

# Package ‘Seqtometry’

August 14, 2026

**Title** Signature scoring for single cell analysis

**Version** 1.0.1

**URL** <https://github.com/HawigerLab/Seqtometry>

**BugReports** <https://github.com/HawigerLab/Seqtometry/issues>

## Description

This package provides functions used in Seqtometry (Kousnetsov et al. 2024), a method for analyzing single cell (scRNA-seq or scATAC-seq) data via signature (gene set) enrichment scores. The Seqtometry scores may be useful for annotating or characterizing cells, either in a flow cytometry like workflow (where scores are standalone features used for progressive partitioning as described in the Seqtometry publication) or in a cluster-based workflow (as features of clusters). The exported impute function (a port of Python's MAGIC-impute, van Dijk et al. 2018), may also be useful for single cell analysis on its own.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Depends** R (>= 4.5.0)

**LinkingTo** Rcpp, RcppArmadillo

**Imports** BiocSingular, checkmate, data.table, DelayedMatrixStats, future.apply, Matrix, MatrixGenerics, purrr, Rcpp, RcppHNSW, RSpectra, sparseMatrixStats, zeallot

**Suggests** BiocStyle, box, dplyr, future, ggplot2, harmony, knitr, MASS, patchwork, rmarkdown, scater, scuttle, SingleCellExperiment, stringr, TENxPBMCDData, testthat (>= 3.0.0), tibble

**biocViews** SingleCell, GeneSetEnrichment, GeneExpression

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/Seqtometry>

**git\_branch** RELEASE\_3\_23

**git\_last\_commit** 6ff53d7

**git\_last\_commit\_date** 2026-07-27  
**Repository** Bioconductor 3.23  
**Date/Publication** 2026-08-13  
**Author** Robert Kousnetsov [aut, cre],  
Daniel Hawiger [cph, fnd]  
**Maintainer** Robert Kousnetsov <robert.kousnetsov@health.slu.edu>

Contents

|                              |           |
|------------------------------|-----------|
| Seqtometry-package . . . . . | 2         |
| .apply_diff_op . . . . .     | 3         |
| .calc_diff_op . . . . .      | 3         |
| .calc_pca . . . . .          | 4         |
| .check_params . . . . .      | 4         |
| .gene_indices . . . . .      | 5         |
| .invert_pca . . . . .        | 5         |
| .minmax_scale . . . . .      | 6         |
| .normalize . . . . .         | 6         |
| .procrustes . . . . .        | 7         |
| impute . . . . .             | 7         |
| score . . . . .              | 9         |
| wks . . . . .                | 11        |
| <b>Index</b>                 | <b>12</b> |

---

|                    |                                                   |
|--------------------|---------------------------------------------------|
| Seqtometry-package | <i>Signature scoring for single cell analysis</i> |
|--------------------|---------------------------------------------------|

---

Description

This package provides functions used in Seqtometry (Kousnetsov et al. 2024), a method for analyzing single cell (scRNA-seq or scATAC-seq) data via signature (gene set) enrichment scores. The Seqtometry scores may be useful for annotating or characterizing cells, either in a flow cytometry like workflow (where scores are standalone features used for progressive partitoning as demonstrated in the Seqtometry publication) or in a cluster-based workflow (as features of clusters). The exported impute function (a port of Python’s MAGIC-impute, van Dijk et al. 2018), may also be useful for single cell analysis on its own.

Author(s)

Robert Kousnetsov <robert.kousnetsov@health.slu.edu>  
Maintainer: Robert Kousnetsov <robert.kousnetsov@health.slu.edu>

---

|                             |                               |
|-----------------------------|-------------------------------|
| <code>.apply_diff_op</code> | <i>Perform data diffusion</i> |
|-----------------------------|-------------------------------|

---

**Description**

Apply diffusion operator to data in order to perform imputation.

**Usage**

```
.apply_diff_op(gex, pcs, aff, dft, t_max, tol, exact_solver)
```

**Arguments**

|                           |                                                      |
|---------------------------|------------------------------------------------------|
| <code>gex</code>          | <b>matrix or Matrix</b> Gene expression matrix       |
| <code>pcs</code>          | <b>matrix</b> Principal components matrix            |
| <code>aff</code>          | <b>dgCMatrix</b> Markov affinity matrix              |
| <code>dft</code>          | <b>NULL or integer(1)</b> Diffusion time             |
| <code>t_max</code>        | <b>integer(1)</b> Maximum diffusion time             |
| <code>tol</code>          | <b>numeric(1)</b> Tolerance for Procrustes disparity |
| <code>exact_solver</code> | <b>logical(1)</b> Perform imputation in gene space   |

**Value**

**list(matrix, integer(1))** Imputed matrix and diffusion time used

---

|                            |                                   |
|----------------------------|-----------------------------------|
| <code>.calc_diff_op</code> | <i>Compute diffusion operator</i> |
|----------------------------|-----------------------------------|

---

**Description**

Calculate graph diffusion operator (Markov affinity matrix).

**Usage**

```
.calc_diff_op(pcs, knn, ka, dist_metric)
```

**Arguments**

|                          |                                                                                       |
|--------------------------|---------------------------------------------------------------------------------------|
| <code>pcs</code>         | <b>matrix</b> Principal components matrix (used for kNN search)                       |
| <code>knn</code>         | <b>integer(1)</b> Number of nearest neighbors to search for                           |
| <code>ka</code>          | <b>integer(1)</b> Number of nearest neighbors to use for adaptive kernel              |
| <code>dist_metric</code> | <b>character(1)</b> Type of metric to use for distance calculations during kNN search |

**Value**

**dgCMatrix** Markov affinity matrix

---

*.calc\_pca**PCA wrapper*

---

**Description**

Calculate leading principal components (via truncated singular value decomposition).

**Usage**

```
.calc_pca(gex, npc, scale)
```

**Arguments**

|                    |                                                                                  |
|--------------------|----------------------------------------------------------------------------------|
| <code>gex</code>   | <b>matrix or Matrix</b> Gene expression matrix (without any zero variance genes) |
| <code>npc</code>   | <b>numeric(1)</b> Number of leading principal components to compute              |
| <code>scale</code> | <b>logical(1)</b> Whether to scale genes to unit variance                        |

**Value**

**list** PC loading/rotation matrices as well as centering/scaling vectors

---

*.check\_params**Parameter validation*

---

**Description**

Checks that all parameters used in `impute` are valid.

**Usage**

```
.check_params(args)
```

**Arguments**

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>args</code> | <b>list</b> The arguments to the <code>magic_impute</code> function |
|-------------------|---------------------------------------------------------------------|

**Value**

NULL (but stops execution for invalid parameters)

---

|               |                                             |
|---------------|---------------------------------------------|
| .gene_indices | <i>Finds row indices of signature genes</i> |
|---------------|---------------------------------------------|

---

### Description

For converting from character to integer based indexing

### Usage

```
.gene_indices(mat, gss)
```

### Arguments

|     |                                                                                       |
|-----|---------------------------------------------------------------------------------------|
| mat | <b>matrix-like</b> Gene expression data (genes x cells)                               |
| gss | <b>named list of character</b> Signature genes (with same nomenclature system as mat) |

### Value

**integer** 0-based indices (for passing to Rcpp function) of signature genes

---

|             |                      |
|-------------|----------------------|
| .invert_pca | <i>PCA inversion</i> |
|-------------|----------------------|

---

### Description

Reverses operations done for PCA: back-rotation, unscaling, and uncentering.

### Usage

```
.invert_pca(pcs, rot, ctr, sdv, low_mem)
```

### Arguments

|         |                                                                               |
|---------|-------------------------------------------------------------------------------|
| pcs     | <b>matrix</b> The principal components (scaled left singular vectors)         |
| rot     | <b>matrix</b> The rotation matrix (right singular vectors)                    |
| ctr     | <b>integer</b> The centering vector                                           |
| sdv     | <b>integer or NULL</b> The scaling vector (or NULL if no scaling was applied) |
| low_mem | <b>logical(1)</b> Whether to use delayed operations to reduce memory usage    |

### Value

**matrix or DelayedMatrix**  $\text{rot} \% \% \text{t}(\text{pcs}) * \text{sdv} + \text{ctr}$

---

|                            |                         |
|----------------------------|-------------------------|
| <code>.minmax_scale</code> | <i>Minmax transform</i> |
|----------------------------|-------------------------|

---

**Description**

Scales input vector to unit range

**Usage**

```
.minmax_scale(x)
```

**Arguments**

|                |                                    |
|----------------|------------------------------------|
| <code>x</code> | <b>numeric</b> Values to be scaled |
|----------------|------------------------------------|

**Value**

**numeric** Minmax transformed values

---

|                         |                           |
|-------------------------|---------------------------|
| <code>.normalize</code> | <i>LogCP10K transform</i> |
|-------------------------|---------------------------|

---

**Description**

Simple normalization method for scRNA-seq data.

**Usage**

```
.normalize(gex)
```

**Arguments**

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>gex</code> | <b>matrix or Matrix</b> Gene expression matrix (cells x genes) |
|------------------|----------------------------------------------------------------|

**Value**

**matrix or Matrix** Transformed (normalized) matrix

---

|             |                             |
|-------------|-----------------------------|
| .procrustes | <i>Procrustes disparity</i> |
|-------------|-----------------------------|

---

### Description

Calculates symmetric Procrustes distance (adapted from MATLAB procrustes).

### Usage

```
.procrustes(x, y)
```

### Arguments

|   |               |
|---|---------------|
| x | <b>matrix</b> |
| y | <b>matrix</b> |

### Value

**numeric(1)** Procrustes disparity between input matrices

---

|        |                                                |
|--------|------------------------------------------------|
| impute | <i>MAGIC imputation (van Dijk et al. 2018)</i> |
|--------|------------------------------------------------|

---

### Description

Calculates a graph diffusion operator for the given input matrix and applies it to produce an imputed matrix.

### Usage

```
impute(  
  gex,  
  transpose = TRUE,  
  do_norm = FALSE,  
  pca = NULL,  
  npc = 100L,  
  scale = TRUE,  
  knn = 16L,  
  ka = 6L,  
  dist_metric = "euclidean",  
  dft = NULL,  
  t_max = 16L,  
  tol = 0.001,  
  exact_solver = TRUE,  
  conserve_memory = FALSE,
```

```

    env_ret = FALSE,
    verbose = FALSE
)

```

### Arguments

|                              |                                                                                                                           |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <code>gex</code>             | <b>matrix or Matrix</b> Gene expression values (that has passed quality control).                                         |
| <code>transpose</code>       | <b>logical(1)</b> Whether to transpose <code>gex</code> (make it cells x genes) prior to downstream operations.           |
| <code>do_norm</code>         | <b>logical(1)</b> Whether to perform LogCP10K normalization on <code>gex</code> .                                         |
| <code>pca</code>             | <b>matrix (cells x PCs) or NULL</b> Precomputed principal component matrix (or NULL to derive it from <code>gex</code> ). |
| <code>npc</code>             | <b>integer(1)</b> Number of principal components (min = 1) to calculate.                                                  |
| <code>scale</code>           | <b>logical(1)</b> Whether to scale columns of input matrix to unit variance prior to PCA.                                 |
| <code>knn</code>             | <b>integer(1)</b> Number of nearest neighbors (min = 2) to consider during distance calculation.                          |
| <code>ka</code>              | <b>integer(1)</b> Number of nearest neighbors (min = 2, max <= <code>knn</code> ) to use for the adaptive kernel.         |
| <code>dist_metric</code>     | <b>character(1)</b> Type of metric to use for distance calculations during kNN search.                                    |
| <code>dft</code>             | <b>NULL or integer(1)</b> Automatic (NULL) or user-defined (integer) diffusion time (min = 1, max = 16).                  |
| <code>t_max</code>           | <b>integer(1)</b> Maximum diffusion time to test when using automatic diffusion time (min = 1, max = 16).                 |
| <code>tol</code>             | <b>numeric(1)</b> Threshold for Procrustes disparity (min = 0, max = 1) between successive diffusion times.               |
| <code>exact_solver</code>    | <b>logical(1)</b> Whether to perform imputation in gene space (TRUE) or PCA space (FALSE).                                |
| <code>conserve_memory</code> | <b>logical(1)</b> Whether to avoid allocating a large dense matrix when <code>exact_solver</code> = FALSE.                |
| <code>env_ret</code>         | <b>logical(1)</b> Return all variables in the environment (TRUE) or just the imputed matrix (FALSE).                      |
| <code>verbose</code>         | <b>logical(1)</b> Whether to print messages at different major parts of the algorithm.                                    |

### Value

**matrix-like or list** If `env_ret` = FALSE, then just the imputed matrix. Otherwise the function environment as a list containing all parameters (possibly modified) as well as

- `imp` **matrix or DelayedMatrix** Imputed matrix.
- `aff` **dgCMatrix** Markov affinity matrix (graph diffusion operator).
- `pca` **list** Possibly computed (if `pca` was NULL), yielding a four element list, where:
  - `x` **matrix (cells x PCs)** The principal components matrix (scaled left singular vectors).



- **v matrix (genes x PCs)** The rotation matrix (right singular vectors).
- center **integer (cells)** The centering vector.
- scale **integer (cells) or NULL** The scaling vector (or NULL if no scaling was applied).

## Examples

```
box::use(
  TENxPBMCData[TENxPBMCData],
  SingleCellExperiment[rowData, logcounts],
  scuttle[quickPerCellQC, logNormCounts],
  scater[runUMAP, plotReducedDim],
  patchwork[wrap_plots])

# PBMC data, basic processing pipeline
dat <- TENxPBMCData(dataset = "pbmc3k")
dimnames(dat) <- list(
  rowData(dat)[["Symbol_TENx"]],
  dat[["Barcode"]])
dat <- dat |>
  quickPerCellQC() |>
  logNormCounts() |>
  runUMAP()

# MAGIC imputation
imp <- logcounts(dat) |>
  as("dgCMatrx") |>
  impute()

# Visualize unimputed versus imputed expression
# on UMAP plots for a gene of interest (GOI)
goi <- "CD19"
dat[["Imputed_GOI"]] <- imp[goi, ]
p1 <- plotReducedDim(dat, "UMAP", color_by = goi)
p2 <- plotReducedDim(dat, "UMAP", color_by = "Imputed_GOI")
wrap_plots(p1, p2, ncol = 2)
```

---

score

*Seqtometry scoring (Kousnetsov et al. 2024)*

---

## Description

Computes signature scores (a weighted KS-like statistic) for single cell expression data

## Usage

```
score(mat, signatures, minmax = TRUE)
```

## Arguments

|                         |                                                                                                          |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| <code>mat</code>        | <b>matrix, Matrix, or DelayedMatrix</b> Gene expression data (genes x cells)                             |
| <code>signatures</code> | <b>named list of character</b> Signature genes (with same nomenclature system used in <code>mat</code> ) |
| <code>minmax</code>     | <b>logical(1)</b> Whether to perform minmax transform on scoring results (default: TRUE)                 |

## Value

**data.table** Single cell scores (cells x signatures) for each signature, where cell barcodes are stored in the "id" column

## Examples

```
box::use(
  TENxPBMCData[TENxPBMCData],
  SingleCellExperiment[rowData, logcounts],
  scuttle[quickPerCellQC, logNormCounts],
  scater[runUMAP, plotReducedDim],
  patchwork[wrap_plots])

# PBMC data, basic processing pipeline
dat <- TENxPBMCData(dataset = "pbmc3k")
dimnames(dat) <- list(
  rowData(dat)[["Symbol_TENx"]],
  dat[["Barcode"]])
dat <- dat |>
  quickPerCellQC() |>
  logNormCounts() |>
  runUMAP()

# MAGIC imputation
imp <- logcounts(dat) |>
  as("dgCMatrx") |>
  impute()

# Score with a B cell signature (gene set)
options(future.globals.maxSize = 1024^3)
b_cell_sig <- list("B_cell" = c("CD19", "MS4A1", "CD79A", "CD79B"))
dat[["B cell signature"]] <- Seqtometry::score(imp, b_cell_sig)[["B_cell"]]

# Visualize a hallmark B cell gene versus a B cell signature score
p1 <- plotReducedDim(dat, "UMAP", color_by = "CD19")
p2 <- plotReducedDim(dat, "UMAP", color_by = "B cell signature")
wrap_plots(p1, p2, ncol = 2)
```

---

|     |                                                                                     |
|-----|-------------------------------------------------------------------------------------|
| wks | <i>Helper function for performing a weighted Kolmogorov-Smirnov-like procedure.</i> |
|-----|-------------------------------------------------------------------------------------|

---

**Description**

Helper function for performing a weighted Kolmogorov-Smirnov-like procedure.

**Usage**

```
wks(gex, gss, mus, sds)
```

**Arguments**

|     |                                            |
|-----|--------------------------------------------|
| gex | numeric: normalized gene expression values |
| gss | list: indices of genes in each gene set    |
| mus | numeric: means of all genes                |
| sds | numeric: standard deviations of all genes  |

**Value**

Modified Kuiper statistic (sum of minimal and maximal deviations during running sum)

# Index

[.apply\\_diff\\_op](#), 3  
[.calc\\_diff\\_op](#), 3  
[.calc\\_pca](#), 4  
[.check\\_params](#), 4  
[.gene\\_indices](#), 5  
[.invert\\_pca](#), 5  
[.minmax\\_scale](#), 6  
[.normalize](#), 6  
[.procrustes](#), 7  
  
[impute](#), 7  
  
[score](#), 9  
[Seqtometry \(Seqtometry-package\)](#), 2  
[Seqtometry-package](#), 2  
  
[wks](#), 11