# Package: SPAROscore (via r-universe)

July 22, 2026

**Type** Package

**Title** A package to compute gene signature scores from transcriptomics data

**Version** 0.99.3

**Description** SPAROscore is a gene signature scoring method designed to be robust across diverse gene expression datasets. SPAROscore adapts to varying levels of sparsity, allowing signature scores to be efficiently computed across bulk, single-cell, and spatial transcriptomic datasets. The resulting scores quantify the relative expression of a gene signature compared with the background expression of each sample/cell/domain, making the scores straightforward to interpret biologically. Internally, SPAROscore ranks genes within each column and computes the Spearman footrule distance between the observed ranks of the signature genes and a background gene expression rank estimated from the geometric mean expression of the sample, cell, or spatial domain.

**License** GPL-3

**URL** https://github.com/MangiolaLaboratory/SPAROscore

**BugReports** https://github.com/MangiolaLaboratory/SPAROscore/issues

**Imports** DelayedArray, DelayedMatrixStats, Matrix, MatrixGenerics, matrixStats, methods, sparseMatrixStats, stats

**Suggests** airway, dplyr, edgeR, ggplot2, ggspavis, GSEABase, knitr, msigdbr, muscData, rmarkdown, scater, Seurat, SingleCellExperiment, SpatialExperiment, spatialLIBD, SummarizedExperiment, testthat (>= 3.0.0), tidyr

**VignetteBuilder** knitr

**biocViews** GeneExpression, SingleCell, Spatial, Transcriptomics, GeneSetEnrichment, Pathways, Software

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** false

**RoxygenNote** 7.3.3

**Config/pak/sysreqs** zlib1g-dev

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

**Date/Publication** 2026-07-22 06:33:11 UTC

**RemoteUrl** https://github.com/BiocStaging/SPAROscore

**RemoteRef** HEAD

**RemoteSha** bb23f80ef8e9fca0dd1d3e8362a266d8dd2a0af6

# Contents



---

`get_ranks` *Compute gene ranks and rank caps for SPAROscore*

---

## Description

Generates gene rank matrices and rank caps required for SPAROscore calculations.

## Usage

```
get_ranks(
  counts,
  count_caps = compute_geometric_average(counts),
  handle_ties = "min"
)

## S4 method for signature 'matrix'
get_ranks(
  counts,
  count_caps = compute_geometric_average(counts),
  handle_ties = "min"
)

## S4 method for signature 'sparseMatrix'
get_ranks(
  counts,
  count_caps = compute_geometric_average(counts),
  handle_ties = "min"
)
```

```
## S4 method for signature 'DelayedMatrix'
get_ranks(
  counts,
  count_caps = compute_geometric_average(counts),
  handle_ties = "min"
)

## S4 method for signature 'data.frame'
get_ranks(
  counts,
  count_caps = compute_geometric_average(counts),
  handle_ties = "min"
)
```

## Arguments

| | |
|---|---|
| counts | A matrix-like object containing gene expression counts, with genes in rows and samples, cells, or spatial locations in columns.<br>Supported input classes include:<br>• matrix<br>• Matrix::sparseMatrix<br>• DelayedArray::DelayedMatrix<br>• data.frame |
| count_caps | Optional numeric vector containing count cap values for each column. These values are used to determine the corresponding rank caps. If NULL, column-wise geometric mean expression values are used. |
| handle_ties | Character string specifying how tied expression values should be ranked. Passed directly to MatrixGenerics::colRanks(ties.method = ...).<br>Supported values are:<br>**"min"** Assign the minimum rank to tied values (default).<br>**"max"** Assign the maximum rank to tied values.<br>**"average"** Assign the average rank to tied values.<br>**"random"** Break ties at random. |

## Details

get_ranks() computes column-wise gene ranks from expression count data and derives a rank cap for each sample, cell, or spatial location. The resulting rank matrix and rank caps can be supplied directly to get_scores.

Internally, ranks are computed using MatrixGenerics::colRanks(). Gene expression values are ranked in descending order, such that the most highly expressed gene in a column receives rank 1.

This function is implemented as an S4 generic and supports multiple input formats for counts including base matrices, sparse matrices, delayed matrices, and data frames.

A count cap value is appended to each column before ranking and its resulting rank is extracted as the column-specific rank cap. By default, count caps are derived from the geometric mean expression value of each column.

The returned ranks matrix and rank_caps vector are intended for use with get_scores.

## Value

A named list with two elements:

**ranks** Matrix of gene ranks with the same dimensions as counts. Ranks are integer-valued unless handle_ties = "average".

**rank_caps** Numeric vector containing the rank cap associated with each column.

## Examples

```
counts <- matrix(
sample(0:10, 500, replace = TRUE),
nrow = 50,
dimnames = list(
    paste0("gene", 1:50),
    paste0("cell", 1:10)
)
)

# Compute ranks
rank_results <- get_ranks(counts)

# Extract outputs
ranks <- rank_results$ranks
rank_caps <- rank_results$rank_caps

# Use custom tie handling
rank_results <- get_ranks(
counts,
handle_ties = "average"
)
```

---

# get_scores *Compute SPAROscores from pre-computed gene ranks*

---

## Description

Computes SPAROscores for one or more gene signatures using the rank matrix and rank caps generated by get_ranks.

## Usage

```
get_scores(
  ranks,
  rank_caps,
  signatures,
  down_signatures = NULL,
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'ANY,ANY,character'
get_scores(
  ranks,
  rank_caps,
  signatures,
  down_signatures = NULL,
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'ANY,ANY,list'
get_scores(
  ranks,
  rank_caps,
  signatures,
  down_signatures = NULL,
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'ANY,ANY,GeneSet'
get_scores(
  ranks,
  rank_caps,
  signatures,
  down_signatures = NULL,
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'ANY,ANY,GeneSetCollection'
get_scores(
  ranks,
  rank_caps,
  signatures,
  down_signatures = NULL,
  handle_missing_genes = "skip",
  prefix = ""
```

)

## Arguments

| | |
|---|---|
| ranks | A numeric matrix of gene ranks named ranks returned by get_ranks. Rows correspond to genes and columns correspond to samples, cells, or spatial domains. |
| rank_caps | A named numeric vector containing the rank cap associated with each column of ranks. Typically obtained from the rank_caps component returned by get_ranks.<br>When supplying custom rank caps, names should match the column names of ranks. |
| signatures | Gene signature(s) to score.<br>Supported inputs include:<br>• A character vector representing a single gene signature.<br>• A named list of character vectors representing multiple gene signatures.<br>• A GeneSet object.<br>• A GeneSetCollection object.<br>Gene identifiers must use the same naming convention as the row names of ranks. |
| down_signatures | Gene signature(s) to be considered for scoring the down-regulation effect. Supported inputs are same as signatures. If provided, names(down_signatures) must match names(signatures). Defaults to NULL.<br>When down_signatures is not NULL, Final Score <- Score(signatures) - Score(down_signatures) |
| handle_missing_genes | Character string specifying how signature genes absent from ranks should be handled.<br>Supported options are:<br>**"skip"** Exclude missing genes from score calculation (default).<br>**"impute"** Include missing genes by assigning them the capped rank value. |
| prefix | Character string to be appended before the headers of the columns returning scores. Defaults to "". |

## Details

SPAROscores quantify signature activity based on the normalized Spearman footrule distance between observed signature gene ranks and a column-specific rank cap. Higher scores indicate that signature genes tend to occupy higher expression ranks, whereas lower scores indicate weaker signature activity.

This function is implemented as an S4 generic and supports multiple input formats for signatures, including character vectors, named lists, GeneSet objects, and GeneSetCollection objects.

The supplied signature genes are first matched against the genes available in ranks. Missing genes are reported and handled according to handle_missing_genes.

For each sample, cell, or spatial domain, SPAROscores are computed as the normalized Spearmanc-footrule distance between observed signature gene ranks and the column-specific rank cap.

Rank caps typically correspond to the rank of the geometric mean expression value estimated by get_ranks, although custom rank caps may also be supplied.

## Value

A numeric matrix of SPAROscores.

For a single signature, the returned matrix contains one column named "SPAROscore" and one row per sample, cell, or spatial domain.

For multiple signatures, rows correspond to samples, cells, or spatial domains and columns correspond to signatures.

## Examples

```
counts <- matrix(
sample(0:10, 500, replace = TRUE),
nrow = 50,
dimnames = list(
    paste0("gene", 1:50),
    paste0("cell", 1:10)
)
)

# Compute ranks and rank caps
rank_results <- get_ranks(counts)

# Score a single gene signature
scores <- get_scores(
ranks = rank_results$ranks,
rank_caps = rank_results$rank_caps,
signatures = c("gene1", "gene3")
)

# Score multiple signatures
scores <- get_scores(
ranks = rank_results$ranks,
rank_caps = rank_results$rank_caps,
signatures = list(
sigA = c("gene1", "gene3"),
sigB = c("gene2", "gene4", "gene6")
)
)
```

---

sparoscore *Compute SPAROscores*

---

## Description

Compute SPAROscores for one or more gene signatures from a variety of supported data containers. ‘sparoscore()‘ is an S4 generic with methods for matrix-like objects, Seurat objects, and Bioconductor ‘SummarizedExperiment‘-derived classes. Depending on the input type, the function either returns a matrix of SPAROscores or appends SPAROscores to the input object’s metadata.

## Usage

```
sparoscore(
  data,
  signatures,
  down_signatures = NULL,
  data_has_ranks = FALSE,
  assay = "RNA",
  layer = "counts",
  count_caps = NULL,
  rank_caps = NULL,
  handle_ties = "min",
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'matrix'
sparoscore(
  data,
  signatures,
  down_signatures = NULL,
  data_has_ranks = FALSE,
  assay = NULL,
  layer = NULL,
  count_caps = NULL,
  rank_caps = NULL,
  handle_ties = "min",
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'sparseMatrix'
sparoscore(
  data,
  signatures,
  down_signatures = NULL,
  data_has_ranks = FALSE,
  assay = NULL,
  layer = NULL,
  count_caps = NULL,
  rank_caps = NULL,
  handle_ties = "min",
```

```
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'DelayedMatrix'
sparoscore(
  data,
  signatures,
  down_signatures = NULL,
  data_has_ranks = FALSE,
  assay = NULL,
  layer = NULL,
  count_caps = NULL,
  rank_caps = NULL,
  handle_ties = "min",
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'data.frame'
sparoscore(
  data,
  signatures,
  down_signatures = NULL,
  data_has_ranks = FALSE,
  assay = NULL,
  layer = NULL,
  count_caps = NULL,
  rank_caps = NULL,
  handle_ties = "min",
  handle_missing_genes = "skip",
  prefix = ""
)

## S4 method for signature 'Seurat'
sparoscore(
  data,
  signatures,
  down_signatures = NULL,
  data_has_ranks = FALSE,
  assay = "RNA",
  layer = "counts",
  count_caps = NULL,
  rank_caps = NULL,
  handle_ties = "min",
  handle_missing_genes = "skip",
  prefix = ""
)
```

```
## S4 method for signature 'SummarizedExperiment'
sparoscore(
  data,
  signatures,
  down_signatures = NULL,
  data_has_ranks = FALSE,
  assay = "counts",
  layer = NULL,
  count_caps = NULL,
  rank_caps = NULL,
  handle_ties = "min",
  handle_missing_genes = "skip",
  prefix = ""
)
```

## Arguments

| | |
|---|---|
| data | Input data object. Supported classes include:<br>• Dense matrices (‘matrix‘)<br>• Sparse matrices (‘sparseMatrix‘)<br>• Delayed matrices (‘DelayedMatrix‘)<br>• Data frames<br>• Seurat objects<br>• ‘SummarizedExperiment‘ (and derived objects)<br>• ‘SingleCellExperiment‘<br>• ‘SpatialExperiment‘<br>• ‘RangedSummarizedExperiment‘ |
| signatures | Gene signature(s) to score. Supported inputs include:<br>• A character vector representing a single gene signature.<br>• A named list of character vectors representing multiple signatures.<br>• A ‘GeneSet‘ object.<br>• A ‘GeneSetCollection‘ object. |
| down_signatures | Gene signature(s) to be considered for scoring the down-regulation effect. Supported inputs are same as signatures. If provided, names(down_signatures) must match names(signatures). Defaults to NULL.<br>When down_signatures is not NULL, Final Score <- Score(signatures) - Score(down_signatures) |
| data_has_ranks | Logical indicating whether the supplied data already contains feature ranks. If ‘FALSE‘ (default), ranks are computed from expression values prior to scoring. |
| assay | Character string specifying the name of the assay containing the data to use. When data_has_ranks = FALSE, this assay should contain expression counts. When data_has_ranks = TRUE, it should contain pre-computed ranks. Defaults to "RNA" for Seurat objects. Defaults to "counts" for SummarizedExperiment-derived objects. Ignored for matrix-like inputs. |

`layer` Character string specifying the name of the assay layer containing the data to use. When data_has_ranks = FALSE, this layer should contain expression counts. When data_has_ranks = TRUE, it should contain pre-computed ranks. Defaults to "counts" Used only for Seurat objects.

`count_caps` Optional numeric vector containing expression thresholds values for each column in the counts data. These values are used to determine the corresponding rank caps. Only used when data_has_ranks = FALSE. If NULL (default), values are computed using `compute_geometric_average`.

`rank_caps` An optional named numeric vector containing the rank cap associated with each column of ranks data used during SPAROscore calculation. Typically obtained from the rank_caps component returned by . When data_has_ranks = TRUE and rank_cpas is NULL (default), rank caps are obtained from `get_ranks` for matrix-like objects, retrieved from the "rank_caps" column of metadata for Seurat and SummarizedExperiment objects

When both count_caps and rank_cpas are provided, rank_caps takes precedence.

`handle_ties` Character string specifying how tied expression values should be ranked. Passed directly to MatrixGenerics::colRanks(ties.method = ...).

Supported values are:

**"min"** Assign the minimum rank to tied values (default).

**"max"** Assign the maximum rank to tied values.

**"average"** Assign the average rank to tied values.

**"random"** Break ties at random.

`handle_missing_genes` Character string specifying how signature genes absent from ranks should be handled.

Supported options are:

**"skip"** Exclude missing genes from score calculation (default).

**"impute"** Include missing genes by assigning them the capped rank value.

`prefix` Character string to be appended before the headers of the columns returning scores. Defaults to "".

## Details

sparoscore() is an S4 generic that dispatches on the class of data. The method selected determines how SPAROscores are computed and where the results are stored.

For matrix-like inputs (matrix, sparseMatrix, DelayedMatrix, and data.frame), the function returns a numeric matrix of SPAROscores.

For Seurat objects, SPAROscores and rank caps are appended to the object's metadata. If ranks are computed from expression data, they are also stored in a "ranks" assay layer for future reuse.

For SummarizedExperiment-derived objects, SPAROscores and rank caps are appended to colData(). If ranks are computed from expression data, they are also stored in a "ranks" assay.

When data_has_ranks = FALSE, feature ranks are first computed using get_ranks(). When data_has_ranks = TRUE, the supplied ranks are used directly and rank computation is skipped.

The supplied signature genes are first matched against the genes available in ranks. Missing genes can either be excluded from scoring ("skip") or assigned the capped rank value and included in score calculation ("impute").

For each sample, cell, or spatial domain, SPAROscores evaluates the normalized Spearman footrule distance between observed signature gene ranks and the column-specific rank cap.

Rank caps typically correspond to the rank of the geometric mean expression value estimated by get_ranks, although custom rank caps may also be supplied.

Scores typically range from 0 to 1, where high scores indicate high signature gene expression, and low scores indicate signature expression closer to the cap value.

## Value

Method-dependent:

- Matrix-like inputs return a numeric matrix of SPAROscores.
- Seurat inputs return the Seurat object with
  - One or more metadata columns containing SPAROscores for the supplied signature(s).
  - A metadata column named "rank_caps" containing the rank cap for each cell.
  - A "ranks" layer added to the selected assay when ranks are calculated from expression data.
- SummarizedExperiment-derived inputs return the same object with
  - One or more SPAROscore columns added to colData().
  - A "rank_caps" column added to colData().
  - A "ranks" assay added when ranks are computed from expression data.

## See Also

get_ranks, get_scores,

## Examples

```
# ------------------------------------------------------------------
# Matrix-like input
# ------------------------------------------------------------------

counts <- matrix(
  sample(0:10, 500, replace = TRUE),
  nrow = 50,
  dimnames = list(
    paste0("gene", 1:50),
    paste0("cell", 1:10)
  )
)

scores <- sparoscore(
  data = counts,
  signatures = c("gene1", "gene2", "gene3")
)
```

```
# Multiple signatures
signatures <- list(
  SignatureA = c("gene1", "gene2", "gene3"),
  SignatureB = c("gene10", "gene11", "gene12")
)

scores <- sparoscore(
  data = counts,
  signatures = signatures
)

# ------------------------------------------------------------------
# Seurat object
# ------------------------------------------------------------------

seurat_object <- Seurat::CreateSeuratObject(counts = counts)

seurat_object <- sparoscore(
  data = seurat_object,
  signatures = c("gene1", "gene2", "gene3")
)

# use custom count caps to measure distance from zero expression
zero_counts <- setNames(rep(0, ncol(seurat_object)), colnames(seurat_object))

seurat_object <- sparoscore(
  data = seurat_object,
  signatures = c("gene1", "gene2", "gene3"),
  count_caps = zero_counts
)

# use custom rank caps to consider top 5% genes
custom_ranks <- setNames(
  rep(0.05*nrow(seurat_object), ncol(seurat_object)),
  colnames(seurat_object)
)

seurat_object <- sparoscore(
  data = seurat_object,
  signatures = c("gene1", "gene2", "gene3"),
  rank_caps = custom_ranks
)

# Reuse previously computed ranks
seurat_object <- sparoscore(
  data = seurat_object,
  signatures = c("gene1", "gene2", "gene3"),
  data_has_ranks = TRUE,
  layer = "ranks"
)

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

```
# SingleCellExperiment / SummarizedExperiment
# ------------------------------------------------------------------

sce <- SingleCellExperiment::SingleCellExperiment(
  assays = list(counts = counts)
)

sce <- sparoscore(
  data = sce,
  signatures = c("gene1", "gene2", "gene3")
)

# Reuse previously computed ranks
sce <- sparoscore(
  data = sce,
  signatures = c("gene1", "gene2", "gene3"),
  data_has_ranks = TRUE,
  assay = "ranks"
)
```

# Index