Package 'MSstatsResponse'

May 20, 2026

Type Package

Title Statistical Methods for Chemoproteomics Dose-Response Analysis

Version 1.3.0

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, tidyr, stringr, plotly, utils

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 ba27038

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-19

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

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

Contents

.extractTemplatesFromData . . . 2
bootstrapIC50 . . . 3
bootstrapIC50LogScale . . . 4
bootstrapIC50Precalculated . . . 4
calculatePeptideWeights . . . 5
calculateTurnoverRatios . . . 7
convertGroupToNumericDose . . . 8
DIA_MSstats_Normalized . . . 9
doseResponseFit . . . 10
fitIsotonicRegression . . . 12
futureExperimentSimulation . . . 13
MSstatsPrepareDoseResponseFit . . . 15
parse_timepoint . . . 17
plotHitRateMSstatsResponse . . . 18
plotIsotonic . . . 18
plot_tpr_power_curve . . . 19
predictIC50 . . . 20
predictIC50Parallel . . . 23
run_tpr_simulation . . . 24
simulateChemoProteinLevelNonParametric . . . 26
visualizeResponseProtein . . . 27

Index **30**

`.extractTemplatesFromData` *Helper function to extract template profiles from user data*

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` *Bootstrap IC50 Estimates and Confidence Interval (ratio scale)*

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,
  weights = NULL
)
```

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(x + 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,
  transform_x = TRUE,
  weights = NULL
)
```

Arguments

x	Numeric vector of dose values.
y	Numeric vector of log2 response values.
n_samples	Number of bootstrap samples (default = 1000).
alpha	Significance level for confidence interval (default = 0.05).
increasing	Logical. Fit non-decreasing if TRUE.
target_response	Numeric value for response level (default = 0.5).
transform_x	Logical. If TRUE, applies log10(x + 1) transformation. Default = TRUE.

Value

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

bootstrapIC50Precalculated *Bootstrap IC50 with pre-calculated ratios*

Description

Bootstrap IC50 with pre-calculated ratios

Usage

```
bootstrapIC50Precalculated(
  dose,
  response,
  n_samples = 1000,
  alpha = 0.1,
  increasing = FALSE,
  target_response = 0.5,
  transform_x = TRUE,
  weights = NULL
)
```

Arguments

dose	Numeric vector of dose values.
response	Numeric vector of response values (pre-calculated ratios).
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.
transform_x	Logical. If TRUE, applies log10(x + 1) transformation. Default = TRUE.

Value

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

---

calculatePeptideWeights

Calculate quality-based weights for peptide measurements

Description

Calculates weights based on coverage, signal intensity, monotonicity, and data validity. Designed for protein turnover data but applicable to any dose/time-response data.

Usage

```
calculatePeptideWeights(
  data,
  protein_col = "Protein",
  peptide_col = "BaseSequence",
  time_col = "TimeVal",
  response_col = "H_frac",
  light_intensity_col = "Light",
  validity_threshold = 1.3,
  top_n_peptides = NULL
)
```

Arguments

data	Data frame with peptide-level measurements (output from calculateTurnoverRatios)
protein_col	Character. Column containing protein identifiers. Default = "Protein"
peptide_col	Character. Column containing peptide identifiers. Default = "BaseSequence"
time_col	Character. Column containing timepoint values. Default = "TimeVal"
response_col	Character. Column containing response values (e.g., H_frac). Default = "H_frac"
light_intensity_col	Character. Column containing light channel intensity. Default = "Light"
validity_threshold	Numeric. Maximum allowed response value. Default = 1.3
top_n_peptides	Numeric. Top n peptides based on median light channel intensity. If NULL, no filtering (all get weight 1).

Value

Input data frame with added columns:

- n_obs: Number of observations for this peptide
- coverage_score: Proportion of timepoints observed
- light_intensity_score: Normalized median light intensity (per protein)
- monotonicity_score: Kendall correlation (time vs response), 0 if decreasing
- validity_flag: 0 if any invalid values, 1 otherwise
- weight: Combined quality weight (product of all components)

Examples

```
## Not run:
# Calculate ratios first
ratios <- calculateTurnoverRatios(feature_data)

# Add quality weights
ratios_weighted <- calculatePeptideWeights(ratios)

# Inspect weights
ratios_weighted %>%
  group_by(Protein, BaseSequence) %>%
  slice(1) %>%
  select(Protein, BaseSequence, coverage_score, monotonicity_score, weight)

# Use with doseResponseFit
result <- doseResponseFit(
  data = ratios_weighted,
  weights = ratios_weighted$weight,
  increasing = TRUE
)

# Use stricter validity threshold
ratios_weighted_strict <- calculatePeptideWeights(ratios, validity_threshold = 1.0)

## End(Not run)
```

`calculateTurnoverRatios` *Calculate turnover ratios from MSstats FeatureLevelData*

Description

Calculate turnover ratios from MSstats FeatureLevelData

Usage

```
calculateTurnoverRatios(
  feature_data,
  channel_col = "LABEL",
  heavy_label = "H",
  light_label = "L",
  time_col = "GROUP",
  peptide_col = "PEPTIDE",
  protein_col = "PROTEIN",
  intensity_col = "INTENSITY",
  run_col = "RUN",
  peptide_selector = NULL,
  agg_function = max,
  normalize_tracer = FALSE,
  tracer_constants = NULL
)
```

Arguments

`feature_data`	Data frame from MSstats dataProcess()$FeatureLevelData
`channel_col`	Character. Name of column containing Heavy/Light labels. Default = "LABEL"
`heavy_label`	Character. Value in channel_col indicating heavy channel. Default = "H"
`light_label`	Character. Value in channel_col indicating light channel. Default = "L"
`time_col`	Character. Column containing timepoint information. Default = "GROUP"
`peptide_col`	Character. Column containing peptide sequences. Default = "PEPTIDE"
`protein_col`	Character. Column containing protein identifiers. Default = "PROTEIN"
`intensity_col`	Character. Column with intensity values. Default = "INTENSITY"
`run_col`	Character. Column identifying technical replicates. Default = "RUN"
`peptide_selector`	Optional function to select peptides. Function should take a data frame (grouped by protein) and return filtered data frame. Default = NULL (use all peptides).
`agg_function`	Function to aggregate duplicate peptide measurements (multiple charge states, transitions, etc.). Default = max (takes highest signal).
`normalize_tracer`	Logical. If TRUE, normalize by tracer incorporation. Default = FALSE
`tracer_constants`	Named numeric vector. Tracer constants for each timepoint. Required if normalize_tracer = TRUE

Value

Data frame with columns: Protein, BaseSequence, TimeVal, Run, Heavy, Light, Total, H_frac, L_frac

Examples

```
## Not run:
# Basic usage - all proteins, all peptides
ratios <- calculateTurnoverRatios(
  feature_data = quant_data$FeatureLevelData
)

# With peptide selection (top 10 by signal per protein)
top10_selector <- function(df) {
  top_peptides <- df %>%
    group_by(BaseSequence) %>%
    summarise(total_signal = sum(Intensity, na.rm = TRUE), .groups = "drop") %>%
    arrange(desc(total_signal)) %>%
    slice_head(n = 10) %>%
    pull(BaseSequence)

  df %>% filter(BaseSequence %in% top_peptides)
}

ratios_top10 <- calculateTurnoverRatios(
  feature_data = quant_data$FeatureLevelData,
  peptide_selector = top10_selector
)

# Use different aggregation function (e.g., median for robustness)
ratios_median <- calculateTurnoverRatios(
  feature_data = quant_data$FeatureLevelData,
  agg_function = median
)

# With tracer normalization
tracer_consts <- c("0hr" = 1.0, "1hr" = 0.95, "12hrs" = 0.85, "168hrs" = 0.75)
ratios_norm <- calculateTurnoverRatios(
  feature_data = quant_data$FeatureLevelData,
  normalize_tracer = TRUE,
  tracer_constants = tracer_consts
)

# Filter for specific protein after calculation
protein_ratios <- ratios %>% filter(Protein == "A0A2K5TXF6")

## End(Not run)
```

`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` *Drug-protein interaction detection tested by F-test (fitted curve vs average response)*

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,
  precalculated_ratios = FALSE
)
```

Arguments

data	Protein-level data, formatted with MSstatsPreparedoseResponseFit().
weights	Optional numeric vector of weights. Defaults to equal weights.
increasing	Logical or character. If TRUE, fits a non-decreasing model. If FALSE, fits non-increasing. If "both", fits both increasing and decreasing models and selects the one with better fit (lower SSE). Recommended for exploratory analysis, but ~2x slower. Default is FALSE.
transform_dose	Logical. If TRUE, applies log10(dose + 1) transformation. Default is TRUE.
ratio_response	Logical. If TRUE, converts log2 abundance to ratios relative to DMSO. Default is FALSE.
precalculated_ratios	Logical. If TRUE, assumes the response column contains pre-calculated ratios (not log2 values) and skips internal ratio calculation. This is useful for protein turnover data or other datasets where ratios are computed externally. 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
)

# Example 2: Fit both directions and select better
```

```
interaction_both <- doseResponseFit(
  data = example_data,
  increasing = "both",
  transform_dose = TRUE,
  ratio_response = FALSE
)

# Check which direction was selected for each protein
table(interaction_both$direction)

# Example 3: Using pre-calculated ratios (e.g., from protein turnover)
# Assume you have pre-calculated ratio data
turnover_data <- example_data
turnover_data$response <- 2^turnover_data$response /
                          mean(2^turnover_data$response[turnover_data$dose == 0])

interaction_results_precalc <- doseResponseFit(
  data = turnover_data,
  increasing = FALSE,
  transform_dose = TRUE,
  ratio_response = FALSE,
  precalculated_ratios = TRUE
)

## Not run:
# Example 4: 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,
  precalculated_ratios = FALSE,
```

```
  test_significance = FALSE
)
```

Arguments

x	Numeric vector of dose values.
y	Numeric vector of response values.
w	Optional numeric vector of weights. Defaults to equal weights.
increasing	Logical. If TRUE, fits a non-decreasing model. If FALSE, fits non-increasing.
transform_x	Logical. If TRUE, applies log10(x + 1) transformation. Default is TRUE.
ratio_y	Logical. If TRUE, converts log2 abundance to ratios relative to DMSO. Default is FALSE.
precalculated_ratios	Logical. If TRUE, assumes y contains pre-calculated ratios and skips internal ratio calculation. Default is FALSE.
test_significance	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

data	A data.frame (e.g. data$ProteinLevelData from MSstats)
dose_column	Name of the column containing dose values (e.g., "dose")
drug_column	Name of column containing treatment name (e.g. drug name)
protein_column	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)
```

`parse_timepoint` *Parse timepoint strings to numeric hours*

Description

Parse timepoint strings to numeric hours

Usage

```
parse_timepoint(time_strings)
```

Arguments

`time_strings`	Character vector of timepoint labels

Value

Numeric vector of hours

Examples

```
MSstatsResponse:::parse_timepoint(c("0hr", "1hr", "12hrs", "168hrs"))
# Returns: 0 1 12 168
```

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,
  weights = NULL,
  show_weights = FALSE,
  ratio = TRUE,
  precalculated_ratios = FALSE,
  show_ic50 = FALSE,
  target_response = 0.5,
  drug_name = NULL,
  protein_name = NULL,
  x_lab = NULL,
  y_lab = NULL,
  title = NULL,
  ci = NULL,
  legend = FALSE,
  theme_style = "classic",
  original_label = FALSE,
```

```
  transform_x = TRUE,
  color_by = NULL,
  original_data = NULL
)
```

Arguments

fit	A model object returned by fitIsotonicRegression().
weights	Numeric vector of weights (same length as data points). Default is NULL.
show_weights	Logical. If TRUE and weights provided, scale point size by weight. Default is FALSE.
ratio	Logical. If TRUE, shows plot on the ratio scale relative to DMSO. Default is FALSE.
precalculated_ratios	Logical. If TRUE, response values are pre-calculated ratios. Default is FALSE.
show_ic50	Logical. If TRUE, adds vertical line and annotation for IC50.
target_response	Numeric. Target response level for IC50/EC50. Default = 0.5.
drug_name	Drug name for plotting data.
protein_name	Protein name for plot.
x_lab	Character. Label for x-axis. If NULL, uses default based on transform_x.
y_lab	Character. Label for y-axis. If NULL, uses default based on ratio/precalculated_ratios.
title	Character. Title for the plot. If NULL, auto-generates.
ci	List with ci_lower and ci_upper. Include IC50 confidence interval bands if provided. Default is NULL.
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.
transform_x	Logical. Whether doses were log-transformed. Default = TRUE.
color_by	Character. Column name to color points by. Default is NULL.
original_data	Data frame containing original data with grouping variables.

Value

A ggplot object.

---

plot_tpr_power_curve *Visualize detection power across experimental designs*

Description

Creates an interactive plot showing how the true positive rate (detection power) changes with the number of doses and replicates. Only the results for the user-selected protein template (passed as the strong interaction category) are displayed. Line color shading goes from light (fewest replicates) to dark (most replicates).

Usage

```
plot_tpr_power_curve(simulation_results, static = FALSE)
```

Arguments

static	Logical. If TRUE, returns a static ggplot object suitable for PDF export. Default: FALSE.

Value

An interactive plotly object, or a static ggplot object if static = TRUE.

Examples

```
## Not run:
results <- run_tpr_simulation(
  rep_range = c(1, 3),
  concentrations = c(0, 10, 100, 1000),
  dose_range = c(2, 4),
  n_proteins = 300
)
plot_tpr_power_curve(results)

## End(Not run)
```

---

predictIC50 *Predict IC50 (dose where response = target) for each protein and drug*

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,
  precalculated_ratios = FALSE,
  bootstrap = TRUE,
  BPPARAM = bpparam(),
  target_response = NULL,
  weights = NULL
)
```

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 or character. If TRUE, fit a non-decreasing trend. If FALSE, fit non-increasing. If "both", fits both directions and selects the better fit. 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.
precalculated_ratios	Logical. If TRUE, assumes response column contains pre-calculated ratios. Default is FALSE.
bootstrap	Logical. If FALSE, skip confidence interval bootstrap estimation and only return IC50. Default = TRUE.
BPPARAM	A BiocParallelParam for parallel processing. Default bpparam().
target_response	Numeric, the response fraction (e.g., 0.5, 0.25, 0.75, 1.5). For decreasing curves: use values < 1 (e.g., 0.5 for IC50). For increasing curves: use values > 1 (e.g., 1.5 for 50% increase). Default = 0.5 (auto-adjusts to 1.5 for increasing curves).

Value

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

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)

# Example 2: Fit both increasing and decreasing curves
ic50_both <- predictIC50(
  data = example_data,
  increasing = "both",
  bootstrap = FALSE
)
print(ic50_both)

## Not run:
# Example 3: 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 4: Parallel processing for large datasets
library(BiocParallel)
ic50_parallel <- predictIC50(
  data = prepared_data,
  n_samples = 1000,
  BPPARAM = bpparam(),
  bootstrap = TRUE
)

# Example 5: IC50 at different response levels (IC25, IC75)
ic25_results <- predictIC50(
  data = prepared_data,
  target_response = 0.25,
  bootstrap = TRUE
)

# Example 6: Increasing curves with EC50 at 1.5x baseline
ec50_results <- predictIC50(
  data = prepared_data,
  increasing = TRUE,
  target_response = 1.5,
  bootstrap = TRUE
)

# Example 7: Using pre-calculated ratios
turnover_data <- prepared_data
turnover_data$response <- 2^turnover_data$response /
                          mean(2^turnover_data$response[turnover_data$dose == 0])

ic50_precalc <- predictIC50(
  data = turnover_data,
  precalculated_ratios = TRUE,
  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)
```

`run_tpr_simulation` *Simulate detection power across experimental design configurations*

Description

Evaluates how well a dose-response proteomics experiment can detect drug-protein interactions under different experimental designs. For each combination of replicate count and number of doses, the function simulates a full experiment and measures the true positive rate (TPR); the percentage of known interacting proteins that are correctly identified.

Usage

```
run_tpr_simulation(
  rep_range,
  concentrations,
  dose_range,
  data,
  protein,
  n_proteins = 1000
)
```

Arguments

`rep_range`	Integer vector of length 2, `c(min, max)`. The range of biological replicates per dose to evaluate. Each value in the sequence `min:max` produces one line in the resulting power curve.
`concentrations`	Numeric vector. The full set of drug concentrations available in the experiment (e.g., `c(0, 1, 3, 10, 30, 100, 300, 1000, 3000)`).
`dose_range`	Integer vector of length 2, `c(min, max)`. The range of dose counts to sweep. For example, `c(2, 9)` evaluates designs with 2 doses, 3 doses, ..., up to 9 doses.

data	User's prepared dose-response data (e.g., from MSstatsPrepareDoseResponseFit). Protein-specific templates are extracted from this data for the simulation.
protein	Character string specifying a protein ID to use as the interaction template for the power analysis.
n_proteins	Integer. Number of proteins to simulate per run. Larger values give more stable estimates but increase runtime. Default: 1000.

Details

This helps experimentalists answer practical questions like: "If I can only afford 3 replicates per dose, how many dose levels do I need to reliably detect weak interactions?"

Concentration subsets are selected automatically from the user's available doses: control (0) and the highest dose are always included, and intermediate doses are added by choosing the one farthest from the log-median of the already-selected set.

Value

A data.frame with columns:

Interaction Character. "Strong" or "Weak".

TPR Numeric. True positive rate as a percentage (0-100).

N_rep Integer. Number of replicates used in this simulation.

NumConcs Integer. Number of concentrations used in this simulation.

Examples

```
## Not run:
# Prepare user data
mock_data <- data.frame(
  protein = rep("P1", 6),
  drug = c("DMSO", "DMSO", "Drug1", "Drug1", "Drug1", "Drug1"),
  dose = c(0, 0, 10, 100, 1000, 1000),
  response = c(20, 20, 18, 15, 12, 12)
)

# Run power analysis
results <- run_tpr_simulation(
  rep_range = c(1, 3),
  concentrations = c(0, 10, 100, 1000),
  dose_range = c(2, 4),
  data = mock_data,
  protein = "P1",
  n_proteins = 300
)

# Visualize results
plot_tpr_power_curve(results)

## End(Not run)
```

`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,
  weights = NULL,
  show_weights = FALSE,
  ratio_response = TRUE,
  precalculated_ratios = FALSE,
  transform_dose = TRUE,
  show_ic50 = TRUE,
  target_response = 0.5,
  add_ci = FALSE,
  n_samples = 1000,
  alpha = 0.1,
  increasing = FALSE,
  color_by = NULL,
  x_lab = NULL,
  y_lab = NULL,
  title = NULL
)
```

Arguments

`data`	Protein-level dataset (e.g., output of MSstatsPrepareDoseResponseFit).
`protein_name`	Character. Protein name to plot.
`drug_name`	Character. Drug name to plot.
`weights`	Optional numeric vector of weights matching nrow(data). Default is NULL (equal weights).
`show_weights`	Logical. If TRUE and weights are provided, scale point size by weight. Default is FALSE.
`ratio_response`	Logical. If TRUE, compute IC50 on ratio scale; if FALSE, use log2 intensities.
`precalculated_ratios`	Logical. If TRUE, assumes response contains pre-calculated ratios. Default is FALSE.
`transform_dose`	Logical. If TRUE, applies log10(dose + 1). Default is TRUE.
`show_ic50`	Logical. If TRUE, adds vertical line and annotation for IC50.
`target_response`	Numeric. Target response level for IC50/EC50 calculation. Default = 0.5.

add_ci	Logical. Include IC50 95% confidence interval bands if TRUE. Default is FALSE.
n_samples	Number of bootstrap samples if including confidence intervals. Default is 1000.
alpha	Alpha level for confidence intervals. Default is 0.10.
increasing	Logical or character. If TRUE, fits a non-decreasing model. If FALSE, fits non-increasing. If "both", fits both and selects better fit.
color_by	Character. Column name to color points by (e.g., "replicate", "peptide"). Default is NULL.
x_lab	Character. Custom label for the x-axis. If NULL, uses default based on transform_dose.
y_lab	Character. Custom label for the y-axis. If NULL, uses default based on ratio_response.
title	Character. Custom plot title. If NULL, auto-generates from drug and protein names.

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: With weights (visualized as point size)
plot2 <- visualizeResponseProtein(
  data = prepared_data,
  protein_name = "PROTEIN_A",
```

```
  drug_name = "Drug1",
  weights = prepared_data$weight,
  show_weights = TRUE,
  ratio_response = TRUE,
  show_ic50 = TRUE
)
print(plot2)

# Example 3: Color points by replicate
## Not run:
plot3 <- visualizeResponseProtein(
  data = prepared_data,
  protein_name = "PROTEIN_A",
  drug_name = "Drug1",
  ratio_response = TRUE,
  show_ic50 = TRUE,
  color_by = "replicate"
)
print(plot3)

## End(Not run)
```

Index

∗ **datasets**
DIA_MSstats_Normalized, 9
∗ **internal**
parse_timepoint, 17
.extractTemplatesFromData, 2

bootstrapIC50, 3
bootstrapIC50LogScale, 4
bootstrapIC50Precalculated, 4

calculatePeptideWeights, 5
calculateTurnoverRatios, 7
convertGroupToNumericDose, 8

DIA_MSstats_Normalized, 9
doseResponseFit, 10

fitIsotonicRegression, 12
futureExperimentSimulation, 13

MSstatsPrepareDoseResponseFit, 15

parse_timepoint, 17
plot_tpr_power_curve, 19
plotHitRateMSstatsResponse, 18
plotIsotonic, 18
predictIC50, 20
predictIC50Parallel, 23

run_tpr_simulation, 24

simulateChemoProteinLevelNonParametric, 26

visualizeResponseProtein, 27