

# Package ‘spoon’

May 26, 2026

**Title** Address the Mean-variance Relationship in Spatial  
Transcriptomics Data

**Version** 1.9.0

**Description** This package addresses the mean-variance relationship in spatially resolved transcriptomics data. Precision weights are generated for individual observations using Empirical Bayes techniques. These weights are used to rescale the data and covariates, which are then used as input in spatially variable gene detection tools.

**URL** <https://github.com/kinnaryshah/spoon>

**BugReports** <https://github.com/kinnaryshah/spoon/issues>

**Imports** SpatialExperiment, BRISC, nnSVG, BiocParallel, Matrix,  
methods, SummarizedExperiment, stats, utils, scuttle

**License** MIT + file LICENSE

**Encoding** UTF-8

**biocViews** Spatial, SingleCell, Transcriptomics, GeneExpression,  
Preprocessing

**Depends** R (>= 4.4)

**Roxygen** list(markdown = TRUE)

**Suggests** testthat, STexampleData, knitr, rmarkdown, BiocStyle

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

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

**git\_branch** devel

**git\_last\_commit** a0a9ed9

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-25

**Author** Kinnary Shah [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-7098-2116>>),  
Boyi Guo [aut] (ORCID: <<https://orcid.org/0000-0003-2950-2349>>),  
Stephanie C. Hicks [aut] (ORCID:  
<<https://orcid.org/0000-0002-7858-0231>>)

**Maintainer** Kinnary Shah <kinnaryshahh@gmail.com>

## Contents

generate_weights . . . . .	2
weighted_nnSVG . . . . .	3
<b>Index</b>	<b>6</b>

---

generate_weights	<i>Generate weights</i>
------------------	-------------------------

---

### Description

Generate weights on the observation level for each gene

### Usage

```
generate_weights(
  input,
  spatial_coords = NULL,
  assay_name = "counts",
  stabilize = TRUE,
  n_threads = 1,
  BPPARAM = NULL
)
```

### Arguments

input	either a SpatialExperiment object which contains a counts matrix, or a counts matrix
spatial_coords	matrix containing columns of spatial coordinates, needed if input is a matrix
assay_name	if using a SpatialExperiment object, name of the assay in which the counts matrix is stored
stabilize	when TRUE, stabilize weights to avoid extrapolation (highly recommended)
n_threads	default = 1, number of threads for parallelization
BPPARAM	optional additional argument for parallelization to use BiocParallel

### Details

This function generates weights for each observation, which are used as input to scale the data and covariates

### Value

weights matrix

**Examples**

```

library(nnSVG)
library(STexampleData)
library(SpatialExperiment)
library(BRISC)
library(BiocParallel)
library(scuttle)

spe <- Visium_humanDLPFC()

# keep spots over tissue
spe <- spe[, colData(spe)$in_tissue == 1]

# filter low-expressed and mitochondrial genes
spe <- filter_genes(spe)

# calculate logcounts (log-transformed normalized counts) using scran package
spe <- computeLibraryFactors(spe)
spe <- logNormCounts(spe)

known_genes <- c("MOBP", "PCP4", "SNAP25", "HBB", "IGKC", "NPY")
ix_known <- which(rowData(spe)$gene_name %in% known_genes)
ix <- c(ix_known)

spe <- spe[ix, ]

spe <- spe[, colSums(logcounts(spe)) > 0]

#EXAMPLE 1 USING SPATIAL EXPERIMENT

set.seed(1)
weights_1 <- generate_weights(input = spe,
                             stabilize = TRUE)

#EXAMPLE 2 USING MATRIX

counts_mat <- counts(spe)
logcounts_mat <- logcounts(spe)
coords_mat <- spatialCoords(spe)

set.seed(1)
weights_2 <- generate_weights(input = counts_mat,
                             spatial_coords = coords_mat,
                             stabilize = TRUE)

```

---

weighted\_nnSVG

*Weighted nnSVG*


---

**Description**

Run nnSVG for SVG detection using the weights

**Usage**

```
weighted_nnSVG(
  input,
  spatial_coords = NULL,
  assay_name = "logcounts",
  w,
  n_threads = 1,
  BPPARAM = MulticoreParam(workers = 1)
)
```

**Arguments**

input	either a SpatialExperiment object which contains a logcounts matrix, or a logcounts matrix
spatial_coords	matrix containing columns of spatial coordinates, needed if input is a matrix
assay_name	if using a SpatialExperiment object, name of the assay in which the logcounts matrix is stored
w	weights matrix
n_threads	default = 1, number of threads for parallelization
BPPARAM	optional additional argument for parallelization to use BiocParallel

**Details**

This function incorporates weights for each observation to run nnSVG

**Value**

either spe with weighted nnSVG statistics, or matrix with weighted nnSVG statistics

**Examples**

```
library(nnSVG)
library(STexampleData)
library(SpatialExperiment)
library(BRISC)
library(BiocParallel)
library(scuttle)
library(Matrix)

spe <- Visium_humanDLPFC()

# keep spots over tissue
spe <- spe[, colData(spe)$in_tissue == 1]

# filter low-expressed and mitochondrial genes
spe <- filter_genes(spe)

# calculate logcounts (log-transformed normalized counts) using scran package
spe <- computeLibraryFactors(spe)
spe <- logNormCounts(spe)

known_genes <- c("MOBP", "PCP4", "SNAP25", "HBB", "IGKC", "NPY")
ix_known <- which(rowData(spe)$gene_name %in% known_genes)
```

```
ix <- c(ix_known)

spe <- spe[ix, ]

spe <- spe[, colSums(logcounts(spe)) > 0]

#EXAMPLE 1 USING SPATIAL EXPERIMENT

set.seed(1)
weights_1 <- generate_weights(input = spe,
                             stabilize = TRUE)
spe_results <- weighted_nnSVG(input = spe,
                              w = weights_1,
                              BPPARAM = MulticoreParam(workers = 1,
                                                         RNGseed = 4))

# display results
rowData(spe_results)

#EXAMPLE 2 USING MATRIX

counts_mat <- counts(spe)
logcounts_mat <- logcounts(spe)
coords_mat <- spatialCoords(spe)

set.seed(1)
weights_2 <- generate_weights(input = counts_mat,
                             spatial_coords = coords_mat,
                             stabilize = TRUE)
results <- weighted_nnSVG(input = logcounts_mat,
                          spatial_coords = coords_mat,
                          w = weights_2,
                          BPPARAM = MulticoreParam(workers = 1, RNGseed = 4))

# display results
print(results)
```

# Index

`generate_weights`, [2](#)

`weighted_nnSVG`, [3](#)