

Package: scDonorAudit (via r-universe)

October 8, 2026

Title Audit Donor Influence in Single-Cell Pseudobulk Analyses

Version 0.99.9

Description Audits sample and donor design for two-condition single-cell pseudobulk differential expression analyses. Fits an edgeR quasi-likelihood model to each cell type and repeats the analysis after omitting each donor. Links planned donor omissions to design checks, fit outcomes and observed gene-effect changes. Provides descriptive summaries of donor-deletion sensitivity together with the coverage of valid effects.

License Artistic-2.0

URL <https://github.com/Emr-27/scDonorAudit>

BugReports <https://github.com/Emr-27/scDonorAudit/issues>

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

biocViews SingleCell, RNASeq, GeneExpression, DifferentialExpression, QualityControl, Software

Imports edgeR, ggplot2, IRanges, Matrix, methods, rlang, S4Vectors, SingleCellExperiment, stats, SummarizedExperiment

Suggests BiocStyle, knitr, rmarkdown, testthat (>= 3.2.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs zlib1g-dev

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-10-08 09:42:54 UTC

RemoteUrl <https://github.com/BiocStaging/scDonorAudit>

RemoteRef HEAD

RemoteSha ae6e56ac8d8e772d91c78e6587e238b6667fe819

Contents

assessDonorInfluence	2
auditDesign	4
plotInfluence	6
preparePseudobulk	7
scDonorAudit	9
summarizeInfluence	10

Index	12
--------------	-----------

assessDonorInfluence	<i>Assess the influence of omitting each donor</i>
----------------------	--

Description

Fits an edgeR quasi-likelihood model to a fixed baseline gene family in each cell type, then removes one donor at a time and refits. This is a sensitivity analysis, not an automatic rule for excluding donors.

Usage

```
assessDonorInfluence(
  pb,
  design,
  contrast,
  min_cells = NULL,
  covariates = character(),
  pair_policy = "strict",
  min_replicates_warn = 3L,
  filter_args = list(),
  backend_args = list()
)
```

Arguments

pb	Output of preparePseudobulk() . The counts assay must remain nonempty with finite, non-negative integer-valued counts and unique, nonempty gene IDs. These requirements and audit metadata consistency are rechecked on every call. Valid count edits, row subsets, and column reordering are supported.
design	Either 'independent', with exactly one registered sample per donor, or 'paired', with exactly one registered sample per condition for each donor.
contrast	Named character vector such as <code>c(numerator='stim', denominator='ctrl')</code> . The sample registry must contain exactly these two conditions.
min_cells	Optional minimum cell count per sample/cell type. The default NULL applies no cell-count threshold. A positive integer requires known <code>n_cells</code> for every observed pseudobulk and excludes samples below the threshold before applying the pairing policy.

covariates	Names of sample-level covariates retained during preparation. Names must be syntactic R names other than . and must not be sample_id, donor_id, condition, cell_type, n_cells, or pb_id.
pair_policy	'strict' blocks a cell type with an incomplete eligible pair; 'complete_pairs' drops both samples of that donor from that cell type's baseline analysis. This handles cell-type eligibility, not an incomplete pair in the sample registry; see auditDesign() for details.
min_replicates_warn	Positive integer threshold (default 3) for low_replication, computed for the baseline and each omission. The count is the smaller per-condition donor count for an independent design or the number of remaining complete pairs for a paired design. Counts below the threshold set the flag; a cell type blocked by the strict pairing policy has NA. This advisory flag leaves design eligibility unchanged.
filter_args	Named list of filterByExpr options. The default empty list resolves to min.count = 10, min.total.count = 15, large.n = 10 and min.prop = 0.7; supplied entries override these values. Filtering uses baseline raw counts grouped by condition within each cell type. Retained genes form the fixed family for all its deletion refits.
backend_args	Named list whose supported entry is the logical robust argument to glmQLFit, defaulting to TRUE.

Details

Each fit uses TMM normalization and calls `edgeR::glmQLFit` with `dispersion = NULL`, `abundance.trend = TRUE`, `legacy = FALSE` and `top.proportion = NULL`. These settings are fixed by the package. Arguments not explicitly supplied to `edgeR`, including `prior.count`, use the defaults of the installed `edgeR` version.

Value

A `SimpleList` with these components:

- `results`: per-cell-type gene-by-run effects, tests and validity flags.
- `fits`: baseline and deletion execution ledger.
- `fit_samples`: sample IDs, library sizes and normalization factors for completed fits.
- `spec`: per-cell-type plan status and planned omission counts.
- `sample_membership`: baseline inclusion decisions and reasons.
- `gene_filter`: the fixed baseline gene family and filtering decisions.
- `coverage`: registered sample/cell-type coverage from preparation.
- `issues`: design problems, backend warnings and errors.
- `config`: resolved analysis settings.
- `provenance`: preparation, package, R and `edgeR` versions and contrast.

In `fits`, `design_reason_code` preserves the pre-fit design audit even when `reason_code` later reports baseline or backend failure. `fit_samples` records the sample ID, library size, and normalization factor for every sample in each completed run.

Examples

```
count_file <- system.file('extdata', 'example_counts.csv',
                          package = 'scDonorAudit')
sample_file <- system.file('extdata', 'example_samples.csv',
                          package = 'scDonorAudit')
counts <- as.matrix(utils::read.csv(count_file, row.names = 1,
                                   check.names = FALSE))
samples <- utils::read.csv(sample_file)
se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(counts = counts),
  colData = S4Vectors::DataFrame(samples, cell_type = 'T'))
pb <- preparePseudobulk(se, 'sample_id', 'donor_id', 'condition',
                      'cell_type', sample_table = samples)
result <- assessDonorInfluence(pb, 'independent',
                              c(numerator = 'stim', denominator = 'ctrl'))
head(as.data.frame(result[['fits']]))
```

auditDesign

Audit a two-condition donor design

Description

Reports which samples enter each cell type's analysis and whether a model can be fit before and after omitting each included donor. Low replication is recorded as a reminder; structural model checks determine whether fitting can proceed.

Usage

```
auditDesign(
  pb,
  design,
  contrast,
  min_cells = NULL,
  covariates = character(),
  pair_policy = "strict",
  min_replicates_warn = 3L
)
```

Arguments

pb	Output of <code>preparePseudobulk()</code> . The counts assay must remain nonempty with finite, non-negative integer-valued counts and unique, nonempty gene IDs. These requirements and audit metadata consistency are rechecked on every call. Valid count edits, row subsets, and column reordering are supported.
design	Either 'independent', with exactly one registered sample per donor, or 'paired', with exactly one registered sample per condition for each donor.
contrast	Named character vector such as <code>c(numerator='stim', denominator='ctrl')</code> . The sample registry must contain exactly these two conditions.

min_cells	Optional minimum cell count per sample/cell type. The default NULL applies no cell-count threshold. A positive integer requires known n_cells for every observed pseudobulk and excludes samples below the threshold before applying the pairing policy.
covariates	Names of sample-level covariates retained during preparation. Names must be syntactic R names other than . and must not be sample_id, donor_id, condition, cell_type, n_cells, or pb_id.
pair_policy	'strict' blocks a cell type with an incomplete eligible pair; 'complete_pairs' drops both samples of that donor from that cell type's baseline analysis. This handles cell-type eligibility, not an incomplete pair in the sample registry; see auditDesign() for details.
min_replicates_warn	Positive integer threshold (default 3) for low_replication, computed for the baseline and each omission. The count is the smaller per-condition donor count for an independent design or the number of remaining complete pairs for a paired design. Counts below the threshold set the flag; a cell type blocked by the strict pairing policy has NA. This advisory flag leaves design eligibility unchanged.

Details

Only two-condition independent or paired designs are supported. Additional repeated measurements or multiple samples per donor and condition are unsupported. For a paired design, every donor must have both conditions in the sample registry before cell-type eligibility is assessed. A registry with a missing condition for any donor is rejected under either pair_policy.

Within a cell type, an observed sample is eligible if it also meets min_cells when that threshold is set. If only one member of a registered pair is eligible, pair_policy either blocks that cell type or excludes the donor from its baseline cohort. Thus 'complete_pairs' can handle a missing cell-type pseudobulk or a sample below the cell-count threshold, but cannot repair an incomplete sample registry.

Value

A SimpleList with these components:

- spec: per-cell-type cohort sizes, plan status and planned omissions.
- sample_membership: registered sample/cell-type combinations, baseline inclusion decisions and their reasons.
- runs: baseline and deletion design checks, including rank, residual degrees of freedom, target estimability and low-replication flags.
- issues: the subset of design run rows with non-OK reason codes.
- config: resolved design, contrast, covariates and thresholds.
- provenance: the preparation's audit universe.

The full field/type/status dictionary is installed at `system.file('OUTPUT_SCHEMA.md', package = 'scDonorAudit')`.

Examples

```
count_file <- system.file('extdata', 'example_counts.csv',
                           package = 'scDonorAudit')
sample_file <- system.file('extdata', 'example_samples.csv',
                           package = 'scDonorAudit')
counts <- as.matrix(utils::read.csv(count_file, row.names = 1,
                                   check.names = FALSE))
samples <- utils::read.csv(sample_file)
se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(counts = counts),
  colData = S4Vectors::DataFrame(samples, cell_type = 'T'))
pb <- preparePseudobulk(se, 'sample_id', 'donor_id', 'condition',
                       'cell_type', sample_table = samples)
audit <- auditDesign(pb, 'independent',
                    c(numerator = 'stim', denominator = 'ctrl'))
head(as.data.frame(audit[['runs']]))
```

plotInfluence

Plot donor coverage or gene-level influence

Description

Plot donor coverage or gene-level influence

Usage

```
plotInfluence(result, type, cell_type = NULL, gene_id = NULL)
```

Arguments

result	Output of assessDonorInfluence() .
type	One of 'coverage', 'influence', or 'effect'.
cell_type	Required for gene-level plots.
gene_id	Explicit gene IDs; a nonempty vector for 'influence' and one ID for 'effect'.

Details

Coverage plots show baseline inclusion for each registered sample/cell-type combination. Included samples have status included; excluded samples show their membership reason, such as missing coverage, insufficient cells or a pairing-policy exclusion. Unique sample IDs determine positions, with donor/sample labels displayed separately.

Influence tiles show the signed change in log2 fold-change: deletion effect minus baseline effect. Grey tiles mark unavailable comparisons, which may arise from design, backend, or gene-level status. Inspect fits and the gene status assay for the specific reason.

Effect plots show baseline and deletion log2 fold-changes; the dashed horizontal line marks the baseline effect. They use unique run IDs for positions and label deletions as omit: <donor ID>. Unavailable effects retain their position without a point.

Value

A ggplot object.

Examples

```
count_file <- system.file('extdata', 'example_counts.csv',
                          package = 'scDonorAudit')
sample_file <- system.file('extdata', 'example_samples.csv',
                          package = 'scDonorAudit')
counts <- as.matrix(utils::read.csv(count_file, row.names = 1,
                                   check.names = FALSE))
samples <- utils::read.csv(sample_file)
se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(counts = counts),
  colData = S4Vectors::DataFrame(samples, cell_type = 'T'))
pb <- preparePseudobulk(se, 'sample_id', 'donor_id', 'condition',
                       'cell_type', sample_table = samples)
result <- assessDonorInfluence(pb, 'independent',
                              c(numerator = 'stim', denominator = 'ctrl'))
plotInfluence(result, 'effect', cell_type = 'T', gene_id = 'g1')
```

preparePseudobulk	<i>Prepare donor-aware pseudobulk counts</i>
-------------------	--

Description

Sums raw counts from an in-memory SingleCellExperiment by gene, sample and cell type, or checks a preaggregated SummarizedExperiment. Missing sample/cell-type combinations are recorded in the preparation metadata's coverage table, rather than inserted as zero-expression columns. A sample registry shows samples that are entirely absent from the input object.

Usage

```
preparePseudobulk(
  x,
  sample_id,
  donor_id,
  condition,
  cell_type,
  assay = "counts",
  sample_vars = character(),
  sample_table = NULL,
  n_cells = NULL
)
```

Arguments

<code>x</code>	A <code>SingleCellExperiment</code> of cells or a <code>SummarizedExperiment</code> with one column for each observed sample/cell-type combination.
<code>sample_id, donor_id, condition, cell_type</code>	Names of metadata columns. If <code>sample_table</code> is supplied, <code>donor_id</code> and <code>condition</code> may be absent from the object; the table supplies them by <code>sample_id</code> .
<code>assay</code>	Name of the raw count assay.
<code>sample_vars</code>	Names of additional sample-level metadata columns to keep.
<code>sample_table</code>	Optional <code>data.frame</code> with canonical columns <code>sample_id</code> , <code>donor_id</code> , <code>condition</code> , and selected <code>sample_vars</code> .
<code>n_cells</code>	For already aggregated input, name of the metadata column containing the number of cells. It is computed for single-cell input.

Details

Counts must be a numeric base matrix or a numeric `Matrix` object containing finite non-negative integer values. Logical and pattern matrices are rejected with `INVALID_COUNTS`. `DelayedArray` and disk-backed assays are not supported. Cell-level sparse counts are aggregated without converting the whole input to a dense matrix. The edgeR backend converts only a cell type's much smaller pseudobulk matrix to dense storage. For larger datasets, aggregate raw counts upstream and supply an in-memory `SummarizedExperiment`, a full sample registry, and `n_cells` if a minimum cell-count threshold will be used. Memory use also depends on the number of retained genes, cell types and planned donor deletions.

Each output column represents one observed sample/cell-type combination. Its `colData` records `pb_id`, `sample_id`, `donor_id`, `condition`, `cell_type`, `n_cells`, and any retained `sample_vars`. For single-cell input, `n_cells` is the number of cells summed into that column. For preaggregated input, it comes from the specified metadata column or is `NA` when `n_cells` is not supplied.

`metadata(output)$scdonoraudit` stores the sample registry, preparation settings, and coverage. The coverage table crosses registered samples with the cell types present in the input. Missing combinations have `observed = FALSE` and `n_cells = NA`; they do not create count columns.

Value

A `SummarizedExperiment` with a raw counts assay, one row per input gene sorted by gene ID, and one column per observed sample/cell-type combination. `colData` maps columns to their sample and cell type; `metadata` stores the sample registry, coverage table, and preparation settings.

Examples

```
count_file <- system.file('extdata', 'example_counts.csv',
  package = 'scDonorAudit')
sample_file <- system.file('extdata', 'example_samples.csv',
  package = 'scDonorAudit')
counts <- as.matrix(utils::read.csv(count_file, row.names = 1,
  check.names = FALSE))
samples <- utils::read.csv(sample_file)
se <- SummarizedExperiment::SummarizedExperiment(
```

```

assays = list(counts = counts),
colData = S4Vectors::DataFrame(samples, cell_type = 'T'))
pb <- preparePseudobulk(se, 'sample_id', 'donor_id', 'condition',
                        'cell_type', sample_table = samples)
pb

```

scDonorAudit

Audit donor omission in two-condition pseudobulk analyses

Description

scDonorAudit describes how a specified edgeR quasi-likelihood analysis changes after omitting each biological donor. It supports independent donors or paired donors with two conditions and optional sample covariates. Planned omissions are linked to design checks, fit outcomes and observed gene-effect changes.

Workflow

[preparePseudobulk\(\)](#) aggregates cells or validates preaggregated counts. [auditDesign\(\)](#) records coverage, cohort choices and design viability. [assessDonorInfluence\(\)](#) fits the baseline and planned donor omissions. [summarizeInfluence\(\)](#) reports observed changes and valid-effect coverage. [plotInfluence\(\)](#) shows coverage, effects and changes with missingness.

Supported designs

The sample registry must contain exactly the two contrast conditions. An independent design requires one registered sample per donor. A paired design requires one registered sample per condition for every donor. Additional repeated measurements or multiple samples per donor and condition are unsupported.

Paired registry entries must be complete under both pairing policies. Within a cell type, a missing pseudobulk or a sample below `min_cells` can leave only one eligible member of a pair. The default `pair_policy='strict'` blocks that cell type; `'complete_pairs'` excludes that donor from the cell type's baseline cohort. It does not repair missing pairs in the registry.

Inputs and interoperability

Inputs use `SingleCellExperiment` or `SummarizedExperiment` and in-memory numeric base or `Matrix` count matrices. Numeric sparse cell-level counts stay sparse during aggregation. Logical/pattern and delayed or disk-backed assays are unsupported. Already aggregated raw counts can be supplied via `SummarizedExperiment` with a sample registry and cell counts. Results use `SimpleList`, `DataFrame`, `SummarizedExperiment` and `CharacterList`.

Output and interpretation

The baseline gene-filtering family and contrast are fixed within each cell type. Library normalization and model parameters are re-estimated for every completed deletion. Failed or skipped omissions remain in `n_planned`. Missing effects are not zero changes; observed effect ranges are not confidence intervals. Empty summaries retain the same 16 typed columns. Read effect

changes together with valid-fit coverage. A large change can reflect biological heterogeneity or limited precision; donor-exclusion decisions require independent quality information and the study design. Adjusted p values describe the edgeR tests within each cell type and run; they do not control error across cell types or deletion runs, or test whether a donor is invalid. The installed output dictionary lists all fields, types and status codes: `system.file('OUTPUT_SCHEMA.md', package = 'scDonorAudit')`.

Data and methods

Help examples use synthetic negative-binomial counts (seed 1103). Bundled data notes and generation scripts are installed in `extdata` and `scripts`. The vignette uses attributed Crowell19 Astrocyte pseudobulk counts and includes method and data citations. For edgeR, use `edgeR::edgeR` and `citation('edgeR')`.

Author(s)

Maintainer: Fuhao Jiang <emr39515@gmail.com> ([ORCID](#))

Authors:

- Fuhao Jiang <emr39515@gmail.com> ([ORCID](#))

See Also

- [preparePseudobulk\(\)](#)
- [auditDesign\(\)](#)
- [assessDonorInfluence\(\)](#)
- [summarizeInfluence\(\)](#)
- [plotInfluence\(\)](#)

summarizeInfluence	<i>Summarize observed donor influence on each gene</i>
--------------------	--

Description

Summaries combine observed effect changes with the coverage of valid deletion effects. Missing or failed deletions remain in `n_planned`. Interpret changes alongside `n_planned` and `n_effect_valid`; the material threshold distinguishes sign reversals from changes near zero.

Usage

```
summarizeInfluence(result, effect_threshold = NULL)
```

Arguments

<code>result</code>	Output of assessDonorInfluence() .
<code>effect_threshold</code>	Optional positive absolute log2 fold-change threshold for a material sign reversal, or NULL. See Details.

Details

A material sign reversal requires opposite signs for baseline and deletion effects, with both absolute values strictly greater than `effect_threshold`. For a valid baseline, `baseline_near_zero` uses `abs(baseline_logFC) <= effect_threshold`.

At a threshold of 0.5, a change from 0.8 to -0.7 is material, but 0.5 to -0.7 is not.

With NULL, `n_material_reversal` is integer NA and `baseline_near_zero` is logical NA. All other summary columns are still calculated.

Value

A `DataFrame` with one row per retained gene and cell type. With no retained genes, the result has zero rows and the same 16 typed columns.

Minimum and maximum effects are observed ranges, not confidence intervals or significance tests. Unavailable effects remain missing.

The `max_influence_donors` column is a `CharacterList`. See [scDonorAudit](#) and the installed `OUTPUT_SCHEMA.md` for field meanings.

Examples

```
count_file <- system.file('extdata', 'example_counts.csv',
  package = 'scDonorAudit')
sample_file <- system.file('extdata', 'example_samples.csv',
  package = 'scDonorAudit')
counts <- as.matrix(utils::read.csv(count_file, row.names = 1,
  check.names = FALSE))
samples <- utils::read.csv(sample_file)
se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(counts = counts),
  colData = S4Vectors::DataFrame(samples, cell_type = 'T'))
pb <- preparePseudobulk(se, 'sample_id', 'donor_id', 'condition',
  'cell_type', sample_table = samples)
result <- assessDonorInfluence(pb, 'independent',
  c(numerator = 'stim', denominator = 'ctrl'))
influence <- summarizeInfluence(result, effect_threshold = 0.5)
head(as.data.frame(influence)[, c('gene_id', 'n_effect_valid',
  'max_abs_delta_observed')])
```

Index

`assessDonorInfluence`, [2](#)
`assessDonorInfluence()`, [6](#), [9](#), [10](#)
`auditDesign`, [4](#)
`auditDesign()`, [3](#), [5](#), [9](#), [10](#)

`edgeR::edgeR`, [10](#)

`plotInfluence`, [6](#)
`plotInfluence()`, [9](#), [10](#)
`preparePseudobulk`, [7](#)
`preparePseudobulk()`, [2](#), [4](#), [9](#), [10](#)

`scDonorAudit`, [9](#), [11](#)
`scDonorAudit-package (scDonorAudit)`, [9](#)
`summarizeInfluence`, [10](#)
`summarizeInfluence()`, [9](#), [10](#)