

Package ‘SpatialCPie’

July 10, 2025

Title Cluster analysis of Spatial Transcriptomics data

Version 1.24.0

Description SpatialCPie is an R package designed to facilitate cluster evaluation for spatial transcriptomics data by providing intuitive visualizations that display the relationships between clusters in order to guide the user during cluster identification and other downstream applications.

The package is built around a shiny ``gadget" to allow the exploration of the data with multiple plots in parallel and an interactive UI. The user can easily toggle between different cluster resolutions in order to choose the most appropriate visual cues.

biocViews Transcriptomics, Clustering, RNASeq, Software

Depends R (>= 3.6)

Imports colorspace (>= 1.3-2), data.table (>= 1.12.2), digest (>= 0.6.21), dplyr (>= 0.7.6), ggforce (>= 0.3.0), ggiraph (>= 0.5.0), ggplot2 (>= 3.0.0), ggrepel (>= 0.8.0), grid (>= 3.5.1), igraph (>= 1.2.2), lpSolve (>= 5.6.13), methods (>= 3.5.0), purrr (>= 0.2.5), readr (>= 1.1.1), rlang (>= 0.2.2), shiny (>= 1.1.0), shinycssloaders (>= 0.2.0), shinyjs (>= 1.0), shinyWidgets (>= 0.4.8), stats (>= 3.6.0), SummarizedExperiment (>= 1.10.1), tibble (>= 1.4.2), tidyr (>= 0.8.1), tidyselect (>= 0.2.4), tools (>= 3.6.0), utils (>= 3.5.0), zeallot (>= 0.1.0)

Suggests BiocStyle (>= 2.8.2), jpeg (>= 0.1-8), knitr (>= 1.20), rmarkdown (>= 1.10), testthat (>= 2.0.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 6.1.1

VignetteBuilder knitr

git_url <https://git.bioconductor.org/packages/SpatialCPie>

git_branch RELEASE_3_21

git_last_commit 8858b53
git_last_commit_date 2025-04-15
Repository Bioconductor 3.21
Date/Publication 2025-07-09
Author Joseph Bergenstraahle [aut, cre]
Maintainer Joseph Bergenstraahle <joseph.bergenstrahle@gmail.com>

Contents

SpatialCPie-package	2
.arrayPlot	3
.clusterGraph	3
.computeClusterColors	4
.likeness	5
.logsumexp	5
.makeApp	6
.makeServer	6
.makeUI	7
.maximizeOverlap	7
.preprocessData	8
.SVGBarplot	9
.tidyAssignments	9
.zscore	10
parseSpotFile	10
runCPie	11
Index	13

SpatialCPie-package	<i>SpatialCPie: Cluster analysis of Spatial Transcriptomics data</i>
---------------------	--

Description

SpatialCPie is an R package designed to facilitate cluster evaluation for spatial transcriptomics data by providing intuitive visualizations that display the relationships between clusters in order to guide the user during cluster identification and other downstream applications. The package is built around a shiny "gadget" to allow the exploration of the data with multiple plots in parallel and an interactive UI. The user can easily toggle between different cluster resolutions in order to choose the most appropriate visual cues.

Author(s)

Maintainer: Joseph Bergenstraahle <joseph.bergenstrahle@gmail.com>

.arrayPlot	Array pie plot
------------	----------------

Description

Array pie plot

Usage

```
.arrayPlot(scores, coordinates, counts = NULL, image = NULL,  
  scoreMultiplier = 1, spotScale = 1, spotOpacity = 1,  
  numTopGenes = 5)
```

Arguments

scores	data.frame with cluster scores for each spot containing the columns "spot", "name", and "score".
coordinates	data.frame with rownames matching those in scores and columns "x" and "y" specifying the plotting position of each observation.
image	a grid.grob to use as background to the plots.
scoreMultiplier	log multiplication factor applied to the score vector.
spotScale	pie chart size.
spotOpacity	pie chart opacity.

Value

[ggplot](#) object of the pie plot.

.clusterGraph	Cluster graph
---------------	---------------

Description

Cluster graph

Usage

```
.clusterGraph(assignments, clusterMeans, featureName,  
  transitionProportions = "To", transitionLabels = FALSE,  
  transitionThreshold = 0, numTopFeatures = 10)
```

Arguments

- `assignments` [data.frame](#) with columns "name", "resolution", and "cluster".
- `clusterMeans` [data.frame](#) with columns "name", "resolution", "cluster", `featureName`, and "mean".
- `featureName` [character](#) with the name of the clustered feature.
- `transitionProportions`
how to compute the transition proportions. Possible values are:
- "From": based on the total number of assignments in the lower-resolution cluster
 - "To": based on the total number of assignments in the higher-resolution cluster
- `transitionLabels`
[logical](#) specifying whether to show edge labels.
- `transitionThreshold`
hide edges with transition proportions below this threshold.
- `numTopFeatures` [integer](#) specifying the number of features to show in the hover tooltips.

Value

[ggplot](#) object of the cluster graph.

`.computeClusterColors` *Compute cluster colors*

Description

Computes colors so that dissimilar clusters are far away in color space.

Usage

```
.computeClusterColors(clusterMeans)
```

Arguments

`clusterMeans` matrix of size (n, K) representing the n feature means for each of the K clusters.

Value

vector of cluster colors.

*.likeness**Likeness score*

Description

Likeness score

Usage

```
.likeness(d, c = 1)
```

Arguments

d distance vector.
c log multiplier.

Value

vector of scores.

*.logsumexp**Logsumexp*

Description

Adapted from <https://stat.ethz.ch/pipermail/r-help/2011-February/269205.html>

Usage

```
.logsumexp(xs)
```

Arguments

xs input vector

Value

log of summed exponentials

*.makeApp**SpatialCPie App*

Description

SpatialCPie App

Usage`.makeApp(image, ...)`**Arguments**

<code>image</code>	background image.
<code>...</code>	arguments passed to <code>.preprocessData</code> .

ValueSpatialCPie `shinyApp` object.

*.makeServer**SpatialCPie server*

Description

SpatialCPie server

Usage`.makeServer(assignments, clusterMeans, counts, scores, colors, image,
coordinates, featureName)`**Arguments**

<code>assignments</code>	<code>data.frame</code> with cluster assignments containing the columns "unit" (name of the observational unit; either a gene name or a spot name), "resolution", "cluster", and "name" (a unique identifier of the (resolution, cluster) pair).
<code>clusterMeans</code>	<code>data.frame</code> with columns "name", "resolution", "cluster", <code>featureName</code> , and "mean".
<code>scores</code>	<code>data.frame</code> with cluster scores for each spot in each resolution containing the columns "spot", "resolution", "cluster", "name", and "score".
<code>colors</code>	vector of colors for each cluster. Names should match the "name" columns of the assignments and scores.
<code>image</code>	background image for the array plots, passed to <code>grid.raster</code> .
<code>coordinates</code>	<code>data.frame</code> with rownames matching the <code>names</code> in scores and columns "x" and "y" specifying the plotting position of each observation.
<code>featureName</code>	<code>character</code> with the name of the clustered feature.

`.makeUI`

7

Value

server function, to be passed to [shinyApp](#).

<code>.makeUI</code>	<i>SpatialCPie UI</i>
----------------------	-----------------------

Description

SpatialCPie UI

Usage

```
.makeUI()
```

Value

SpatialCPie UI, to be passed to [shinyApp](#).

<code>.maximizeOverlap</code>	<i>Maximize overlap</i>
-------------------------------	-------------------------

Description

Maximize overlap

Usage

```
.maximizeOverlap(xss)
```

Arguments

<code>xss</code>	list of lists of labels.
------------------	--------------------------

Value

`xss`, relabeled so as to maximize the overlap between labels in consecutive label lists.

<code>.preprocessData</code>	<i>Preprocess data</i>
------------------------------	------------------------

Description

Preprocesses input data for `.makeServer`.

Usage

```
.preprocessData(counts, margin, resolutions, assignmentFunction,
  coordinates = NULL)
```

Arguments

<code>counts</code>	count matrix. rownames should correspond to genes and colnames should correspond to spot coordinates.
<code>margin</code>	which margin of the count matrix to cluster. Valid values are <code>c("spot", "sample", "gene", "feature")</code> .
<code>resolutions</code>	vector of resolutions to cluster.
<code>assignmentFunction</code>	function to compute cluster assignments. The function should have the following signature: <code>integer (number of clusters) -> (m, n) feature matrix -> m-length vector (cluster assignment of each data point)</code> .
<code>coordinates</code>	optional <code>data.frame</code> with pixel coordinates for each spot. rownames should correspond to the colnames of counts and the columns <code>x</code> and <code>y</code> should specify the pixel coordinates of the spots.

Value

list with the following elements:

- `$assignments`: tidy assignments
- `$means`: cluster means
- `$scores`: cluster scores for each spot in each resolution
- `$colors`: cluster colors
- `$coordinates`: spot coordinates, either from `coordinates` or parsed from `assignments`
- `$featureName`: name of the clustered feature (the "opposite" of `margin`)

<code>.SVGBarplot</code>	<i>SVG barplot</i>
--------------------------	--------------------

Description

SVG barplot

Usage

```
.SVGBarplot(xs)
```

Arguments

`xs` named vector with observations

Value

`character` SVG barplot

<code>.tidyAssignments</code>	<i>Tidy assignments</i>
-------------------------------	-------------------------

Description

Tidy assignments

Usage

```
.tidyAssignments(assignments)
```

Arguments

`assignments` list of assignment vectors.

Value

a `data.frame` containing the assignments, with the data relabeled so that the overlap between consecutive assignment vectors is maximized. Additionally, a "root" resolution is added.

<code>.zscore</code>	<i>Z-score</i>
----------------------	----------------

Description

Z-score

Usage

```
.zscore(xs)
```

Arguments

<code>xs</code>	vector of observations
-----------------	------------------------

Value

`xs`, z-normalized. if all elements of `xs` are equal, a vector of zeros will be returned instead.

<code>parseSpotFile</code>	<i>Parse spot detector output</i>
----------------------------	-----------------------------------

Description

Parses the output from the ST spot detector tool for use with SpatialCPie.

Usage

```
parseSpotFile(file)
```

Arguments

<code>file</code>	spot file
-------------------	-----------

Value

`data.frame` with columns "x" and "y" specifying the pixel coordinates of each spot

Examples

```
## Create spot file
data <- rbind(
  c(7, 18, 7.00, 18.07, 563.2, 947.0),
  c(8, 11, 8.00, 11.04, 612.5, 627.7)
)
filename <- tempfile()
write.table(
  data,
  file = filename,
  sep = "\t",
  quote = FALSE,
  col.names = c("x", "y", "new_x", "new_y", "pixel_x", "pixel_y")
)

## Parse spot file
parseSpotFile(filename)

## Delete spot file
unlink(filename)
```

runCPie

Run SpatialCPie

Description

Runs the SpatialCPie gadget.

Usage

```
runCPie(counts, image = NULL, spotCoordinates = NULL,
  margin = "spot", resolutions = 2:4,
  assignmentFunction = function(k, x) kmeans(x, centers = k)$cluster,
  view = NULL)
```

Arguments

counts	gene count matrix or a SummarizedExperiment-class object containing count values.
image	image to be used as background to the plot.
spotCoordinates	data.frame with pixel coordinates. The rows should correspond to the columns (spatial areas) in the count file.
margin	which margin to cluster.
resolutions	numeric vector specifying the clustering resolutions.
assignmentFunction	function to compute cluster assignments.
view	viewer object.

Value

a list with the following items:

- "clusters": Cluster assignments (may differ from assignments)
- "clusterGraph": The cluster tree ggplot object
- "arrayPlot": The pie plot ggplot objects

Examples

```
if (interactive()) {
  options(device.ask.default = FALSE)

  ## Set up coordinate system
  coordinates <- as.matrix(expand.grid(1:10, 1:10))

  ## Generate data set with three distinct genes generated by three
  ## distinct cell types
  profiles <- diag(rep(1, 3)) + runif(9)
  centers <- cbind(c(5, 2), c(2, 8), c(8, 2))
  mixes <- apply(coordinates, 1, function(x) {
    x <- exp(-colSums((centers - x) ^ 2) / 50)
    x / sum(x)
  })
  means <- 100 * profiles %*% mixes
  counts <- matrix(rpois(prod(dim(means)), means), nrow = nrow(profiles))
  colnames(counts) <- apply(
    coordinates,
    1,
    function(x) do.call(paste, c(as.list(x), list(sep = "x"))))
  )
  rownames(counts) <- paste("gene", 1:nrow(counts))

  ## Run SpatialCPie
  runCPie(counts)
}
```

Index

* **internal**

- .SVGBarplot, [9](#)
- .arrayPlot, [3](#)
- .clusterGraph, [3](#)
- .computeClusterColors, [4](#)
- .likeness, [5](#)
- .logsumexp, [5](#)
- .makeApp, [6](#)
- .makeServer, [6](#)
- .makeUI, [7](#)
- .maximizeOverlap, [7](#)
- .preprocessData, [8](#)
- .tidyAssignments, [9](#)
- .zscore, [10](#)

- .SVGBarplot, [9](#)
- .arrayPlot, [3](#)
- .clusterGraph, [3](#)
- .computeClusterColors, [4](#)
- .likeness, [5](#)
- .logsumexp, [5](#)
- .makeApp, [6](#)
- .makeServer, [6, 8](#)
- .makeUI, [7](#)
- .maximizeOverlap, [7](#)
- .preprocessData, [6, 8](#)
- .tidyAssignments, [9](#)
- .zscore, [10](#)

character, [4, 6, 9](#)

data.frame, [3, 4, 6, 8–11](#)

ggplot, [3, 4](#)

grid.grob, [3](#)

grid.raster, [6](#)

integer, [4](#)

logical, [4](#)

names, [6](#)

numeric, [11](#)

parseSpotFile, [10](#)

runCPie, [11](#)

shinyApp, [6, 7](#)

SpatialCPie (SpatialCPie-package), [2](#)

SpatialCPie-package, [2](#)

viewer, [11](#)