Package 'SpaTopic'

May 29, 2026

Type Package

Title Topic Inference to Identify Tissue Architecture in Multiplexed Images

Version 1.3.1

Date 2026-05-27

Description A novel spatial topic model to integrate both cell type and spatial information to identify the complex spatial tissue architecture on multiplexed tissue images without human intervention. The Package implements a collapsed Gibbs sampling algorithm for inference. The method is highly scalable to large-scale image datasets without extracting neighborhood information for every single cell. The package supports spatially resolved cell-level data analysis, topic inference, visualization, and downstream biological interpretation of tissue microenvironments.

License GPL (>= 3)

Depends R (>= 3.5.0),

Imports Rcpp (>= 0.12.0), RANN (>= 2.6.0), sf (>= 1.0-12), methods (>= 3.4), foreach (>= 1.5.0), iterators (>= 1.0),

LinkingTo Rcpp, RcppArmadillo, RcppProgress,

Suggests knitr, rmarkdown, SeuratObject (>= 4.9.9.9086), doParallel (>= 1.0), spelling

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

LazyData true

URL https://xiyupeng.github.io/SpaTopic/,
https://github.com/xiyupeng/SpaTopic,
https://doi.org/10.1038/s41467-025-61821-y

BugReports https://github.com/xiyupeng/SpaTopic/issues

Language en-US

NeedsCompilation yes

Author Xiyu Peng [aut, cre] (ORCID: <https://orcid.org/0000-0003-4232-0910>), Nikki Xiao [aut]

Maintainer Xiyu Peng <pansypeng124@gmail.com>

Repository CRAN

Date/Publication 2026-05-29 15:00:02 UTC

Contents

`frex_topic` *Calculate FREX ranked words and values for a topic*

Description

Calculate FREX ranked words and values for a topic

Usage

```
frex_topic(beta, vocab, topic = 1, top_n = NULL, frex_weight = 0.5)
```

Arguments

`beta`	A numeric matrix of dimension (topics x words) representing the probability distribution of words within each topic. Each row should sum to 1. Beta must be on the probability scale (not log scale).
`vocab`	a character vector of vocabulary terms corresponding to the columns of beta.
`topic`	the topic index that we want to calculate, the default is 1.
`top_n`	the number of top words to return, the default is to return all words.
`frex_weight`	the weight between 0 and 1 controlling the balance between frequency and exclusivity in the FREX metric. Weight closer to 1 is favoring exclusivity and closer to 0 is favoring frequency, we set the default as 0.5.

Value

a data frame with ranks, words, and FREX values of the words

`label_topics` *Label topics function*

Description

This function generates topic labels using four metrics: highest probability, FREX, lift, and score. For each topic, it returns the top n vocabulary terms according to each metric.

Usage

```
label_topics(beta, vocab, wordcounts, n = 8, frex_weight = 0.5)
```

Arguments

`beta`	A numeric matrix of dimension (topics x words) representing the probability distribution of words within each topic. Each row should sum to 1. Beta must be on the probability scale (not log scale).
`vocab`	a character vector of vocabulary terms corresponding to the columns of beta.
`wordcounts`	a numeric vector giving the total count of each word across the entire dataset.
`n`	the number of top words to return for each topic, the default value is 8.
`frex_weight`	the weight between 0 and 1 controlling the balance between frequency and exclusivity in the FREX metric. Weight closer to 1 is favoring exclusivity and closer to 0 is favoring frequency, we set the default as 0.5.

Details

Highest Probability: For each topic, words are ranked by their probability within that topic. The top n words with the largest probabilities are selected.

FREX: FREX is calculated by combining frequency and exclusivity for each word in each topic. Frequency is the word probabilities ranked and scaled to values between 0 and 1. Each word's probability is divided by its total probability to calculate how exclusive the word is to each topic. Then the exclusivity values are ranked within each topic and scaled to values between 0 and 1. The FREX score is the weighted harmonic mean of frequency rank and exclusivity rank, according to this formula frex<- 1 / (w / freq_rank + (1 - w) / ex_rank).

Lift: We first calculate the overall frequency of each word by dividing its total count by the total count of all words in the dataset. Then each word's probability is divided by its overall frequency.

Score: The score is computed by first taking the logarithm of the topic-word probabilities. Then calculate the average log probability across all topics for each word to represent its overall baseline level. For each topic and word, compute the difference between its log probability in that topic and its average log probability, and multiply by beta to get the final score.

Value

a list of top n vocabulary terms for each topic, ranked according to four metrics: highest probability, FREX, lift, and score.

`lift_topic` *Calculate lift ranked words and values for a topic*

Description

Calculate lift ranked words and values for a topic

Usage

```
lift_topic(beta, vocab, wordcounts, topic = 1, top_n = NULL)
```

Arguments

`beta`	A numeric matrix of dimension (topics x words) representing the probability distribution of words within each topic. Each row should sum to 1. Beta must be on the probability scale (not log scale).
`vocab`	a character vector of vocabulary terms corresponding to the columns of beta.
`wordcounts`	a numeric vector giving the total count of each word across the entire dataset.
`topic`	the topic index that we want to calculate, the default is 1.
`top_n`	the number of top words to return, the default is to return all words.

Value

a data frame with ranks, words, and lift values of the words

`lung5` *Example input data for 'SpaTopic'*

Description

multiplexed image data on tumor tissue sample from non small cell lung cancer patient

Usage

```
lung5
```

Format

‘lung5‘ A data frame with 100149 rows and 4 columns:

image Image ID

X X coordinate of the cell

Y Y coordinate of the cell

type cell type

Source

<https://nanostring.com/products/cosmx-spatial-molecular-imager/ffpe-dataset/nsclc-ffpe-dataset/>

See Also

SpaTopic_inference,Seurat5obj_to_SpaTopic

---

print.SpaTopic *Print method for SpaTopic objects*

Description

Provides a formatted summary of SpaTopic results when the object is printed. This method displays key model metrics and explains how to access different components of the model output.

Usage

```
## S3 method for class 'SpaTopic'
print(x, ...)
```

Arguments

x	An object of class "SpaTopic" returned by the SpaTopic_inference function
...	Additional arguments passed to print methods (not used)

Details

The method displays: - Number of topics identified - Model perplexity (lower is better) - DIC (Deviance Information Criterion) for model comparison - A preview of the topic distributions across cell types - Instructions on how to access full results

Value

No return value, called for side effect of printing

Examples

```
# If gibbs.res is a SpaTopic object:
# print(gibbs.res)
```

prob_topic *Calculate highest probability words and values for a topic*

Description

Calculate highest probability words and values for a topic

Usage

```
prob_topic(beta, vocab, topic = 1, top_n = NULL)
```

Arguments

beta	A numeric matrix of dimension (topics x words) representing the probability distribution of words within each topic. Each row should sum to 1. Beta must be on the probability scale (not log scale).
vocab	a character vector of vocabulary terms corresponding to the columns of beta.
topic	the topic index that we want to calculate, the default is 1.
top_n	the number of top words to return, the default is to return all words.

Value

a data frame with ranks, words, and the probabilities of the words

score_topic *Calculate score ranked words and values for a topic*

Description

Calculate score ranked words and values for a topic

Usage

```
score_topic(beta, vocab, topic = 1, top_n = NULL)
```

Arguments

beta	A numeric matrix of dimension (topics x words) representing the probability distribution of words within each topic. Each row should sum to 1. Beta must be on the probability scale (not log scale).
vocab	a character vector of vocabulary terms corresponding to the columns of beta.
topic	the topic index that we want to calculate, the default is 1.
top_n	the number of top words to return, the default is to return all words.

Value

a data frame with ranks, words, and score values of the words

Seurat5obj_to_SpaTopic *Convert a Seurat v5 object as the input of 'SpaTopic'*

Description

Prepare 'SpaTopic' input from one Seurat v5 object

Usage

```
Seurat5obj_to_SpaTopic(object, group.by, image = "image1")
```

Arguments

object	Seurat v5 object
group.by	character. The name of the column that contains celltype information in the Seurat object.
image	character. The name of the image. Default is "image1".

Value

Return a data frame as the input of 'SpaTopic'

See Also

lung5

Examples

```
## nano.obj is a Seurat v5 object
#dataset<-Seurat5obj_to_SpaTopic(object = nano.obj,
#                 group.by = "predicted.annotation.l1",image = "image1")
## Expect output
data("lung5")
```

SpaTopic-class *A class of the output from 'SpaTopic'*

Description

Outputs from function SpaTopic_inference. A list contains the following members:

- $Perplexity. The perplexity is for the training data. Let N be the total number of cells across all images. $Perplexity = exp(-loglikelihood/N)$
- $Deviance. $Deviance = -2loglikelihood$.
- $loglikelihood. The model log-likelihood.
- $loglike.trace. The log-likelihood for every collected posterior sample. NULL if trace = FALSE.
- $DIC. Deviance Information Criterion. NULL if trace = FALSE.
- $Beta. Topic content matrix with rows as celltypes and columns as topics
- $Theta. Topic prevalent matrix with rows as regions and columns as topics
- $Ndk. Number of cells per topic (col) per region (row).
- $Nwk. Number of cells per topic (col) per celltype (row).
- $Z.trace. Number of times cell being assigned to each topic across all posterior samples. We can further compute the posterior distributions of Z (topic assignment) for individual cells.
- $doc.trace. Ndk for every collected posterior sample. NULL if trace = FALSE.
- $word.trace. Nwk for every collected posterior sample. NULL if trace = FALSE.
- $cell_topics. Final topic assignments Z for individual cells.
- $parameters. Model parameters used in the analysis.

See Also

SpaTopic_inference

SpaTopic-Package *SpaTopic: Spatial Topic Modeling for Multiplexed Images*

Description

SpaTopic is a package for spatial topic modeling of Multiplexed images. It adapts an approach originally developed for image segmentation in computer vision, incorporating spatial information into the flexible design of regions (image partitions, analogous to documents in language modeling). We further refined the approach to address unique challenges in cellular images and provide an efficient C++ implementation of the algorithm in this R package.

Compared to other KNN-based methods (such as KNN-kmeans, the default neighborhood analysis in Seurat v5 R package), SpaTopic runs much faster on large-scale image datasets.

The main functions in the 'SpaTopic' package

- Prepare input Seurat5obj_to_SpaTopic
- Model Inference SpaTopic_inference
- Print results print.SpaTopic

Author(s)

Xiyu Peng <pansypeng124@gmail.com>

Source

<https://github.com/xiyupeng/SpaTopic>

See Also

Useful links:

- https://xiyupeng.github.io/SpaTopic/
- https://github.com/xiyupeng/SpaTopic
- doi:10.1038/s4146702561821y
- Report bugs at https://github.com/xiyupeng/SpaTopic/issues

SpaTopic_inference *'SpaTopic': fast topic inference to identify tissue architecture in multiplexed images*

Description

This is the main function of 'SpaTopic', implementing a Collapsed Gibbs Sampling algorithm to learn topics, which referred to different tissue microenvironments, across multiple multiplexed tissue images. The function takes cell labels and coordinates on tissue images as input, and returns the inferred topic labels for every cell, as well as topic contents, a distribution over celltypes. The function recovers spatial tissue architectures across images, as well as indicating cell-cell interactions in each domain.

Usage

```
SpaTopic_inference(
  tissue,
  ntopics,
  sigma = 50,
  region_radius = 400,
  kneigh = 5,
  npoints_selected = 1,
  ini_LDA = TRUE,
  ninit = 10,
  niter_init = 100,
```

```
  beta = 0.05,
  alpha = 0.01,
  trace = FALSE,
  seed = 123,
  thin = 20,
  burnin = 1000,
  niter = 200,
  display_progress = TRUE,
  z_cellsize = region_radius * 2,
  do.parallel = FALSE,
  n.cores = 1,
  axis = "2D"
)
```

Arguments

`tissue`	(Required). A data frame or a list of data frames. One for each image. Each row represent a cell with its image ID, X, Y coordinates on the image, celltype, with column names (image, X, Y, type), respectively. For 3D tissue images, you may add either a 'Z' column (preferred) or 'Y2' column (legacy support) for the third dimension.
`ntopics`	(Required). Number of topics. Topics will be obtained as distributions of cell types.
`sigma`	Default is 50. The lengthscale of the Nearest-neighbor Exponential Kernel. Sigma controls the strength of decay of correlation with distance in the kernel function. Please check the paper for more information. Need to be adjusted based on the image resolution
`region_radius`	Default is 400. The radius for each grid square when sampling region centers for each image. Need to be adjusted based on the image resolution and pattern complexity.
`kneigh`	Default is 5. Only consider the top 5 closest region centers for each cell.
`npoints_selected`	Default is 1. Number of points sampled for each grid square when sampling region centers for each image. Used with `region_radius`.
`ini_LDA`	Default is TRUE. Use warm start strategy for initialization and choose the best one to continue. If 0, it simply uses the first initialization.
`ninit`	Default is 10. Number of initialization. Only retain the initialization with the highest log likelihood (perplexity).
`niter_init`	Default is 100. Warm start with 100 iterations in the Gibbs sampling during initialization.
`beta`	Default is 0.05. A hyperparameter to control the sparsity of topic content (topic-celltype) matrix `Beta`. A smaller value introduces more sparse in `Beta`.
`alpha`	Default is 0.01. A hyperparameter to control the sparsity of document (region) content (region-topic) matrix `Theta`. For our application, we keep it very small for the sparsity in `Theta`.

trace	Default is FALSE. Compute and save log likelihood, Ndk, Nwk for every posterior samples. Useful when you want to use DIC to select number of topics, but it is time consuming to compute the likelihood for every posterior samples.
seed	Default is 123. Random seed.
thin	Default is 20. Key parameter in Gibbs sampling. Collect a posterior sample for every thin=20 iterations.
burnin	Default is 1000. Key parameter in Gibbs sampling. Start to collect posterior samples after 1000 iterations. You may increase the number of iterations for burn-in for highly complex tissue images.
niter	Default is 200. Key parameter in Gibbs sampling. Number of posterior samples collected for model inference.
display_progress	Default is TRUE. Display the progress bar.
z_cellsize	Default is region_radius*2. The thickness of each Z slice when performing 3D stratified sampling. Only used when axis = "3D". Controls the Z-dimension binning resolution for region center selection in 3D tissue images. Need to be adjusted based on the tissue thickness and Z-resolution.
do.parallel	Default is FALSE. Use parallel computing through R package foreach.
n.cores	Default is 1. Number of cores used in parallel computing.
axis	Default is "2D". You may switch to "3D" for 3D tissue images. However, the model inference for 3D tissue is still under test.

Value

Return a SpaTopic-class object. A list of outputs from Gibbs sampling.

See Also

SpaTopic-class

Examples

```
## tissue is a data frame containing cellular information from one image or
## multiple data frames from multiple images.

data("lung5")
## NOT RUN, it takes about 90s
library(sf)
#gibbs.res<-SpaTopic_inference(lung5, ntopics = 7,
#                              sigma = 50, region_radius = 400)


## generate a fake image 2 and make an example for multiple images
## NOT RUN
#lung6<-lung5
#lung6$image<-"image2"  ## The image ID of two images should be different
#gibbs.res<-SpaTopic_inference(list(A = lung5, B = lung6),
#                ntopics = 7, sigma = 50, region_radius = 400)
```

`spatopic_message` *Format messages for SpaTopic package*

Description

Creates consistently formatted messages for the SpaTopic package with timestamps and message type indicators. This function helps standardize all output messages across the package. Error messages will stop execution.

Usage

```
spatopic_message(type = "INFO", message, timestamp = TRUE)
```

Arguments

`type`	Character string indicating message type (e.g., "INFO", "WARNING", "ERROR", "PROGRESS")
`message`	The message content to display
`timestamp`	Logical; whether to include a timestamp in the message (default: TRUE)

Details

This function prefixes messages with a timestamp and the SpaTopic tag, creating a consistent message format throughout the package. When type="ERROR", this will stop execution with stop(). When type="WARNING", this will use warning() for non-fatal warnings. All other message types will use message() for informational output.

Value

No return value, called for side effect of displaying a message

Examples

```
## Not run:
spatopic_message("INFO", "Starting analysis...")
spatopic_message("WARNING", "Parameter out of recommended range", timestamp = FALSE)
spatopic_message("ERROR", "Required input missing") # This will stop execution
spatopic_message("PROGRESS", "Processing complete")

## End(Not run)
```

stratified_sampling_3D_via_2D *Spatially stratified random sample points from a 3D image using 2D slicing.*

Description

Spatially stratified random sample points from a 3D image by slicing along the Z-axis and applying 2D stratified sampling using R package `sf` to each slice. This approach leverages the efficiency of sf's 2D spatial indexing for 3D data.

Usage

```
stratified_sampling_3D_via_2D(
  points,
  cellsize = c(600, 600),
  z_cellsize = 600,
  num_samples_per_stratum = 1
)
```

Arguments

`points`	a data frame contains all points in a 3D image with X, Y, Z coordinates.
`cellsize`	a vector of length 2 contains the size of each grid square in X and Y dimensions. Default c(600,600).
`z_cellsize`	numeric value specifying the thickness of each Z slice. Default 600.
`num_samples_per_stratum`	number of points selected from each 3D grid cell (X,Y,Z). Default 1.

Details

This function performs 3D stratified sampling by: 1. Binning points into Z slices of thickness `z_cellsize` 2. For each Z slice, applying 2D stratified sampling using `stratified_sampling_sf` 3. Combining sampled indices from all slices

The approach is efficient for large 3D datasets as it leverages sf's optimized 2D spatial operations rather than implementing full 3D spatial indexing.

Value

Return a vector contains indices of sampled points from the original data frame.

See Also

`stratified_sampling_sf` for 2D stratified sampling

Examples

```
# Create example 3D data
set.seed(123)
tissue_3d <- data.frame(
  X = runif(1000, 0, 1000),
  Y = runif(1000, 0, 1000),
  Z = runif(1000, 0, 100)
)

# Sample points with 3D stratification
pt_idx <- stratified_sampling_3D_via_2D(tissue_3d,
                                        cellsize = c(200, 200),
                                        z_cellsize = 200)
```

`stratified_sampling_sf` *Spatially stratified random sample points from an image.*

Description

Spatially stratified random sample points from an image by R package `sf`

Usage

```
stratified_sampling_sf(
  points,
  cellsize = c(600, 600),
  num_samples_per_stratum = 1
)
```

Arguments

`points`	a data frame contains all points in a image with X, Y coordinates.
`cellsize`	a vector of length 2 contains the size of each grid square. Default c(600,600).
`num_samples_per_stratum`	number of point selected from each grid square. Default 1.

Value

Return a vector contains index of sampled points.

Examples

```
data("lung5")
pt_idx<-stratified_sampling_sf(lung5, cellsize = c(600,600))
```

Index