Package 'pepdiff'

April 2, 2026

Type Package

Title Differential Abundance Analysis for Phosphoproteomics Data

Version 1.0.0

Author Dan MacLean [aut, cre]

Description
Provides tools for analyzing differential abundance in proteomics experiments. Implements S3 classes for data management and supports Generalized Linear Models (GLM; Nelder and Wedderburn (1972) <doi:10.2307/2344614>), Aligned Rank Transform (ART; Wobbrock et al. (2011) <doi:10.1145/1978942.1978963>), and pairwise test methods for statistical analysis. Includes visualization functions for Principal Component Analysis (PCA), volcano plots, and heatmaps.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports dplyr, tidyr, tibble, rlang, readr, ggplot2, cowplot, emmeans, ARTool, stats, magrittr, stringr, forcats, grid

Suggests ComplexHeatmap, UpSetR, RColorBrewer, viridis, circlize, factoextra, RankProd, MKinfer, testthat (>= 3.0.0), knitr, rmarkdown

Depends R (>= 4.1.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Maintainer Dan MacLean <dan.maclean@tsl.ac.uk>

Repository CRAN

Date/Publication 2026-04-02 20:00:02 UTC

Contents

`assess_missing` *calculate the proportion of peptides with missing values per group in a data set.*

Description

Group the data by treatment, seconds, bio rep and tech rep, then calculate the percent of NA in each group.

Usage

```
assess_missing(df)
```

Arguments

`df`	dataframe with unmerged tech reps; typically from ‘import_data()‘

Value

grouped summary dataframe

`classify_bf_evidence` *Classify Bayes factor into evidence categories*

Description

Converts numeric Bayes factors into categorical evidence levels following conventional thresholds (Jeffreys, 1961; Lee & Wagenmakers, 2013).

Usage

```
classify_bf_evidence(bf)
```

Arguments

`bf`	Numeric vector of Bayes factors (BF10)

Value

An ordered factor with levels:

`strong_null`	BF < 0.1 - Strong evidence for null hypothesis
`moderate_null`	BF 0.1-0.33 - Moderate evidence for null
`inconclusive`	BF 0.33-3 - Evidence is inconclusive
`moderate_alt`	BF 3-10 - Moderate evidence for alternative
`strong_alt`	BF > 10 - Strong evidence for alternative

Examples

```
classify_bf_evidence(c(0.05, 0.2, 1, 5, 20))
```

---

combine_tech_reps *Combine technical replicates*

Description

Explicitly combines technical replicates by averaging values within each combination of peptide, factors, and biological replicate.

Usage

```
combine_tech_reps(data, fun = mean)
```

Arguments

`data`	A pepdiff_data object with a tech_rep column
`fun`	Function to use for combining (default: mean)

Value

A pepdiff_data object with technical replicates combined

Examples

```
# Create data with technical replicates
tech_data <- data.frame(
  peptide = rep(paste0("PEP_", sprintf("%03d", 1:4)), each = 12),
  gene_id = rep(paste0("GENE_", 1:2), each = 24),
  bio_rep = rep(rep(1:3, each = 4), 4),
  tech_rep = rep(c(1, 2, 1, 2), 12),
  treatment = rep(c("ctrl", "ctrl", "trt", "trt"), 12),
  value = c(1466635, 1420000, 3106327, 3200000,  # PEP_001, bio_rep 1
            620128, 640000, 2744616, 2800000,    # PEP_001, bio_rep 2
            975783, 990000, 1943566, 1980000,    # PEP_001, bio_rep 3
            1171378, 1180000, 1949132, 1970000,  # PEP_002, bio_rep 1
```

```
              993280, 1000000, 1568840, 1590000,   # PEP_002, bio_rep 2
              1115054, 1130000, 2230232, 2250000,  # PEP_002, bio_rep 3
              1523992, 1540000, 3051384, 3100000,  # PEP_003, bio_rep 1
              515740, 520000, 1076568, 1090000,    # PEP_003, bio_rep 2
              1094908, 1110000, 2188616, 2210000,  # PEP_003, bio_rep 3
              736552, 750000, 1474984, 1490000,    # PEP_004, bio_rep 1
              1200000, 1210000, 1800000, 1820000,  # PEP_004, bio_rep 2
              980000, 990000, 1650000, 1670000)    # PEP_004, bio_rep 3
)

# Write to temporary file and read with pepdiff (including tech_rep)
temp_file <- tempfile(fileext = ".csv")
write.csv(tech_data, temp_file, row.names = FALSE)
dat_with_techreps <- read_pepdiff(
  file = temp_file,
  id = "peptide", gene = "gene_id", value = "value",
  factors = "treatment", replicate = "bio_rep", tech_rep = "tech_rep"
)
unlink(temp_file)  # Clean up

# Combine by averaging (default)
dat_avg <- combine_tech_reps(dat_with_techreps)

# Or combine by taking median
dat_median <- combine_tech_reps(dat_with_techreps, fun = median)

# Compare dimensions
nrow(dat_with_techreps$data)  # Before: 48 rows (with tech reps)
nrow(dat_avg$data)            # After: 24 rows (combined)
```

`compare` *Compare peptide abundances between conditions*

Description

Performs differential abundance analysis on proteomics data. Supports three methods: GLM (default), ART (non-parametric), and pairwise tests.

Usage

```
compare(data, ...)

## S3 method for class 'pepdiff_data'
compare(
  data,
  compare,
  ref,
  within = NULL,
  method = c("glm", "art", "pairwise"),
```

```
  test = c("wilcoxon", "bootstrap_t", "bayes_t", "rankprod"),
  alpha = 0.05,
  fdr_method = "BH",
  bf_threshold = 3,
  ...
)
```

Arguments

data	A pepdiff_data object from [read_pepdiff()]
...	Additional arguments passed to methods
compare	Factor to compare (character string)
ref	Reference level for comparisons
within	Optional factor(s) to stratify by
method	Analysis method: "glm" (default), "art", or "pairwise"
test	For pairwise method: "wilcoxon", "bootstrap_t", "bayes_t", or "rankprod"
alpha	Significance threshold (default 0.05). Used for p-value based tests.
fdr_method	FDR correction method (default "BH"). Not applied for bayes_t.
bf_threshold	Bayes factor threshold for significance (default 3). Only used when test = "bayes_t". BF > threshold marks peptide as significant.

Value

A pepdiff_results object containing:

results	Tibble with peptide, gene_id, comparison, fold_change, log2_fc, p_value, fdr, significant. For bayes_t: p_value/fdr are NA, includes bf and evidence columns.
comparisons	Tibble defining the comparisons made
method	Statistical method used
diagnostics	Model convergence information (for GLM/ART)
params	Analysis parameters
data	The original pepdiff_data object
call	The function call

Examples

```
# Create toy dataset for demonstration
toy_data <- data.frame(
  peptide = rep(paste0("PEP_", sprintf("%03d", 1:5)), each = 4),
  gene_id = rep(paste0("GENE_", ceiling((1:5)/2)), each = 4),
  bio_rep = rep(c(1, 2, 1, 2), 5),
  treatment = rep(c("ctrl", "ctrl", "trt", "trt"), 5),
  value = c(1466635, 620128, 3106327, 2744616,  # PEP_001
            975783, 1171378, 1943566, 1949132,   # PEP_002
            993280, 1115054, 1568840, 2230232,   # PEP_003
            1523992, 515740, 3051384, 1076568,   # PEP_004
```

```
            1094908, 736552, 2188616, 1474984)   # PEP_005
)

# Write to temporary file and read with pepdiff
temp_file <- tempfile(fileext = ".csv")
write.csv(toy_data, temp_file, row.names = FALSE)
dat <- read_pepdiff(
  file = temp_file,
  id = "peptide", gene = "gene_id", value = "value",
  factors = "treatment", replicate = "bio_rep"
)
unlink(temp_file)  # Clean up

# Simple comparison
results <- compare(dat, compare = "treatment", ref = "ctrl")

# Create factorial data for stratified comparison
factorial_data <- data.frame(
  peptide = rep(paste0("PEP_", sprintf("%03d", 1:3)), each = 12),
  gene_id = rep(paste0("GENE_", 1:3), each = 12),
  bio_rep = rep(rep(1:3, each = 4), 3),
  treatment = rep(c("ctrl", "ctrl", "trt", "trt"), 9),
  timepoint = rep(c("0h", "24h", "0h", "24h"), 9),
  value = c(1200000, 980000, 1800000, 1650000, 1100000, 1050000, 1950000, 1720000,
            950000, 1150000, 1600000, 1580000, 1300000, 1250000, 2100000, 1890000,
            1080000, 990000, 1750000, 1680000, 1150000, 1200000, 1850000, 1780000,
            1250000, 1180000, 1950000, 1820000, 1050000, 1100000, 1700000, 1650000,
            1180000, 1220000, 1800000, 1750000)
)

temp_file2 <- tempfile(fileext = ".csv")
write.csv(factorial_data, temp_file2, row.names = FALSE)
dat_factorial <- read_pepdiff(
  file = temp_file2,
  id = "peptide", gene = "gene_id", value = "value",
  factors = c("treatment", "timepoint"), replicate = "bio_rep"
)
unlink(temp_file2)  # Clean up

# Stratified comparison
results_stratified <- compare(dat_factorial, compare = "treatment",
                              ref = "ctrl", within = "timepoint")

# Pairwise test
results_pairwise <- compare(dat, compare = "treatment", ref = "ctrl",
                            method = "pairwise", test = "wilcoxon")

# Bayes factor test (uses bf_threshold instead of alpha/FDR)
results_bayes <- compare(dat, compare = "treatment", ref = "ctrl",
                         method = "pairwise", test = "bayes_t", bf_threshold = 10)
```

`compare.data.frame` *Default compare method for legacy data frames*

Description

This method handles calls to compare() with data frames from the old import_data() function. It issues a deprecation warning and delegates to compare_legacy().

Usage

```
## S3 method for class 'data.frame'
compare(data, ...)
```

Arguments

`data`	A data frame from import_data()
`...`	Arguments passed to compare_legacy()

Value

Results from compare_legacy()

`compare_calls` *compare sets of significant peptides called by the used data*

Description

produces an UpSet plot showing intersections and set-size of the different sets of significant peptides called by the methods used in the provided result dataframe

Usage

```
compare_calls(r, sig = 0.05)
```

Arguments

`r`	result dataframe typically from ‘compare()‘
`sig`	significance cut-off to select peptides

Value

UpSet plot

compare_many *compare many combinations of treatment and control*

Description

for each combination of treatment and control condition, runs the ‘compare()‘ function and collates the results

Usage

```
compare_many(df, comparison, iters = 1000, tests = c("bootstrap_t"))
```

Arguments

df	dataframe. Typically from ‘import_data()‘
comparison	path to file or dataframe of comparisons with columns treatment, t_seconds, control, c_seconds
iters	number of iterations to perform for iterative tests
tests	character vector of tests to use, one or more of: ‘norm_quantile‘, ‘bootstrap_t‘, ‘wilcoxon‘, ‘kruskal-wallis‘, ‘rank_product‘

Value

Named list of dataframes. Each element contains statistical comparison results from `compare_legacy()`, with names derived from treatment-control combinations.

---

estimate_result_clusters *plots a Figure of Merit curve to help estimate the number of clusters in the results*

Description

plots a Figure of Merit curve to help estimate the number of clusters in the results

Usage

```
estimate_result_clusters(r)
```

Arguments

r	the results object from ‘compare_many()‘

Value

A ggplot2 object displaying a Figure of Merit curve to help determine optimal number of clusters in the results.

fc_qqplot *plot qqplot of fold changes from a comparison*

Description

plot qqplot of fold changes from a comparison

Usage

```
fc_qqplot(df, log = FALSE, base = 2)
```

Arguments

df	result dataframe, typically from ‘compare()‘
log	log the fold change values
base	base of the log, if used

Value

A ggplot2 object displaying a quantile-quantile plot of fold change values for assessing distribution normality.

fold_change_matrix *returns a matrix of fold change values*

Description

Computes the fold change relative to the control sample and returns a matrix with comparisons in columns and peptides in rows. Use this if you want data for a customised heatmap

Usage

```
fold_change_matrix(
  l,
  log = TRUE,
  base = 2,
  sig_only = FALSE,
  sig_level = 0.05,
  metric = "bootstrap_t_fdr"
)
```

Arguments

l	list of results, usually from ‘compare_many()‘
log	whether to log the data
base	base used in logging (default = 2)
sig_only	return only rows with 1 or more values significant at ‘sig_level‘ of ‘metric‘
sig_level	significance level cutoff
metric	the test metric used to determine significance one of: ‘bootstrap_t_p_val‘, ‘bootstrap_t_fdr‘ ‘wilcoxon_p_val‘, ‘wilcoxon_fdr‘ ‘kruskal_p_val‘, ‘kruskal_fdr‘ ‘rank_prod_p1_p_val‘, ‘rank_prod_p2_p_val‘, ‘rank_prod_p1_fdr‘, ‘rank_prod_p2_fdr‘.

Value

matrix

---

get_bootstrap_percentile

get p values for contrast using boostrap t test

Description

get p values for contrast using boostrap t test

Legacy: Get bootstrap t-test p-values for matrix data

Usage

```
get_bootstrap_percentile(treatment, control, iters = 1000)

get_bootstrap_percentile(treatment, control, iters = 1000)
```

Arguments

treatment	Matrix of treatment data (rows = peptides, cols = replicates)
control	Matrix of control data
iters	Number of bootstrap iterations

Value

dataframe with two columns ‘bootstrap_t_p_val‘ and ‘bootstrap_t_fdr‘

Data frame with bootstrap_t_p_val and bootstrap_t_fdr columns

get_comparison *Get results for a specific comparison*

Description

Get results for a specific comparison

Usage

```
get_comparison(x, comparison)
```

Arguments

x	A pepdiff_results object
comparison	Comparison name to retrieve

Value

A tibble with results for the specified comparison

get_kruskal_percentile *get p values for contrast using Kruskal-Wallis test*

Description

get p values for contrast using Kruskal-Wallis test

Legacy: Get Kruskal-Wallis test p-values for matrix data

Usage

```
get_kruskal_percentile(treatment, control)

get_kruskal_percentile(treatment, control)
```

Arguments

treatment	Matrix of treatment data
control	Matrix of control data

Value

dataframe with two columns ‘kruskal_p_val‘ and ‘kruskal_fdr‘

Data frame with kruskal_p_val and kruskal_fdr columns

get_peptide *Get results for a specific peptide*

Description

Get results for a specific peptide

Usage

```
get_peptide(x, peptide)
```

Arguments

x	A pepdiff_results object
peptide	Peptide ID to retrieve

Value

A tibble with results for the specified peptide

get_rp_percentile *get p values for contrast using Rank Products test*

Description

get p values for contrast using Rank Products test

Legacy: Get Rank Products test p-values for matrix data

Usage

```
get_rp_percentile(treatment, control)

get_rp_percentile(treatment, control)
```

Arguments

treatment	Matrix of treatment data
control	Matrix of control data

Value

dataframe with four columns, two for the test each way from RankProducts ‘rank_prod_p1_p_val‘, ‘rank_prod_p2_p_val‘ and ‘rank_prod_p1_fdr‘, ‘rank_prod_p2_fdr‘.

Data frame with rank product p-values and FDR

`get_sig_rows` *works out if a peptide has at least one significant value across the experiment Composes a matrix of the 'metric' significance values with peptides in rows, experiments in columns and works out if each peptide row has a value below the stated cut off*

Description

#' returns a logical vector of length equal to row number of matrix

Usage

```
get_sig_rows(l, metric = "bootstrap_t_pval", sig_level = 0.05)
```

Arguments

`l`	list of results, usually from 'compare_many()'
`metric`	the test metric used to determine significance one of:
`sig_level`	significance level cutoff

`get_wilcoxon_percentile` *get p values for contrast using Wilcoxon test*

Description

get p values for contrast using Wilcoxon test

Legacy: Get Wilcoxon test p-values for matrix data

Usage

```
get_wilcoxon_percentile(treatment, control)

get_wilcoxon_percentile(treatment, control)
```

Arguments

`treatment`	Matrix of treatment data
`control`	Matrix of control data

Value

dataframe with two columns 'wilcoxon_p_val' and 'wilcoxon_fdr'

Data frame with wilcoxon_p_val and wilcoxon_fdr columns

`import_data` *read data from a file*

Description

reads data, renames columns appropriately, discards unused columns, factors and reorders, discards duplicate rows

Usage

```
import_data(
  file,
  treatment = "genotype",
  bio_rep = "bio_rep",
  tech_rep = "tech_rep",
  quant = "total_area",
  seconds = "seconds",
  gene_id = "gene_id",
  peptide = "peptide_sequence"
)
```

Arguments

`file`	Path to the file to load - must be a csv file
`treatment`	Column containing the treatment of the observation
`bio_rep`	Column containing the biological replicate of the observation
`tech_rep`	Column containing the technical replicate of the observation
`quant`	Column containing the quantitation data
`seconds`	Column containing timepoint of observation
`gene_id`	Column containing the id of the gene this hit
`peptide`	Column containing the sequence of this peptide

Value

tibble with columns id, gene_id, peptide, treatment, seconds, bio_rep, tech_rep, quant

`kmeans_by_selected_cols` *Perform kmeans of a dataset using just data in selected columns, then return matrices of all columns*

Description

Perform kmeans of a dataset using just data in selected columns, then return matrices of all columns

Usage

```
kmeans_by_selected_cols(
  l,
  cols = NULL,
  log = TRUE,
  base = 2,
  sig_only = TRUE,
  sig_level = 0.05,
  metric = "bootstrap_t_p_val",
  k = NA,
  nstart = 25,
  itermax = 1000
)
```

Arguments

`l`	list of results, usually from ‘compare_many()‘
`cols`	names of columns to perform the k-means with
`log`	whether to log the data
`base`	base used in logging (default = 2)
`sig_only`	return only rows with 1 or more values significant at ‘sig_level‘ of ‘metric‘
`sig_level`	significance level cutoff
`metric`	the test metric used to determine significance one of: ‘bootstrap_t_p_val‘, ‘bootstrap_t_fdr‘ ‘wilcoxon_p_val‘, ‘wilcoxon_fdr‘ ‘kruskal_p_val‘, ‘kruskal_fdr‘ ‘rank_prod_p1_p_val‘, ‘rank_prod_p2_p_val‘, ‘rank_prod_p1_fdr‘, ‘rank_prod_p2_fdr‘.
`k`	number of clusters to make
`nstart`	nstart value for ‘kmeans()‘
`itermax`	number of ‘kmeans()‘ iterations (1000)

Value

list of matrices

`list2mat` *converts a results object to a matrix as if for direct use in external heatmap functions*

Description

converts a results object to a matrix as if for direct use in external heatmap functions

Usage

```
list2mat(r, column = "fold_change")
```

Arguments

`r`	results object, usually from ‘compare_many()‘
`column`	column from results data to put into matrix, default = "fold_change"

`long_results` *Convert wide format results table to long format*

Description

Tidies up the wide results table from ‘compare()‘ to a long format.

Usage

```
long_results(r)
```

Arguments

`r`	Results dataframe typically from ‘compare()‘

Value

Dataframe in long format

`metrics` *reports metrics available for significance values*

Description

reports metrics available for significance values

Usage

```
metrics()
```

missing_peptides_plot *plot the representation of peptides in each group.*

Description

Shows what proportion of the whole set of peptides is missing in each group of treatment, seconds, bio rep and tech rep.

Usage

```
missing_peptides_plot(df)
```

Arguments

df	dataframe with unmerged tech reps; typically from ‘import_data()‘

Value

ggplot2 plot

norm_qqplot *draw qqplots for data*

Description

Plot qqplot of distribution of quantifications in data for each treatment, seconds and biological replicate

Usage

```
norm_qqplot(df, log = FALSE, base = 2)
```

Arguments

df	dataframe; typically from ‘import_data()‘
log	perform log transform of data
base	base to use in log transform

Value

ggplot2 plot

plot.pepdiff_data *Plot method for pepdiff_data*

Description

Creates a multi-panel diagnostic plot showing PCA, distributions, and missingness.

Usage

```
## S3 method for class 'pepdiff_data'
plot(x, ...)
```

Arguments

x	A pepdiff_data object
...	Additional arguments (ignored)

Value

A cowplot grid of plots

plot.pepdiff_results *Plot method for pepdiff_results*

Description

Creates a multi-panel plot showing volcano, p-value/BF histogram, and FC distribution. Automatically dispatches to BF-specific plots when results are from bayes_t test.

Usage

```
## S3 method for class 'pepdiff_results'
plot(x, ...)
```

Arguments

x	A pepdiff_results object
...	Additional arguments (ignored)

Value

A cowplot grid of plots

plot_bf_distribution *Bayes factor distribution plot*

Description

Creates a histogram of log10(BF) values with reference lines at standard thresholds.

Usage

```
plot_bf_distribution(results, comparison = NULL)
```

Arguments

results	A pepdiff_results object from bayes_t test
comparison	Optional comparison to filter by

Value

A ggplot object

plot_distributions_simple *Simple distribution plot for pepdiff_data*

Description

Simple distribution plot for pepdiff_data

Usage

```
plot_distributions_simple(data, facet_by = NULL)
```

Arguments

data	A pepdiff_data object
facet_by	Factor to facet by (default: first factor)

Value

A ggplot object

plot_fc *plot histogram of fold change distribution for a comparison*

Description

plot histogram of fold change distribution for a comparison

Usage

```
plot_fc(df, log = FALSE, base = 2)
```

Arguments

df	result dataframe, typically from ‘compare()‘
log	log the fold change values
base	base of the log, if used

Value

ggplot2 plot

plot_fc_distribution_new *Fold change distribution for pepdiff_results*

Description

Fold change distribution for pepdiff_results

Usage

```
plot_fc_distribution_new(results, comparison = NULL)
```

Arguments

results	A pepdiff_results object
comparison	Optional comparison to filter by

Value

A ggplot object

`plot_fit_diagnostics` *Plot GLM fit diagnostics*

Description

Creates a multi-panel diagnostic plot to help assess whether GLM models fit the data well. This is useful for deciding whether to use GLM or switch to ART (Aligned Rank Transform).

Usage

```
plot_fit_diagnostics(
  results,
  n_sample = 6,
  deviance_threshold = NULL,
  full_qq = FALSE
)
```

Arguments

`results`	A ‘pepdiff_results‘ object from ‘compare()‘ with ‘method = "glm"‘
`n_sample`	Number of peptides to show in sample residual plots (default 6)
`deviance_threshold`	Optional threshold for flagging high-deviance peptides. If NULL (default), uses the 95th percentile of deviance values.
`full_qq`	Deprecated. Residuals are now stored during ‘compare()‘ so accurate QQ plots are always available without refitting.

Details

The function generates a 4-panel diagnostic plot:

Panel 1: Deviance Distribution - Histogram showing the distribution of residual deviance across all converged peptides. A long right tail suggests some peptides fit poorly.

Panel 2: Deviance vs Fold Change - Scatter plot of deviance against absolute log2 fold change. If high-deviance points cluster at extreme fold changes, this may indicate outlier-driven "significant" results.

Panel 3: Sample Residual Plots - Residuals vs fitted values for a sample of peptides (2 with highest deviance, 2 median, 2 lowest). Look for random scatter around zero; patterns or funnels indicate poor fit.

Panel 4: Pooled QQ Plot - Quantile-quantile plot of pooled residuals. Points should fall on the diagonal line. S-curves indicate heavy tails (consider ART), systematic deviation suggests wrong distributional assumption.

Value

Invisibly returns a list with:

`plot`	The diagnostic plot (ggplot/cowplot grid)
`flagged`	Tibble of peptides with potential fit issues
`summary`	List with summary statistics (n_flagged, median_deviance, etc.)

Interpretation

Use GLM when: - Deviance distribution looks reasonable (few flagged peptides) - No systematic patterns in residual plots - QQ plot is reasonably linear

Consider ART when: - Many peptides (>15 - Residual plots show systematic curves or funnels - QQ plot shows heavy tails (S-curve)

See Also

[compare()] for running the analysis, ‘vignette("checking_fit")‘ for detailed guidance on interpreting diagnostics

Examples

```
# Create toy dataset for GLM analysis
toy_data <- data.frame(
  peptide = rep(paste0("PEP_", sprintf("%03d", 1:8)), each = 6),
  gene_id = rep(paste0("GENE_", 1:4), each = 12),
  bio_rep = rep(rep(1:3, each = 2), 8),
  treatment = rep(c("ctrl", "trt"), 24),
  value = c(1466635, 3106327, 620128, 2744616, 975783, 1943566,  # PEP_001
            1171378, 1949132, 993280, 1568840, 1115054, 2230232,  # PEP_002
            1523992, 3051384, 515740, 1076568, 1094908, 2188616,  # PEP_003
            736552, 1474984, 1200000, 1800000, 980000, 1650000,   # PEP_004
            1100000, 1950000, 1050000, 1720000, 950000, 1600000,  # PEP_005
            1150000, 1580000, 1300000, 2100000, 1250000, 1890000, # PEP_006
            1080000, 1750000, 990000, 1680000, 1150000, 1850000,  # PEP_007
            1200000, 1780000, 1250000, 1950000, 1180000, 1820000) # PEP_008
)

# Write to temporary file and read with pepdiff
temp_file <- tempfile(fileext = ".csv")
write.csv(toy_data, temp_file, row.names = FALSE)
dat <- read_pepdiff(
  file = temp_file,
  id = "peptide", gene = "gene_id", value = "value",
  factors = "treatment", replicate = "bio_rep"
)
unlink(temp_file)  # Clean up

# Run GLM analysis
results <- compare(dat, compare = "treatment", ref = "ctrl", method = "glm")
```

```
# Check fit diagnostics
diag <- plot_fit_diagnostics(results)

# View flagged peptides (if any)
head(diag$flagged)

# Get summary statistics
diag$summary
```

`plot_heatmap` *makes heatmap from all experiments, filter on a single metric and sig value*

Description

reduces dataframes and makes long list, makes a basic heatmap. Use ‘fold_change_matrix()‘ to extract data in a heatmappable format

Usage

```
plot_heatmap(
  l,
  sig_level = 0.05,
  metric = "bootstrap_t_fdr",
  log = TRUE,
  base = 2,
  col_order = NULL,
  sig_only = TRUE,
  pal = "RdBu",
  lgd_x = 1.7,
  lgd_y = 1,
  padding = c(0, 0, 0, 3)
)
```

Arguments

`l`	list of results, usually from ‘compare_many()‘
`sig_level`	significance level cutoff
`metric`	the test metric used to determine significance one of: ‘bootstrap_t_p_val‘, ‘bootstrap_t_fdr‘ ‘wilcoxon_p_val‘, ‘wilcoxon_fdr‘ ‘kruskal_p_val‘, ‘kruskal_fdr‘ ‘rank_prod_p1_p_val‘, ‘rank_prod_p2_p_val‘, ‘rank_prod_p1_fdr‘, ‘rank_prod_p2_fdr‘.
`log`	whether to log the data
`base`	base used in logging (default = 2)
`col_order`	specify a column order for the plot, default is names(l)
`sig_only`	return only rows with 1 or more values significant at ‘sig_level‘ of ‘metric‘

pal	cbrewer palette to use "RdBu", needs minimum 11 colours
lgd_x	x offset of legend placement in ‘in‘ units
lgd_y	y offset of legend placement in ‘in‘ units
padding	vector of padding values to pass to ComplexHeatmap::draw for padding of heatmap sections

Value

No return value, called for side effects. Displays a heatmap visualization of fold change data.

`plot_kmeans` *K-means cluster the data on the samples*

Description

Performs and draws a K-means cluster on the samples. Estimates number of clusters as the product of the number of treatments and seconds. So tries to group the bio reps together

Usage

```
plot_kmeans(df, nstart = 25, iter.max = 1000)
```

Arguments

df	dataframe, typically from ‘import_data()‘
nstart	nstart points for ‘kmeans()‘ function
iter.max	max iterations to perform for ‘kmeans()‘ function

Value

ggplot2 plot

`plot_missingness_simple`
Simple missingness plot for pepdiff_data

Description

Simple missingness plot for pepdiff_data

Usage

```
plot_missingness_simple(data)
```

Arguments

data	A pepdiff_data object

Value

A ggplot object

---

plot_pca *plots a pca on the treatment, seconds, bio-rep*

Description

Performs and draws a PCA plot with four panels, PCA with sample names coloured by treatment, seconds and biorep and a scree plot of the PCA dimensions

Usage

```
plot_pca(df)
```

Arguments

df	dataframe, typically from ‘import_data()‘

Value

ggplot2 plot

---

plot_pca_simple *Simple PCA plot for pepdiff_data*

Description

Simple PCA plot for pepdiff_data

Usage

```
plot_pca_simple(data, color_by = NULL)
```

Arguments

data	A pepdiff_data object
color_by	Factor to color points by (default: first factor)

Value

A ggplot object

plot_pvalue_histogram *P-value histogram for pepdiff_results*

Description

P-value histogram for pepdiff_results

Usage

```
plot_pvalue_histogram(results, comparison = NULL)
```

Arguments

results	A pepdiff_results object
comparison	Optional comparison to filter by

Value

A ggplot object

plot_quant_distributions *draw density plots for data*

Description

Plot density of quantities in data for each treatment, seconds and biological replicate

Usage

```
plot_quant_distributions(df, log = FALSE, base = 2)
```

Arguments

df	dataframe; typically from ‘import_data()‘
log	perform log transform of data
base	base to use in log transform

Value

ggplot2 plot

`plot_result` *plot the p-values against fold change for the tests used in 'compare()'*

Description

plots fold change against p-value in all tests used, also splits data on the number of biological replicates were available before value replacement for each peptide

Usage

```
plot_result(df)
```

Arguments

`df`	result dataframe typically from 'compare()'

Value

ggplot2 plot

`plot_volcano_bf` *Volcano plot for Bayes factor results*

Description

Creates a volcano plot with log10(BF) on the y-axis instead of -log10(p-value). Reference lines are drawn at BF thresholds (3, 10) and their reciprocals (0.33, 0.1).

Usage

```
plot_volcano_bf(results, comparison = NULL, bf_threshold = 3, fc_threshold = 1)
```

Arguments

`results`	A pepdiff_results object from bayes_t test
`comparison`	Optional comparison name to filter by
`bf_threshold`	BF threshold for coloring (default 3)
`fc_threshold`	Fold change threshold for labeling (default 1)

Value

A ggplot object

plot_volcano_new *Volcano plot for pepdiff_results*

Description

Volcano plot for pepdiff_results

Usage

```
plot_volcano_new(results, comparison = NULL, alpha = 0.05, fc_threshold = 1)
```

Arguments

results	A pepdiff_results object
comparison	Optional comparison name to filter by
alpha	Significance threshold for coloring
fc_threshold	Fold change threshold for labeling

Value

A ggplot object

print.pepdiff_data *Print method for pepdiff_data*

Description

Print method for pepdiff_data

Usage

```
## S3 method for class 'pepdiff_data'
print(x, ...)
```

Arguments

x	A pepdiff_data object
...	Additional arguments (ignored)

Value

The object invisibly

print.pepdiff_results *Print method for pepdiff_results*

Description

Print method for pepdiff_results

Usage

```
## S3 method for class 'pepdiff_results'
print(x, ...)
```

Arguments

x	A pepdiff_results object
...	Additional arguments (ignored)

Value

The object invisibly

p_value_hist *plot histograms of p-values for each test used*

Description

plot histograms of p-values for each test used

Usage

```
p_value_hist(r)
```

Arguments

r	list of result dataframes typically from ‘compare()‘

Value

ggplot2 plot

read_pepdiff *Read proteomics data into a pepdiff_data object*

Description

Imports a CSV file containing PRM proteomics data and creates a pepdiff_data object suitable for analysis with [compare()].

Usage

```
read_pepdiff(file, id, gene, value, factors, replicate, tech_rep = NULL)
```

Arguments

file	Path to CSV file
id	Column name containing peptide identifiers
gene	Column name containing gene identifiers
value	Column name containing abundance values
factors	Character vector of column names to use as experimental factors
replicate	Column name containing biological replicate identifiers
tech_rep	Optional column name containing technical replicate identifiers. If provided, data will NOT be automatically combined - use [combine_tech_reps()] explicitly after import.

Value

A pepdiff_data object with components:

data	Tibble with columns: peptide, gene_id, [factors], bio_rep, value
factors	Character vector of factor names
design	Tibble of factor combinations with n_reps, n_peptides
missingness	Tibble of peptide missingness statistics
peptides	Character vector of unique peptide IDs
call	The original function call

Examples

```
## Not run:
# Simple import with one factor
dat <- read_pepdiff(
  "data.csv",
  id = "peptide_sequence",
  gene = "gene_name",
  value = "intensity",
  factors = "treatment",
```

```
  replicate = "bio_rep"
)

# Multi-factor import
dat <- read_pepdiff(
  "data.csv",
  id = "peptide",
  gene = "gene_id",
  value = "total_area",
  factors = c("treatment", "timepoint"),
  replicate = "bio_rep"
)

## End(Not run)
```

`significant` *Extract significant results*

Description

Extract significant results

Usage

```
significant(x, alpha = NULL, by_fdr = TRUE, bf_threshold = NULL)
```

Arguments

`x`	A pepdiff_results object
`alpha`	Significance threshold for p-value tests (default uses analysis alpha)
`by_fdr`	Logical, use FDR-adjusted p-values for p-value tests (default TRUE)
`bf_threshold`	BF threshold for bayes_t results (default uses analysis bf_threshold)

Value

A tibble of significant results

subset.pepdiff_data *Subset a pepdiff_data object*

Description

Subset a pepdiff_data object

Usage

```
## S3 method for class 'pepdiff_data'
subset(x, peptides = NULL, ...)
```

Arguments

x	A pepdiff_data object
peptides	Character vector of peptide IDs to keep (optional)
...	Additional subsetting expressions evaluated in the data

Value

A new pepdiff_data object

summary.pepdiff_data *Summary method for pepdiff_data*

Description

Summary method for pepdiff_data

Usage

```
## S3 method for class 'pepdiff_data'
summary(object, ...)
```

Arguments

object	A pepdiff_data object
...	Additional arguments (ignored)

Value

A summary list invisibly

summary.pepdiff_results *Summary method for pepdiff_results*

Description

Summary method for pepdiff_results

Usage

```
## S3 method for class 'pepdiff_results'
summary(object, ...)
```

Arguments

object	A pepdiff_results object
...	Additional arguments (ignored)

Value

A summary list invisibly

test_bayes_t *Bayes factor t-test for two groups*

Description

Computes a Bayes factor comparing the alternative hypothesis (group difference) to the null hypothesis (no difference) using the JZS (Jeffreys-Zellner-Siow) prior. Uses an analytical approximation for computational efficiency.

Usage

```
test_bayes_t(control, treatment, r_scale = 0.707)
```

Arguments

control	Numeric vector of control group values
treatment	Numeric vector of treatment group values
r_scale	Scale parameter for the Cauchy prior on effect size (default 0.707)

Details

The Bayes factor is interpreted as: - BF10 > 10: Strong evidence for difference - BF10 > 3: Moderate evidence for difference - BF10 0.33-3: Inconclusive - BF10 < 0.33: Moderate evidence for no difference - BF10 < 0.1: Strong evidence for no difference

Unlike p-values, Bayes factors are NOT converted to pseudo-p-values. Use [classify_bf_evidence()] to interpret BF values categorically.

Value

A list with components:

`bf`	Bayes factor (BF10) - evidence for alternative vs null
`effect_size`	Cohen's d effect size
`method`	"bayes_t"

Examples

```
ctrl <- c(100, 120, 110, 105)
trt <- c(200, 220, 180, 210)
test_bayes_t(ctrl, trt)
```

`test_bootstrap_t` *Bootstrap t-test for two groups*

Description

Performs a bootstrap-based t-test comparing two groups. This is more robust than a standard t-test when assumptions of normality may not hold.

Usage

```
test_bootstrap_t(control, treatment, n_boot = 1000, seed = NULL)
```

Arguments

`control`	Numeric vector of control group values
`treatment`	Numeric vector of treatment group values
`n_boot`	Number of bootstrap iterations (default 1000)
`seed`	Optional random seed for reproducibility

Value

A list with components:

`p_value`	The two-sided p-value
`t_obs`	The observed t-statistic
`method`	"bootstrap_t"

Examples

```
ctrl <- c(100, 120, 110, 105)
trt <- c(200, 220, 180, 210)
test_bootstrap_t(ctrl, trt, n_boot = 500)
```

`test_rankprod` *Rank products test for two groups*

Description

‘r lifecycle::badge("deprecated")‘

This function is deprecated because Rank Products requires ranking across ALL peptides, not within single peptides. The per-peptide permutation approach produces unreliable p-values.

Use ‘compare()‘ with ‘test = "rankprod"‘ instead, which properly uses the RankProd package to rank across all peptides.

Usage

```
test_rankprod(control, treatment, n_perm = 1000, seed = NULL)
```

Arguments

`control`	Numeric vector of control group values
`treatment`	Numeric vector of treatment group values
`n_perm`	Number of permutations for p-value estimation (default 1000)
`seed`	Optional random seed for reproducibility

Value

A list with components:

`p_value_up`	P-value for upregulation (treatment > control)
`p_value_down`	P-value for downregulation (treatment < control)
`p_value`	Combined two-sided p-value (minimum of up/down)
`rp_up`	Rank product for upregulation
`rp_down`	Rank product for downregulation
`method`	"rankprod"

Examples

```
ctrl <- c(100, 120, 110, 105)
trt <- c(200, 220, 180, 210)
test_rankprod(ctrl, trt, n_perm = 100)  # Deprecated
```

test_wilcoxon *Wilcoxon rank-sum test for two groups*

Description

Performs a two-sample Wilcoxon rank-sum test (Mann-Whitney U test) to compare abundance values between control and treatment groups.

Usage

```
test_wilcoxon(control, treatment, ...)
```

Arguments

control	Numeric vector of control group values
treatment	Numeric vector of treatment group values
...	Additional arguments passed to [stats::wilcox.test()]

Value

A list with components:

p_value	The p-value from the test
statistic	The test statistic W
method	"wilcoxon"

Examples

```
ctrl <- c(100, 120, 110, 105)
trt <- c(200, 220, 180, 210)
test_wilcoxon(ctrl, trt)
```

times_measured *calculate number of measurements of each peptide in each treatment and time*

Description

For each peptide, works out how many biologically replicated measurements are available in the different combinations of treatment and seconds

Usage

```
times_measured(df)
```

Arguments

df	dataframe. Typically from ‘import_data()‘

Value

dataframe

---

times_measured_plot *plot the count of the number of times peptides were measured.*

Description

Calculates and plots the number of times each peptide was measured in each combination of treatment and seconds and presents a summary plot

Usage

```
times_measured_plot(df)
```

Arguments

df	dataframe. Typically from ‘import_data()‘

Value

ggplot2 plot

---

volcano_plot *volcano plot the data*

Description

draws a plot of peptide count against log fc at either protein or peptide level for samples

Usage

```
volcano_plot(
  l,
  log = FALSE,
  base = 2,
  sig_level = 0.05,
  metric = "bootstrap_t_p_val",
  option = "E",
  direction = -1
)
```

Arguments

l	list of results data frames, typically from ‘compare_many()‘
log	log the data
base	base for logging
sig_level	significance cutoff for colour
metric	metric to use for significance
option	viridis colour scheme key to use
direction	viridis colour scheme direction (1/-1)

Value

ggplot2 plot

Index