Package: CalibraCurve (via r-universe)

May 29, 2026

Title Calibration curves for targeted proteomics, lipidomics and metabolomics data

Version 1.2.0

Description CalibraCurve is a computational tool designed to generate calibration curves for targeted mass spectrometry-based quantitative data. It is applicable to various omics disciplines, including proteomics, lipidomics, and metabolomics. The package also offers functionalities for data and calibration curve visualization and concentration prediction from new datasets based on the established curves.

License BSD 3-clause License + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports checkmate, dplyr, ggplot2, magrittr, openxlsx, scales, SummarizedExperiment, tidyr

Suggests BiocStyle, knitr, msqc1, RefManageR, rmarkdown, sessioninfo, testthat, vdiffr

biocViews Proteomics, Lipidomics, Metabolomics, Regression, MassSpectrometry, Visualization

URL https://github.com/mpc-bioinformatics/CalibraCurve

BugReports https://github.com/mpc-bioinformatics/CalibraCurve/issues

Depends R (>= 4.5.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev zlib1g-dev

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

Date/Publication 2026-04-28 13:05:42 UTC

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

RemoteRef RELEASE_3_23

RemoteSha 00cd320e9683189062c82464d5b9a02f2a2327d7

Contents

`.calcContPrelimRanges` *PLR: Calculate key features for the existing continuous preliminary ranges*

Description

PLR: Calculate key features for the existing continuous preliminary ranges

Usage

```
.calcContPrelimRanges(index)
```

Arguments

index	**integer** Integer vector containing indices of concentration levels with CV < threshold.

Value

Data.frame with one range in each row. Contains the columns "startPoints", "endPoints", and "extent".

.calcCV *PLR: calculate CV for one concentration level*

Description

PLR: calculate CV for one concentration level

Usage

```
.calcCV(x)
```

Arguments

x	**data.frame** Data.frame containing data for a specific concentration level.

Value

numeric(1)
Coefficient of variation (CV) for the given concentration level.

.calcLinearModel *FLR: calculate weighted or unweighted linear model*

Description

FLR: calculate weighted or unweighted linear model

Usage

```
.calcLinearModel(x, weights = NULL)
```

Arguments

x	**list of data.frames** List of data frames (one entry for each concentration level), e.g. output "dataPrelim" from calculate_PLR.
weights	**numeric** Vector of weights as calculated by applying .calcWeights (default is NULL and will result in an unweighted model).

Value

Fit of the linear model as an object of class "lm".

.calcPerBias *FLR: calculate percent bias for a specific concentration level*

Description

FLR: calculate percent bias for a specific concentration level

Usage

```
.calcPerBias(x, LMfit, expConc)
```

Arguments

x	**data.frame** Data.frame containing data for a specific concentration level.
LMfit	**lm object** Linear model fit as calculated by .calcLinearModel.
expConc	**numeric(1)** Expected (known) concentration level.

Value

Vector of percent bias values for each data point in this concentration level.

.calcPerBiasAvgSDCV *FLR: calculate average, SD and CV percent bias for each concentration level*

Description

FLR: calculate average, SD and CV percent bias for each concentration level

Usage

```
.calcPerBiasAvgSDCV(x, method = "mean")
```

Arguments

x	**list** Result of .calcPerBiasLevels.
method	**character(1)** Method for calculating the average percent bias: "mean" (default) or "median".

Value

data frame with 3 columns: avgPerBias, stdDevPerBias, CV_PerBias
each row is one concentration level

.calcPerBiasLevels *FLR: calculate list of percent bias values for all concentration levels*

Description

FLR: calculate list of percent bias values for all concentration levels

Usage

```
.calcPerBiasLevels(x, LMfit)
```

Arguments

x	**list of data.frames** List of data frames (one entry for each concentration level), e.g. output "dataPrelim" from calculate_PLR.
LMfit	**lm object** Linear model fit as calculated by .calcLinearModel.

Value

List with percent bias values for each data point per concentration level

.calcResponseFactors *Calculate Response factors*

Description

Function, which calculates a response factor for a single data point

Usage

```
.calcResponseFactors(x, intercept, expConc)
```

Arguments

x	*data.frame** Data.frame containing data for a specific concentration level.
intercept	**numeric(1)** Intercept of the linear model.
expConc	**numeric(1)** Expected concentration (known concentration value).

Details

Formula obtained from: Green, J. M., A practical guide to analytical method validation. Analytical Chemistry 1996, 68, 305A-309A.

Value

vector of response factors for this specific concentration level

.calcWeights *FLR: calculate weights for linear model for a specific concentration level*

Description

FLR: calculate weights for linear model for a specific concentration level

Usage

```
.calcWeights(x, weightingMethod = "1/x^2")
```

Arguments

x	**data.frame** Data.frame containing data for a specific concentration level.
weightingMethod	**character(1)** Method for weighting (currently "1/x", "1/x^2" and "None" are supported, default is "1/x^2").

Value

Vector of a constant weights for each measurement in this concentration level.

.checkFinalRange *FLR: checks if final linear range has been reached (compare average percent bias with threshold)*

Description

FLR: checks if final linear range has been reached (compare average percent bias with threshold)

Usage

```
.checkFinalRange(perBiasInfo, perBiasThres = 20)
```

Arguments

perBiasInfo	**data.frame** Result of .calcPerBiasAvgSDCV.
perBiasThres	**numeric(1) numeric(1)** Threshold for average percent bias in percent, default is 20.

Value

TRUE if both lowest and highest concentration level passed the check (and the final linear range is reached), FALSE otherwise

`.checkNumberReplicates` *Data preprocessing: Helper function to check for sufficient number of replicates for a specific concentration level*

Description

Data preprocessing: Helper function to check for sufficient number of replicates for a specific concentration level

Usage

```
.checkNumberReplicates(x, data, minReplicates)
```

Arguments

`x`	**numeric(1)** concentration level to check
`data`	**list of data.frames** list of data.frames (each dataframe contains data for a specific concentration level)
`minReplicates`	**integer(1)** minimal number of data points per concentration level

Value

logical(1)
TRUE if there are enough replicates, else FALSE

`.filterConcentrationLevel` *Data preprocessing: Helper function to select all rows from a specific concentration level*

Description

Data preprocessing: Helper function to select all rows from a specific concentration level

Usage

```
.filterConcentrationLevel(x, data)
```

Arguments

x	**numeric(1)** concentration level to select
data	**data.frame** data set to be filtered (e.g., result of readDataTable)

Value

data.frame

.prepareAnnotationData *Prepare annotation data*

Description

Prepare annotation data

Usage

```
.prepareAnnotationData(D_calib)
```

Arguments

D_calib	**data.frame** Result of .prepareCalibData

Value

Dataframe with annotation data for the plot.

.prepareCalibData *Prepare calibration data*

Description

Prepare calibration data

Usage

```
.prepareCalibData(CC_RES)
```

Arguments

CC_RES	**list** Result object of CalibraCurve.

Value

Dataframe with calibration data for plotting.

.prepareCurveData *Prepare curve data*

Description

Prepare curve data

Usage

```
.prepareCurveData(CC_RES, D_calib)
```

Arguments

CC_RES	**list** Results of CalibraCurve. Each list element is the result of calc_single_curve
D_calib	**data.frame** Result of .prepareCalibData

Value

Dataframe with data points on a grid to plot the calibration curve.

.saveCCResult *Save results of CalibraCurve*

Description

Save results of CalibraCurve

Usage

```
.saveCCResult(CC_res, output_path, suffix = "")
```

Arguments

CC_res	**list** Result object of CalibraCurve.
output_path	**character(1)** Path to the output directory.
suffix	**character(1)** Suffix for the output files, ideally starting with "_" (default is "").

Value

Returns nothing, but the function saves the results to the specified output path.

`.saveTablesAndPlots` *Save result tables and plots*

Description

Save result tables and plots

Usage

```
.saveTablesAndPlots(
  RES,
  output_path,
  pl_CC_list,
  pl_RF_list,
  summary_tab,
  CC_plot_width,
  CC_plot_height,
  RF_plot_width,
  RF_plot_height,
  plot_type,
  plot_dpi,
  device
)
```

Arguments

`RES`	**list** Results of `CalibraCurve`. Each list element is the result of `calc_single_curve`
`output_path`	**character(1)** Folder to save results (table and plots). If NULL (default), results are not saved.
`pl_CC_list`	**list** List of calibration curves (ggplot objects, results of `plotCalibraCurve` (CC_plot)).
`pl_RF_list`	**list** List of response factor plots (ggplot objects, results of `plotResponseFactors`).
`summary_tab`	**data.frame** Table with summary information, result of `plotCalibraCurve` (annotation_dat))
`CC_plot_width`	**numeric(1)** Plot width in cm (default is 10).
`CC_plot_height`	**numeric(1)** Plot height in cm (default is 10).
`RF_plot_width`	**numeric(1)** Plot width in cm (default is 10).
`RF_plot_height`	**numeric(1)** Plot height in cm (default is 10).

`plot_type` **character(1)**
Type of plot for calibration curves: "single_plots" (default, generate a separate plot for each substance), "multiplot" (generate a graphic with subplots for each substance) or "all_in_one" (generate a single plot with all substances).

`plot_dpi` **numeric(1)**
Plot resolution in dpi (default is 300).

`device` **character(1)**
Device for saving the plot (default is "png"). Other options include "pdf", "jpeg", "tiff", "svg" etc. For details see `ggsave`.

Value

Invisible NULL. Plots and the summary table are saved to the hard drive.

`.selctConcLevel` *FLR: selects the highest or lowest concentration level for removal*

Description

FLR: selects the highest or lowest concentration level for removal

Usage

```
.selctConcLevel(x, perBiasT = 20, consPerBiasCV = TRUE, perBiasDistT = 10)
```

Arguments

`x` **data.frame** Result of `.calcPerBiasAvgSDCV`.

`perBiasT` **numeric(1)**
Threshold for average percent bias in percent, default is 20.

`consPerBiasCV` consider CV? default is TRUE

`perBiasDistT` **numeric(1)**
Threshold for the difference in average percent bias in percent (for lower differences, CV will be considered), default is 10. Only used if consPerBiasCV is TRUE.

Value

TRUE, if lowest concentration will be removed, FALSE if highest will be removed

`assemble_results` *Assemble result tables*

Description

Assemble result tables

Usage

```
assemble_results(
  X,
  dataCleaned,
  PLR_res,
  resFacDataV,
  avgResFacDataV,
  FLR_res,
  mod,
  cvThres = 20,
  RfThresL = 80,
  RfThresU = 120,
  substance = "substance1"
)
```

Arguments

`X`	**data.frame** Original data set, e.g. result of `readDataTable`.
`dataCleaned`	**list of data.frames** Cleaned data, result of `cleanData`.
`PLR_res`	**list** Result object of `calculate_PLR`.
`resFacDataV`	**list** List of response factor values for each concentration level.
`avgResFacDataV`	**numeric** Vector of mean response factor values for the different concentration levels.
`FLR_res`	**list** Result object of `calculate_FLR`.
`mod`	**lm object** Final linear model fit (object "mod" from results of `calculate_FLR`).
`cvThres`	**numeric(1)** Threshold for CV per concentration level in percent (default is 20).
`RfThresL`	**numeric(1)** Lower threshold for response factor in percent (default is 80).
`RfThresU`	**numeric(1)** Upper threshold for response factor in percent (default is 120).
`substance`	**character(1)** Name of the substance (default is "substance1").

Value

List with the following elements:

- result_table_conc_levels: Result table with one line for each concentration level.
- result_table_obs: Result table with one line per observation (e.g. individual response factors for each data point).

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
package = "CalibraCurve")
D_list <- readDataSE(file, concColName = "amount_fmol",
        substColName = "Substance", assayNumber = 1)
data_cleaned <- cleanData(D_list[[1]])

RES_PLR <- calculate_PLR(data_cleaned, calcContinuousPrelimRanges = FALSE)
RES_FLR <- calculate_FLR(RES_PLR$dataPrelim)
RFs <- calcRF(data_cleaned, mod = RES_FLR$mod)

assemble_results(
    X = D_list[[1]],
    dataCleaned = data_cleaned,
    PLR_res = RES_PLR,
    resFacDataV = RFs$RFs,
    avgResFacDataV = RFs$meanRFs,
    FLR_res = RES_FLR,
    mod = RES_FLR$mod
)
```

---

calc_single_curve *Calculate all necessary steps for a single calibration curve*

Description

Calculate all necessary steps for a single calibration curve

Usage

```
calc_single_curve(
  D,
  substance = "substance",
  minReplicates = 3,
  cvThres = 20,
  calcContinuousPrelimRanges = FALSE,
  weightingMethod = "1/x^2",
  centralTendencyMeasure = "mean",
  perBiasThres = 20,
  considerPerBiasCV = TRUE,
```

```
  perBiasDistThres = 10,
  RfThresL = 80,
  RfThresU = 120
)
```

Arguments

D	**data.frame** data set, e.g. result of readDataTable or readDataSE. Has to include exactly two columns: "Concentration" and "Measurement".
substance	**character(1)** Name of the substance (default is "substance"). Will be added to the result files and may be used when plotting multiple calibration curves in one plot.
minReplicates	**integer(1)** Minimal number of replicates per concentration level. Concentration levels with too few data points will be removed.
cvThres	**numeric(1)** Threshold for CV per concentration level in percent (default is 20).
calcContinuousPrelimRanges	**logical(1)** If TRUE, the longest continuous range is selected (default is TRUE). If FALSE, gaps with CVs larger than the threshold may be included.
weightingMethod	**character(1)** Method for weighting (currently "1/x", "1/x^2" and "None" are supported, default is 1/x^2).
centralTendencyMeasure	**character(1)** Method for calculating average percent bias, "mean" (default) or "median".
perBiasThres	**numeric(1)** Threshold for average percent bias in percent, default is 20.
considerPerBiasCV	**logical(1)** If TRUE, CV is considered for the elimination of the concentration level (default). CV will only be considered if the difference in percent bias values is lower than perBiasDistThres.
perBiasDistThres	**numeric(1)** Threshold for the difference in average percent bias in percent (for lower differences, CV will be considered), default is 10.
RfThresL	**numeric(1)** Lower threshold for response factor in percent (default is 80).
RfThresU	**numeric(1)** Upper threshold for response factor in percent (default is 120).

Value

List with the following elements:

- mod: lm-object containing the final linear model.
- final_linear_range: vector of concentration levels that are part of the final linear range.
- weightingMethod: weighting method used for the linear model.
- result_table_conc_levels: Result table with one line for each concentration level (generated by assemble_results).
- result_table_obs: Result table with one line per observation (e.g. individual response factors for each data point) (generated by assemble_results).

Examples

```
data_path <- system.file("extdata", "MSQC1_xlsx", "GGPFSDSYR_QTRAP_y5.xlsx",
        package = "CalibraCurve")
D <- readDataTable(dataPath = data_path, concCol = 16, measCol = 12,
        fileType = "xlsx")
calc_single_curve(D = D)
```

---

calcRF *Calculate Response factors*

Description

Final linear range: Function, which returns a list with response factor values for a data set (given as list)

Usage

```
calcRF(x, mod)
```

Arguments

x	**list of data.frames** List of data.frames containing data for each concentration level (result from cleanData).
mod	**lm object** Final linear model fit (object "mod" from results of calculate_FLR).

Value

List with the following elements:

- RFs: List with response factors for each concentration level.
- meanRFs: Vector with mean response factors for each concentration level.

response factor values for each concentration level.

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
package = "CalibraCurve")
D_list <- readDataSE(file, concColName = "amount_fmol",
        substColName = "Substance", assayNumber = 1)
data_cleaned <- cleanData(D_list[[1]])
RES_PLR <- calculate_PLR(data_cleaned, calcContinuousPrelimRanges = FALSE)
RES_FLR <- calculate_FLR(RES_PLR$dataPrelim)

calcRF(data_cleaned, mod = RES_FLR$mod)
```

`calculate_FLR` *Calculate the final linear range*

Description

Calculate the final linear range

Usage

```
calculate_FLR(
  dataPrelim,
  weightingMethod = "1/x^2",
  centralTendencyMeasure = "mean",
  perBiasThres = 20,
  considerPerBiasCV = TRUE,
  perBiasDistThres = 10
)
```

Arguments

`dataPrelim` **list of data.frames**
List of data.frames containing data only within the preliminary linear range (result from `calculate_PLR`).

`weightingMethod` **character(1)**
Method for weighting (currently "1/x", "1/x^2" and "None" are supported, default is 1/x^2).

`centralTendencyMeasure` **character(1)**
Method for calculating average percent bias, "mean" (default) or "median".

`perBiasThres` **numeric(1)**
Threshold for average percent bias in percent, default is 20.

`considerPerBiasCV` **logical(1)**
If TRUE, CV is considered for the elimination of the concentration level (default). CV will only be considered if the difference in percent bias values is lower than perBiasDistThres.

perBiasDistThres
: **numeric(1)**
Threshold for the difference in average percent bias in percent (for lower differences, CV will be considered), default is 10. Only relevant if considerPerBiasCV = TRUE.

Value

List with the following elements:

- dataFinal: List of data.frames containing data only within the final linear range.
- mod: lm-object containing the final linear model (weighted or unweighted, depending on weightingMethod).
- perBias: result list of .calcPerBiasLevels.
- perBiasAvgSDCV: result data.frame of .calcPerBiasAvgSDCV.

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
        package = "CalibraCurve")
D_list <- readDataSE(file, concColName = "amount_fmol",
        substColName = "Substance", assayNumber = 1)
data_cleaned <- cleanData(D_list[[1]])
RES_PLR <- calculate_PLR(data_cleaned, calcContinuousPrelimRanges = FALSE)

calculate_FLR(RES_PLR$dataPrelim)
```

---

calculate_PLR *Calculate preliminary linear range (PLR)*

Description

Calculate preliminary linear range (PLR)

Usage

```
calculate_PLR(dataCleaned, cvThres = 20, calcContinuousPrelimRanges = TRUE)
```

Arguments

dataCleaned
: **data.frame**
Data cleaned by cleanData.

cvThres
: **numeric(1)**
Threshold for CV per concentration level in percent (default is 20).

calcContinuousPrelimRanges
: **logical(1)**
If TRUE, the longest continuous range fulfilling the CV threshold is selected (default is TRUE). If FALSE, gaps with CVs larger than the threshold may be included.

Value

List with the following elements:

- dataPrelim: List of data.frames containing data only within the preliminary linear range.
- concLevelsCV: Vector with the calculated CV for each concentration level.
- prelimConcLevels: Vector with the concentration levels within the preliminary linear range.

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
package = "CalibraCurve")
D_list <- readDataSE(file, concColName = "amount_fmol",
        substColName = "Substance", assayNumber = 1)
data_cleaned <- cleanData(D_list[[1]])

calculate_PLR(data_cleaned, calcContinuousPrelimRanges = FALSE)
```

CalibraCurve *CalibraCurve*

Description

CalibraCurve

Usage

```
CalibraCurve(
  D_list,
  output_path = NULL,
  substance = "substance",
  minReplicates = 3,
  cvThres = 20,
  calcContinuousPrelimRanges = FALSE,
  weightingMethod = "1/x^2",
  centralTendencyMeasure = "mean",
  perBiasThres = 20,
  considerPerBiasCV = TRUE,
  perBiasDistThres = 10,
  RfThresL = 80,
  RfThresU = 120,
  plot_type = "single_plots",
  RF_colour_threshold = "orange",
  RF_colour_within = "#00BFC4",
  RF_colour_outside = "#F8766D",
  device = "png",
  CC_plot_width = 15,
  CC_plot_height = 10,
```

```
  RF_plot_width = 15,
  RF_plot_height = 10,
  base_size = 11,
  RF_legend = FALSE,
  plot_dpi = 300,
  verbose = TRUE,
  ...
)
```

Arguments

`D_list`	**data.frame/list** Data.frame or list of data.frames (one for one specific substance, each must contain two columns: Concentration and Measurement)
`output_path`	**character(1)** Folder to save results (table and plots). If NULL (default), results are not saved.
`substance`	**character(1)** Name of the substance (default is "substance1"). Will be added to the result files and will be used as labels in the plots.
`minReplicates`	**integer(1)** Minimal number of replicates/data points per concentration level. Concentration levels with too few data points will be removed.
`cvThres`	**numeric(1)** Threshold for CV per concentration level in percent (default is 20).
`calcContinuousPrelimRanges`	**logical(1)** If TRUE, the longest continuous range is selected (default is TRUE). If FALSE, gaps with CVs larger than the threshold may be included.
`weightingMethod`	**character(1)** Method for weighting (currently "1/x", "1/x^2" and "None" are supported, default is 1/x^2).
`centralTendencyMeasure`	**character(1)** Method for calculating average percent bias, "mean" (default) or "median".
`perBiasThres`	**numeric(1)** Threshold for average percent bias in percent, default is 20.
`considerPerBiasCV`	**logical(1)** If TRUE, CV is considered for the elimination of the concentration level (default). CV will only be considered if the difference in percent bias values is lower than perBiasDistThres.
`perBiasDistThres`	**numeric(1)** Threshold for the difference in average percent bias in percent (for lower differences, CV will be considered), default is 10.

RfThresL **numeric(1)**
Lower threshold for response factor in percent (default is 80).

RfThresU **numeric(1)**
Upper threshold for response factor in percent (default is 120).

plot_type **character(1)**
Type of plot for calibration curves: "single_plots" (default, generate a separate plot for each substance), "multiplot" (generate a graphic with subplots for each substance) or "all_in_one" (generate a single plot with all substances).

RF_colour_threshold
character(1)
Response factor plot: Colour for horizontal threshold lines, default is "orange".

RF_colour_within
character(1)
Response factor plot: Colour for points and lines within the final linear range, default is "#00BFC4" (default ggplot colour).

RF_colour_outside
character(1)
Response factor plot: Colour for horizontal outside of the final linear range, default is "#F8766D" (default ggplot colour).

device **character(1)**
Device for saving the plot (default is "png"). Other options include "pdf", "jpeg", "tiff", "svg" etc. For details see ggsave.

CC_plot_width **numeric(1)**
Plot width in cm (default is 10).

CC_plot_height **numeric(1)**
Plot height in cm (default is 10).

RF_plot_width **numeric(1)**
Plot width in cm (default is 10).

RF_plot_height **numeric(1)**
Plot height in cm (default is 10).

base_size **numeric(1)**
Base size for the plot theme, default is 11.

RF_legend **logical(1)**
If TRUE, a legend will be added to the response factor plots.

plot_dpi **numeric(1)**
Plot resolution in dpi (default is 300).

verbose **logical(1)**
If FALSE, no messages will be printed.

... additional parameters for plotCalibraCurve

Value

List with the following elements:

- RES: List of CalibraCurve results (one item per substance, output from calc_single_curve).

- summary_tab: Data.frame with summary information about the calibration curves, one row per substance.
- plot_CC_list: List of ggplot2 objects of the calibration curves (one per substance for "single_plots", otherwise only one item).
- plot_RF_list: List of ggplot2 objects of the response factor plots (one item).

Examples

```
### NOTE: as output_path is not specified here, no files will be saved.
### Set output_path to a folder of your choice to save the results.

### single xlsx file:
data_path <- system.file("extdata", "MSQC1_xlsx", "GGPFSDSYR_QTRAP_y5.xlsx",
    package = "CalibraCurve")
D <- readDataTable(dataPath = data_path, concCol = 16, measCol = 12,
    fileType = "xlsx")
RES <- CalibraCurve(D_list = D)
RES$plot_CC_list
RES$plot_RF_list

### multiple xlsx files (in a folder) as multiplot:
data_folder <- system.file("extdata", "MSQC1_xlsx", package = "CalibraCurve")
D_list <- readMultipleTables(dataFolder = data_folder, fileType = "xlsx",
    concCol = 16, measCol = 12)
RES <- CalibraCurve(D_list = D_list, plot_type = "multiplot")
RES$plot_CC_list

### single rds file (SummarizedExperiment) as an all-in-one plot:
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
    package = "CalibraCurve")
D_list <- readDataSE(dataPath = file, concColName = "amount_fmol",
        substColName = "Substance", assayNumber = 1)
RES <- CalibraCurve(D_list, plot_type = "all_in_one")
RES$plot_CC_list
```

---

cleanData — *Clean data (remove 0s and NAs, remove concentration levels with insufficient number of replicates)*

Description

Clean data (remove 0s and NAs, remove concentration levels with insufficient number of replicates)

Usage

```
cleanData(rawData, minReplicates = 3)
```

Arguments

rawData	**data.frame** data set to be cleaned, result of readDataTable or readDataSE.
minReplicates	**integer(1)** Minimal number of replicates per concentration level. Concentration levels with too few data points will be removed.

Value

list of data.frames, each element contains data for a specific concentration level

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
    package = "CalibraCurve"
)
D_list <- readDataSE(file,
    concColName = "amount_fmol",
    substColName = "Substance", assayNumber = 1
)
cleanData(D_list[[1]])
```

---

plotCalibraCurve *Plot the calibration curve*

Description

Plot the calibration curve

Usage

```
plotCalibraCurve(
  CC_RES,
  ylab = "Intensity",
  xlab = "Concentration",
  show_regression_info = FALSE,
  show_linear_range = TRUE,
  show_data_points = TRUE,
  plot_type = "multiplot",
  point_colour = "black",
  curve_colour = "red",
  linear_range_colour = "black",
  multiplot_nrow = NULL,
  multiplot_ncol = NULL,
  multiplot_scales = "free",
  regression_info_size = 3,
  base_size = 11
)
```

Arguments

CC_RES	**list** Results of CalibraCurve. Each list element is the result of calc_single_curve
ylab	**character(1)** y-axis label.
xlab	**character(1)** x-axis label.
show_regression_info	**logical(1)** If TRUE, show regression information (R2, slope, intercept) on the plot.
show_linear_range	**logical(1)** If TRUE, show the linear range of the calibration curve as a rectangle in the plot.
show_data_points	**logical(1)** If TRUE, show the data points on the plot.
plot_type	**character(1)** Type of plot for calibration curves: "single_plots" (default, generate a separate plot for each substance), "multiplot" (generate a graphic with subplots for each substance) or "all_in_one" (generate a single plot with all substances).
point_colour	**character(1)** Colour of the data points, default is "black".
curve_colour	**character(1)** Colour of the calibration curve, default is "red".
linear_range_colour	**character(1)** Colour of the linear range background, default is "black" (colour is weakened by alpha = 0.1).
multiplot_nrow	**integer(1)** Number of rows for the multiplot layout (default is NULL, which means that there is only one row).
multiplot_ncol	**integer(1)** Number of columns for the multiplot layout (default is NULL, which means that the number of columns is determined automatically based on the number of curves).
multiplot_scales	**character(1)** Scales for the multiplot layout, default is "free" (which means that each plot has its own scales). Other options are "fixed", "free_x", "free_y".
regression_info_size	**numeric(1)** Size of the regression information text on the plot, default is 3.
base_size	**numeric(1)** Base size for the plot theme, default is 11.

Value

List of a ggplot2 object containing the calibration curve plot (CC_plot) and a data.frame containing summary data for annotations (annotation_dat).

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
    package = "CalibraCurve"
)
D_list <- readDataSE(file,
    concColName = "amount_fmol",
    substColName = "Substance", assayNumber = 1
)
CC_RES <- calc_single_curve(D_list[[1]], calcContinuousPrelimRanges = FALSE)

plotCalibraCurve(list(substance = CC_RES))
```

---

plotResponseFactors *Plot response factors*

Description

Plot response factors

Usage

```
plotResponseFactors(
  RES,
  RfThresL = 80,
  RfThresU = 120,
  ylab = "Response Factor",
  xlab = "Concentration",
  colour_threshold = "orange",
  colour_within = "#00BFC4",
  colour_outside = "#F8766D",
  legend = FALSE,
  base_size = 11,
  substance = NULL
)
```

Arguments

RES	**list** Results of calc_single_curve.
RfThresL	**numeric(1)** Lower threshold for response factor in percent (default is 80).

RfThresU **numeric(1)**
Upper threshold for response factor in percent (default is 120).

ylab **character(1)**
y-axis label.

xlab **character(1)**
x-axis label.

colour_threshold **character(1)**
Colour for horizontal threshold lines, default is "orange".

colour_within **character(1)**
Colour for points and lines within the threshold, default is "#00BFC4" (default ggplot colour).

colour_outside **character(1)**
Colour for horizontal outside of the threshold, default is "#F8766D" (default ggplot colour).

legend **logical(1)**
If TRUE, a legend is added to the plot, default is FALSE.

base_size **numeric(1)**
Base size for the plot theme, default is 11.

substance **character(1)**
Name of the substance (default is "substance1"). Will be added to the result files and will be used as labels in the plots.

Value

A ggplot2 object containing the response factor plot.

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
    package = "CalibraCurve"
)
D_list <- readDataSE(file,
    concColName = "amount_fmol",
    substColName = "Substance", assayNumber = 1
)
CC_RES <- calc_single_curve(D_list[[1]], calcContinuousPrelimRanges = FALSE)

plotResponseFactors(RES = CC_RES)
```

---

predictConcentration *Predict concentrations from intensity values based on a given calibration curve*

Description

Predict concentrations from intensity values based on a given calibration curve

Usage

```
predictConcentration(CC_res, newdata, verbose = TRUE)
```

Arguments

CC_res	**list** Results of CalibraCurve.
newdata	**numeric** A vector of intensity values for which to predict concentrations.
verbose	**logical(1)** If TRUE, a warning message is given if estimated concentrations are outside of the linear range.

Details

The function will give a warning if any of the predicted concentrations are outside the final linear range. This is important to ensure that the predictions are reliable and within the linear range of the calibration curve.

Value

A data frame with the following columns:

- intensity: The input intensity values.
- predicted_concentrations: The predicted concentrations based on the calibration curve.
- linear_range: A logical vector indicating whether the predicted concentrations are within the final linear range.

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
        package = "CalibraCurve")
D <- readDataSE(file, concColName = "amount_fmol",
        substColName = "Substance")
RES <- CalibraCurve(D)
newdata <- c(1000000, 10000000, 100000000) # 1e6, 1e7, 1e8
predictConcentration(RES$RES[[4]], newdata = newdata)
```

---

readDataSE *Read data stored as an SummarizedExperiment object (directly or stored in an .rds file). Extracts the two relevant columns (concentration and measurement) and orders the data by increasing concentration level.*

Description

Read data stored as an SummarizedExperiment object (directly or stored in an .rds file). Extracts the two relevant columns (concentration and measurement) and orders the data by increasing concentration level.

Usage

```
readDataSE(
  dataPath = NULL,
  rawDataSE = NULL,
  concColName,
  substColName,
  assayNumber = 1,
  rowNumbers = NULL
)
```

Arguments

`dataPath`	**character(1)** Path to the data file (.rds file)
`rawDataSE`	**SummarizedExperiment** SummarizedExperiment object
`concColName`	**character(1)** Name of the column in the colData() containing the concentration levels.
`substColName`	**character(1)** column name of rowData() containing the substance name (must be a unique value in each row)
`assayNumber`	**integer(1)** Number of assay to be extracted from the SummarizedExperiment object
`rowNumbers`	**integer** Row numbers to extract from the SummarizedExperiment object. Default is NULL, which means that all rows in the object will be used.

Details

The SummarizedExperiments object may contain quantitative values from targeted proteomics, lipidomics or metabolomics experiments. The colData has to contain a column with the concentration levels (concColName). The rowData has to contain a column with the substance names (e.g. peptide sequence, name of lipid or metabolite etc).

Value

List of data.frame, each with two numeric columns: Concentration and Measurement

Examples

```
file <- system.file("extdata", "MSQC1", "msqc1_dil_GGPFSDSYR.rds",
    package = "CalibraCurve")

D_list <- readDataSE(file,
    concColName = "amount_fmol",
    substColName = "Substance", assayNumber = 1)

# Alternative: import SummarizedExperiment object directly
rawDataSE <- readRDS(file)

D_list2 <- readDataSE(rawDataSE = rawDataSE,
    concColName = "amount_fmol",
    substColName = "Substance", assayNumber = 1
)
```

`readDataTable` *Read data in different table input formats (xlsx, csv or txt). Extracts the two relevant columns (concentration and measurement) and orders the data by increasing concentration.*

Description

Read data in different table input formats (xlsx, csv or txt). Extracts the two relevant columns (concentration and measurement) and orders the data by increasing concentration.

Usage

```
readDataTable(
  dataPath,
  fileType,
  concCol,
  measCol,
  substCol = NULL,
  sep = ",",
  dec = ".",
  header = TRUE,
  naStrings = c("NA", "NaN", "Filtered", "#NV"),
  sheet = 1
)
```

Arguments

`dataPath`	**character(1)** Path to the data file (.csv, .txt or .xlsx file).
`fileType`	**character(1)** Type of file: "csv", "txt" or "xlsx".

concCol	**integer(1)** Column number of the concentration values.
measCol	**integer** Column number of the concentration values.
substCol	**integer** Column number of the substance names (optional). If not set, all rows will be interpreted as belonging to the same substance.
sep	**character(1)** The field separator, default is ",".
dec	**character(1)** Decimal separator, default is ".".
header	**logical(1)** If TRUE, first line is counted as column names. The default is TRUE.
naStrings	**character** Vector of strings which are to be interpreted as NA. The default is c("NA", "NaN", "Filtered", "#NV").
sheet	**integer(1)** Sheet number (only needed for xlsx files, default is to use the first sheet).

Value

Data.frame with two numeric columns: Concentration and Measurement

Examples

```
### xlsx file:
file <- system.file("extdata", "MSQC1_xlsx", "GGPFSDSYR_QTRAP_y5.xlsx",
    package = "CalibraCurve"
)
D <- readDataTable(file, fileType = "xlsx", concCol = 16, measCol = 12)
```

---

readMultipleTables *Read folder of files in different table input formats (xlsx, csv or txt).*

Description

Read folder of files in different table input formats (xlsx, csv or txt).

Usage

```
readMultipleTables(dataFolder, fileType, concCol, measCol, ...)
```

Arguments

`dataFolder`	**character(1)** Folder containing either xlsx, csv or txt files
`fileType`	**character(1)** Type of file: "csv", "txt" or "xlsx".
`concCol`	**integer(1)** Column number of the concentration values.
`measCol`	**integer** Column number of the concentration values.
`...`	additional parameters to `readDataTable`

Value

List of data.frame, each with two numeric columns: Concentration and Measurement

Examples

```
data_folder <- system.file("extdata", "MSQC1_xlsx",
    package = "CalibraCurve")
D_list <- readMultipleTables(
    dataFolder = data_folder, fileType = "xlsx",
    concCol = 16, measCol = 12
)
```

Index