Package 'clustree'

April 20, 2018

Type Package

Title Visualise Clusterings at Different Resolutions

Version 0.1.2

Date 2018-04-11

Maintainer Luke Zappia <luke.zappia@mcri.edu.au>

Description Deciding what resolution to use can be a difficult question when approaching a clustering analysis. One way to approach this problem is to look at how samples move as the number of clusters increases. This package allows you to produce clustering trees, a visualisation for interrogating clusterings as resolution increases.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/lazappi/clustree

BugReports https://github.com/lazappi/clustree/issues

VignetteBuilder knitr

Depends R (>= 3.4), ggraph

Imports checkmate, igraph, dplyr, grid, ggplot2, viridis, methods, rlang

Suggests testthat, knitr, rmarkdown, SingleCellExperiment, Seurat, covr

RoxygenNote 6.0.1

NeedsCompilation no

Author Luke Zappia [aut, cre] (<https://orcid.org/0000-0001-7744-8565>), Alicia Oshlack [aut] (<https://orcid.org/0000-0001-9788-5690>), Andrea Rau [ctb]

Repository CRAN

Date/Publication 2018-04-20 03:18:46 UTC

R topics documented:

`clustree-package` *Clustree*

Description

Deciding what resolution to use can be a difficult question when approaching a clustering analysis. One way to approach this problem is to look at how samples move as the number of clusters increases. This package allows you to produce clustering trees, a visualisation for interrogating clusterings as resolution increases.

`add_node_points` *Add node points*

Description

Add node points to a clustering tree plot with the specified aesthetics.

Usage

```
add_node_points(prefix, node_colour, node_size, node_alpha, allowed)
```

Arguments

prefix	string indicating columns containing clustering information
node_colour	either a value indicating a colour to use for all nodes or the name of a metadata column to colour nodes by
node_size	either a numeric value giving the size of all nodes or the name of a metadata column to use for node sizes
node_alpha	either a numeric value giving the alpha of all nodes or the name of a metadata column to use for node transparency
allowed	vector of allowed node attributes to use as aesthetics

---

aggr_metadata *Aggregate metadata*

Description

Aggregate a metadata column to get a summarized value for a cluster node

Usage

```
aggr_metadata(node_data, col_name, col_aggr, metadata, is_cluster)
```

Arguments

node_data	data.frame containing information about a set of cluster nodes
col_name	the name of the metadata column to aggregate
col_aggr	string naming a function used to aggregate the column
metadata	data.frame providing metadata on samples
is_cluster	logical vector indicating which rows of metadata are in the node to be summarized

Value

data.frame with aggregated data

assert_colour_node_aes *Assert colour node aesthetics*

Description

Raise error if an incorrect set of colour node parameters has been supplied.

Usage

```
assert_colour_node_aes(node_aes_name, prefix, metadata, node_aes, node_aes_aggr,
  min, max)
```

Arguments

node_aes_name	name of the node aesthetic to check
prefix	string indicating columns containing clustering information
metadata	data.frame containing metadata on each sample that can be used as node aesthetics
node_aes	value of the node aesthetic to check
node_aes_aggr	aggregation funciton associated with the node aesthetic
min	minimum numeric value allowed
max	maximum numeric value allowed

assert_node_aes *Assert node aesthetics*

Description

Raise error if an incorrect set of node parameters has been supplied.

Usage

```
assert_node_aes(node_aes_name, prefix, metadata, node_aes, node_aes_aggr)
```

Arguments

node_aes_name	name of the node aesthetic to check
prefix	string indicating columns containing clustering information
metadata	data.frame containing metadata on each sample that can be used as node aesthetics
node_aes	value of the node aesthetic to check
node_aes_aggr	aggregation funciton associated with the node aesthetic

assert_numeric_node_aes *Assert numeric node aesthetics*

Description

Raise error if an incorrect set of numeric node parameters has been supplied.

Usage

```
assert_numeric_node_aes(node_aes_name, prefix, metadata, node_aes,
  node_aes_aggr, min, max)
```

Arguments

node_aes_name	name of the node aesthetic to check
prefix	string indicating columns containing clustering information
metadata	data.frame containing metadata on each sample that can be used as node aesthetics
node_aes	value of the node aesthetic to check
node_aes_aggr	aggregation funciton associated with the node aesthetic
min	minimum numeric value allowed
max	maximum numeric value allowed

build_tree_graph *Build tree graph*

Description

Build a tree graph from a set of clusterings, metadata and associated aesthetics

Usage

```
build_tree_graph(clusterings, prefix, count_filter, prop_filter, metadata,
  node_colour, node_colour_aggr, node_size, node_size_aggr, node_alpha,
  node_alpha_aggr)
```

Arguments

clusterings	numeric matrix containing clustering information, each column contains clustering at a separate resolution
prefix	string indicating columns containing clustering information
count_filter	count threshold for filtering edges in the clustering graph
prop_filter	in proportion threshold for filtering edges in the clustering graph
metadata	data.frame containing metadata on each sample that can be used as node aesthetics
node_colour	either a value indicating a colour to use for all nodes or the name of a metadata column to colour nodes by
node_colour_aggr	if node_colour is a column name than a function to aggregate that column for samples in each cluster
node_size	either a numeric value giving the size of all nodes or the name of a metadata column to use for node sizes
node_size_aggr	if node_size is a column name than a function to aggregate that column for samples in each cluster
node_alpha	either a numeric value giving the alpha of all nodes or the name of a metadata column to use for node transparency
node_alpha_aggr	if node_size is a column name than a function to aggregate that column for samples in each cluster

Value

igraph::igraph object containing the tree graph

---

clustree *Plot a clustering tree*

Description

Creates a plot of a clustering tree showing the relationship between clusterings at different resolutions.

Usage

```
clustree(x, ...)

## S3 method for class 'matrix'
clustree(x, prefix, suffix = NULL, count_filter = 0,
  prop_filter = 0.1, metadata = NULL, node_colour = prefix,
  node_colour_aggr = NULL, node_size = "size", node_size_aggr = NULL,
  node_size_range = c(4, 15), node_alpha = 1, node_alpha_aggr = NULL,
```

```
  node_text_size = 3, scale_node_text = FALSE, node_text_colour = "black",
  edge_width = 1.5, edge_arrow = TRUE, edge_arrow_ends = "last",
  layout = c("tree", "sugiyama"), ...)

## S3 method for class 'data.frame'
clustree(x, prefix, ...)

## S3 method for class 'SingleCellExperiment'
clustree(x, prefix, exprs = "counts", ...)

## S3 method for class 'seurat'
clustree(x, prefix = "res.", exprs = c("data", "raw.data",
  "scale.data"), ...)
```

Arguments

x	object containing clustering data
...	extra parameters passed to other methods
prefix	string indicating columns containing clustering information
suffix	string at the end of column names containing clustering information
count_filter	count threshold for filtering edges in the clustering graph
prop_filter	in proportion threshold for filtering edges in the clustering graph
metadata	data.frame containing metadata on each sample that can be used as node aesthetics
node_colour	either a value indicating a colour to use for all nodes or the name of a metadata column to colour nodes by
node_colour_aggr	if node_colour is a column name than a string giving the name of a function to aggregate that column for samples in each cluster
node_size	either a numeric value giving the size of all nodes or the name of a metadata column to use for node sizes
node_size_aggr	if node_size is a column name than a string giving the name of a function to aggregate that column for samples in each cluster
node_size_range	numeric vector of length two giving the maximum and minimum point size for plotting nodes
node_alpha	either a numeric value giving the alpha of all nodes or the name of a metadata column to use for node transparency
node_alpha_aggr	if node_aggr is a column name than a string giving the name of a function to aggregate that column for samples in each cluster
node_text_size	numeric value giving the size of node labels if scale_node_text is FALSE
scale_node_text	logical indicating whether to scale node labels along with the node size

`node_text_colour`
: colour value for node labels

`edge_width`
: numeric value giving the width of plotted edges

`edge_arrow`
: logical indicating whether to add an arrow to edges

`edge_arrow_ends`
: One of `"last"`, `"first"`, or `"both"`, indicating which ends of the line to draw arrow heads if `edge_arrow = "TRUE"`.

`layout`
: character specifying the "tree" or "sugiyama" layout, see `igraph::layout_as_tree()` and `igraph::layout_with_sugiyama()` for details

`exprs`
: source of gene expression information to use as node aesthetics, for `SingleCellExperiment` objects it must be a name in `SummarizedExperiment::assayNames()`, for a `seurat` object it must be one of `data`, `raw.data` or `scale.data`

Details

Data sources

Plotting a clustering tree requires information about which cluster each sample has been assigned to at different resolutions. This information can be supplied in various forms, as a matrix, data.frame or more specialised object. In all cases the object provided must contain numeric columns with the naming structure `PXS` where `P` is a prefix indicating that the column contains clustering information, `X` is a numeric value indicating the clustering resolution and `S` is any additional suffix to be removed. For `SingleCellExperiment` objects this information must be in the `colData` slot and for `Seurat` objects it must be in the `meta.data` slot. For all objects except matrices any additional columns can be used as aesthetics, for matrices an additional metadata data.frame can be supplied if required.

Filtering

Edges in the graph can be filtered by adjusting the `count_filter` and `prop_filter` parameters. The `count_filter` removes any edges that represent less than that number of samples, while the `prop_filter` removes edges that represent less than that proportion of cells in the node it points towards.

Node aesthetics

The aesthetics of the plotted nodes can be controlled in various ways. By default the colour indicates the clustering resolution, the size indicates the number of samples in that cluster and the transparency is set to 100 Each of these can be set to a specific value or linked to a supplied metadata column. For a `SingleCellExperiment` or `Seurat` object the names of genes can also be used. If a metadata column is used than an aggregation function must also be supplied to combine the samples in each cluster. This function must take a vector of values and return a single value.

Value

ggplot2::ggplot object containing a clustering tree

Examples

```
data(iris_clusts)
clustree(iris_clusts, prefix = "K")
```

get_tree_edges *Get tree edges*

Description

Extract the edges from a set of clusterings

Usage

```
get_tree_edges(clusterings, prefix)
```

Arguments

clusterings	numeric matrix containing clustering information, each column contains clustering at a separate resolution
prefix	string indicating columns containing clustering information

Value

data.frame containing edge information

get_tree_nodes *Get tree nodes*

Description

Extract the nodes from a set of clusterings and add relevant attributes

Usage

```
get_tree_nodes(clusterings, prefix, metadata, node_colour, node_colour_aggr,
  node_size, node_size_aggr, node_alpha, node_alpha_aggr)
```

Arguments

clusterings	numeric matrix containing clustering information, each column contains clustering at a separate resolution
prefix	string indicating columns containing clustering information
metadata	data.frame containing metadata on each sample that can be used as node aesthetics
node_colour	either a value indicating a colour to use for all nodes or the name of a metadata column to colour nodes by
node_colour_aggr	if node_colour is a column name than a function to aggregate that column for samples in each cluster

node_size	either a numeric value giving the size of all nodes or the name of a metadata column to use for node sizes
node_size_aggr	if node_size is a column name than a function to aggregate that column for samples in each cluster
node_alpha	either a numeric value giving the alpha of all nodes or the name of a metadata column to use for node transparency
node_alpha_aggr	if node_size is a column name than a function to aggregate that column for samples in each cluster

Value

data.frame containing node information

---

iris_clusts *Clustered Iris dataset*

Description

Iris dataset clustered using k-means with a range of values of k

Usage

```
iris_clusts
```

Format

iris_clusts is a data.frame containing the normal iris dataset with additional columns holding k-means clusterings at different values of k

Source

```
set.seed(1)
iris_mat <- as.matrix(iris[1:4])
iris_km <- sapply(1:5, function(x) {
    km <- kmeans(iris_mat, centers = x, iter.max = 100, nstart = 10)
    km$cluster
})
colnames(iris_km) <- paste0("K", 1:5)
iris_clusts <- cbind(iris, iris_km)
```

sim_sc3 *SC3 Clustered simulated scRNA-seq dataset*

Description

A simulated scRNA-seq dataset generated using the `splatter` package and clustered using `SC3`.

Usage

```
sim_sc3
```

Format

`sim_sc3` is a SingleCellExperiment::SingleCellExperiment object holding a simulated scRNA-seq dataset. The dataset has been clustered using the `SC3` package with a range of values for k. The results of the clustering are held in the `colData` slot with columns named `sc3_X_clusters` where `X` is the value of k.

Source

```
# Simulation
library("splatter") # Version 1.2.1

sim <- splatSimulate(batchCells = 200, nGenes = 10000,
                     group.prob = c(0.4, 0.2, 0.2, 0.15, 0.05),
                     de.prob = c(0.1, 0.2, 0.05, 0.1, 0.05),
                     method = "groups", seed = 1)
sim_counts <- counts(sim)[1:1000, ]

# Clustering
library("SC3") # Version 1.7.6
library("scater") # Version 1.6.2

sim_sc3 <- SingleCellExperiment(assays = list(counts = sim_counts))
rowData(sim_sc3)$feature_symbol <- rownames(sim_counts)
sim_sc3 <- normalise(sim_sc3)
sim_sc3 <- sc3(sim_sc3, ks = 1:8, biology = FALSE, n_cores = 1)
```

sim_seurat *Seurat Clustered simulated scRNA-seq dataset*

Description

A simulated scRNA-seq dataset generated using the `splatter` package and clustered using `Seurat`.

Usage

```
sim_seurat
```

Format

`sim_seurat` is a Seurat::seurat object holding a simulated scRNA-seq dataset. The dataset has been clustered using the `Seurat` package with a range of values for the resolution parameter. The results of the clustering are held in the `meta.data` slot with columns named `res.X` where `X` is the value of the resolution parameter.

Source

```
# Simulation
library("splatter") # Version 1.2.1

sim <- splatSimulate(batchCells = 200, nGenes = 10000,
                     group.prob = c(0.4, 0.2, 0.2, 0.15, 0.05),
                     de.prob = c(0.1, 0.2, 0.05, 0.1, 0.05),
                     method = "groups", seed = 1)
sim_counts <- counts(sim)[1:1000, ]

library("Seurat") # Version 2.2.0
sim_seurat <- CreateSeuratObject(sim_counts)
sim_seurat <- NormalizeData(sim_seurat, display.progress = FALSE)
sim_seurat <- FindVariableGenes(sim_seurat, do.plot = FALSE,
                                display.progress = FALSE)
sim_seurat <- ScaleData(sim_seurat, display.progress = FALSE)
sim_seurat <- RunPCA(sim_seurat, do.print = FALSE)
sim_seurat <- FindClusters(sim_seurat, dims.use = 1:6,
                           resolution = seq(0, 1, 0.1),
                           print.output = FALSE)
```

---

store_node_aes *Store node aesthetics*

Description

Store the names of node attributes to use as aesthetics as graph attributes

Usage

```
store_node_aes(graph, node_aes_name, node_aes, node_aes_aggr, metadata)
```

Arguments

graph	graph to store attributes in
node_aes_name	name of the aesthetic to store
node_aes	value of the aesthetic to store
node_aes_aggr	name of an aggregation function associated with the aesthetic to store
metadata	data.frame containing metadata that can be used as aesthetics

Value

graph with additional attributes

Index