Package ‘scTreeViz’

March 16, 2024

Type Package

Title R/Bioconductor package to interactively explore and visualize single cell RNA-seq datasets with hierarchical annotations

Version 1.8.0

Description scTreeViz provides classes to support interactive data aggregation and visualization of single cell RNA-seq datasets with hierarchies for e.g. cell clusters at different resolutions. The `TreeIndex` class provides methods to manage hierarchy and split the tree at a given resolution or across resolutions. The `TreeViz` class extends `SummarizedExperiment` and can performs quick aggregations on the count matrix defined by clusters.

License Artistic-2.0

Depends R (>= 4.0), methods, epivizr, SummarizedExperiment

Imports data.table, S4Vectors, digest, Matrix, Rtsne, httr, igraph, clustree, scran, sys, epivizrData, epivizrServer, gggraph, scater, Seurat, SingleCellExperiment, ggplot2, stats, utils

Suggests knitr, BiocStyle, testthat, SC3, scRNAseq, rmarkdown, msd16s, metagenomeSeq, epivizrStandalone, GenomeInfoDb

VignetteBuilder knitr

biocViews Visualization, Infrastructure, GUI, SingleCell

Encoding UTF-8

LazyData true


RoxygenNote 7.1.1

git_url https://git.bioconductor.org/packages/scTreeViz
git_branch RELEASE_3_18
git_last_commit 2e0db4d
git_last_commit_date 2023-10-24
Repository  Bioconductor 3.18
Date/Publication  2024-03-15
Author  Jayaram Kancherla [aut, cre],
        Hector Corrada Bravo [aut],
        Kazi Tasnim Zinat [aut],
        Stephanie Hicks [aut]
Maintainer  Jayaram Kancherla <jayaram.kancherla@gmail.com>

R topics documented:

  .generate_hierarchy_tree ........................................ 2
  .generate_leaf_of_table ......................................... 3
  .generate_nodes_table ........................................... 4
  .generate_node_ids .............................................. 4
  .replaceNAFeatures ............................................. 5
  ClusterHierarchy ............................................... 5
  ClusterHierarchy-class .......................................... 6
  createFromSCE .................................................. 6
  createFromSeurat ................................................. 7
  createTreeViz .................................................. 8
  EpivizTreeData-class ........................................... 9
  set_gene_list .................................................. 10
  show,TreeViz-method ........................................... 10
  startTreeviz ................................................... 12
  TreeIndex ....................................................... 13
  TreeIndex-class ................................................ 14
  TreeViz ........................................................ 14
  TreeViz-class ................................................... 15
  TreeVizApp-class ................................................ 15
  [,TreeIndex,ANY,ANY,ANY-method ......................... 15

Index  18

  .generate_hierarchy_tree
    generate hierarchy tree

Description

  generate hierarchy tree

Usage

  .generate_hierarchy_tree(hierarchy, feature_order)
.generate_leaf_of_table

Arguments

- hierarchy: hierarchy as a data.table
- feature_order: order of the tree if different from colnames

Value

a data frame object

Description

generate leaf of table

Usage

.generate_leaf_of_table(
  hierarchy_tree,
  node_ids_table,
  nodes_table,
  feature_order
)

Arguments

- hierarchy_tree: hierarchy as a data.table
- node_ids_table: node ids
- nodes_table: nodes table
- feature_order: order of the tree if different from colnames

Value

a data frame object
### .generate_nodes_table  generate nodes table tree

**Description**

generate nodes table tree

**Usage**

```
.generate_nodes_table(hierarchy_tree, node_ids_table, feature_order)
```

**Arguments**

- `hierarchy_tree`: hierarchy as a data.table
- `node_ids_table`: node ids
- `feature_order`: order of the tree if different from colnames

**Value**

a data frame object

### .generate_node_ids  generate node ids in the tree

**Description**

generate node ids in the tree

**Usage**

```
.generate_node_ids(hierarchy_tree, feature_order)
```

**Arguments**

- `hierarchy_tree`: hierarchy as a data.table
- `feature_order`: order of the tree if different from colnames

**Value**

a data frame object
.replaceNAFeatures  
replace if there are NA's in the hierarchy

Description
replace if there are NA's in the hierarchy

Usage
.replaceNAFeatures(replacing_na_obj_fData, feature_order)

Arguments
replacing_na_obj_fData
    hierarchy data table
feature_order  order of the tree if different from colnames

Value
a data frame object

ClusterHierarchy  
Creates a new ClusterHierarchy object.

Description
Works as a validation check for multiple issues user passed dataframe might have. For example, multiple root nodes, incompatible naming, multiple parents of a single node, etc. This function performs all this checks and tries to resolve the issues by making changes in cluster assignment. User can give either col_regex or columns option to filter the columns or specify the column order.

Usage
ClusterHierarchy(hierarchy, col_regex = NULL, columns = NULL)

Arguments
hierarchy    hierarchy as a dataFrame
col_regex    Regular Expression for choosing columns
columns      Vector containing list of columns to choose from with ordering

Value
'ClusterHierarchy' return an object of class ClusterHierarchy containing cluster information that ensures a valid dataframe for treeviz input.
Examples

n=64
# create a hierarchy
df<- data.frame(cluster0=rep(1,n))
for(i in seq(1,5)){
  df[[paste0("cluster",i)]]<- rep(seq(1:(2**i)),each=ceiling(n/(2**i)),len=n)
}
clus_hier<-ClusterHierarchy(df, col_regex = "clus")

ClusterHierarchy-class

ClusterHierarchy class to manage treeviz cluster data

createFromSCE

Creates a 'TreeViz' object from 'SingleCellExperiment'. Generates clusters based on Walktrap algorithm if no default is provided

Description

Creates a ‘TreeViz’ object from ‘SingleCellExperiment’. Generates clusters based on Walktrap algorithm if no default is provided

Usage

createFromSCE(
  object,
  check_coldata = FALSE,
  col_regex = NULL,
  columns = NULL,
  reduced_dim = c("TSNE")
)

Arguments

object 'SingleCellExperiment' object to be visualized
check_coldata whether to colData of ‘SingleCellExperiment’ object for cluster information or not
col_regex common regular expression shared across all columns with cluster information
columns vector containing columns with cluster information
reduced_dim Vector of Dimensionality reduction information provided in ‘SingleCellExperiment’ object to be added in ‘TreeViz’ (if exists)
createFromSeurat

Value

‘TreeViz’ Object

Examples

library(SingleCellExperiment)
library(scater)
sce <- mockSCE()
sce <- logNormCounts(sce)
sce <- runTSNE(sce)
sce <- runUMAP(sce)
set.seed(1000)
for (i in seq_len(5)) {
  clust.kmeans <- kmeans(reducedDim(sce, “TSNE”), centers = i)
sce[[paste0(“clust”, i)]] <- factor(clust.kmeans$cluster)
}
treeviz <-createFromSCE(sce, check_coldata = TRUE, col_regex = “clust”, reduced_dim = c(“TSNE”, “UMAP”))

Describe

Creates a ‘TreeViz’ object from ‘Seurat’

Usage

createFromSeurat(
  object,
  check_metadata = FALSE,
  col_regex = “*snn*”,
  columns = NULL,
  reduced_dim = c(“TSNE”)
)

Arguments

object ‘Seurat’ class containing cluster information at different resolutions
check_metadata whether to metaData of ‘Seurat’ object for cluster information or not
col_regex common regular expression shared across all columns with cluster information
columns vector containing columns with cluster information
reduced_dim Vector of Dimensionality reduction information provided in ‘Seurat’ object to
be added in ‘TreeViz’ (if exists)

Value

‘TreeViz’ Object
createTreeViz

Description

Provided with a count matrix and a dataframe or 'ClusterHierarchy' object, this module runs the necessary checks on the dataframe and tries to convert it to a tree by making necessary changes. Returns the 'TreeViz' object if a tree is successfully generated from dataframe, throws error otherwise.

Usage

createTreeViz(clusters, counts)

Arguments

clusters 'ClusterHierarchy' object or a dataframe containing cluster information at different resolutions

counts matrix Dense or sparse matrix containing the count matrix

Value

'TreeViz' Object

Examples

n=64
# create a hierarchy
df<- data.frame(cluster0=rep(1,n))
for(i in seq_len(5)){
  df[[paste0("cluster",i)]]<- rep(seq(1:(2*i)),each=ceiling(n/(2*i)),len=n)
}
# generate a count matrix
counts <- matrix(rpois(6400, lambda = 10), ncol=n, nrow=100)
colnames(counts)<- seq_len(64)
# create a `TreeViz` object
treeViz <- createTreeViz(df, counts)
EpivizTreeData-class  

Data container for MRexperiment objects

Description

Used to serve hierarchical data (used in e.g., icicle plots and heatmaps).

Methods

df_to_tree(root, df)  Helper function to recursively build nested response for getHierarchy

  root  Root of subtree
  df    data.frame containing children to process

get_default_chart_type()  Get name of default chart type for this data type

get_measurements()  Get description of measurements served by this object

getCombined(measurements = NULL, seqName, start = 1, end = 1000, order = NULL, nodeSelection = NULL, selectedLevels = NULL)  Return the counts aggregated to selected nodes for the given samples

  measurements  Samples to get counts for
  seqName  name of datasource
  start  Start of feature range to query
  end  End of feature range to query
  order  Ordering of nodes
  nodeSelection  Node-id and selectionType pairs
  selectedLevels  Current aggregation level

getHierarchy(nodeId = NULL)  Retrieve feature hierarchy information for subtree with specified root

  nodeId  Feature identifier with level info

getReducedDim(method = NULL, gene = NULL)  Compute PCA over all features for given samples

  method  which dimension to access
  gene  send expression of a gene back with the dimensions

getRows(measurements = NULL, start = 1, end = 1000, selectedLevels = 3, selections = NULL)  Return the sample annotation and features within the specified range and level for a given sample and features

  measurements  Samples to retrieve for
  start  Start of feature range to query
  end  End of feature range to query
  selections  Node-id and selectionType pairs
  selectedLevels  Current aggregation level

propagateHierarchyChanges( selection = NULL, order = NULL, selectedLevels = NULL, request_with_labels = FALSE)  Update internal state for hierarchy

  selection  Node-id and selectionType pairs
  order  Ordering of features
**selectedLevels**  Current aggregation level

**request_with_labels**  For handling requests using fData entries from MRexperiment

**row_to_dict(row)**  Helper function to format each node entry for getHierarchy response

**row**  Information for current node.

**searchTaxonomy(query = NULL, max_results = 15)**  Return list of features matching a text-based query

  **query**  String of feature for which to search

  **max_results**  Maximum results to return

---

**set_gene_list**  Sets gene list for visualization

**Description**

Sets gene list for visualization

**Usage**

```r
set_gene_list(treeviz, genes)
```

**Arguments**

- **treeviz**  TreeViz object
- **genes**  list of genes to use

**Value**

TreeViz object set with gene list

---

**show,TreeViz-method**  show object

**Description**

show object

Method to aggregate a TreeViz object

Method to aggregate a TreeViz object

Generic method to register data to the epiviz data server

plot tree from TreeViz
Usage

```r
## S4 method for signature 'TreeViz'
show(object)

aggregateTree(x, ...)

## S4 method for signature 'TreeViz'
aggregateTree(
  x,
  selectedLevel = 3,
  selectedNodes = NULL,
  aggFun = colSums,
  start = 1,
  end = NULL,
  by = "row",
  format = "TreeViz"
)

## S4 method for signature 'TreeViz'
register(object, tree = "row", columns = NULL, ...)

## S4 method for signature 'TreeViz,ANY'
plot(x, y)
```

Arguments

- `object`: The object to register to data server
- `x`: treenviz object
- `...`: Additional arguments passed to object constructors
- `selectedLevel`: level to select nodes from
- `selectedNodes`: used to set states on individual nodes to define a cut on the tree
- `aggFun`: aggregate function to use, by default colSums if by="row", rowSums if by="col"
- `start, end`: indices to filter nodes
- `by`: "row" to aggregate the TreeIndex on rowData, "col" to aggregate TreeIndex on colData
- `format`: return format can be one of "counts" or "TreeViz"
- `tree`: Is tree over rows or columns of the object (default: "row")
- `columns`: Name of columns containing data to register
- `y`: none

Value

- describe a TreeIndex object
- a generic
a Treeviz object or type specified by format
An \texttt{EpivizTreeData-class} object
Dataframe containing cluster information at different resolutions

Functions

- \texttt{show,TreeViz-method}:
- \texttt{aggregateTree}:
- \texttt{aggregateTree,TreeViz-method}:
- \texttt{register,TreeViz-method}:
- \texttt{plot,TreeViz,ANY-method}:

Examples

```r
library(metagenomeSeq)
data(mouseData)counts <- MRcounts(mouseData)hierarchy <- fData(mouseData)tree <- TreeIndex(hierarchy)mbiome <- TreeViz(SimpleList(counts=counts), rowData=tree)aggregateTree(mbiome)
```

```r
library(metagenomeSeq)
data(mouseData)counts <- MRcounts(mouseData)hierarchy <- fData(mouseData)tree <- TreeIndex(hierarchy)mbiome <- TreeViz(SimpleList(counts=counts), rowData=tree)aggregateTree(mbiome)
```

```r
library(metagenomeSeq)
data(mouseData)counts <- MRcounts(mouseData)hierarchy <- fData(mouseData)tree <- TreeIndex(hierarchy)mbiome <- TreeViz(SimpleList(counts=counts), rowData=tree)plot(mbiome)
```

---

\texttt{startTreeviz} \hspace{1cm} \textit{Start treeviz app and create \texttt{TreeVizApp} object to manage connection.}

Description

Start treeviz app and create \texttt{TreeVizApp} object to manage connection.
Usage

```
startTreeviz(
    data = NULL,
    genes = NULL,
    top_genes = 100,
    host = "http://epiviz.cbcb.umd.edu/treeviz",
    register_function = .register_all_treeviz_things,
    delay = 2L,
    ...
)
```

Arguments

- `data` (TreeViz object to explore)
- `genes` (character vector) genes (rownames) to include in heatmap
- `top_genes` (integer) number of top variable genes to include in the heatmap
- `host` (character) host address to launch.
- `register_function` (function) function used to register actions and charts on the treeviz app.
- `delay` (integer) number of seconds to wait for application to load in browser
- `...` additional parameters passed to `startEpiviz`.

Value

An object of class `TreeVizApp`

See Also

- `TreeVizApp`

Examples

```
# see package vignette for example usage
app <- startTreeviz(non_interactive=TRUE, open_browser=FALSE)
app$stop_app()
```

---

**TreeIndex**

create a new `TreeIndex` object

**Description**

create a new `TreeIndex` object

**Usage**

```
TreeIndex(hierarchy = NULL, feature_order = NULL)
```
TreeViz

Arguments

- hierarchy: hierarchy as a data.table
- feature_order: order of the tree if different from colnames

Value

- a 'TreeIndex' object

Examples

```r
library(metagenomeSeq)
data(mouseData)
counts <- MRcounts(mouseData)
hierarchy <- fData(mouseData)
tree <- TreeIndex(hierarchy)
```

---

TreeIndex-class

TreeIndex class to manage and query hierarchical data

Description

TreeIndex class to manage and query hierarchical data

---

TreeViz

The TreeViz class.

Description

SummarizedExperiment-like class for datasets that have hierarchies on either rowData or colData. For microbiome data, rowData is a tree hierarchy. For single cell data, colData is a tree hierarchy.

Usage

```r
TreeViz(assays = SimpleList(), rowData = NULL, colData = NULL, ...)
```

Arguments

- assays: simple list of counts
- rowData: rowData
- colData: colData
- ...: other parameters for SummarizedExperiment

Value

- a 'TreeViz' object
TreeViz-class

Examples

```r
library(metagenomeSeq)
data(mouseData)
counts <- MRcounts(mouseData)
hierarchy <- fData(mouseData)
tree <- TreeIndex(hierarchy)
mbiome <- TreeViz(SimpleList(counts=counts), rowData=tree)
```

Description

TreeViz class wrapper for SummarizedExperiment objects

TreeVizApp-class

Class managing connection to metaviz application.

Methods

plotGene(gene = NULL, datasource_name = "SCRNA_1") Plot a bar plot for a gene across cell types
gene gene to extract expression values
datasource_name object to extract from (automatically selected)

[,TreeIndex,ANY,ANY,ANY-method

Subset TreeIndex

Description

Subset TreeIndex
Generic method to get nodes at a tree level
Method to get nodes at a tree level
Generic method for possible node states
Method to get possible node states a node state is 0 if removed, 1 if expanded to show children & 2 if counts are aggregated to the node
Generic method to split the tree
splitAt divides the TreeIndex into groups defined by the level, node selections and filters(start, end)
Show the TreeIndex object
Usage

## S4 method for signature 'TreeIndex,ANY,ANY,ANY'

x[i, j, ..., drop = FALSE]

getNodes(x, ...)

## S4 method for signature 'TreeIndex'

getNodes(x, selectedLevel = NULL)

goListNodeStates(x)

## S4 method for signature 'TreeIndex'

getNodeStates(x)

## S4 method for signature 'TreeIndex'

splitAt(x, ...)

## S4 method for signature 'TreeIndex'

splitAt(  
  x,  
  selectedLevel = 3,  
  selectedNodes = NULL,  
  start = 1,  
  end = NULL,  
  format = "list"  
)

## S4 method for signature 'TreeIndex'

show(object)

Arguments

x TreeIndex object
i, j indices to subset or keep
... other parameters
drop drop the dimensions of the object. defaults to FALSE
selectedLevel tree level to select nodes from
selectedNodes used to set states on individual nodes to define a cut on the tree
start, end indices to filter nodes by
format return format can be one of "list" or "TreeIndex"
object TreeIndex object

Value

a `TreeIndex` subset object
a generic
levels at node cut
node state
node states
a generic
a ‘TreeIndex’ object or type set in format
object description of the ‘TreeIndex’ object

**Examples**

```r
library(metagenomeSeq)
data(mouseData)
counts <- MRcounts(mouseData)
hierarchy <- fData(mouseData)
tree <- TreeIndex(hierarchy)
getNodes(tree)
library(metagenomeSeq)
data(mouseData)
counts <- MRcounts(mouseData)
hierarchy <- fData(mouseData)
tree <- TreeIndex(hierarchy)
getNodes(tree)

library(metagenomeSeq)
data(mouseData)
counts <- MRcounts(mouseData)
hierarchy <- fData(mouseData)
tree <- TreeIndex(hierarchy)
splitAt(tree)

library(metagenomeSeq)
data(mouseData)
counts <- MRcounts(mouseData)
hierarchy <- fData(mouseData)
tree <- TreeIndex(hierarchy)
splitAt(tree)
```
Index

.generate_hierarchy_tree, 2
.generate_leaf_of_table, 3
.generate_node_ids, 4
.generate_nodes_table, 4
.replaceNAFeatures, 5
[,TreeIndex,ANY,ANY,ANY-method, 15
aggregateTree(show,TreeViz-method), 10
aggregateTree,TreeViz-method
(show,TreeViz-method), 10

ClusterHierarchy, 5
ClusterHierarchy-class, 6
createFromSCE, 6
createFromSeurat, 7
createTreeViz, 8

EpivizTreeData (EpivizTreeData-class), 9
EpivizTreeData-class, 9

getNodes
([],TreeIndex,ANY,ANY,ANY-method), 15
getNodes,TreeIndex-method
([],TreeIndex,ANY,ANY,ANY-method), 15
getNodeStates
([],TreeIndex,ANY,ANY,ANY-method), 15
getNodeStates,TreeIndex-method
([],TreeIndex,ANY,ANY,ANY-method), 15

plot,TreeViz,ANY-method
(show,TreeViz-method), 10

register,TreeViz-method
(show,TreeViz-method), 10

set_gene_list, 10

show,TreeIndex-method
([],TreeIndex,ANY,ANY,ANY-method), 15
show,TreeViz-method, 10
splitAt
([],TreeIndex,ANY,ANY,ANY-method), 15
splitAt,TreeIndex-method
([],TreeIndex,ANY,ANY,ANY-method), 15
startEpiviz, 13
startTreeviz, 12

TreeIndex, 13
TreeIndex-class, 14
TreeViz, 14
TreeViz-class, 15
TreeVizApp, 12, 13
TreeVizApp (TreeVizApp-class), 15
TreeVizApp-class, 15