Package: tpSVG (via r-universe)

November 10, 2026

Title Thin plate models to detect spatially variable genes

Version 1.8.0

Description The goal of `tpSVG` is to detect and visualize spatial variation in the gene expression for spatially resolved transcriptomics data analysis. Specifically, `tpSVG` introduces a family of count-based models, with generalizable parametric assumptions such as Poisson distribution or negative binomial distribution. In addition, comparing to currently available count-based model for spatially resolved data analysis, the `tpSVG` models improves computational time, and hence greatly improves the applicability of count-based models in SRT data analysis.

License MIT + file LICENSE

URL https://github.com/boyiguo1/tpSVG

BugReports https://github.com/boyiguo1/tpSVG/issues

biocViews Spatial, Transcriptomics, GeneExpression, Software, StatisticalMethod, DimensionReduction, Regression, Preprocessing

Encoding UTF-8

Depends mgcv, R (>= 4.4)

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports stats, BiocParallel, MatrixGenerics, methods, SingleCellExperiment, SummarizedExperiment, SpatialExperiment

Suggests BiocStyle, knitr, nnSVG, rmarkdown, scran, scuttle, STexampleData, escheR, ggpubr, colorspace, BumpyMatrix, sessioninfo, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libmagick++-dev gsfonts libicu-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 13:02:29 UTC

RemoteUrl https://github.com/bioc/tpSVG

RemoteRef RELEASE_3_23

RemoteSha ca85de0f0b9771b434f29a3b71aa527d883d321e

Contents

tpSVG 2

Index **5**

`tpSVG` *Thin Plate Spline Model to Detect Spatially Variable Genes*

Description

Thin Plate Spline Model to Detect Spatially Variable Genes

Usage

```
tpSVG(
  input,
  spatial_coords = NULL,
  X = NULL,
  family = poisson(),
  offset = log(input$sizeFactor),
  weights = NULL,
  assay_name = "counts",
  n_threads = 1,
  BPPARAM = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`input`	`SpatialExperiment` or `numeric` matrix: Input data, which can either be a `SpatialExperiment` object or a `numeric` matrix of values. If it is a `SpatialExperiment` object, it is assumed to have an `assay` slot containing either logcounts (e.g. from the `scran` package) or deviance residuals (e.g. from the `scry` package), and a `spatialCoords` slot containing spatial coordinates of the measurements. If it is a `numeric` matrix, the values are assumed to already be normalized and transformed (e.g. logcounts), formatted as `rows` = `genes` and `columns` = `spots`, and a separate `numeric` matrix of spatial coordinates must also be provided with the `spatial_coords` argument.

spatial_coords — numeric matrix: Matrix containing columns of spatial coordinates, formatted as rows = spots. This must be provided if input is provied as a numeric matrix of values, and is ignored if input is provided as a SpatialExperiment object. Default = NULL.

X — numeric matrix: Optional design matrix containing columns of covariates per spatial location, e.g. known spatial domains. Number of rows must match the number of spatial locations. Default = NULL, which fits an intercept-only model.

family — a description of the error distribution and link function to be used in the model. Currently support two distributions poisson and gaussian

offset — This can be used to account for technician variation when family = poisson model is used to model raw counts. offset should take in the log-transformed scale factor, e.g. offset = log(spe$sizeFactor), library size, or other normalization factor.

weights — Reserved for future development, e.g. correcting mean-var relationship for Gaussian models. Please use with caution.

assay_name — character: If input is provided as a SpatialExperiment object, this argument selects the name of the assay slot in the input object containing the preprocessed gene expression values. For example, logcounts for log-transformed normalized counts from the scran package, or binomial_deviance_residuals for deviance residuals from the scry package. Default = "logcounts", or ignored if input is provided as a numeric matrix of values.

n_threads — integer: Number of threads for parallelization. Default = 1. We recommend setting this equal to the number of cores available (if working on a laptop or desktop) or around 10 or more (if working on a compute cluster).

BPPARAM — BiocParallelParam: Optional additional argument for parallelization. This argument is provided for advanced users of BiocParallel for further flexibility for parallelization on some operating systems. If provided, this should be an instance of BiocParallelParam. For most users, the recommended option is to use the n_threads argument instead. Default = NULL, in which case n_threads will be used instead.

verbose — logical: Whether to display verbose output for model fitting and parameter estimation from BRISC. Default = FALSE.

... — Reserved for future arguments.

Value

If the input was provided as a SpatialExperiment object, the output values are returned as additional columns in the rowData slot of the input object. If the input was provided as a numeric matrix of values, the output is returned as a numeric matrix. The output values include p-values without any adjustment and statistics reporting reporting the thinplate spline model. The test_stat entry of the returned object is the test statistic for the corresponding model, that is F statistics for the gaussian model and the Chi-squared statistics for generalized models.

Examples

```
library(SpatialExperiment)
library(STexampleData)
library(scran)
library(nnSVG)

# load example dataset from STexampleData package
spe <- Visium_humanDLPFC()

# preprocessing steps

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

# skip spot-level quality control, since this has been performed previously
# on this dataset
# Add library size
spe <- addPerCellQCMetrics(spe)

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

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



# select small number of genes for faster runtime in this example
set.seed(123)
ix <- sample(seq_len(nrow(spe)), 4)
spe <- spe[ix, ]

# run tpSVG
set.seed(123)

# Gaussian Model
spe_gaus <- tpSVG(
 spe,
 family = gaussian(),
 assay_name = "logcounts"
 )

# Poisson Model
spe_poisson  <- tpSVG(
 spe,
 family = poisson,
 assay_name = "counts",
 offset = log(spe$sizeFactor)   # Natural log library size
 )
```

Index

tpSVG, 2