Package 'smoothclust'

January 11, 2025

Version 1.3.0

Title smoothclust

Description Method for segmentation of spatial domains and spatially-aware clustering in spatial transcriptomics data. The method generates spatial domains with smooth boundaries by smoothing gene expression profiles across neighboring spatial locations, followed by unsupervised clustering. Spatial domains consisting of consistent mixtures of cell types may then be further investigated by applying cell type compositional analyses or differential analyses.

URL https://github.com/lmweber/smoothclust

BugReports https://github.com/lmweber/smoothclust/issues

License MIT + file LICENSE

Encoding UTF-8

biocViews Spatial, SingleCell, Transcriptomics, GeneExpression, Clustering

Depends R (>= 4.4.0)

Imports SpatialExperiment, SummarizedExperiment, sparseMatrixStats, spdep, methods, utils

VignetteBuilder knitr

Suggests BiocStyle, knitr, STexampleData, scuttle, scran, scater, ggspavis, testthat

RoxygenNote 7.3.2

git_url https://git.bioconductor.org/packages/smoothclust

git_branch devel

git_last_commit 28cfdec

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-01-10

Author Lukas M. Weber [aut, cre] (ORCID: <https://orcid.org/0000-0002-3282-1730>)

Maintainer Lukas M. Weber <lmweberedu@gmail.com>

Contents

`smoothclust` *smoothclust*

Description

Method for segmentation of spatial domains and spatially-aware clustering.

Usage

```
smoothclust(
  input,
  assay_name = "counts",
  spatial_coords = NULL,
  method = c("uniform", "kernel", "knn"),
  bandwidth = 0.05,
  k = 18,
  truncate = 0.05,
  sparse = TRUE
)
```

Arguments

`input`	Input data, which can be provided as either a `SpatialExperiment` object or a numeric matrix. If this is a `SpatialExperiment` object, it is assumed to contain either raw expression counts or logcounts in the `assay` slots and spatial coordinates in the `spatialCoords` slot. If this is a numeric matrix, it is assumed to contain either raw expression counts or logcounts, and spatial coordinates need to be provided separately with the `spatial_coords` argument.
`assay_name`	For a `SpatialExperiment` input object, this argument specifies the name of the `assay` containing the expression values to be smoothed. In most cases, this will be `counts`, which contains raw expression counts. Alternatively, `logcounts` may also be used. Note that if `logcounts` are used, the smoothed values represent geometric averages, which are more difficult to interpret. We recommend using raw counts if possible. This argument is only used if the input is a `SpatialExperiment` object. Default = `counts`.
`spatial_coords`	Numeric matrix of spatial coordinates, assumed to contain x coordinates in first column and y coordinates in second column. This argument is only used if the input is a numeric matrix.

`method` Method used for smoothing. Options are `uniform`, `kernel`, and `knn`. The `uniform` method calculates unweighted averages across spatial locations within a circular window with radius `bandwidth` at each spatial location, which smooths out spatial variability as well as sparsity due to sampling variability. The `kernel` method calculates a weighted average using a truncated exponential kernel applied to Euclidean distances with a length scale parameter equal to `bandwidth`, which provides a more sophisticated approach to smoothing out spatial variability but may be affected by sparsity due to sampling variability (especially sparsity at the index point), and is computationally slower. The `knn` method calculates an unweighted average across the index point and its k nearest neighbors, and is the fastest method. Default = `uniform`.

`bandwidth` Bandwidth parameter for smoothing, expressed as proportion of width or height (whichever is greater) of tissue area. Only used for `method = "uniform"` or `method = "kernel"`. For `method = "uniform"`, the bandwidth represents the radius of a circle, and unweighted averages are calculated across neighboring points within this circle. For `method = "kernel"`, the averaging is weighted by distances scaled using a truncated exponential kernel applied to Euclidean distances. For example, a bandwidth of 0.05 will smooth values across neighbors weighted by distances scaled using a truncated exponential kernel with length scale equal to 5 area. Weights for `method = "kernel"` are truncated at small values for computational efficiency. Default = 0.05.

`k` Number of nearest neighbors parameter for `method = "knn"`. Only used for `method == "knn"`. Unweighted averages are calculated across the index point and its k nearest neighbors. Default = 18 (based on two layers in honeycomb pattern for 10x Genomics Visium platform).

`truncate` Truncation threshold parameter if `method = "kernel"`. Kernel weights below this value are set to zero for computational efficiency. Only used for `method = "kernel"`. Default = 0.05.

`sparse` Whether to return output assay or numeric matrix as sparse matrix. Default = TRUE. In most cases (e.g. if using `SpatialExperiment` objects) this should be left as TRUE. Set to FALSE to return a dense matrix instead.

Details

Method for segmentation of spatial domains and spatially-aware clustering in spatial transcriptomics data.

Method for segmentation of spatial domains and spatially-aware clustering in spatial transcriptomics data. The method generates spatial domains with smooth boundaries by smoothing gene expression profiles across neighboring spatial locations, followed by unsupervised clustering. Spatial domains consisting of consistent mixtures of cell types may then be further investigated by applying cell type compositional analyses or differential analyses.

Value

Returns spatially smoothed expression values, which can then be used as the input for further downstream analyses. Results are returned either as a `SpatialExperiment` object containing a new `assay` named `<assay_name>_smooth` (e.g. `counts_smooth` or `logcounts_smooth`), or as a numeric matrix, depending on the input type.

Examples

```
library(STexampleData)

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

# run smoothclust
# using "knn" method for faster runtime in this example
spe <- smoothclust(spe, method = "knn", k = 6)

# see vignette for extended example using default method and including
# downstream analysis steps
```

`smoothness_metric` *Function for smoothness metric*

Description

Function for clustering smoothness evaluation metric

Usage

```
smoothness_metric(spatial_coords, labels, k = 6)
```

Arguments

`spatial_coords`	Numeric matrix containing spatial coordinates of points, formatted as nrow = number of points, ncol = 2 (assuming x and y dimensions). For example, 'spatial_coords = spatialCoords(spe)' if using a `SpatialExperiment` object.
`labels`	Numeric vector of cluster labels for each point. For example, 'labels <- as.numeric(colData(spe)$label)' if using a `SpatialExperiment` object.
`k`	Number of k nearest neighbors to use in calculation. Default = 6 (from 10x Genomics Visium platform).

Details

Function to calculate clustering smoothness evaluation metric, defined as the average number of nearest neighbors per point that are from a different cluster. This metric can be used to quantify and compare the relative smoothness of the boundaries of clusters or spatial domains.

Value

Returns a list containing (i) a vector of values at each point (i.e. the number of nearest neighbors that are from a different cluster at each point) and (ii) the average value across all points.

Examples

```
library(STexampleData)
library(scran)
library(scater)

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

# run smoothclust
# using "knn" method for faster runtime in this example
# see vignette for example using default method
spe <- smoothclust(spe, method = "knn", k = 6)

# calculate logcounts
spe <- logNormCounts(spe, assay.type = "counts_smooth")

# preprocessing steps for clustering
# remove mitochondrial genes
is_mito <- grepl("(^mt-)", rowData(spe)$gene_name, ignore.case = TRUE)
spe <- spe[!is_mito, ]
# select top highly variable genes (HVGs)
dec <- modelGeneVar(spe)
top_hvgs <- getTopHVGs(dec, prop = 0.1)
spe <- spe[top_hvgs, ]

# dimensionality reduction
set.seed(123)
spe <- runPCA(spe)

# run k-means clustering
set.seed(123)
k <- 5
clus <- kmeans(reducedDim(spe, "PCA"), centers = k)$cluster
colLabels(spe) <- factor(clus)

# calculate smoothness metric
res <- smoothness_metric(spatialCoords(spe), as.numeric(colData(spe)$label))

# results
str(res)
head(res$n_discordant)
res$mean_discordant
```

Index