Package 'SpaceMarkers'

April 4, 2025

Type Package

Title Spatial Interaction Markers

Version 1.3.5

BugReports https://github.com/DeshpandeLab/SpaceMarkers/issues

URL https://github.com/DeshpandeLab/SpaceMarkers

Description Spatial transcriptomic technologies have helped to resolve the connection between gene expression and the 2D orientation of tissues relative to each other. However, the limited single-cell resolution makes it difficult to highlight the most important molecular interactions in these tissues. SpaceMarkers, R/Bioconductor software, can help to find molecular interactions, by identifying genes associated with latent space interactions in spatial transcriptomics.

Depends R (>= 4.4.0)

biocViews SingleCell, GeneExpression, Software, Spatial, Transcriptomics

Imports matrixStats, matrixTests, rstatix, spatstat.explore, spatstat.geom, ape, hdf5r, nanoparquet, jsonlite, Matrix, qvalue, stats, utils, methods, ggplot2, reshape2

Suggests data.table, devtools, dplyr, hrbrthemes, knitr, RColorBrewer, cowplot, readbitmap, rjson, rmarkdown, BiocStyle, testthat (>= 3.0.0), viridis, CoGAPS

Enhances BiocParallel

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

License MIT + file LICENSE

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

git_branch devel

git_last_commit d33ec2f

git_last_commit_date 2025-03-17

Repository Bioconductor 3.21

Date/Publication 2025-04-03

Author Atul Deshpande [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5144-6924>),
Ludmila Danilova [ctb],
Dmitrijs Lvovs [ctb] (ORCID: <https://orcid.org/0009-0003-2152-6853>)

Maintainer Atul Deshpande <adeshpande@jhu.edu>

Contents

`.getBTMEfeatures` *.getBTMEfeatures Load features BayesTME object*

Description

.getBTMEfeatures Load features BayesTME object

Usage

```
.getBTMEfeatures(hf)
```

.getCogapsFeatures *.getCogapsFeatures Load features CoGAPS object*

Description

.getCogapsFeatures Load features CoGAPS object

Usage

```
.getCogapsFeatures(obj)
```

.getCSVfeatures *.getCSVFeatures Load features from dataframe*

Description

.getCSVFeatures Load features from dataframe

Usage

```
.getCSVfeatures(obj)
```

.getSeuratFeatures *.getSeuratFeatures Load features Seurat object*

Description

.getSeuratFeatures Load features Seurat object

Usage

```
.getSeuratFeatures(obj)
```

.inferMethod *inferMethod Infer the method used to obtain spatial features*

Description

inferMethod Infer the method used to obtain spatial features

Usage

```
.inferMethod(spObject, method)
```

.readFormat *readFormat Reads a format into an R object*

Description

readFormat Reads a format into an R object

Usage

```
.readFormat(path)
```

curated_genes *Curated Genes for example purposes*

Description

A vector with genes selected based on previous runs of SpaceMarkers on the Visium 10x breast ductal carcinoma spatial transcriptomics dataset

Format

A vector with 114 pre-selected genes

Value

a vector of genes

`findAllHotspots` *Find hotSpots for all spatial patterns*

Description

Convenience function to find hotspots for all spatial patterns

Usage

```
findAllHotspots(
  spPatterns,
  params = NULL,
  outlier = "positive",
  nullSamples = 1000,
  includeSelf = TRUE,
  ...
)
```

Arguments

`spPatterns`	A data frame that contains the spatial coordinates and metrics for spatial features (cell types/cell processes). The column names must include 'x' and 'y' as well as the spatially varying features.
`params`	a named vector of the optimal sigma and threshold for a given spatial pattern. The names are should be 'sigmaOpt' and 'threshOpt'. The default value is NULL.
`outlier`	a character string specifying whether to apply the outlier threshold to the kernel density distribution in a one-sided manner (specify 'positive' the default) or in a two sided manner (specify 'two.sided').
`nullSamples`	a numeric values specifying the number of spatial patterns to randomly sample for a null distribution.
`includeSelf`	a logic value specifying whether to consider the spatial influence the pattern has on surrounding regions only (set to FALSE), or whether to also consider the influence of the pattern itself (set to TRUE , the default).
`...`	Arguments passed to methods

`findGenesOfInterest` *findGenesOfInterest Identify genes associated with pattern interaction. This function identifies genes exhibiting significantly higher values of testMat in the Interaction region of the two patterns compared to regions with exclusive influence from either pattern. It uses Kruskal-Wallis test followed by posthoc analysis using Dunn's Test to identify the genes.*

Description

findGenesOfInterest Identify genes associated with pattern interaction. This function identifies genes exhibiting significantly higher values of testMat in the Interaction region of the two patterns compared to regions with exclusive influence from either pattern. It uses Kruskal-Wallis test followed by posthoc analysis using Dunn's Test to identify the genes.

Usage

```
findGenesOfInterest(testMat, goodGenes, region, fdr.level=0.05,
       analysis=c("enrichment","overlap"),...)
```

Arguments

testMat	A matrix of counts with cells as columns and genes as rows
goodGenes	A vector of user specified genes expected to interact a priori. The default for this is NULL as the function can find these genes itself
region	A data frame of the reference pattern regions that overlap with the other patterns
fdr.level	False Discovery Rate. The default value is 0.05.
analysis	a character string that specifies the type of analysis to carry out, whether overlap or enrichment.
...	Additional arguments to be passed to lower level functions

Value

a list of genes exhibiting significantly higher values of testMat in the Interaction region of the two #' patterns compared to regions with exclusive influence from either pattern.

---

findPatternHotspots *Identify hotspots of spatial pattern influence*

Description

This function calculates 'hotspots' which are regions of high spatial influence based on an outlier threshold from a null distribution.

Usage

```
findPatternHotspots(
  spPatterns,
  params = NULL,
  patternName = "Pattern_1",
  outlier = "positive",
  nullSamples = 1000,
  includeSelf = TRUE,
  ...
)
```

Arguments

spPatterns	A data frame that contains the spatial coordinates and metrics for spatial features (cell types/cell processes). The column names must include 'x' and 'y' as well as the spatially varying features.
params	a named vector of the optimal sigma and threshold for a given spatial pattern. The names are should be 'sigmaOpt' and 'threshOpt'. The default value is NULL.
patternName	a character string that specifies the pattern of interest
outlier	a character string specifying whether to apply the outlier threshold to the kernel density distribution in a one-sided manner (specify 'positive' the default) or in a two sided manner (specify 'two.sided').
nullSamples	a numeric values specifying the number of spatial patterns to randomly sample for a null distribution.
includeSelf	a logic value specifying whether to consider the spatial influence the pattern has on surrounding regions only (set to FALSE), or whether to also consider the influence of the pattern itself (set to TRUE , the default).
...	Arguments passed to methods

Value

a character vector with the spatial feature name if the spatial influence exceeded the threshold for that spot/cell, and NA otherwise

See Also

Other getIntGenes: `getInteractingGenes()`, `getPairwiseInteractingGenes()`

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
                             package="SpaceMarkers",mustWork = TRUE))
sp_url <- urls[["visium_url"]][2]
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
#Obtaining CoGAPS Patterns i.e Spatial Features
cogaps_result <- readRDS(system.file("extdata","CoGAPS_result.rds",
                                     package="SpaceMarkers",mustWork = TRUE))
spFeatures <- slot(cogaps_result,"sampleFactors")
#Obtaining Spatial Coordinates
download.file(sp_url, basename(sp_url), mode = "wb")
untar(basename(sp_url))
spCoords <- load10XCoords(visiumDir = ".", version = "1.0")
rownames(spCoords) <- spCoords$barcode
#Match Dimensions
barcodes <- intersect(rownames(spFeatures),spCoords$barcode)
```

```
spCoords <- spCoords[barcodes,]
spFeatures <- spFeatures[barcodes,]
spPatterns <- cbind(spCoords,spFeatures[barcodes,])
spPatterns<-spPatterns[c("barcode","y","x","Pattern_1","Pattern_5")]
data("optParams")
hotspots <- findPatternHotspots(
spPatterns = spPatterns,
patternName = "Pattern_1",
params = optParams[,"Pattern_1"],
outlier = "positive",nullSamples = 1000,includeSelf = TRUE)
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
```

`getIMScores` *getIMScores*

Description

Get the interaction scores for SpaceMarkers

Usage

```
getIMScores(SpaceMarkers)
```

Arguments

`SpaceMarkers`	A list of SpaceMarkers objects

Value

A data frame with columns Gene and SpaceMarkersMetric

Examples

```
example(getPairwiseInteractingGenes)
getIMScores(SpaceMarkers)
```

getInteractingGenes *Calculate Interaction Regions and Associated Genes*

Description

This function calculates statistically significant genes using a non-parametric Kruskal-Wallis test for genes in any one region of influence and a post hoc Dunn's test is used for analysis of genes between regions.

Usage

```
getInteractingGenes(
  data,
  spPatterns,
  refPattern = "Pattern_1",
  mode = c("DE", "residual"),
  optParams = NULL,
  reconstruction = NULL,
  hotspots = NULL,
  analysis = c("enrichment", "overlap"),
  minOverlap = 50,
  ...
)
```

Arguments

data	original spatial data matrix.
spPatterns	A data frame that contains the spatial coordinates and metrics for spatial features (cell types/cell processes). The column names must include 'x' and 'y' as well as the spatially varying features.
refPattern	a character string that specifies the pattern whose "interaction" with every other pattern we want to study. The default value is "Pattern_1".
mode	SpaceMarkers mode of operation. Possible values are "DE" (the default) or "residual".
optParams	a matrix with dimensions 2 X N, where N is the number of spatial patterns with optimal parameters. The first row contains the kernel width 'sigmaOpt' for each pattern, and the second row is the threshOpt (outlier threshold) for each pattern. Users can also input their preferred param values. The default value is NULL.
reconstruction	reconstruction of the data matrix from latent spaces. Required for "residual" mode.
hotspots	a vector that specifies the patterns to compare to the 'refPattern'. The default is NULL which indicates that all patterns would be compared to the 'refPattern'.
analysis	a character string that specifies the type of downstream analysis to be performed. Possible values are "enrichment" (default) and "overlap". In enrichment mode, all genes are returned, ranked by the SpaceMarkers metric. In overlap mode,

only the genes which are significantly overexpressed in the interaction region are returned.

`minOverlap` a number that specifies the minimum overlap between genes in two patterns to be considered for the statistical tests. The default is 50.

`...` Arguments passed to methods

Value

a list of data frames with information about the interacting genes of the refPattern and each latent feature pattern matrix (interacting_genes object). There is also a data frame with all of the regions of influence for any two of patterns (the hotspots object).

See Also

Other getIntGenes: `findPatternHotspots()`, `getPairwiseInteractingGenes()`

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
package="SpaceMarkers",mustWork = TRUE))
counts_url <- urls[["visium_url"]][1]
sp_url <- urls[["visium_url"]][2]
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
download.file(counts_url,basename(counts_url), mode = "wb")
counts_matrix<-load10XExpr(visiumDir=".",h5filename = basename(counts_url))
#Obtaining CoGAPS Patterns
cogaps_result <- readRDS(system.file("extdata","CoGAPS_result.rds",
package="SpaceMarkers",mustWork = TRUE))
features <- intersect(rownames(counts_matrix),rownames(
    slot(cogaps_result,"featureLoadings")))
barcodes <- intersect(colnames(counts_matrix),rownames(
    slot(cogaps_result,"sampleFactors")))
counts_matrix <- counts_matrix[features,barcodes]
cogaps_matrix <- slot(cogaps_result,"featureLoadings")[features,]%*%
    t(slot(cogaps_result,"sampleFactors")[barcodes,])
#Obtaining Spatial Coordinates
download.file(sp_url, basename(sp_url), mode = "wb")
untar(basename(sp_url))
spCoords <- load10XCoords(visiumDir = ".", version = "1.0")
rownames(spCoords) <- spCoords$barcode
spCoords <- spCoords[barcodes,]
spPatterns <- cbind(spCoords,slot(cogaps_result,
"sampleFactors")[barcodes,])
data("curated_genes")
spPatterns<-spPatterns[c("barcode","y","x","Pattern_1","Pattern_5")]
```

```
counts_matrix <- counts_matrix[curated_genes,]
cogaps_matrix <- cogaps_matrix[curated_genes, ]
data("optParams")
SpaceMarkersMode <- "DE"
ref_Pattern <- "Pattern_1"
SpaceMarkers_test <- getInteractingGenes(
    data=counts_matrix,reconstruction=NULL,
    optParams = optParams,
    spPatterns = spPatterns,
    refPattern = "Pattern_1",
    mode="DE",analysis="overlap")
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
```

`getOverlapScores` *getOverlapScores*

Description

Calculate the overlap scores between patterns in hotspots

Usage

```
getOverlapScores(hotspots, patternList = NULL)
```

Arguments

`hotspots`	A data frame with columns x, y, barcode and pattern names
`patternList`	A character vector of pattern names to calculate overlap scores for

Value

A data frame with columns pattern1, pattern2 and overlapScore

Examples

```
hotspots <- data.frame(x = c(1,2,3,4,5),
                       y = c(1,2,3,4,5),
                       barcode = c("A","B","C","D","E"),
                       pattern1 = c(1,0,1,0,1),
                       pattern2 = c(1,1,0,0,1))
getOverlapScores(hotspots)
getOverlapScores(hotspots, c("pattern1","pattern2"))
```

getPairwiseInteractingGenes *getPairwiseInteractingGenes*

Description

Performs pairwise analysis to find genes associated with spatial interaction between pairs of spatially varying patterns.

Usage

```
getPairwiseInteractingGenes(
  data,
  spPatterns,
  mode = c("DE", "residual"),
  optParams = NULL,
  reconstruction = NULL,
  hotspots = NULL,
  minOverlap = 50,
  analysis = c("enrichment", "overlap"),
  patternPairs = NULL,
  ...,
  workers = NULL
)
```

Arguments

data	original spatial data matrix.
spPatterns	A data frame that contains the spatial coordinates and metrics for spatial features (cell types/cell processes). The column names must include 'x' and 'y' as well as the spatially varying features.
mode	SpaceMarkers mode of operation. Possible values are "DE" (the default) or "residual".
optParams	a matrix with dimensions 2 X N, where N is the number of spatial patterns with optimal parameters. The first row contains the kernel width 'sigmaOpt' for each pattern, and the second row is the threshOpt (outlier threshold) for each pattern. Users can also input their preferred param values. The default value is NULL.
reconstruction	reconstruction of the data matrix from latent spaces. Required for "residual" mode.
hotspots	a vector that specifies the patterns to compare to the 'refPattern'. The default is NULL which indicates that all patterns would be compared to the 'refPattern'.
minOverlap	a number that specifies the minimum overlap between genes in two patterns to be considered for the statistical tests. The default is 50.

analysis	a character string that specifies the type of downstream analysis to be performed. Possible values are "enrichment" (default) and "overlap". In enrichment mode, all genes are returned, ranked by the SpaceMarkers metric. In overlap mode, only the genes which are significantly overexpressed in the interaction region are returned.
patternPairs	A matrix of pattern pairs to be analyzed. Default is
...	Arguments passed to methods
workers	(optional) Number of workers to be used for parallel processing.

Details

==========

Value

a list of data frames for each pattern with 1) names of the patterns (patterns object) 2) data frame with the hotspots of influence for the two patterns (the hotspots object). 3) data frame with the genes associated with the interaction between the two patterns (interacting genes object, empty if insufficient interaction).

See Also

Other getIntGenes: `findPatternHotspots()`, `getInteractingGenes()`

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
package="SpaceMarkers",mustWork = TRUE))
counts_url <- urls[["visium_url"]][1]
sp_url <- urls[["visium_url"]][2]
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
download.file(counts_url,basename(counts_url), mode = "wb")
counts_matrix<-load10XExpr(visiumDir=".",
h5filename = basename(counts_url))
#Obtaining CoGAPS Patterns
cogaps_result <- readRDS(system.file("extdata","CoGAPS_result.rds",
package="SpaceMarkers",mustWork = TRUE))
features <- intersect(rownames(counts_matrix),rownames(
    slot(cogaps_result,"featureLoadings")))
barcodes <- intersect(colnames(counts_matrix),rownames(
    slot(cogaps_result,"sampleFactors")))
counts_matrix <- counts_matrix[features,barcodes]
cogaps_matrix <- slot(cogaps_result,"featureLoadings")[features,]%*%
    t(slot(cogaps_result,"sampleFactors")[barcodes,])
#Obtaining Spatial Coordinates
```

```
download.file(sp_url, basename(sp_url), mode = "wb")
untar(basename(sp_url))
spCoords <- load10XCoords(visiumDir = ".", version = "1.0")
rownames(spCoords) <- spCoords$barcode
spCoords <- spCoords[barcodes,]
spPatterns <- cbind(spCoords,
slot(cogaps_result,"sampleFactors")[barcodes,])
data("curated_genes")
spPatterns<-spPatterns[c("barcode","y","x","Pattern_1",
"Pattern_3","Pattern_5")]
counts_matrix <- counts_matrix[curated_genes,]
cogaps_matrix <- cogaps_matrix[curated_genes, ]
optParams <- matrix(c(6, 2, 6, 2, 6, 2), nrow = 2)
rownames(optParams) <- c("sigmaOpt","threshOpt")
colnames(optParams) <- c("Pattern_1","Pattern_3","Pattern_5")
SpaceMarkersMode <- "DE"
patternPairs <- matrix(c("Pattern_1", "Pattern_1",
           "Pattern_3", "Pattern_5"), nrow=2)
SpaceMarkers <- getPairwiseInteractingGenes(
    data=counts_matrix,reconstruction=NULL,
    optParams = optParams,
    spPatterns = spPatterns,
    mode="DE",analysis="enrichment", patternPairs=patternPairs)
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
```

`getSpatialFeatures` *Load spatial features*

Description

This function loads spatial features from a file containing spatial features

Usage

```
getSpatialFeatures(filePath, method = NULL, featureNames = ".")
```

Arguments

`filePath`	A string path to the location of the file containing the spatial features.
`method`	A string specifying the type of object to obtain spatial feature from. Default NULL, where the method is inferred based on object type. Other methods are: "CoGAPS", "Seurat", or "BayesTME".
`featureNames`	An array of strings specifying the column names corresponding to the feature names or a regex string. In the case of Seurat, all metadata columns with "_Feature" suffix are selected.

Value

a matrix of spatial features with barcodes associated with individual coordinates

Examples

```
library(SpaceMarkers)
#CoGAPS data filePath
filePath <- system.file("extdata","CoGAPS_result.rds",
package = "SpaceMarkers",mustWork = TRUE)
spFeatures <- getSpatialFeatures(filePath, method = "CoGAPS")
head(spFeatures)
```

---

getSpatialParameters *Read optimal parameters for spatial kernel density from user input or .json file*

Description

This function obtains the width of a spatial kernel density (sigma) from either the user input or from a scale factors .json file. The outlier threshold around the set of spots (threshold) for each pattern is specified by the user (default is 4).

Usage

```
getSpatialParameters(
  spatialPatterns,
  visiumDir = ".",
  spatialDir = "spatial",
  pattern = "scalefactors_json.json",
  sigma = NULL,
  threshold = 4,
  resolution = c("lowres", "hires", "fullres"),
  ...
)
```

Arguments

spatialPatterns	A data frame that contains the spatial coordinates for each cell type. The column names must include 'x' and 'y' as well as a set of numbered columns named 'Pattern_1.....N'.
visiumDir	A string path specifying the location of the 10xVisium directory
spatialDir	A string path specifying the location of the spatial folder containing the .json file of the spot characteristics
pattern	A string specifying the name of the .json file

sigma	A numeric value specifying the width of the kernel density estimate to be used for smoothing
threshold	A numeric value specifying how many standard deviations above the mean of a null distribution to use an outlier threshold for identifying 'hotspots'
resolution	A string specifying image resolution
...	Arguments passed to methods

Value

a numeric matrix of sigmaOpts - the optimal width of the gaussian distribution, and the threshOpt - outlier threshold around the set of spots for each pattern

Examples

```
library(SpaceMarkers)
# Create test data
cells <- c()
test_num <- 500
for(i in 1:test_num){
    cells[length(cells)+1] <- paste0("cell_",i)
}
spPatterns <- data.frame(barcode = cells,
y = runif(test_num, min=0, max=test_num),
x = runif(test_num, min=0, max=test_num),
Pattern_1 = runif(test_num, min=0, max=1),
Pattern_2 = runif(test_num, min=0, max=1) )
# Call the getSpatialParameters function with the test data
optParams <- getSpatialParameters(spPatterns, sigma = 10)
```

---

getSpatialParamsMoransI

Calculate the optimal parameters from spatial kernel density for cell-cell interactions

---

Description

This function uses Morans.I to calculate the optimal width of the kernel density (sigmaOpt) as well as the outlier threshold around the set of spots (threshOpt) for a null distribution.

Usage

```
getSpatialParamsMoransI(spatialPatterns, ...)
```

Arguments

spatialPatterns	A data frame that contains the spatial coordinates for each cell type. The column names must include 'x' and 'y' as well as a set of numbered columns named 'Pattern_1.....N'.
...	Arguments passed to methods

Value

a numeric matrix of sigmaOpts - the optimal width of the gaussian distribution, and the threshOpt - outlier threshold around the set of spots for each pattern

Examples

```
library(SpaceMarkers)
# Create test data
cells <- c()
test_num <- 500
for(i in 1:test_num){
    cells[length(cells)+1] <- paste0("cell_",i)
}
spPatterns <- data.frame(barcode = cells,
y = runif(test_num, min=0, max=test_num),
x = runif(test_num, min=0, max=test_num),
Pattern_1 = runif(test_num, min=0, max=1),
Pattern_2 = runif(test_num, min=0, max=1) )
# Call the getSpatialParamsMoransI function with the test data
optParams <- getSpatialParamsMoransI(spPatterns)
```

---

load10XCoords *Load 10x Visium Spatial Coordinates*

Description

This function loads spatial coordinates for each cell from a 10X Visium spatial folder.

Usage

```
load10XCoords(visiumDir, resolution = "lowres", version = NULL)
```

Arguments

visiumDir	A string path to the location of the folder containing the spatial coordinates. The folder in your visiumDir must be named 'spatial' and must contain files 'scalefactors_json.json' and 'tissue_positions_list.csv.'
resolution	A string specifying which values to look for in the .json object. Can be either lowres or highres.
version	A string specifying the version of the spaceranger data.

Value

a data frame of the spatial coordinates (x and y) for each spot/cell

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
package = "SpaceMarkers",mustWork = TRUE))
sp_url <- urls[["visium_url"]][2]
# Spatial Coordinates
download.file(sp_url, basename(sp_url), mode = "wb")
untar(basename(sp_url))
spCoords <- load10XCoords(visiumDir = ".", version = "1.0")
unlink("spatial", recursive = TRUE)
unlink("Visium_Human_Breast_Cancer_spatial.tar.gz")
```

---

load10XExpr *Load 10X Visium Expression Data*

Description

This loads log-transformed 10X Visium expression data from standard 10X Visium folder.

Usage

```
load10XExpr(visiumDir = NULL, h5filename = "filtered_feature_bc_matrix.h5")
```

Arguments

visiumDir	A string path to the h5 file with expression information.
h5filename	A string of the name of the h5 file in the directory.

Value

A matrix of class dgeMatrix or Matrix that contains the expression info for each sample (cells) across multiple features (genes)

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
package = "SpaceMarkers",mustWork = TRUE))
counts_url <- urls[["visium_url"]][1]
#Remove present Directories if any
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
```

```
download.file(counts_url,basename(counts_url), mode = "wb")
counts_matrix<-load10XExpr(visiumDir=".",h5filename = basename(counts_url))
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
```

`optParams` *Optimal paramters of 5 patterns from CoGAPS.*

Description

A dataset with the optimal width of the gaussian distribution (sigmaOpt) and the outlier threshold around the set of spots (thresOpt) for each pattern obtained from CoGAPS. CoGAPS was ran on spatial transcriptomic data from a breast cancer sample.

Format

A data frame with 2 rows and 5 columns:

Pattern_1 immune cell pattern paramters

Pattern_2 Disp.1 parameters

Pattern_3 intraductal carcinoma (DCIS) parameters

Pattern_4 Disp.2 parameters

Pattern_5 invasive carcinoma lesion pattern paramters

Value

A matrix of optimal parameters for patterns identified by CoGAPS

`plotIMScores` *plotIMScores*

Description

Plot the top SpaceMarkers IMScores

Usage

```
plotIMScores(
  df,
  interaction,
  cutOff = 0,
  nGenes = 20,
  geneText = 12,
  metricText = 12,
  increments = 1,
  out = NULL
)
```

Arguments

df	A data frame with columns Gene and SpaceMarkersMetric
interaction	The interaction to plot
cutOff	The cut off value for the plot
nGenes	The number of genes to plot
geneText	The font size for the gene text
metricText	The font size for the metric text
increments	The increments for the y-axis
out	The output path for the plot

Examples

```
example(getPairwiseInteractingGenes)
plotIMScores(getIMScores(SpaceMarkers), "Pattern_1_Pattern_3")
```

---

plotOverlapScores *plotOverlapScores*

Description

Plot the overlap scores between patterns in hotspots

Usage

```
plotOverlapScores(
  df,
  title = "Spatial Overlap Scores",
  out = NULL,
  fontsize = 15
)
```

Arguments

df	A data frame with columns pattern1, pattern2 and overlapScore
title	The title of the plot
out	The output path for the plot
fontsize	The font size of the plot

Value

A ggplot object

Examples

```
df <- data.frame(pattern1 = c("pattern1","pattern1","pattern2","pattern2"), pattern2 = c("pattern1","pattern2","p
plotOverlapScores(df)
plotOverlapScores(df, "Overlap Scores", "overlapScores.png", 15)
```

Index