

Package ‘MSstatsResponse’

December 10, 2025

Type Package

Title Statistical Methods for Chemoproteomics Dose-Response Analysis

Version 1.1.0

Description Tools for detecting drug-protein interactions and estimating IC50 values from chemoproteomics data. Implements semi-parametric isotonic regression, bootstrapping, and curve fitting to evaluate compound effects on protein abundance.

URL <https://github.com/Vitek-Lab/MSstatsResponse>

BugReports <https://github.com/Vitek-Lab/MSstatsResponse/issues>

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.5.0)

LazyData false

RoxygenNote 7.3.3

Imports BiocParallel, ggplot2, dplyr, stats, parallel, data.table

Suggests MSstats, MSstatsTMT, tidyverse, boot, purrr, gridExtra, knitr, rmarkdown, BiocStyle, testthat

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

biocViews Proteomics, MassSpectrometry, StatisticalMethod, Software, Regression

git_url <https://git.bioconductor.org/packages/MSstatsResponse>

git_branch devel

git_last_commit 048d39d

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-12-09

Author Sarah Szvetecz [aut, cre],
Devon Kohler [aut],
Olga Vitek [aut]

Maintainer Sarah Szvetecz <szvetecz.s@northeastern.edu>

Contents

.extractTemplatesFromData	2
bootstrapIC50	3
bootstrapIC50LogScale	4
convertGroupToNumericDose	4
DIA_MSstats_Normalized	5
doseResponseFit	6
fitIsotonicRegression	8
futureExperimentSimulation	9
MSstatsPrepareDoseResponseFit	11
plotHitRateMSstatsResponse	13
plotIsotonic	14
predictIC50	15
predictIC50Parallel	17
simulateChemoProteinLevelNonParametric	18
visualizeResponseProtein	19
Index	22

.extractTemplatesFromData	<i>Helper function to extract template profiles from user data</i>
---------------------------	--

Description

Helper function to extract template profiles from user data

Usage

```
.extractTemplatesFromData(  
  data,  
  strong_proteins,  
  weak_proteins,  
  no_interaction_proteins,  
  drug_name,  
  concentrations  
)
```

Arguments

data	Prepared dose-response data
strong_proteins	Vector of protein IDs for strong responders
weak_proteins	Vector of protein IDs for weak responders
no_interaction_proteins	Vector of protein IDs for non-responders
drug_name	Drug to extract templates for
concentrations	Concentrations to include in template (in nM)

Value

List with template data for each interaction type

bootstrapIC50	<i>Bootstrap IC50 Estimates and Confidence Interval (ratio scale)</i>
---------------	---

Description

Bootstrap IC50 Estimates and Confidence Interval (ratio scale)

Usage

```
bootstrapIC50(  
  dose,  
  response,  
  n_samples = 1000,  
  alpha = 0.1,  
  increasing = FALSE,  
  target_response = 0.5,  
  transform_x = TRUE  
)
```

Arguments

- dose Numeric vector of dose values.
- response Numeric vector of response values (on log2 scale).
- n_samples Number of bootstrap samples (default = 1000).
- alpha Significance level for confidence interval (default = 0.10).
- increasing Logical. Fit non-decreasing if TRUE.
- target_response Numeric value for response level (default = 0.5).
- transform_x Logical. If TRUE, applies log10(dose + 1) transformation. Default = TRUE.

Value

List with mean IC50, CI bounds, and transformed estimates.

`bootstrapIC50LogScale` *Bootstrap IC50 Estimates and Confidence Interval (log scale)*

Description

Bootstrap IC50 Estimates and Confidence Interval (log scale)

Usage

```
bootstrapIC50LogScale(
  x,
  y,
  n_samples = 1000,
  alpha = 0.05,
  increasing = FALSE,
  target_response = 0.5
)
```

Arguments

<code>x</code>	Numeric vector of dose values.
<code>y</code>	Numeric vector of log2 response values.
<code>n_samples</code>	Number of bootstrap samples (default = 1000).
<code>alpha</code>	Significance level for confidence interval (default = 0.05).
<code>increasing</code>	Logical. Fit non-decreasing if TRUE.
<code>target_response</code>	Numeric value for response level (default = 0.5).

Value

List with mean IC50, CI bounds, and transformed estimates.

`convertGroupToNumericDose`
Convert MSstats GROUP labels to numeric dose in nM and extract drug name

Description

Convert MSstats GROUP labels to numeric dose in nM and extract drug name

Usage

```
convertGroupToNumericDose(group_vector)
```

Arguments

group_vector A character or factor vector with GROUP labels (e.g., "Dasatinib_003uM")

Value

A data frame with two columns: dose_nM (numeric), and drug (character).

Examples

```
# Example 1: Basic conversion with mixed units
groups <- c("DMSO", "Dasatinib_001uM", "Dasatinib_010uM",
           "Dasatinib_100nM", "Dasatinib_1000nM")
dose_info <- convertGroupToNumericDose(groups)
print(dose_info)

# Example 2: Handle multiple drugs
multi_drug_groups <- c("DMSO",
                      "Dasatinib_001uM", "Dasatinib_010uM",
                      "Imatinib_001uM", "Imatinib_010uM")
multi_dose_info <- convertGroupToNumericDose(multi_drug_groups)
print(multi_dose_info)

# Show unique drugs found
print(unique(multi_dose_info$drug))
```

DIA_MSstats_Normalized

Example pre-processed DIA-MS dataset

Description

This dataset contains normalized protein-level data from a DIA-MS chemoproteomics experiment, pre-processed using MSstats.

Usage

DIA_MSstats_Normalized

Format

A data frame with protein-level abundance values and associated MSstats metadata column names.

Details

It is used in the MSstatsResponse vignette to demonstrate data formatting and downstream dose–response analysis.

The dataset is formatted using the standard MSstats preprocessing workflow. For more information on preprocessing mass spectrometry–based proteomics experiments, see the vignettes for MSstats and/or MSstatsTMT.

Below is an example of how such data can be prepared:

```
# Read raw data (example with Spectronaut output)
raw_data <- readr::read_tsv("path/to/spectronaut_report.tsv")

# Convert to MSstats format
msstats_data <- MSstats::SpectronauttoMSstatsFormat(raw_data)

# Process data: normalization and protein summarization
processed_data <- MSstats::dataProcess(
  msstats_data,
  normalization = "equalizeMedians", # or FALSE for no normalization
  summaryMethod = "TMP",             # Tukey's median polish
  MBimpute = TRUE,                   # Impute missing values
  maxQuantileforCensored = 0.999
)
```

Examples

```
data("DIA_MSstats_Normalized")
head(DIA_MSstats_Normalized)
```

doseResponseFit	<i>Drug-protein interaction detection tested by F-test (fitted curve vs average response)</i>
-----------------	---

Description

Fits an isotonic regression model to protein abundance data. Performs an F-test to assess the significance of the dose-response curve and applies FDR correction.

Usage

```
doseResponseFit(
  data,
  weights = NULL,
  increasing = FALSE,
  transform_dose = TRUE,
  ratio_response = FALSE
)
```

Arguments

data Protein-level data, formatted with MSstatsPreparedoseResponseFit().

weights Optional numeric vector of weights. Defaults to equal weights.

increasing Logical. If TRUE, fits a non-decreasing model. If FALSE, fits non-increasing.

transform_dose Logical. If TRUE, applies $\log_{10}(\text{dose} + 1)$ transformation. Default is TRUE.

ratio_response Logical. If TRUE, converts $\log_2$ abundance to ratios relative to DMSO. Default is FALSE.

Value

A data frame with protein-wise F-test results and BH-adjusted p-values.

Examples

```
# Load example data
data_path <- system.file("extdata", "DIA_MSstats_Normalized.RDS",
                          package = "MSstatsResponse")
dia_data <- readRDS(data_path)

# Convert GROUP to dose
dose_info <- convertGroupToNumericDose(dia_data$ProteinLevelData$GROUP)
dia_data$ProteinLevelData$dose <- dose_info$dose_nM * 1e-9
dia_data$ProteinLevelData$drug <- dose_info$drug

# Prepare data for analysis
prepared_data <- MSstatsPrepareDoseResponseFit(
  dia_data$ProteinLevelData,
  dose_column = "dose",
  drug_column = "drug",
  protein_column = "Protein",
  log_abundance_column = "LogIntensities"
)

# Subset for quick example
example_data <- prepared_data[prepared_data$protein %in%
                              unique(prepared_data$protein)[1:5], ]

# Example 1: Basic interaction detection on log2 scale
interaction_results <- doseResponseFit(
  data = example_data,
  increasing = FALSE,
  transform_dose = TRUE,
  ratio_response = FALSE
)

# View results
print(interaction_results)

# Check significant interactions
significant <- interaction_results[interaction_results$adjust_pval < 0.05, ]
```

```

print(paste("Found", nrow(significant), "significant interactions"))

## Not run:
# Example 2: Full dataset analysis
full_results <- doseResponseFit(
  data = prepared_data,
  increasing = FALSE,
  transform_dose = TRUE,
  ratio_response = FALSE
)

## End(Not run)

```

`fitIsotonicRegression` *Fit Isotonic Regression Model*

Description

Fits an isotonic regression model to protein intensity data with log-transformed drug doses. Optionally performs an F-test to assess the significance of the dose-response curve.

Usage

```

fitIsotonicRegression(
  x,
  y,
  w = rep(1, length(y)),
  increasing = FALSE,
  transform_x = TRUE,
  ratio_y = FALSE,
  test_significance = FALSE
)

```

Arguments

<code>x</code>	Numeric vector of dose values.
<code>y</code>	Numeric vector of response values.
<code>w</code>	Optional numeric vector of weights. Defaults to equal weights.
<code>increasing</code>	Logical. If TRUE, fits a non-decreasing model. If FALSE, fits non-increasing.
<code>transform_x</code>	Logical. If TRUE, applies $\log_{10}(x + 1)$ transformation. Default is TRUE.
<code>ratio_y</code>	Logical. If TRUE, converts $\log_2$ abundance to ratios relative to DMSO. Default is FALSE.
<code>test_significance</code>	Logical. If TRUE, performs F-test to assess significance.

Value

A list representing the isotonic regression model (class = "isotonic_model").

futureExperimentSimulation

Test future experimental design using simulated data with user-defined or default templates

Description

Test future experimental design using simulated data with user-defined or default templates

Usage

```
futureExperimentSimulation(
  N_proteins = 300,
  N_rep = 3,
  N_Control_Rep = NULL,
  Concentrations = c(0, 1, 3, 10, 30, 100, 300, 1000, 3000),
  IC50_Prediction = FALSE,
  data = NULL,
  strong_proteins = NULL,
  weak_proteins = NULL,
  no_interaction_proteins = NULL,
  drug_name = NULL
)
```

Arguments

N_proteins	Number of proteins to simulate. Default = 300.
N_rep	Number of replicates for each drug concentration. Default = 3.
N_Control_Rep	Number of control replicates. If NULL, uses N_rep.
Concentrations	Numeric vector of drug concentrations (in nM scale). Default = c(0, 1, 3, 10, 30, 100, 300, 1000, 3000).
IC50_Prediction	Logical. If TRUE, perform IC50 prediction. Default = FALSE.
data	Optional. User's prepared dose-response data (e.g., from MSstatsPrepareDoseResponseFit). If provided, will extract templates from this data instead of using defaults.
strong_proteins	Character vector of protein IDs to use as strong interaction templates. Only used if data is provided.
weak_proteins	Character vector of protein IDs to use as weak interaction templates. Only used if data is provided.

no_interaction_proteins Character vector of protein IDs to use as no interaction templates. Only used if data is provided.

drug_name Character. Name of drug to extract templates for. Default = first non-DMSO drug in data.

Value

A list containing simulated data, MSstats formatted data, dose-response fit results, hit rate plots, and optionally IC50 predictions.

Examples

```
# Example 1: Quick simulation with default templates (small scale for speed)
sim_results <- futureExperimentSimulation(
  N_proteins = 50, # Small number for quick example
  N_rep = 2,
  N_Control_Rep = 3,
  Concentrations = c(0, 10, 100, 1000), # Fewer doses for speed
  IC50_Prediction = FALSE
)

# View hit rates
print(sim_results$Hit_Rates_Data)

# Check simulation results
print(paste("Simulated", nrow(sim_results$Simulated_Data), "data points"))

## Not run:
# Example 2: Full simulation with standard parameters
full_sim <- futureExperimentSimulation(
  N_proteins = 3000,
  N_rep = 3,
  N_Control_Rep = 6,
  Concentrations = c(0, 1, 3, 10, 30, 100, 300, 1000, 3000),
  IC50_Prediction = TRUE
)

# Display power analysis plot
print(full_sim$Hit_Rates_Plot)

# Example 3: Using custom templates from your own data
# Load and prepare your data
data_path <- system.file("extdata", "DIA_MSstats_Normalized.RDS",
  package = "MSstatsResponse")
dia_data <- readRDS(data_path)

dose_info <- convertGroupToNumericDose(dia_data$ProteinLevelData$GROUP)
dia_data$ProteinLevelData$dose <- dose_info$dose_nM * 1e-9
dia_data$ProteinLevelData$drug <- dose_info$drug

prepared_data <- MSstatsPrepareDoseResponseFit(
```

```

    dia_data$ProteinLevelData,
    dose_column = "dose",
    drug_column = "drug",
    protein_column = "Protein",
    log_abundance_column = "LogIntensities"
  )

# Run simulation with custom templates
custom_sim <- futureExperimentSimulation(
  N_proteins = 1000,
  N_rep = 3,
  data = prepared_data,
  strong_proteins = c("PROTEIN_A"),
  weak_proteins = c("PROTEIN_B"),
  no_interaction_proteins = c("PROTEIN_C"),
  drug_name = "Drug1",
  Concentrations = c(0, 1, 10, 100, 1000, 3000)
)

## End(Not run)

```

MSstatsPrepareDoseResponseFit

Prepare data for dose-response fitting with isotonic regression

Description

Prepare data for dose-response fitting with isotonic regression

Usage

```

MSstatsPrepareDoseResponseFit(
  data,
  dose_column = "dose",
  drug_column = "drug",
  protein_column = "Protein",
  log_abundance_column = "LogIntensities",
  transform_nM_to_M = NULL
)

```

Arguments

<code>data</code>	A data.frame (e.g. <code>data\$ProteinLevelData</code> from MSstats)
<code>dose_column</code>	Name of the column containing dose values (e.g., "dose")
<code>drug_column</code>	Name of column containing treatment name (e.g. drug name)
<code>protein_column</code>	Name of the column containing protein identifiers (e.g., "Protein")

`log_abundance_column`
 Name of the column with log-transformed abundance values (e.g., "LogIntensities")

`transform_nM_to_M`
 Logical. If TRUE, converts dose values from nanomolar (nM) to molar (M) by multiplying by 10^{-9} . Use when `dose_column` contains nM values but analysis requires M units. Default is NULL (no transformation applied).

Value

A standardized data.frame with columns: dose, response, protein

Examples

```
# Load example data
data_path <- system.file("extdata", "DIA_MSstats_Normalized.RDS",
  package = "MSstatsResponse")
dia_data <- readRDS(data_path)

# Example 1: Basic data preparation with dose already in M
# First add dose column if using GROUP labels
dose_info <- convertGroupToNumericDose(dia_data$ProteinLevelData$GROUP)
dia_data$ProteinLevelData$dose <- dose_info$dose_nM * 1e-9 # Convert to M
dia_data$ProteinLevelData$drug <- dose_info$drug

prepared_data <- MSstatsPrepareDoseResponseFit(
  data = dia_data$ProteinLevelData,
  dose_column = "dose",
  drug_column = "drug",
  protein_column = "Protein",
  log_abundance_column = "LogIntensities"
)

# Check structure
str(prepared_data)
head(prepared_data)

# Example 2: Convert dose from nM to M during preparation
dia_data$ProteinLevelData$dose_nM <- dose_info$dose_nM # Keep in nM

prepared_data_converted <- MSstatsPrepareDoseResponseFit(
  data = dia_data$ProteinLevelData,
  dose_column = "dose_nM",
  drug_column = "drug",
  protein_column = "Protein",
  log_abundance_column = "LogIntensities",
  transform_nM_to_M = TRUE # Convert nM to M
)

# Verify conversion
print(unique(prepared_data_converted$dose))
```

```
## Not run:
# Example 3: Working with custom column names
custom_data <- data.frame(
  ProteinID = rep(c("P1", "P2"), each = 10),
  Treatment = rep(c("DMSO", "Drug1"), 10),
  Concentration = rep(c(0, 1, 10, 100, 1000), 4),
  Log2Abundance = rnorm(20, mean = 20, sd = 1)
)

prepared_custom <- MSstatsPrepareDoseResponseFit(
  data = custom_data,
  dose_column = "Concentration",
  drug_column = "Treatment",
  protein_column = "ProteinID",
  log_abundance_column = "Log2Abundance",
  transform_nM_to_M = TRUE
)

## End(Not run)
```

plotHitRateMSstatsResponse

Plot hit rates by category

Description

Plot hit rates by category

Usage

```
plotHitRateMSstatsResponse(results, rep_count, concentration_count)
```

Arguments

results	Output of interaction test from doseResponseFit()
rep_count	Number of replicates per concentration in simulation
concentration_count	Number of concentrations in simulation

Value

A list containing the plot and plot data

plotIsotonic

Plot Isotonic Regression Model

Description

Plot Isotonic Regression Model

Usage

```
plotIsotonic(
  fit,
  ratio = TRUE,
  show_ic50 = FALSE,
  drug_name = NULL,
  protein_name = NULL,
  x_lab = expression(Log[10] ~ "[drug (M)]"),
  y_lab = "Log2 Intensity",
  title = NULL,
  ci = NULL,
  legend = FALSE,
  theme_style = "classic",
  original_label = FALSE
)
```

Arguments

fit	A model object returned by fitIsotonicRegression().
ratio	Logical. If TRUE, shows plot on the ratio scale relative to DMSO (i.e. 0-1 scale). Default is FALSE.
show_ic50	Logical. If TRUE, adds vertical line and annotation for IC50.
drug_name	Drug name for plotting data.
protein_name	Protein name for plot.
x_lab	Label for x-axis.
y_lab	Label for y-axis.
title	Title for the plot.
ci	Logical. Include IC50 95% confidence interval bands if TRUE. Default is FALSE.
legend	Logical. Show legend if TRUE.
theme_style	ggplot2 theme name to apply (default = "classic").
original_label	Logical. If TRUE, replace x-axis tick labels with original dose labels.

Value

A ggplot object.

predictIC50	<i>Predict IC50 (dose where response = target) for each protein and drug</i>
-------------	--

Description

Predict IC50 (dose where response = target) for each protein and drug

Usage

```
predictIC50(
  data,
  n_samples = 1000,
  alpha = 0.1,
  increasing = FALSE,
  transform_dose = TRUE,
  ratio_response = TRUE,
  bootstrap = TRUE,
  BPPARAM = bpparam(),
  target_response = 0.5
)
```

Arguments

<code>data</code>	A data frame with columns: protein, drug, dose, response.
<code>n_samples</code>	Number of bootstrap samples. Default = 1000.
<code>alpha</code>	Confidence level. Default = 0.10.
<code>increasing</code>	Logical. If TRUE, fit a non-decreasing trend. Default = FALSE.
<code>transform_dose</code>	Logical. If TRUE, applies log10(dose + 1) transformation. Default = TRUE.
<code>ratio_response</code>	Logical. If TRUE, use ratio response; else use log2 scale. Default = TRUE.
<code>bootstrap</code>	Logical. If FALSE, skip confidence interval bootstrap estimation and only return IC50. Default = TRUE.
<code>BPPARAM</code>	A BiocParallelParam for parallel processing. The recommended usage is to <i>register</i> a backend once (e.g., <code>register(MulticoreParam(workers=4))</code> on Linux/macOS or <code>register(SnowParam(workers=4, type="SOCK"))</code> on Windows) and pass <code>BPPARAM = bpparam()</code> . Default <code>bpparam()</code> .
<code>target_response</code>	Numeric, the response fraction (e.g., 0.5, 0.25, 0.75). Default = 0.5.

Value

A data frame with columns: protein, drug, IC50, IC50_lower_bound, IC50_upper_bound.

Examples

```

# Load example data
data_path <- system.file("extdata", "DIA_MSstats_Normalized.RDS",
                          package = "MSstatsResponse")
dia_data <- readRDS(data_path)

# Convert GROUP to dose
dose_info <- convertGroupToNumericDose(dia_data$ProteinLevelData$GROUP)
dia_data$ProteinLevelData$dose <- dose_info$dose_nM * 1e-9
dia_data$ProteinLevelData$drug <- dose_info$drug

# Prepare data for analysis
prepared_data <- MSstatsPrepareDoseResponseFit(
  dia_data$ProteinLevelData,
  dose_column = "dose",
  drug_column = "drug",
  protein_column = "Protein",
  log_abundance_column = "LogIntensities"
)

# Subset to fewer proteins for example
example_data <- prepared_data[prepared_data$protein %in%
                              unique(prepared_data$protein)[1:3], ]

# Example 1: Quick IC50 estimation without bootstrap (fast)
ic50_quick <- predictIC50(
  data = example_data,
  bootstrap = FALSE
)
print(ic50_quick)

## Not run:
# Example 2: Full IC50 estimation with bootstrap confidence intervals
ic50_results <- predictIC50(
  data = prepared_data,
  n_samples = 1000,
  alpha = 0.10,
  ratio_response = TRUE,
  bootstrap = TRUE
)

# Example 3: Parallel processing for large datasets
library(BiocParallel)
ic50_parallel <- predictIC50(
  data = prepared_data,
  n_samples = 1000,
  BPPARAM = bpparam(),
  bootstrap = TRUE
)

# Example 4: IC50 at different response levels (IC25, IC75)

```



```
ic25_results <- predictIC50(  
  data = prepared_data,  
  target_response = 0.25,  
  bootstrap = TRUE  
)  
  
## End(Not run)
```

predictIC50Parallel *Parallel version of predictIC50 function*

Description

Runs predictIC50 on the entire dataset in parallel across proteins.

Usage

```
predictIC50Parallel(  
  data,  
  n_samples = 1000,  
  alpha = 0.1,  
  increasing = FALSE,  
  transform_dose = TRUE,  
  ratio_response = TRUE,  
  bootstrap = TRUE,  
  numberOfCores = 2  
)
```

Arguments

data	A data frame with columns: protein, drug, dose, response.
n_samples	Number of bootstrap samples. Default = 1000.
alpha	Confidence level. Default = 0.10.
increasing	Logical. If TRUE, fit non-decreasing trend. Default = FALSE.
transform_dose	Logical. If TRUE, applies log10(dose + 1) transformation. Default = TRUE.
ratio_response	Logical. If TRUE, use ratio response; else use log2 scale. Default = TRUE.
bootstrap	Logical. If TRUE, compute bootstrap CIs. Default = TRUE.
numberOfCores	Number of cores for parallel processing. Default = 2.

Value

A data frame with columns: protein, drug, IC50, lower CI, upper CI.

Examples

```

# Load example data
data_path <- system.file("extdata", "DIA_MSstats_Normalized.RDS",
                          package = "MSstatsResponse")
dia_data <- readRDS(data_path)

# Convert GROUP to dose
dose_info <- convertGroupToNumericDose(dia_data$ProteinLevelData$GROUP)
dia_data$ProteinLevelData$dose <- dose_info$dose_nM * 1e-9
dia_data$ProteinLevelData$drug <- dose_info$drug

# Prepare data for analysis
prepared_data <- MSstatsPrepareDoseResponseFit(
  dia_data$ProteinLevelData,
  dose_column = "dose",
  drug_column = "drug",
  protein_column = "Protein",
  log_abundance_column = "LogIntensities"
)

# Subset for quick example
example_data <- prepared_data[prepared_data$protein %in%
                              unique(prepared_data$protein)[1:5], ]

# Example 1: Quick parallel IC50 without bootstrap (2 cores)
ic50_quick_parallel <- predictIC50Parallel(
  data = example_data,
  bootstrap = FALSE,
  numberOfCores = 2
)
print(ic50_quick_parallel)

```

```
simulateChemoProteinLevelNonParametric
```

Simulate chemoproteomics data at the protein level - non-parametric approach

Description

Simulate chemoproteomics data at the protein level - non-parametric approach

Usage

```

simulateChemoProteinLevelNonParametric(
  N_proteins = 3000,
  TP = 0.333,
  TW = 0.333,
  TN = 0.333,

```

```

concentrations = c(0, 1, 3, 10, 30, 100, 300, 1000, 3000),
rep = 3,
seed = NULL,
var_tech = 0.4,
control_rep = NULL,
template = list(strong_interaction = data.frame(dose = c(), LogIntensities = c()),
  weak_interaction = data.frame(dose = c(), LogIntensities = c()), no_interaction =
    data.frame(dose = c(), LogIntensities = c())),
outlier_prob = 0.05
)

```

Arguments

N_proteins	Number of proteins in simulation. Default = 3000.
TP	Proportion of strong interacting proteins. Default = 0.333.
TW	Proportion of weak interacting proteins. Default = 0.333.
TN	Proportion of non-interacting proteins. Default = 0.333.
concentrations	Numeric vector of drug concentrations in simulation experiment. Default = c(0, 1, 3, 10, 30, 100, 300, 1000, 3000).
rep	Number of replicates for each drug concentration. Default = 3.
seed	Simulation seed for reproducibility. Default = 3.
var_tech	Combined technical and biological variation. Default = 0.4.
control_rep	Number of control replicates. If NULL, uses rep.
template	List containing real protein level data representing different levels of interactions.
outlier_prob	Probability of sample outlier. Default = 0.05.

Value

A data.frame with simulated chemoproteomics data.

visualizeResponseProtein

Plot isotonic regression fit with optional IC50 for a single protein and drug

Description

Plot isotonic regression fit with optional IC50 for a single protein and drug

Usage

```
visualizeResponseProtein(
  data,
  protein_name,
  drug_name,
  ratio_response = TRUE,
  transform_dose = TRUE,
  show_ic50 = TRUE,
  add_ci = FALSE,
  n_samples = 1000,
  alpha = 0.1,
  increasing = FALSE,
  y_lab = "Ratio Response"
)
```

Arguments

<code>data</code>	Protein-level dataset (e.g., output of <code>MSstatsPrepareDoseResponseFit</code>).
<code>protein_name</code>	Character. Protein name to plot.
<code>drug_name</code>	Character. Drug name to plot.
<code>ratio_response</code>	Logical. If TRUE, compute IC50 on ratio scale; if FALSE, use log2 intensities.
<code>transform_dose</code>	Logical. If TRUE, applies $\log_{10}(\text{dose} + 1)$. Default is TRUE.
<code>show_ic50</code>	Logical. If TRUE, adds vertical line and annotation for IC50.
<code>add_ci</code>	Logical. Include IC50 95% confidence interval bands if TRUE. Default is FALSE.
<code>n_samples</code>	Number of bootstrap samples if including confidence intervals. Default is 1000.
<code>alpha</code>	Alpha level for confidence intervals. Default is 0.05.
<code>increasing</code>	Logical. If TRUE, fits a non-decreasing model. If FALSE, fits non-increasing.
<code>y_lab</code>	Character. Label for the y-axis. Default is "Ratio Response".

Value

A ggplot object.

Examples

```
# Load example data
data_path <- system.file("extdata", "DIA_MSstats_Normalized.RDS",
  package = "MSstatsResponse")
dia_data <- readRDS(data_path)

# Convert GROUP to dose
dose_info <- convertGroupToNumericDose(dia_data$ProteinLevelData$GROUP)
dia_data$ProteinLevelData$dose <- dose_info$dose_nM * 1e-9
dia_data$ProteinLevelData$drug <- dose_info$drug

# Prepare data for analysis
```

```
prepared_data <- MSstatsPrepareDoseResponseFit(  
  dia_data$ProteinLevelData,  
  dose_column = "dose",  
  drug_column = "drug",  
  protein_column = "Protein",  
  log_abundance_column = "LogIntensities"  
)  
  
# Example 1: Basic dose-response visualization  
plot1 <- visualizeResponseProtein(  
  data = prepared_data,  
  protein_name = "PROTEIN_A",  
  drug_name = "Drug1",  
  ratio_response = TRUE,  
  show_ic50 = FALSE,  
  add_ci = FALSE  
)  
print(plot1)  
  
# Example 2: Add IC50 annotation  
plot2 <- visualizeResponseProtein(  
  data = prepared_data,  
  protein_name = "PROTEIN_A",  
  drug_name = "Drug1",  
  ratio_response = TRUE,  
  show_ic50 = TRUE,  
  add_ci = FALSE  
)  
print(plot2)
```

Index

* **datasets**

- DIA_MSstats_Normalized, [5](#)
- .extractTemplatesFromData, [2](#)

- bootstrapIC50, [3](#)
- bootstrapIC50LogScale, [4](#)

- convertGroupToNumericDose, [4](#)

- DIA_MSstats_Normalized, [5](#)
- doseResponseFit, [6](#)

- fitIsotonicRegression, [8](#)
- futureExperimentSimulation, [9](#)

- MSstatsPrepareDoseResponseFit, [11](#)

- plotHitRateMSstatsResponse, [13](#)
- plotIsotonic, [14](#)
- predictIC50, [15](#)
- predictIC50Parallel, [17](#)

- simulateChemoProteinLevelNonParametric,
[18](#)

- visualizeResponseProtein, [19](#)