import is necessarily partial: the single reported abundance per feature is mapped to `feature_intensity`, and feature identities (name, formula, neutral mass, m/z, retention time) are taken from the SMF/SML sections where available. Internal-standard relationships, QC-type assignments and calibration metadata are **not** part of mzTab-M and must be supplied afterwards with [add_metadata\(\)](#). `study_variable` groups are not imported; all analyses are in one batch.

Value

The updated [MRMhubExperiment](#).

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from `analysis_id`.

The same normalization is applied to both the data and the metadata, which is what lets an `analysis_id` typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

See Also

[import_data_mrmhub\(\)](#), [save_dataset_mztab\(\)](#), [add_metadata\(\)](#)

Examples

```
## Not run:
mexp <- MRMhubExperiment()
mexp <- import_data_mztab(mexp, "LDA_export.mzTab")

## End(Not run)
```

import_data_skyline	<i>Import Skyline peak integration results</i>
---------------------	--

Description

This function imports tabular data files (*.csv) exported from Skyline, containing peak integration results.

Usage

```
import_data_skyline(
  data = NULL,
  path,
  transition_id_columns = c("name", "mz", "none"),
  import_metadata = TRUE,
  silent = FALSE
)
```

Arguments

data	A MRMhubExperiment object.
path	One or more file paths, or a directory path from which all matching files will be imported.
transition_id_columns	A character vector specifying the columns that define the transition (precursor and product) to use for unique feature_id generation. Options are "name", "mz", or "none". If "none", feature_id is derived from Molecule Name and Precursor Name or Product Name. If "mz", feature_id is based on Precursor Mz and Product Mz. Using "none" will result in feature_id being a copy of Molecule Name, with an error raised if it is not unique for each transition.
import_metadata	Logical; whether to import additional metadata columns (e.g., precursor/product m/z values).
silent	Logical; whether to suppress most notifications.

Details

In Skyline, transitions are defined by the Molecule Name and corresponding precursor and product m/z values, rather than any identifier. When importing this data, the feature_id is generated using the Molecule Name with either precursor/product names or m/z values, unless Molecule Name uniquely identifies the features (refer to the transition_id_columns argument below).

The following supported columns from Skyline can be imported:

Skyline Column Name	MRMhub Column Name	Required
Replicate Name	analysis_id	Yes
Molecule List Name	feature_class	No

Molecule Name	feature_id	Yes
Precursor Name	feature_id	Yes*
Product Name	feature_id	Yes*
Precursor Mz	method_precursor_mz	Yes*
Product Mz	method_product_mz	Yes*
Area	feature_area	Yes
Retention Time	feature_rt	No

*Requirements for these columns are described in `transition_id_columns`.

To export results from Skyline, use the 'Molecule Transition Results' format and include Replicate Name, Molecule Name, and either Precursor Mz/Product Mz or Precursor Name/Product Name columns. At least one feature variable, such as Area or RT, must also be exported.

Value

A `MRMhubExperiment` object containing the imported data.

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from `analysis_id`.

The same normalization is applied to both the data and the metadata, which is what lets an `analysis_id` typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

Examples

```
mexp <- MRMhubExperiment()
file_path <- system.file("extdata", "Skyline_MoleculeTransitionResults.csv", package = "mrmhub")
mexp <- import_data_skyline(
  data = mexp,
  path = file_path,
  transition_id_columns = "mz",
  import_metadata = TRUE
)
print(mexp)
```

Description

Imports analysis metadata (annotation) from a preloaded data frame or tibble via the `data` argument, or from data from a file (CSV or Excel) via the `path` argument. The analysis metadata must contain following columns: `analysis_id` and `qc_type`. Additional analysis metadata columns are described under details below.

Usage

```
import_metadata_analyses(
  data = NULL,
  table = NULL,
  path = NULL,
  sheet = NULL,
  ignore_warnings = FALSE,
  excl_unmatched_analyses = FALSE
)
```

Arguments

<code>data</code>	A MRMhubExperiment object
<code>table</code>	A data frame or tibble with analysis (sample) metadata. If <code>path</code> is also provided, an error will be raised.
<code>path</code>	A character string specifying the path to a CSV (.csv) or Excel (.xlsx) file. If <code>table</code> is also provided, an error will be raised.
<code>sheet</code>	Defines the sheet name in case an Excel file is provided.
<code>ignore_warnings</code>	Ignore warnings from data validation and proceed with importing metadata
<code>excl_unmatched_analyses</code>	Exclude analyses (samples) that have no matching metadata

Value

An updated [MRMhubExperiment](#) object

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from `analysis_id`.

The same normalization is applied to both the data and the metadata, which is what lets an `analysis_id` typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

Examples

```

mexp <- MRMhubExperiment()
file_path = system.file("extdata", "MHQuant_demo.csv", package = "mrmhub")
mexp <- import_data_masshunter(
  data = mexp,
  path = file_path,
  import_metadata = FALSE)

meta_path = system.file("extdata", "MHQuant_demo_metadata_analyses.csv", package = "mrmhub")

mexp <- import_metadata_analyses(
  data = mexp,
  path = meta_path,
  excl_unmatched_analyses = TRUE)

print(mexp)

```

import_metadata_features

Import feature metadata

Description

Imports feature metadata (annotation) from a preloaded data frame or tibble via the `data` argument, or from data from a file (CSV or Excel) via the `path` argument. The feature metadata must contain following column: `feature_id`. Additional feature metadata columns are described under details below.

Usage

```

import_metadata_features(
  data = NULL,
  table = NULL,
  path = NULL,
  sheet = NULL,
  ignore_warnings = FALSE
)

```

Arguments

<code>data</code>	A MRMhubExperiment object
<code>table</code>	A data frame or tibble with feature metadata. If <code>path</code> is also provided, an error will be raised.
<code>path</code>	A character string specifying the path to a CSV (.csv) or Excel (.xlsx) file. If <code>table</code> is also provided, an error will be raised.
<code>sheet</code>	Defines the sheet name in case an Excel file is provided.
<code>ignore_warnings</code>	Ignore warnings from data validation and proceed with importing metadata

Value

An updated [MRMhubExperiment](#) object

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from analysis_id.

The same normalization is applied to both the data and the metadata, which is what lets an analysis_id typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

`import_metadata_from_data`*Retrieve metadata from imported analysis data*

Description

Retrieves available metadata from the imported analysis data and associates it with the provided MRMhubExperiment object.

Usage

```
import_metadata_from_data(data = NULL, qc_type_column_name = "qc_type")
```

Arguments

`data` A [MRMhubExperiment](#) object

`qc_type_column_name` Column name in the imported raw data representing the qc_type

Value

An updated [MRMhubExperiment](#) object

import_metadata_istds *Import internal standards (ISTD) metadata*

Description

Imports ISTD metadata (annotation) from a preloaded data frame or tibble via the `data` argument, or from data from a file (CSV or Excel) via the `path` argument. The ISTD metadata must contain following columns: `istd_feature_id` and one of `istd_conc_nmolar` or `istd_conc_ngml`.

Usage

```
import_metadata_istds(  
  data = NULL,  
  table = NULL,  
  path = NULL,  
  sheet = NULL,  
  ignore_warnings = FALSE  
)
```

Arguments

<code>data</code>	A MRMhubExperiment object
<code>table</code>	A data frame or tibble with ISTD metadata. If <code>path</code> is also provided, an error will be raised.
<code>path</code>	A character string specifying the path to a CSV (.csv) or Excel (.xlsx) file. If <code>table</code> is also provided, an error will be raised.
<code>sheet</code>	Defines the sheet name in case an Excel file is provided.
<code>ignore_warnings</code>	Ignore warnings from data validation and proceed with importing metadata

Value

An updated [MRMhubExperiment](#) object

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from `analysis_id`.

The same normalization is applied to both the data and the metadata, which is what lets an `analysis_id` typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

`import_metadata_msorganiser`*Import metadata from a MRMhub Metadata Organizer file*

Description

Imports metadata from a 'MRMhub Metadata Organizer' file (.xlsx) file and associates it with analysis data.

Usage

```
import_metadata_msorganiser(  
    data = NULL,  
    path,  
    ignore_warnings = FALSE,  
    excl_unmatched_analyses = FALSE  
)
```

Arguments

<code>data</code>	A MRMhubExperiment object
<code>path</code>	File name and path of the 'MRMhub Metadata Organizer' file (.xlsx) file
<code>ignore_warnings</code>	Ignore warnings from data validation and proceed with importing metadata
<code>excl_unmatched_analyses</code>	Exclude analyses (samples) that have no matching metadata

Value

An updated [MRMhubExperiment](#) object

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from `analysis_id`.

The same normalization is applied to both the data and the metadata, which is what lets an `analysis_id` typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

Examples

```
mexp <- MRMhubExperiment()

mexp <- import_data_mrmhub(
  data = mexp,
  path = system.file("extdata", "MRMhub_demo.tsv", package = "mrmhub"),
  import_metadata = TRUE)

mexp <- import_metadata_msorganiser(
  data = mexp,
  path = system.file("extdata", "Example_Metadata_1.xlsx", package = "mrmhub"),
  excl_unmatched_analyses = FALSE,
  ignore_warnings = TRUE)

print(mexp)
```

import_metadata_qcconcentrations

Import calibration curves metadata

Description

Imports calibration curve metadata (annotation) from a preloaded data frame or tibble via the data argument, or from data from a file (CSV or Excel) via the path argument. The calibration curve metadata must contain following columns: sample_id, analyte_id, concentration, and concentration_unit.

Usage

```
import_metadata_qcconcentrations(
  data = NULL,
  table = NULL,
  path = NULL,
  sheet = NULL,
  ignore_warnings = FALSE
)
```

Arguments

data	A MRMhubExperiment object
table	A data frame or tibble with calibration curve metadata. If path is also provided, an error will be raised.
path	A character string specifying the path to a CSV (.csv) or Excel (.xlsx) file. If table is also provided, an error will be raised.
sheet	Defines the sheet name in case an Excel file is provided.
ignore_warnings	Ignore warnings from data validation and proceed with importing metadata

Value

An updated `MRMhubExperiment` object

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from `analysis_id`.

The same normalization is applied to both the data and the metadata, which is what lets an `analysis_id` typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

```
import_metadata_responsecurves
    Import response curves metadata
```

Description

Imports response curve metadata (annotation) from a preloaded data frame or tibble via the `data` argument, or from data from a file (CSV or Excel) via the `path` argument. The response curve metadata must contain following columns: `analysis_id`, `curve_id`, `analyzed_amount` and `analyzed_amount_unit`.

Usage

```
import_metadata_responsecurves(
  data = NULL,
  table = NULL,
  path = NULL,
  sheet = NULL,
  ignore_warnings = FALSE
)
```

Arguments

<code>data</code>	A <code>MRMhubExperiment</code> object
<code>table</code>	A data frame or tibble with response curve metadata. If <code>path</code> is also provided, an error will be raised.
<code>path</code>	A character string specifying the path to a CSV (.csv) or Excel (.xlsx) file. If <code>table</code> is also provided, an error will be raised.
<code>sheet</code>	Defines the sheet name in case an Excel file is provided.
<code>ignore_warnings</code>	Ignore warnings from data validation and proceed with importing metadata

Value

An updated [MRMhubExperiment](#) object

Identifier normalization

All imported identifiers are whitespace-normalized on import: leading and trailing spaces are removed and internal runs of whitespace are collapsed to a single space (for example "QC 01" becomes "QC 01"). Raw-data file extensions (.mzML, .d, .raw, .wiff, .wiff2, .lcd, .chrom, case-insensitive) are stripped from analysis_id.

The same normalization is applied to both the data and the metadata, which is what lets an analysis_id typed into metadata match the one derived from a data-file name instead of silently failing to join. A consequence is that two identifiers differing only by whitespace collapse to one and are then reported as duplicates.

licar_pattern_choices *Valid MRM-pattern labels*

Description

The labels a feature's mrm_pattern may take, optionally for one product-ion origin. Single source of truth shared by metadata validation and the Excel template dropdown (so they cannot drift).

Usage

```
licar_pattern_choices(origin = NULL)
```

Arguments

origin Optional origin filter ("Head Group", "FA", "LCB", "Neutral", "RPLC").

Value

A character vector of labels.

lipidomics_dataset *Plasma lipidomics dataset with metadata*

Description

This demo dataset is included for use in function examples and user testing. It is a small, preprocessed subset of a plasma lipidomics dataset, containing raw peak areas and analytical metadata. The original dataset was published in Tan et al., ATVB, 2022.

Usage

```
lipidomics_dataset
```

Format

A MRMhubExperiment object with the following data and metadata:

dataset_orig A tibble containing the original peak data.

dataset A tibble with annotated lipidomics data.

annot_analyses Analysis-level metadata

annot_features Feature-level metadata

annot_batches Batch annotations.

annot_istds ISTD concentrations

annot_responsecurves Response curves metadata

mrmhub_enable_cli_color

Enable coloured mrmhub console output in notebooks

Description

mrmhub's console feedback is coloured with cli. In an interactive session this works out of the box, but in a non-interactive render (Quarto or R Markdown) cli suppresses colour by default, and its alerts travel on the *message* stream, which knitr does not colour-convert. Call this once in a setup chunk to (1) advertise colour support to cli via `cli.num_colors / crayon.enabled` and (2), when rendering to HTML, register a knitr message hook that converts the emitted ANSI sequences to coloured HTML using the **fansi** package. For PDF and Word the same hook strips the ANSI so messages stay clean plain text. Advertising colour also tints ordinary output (e.g. tibble prints) on the output stream, which knitr leaves untouched, so an output hook strips that ANSI to keep printed data plain text; only the message-stream alerts colour.

Usage

```
mrmhub_enable_cli_color(num_colors = 256L)
```

Arguments

`num_colors` Number of colours to advertise to cli. Defaults to 256.

Value

Invisibly, the previous values of the options it changed (as returned by `options()`), so the caller can restore them.

Examples

```
## Not run:
# In a Quarto / R Markdown setup chunk:
mrmhub_enable_cli_color()

## End(Not run)
```

`mrmhub_get_plot_defaults`*Report the global MRMhub plot defaults*

Description

Returns the currently set `mrmhub.*` plot-appearance options as a named list. Options that are not set are omitted.

Usage

```
mrmhub_get_plot_defaults()
```

Value

A named list of the active plot defaults (empty if none are set).

See Also

[mrmhub_set_plot_defaults\(\)](#)

Examples

```
mrmhub_set_plot_defaults(font_base_size = 8)
mrmhub_get_plot_defaults()
mrmhub_reset_plot_defaults()
```

`mrmhub_reset_plot_defaults`*Clear the global MRMhub plot defaults*

Description

Removes all `mrmhub.*` plot-appearance options set by [mrmhub_set_plot_defaults\(\)](#), restoring the built-in per-function defaults.

Usage

```
mrmhub_reset_plot_defaults()
```

Value

Invisibly, the previous option values (as returned by [options\(\)](#)).

See Also

[mrmhub_set_plot_defaults\(\)](#)

Examples

```
mrmhub_set_plot_defaults(font_base_size = 8)
mrmhub_reset_plot_defaults()
```

```
mrmhub_set_plot_defaults
```

Set global default appearance for MRMhub plots

Description

Sets session-wide defaults for the shared appearance arguments of the `plot_*`() functions, so common choices such as a smaller base font or point size can be made once at the top of a notebook instead of on every call. Defaults are stored as `mrmhub.*` options. The precedence for every argument is: a value passed explicitly to a plotting function wins, then the global default set here, then the function's built-in default (including automatic `cols_page`-based sizing on faceted plots).

Usage

```
mrmhub_set_plot_defaults(
  font_base_size = NULL,
  point_size = NULL,
  legend_position = NULL,
  legend_size = NULL,
  show_legend_title = NULL,
  strip_bg_color = NULL,
  units = NULL,
  dpi = NULL
)
```

Arguments

<code>font_base_size</code>	Base font size in points; all plot text scales from it.
<code>point_size</code>	Point/marker size.
<code>legend_position</code>	Legend placement: a keyword ("right", "left", "top", "bottom", "none"), a corner ("inside-tr", "inside-tl", "inside-br", "inside-bl"), or a numeric <code>c(x, y)</code> .
<code>legend_size</code>	Scales the whole legend (text, title, key and glyphs); a value ≤ 3 is a multiplier of <code>font_base_size</code> , larger values are points.
<code>show_legend_title</code>	FALSE hides legend titles.
<code>strip_bg_color</code>	Facet strip background fill (strip text auto-contrasts).
<code>units</code>	Default unit for the width and height of <code>save_plot()</code> : "mm" (the built-in default), "cm", "in", "pt" or "px".
<code>dpi</code>	Default resolution in dots per inch for the raster formats of <code>save_plot()</code> . The built-in default is 300.

Details

Only the arguments you supply are changed; the rest are left untouched. Pass `NULL` (the default) to leave a setting as it is, and use `mrmhub_reset_plot_defaults()` to clear them all. The previous option values are returned invisibly, so a set can be undone with `options(old)` or scoped to a block with `withr::local_options()`.

Value

Invisibly, a named list of the option values as they were before this call (as returned by `options()`).

See Also

`mrmhub_reset_plot_defaults()`, `mrmhub_get_plot_defaults()`, `save_plot()` and the vignette("manual-08-visual article).

Examples

```
old <- mrmhub_set_plot_defaults(font_base_size = 8, point_size = 0.8)
# ... make plots with the smaller defaults ...
options(old) # restore
```

mrmhub_status

Detailed processing and metadata report for a MRMhubExperiment

Description

Prints the full status dashboard for a `MRMhubExperiment`: sample and feature composition, which metadata tables are populated, the state of each processing step, and any manually excluded analyses or features. Printing the object directly gives the compact one-screen overview instead.

Usage

```
mrmhub_status(object)
```

Arguments

object A `MRMhubExperiment` object.

Value

The object, invisibly.

Examples

```
mrmhub_status(MRMhubExperiment(title = "Test", analysis_type = "lipidomics"))
```

MRMhubExperiment	<i>Constructor for the MRMhubExperiment object</i>
------------------	--

Description

Constructor for the MRMhubExperiment object

Usage

```
MRMhubExperiment(title = "", analysis_type = NA_character_)
```

Arguments

title	Title of experiment
analysis_type	Analysis type, one of "lipidomics", "metabolomics", "externalcalib", "others"

Value

[MRMhubExperiment](#) object

MRMhubExperiment-class	<i>S4 class representing the MRMhub dataset</i>
------------------------	---

Description

The MRMhubExperiment object is the core data structure utilized within the MRMhub workflow, encapsulating all relevant experimental data and metadata. It also includes processing results, details of the applied processing steps, and the current status of the data.

Slots

title	Title of the experiment
analysis_type	Analysis type, one of "lipidomics", "metabolomics", "externalcalib", "others"
feature_intensity_var	Feature variable used as default for calculations
conc_analyte_unit	Unit of the analyte amount underlying feature_conc, set when the data are quantitated. "pmol" or "ng" for ISTD quantitation (see quantify_by_istd()), or the calibrant concentration unit (e.g. "nmol/L") for calibration quantitation. Divided by sample_amount_unit by get_conc_unit() to name the unit of feature_conc. NA when not quantitated.
dataset_orig	Original imported analysis data.
dataset	Processed analysis data.
dataset_filtered	QC-filtered processed analysis data.
annot_analyses	Annotation of analyses/runs

annot_features Annotation of measured features.
 annot_istds Annotation of Internal Standard concs.
 annot_responsecurves Annotation of response curves (RQC).
 annot_qcconcentrations Annotation of calibration curves.
 annot_studysamples Annotation of study samples.
 annot_batches Annotation of batches.
 annot_interferences Interference relationships (feature_id, interference_feature_id, interference_contribution, overlap_type, source) feeding the correction engine; derived (auto) and/or manual.
 metrics_qc QC information for each measured feature
 metrics_calibration Calibration metrics calculated from external calibration curves for each measured feature
 status_processing Status within the data processing workflow
 is_istd_normalized Flag if data has been ISTD normalized
 is_quantitated Flag if data has been quantitated using ISTD and sample amount
 is_filtered Flag if data has been filtered based on QC parameters
 is_isotope_corr Flag if one or more features have been isotope corrected
 analyses_excluded Analyses excluded from processing, plots and reporting, unless explicitly requested
 features_excluded Features excluded from processing, plots and reporting, unless explicitly requested
 var_drift_corrected List indicating which variables are drift corrected
 var_batch_corrected List indicating which variables are batch corrected

normalize_by_istd	<i>Normalize feature intensities using internal standards</i>
-------------------	---

Description

Normalize feature intensities by dividing them by the intensities of the corresponding internal standards (ISTDs). Each feature must have a defined internal standard (ISTD) in the feature metadata.

Usage

```
normalize_by_istd(data = NULL, ignore_missing_annotation = FALSE)
```

Arguments

data A [MRMhubExperiment](#) object.
 ignore_missing_annotation
 If FALSE, the function will raise an error when an ISTD is not defined for one or more features (excluding the ISTDs themselves). If TRUE, features with missing ISTD annotations will have NA values in the normalized intensities.

Value

A [MRMhubExperiment](#) object with normalized feature intensities

parse_lipid_feature_names

Get lipid class, species and transition names

Description

This function retrieves lipid class, species and transition names from the `feature_id` column and adds them as columns to the dataset.

Usage

```
parse_lipid_feature_names(  
  tbl,  
  use_as_feature_class = "lipid_class_lcb",  
  add_transition_names = FALSE,  
  add_chain_composition = TRUE  
)
```

Arguments

<code>tbl</code>	A data frame containing a <code>feature_id</code> column
<code>use_as_feature_class</code>	Set <code>feature_class</code> to <code>lipid_class</code>
<code>add_transition_names</code>	add transition name and transition group, based on information in square brackets in <code>feature_id</code>
<code>add_chain_composition</code>	add <code>total_c</code> and <code>total_db</code> to the dataset

Value

The input data frame with added lipid class, species and transition name columns.

parse_masshunter_csv *Reads and parses one Agilent MassHunter Quant CSV result file*

Description

Reads and parses one Agilent MassHunter Quant CSV result file

Usage

```
parse_masshunter_csv(  
  path,  
  expand_qualifier_names = TRUE,  
  silent = FALSE,  
  conc_column = "conc_final"  
)
```

Arguments

path	File path of MassHunter Quant CSV file
expand_qualifier_names	If TRUE, each qualifier is renamed by adding its quantifier's name in front and placing the qualifier's m/z transition in square brackets (e.g. Qualifier (422.3 -> 113.0)).
silent	Suppress messages
conc_column	Which concentration field of the masshunter data to use, in case "Calc. Conc." and "Final Conc" are present. Default is "conc_final".

Value

A tibble with the parse results in the long format

Examples

```
file_path = system.file("extdata", "MHQuant_demo.csv", package = "mrhub")  
  
tbl <- parse_masshunter_csv(  
  path = file_path,  
  expand_qualifier_names = TRUE)  
  
head(tbl)
```

`parse_mrmhub_result` *Parses MRMhub peak integration results into a tibble*

Description

Parses MRMhub peak integration results into a tibble

Usage

```
parse_mrmhub_result(path, silent = FALSE)
```

Arguments

<code>path</code>	File name of the MRMhub result file (*.tsv or *.csv)
<code>silent</code>	No comments printed

Value

A tibble in the long format

Examples

```
file_path = system.file("extdata", "MRMhub_demo.tsv", package = "mrmhub")

tbl <- parse_mrmhub_result(path = file_path)

head(tbl)
```

`parse_plain_long_csv` *Parses a plain long CSV file*

Description

Parses a CSV table with analysis/samples and feature pairs in rows, columns representing the feature variables.

Usage

```
parse_plain_long_csv(
  path,
  na_strings = "NA",
  silent = FALSE,
  column_mapping = NULL,
  warn_unrecognized_columns = TRUE,
  ...
)
```

Arguments

path	File name (*.tsv or *.csv)
na_strings	A character vector of strings to be interpreted as NA values. Blank fields are also considered to be missing values.
silent	No comments printed
column_mapping	A named character vector with the mapping of the columns in the input file to the columns in the output table. If NULL (default), the function will use the default mapping. See import_data_csv_long() for details.
warn_unrecognized_columns	Logical indicating whether to issue a warning for unknown columns in the dataset.
...	Additional arguments passed to the function. Currently only transition_id_columns, which is used for Skyline-like data files. The column mapped to feature_id will be appended with following info: if transition_id_columns = "name", the function will use the method_precursor_name and method_product_name columns to create unique feature IDs. If transition_id_columns = "mz", the function will use the method_precursor_mz and method_product_mz columns to create unique feature IDs.

Value

A tibble in the long format

Examples

```
file_path = system.file("extdata", "plain_long_dataset.csv", package = "mrmhub")

tbl <- parse_plain_long_csv(path = file_path)

head(tbl)
```

parse_plain_wide_csv *Parses a plain wide CSV file*

Description

Parses a CSV table with analysis/samples in rows, features values in columns.

Usage

```
parse_plain_wide_csv(
  path,
  variable_name,
  analysis_id_col = NA,
  import_metadata = TRUE,
  first_feature_column = NA,
  na_strings = "NA"
)
```

Arguments

path	File name and path of a plain wide-format CSV file
variable_name	Name of the variable representing the values in the table. Must be one of "intensity", "norm_intensity", "conc", "area", "height", "response".
analysis_id_col	Column to be used as analysis_id
import_metadata	Import additional metadata columns (e.g. batch ID, sample type) and add to the MRMhubExperiment object
first_feature_column	Column number of the first column representing the feature values
na_strings	A character vector of strings which are to be interpreted as NA values. Blank fields are also considered to be missing values.

Value

A tibble in the long format

Examples

```
file_path <- system.file("extdata", "plain_wide_dataset.csv", package = "mrmhub")

tbl <- parse_plain_wide_csv(
  path = file_path,
  variable_name = "conc",
  analysis_id_col = "analysis_id",
  import_metadata = TRUE)

head(tbl)
```

parse_skyline_result *Parses skyline peak integration results into a tibble*

Description

Parses skyline peak integration results into a tibble

Usage

```
parse_skyline_result(path, na_strings, silent = FALSE, ...)
```

Arguments

path	File name of the Skyline result file (*.tsv or *.csv)
na_strings	A character vector of strings to be interpreted as NA values.
silent	No comments printed
...	Additional arguments passed to the function. Currently only transition_id_columns, which is used for Skyline-like data files. The column mapped to feature_id will be appended with following info: if transition_id_columns = "name", the function will use the method_precursor_name and method_product_name columns to create unique feature IDs. If transition_id_columns = "mz", the function will use the method_precursor_mz and method_product_mz columns to create unique feature IDs.

Value

A tibble in the long format

Examples

```
file_path = system.file("extdata", "Skyline_MoleculeTransitionResults.csv", package = "mrmhub")

tbl <- parse_skyline_result(path = file_path, na_strings = "NA")

head(tbl)
```

plot_abundanceprofile *Plot abundance profile*

Description

Creates a profile plot showing the abundance distribution of features across different classes.

Usage

```
plot_abundanceprofile(
  data,
  variable,
  qc_types,
  log_scale,
  use_qc_metrics = FALSE,
  feature_map = NA,
  show_sum = NA,
  exclude_classes = NA,
  filter_data = FALSE,
  include_qualifier = FALSE,
  include_istd = FALSE,
  include_feature_filter = NA,
```

```

exclude_feature_filter = NA,
analysis_range = NA,
drop_empty_classes = TRUE,
density_strip = FALSE,
scale_factor = 1,
x_lim = NA,
x_label = NA,
y_axis_position = "right",
segment_width = 0.25,
font_base_size = NULL,
grid_major_color = "grey50",
grid_major_linewidth = 0.1,
grid_minor_color = "grey50",
grid_minor_linewidth = 0.1
)

```

Arguments

data	A MRMhubExperiment object.
variable	A character string indicating the variable to plot. For use_qc_metrics = FALSE, this must be a base name like "area" or "conc". For use_qc_metrics = TRUE, this is the name of a column in the metrics_qc table (e.g., "conc_median_spl").
qc_types	A character vector specifying the QC types to be averaged and plotted. Ignored if use_qc_metrics is TRUE, where the QC type is part of variable.
log_scale	A logical value indicating whether to use a log10 scale for the x-axis.
use_qc_metrics	A logical value. If FALSE (default), data is summarized on the fly from the main dataset. If TRUE, pre-calculated summary data is used from the metrics_qc table, which is much faster.
feature_map	A named character vector specifying feature classes, their order, and color. The order of the vector determines the y-axis order. Alternatively, can be a single string: lipidomics to use a default theme comprising a mapping for lipid classes frequently measured in lipidomics methods. When using the default theme, set drop_empty_classes = TRUE to avoid showing classes not present in the data. If NA (default), the default theme is used.
show_sum	A logical value indicating whether to plot a summary point (diamond shape) representing the mean of the summed abundances for each class. Defaults to TRUE for abundance-related variables and FALSE for others (e.g., "rt", "fwhm").
exclude_classes	A character vector of feature classes to be excluded from the plot.
filter_data	A logical value indicating whether to use all data (FALSE, default) or only QC-filtered data (TRUE, via filter_features_qc()).
include_qualifier	A logical value indicating whether to include qualifier features. Default is FALSE.
include_istd	A logical value indicating whether to include internal standard (ISTD) features. Default is FALSE.

include_feature_filter	Feature(s) to include by feature_id, as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
exclude_feature_filter	Feature(s) to exclude by feature_id, matched the same way as include_feature_filter. NA or "" ignores the filter.
analysis_range	A numeric vector of length 2 specifying the inclusive range of analysis_order. If NA (default), all analyses are included.
drop_empty_classes	A logical value. If TRUE (default), feature classes from feature_map that are not found in the data are removed from the plot. If FALSE, they are kept as empty rows.
density_strip	A logical value. If TRUE, a density strip showing the distribution of all features is added to the top of the plot.
scale_factor	A numeric value to scale the selected variable. Default is 1.
x_lim	A numeric vector of length 2 specifying the x-axis limits. If log_scale is TRUE, these are interpreted as exponents (e.g., c(2, 6) for 10^2 to 10^6). If NA, limits are determined automatically from the data.
x_label	A character string for the x-axis label. If NA, a default label is generated.
y_axis_position	A character string specifying the y-axis position ("left" or "right"). Default is "right".
segment_width	A numeric value for the linewidth of the feature segments. Default is 0.25.
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
grid_major_color	Color for major grid lines. Default is "grey50".
grid_major_linewidth	Linewidth for major grid lines. Default is 0.1.
grid_minor_color	Color for minor grid lines. Default is "grey50".
grid_minor_linewidth	Linewidth for minor grid lines. Default is 0.1.

Value

A ggplot object representing the abundance profile plot.

plot_calibrationcurves

Plot calibration curves

Description

This function plots calibration curves of each feature where defined and displays QC samples with defined concentrations within the plot. Users can select a regression model (linear or quadratic) and apply weighting (none, "1/x", "1/x^2", or "1/sqrt(x)"), either through function arguments or feature metadata.

Usage

```
plot_calibrationcurves(
  data = NULL,
  variable = "norm_intensity",
  qc_types = NA,
  fit_overwrite,
  fit_model = c("linear", "quadratic"),
  fit_weighting = c(NA, "none", "1/x", "1/x^2", "1/sqrt(x)"),
  ci_show = NA,
  ci_clip = TRUE,
  zoom_n_points = NA,
  log_scale = FALSE,
  filter_data = FALSE,
  include_qualifier = TRUE,
  include_istd = FALSE,
  include_feature_filter = NA,
  exclude_feature_filter = NA,
  output_pdf = FALSE,
  path = NA,
  create_dir = TRUE,
  return_plots = FALSE,
  point_size = NULL,
  point_color = NA,
  point_fill = NA,
  point_shape = NA,
  line_width = 0.7,
  line_color = "#4575b4",
  ribbon_fill = "#e6f6ff",
  font_base_size = NULL,
  rows_page = 4,
  cols_page = 5,
  specific_page = NA,
  page_orientation = "LANDSCAPE",
  page_width = NULL,
  page_height = NULL,
```

```

    page_units = "mm",
    show_progress = TRUE
)

```

Arguments

data	A MRHubExperiment object.
variable	Variable to plot on the y-axis, usually intensity. Default is "intensity".
qc_types	A character vector specifying the QC types to plot. It must contain at least CAL, which represents calibration curve samples. Other QC types will be plotted as points when they have assigned concentrations (see QC-concentration metadata). These QC types need to be present in the data and defined in the analysis metadata. The default is NA, which means any of the QC types "CAL", "HQC", "MQC", "LQC", "EQA", "QC", will be plotted if present and have assigned concentrations.
fit_overwrite	If TRUE, the function will use the provided fit_model and fit_weighting values for all analytes and ignore any fit method and weighting settings defined in the metadata. If omitted, the fit model and weighting stored in metrics_calibration (i.e. those used by <code>quantify_by_calibration()</code>) are plotted; this requires calibration results. When given, a warning is shown if the plotted fit differs from the stored one.
fit_model	A character string specifying the default regression fit method to use for the calibration curve. Must be one of "linear" or "quadratic". This method will be applied if no specific fit method is defined for a feature in the metadata, or when fit_overwrite = TRUE.
fit_weighting	A character string specifying the default weighting method for the regression points in the calibration curve. Must be one of "none", "1/x", "1/x^2", or "1/sqrt(x)". This method will be applied if no specific weighting method is defined for a feature in the metadata, or when fit_overwrite = TRUE.
ci_show	Logical, if TRUE, displays the confidence interval as ribbon. Default is NA, in which case confidence intervals are plotted in a linear scale and omitted in log-log scale.
ci_clip	Logical, if TRUE, clips the confidence interval above or below the highest and lowest data point, respectively.
zoom_n_points	Number of x lowest concentration points to display, used for zooming. Set to NULL or NA (default) to show all points.
log_scale	Logical. Determines whether the x and y axes are displayed in a logarithmic scale (log-log scale). Set to TRUE to enable logarithmic scaling; otherwise, set to FALSE for a linear scale. Note: If TRUE, any regression curves or standard error regions with negative values will be omitted from display.
filter_data	Logical, if TRUE, uses QC filtered data; otherwise uses raw data. Default is FALSE.
include_qualifier	Logical, whether to include qualifier features. Default is TRUE.
include_istd	Logical, whether to include internal standard (ISTD) features. Default is TRUE.

include_feature_filter	Feature(s) to include by feature_id, as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
exclude_feature_filter	Feature(s) to exclude by feature_id, matched the same way as include_feature_filter. NA or "" ignores the filter.
output_pdf	Logical, if TRUE, saves plots as a PDF file. Default is FALSE.
path	File path for saving the PDF. Default is an empty string.
create_dir	A logical value. If TRUE (the default) and output_pdf is TRUE, the parent directory of path is created if it does not yet exist.
return_plots	Logical, if TRUE, returns plots as a list of ggplot objects. Default is FALSE.
point_size	Size of points in the plot. Default is 1.5.
point_color	A vector specifying the colors for points corresponding to different QC types. This can be either an unnamed vector or a named vector, with names corresponding to QC types. Unused colors will be ignored. Default is NA which corresponds to the default colors for QC types defined in the package.
point_fill	A vector specifying the fill colors for points corresponding to different QC types. This can be either an unnamed vector or a named vector, with names corresponding to QC types. Unused fill colors will be ignored. Default is NA which corresponds to the default fill colors for QC types defined in the package.
point_shape	A vector specifying the shapes for points corresponding to different QC types. This can be either an unnamed vector or a named vector, with names corresponding to QC types. Unused shapes will be ignored. Default is NA which corresponds to the default shapes for QC types defined in the package.
line_width	Width of regression lines. Default is 0.7.
line_color	Color of the regression line. Default is "#4575b4".
ribbon_fill	Color for the confidence interval ribbon. Default is "#91bfdb40".
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by <code>mrmhub_set_plot_defaults()</code> if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
rows_page	Number of plot rows. Default is 4.
cols_page	Number of plot columns. Default is 5.
specific_page	Show/save a specific page number only. NA plots/saves all pages.
page_orientation	Orientation of PDF, either "LANDSCAPE" or "PORTRAIT". Default is "LANDSCAPE". Ignored when page_width and page_height are given.
page_width, page_height	Size of a PDF page, in page_units. Both must be given together. NULL (default) uses an A4 page of 280 x 200 mm, oriented by page_orientation. When an explicit size is given, page_orientation has no effect.
page_units	Unit of page_width and page_height: "mm" (default), "cm", "in" or "pt".
show_progress	Logical. If TRUE, displays a progress bar during plot creation.

Details

Features for plotting can be filtered using QC filters defined via `filter_features_qc()` or through `include_feature_filter` and `exclude_feature_filter` arguments. The resulting plots offer extensive customization options, including point size, line width, point color, point fill, point shape, line color, ribbon fill, and font base size.

Plots will be divided into multiple pages if the number of features exceeds the product of `rows_page` and `cols_page` settings. The function supports both direct plotting within R and saving plots as PDF files. Additionally, plots can be returned as a list of `ggplot2` objects for further manipulation or integration into other analyses.

Value

A list of `ggplot` objects if `return_plots = TRUE`, otherwise `NULL` (the plots are drawn to the active device or written to a PDF).

Preferred formats and devices

Purpose	Format	Device used	Typical dpi
Journal figure, vector (default choice)	"pdf"	<code>grDevices::cairo_pdf</code> , else <code>grDevices::pdf</code>	n/a
Figure for further editing (Illustrator, Inkscape)	"svg"	<code>svglite::svglite</code> , else <code>grDevices::svg</code>	n/a
Slides, Quarto HTML, GitHub	"png"	<code>ragg::agg_png</code> , else <code>grDevices::png</code>	150-300
Journal requiring raster submission	"tiff"	<code>ragg::agg_tiff</code> , else <code>grDevices::tiff</code>	300-600

Prefer a **vector** format (pdf, svg) for publication: text stays selectable and searchable, and lines stay sharp at any magnification.

Prefer a **raster** format (png, tiff) when a plot draws very many marks – a `plot_runscatter()` page covering several thousand analyses, or a dense `plot_pca()` score plot. Every point becomes a separate object in a PDF, so such figures produce very large files that are slow to open and to typeset. Saving them at 300-600 dpi instead keeps the file small with no visible loss.

The optional packages `ragg` and `svglite` are used automatically when installed, giving better text rendering, system-font support and smaller SVG files. When they are absent the equivalent `grDevices` device is used and the output is still correct. Installing both is recommended: `install.packages(c("ragg", "svglite"))`.

PDF output uses the `cairo` device wherever R was built with `cairo` support (`capabilities("cairo")`), because plain `grDevices::pdf()` writes text in a single-byte encoding and silently transliterates anything outside it – an en dash becomes `-`, `>=` replaces the proper symbol. Unit labels such as `umol/L` and statistical annotations routinely depend on those glyphs.

Multi-page output from the paged plot functions (`plot_runscatter()`, `plot_calibrationcurves()`, `plot_responsecurves()`, `plot_feature_correlations()`) is PDF only, which is the only format that holds many pages in one file. Use `save_plot()` for single figures in any of the other formats.

See Also

[save_plot\(\)](#) to save a single figure in any format.

Other calibration plots: [plot_responsecurves\(\)](#)

plot_feature_correlations

Plot highly correlated feature pairs

Description

Creates scatter plots for pairs of features that have correlations outside specified thresholds. Each pair is displayed in a separate facet with its correlation coefficient.

This plot can be used to visually inspect highly correlated features, that may represent duplicate identifications or represent isomers.

Correlations are Pearson's r over the analyses where both features have values; pairs sharing values in fewer than half of the analyses are skipped.

Usage

```
plot_feature_correlations(  
  data,  
  variable,  
  qc_types = NA,  
  cor_min,  
  cor_min_neg = -0.99,  
  log_scale = FALSE,  
  sort_by_corr = TRUE,  
  filter_data = FALSE,  
  include_qualifier = FALSE,  
  include_istd = FALSE,  
  include_feature_filter = NA,  
  exclude_feature_filter = NA,  
  min_median_value = NA,  
  output_pdf = FALSE,  
  path = NA,  
  create_dir = TRUE,  
  return_plots = FALSE,  
  rows_page = 4,  
  cols_page = 5,  
  specific_page = NA,  
  page_orientation = "LANDSCAPE",  
  page_width = NULL,  
  page_height = NULL,  
  page_units = "mm",  
  point_size = NULL,
```

```

    point_alpha = 0.8,
    point_stroke = 0.3,
    line_width = 0.5,
    line_color = "orange",
    line_alpha = 0.5,
    font_base_size = NULL,
    show_progress = TRUE
  )

```

Arguments

data	A MRMhubExperiment object.
variable	A character string indicating the signal variable to plot. Must be one of: "area", "height", "intensity", "norm_intensity", "response", "conc", "conc_raw", "rt", "fwhm".
qc_types	A character vector specifying the QC types to plot. It must contain at least one element. The default NA plots any of the non-blank QC types ("SPL", "TQC", "BQC", "HQC", "MQC", "LQC", "NIST", "LTR") present in the dataset. A single value that is a QC type is matched exactly; any other single value is a regular expression, e.g. "QC\$" for all QC types ending in "QC".
cor_min	Numeric. Minimum correlation threshold. Only feature pairs with positive correlations above this value will be shown. Set to Inf to exclude positive correlations.
cor_min_neg	Numeric. Minimum negative correlation threshold. Only feature pairs with negative correlations above this value will be shown. Set to -Inf to exclude negative correlations.
log_scale	A logical value indicating whether to use a log10 scale for both axes. Default is FALSE.
sort_by_corr	A logical value indicating whether to sort the features in the plot by correlation or alphabetically by feature ID. Default is TRUE.
filter_data	A logical value indicating whether to use all data (default) or only QC-filtered data (filtered via filter_features_qc()).
include_qualifier	A logical value indicating whether to include qualifier features. Default is FALSE.
include_istd	A logical value indicating whether to include internal standard (ISTD) features. Default is FALSE.
include_feature_filter	Feature(s) to include by feature_id, as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
exclude_feature_filter	Feature(s) to exclude by feature_id, matched the same way as include_feature_filter. NA or "" ignores the filter.

min_median_value	Minimum median feature value across the selected QC-type samples required for a feature to be included. NA (default) applies no filtering. This is a fast way to exclude noisy features; for principled QC-based filtering use filter_features_qc() .
output_pdf	If TRUE, saves the generated plots as a PDF file. When FALSE, plots are directly plotted.
path	The file path for saving the PDF. Must be defined if output_pdf is TRUE.
create_dir	A logical value. If TRUE (the default) and output_pdf is TRUE, the parent directory of path is created if it does not yet exist.
return_plots	Logical. If TRUE, returns the plots as a list of ggplot objects.
rows_page	Number of rows of plots per page.
cols_page	Number of columns of plots per page.
specific_page	An integer specifying a specific page to plot. If NA (default), all pages are plotted.
page_orientation	Orientation of the PDF paper: "LANDSCAPE" or "PORTRAIT". Ignored when page_width and page_height are given.
page_width, page_height	Size of a PDF page, in page_units. Both must be given together. NULL (default) uses an A4 page of 280 x 200 mm, oriented by page_orientation. When an explicit size is given, page_orientation has no effect.
page_units	Unit of page_width and page_height: "mm" (default), "cm", "in" or "pt".
point_size	A numeric value indicating the size of points in millimeters. Default is 1.
point_alpha	A numeric value indicating the transparency of points (0-1). Default is 0.8.
point_stroke	A numeric value indicating the stroke width of the points. Default is 0.3.
line_width	A numeric value indicating the size of the correlation line. Default is 0.5.
line_color	A character string indicating the color of the correlation line. Default is orange.
line_alpha	A numeric value indicating the transparency of the correlation line (0-1). Default is 0.5.
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
show_progress	Logical. If TRUE, displays a progress bar during plot creation.

Value

A ggplot object showing scatter plots of highly correlated feature pairs. Returns NULL if no correlations meet the threshold criteria.

Preferred formats and devices

Purpose	Format	Device used	Typical dpi
Journal figure, vector (default choice)	"pdf"	grDevices::cairo_pdf, else grDevices::pdf	n/a

Figure for further editing (Illustrator, Inkscape)	"svg"	svglite::svglite, else grDevices::svg	n/a
Slides, Quarto HTML, GitHub	"png"	ragg::agg_png, else grDevices::png	150-300
Journal requiring raster submission	"tiff"	ragg::agg_tiff, else grDevices::tiff	300-600

Prefer a **vector** format (pdf, svg) for publication: text stays selectable and searchable, and lines stay sharp at any magnification.

Prefer a **raster** format (png, tiff) when a plot draws very many marks – a [plot_runscatter\(\)](#) page covering several thousand analyses, or a dense [plot_pca\(\)](#) score plot. Every point becomes a separate object in a PDF, so such figures produce very large files that are slow to open and to typeset. Saving them at 300-600 dpi instead keeps the file small with no visible loss.

The optional packages `ragg` and `svglite` are used automatically when installed, giving better text rendering, system-font support and smaller SVG files. When they are absent the equivalent `grDevices` device is used and the output is still correct. Installing both is recommended: `install.packages(c("ragg", "svglite"))`.

PDF output uses the `cairo` device wherever R was built with `cairo` support (`capabilities("cairo")`), because plain `grDevices::pdf()` writes text in a single-byte encoding and silently transliterates anything outside it – an en dash becomes `-`, `>=` replaces the proper symbol. Unit labels such as `umol/L` and statistical annotations routinely depend on those glyphs.

Multi-page output from the paged plot functions ([plot_runscatter\(\)](#), [plot_calibrationcurves\(\)](#), [plot_responsecurves\(\)](#), [plot_feature_correlations\(\)](#)) is PDF only, which is the only format that holds many pages in one file. Use [save_plot\(\)](#) for single figures in any of the other formats.

See Also

[save_plot\(\)](#) to save a single figure in any format.

Other QC plots: [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_interference_impact\(\)](#), [plot_qc_summary_byclass\(\)](#), [plot_qc_summary_overall\(\)](#), [plot_qcmetrics_comparison\(\)](#), [plot_rla_boxplot\(\)](#), [plot_rt_vs_chain\(\)](#), [plot_runscatter\(\)](#), [plot_runsequence\(\)](#)

plot_interference_correction

Plot the results of interference correction

Description

This function generates grouped standardized beeswarm plots to visualize the results of interference correction across different QC types.

Usage

```
plot_interference_correction(
  data,
  qc_types = c("SPL", "TQC", "PBLK", "BQC"),
  include_qualifier = FALSE,
  include_istd = TRUE,
  include_feature_filter = NA,
  exclude_feature_filter = NA,
  min_median_value = NA,
  min_correction_pct = NA,
  sort_by_effect = c("none", "desc", "asc"),
  top_n = NA,
  y_lim = c(-NA, NA),
  point_size = NULL,
  dodge_width = 0.6,
  point_alpha = 0.3,
  box_alpha = 0.3,
  box_linewidth = 0.5,
  font_base_size = NULL,
  legend_position = NULL,
  legend_size = NULL,
  show_legend_title = NULL,
  title = NULL,
  angle_x = 45
)
```

Arguments

<code>data</code>	A MRHubExperiment object.
<code>qc_types</code>	A character vector specifying the QC types to plot. It must contain at least one element. The default NA plots any of the non-blank QC types ("SPL", "TQC", "BQC", "HQC", "MQC", "LQC", "NIST", "LTR") present in the dataset. A single value that is a QC type is matched exactly; any other single value is a regular expression, e.g. "QC\$" for all QC types ending in "QC".
<code>include_qualifier</code>	A logical value indicating whether to include qualifier features. Default is FALSE.
<code>include_istd</code>	A logical value indicating whether to include internal standards (ISTD) features. Default is TRUE.
<code>include_feature_filter</code>	Feature(s) to include by <code>feature_id</code> , as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
<code>exclude_feature_filter</code>	Feature(s) to exclude by <code>feature_id</code> , matched the same way as <code>include_feature_filter</code> . NA or "" ignores the filter.

min_median_value	Median raw-signal abundance floor: drop features whose median feature_intensity across the selected QC types is below this value. NA (default) applies no threshold. Use to hide low-signal features.
min_correction_pct	Keep only features whose median correction (percent of raw signal removed, across the selected QC types) is at least this value. NA (default) applies no threshold. Use to focus the plot on substantially-corrected features.
sort_by_effect	Order the x-axis by correction effect, defined per feature as the deviation of its median % change from 100% (pooled across the displayed QC-type points). One of "none" (default, alphabetical), "desc" (largest effect first) or "asc".
top_n	Keep only the top_n features with the largest correction effect (see sort_by_effect). NA (default) keeps all. Applied after the min_median_value / min_correction_pct filters.
y_lim	A numeric vector of length 2 specifying the y-axis limits.
point_size	A numeric value indicating the size of points in millimeters. Default is 0.5.
dodge_width	Numeric. Width used to dodge overlapping points by qc_type. Default is 0.6.
point_alpha	Numeric. Transparency of the plotted points. Default is 0.3.
box_alpha	Numeric. Transparency of the boxplot. Default is 0.3.
box_linewidth	Numeric. Width of the boxplot lines. Default is 0.5.
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by <code>mrmhub_set_plot_defaults()</code> if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> .
legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
angle_x	Numeric. Angle of the x-axis text labels. Default is 45.

Value

A ggplot object showing the grouped standardized beeswarm plot.

See Also

Other QC plots: `plot_feature_correlations()`, `plot_matrixeffects()`, `plot_normalization_qc()`, `plot_pca()`, `plot_pca_loading()`, `plot_qc_interference_impact()`, `plot_qc_summary_byclass()`, `plot_qc_summary_overall()`, `plot_qcmetrics_comparison()`, `plot_rla_boxplot()`, `plot_rt_vs_chain()`, `plot_runscatter()`, `plot_runsequence()`

<code>plot_matrixeffects</code>	<i>Plot standardized feature intensities grouped by QC type</i>
---------------------------------	---

Description

This function creates a grouped beeswarm plot of standardized feature intensities, where each value is shown as a percentage of the feature's median over the plotted non-blank analyses (per batch with `batchwise_normalization = TRUE`). Points are grouped by `qc_type` and spread using quasirandom jitter. An ISTD signal higher in process blanks (ISTD without matrix) than in the samples may indicate ion suppression by the sample matrix. The spread between study samples and pooled QCs can reflect sample-to-sample differences in matrix effects.

Usage

```
plot_matrixeffects(
  data,
  variable = "intensity",
  qc_types = c("SPL", "TQC", "PBLK", "BQC"),
  batchwise_normalization = TRUE,
  include_qualifier = FALSE,
  only_istd = TRUE,
  include_feature_filter = NA,
  exclude_feature_filter = NA,
  min_median_value = NA,
  y_lim = c(-NA, NA),
  point_size = NULL,
  dodge_width = 0.6,
  point_alpha = 0.3,
  box_alpha = 0.3,
  box_linewidth = 0.5,
  font_base_size = NULL,
  legend_position = NULL,
  legend_size = NULL,
  show_legend_title = NULL,
  title = NULL,
  angle_x = 45
)
```

Arguments

<code>data</code>	A <code>MRMhubExperiment</code> object.
<code>variable</code>	A character string indicating the signal variable to plot. Must be one of: "area", "height", "intensity", "norm_intensity", "response", "conc", "conc_raw", "rt", "fwhm".
<code>qc_types</code>	A character vector specifying the QC types to plot. It must contain at least one element. The default NA plots any of the non-blank QC types ("SPL", "TQC", "BQC", "HQC", "MQC", "LQC", "NIST", "LTR") present in the dataset. A single value that is a QC type is matched exactly; any other single value is a regular expression, e.g. "QC\$" for all QC types ending in "QC".
<code>batchwise_normalization</code>	A logical value indicating whether to normalize the signals by batch instead of globally.
<code>include_qualifier</code>	A logical value indicating whether to include qualifier features. Default is FALSE.
<code>only_istd</code>	A logical value indicating whether to show only internal standard (ISTD) features. Default is TRUE. Set to FALSE in combination with <code>feature_filter</code> parameters to show other features.
<code>include_feature_filter</code>	Feature(s) to include by <code>feature_id</code> , as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
<code>exclude_feature_filter</code>	Feature(s) to exclude by <code>feature_id</code> , matched the same way as <code>include_feature_filter</code> . NA or "" ignores the filter.
<code>min_median_value</code>	Minimum median feature value across the selected QC-type samples required for a feature to be included. NA (default) applies no filtering. This is a fast way to exclude noisy features; for principled QC-based filtering use filter_features_qc() .
<code>y_lim</code>	A numeric vector of length 2 specifying the y-axis limits.
<code>point_size</code>	A numeric value indicating the size of points in millimeters. Default is 0.5.
<code>dodge_width</code>	Numeric. Width used to dodge overlapping points by <code>qc_type</code> . Default is 0.6.
<code>point_alpha</code>	Numeric. Transparency of the plotted points. Default is 0.3.
<code>box_alpha</code>	Numeric. Transparency of the boxplot. Default is 0.3.
<code>box_linewidth</code>	Numeric. Width of the boxplot lines. Default is 0.5.
<code>font_base_size</code>	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
<code>legend_position</code>	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with mrmhub_set_plot_defaults() .

legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
angle_x	Numeric. Angle of the x-axis text labels. Default is 45.

Details

Include at least one QC type besides the study samples: the reference is computed from the plotted analyses, so with study samples only they centre on 100% by construction. The plot compares ISTD signals between samples; it does not measure the matrix effect itself.

Value

A ggplot object showing the grouped standardized beeswarm plot.

See Also

Other QC plots: `plot_feature_correlations()`, `plot_interference_correction()`, `plot_normalization_qc()`, `plot_pca()`, `plot_pca_loading()`, `plot_qc_interference_impact()`, `plot_qc_summary_byclass()`, `plot_qc_summary_overall()`, `plot_qcmetrics_comparison()`, `plot_rla_boxplot()`, `plot_rt_vs_chain()`, `plot_runscatter()`, `plot_runsequence()`

plot_normalization_qc	<i>Compare feature variability before and after normalization</i>
-----------------------	---

Description

Evaluates the effectiveness of normalization by comparing feature variability (measured as %CV) in QC and/or study samples before and after normalization. The comparison is visualized through one of three plot types:

- Scatter plot: CV values before vs after normalization
- Difference plot: (CV after - CV before) vs mean CV
- Ratio plot: log2 of (CV after / CV before) vs mean CV

Features can be grouped and visualized by their feature class using facets.
The resulting visualization helps assess whether normalization improved measurement precision across different features and sample/QC types.

Usage

```
plot_normalization_qc(
  data = NULL,
  before_norm_var,
  after_norm_var,
  plot_type,
  qc_types = NA,
  facet_by_class = FALSE,
  y_shared = FALSE,
  filter_data = FALSE,
  include_qualifier = FALSE,
  include_istd = FALSE,
  cv_threshold_value = 25,
  x_lim = c(NA_real_, NA_real_),
  y_lim = c(NA_real_, NA_real_),
  cols_page = 5,
  point_size = NULL,
  point_alpha = 0.5,
  font_base_size = NULL,
  autoscale = TRUE,
  legend_position = NULL,
  legend_size = NULL,
  show_legend_title = NULL,
  title = NULL,
  strip_text_size = NULL,
  strip_bg_color = NULL,
  legend_bg_alpha = NULL
)
```

Arguments

<code>data</code>	A MRMhubExperiment object.
<code>before_norm_var</code>	A string specifying the variable from the QC metrics table to be used for the x-axis (before normalization).
<code>after_norm_var</code>	A string specifying the variable from the QC metrics table to be used for the y-axis (after normalization).
<code>plot_type</code>	A character string specifying the type of plot to generate. Must be one of "scatter", "diff", or "ratio". Selecting "scatter" plots the before and after normalization CV values as a scatter plot, "diff" plots the difference between the two CV values against the average CV, and "ratio" plots the log2 ratio of the two CV values against the average CV.
<code>qc_types</code>	A character vector specifying the QC types to plot. It must contain at least one element. The default NA plots any of the non-blank QC types ("SPL", "TQC", "BQC", "HQC", "MQC", "LQC", "NIST", "LTR") present in the dataset. A single value that is a QC type is matched exactly; any other single value is a regular expression, e.g. "QC\$" for all QC types ending in "QC".

facet_by_class	If TRUE, facets the plot by feature_class, as defined in the feature metadata.
y_shared	Logical; if TRUE, all facets share the same y-axis scale. If FALSE (default), each facet has its own y-axis scale.
filter_data	Whether to use all data (default) or only QC-filtered data (filtered via filter_features_qc()).
include_qualifier	Whether to include qualifier features (default is FALSE).
include_istd	Whether to include internal standards (ISTDs) (default is FALSE).
cv_threshold_value	Numerical threshold value to be shown as dashed lines in the plot (default is 25).
x_lim	Numeric vector of length 2 for x-axis limits. Use NA for auto-scaling (default is c(NA, NA)).
y_lim	Numeric vector of length 2 for y-axis limits. Use NA for auto-scaling (default is c(NA, NA)). A fixed lower limit of 0 would clip the negative values of plot_type = "diff" and "ratio", which are the features whose CV the normalization reduced.
cols_page	Number of facet columns per page, representing different feature classes (default is 5). Only used if facet_by_class = TRUE.
point_size	Size of points in millimeters (default is 1).
point_alpha	Transparency of points (default is 0.5).
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
autoscale	Logical. When TRUE (default), font_base_size and point_size left as NULL are sized automatically from cols_page (more facet columns per page give smaller text and points). Any value passed explicitly always takes precedence. When FALSE, unset sizes fall back to the single-plot defaults.
legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with mrmhub_set_plot_defaults() .
legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with mrmhub_set_plot_defaults() ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
strip_text_size	Optional facet strip text size, as a multiplier of font_base_size (when <= 3) or an absolute point size (when > 3). NULL (default) inherits from font_base_size.

- `strip_bg_color` Optional facet strip background fill colour. The strip text colour is set automatically for contrast (white on a dark fill, black on a light one). NULL (default) keeps the house dark-navy strips.
- `legend_bg_alpha` Optional opacity ([0, 1]) of a white legend background box, useful for a readable inside legend drawn over points. NULL (default) leaves the legend background unchanged.

Details

The function preselects the corresponding variables from the QC metrics and uses `plot_qcmetrics_comparison()` to visualize the results.

- The data must be normalized before using `normalize_by_istd()` followed by calculation of the QC metrics table via `calc_qc_metrics()` or `filter_features_qc()`, see examples below.
- When `facet_by_class = TRUE`, then the `feature_class` must be defined in the metadata or retrieved via specific functions, e.g., `parse_lipid_feature_names()`.

Value

A ggplot object representing the scatter plot comparing CV values before and after normalization.

See Also

`plot_qcmetrics_comparison()`, `calc_qc_metrics()`, `filter_features_qc()`,

Other QC plots: `plot_feature_correlations()`, `plot_interference_correction()`, `plot_matrixeffects()`, `plot_pca()`, `plot_pca_loading()`, `plot_qc_interference_impact()`, `plot_qc_summary_byclass()`, `plot_qc_summary_overall()`, `plot_qcmetrics_comparison()`, `plot_rla_boxplot()`, `plot_rt_vs_chain()`, `plot_runscatter()`, `plot_runsequence()`

Examples

```
# Example usage:
mexp <- lipidomics_dataset
mexp <- normalize_by_istd(mexp)
mexp <- calc_qc_metrics(mexp)
plot_normalization_qc(
  data = mexp,
  before_norm_var = "intensity",
  after_norm_var = "norm_intensity",
  plot_type = "scatter",
  qc_types = "SPL",
  filter_data = FALSE,
  facet_by_class = TRUE,
  cv_threshold_value = 25
)
```

plot_pca*PCA plot for quality control*

Description

Generates a Principal Component Analysis (PCA) plot for visualizing samples including quality control (QC) samples. This function provides options for filtering data, applying transformations, and labelling of outliers.

Experimental batches can be visualized as ellipses to assess batch effects.

This function returns a ggplot object. Identified outliers can be printed to the console.

Usage

```
plot_pca(  
  data = NULL,  
  variable,  
  qc_types = NA,  
  ellipse_variable = "qc_type",  
  ellipse_levels = NA,  
  pca_dims = c(1, 2),  
  log_transform = TRUE,  
  filter_data = FALSE,  
  include_qualifier = FALSE,  
  include_istd = FALSE,  
  include_feature_filter = NA,  
  exclude_feature_filter = NA,  
  min_median_value = NA,  
  show_labels = TRUE,  
  labels_column = "analysis_id",  
  labels_threshold_mad = 3,  
  shared_labeltext_hide = NA,  
  label_font_size = 3,  
  point_size = NULL,  
  point_alpha = 0.7,  
  font_base_size = NULL,  
  legend_position = NULL,  
  legend_size = NULL,  
  show_legend_title = NULL,  
  title = NULL,  
  strip_text_size = NULL,  
  strip_bg_color = NULL,  
  legend_bg_alpha = NULL,  
  aspect_ratio = 1,  
  ellipse_confidence_level = 0.95,  
  ellipse_linewidth = 1,  
  ellipse_fill = TRUE,
```

```

    ellipse_fillcolor = NA,
    ellipse_alpha = 0.1
)

```

Arguments

data	A MRMhubExperiment object.
variable	A character string indicating the variable to use for PCA analysis. Must be one of: "area", "height", "intensity", "norm_intensity", "response", "conc", "conc_raw", "rt", "fwhm".
qc_types	A character vector specifying the QC types to plot. It must contain at least one element. The default NA plots any of the non-blank QC types ("SPL", "TQC", "BQC", "HQC", "MQC", "LQC", "NIST", "LTR") present in the dataset. A single value that is a QC type is matched exactly; any other single value is a regular expression, e.g. "QC\$" for all QC types ending in "QC".
ellipse_variable	String specifying which sample variable to show as ellipses. Must be one of: "none", "qc_type", "batch_id". "none" omits ellipses.
ellipse_levels	A character vector specifying the levels of ellipse_variable to display as ellipses.
pca_dims	A numeric vector of length 2 indicating the PCA dimensions to plot. Default is c(1, 2).
log_transform	A logical value indicating whether to log-transform the data before the PCA. Default is TRUE.
filter_data	A logical value indicating whether to use all data (default) or only QC-filtered data (filtered via filter_features_qc()).
include_qualifier	A logical value indicating whether to include qualifier features. Default is TRUE.
include_istd	A logical value indicating whether to include internal standard (ISTD) features. Default is TRUE.
include_feature_filter	Feature(s) to include by feature_id, as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
exclude_feature_filter	Feature(s) to exclude by feature_id, matched the same way as include_feature_filter. NA or "" ignores the filter.
min_median_value	Minimum median feature value across the selected QC-type samples required for a feature to be included. NA (default) applies no filtering. This is a fast way to exclude noisy features; for principled QC-based filtering use filter_features_qc() .
show_labels	A logical value indicating whether to show analysis_id labels for points outside k * MAD of the selected PCA dimensions. Default is TRUE.
labels_column	A character string indicating the column to be used for the point labels. Typically "analysis_id" or "analysis_order". Default is "analysis_id".

labels_threshold_mad	A numeric value determining the threshold for showing labels based on the median absolute deviation (MAD): samples whose score on either shown PC lies more than this many MADs from the median are labelled. Default is 3. Set to NULL to suppress labels.
shared_labeltext_hide	A character string representing text shared across labels to be hidden (case-sensitive). If this results in non-unique analysis_id's, an error will be raised.
label_font_size	Number indicating the font size for labels in 'mm'. Note the unit is different from font_base_size that is in 'pt'.
point_size	A numeric value indicating the size of points in millimeters. Default is 2.
point_alpha	A numeric value indicating the transparency of points (0-1). Default is 0.5.
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with mrmhub_set_plot_defaults() .
legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with mrmhub_set_plot_defaults() ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
strip_text_size	Optional facet strip text size, as a multiplier of font_base_size (when <= 3) or an absolute point size (when > 3). NULL (default) inherits from font_base_size.
strip_bg_color	Optional facet strip background fill colour. The strip text colour is set automatically for contrast (white on a dark fill, black on a light one). NULL (default) keeps the house dark-navy strips.
legend_bg_alpha	Optional opacity ([0, 1]) of a white legend background box, useful for a readable inside legend drawn over points. NULL (default) leaves the legend background unchanged.
aspect_ratio	Panel aspect ratio (height/width). Default 1 gives a square score plot (PC1/PC2 on the same visual scale); NULL leaves it free.
ellipse_confidence_level	A numeric value indicating the confidence level for the ellipses. Default is 0.95.
ellipse_linewidth	A numeric value indicating the line width of the ellipses. Default is 1.

`ellipse_fill` A logical value indicating whether to fill the ellipses.

`ellipse_fillcolor` A vector specifying the fill colors for ellipse corresponding to different `ellipse_variable` levels. This can be either an unnamed vector or a named vector, with names corresponding to levels in `ellipse_variable`. Unused fill colors will be ignored. Default is NA which corresponds to the default fill colors in case of `ellipse_variable = qc_type`, and to automatically generated colors otherwise.

`ellipse_alpha` A numeric value indicating the transparency of the ellipse fill (0-1). Default is 0.3.

Value

A ggplot object with the PCA plot

See Also

Other QC plots: [plot_feature_correlations\(\)](#), [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_interference_impact\(\)](#), [plot_qc_summary_byclass\(\)](#), [plot_qc_summary_overall\(\)](#), [plot_qcmetrics_comparison\(\)](#), [plot_rla_boxplot\(\)](#), [plot_rt_vs_chain\(\)](#), [plot_runscatter\(\)](#), [plot_runsequence\(\)](#)

plot_pca_loading	<i>Plot PCA loadings</i>
------------------	--------------------------

Description

Generates a plot of PCA loadings, illustrating the contribution of features to each principal component. This function can be used to investigate which feature (groups) are contributing to the variance seen in the plot and which need further investigation.

Usage

```
plot_pca_loading(
  data = NULL,
  variable,
  qc_types = NA,
  pca_dims = c(1, 2, 3, 4),
  log_transform = TRUE,
  top_n = 30,
  verticalBars = FALSE,
  abs_loading = TRUE,
  filter_data = FALSE,
  include_qualifier = FALSE,
  include_istd = FALSE,
  include_feature_filter = NA,
  exclude_feature_filter = NA,
```

```

    min_median_value = NA,
    font_base_size = NULL,
    legend_position = NULL,
    legend_size = NULL,
    show_legend_title = NULL,
    title = NULL,
    strip_text_size = NULL,
    strip_bg_color = NULL,
    legend_bg_alpha = NULL
  )

```

Arguments

<code>data</code>	A MRMhubExperiment object.
<code>variable</code>	A character string indicating the variable to use for PCA analysis. Must be one of: "area", "height", "intensity", "norm_intensity", "response", "conc", "conc_raw", "rt", "fwhm".
<code>qc_types</code>	A character vector specifying the QC types to plot. It must contain at least one element. The default NA plots any of the non-blank QC types ("SPL", "TQC", "BQC", "HQC", "MQC", "LQC", "NIST", "LTR") present in the dataset. A single value that is a QC type is matched exactly; any other single value is a regular expression, e.g. "QC\$" for all QC types ending in "QC".
<code>pca_dims</code>	A numeric vector indicating for which PCA dimensions the loadings should be shown. Default is c(1, 2, 3, 4).
<code>log_transform</code>	A logical value indicating whether to log-transform the data before the PCA. Default is TRUE.
<code>top_n</code>	Number of top features with highest absolute loading that will be shown for each PC dimension. Default is 30.
<code>verticalBars</code>	Show vertical bars instead of horizontal bars in the plot. Default is FALSE.
<code>abs_loading</code>	Show absolute loading values instead of signed loadings. Default is TRUE.
<code>filter_data</code>	A logical value indicating whether to use all data (default) or only QC-filtered data (filtered via <code>filter_features_qc()</code>).
<code>include_qualifier</code>	A logical value indicating whether to include qualifier features. Default is FALSE.
<code>include_istd</code>	A logical value indicating whether to include internal standard (ISTD) features. Default is FALSE.
<code>include_feature_filter</code>	Feature(s) to include by <code>feature_id</code> , as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
<code>exclude_feature_filter</code>	Feature(s) to exclude by <code>feature_id</code> , matched the same way as <code>include_feature_filter</code> . NA or "" ignores the filter.

<code>min_median_value</code>	Minimum median feature value across the selected QC-type samples required for a feature to be included. NA (default) applies no filtering. This is a fast way to exclude noisy features; for principled QC-based filtering use <code>filter_features_qc()</code> .
<code>font_base_size</code>	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by <code>mrmhub_set_plot_defaults()</code> if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
<code>legend_position</code>	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> .
<code>legend_size</code>	Optional single multiplier of <code>font_base_size</code> (when ≤ 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
<code>show_legend_title</code>	Logical. NULL (default) keeps the legend title, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> ; FALSE hides it, TRUE forces it shown.
<code>title</code>	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
<code>strip_text_size</code>	Optional facet strip text size, as a multiplier of <code>font_base_size</code> (when ≤ 3) or an absolute point size (when > 3). NULL (default) inherits from <code>font_base_size</code> .
<code>strip_bg_color</code>	Optional facet strip background fill colour. The strip text colour is set automatically for contrast (white on a dark fill, black on a light one). NULL (default) keeps the house dark-navy strips.
<code>legend_bg_alpha</code>	Optional opacity ([0, 1]) of a white legend background box, useful for a readable inside legend drawn over points. NULL (default) leaves the legend background unchanged.

Value

A ggplot object with PCA loadings plot

See Also

Other QC plots: `plot_feature_correlations()`, `plot_interference_correction()`, `plot_matrixeffects()`, `plot_normalization_qc()`, `plot_pca()`, `plot_qc_interference_impact()`, `plot_qc_summary_byclass()`, `plot_qc_summary_overall()`, `plot_qcmetrics_comparison()`, `plot_rla_boxplot()`, `plot_rt_vs_chain()`, `plot_runscatter()`, `plot_runsequence()`

plot_qc_interference_impact

Plot the magnitude of interference correction as a histogram

Description

Shows how many features were affected by which magnitude of interference correction, as a histogram of the per-feature correction impact (percent of raw signal removed) within one or more QC types (study samples by default). The data must already be interference-corrected. Features with no impact are excluded and reported.

Usage

```
plot_qc_interference_impact(
  data,
  qc_types = "SPL",
  include_qualifier = FALSE,
  include_istd = TRUE,
  include_feature_filter = NA,
  exclude_feature_filter = NA,
  min_correction_pct = NA,
  binwidth = NA,
  font_base_size = 8
)
```

Arguments

data	A MRMhubExperiment (already interference-corrected).
qc_types	QC type(s) to summarize. Default "SPL" (study samples). Set to NA to use all non-blank sample/QC types.
include_qualifier	Include qualifier features. Default FALSE.
include_istd	Include internal standards. Default TRUE.
include_feature_filter	Feature(s) to include by feature_id, as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
exclude_feature_filter	Feature(s) to exclude by feature_id, matched the same way as include_feature_filter. NA or "" ignores the filter.
min_correction_pct	Keep only features whose median correction (percent of raw signal removed) is at least this value. NA (default) keeps every feature with a positive impact. Use to drop negligible corrections.
binwidth	Histogram bin width (percent). Default NA (30 bins).
font_base_size	Base font size. Default 8.

Value

A ggplot object: feature count vs. percent of signal removed.

See Also

Other QC plots: [plot_feature_correlations\(\)](#), [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_summary_byclass\(\)](#), [plot_qc_summary_overall\(\)](#), [plot_qcmetrics_comparison\(\)](#), [plot_rla_boxplot\(\)](#), [plot_rt_vs_chain\(\)](#), [plot_runscatter\(\)](#), [plot_runsequence\(\)](#)

plot_qc_summary_byclass

Plot QC filtering summary by feature class

Description

This function provides a summary of feature QC filtering based on feature class, showing the number of features that passed or failed various quality control criteria. It visualizes the filtering in a hierarchical sequence. Features are first evaluated against lower-level filters such as signal-to-blank (S/B) ratios and minimum intensity, followed by higher-level filters like the coefficient of variation (CV) or linear regression results. This means that a feature is classified as failing a given criterion (e.g., CV) only if it has passed all hierarchically lower filters (e.g., S/B ratio and minimum intensity). Each feature is counted once; features retained via `features.to.keep` despite failing are shown as "QC failed, kept".

Usage

```
plot_qc_summary_byclass(
  data = NULL,
  font_base_size = NULL,
  legend_position = NULL,
  legend_size = NULL,
  show_legend_title = NULL,
  title = NULL
)
```

Arguments

<code>data</code>	A MRMhubExperiment object.
<code>font_base_size</code>	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).

legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with mrmhub_set_plot_defaults() .
legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with mrmhub_set_plot_defaults() ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.

Value

A ggplot object showing the feature QC filtering summary by feature class.

See Also

[plot_qc_summary_overall\(\)](#) for an overall summary plot [filter_features_qc\(\)](#) for comparing QC metrics

Other QC plots: [plot_feature_correlations\(\)](#), [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_interference_impact\(\)](#), [plot_qc_summary_overall\(\)](#), [plot_qcmetrics_comparison\(\)](#), [plot_rla_boxplot\(\)](#), [plot_rt_vs_chain\(\)](#), [plot_runscatter\(\)](#), [plot_runsequence\(\)](#)

plot_qc_summary_overall

Plot overall QC filtering summary

Description

This function generates a summary of the feature QC filtering process, visualizing the number of features that passed or failed the various QC criteria. The bars apply the criteria hierarchically: a feature is counted once, under the first criterion it fails, and features retained via `features.to.keep` despite failing are shown as "QC failed, kept". See [plot_qc_summary_byclass\(\)](#) for more information. The optional Venn diagram shows, for the features passing the missing-value and minimum-intensity criteria, the overlap of features failing the signal-to-blank, CV and linearity criteria; unlike the bars, it is not hierarchical.

Usage

```
plot_qc_summary_overall(data = NULL, with_venn = TRUE, font_base_size = NULL)
```

Arguments

data	A MRMhubExperiment object.
with_venn	Whether to include a Venn diagram summarizing the features excluded due to different QC criteria. Default is TRUE.
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).

Details

The QC filtering process follows a hierarchical structure, where features are first evaluated against lower-level filters such as signal-to-blank ratios and minimum intensity. Only features that pass these basic criteria are then subjected to higher-level filters like the coefficient of variation (CV) or linear regression results. A feature will only fail a higher-level filter (such as CV or R^2) if it has passed all previous lower-level filters. This ensures that features are evaluated progressively, starting from fundamental quality checks up to more stringent filtering criteria.

Note: The function currently shows a warning Using size aesthetic for lines was deprecated in ggplot2 3.4.0. which can be ignored.

Value

A ggplot object showing the feature QC filtering summary with or without a Venn diagram.

See Also

Other QC plots: [plot_feature_correlations\(\)](#), [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_interference_impact\(\)](#), [plot_qc_summary_byclass\(\)](#), [plot_qcmetrics_comparison\(\)](#), [plot_rla_boxplot\(\)](#), [plot_rt_vs_chain\(\)](#), [plot_runscatter\(\)](#), [plot_runsequence\(\)](#)

plot_qcmetrics_comparison

Comparison of two feature QC metrics variables

Description

This function generates scatter plots comparing two QC metrics variables across feature classes. A list of available QC metrics is available from the [calc_qc_metrics\(\)](#) documentation.

Usage

```
plot_qcmetrics_comparison(
  data = NULL,
  plot_type,
  x_variable,
  y_variable,
  qc_types = NA,
  facet_by_class = FALSE,
  y_shared = FALSE,
  filter_data = FALSE,
  include_qualifier = FALSE,
  include_istd = FALSE,
  equality_line = FALSE,
  threshold_values = NA_real_,
  log_scale = FALSE,
  x_lim = c(NA_real_, NA_real_),
  y_lim = c(NA_real_, NA_real_),
  cols_page = 5,
  point_size = NULL,
  point_alpha = 0.5,
  point_color = "#0460acff",
  point_fill = "#4da2e7ff",
  point_shape = 21,
  point_stroke = 0.5,
  font_base_size = NULL,
  autoscale = TRUE,
  legend_position = NULL,
  legend_size = NULL,
  show_legend_title = NULL,
  title = NULL,
  strip_text_size = NULL,
  strip_bg_color = NULL,
  legend_bg_alpha = NULL
)
```

Arguments

<code>data</code>	A MRMhubExperiment object.
<code>plot_type</code>	A character string specifying the type of plot to generate. Must be one of "scatter", "diff", or "ratio". Selecting "scatter" plots the "y_variable" against the "x_variable" values as a scatter plot, "diff" plots the difference between the two values against the average value, and "ratio" plots the log2 ratio of the two values against the average value.
<code>x_variable</code>	The name of the QC metric variable to be plotted on the x-axis.
<code>y_variable</code>	The name of the QC metric variable to be plotted on the y-axis.
<code>qc_types</code>	A character vector specifying the QC types to plot.

facet_by_class	Logical; if TRUE, facets the plot by feature_class, as defined in the feature metadata.
y_shared	Logical; if TRUE, all facets share the same y-axis scale. If FALSE (default), each facet has its own y-axis scale.
filter_data	Logical; whether to use all data (default) or only QC-filtered data (filtered via filter_features_qc()).
include_qualifier	Logical; whether to include qualifier features (default is FALSE).
include_istd	Logical; whether to include internal standards (ISTDs) (default is FALSE). An ISTD normalized by itself has a CV of 0 for normalized intensities and concentrations.
equality_line	Logical; whether to show a line indicating identical values in both compared variables (default is FALSE).
threshold_values	Numeric single value or vector with 2 elements; threshold value to be shown as dashed lines from both axes on the plot (default is NA).
log_scale	Logical, whether to use a log10 scale the axes.
x_lim	Numeric vector of length 2 for x-axis limits. Use NA for auto-scaling (default is <code>c(0, NA)</code>).
y_lim	Numeric vector of length 2 for y-axis limits. Use NA for auto-scaling (default is <code>c(0, NA)</code>).
cols_page	Integer; number of facet columns per page (default is 5).
point_size	Numeric; size of points in millimeters (default is 1).
point_alpha	Numeric; transparency of points (default is 0.5).
point_color	A vector specifying the colors for points corresponding to different QC types. This can be either an unnamed vector or a named vector, with names corresponding to QC types. Unused colors will be ignored. Default is NA which corresponds to the default colors for QC types defined in the package.
point_fill	A vector specifying the fill colors for points corresponding to different QC types. This can be either an unnamed vector or a named vector, with names corresponding to QC types. Unused fill colors will be ignored. Default is NA which corresponds to the default fill colors for QC types defined in the package.
point_shape	A vector specifying the shapes for points corresponding to different QC types. This can be either an unnamed vector or a named vector, with names corresponding to QC types. Unused shapes will be ignored. Default is NA which corresponds to the default shapes for QC types defined in the package.
point_stroke	Numeric; thickness of point borders (default is 0.5).
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
autoscale	Logical. When TRUE (default), font_base_size and point_size left as NULL are sized automatically from cols_page (more facet columns per page give smaller text and points). Any value passed explicitly always takes precedence. When FALSE, unset sizes fall back to the single-plot defaults.

legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with mrmhub_set_plot_defaults() .
legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with mrmhub_set_plot_defaults() ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
strip_text_size	Optional facet strip text size, as a multiplier of font_base_size (when <= 3) or an absolute point size (when > 3). NULL (default) inherits from font_base_size.
strip_bg_color	Optional facet strip background fill colour. The strip text colour is set automatically for contrast (white on a dark fill, black on a light one). NULL (default) keeps the house dark-navy strips.
legend_bg_alpha	Optional opacity ([0, 1]) of a white legend background box, useful for a readable inside legend drawn over points. NULL (default) leaves the legend background unchanged.

Details

The comparison is visualized through one of three plot types:

- Scatter plot: Values of y_variable vs x_variable
- Difference plot: (y_variable - x_variable) vs mean of both values
- Ratio plot: log2(y_variable / x_variable) vs mean of both values
- x_variable and y_variable must be available in the QC metrics table. Please refer to the help page of [calc_qc_metrics\(\)](#) for more information on the available QC metric variables.
- When facet_by_class = TRUE, then the feature_class must be defined in the metadata or retrieved via specific functions, e.g., [parse_lipid_feature_names\(\)](#).

Value

A ggplot object representing the scatter plot.

See Also

[calc_qc_metrics\(\)](#), [filter_features_qc\(\)](#), [plot_normalization_qc\(\)](#), [normalize_by_istd\(\)](#)

Other QC plots: [plot_feature_correlations\(\)](#), [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_interference_impact\(\)](#), [plot_qc_summary_byclass\(\)](#), [plot_qc_summary_overall\(\)](#), [plot_rla_boxplot\(\)](#), [plot_rt_vs_chain\(\)](#), [plot_runscatter\(\)](#), [plot_runsequence\(\)](#)

plot_responsecurves	<i>Plot response curves</i>
---------------------	-----------------------------

Description

This function plots response curves for each feature. Multiple response curves, each with a linear regression line, can be plotted on the same graph. Each feature is displayed as a separate facet.

Usage

```
plot_responsecurves(  
  data = NULL,  
  variable = "intensity",  
  filter_data = FALSE,  
  include_qualifier = TRUE,  
  include_istd = TRUE,  
  include_feature_filter = NA,  
  exclude_feature_filter = NA,  
  max_regression_value = NA,  
  output_pdf = FALSE,  
  path = NA,  
  create_dir = TRUE,  
  return_plots = FALSE,  
  color_curves = NULL,  
  point_size = NULL,  
  line_width = 0.7,  
  label_wrap = FALSE,  
  label_wrap_width = 25,  
  font_base_size = NULL,  
  autoscale = TRUE,  
  legend_position = NULL,  
  legend_size = NULL,  
  show_legend_title = NULL,  
  title = NULL,  
  strip_text_size = NULL,  
  strip_bg_color = NULL,  
  rows_page = 4,  
  cols_page = 5,  
  curve_layout = "overlay",  
  fixed_scale_curves = FALSE,  
  r2_vstep = 0.06,  
  specific_page = NA,  
  page_orientation = "LANDSCAPE",  
  page_width = NULL,  
  page_height = NULL,  
  page_units = "mm",  
  show_progress = TRUE
```

)

Arguments

<code>data</code>	A MRMhubExperiment object.
<code>variable</code>	The variable to plot on the y-axis.
<code>filter_data</code>	Whether to use all data (default) or only QC-filtered data (filtered via <code>filter_features_qc()</code>).
<code>include_qualifier</code>	Logical, whether to include qualifier features. Default is TRUE.
<code>include_istd</code>	Logical, whether to include internal standard (ISTD) features. Default is TRUE.
<code>include_feature_filter</code>	Feature(s) to include by <code>feature_id</code> , as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
<code>exclude_feature_filter</code>	Feature(s) to exclude by <code>feature_id</code> , matched the same way as <code>include_feature_filter</code> . NA or "" ignores the filter.
<code>max_regression_value</code>	The maximum <code>sample_amount</code> (x) value for fitting the regression line. If NA, regression is based on all data points.
<code>output_pdf</code>	If TRUE, saves the generated plots as a PDF file. When FALSE, plots are directly plotted.
<code>path</code>	The file path for saving the PDF. Must be defined if <code>output_pdf</code> is TRUE.
<code>create_dir</code>	A logical value. If TRUE (the default) and <code>output_pdf</code> is TRUE, the parent directory of path is created if it does not yet exist.
<code>return_plots</code>	Logical. If TRUE, returns the plots as a list of ggplot objects.
<code>color_curves</code>	A vector of colors for the curves. If NULL (default), the colors for each curve are generated automatically. If colors are provided, the number of colors must match the number of curves.
<code>point_size</code>	Size of points in millimeters.
<code>line_width</code>	Width of regression lines.
<code>label_wrap</code>	Logical. If TRUE, long <code>feature_id</code> labels are wrapped to multiple lines using <code>label_wrap_width</code> . Default is FALSE.
<code>label_wrap_width</code>	Integer. Maximum width in characters for wrapped labels when <code>label_wrap</code> = TRUE. Default is 25. Ignored when <code>label_wrap</code> = FALSE.
<code>font_base_size</code>	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by <code>mrhub_set_plot_defaults()</code> if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
<code>autoscale</code>	Logical. When TRUE (default), <code>font_base_size</code> and <code>point_size</code> left as NULL are sized automatically from <code>cols_page</code> (more facet columns per page give smaller text and points). Any value passed explicitly always takes precedence. When FALSE, unset sizes fall back to the single-plot defaults.

legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> .
legend_size	Optional single multiplier of font_base_size (when ≤ 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
strip_text_size	Optional facet strip text size, as a multiplier of font_base_size (when ≤ 3) or an absolute point size (when > 3). NULL (default) inherits from font_base_size.
strip_bg_color	Optional facet strip background fill colour. The strip text colour is set automatically for contrast (white on a dark fill, black on a light one). NULL (default) keeps the house dark-navy strips.
rows_page	Number of rows of plots per page. Used for pagination in curve_layout = "overlay" and "cols". Ignored in curve_layout = "rows".
cols_page	Number of columns of plots per page. Used for pagination in curve_layout = "overlay" and "rows". Ignored in curve_layout = "cols".
curve_layout	Controls how multiple curves are displayed. One of: "overlay" (default) All curves overlaid in each feature panel using facet_wrap2. Pagination uses rows_page * cols_page. "cols" Grid layout with features as rows and curves as columns using facet_grid2. Pagination uses rows_page only. "rows" Grid layout with curves as rows and features as columns using facet_grid2. Pagination uses cols_page only.
fixed_scale_curves	Logical. If TRUE, fixes the y-axis scale per feature row (curve_layout = "cols") or per curve row (curve_layout = "rows"). If FALSE (default), each panel auto-scales. Silently ignored when curve_layout = "overlay".
r2_vstep	Numeric. Vertical step between stacked R ² labels when multiple curves are plotted in the same panel (curve_layout = "overlay"). Default is 0.06. Ignored when curve_layout is "cols" or "rows", where each panel has one curve.
specific_page	An integer specifying a specific page to plot. If NA (default), all pages are plotted.
page_orientation	Orientation of the PDF paper: "LANDSCAPE" or "PORTRAIT". Ignored when page_width and page_height are given.
page_width, page_height	Size of a PDF page, in page_units. Both must be given together. NULL (default) uses an A4 page of 280 x 200 mm, oriented by page_orientation. When an explicit size is given, page_orientation has no effect.

page_units	Unit of page_width and page_height: "mm" (default), "cm", "in" or "pt".
show_progress	Logical. If TRUE, displays a progress bar during plot creation.

Details

Features for plotting can be filtered using QC filters defined via `filter_features_qc()` or through `include_feature_filter` and `exclude_feature_filter` arguments. The resulting plots offer extensive customization options, including point size, line width, point color, point fill, point shape, line color, ribbon fill, and font base size.

Plots will be divided into multiple pages if the number of features exceeds the product of `rows_page` and `cols_page` settings. The function supports both direct plotting within R and saving plots as PDF files. Additionally, plots can be returned as a list of `ggplot2` objects for further manipulation or integration into other analyses.

Value

If `return_plots` is TRUE, a list of `ggplot` objects is returned. Otherwise, the function saves the plot output or does not return anything.

Preferred formats and devices

Purpose	Format	Device used	Typical dpi
Journal figure, vector (default choice)	"pdf"	<code>grDevices::cairo_pdf</code> , else <code>grDevices::pdf</code>	n/a
Figure for further editing (Illustrator, Inkscape)	"svg"	<code>svglite::svglite</code> , else <code>grDevices::svg</code>	n/a
Slides, Quarto HTML, GitHub	"png"	<code>ragg::agg_png</code> , else <code>grDevices::png</code>	150-300
Journal requiring raster submission	"tiff"	<code>ragg::agg_tiff</code> , else <code>grDevices::tiff</code>	300-600

Prefer a **vector** format (pdf, svg) for publication: text stays selectable and searchable, and lines stay sharp at any magnification.

Prefer a **raster** format (png, tiff) when a plot draws very many marks – a `plot_runscatter()` page covering several thousand analyses, or a dense `plot_pca()` score plot. Every point becomes a separate object in a PDF, so such figures produce very large files that are slow to open and to typeset. Saving them at 300-600 dpi instead keeps the file small with no visible loss.

The optional packages `ragg` and `svglite` are used automatically when installed, giving better text rendering, system-font support and smaller SVG files. When they are absent the equivalent `grDevices` device is used and the output is still correct. Installing both is recommended: `install.packages(c("ragg", "svglite"))`.

PDF output uses the `cairo` device wherever R was built with `cairo` support (`capabilities("cairo")`), because plain `grDevices::pdf()` writes text in a single-byte encoding and silently transliterates anything outside it – an en dash becomes `-`, `>=` replaces the proper symbol. Unit labels such as `umol/L` and statistical annotations routinely depend on those glyphs.

Multi-page output from the paged plot functions (`plot_runscatter()`, `plot_calibrationcurves()`, `plot_responsecurves()`, `plot_feature_correlations()`) is PDF only, which is the only format that holds many pages in one file. Use `save_plot()` for single figures in any of the other formats.

See Also

[save_plot\(\)](#) to save a single figure in any format.

Other calibration plots: [plot_calibrationcurves\(\)](#)

plot_rla_boxplot	<i>Relative log abundance (RLA) plot</i>
------------------	--

Description

The Relative Log Abundance (RLA) plot visualizes standardized feature abundances distributions across samples. RLA standardization involves subtracting either the within-batch or across-batch median from each feature's log-transformed abundance. These plots are effective for identifying systematic technical variations, such as batch effects, instrument drift, or sample handling inconsistencies, by providing a robust representation less susceptible to global intensity shifts.

The function also incorporates optional outlier detection and visualization functionalities to identify anomalous samples based on their median RLA values.

This function returns a list with the ggplot object representing the RLA plot and a table with detected outliers (if outlier_detection = TRUE).

Usage

```
plot_rla_boxplot(
  data = NULL,
  variable,
  rla_type_batch,
  qc_types = NA,
  plot_range = NA,
  rla_limit_to_range = FALSE,
  collapse_excluded = FALSE,
  remove_gaps = FALSE,
  gap_line_color = "#e34a33",
  gap_line_width = 0.3,
  gap_label_size = 2.5,
  gap_scale = 1,
  filter_data = FALSE,
  include_qualifier = TRUE,
  include_istd = TRUE,
  include_feature_filter = NA,
  exclude_feature_filter = NA,
  show_timestamp = FALSE,
  min_feature_intensity = 0,
  y_lim = NA,
  outlier_detection = TRUE,
  outlier_exclude = FALSE,
  outlier_method = "mad",
```

```

outlier_qctypes = c("SPL", "TQC", "BQC", "LTR", "NIST"),
outlier_k = NULL,
show_batches = TRUE,
batch_zebra_stripe = FALSE,
batch_line_color = "#b6f0c5",
batch_fill_color = "grey93",
x_gridlines = FALSE,
linewidth = 0.2,
font_base_size = NULL,
legend_position = NULL,
legend_size = NULL,
show_legend_title = NULL,
title = NULL,
strip_text_size = NULL,
strip_bg_color = NULL,
legend_bg_alpha = NULL,
relative_log_abundances = TRUE,
show_plot = TRUE
)

```

Arguments

<code>data</code>	A MRHubExperiment object.
<code>variable</code>	Variable to plot, must be one of "intensity", "norm_intensity", "conc", "area", "height", "fwhm", or one of "intensity_raw", "intensity_before", "norm_intensity_raw", "norm_intensity_before", "conc_raw", "conc_before"
<code>rla_type_batch</code>	Character, must be either "within" or "across", defining whether to use within-batch or across-batch RLA
<code>qc_types</code>	QC types to be plotted. Can be a vector of QC types or a regular expression pattern. NA (default) displays all available QC/Sample types.
<code>plot_range</code>	Numeric vector of length 2, specifying the start and end indices of the analysis order to be plotted. NA plots all samples.
<code>rla_limit_to_range</code>	Logical, whether to limit the RLA values to the specified <code>plot_range</code> . Default is FALSE, which means RLA values are calculated for all samples.
<code>collapse_excluded</code>	Logical, whether to collapse gaps in the x-axis caused by QC types that were not selected, re-indexing x to a contiguous sequence. Default is FALSE.
<code>remove_gaps</code>	Logical. If TRUE, contiguous indices replace the original <code>analysis_order</code> on the x-axis so that missing/filtered samples no longer leave large gaps. Each gap is highlighted with a thick vertical line and annotated with the flanking analysis-order IDs. Default is FALSE.
<code>gap_line_color</code>	Color of the vertical gap-indicator lines. Default is "#e34a33".
<code>gap_line_width</code>	Line width of the vertical gap-indicator lines. Default is 0.3.
<code>gap_label_size</code>	Font size of the gap-boundary labels. Default is 2.5.

gap_scale	Numeric multiplication factor for the gap band width. Default is 1. Increase (e.g. 2) for wider gaps, decrease for narrower.
filter_data	Logical, whether to use QC-filtered data based on criteria set via filter_features_qc().
include_qualifier	Logical, whether to include qualifier features. Default is TRUE.
include_istd	Logical, whether to include internal standard (ISTD) features. Default is TRUE.
include_feature_filter	Feature(s) to include by feature_id, as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
exclude_feature_filter	Feature(s) to exclude by feature_id, matched the same way as include_feature_filter. NA or "" ignores the filter.
show_timestamp	Logical, whether to use the acquisition timestamp as the x-axis instead of the run sequence number
min_feature_intensity	Numeric, exclude features with overall median intensity below this value
y_lim	Numeric vector of length 2, specifying the lower and upper y-axis limits. Default is NA, which uses limits calculated based on outlier_exclude.
outlier_detection	Logical, whether to show outlier fences on the plot and return a table with detect outliers based on the method defined by outlier_method.
outlier_exclude	Logical, whether to exclude outlier values from the plot. Default is FALSE, which means outliers are shown.
outlier_method	Character, method used for outlier detection. Default is "mad" (median absolute deviation). Other possible values are "iqr", "sd", "z_normal", "z_robust", "quantile", and "fold". See get_outlier_bounds() for details.
outlier_qctypes	Character vector, QC types to use for outlier detection. Default is c("SPL", "TQC", "BQC").
outlier_k	Numeric, multiplier for the outlier detection method. Default is NULL, which uses the default value for the selected method. See get_outlier_bounds() for details. When using the "fold" method, either single numeric value or a vector with two values (lower and upper fences) can be supplied.
show_batches	Logical, whether to show batch separators in the plot
batch_zebra_stripe	Logical, whether to show batches as shaded areas instead of line separators
batch_line_color	Character, color of the batch separator lines
batch_fill_color	Character, color of the batch shaded areas
x_gridlines	Logical, whether to show major x-axis gridlines

linewidth	Numeric, line width used for whiskers of the boxplot
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by <code>mrmhub_set_plot_defaults()</code> if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> .
legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with <code>mrmhub_set_plot_defaults()</code> ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
strip_text_size	Optional facet strip text size, as a multiplier of font_base_size (when <= 3) or an absolute point size (when > 3). NULL (default) inherits from font_base_size.
strip_bg_color	Optional facet strip background fill colour. The strip text colour is set automatically for contrast (white on a dark fill, black on a light one). NULL (default) keeps the house dark-navy strips.
legend_bg_alpha	Optional opacity ([0, 1]) of a white legend background box, useful for a readable inside legend drawn over points. NULL (default) leaves the legend background unchanged.
relative_log_abundances	Logical, whether to use relative log abundances (RLA) or just log-transformed values
show_plot	Logical, whether to display the plot. Default is TRUE.

Value

A list with the ggplot object representing the RLA plot and a table with detected outliers if outlier_detection = TRUE.

References

De Livera et al. (2012) Normalizing and integrating metabolomics data. Analytical Chemistry 10768-10776 DOI: [10.1021/ac302748b](https://doi.org/10.1021/ac302748b) De Livera et al. (2015) Statistical Methods for Handling Unwanted Variation in Metabolomics Data. Analytical Chemistry 87(7):3606-3615 DOI: [10.1021/ac502439y](https://doi.org/10.1021/ac502439y)

See Also

Other QC plots: `plot_feature_correlations()`, `plot_interference_correction()`, `plot_matrixeffects()`, `plot_normalization_qc()`, `plot_pca()`, `plot_pca_loading()`, `plot_qc_interference_impact()`, `plot_qc_summary_byclass()`, `plot_qc_summary_overall()`, `plot_qcmetrics_comparison()`, `plot_rt_vs_chain()`, `plot_runscatter()`, `plot_runsequence()`

plot_rt_vs_chain

*Plot retention time versus chain length and saturation***Description**

Generates scatter plots of retention time (RT) versus either chain length, degree of saturation (double bonds), or equivalent carbon number (ECN) of lipid features of different feature classes. This visualization can be useful in identifying annotation (peak picking) errors in reversed-phase (RP)-LC lipidomics dataset arising from isotopic, isobaric, isomeric, or unknown interferences.

Usage

```
plot_rt_vs_chain(
  data = NULL,
  x_var = c("total_c", "total_db", "ecn"),
  qc_types = NA,
  outliers_highlight = TRUE,
  outlier_residual_min = 0.15,
  outlier_print = TRUE,
  ecn_k = 1.5,
  include_qualifier = FALSE,
  robust_regression = TRUE,
  cols_page = 5,
  point_size = NULL,
  point_alpha = 0.9,
  line_transparency = 0.5,
  font_base_size = NULL,
  autoscale = TRUE,
  legend_position = NULL,
  legend_size = NULL,
  show_legend_title = NULL,
  title = NULL,
  strip_text_size = NULL,
  strip_bg_color = NULL
)
```

Arguments

<code>data</code>	A MRMhubExperiment object.
<code>x_var</code>	Variable to use for the x-axis. One of "total_c", "total_db", or "ecn".

qc_types	A character vector specifying the QC types to plot. It must contain at least one element. The default NA plots any of the non-blank QC types ("SPL", "TQC", "BQC", "HQC", "MQC", "LQC", "NIST", "LTR") present in the dataset. A single value that is a QC type is matched exactly; any other single value is a regular expression, e.g. "QC\$" for all QC types ending in "QC".
outliers_highlight	Whether to highlight potential outliers in the plot. Default is TRUE.
outlier_residual_min	Minimum value for the residuals to be considered an outlier (default is 0.15). The value corresponds to the RT difference between the fitted line and the median RT of the feature. The value is used to flag outliers.
outlier_print	Whether to print the features that are flagged as potential outliers to the console. Default is TRUE.
ecn_k	Constant for ECN calculation ($ECN = C - ecn_k * DB$), see Details. Default is 1.5.
include_qualifier	Whether to include qualifier features.
robust_regression	Whether to use robust regression, which is less sensitive to outlier (default is TRUE).
cols_page	Number of facet columns, representing different feature classes, shown per page (default is 5).
point_size	Size of the data points. Default is 2
point_alpha	Alpha transparency of the data point. Default is 0.9
line_transparency	Alpha transparency of the regression lines. Default is 0.9
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
autoscale	Logical. When TRUE (default), font_base_size and point_size left as NULL are sized automatically from cols_page (more facet columns per page give smaller text and points). Any value passed explicitly always takes precedence. When FALSE, unset sizes fall back to the single-plot defaults.
legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with mrmhub_set_plot_defaults() .
legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with mrmhub_set_plot_defaults() ; FALSE hides it, TRUE forces it shown.

title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
strip_text_size	Optional facet strip text size, as a multiplier of font_base_size (when ≤ 3) or an absolute point size (when > 3). NULL (default) inherits from font_base_size.
strip_bg_color	Optional facet strip background fill colour. The strip text colour is set automatically for contrast (white on a dark fill, black on a light one). NULL (default) keeps the house dark-navy strips.

Details

The retention time can be either plotted against the total number of carbon atoms with the total number of double bonds as curves, or opposite, with the total double bond number as x axis and the total number of carbon atoms as curves. Alternatively, the retention time can be plotted against the ECN, which is calculated as $ECN = C_{total} - ecn_k \times DB_{total}$, where ecn_k is a constant that may need to be adjusted to the specific chromatographic properties. The default value is $ecn_k = 1.5$.

Value

A ggplot object representing faceted scatter plots

See Also

Other QC plots: [plot_feature_correlations\(\)](#), [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_interference_impact\(\)](#), [plot_qc_summary_byclass\(\)](#), [plot_qc_summary_overall\(\)](#), [plot_qcmetrics_comparison\(\)](#), [plot_rla_boxplot\(\)](#), [plot_runscatter\(\)](#), [plot_runsequence\(\)](#)

plot_runscatter	<i>RunScatter plot</i>
-----------------	------------------------

Description

The runscatter function visualizes raw or processed feature signals across different sample/QC types along the analysis sequence. It helps identify trends, detect outliers, and assess analytical performance. Available feature variables, such as retention time (RT) and full width at half maximum (FWHM), can be plotted against analysis order or timestamps.

By default, all QC types present in the dataset will be plotted. User-defined QC types that have no predefined color or shape in mrmhub are assigned black shapes. To show specific QC types, use the qc_types argument.

To plot the feature values before the last applied drift/batch correction, add *_before to the variable name, e.g., intensity_before or conc_before. To plot the uncorrected feature values (before any drift/batch correction), add *_raw to the variable name, e.g., intensity_raw or conc_raw. To show corresponding fit curves, set show_trend = TRUE.

The function also supports visualizing analysis batches, reference lines (mean $\pm$ SD), and trends. It offers customization options to display batch separators, apply outlier capping, show smoothed

trend curves, add reference lines, and incorporate other features. Outlier capping is particularly useful to focus on QC or study sample trends that might otherwise be obscured by extreme values or high variability.

The runcscatter function serves as a central QC tool in the workflow, providing critical insights into data quality.

Usage

```
plot_runcscatter(  
  data = NULL,  
  variable,  
  qc_types = NA,  
  filter_data = FALSE,  
  include_qualifier = TRUE,  
  include_istd = TRUE,  
  include_feature_filter = NA,  
  exclude_feature_filter = NA,  
  plot_range = NA,  
  output_pdf = FALSE,  
  path = NA,  
  create_dir = TRUE,  
  multithreading = FALSE,  
  return_plots = FALSE,  
  show_batches = TRUE,  
  batch_zebra_stripe = FALSE,  
  batch_line_color = "#b6f0c5",  
  batch_fill_color = "grey93",  
  cap_outliers = FALSE,  
  cap_sample_k_mad = 4,  
  cap_qc_k_mad = 4,  
  cap_top_n_outliers = NA,  
  show_reference_lines = FALSE,  
  ref_qc_types = NA,  
  reference_k_sd = 2,  
  reference_batchwise = FALSE,  
  reference_line_color = "#04bf9a",  
  reference_sd_shade = FALSE,  
  reference_fill_color = NA,  
  reference_linewidth = 0.75,  
  show_trend = FALSE,  
  trend_color = "#22e06b",  
  y_lim = c(0, NA),  
  log_scale = FALSE,  
  show_gridlines = FALSE,  
  point_size = NULL,  
  point_alpha = 1,  
  point_border_width = NA,  
  font_base_size = NULL,
```

```

    autoscale = TRUE,
    legend_position = NULL,
    legend_size = NULL,
    show_legend_title = NULL,
    title = NULL,
    strip_text_size = NULL,
    strip_bg_color = NULL,
    legend_bg_alpha = NULL,
    rows_page = 3,
    cols_page = 3,
    specific_page = NA,
    page_orientation = "LANDSCAPE",
    page_width = NULL,
    page_height = NULL,
    page_units = "mm",
    y_label_text = NA,
    pages_per_core = 1,
    use_dingbats = TRUE,
    show_progress = TRUE,
    remove_gaps = FALSE,
    collapse_excluded = FALSE,
    gap_line_color = "#e34a33",
    gap_line_width = 0.3,
    gap_label_size = 2.5,
    gap_scale = 1,
    label_wrap = FALSE,
    label_wrap_width = 25
  )

```

Arguments

<code>data</code>	A MRMhubExperiment object.
<code>variable</code>	The variable to plot on the y-axis, one of 'intensity', 'norm_intensity', 'conc', 'rt', 'fwhm', 'area', 'height', 'response'. Add <code>_before</code> after the variable name to plot the feature values before the last applied drift/batch correction, (e.g., <code>conc_before</code>). Add <code>_raw</code> after the variable name to plot the raw uncorrected feature values (e.g., <code>conc_raw</code>).
<code>qc_types</code>	QC types to be plotted. Can be a vector of QC types or a regular expression pattern. NA (default) displays all available QC/Sample types.
<code>filter_data</code>	Logical, whether to use QC-filtered data based on criteria set via <code>filter_features_qc()</code> .
<code>include_qualifier</code>	Logical, whether to include qualifier features. Default is TRUE.
<code>include_istd</code>	Logical, whether to include internal standard (ISTD) features. Default is TRUE.
<code>include_feature_filter</code>	Feature(s) to include by <code>feature_id</code> , as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.

exclude_feature_filter	Feature(s) to exclude by feature_id, matched the same way as include_feature_filter. NA or "" ignores the filter.
plot_range	Numeric vector of length 2, specifying the start and end indices of the analysis order to be plotted. NA plots all samples.
output_pdf	Logical, whether to save the plot as a PDF file.
path	File name for the PDF output.
create_dir	A logical value. If TRUE (the default) and output_pdf is TRUE, the parent directory of path is created if it does not yet exist.
multithreading	Logical, whether to use parallel processing to speed up plot generation.
return_plots	Logical, whether to return the list of ggplot objects.
show_batches	Logical, whether to show batch separators in the plot.
batch_zebra_stripe	Logical, whether to display batches with alternating shaded and non-shaded areas.
batch_line_color	Color of the batch separator lines.
batch_fill_color	Color for the shaded areas representing batches.
cap_outliers	Logical, whether to cap upper outliers based on MAD fences of SPL and QC samples.
cap_sample_k_mad	Numeric, $k * \text{MAD}$ (median absolute deviation) for outlier capping of SPL samples.
cap_qc_k_mad	Numeric, $k * \text{MAD}$ (median absolute deviation) for outlier capping of QC samples.
cap_top_n_outliers	Numeric, cap the top n outliers regardless of MAD fences. NA or 0 ignores this filter.
show_reference_lines	Whether to display reference lines ($\text{mean} \pm n \times \text{SD}$).
ref_qc_types	QC type for which the reference lines are calculated.
reference_k_sd	Multiplier for standard deviations to define SD reference lines.
reference_batchwise	Whether to calculate reference lines per batch.
reference_line_color	Color of the reference lines.
reference_sd_shade	TRUE plots a colored band indicating the $\pm n \times \text{SD}$ reference range. FALSE (default) shows reference lines instead.
reference_fill_color	Fill color of the batch-wise reference ranges. If NA (default), the color assigned to the qc_type is used.

reference_linewidth	Width of the reference lines.
show_trend	If TRUE trend curves before or after drift/batch correction are shown.
trend_color	Color of the trend curve.
y_lim	Numeric vector of length 2, specifying the lower and upper y-axis limits. Default is <code>c(0, NA)</code> , which sets the lower limit to 0 and the upper limit automatically.
log_scale	Logical, whether to use a log10 scale for the y-axis.
show_gridlines	Whether to show major x and y gridlines.
point_size	Size of the data points. Default is 1.5.
point_alpha	Alpha transparency of the data points.
point_border_width	Width of the data point borders.
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by <code>mrhub_set_plot_defaults()</code> if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
autoscale	Logical. When TRUE (default), <code>font_base_size</code> and <code>point_size</code> left as NULL are sized automatically from <code>cols_page</code> (more facet columns per page give smaller text and points). Any value passed explicitly always takes precedence. When FALSE, unset sizes fall back to the single-plot defaults.
legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric <code>c(x, y)</code> in $[0, 1]$ coordinates. NULL (default) keeps the current placement, unless a global default is set with <code>mrhub_set_plot_defaults()</code> .
legend_size	Optional single multiplier of <code>font_base_size</code> (when ≤ 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with <code>mrhub_set_plot_defaults()</code> ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.
strip_text_size	Optional facet strip text size, as a multiplier of <code>font_base_size</code> (when ≤ 3) or an absolute point size (when > 3). NULL (default) inherits from <code>font_base_size</code> .
strip_bg_color	Optional facet strip background fill colour. The strip text colour is set automatically for contrast (white on a dark fill, black on a light one). NULL (default) keeps the house dark-navy strips.
legend_bg_alpha	Optional opacity ($[0, 1]$) of a white legend background box, useful for a readable inside legend drawn over points. NULL (default) leaves the legend background unchanged.
rows_page	Number of rows per page.

cols_page	Number of columns per page.
specific_page	Show/save a specific page number only. NA plots/saves all pages.
page_orientation	Page orientation, "LANDSCAPE" or "PORTRAIT". Ignored when page_width and page_height are given.
page_width, page_height	Size of a PDF page, in page_units. Both must be given together. NULL (default) uses an A4 page of 280 x 200 mm, oriented by page_orientation. When an explicit size is given, page_orientation has no effect.
page_units	Unit of page_width and page_height: "mm" (default), "cm", "in" or "pt".
y_label_text	Override the default y-axis label text.
pages_per_core	Number of pages to be plotted by core when multithreading is enabled. Default is 6. Changing this number may improve performance.
use_dingbats	Logical, whether to use Dingbats font in the PDF output for improved plotting speed. Default is TRUE. Set to FALSE if your PDF viewer does not show points correctly.
show_progress	Logical, whether to show a progress bar. Default is TRUE.
remove_gaps	Logical. If TRUE, contiguous indices replace the original analysis_order on the x-axis so that missing/filtered samples no longer leave large gaps. Each gap is highlighted with a thick vertical line and annotated with the flanking analysis-order IDs. Default is FALSE.
collapse_excluded	Logical. If TRUE, gaps in the x-axis caused by QC types that were not selected (via qc_types) are collapsed, re-indexing x to a contiguous sequence. Default is FALSE.
gap_line_color	Color of the vertical gap-indicator lines. Default is "#e34a33".
gap_line_width	Line width of the vertical gap-indicator lines. Default is 0.3.
gap_label_size	Font size of the gap-boundary labels. Default is 2.5.
gap_scale	Numeric multiplication factor for the gap band width. Default is 1. Increase (e.g. 2) for wider gaps, decrease for narrower.
label_wrap	Logical. If TRUE, long feature_id labels are wrapped to multiple lines using label_wrap_width. Default is FALSE.
label_wrap_width	Integer. Maximum width in characters for wrapped labels when label_wrap = TRUE. Default is 25. Ignored when label_wrap = FALSE.

Details

- The outlier capping feature (cap_outliers) allows you to cap upper outliers based on median absolute deviation (MAD) fences of SPL and QC samples, or to remove the top n points. This can help to focus on the trends of interest when there are outlier or a high variability in the data, e.g. in the study samples.
- When using log-scale (log_scale = TRUE), zero or negative values will be replaced with the minimum positive value divided by 5 to avoid log 0 errors

- Reference lines/ranges corresponding to mean $\pm k \times \text{SD}$ can be shown across or within batches as lines or shaded stripes.
- Trend curves can be displayed before or after drift/batch correction. In either case, a drift and/or batch correction must be applied to the data to enable plotting of trend curves. To show trend curves used for the last drift or batch correction, add "_before" to the variable name, e.g. conc_before or intensity_before and set show_trend = TRUE.

Value

A list of ggplot objects if return_plots = TRUE, otherwise NULL (the plots are drawn to the active device or written to a PDF).

Preferred formats and devices

Purpose	Format	Device used	Typical dpi
Journal figure, vector (default choice)	"pdf"	grDevices::cairo_pdf, else grDevices::pdf	n/a
Figure for further editing (Illustrator, Inkscape)	"svg"	svglite::svglite, else grDevices::svg	n/a
Slides, Quarto HTML, GitHub	"png"	ragg::agg_png, else grDevices::png	150-300
Journal requiring raster submission	"tiff"	ragg::agg_tiff, else grDevices::tiff	300-600

Prefer a **vector** format (pdf, svg) for publication: text stays selectable and searchable, and lines stay sharp at any magnification.

Prefer a **raster** format (png, tiff) when a plot draws very many marks – a [plot_runscatter\(\)](#) page covering several thousand analyses, or a dense [plot_pca\(\)](#) score plot. Every point becomes a separate object in a PDF, so such figures produce very large files that are slow to open and to typeset. Saving them at 300-600 dpi instead keeps the file small with no visible loss.

The optional packages ragg and svglite are used automatically when installed, giving better text rendering, system-font support and smaller SVG files. When they are absent the equivalent grDevices device is used and the output is still correct. Installing both is recommended: `install.packages(c("ragg", "svglite"))`.

PDF output uses the cairo device wherever R was built with cairo support (`capabilities("cairo")`), because plain `grDevices::pdf()` writes text in a single-byte encoding and silently transliterates anything outside it – an en dash becomes -, >= replaces the proper symbol. Unit labels such as $\mu\text{mol/L}$ and statistical annotations routinely depend on those glyphs.

Multi-page output from the paged plot functions ([plot_runscatter\(\)](#), [plot_calibrationcurves\(\)](#), [plot_responsecurves\(\)](#), [plot_feature_correlations\(\)](#)) is PDF only, which is the only format that holds many pages in one file. Use [save_plot\(\)](#) for single figures in any of the other formats.

See Also

[save_plot\(\)](#) to save a single figure in any format.

Other QC plots: [plot_feature_correlations\(\)](#), [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_interference_impact\(\)](#), [plot_qc_summary_byclass\(\)](#), [plot_qc_summary_overall\(\)](#), [plot_qcmetrics_comparison\(\)](#), [plot_qla_boxplot\(\)](#), [plot_rt_vs_chain\(\)](#), [plot_runsequence\(\)](#)

plot_runsequence	<i>RunSequence plot</i>
------------------	-------------------------

Description

The RunSequence plot provides an overview of the analysis design and timelines, which can be useful for subsequent processing steps. The plot illustrates the batch structure, the quality control (QC) samples included with their respective positions, and additional information regarding the date, duration, and run time of the analysis.

Setting `show_timestamp = TRUE` allows you to check for any interruptions in the analysis timeline.

Usage

```
plot_runsequence(
  data = NULL,
  qc_types = NA,
  show_batches = TRUE,
  show_timestamp = FALSE,
  add_info_title = TRUE,
  single_row = FALSE,
  segment_linewidth = 0.5,
  batch_zebra_stripe = FALSE,
  batch_line_color = "#b6f0c5",
  batch_fill_color = "grey93",
  font_base_size = NULL,
  legend_position = NULL,
  legend_size = NULL,
  show_legend_title = NULL,
  title = NULL
)
```

Arguments

<code>data</code>	A MRHubExperiment object.
<code>qc_types</code>	QC types to be plotted. Can be a vector of QC types or a regular expression pattern. NA (default) displays all available QC/Sample types.
<code>show_batches</code>	Logical, whether to show batch separators in the plot.
<code>show_timestamp</code>	Logical, whether to use the acquisition timestamp as the x-axis instead of the run sequence number.
<code>add_info_title</code>	Logical, whether to add a title with the experiment title, analysis date, and analysis times.
<code>single_row</code>	Logical, whether to show all QC types in a single row.
<code>segment_linewidth</code>	Width of the segment lines, default is 0.5.

batch_zebra_stripe	Logical, whether to show batches as shaded areas instead of line separators.
batch_line_color	Color of the batch separator lines.
batch_fill_color	Color of the batch shaded areas.
font_base_size	Numeric. Base font size (in points) for plot text; all plot text scales proportionally with this value. NULL (default) uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise an automatic size (derived from the facet-column count on paged plots, or the per-plot default shown in the Usage section above).
legend_position	Optional legend placement. One of "right", "left", "top", "bottom", "none"; a corner keyword "inside-tr", "inside-tl", "inside-br", "inside-bl"; or a numeric c(x, y) in [0, 1] coordinates. NULL (default) keeps the current placement, unless a global default is set with mrmhub_set_plot_defaults() .
legend_size	Optional single multiplier of font_base_size (when <= 3) or absolute point size (when > 3) that scales the whole legend: text, title, key and the plotted symbols. NULL (default) leaves the legend unchanged.
show_legend_title	Logical. NULL (default) keeps the legend title, unless a global default is set with mrmhub_set_plot_defaults() ; FALSE hides it, TRUE forces it shown.
title	Optional plot title. NULL (default) or NA shows no title; a character string is shown as the title.

Value

A ggplot object representing the run sequence plot.

See Also

Other QC plots: [plot_feature_correlations\(\)](#), [plot_interference_correction\(\)](#), [plot_matrixeffects\(\)](#), [plot_normalization_qc\(\)](#), [plot_pca\(\)](#), [plot_pca_loading\(\)](#), [plot_qc_interference_impact\(\)](#), [plot_qc_summary_byclass\(\)](#), [plot_qc_summary_overall\(\)](#), [plot_qcmetrics_comparison\(\)](#), [plot_ria_boxplot\(\)](#), [plot_rt_vs_chain\(\)](#), [plot_runscatter\(\)](#)

pooled_rsd

Pooled percent relative standard deviation (%RSD) across groups

Description

[pooled_sd\(\)](#) expressed as a percentage of the grand mean of x, i.e. a batch-pooled %CV. The pooled SD weights each group by its degrees of freedom and the grand mean weights each value equally, so both are consistently weighted by group size. See [pooled_sd\(\)](#) for how groups are pooled.

Usage

```
pooled_rsd(x, group, na.rm = FALSE, min_n = 2L)
```

Arguments

<code>x</code>	a numeric vector
<code>group</code>	a vector the same length as <code>x</code> giving the pooling group of each value (e.g. <code>batch_id</code>)
<code>na.rm</code>	logical, if TRUE then NA values in <code>x</code> are stripped before pooling
<code>min_n</code>	integer, the minimum number of non-missing values a group must have to contribute. Groups with fewer than <code>min_n</code> are dropped; a group is always dropped below two values, which have no within-group variance. The default is 2.

Value

a single numeric value. `NA_real_` if the grand mean is zero or NA, `x` is not numeric, or no group has enough replicates.

Examples

```
pooled_rsd(c(5, 6, 4, 15, 16, 14), group = c(1, 1, 1, 2, 2, 2))
```

<code>pooled_sd</code>	<i>Pooled standard deviation across groups</i>
------------------------	--

Description

Pools the within-group variances of `x` across the levels of `group`, weighting each group by its degrees of freedom ($n_i - 1$), and returns the square root. This is the within-group precision estimate: unlike a plain `sd(x)` it is not inflated by systematic shifts *between* groups (e.g. batch offsets or drift), so it is the appropriate spread for a batch-pooled %RSD.

Usage

```
pooled_sd(x, group, na.rm = FALSE, min_n = 2L)
```

Arguments

<code>x</code>	a numeric vector
<code>group</code>	a vector the same length as <code>x</code> giving the pooling group of each value (e.g. <code>batch_id</code>)
<code>na.rm</code>	logical, if TRUE then NA values in <code>x</code> are stripped before pooling
<code>min_n</code>	integer, the minimum number of non-missing values a group must have to contribute. Groups with fewer than <code>min_n</code> are dropped; a group is always dropped below two values, which have no within-group variance. The default is 2.

Value

a single numeric value. NA_real_ if x is not numeric or no group has enough replicates to pool.

Examples

```
pooled_sd(c(5, 6, 4, 15, 16, 14), group = c(1, 1, 1, 2, 2, 2))
```

quant_lcms_dataset	<i>LC-MS dataset with external calibration curve and metadata</i>
--------------------	---

Description

This demo dataset is included for use in function examples and user testing. It is a subset of an LC-MS analysis of plasma steroids, containing an external calibration curve for each analyte, QC samples with known concentrations, and unknown samples.

Usage

```
quant_lcms_dataset
```

Format

A MRMhubExperiment object with the following data and metadata:

dataset_orig Original data (peak data).

dataset Annotated data

annot_analyses Analysis-level metadata

annot_features Feature-level annotations

annot_istds ISTD concentrations

annot_qcconcentrations Calibrant (CAL) and QC concentrations

quantify_by_calibration	<i>Calculate concentrations based on external calibration</i>
-------------------------	---

Description

Concentrations of all features in all analyses are determined using ISTD-normalized intensities and corresponding external calibration curves. Calibration curves are calculated for each feature based on calibration sample concentrations defined in the qc_concentrations metadata. The regression fit model (linear or quadratic) and the weighting method (either "none", "1/x", "1/x^2", or "1/sqrt(x)") can be defined globally via the arguments fit_model and fit_weighting for all features, if fit_overwrite is TRUE. Alternatively, the model and weighting can be defined individually for each feature in the feature metadata (columns curve_fit_model and curve_fit_weighting). If these details are missing in the metadata, the default values provided via fit_model and fit_weighting will be used.

Usage

```
quantify_by_calibration(
  data = NULL,
  include_qualifier = TRUE,
  fit_overwrite,
  fit_model = c("linear", "quadratic"),
  fit_weighting = c("none", "1/x", "1/x^2", "1/sqrt(x)"),
  ignore_failed_calibration = FALSE,
  ignore_missing_annotation = FALSE,
  lod_sigma = c("residual", "intercept")
)
```

Arguments

data	A MRMhubExperiment object
include_qualifier	A logical value. If TRUE, the function will include qualifier features in the calibration curve calculations.
fit_overwrite	If TRUE, the function will use the provided <code>fit_model</code> and <code>fit_weighting</code> values for all analytes and ignore any fit method and weighting settings defined in the metadata.
fit_model	A character string specifying the default regression fit method to use for the calibration curve. Must be one of "linear" or "quadratic". This method will be applied if no specific fit method is defined for a feature in the metadata, or when <code>fit_overwrite = TRUE</code> .
fit_weighting	A character string specifying the default weighting method for the regression points in the calibration curve. Must be one of "none", "1/x", "1/x^2", or "1/sqrt(x)". This method will be applied if no specific weighting method is defined for a feature in the metadata, or when <code>fit_overwrite = TRUE</code> .
ignore_failed_calibration	If FALSE, raises error if calibration curve fit fails for any feature. If TRUE, failed fits will be ignored, and resulting feature concentration will be NA.
ignore_missing_annotation	If FALSE, raises error if any of the following information is missing: calibration curve data, ISTD mix volume and sample amounts for any feature. If TRUE, missing annotations will be ignored, and resulting feature concentration will be NA
lod_sigma	A character string selecting the standard deviation of the response (sigma) used in the ICH Q2 LoD/LoQ formulas. Must be one of "residual" (the residual standard error of the regression, Sy/x ; the default) or "intercept" (the standard error of the intercept). See calc_calibration_results() for details.

Details

The concentrations are added to the dataset table as `feature_conc` column. The results of the regression and the calculated LoD and LoQ values are stored in the `metrics_calibration` table of the returned `MRMhubExperiment` object.

Value

A modified [MRMhubExperiment](#) object with updated concentration values.

See Also

[calc_calibration_results\(\)](#) for calculating the calibration curve results including LoD and LoQ.

[quantify_by_istd\(\)](#) for calculation of concentrations based on spiked-in internal standard concentration.

quantify_by_istd	<i>Calculate analyte concentrations using internal standards</i>
------------------	--

Description

This function calculates analyte concentrations based on internal standard (ISTD) normalized intensities, the corresponding spiked-in ISTD amount, and the per-feature response factor (`response_factor`, from the feature metadata), normalized by the sample amount.

Usage

```
quantify_by_istd(  
  data = NULL,  
  concentration_unit = "molar",  
  ignore_missing_annotation = FALSE,  
  ignore_istds = FALSE  
)
```

Arguments

<code>data</code>	A MRMhubExperiment object
<code>concentration_unit</code>	Character string indicating the type of concentration to calculate and export. Must be either "molar" for molar concentrations (e.g., $\mu\text{mol/L}$) or "mass" for mass concentrations (e.g., $\mu\text{g/L}$).
<code>ignore_missing_annotation</code>	If FALSE, an error will be raised if any of the following information is missing: ISTD concentration, ISTD mix volume, and sample amounts for any feature. If TRUE, missing annotations will be ignored, and resulting feature concentration will be NA
<code>ignore_istds</code>	If TRUE, ISTD features will be ignored in the concentration calculation and the resulting concentration will be NA. Default is FALSE.

Details

The ISTD used for quantitation is taken from the `quant_istd_feature_id` column, which is currently always derived from the feature's normalization ISTD (`istd_feature_id`).

By default, concentrations are returned in molar units (e.g., $\mu\text{mol/L}$). To return concentrations in mass units (e.g., $\mu\text{g/L}$), set `concentration_unit = "mass"`. This requires either the chemical formula or molecular weight for each feature to be specified in the feature metadata.

Internal standard concentrations can also be provided in ng/mL . In such cases, the function will convert these concentrations to molar units for internal calculations. Here too, either the chemical formula or the molecular weight must be defined for each ISTD in the feature metadata.

The unit of the calculated concentrations is determined by the `concentration_unit` argument and the `sample_amount_unit` field in the analysis metadata of the `MRMhubExperiment` object. The function will automatically adjust the sample amount to the appropriate unit based on the provided `concentration_unit`. For example, if `concentration_unit = "molar"` and `sample_amount_unit = "uL"`, the calculated concentrations will be in $\mu\text{mol/L}$. If `concentration_unit = "mass"`, the concentrations will be in $\mu\text{g/L}$.

The calculated concentrations are added to the dataset table as a new column named `feature_conc`, along with the intermediate total amount per analysis in `feature_pmol_total`.

Value

A `MRMhubExperiment` object with the calculated analyte concentrations added to the dataset table in the `feature_conc` column (and the intermediate `feature_pmol_total`).

See Also

[`quantify\_by\_calibration\(\)`](#) for calculating concentrations based on external calibration curves.

<code>read_dataset_rds</code>	<i>Read a complete MRMhubExperiment from an RDS file</i>
-------------------------------	--

Description

Reads an `.rds` file written by [`save\_dataset\_rds\(\)`](#) (or a bare `saveRDS()`) back into an `MRMhubExperiment`, with feedback and a class check. The `.rds` is a self-contained snapshot of a complete analysis project, so the reloaded object can be inspected, re-plotted or re-processed with **mrhub** without the original input files or metadata.

If the file carries an embedded content fingerprint (see [`save\_dataset\_rds\(\)`](#)), it is verified against the reloaded object: a match confirms the file holds the object it was saved from, and the fingerprint is printed so it can be compared with a recorded value.

This function only adds the class check, fingerprint verification and console feedback on top of [`readRDS\(\)`](#); the file itself is a plain R serialization. It can therefore always be reopened with base R's `readRDS()` without **mrhub**, and every slot (the dataset, the `annot_*` metadata tables, the QC and calibration metrics) is stored as an ordinary tibble that can be inspected via the `@` slots or [`attributes\(\)`](#). **mrhub** is only required to re-plot or re-process the object — which makes the `.rds` a convenient self-contained archive of a complete analysis.

Usage

```
read_dataset_rds(path, verify = TRUE, show_status = FALSE)
```

Arguments

path	A character string with the path to the .rds file.
verify	A logical value. If TRUE (default) and the file carries an embedded fingerprint, it is recomputed and compared; a mismatch triggers a warning (the object is still returned).
show_status	A logical value. If TRUE, the full processing and metadata report (mrmhub_status()) is printed after loading. Default is FALSE, which prints only the compact one-line overview.

Value

The loaded [MRMhubExperiment](#) object.

See Also

[save_dataset_rds\(\)](#), [mrmhub_status\(\)](#)

Examples

```
mexp <- data_load_example()
path <- file.path(tempdir(), "example_mexp.rds")
save_dataset_rds(mexp, path)
mexp2 <- read_dataset_rds(path)
```

save_dataset_csv	<i>Export data to a CSV file</i>
------------------	----------------------------------

Description

This function exports specific unprocessed or processed feature variable (e.g. intensities or concentrations) from a [MRMhubExperiment](#) object to a CSV file. Allows selection of features and optional QC filtering.

Usage

```
save_dataset_csv(
  data = NULL,
  path,
  variable,
  qc_types = NA,
  filter_data = FALSE,
  include_qualifier = NA,
```

```

include_istd = NA,
include_feature_filter = NA,
exclude_feature_filter = NA,
add_qctype = NA,
create_dir = TRUE
)

```

Arguments

data	MRMhubExperiment object
path	File name with path of exported CSV file
variable	Variable to be exported, must be present in the data and any of "area", "height", "intensity", "norm_intensity", "response", "conc", "conc_raw", "rt", "fwhm".
qc_types	QC types to be exported. Can be a vector of QC types or a regular expression pattern. NA (default) exports all available QC/Sample types.
filter_data	A logical value indicating whether to use all data (default) or only QC-filtered data (filtered via filter_features_qc()). Default is FALSE.
include_qualifier	A logical value indicating whether to include qualifier features. Default is NA, which will be automatically set to FALSE if variable is conc or conc_raw, and TRUE otherwise.
include_istd	A logical value indicating whether to include internal standard (ISTD) features. Default is NA, which will be automatically set to FALSE if variable is norm_intensity, conc or conc_raw, and TRUE otherwise.
include_feature_filter	Feature(s) to include by feature_id, as a character vector. Each element is matched exactly when it names an existing feature, otherwise treated as a regex; elements combine with OR. A full ID (e.g. "S1P d18:0 [M>60]") needs no escaping, while patterns like "PC PE" still work. NA or "" ignores the filter.
exclude_feature_filter	Feature(s) to exclude by feature_id, matched the same way as include_feature_filter. NA or "" ignores the filter.
add_qctype	Add the QC type as column
create_dir	A logical value. If TRUE (the default), the parent directory of path is created if it does not yet exist.

See Also

[normalize_by_istd\(\)](#), [quantify_by_istd\(\)](#), [quantify_by_calibration\(\)](#), [calibrate_by_reference\(\)](#)

save_dataset_mztab	Export an experiment to mzTab-M (HUPO-PSI)
--------------------	--

Description

Writes an [MRMhubExperiment](#) to an **mzTab-M 2.0.0-M** file, the HUPO-PSI community standard for reporting quantitative metabolomics / lipidomics results (and the format expected by repositories such as MetaboLights).

Usage

```
save_dataset_mztab(  
  data = NULL,  
  path,  
  variable = "conc",  
  instrument = NULL,  
  contact = NULL,  
  publication = NULL,  
  rt_in_minutes = TRUE,  
  overwrite = TRUE  
)
```

Arguments

data	An MRMhubExperiment object.
path	Output file path. A .mzTab extension is appended if missing.
variable	Feature variable used as abundance. One of "conc", "intensity", "norm_intensity", "area" or "height". Default "conc" (falls back to "intensity" if not quantified).
instrument, contact, publication	Optional single strings used to enrich the MTD metadata header (instrument name, contact name, publication identifier). NULL (default) omits them.
rt_in_minutes	Logical; are feature_rt values in minutes? mzTab-M stores retention time in seconds, so values are multiplied by 60 when TRUE (default).
overwrite	Logical; overwrite an existing file. Default TRUE.

Details

The full dataset is exported: every analysis becomes an assay (QC, blank and calibration samples included), every feature a Small Molecule Feature (SMF) row, and features are grouped by analyte into Small Molecule Summary (SML) rows. A minimal Small Molecule Evidence (SME) row is emitted per feature so all three table types are present.

Abundance. variable selects which feature_* value is written to the abundance_assay[n] columns. The default "conc" uses final concentrations and declares the matching concentration unit; if the experiment has not been quantified, the exporter falls back to raw feature_intensity with an "Arbitrary quantification unit".

What is *not* exported. `mzTab-M` is a quantification report, not a processing-state container. Internal-standard relationships, QC/calibration metrics, drift/batch-correction state and the `qc_type`/batch structure are not representable and are therefore not written (ISTD features are merely flagged via an `opt_global_is_internal_standard` column). See the *mzTab-M export* recipe article for the full mapping.

Value

Invisibly returns the (normalised) output path. Called for its side effect of writing the file.

See Also

[save_report_xlsx\(\)](#), [save_dataset_csv\(\)](#)

Examples

```
## Not run:
save_dataset_mztab(mexp, "experiment.mzTab")
save_dataset_mztab(mexp, "raw.mzTab", variable = "area")

## End(Not run)
```

save_dataset_rds

Save a complete MRMhubExperiment as an RDS file

Description

Writes the whole [MRMhubExperiment](#) to a single `.rds` file: raw data, metadata joins, calibration fits, quantification results and processing history all travel together. This makes the `.rds` a self-contained, portable snapshot of a complete analysis project — a collaborator can open it with `readRDS()` to inspect, or load **mrhub** to re-plot and re-process, without reconstructing the original file paths, input files or metadata versions.

By default a content fingerprint of the object (`rlang::hash()`) is embedded in the saved file and printed. Record it (or compare printed fingerprints) to confirm that a reloaded file — or a copy shared with a colleague — holds the identical object. [read_dataset_rds\(\)](#) reads the file back and verifies the fingerprint.

Usage

```
save_dataset_rds(
  data = NULL,
  path,
  hash = TRUE,
  compress = TRUE,
  overwrite = TRUE,
  create_dir = TRUE
)
```

Arguments

data	A MRMhubExperiment object to save.
path	A character string with the file path. If it does not end in <code>.rds</code> , the extension is appended automatically.
hash	A logical value. If TRUE (default), a content fingerprint of the object is computed with rlang::hash() , embedded in the saved file and printed. The fingerprint is stored as an object attribute and is stripped again by read_dataset_rds() , so it does not affect the reloaded object.
compress	A logical value passed to saveRDS() . If TRUE (the default), the file is gzip-compressed (typically about ten times smaller); FALSE writes it uncompressed (faster, but much larger). Either way the file is read back identically — the compression is detected automatically on load.
overwrite	A logical value indicating whether to overwrite the file if it already exists. Default is TRUE.
create_dir	A logical value. If TRUE (the default), the parent directory of path is created if it does not yet exist.

Value

The path written to, invisibly.

Archiving

The file is a plain R serialization, so it can always be reopened with base R's `readRDS()` — **mrhub** is not required just to read it. Every slot (the dataset, the `annot_*` metadata tables, and the QC and calibration metrics) is stored as an ordinary tibble and can be inspected via the `@` slots or [attributes\(\)](#), which makes the `.rds` a convenient self-contained archive of a complete analysis. **mrhub** (a compatible version) is only needed to re-plot or re-process the object through its methods. For long-term, cross-tool preservation, complement the `.rds` with an open-standard export ([save_dataset_mztab\(\)](#), [save_dataset_summarizedexperiment\(\)](#) or [save_dataset_csv\(\)](#)).

See Also

[read_dataset_rds\(\)](#), [save_report_xlsx\(\)](#), [save_dataset_summarizedexperiment\(\)](#)

Examples

```
mexp <- data_load_example()
path <- file.path(tempdir(), "example_mexp.rds")
save_dataset_rds(mexp, path)
```

save_dataset_summarizedexperiment

Export an experiment to a Bioconductor SummarizedExperiment

Description

Converts an [MRMhubExperiment](#) to a [SummarizedExperiment](#), the Bioconductor container for feature x sample data, and optionally to a [LipidomicsExperiment](#) for use with [lipidr](#). This opens the experiment to the Bioconductor ecosystem - limma for differential abundance, POMA and pmp for preprocessing.

Usage

```
save_dataset_summarizedexperiment(
  data = NULL,
  path = NULL,
  variable = NULL,
  as = c("SummarizedExperiment", "LipidomicsExperiment"),
  filter_data = FALSE,
  overwrite = TRUE
)
```

Arguments

data	An MRMhubExperiment object.
path	Optional file path. When given, the object is written there with saveRDS() and returned invisibly; a .rds extension is appended if missing. When NULL (default) the object is returned.
variable	Feature variables to export as assays, e.g. "conc" or c("intensity", "conc"). NULL (default) exports the numeric feature_* variables present in the data, except the backup snapshots (_orig, _before, _beforecal) and drift-model fit points (_fit), which are internal bookkeeping rather than measurements. Name one of those explicitly to export it.
as	Class to produce. "SummarizedExperiment" (default) or "LipidomicsExperiment", which additionally requires the lipidr package and is only meaningful for lipidomics data.
filter_data	Use QC-filtered data (dataset_filtered, see filter_features_qc()) instead of the full dataset. Default FALSE.
overwrite	Overwrite an existing file at path. Default TRUE.

Details

Layout. Features are rows and analyses are columns, following the [SummarizedExperiment](#) convention. Each feature variable becomes one assay, so feature_intensity, feature_norm_intensity and feature_conc sit side-by-side in the same object and are addressed with `SummarizedExperiment::assay(se,`

"conc"). Assay names drop the feature_ prefix. annot_features becomes rowData(), annot_analyses becomes colData(), and the processing state (status, flags, concentration unit) becomes metadata().

Everything is exported. Internal standards, QC samples, blanks and calibrants are all included and flagged rather than dropped, because downstream tools need them: lipidr requires the istd annotation and pmp's blank filter needs blanks present. Subset when you need to:

```
se[!rowData(se)$is_istd, se$qc_type == "SPL"]
```

Most statistical tools will otherwise happily include blanks and calibrants and return nonsense.

QC metrics are not written to rowData(). To filter features by QC criteria, use `filter_features_qc()` and export with `filter_data = TRUE`. `save_feature_qc_metrics()` exports the metrics themselves.

Value

A SummarizedExperiment (or LipidomicsExperiment). Returned invisibly when path is given.

References

Morgan M, Obenchain V, Hester J, & Pagès H (2026). SummarizedExperiment: A container (S4 class) for matrix-like assays. R package version 1.42.0. doi:10.18129/B9.bioc.SummarizedExperiment <https://bioconductor.org/packages/SummarizedExperiment>

Mohamed A, Molendijk J, & Hill MM (2020). lipidr: A Software Tool for Data Mining and Analysis of Lipidomics Datasets. *Journal of Proteome Research*, 19(7), 2890-2897. doi:10.1021/acs.jproteome.0c00082

See Also

`save_dataset_csv()`, `save_dataset_mztab()`, `save_report_xlsx()`

Examples

```
mexp <- normalize_by_istd(lipidomics_dataset)
mexp <- quantify_by_istd(mexp)

se <- save_dataset_summarizedexperiment(mexp)
SummarizedExperiment::assayNames(se)

# study samples only, internal standards dropped
se[!SummarizedExperiment::rowData(se)$is_istd, se$qc_type == "SPL"]
```

`save_feature_qc_metrics`*Save feature QC metrics to CSV*

Description

This function exports the feature information and QC (Quality Control) metrics from a `MRMhubExperiment` object to a CSV file.

Usage

```
save_feature_qc_metrics(data = NULL, path)
```

Arguments

<code>data</code>	A <code>MRMhubExperiment</code> object containing the QC metrics.
<code>path</code>	A string specifying the file path where the CSV file will be saved.

Value

A tibble with the QC metrics that have been exported.

`save_metadata_msorganiser_template`*Saves a MRMhub Metadata Organizer template*

Description

This function saves a XLSX file with metadata template to the specified location.

Usage

```
save_metadata_msorganiser_template(path = "metadata_msorganiser_template.xlsx")
```

Arguments

<code>path</code>	File path where the MRMhub Metadata Organizer file will be saved. If left empty (default), the file will be saved in the current working directory under the file "metadata_msorganiser_template.xlsx"
-------------------	--

`save_metadata_templates`*Saves an Excel (xlsx) file with metadata templates*

Description

This function saves a XLSX file with metadata template to the specified location.

Usage

```
save_metadata_templates(path = "metadata_template.xlsx")
```

Arguments

path	File path where the XLSX file with templates will be saved. If left empty (default), the file will be saved in the current working directory under the file "metadata_template.xlsx"
------	--

`save_plot`*Save a plot to a file*

Description

Writes a plot created by any of the `plot_*`() functions to a file at a defined physical size and resolution, so figures do not have to be exported with hand-written `ggplot2::ggsave()` calls. Dimensions can be given in millimetres, centimetres, inches, points or pixels, and the same figure can be written in several formats in one call.

Usage

```
save_plot(  
  plot,  
  path,  
  width,  
  height,  
  units = NULL,  
  dpi = NULL,  
  format = NULL,  
  scale = 1,  
  bg = NULL,  
  create_dir = TRUE,  
  overwrite = TRUE,  
  show_plot = TRUE,  
  ...  
)
```

Arguments

plot	The plot to save. A ggplot object (what most plot_*() functions return), a patchwork composition, a list carrying the plot in a plot element (as plot_rla_boxplot() returns), or a list of ggplot objects, which is written as a multi-page PDF. Lists of plots are returned by the paged plot functions with return_plots = TRUE.
path	Output file path. The extension selects the format unless format is given, in which case a known extension is replaced.
width, height	Figure size, in units. Both are required: the physical size of a saved figure is always explicit at the call site, so it can be read off the code rather than inherited from a session setting.
units	Unit of width and height: "mm" (default), "cm", "in", "pt" or "px". NULL uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise "mm".
dpi	Resolution in dots per inch for the raster formats (png, tiff, jpeg), and the reference resolution when units = "px". Ignored for the vector formats. NULL uses the global default set by mrmhub_set_plot_defaults() if one is in effect, otherwise 300.
format	Output format(s): one or more of "pdf", "svg", "png", "tiff", "jpeg". NULL (default) takes the format from the extension of path. Several formats write one file each, sharing the same base name.
scale	Multiplicative scaling factor applied to the plot, as in ggplot2::ggsave(). Values > 1 make text and symbols smaller relative to the figure.
bg	Background colour. NULL (default) uses the plot's own background.
create_dir	A logical value. If TRUE (the default), the parent directory of path is created if it does not yet exist.
overwrite	A logical value indicating whether existing files may be overwritten. Default is TRUE.
show_plot	A logical value. If TRUE (the default), the plot is returned <i>visibly</i> , so that piping into save_plot() still renders the figure in a Quarto or R Markdown chunk and the call reads as one statement. The written paths are then attached as a "paths" attribute. If FALSE, nothing is drawn and the paths are returned invisibly, which is preferable in scripts and loops where re-drawing a dense figure is wasted work.
...	Further arguments passed to the graphics device.

Value

If show_plot = TRUE, the plot itself, visibly, with the written paths in its "paths" attribute. If show_plot = FALSE, invisibly, a character vector of the paths written. For multi-page output the list of plots is returned in place of a single plot.

Preferred formats and devices

Purpose	Format	Device used	Typical dpi
---------	--------	-------------	-------------

Journal figure, vector (default choice)	"pdf"	grDevices::cairo_pdf, else grDevices::pdf	n/a
Figure for further editing (Illustrator, Inkscape)	"svg"	svglite::svglite, else grDevices::svg	n/a
Slides, Quarto HTML, GitHub	"png"	ragg::agg_png, else grDevices::png	150-300
Journal requiring raster submission	"tiff"	ragg::agg_tiff, else grDevices::tiff	300-600

Prefer a **vector** format (pdf, svg) for publication: text stays selectable and searchable, and lines stay sharp at any magnification.

Prefer a **raster** format (png, tiff) when a plot draws very many marks – a [plot_runscatter\(\)](#) page covering several thousand analyses, or a dense [plot_pca\(\)](#) score plot. Every point becomes a separate object in a PDF, so such figures produce very large files that are slow to open and to typeset. Saving them at 300-600 dpi instead keeps the file small with no visible loss.

The optional packages `ragg` and `svglite` are used automatically when installed, giving better text rendering, system-font support and smaller SVG files. When they are absent the equivalent `grDevices` device is used and the output is still correct. Installing both is recommended: `install.packages(c("ragg", "svglite"))`.

PDF output uses the `cairo` device wherever R was built with `cairo` support (`capabilities("cairo")`), because plain `grDevices::pdf()` writes text in a single-byte encoding and silently transliterates anything outside it – an en dash becomes `-`, `>=` replaces the proper symbol. Unit labels such as `umol/L` and statistical annotations routinely depend on those glyphs.

Multi-page output from the paged plot functions ([plot_runscatter\(\)](#), [plot_calibrationcurves\(\)](#), [plot_responsecurves\(\)](#), [plot_feature_correlations\(\)](#)) is PDF only, which is the only format that holds many pages in one file. Use [save_plot\(\)](#) for single figures in any of the other formats.

See Also

[mrmhub_set_plot_defaults\(\)](#) to set units and dpi once for a whole notebook, [plot_runscatter\(\)](#) and the other paged plot functions for multi-page PDF output, and [save_report_xlsx\(\)](#) to export the data.

Examples

```
## Not run:
p <- plot_pca(mexp, variable = "norm_intensity", qc_types = c("BQC", "SPL"))

# A single figure, sized in mm (the default unit)
save_plot(p, "output/pca.pdf", width = 180, height = 120)

# In a notebook: save and show the figure in one statement
plot_pca(mexp, variable = "norm_intensity") |>
  save_plot("output/pca.pdf", width = 180, height = 120)

# The same figure as vector and raster in one call
save_plot(p, "output/pca", format = c("pdf", "png"), width = 180, height = 120)

# In a script or loop, skip the re-draw and collect the paths
paths <- save_plot(p, "output/pca.pdf", width = 180, height = 120,
```

```

        show_plot = FALSE)

# Every runscatter page in one multi-page PDF
pages <- plot_runscatter(mexp, variable = "conc", return_plots = TRUE)
save_plot(pages, "output/runscatter.pdf", width = 280, height = 200,
          show_plot = FALSE)

## End(Not run)

```

save_report_xlsx

Write a data-processing report (Excel)

Description

Generates a data processing report from a `MRMhubExperiment` object and writes it to an Excel file. The report includes information on the data processing steps, quality control metrics, feature concentrations, and metadata. Following tables will be created as sheets in the EXCEL file, in this order:

Usage

```

save_report_xlsx(
  data = NULL,
  path,
  filtered_variable = "conc",
  normalized_variable = NA,
  overwrite = TRUE,
  create_dir = TRUE
)

```

Arguments

- | | |
|---------------------|---|
| data | A MRMhubExperiment object containing original and processed data and meta-data. |
| path | A character string specifying the file name and path for the Excel file. If the path does not include an <code>.xlsx</code> extension, it is added automatically. |
| filtered_variable | A character string specifying the variable name in the filtered data to be exported. It must be one of "conc", "intensity", "norm_intensity", "response", "area", "height", "conc_raw", "rt", or "fwhm". The defined variable name will be included in the sheet name. Default is "conc". |
| normalized_variable | A character string indicating if and which normalized feature values (by reference sample) to include in the report. See also <code>[calibrate_by_reference()]</code> . |
| overwrite | A logical value indicating whether to overwrite the file if it already exists. Default is TRUE. |
| create_dir | A logical value. If TRUE (the default), the parent directory of path is created if it does not yet exist. |

Details

- Info: Report date, author, MRMhub version and the concentration unit.
- Feature_QC_metrics: Quality control metrics of all features. An infinite signal-to-blank ratio (feature not detected in the blank) is written as the text Inf.
- Calibration_metrics: External calibration results per feature.
- QCfilt_x_StudySamples: Feature (QC)-filtered data (variable defined via `filtered_variable`) in study samples ('SPL'). Filter have to be set via `filter_features_qc()`. The *x* corresponds to the `filtered_variable` argument.
- QCfilt_x_AllSamples: Feature (QC)-filtered data (variable defined via `filtered_variable`) in all samples. Filter have to be set via `filter_features_qc()`. The *x* corresponds to the `filtered_variable` argument.
- Raw_Intensity_FullDataset: Raw feature intensities from the full, non-filtered dataset.
- Norm_Intensity_FullDataset: Normalized feature intensities from the full, non-filtered dataset.
- Conc_FullDataset: Final feature concentrations from the full, non-filtered dataset.
- x_NormalizedByRef_Full: Study-sample values normalized by a reference sample (see `calibrate_by_reference()`), if available.
- SampleMetadata: Analysis metadata that was imported and used for processing steps
- FeatureMetadata: Feature metadata that was imported and used for processing steps
- InternalStandards: Internal standards metadata with concentrations
- BatchInfo: Information on batches and positions of first and last analysis/sample in each batch
- Interferences: Derived and declared interference relationships (interfering feature, contribution factor, overlap type, source) with the per-feature correction impact when the correction has been applied.

Internal standards are not included in the concentration and QC-filtered sheets. For reference-normalized variables, sheet names use short labels to stay within Excel's 31 characters, with "Ref" marking values normalized by a reference sample (e.g. QCfilt_ConcRef_StudySamples).

If certain data sets are not available, the function includes empty tables for the corresponding dataset.

Concentration corresponds to the final concentration values after applying isotope correction, and drift and batch correction, if applicable. If any corrections, such as drift or batch correction, were applied to raw or normalized intensities, the exported values will reflect these corrections.

Value

The function does not return a value. It writes the report to the specified Excel file.

See Also

`normalize_by_istd()`, `quantify_by_istd()`, `quantify_by_calibration()`, `calibrate_by_reference()`

Examples

```
## Not run:
# Assuming `mrmhubexp` is a MRMhubExperiment object and `output_path` is a valid path
save_report_xlsx(data = mrmhubexp, path = "output_path/report.xlsx")

## End(Not run)
```

set_analysis_order	<i>Set analysis order</i>
--------------------	---------------------------

Description

Determines the sequence of analyses using either instrument timestamps, the order in the imported raw data file, or the order defined in the Analysis metadata. Note: After changing the analysis order, all post processing steps must be rerun.

Usage

```
set_analysis_order(
  data = NULL,
  order_by = c("timestamp", "resultfile", "metadata")
)
```

Arguments

data	A MRMhubExperiment object
order_by	Character string specifying the ordering method. Must be one of "timestamp" (requires timestamp data in imported results), "resultfile" (uses order from imported data file), or "metadata" (uses order from analysis metadata)

Value

An updated [MRMhubExperiment](#) object with ordered analyses

Examples

```
file_path <- system.file("extdata", "MRMhub_demo.tsv", package = "mrmhub")
mexp <- MRMhubExperiment()
mexp <- import_data_mrmhub(mexp, path = file_path, import_metadata = TRUE)

# Order by timestamp (if available)
mexp <- set_analysis_order(mexp, "timestamp")

# Order by metadata definition
mexp <- set_analysis_order(mexp, "metadata")
```

set_intensity_var	<i>Set default variable to be used as feature raw signal value</i>
-------------------	--

Description

Sets the raw signal variable used for calculations starting from raw signal values (i.e., normalization). Note that this set variable must be part of the originally imported data. Processed data variables (e.g., normalized intensities and concentrations) can not be set as default feature intensity variable.

Usage

```
set_intensity_var(  
  data = NULL,  
  variable_name,  
  auto_select = FALSE,  
  warnings = TRUE,  
  ...  
)
```

Arguments

data	MRMhubExperiment object
variable_name	Feature variable to be used as default feature intensity for downstream processing.
auto_select	If TRUE then the first available of these will be used as default: "intensity", "response", "area", "height".
warnings	Suppress warnings
...	Feature variables to search for one-by-one when auto_select = TRUE

Value

[MRMhubExperiment](#) object

set_lipid_class	<i>Set feature classes from lipid names</i>
-----------------	---

Description

Derives lipid classes from the feature_ids with the Goslin lipid name parser and writes them to feature_class in the feature metadata, the dataset and the QC metrics. For sphingolipids, the class includes the number of oxygens on the sphingoid base (e.g. Cer;02, SM;02). Requires the Bioconductor package rgoslin (BiocManager::install("rgoslin")).

Usage

```
set_lipid_class(data = NULL, overwrite = FALSE)
```

Arguments

data	A MRMhubExperiment object.
overwrite	Logical. If FALSE (default), only features without a feature_class get one; if TRUE, all classes are replaced. Features whose name cannot be parsed keep their class.

Value

MRMhubExperiment object with updated feature_class.

See Also

[parse_lipid_feature_names\(\)](#)

summarize_interferences

Summarize interference relationships

Description

Prints and returns a rollup of the interference relationships defined for the experiment – automatically derived ([calc_isotopic_interferences\(\)](#)) and declared (custom) – so they can be reviewed before, and after, applying a correction. Reports the affected features, a split by source and overlap type, the contribution-factor range and, once the data are corrected, the per-feature median impact.

Usage

```
summarize_interferences(data = NULL)
```

Arguments

data	A MRMhubExperiment.
------	---------------------

Value

Invisibly, a tibble of the assembled, de-duplicated interference edges (with a pct_impact column when the data are already corrected). Called mainly for the printed summary.

See Also

[calc_isotopic_interferences\(\)](#), [correct_isotopic_interferences\(\)](#), [correct_custom_interferences\(\)](#)

Index

- * **EDA plots**
 - plot_abundanceprofile, 93
- * **QC plots**
 - plot_feature_correlations, 100
 - plot_interference_correction, 103
 - plot_matrixeffects, 106
 - plot_normalization_qc, 108
 - plot_pca, 112
 - plot_pca_loading, 115
 - plot_qc_interference_impact, 118
 - plot_qc_summary_byclass, 119
 - plot_qc_summary_overall, 120
 - plot_qcmetrics_comparison, 121
 - plot_qla_boxplot, 129
 - plot_rt_vs_chain, 133
 - plot_runscatter, 135
 - plot_runsequence, 142
- * **calibration plots**
 - plot_calibrationcurves, 96
 - plot_responsecurves, 125
- * **datasets**
 - lipidomics_dataset, 81
 - quant_lcms_dataset, 145
- \$, MRMhubExperiment-method, 5
- add_metadata, 5
- add_metadata(), 71
- attributes(), 148, 153
- build_workflow, 6
- build_workflow(), 50
- calc_average_molweight, 7
- calc_calibration_results, 8
- calc_calibration_results(), 55, 56, 146, 147
- calc_isotopic_interferences, 10
- calc_isotopic_interferences(), 37, 164
- calc_qc_metrics, 12
- calc_qc_metrics(), 111, 121, 124
- calibrate_by_reference, 15
- calibrate_by_reference(), 150, 161
- check_setup, 17
- correct_batch_centering, 18
- correct_batch_centering(), 20, 21, 23
- correct_batch_combat, 19
- correct_batch_combat(), 23
- correct_batch_serrf, 21
- correct_batch_serrf(), 20, 21
- correct_custom_interferences, 23
- correct_custom_interferences(), 36, 37, 164
- correct_drift, 24
- correct_drift_cubicspline, 26
- correct_drift_gam, 29
- correct_drift_gaussiankernel, 31
- correct_drift_loess, 33
- correct_drift_loess(), 21, 23
- correct_interference_manual, 36
- correct_interference_manual(), 24
- correct_isotopic_interferences, 37
- correct_isotopic_interferences(), 10, 11, 23, 24, 36, 164
- cv, 38
- cv_log, 39
- data_load_example, 39
- data_sum_features, 40
- detect_outlier_pca, 41
- exclude_analyses, 42
- exclude_features, 43
- filter_features_qc, 43
- filter_features_qc(), 94, 99, 101, 102, 107, 110, 111, 113, 116, 117, 120, 123, 124, 126, 128, 150, 154, 155, 161
- fun_cspline, 47
- fun_gam_smooth, 48

- `fun_gauss.kernel.smooth`, 49
- `fun_loess`, 49
- `fun_loess()`, 25
- `generate_workflow_qmd`, 50
- `generate_workflow_qmd()`, 6
- `get_analyis_end`, 51
- `get_analyis_start`, 52
- `get_analysis_breaks`, 52
- `get_analysis_count`, 53
- `get_analysis_duration`, 53
- `get_analyticaldata`, 54
- `get_batch_boundaries`, 54
- `get_calibration_metrics`, 55
- `get_feature_count`, 56
- `get_featurelist`, 57
- `get_iqr_tails`, 58
- `get_mad_tails`, 58
- `get_outlier_bounds`, 59
- `get_qc_bias_variability`, 60
- `get_response_curve_stats`, 62
- `get_response_curve_stats()`, 14
- `get_runtime_median`, 63
- `ggplot2::ggsave()`, 157, 158
- `import_data_csv_long`, 63
- `import_data_csv_long()`, 91
- `import_data_csv_wide`, 66
- `import_data_masshunter`, 68
- `import_data_mrmhub`, 69
- `import_data_mrmhub()`, 71
- `import_data_mztab`, 70
- `import_data_skyline`, 72
- `import_metadata_analyses`, 73
- `import_metadata_features`, 75
- `import_metadata_from_data`, 76
- `import_metadata_from_data()`, 71
- `import_metadata_istds`, 77
- `import_metadata_msorganiser`, 78
- `import_metadata_qcconcentrations`, 79
- `import_metadata_responsecurves`, 80
- `licar_pattern_choices`, 81
- `lipidomics_dataset`, 81
- `mgcv::gam()`, 29, 31
- `mirai::daemons()`, 23
- `mrmhub_enable_cli_color`, 82
- `mrmhub_get_plot_defaults`, 83
- `mrmhub_get_plot_defaults()`, 85
- `mrmhub_reset_plot_defaults`, 83
- `mrmhub_reset_plot_defaults()`, 85
- `mrmhub_set_plot_defaults`, 84
- `mrmhub_set_plot_defaults()`, 83, 95, 98, 102, 105, 107, 108, 110, 114, 117, 119–121, 123, 124, 126, 127, 132, 134, 139, 143, 158, 159
- `mrmhub_status`, 85
- `mrmhub_status()`, 149
- `MRMhubExperiment`, 5, 6, 8, 10–12, 15, 17–28, 30–37, 40–44, 47, 51–55, 57, 61–81, 85, 86, 86–88, 92, 118, 146–154, 156, 160, 162–164
- `MRMhubExperiment()`, 71
- `MRMhubExperiment-class`, 86
- `normalize_by_istd`, 87
- `normalize_by_istd()`, 14, 17, 41, 111, 124, 150, 161
- `options()`, 82, 83, 85
- `parse_lipid_feature_names`, 88
- `parse_lipid_feature_names()`, 111, 124, 164
- `parse_masshunter_csv`, 89
- `parse_mrmhub_result`, 90
- `parse_plain_long_csv`, 90
- `parse_plain_wide_csv`, 91
- `parse_skyline_result`, 92
- `plot_abundanceprofile`, 93
- `plot_calibrationcurves`, 96
- `plot_calibrationcurves()`, 99, 103, 128, 129, 141, 159
- `plot_feature_correlations`, 100
- `plot_feature_correlations()`, 99, 103, 106, 108, 111, 115, 117, 119–121, 124, 128, 133, 135, 141, 143, 159
- `plot_interference_correction`, 103
- `plot_interference_correction()`, 103, 108, 111, 115, 117, 119–121, 124, 133, 135, 141, 143
- `plot_matrixeffects`, 106
- `plot_matrixeffects()`, 103, 106, 111, 115, 117, 119–121, 124, 133, 135, 141, 143
- `plot_normalization_qc`, 108

- `plot_normalization_qc()`, [103](#), [106](#), [108](#),
[115](#), [117](#), [119–121](#), [124](#), [133](#), [135](#),
[141](#), [143](#)
- `plot_pca`, [112](#)
- `plot_pca()`, [99](#), [103](#), [106](#), [108](#), [111](#), [117](#),
[119–121](#), [124](#), [128](#), [133](#), [135](#), [141](#),
[143](#), [159](#)
- `plot_pca_loading`, [115](#)
- `plot_pca_loading()`, [103](#), [106](#), [108](#), [111](#),
[115](#), [119–121](#), [124](#), [133](#), [135](#), [141](#),
[143](#)
- `plot_qc_interference_impact`, [118](#)
- `plot_qc_interference_impact()`, [103](#), [106](#),
[108](#), [111](#), [115](#), [117](#), [120](#), [121](#), [124](#),
[133](#), [135](#), [141](#), [143](#)
- `plot_qc_summary_byclass`, [119](#)
- `plot_qc_summary_byclass()`, [103](#), [106](#), [108](#),
[111](#), [115](#), [117](#), [119–121](#), [124](#), [133](#),
[135](#), [141](#), [143](#)
- `plot_qc_summary_overall`, [120](#)
- `plot_qc_summary_overall()`, [103](#), [106](#), [108](#),
[111](#), [115](#), [117](#), [119](#), [120](#), [124](#), [133](#),
[135](#), [141](#), [143](#)
- `plot_qcmetrics_comparison`, [121](#)
- `plot_qcmetrics_comparison()`, [103](#), [106](#),
[108](#), [111](#), [115](#), [117](#), [119–121](#), [133](#),
[135](#), [141](#), [143](#)
- `plot_responsecurves`, [125](#)
- `plot_responsecurves()`, [99](#), [100](#), [103](#), [128](#),
[141](#), [159](#)
- `plot_rla_boxplot`, [129](#)
- `plot_rla_boxplot()`, [103](#), [106](#), [108](#), [111](#),
[115](#), [117](#), [119–121](#), [124](#), [135](#), [141](#),
[143](#), [158](#)
- `plot_rt_vs_chain`, [133](#)
- `plot_rt_vs_chain()`, [103](#), [106](#), [108](#), [111](#),
[115](#), [117](#), [119–121](#), [124](#), [133](#), [141](#),
[143](#)
- `plot_runscatter`, [135](#)
- `plot_runscatter()`, [19](#), [21](#), [23](#), [26](#), [29](#), [31](#),
[33](#), [99](#), [103](#), [106](#), [108](#), [111](#), [115](#), [117](#),
[119–121](#), [124](#), [128](#), [133](#), [135](#), [141](#),
[143](#), [159](#)
- `plot_runsequence`, [142](#)
- `plot_runsequence()`, [103](#), [106](#), [108](#), [111](#),
[115](#), [117](#), [119–121](#), [124](#), [133](#), [135](#),
[141](#)
- `pooled_rsd`, [143](#)
- `pooled_sd`, [144](#)
- `pooled_sd()`, [143](#)
- `purrr::in_parallel()`, [23](#)
- `quant_lcms_dataset`, [145](#)
- `quantify_by_calibration`, [145](#)
- `quantify_by_calibration()`, [10](#), [14](#), [17](#), [61](#),
[97](#), [148](#), [150](#), [161](#)
- `quantify_by_istd`, [147](#)
- `quantify_by_istd()`, [14](#), [17](#), [86](#), [147](#), [150](#),
[161](#)
- `ranger::ranger()`, [22](#), [23](#)
- `read_dataset_rds`, [148](#)
- `read_dataset_rds()`, [152](#), [153](#)
- `readRDS()`, [148](#)
- `rlang::hash()`, [152](#), [153](#)
- `save_dataset_csv`, [149](#)
- `save_dataset_csv()`, [152](#), [153](#), [155](#)
- `save_dataset_mztab`, [151](#)
- `save_dataset_mztab()`, [71](#), [153](#), [155](#)
- `save_dataset_rds`, [152](#)
- `save_dataset_rds()`, [148](#), [149](#)
- `save_dataset_summarizedexperiment`, [154](#)
- `save_dataset_summarizedexperiment()`,
[153](#)
- `save_feature_qc_metrics`, [156](#)
- `save_feature_qc_metrics()`, [155](#)
- `save_metadata_msorganiser_template`,
[156](#)
- `save_metadata_templates`, [157](#)
- `save_plot`, [157](#)
- `save_plot()`, [84](#), [85](#), [99](#), [100](#), [103](#), [128](#), [129](#),
[141](#), [159](#)
- `save_report_xlsx`, [160](#)
- `save_report_xlsx()`, [152](#), [153](#), [155](#), [159](#)
- `saveRDS()`, [153](#), [154](#)
- `set_analysis_order`, [162](#)
- `set_analysis_order()`, [41](#)
- `set_intensity_var`, [163](#)
- `set_intensity_var()`, [41](#)
- `set_lipid_class`, [163](#)
- `stats::loess()`, [34](#)
- `stats::smooth.spline()`, [26](#), [28](#)
- `summarize_interferences`, [164](#)
- `sva::ComBat()`, [20](#), [21](#)
- `withr::local_options()`, [85](#)