Package ‘spicyR’

January 16, 2024

Type  Package
Title  Spatial analysis of in situ cytometry data
Version  1.14.3
Description  The spicyR package provides a framework for performing inference on changes in spa-
tial relationships between pairs of cell types for cell-resolution spatial omics technolo-
gies. spicyR consists of three primary steps: (i) summarizing the degree of spatial localiza-
tion between pairs of cell types for each image; (ii) modelling the variability in localization sum-
mary statistics as a function of cell counts and (iii) testing for changes in spatial localizations as-
sociated with a response variable.
License  GPL (>=2)
LazyData  true
biocViews  SingleCell, CellBasedAssays, Spatial
Encoding  UTF-8
Depends  R (>= 4.1)
VignetteBuilder  knitr

BugReports  https://github.com/SydneyBioX/spicyR/issues
URL  https://ellispatrick.github.io/spicyR/
     https://github.com/SydneyBioX/spicyR
Imports  ggplot2, concaveman, BiocParallel, spatstat.explore,
         spatstat.geom, lmerTest, BiocGenerics, S4Vectors, methods,
         mgcv, pheatmap, rlang, grDevices, IRanges, stats, data.table,
         dplyr, tidyr, scam, SingleCellExperiment, SpatialExperiment,
         SummarizedExperiment, ggforce, ClassifyR, tibble
Suggests  BiocStyle, knitr, rmarkdown, pkgdown, imcRtools
RoxygenNote  7.2.3

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

git_branch  RELEASE_3_18

git_last_commit  79d608c

git_last_commit_date  2024-01-08
Accessors

Repository  Bioconductor 3.18
Date/Publication  2024-01-15
Author  Nicolas Canete [aut],
        Ellis Patrick [aut, cre]
Maintainer  Ellis Patrick <ellis.patrick@sydney.edu.au>

R topics documented:

Accessors  ................................................. 2
as.data.frame.SegmentedCells  .................................. 4
bind  ....................................................... 5
colTest  ..................................................... 6
covPairs  ....................................................... 6
diabetesData  ................................................... 7
diabetesData_SCE  ............................................ 8
getPairwise  .................................................... 8
getProp  ....................................................... 10
plot.SegmentedCells,ANY-method  ................................ 10
SegmentedCells-class  .......................................... 11
show-SegmentedCells  .......................................... 13
signifPlot  .................................................... 14
spicyBoxPlot  .................................................. 15
SpicyResults-class  ............................................ 16
spicyTest  ...................................................... 18
topPairs  ...................................................... 18

Index  ................................................. 20

Accessors  Accessors for SegmentedCells

Description

Methods to access various components of the ‘SegmentedCells’ object.

Usage

cellSummary(x, imageID = NULL, bind = TRUE)

cellSummary(x, imageID = NULL) <- value

cellMarks(x, imageID = NULL, bind = TRUE)

cellMarks(x, imageID = NULL) <- value

cellMorph(x, imageID = NULL, bind = TRUE)
```r

Accessors

`cellMorph(x, imageID = NULL) <- value`

`imagePheno(x, imageID = NULL, bind = TRUE, expand = FALSE)`

`imagePheno(x, imageID = NULL) <- value`

`imageID(x, imageID = NULL)`

`cellID(x, imageID = NULL)`

`cellID(x) <- value`

`imageCellID(x, imageID = NULL)`

`imageCellID(x) <- value`

`cellType(x, imageID = NULL)`

`cellType(x, imageID = NULL) <- value`

`filterCells(x, select)`

`cellAnnotation(x, variable, imageID = NULL)`

`cellAnnotation(x, variable, imageID = NULL) <- value`

Arguments

- **x**: A `SegmentedCells` object.
- **imageID**: A vector of imageIDs to specifically extract.
- **bind**: When false outputs a list of DataFrames split by imageID.
- **expand**: Used to expand the phenotype information from per image to per cell.
- **value**: The relevant information used to replace.
- **select**: A logical vector of the cells to be kept.
- **variable**: A variable to add or retrieve from `cellSummary`.

Value

DataFrame or a list of DataFrames

Descriptions

- **`cellSummary`**: Retrieves the DataFrame containing ‘x’ and ‘y’ coordinates of each cell as well as ‘cellID’, ‘imageID’ and ‘cellType’. imageID can be used to select specific images and bind=FALSE outputs the information as a list split by imageID.

- **`cellMorph`**: Retrieves the DataFrame containing morphology information.
`cellMarks`: Retrieves the DataFrame containing intensity of gene or protein markers.

`imagePheno`: Retrieves the DataFrame containing the phenotype information for each image. Using `expand = TRUE` will produce a DataFrame with the number of rows equal to the number of cells.

**Examples**

```r
### Something that resembles cellProfiler data
set.seed(51773)

n = 10

cells <- data.frame(row.names = seq_len(n))
cells$ObjectNumber <- seq_len(n)
cells$ImageNumber <- rep(1:2, c(n/2, n/2))
cells$AreaShape_Center_X <- runif(n)
cells$AreaShape_Center_Y <- runif(n)
cells$AreaShape_round <- rexp(n)
cells$AreaShape_diameter <- rexp(n, 2)
cells$Intensity_Mean_CD8 <- rexp(n, 10)
cells$Intensity_Mean_CD4 <- rexp(n, 10)

cellExp <- SegmentedCells(cells, cellProfiler = TRUE)

### Cluster cell types
intensities <- cellMarks(cellExp)
kM <- kmeans(intensities, 2)
cellType(cellExp) <- paste('cluster', kM$cluster, sep = '')

cellSummary(cellExp, imageID = 1)
```

---

**Description**

Function to coerce a SegmentedCells object to a data frame.

**Usage**

```r
## S3 method for class 'SegmentedCells'
as.data.frame(x, ...)
```

**Arguments**

- **x**: A SegmentedCells object.
- **...**: Other arguments.
bind

Value

A data.frame

```r
## Generate toy data set.seed(51773) x <- round(c(runif(200),runif(200)+1,runif(200)+2,runif(200)+3,
runif(200)+3,runif(200)+2,runif(200)+1,runif(200)),4) y <- round(c(runif(200),runif(200)+1,runif(200)+2,runif(200)+3,runif(200),runif(200)+1,runif(200)+2,runif(200)+3),4) cellType <- factor(paste('c',rep(rep(c(1:2),rep(200,2)),4),sep = '')) imageID <- rep(c('s1', 's2'),c(800,800)) cells <- data.frame(x, y, cellType, imageID)
```

```r
## Store data in SegmentedCells object cellExp <- SegmentedCells(cells, cellTypeString = 'cell-Type')
## Generate LISA cellsDF <- as.data.frame(cellExp)
```

```
bind(results, pairName = NULL)
```

Description

Produces a dataframe showing L-function metric for each imageID entry.

Usage

```r
bind(results, pairName = NULL)
```

Arguments

- `results` Spicy test result obtained from spicy.
- `pairName` A string specifying the pairwise interaction of interest. If NULL, all pairwise interactions are shown.

Value

A data.frame containing the colData related to the results.

Examples

```r
data(spicyTest)
df <- bind(spicyTest)
```
colTest

Perform a simple wilcoxon-rank-sum test or t-test on the columns of a data frame

Description

Perform a simple wilcoxon-rank-sum test or t-test on the columns of a data frame

Usage

colTest(df, condition, type = NULL, feature = NULL, imageID = "imageID")

Arguments

df A data.frame or SingleCellExperiment, SpatialExperiment
condition The condition of interest
type The type of test, "wilcox", "ttest" or "survival".
feature Can be used to calculate the proportions of this feature for each image
imageID The imageID's if presenting a SingleCellExperiment

Value

Proportions

Examples

# Test for an association with long-duration diabetes
# This is clearly ignoring the repeated measures...
data("diabetesData")
props <- getProp(diabetesData)
condition <- imagePheno(diabetesData)$stage
names(condition) <- imagePheno(diabetesData)$imageID
condition <- condition[condition %in% c("Long-duration", "Onset")]
test <- colTest(props[names(condition), ], condition)

convPairs

Converts colPairs object into an abundance matrix based on number of nearby interactions for every cell type.

Description

Converts colPairs object into an abundance matrix based on number of nearby interactions for every cell type.
Usage

`convPairs(cells, colPair, cellType = "cellType", imageID = "imageID")`

Arguments

- `cells`: A SingleCellExperiment that contains objects in the colPairs slot.
- `colPair`: The name of the object in the colPairs slot for which the dataframe is constructed from.
- `cellType`: The cell type if using SingleCellExperiment.
- `imageID`: The image ID if using SingleCellExperiment.

Value

Matrix of abundances

Examples

```r
data("diabetesData_SCE")

diabetesData_SPE <- SpatialExperiment::SpatialExperiment(diabetesData_SCE, colData = SingleCellExperiment::colData(diabetesData_SCE))
SpatialExperiment::spatialCoords(diabetesData_SPE) <- data.frame(
  SingleCellExperiment::colData(diabetesData_SPE)$x, 
  SingleCellExperiment::colData(diabetesData_SPE)$y) |>
  as.matrix()

SpatialExperiment::spatialCoordsNames(diabetesData_SPE) <- c("x", "y")

diabetesData_SPE <- imcRtools::buildSpatialGraph(diabetesData_SPE, img_id = "imageID", type = "knn", k = 20, coords = c("x", "y"))

pairAbundances <- convPairs(diabetesData_SPE, colPair = "knn_interaction_graph")
```

---

**diabetesData**

**Diabetes IMC data**

Description

This is a subset of the Damond et al 2019 imaging mass cytometry dataset. The data contains cells in the pancreatic islets of individuals with early onset diabetes and healthy controls. The object contains single-cell data of 160 images from 8 subjects, with 20 images per subject.
Usage
diabetesData

Format
diabetesData a SegmentedCells object

diabetesData_SCE  Diabetes IMC data in SCE format.

Description
This is a subset of the Damond et al 2019 imaging mass cytometry dataset. The data contains cells in the pancreatic islets of individuals with early onset diabetes and healthy controls. The object contains single-cell data of 160 images from 8 subjects, with 20 images per subject.

Usage
diabetesData_SCE

Format
diabetesData_SCE a SingleCellExperiment object

Details
Converted into a SingleCellExperiment format.

getPairwise  Get statistic from pairwise L curve of a single image.

Description
Get statistic from pairwise L curve of a single image.

Usage
getPairwise(
cells,
from = NULL,
to = NULL,
window = "convex",
window.length = NULL,
Rs = c(20, 50, 100),
sigma = NULL,
getPairwise

minLambda = 0.05,
edgeCorrect = TRUE,
includeZeroCells = TRUE,
BPPARAM = BiocParallel::SerialParam(),
imageID = "imageID",
cellType = "cellType",
spatialCoords = c("x", "y")
)

Arguments

cells A SegmentedCells or data frame that contains at least the variables x and y, giving the location coordinates of each cell, and cellType.
from The 'from' cellType for generating the L curve.
to The 'to' cellType for generating the L curve.
window Should the window around the regions be 'square', 'convex' or 'concave'.
window.length A tuning parameter for controlling the level of concavity when estimating concave windows.
Rs A vector of the radii that the measures of association should be calculated.
sigma A numeric variable used for scaling when fitting inhomogeneous L-curves.
minLambda Minimum value for density for scaling when fitting inhomogeneous L-curves.
edgeCorrect A logical indicating whether to perform edge correction.
includeZeroCells A logical indicating whether to include cells with zero counts in the pairwise association calculation.
BPPARAM A BiocParallelParam object.
imageID The imageID if using a SingleCellExperiment or SpatialExperiment.
cellType The cellType if using a SingleCellExperiment or SpatialExperiment.
spatialCoords The spatialCoords if using a SingleCellExperiment or SpatialExperiment.

Value

Statistic from pairwise L curve of a single image.

Examples

data("diabetesData")
pairAssoc <- getPairwise(diabetesData[1, ])

getProp

Description
Get proportions from a SegmentedCells, SingleCellExperiment, SpatialExperiment or data.frame.

Usage
getProp(cells, feature = "cellType", imageID = "imageID")

Arguments
- cells: SegmentedCells, SingleCellExperiment, SpatialExperiment or data.frame
- feature: The feature of interest
- imageID: The imageID’s

Value
Proportions

Examples
data("diabetesData")
prop <- getProp(diabetesData)

plot(SegmentedCells,ANY-method

Description
A basic plot for SegmentedCells object

Usage
## S4 method for signature 'SegmentedCells,ANY'
plot(x, imageID = NULL)

Arguments
- x: A SegmentedCells object.
- imageID: The image that should be plotted.
Value

A ggplot object.

usage

‘plot(x, imageID = NULL)’

Examples

```r
### Something that resembles cellProfiler data

set.seed(51773)

n = 10

cells <- data.frame(row.names = seq_len(n))
cells$ObjectNumber <- seq_len(n)
cells$ImageNumber <- rep(1:2, c(n/2, n/2))
cells$AreaShape_Center_X <- runif(n)
cells$AreaShape_Center_Y <- runif(n)
cells$AreaShape_round <- rexp(n)
cells$AreaShape_diameter <- rexp(n, 2)
cells$Intensity_Mean_CD8 <- rexp(n, 10)
cells$Intensity_Mean_CD4 <- rexp(n, 10)

cellExp <- SegmentedCells(cells, cellProfiler = TRUE)

### Cluster cell types

markers <- cellMarks(cellExp)
kM <- kmeans(markers, 2)
cellType(cellExp) <- paste('cluster', kM$cluster, sep = '')

#plot(cellExp, imageID=1)
```

SegmentedCells-class

The SegmentedCells class

Description

The SegmentedCells S4 class is for storing data from segmented imaging cytometry and spatial omics data. It extends DataFrame and defines methods that take advantage of DataFrame nesting to represent elements of cell-based experiments with spatial orientation that are commonly encountered. This object is able to store information on a cell’s spatial location, cellType, morphology, intensity of gene/protein markers as well as image level phenotype information.
Usage

SegmentedCells(
  cellData,
  cellProfiler = FALSE,
  spatialCoords = c("x", "y"),
  cellTypeString = "cellType",
  intensityString = "intensity",
  morphologyString = "morphology",
  phenotypeString = "phenotype",
  cellIDString = "cellID",
  cellAnnotations = NULL,
  imageCellIDString = "imageCellID",
  imageIDString = "imageID",
  verbose = TRUE
)

Arguments

  cellData  A data frame that contains at least the columns x and y giving the location coordinates of each cell.
  cellProfiler  A logical indicating that cellData is in a format similar to what cellProfiler outputs.
  spatialCoords  The column names corresponding to spatial coordinates. eg. x, y, z...
  cellTypeString  The name of the column that contains cell type calls.
  intensityString  A string which can be used to identify the columns which contain marker intensities. (This needs to be extended to take the column names themselves.)
  morphologyString  A string which can be used to identify the columns which contains morphology information.
  phenotypeString  A string which can be used to identify the columns which contains phenotype information.
  cellIDString  The column name for cellID.
  cellAnnotations  A vector of variables that provide additional annotation of a cell.
  imageCellIDString  The column name for imageCellID.
  imageIDString  The column name for imageIDString.
  verbose  logical indicating whether to output messages.

Value

A SegmentedCells object
Examples

### Something that resembles cellProfiler data

```r
set.seed(51773)

n = 10

cells <- data.frame(row.names = seq_len(n))
cells$ObjectNumber <- seq_len(n)
cells$ImageNumber <- rep(seq_len(2),c(n/2,n/2))
cells$AreaShape_Center_X <- runif(n)
cells$AreaShape_Center_Y <- runif(n)
cells$AreaShape_round <- rexp(n)
cells$AreaShape_diameter <- rexp(n, 2)
cells$Intensity_Mean_CD8 <- rexp(n, 10)
cells$Intensity_Mean_CD4 <- rexp(n, 10)

cellExp <- SegmentedCells(cells, cellProfiler = TRUE)

### Cluster cell types
intensities <- cellMarks(cellExp)
kM <- kmeans(intensities,2)
cellType(cellExp) <- paste('cluster',kM$cluster, sep = '')
cellSummary(cellExp)
```

---

**show-SegmentedCells**  
*Show SegmentedCells*

**Description**

This outputs critical information about a `SegmentedCells`.

**Arguments**

- `object`  
  A `SegmentedCells`.

**Value**

Information of the number of images, cells, intensities, morphologies and phenotypes.

**usage**

`'show(object)'`
### Examples

```r
### Something that resembles cellProfiler data

set.seed(51773)

n = 10

cells <- data.frame(row.names = seq_len(n))
cells$ObjectNumber <- seq_len(n)
cells$ImageNumber <- rep(1:2,c(n/2,n/2))
cells$AreaShape_Center_X <- runif(n)
cells$AreaShape_Center_Y <- runif(n)
cells$AreaShape_round <- rexp(n)
cells$AreaShape_diameter <- rexp(n, 2)
cells$Intensity_Mean_CD8 <- rexp(n, 10)
cells$Intensity_Mean_CD4 <- rexp(n, 10)

cellExp <- SegmentedCells(cells, cellProfiler = TRUE)

### Cluster cell types

markers <- cellMarks(cellExp)
kM <- kmeans(markers,2)
cellType(cellExp) <- paste('cluster',kM$cluster, sep = '')

cellExp
```

---

**signifPlot**  
Plots result of `signifPlot`.

---

**Description**

Plots result of `signifPlot`.

**Usage**

```r
signifPlot(
  results,
  fdr = FALSE,
  type = "bubble",
  breaks = NULL,
  comparisonGroup = NULL,
  colours = c("#4575B4", "white", "#D73027"),
  marksToPlot = NULL,
  cutoff = 0.05
)
```
spicyBoxPlot

**Arguments**

- **results**: Data frame obtained from spicy.
- **fdr**: TRUE if FDR correction is used.
- **type**: Where to make a bubble plot or heatmap.
- **breaks**: Vector of 3 numbers giving breaks used in heatmap. The first number is the minimum, the second is the maximum, the third is the number of breaks.
- **comparisonGroup**: A string specifying the name of the outcome group to compare with the base group.
- **colours**: Vector of colours to use in heatmap.
- **marksToPlot**: Vector of marks to include in heatmap.
- **cutoff**: Significance threshold for circles in bubble plot

**Value**

A heatmap object

**Examples**

```r
data(spicyTest)
signifPlot(spicyTest, breaks = c(-3, 3, 0.5))
```

---

**spicyBoxPlot**

*Plots boxplot for a specified cell-cell relationship*

**Description**

Plots boxplot for a specified cell-cell relationship

**Usage**

```
spicyBoxPlot(results, from = NULL, to = NULL, rank = NULL)
```

**Arguments**

- **results**: Data frame obtained from spicy.
- **from**: Cell type which you would like to compare to the to cell type.
- **to**: Cell type which you would like to compare to the from cell type.
- **rank**: Ranking of cell type in terms of p-value, the smaller the p-value the higher the rank.

**Value**

A ggplot2 boxplot
SpicyResults-class

Performs spatial tests on spatial cytometry data.

Description

Performs spatial tests on spatial cytometry data.

Usage

spicy(
  cells,
  condition = NULL,
  subject = NULL,
  covariates = NULL,
  from = NULL,
  to = NULL,
  alternateResult = NULL,
  verbose = TRUE,
  weights = TRUE,
  weightsByPair = FALSE,
  weightFactor = 1,
  window = "convex",
  window.length = NULL,
  BPPARAM = BiocParallel::SerialParam(),
  sigma = NULL,
  Rs = NULL,
  minLambda = 0.05,
  edgeCorrect = TRUE,
  includeZeroCells = FALSE,
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  ...
)

Arguments

cells A SegmentedCells or data frame that contains at least the variables x and y, giving the location coordinates of each cell, and cellType.
condition Vector of conditions to be tested corresponding to each image if cells is a data frame.
subject Vector of subject IDs corresponding to each image if cells is a data frame.
covariates Vector of covariate names that should be included in the mixed effects model as fixed effects.
from vector of cell types which you would like to compare to the to vector
to vector of cell types which you would like to compare to the from vector
alternateResult An pairwise association statistic between each combination of cell types in each image.
verbose logical indicating whether to output messages.
weights logical indicating whether to include weights based on cell counts.
weightsByPair logical indicating whether weights should be calculated for each cell type pair.
weightFactor numeric that controls the convexity of the weight function.
window Should the window around the regions be 'square', 'convex' or 'concave'.
window.length A tuning parameter for controlling the level of concavity when estimating concave windows.
BPARAM A BiocParallelParam object.
sigma A numeric variable used for scaling when fitting inhomogeneous L-curves.
Rs A vector of radii that the measures of association should be calculated.
minLambda Minimum value for density for scaling when fitting inhomogeneous L-curves.
edgeCorrect A logical indicating whether to perform edge correction.
includeZeroCells A logical indicating whether to include cells with zero counts in the pairwise association calculation.
imageID The image ID if using SingleCellExperiment.
cellType The cell type if using SingleCellExperiment.
spatialCoords The spatial coordinates if using a SingleCellExperiment.
... Other options.

Value
Data frame of p-values.

Examples

data("diabetesData")

# Test with random effect for patient on a pairwise combination of cell types.
spicy(diabetesData,
    condition = "stage", subject = "case",
    from = "Tc", to = "Th"
# Test all pairwise combinations of cell types without random effect of patient.
# spicyTest <- spicy(diabetesData, condition = "stage", subject = "case")

# Test all pairwise combination of cell types with random effect of patient.
# spicy(diabetesData, condition = "condition", subject = "subject")

---

**spicyTest**

*Results from spicy for diabetesData*

**Description**

Results from the call: `spicyTest <- spicy(diabetesData, condition = "condition", subject = "subject")`

**Usage**

`spicyTest`

**Format**

`spicyTest` a spicy object

---

**topPairs**

*A table of the significant results from spicy tests*

**Description**

A table of the significant results from spicy tests

**Usage**

`topPairs(x, coef = NULL, n = 10, adj = "fdr", cutoff = NULL, figures = NULL)`

**Arguments**

- `x` The output from spicy.
- `coef` Which coefficient to list.
- `n` Extract the top n most significant pairs.
- `adj` Which p-value adjustment method to use, argument for p.adjust().
- `cutoff` A p-value threshold to extract significant pairs.
- `figures` Round to ‘figures’ significant figures.
topPairs

Value

A data.frame

Examples

data(spicyTest)
topPairs(spicyTest)
## Index

### * datasets
- `diabetesData`, 7
- `diabetesData_SCE`, 8
- `spicyTest`, 18

### Accessors
- `as.data.frame.SegmentedCells`, 4
- `bind`, 5
- `cellAnnotation`, 2
- `cellAnnotation<-(Accessors)`, 2
- `cellAnnotation<-,SegmentedCells-method (Accessors)`, 2
- `cellID`, 2
- `cellID<-(Accessors)`, 2
- `cellID<-,SegmentedCells-method (Accessors)`, 2
- `cellMarks`, 2
- `cellMarks<-(Accessors)`, 2
- `cellMarks<-,SegmentedCells-method (Accessors)`, 2
- `cellMorph`, 2
- `cellMorph<-(Accessors)`, 2
- `cellMorph<-,SegmentedCells-method (Accessors)`, 2
- `cellSummary`, 2
- `cellSummary<-(Accessors)`, 2
- `cellSummary<-,SegmentedCells-method (Accessors)`, 2
- `cellType`, 2
- `cellType<-(Accessors)`, 2
- `cellType<-,SegmentedCells-method (Accessors)`, 2
- `colTest`, 6
- `convPairs`, 6
- `diabetesData`, 7
- `diabetesData_SCE`, 8
- `filterCells`, 2
- `filterCells<-,SegmentedCells-method (Accessors)`, 2
- `getPairwise`, 8
- `getProp`, 10
- `imageCellID`, 2
- `imageCellID<-(Accessors)`, 2
- `imageCellID<-,SegmentedCells-method (Accessors)`, 2
- `imageID`, 2
- `imageID<-(Accessors)`, 2
- `imageID<-,SegmentedCells-method (Accessors)`, 2
- `imagePheno`, 2
- `imagePheno<-(Accessors)`, 2
- `imagePheno<-,SegmentedCells-method (Accessors)`, 2
- `plot (plot,SegmentedCells,ANY-method)`, 10
- `plot,SegmentedCells`
  - (plot,SegmentedCells,ANY-method), 10
- `plot,SegmentedCells,ANY-method`, 10
INDEX

SegmentedCells (SegmentedCells-class), 11
SegmentedCells,SegmentedCells-method
  (SegmentedCells-class), 11
SegmentedCells-class, 11
show (show-SegmentedCells), 13
show-SegmentedCells, 13
signifPlot, 14
spicy (SpicyResults-class), 16
spicy,spicy-method
  (SpicyResults-class), 16
spicyBoxPlot, 15
SpicyResults,list,ANY-method
  (SpicyResults-class), 16
SpicyResults-class, 16
spicyTest, 18
topPairs, 18
topPairs,SpicyResults-method
  (topPairs), 18