Package 'exposomeSC'

November 25, 2026

Title Cell-Type-Resolved Exposome-Wide Association Studies for Single-Cell RNA-seq

Version 0.99.4

Description Tests donor-level environmental exposures for association with gene expression within cell types in single-cell RNA-seq cohorts (single-cell exposome-wide association studies, sc-ExWAS). The package extends SingleCellExperiment to hold donor-level exposure measurements alongside the cells, and aggregates raw counts by donor and cell type so that the donor, not the cell, is the unit of inference (Squair et al. 2021 <doi:10.1038/s41467-021-25960-2>). The default analysis fits robust edgeR quasi-likelihood models with exposure-independent expression filtering, complete-transcriptome library sizes, design-rank checks and multiplicity correction across all genes and cell types; DESeq2 and voom-dream backends are also available. Further functions cover several exposures at once, donor-level cell-type composition, dose-response curves, exposure mixtures, meta-analysis across cohorts, power calculation, gene set testing with fgsea and imputation below the limit of detection. Exploratory functions estimate cross-omic networks within a cell type. All results are associations; the package does not identify causal effects.

License Artistic-2.0

Depends R (>= 4.6.0)

Imports methods, BiocGenerics, Matrix, S4Vectors (>= 0.32.0), SummarizedExperiment, SingleCellExperiment, stats, BiocParallel, MASS, edgeR (>= 4.10.0), limma, scrapper (>= 1.6.0), withr

Suggests DESeq2, variancePartition, fgsea, lme4, lmerTest, pbkrtest, mgcv, qgcomp, glasso, Biobase, coglasso (>= 1.1.0), igraph, ggraph, tidygraph, ggplot2, metafor, ppcor, testthat (>= 3.1.7), knitr, rmarkdown, BiocStyle

Config/testthat/edition 3

biocViews SingleCell, RNASeq, GeneExpression, DifferentialExpression, Transcriptomics, StatisticalMethod, Regression, MultipleComparison, GeneSetEnrichment, Network, NetworkInference

URL https://github.com/CuiweiG/exposomeSC

BugReports https://github.com/CuiweiG/exposomeSC/issues

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

Collate 'AllClasses.R' 'AllGenerics.R' 'below-lod.R' 'build-scee.R' 'bridge.R' 'cell-coupling.R' 'utils.R' 'composition.R' 'data-docs.R' 'dose-response.R' 'enrichment.R' 'exposomeSC-package.R' 'gam-dr.R' 'glmm-inference.R' 'interaction.R' 'meta-analysis.R' 'methods.R' 'mixture.R' 'sc-exwas.R' 'multi-exwas.R' 'network-accessors.R' 'network-utils.R' 'network-build.R' 'network-compare.R' 'network-plots.R' 'network-simulate.R' 'power.R' 'pseudobulk.R' 'qgcomp-mixture.R'

git_url https://git.bioconductor.org/packages/exposomeSC

git_branch devel

git_last_commit 03abd8f

git_last_commit_date 2026-11-21

Repository Bioconductor 3.25

Date/Publication 2026-11-24

Author Cuiwei Gao [aut, cre]

Maintainer Cuiwei Gao <48gaocuiwei@gmail.com>

Contents

exposomeSC-package . . . 3
as_scee . . . 5
build_scee . . . 6
CelltypeNetwork-class . . . 7
CelltypeNetworkResult . . . 7
CelltypeNetworkResult-class . . . 8
donor_pseudobulk . . . 9
estimate_power . . . 11
exposureData . . . 12
exposure_impute_lod . . . 14
hlca_airway_subset . . . 16
incov_pbmc_subset . . . 18
network-accessors . . . 20
NetworkComparison . . . 23
NetworkComparison-class . . . 24
plot_celltype_network . . . 24
plot_edge_stability . . . 26
plot_network_comparison . . . 27
run_celltype_network . . . 28
run_cell_coupling . . . 31
run_comparative_network . . . 35
run_dose_response . . . 36
run_dose_response_gam . . . 37
run_exposure_composition . . . 39

run_exposure_network 42
run_gsea 46
run_interaction_test 47
run_meta_exwas 49
run_mixture_qgcomp 50
run_multi_exwas 53
run_sc_exwas 54
run_sc_exwas_glmm 58
run_sc_exwas_pb_offset 61
run_sc_mixture 63
seurat_to_exposure 65
simulate_crossomic_network 66
SingleCellExposomeExperiment-class 68

Index **70**

`exposomeSC-package` *exposomeSC: Cell-Type-Resolved Exposome-Wide Association Studies for Single-Cell RNA-seq*

Description

Tests donor-level exposures for association with gene expression within cell types, with the donor as the unit of inference. Results are associations; the package does not identify causal effects.

Core analysis

`build_scee` Build a SingleCellExposomeExperiment from SCE + exposure matrix

`run_sc_exwas` Cell-type-specific ExWAS via pseudobulk (edgeR, DESeq2 or voom-dream backend)

`run_sc_exwas_pb_offset` Negative binomial model of the pseudobulk counts with a donor offset

`run_sc_exwas_glmm` Mixed model of the cells with donor random effects, linear by default

`run_multi_exwas` Batch ExWAS across multiple exposures with global FDR

`donor_pseudobulk` Donor pseudobulk counts per cell type as SummarizedExperiment objects

Dose-response and mixtures

`run_dose_response` Polynomial dose-response (AIC selection)

`run_dose_response_gam` GAM dose-response via mgcv

`run_sc_mixture` Quantile-scored mixture screening

`run_mixture_qgcomp` Quantile g-computation with qgcomp

`run_interaction_test` Exposure x celltype interaction F-test

Cell-level coupling

`run_cell_coupling` Gene-protein coupling via meta-regression with confounders

Enrichment and meta-analysis

`run_gsea` Pathway enrichment via fgsea

`run_meta_exwas` Cross-cohort meta-analysis (fixed/random effects)

`estimate_power` Power/sample size calculator

Composition

`run_exposure_composition` Exposure association with donor-level cell-type composition

Cross-omic networks (exploratory)

`run_celltype_network` Cell-type-resolved network of transcripts and donor-level molecular measurements

`run_exposure_network` Network restricted to exposure-associated features

`run_comparative_network` Descriptive comparison of networks across cell types

`simulate_crossomic_network` Simulated data with a known network

`plot_celltype_network`, `plot_network_comparison`, `plot_edge_stability` Plots

Preprocessing

`exposure_impute_lod` Below-LOD imputation

`as_scee` Convert from rexposome

`seurat_to_exposure` Extract from Seurat

Author(s)

Maintainer: Cuiwei Gao <48gaocuiwei@gmail.com>

References

Squair JW et al. (2021). Confronting false discoveries in single-cell differential expression. *Nat Commun* 12:5692. doi:10.1038/s41467021259602

Aitchison J (1986). *The Statistical Analysis of Compositional Data*. Chapman and Hall.

See Also

Useful links:

- https://github.com/CuiweiG/exposomeSC
- Report bugs at https://github.com/CuiweiG/exposomeSC/issues

as_scee *Convert rexposome ExposomeSet to exposomeSC*

Description

Bridges the bulk exposome ecosystem with single-cell analysis by extracting the exposure matrix from an `ExposomeSet` (rexposome) and combining it with a `SingleCellExperiment`.

Usage

```
as_scee(exposome_set, sce, sample_col, exposures = NULL)
```

Arguments

`exposome_set`	An `ExposomeSet` from rexposome.
`sce`	A `SingleCellExperiment` with matching donor/sample IDs.
`sample_col`	Character. Column in `colData(sce)` with sample IDs matching `sampleNames(exposome_set)`.
`exposures`	Character vector (optional). Subset of exposures to include; a name that the ExposomeSet does not have is an error. Default: all.

Details

This function enables a common workflow: researchers have existing bulk exposome data in rexposome format and want to add single-cell resolution. The exposure matrix and metadata are extracted from the `ExposomeSet` and combined with the SCE to create an SCEE.

Every sample in `colData(sce)[[sample_col]]` must be present in the ExposomeSet; ExposomeSet samples without cells are dropped.

Value

A `SingleCellExposomeExperiment`.

References

Hernandez-Ferrer C et al. (2019). Comprehensive study of the exposome and omic data using rexposome Bioconductor packages. *Bioinformatics* 35:5344-5345.

Examples

```
if (requireNamespace("Biobase", quietly = TRUE)) {
    ## as_scee() needs only what an ExposomeSet is: a Biobase eSet whose
    ## assayData element "exp" holds exposures (rows) by samples
    ## (columns). A minimal eSet of that shape stands in here; an object
    ## from rexposome::loadExposome() is converted the same way.
    setClass("ExposomeSetExample", contains = "eSet")
    samples <- paste0("D", 1:4)
    exposures <- matrix(c(12, 25, 8, 30, 0.4, 0.9, 0.2, 1.1), nrow = 2,
        byrow = TRUE, dimnames = list(c("PM2.5", "Pb"), samples))
    es <- methods::new("ExposomeSetExample",
        assayData = Biobase::assayDataNew("environment", exp = exposures),
        phenoData = Biobase::AnnotatedDataFrame(
            data.frame(sex = c(0, 1, 0, 1), row.names = samples)),
```

```
        featureData = Biobase::AnnotatedDataFrame(
            data.frame(family = c("air", "metal"),
                       row.names = c("PM2.5", "Pb"))))
    sce <- SingleCellExperiment::SingleCellExperiment(
        assays = list(counts = matrix(1L, nrow = 5, ncol = 8,
            dimnames = list(paste0("G", 1:5), paste0("c", 1:8)))),
        colData = S4Vectors::DataFrame(
            cell_id = paste0("c", 1:8),
            donor_id = rep(samples, each = 2)))
    scee <- as_scee(es, sce, sample_col = "donor_id")
    exposureData(scee)
    exposureInfo(scee)
}
```

`build_scee` *Build a SingleCellExposomeExperiment*

Description

Combines a `SingleCellExperiment` with sample-level exposure measurements into a unified container for sc-ExWAS.

Usage

```
build_scee(sce, exposure_matrix, sample_col, exposure_info = NULL)
```

Arguments

`sce`	A `SingleCellExperiment`.
`exposure_matrix`	Numeric matrix. Rows = samples, columns = exposure variables, with unique row and column names. The row names must be exactly the sample IDs in `colData(sce)[[sample_col]]`: every sample of the SCE must have a row, and rows for samples absent from the SCE are an error. `NA` is allowed; `NaN` and infinite values are not.
`sample_col`	Character. Column in `colData(sce)` containing the sample/donor ID. Its name is stored in the object (see `sampleColumn`), and the analysis functions read each cell's donor from this column.
`exposure_info`	`DataFrame` (optional). Metadata for exposures (for example family, unit and LOD). It must contain an `exposure` column equal to `colnames(exposure_matrix)`, in the same order.

Value

A `SingleCellExposomeExperiment`.

Examples

```
library(SingleCellExperiment)
library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(2000, 5), nrow = 50,
    dimnames = list(paste0("G", 1:50), paste0("c", 1:40)))
sce <- SingleCellExperiment(
    assays = list(counts = counts),
    colData = DataFrame(
        cell_id = paste0("c", seq_len(40)),
        donor_id = rep(paste0("D", 1:4), each = 10),
        cell_type = rep(c("Mono", "NK"), 20)))
exp_mat <- matrix(rnorm(12), nrow = 4,
    dimnames = list(paste0("D", 1:4),
                    c("PM2.5", "Pb", "BPA")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
scee
```

CelltypeNetwork-class *CelltypeNetwork: virtual base class for cell-type networks*

Description

CelltypeNetwork: virtual base class for cell-type networks

Examples

```
showClass("CelltypeNetwork")
```

CelltypeNetworkResult *Create a CelltypeNetworkResult*

Description

Wraps a network estimated elsewhere, or built by hand, in a `CelltypeNetworkResult` so that it can be compared with `run_comparative_network` and plotted. The networks returned by `run_celltype_network` and `run_exposure_network` are already of this class.

Usage

```
CelltypeNetworkResult(
  precision,
  adjacency,
  node_info,
  celltype,
  method = "precomputed",
  stability = NULL,
  metadata = list()
)
```

Arguments

precision	Numeric square matrix: the precision (inverse covariance) matrix, one row and column per feature.
adjacency	Symmetric matrix of edge indicators (1 = edge, 0 = no edge) with the dimensions of precision and a zero diagonal; a logical matrix is stored as 0/1 integers.
node_info	DataFrame or data.frame with one row per feature, in matrix order, a feature column without missing values and an omic_layer column, which the summaries, plots and comparisons read; the summaries and plots recognise the layers "transcript" and "metabolite". Row and column names of the matrices, where present, must equal feature, so that a misordered table is an error.
celltype	Character scalar: the cell type.
method	Character scalar: how the network was estimated. Default "precomputed".
stability	Optional matrix of edge selection frequencies between 0 and 1 with the dimensions of adjacency; NULL (default) stores none.
metadata	List of further information, for example n_donors, the number of donors as a whole number, which the show method and plot_celltype_network print.

Value

A valid CelltypeNetworkResult.

See Also

network-accessors

Examples

```
features <- c("G1", "G2", "M1")
precision <- diag(3)
precision[1, 3] <- precision[3, 1] <- -0.4
adjacency <- (precision != 0) * 1L
diag(adjacency) <- 0L
dimnames(precision) <- dimnames(adjacency) <- list(features, features)
network <- CelltypeNetworkResult(
    precision = precision,
    adjacency = adjacency,
    node_info = S4Vectors::DataFrame(
        feature = features,
        omic_layer = c("transcript", "transcript", "metabolite")),
    celltype = "Monocyte",
    metadata = list(n_donors = 20L))
network
```

CelltypeNetworkResult-class

CelltypeNetworkResult: cell-type-resolved cross-omic network

Description

Stores the estimated precision matrix, adjacency matrix, stability scores, and node metadata from run_celltype_network and run_exposure_network. Use the accessors in network-accessors to extract them, and CelltypeNetworkResult() to wrap a network estimated elsewhere.

Usage

```
## S4 method for signature 'CelltypeNetworkResult'
show(object)
```

Arguments

object	A CelltypeNetworkResult object.

Value

The show method prints a summary and returns NULL invisibly.

Slots

precision_matrix Numeric matrix. Estimated block precision (inverse covariance) matrix.

adjacency_matrix Numeric matrix. Symmetric binary edge indicators (1 = edge present, 0 = absent) with the dimensions of precision_matrix and a zero diagonal.

stability_scores Numeric matrix. Edge-level selection frequencies (0-1) across the XStARS subsamples at the selected penalties, for coglasso networks built with stability = TRUE. Empty otherwise.

node_info DataFrame. Maps each node (row/col index) to its omic layer ("transcript" or "metabolite"), feature name, and optional module/community assignment. It has one row per node, in matrix order; when the matrices have row or column names, these must equal its feature column.

celltype Character. The cell type this network represents.

method Character. Estimation method: "coglasso" or "block_glasso" from run_celltype_network(), "exposure_restricted" from run_exposure_network(), or "precomputed" for a network supplied by the user.

metadata List. Additional parameters: n_donors, n_features, lambda values, exposure, covariates.

See Also

network-accessors, CelltypeNetworkResult

Examples

```
showClass("CelltypeNetworkResult")
```

---

donor_pseudobulk *Donor pseudobulk counts for each cell type*

Description

Sums the raw counts of each cell type within each donor and returns one SummarizedExperiment per cell type, with the donors' exposure data as column data. These are the pseudobulk counts that run_sc_exwas and the other pseudobulk functions of the package aggregate internally, so they can be passed to other Bioconductor tools, inspected, or plotted.

Usage

```
donor_pseudobulk(
  x,
  celltype_col = "cell_type",
  celltypes = NULL,
  min_cells = 10L
)
```

Arguments

x	A SingleCellExposomeExperiment with a "counts" assay. Each cell's donor is read from the colData column named by sampleColumn(x).
celltype_col	Character. Column in colData containing cell type annotations. Default "cell_type".
celltypes	Character vector (optional). Cell types to aggregate. Default: all non-missing labels, sorted.
min_cells	Integer. A donor is included for a cell type only when it has at least this many cells of that type. Default 10, the default of run_sc_exwas.

Details

The summation is done by scrapper::aggregateAcrossCells(). Donors with missing exposure values are kept here; the analysis functions drop them, and filter genes, when they fit a model, so a fitted model may use fewer donors and genes than the object returned here. Model diagnostics (metadata() of the run_sc_exwas result) list the donors and gene counts actually used.

Value

A SimpleList named by cell type. Each element is a SummarizedExperiment with one column per included donor and

- an assay "counts" with the summed raw counts (features by donors);
- rowData copied from x;
- colData with the donor identifier (donor_id), the cell type (cell_type), the number of cells summed (n_cells) and one column per variable of exposureData(x);
- metadata recording the cell type, min_cells and the colData columns used for donors and cell types.

Cell types in which no donor reaches min_cells are left out and listed in metadata(result)$skipped.

References

Lun ATL, Kancherla J (2023). Powering single-cell analyses in the browser with WebAssembly. *Journal of Open Source Software* 8(89):5603. doi:10.21105/joss.05603

See Also

run_sc_exwas

Examples

```
library(SingleCellExperiment)
library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(2000, 5), nrow = 20,
    dimnames = list(paste0("G", 1:20), paste0("c", 1:100)))
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = DataFrame(donor_id = rep(paste0("D", 1:5), each = 20),
        cell_type = rep(c("Mono", "NK"), 50)))
exp_mat <- matrix(rnorm(5), ncol = 1,
    dimnames = list(paste0("D", 1:5), "E1"))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
pb <- donor_pseudobulk(scee, min_cells = 5L)
pb
SummarizedExperiment::colData(pb[["Mono"]])
```

`estimate_power` *Power analysis for sc-ExWAS study design*

Description

Approximates the power to detect a donor-level exposure effect on one gene in one cell type at the pseudobulk level, for a range of donor numbers.

Usage

```
estimate_power(
  n_donors = seq(10, 100, by = 10),
  effect_size = 0.5,
  n_genes = 5000L,
  n_celltypes = 5L,
  alpha = 0.05,
  dispersion = 0.1,
  base_mean = 100,
  exposure_sd = 1
)
```

Arguments

`n_donors`	Integer vector. Donor sample sizes to evaluate. Default: `seq(10, 100, by = 10)`.
`effect_size`	Numeric. Expected log2 fold change per unit exposure. Default 0.5.
`n_genes`	Integer. Number of genes tested (for multiple testing correction). Default 5000.
`n_celltypes`	Integer. Number of cell types tested. Default 5.
`alpha`	Numeric. Family-wise significance level; the per-test level is `alpha / (n_genes * n_celltypes)`. Default 0.05.
`dispersion`	Numeric. Negative-binomial dispersion ϕ of donor pseudobulk counts, whose variance is $\mu + \phi\mu^2$. Default 0.1.
`base_mean`	Numeric. Mean donor pseudobulk count μ of the gene. Default 100; `Inf` gives the large-count limit.

`exposure_sd` Numeric. Standard deviation of the exposure variable across donors. Default 1 (standardised). For a binary exposure that a proportion q of the donors have, the standard deviation is $\sqrt{q(1-q)}$, 0.5 for two groups of equal size; `effect_size` is then the log2 fold change between the groups.

Details

For a negative-binomial GLM with a log link, the large-sample standard error of the exposure coefficient on the $\log_2$ scale is

$$SE = \frac{\sqrt{1/\mu + \phi}}{\log(2)\, s_x \sqrt{n}},$$

where s_x is `exposure_sd` and n the number of donors. Power is that of a two-sided Wald test at the Bonferroni per-test level. The approximation treats the dispersion as known and uses the normal distribution, so it is optimistic for small numbers of donors; simulate from the planned design when an accurate figure matters. Bonferroni is conservative relative to Benjamini-Hochberg control of the false discovery rate.

Pseudobulk inference has donors, not cells, as its independent units. More cells per donor raise μ and so shrink the $1/\mu$ term, but they do not increase n.

Value

A `DataFrame` with columns: n_donors, power, effect_size, exposure_sd, n_tests, alpha_adjusted, dispersion, base_mean.

References

Squair JW et al. (2021). Confronting false discoveries in single-cell differential expression. *Nat Commun* 12:5692.

Examples

```
pw <- estimate_power(n_donors = seq(10, 50, by = 10),
    effect_size = 0.5, n_genes = 2000, n_celltypes = 3)
pw
```

`exposureData` *Accessors for SingleCellExposomeExperiment objects*

Description

Get and set the donor-level exposure data of a `SingleCellExposomeExperiment` and the column of `colData` that links cells to donors.

Usage

```
exposureData(x)

exposureInfo(x)

cellSampleMap(x)
```

```
sampleColumn(x)

sampleColumn(x) <- value

exposureVariables(x)

exposureData(x) <- value

exposureInfo(x) <- value

## S4 method for signature 'SingleCellExposomeExperiment'
exposureData(x)

## S4 method for signature 'SingleCellExposomeExperiment'
exposureInfo(x)

## S4 method for signature 'SingleCellExposomeExperiment'
cellSampleMap(x)

## S4 method for signature 'SingleCellExposomeExperiment'
sampleColumn(x)

## S4 replacement method for signature 'SingleCellExposomeExperiment'
sampleColumn(x) <- value

## S4 method for signature 'SingleCellExposomeExperiment'
exposureVariables(x)

## S4 replacement method for signature 'SingleCellExposomeExperiment'
exposureData(x) <- value

## S4 replacement method for signature 'SingleCellExposomeExperiment'
exposureInfo(x) <- value
```

Arguments

x	A SingleCellExposomeExperiment.
value	For exposureData<-, a numeric matrix with one row per sample, named by sample ID, and one column per exposure; the rows of exposureInfo(x) follow the new columns, so an exposure can be added (its metadata row is filled with NA), dropped or reordered. For exposureInfo<-, a DataFrame whose exposure column equals colnames(exposureData(x)); for sampleColumn<-, the name of a colData column whose sample IDs match the rows of exposureData(x).

Value

`exposureData()` Numeric matrix, samples by exposures.

`exposureInfo()` DataFrame with one row per exposure.

`exposureVariables()` Character vector of exposure names.

`sampleColumn()` Character scalar: the colData column holding each cell's sample (donor) ID. The analysis functions read each cell's donor from this column.

`cellSampleMap()` `DataFrame` with columns `cell_id` (the cell names) and `sample_id`, derived from that column.

Replacement functions The modified object, after a validity check.

Examples

```
library(SingleCellExperiment)
counts <- matrix(1L, nrow = 5, ncol = 12,
    dimnames = list(paste0("G", 1:5), paste0("c", 1:12)))
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = S4Vectors::DataFrame(
        cell_id = paste0("c", 1:12),
        donor_id = rep(paste0("D", 1:4), each = 3)))
exp_mat <- matrix(c(12, 25, 8, 30, 45, 32, 67, 51), nrow = 4,
    dimnames = list(paste0("D", 1:4), c("PM2.5", "age")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")

exposureVariables(scee)
exposureData(scee)
exposureInfo(scee)
sampleColumn(scee)
head(cellSampleMap(scee))

## Replace the exposure matrix, here rescaling PM2.5
exp_new <- exposureData(scee)
exp_new[, "PM2.5"] <- exp_new[, "PM2.5"] / 10
exposureData(scee) <- exp_new
exposureData(scee)[, "PM2.5"]
```

---

exposure_impute_lod *Impute exposure values below the limit of detection*

Description

Environmental exposure measurements often contain values reported only as below the limit of detection (LOD), that is, left-censored values. This function replaces them by substitution, by regression on order statistics or by multiple imputation from a censored log-normal model. Missing values are kept apart from censored ones and are never imputed.

Usage

```
exposure_impute_lod(
  exposure_matrix,
  lod,
  method = c("lod_sqrt2", "lod_2", "ros", "multiple"),
  censored = NULL,
  M = 5L,
  verbose = TRUE
)
```

Arguments

`exposure_matrix`	Numeric matrix. Rows = samples, columns = exposure variables. In the exposures named in `lod`, an infinite value that is not censored is an error.
`lod`	Named numeric vector of positive, finite detection limits, one per exposure. Names must match column names of `exposure_matrix`: a name that is not a column is ignored with a warning, and the function stops when no name is. Exposures not in `lod` are left unchanged.
`method`	Character. Imputation method: `"lod_sqrt2"` Substitution with LOD / sqrt(2). `"lod_2"` Substitution with LOD / 2. `"ros"` Regression on order statistics (ROS) for one detection limit per exposure; see Details. `"multiple"` Multiple imputation from a log-normal model fitted by censored maximum likelihood to a bootstrap sample of the data, as proposed by Lubin et al. (2004); returns `M` completed matrices.
`censored`	`NULL` (default) or a logical matrix with the same dimensions as `exposure_matrix`, `TRUE` where a value was reported as below the LOD. The numeric value at a censored position is ignored and may be `NA`. With `NULL`, a value is censored when it is not `NA` and is at or below the LOD. Either way, `NA` values that are not marked as censored are missing and stay `NA`.
`M`	Integer. Number of imputations (only for `method = "multiple"`). Default 5.
`verbose`	Logical. Report the numbers of censored and missing values per exposure. Default TRUE.

Details

Substitution is simple, but it distorts variances and associations, increasingly so as the censored fraction grows (Lubin et al. 2004).

ROS. With a single detection limit every censored value lies below every detected value, so the censored values take the lowest ranks among the non-missing values of the exposure. Blom plotting positions $(i - 3/8)/(n + 1/4)$ are computed for all n non-missing values, the logarithms of the detected values are regressed on the normal quantiles of their plotting positions, and each censored value is replaced by the value of the fitted line at one of the lowest plotting positions, capped just below the LOD (Helsel 2012). ROS reproduces the distribution of the censored values, not their individual values: the imputed values are assigned to the censored samples in row order, so they carry no information on which censored sample had the larger true value. Use the ROS values for summaries of the distribution; for association analyses prefer `method = "multiple"`. ROS is not the reverse Kaplan-Meier estimator, which is not implemented here.

Multiple imputation. For each of the `M` imputations the non-missing values of an exposure are resampled with replacement, the mean and standard deviation of the log values are estimated by maximum likelihood with the censored values contributing $P(X \leq \mathrm{LOD})$, and every censored value is drawn from the fitted log-normal distribution truncated to $(0, \mathrm{LOD}]$. The bootstrap step carries the uncertainty of the fitted distribution into the imputations (Lubin et al. 2004). Analyse each completed matrix and combine the estimates with Rubin's rules (Rubin 1987). The draws use the random-number stream of the session, so call `set.seed()` first for reproducible imputations. If the fit fails for an exposure even on its observed values, an error names it; impute that exposure with another method.

ROS and multiple imputation assume a log-normal distribution and need at least three detected values; an exposure with fewer is imputed with LOD / sqrt(2), with a warning, and this is recorded

in `lod_summary`. An exposure with more than 80% of its non-missing values censored gives a warning.

Other implementations: `rexposome::ilod()` and `rexposome::imputeLOD()` impute values below the LOD with the QRILC or MinProb methods of the **imputeLCMD** package, which are not the methods offered here.

Value

For `method != "multiple"`: the numeric matrix with censored values replaced, of the same dimensions as the input. For `method = "multiple"`: an `S4Vectors::SimpleList` of `M` such matrices. In both cases a `data.frame` with one row per exposure of `lod` that is a column of `exposure_matrix` (`exposure`, `n_detected`, `n_censored`, `n_missing`, `method`) records what was done; it is the `"lod_summary"` attribute of the matrix, or `metadata(result)$lod_summary` for the list.

References

Lubin JH, Colt JS, Camann D, et al. (2004). Epidemiologic evaluation of measurement data in the presence of detection limits. *Environ Health Perspect* 112(17):1691–1696. doi:10.1289/ehp.7199

Helsel DR (2012). Statistics for Censored Environmental Data Using Minitab and R. 2nd ed. Wiley. doi:10.1002/9781118162729

Rubin DB (1987). Multiple Imputation for Nonresponse in Surveys. Wiley. doi:10.1002/9780470316696

Examples

```
set.seed(1)
mat <- matrix(rlnorm(200, 0, 1), nrow = 40, ncol = 5,
    dimnames = list(paste0("S", 1:40),
        c("Pb", "Cd", "Hg", "As", "BPA")))
lod <- c(Pb = 0.5, Cd = 0.3, Hg = 0.2, As = 0.1, BPA = 0.4)
## Values reported as below the LOD, and one genuinely missing value
censored <- sweep(mat, 2, lod[colnames(mat)], "<")
mat[censored] <- NA
mat["S1", "Pb"] <- NA
censored["S1", "Pb"] <- FALSE
imputed <- exposure_impute_lod(mat, lod, method = "ros",
    censored = censored)
attr(imputed, "lod_summary")
imputations <- exposure_impute_lod(mat, lod, method = "multiple",
    censored = censored, M = 3L, verbose = FALSE)
length(imputations)
```

---

hlca_airway_subset *Airway epithelial cells of smokers and never smokers from the Human Lung Cell Atlas*

Description

A `SingleCellExposomeExperiment` with the raw counts of 352 tracheal epithelial cells from 12 donors of the Human Lung Cell Atlas (HLCA) core: six heavy smokers and six never smokers, with up to 10 basal, 10 secretory and 10 multiciliated-lineage cells per donor. It is a small real data set for the vignettes and the tests. It shows how the functions are used on real counts; with 12 donors and at most 10 cells per donor and cell type it is far too small to support conclusions about smoking.

Format

An RDS file, `inst/extdata/hlca_airway_subset.rds`, read with `readRDS(system.file("extdata", "hlca_airway_subset.rds", package = "exposomeSC"))`.

Details

Cells Study `"Seibold_2020"` of the HLCA core (Goldfarbmuren et al. 2020): the heavy smokers T85, T90, T101, T120, T154 and T167 (smokers with at least 15 pack-years) and the never smokers T126, T137, T153, T164, T165 and T166, the two groups that the authors compare, as classified in their public metadata. The other two donors of the study in the atlas, a light smoker and a child who never smoked, are left out. Cells annotated `"Basal"`, `"Secretory"` or `"Multiciliated lineage"` at HLCA annotation level 3 were eligible, and up to 10 per donor and cell type were drawn at random with a fixed seed, without looking at expression; donor T154 has only two multiciliated-lineage cells. The donors and the cell types were fixed before any result on cells of these donors was seen. The number of cells per donor and cell type was set later, by the size limit of a Bioconductor source package: before this subset was drawn, but after the smoking model had been run on larger sets of cells of the same donors.

Counts `assay(x, "counts")`: the raw counts of the H5AD (`/raw/X`), 20,111 genes by 352 cells, as a sparse matrix. Genes without a count in these cells are left out, so the column sums are those of the complete transcriptome. The rows are Ensembl gene identifiers. `rowData(x)$symbol` holds the gene names of the atlas (`feature_name`): for 2,301 genes that name is the Ensembl identifier again, and three names (MATR3, RAET1E-AS1 and SPATA13) are each on two rows.

Cell annotations `colData(x)`: `donor_id` (the donor code of the study), `cell_type`, `assay` (10x 3' v2 or v3) and `hlca_donor_id` (the donor identifier of the HLCA).

Donor variables `exposureData(x)`, one row per donor: `smoking` (1 = heavy smoker, 0 = never smoker), `chemistry_v3` (1 = 10x 3' v3, 0 = 10x 3' v2), `age` (years) and `sex_male` (1 = male, 0 = female).

Provenance `metadata(x)$source`: the URL, dataset version, size and SHA-256 of the H5AD, the licence, the references, the selection rule and, in `eligible_cells`, the number of eligible cells of each donor and cell type in the atlas (29,217 in all), from which the cells of the subset were drawn.

Source

The CELLxGENE H5AD of the HLCA core, dataset version `688185ad-11c2-4172-a53a-f4f1f4076860` (https://datasets.cellxgene.cziscience.com/688185ad-11c2-4172-a53a-f4f1f4076860.h5ad, 5,873,612,847 bytes, SHA-256 `1cbdde1e513a31bdb6a3f70296e2d0612760d2af4032e239c94596ea87b44007`), distributed under the Creative Commons Attribution 4.0 International licence (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/). The subset was made by `inst/scripts/make_hlca_airway_subset.R`, which checks the file and carries out the selection described above. Cells and genes were selected; the counts themselves are unchanged.

References

Sikkema L, et al. (2023). An integrated cell atlas of the lung in health and disease. *Nature Medicine*, 29, 1563-1577. doi:10.1038/s41591023023272

Goldfarbmuren KC, et al. (2020). Dissecting the cellular specificity of smoking effects and reconstructing lineages in the human airway epithelium. *Nature Communications*, 11, 2485. doi:10.1038/s4146702016239z

Examples

```
x <- readRDS(system.file("extdata", "hlca_airway_subset.rds",
    package = "exposomeSC"))
x
table(x$donor_id, x$cell_type)
exposureData(x)
S4Vectors::metadata(x)$source$licence
```

`incov_pbmc_subset` *Blood cells, surface proteins and plasma measurements of patients with COVID-19 from the INCOV cohort*

Description

A `SingleCellExposomeExperiment` with the raw counts of 1,032 blood mononuclear cells from 43 patients of the INCOV cohort (Su et al. 2020) at their first blood draw: 14 monocytes and 10 T cells per patient, with the counts of 192 antibodies for the same cells, four of them isotype controls, 13 plasma compounds of exogenous origin among the donor variables and 182 plasma metabolites of the same draw. It is a small real data set for the functions that need more than transcript counts: surface proteins, exposure measurements with missing values, and a second omic layer measured once per donor. It shows how these functions are used on real data. With at most 14 cells per donor and cell type it is far too small to support conclusions, and all its donors had an acute viral infection.

Format

An RDS file, `inst/extdata/incov_pbmc_subset.rds`, read with `readRDS(system.file("extdata", "incov_pbmc_subset.rds", package = "exposomeSC"))`.

Details

Donors Patients of the INCOV cohort who, at their first blood draw, had single cells, plasma metabolomics and plasma proteomics measured, a recorded history of cigarette smoking, and were not in hospital (WHO ordinal scale 1 or 2). 43 patients meet these conditions: 37 women and 6 men aged 19 to 81, 16 of them current or former smokers. The rule was fixed from the clinical sheet and the list of files before any single-cell value of these donors was read.

Counts `assay(x, "counts")`: 15,406 genes by 1,032 cells, as a sparse matrix; the rows are gene symbols. The archive holds the natural logarithm of one plus the counts per million of each cell, from which the counts were recovered: every value of a cell is a whole multiple of the value of one count, and that value times the total of the cell is one million, which the script checks for every cell. Genes without a count in these cells are left out, so the column sums are those of all 24,966 genes of the archive files. The authors had kept cells with at least 200 genes, fewer than 10,000 counts, fewer than 2,500 genes and under 10 per cent mitochondrial counts.

Surface proteins `altExp(x, "CITE")`: the counts of 192 antibodies, four of them isotype controls, for the same cells, recovered in the same way.

Cell types `x$cell_type`, assigned for this data set; the archive holds no cell types. Each sample has two gates, one for the protein CD3 and one for the sum of the proteins CD14, CD11c, CD33 and CD64. A gate is the midpoint of the two centres that a split of the cells into two groups by the logarithm of one plus the count gives. A `"T cell"` has CD3 above its gate,

the sum not above its gate, and fewer than 10 counts of the genes *LYZ*, *S100A8* and *S100A9* together. A `"Monocyte"` has the sum above its gate, CD3 not above its gate, at least 5 counts of those three genes and fewer than 3 counts of *CD3D*, *CD3E* and *TRAC* together. Of the 63,865 cells of the 43 samples, 9,889 are monocytes and 38,213 T cells by this rule. In one sample, INCOV132, the two groups of the sum lie close together (its gate is 8.8 counts, the lowest; the others are 12 or more): 121 of its 328 cells lie above both gates and are neither kind, against fewer than a tenth of the cells of any other sample, and of the 43 cells that the gates alone would make monocytes the conditions on the genes keep 16. The rule was chosen by how well alternatives agreed with these genes in 42 of the samples and was applied to the 43rd as it stood. The five proteins and six genes of the rule are constrained by it and must not be read as results.

Cells 14 monocytes and 10 T cells per donor, drawn at random with a fixed seed from the cells of each kind. The two numbers were set by the size limit of a Bioconductor source package.

Cell annotations `colData(x)`: `donor_id` (the patient identifier of the study), `cell_type`, `sample_id` (the name of the sample in the archive, the patient identifier followed by `-BL`) and `archive_cell` (the name of the cell in the files of the archive). The column names of the object are the patient identifier and the barcode of the cell. The rows are named by gene symbol, which `rowData(x)$symbol` repeats.

Donor variables `exposureData(x)`, one row per donor: `ever_smoker` (1 = current or former smoker, 0 = never smoker), `age` (years), `sex_male` (1 = male, 0 = female), and the plasma levels of 13 compounds of exogenous origin: `cotinine`; `caffeine`, `paraxanthine` and `theobromine`; `pfos` and `pfoa` (perfluorooctanesulfonate and perfluorooctanoate); `methylparaben_sulfate` and `propylparaben_sulfate`; `hydroxychlorothalonil`; `naphthol_sulfate`; `ethyl_glucuronide`; `saccharin` and `acesulfame`. The levels are relative abundances, scaled by the authors so that the median of a compound within a batch is 1, and `NA` where the table has no value for the sample. Ten of the 13 compounds lack a value for at least one donor: cotinine has a value for only 5 donors, propylparaben sulfate for 17, acesulfame for 19 and saccharin for 27. The table does not say why a value is missing. In measurements of this kind the usual reason is that the compound was not detected in the sample, but it is not the only one: the table also lacks a value of glucose for one of these donors. `exposureInfo(x)` gives the name of each compound in the source table (`source_name`), its family, and `lowest_reported`, the smallest value of the compound among the 387 samples of the table. The table reports no limits of detection. For the metabolites that they analysed, those with fewer than 20 per cent of the values missing, the authors put this smallest value in place of a missing one; six of the 13 compounds here have more missing values than that. The 13 compounds were chosen by their kind, from their names and the numbers of samples with a value.

Plasma metabolites `metadata(x)$plasma_metabolites`, 43 donors by 182 metabolites: the metabolites that the source table assigns to a super-pathway other than xenobiotics and reports for all 43 donors, on the same relative scale. Their pathways are in `metadata(x)$metabolite_info`.

Provenance `metadata(x)$source`: the reference, the sources and their licences, the permission for the single-cell values (`single_cells$permission`), the names, sizes and SHA-256 of the 88 input files, the selection rule, the gates of each sample (`gates`) and the numbers of monocytes, T cells and other cells of each donor (`gated_cells`).

The rule for the donors, the rule for the cell types, the numbers of cells and the seed were fixed before any association of a donor variable with expression, proteins or metabolites was computed on these donors. The object was first made from 42 of the 43 donors, because the first transfer of the expression file of one was incomplete, and the vignettes were first written on that version. It was then made again from all 43 by the script without that exclusion; the cells of the 14 donors that follow that donor in the order of the draw are a new random draw.

Source

The clinical records and the plasma measurements are Table S1 of Mendeley Data, doi:10.17632/tzydswhhb5.5 (version 5, published on 4 November 2020), distributed under the Creative Commons Attribution 4.0 International licence (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/). The plasma values were selected, not changed; smoking and sex were recoded as described above.

The single-cell values of genes and surface proteins are from ArrayExpress accession E-MTAB-9357 (https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-9357), released on 29 October 2020 and downloaded on 19 November 2026. On 21 November 2026 the authors who deposited the accession gave permission in these words: "The single-cell values from E-MTAB-9357 included in exposomeSC may be used and redistributed under CC BY 4.0, with citation of Su et al. (2020)." The object holds a subset of the values that the authors released, transformed as described above.

The object as a whole is therefore distributed under CC BY 4.0. Work that uses it should cite Su et al. (2020). The permission does not imply that the authors endorse exposomeSC.

The article itself is published under CC BY-NC-ND 4.0. The values in this object come from the data that the authors deposited in Mendeley Data and ArrayExpress, not from the article.

The object was made by `inst/scripts/make_incov_pbmc_subset.R`, which checks the size and the SHA-256 of every input file and carries out the steps described above.

References

Su Y, Chen D, Yuan D, et al. (2020). Multi-omics resolves a sharp disease-state shift between mild and moderate COVID-19. *Cell*, 183(6), 1479-1495.e20. doi:10.1016/j.cell.2020.10.037

Examples

```
x <- readRDS(system.file("extdata", "incov_pbmc_subset.rds",
    package = "exposomeSC"))
x
table(x$cell_type)
exposureData(x)[1:3, 1:6]
SingleCellExperiment::altExp(x, "CITE")
dim(S4Vectors::metadata(x)$plasma_metabolites)
S4Vectors::metadata(x)$source$single_cells$licence
```

`network-accessors` *Accessors for network results*

Description

Extract the parts of a `CelltypeNetworkResult` (returned by `run_celltype_network` and `run_exposure_network`) and of a `NetworkComparison` (returned by `run_comparative_network`).

Usage

```
networkPrecision(x)

networkAdjacency(x)

networkStability(x)

networkNodes(x)

networkCelltype(x)

networkMethod(x)

diffEdges(x)

sharedEdges(x)

celltypeSpecificEdges(x)

comparisonSummary(x)

## S4 method for signature 'CelltypeNetworkResult'
networkPrecision(x)

## S4 method for signature 'CelltypeNetworkResult'
networkAdjacency(x)

## S4 method for signature 'CelltypeNetworkResult'
networkStability(x)

## S4 method for signature 'CelltypeNetworkResult'
networkNodes(x)

## S4 method for signature 'CelltypeNetworkResult'
networkCelltype(x)

## S4 method for signature 'CelltypeNetworkResult'
networkMethod(x)

## S4 method for signature 'CelltypeNetworkResult'
metadata(x, ...)

## S4 method for signature 'NetworkComparison'
diffEdges(x)

## S4 method for signature 'NetworkComparison'
sharedEdges(x)

## S4 method for signature 'NetworkComparison'
celltypeSpecificEdges(x)

## S4 method for signature 'NetworkComparison'
```

```
comparisonSummary(x)
```

Arguments

x	A CelltypeNetworkResult or, for diffEdges(), sharedEdges(), celltypeSpecificEdges() and comparisonSummary(), a NetworkComparison.
...	Ignored by metadata().

Value

networkPrecision() Numeric feature-by-feature precision matrix.

networkAdjacency() Feature-by-feature matrix of edge indicators (1 = edge, 0 = no edge).

networkStability() Feature-by-feature matrix of edge selection frequencies, or a 0 by 0 matrix when none were stored.

networkNodes() DataFrame with one row per feature, in the order of the matrix rows.

networkCelltype() Character scalar: the cell type.

networkMethod() Character scalar: the estimation method.

metadata() List of parameters and summaries, for example n_donors, transform and edge_counts.

diffEdges() DataFrame of edges that are unique to one cell type or present in some but not all networks.

sharedEdges() DataFrame of edges present in every network.

celltypeSpecificEdges() Named list of DataFrames, one per cell type with edges found only in that cell type.

comparisonSummary() List with the compared cell types, edge counts and pairwise Jaccard overlap.

Examples

```
features <- c("G1", "G2", "M1")
make_network <- function(celltype, cross_weight) {
    precision <- diag(3)
    precision[1, 3] <- precision[3, 1] <- cross_weight
    adjacency <- (precision != 0) * 1L
    diag(adjacency) <- 0L
    dimnames(precision) <- dimnames(adjacency) <- list(features, features)
    CelltypeNetworkResult(
        precision = precision,
        adjacency = adjacency,
        node_info = S4Vectors::DataFrame(
            feature = features,
            omic_layer = c("transcript", "transcript", "metabolite")),
        celltype = celltype,
        metadata = list(n_donors = 20L))
}
mono <- make_network("Monocyte", -0.45)
networkCelltype(mono)
networkAdjacency(mono)
networkNodes(mono)
metadata(mono)$n_donors

comparison <- run_comparative_network(mono, make_network("B cell", 0))
diffEdges(comparison)
```

```
celltypeSpecificEdges(comparison)
comparisonSummary(comparison)$jaccard
```

NetworkComparison *Create a NetworkComparison*

Description

Builds a NetworkComparison from edge tables. Most users obtain one from run_comparative_network instead.

Usage

```
NetworkComparison(
  diff_edges = S4Vectors::DataFrame(),
  shared_edges = S4Vectors::DataFrame(),
  celltype_specific = list(),
  summary = list()
)
```

Arguments

diff_edges, shared_edges	DataFrames with one row per edge and at least the columns node_i and node_j. plot_network_comparison() also uses category and max_diff. Edges in shared_edges have category "shared".
celltype_specific	Named list of DataFrames, one per cell type, with the edges found only in that cell type.
summary	List with, for example, celltypes (the compared cell types), n_edges (named edge counts) and jaccard (named pairwise overlaps).

Value

A valid NetworkComparison.

See Also

network-accessors

Examples

```
shared <- S4Vectors::DataFrame(node_i = "G1", node_j = "M1",
    category = "shared", max_diff = 0.1)
differential <- S4Vectors::DataFrame(node_i = "G2", node_j = "M1",
    category = "differential", max_diff = 0.5)
NetworkComparison(
    diff_edges = differential,
    shared_edges = shared,
    summary = list(celltypes = c("Monocyte", "B cell")))
```

NetworkComparison-class *NetworkComparison: differential network analysis result*

Description

Stores results from run_comparative_network, including differential, shared, and cell-type-specific edges. Use the accessors in network-accessors to extract them.

Usage

```
## S4 method for signature 'NetworkComparison'
show(object)
```

Arguments

object	A NetworkComparison object.

Value

The show method prints a summary and returns NULL invisibly.

Slots

diff_edges DataFrame. Edges present in some but not all of the compared networks (including edges unique to one cell type), with the range of their partial correlations.

shared_edges DataFrame. Edges present in all compared cell types.

celltype_specific Named list of DataFrames. Edges unique to each cell type.

summary List. Summary statistics: Jaccard similarity, edge counts, cell types compared.

See Also

network-accessors, NetworkComparison

Examples

```
showClass("NetworkComparison")
```

plot_celltype_network *Plot cell-type-specific cross-omic network*

Description

Renders a cell-type network graph with nodes coloured by omic layer and edges weighted by the absolute entries of the precision matrix or by stability.

Usage

```
plot_celltype_network(
  network,
  layout = c("fr", "circle", "grid", "bipartite"),
  color_by = c("omic_layer", "stability", "community"),
  highlight_cross_omic = TRUE,
  label_top = 10L,
  edge_alpha_by = c("weight", "stability")
)
```

Arguments

network	CelltypeNetworkResult.
layout	Character; graph layout algorithm. "fr" (Fruchterman-Reingold, default), "circle", "grid", or "bipartite".
color_by	Character; node colouring scheme. "omic_layer" (default) colours nodes by omic layer, "stability" by the largest selection probability among a node's edges (an error when the network has no stability scores), and "community" by Louvain community.
highlight_cross_omic	Logical; emphasize transcript-metabolite edges. Default TRUE.
label_top	Integer; label top N nodes by degree. Default 10.
edge_alpha_by	Character; "weight" (default), the absolute precision-matrix entry of each edge, or "stability" (an error unless every edge has a stability score). Precision entries are not scaled by the diagonal, so their order can differ from that of the partial correlations, which run_comparative_network compares.

Value

A ggplot object (requires **ggraph** and **igraph**).

Examples

```
features <- c("G1", "G2", "M1")
precision <- diag(3)
precision[1, 3] <- precision[3, 1] <- -0.4
adjacency <- (precision != 0) * 1L
diag(adjacency) <- 0L
dimnames(precision) <- dimnames(adjacency) <- list(features, features)
network <- CelltypeNetworkResult(
    precision = precision,
    adjacency = adjacency,
    node_info = S4Vectors::DataFrame(
        feature = features,
        omic_layer = c("transcript", "transcript", "metabolite")
    ),
    celltype = "Monocyte",
    metadata = list(n_donors = 20L)
)
plot_packages <- c("igraph", "ggraph", "ggplot2", "tidygraph")
if (all(vapply(plot_packages, requireNamespace,
        logical(1), quietly = TRUE))) {
    network_plot <- plot_celltype_network(
```

```
        network,
        layout = "circle",
        label_top = 3L
    )
    inherits(network_plot, "ggplot")
}
```

`plot_edge_stability` *Edge stability plot*

Description

Plots the stability-selection frequency of the most stable edges (up to 50) at the selected penalty, coloured by whether the edge joins two omic layers. The network object stores frequencies at the selected penalty only, so this is not a stability surface over the tuning grid; keep the grid from the estimator if one is needed. A reference line is drawn only when `threshold` is given.

Usage

```
plot_edge_stability(network, threshold = NULL)
```

Arguments

`network`	`CelltypeNetworkResult` with `stability=TRUE`.
`threshold`	A single number between 0 and 1, or `NULL` (default). If given, a dashed reference line is drawn at this selection frequency, for example the cut-off used to call stable edges.

Value

A `ggplot` object.

Examples

```
features <- c("G1", "G2", "M1")
stability <- matrix(0, 3, 3, dimnames = list(features, features))
stability[1, 3] <- stability[3, 1] <- 0.85
precision <- diag(3)
dimnames(precision) <- list(features, features)
network <- CelltypeNetworkResult(
    precision = precision,
    adjacency = (stability > 0) * 1L,
    stability = stability,
    node_info = S4Vectors::DataFrame(
        feature = features,
        omic_layer = c("transcript", "transcript", "metabolite")
    ),
    celltype = "Monocyte",
    metadata = list(n_donors = 20L)
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
    stability_plot <- plot_edge_stability(network, threshold = 0.8)
    inherits(stability_plot, "ggplot")
}
```

plot_network_comparison *Plot the edges of a network comparison*

Description

Plots, for the selected edge set, how much each edge's partial correlation varies across the compared cell types (`max_diff`), with the per-cell-type edge counts and pairwise Jaccard overlap in the subtitle. Up to 50 edges are shown.

Usage

```
plot_network_comparison(
  comparison,
  highlight = c("differential", "shared", "all")
)
```

Arguments

`comparison`	`NetworkComparison`.
`highlight`	Character; which edges to plot: `"differential"` (default), `"shared"`, or `"all"`.

Value

A `ggplot` object.

Examples

```
shared_edge <- S4Vectors::DataFrame(
    node_i = "G1", node_j = "M1", category = "shared",
    max_diff = 0.1
)
differential_edge <- S4Vectors::DataFrame(
    node_i = "G2", node_j = "M1", category = "differential",
    max_diff = 0.5
)
comparison <- NetworkComparison(
    diff_edges = differential_edge,
    shared_edges = shared_edge,
    celltype_specific = list(),
    summary = list(
        celltypes = c("Monocyte", "B cell"),
        n_edges = c(Monocyte = 2L, `B cell` = 1L),
        n_shared = 1L,
        n_differential = 1L,
        jaccard = list(Monocyte_vs_B_cell = 0.5)
    )
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
    comparison_plot <- plot_network_comparison(
        comparison,
        highlight = "differential"
    )
```

```
    inherits(comparison_plot, "ggplot")
}
```

`run_celltype_network` *Build cell-type-resolved cross-omic network*

Description

Given a SCEE with pseudobulk transcriptomics and donor-level metabolomics, constructs a block-calibrated conditional independence network for a specified cell type.

Usage

```
run_celltype_network(
  x,
  metabolites = NULL,
  celltype,
  celltype_col = "cell_type",
  exposure = NULL,
  covariates = NULL,
  method = c("coglasso", "block_glasso"),
  stability = TRUE,
  nlambda_w = 15L,
  nlambda_b = 15L,
  subsample_ratio = 0.8,
  rep_num = 20L,
  min_cells = 10L,
  top_var_genes = NULL,
  lambda_w = 0.3,
  lambda_b = 0.5,
  precomputed_network = NULL,
  transform = c("vst", "log_cpm"),
  genes = NULL
)
```

Arguments

`x`	`SingleCellExposomeExperiment`. Each cell's donor is read from the `colData` column named by `sampleColumn(x)`.
`metabolites`	Numeric matrix; rows = donors, cols = metabolites. Row names must be donor IDs. A matrix without columns, or a missing or infinite value for an analysed donor, is an error, and so is a column with the name of another column or of a gene that becomes a node: two nodes cannot have the same name. The matrix is needed to estimate a network; the default, `NULL`, is for a call with `precomputed_network`, which does not use it.
`celltype`	Character; which cell type to build the network for.
`celltype_col`	Character; column in `colData` identifying cell types. Default `"cell_type"`.
`exposure`	Character or NULL; exposure variable to condition on (regressed out), a column of `exposureData(x)`; another name is an error. Default NULL.

`covariates` Character vector or NULL; adjustment covariates, columns of `exposureData(x)`; another name is an error. Donors with a missing value of `exposure` or of a covariate are left out, with a message. Default NULL.

`method` Character; `"coglasso"` (default) or `"block_glasso"`. The coglasso method uses collaborative graphical lasso with block-calibrated penalties (Albanese et al. 2026) and needs the **coglasso** package, version 1.1.0 or later.

`stability` Logical; for `method = "coglasso"`, whether to return the edge-level selection frequencies from XStARS in `stability_scores`. The coglasso penalties are chosen by XStARS either way. `block_glasso` uses the fixed penalties `lambda_w` and `lambda_b`, performs no stability selection and ignores this argument. Default TRUE.

`nlambda_w` Integer; within-omic lambda grid size. Default 15.

`nlambda_b` Integer; between-omic lambda grid size. Default 15.

`subsample_ratio`
Numeric strictly between 0 and 1; proportion of donors drawn in each XStARS subsample, passed to `coglasso::xstars()` as `stars_subsample_ratio`. Default 0.8.

`rep_num` Integer, at least 2; number of XStARS subsamples. Default 20, as in `coglasso::xstars()`.

`min_cells` Integer; minimum cells per pseudobulk sample. Default 10.

`top_var_genes` Integer or NULL; restrict to top N most variable genes for dimensionality reduction. Default NULL (all genes). Recommended: 50-200.

`lambda_w` Numeric; graphical-lasso penalty for pairs of features in the same omic layer, on the correlation scale, for the block_glasso method. Default 0.3.

`lambda_b` Numeric; graphical-lasso penalty for pairs of features in different omic layers, on the correlation scale, for the block_glasso method. Default 0.5.

`precomputed_network`
Optional named list holding a pre-estimated network to wrap into a CelltypeNetworkResult without re-estimation, with elements `adjacency` (symmetric binary matrix, required), `precision` (numeric matrix, optional), `stability` (numeric matrix, optional), `feature_names` (character vector of unique, non-missing and non-empty names, optional; taken from the column names of `adjacency` when absent), and either `block_sizes` (a named vector giving the size of each omic block) or `feature_blocks` (a character vector labelling each feature), both optional. The block labels become the `omic_layer` column of the node table, so use `"transcript"` and `"metabolite"` for the summaries and plots to recognise the layers. A logical adjacency is stored as 0/1 and its diagonal is set to zero. When provided, nothing is estimated, and of the other arguments only `celltype` is used. `x` must still be a `SingleCellExposomeExperiment`.

`transform` Character; the transform of the pseudobulk counts. `"vst"` (default) is the variance-stabilising transformation of `DESeq2::vst()`, blind to the design, and needs **DESeq2** and at least 1000 genes with a mean normalised count above 5; `"log_cpm"` is the base-2 logarithm of the counts per million plus one. The call stops when `"vst"` cannot be computed; it does not switch to the other transform. One choice stays with DESeq2: when the parametric fit of the dispersion trend fails, `vst()` uses a local fit, and a message says so.

`genes` Character vector or `NULL`; row names of `x`, the genes that become the transcript nodes of the network, in the order given. The transform is computed from all genes of `x` before these are taken, so the values of a gene do not depend on which genes are kept, as they would after subsetting `x`. When `top_var_genes` is

smaller than their number, the most variable of these genes are kept, in the order of their variances. Networks of several cell types built with the same genes have the same transcripts, which run_comparative_network needs, unless one of the genes is constant in a cell type and is left out there (see Details). A name that is not a row name of x, or that is the name of more than one row, is an error. Default NULL (all genes).

Details

The workflow proceeds as follows:

1. Pseudobulk aggregation for the specified cell type, reusing existing **exposomeSC** infrastructure.
2. Transformation of the pseudobulk counts, chosen by transform: DESeq2's variance-stabilising transformation or log-CPM. The transform is recorded in metadata(result)$transform.
3. Constant features: a transcript or metabolite that takes one value in every donor (for example a gene with no counts in the cell type) is left out, with a message that names up to five transcripts and five metabolites; all are recorded in metadata(result)$constant_features; the call stops when every transcript or every metabolite is constant. Networks of different cell types can therefore differ in their features, which run_comparative_network does not accept. For a comparison, name the genes with genes and leave a gene that is constant in one of the cell types out of genes for all of them. top_var_genes on its own selects different genes in different cell types.
4. Adjustment: if exposure or covariates are specified, regresses them out from both transcriptomic and metabolomic matrices. The resulting network captures the exposure-adjusted conditional dependency structure.
5. Column-binding of the transcript values and the metabolite values (the residuals of both, where something was regressed out), each column scaled to unit variance.
6. Block-calibrated graphical lasso estimation. With coglasso: collaborative graphical lasso with separate within/between-omic penalties and XStARS stability selection. With block_glasso: the graphical lasso (glasso::glasso()) on the correlation matrix, with the fixed penalties lambda_w within and lambda_b between omic layers, the diagonal unpenalised and no stability selection; it needs the **glasso** package.
7. Returns a CelltypeNetworkResult object.

Value

A CelltypeNetworkResult object.

References

Albanese A, Kohlen W, Behrouzi P (2026). Multi-omics network reconstruction with collaborative graphical lasso. *Bioinformatics* 42(7):btag477. doi:10.1093/bioinformatics/btag477

Friedman J, Hastie T, Tibshirani R (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9(3):432–441. doi:10.1093/biostatistics/kxm045

Liu H, Roeder K, Wasserman L (2010). Stability approach to regularization selection (StARS) for high dimensional graphical models. *Advances in Neural Information Processing Systems*, 23.

Examples

```
donor_ids <- paste0("D", seq_len(8L))
cell_donor <- rep(donor_ids, each = 10L)
cell_ids <- paste0("cell", seq_along(cell_donor))
counts <- outer(
    seq_len(4L),
    seq_along(cell_ids),
    function(gene, cell) 2L + ((5L * gene + 3L * cell) %% 9L)
)
storage.mode(counts) <- "integer"
dimnames(counts) <- list(paste0("G", seq_len(4L)), cell_ids)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(counts = counts),
    colData = S4Vectors::DataFrame(
        cell_id = cell_ids,
        donor_id = cell_donor,
        cell_type = "Monocyte"
    )
)
exposure <- matrix(seq_len(8L), ncol = 1L,
    dimnames = list(donor_ids, "exposure"))
scee <- build_scee(sce, exposure, sample_col = "donor_id")
metabolites <- outer(
    seq_len(8L),
    seq_len(2L),
    function(donor, feature) (donor + feature)^2 / 10
)
dimnames(metabolites) <- list(donor_ids, c("M1", "M2"))
if (requireNamespace("glasso", quietly = TRUE)) {
    network <- run_celltype_network(
        scee,
        metabolites,
        celltype = "Monocyte",
        method = "block_glasso",
        stability = FALSE,
        min_cells = 5L,
        top_var_genes = 3L,
        transform = "log_cpm"
    )
    network
}
```

`run_cell_coupling` *Cell-Level Gene-Protein Coupling Analysis*

Description

Computes within-donor Spearman partial correlations between gene expression and protein abundance at the single-cell level, then uses random-effects meta-regression to test whether coupling strength varies with a donor-level exposure.

Usage

```
run_cell_coupling(
```

```
  x,
  genes,
  proteins,
  exposure,
  celltype,
  celltype_col = "cell_type",
  altexp_name = "CITE",
  confounders = NULL,
  min_cells = 30L,
  min_donors = 10L,
  meta_moderators = NULL,
  var_method = c("asymptotic", "bootstrap"),
  n_boot = 200L
)
```

Arguments

x	A SingleCellExposomeExperiment object. Each cell's donor is read from the colData column named by sampleColumn(x). Must contain an altExp named altexp_name with cell-level protein counts.
genes	Character vector of gene names to test.
proteins	Character vector of protein names to test (must be present in altExp).
exposure	Character; name of the exposure variable in exposureData(x).
celltype	Character; cell type to analyse.
celltype_col	Character; column in colData containing cell type labels. Default "cell_type".
altexp_name	Character; name of the altExp containing protein data. Default "CITE".
confounders	Character vector of column names in colData(x) to partial out (e.g., c("S.Score", "G2M.Score", "percent.mt")). Library sizes (nCount_RNA, nCount_protein) are always controlled regardless of this parameter. Each confounder is ranked within a donor, so it must be numeric, logical, an ordered factor or a label with at most two values; an unordered label with more categories, such as a cell-cycle phase, is an error and must be given as 0/1 indicator columns for all its categories but one, leaving out a category that every donor has cells of (for G1, S and G2M, for example S and G2M when every donor has G1 cells). The names gene, protein, lib_rna and lib_prot are used for the variables of the partial correlation and cannot be confounder names. Nor can a confounder be a column that is not passed on as one column under its own name: a matrix or nested table with several columns or with a column name other than that of the confounder, or a column with the name of an argument of data.frame (row.names, check.rows, check.names, fix.empty.names or stringsAsFactors). Default NULL (library sizes only).
min_cells	Integer; minimum cells per donor to include. Default 30.
min_donors	Integer; minimum number of donors with enough cells (at least min_cells, and at least 7 plus the number of confounders, because the two library sizes are always adjusted for) and values of the exposure and of every moderator. A gene-protein pair also needs a partial correlation in that many donors. Default 10.
meta_moderators	Character vector or one-sided formula of additional donor-level moderators for the meta-regression (e.g., "age", c("age", "sex") or ~ age + sex). These must

be columns in `exposureData(x)` with syntactic R names (see `make.names`) other than `.`, `...` and names such as `..1`, which R reads specially in a formula; a formula may only list such columns, without transformations or interactions. The exposure itself cannot be a moderator, and the names `yi`, `vi`, `donor_exp`, `intrcpt` and `meta_formula`, which the meta-regression uses for its own variables, cannot be moderator names. Default `NULL` (exposure only).

`var_method` Character. Method for estimating the sampling variance of Fisher z-transformed correlations. `"asymptotic"` (default) uses the formula $1.06/(n-q-3)$. `"bootstrap"` resamples cells within each donor to compute empirical variance, which is more accurate for small cell counts or non-normal data but slower. The bootstrap uses the random number generator, so set a seed for reproducible variances.

`n_boot` Integer. Number of bootstrap replicates per donor when `var_method = "bootstrap"`. Default 200.

Details

For each gene-protein pair, the method proceeds in two stages:

Stage 1 (within-donor): For each donor d with at least `min_cells` cells of the specified cell type, compute the Spearman partial correlation between gene expression and protein abundance, controlling for library sizes and any user-specified confounders:

$$r_d = \rho_S(\text{gene}_i, \text{protein}_j \mid \text{nCount_RNA}, \text{nCount_protein}, \text{confounders})$$

Then apply Fisher's z-transform:

$$z_d = \operatorname{arctanh}(r_d), \quad \operatorname{Var}(z_d) \approx 1.06/(n_d - q - 3)$$

where q is the number of conditioning variables (2 + length of `confounders`, less any that are constant within the donor and therefore dropped). The factor 1.06 corrects for the efficiency loss of rank-based correlation (Fieller et al. 1957).

Stage 2 (across-donor meta-regression): Fit a random-effects model via REML with Knapp-Hartung adjustment:

$$z_d = \beta_0 + \beta_1 \cdot \text{exposure}_d + \sum_k \gamma_k \cdot \text{moderator}_{dk} + u_d + \varepsilon_d$$

where $\varepsilon_d \sim N(0, v_d)$ (known sampling variance) and $u_d \sim N(0, \tau^2)$ (between-donor heterogeneity). The Knapp-Hartung adjustment provides more accurate confidence intervals for small numbers of studies (donors), using a t-distribution with $D - p - 1$ degrees of freedom rather than a z-distribution.

Confounder control: Cell-level confounders (cell cycle, mitochondrial content, ambient RNA scores) are partialed out in Stage 1, ensuring that the correlation reflects genuine gene-protein coupling rather than shared technical or biological confounds. Donor-level moderators (age, sex) are included in Stage 2.

Value

A `DataFrame` with one row per gene-protein pair (zero rows when too few donors qualify). A pair whose exposure does not vary among its donors, or is collinear with the moderators, is not tested, with a warning that counts such pairs:

gene, protein Feature names.

n_donors Number of donors in the meta-regression: donors with enough cells, a partial correlation and values of the exposure and of every moderator.

beta0, se0, pval0 Intercept (baseline coupling).

beta1, se1, pval1 Exposure coefficient: the slope, across donors, of the Fisher z of the coupling on the exposure.

tau2 Between-donor heterogeneity variance (REML).

I2 Heterogeneity percentage.

mean_r Mean partial correlation across donors.

mean_n_cells Mean cells per donor.

n_confounders Number of conditioning variables requested (library sizes plus confounders); variables constant within a donor are dropped for that donor.

References

Viechtbauer W (2010). Conducting meta-analyses in R with the metafor package. *J Stat Softw*, 36(3), 1-48. doi:10.18637/jss.v036.i03

Fieller EC, Hartley HO, Pearson ES (1957). Tests for rank correlation coefficients. I. *Biometrika*, 44, 470-481.

Kim S (2015). ppcor: An R Package for a Fast Calculation to Semi-partial Correlation Coefficients. *Commun Stat Appl Methods*, 22(6), 665-674.

Examples

```
set.seed(1)
donors <- sprintf("D%02d", 1:12)
donor <- rep(donors, each = 60)
gene <- matrix(stats::rpois(2 * length(donor), 10), nrow = 2,
    dimnames = list(c("Gene1", "Gene2"), paste0("c", seq_along(donor))))
protein <- matrix(stats::rpois(2 * length(donor), 5), nrow = 2,
    dimnames = list(c("Prot1", "Prot2"), colnames(gene)))
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(counts = gene),
    colData = S4Vectors::DataFrame(donor_id = donor, cell_type = "T"))
SingleCellExperiment::altExp(sce, "CITE") <-
    SummarizedExperiment::SummarizedExperiment(
        assays = list(counts = protein))
exp_mat <- matrix(seq(0, 2, length.out = 12), ncol = 1,
    dimnames = list(donors, "exposure"))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
if (requireNamespace("metafor", quietly = TRUE) &&
    requireNamespace("ppcor", quietly = TRUE)) {
    run_cell_coupling(scee, genes = "Gene1", proteins = "Prot1",
        exposure = "exposure", celltype = "T",
        min_cells = 20L, min_donors = 10L)
}
```

`run_comparative_network` *Compare networks across cell types*

Description

Classifies the union of edges from two or more cell-type-specific networks estimated on the same features as shared, unique to one cell type or differential, and reports pairwise Jaccard overlap. The comparison is descriptive: no edge is tested and no p-value is reported.

Usage

```
run_comparative_network(...)
```

Arguments

`...`	Two or more `CelltypeNetworkResult` objects. All must share the same feature set (same genes and metabolites).

Details

Partial correlations are computed from each precision matrix Θ as $-\Theta_{ij}/\sqrt{\Theta_{ii}\Theta_{jj}}$.

This is the single-cell analogue of the temporal edge comparison in Cheng et al. (ES&T 2024).

Value

A `NetworkComparison` object. Its edge tables contain `max_diff`, the range of the edge's partial correlation across networks.

References

Cheng SL et al. (2024). Multiomic signatures of traffic-related air pollution in London reveal potential short-term perturbations in gut microbiome-related pathways. *Environ Sci Technol* 58:8771-8782. doi:10.1021/acs.est.3c09148

Examples

```
features <- c("G1", "G2", "M1")
make_network <- function(celltype, cross_weight) {
    precision <- diag(3)
    precision[1, 3] <- precision[3, 1] <- cross_weight
    adjacency <- (precision != 0) * 1L
    diag(adjacency) <- 0L
    dimnames(precision) <- dimnames(adjacency) <- list(features, features)
    CelltypeNetworkResult(
        precision = precision,
        adjacency = adjacency,
        node_info = S4Vectors::DataFrame(
            feature = features,
            omic_layer = c("transcript", "transcript", "metabolite")
        ),
        celltype = celltype,
        metadata = list(n_donors = 20L)
```

```
    )
}
monocyte_network <- make_network("Monocyte", -0.45)
b_cell_network <- make_network("B cell", -0.15)
comparison <- run_comparative_network(monocyte_network, b_cell_network)
comparison
```

`run_dose_response` *Cell-type-specific exposure dose-response analysis*

Description

Fits linear and polynomial dose-response models for each gene within a cell type, testing whether the exposure-expression relationship is nonlinear. Linear exposure-response models are common, but some exposures, such as endocrine-disrupting chemicals, show non-monotonic dose responses (Vandenberg et al. 2012).

Usage

```
run_dose_response(
  x,
  exposure,
  celltype,
  celltype_col = "cell_type",
  target_genes = NULL,
  max_degree = 3L,
  min_cells = 10L
)
```

Arguments

`x`	A `SingleCellExposomeExperiment`. Each cell's donor is read from the `colData` column named by `sampleColumn(x)`.
`exposure`	Character. Exposure variable name.
`celltype`	Character. Target cell type.
`celltype_col`	Character. `colData` column with cell type labels. Default `"cell_type"`.
`target_genes`	Character vector (optional). Genes to test. Default: top 50 most variable genes.
`max_degree`	Integer. Maximum polynomial degree to test. Default 3 (cubic). Higher values risk overfitting with small sample sizes.
`min_cells`	Integer. Minimum cells per donor. Default 10.

Details

For each gene, the function fits:

1. `y ~ exposure` (linear)
2. `y ~ poly(exposure, 2)` (quadratic)
3. `y ~ poly(exposure, 3)` (cubic, if max_degree >= 3)

on pseudobulk log-CPM values. `best_model` is the model with the lowest AIC and is descriptive.

`p_nonlinear` is an F-test of the highest-degree polynomial that could be fitted (cubic when `max_degree = 3` and at least six donors are available, otherwise quadratic) against the linear model. It is fixed in advance and does not depend on the AIC selection, because testing the model chosen by AIC would overstate the evidence for nonlinearity.

Value

A `DataFrame` with columns: gene, best_model ("linear", "quadratic", "cubic"), linear_coef, AIC_linear, AIC_best, p_nonlinear (F-test of `nonlinear_model` against the linear model), nonlinear_model, celltype, n_donors (donors with at least `min_cells` cells and a non-missing exposure; donors with a missing exposure are left out, with a message).

References

Vandenberg LN et al. (2012). Hormones and endocrine- disrupting chemicals: low-dose effects and nonmonotonic dose responses. *Endocr Rev* 33:378-455. doi:10.1210/er.20111050

Examples

```
library(SingleCellExperiment); library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(5000, 8), nrow = 50,
    dimnames = list(paste0("G", 1:50), paste0("c", 1:100)))
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = DataFrame(cell_id = paste0("c", 1:100),
        donor_id = rep(paste0("D", 1:10), each = 10),
        cell_type = rep(c("Mono", "NK"), 50)))
exp_mat <- matrix(rnorm(20), nrow = 10,
    dimnames = list(paste0("D", 1:10), c("E1", "E2")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
dr <- run_dose_response(scee, exposure = "E1",
    celltype = "Mono", celltype_col = "cell_type",
    min_cells = 3L,
    max_degree = 2L)
head(dr)
```

---

run_dose_response_gam *GAM-based dose-response analysis*

Description

Fits generalized additive models (GAMs) for cell-type- specific dose-response curves, allowing the data to determine the shape of the exposure-expression relationship without pre-specifying a polynomial degree. Unlike `run_dose_response`, the shape is not restricted to a polynomial.

Usage

```
run_dose_response_gam(
  x,
  exposure,
  celltype,
  celltype_col = "cell_type",
```

```
  target_genes = NULL,
  k = NULL,
  min_cells = 10L,
  loocv = TRUE
)
```

Arguments

x	A SingleCellExposomeExperiment. Each cell's donor is read from the colData column named by sampleColumn(x).
exposure	Character. Exposure variable name.
celltype	Character. Target cell type.
celltype_col	Character. colData column with cell type labels. Default "cell_type".
target_genes	Character vector (optional). Genes to test. Default: top 50 most variable.
k	Integer. Maximum basis dimension for the smooth term, at least 4 and at most the number of distinct exposure values. Default min(n_donors - 1, 10), capped at that number. Higher k allows more wiggliness; the REML smoothing penalty limits overfitting. The exposure needs at least 4 distinct values, because the test of nonlinearity needs k of at least 4; for an exposure with 3 values use run_dose_response.
min_cells	Integer. Min cells per donor. Default 10.
loocv	Logical. If TRUE, compute leave-one-out cross-validated R^2 for the linear and GAM models. Default TRUE.

Details

For each gene, two models are fit:

1. Linear: y ~ exposure (OLS)
2. GAM: y ~ s(exposure, k = k) (penalized thin-plate regression spline via mgcv::gam)

The GAM smooth is estimated using restricted maximum likelihood (REML), which selects the smoothness penalty. The effective degrees of freedom (edf) indicate the complexity: edf close to 1 means approximately linear, and larger values mean more curvature.

p_nonlinear comes from a third model, y ~ exposure + s(exposure, bs = "tp", m = c(2, 0)), whose smooth has no unpenalised null space, so the linear trend is carried by the parametric term and the smooth can only represent departures from linearity. Its p-value is the approximate Wald-type test for smooth terms of Wood (2013). Comparing the linear and GAM fits by an F-test on their residual deviances is not valid here: the GAM's effective degrees of freedom are estimated and often close to one.

Leave-one-out cross-validated R^2 (loocv) provides an honest estimate of out-of-sample prediction accuracy, unlike apparent R^2 which inflates with model complexity. This is critical for small-sample dose-response where overfitting is a real risk.

Why GAM over polynomial:

- Polynomials impose a global parametric form
- GAMs adapt locally – can capture threshold effects, plateaus, U-shapes
- Penalised smoothness selection limits overfitting
- edf provides an interpretable complexity measure

Value

A DataFrame with columns: gene, edf (effective degrees of freedom for smooth term), p_smooth (significance of smooth term vs intercept), p_nonlinear (approximate test that a fully penalised smooth added to a linear exposure term is zero, that is, of departure from linearity; see Details), AIC_linear, AIC_gam, R2_linear, R2_gam, R2_loocv_linear (if loocv=TRUE), R2_loocv_gam, deviance_explained, celltype, n_donors (donors with at least min_cells cells and a non-missing exposure; donors with a missing exposure are left out, with a message).

References

Wood SN (2017). Generalized Additive Models: An Introduction with R. 2nd ed. Chapman and Hall/CRC.

Wood SN (2013). On p-values for smooth components of an extended generalized additive model. *Biometrika* 100:221-228. doi:10.1093/biomet/ass048

Vandenberg LN et al. (2012). Hormones and endocrine- disrupting chemicals: low-dose effects and nonmonotonic dose responses. *Endocr Rev* 33:378-455. doi:10.1210/er.20111050

Examples

```
library(SingleCellExperiment); library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(5000, 8), nrow = 50,
    dimnames = list(paste0("G", 1:50), paste0("c", 1:100)))
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = DataFrame(cell_id = paste0("c", 1:100),
        donor_id = rep(paste0("D", 1:10), each = 10),
        cell_type = rep(c("Mono", "NK"), 50)))
exp_mat <- matrix(rnorm(20), nrow = 10,
    dimnames = list(paste0("D", 1:10), c("E1", "E2")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
if (requireNamespace("mgcv", quietly = TRUE)) {
    dr <- run_dose_response_gam(scee, exposure = "E1",
        celltype = "Mono", celltype_col = "cell_type",
        min_cells = 3L,
        target_genes = paste0("G", 1:3))
    head(dr)
}
```

---

run_exposure_composition

Test exposure associations with donor-level cell composition

Description

Tests whether an exposure is associated with the joint cell-composition profile while retaining donors, rather than cells or repeated biological samples, as the independent units of inference.

Usage

```
run_exposure_composition(
  x,
  exposure,
  celltype_col = "cell_type",
  replicate_col = NULL,
  covariates = NULL,
  min_cells = 10L,
  support_min_donors = 10L,
  support_min_total_cells = 100L,
  celltype_universe = NULL,
  exclude_celltypes = character(),
  zero_prior = 0.5,
  replicate_aggregation = c("equal_replicate", "pooled_counts"),
  n_permutations = 0L,
  permutation_strata = NULL,
  seed = NULL
)
```

Arguments

x	A SingleCellExposomeExperiment. Each cell's donor is read from the colData column named by sampleColumn(x).
exposure	Character scalar naming the exposure variable.
celltype_col	Character scalar naming the cell-type column in colData(x).
replicate_col	Optional character scalar naming biological samples nested within donors. If NULL, each donor is treated as one sample.
covariates	Character vector naming numeric design columns in exposureData(x).
min_cells	Minimum number of cells from the retained universe per biological sample.
support_min_donors	Minimum number of donors in which a cell type must be observed. Support is evaluated without reference to exposure values.
support_min_total_cells	Minimum total number of cells required for a cell type, also evaluated without reference to exposure values.
celltype_universe	Optional pre-specified character vector of cell types eligible for analysis.
exclude_celltypes	Character vector of annotation labels excluded before support filtering.
zero_prior	Positive symmetric Dirichlet prior. The default 0.5 is the Jeffreys prior.
replicate_aggregation	Either "equal_replicate", the primary approach that averages log-ratio coordinates across samples within each donor, or "pooled_counts", which pools cells within donor and is intended as a sensitivity analysis.
n_permutations	Number of Freedman–Lane residual permutations for the omnibus test. Zero reports the asymptotic partial Pillai test only.
permutation_strata	Optional exchangeability strata for permutation. Supply a named vector indexed by donor ID for study-stratified inference.

seed — NULL (default) or a single integer, used only for permutation. With NULL the permutations draw from the caller's random-number stream, so set.seed() before the call makes them reproducible. With an integer, that seed is used inside the call and the caller's random-number state is restored before return.

Details

Counts are formed for every biological sample by supported cell type. Zeros are handled through posterior expected log proportions under a symmetric Dirichlet(1/2) prior. Orthonormal isometric log-ratio coordinates define a basis-invariant partial Pillai omnibus test. Repeated samples are averaged within donor in log-ratio space, giving every donor equal regression weight. Cell-type contrasts are exposure coefficients for log2(cell type / geometric mean of all other supported types) with HC3 heteroscedasticity-consistent standard errors. Nominal 95 percent confidence intervals and Bonferroni simultaneous 95 percent family-wise intervals are both returned and labelled explicitly.

The method estimates relative composition within the supported, retained cell-type universe. It does not identify absolute cell abundance, and the Dirichlet prior handles observed zeros but does not model dissociation, capture, annotation, or tissue-sampling selection mechanisms.

Relation to other methods. speckle::propeller() (Phipson et al. 2022) transforms the cell-type proportions of each sample (logit or arcsine square root) and fits a **limma** linear model for each cell type to compare groups of samples. crumblr::crumblr() (Hoffman and Roussos 2026) returns centred log-ratio transformed counts with observation-level precision weights for weighted linear or linear mixed models. **sccomp** (Mangiola et al. 2023) fits a Bayesian beta-binomial model of composition and variability. voomCLR (Assefa et al. 2026), distributed outside Bioconductor, applies the centred log-ratio transform with observation-level variances from a voom-type mean-variance trend and offers a bias correction of the fitted coefficients. run_exposure_composition() differs from these in three respects: by default (replicate_aggregation = "equal_replicate") repeated samples of a donor are averaged in log-ratio space, so that each donor carries equal weight; the omnibus test of the whole composition uses orthonormal isometric log-ratio coordinates and does not depend on which basis is chosen; and the permutation test can be restricted to exchangeability strata such as studies (permutation_strata). Its per-cell-type contrasts use HC3 standard errors, without precision weights or empirical Bayes moderation of the variances.

Value

A DataFrame with one HC3-adjusted cell-type-versus-rest log-ratio contrast per supported cell type. metadata(result) stores the partial Pillai omnibus test, support audit, design diagnostics, and the donor-level model matrices used for inference. padj is the Holm family-wise adjusted p-value; padj_bh is supplied as a secondary false-discovery-rate summary.

References

Aitchison J (1986). The Statistical Analysis of Compositional Data. Chapman and Hall.

Freedman D, Lane D (1983). A nonstochastic interpretation of reported significance levels. Journal of Business and Economic Statistics, 1, 292–298.

Phipson B, Sim CB, Porrello ER, Hewitt AW, Powell J, Oshlack A (2022). propeller: testing for differences in cell type proportions in single cell data. *Bioinformatics* 38(20):4720–4726. doi:10.1093/bioinformatics/btac582

Hoffman GE, Roussos P (2026). Fast, flexible analysis of differences in cellular composition with crumblr. *Nat Commun* 17:9598. doi:10.1038/s41467026756817

Mangiola S, Roth-Schulze AJ, Trussart M, et al. (2023). sccomp: robust differential composition and variability analysis for single-cell data. *Proc Natl Acad Sci USA* 120(33):e2203828120. doi:10.1073/pnas.2203828120

Assefa AT, Verbist B, Van den Berge K (2026). Assessing differential cell composition in single-cell studies using voomCLR. *Bioinformatics* 42(1):btaf637. doi:10.1093/bioinformatics/btaf637

Examples

```
library(SingleCellExperiment)
library(S4Vectors)
set.seed(1)
donors <- paste0("D", seq_len(12))
donor_type_counts <- t(rmultinom(
    length(donors),
    size = 30,
    prob = c(A = 0.3, B = 0.4, C = 0.3)
))
rownames(donor_type_counts) <- donors
cell_donor <- rep(donors, each = 30)
cell_type <- unlist(
    lapply(seq_along(donors), function(i) {
        rep(colnames(donor_type_counts), donor_type_counts[i, ])
    }),
    use.names = FALSE
)
counts <- matrix(rpois(20 * length(cell_donor), 5), nrow = 20,
    dimnames = list(paste0("G", seq_len(20)),
        paste0("c", seq_along(cell_donor))))
sce <- SingleCellExperiment(
    assays = list(counts = counts),
    colData = DataFrame(
        cell_id = colnames(counts),
        donor_id = cell_donor,
        cell_type = cell_type))
exp_mat <- cbind(X1 = rep(0:1, 6), age = seq(40, 62, by = 2))
rownames(exp_mat) <- donors
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
run_exposure_composition(
    scee,
    exposure = "X1",
    covariates = "age",
    min_cells = 10L,
    support_min_donors = 3L,
    support_min_total_cells = 10L
)
```

---

run_exposure_network *Network among exposure-associated features*

Description

Pre-selects the transcripts and metabolites associated with an exposure, by stability selection or univariate testing, and estimates a cross-omic network among the selected features only. Cheng et al. (2024) analysed transcriptomic and metabolomic signatures of air pollution in two such stages:

features associated with the exposures were identified first, and conditional multi-omic networks estimated afterwards.

Usage

```
run_exposure_network(
  x,
  metabolites,
  celltype,
  exposure,
  celltype_col = "cell_type",
  covariates = NULL,
  selection_method = c("stability", "univariate"),
  fdr_threshold = 0.05,
  pi_threshold = 0.6,
  stability_q = NULL,
  min_cells = 10L,
  network_method = c("coglasso", "block_glasso"),
  lambda_w = 0.3,
  lambda_b = 0.5,
  transform = c("vst", "log_cpm"),
  genes = NULL
)
```

Arguments

x	SingleCellExposomeExperiment. Each cell's donor is read from the colData column named by sampleColumn(x).
metabolites	Numeric matrix, or a data frame of numeric columns; rows = donors, cols = metabolites. Row names must be donor IDs. A matrix without columns, or a missing or infinite value for an analysed donor, is an error, and so is a column with the name of another column or of a candidate gene: two nodes cannot have the same name. Leave the exposure itself out of the columns: a column that holds the values of the exposure, or a monotone transform of them, has a rank correlation of 1 or -1 with it, is always selected and becomes a node.
celltype	Character; cell type to analyse.
exposure	Character; exposure variable name, a column of exposureData(x); another name is an error.
celltype_col	Character; cell type column. Default "cell_type".
covariates	Character vector or NULL; columns of exposureData(x). The selection is made given the covariates (see Details), and they are regressed out of the transcript and metabolite values before the network is estimated; another name is an error, and so is the exposure itself. Donors with a missing covariate value are left out, with a message. A covariate enters the selection by its ranks and the adjustment of the values as the number it is, so either way it is taken as ordered; give a covariate with more than two unordered categories as one 0/1 column for each category but one.
selection_method	Character; "stability" (default) or "univariate".
fdr_threshold	Numeric; FDR threshold for univariate selection. Default 0.05.

`pi_threshold` Numeric in (0.5, 1]; minimum proportion of subsamples in which a feature must be selected under `selection_method = "stability"`. Default 0.6.

`stability_q` Integer or `NULL`; number of features selected in each subsample under `selection_method = "stability"`. The default `NULL` uses $\max(1, \lfloor\sqrt{(2\pi_{thr} - 1)p}\rfloor)$, the largest value for which the Meinshausen-Buhlmann bound on the expected number of falsely selected features is at most 1 (when $(2\pi_{thr} - 1)p \geq 1$). Here p is the number of candidate features that vary across the donors with a value of the exposure.

`min_cells` Integer; minimum cells per pseudobulk.

`network_method` Character; `"coglasso"` (default) or `"block_glasso"`, the estimator for the network on the selected features. `"coglasso"` needs at least one selected transcript and metabolite; when only one omic layer is selected the block graphical lasso is used instead, with a warning. Otherwise it requires the **coglasso** package, version 1.1.0 or later (an error when it is not installed), and chooses the penalties from a 10 by 10 grid with the defaults of `coglasso::xstars()`. The estimator used is recorded in `metadata(result)$network_method`.

`lambda_w` Numeric; graphical-lasso penalty for pairs of features in the same omic layer, on the correlation scale, for the block graphical lasso (`network_method = "block_glasso"`, or the fallback when only one omic layer is selected). Default 0.3.

`lambda_b` Numeric; graphical-lasso penalty for pairs of features in different omic layers, on the correlation scale, for the block graphical lasso. Default 0.5.

`transform` Character; the transform of the pseudobulk counts, `"vst"` (default) or `"log_cpm"`, as in `run_celltype_network`. The call stops when `"vst"` cannot be computed; it does not switch to the other transform.

`genes` Character vector or `NULL`; row names of `x`, the genes that are candidates for the selection, taken after the transform as in `run_celltype_network`. Fewer candidates leave fewer tests to adjust for and a smaller `stability_q`. A name that is not a row name of `x`, or that is the name of more than one row, is an error. Default `NULL` (all genes).

Details

Stage 1: Feature selection, among the donors with a value of the exposure. Each transcript and metabolite is screened by its Spearman correlation with the exposure. With `covariates` it is the partial rank correlation: the ranks of the covariates are regressed out of the ranks of the feature and of the exposure, and the residuals are correlated. A feature without variation across these donors, such as a gene without a count in the cell type, is neither tested nor ranked by either method, and p below counts the features that vary. `"univariate"` tests the correlation (asymptotic t approximation, with one degree of freedom fewer for each covariate that is not constant or a combination of the others among these donors) with BH correction over these features. The approximation is poor when both the exposure and a feature take one value in all but a few donors, as an exposure that only a few donors have, or that is above its limit of detection in a few, and a gene with a count in one or two donors: the p-value of such a gene is far too small whenever its donors are among those few, and with many such genes among the candidates some pass the BH correction together. For an exposure of this kind, name the candidate genes with `genes` and leave out those with a count in few donors. `"stability"` applies stability selection (Meinshausen and Buhlmann 2010) to a marginal screen: in each of 100 random subsamples of half the donors with an observed exposure, the `stability_q` features with the largest absolute correlation with the exposure are selected, and features selected in a proportion of at least `pi_threshold` of subsamples are retained. The screen remains defined when features outnumber donors. Under the exchangeability and better-than-random-guessing assumptions of Meinshausen and Buhlmann, the expected number of falsely selected features is at

most $q^2/((2\pi_{thr}-1)p)$, reported as `pfer_bound`. Candidates with a count in very few donors are hardly ever among the `stability_q` highest of a subsample, which the exchangeability assumption does not allow for; with many of them among the candidates the bound is too small. `genes` can leave them out. At least three features must be selected.

Transcript values are log-CPM or VST values and therefore relative to each donor's library size. When the exposure strongly changes genes that carry a large share of the counts, the remaining genes shift in the opposite direction and can pass the screen; with few measured genes this can dominate the univariate screen.

Stage 2: Network estimation on the selected features only, with collaborative graphical lasso model selection by XStARS (`"coglasso"`) or the block graphical lasso (`"block_glasso"`). It is one network for all the donors analysed. The exposure is not regressed out of the features and is not a node, so an edge can reflect the association that two features share with the exposure, and the network does not estimate how the dependence between features changes with the exposure.

Value

A `CelltypeNetworkResult` restricted to exposure-associated features. Its metadata record the expression transform, the covariates, the donors of the network (`donors`) and the number of them with a value of the exposure (`n_donors_exposure`), on whom the selection was made, the selection settings, for stability selection `pfer_bound`, in `constant_features` the candidates that are constant across the donors with a value of the exposure and so could not be selected, and in `selection` the penalties of the network: `lambda_w` and `lambda_b` for the block graphical lasso, or the size of the grid and the `selected_lambda_w`, `selected_lambda_b` and `selected_c` that coglasso selected, with `stars_variability`.

References

Cheng SL et al. (2024). Multiomic signatures of traffic-related air pollution in London reveal potential short-term perturbations in gut microbiome-related pathways. *Environ Sci Technol* 58:8771-8782. doi:10.1021/acs.est.3c09148

Meinshausen N, Buhlmann P (2010). Stability selection. *J R Stat Soc Series B*, 72(4):417-473.

Examples

```
set.seed(3)
donors <- sprintf("D%02d", 1:20)
donor <- rep(donors, each = 40)
exposure <- stats::setNames(stats::rnorm(20), donors)
counts <- matrix(stats::rpois(20 * length(donor), 20), nrow = 20,
    dimnames = list(paste0("G", 1:20), paste0("c", seq_along(donor))))
## two genes rise and two fall with the exposure
effect <- c(0.8, 0.8, -0.8, -0.8)
for (g in 1:4)
    counts[g, ] <- stats::rpois(length(donor),
        20 * exp(effect[g] * exposure[donor]))
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(counts = counts),
    colData = S4Vectors::DataFrame(donor_id = donor, cell_type = "Mono"))
scee <- build_scee(sce, matrix(exposure, ncol = 1,
    dimnames = list(donors, "PM2.5")), sample_col = "donor_id")
metabolites <- matrix(stats::rnorm(20 * 3), nrow = 20,
    dimnames = list(donors, paste0("M", 1:3)))
metabolites[, "M1"] <- metabolites[, "M1"] + 1.5 * exposure
if (requireNamespace("glasso", quietly = TRUE)) {
```

```
    net <- run_exposure_network(scee, metabolites, celltype = "Mono",
        exposure = "PM2.5",
        selection_method = "univariate",
        network_method = "block_glasso",
        transform = "log_cpm")
    net
}
```

`run_gsea` *Gene set enrichment on sc-ExWAS results*

Description

Runs fast gene set enrichment analysis (fgsea) on the ranked gene list from `run_sc_exwas`, separately for each cell type and exposure.

Usage

```
run_gsea(
  exwas_result,
  gene_sets,
  rank_by = "statistic",
  min_size = 10L,
  max_size = 500L
)
```

Arguments

`exwas_result`	A `DataFrame` returned by `run_sc_exwas` or `run_multi_exwas`.
`gene_sets`	A named list of character vectors, one gene set each. The genes are named as in the gene column of `exwas_result`, which holds the row names of the object that was analysed: Ensembl identifiers when the rows are Ensembl identifiers, symbols when they are symbols.
`rank_by`	Character scalar naming the numeric column used to rank genes. Default `"statistic"`, whose scale depends on the sc-ExWAS backend: signed square-root QL F for edgeR, Wald statistic for DESeq2, and moderated t statistic for voom-dream. `"log2FC"` ranks by effect size; any other numeric column of `exwas_result` may be named.
`min_size`	Integer. Minimum gene set size. Default 10.
`max_size`	Integer. Maximum gene set size. Default 500.

Details

This function requires the `fgsea` package (Bioconductor). Gene sets can come from any source, for example from MSigDB via `msigdbr::msigdbr()`, as long as they use the identifiers of the result.

Genes are ranked by the backend-specific association statistic (default) within each cell type, then fgsea is run per cell type. For edgeR QL this is the signed square root of the QL F statistic and is not a Wald z statistic. Results are combined with a final cross-cell-type FDR adjustment.

Ranking by the signed statistic preserves direction and within-backend evidence ordering. Statistic magnitudes are not calibrated for comparison across different backends; use `rank_by = "log2FC"` when an effect-size ranking is the scientific target.

`fgsea` estimates p-values by Monte Carlo sampling, so they depend on the random number generator state. `run_gsea()` runs `fgsea` serially, so calling `set.seed()` beforehand makes them reproducible. When a gene appears more than once in a stratum, its largest ranking value is kept.

Value

A `DataFrame` with the `fgsea` columns pathway, pval, padj (within cell type and exposure), log2err, ES, NES, size and leadingEdge (semicolon-separated), plus celltype, exposure and padj_global (Benjamini-Hochberg across all cell types and exposures).

References

Korotkevich G et al. (2021). Fast gene set enrichment analysis. *bioRxiv*. doi:10.1101/060012

Subramanian A et al. (2005). Gene set enrichment analysis. *PNAS* 102:15545-15550. doi:10.1073/pnas.0506580102

Examples

```
set.seed(1)
genes <- paste0("G", 1:200)
exwas <- S4Vectors::DataFrame(gene = genes, celltype = "Mono",
    exposure = "PM2.5",
    statistic = c(stats::rnorm(20, mean = 3), stats::rnorm(180)))
gene_sets <- list(shifted = genes[1:20], random = sample(genes, 25))
if (requireNamespace("fgsea", quietly = TRUE)) {
    run_gsea(exwas, gene_sets, min_size = 10L)
}
```

---

run_interaction_test *Test exposure-by-celltype interaction*

Description

Formally tests whether an exposure effect differs across cell types using a pseudobulk interaction model. This prevents the common mistake of concluding "the exposure affects cell type A but not B" based on separate per- celltype p-values (Gelman & Stern 2006).

Usage

```
run_interaction_test(
  x,
  exposure,
  celltype_col = "cell_type",
  target_genes = NULL,
  covariates = NULL,
  min_cells = 10L,
  min_donors = 5L
)
```

Arguments

x	A SingleCellExposomeExperiment. Each cell's donor is read from the colData column named by sampleColumn(x).
exposure	Character. Exposure variable name.
celltype_col	Character. colData column with cell type labels. Default "cell_type".
target_genes	Character vector (optional). Genes to test. Default: top 50 most variable genes across all cell types.
covariates	Character vector (optional). Columns of exposureData adjusted for; listing the exposure here is an error.
min_cells	Integer. Min cells per donor per celltype. Default 10.
min_donors	Integer. Min donors per celltype. Default 5.

Details

For each gene, pseudobulk log-CPM values of all cell types are stacked, one row per donor and cell type, and two linear models are compared with an F-test:

```
reduced: y ~ donor + celltype + covariates:celltype
full:    y ~ donor + celltype + covariates:celltype + exposure:celltype
```

Donor fixed effects absorb every donor-level quantity, including the exposure and covariate main effects, and the shared donor component of expression in different cell types, so the test compares exposure slopes between cell types through within-donor contrasts rather than treating cell types from the same donor as independent. Covariate-by-cell-type terms allow covariate effects to differ between cell types. Donors observed in only one cell type do not inform the interaction.

Comparing significance levels across cell types is not the same as testing for a difference between them (Gelman & Stern 2006, Nieuwenhuis et al. 2011); this function tests the difference directly.

Value

A DataFrame with columns: gene, p_interaction (F-test for the exposure-by-cell-type terms), df_interaction (numerator degrees of freedom), padj_interaction (BH-corrected), n_celltypes, n_donors.

References

Gelman A, Stern H (2006). The difference between 'significant' and 'not significant' is not itself statistically significant. *Am Stat* 60:328-331.

Nieuwenhuis S et al. (2011). Erroneous analyses of interactions in neuroscience. *Nat Neurosci* 14:1105-1107. doi:10.1038/nn.2886

Examples

```
library(SingleCellExperiment); library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(5000, 8), nrow = 50,
    dimnames = list(paste0("G", 1:50), paste0("c", 1:100)))
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = DataFrame(cell_id = paste0("c", 1:100),
        donor_id = rep(paste0("D", 1:10), each = 10),
        cell_type = rep(c("Mono", "NK"), 50)))
exp_mat <- matrix(rnorm(20), nrow = 10,
```

```
    dimnames = list(paste0("D", 1:10), c("E1", "E2")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
ix <- run_interaction_test(scee, exposure = "E1",
    celltype_col = "cell_type", min_cells = 3L, min_donors = 3L)
head(ix)
```

`run_meta_exwas` *Meta-analysis across cohorts for sc-ExWAS*

Description

Combines independent-cohort sc-ExWAS estimates using inverse-variance weighted fixed-effect synthesis or random-effects synthesis. Keys are matched exactly on gene, cell type, and exposure.

Usage

```
run_meta_exwas(
  results_list,
  cohort_names = NULL,
  method = c("fixed", "random"),
  effect_col = NULL,
  se_col = NULL
)
```

Arguments

`results_list`	A list of `DataFrames` or data frames. Each cohort must contain unique `gene`-`celltype`-`exposure` keys and an effect estimate plus standard error.
`cohort_names`	Character vector with one unique name per cohort. Defaults to `Cohort_1`, `Cohort_2`, and so forth.
`method`	Character. `"fixed"` (default), or `"random"` for REML between-cohort variance with Hartung–Knapp inference. Random-effects synthesis requires **metafor** and at least three cohorts per key.
`effect_col`	Optional character scalar naming the effect column. When `NULL`, the first available column among `log2FC`, `log2FoldChange`, and `estimate` is used separately in each cohort.
`se_col`	Optional character scalar naming the standard-error column. When `NULL`, the first available column among `se` and `lfcSE` is used separately in each cohort. edgeR QL results from `run_sc_exwas` are deliberately unsupported because `glmQLFTest` does not supply coefficient standard errors.

Details

Fixed-effect synthesis uses

$$\hat{\beta} = \frac{\sum_i w_i \hat{\beta}_i}{\sum_i w_i}, \quad w_i = 1/SE_i^2.$$

Random-effects synthesis uses restricted maximum likelihood for τ^2 and Hartung–Knapp small-sample inference. It is deliberately unavailable for only two cohorts. The I^2 and Cochran Q statistics are reported as heterogeneity diagnostics; they do not establish transportability or independence of the contributing cohorts.

The function never maps gene identifiers or cell-type labels. Harmonisation must occur before calling it. Within-cohort duplicate keys are an error, preventing accidental double weighting after label aggregation. Inverse-variance synthesis also rejects edgeR QL results from `run_sc_exwas`, recognised by their `method` label or their metadata, because the QL F test supplies no coefficient standard errors. Use a backend that estimates valid coefficient standard errors for each independent cohort before meta-analysis.

Value

A `DataFrame` with one row per exact key and columns `meta_effect`, `meta_se`, `meta_statistic`, `meta_df`, `meta_pvalue`, `meta_padj`, `I2`, `Q_pvalue`, `tau2`, `n_cohorts`, `cohorts`, `direction`, and `method`. Under Hartung–Knapp inference `meta_statistic` is a t statistic with `meta_df` degrees of freedom; under fixed-effect synthesis it is a z statistic and `meta_df` is `Inf`.

References

Borenstein M et al. (2009). *Introduction to Meta-Analysis*. Wiley.

Viechtbauer W (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software* 36:1–48. doi:10.18637/jss.v036.i03

Examples

```
library(S4Vectors)
r1 <- DataFrame(
    gene = c("A", "B"), celltype = "Mono",
    log2FC = c(0.5, -0.3), se = c(0.1, 0.15),
    exposure = "PM2.5"
)
r2 <- DataFrame(
    gene = c("A", "B", "C"), celltype = "Mono",
    log2FC = c(0.4, -0.1, 0.2), se = c(0.12, 0.2, 0.1),
    exposure = "PM2.5"
)
meta <- run_meta_exwas(
    list(r1, r2),
    cohort_names = c("CohortA", "CohortB")
)
meta
```

---

run_mixture_qgcomp *Formal mixture analysis via quantile g-computation*

Description

Wraps `qgcomp::qgcomp.noboot()` or `qgcomp::qgcomp.boot()` on pseudobulk data for cell-type-specific exposure mixture analysis. Unlike `run_sc_mixture` (simplified screening), it reports a confidence interval and p-value for the overall mixture effect and, without bootstrapping, the positive and negative exposure weights estimated by **qgcomp**.

Usage

```
run_mixture_qgcomp(
  x,
  exposures,
  celltype,
  celltype_col = "cell_type",
  target_genes = NULL,
  covariates = NULL,
  q = 4L,
  bootstrap = FALSE,
  B = 200L,
  min_cells = 10L
)
```

Arguments

x	A SingleCellExposomeExperiment. Each cell's donor is read from the colData column named by sampleColumn(x).
exposures	Character vector. Exposure variable names, distinct. Each name must be a syntactic R name (see make.names) because it goes into the model formula, and cannot be ., ... or a name such as ..1, which R reads specially there. The names y, weights and id__ cannot be used: the response is called y, and **qgcomp** adds columns called weights and id__ to the model data. With bootstrap = FALSE a name cannot contain _q either, which **qgcomp** removes from the names of the weights it returns.
celltype	Character. Target cell type.
celltype_col	Character. colData column with cell type labels. Default "cell_type".
target_genes	Character vector (optional). Genes to summarise. Default: top 20 most variable.
covariates	Character vector (optional). Columns of exposureData; an unknown name is an error, and so is an exposure listed here. Names must be syntactic and cannot be y, weights, id__, ., ... or a name such as ..1, as for exposures.
q	Integer. Number of quantile bins. Default 4.
bootstrap	Logical. If TRUE, use qgcomp.boot() for bootstrap confidence intervals. Default FALSE (faster; a t interval on the residual degrees of freedom, see Details).
B	Integer. Number of bootstrap replicates (only if bootstrap = TRUE). Default 200.
min_cells	Integer. Default 10.

Details

Quantile g-computation (Keil et al. 2020) estimates the overall effect of jointly increasing all exposures by one quantile, and without bootstrapping decomposes it into positive and negative contributions. The weights are interpretable only when the exposure effects are of a similar shape (Keil et al. 2020).

The pseudobulk response is the mean log-CPM of the target genes over donors with complete exposure and covariate data.

Without bootstrapping, mixture_pvalue is a two-sided t test of ψ on the residual degrees of freedom of the fitted model, and ci_lower and ci_upper are the matching t interval. qgcomp.noboot() itself uses the null degrees of freedom minus the number of exposures for its p-value, which leaves

the covariates out, and a normal interval; its own values are kept in `metadata(result)$fit`. With `bootstrap = TRUE` the p-value and the interval are those of `qgcomp.boot()`; the call stops when `qgcomp.boot()` returns no p-value for ψ.

Comparison with `run_sc_mixture`:

Feature	run_sc_mixture	run_mixture_qgcomp
CI for overall effect	No	Yes (t interval or bootstrap)
Pos/neg weights	No	Yes (without bootstrap)
P-value for overall effect	No	Yes
Speed	Fast	Moderate
Minimum n	5	10+ recommended

Value

A one-row `DataFrame` with columns `celltype`, `mixture_effect` (the overall mixture effect ψ), `mixture_se`, `ci_lower` and `ci_upper` (95% confidence interval for ψ), `mixture_pvalue` (for the interval and the p-value see Details), `n_donors`, `n_genes` and `method` (`"qgcomp.noboot"` or `"qgcomp.boot"`). `metadata(result)` holds:

positive_weights Named numeric. Exposures that increase the response, with relative weights; `NULL` when `bootstrap = TRUE`, because `qgcomp.boot()` does not estimate weights.

negative_weights Named numeric. Exposures that decrease the response; `NULL` when `bootstrap = TRUE`.

target_genes The genes whose mean log-CPM is the response.

fit The `qgcomp` model object (for plotting and further analysis).

References

Keil AP et al. (2020). A quantile-based g-computation approach to addressing the effects of exposure mixtures. *Environ Health Perspect* 128:047004. doi:10.1289/EHP5838

Examples

```
# Requires the qgcomp package
library(SingleCellExperiment); library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(5000, 8), nrow = 50,
    dimnames = list(paste0("G", 1:50), paste0("c", 1:100)))
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = DataFrame(cell_id = paste0("c", 1:100),
        donor_id = rep(paste0("D", 1:10), each = 10),
        cell_type = rep(c("Mono", "NK"), 50)))
exp_mat <- matrix(rnorm(30), nrow = 10,
    dimnames = list(paste0("D", 1:10), c("E1", "E2", "E3")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
if (requireNamespace("qgcomp", quietly = TRUE)) {
    mix <- run_mixture_qgcomp(scee,
        exposures = c("E1", "E2", "E3"),
        celltype = "Mono", celltype_col = "cell_type",
        min_cells = 3L)
    mix
    metadata(mix)$positive_weights
    metadata(mix)$negative_weights
}
```

`run_multi_exwas` *Run sc-ExWAS across multiple exposures*

Description

Convenience wrapper that runs `run_sc_exwas` for each exposure and combines results with a final global FDR correction across all exposures and cell types.

Usage

```
run_multi_exwas(
  x,
  exposures = NULL,
  celltype_col = "cell_type",
  covariates = NULL,
  min_cells = 10L,
  min_donors = 5L,
  filter_genes = TRUE,
  ...
)
```

Arguments

`x`	A `SingleCellExposomeExperiment`. Each cell's donor is read from the `colData` column named by `sampleColumn(x)`.
`exposures`	Character vector. Exposure names to test. Default: all exposures in `exposureData` except the `covariates`.
`celltype_col`	Character. `colData` column with cell type labels. Default `"cell_type"`.
`covariates`	Character vector (optional). Columns of `exposureData` adjusted for in every model; they are not tested as exposures, and listing one in `exposures` is an error.
`min_cells`	Integer. Default 10.
`min_donors`	Integer. Default 5.
`filter_genes`	Logical. Default TRUE.
`...`	Further arguments of `run_sc_exwas`, such as `method` or `celltypes`. An argument that `run_sc_exwas` does not accept is an error.

Details

If `run_sc_exwas` stops for one of the exposures, the call stops too, with an error that names the exposure, instead of leaving that exposure out of `padj_all`. A cell type that `run_sc_exwas` skips for an exposure with a warning, for example for too few donors with complete data, adds no tests to `padj_all`; the diagnostics in `metadata(result)$per_exposure` list the cell types fitted for each exposure.

Value

A DataFrame with all results combined and padj_all for global FDR across all tests. The se and statistic fields retain the selected backend's semantics: edgeR QL has numeric-NA coefficient standard errors and a signed square-root QL F statistic, whereas DESeq2 and voom-dream return their coefficient standard errors with Wald and moderated t statistics, respectively. The columns are those of run_sc_exwas followed by padj_all; metadata(result)$per_exposure keeps the metadata() (parameters and diagnostics) of each exposure's run_sc_exwas result.

Examples

```
library(SingleCellExperiment); library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(5000, 8), nrow = 50,
    dimnames = list(paste0("G", 1:50), paste0("c", 1:100)))
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = DataFrame(cell_id = paste0("c", 1:100),
        donor_id = rep(paste0("D", 1:5), each = 20),
        cell_type = rep(c("Mono", "NK"), 50)))
exp_mat <- matrix(rnorm(15), nrow = 5,
    dimnames = list(paste0("D", 1:5), c("E1", "E2", "E3")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
multi <- run_multi_exwas(scee,
    exposures = c("E1", "E2"),
    celltype_col = "cell_type",
    min_cells = 5L, min_donors = 3L)
head(multi)
```

---

run_sc_exwas *Run cell-type-specific ExWAS*

Description

Performs pseudobulk differential expression analysis with a continuous exposure variable as covariate. Aggregates counts within each donor-celltype stratum to avoid pseudoreplication, then uses an empirical-Bayes pseudobulk model for inference.

Usage

```
run_sc_exwas(x, ...)

## S4 method for signature 'SingleCellExposomeExperiment'
run_sc_exwas(
  x,
  exposure,
  celltype_col = "cell_type",
  celltypes = NULL,
  covariates = NULL,
  min_cells = 10L,
  min_donors = 5L,
  filter_genes = TRUE,
  method = c("edgeR", "DESeq2", "voom_dream"),
  min_group_donors = 5L,
```

```
  filter_min_count = 10L,
  filter_min_total_count = 15L,
  filter_method = c("fixed_support", "filterByExpr"),
  filter_min_cpm = 1,
  filter_min_donors = 5L,
  test_features = NULL,
  robust = TRUE,
  ...
)
```

Arguments

x	A SingleCellExposomeExperiment. Each cell's donor is read from the colData column named by sampleColumn(x).
...	Must be empty. Any further argument is an error, so that a misspelled argument name cannot silently change the analysis.
exposure	Character. Name of the exposure variable (column in exposureData).
celltype_col	Character. Column in colData containing cell type annotations. Default "cell_type".
celltypes	Character vector (optional). Which cell types to test. Default: all.
covariates	Character vector (optional). Additional covariates from exposureData. A covariate cannot be called "exposure", the name of the model's exposure term, or be the exposure itself, and each name must be a syntactic R name (see make.names) because it goes into the model formula; ., ... and names such as ..1, which R reads specially there, cannot be used.
min_cells	Integer. Minimum cells per donor-celltype stratum. Default 10.
min_donors	Integer. Minimum donors per cell type. Default 5, and at least 3, for every backend. Cell types with fewer valid donors are skipped with a warning.
filter_genes	Logical. If TRUE (default), filter low-expression genes before model fitting. The edgeR backend defaults to exposure-label-independent donor-support filtering; the DESeq2 and voom-dream backends retain genes detected in at least 20% of donors. If FALSE, every backend leaves out, in each cell type, only the genes without any count in the donors analysed for it.
method	Character. Backend for differential expression. "edgeR" (default) uses the edgeR v4 quasi-likelihood pipeline (see Details). "DESeq2" uses DESeq2 on aggregated counts. "voom_dream" uses variancePartition's voomWithDreamWeights() and dream() with precision weights from the mean-variance trend (Hoffman and Roussos 2021). Pseudobulk data have one observation per donor and the model has no random effects, so dream() fits it with limma and the moderated t statistic uses the residual degrees of freedom. This backend does not use the **dreamlet** package.
min_group_donors	Integer. For a binary exposure, minimum complete donors required in each group for the edgeR backend, at least 2. Default 5.
filter_min_count	Numeric. Minimum count passed to edgeR::filterByExpr when filter_method = "filterByExpr".
filter_min_total_count	Numeric. Minimum total pseudobulk count for either edgeR filtering method.

`filter_method` Character. edgeR expression-filtering rule. `"fixed_support"` (default) requires a fixed CPM threshold in a fixed number of independent donors and does not use exposure labels. `"filterByExpr"` retains the edgeR compatibility option.

`filter_min_cpm` Numeric. CPM threshold for `filter_method = "fixed_support"`. Default 1.

`filter_min_donors` Integer. Minimum number of independent donors meeting `filter_min_cpm`. Default 5.

`test_features` Optional character vector of feature identifiers that defines the reported hypothesis family. Library-size calculation, expression filtering, normalisation and dispersion estimation still use all available features; subsetting occurs only after model fitting. Every requested feature must pass the pre-specified expression filter in every tested cell type, otherwise the call fails rather than silently changing the requested hypothesis family. Currently supported only by the edgeR backend; other backends fail explicitly rather than ignoring it.

`robust` Logical. For the edgeR backend, moderate the quasi-likelihood dispersions with robust empirical Bayes (`glmQLFit(robust = TRUE)`). Default `TRUE`.

Details

The pseudobulk approach aggregates single-cell counts within each (donor x cell type) stratum, creating one observation per donor per cell type. This correctly treats cells from the same donor as non-independent, avoiding the inflated Type I error rates that plague cell-level analyses (Squair et al. 2021, *Nat Commun*). The pseudobulk counts that enter the models are available as `SummarizedExperiment` objects from `donor_pseudobulk`.

The exposure variable and specified covariates enter one donor-level design matrix. The default edgeR backend fits the edgeR v4 quasi-likelihood model, `glmQLFit(legacy = FALSE)`, which sets the negative-binomial dispersion internally from the most highly expressed genes and moderates the quasi-likelihood dispersions by empirical Bayes; the exposure coefficient is tested with a quasi-likelihood F-test (Chen et al. 2025). No separate `estimateDisp()` step is run: from edgeR 4.10.0 the method gives the same result whether or not the data hold such estimates, whereas earlier edgeR 4.x releases did not. The edgeR version and `legacy = FALSE` are recorded in `metadata(result)$parameters`. Complete-transcriptome library sizes are retained when expression filtering or `test_features` restricts the reported hypothesis family.

`edgeR::glmQLFTest` does not return a coefficient standard error. Consequently, the edgeR backend never reconstructs `se` from the ratio of the log-fold change to the QL statistic. Its `statistic` field is a directional ranking representation of the QL F statistic, not a Wald z or t statistic, and must not be used to derive a coefficient confidence interval. The edgeR result metadata records `se_method = "not_available_edgeR_QL"` and `statistic_type = "signed_sqrt_qlf"`.

What is delegated and what is kept. The raw counts of each donor and cell type are summed with `scrapper::aggregateAcrossCells()`, and the models are fitted by **edgeR**, **DESeq2** or **variancePartition** and **limma**. Four choices are made by **exposomeSC** itself, on purpose, because they define the hypotheses of an exposome-wide study:

1. The exposure is a donor-level variable from `exposureData(x)`, continuous or binary, and enters the design matrix together with the covariates; cells are never grouped by exposure.
2. Gene filtering does not use the exposure. The default edgeR rule (`filter_method = "fixed_support"`) keeps genes with at least `filter_min_cpm` CPM in `filter_min_donors` donors, and the DESeq2 and voom-dream backends keep genes detected in at least 20% of donors (and in at least two); which genes are tested therefore does not depend on the exposure values. By contrast, `scran::pseudoBulkDGE()`, for example, filters each label with `edgeR::filterByExpr()` using the design matrix.

3. In the edgeR backend the library sizes are the column sums of the complete pseudobulk transcriptome, computed before filtering, and are kept when filtering or `test_features` reduces the tested genes. The voom-dream backend computes library sizes from the retained genes and DESeq2 uses its own size factors.
4. `padj_global` applies the Benjamini-Hochberg procedure once across all tested gene-by-cell-type hypotheses, next to the within-cell-type `padj`.

Value

A `DataFrame` with one row per tested gene and cell type and the same columns, in the same order, for every backend: `gene`, `celltype`, `log2FC`, `se`, `statistic`, `pvalue`, `pvalue_underflow_clamped` (`TRUE` where a p-value of exactly zero was replaced by the smallest positive double), `padj` (BH within the cell type), `padj_global` (BH across all tested gene-by-cell-type hypotheses), `logCPM`, `exposure`, `n_donors`, `n_unexposed` and `n_exposed` (group sizes when the exposure takes two values, otherwise `NA`), `min_cells` and `median_cells` (cells per donor in the cell type), and `method`. Rows are ordered by `padj_global`, then `pvalue`.

`logCPM` is the average log2 count per million reported by the backend: edgeR's `logCPM` for the edgeR backend, the mean of the voom log-CPM values for voom-dream, and `edgeR::aveLogCPM()` of the filtered pseudobulk counts for DESeq2.

Uncertainty fields are backend-specific: edgeR QL reports `se` as numeric `NA` and `statistic` as the directional signed square root of the QL F statistic; DESeq2 reports its coefficient standard error and Wald statistic; voom-dream reports the moderated standard error (`|log2FC / t|`) and the moderated t statistic. `metadata(result)$parameters` records the settings, the backend and its package version, `se_method` and `statistic_type`; `metadata(result)$diagnostics` holds, for each fitted cell type, the donors used, the numbers of genes before and after filtering and the design columns; for the edgeR backend also `dropped_design_columns`, the covariate columns left out of the design as collinear (each is also reported in a warning), and `residual_df`, the smallest adjusted residual degrees of freedom over the fitted genes (`df.residual.adj`).

References

Squair JW et al. (2021). Confronting false discoveries in single-cell differential expression. *Nat Commun* 12:5692. doi:10.1038/s41467021259602

Chen Y, Chen L, Lun ATL, Baldoni PL, Smyth GK (2025). edgeR v4: powerful differential analysis of sequencing data with expanded functionality and improved support for small counts and larger datasets. *Nucleic Acids Research* 53(2):gkaf018. doi:10.1093/nar/gkaf018

Hoffman GE, Roussos P (2021). Dream: powerful differential expression analysis for repeated measures designs. *Bioinformatics* 37(2):192–201. doi:10.1093/bioinformatics/btaa687

Hoffman GE, Schadt EE (2016). variancePartition: interpreting drivers of variation in complex gene expression studies. *BMC Bioinformatics* 17:483. doi:10.1186/s128590161323z

See Also

`donor_pseudobulk` for the pseudobulk counts, `run_multi_exwas` for several exposures.

Examples

```
library(SingleCellExperiment)
library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(5000, 8), nrow = 50,
    dimnames = list(paste0("G", 1:50), paste0("c", 1:100)))
```

```
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = DataFrame(cell_id = paste0("c", 1:100),
        donor_id = rep(paste0("D", 1:5), each = 20),
        cell_type = rep(c("Mono", "NK"), 50)))
exp_mat <- matrix(rnorm(15), nrow = 5,
    dimnames = list(paste0("D", 1:5), c("E1", "E2", "E3")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
result <- run_sc_exwas(scee, exposure = "E1",
    celltype_col = "cell_type", min_cells = 5L)
head(result)
metadata(result)$parameters$statistic_type
```

`run_sc_exwas_glmm` *Cell-level linear mixed model sc-ExWAS with finite-df inference*

Description

Fits, per gene, the cell-level model `y ~ exposure * celltype + covariates + (1 | donor) + (1 | donor:celltype)` and reports cell-type-specific exposure effects. For `family = "gaussian"` the model is a linear mixed model fitted by `lmerTest::lmer` with **REML = TRUE** (mandatory), and p-values are referred to a finite denominator degrees of freedom – **Satterthwaite** (default) or **Kenward-Roger** – via `lmerTest::contest1D`. A Wald test referred to the normal distribution would be anticonservative with few donors.

Usage

```
run_sc_exwas_glmm(
  x,
  exposure,
  celltype_col = "cell_type",
  covariates = NULL,
  target_genes = NULL,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  family = c("gaussian", "nbinom"),
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

`x`	A `SingleCellExposomeExperiment`. Each cell's donor is read from the `colData` column named by `sampleColumn(x)`.
`exposure`	Character; a donor-level column of `exposureData(x)`.
`celltype_col`	Character. Column in `colData` containing cell type annotations. Default `"cell_type"`.
`covariates`	Character vector; donor covariate columns in `exposureData` (numeric), added as fixed-effect main effects. The names `exposure`, `donor`, `celltype` and `y` are used by the model and cannot be covariate names; with `family = "nbinom"` the same holds for `logoff` and the `expo_<cell type>` columns. Each name must be a syntactic R name (see `make.names`) other than `.`, `...` and names such as `..1`, which R reads specially in a formula, and the exposure itself cannot be a covariate.

target_genes	Character; genes to test (default: all rows of x). A cell-level LMM per gene is costly – passing a focused gene set is strongly recommended.
ddf	Denominator degrees-of-freedom method for family = "gaussian": "Satterthwaite" (default) or "Kenward-Roger" (needs **pbkrtest**; use for $n \lesssim 30$ donors).
family	"gaussian" (LMM on cell-level $\log_2$ CP10k, default) or "nbinom" (negative-binomial GLMM on cell counts).
BPPARAM	A BiocParallel param used to parallelise over genes. Defaults to BiocParallel::SerialParam(), so no worker process is started unless one is asked for; pass, for example, BiocParallel::SnowParam(workers = 2L) to fit genes in parallel.

Details

Why REML. Maximum likelihood biases the variance components downward, which shrinks the residual scale and inflates the exposure t-statistic; the Kenward-Roger adjusted covariance and the Satterthwaite degrees-of-freedom approximation are both defined only on the REML fit. The function therefore forces REML = TRUE and does not expose a switch.

Random effects. The exposure varies between donors, so the replicates for a cell-type-specific exposure effect are donor-by-cell-type units, not cells. The model therefore includes a donor-by-cell-type random intercept as well as a donor random intercept; without it, variation shared by the cells of one donor and cell type is treated as independent and the cell-type-specific tests are anticonservative. With a single cell type the model reduces to y ~ exposure + covariates + (1 | donor). A variance component estimated at zero (a singular fit) is expected when that source of variation is absent and is not reported.

Cell-type-specific effects. Under treatment contrasts the exposure coefficient is the effect at the reference cell type and each exposure:celltype coefficient is a difference from it. Each cell type's effect is therefore a linear combination of fixed effects, tested with contest1D so that the degrees of freedom and standard error account for the covariance between the main effect and the interaction.

family = "nbinom". Satterthwaite and Kenward-Roger are linear -mixed-model constructions and do not extend to a generalised LMM, so the negative-binomial path fits lme4::glmer.nb on an equivalent cell-type-nested reparametrisation of exposure * celltype (one exposure slope column per cell type) and tests each cell type's effect by a likelihood-ratio test (anova of the full versus the column-dropped model) – again not a Wald-normal test. This path is markedly more expensive (one glmer.nb refit per cell type per gene); for count data prefer the donor-level run_sc_exwas_pb_offset, which is calibration- equivalent and far faster (Lee & Han 2024). The ddf argument is ignored for family = "nbinom".

Missing values. The cells of a donor whose exposure or a covariate is missing, and cells without a cell-type label, are left out, with a message, before either family is fitted, so that the full and reduced models of a likelihood-ratio test use the same cells.

Value

A DataFrame, one row per gene x cell type: gene, celltype, log2FC, se, statistic (Satterthwaite /KR t for gaussian, LR χ^2 for nbinom), df (finite denominator df; 1 for the nbinom LRT), pvalue, padj_global (BH across all gene-by-cell-type strata, as in run_sc_exwas; there is no within-cell-type padj), and method. Rows ordered by padj_global, then pvalue. A gene-by-cell-type stratum that cannot be fitted is left out of the result; with family = "nbinom", a stratum whose likelihood-ratio test cannot be computed is kept with NA pvalue and padj_global. Its statistic is NA when the reduced model or anova() fails, and close to 0 when the test has no degrees of freedom, as when a covariate is a linear function of the exposure. padj_global adjusts over the strata with a p-value, and the function stops when no stratum can be fitted.

Note

Calibrate against a donor-permutation null, with the exposure permuted across donors, before quoting any type-I rate; Satterthwaite/KR control the finite-sample error only under the working model, so the empirical check is the headline.

References

Kuznetsova A, Brockhoff PB & Christensen RHB (2017). lmerTest package: tests in linear mixed effects models. *J Stat Softw* 82(13).

Kenward MG & Roger JH (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics* 53:983-997.

Halekoh U & Hojsgaard S (2014). A Kenward-Roger approximation and parametric bootstrap methods for tests in linear mixed models – the R package pbkrtest. *J Stat Softw* 59(9).

See Also

`run_sc_exwas_pb_offset` for the donor-level default.

Examples

```
donor_ids <- paste0("D", seq_len(10L))
exposure <- rep(c(0, 1), each = 5L)
cell_donor <- rep(donor_ids, each = 12L)
cell_type <- rep(
    rep(c("TypeA", "TypeB"), each = 6L),
    times = length(donor_ids)
)
cell_ids <- paste0("cell", seq_along(cell_donor))
counts <- outer(
    seq_len(2L),
    seq_along(cell_ids),
    function(gene, cell) 2L + ((7L * gene + 5L * cell) %% 11L)
)
exposed_type_a <- exposure[match(cell_donor, donor_ids)] == 1 &
    cell_type == "TypeA"
counts[1L, exposed_type_a] <- counts[1L, exposed_type_a] + 3L
storage.mode(counts) <- "integer"
dimnames(counts) <- list(c("G1", "G2"), cell_ids)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(counts = counts),
    colData = S4Vectors::DataFrame(
        cell_id = cell_ids,
        donor_id = cell_donor,
        cell_type = cell_type
    )
)
donor_design <- matrix(exposure, ncol = 1L,
    dimnames = list(donor_ids, "exposure"))
scee <- build_scee(sce, donor_design, sample_col = "donor_id")
if (requireNamespace("lme4", quietly = TRUE) &&
        requireNamespace("lmerTest", quietly = TRUE)) {
    mixed_result <- run_sc_exwas_glmm(
        scee,
        exposure = "exposure",
        celltype_col = "cell_type",
```

```
        target_genes = "G1",
        family = "gaussian",
        BPPARAM = BiocParallel::SerialParam()
    )
    mixed_result
}
```

`run_sc_exwas_pb_offset` *Donor offset-pseudobulk negative-binomial sc-ExWAS*

Description

A donor-level negative-binomial alternative to `run_sc_exwas`. Per cell type and per gene, aggregates cells to a donor pseudobulk count and fits a negative-binomial GLM with a library-size offset $\log s_i = \log(\text{donor total counts})$. Lee and Han (2024) report that pseudobulk models with proper offsets have the same statistical properties as generalised linear mixed models in single-cell case-control studies, without the pseudoreplication that inflates cell-level tests. P-values are Wald tests with an estimated dispersion and can be anticonservative when there are few donors. The function has no threshold on the number of cells: every donor with at least one cell of a cell type enters the model of that cell type, where `run_sc_exwas` leaves out a donor with fewer than `min_cells` cells.

Usage

```
run_sc_exwas_pb_offset(
  x,
  exposure,
  celltype_col = "cell_type",
  covariates = NULL,
  target_genes = NULL,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

`x`	A `SingleCellExposomeExperiment`. Each cell's donor is read from the `colData` column named by `sampleColumn(x)`.
`exposure`	Character; a donor-level column of `exposureData(x)`.
`celltype_col`	Character. Column in `colData` containing cell type annotations. Default `"cell_type"`.
`covariates`	Character vector; donor covariate columns in `exposureData` (numeric), added as GLM main effects. The names `y`, `A` and `logoff` are used by the model and cannot be covariate names, and each name must be a syntactic R name (see `make.names`) other than `.`, `...` and names such as `..1`, which R reads specially in a formula. The exposure itself cannot be a covariate.
`target_genes`	Character; genes to test (default: all rows of `x`).
`BPPARAM`	A `BiocParallel` param used to parallelise over genes. Defaults to `BiocParallel::SerialParam()`, so no worker process is started unless one is asked for; pass, for example, `BiocParallel::SnowParam(workers = 2L)` to fit genes in parallel. Pass `NULL` to use a plain `lapply`.

Value

A DataFrame, one row per gene x cell type: gene, celltype, log2FC (exposure effect on the $\log_2$ rate scale), se (on the same scale), statistic (donor-level Wald z from the NB GLM), pvalue, and padj_global (BH across all tested gene-by-cell-type strata, as in run_sc_exwas; there is no within-cell-type padj). Rows ordered by padj_global, then pvalue. A gene whose GLM cannot be fitted gets NA estimates; the function stops when no cell type has at least three donors with cells.

Note

Report empirical calibration from a donor-permutation null, with the exposure permuted across donors, before quoting any type-I rate.

References

Lee H & Han B (2024). Pseudobulk with proper offsets has the same statistical properties as generalized linear mixed models in single-cell case-control studies. *Bioinformatics* 40(8):btae498. doi:10.1093/bioinformatics/btae498

Squair JW et al. (2021). Confronting false discoveries in single-cell differential expression. *Nat Commun* 12:5692.

See Also

run_sc_exwas_glmm for the cell-level LMM reference.

Examples

```
donor_ids <- paste0("D", seq_len(10L))
exposure <- rep(c(0, 1), each = 5L)
cell_donor <- rep(donor_ids, each = 10L)
cell_type <- rep("TypeA", length(cell_donor))
cell_ids <- paste0("cell", seq_along(cell_donor))
pb_targets <- rbind(
    G1 = c(30, 140, 50, 180, 40, 110, 260, 70, 220, 130),
    G2 = c(200, 40, 240, 60, 300, 50, 270, 80, 330, 70),
    G3 = c(100, 120, 90, 150, 110, 130, 170, 100, 160, 140),
    G4 = c(250, 200, 300, 180, 270, 220, 320, 210, 290, 230)
)
counts <- matrix(0L, nrow = 4L, ncol = length(cell_ids))
for (donor in seq_along(donor_ids)) {
    counts[, cell_donor == donor_ids[donor]] <- pb_targets[, donor] / 10L
}
storage.mode(counts) <- "integer"
dimnames(counts) <- list(paste0("G", seq_len(4L)), cell_ids)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(counts = counts),
    colData = S4Vectors::DataFrame(
        cell_id = cell_ids,
        donor_id = cell_donor,
        cell_type = cell_type
    )
)
donor_design <- matrix(exposure, ncol = 1L,
    dimnames = list(donor_ids, "exposure"))
scee <- build_scee(sce, donor_design, sample_col = "donor_id")
offset_result <- run_sc_exwas_pb_offset(
```

```
    scee,
    exposure = "exposure",
    celltype_col = "cell_type",
    target_genes = c("G1", "G2"),
    BPPARAM = BiocParallel::SerialParam()
)
head(offset_result)
```

`run_sc_mixture` *Cell-type-specific mixture exposure analysis*

Description

Estimates the relative contribution of multiple correlated exposures to gene expression within a specific cell type using a simplified quantile-scored linear model on pseudobulk data.

Usage

```
run_sc_mixture(
  x,
  exposures,
  celltype,
  celltype_col = "cell_type",
  target_genes = NULL,
  covariates = NULL,
  q = 4L,
  min_cells = 10L
)
```

Arguments

`x`	A `SingleCellExposomeExperiment`. Each cell's donor is read from the `colData` column named by `sampleColumn(x)`.
`exposures`	Character vector. Exposure variable names, distinct. Each name must be a syntactic R name (see `make.names`) because it goes into the model formula, and cannot be `y`, the name of the response. Nor can it be `.`, `...` or a name such as `..1`, which R reads specially in a formula.
`celltype`	Character. Which cell type to analyse.
`celltype_col`	Character. `colData` column with cell type labels. Default `"cell_type"`.
`target_genes`	Character vector (optional). Genes to test. Default: top 20 most variable genes.
`covariates`	Character vector (optional). Additional covariates from `exposureData`; a name that is not a column of `exposureData` is an error, and so is an exposure listed here. Names must be syntactic and cannot be `y`, `.`, `...` or a name such as `..1`, as for `exposures`.
`q`	Integer. Number of quantile bins, at least 2. Default 4. Tied exposure values (for example at a limit of detection) can give fewer bins.
`min_cells`	Integer. Minimum cells per donor. Default 10.

Details

This is a **simplified screening method**, not a formal quantile-based g-computation (Keil et al. 2020) or weighted quantile sum regression (Carrico et al. 2015). The approach:

1. Donors with a missing exposure or covariate are removed, and exposures are scored into quantile bins (1 to q; fewer when quantiles are tied).
2. For each target gene, a linear model is fit on pseudobulk log-CPM with all quantile-scored exposures and covariates as predictors.
3. Weights are the absolute coefficients of the exposures that have one, averaged across target genes and normalised to sum to 1; an exposure without an estimable coefficient gets the weight `NA` (see Value). Because every exposure is on the same quantile scale, the coefficients are directly comparable.

Limitations:

- No bootstrap confidence intervals on weights.
- Does not separate positive/negative weight directions as in formal qgcomp.
- Log-CPM may have heteroscedasticity at low counts.

For rigorous mixture analyses suitable for publication, consider `qgcomp::qgcomp.noboot()` or `gWQS::gwqs()` applied to the pseudobulk counts returned by `donor_pseudobulk`, or `run_mixture_qgcomp`.

Value

A `DataFrame` with one row per target gene with a fitted model: `gene` and `mixture_effect` (the sum of that gene's estimable quantile-scored exposure coefficients; `NA` when none is estimable). `metadata(result)` holds:

weights Named numeric vector. Relative contribution of each exposure (the weights that are not `NA` sum to 1): the mean over target genes of the absolute coefficients of the quantile-scored exposures, normalised. When the quantile scores of exposures are collinear, for example for two exposures with the same ranks, `lm()` estimates a coefficient only for the one listed first in `exposures`; it carries their joint effect, and the later ones get the weight `NA`, with a warning. When no coefficient can be estimated, every weight is `NA`.

mixture_coef Overall mixture coefficient (sum of quantile-scored exposure coefficients, averaged across target genes); `NA` when no coefficient can be estimated.

method Character. Always `"quantile_linear"`.

celltype Character. Cell type analysed.

n_donors Integer. Donors with complete exposure and covariate data used in the analysis.

n_genes Integer. Genes with a fitted model.

References

Keil AP et al. (2020). A quantile-based g-computation approach to addressing the effects of exposure mixtures. *Environ Health Perspect* 128:047004. doi:10.1289/EHP5838

Carrico C et al. (2015). Characterization of weighted quantile sum regression for highly correlated data in a risk analysis setting. *J Agric Biol Environ Stat* 20:100-120.

Examples

```
library(SingleCellExperiment); library(S4Vectors)
set.seed(1)
counts <- matrix(rpois(5000, 8), nrow = 50,
    dimnames = list(paste0("G", 1:50), paste0("c", 1:100)))
sce <- SingleCellExperiment(assays = list(counts = counts),
    colData = DataFrame(cell_id = paste0("c", 1:100),
        donor_id = rep(paste0("D", 1:10), each = 10),
        cell_type = rep(c("Mono", "NK"), 50)))
exp_mat <- matrix(rnorm(30), nrow = 10,
    dimnames = list(paste0("D", 1:10), c("E1", "E2", "E3")))
scee <- build_scee(sce, exp_mat, sample_col = "donor_id")
mix <- run_sc_mixture(scee, exposures = c("E1", "E2", "E3"),
    celltype = "Mono", celltype_col = "cell_type",
    min_cells = 3L)
metadata(mix)$weights
```

---

seurat_to_exposure *Convert Seurat metadata to exposure matrix*

Description

Extracts donor-level metadata from a Seurat object's `meta.data` and formats it as an exposure matrix for `build_scee`.

Usage

```
seurat_to_exposure(seurat_meta, sample_col, exposure_cols)
```

Arguments

`seurat_meta`	A data.frame (from `seurat_obj@meta.data`).
`sample_col`	Character. Column with donor IDs.
`exposure_cols`	Character vector. Columns to use as exposures.

Value

A numeric matrix (donors x exposures). Each value is the mean over the donor's cells; a warning names columns whose values vary within a donor, and non-numeric columns are an error.

Examples

```
meta <- data.frame(
    donor = rep(c("D1","D2","D3"), each = 10),
    PM2.5 = rep(c(12, 25, 8), each = 10),
    age = rep(c(45, 32, 67), each = 10))
exp_mat <- seurat_to_exposure(meta,
    sample_col = "donor",
    exposure_cols = c("PM2.5", "age"))
dim(exp_mat)
```

`simulate_crossomic_network` *Simulate cross-omic network data with known structure*

Description

Generates donor-level metabolite data and cell-type-specific transcript pseudobulk counts whose latent log-expression follows a known Gaussian graphical model, with cell-type heterogeneity and optional exposure-dependent composition. Intended for benchmarking edge recovery and false-positive control.

Usage

```
simulate_crossomic_network(
  n_donors = 100L,
  n_celltypes = 4L,
  n_transcripts = 30L,
  n_metabolites = 10L,
  n_cells_per_donor = 200L,
  edge_density = 0.1,
  shared_edge_frac = 0.6,
  cross_omic_frac = 0.2,
  exposure_effect = 0.5,
  composition_confounding = TRUE,
  seed = NULL
)
```

Arguments

`n_donors`	Integer; number of donors. Default 100.
`n_celltypes`	Integer; number of cell types. Default 4.
`n_transcripts`	Integer; number of transcript features. Default 30.
`n_metabolites`	Integer; number of metabolite features. Default 10.
`n_cells_per_donor`	Integer; cells per donor, allocated to cell types by the composition. Pseudobulk depth scales with the number of cells. Default 200.
`edge_density`	Numeric; proportion of possible edges present in the base network. Default 0.1.
`shared_edge_frac`	Numeric; fraction of base-network edges kept by cell types 1 and 2, each of which also gains the same number of new edges. Default 0.6.
`cross_omic_frac`	Numeric; fraction of edges that are cross-omic. Default 0.2.
`exposure_effect`	Numeric; effect of the exposure on the mean of affected features and on composition. Default 0.5.
`composition_confounding`	Logical; make the proportions of cell types 1 and 2 depend on the exposure. Default TRUE.
`seed`	Integer or NULL; random seed. The caller's random number generator state is restored on return. Default NULL.

Details

Simulation design:

1. A sparse positive-definite base precision matrix Θ is generated over transcripts and metabolites with within-omic and cross-omic edges.
2. Cell types 1 and 2 keep a fraction `shared_edge_frac` of the base edges and gain as many new edges; cell type 3 has an independently generated precision matrix; cell types 4 and above have no transcript-transcript or cross-omic edges.
3. Metabolites are drawn once per donor from a multivariate normal distribution whose precision is the metabolite block of the base matrix, P_m.
4. For each cell type, latent transcript log-expression is drawn from its conditional distribution given the metabolites, $N(-\Theta_{tt}^{-1}\Theta_{tm}m, \Theta_{tt}^{-1})$. The joint precision of transcripts and metabolites is then Θ_{tt}, Θ_{tm} and $P_m + \Theta_{mt}\Theta_{tt}^{-1}\Theta_{tm}$; it is returned in `true_networks`, so its metabolite block can contain edges induced by the cross-omic edges.
5. The exposure shifts the mean of the first 30% of metabolites and, in cell types 1 and 2, of the first 20% of transcripts; it does not change any network.
6. Pseudobulk counts are Poisson with mean equal to the number of cells times $\exp(\text{latent} + 1)$, which makes them overdispersed relative to a Poisson with a fixed mean.
7. If `composition_confounding = TRUE`, the proportions of cell types 1 and 2 increase with the exposure, so pseudobulk depth varies with the exposure.

Value

List with:

pseudobulk List of count matrices (donors x transcripts), one per cell type.

metabolites Metabolite matrix (donors x metabolites), shared by all cell types.

exposure Named numeric vector of exposure values.

composition Donor-by-cell-type matrix of proportions.

n_cells Donor-by-cell-type matrix of simulated cell numbers.

true_networks List of precision matrices, one per cell type, of the latent transcript log-expression and the metabolites conditional on the exposure.

true_adjacency List of the corresponding adjacency matrices.

celltype_names Character vector of cell type names.

node_info DataFrame of feature metadata.

params List of simulation parameters.

Examples

```
sim <- simulate_crossomic_network(
    n_donors = 50, n_transcripts = 20,
    n_metabolites = 5, seed = 42)
names(sim)
sim$true_adjacency[[1]][1:5, 1:5]
```

`SingleCellExposomeExperiment-class` *SingleCellExposomeExperiment: Container for single-cell exposome data*

Description

Extends `SingleCellExperiment` with sample-level environmental exposure measurements. This enables cell-type-specific exposome-wide association studies (sc-ExWAS).

Usage

```
## S4 method for signature 'SingleCellExposomeExperiment,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'SingleCellExposomeExperiment'
cbind(..., deparse.level = 1)

## S4 method for signature 'SingleCellExposomeExperiment'
rbind(..., deparse.level = 1)

## S4 replacement method for signature 'SingleCellExposomeExperiment,DataFrame'
colData(x, ...) <- value

## S4 replacement method for signature 'SingleCellExposomeExperiment,NULL'
colData(x, ...) <- value

## S4 replacement method for signature 'SingleCellExposomeExperiment,list'
dimnames(x) <- value

## S4 replacement method for signature 'SingleCellExposomeExperiment,NULL'
dimnames(x) <- value

## S4 method for signature 'SingleCellExposomeExperiment'
show(object)
```

Arguments

`x`	A SingleCellExposomeExperiment.
`i`	Row (gene) subscript.
`j`	Column (cell) subscript.
`...`	Additional arguments.
`drop`	Logical. Ignored (kept for S4 compatibility).
`deparse.level`	Ignored.
`value`	The replacement `colData` or `dimnames`.
`object`	A SingleCellExposomeExperiment.

Details

The donor of each cell is read from one column of `colData`, whose name is stored in the object (see `sampleColumn`). The cell-to-donor mapping is not stored a second time, so renaming cells, combining objects and editing `colData` cannot leave two copies out of step. The object is valid when every donor in that column has exactly one row in `exposureData` and every row of `exposureData` belongs to at least one cell.

Replacing `colData`, or a column of it with `$<-` or `[[<-`, checks validity again, so an edit that removes the donor column or introduces a donor without exposure data is an error; so does replacing the cell names with `colnames<-`. To change donor identifiers, edit `exposureData` and `colData` together and rebuild the object with `build_scee`.

Value

The subsetting method returns a valid `SingleCellExposomeExperiment`. When cells are subset, its `exposureData` is restricted to the donors that keep at least one cell; feature subsetting leaves the donor-level slots unchanged. `cbind` combines objects with the same exposure variables, exposure metadata and sample column; donors present in more than one object must have identical exposure values. Every argument to `cbind` or `rbind` must be a `SingleCellExposomeExperiment`: when a plain `SingleCellExperiment` comes first, R dispatches to the `SingleCellExperiment` method, which returns a `SingleCellExperiment` without the exposure data (with only a warning about ambiguous dispatch), so convert such objects with `build_scee` first. `rbind` combines objects with the same cells and different features; they must have the same sample column and `exposureInfo`, and the same exposure values for each donor. The replacement methods return the modified object. The `show` method is called for its display side effect and returns `NULL` invisibly.

Slots

`exposureData` Numeric matrix. Rows = samples (donors), columns = exposure variables.

`exposureInfo` `DataFrame`. Metadata for each exposure (family, unit, LOD).

`sampleColumn` Character scalar. Name of the `colData` column holding each cell's sample (donor) identifier.

Data hierarchy

```
Donor (n=50-1000)
  |-- Exposures (PM2.5, Pb, BPA, ...)   <- sample level
  +-- Cells x Genes matrix
      |-- T cells (1000 cells)
      |-- NK cells (500 cells)            <- cell level
      +-- Monocytes (800 cells)
```

References

van der Wijst MGP et al. (2020). The single-cell eQTLGen consortium. *eLife* 9:e52155. doi:10.7554/eLife.52155

Examples

```
showClass("SingleCellExposomeExperiment")
```

Index

∗ **datasets**
hlca_airway_subset, 16
incov_pbmc_subset, 18
∗ **package**
exposomeSC-package, 3
.SCEE
(SingleCellExposomeExperiment-class), 68
[,SingleCellExposomeExperiment,ANY,ANY,ANY-method
(SingleCellExposomeExperiment-class), 68

as_scee, *4*, 5

build_scee, *3*, 6, *65*, *69*

cbind,SingleCellExposomeExperiment-method
(SingleCellExposomeExperiment-class), 68
cellSampleMap (exposureData), 12
cellSampleMap,SingleCellExposomeExperiment-method
(exposureData), 12
CelltypeNetwork-class, 7
CelltypeNetworkResult, 7, *7–9*, *20*, *25*, *26*, *30*, *35*, *45*
CelltypeNetworkResult-class, 8
celltypeSpecificEdges
(network-accessors), 20
celltypeSpecificEdges,NetworkComparison-method
(network-accessors), 20
colData<-,SingleCellExposomeExperiment,DataFrame-method
(SingleCellExposomeExperiment-class), 68
colData<-,SingleCellExposomeExperiment,NULL-method
(SingleCellExposomeExperiment-class), 68
comparisonSummary (network-accessors), 20
comparisonSummary,NetworkComparison-method
(network-accessors), 20

data.frame, *32*
diffEdges (network-accessors), 20
diffEdges,NetworkComparison-method
(network-accessors), 20
dimnames<-,SingleCellExposomeExperiment,list-method
(SingleCellExposomeExperiment-class), 68
dimnames<-,SingleCellExposomeExperiment,NULL-method
(SingleCellExposomeExperiment-class), 68
donor_pseudobulk, *3*, 9, *56*, *57*, *64*

estimate_power, *4*, 11
exposomeSC (exposomeSC-package), 3
exposomeSC-package, 3
exposure_impute_lod, *4*, 14
exposureData, 12
exposureData,SingleCellExposomeExperiment-method
(exposureData), 12
exposureData<- (exposureData), 12
exposureData<-,SingleCellExposomeExperiment-method
(exposureData), 12
exposureInfo (exposureData), 12
exposureInfo,SingleCellExposomeExperiment-method
(exposureData), 12
exposureInfo<- (exposureData), 12
exposureInfo<-,SingleCellExposomeExperiment-method
(exposureData), 12
exposureVariables (exposureData), 12
exposureVariables,SingleCellExposomeExperiment-method
(exposureData), 12

hlca_airway_subset, 16

incov_pbmc_subset, 18

make.names, *33*, *51*, *55*, *58*, *61*, *63*
metadata,CelltypeNetworkResult-method
(network-accessors), 20

network-accessors, 20
networkAdjacency (network-accessors), 20
networkAdjacency,CelltypeNetworkResult-method
(network-accessors), 20
networkCelltype (network-accessors), 20
networkCelltype,CelltypeNetworkResult-method
(network-accessors), 20
NetworkComparison, *20*, 23, *23*, *24*, *27*, *35*
NetworkComparison-class, 24

networkMethod (network-accessors), 20
networkMethod,CelltypeNetworkResult-method (network-accessors), 20
networkNodes (network-accessors), 20
networkNodes,CelltypeNetworkResult-method (network-accessors), 20
networkPrecision (network-accessors), 20
networkPrecision,CelltypeNetworkResult-method (network-accessors), 20
networkStability (network-accessors), 20
networkStability,CelltypeNetworkResult-method (network-accessors), 20

plot_celltype_network, *4*, *8*, 24
plot_edge_stability, *4*, 26
plot_network_comparison, *4*, 27

rbind,SingleCellExposomeExperiment-method (SingleCellExposomeExperiment-class), 68
run_cell_coupling, *3*, 31
run_celltype_network, *4*, *7, 8*, *20*, 28, *44*
run_comparative_network, *4*, *7*, *20*, *23–25*, *30*, 35
run_dose_response, *3*, 36, *37, 38*
run_dose_response_gam, *3*, 37
run_exposure_composition, *4*, 39
run_exposure_network, *4*, *7, 8*, *20*, 42
run_gsea, *4*, 46
run_interaction_test, *3*, 47
run_meta_exwas, *4*, 49
run_mixture_qgcomp, *3*, 50, *64*
run_multi_exwas, *3*, *46*, 53, *57*
run_sc_exwas, *3*, *9, 10*, *46*, *50*, *53*, 54, *54*, *59*, *61, 62*
run_sc_exwas,SingleCellExposomeExperiment-method (run_sc_exwas), 54
run_sc_exwas_glmm, *3*, 58, *62*
run_sc_exwas_pb_offset, *3*, *59, 60*, 61
run_sc_mixture, *3*, *50*, 63

sampleColumn, *6*, *10*, *28*, *32*, *36*, *38*, *40*, *43*, *48*, *51*, *53*, *55*, *58*, *61*, *63*, *69*
sampleColumn (exposureData), 12
sampleColumn,SingleCellExposomeExperiment-method (exposureData), 12
sampleColumn<- (exposureData), 12
sampleColumn<-,SingleCellExposomeExperiment-method (exposureData), 12
seurat_to_exposure, *4*, 65
sharedEdges (network-accessors), 20
sharedEdges,NetworkComparison-method (network-accessors), 20
show,CelltypeNetworkResult-method (CelltypeNetworkResult-class), 8
show,NetworkComparison-method (NetworkComparison-class), 24
show,SingleCellExposomeExperiment-method (SingleCellExposomeExperiment-class), 68
SimpleList, *10*
simulate_crossomic_network, *4*, 66
SingleCellExperiment, *6*, *68*
SingleCellExposomeExperiment, *5, 6*, *10*, *12, 13*, *16*, *18*, *28*, *36*, *38*, *40*, *43*, *48*, *51*, *53*, *55*, *58*, *61*, *63*
SingleCellExposomeExperiment-class, 68
SummarizedExperiment, *10*