Package: RscDRS (via r-universe)

November 26, 2026

Title Single-Cell Disease-Relevance Score

Version 0.99.0

Description A native R implementation of the Python package scDRS (single-cell disease-relevance score; Zhang, Hou et al., Nature Genetics 2022; <https://github.com/martinjzhang/scDRS>). scDRS links single-cell RNA-seq data to polygenic disease risk by scoring each cell for excess expression of a GWAS-derived disease gene set relative to matched control gene sets, and provides downstream tests for cell-type association and heterogeneity. The package works on SingleCellExperiment objects.

License MIT + file LICENSE

URL https://github.com/sinar9776/RscDRS, https://github.com/martinjzhang/scDRS

BugReports https://github.com/sinar9776/RscDRS/issues

Depends R (>= 4.5.0), SingleCellExperiment

Imports BiocNeighbors, BiocSingular, Matrix, methods, S4Vectors, stats, SummarizedExperiment, utils

Suggests BiocStyle, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), uwot

VignetteBuilder knitr

biocViews Software, SingleCell, GeneExpression, Transcriptomics, GenomeWideAssociation, GeneSetEnrichment

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-11-26 20:21:21 UTC

RemoteUrl https://github.com/BiocStaging/RscDRS

RemoteRef HEAD

RemoteSha 9ceb0b23fba3dca7c7da0ca302818f88e3803c97

Contents

scdrsBuildGraph . . . 2
scdrsComputeGearysC . . . 3
scdrsComputeGeneStats . . . 4
scdrsComputeScore . . . 6
scdrsGetHomologs . . . 7
scdrsLoadToyData . . . 8
scdrsMetaAnalyze . . . 8
scdrsMungeGeneSets . . . 9
scdrsNormalize . . . 10
scdrsPerformDownstream . . . 11
scdrsPreprocess . . . 12
scdrsRankGenes . . . 14
scdrsReadGeneSets . . . 15
scdrsScoreCells . . . 16
scdrsSelectControlGenes . . . 18
scdrsTestCorrelation . . . 19
scdrsTestGearysC . . . 20
scdrsTestGroups . . . 21
scdrsTestOverlap . . . 23
scdrsWriteScores . . . 24
scdrsZToP . . . 25

Index **26**

`scdrsBuildGraph` *Build a cell-cell kNN graph*

Description

Builds the nearest-neighbour graph that the scDRS heterogeneity test (`scdrsTestGroups()`) uses, in the same way as `scdrs perform-downstream` does with scanpy (`sc.pp.pca(n_comps = 20)` and `sc.pp.neighbors(n_neighbors = 15)`): a PCA of the gene-centered expression, exact Euclidean nearest neighbours in PCA space, and UMAP connectivities (fuzzy simplicial set) as edge weights.

Usage

```
scdrsBuildGraph(
  sce,
  assay = "logcounts",
  nNeighbors = 15L,
```

```
  nPCs = 20L,
  name = "connectivities"
)
```

Arguments

sce	A SingleCellExperiment::SingleCellExperiment.
assay	Expression assay to use (log-normalized).
nNeighbors	Number of neighbours per cell, the cell itself included (as in scanpy).
nPCs	Number of principal components; at most `min(dim(sce)) - 1`.
name	Name under which the graph is stored in `colPair(sce)`.

Details

The PCA uses **BiocSingular** (exact SVD for up to 2e7 matrix entries, IRLBA otherwise) and the exact neighbour search uses **BiocNeighbors**; the edge weights are UMAP connectivities, as in scanpy and scDRS. Any other graph, e.g. from **bluster** or **scran**, can be used instead by storing it in `colPair(sce)` or passing it to `scdrsTestGroups()`. The graph is close to scanpy's but not always identical, because PCA and neighbour search differ slightly between implementations.

Value

`sce` with the graph in `colPair(sce, name)` (a symmetric weighted graph) and the PCA in `reducedDim(sce, "scdrs_pca")`.

Examples

```
sce <- scdrsBuildGraph(scdrsLoadToyData())
sce
g <- SingleCellExperiment::colPair(sce, "connectivities", asSparse = TRUE)
summary(Matrix::rowSums(g > 0))
```

---

scdrsComputeGearysC *Geary's C statistic*

Description

Computes Geary's C, a measure of spatial autocorrelation, of one or several cell-level variables on a cell-cell graph. Small values mean that neighbouring cells have similar values. This is the R port of `scdrs.method.gearys_c()`.

Usage

```
scdrsComputeGearysC(graph, x)
```

Arguments

graph	A cells x cells matrix of edge weights (sparse or dense), e.g. from scdrsBuildGraph().
x	A numeric vector with one value per cell, or a matrix with one row per cell (one statistic per column).

Details

$$C = \frac{(N-1)\sum_{i,j} w_{ij}(x_i - x_j)^2}{2W\sum_i (x_i - \bar{x})^2},$$

with $W = \sum_{ij} w_{ij}$. It is computed for all columns at once from $\sum_{ij} w_{ij}(x_i - x_j)^2 = \sum_i r_i x_i^2 + \sum_j k_j x_j^2 - 2x^\top W x$ (row sums r, column sums k), on the centered values.

Value

A numeric vector with one value per column of x.

Examples

```
g <- Matrix::sparseMatrix(i = c(1, 2, 2, 3), j = c(2, 1, 3, 2),
    x = 1, dims = c(3, 3))
scdrsComputeGearysC(g, cbind(smooth = c(1, 2, 3), rough = c(1, 3, 1)))
```

---

scdrsComputeGeneStats *Compute gene- and cell-level statistics for scDRS*

Description

Computes the gene-level statistics used to match control genes (mean, variance, technical variance and mean-variance bins) and the cell-level mean and variance. This is the R port of scdrs.pp.compute_stats(). Most users call scdrsPreprocess(), which runs this function.

Usage

```
scdrsComputeGeneStats(
  sce,
  assay = "logcounts",
  implicit = FALSE,
  cellWeight = NULL,
  nMeanBin = 20L,
  nVarBin = 20L,
  nChunk = 20L
)
```

Arguments

sce	A SingleCellExperiment::SingleCellExperiment with log-normalized expression (genes in rows, cells in columns).
assay	Name of the assay to use.
implicit	Logical. If TRUE, the statistics are computed for the covariate-corrected data X + COV_BETA %*% t(COV_MAT) + COV_GENE_MEAN without building it (implicit covariate correction). The correction is read from reducedDim(sce, "scdrs_cov"), rowData(sce)$scdrs_cov_beta and rowData(sce)$scdrs_cov_gene_mean, as stored by scdrsPreprocess(). If FALSE, the statistics are computed for the assay as it is.
cellWeight	NULL or a numeric vector with one weight per cell, used for the gene-level statistics.
nMeanBin, nVarBin	Number of mean-expression bins and of variance bins within each mean bin. If either is larger than floor(sqrt(nrow(sce) / 10)), both are set to that value.
nChunk	Number of chunks used in implicit mode (controls memory use only; the results do not depend on it).

Details

Variances use the population formula (divisor n). Technical variance follows the Seurat v3 recipe: a LOESS fit (span 0.3, degree 2) of log10(ct_var) on log10(ct_mean), and var_tech = var * ct_var_tech / ct_var. Bins are quantile bins as pandas.qcut(), labelled from 0.

Value

A list of two S4Vectors::DataFrames:

gene One row per gene: mean, var (log scale), var_tech (technical variance, log scale), ct_mean, ct_var, ct_var_tech (the same on the expm1() scale) and mean_var (mean-variance bin, e.g. "3.12").

cell One row per cell: mean and var (log scale, unweighted).

References

Zhang MJ, Hou K, et al. (2022). Polygenic enrichment distinguishes disease associations of individual cells in single-cell RNA-seq data. *Nature Genetics* 54, 1572-1580.

See Also

scdrsPreprocess()

Examples

```
sce <- scdrsLoadToyData()
stats <- scdrsComputeGeneStats(sce)
head(stats$gene)
head(stats$cell)
```

`scdrsComputeScore` *Compute scDRS scores for several gene sets*

Description

Preprocesses the data if needed (`scdrsPreprocess()`) and scores every gene set with at least `minGenes` genes in the data (`scdrsScoreCells()`). This is the R port of the `scdrs compute-score` command, with the same defaults.

Usage

```
scdrsComputeScore(
  sce,
  geneSets,
  assay = "logcounts",
  cov = NULL,
  adjProp = NULL,
  ctrlMatchKey = "mean_var",
  weightOpt = "vs",
  nCtrl = 1000L,
  returnCtrlRawScore = FALSE,
  returnCtrlNormScore = TRUE,
  minGenes = 10L
)
```

Arguments

`sce`	A SingleCellExperiment::SingleCellExperiment with log-normalized data (see `scdrsNormalize()` for raw counts).
`geneSets`	A named list of gene sets as returned by `scdrsReadGeneSets()` (named numeric vectors of weights, or character vectors), or the path of a `.gs` file with genes named as in `sce`.
`assay`	Assay to use.
`cov, adjProp`	Covariates and cell-group column for `scdrsPreprocess()`. The data is preprocessed when these are given or when `sce` has not been preprocessed yet; otherwise the existing preprocessing is used.
`ctrlMatchKey, weightOpt, nCtrl`	See `scdrsScoreCells()`.
`returnCtrlRawScore, returnCtrlNormScore`	Return the control scores? The normalized ones are returned by default, because the downstream analyses need them.
`minGenes`	Gene sets with fewer genes in `sce` are skipped.

Value

A named S4Vectors::SimpleList of score tables (see `scdrsScoreCells()`), one per scored gene set.

See Also

scdrsWriteScores(), scdrsPerformDownstream().

Examples

```
sce <- scdrsLoadToyData()
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_human.gs",
    package = "RscDRS"), srcSpecies = "human", dstSpecies = "mouse")
set.seed(1)
res <- scdrsComputeScore(sce, gs, cov = "covariate", nCtrl = 100)
names(res)
head(res[[1]][, 1:6])
```

---

scdrsGetHomologs *Mouse-human gene homologs*

Description

Returns the mouse-human gene symbol map shipped with scDRS (16,466 pairs, `system.file("extdata", "mouse_human_homologs.txt", package = "RscDRS")`). This is the R port of `scdrs.util.load_homolog_mapping()`.

Usage

```
scdrsGetHomologs(src, dst)
```

Arguments

src, dst	Source and destination species: "mouse" or "human" (also "mmusculus", "Mus_musculus", "hsapiens", "Homo_sapiens", ...). They must differ.

Value

A named character vector: names are `src` gene symbols, values the `dst` homologs.

Examples

```
map <- scdrsGetHomologs("human", "mouse")
map[c("ACADM", "APOE")]
```

scdrsLoadToyData *Load the scDRS toy data*

Description

Loads the small example data set shipped with the original scDRS (a subset of Tabula Muris Senis: 2,500 mouse genes and 30 cells, 10 "causal" and 20 "non-causal") as a `SingleCellExperiment`.

Usage

```
scdrsLoadToyData()
```

Value

A SingleCellExperiment::SingleCellExperiment with a sparse `logcounts` assay (size-factor normalized, `log1p`), cell metadata in `colData` (`label`, `cell_type`, `causal_variable`, `non_causal_variable`, `covariate`) and gene metadata in `rowData` (`label`). The files are in `system.file("extdata", package = "RscDRS")`, including the disease gene sets `toydata_mouse.gs` and `toydata_human.gs`; see `inst/scripts/python/README.md` for their origin.

Examples

```
sce <- scdrsLoadToyData()
sce
table(sce$cell_type)
```

scdrsMetaAnalyze *Meta-analysis of effect sizes*

Description

Fixed-effect or DerSimonian-Laird random-effects meta-analysis, e.g. to combine scDRS results across data sets. This is the R port of `scdrs.util.meta_analysis()`.

Usage

```
scdrsMetaAnalyze(effects, se, method = c("random", "fixed"), weights = NULL)
```

Arguments

`effects`	Numeric vector of effect sizes.
`se`	Numeric vector of their standard errors.
`method`	`"random"` (default) or `"fixed"`.
`weights`	Optional weights to use instead of the inverse variances.

Details

With $v_i = 1/\mathrm{se}_i^2$, the between-study variance is $\tau^2 = \max(0, (Q-(m-1))/(\sum v - \sum v^2/\sum v))$, where Q is Cochran's Q. Random-effects weights are $1/(\mathrm{se}^2 + \tau^2)$.

Value

A named numeric vector: `summary` (the combined effect) and `se` (its standard error).

Examples

```
scdrsMetaAnalyze(c(0.2, 0.5, -0.1), c(0.1, 0.2, 0.15))
scdrsMetaAnalyze(c(0.2, 0.5, -0.1), c(0.1, 0.2, 0.15), method = "fixed")
```

---

scdrsMungeGeneSets *Make scDRS gene sets from gene-level GWAS statistics*

Description

For each trait, selects the disease genes with the smallest gene-level p-values (e.g. from MAGMA) and weights them by their z-scores. This is the R port of the `scdrs munge-gs` command.

Usage

```
scdrsMungeGeneSets(
  pvals = NULL,
  zscores = NULL,
  weight = c("zscore", "uniform"),
  fdr = NULL,
  fwer = NULL,
  nMin = 100L,
  nMax = 1000L,
  file = NULL
)
```

Arguments

`pvals`, `zscores`	Exactly one of: gene-level p-values or one-sided z-scores, as a `data.frame` (genes as row names, one column per trait) or the path of a tab-separated file with the genes in the first column (header `GENE`). Z-scores are converted to p-values with `pnorm(-z)`. Missing values are dropped per trait.
`weight`	`"zscore"` (default): weight $\min(-\Phi^{-1}(\max(p, 10^{-100})), 10)$; or `"uniform"`: weight 1.
`fdr`, `fwer`	At most one of them: the FDR or FWER level used to choose the number of genes.
`nMin`, `nMax`	Minimum and maximum number of genes per trait.
`file`	Optional path: also write the gene sets to this `.gs` file, in the format of `scdrs munge-gs` (weights with 5 significant digits).

Details

The number of genes per trait is the number of discoveries at `fdr` (Benjamini-Hochberg) or `fwer` (Bonferroni), or `nMax` if neither is given, and is then restricted to `[nMin, nMax]`.

Value

A named list (traits in sorted order) of named numeric vectors (names = genes, ordered by p-value; values = weights rounded to 5 significant digits, as in the file), like `scdrsReadGeneSets()`.

Examples

```
set.seed(1)
p <- data.frame(BMI = runif(500)^4, HEIGHT = runif(500)^2,
    row.names = paste0("GENE", seq_len(500)))
gs <- scdrsMungeGeneSets(pvals = p, fdr = 0.05, nMin = 10, nMax = 100)
lengths(gs)
head(gs$BMI)
```

---

scdrsNormalize *Filter and normalize raw counts as scDRS does*

Description

Optionally removes cells with fewer than `minGenes` expressed genes and then genes expressed in fewer than `minCells` cells, and normalizes each cell to 10,000 total counts followed by a natural-log `log1p`. This copies what `scdrs compute-score` does with `--flag-filter-data` and `--flag-raw-count` (scanpy's `filter_cells`, `filter_genes`, `normalize_per_cell(counts_per_cell_after = 1e4)` and `log1p`; R port of `scdrs.util.load_h5ad()`).

Usage

```
scdrsNormalize(
  sce,
  assay = "counts",
  filter = TRUE,
  minGenes = 250L,
  minCells = 50L,
  normalize = TRUE,
  name = "logcounts"
)
```

Arguments

`sce`	A SingleCellExperiment::SingleCellExperiment with raw counts.
`assay`	Name of the counts assay.
`filter`	Filter cells and genes first?

`minGenes`, `minCells`
: Filtering thresholds.

`normalize`
: Normalize and log-transform? If `FALSE`, only the filtering is done.

`name`
: Name of the new assay.

Details

Note that `scuttle::logNormCounts()` gives different values (log2 and centred size factors); scDRS expects natural-log data.

Value

`sce`, possibly with fewer cells and genes, with the normalized data in assay `name` (sparse if the counts are sparse). Cells without any counts are removed when normalizing, as in scanpy.

Examples

```
set.seed(1)
counts <- matrix(rpois(2000, 2), 50, 40,
    dimnames = list(paste0("g", 1:50), paste0("c", 1:40)))
sce <- SingleCellExperiment::SingleCellExperiment(list(counts = counts))
sce <- scdrsNormalize(sce, minGenes = 10, minCells = 5)
sce
```

---

scdrsPerformDownstream

Run the scDRS downstream analyses

Description

Runs the group-level analysis (`scdrsTestGroups()`), the correlation analysis (`scdrsTestCorrelation()`) and the gene analysis (`scdrsRankGenes()`) for one or several traits. This is the R port of the `scdrs perform-downstream` command.

Usage

```
scdrsPerformDownstream(
  sce,
  scores,
  groupCols = NULL,
  varCols = NULL,
  geneAnalysis = FALSE,
  graph = "connectivities",
  assay = "logcounts"
)
```

Arguments

sce	A SingleCellExperiment::SingleCellExperiment with the annotations in colData and a cell-cell graph (see graph).
scores	One score table, or a named list of them (e.g. from scdrsComputeScore() or scdrsReadScores()), with normalized control scores.
groupCols, varCols	NULL, or colData(sce) columns for the group analysis and the correlation analysis.
geneAnalysis	Run the gene analysis?
graph	Name of a colPair(sce) graph (default "connectivities", made by scdrsBuildGraph()), or a cells x cells matrix.
assay	Assay used by the gene analysis.

Value

For one score table, a list with elements group, corr and gene (NULL for analyses not run); for a list of score tables, a named list of such lists.

Examples

```
sce <- scdrsBuildGraph(scdrsLoadToyData())
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_mouse.gs",
    package = "RscDRS"))
set.seed(1)
res <- scdrsComputeScore(sce, gs, cov = "covariate", nCtrl = 100)
out <- scdrsPerformDownstream(sce, res, groupCols = "cell_type",
    varCols = "causal_variable")
out$toydata_gs_mouse$group$cell_type
```

---

scdrsPreprocess *Preprocess single-cell data for scDRS*

Description

Regresses covariates out of the expression data (adding back each gene's mean), then computes the gene- and cell-level statistics that scDRS uses to choose control genes. This is the R port of scdrs.preprocess().

Usage

```
scdrsPreprocess(
  sce,
  assay = "logcounts",
  cov = NULL,
  adjProp = NULL,
```

```
  nMeanBin = 20L,
  nVarBin = 20L,
  nChunk = NULL,
  implicit = NULL
)
```

Arguments

sce	A SingleCellExperiment::SingleCellExperiment. The assay must be size-factor normalized and log1p-transformed (natural log), with genes in rows and cells in columns. Gene and cell names must be unique.
assay	Name of the assay to use. It is never modified.
cov	Covariates to regress out, or NULL for none. Either a character vector of colData(sce) column names, or a data.frame / DataFrame with cell names as row names (like the scDRS .cov file; at least 75% of the cells must be found). Non-numeric columns are turned into dummy variables, missing values are replaced by the column mean, and a constant term SCDRS_CONST is added unless the covariates already contain one.
adjProp	NULL, or the name of a colData(sce) column with cell groups (e.g. cell types). Cells are then weighted inversely to the size of their group when computing gene statistics (groups smaller than 1% of the largest one count as that size).
nMeanBin, nVarBin	Number of mean-expression bins and of variance bins within each mean bin, used to match control genes.
nChunk	Number of chunks for implicit covariate correction (memory use only). NULL uses 5 * nGene * nCell / nnz + 1 in implicit mode and 20 otherwise, as in Python.
implicit	Use implicit covariate correction? NULL (default) follows scDRS: TRUE when the assay is sparse and cov is given. In implicit mode the corrected data is never stored, which keeps the memory use low; otherwise it is stored as a new dense assay "scdrs_corrected".

Details

The corrected data is

$$\tilde{X} = X - C\beta + \mu,$$

where C is the covariate matrix, β the least-squares coefficients and μ the gene means. Python's COV_BETA is $-\beta^\top$, so X~ = X + scdrs_cov_beta %*% t(scdrs_cov) + scdrs_cov_gene_mean.

Value

sce, with:

rowData(sce) scdrs_mean, scdrs_var, scdrs_var_tech, scdrs_ct_mean, scdrs_ct_var, scdrs_ct_var_tech, scdrs_mean_var (see scdrsComputeGeneStats()); with covariates also scdrs_cov_beta (a genes x covariates matrix, the negated regression coefficients, Python's COV_BETA) and scdrs_cov_gene_mean.

colData(sce) scdrs_mean, scdrs_var.

`reducedDim(sce, "scdrs_cov")` The covariate matrix (cells x covariates), with covariates only.

`assay(sce, "scdrs_corrected")` The covariate-corrected data, with covariates and `implicit = FALSE` only.

`metadata(sce)$scdrs` A list of flags and settings (`assay` is the assay that later steps should score).

Results of an earlier run of `scdrsPreprocess()` are replaced.

References

Zhang MJ, Hou K, et al. (2022). Polygenic enrichment distinguishes disease associations of individual cells in single-cell RNA-seq data. *Nature Genetics* 54, 1572-1580.

See Also

`scdrsComputeGeneStats()`

Examples

```
sce <- scdrsLoadToyData()
sce <- scdrsPreprocess(sce, cov = "covariate")
rowData(sce)[1:3, c("scdrs_mean", "scdrs_var", "scdrs_mean_var")]
head(reducedDim(sce, "scdrs_cov"))
str(metadata(sce)$scdrs)
```

---

scdrsRankGenes *Rank genes by their correlation with the disease score*

Description

Computes the Pearson correlation across cells between each gene's expression and the normalized disease score, to prioritize genes. This is the R port of `scdrs.method.downstream_gene_analysis()`.

Usage

```
scdrsRankGenes(sce, scores, assay = "logcounts")
```

Arguments

`sce`	A SingleCellExperiment::SingleCellExperiment with the annotations in `colData` and a cell-cell graph (see `graph`).
`scores`	A score table with normalized control scores: the output of `scdrsScoreCells()` with `returnCtrlNormScore = TRUE`, one element of `scdrsComputeScore()`, or a table read with `scdrsReadScores()`.
`assay`	Expression assay to use. As in Python, this is the log-normalized data, not the covariate-corrected data.

Value

A S4Vectors::DataFrame with one row per gene (row names), sorted by decreasing correlation, with columns `CORR` and `RANK` (1 = highest correlation; Python's `RANK` starts at 0).

Examples

```
sce <- scdrsPreprocess(scdrsLoadToyData(), cov = "covariate")
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_mouse.gs",
    package = "RscDRS"))
set.seed(1)
res <- scdrsScoreCells(sce, gs[[1]], nCtrl = 100)
head(scdrsRankGenes(sce, res))
```

---

scdrsReadGeneSets *Read and write scDRS gene set files*

Description

`scdrsReadGeneSets()` reads a `.gs` file: a tab-separated file with columns `TRAIT` and `GENESET`, where `GENESET` is a comma-separated list of genes (`"g1,g2,g3"`) or of genes and weights (`"g1:w1,g2:w2"`). Genes can be mapped between mouse and human and intersected with the genes of a data set. `scdrsWriteGeneSets()` writes such a file. These are the R ports of `scdrs.util.load_gs()` and `scdrs.util.save_gs()`.

Usage

```
scdrsReadGeneSets(file, srcSpecies = NULL, dstSpecies = NULL, intersect = NULL)

scdrsWriteGeneSets(geneSets, file)
```

Arguments

`file`	Path of the `.gs` file.
`srcSpecies, dstSpecies`	`NULL` (default, no mapping), or the species of the gene set file and of the data (`"mouse"` or `"human"`). Genes without a homolog are dropped.
`intersect`	`NULL`, or a character vector of genes (e.g. `rownames(sce)`) to keep.
`geneSets`	A named list of gene sets: named numeric vectors (genes and weights, written as `"g:w"`) or character vectors (genes only).

Value

`scdrsReadGeneSets()`: a named list (one element per trait) of named numeric vectors (names = genes, values = weights; 1 for an unweighted gene set). A gene listed twice keeps its first position and its last weight, as in Python.

`scdrsWriteGeneSets()`: `file`, invisibly.

See Also

scdrsMungeGeneSets() to make gene sets from GWAS gene statistics, scdrsComputeScore().

Examples

```
f <- system.file("extdata", "toydata_human.gs", package = "RscDRS")
sce <- scdrsLoadToyData()
gs <- scdrsReadGeneSets(f, srcSpecies = "human", dstSpecies = "mouse",
    intersect = rownames(sce))
lengths(gs)
out <- tempfile(fileext = ".gs")
scdrsWriteGeneSets(gs, out)
identical(scdrsReadGeneSets(out), gs)
```

---

scdrsScoreCells *Compute scDRS disease scores for each cell*

Description

For one disease gene set, computes each cell's raw disease score (a weighted mean of the covariate-corrected expression of the disease genes), the same score for `nCtrl` matched control gene sets, the normalized score, and a cell-level p-value from the empirical distribution of the control scores pooled over all cells. This is the R port of `scdrs.score_cell()`.

Usage

```
scdrsScoreCells(
  sce,
  genes,
  weights = NULL,
  ctrlMatchKey = "mean_var",
  nCtrl = 1000L,
  nGeneBin = 200L,
  weightOpt = c("vs", "uniform", "inv_std", "od"),
  returnCtrlRawScore = FALSE,
  returnCtrlNormScore = FALSE,
  controlSets = NULL
)
```

Arguments

`sce`	A SingleCellExperiment::SingleCellExperiment processed by scdrsPreprocess().
`genes`	Disease genes: a character vector, or a named numeric vector of weights (names = genes), e.g. one element of scdrsReadGeneSets(). Genes not in `sce` are dropped.
`weights`	Optional gene weights when `genes` is a character vector (default 1).

`ctrlMatchKey` Gene statistic used to match control genes (`"mean_var"`, the default, or e.g. `"mean"`); see `scdrsSelectControlGenes()`.

`nCtrl` Number of control gene sets.

`nGeneBin` Number of quantile bins for a continuous `ctrlMatchKey`.

`weightOpt` How the raw score weights genes: `"vs"` (default, $1/\sqrt{\text{var_tech} + 0.01}$), `"uniform"`, `"inv_std"` ($1/\sqrt{\text{var} + 0.01}$), or `"od"` (overdispersion score, a benchmarking option); always multiplied by the gene weights.

`returnCtrlRawScore`, `returnCtrlNormScore` Also return the raw / normalized control scores? The normalized control scores are needed by the downstream analyses (`scdrsTestGroups()` etc.).

`controlSets` Optional precomputed control gene sets, either a list of named numeric vectors as returned by `scdrsSelectControlGenes()`, or a `data.frame` with columns `ctrl_id`, `gene` and `weight`. If given, `nCtrl`, `ctrlMatchKey` and `nGeneBin` only matter for the variance ratio (computed when `ctrlMatchKey = "mean_var"`), and no random numbers are used.

Details

Run `scdrsPreprocess()` first. Control gene sets are drawn at random (see `scdrsSelectControlGenes()`); call `set.seed()` first for reproducible results.

Let s_c be the raw disease score of cell c and S_{ck} that of control set k. The scores are centered per gene set, the control scores are scaled to the same variance as the disease score (assuming independent genes), each cell is standardized by the mean and SD of its control scores, and the scores are centered again. Cells with a raw score of exactly 0 get the smallest score. The p-value is $(1 + \#\{(c', k) : Z_{c'k} \geq z_c\})/(1 + nK)$.

Value

A S4Vectors::DataFrame with one row per cell (row names = cell names) and columns

`raw_score` Raw disease score.

`norm_score` Normalized disease score.

`mc_pval` Monte Carlo p-value from the cell's own control scores.

`pval` scDRS cell-level p-value (control scores pooled over all cells).

`nlog10_pval` `-log10(pval)`.

`zscore` One-sided z-score of `pval`, clipped to [-10, 10].

`ctrl_raw_score`, `ctrl_norm_score` Only if requested: cells x `nCtrl` matrices of control scores (matrix columns, columns named `0`, `1`, ... as in Python's `ctrl_norm_score_0`, ...).

`metadata()` of the result holds the disease genes and weights used (`genes`), the control sets (`controlSets`) and the settings.

References

Zhang MJ, Hou K, et al. (2022). Polygenic enrichment distinguishes disease associations of individual cells in single-cell RNA-seq data. *Nature Genetics* 54, 1572-1580.

See Also

scdrsPreprocess(), scdrsComputeScore() to score several gene sets, scdrsTestGroups().

Examples

```
sce <- scdrsPreprocess(scdrsLoadToyData(), cov = "covariate")
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_mouse.gs",
    package = "RscDRS"))
set.seed(1)
res <- scdrsScoreCells(sce, gs[["toydata_gs_mouse"]], nCtrl = 100)
head(res)
tapply(res$norm_score, sce$cell_type, mean)
```

---

scdrsSelectControlGenes

Select matched control gene sets

Description

Draws nCtrl control gene sets that match the disease gene set gene by gene: within each bin of ctrlMatchKey (by default the mean-variance bins made by scdrsPreprocess()) that contains d disease genes, each control set gets d genes drawn at random, without replacement, from all genes of that bin (disease genes included). A control gene takes the weight of a disease gene of the same bin. This is the R port of scdrs.method._select_ctrl_geneset().

Usage

```
scdrsSelectControlGenes(
  geneStats,
  genes,
  weights = NULL,
  ctrlMatchKey = "mean_var",
  nCtrl = 1000L,
  nGeneBin = 200L
)
```

Arguments

geneStats	A SingleCellExperiment::SingleCellExperiment processed by scdrsPreprocess() (the statistics are taken from rowData, without the scdrs_ prefix), or a data.frame / DataFrame of gene statistics with gene names as row names.
genes	Disease genes: a character vector, or a named numeric vector of weights (names = genes), as returned by scdrsReadGeneSets(). Genes not in geneStats are dropped; a gene given twice keeps its last weight.
weights	Optional gene weights when genes is a character vector (default 1).

ctrlMatchKey	Gene statistic used for matching, e.g. "mean_var" (default) or "mean". A statistic with fewer distinct values than 10% of the genes is used as categories; otherwise genes are split into nGeneBin quantile bins.
nCtrl	Number of control gene sets.
nGeneBin	Number of quantile bins for a continuous ctrlMatchKey.

Details

Random numbers come from R's generator, so call set.seed() first for reproducible results. The control sets are not the same as the ones Python draws with the same seed (see the vignette).

Value

A list of nCtrl named numeric vectors (names = control genes, values = weights).

References

Zhang MJ, Hou K, et al. (2022). Polygenic enrichment distinguishes disease associations of individual cells in single-cell RNA-seq data. *Nature Genetics* 54, 1572-1580.

See Also

scdrsScoreCells()

Examples

```
sce <- scdrsPreprocess(scdrsLoadToyData())
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_mouse.gs",
    package = "RscDRS"))
set.seed(1)
ctrl <- scdrsSelectControlGenes(sce, gs[[1]], nCtrl = 5)
lengths(ctrl)
head(ctrl[[1]])
```

---

scdrsTestCorrelation *Correlation of disease scores with cell-level variables*

Description

For each continuous cell-level variable (e.g. a T cell effector gradient), tests the Pearson correlation between the variable and the normalized disease score against the correlations with the control scores. This is the R port of scdrs.method.downstream_corr_analysis().

Usage

```
scdrsTestCorrelation(sce, scores, varCols)
```

Arguments

sce	A SingleCellExperiment::SingleCellExperiment with the annotations in colData and a cell-cell graph (see graph).
scores	A score table with normalized control scores: the output of scdrsScoreCells() with returnCtrlNormScore = TRUE, one element of scdrsComputeScore(), or a table read with scdrsReadScores().
varCols	Names of numeric colData(sce) columns.

Value

A S4Vectors::DataFrame with one row per variable and columns n_ctrl, corr_mcp (Monte Carlo p-value) and corr_mcz (z-score against the control correlations).

Examples

```
sce <- scdrsPreprocess(scdrsLoadToyData(), cov = "covariate")
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_mouse.gs",
    package = "RscDRS"))
set.seed(1)
res <- scdrsScoreCells(sce, gs[[1]], nCtrl = 100,
    returnCtrlNormScore = TRUE)
scdrsTestCorrelation(sce, res, c("causal_variable",
    "non_causal_variable"))
```

---

scdrsTestGearysC *Test the heterogeneity of disease scores within cell groups*

Description

For each group of cells, computes Geary's C of the normalized disease score on the group's part of the cell-cell graph and compares it with null statistics, to test whether disease scores are more similar among neighbouring cells than expected (heterogeneity within the group). This is the R port of scdrs.method.test_gearysc().

Usage

```
scdrsTestGearysC(
  graph,
  scores,
  groups,
  method = c("control_distribution_match", "permutation", "control")
)
```

Arguments

graph	A cells x cells matrix of edge weights, with the cells in the order of the rows of scores (or with cell names as dimnames).
scores	A score table with normalized control scores, from scdrsScoreCells() with returnCtrlNormScore = TRUE (or read from a Python .full_score.gz file with scdrsReadScores()).
groups	A vector with the group of each cell (rows of scores).
method	How the null statistics are made: "control_distribution_match" (default: the disease scores of the group, rearranged in the order of each control score), "permutation" (random permutations of the disease scores; uses random numbers) or "control" (the control scores themselves).

Value

A S4Vectors::DataFrame with one row per group (sorted): pval ($(1 + \#\{k : C > C_k\})/(1 + K)$; small values mean heterogeneity), trait (Geary's C of the disease score), ctrl_mean, ctrl_std (mean and sample SD of the null statistics) and zsc $(-(C - \text{ctrl_mean})/\text{ctrl_std})$.

See Also

scdrsTestGroups(), which calls this function.

Examples

```
sce <- scdrsPreprocess(scdrsBuildGraph(scdrsLoadToyData()))
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_mouse.gs",
    package = "RscDRS"))
set.seed(1)
res <- scdrsScoreCells(sce, gs[[1]], nCtrl = 100,
    returnCtrlNormScore = TRUE)
g <- SingleCellExperiment::colPair(sce, "connectivities", asSparse = TRUE)
scdrsTestGearysC(g, res, sce$cell_type)
```

---

scdrsTestGroups *Group-level scDRS analysis*

Description

For each annotation in groupCols (e.g. cell types) and each group of cells, counts the significant cells and tests whether the group as a whole is associated with the disease (association) and whether disease scores vary among the group's cells beyond chance (heterogeneity). This is the R port of scdrs.method.downstream_group_analysis().

Usage

```
scdrsTestGroups(
  sce,
  scores,
  groupCols,
  fdrThresholds = c(0.05, 0.1, 0.2),
  graph = "connectivities"
)
```

Arguments

sce	A SingleCellExperiment::SingleCellExperiment with the annotations in colData and a cell-cell graph (see graph).
scores	A score table with normalized control scores: the output of scdrsScoreCells() with returnCtrlNormScore = TRUE, one element of scdrsComputeScore(), or a table read with scdrsReadScores().
groupCols	Names of colData(sce) columns defining the groups.
fdrThresholds	FDR levels at which significant cells are counted.
graph	Name of a colPair(sce) graph (default "connectivities", made by scdrsBuildGraph()), or a cells x cells matrix.

Value

A named S4Vectors::SimpleList with one S4Vectors::DataFrame per element of groupCols, with one row per group (sorted) and columns:

n_cell, n_ctrl Numbers of cells and of control sets.

assoc_mcp, assoc_mcz Association: Monte Carlo p-value and z-score of the 95% quantile of the group's disease scores against the same quantile of each control score.

hetero_mcp, hetero_mcz Heterogeneity: p-value and z-score of Geary's C (see scdrsTestGearysC()).

n_fdr_<t> Number of cells with FDR (Benjamini-Hochberg, over all cells) below t, for each of fdrThresholds.

References

Zhang MJ, Hou K, et al. (2022). Polygenic enrichment distinguishes disease associations of individual cells in single-cell RNA-seq data. *Nature Genetics* 54, 1572-1580.

See Also

scdrsTestCorrelation(), scdrsRankGenes(), scdrsPerformDownstream().

Examples

```
sce <- scdrsBuildGraph(scdrsLoadToyData())
sce <- scdrsPreprocess(sce, cov = "covariate")
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_mouse.gs",
    package = "RscDRS"))
```

```
set.seed(1)
res <- scdrsScoreCells(sce, gs[[1]], nCtrl = 100,
    returnCtrlNormScore = TRUE)
scdrsTestGroups(sce, res, "cell_type")[["cell_type"]]
```

`scdrsTestOverlap` *Test the overlap of two gene sets*

Description

Fisher's exact test for the overlap of two gene sets within a background, with the sample odds ratio and its 95% confidence interval. This is the R port of `scdrs.util.test_overlap()`.

Usage

```
scdrsTestOverlap(set1, set2, background)
```

Arguments

`set1`, `set2`	Character vectors of genes.
`background`	Character vector of all genes considered.

Details

The table counts background genes in neither set, in only one of them, and in both. Unlike `stats::fisher.test()`, the odds ratio is the sample odds ratio (as `scipy.stats.fisher_exact()`), not the conditional maximum likelihood estimate.

Value

A named numeric vector: `pval` (two-sided Fisher's exact test), `odds_ratio` (sample odds ratio), `or_ub` and `or_lb` (upper and lower 95% Wald limits on the log scale; both 0 if a cell of the 2 x 2 table is 0, as in Python).

Examples

```
bg <- paste0("g", seq_len(100))
scdrsTestOverlap(bg[1:30], bg[21:45], bg)
```

scdrsWriteScores *Read and write scDRS score files*

Description

scdrsWriteScores() writes score tables in the format of scdrs compute-score: <trait>.score.gz (the six main columns) and, if the tables contain control scores, <trait>.full_score.gz (all columns, with ctrl_raw_score_0, ..., ctrl_norm_score_0, ...). scdrsReadScores() reads such files, written by R or by Python, back into score tables that the downstream functions accept. This is the R port of scdrs.util.load_scdrs_score().

Usage

```
scdrsWriteScores(scores, dir)

scdrsReadScores(file, cells = NULL)
```

Arguments

scores	A named list of score tables (e.g. from scdrsComputeScore()).
dir	Output directory (created if needed).
file	Path of a .full_score.gz or .score.gz file. Use @ in the file name to read several files at once, e.g. "out/@.full_score.gz" (the directory itself must not contain @).
cells	Optional cell names: files containing fewer than 10% of them are skipped, with a message.

Value

scdrsWriteScores(): the paths of the files written, invisibly.

scdrsReadScores(): a named list (traits from the file names) of S4Vectors::DataFrames with cell names as row names; control scores are returned as the matrix columns ctrl_raw_score and ctrl_norm_score.

Examples

```
sce <- scdrsLoadToyData()
gs <- scdrsReadGeneSets(system.file("extdata", "toydata_mouse.gs",
    package = "RscDRS"))
set.seed(1)
res <- scdrsComputeScore(sce, gs, nCtrl = 20)
dir <- file.path(tempdir(), "scdrs_out")
scdrsWriteScores(res, dir)
list.files(dir)
back <- scdrsReadScores(file.path(dir, "@.full_score.gz"))
all.equal(back$toydata_gs_mouse$norm_score, res[[1]]$norm_score)
```

scdrsZToP *Convert between one-sided p-values and z-scores*

Description

`scdrsZToP()` returns $p = \Phi(-z)$ and `scdrsPToZ()` returns $z = -\Phi^{-1}(p)$, accurate far in the tails. These are the R ports of `scdrs.util.zsc2pval()` and `scdrs.util.pval2zsc()`.

Usage

```
scdrsZToP(z)

scdrsPToZ(p)
```

Arguments

z	Numeric vector of z-scores.
p	Numeric vector of p-values.

Value

A numeric vector of the same length.

Examples

```
scdrsZToP(c(0, 1.96, 10))
scdrsPToZ(c(0.5, 0.025, 1e-30))
```

Index

S4Vectors::DataFrame, *5*, *15*, *17*, *20–22*, *24*
S4Vectors::SimpleList, *6*, *22*
scdrsBuildGraph, 2
scdrsBuildGraph(), *4*, *12*, *22*
scdrsComputeGearysC, 3
scdrsComputeGeneStats, 4
scdrsComputeGeneStats(), *13, 14*
scdrsComputeScore, 6
scdrsComputeScore(), *12*, *14*, *16*, *18*, *20*, *22*, *24*
scdrsGetHomologs, 7
scdrsLoadToyData, 8
scdrsMetaAnalyze, 8
scdrsMungeGeneSets, 9
scdrsMungeGeneSets(), *16*
scdrsNormalize, 10
scdrsNormalize(), *6*
scdrsPerformDownstream, 11
scdrsPerformDownstream(), *7*, *22*
scdrsPreprocess, 12
scdrsPreprocess(), *4–6*, *16–18*
scdrsPToZ *(*scdrsZToP*)*, 25
scdrsRankGenes, 14
scdrsRankGenes(), *11*, *22*
scdrsReadGeneSets, 15
scdrsReadGeneSets(), *6*, *10*, *16*, *18*
scdrsReadScores *(*scdrsWriteScores*)*, 24
scdrsReadScores(), *12*, *14*, *20–22*
scdrsScoreCells, 16
scdrsScoreCells(), *6*, *14*, *19–22*
scdrsSelectControlGenes, 18
scdrsSelectControlGenes(), *17*
scdrsTestCorrelation, 19
scdrsTestCorrelation(), *11*, *22*
scdrsTestGearysC, 20
scdrsTestGearysC(), *22*
scdrsTestGroups, 21
scdrsTestGroups(), *2, 3*, *11*, *17, 18*, *21*
scdrsTestOverlap, 23
scdrsWriteGeneSets *(*scdrsReadGeneSets*)*, 15
scdrsWriteScores, 24
scdrsWriteScores(), *7*
scdrsZToP, 25
set.seed(), *17*, *19*
SingleCellExperiment::SingleCellExperiment, *3*, *5, 6*, *8*, *10*, *12–14*, *16*, *18*, *20*, *22*
stats::fisher.test(), *23*