Package: ProteinBatcher (via r-universe)

November 25, 2026

Title An end-to-end proteomics workflow with condition-aware imputation, flexible statistical modelling and interactive visualization

Version 1.0.0

Description ProteinBatcher provides utilities to read, validate, and harmonize quantitative proteomics result tables generated by DIA-NN from data-independent acquisition mass spectrometry experiments. The package enables batch processing of multiple DIA-NN output files, consistent handling of protein identifiers and associated metadata, and transformation of tabular data into analysis-ready formats. It facilitates downstream differential abundance analysis using linear modeling frameworks based on limma, with explicit support for multivariable experimental designs including batch effects and other confounding covariates, as well as correlation-aware designs via duplicateCorrelation.

License MIT + file LICENSE

URL https://github.com/DataScienceRD-Almirall/ProteinBatcher

BugReports https://github.com/DataScienceRD-Almirall/ProteinBatcher/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports data.table, ggplot2, ggrepel, grDevices, grid, limma, methods, S4Vectors, stats, SummarizedExperiment, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

biocViews Proteomics, MassSpectrometry, DataImport, DataRepresentation, Software, BatchEffect

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:53 UTC

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

RemoteRef RELEASE_3_24

RemoteSha ffbf19397bf870f036fd6920889c236a320c6e35

Contents

filter_se_missing 2
impute_se 4
plot_deregulogram 6
plot_volcano_customized 9
run_proteomics_pipeline 11
test_limma_customized 16

Index **20**

`filter_se_missing` *Filter proteins by missingness within conditions (SummarizedExperiment)*

Description

`filter_se_missing()` removes proteins (rows) that fail a per-condition missingness threshold. For each condition in `colData(se)$condition`, the function computes the **proportion of non-missing intensities per protein** and keeps a protein if it meets the threshold in **at least one** condition.

This is useful in label-free proteomics pipelines prior to imputation and differential analysis, ensuring that proteins are not uniformly sparse across all conditions.

Usage

```
filter_se_missing(se, percentage = 50)
```

Arguments

`se`	A `SummarizedExperiment` (or compatible) object with: • an assay matrix accessible via `assay(se)` (numeric, may contain `NA`) • `colData(se)$condition` (character or factor) giving the experimental condition per sample. • (optional) `rowData(se)$Protein.Names` and `rowData(se)$Genes` used to annotate the returned table of removed proteins. If missing, they will be safely filled with `NA_character_`.
`percentage`	Numeric scalar in `[0, 100]`. Minimum proportion of **non-missing** values required within a condition for a protein to be retained. Default: `50` (i.e., at least half of samples observed within any one condition).

Details

Let X be the assay matrix (`assay(se)`), rows = proteins and columns = samples. For every distinct `condition` in `colData(se)$condition`, we compute `rowMeans(!is.na(X[, condition == cn]))`. A protein is **kept** if the per-condition proportion of observed values is `>= percentage / 100` for **any** condition. Otherwise, it is **removed**.

Value

A named list with two elements:

`se_filt` `SummarizedExperiment` containing only retained proteins.

`removed` A `data.frame` with metadata for proteins that were filtered out, columns: `protein_id`, `gene_name`, and `reason`.

Notes

- Samples with `NA` in `colData(se)$condition` are **excluded** from per-condition computations and a message is emitted.
- If `percentage = 0`, all proteins with at least one observed value in any condition are kept. If `percentage = 100`, only proteins fully observed within at least one condition are kept.

See Also

- `SummarizedExperiment::SummarizedExperiment`
- Downstream steps like `impute_se()` for imputation.

Examples

```
library(SummarizedExperiment)

## Toy expression matrix: 4 proteins x 4 samples
mat <- matrix(
  c(
    1,  NA,  2,  3,   # p1: OK in cond B
    NA, NA,  NA, NA, # p2: always missing -> removed
    5,  6,   NA, NA, # p3: OK in cond A
    NA, NA,  1,  NA  # p4: too sparse everywhere
  ),
  nrow = 4, byrow = TRUE,
  dimnames = list(
    paste0("p", 1:4),
    paste0("s", 1:4)
  )
)

## Sample conditions
cd <- DataFrame(
  condition = c("A", "A", "B", "B"),
  row.names = colnames(mat)
)
```

```
## Protein annotations (optional)
rd <- DataFrame(
  Protein.Names = paste0("Prot", 1:4),
  Genes = paste0("Gene", 1:4),
  row.names = rownames(mat)
)

se <- SummarizedExperiment(
  assays = list(intensity = mat),
  colData = cd,
  rowData = rd
)

## Keep proteins with >= 50% observed values in at least one condition
out <- filter_se_missing(se, percentage = 50)

## Filtered SummarizedExperiment
out$se_filt

## Table of removed proteins with reasons
out$removed
```

`impute_se` *Impute missing proteomics intensities within conditions (SummarizedExperiment)*

Description

`impute_se()` imputes missing values using a two-step, condition-aware strategy: (1) within-condition mean when missingness < `threshold`; (2) left-censored draws (LDV) for remaining NAs, using a low-intensity reference and per-protein SD.

Usage

```
impute_se(
  se,
  threshold = 0.5,
  ldv_source = c("global", "per_condition"),
  sd_fallback = NULL,
  lower_bound = NULL
)
```

Arguments

`se`	A `SummarizedExperiment` with numeric assay and `colData(se)$condition`.
`threshold`	Numeric in `(0,1]`, default `0.5`.

ldv_source	c("global","per_condition"), default "global".
sd_fallback	Optional numeric fallback SD; default is median non-zero SD.
lower_bound	Optional minimum clip for imputed values (avoid on log scale).

Details

Let assay(se) be a numeric matrix (rows = proteins, cols = samples); colData(se)$condition provides the condition for each sample. Step (1) replaces NAs in a condition with the mean of observed values for that protein in the same condition when the NA proportion is below threshold. Step (2) replaces remaining NAs with rnorm(1, mean = LDV, sd = sd_protein), where LDV is the global (or per-condition) minimum observed intensity; per-protein SDs are computed from available values with a robust fall-back when needed.

The per-cell imputation method is recorded in metadata(se)$imputation_map ("none"|"mean"|"ldv"). A row-level flag is added at rowData(se)$imputed.

Value

A SummarizedExperiment with imputed assay, updated metadata/rowData.

See Also

SummarizedExperiment::SummarizedExperiment

Examples

```
library(SummarizedExperiment)

## Toy matrix with missing values
mat <- matrix(
  c(
    10, 11,  NA, NA,  # p1: partial missing
    NA, NA,  NA, NA, # p2: fully missing
    5,  NA,  6,  7   # p3: mixed
  ),
  nrow = 3, byrow = TRUE,
  dimnames = list(
    paste0("p", 1:3),
    paste0("s", 1:4)
  )
)

## Conditions for samples
cd <- DataFrame(
  condition = c("A", "A", "B", "B"),
  row.names = colnames(mat)
)

## Build SummarizedExperiment
se <- SummarizedExperiment(
  assays = list(intensity = mat),
  colData = cd
```

```
)

## Impute with default strategy:
## - mean within condition if missingness < threshold
## - LDV otherwise
set.seed(123)
se_imp <- impute_se(se, threshold = 0.5, ldv_source = "global")

## Imputed assay
assay(se_imp)

## Per-cell imputation method
metadata(se_imp)$imputation_map

## Row-level flag: was any value imputed?
rowData(se_imp)$imputed
```

`plot_deregulogram` *Deregulogram plot for interaction models (2-level factors)*

Description

Creates a deregulogram: a scatter of full effects in two levels of an interaction factor (e.g. sex female vs male, batch day1 vs day2), colored by whether the interaction is significant. This visualization complements volcano plots by showing effect concordance and sex-/batch-bias in magnitude and direction, as described in the proteomics workflow documentation.

Usage

```
plot_deregulogram(
  se_limma,
  tests,
  tests_interaction,
  factor_name,
  factor_levels,
  path_output,
  experiment,
  alpha = 0.05,
  lfc = 1,
  label_col = "Genes"
)
```

Arguments

`se_limma`	A SummarizedExperiment containing limma results in `rowData()` (e.g., output of `test_limma_customized()`).
`tests`	Character vector of main contrasts (e.g. `"IL13_vs_NoTreated"`).

`tests_interaction`
: Character vector of interaction contrasts/coefficients aligned with `tests` (e.g. `"IL13.sexmale"` or `"IL13.batchday2"`).

`factor_name`
: Character scalar, name of the interaction factor (e.g. `"sex"`, `"batch"`). Used only for titles/labels.

`factor_levels`
: Character vector of length 2 giving the factor levels in order `c(ref, other)` (e.g. `c("female","male")`).

`path_output`
: Output directory where PDFs are written.

`experiment`
: Experiment identifier used in output filenames.

`alpha`
: Numeric. Significance threshold (default 0.05).

`lfc`
: Numeric. Absolute log2FC threshold used for highlighting (default 1).

`label_col`
: Column in `rowData(se_limma)` used for point labels (default `"Genes"`).

Details

The function assumes that differential testing has already been performed and stored in `rowData(se_limma)` using the `"<stat>_<contrast>"` naming convention (e.g., `"log2FC_IL13_vs_NoTreated"`, `"p.adj_IL13_vs_NoTreated"`, `"log2FC_IL13.batchday2"`, `"p.adj_IL13.batchday2"`).

Deregulogram axes:

- y-axis: full effect in the reference level (level 1)
- x-axis: full effect in the non-reference level (level 2), computed as `log2FC_main` + `log2FC_interaction`

This plot is meaningful only when the interaction factor has exactly two levels. Use `.pp_infer_interaction_factor()` (internal) to infer the factor from `colData()` and `tests_interaction`, or pass factor name and levels explicitly.

Important: gene sets do not have to match the interaction volcano plot. The interaction volcano plot (see `plot_volcano_customized` on the `interaction_effect` object) identifies proteins with a statistically significant interaction term (e.g. `p.adj_interaction` < `alpha` and `|log2FC_interaction| >= lfc`). In contrast, the deregulogram is designed to highlight a more interpretable subset where the interaction is not only statistically significant, but also occurs in the context of a relevant condition effect (i.e., the main/full effects reach the chosen `alpha` and `lfc` thresholds). Therefore, the deregulogram typically shows a subset of the interaction-volcano hits and is not expected to reproduce the same labeled proteins one-to-one.

Deregulogram axes compare full effects between two levels of the interaction factor. For a main contrast `C` and a 2-level factor (ref vs level2), the full effect in the reference level is `log2FC_C`, while the full effect in the second level is `log2FC_C` + `log2FC_{C:factor}`. Points are colored by whether the interaction is significant and optionally labeled to emphasize proteins whose condition effect is factor-dependent.

Value

Invisibly returns `NULL`. Side effect: writes one PDF per contrast.

Examples

```
library(SummarizedExperiment)
library(S4Vectors)

## ------------------------------------------------------------
## Minimal SummarizedExperiment with main + interaction statistics
## ------------------------------------------------------------

mat <- matrix(
  rnorm(12),
  nrow = 3,
  dimnames = list(
    paste0("p", 1:3),
    paste0("s", 1:4)
  )
)

## Sample metadata (interaction factor has 2 levels)
cd <- DataFrame(
  condition = c("A","A","B","B"),
  sex       = c("female","male","female","male"),
  row.names = colnames(mat)
)

## Main and interaction contrasts
main_contrast  <- "B_vs_A"
inter_contrast <- "B_vs_A.sexmale"

## rowData with columns expected by plot_deregulogram()
rd <- DataFrame(
  Genes = c("G1","G2","G3"),
  row.names = rownames(mat)
)

## Main effect
rd[[paste0("log2FC_", main_contrast)]] <- c( 1.5,  0.2, -1.3)
rd[[paste0("p.adj_",  main_contrast)]] <- c( 0.01, 0.8,  0.03)

## Interaction effect
rd[[paste0("log2FC_", inter_contrast)]] <- c( 0.7, -0.1,  1.1)
rd[[paste0("p.adj_",  inter_contrast)]] <- c( 0.02, 0.9,  0.01)

se_limma <- SummarizedExperiment(
  assays  = list(intensity = mat),
  colData = cd,
  rowData = rd
)

## ------------------------------------------------------------
## Create deregulogram (PDF written to tempdir())
## ------------------------------------------------------------
```

```
plot_deregulogram(
  se_limma = se_limma,
  tests = main_contrast,
  tests_interaction = inter_contrast,
  factor_name = "sex",
  factor_levels = c("female", "male"),
  path_output = tempdir(),
  experiment = "EXAMPLE",
  alpha = 0.05,
  lfc = 1,
  label_col = "Genes"
)
```

`plot_volcano_customized` *Volcano plot for differential abundance results*

Description

Generates a volcano plot from differential testing results stored in a SummarizedExperiment. The function expects contrast-specific columns in `rowData(dep)` following the naming convention `"<contrast>_diff"` and either `"<contrast>_p.adj"` or `"<contrast>_p.val"`.

Usage

```
plot_volcano_customized(
  dep,
  contrast,
  label_size = 3,
  name_col = "ID",
  name_imputed = "imputed",
  add_names = TRUE,
  adjusted = TRUE,
  plot = TRUE,
  alpha = 0.05,
  lfc = 1
)
```

Arguments

`dep`	A SummarizedExperiment containing differential testing results in `rowData()`.
`contrast`	Character scalar specifying the contrast to plot. Must match the prefix used in `rowData(dep)` (e.g. `"IL13_vs_NoTreated"`).
`label_size`	Numeric. Text size for feature labels in the plot.
`name_col`	Character. Column in `rowData(dep)` used for labeling significant points (default: `"ID"`).

name_imputed	Character or NULL. Column in rowData(dep) indicating imputed features (logical). Used to color points. If NULL or not present, all features are treated as non-imputed.
add_names	Logical. If TRUE, labels significant features using **ggrepel**.
adjusted	Logical. If TRUE, uses adjusted p-values ("<contrast>_p.adj"); otherwise uses raw p-values ("<contrast>_p.val").
plot	Logical. If TRUE, returns a ggplot object; otherwise returns a data.frame with volcano statistics.
alpha	Numeric. Significance threshold for p-values.
lfc	Numeric. Absolute log2 fold-change threshold.

Details

Significance is determined internally using a log2 fold-change threshold (lfc) and a p-value or adjusted p-value threshold (alpha). Optionally, significant features can be labeled using a column from rowData(dep) (e.g. gene symbols).

Points are colored according to significance and imputation status: imputed & significant, non-imputed & significant, or non-significant. The plot title and axis annotations are derived from the contrast name.

This function does not rely on precomputed *_significant columns; significance is evaluated on the fly from diff, p.adj/p.val, alpha, and lfc.

Value

If plot = TRUE, a ggplot volcano plot. If plot = FALSE, a data.frame with log2 fold change, -log10(p-value), and significance flag.

Examples

```
library(SummarizedExperiment)
library(S4Vectors)

## Minimal SummarizedExperiment with differential results in rowData
mat <- matrix(
  rnorm(12),
  nrow = 3,
  dimnames = list(
    paste0("p", 1:3),
    paste0("s", 1:4)
  )
)

cd <- DataFrame(
  condition = c("A","A","B","B"),
  row.names = colnames(mat)
)

rd <- DataFrame(
  ID = c("Prot1","Prot2","Prot3"),
```

```
  imputed = c(FALSE, TRUE, FALSE),
  row.names = rownames(mat)
)

contrast <- "B_vs_A"

## Columns expected by plot_volcano_customized()
rd[[paste0("log2FC_", contrast)]] <- c(2.0, 0.3, -1.5)
rd[[paste0("p.adj_",  contrast)]] <- c(0.01, 0.9, 0.03)

se <- SummarizedExperiment(
  assays  = list(intensity = mat),
  colData = cd,
  rowData = rd
)

## Return volcano statistics as a data.frame (no plotting)
df <- plot_volcano_customized(
  dep = se,
  contrast = contrast,
  plot = FALSE
)

df
```

`run_proteomics_pipeline` *Run a the downstream proteomics pipeline (filter, impute, limma, reporting)*

Description

This workflow performs a reproducible downstream analysis for quantitative proteomics: (i) imports quantification and sample annotation into a SummarizedExperiment, (ii) filters features by missingness, (iii) imputes remaining missing values using a mean vs LDV (left-censored) strategy, (iv) runs differential abundance testing with limma (optionally including an interaction term), (v) organizes results into plot-ready effect objects, and (vi) optionally exports volcano plots, deregulograms and results to disk.

Usage

```
run_proteomics_pipeline(
  path_pgmatrix,
  path_annotation,
  path_output,
  level = "protein",
  type = "DIA",
  experiment,
  percent_missing,
```

```
  ldv_source = c("global", "per-condition"),
  threshold = 0.3,
  tests,
  tests_interaction,
  formula,
  reference_condition,
  paired = FALSE,
  block_effect = FALSE,
  block_var = "donor_id",
  annotation_format = c("standard", "sdrf"),
  sdrf_map = NULL,
  plots = FALSE,
  ...
)
```

Arguments

path_pgmatrix	Character scalar. Path to the quantitative matrix file.
path_annotation	Character scalar. Path to the sample annotation TSV.
path_output	Character scalar. Output directory where files may be written.
level	Character. Quantification level (e.g. "protein").
type	Character. Quantification type (e.g. "DIA").
experiment	Character scalar. Experiment identifier used in output filenames.
percent_missing	Numeric. Missingness threshold used for filtering (percentage, e.g. 50).
ldv_source	Character. One of "global" or "per-condition" controlling LDV definition.
threshold	Numeric in (0, 1]. Forwarded to impute_se(). Missingness proportion below which within-condition mean imputation is used; at or above it, LDV (left-censored) imputation is used instead.
tests	Character vector of main contrasts to test. Each element must correspond to a valid condition contrast present in the design matrix (e.g. "IL13_vs_NoTreated").
tests_interaction	Character vector defining interaction contrasts or interaction coefficients aligned with tests. Use "NA" to disable interaction testing for a given contrast. Typical values encode condition-by-factor interactions (e.g. "IL13.batchday2").
formula	A model formula specifying the design matrix for limma. Common examples include ~ 0 + condition + batch + condition:batch. All variables referenced in the formula must be present in colData().
reference_condition	Character scalar specifying the reference level for the condition variable. This level is used to relevel the design prior to fitting the limma model.
paired	Logical. Forwarded to test_limma_customized(). If TRUE, treats the design as paired/repeated-measures (see "Paired designs and blocking" section below).

block_effect Logical. Forwarded to test_limma_customized(). If TRUE, models correlation between repeated observations via limma::duplicateCorrelation (see "Paired designs and blocking" section below); the blocking variable is specified by block_var.

block_var Character. Forwarded to test_limma_customized(). Name of the colData column to use as the blocking variable when block_effect = TRUE. Defaults to "donor_id". Ignored when block_effect = FALSE.

annotation_format Character. One of "standard" (default) or "sdrf". Use "sdrf" when path_annotation is an SDRF-Proteomics file rather than a plain annotation TSV with file/sample_name/condition columns; see sdrf_map.

sdrf_map Named list, only used when annotation_format = "sdrf". Maps SDRF column names (which are not fixed by the standard) to the roles the pipeline needs, e.g. list(condition = "factor value[treatment]", batch = "comment[batch]",donor_id = "characteristics[individual]"). At minimum condition must be supplied. Ignored when annotation_format = "standard".

plots Logical. If TRUE, exports plots/tables to path_output.

... Additional arguments forwarded only to test_limma_customized() (beyond test, test_interaction, design_formula, ref_condition, paired and block_effect, which are already explicit parameters above).

Value

A named list with:

se_raw Raw SummarizedExperiment imported from files.

se_filt Filtered SummarizedExperiment.

removed A data.frame describing filtered-out features.

se_imp Imputed SummarizedExperiment.

effects A named list keyed by main contrast (the values of tests). Each element is itself a list of three SummarizedExperiment objects: all_common_effect, common_effect and interaction_effect. Per-contrast statistics are stored in their rowData() using the "<stat>_<contrast>" naming convention (e.g. log2FC_IL13_vs_NoTreated). This element is always returned, independently of plots, so a contrast can be accessed with $, e.g. res$effects$IL13_vs_NoTreated$all_common_effe

Input files

path_pgmatrix Path to a quantitative matrix file produced by the upstream quantification tool (e.g., a TSV). Rows represent quantified features (e.g. protein groups) and columns represent samples.

path_annotation Path to a tab-delimited sample annotation file describing the experimental design. At minimum it must encode sample-to-condition mapping. Additional columns are required (batch, replicate, block). Accepts either a standard annotation TSV (default, annotation_format = "standard") or an SDRF-Proteomics file (annotation_format = "sdrf", with column roles supplied via sdrf_map).

Main steps

1. **Import**: creates a raw SummarizedExperiment (`se_raw`).
2. **Missingness filter**: removes features not sufficiently observed across samples/conditions according to `percent_missing`. Filtered-out features are returned and may be written to disk.
3. **Imputation**: imputes remaining missing values using a two-step strategy: within-condition mean imputation when missingness is below a threshold and LDV (left-censored) imputation when missingness is above the threshold. The imputation method is controlled by `ldv_source` and `threshold`.
4. **Differential testing**: calls `test_limma_customized` to fit a limma model and evaluate user-specified contrasts. Interaction testing is optional.
5. **Organize outputs**: splits the limma results into plot-ready effect objects using `.pp_organize_effects`:
   - `all_common_effect`: main-effect statistics for all proteins (no interaction filtering).
   - `common_effect`: proteins not significant for the interaction term (main effect stable).
   - `interaction_effect`: proteins significant for the interaction term.
6. **Optional reporting** (`plots = TRUE`): exports volcano plots and tables for the three effect types above, and generates deregulograms when an interaction factor with exactly two levels is detected (e.g., female vs male, day1 vs day2, ...).

Differential testing inputs

The limma step requires the following explicit parameters:

`tests` Character vector of main contrasts to test, using a consistent naming scheme (e.g. `"A_vs_B"`). These define the primary condition effects.

`tests_interaction` Character vector aligned with `tests` defining interaction contrasts or interaction coefficients (or `"NA"` to disable interaction testing). These represent factor-dependent differences in the condition effect (e.g. condition-by-batch).

`formula` A model formula describing the design matrix (e.g. `~ 0 + condition + batch + condition:batch`). Variables referenced in the formula must be present in `colData(se)`.

`reference_condition` Character scalar indicating the reference level for `condition`. The function will relevel `colData(se)$condition` to this value before model fitting.

Paired designs and blocking

`paired` Logical (forwarded to `test_limma_customized`). If `TRUE`, the model is treated as a paired / repeated-measures design by ensuring `replicate` is included in the design formula. In the current implementation, if `replicate` is not already present in `formula`, it is added automatically. This preserves any additional covariates already present in the formula.

`block_effect` Logical (forwarded to `test_limma_customized`). If `TRUE`, correlation between repeated observations is modeled using `limma::duplicateCorrelation` and `lmFit(..., block=..., correlation=...)`. The blocking variable is specified by `block_var` (default: `"donor_id"`). Use this when there is a known blocking factor (e.g., donor/subject) inducing correlation across samples.

`block_var` Character (forwarded to `test_limma_customized`). Name of the `colData` column to use as the blocking variable for `duplicateCorrelation` when `block_effect = TRUE`. Defaults to `"donor_id"`. Any column present in the annotation file can be used (e.g., `"batch"`, `"patient_id"`). Ignored when `block_effect = FALSE`.

Volcano vs deregulogram (interpretation note)

Interaction volcano plots highlight proteins with significant interaction coefficients (i.e., where the condition effect depends on a second factor). Deregulograms are more restrictive: they visualize full effects across the two levels of the interaction factor and emphasize proteins where interaction occurs in the context of a relevant condition effect (effect-size and FDR thresholds). Therefore, the labeled proteins in the deregulogram are not expected to match one-to-one with those in the interaction volcano plot.

See Also

test_limma_customized, plot_volcano_customized, plot_deregulogram

Examples

```
## Load package
library(ProteinBatcher)

## ------------------------------------------------------------------
## Input files shipped with the package
## ------------------------------------------------------------------

path_annotation <- system.file(
  "extdata", "annotation_HaCaT.tsv",
  package = "ProteinBatcher"
)

path_pgmatrix <- system.file(
  "extdata",
  "2024MK017_HaCaT_Stimulation_1to24_Astral_report.pg_matrix.tsv",
  package = "ProteinBatcher"
)

## Output directory
path_output <- tempdir()

## ------------------------------------------------------------------
## Workflow parameters (single contrast example)
## ------------------------------------------------------------------

experiment <- "HaCaT"
percent_missing <- 50

formula <- ~ 0 + condition + batch + condition:batch

tests <- c("IL13_vs_NoTreated")
tests_interaction <- c("IL13.batchday2")

reference_condition <- "NoTreated"

## ------------------------------------------------------------------
## Run downstream proteomics workflow
```

```
## ------------------------------------------------------------

res <- run_proteomics_pipeline(
  path_pgmatrix        = path_pgmatrix,
  path_annotation      = path_annotation,
  path_output          = path_output,
  tests                = tests,
  tests_interaction    = tests_interaction,
  formula              = formula,
  reference_condition  = reference_condition,
  percent_missing      = percent_missing,
  ldv_source           = "per-condition",
  experiment           = experiment,
  plots                = FALSE
)

## Inspect outputs
names(res)
res$se_imp

## `effects` is keyed by contrast, so it can be accessed with `$`:
names(res$effects)
res$effects$IL13_vs_NoTreated$all_common_effect
```

`test_limma_customized` *Differential abundance testing with limma (imputation-aware FDR)*

Description

Performs differential testing on an imputed SummarizedExperiment using **limma**. The function fits a linear model defined by `design_formula`, applies user-specified contrasts (`test`) and, optionally, interaction-related contrasts (`test_interaction`), and stores the resulting statistics in `rowData(se)`.

Usage

```
test_limma_customized(
  se,
  type = c("manual"),
  test = NULL,
  test_interaction = "NA",
  design_formula = stats::formula(~0 + condition),
  ref_condition = NULL,
  paired = FALSE,
  block_effect = FALSE,
  block_var = "donor_id"
)
```

Arguments

se	A SummarizedExperiment. The assay is taken from SummarizedExperiment::assay(se).
type	Character. Currently only "manual" is supported (kept for API compatibility).
test	Character vector of contrasts to test for the main effect. Contrasts must be supplied in the "A_vs_B" format (internally translated to "A - B" for **limma**).
test_interaction	Character vector defining interaction-related tests. Use "NA" (default) to skip interaction testing. If provided (not "NA"), results for these contrasts will also be added to rowData(se).
design_formula	A model formula describing the linear model. Default is ~ 0 + condition. The function expects condition to be present in colData(se) and will relevel it to ref_condition. If paired = TRUE and replicate is not present in the formula, replicate is added.
ref_condition	Character scalar. Reference level for condition. Must be a level present in colData(se)$condition.
paired	Logical. If TRUE, replicate is added to the design (if absent) to support paired/repeated-measure designs for technical replicates.
block_effect	Logical. If TRUE, fits limma with a blocking factor using limma::duplicateCorrelation() and the colData column specified by block_var.
block_var	Character. Name of the colData(se) column to use as the blocking variable for duplicateCorrelation when block_effect = TRUE. Defaults to "donor_id". The column must exist in colData(se) and have at least two levels. Ignored when block_effect = FALSE.

Details

A key feature of this customized implementation is that adjusted p-values (BH-FDR) are recomputed after excluding features that were imputed as *LDV* in both groups of a tested contrast, using an imputation_map stored in metadata(se). This prevents artificially significant results driven by LDV/LDV imputations.

When paired = TRUE, the function ensures the design includes replicate by updating the right-hand side of design_formula (it does not overwrite other covariates such as batch). When block_effect = TRUE, the function fits the model using limma::duplicateCorrelation() with the column specified by block_var (default: donor_id) as the blocking variable.

Results are written into rowData(se) as wide columns per contrast, using prefixes corresponding to each tested comparison. For each contrast, the following statistics are stored (naming depends on internal helpers):

- diff: log2 fold-change estimate (logFC)
- CI.L, CI.R: confidence interval bounds (if computed)
- p.val: raw p-value
- p.adj: BH-adjusted p-value, recomputed after excluding LDV/LDV

Interaction handling is optional and depends on test_interaction. In the associated proteomics workflow, interaction tests are used to distinguish "common" effects from "interaction" effects (e.g., condition-by-sex or condition-by-batch interactions).

Value

The input SummarizedExperiment with additional columns appended to `rowData(se)` containing differential testing results.

See Also

lmFit, makeContrasts, contrasts.fit, eBayes, topTable, duplicateCorrelation

Examples

```
library(SummarizedExperiment)

## Toy proteomics matrix: 4 proteins x 4 samples
mat <- matrix(
  c(
    10, 11,  9, 10,
    20, 21, 19, 20,
    5,  NA,  6,  7,
    NA, NA, NA, NA
  ),
  nrow = 4, byrow = TRUE,
  dimnames = list(
    paste0("p", 1:4),
    paste0("s", 1:4)
  )
)

## Sample metadata
cd <- DataFrame(
  condition = c("A", "A", "B", "B"),
  batch     = c("d1","d1","d2","d2"),
  replicate = c("r1","r2","r1","r2"),
  donor_id  = c("1","1","1","1"),
  label     = colnames(mat),
  row.names = colnames(mat)
)

## Feature metadata
rd <- DataFrame(
  name  = rownames(mat),
  Genes = paste0("G", 1:4),
  row.names = rownames(mat)
)

se <- SummarizedExperiment(
  assays  = list(intensity = mat),
  colData = cd,
  rowData = rd
)

## Imputation map required by the FDR logic
imp_map <- matrix(
```

```
  "none", nrow = nrow(mat), ncol = ncol(mat),
  dimnames = dimnames(mat)
)
metadata(se)$imputation_map <- imp_map

## Run differential testing (main effect only)
se_out <- test_limma_customized(
  se = se,
  test = "B_vs_A",
  test_interaction = "NA",
  design_formula = ~ 0 + condition + batch,
  ref_condition = "A"
)

## Results are appended to rowData(se_out)
rowData(se_out)
```

Index

contrasts.fit, *18*

duplicateCorrelation, *18*

eBayes, *18*

filter_se_missing, 2

ggplot, *10*

impute_se, 4

lmFit, *18*

makeContrasts, *18*

plot_deregulogram, 6, *15*
plot_volcano_customized, 9, *15*

run_proteomics_pipeline, 11

test_limma_customized, *14, 15*, 16
topTable, *18*