Package ‘sccore’

September 30, 2020

Title  Core Utilities for Single-Cell RNA-Seq
Version  0.1.0
Description  Core utilities for single-cell RNA-seq data analysis. Contained within are utility functions for working with differential expression (DE) matrices and count matrices, a collection of functions for manipulating and plotting data via 'ggplot2', and functions to work with cell graphs and cell embeddings. Graph-based methods include embedding kNN cell graphs into a UMAP <doi:10.21105/joss.00861>, collapsing vertices of each cluster in the graph, and propagating graph labels.
License  GPL-3
Encoding  UTF-8
LazyData  true
Imports  dplyr, ggplot2, ggrepel, graphics, grDevices, igraph, magrittr, Matrix, methods, pROC, parallel, Rcpp, rlang, scales, tibble, utils, uwot, withr
Depends  R (>= 3.5.0)
Suggests  ggrastr (>= 0.1.7), pbapply, rmumps, testthat
RoxygenNote  7.1.1
LinkingTo  Rcpp, RcppArmadillo, RcppProgress, RcppEigen
NeedsCompilation  yes
URL  https://github.com/kharchenkolab/sccore
BugReports  https://github.com/kharchenkolab/sccore/issues
Author  Peter Kharchenko [aut], Viktor Petukhov [aut], Ramus Rydbirk [aut], Evan Biederstedt [aut, cre]
Maintainer  Evan Biederstedt <evan.biederstedt@gmail.com>
Repository  CRAN
Date/Publication  2020-09-30 10:00:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>adjacentVertices</td>
<td>3</td>
</tr>
<tr>
<td>adjacent_vertex_weights</td>
<td>3</td>
</tr>
<tr>
<td>appendSpecificityMetricsToDE</td>
<td>4</td>
</tr>
<tr>
<td>as_factor</td>
<td>5</td>
</tr>
<tr>
<td>cellAnnotations</td>
<td>5</td>
</tr>
<tr>
<td>collapseGraphPaga</td>
<td>6</td>
</tr>
<tr>
<td>collapseGraphSum</td>
<td>6</td>
</tr>
<tr>
<td>colSumByFac</td>
<td>7</td>
</tr>
<tr>
<td>conosClusterList</td>
<td>7</td>
</tr>
<tr>
<td>conosGraph</td>
<td>8</td>
</tr>
<tr>
<td>dotPlot</td>
<td>8</td>
</tr>
<tr>
<td>embeddingColorsPlot</td>
<td>10</td>
</tr>
<tr>
<td>embeddingGroupPlot</td>
<td>11</td>
</tr>
<tr>
<td>embeddingPlot</td>
<td>12</td>
</tr>
<tr>
<td>embedGraphUmap</td>
<td>15</td>
</tr>
<tr>
<td>embedKnnGraph</td>
<td>16</td>
</tr>
<tr>
<td>extendMatrix</td>
<td>18</td>
</tr>
<tr>
<td>fac2col</td>
<td>18</td>
</tr>
<tr>
<td>fac2palette</td>
<td>19</td>
</tr>
<tr>
<td>getClusterGraph</td>
<td>20</td>
</tr>
<tr>
<td>get_nearest_neighbors</td>
<td>21</td>
</tr>
<tr>
<td>graphToAdjList</td>
<td>22</td>
</tr>
<tr>
<td>jsDist</td>
<td>22</td>
</tr>
<tr>
<td>mergeCountMatrices</td>
<td>23</td>
</tr>
<tr>
<td>multi2dend</td>
<td>24</td>
</tr>
<tr>
<td>plapply</td>
<td>24</td>
</tr>
<tr>
<td>propagateLabels</td>
<td>25</td>
</tr>
<tr>
<td>propagateLabelsDiffusion</td>
<td>26</td>
</tr>
<tr>
<td>propagateLabelsSolver</td>
<td>27</td>
</tr>
<tr>
<td>propagate_labels</td>
<td>28</td>
</tr>
<tr>
<td>setMinMax</td>
<td>29</td>
</tr>
<tr>
<td>smooth_count_matrix</td>
<td>29</td>
</tr>
<tr>
<td>sn</td>
<td>30</td>
</tr>
<tr>
<td>splitVectorByNodes</td>
<td>31</td>
</tr>
<tr>
<td>styleEmbeddingPlot</td>
<td>32</td>
</tr>
<tr>
<td>umapEmbedding</td>
<td>33</td>
</tr>
<tr>
<td>val2ggcol</td>
<td>33</td>
</tr>
</tbody>
</table>
adjacentVertices

List of adjacent vertices from igraph object

Description

List of adjacent vertices from igraph object

Usage

adjacentVertices(edge_verts)

Arguments

edge_verts edge vertices of igraph graph object

Value

list of adjacent vertices

Examples

## Not run:
edges <- igraph::as_edgelist(conosGraph)
adjacentVertices(edges)

## End(Not run)

adjacent_vertex_weights

List of adjacent vertex weights from igraph object

Description

List of adjacent vertex weights from igraph object

Usage

adjacent_vertex_weights(edge_verts, edge_weights)

Arguments

dge_verts edge vertices of igraph graph object
dge_weights edge weights of igraph graph object

Value

list of adjacent vertices
Examples

```r
## Not run:
edges <- igraph::as_edgelist(conosGraph)
edge.weights <- igraph::edge.attributes(conosGraph)$weight
adjacent_vertex_weights(edges, edge.weights)
## End(Not run)
```

appendSpecificityMetricsToDE

*Append specificity metrics to DE*

Description

Append specificity metrics to DE

Usage

```r
appendSpecificityMetricsToDE(
  de.df,
  clusters,
  cluster.id,
  p2.counts,
  low.expression.threshold = 0,
  append.auc = FALSE
)
```

Arguments

- `de.df`: data.frame of differential expression values
- `clusters`: factor of clusters
- `cluster.id`: names of `clusters` factor. If a cluster.id doesn’t exist in cluster names, an error is thrown.
- `p2.counts`: counts from Pagoda2, refer to <https://github.com/kharchenkolab/pagoda2>
- `low.expression.threshold`: numeric Threshold to remove expression values (default=0). Values under this threshold are discarded.
- `append.auc`: boolean If TRUE, append AUC values (default=FALSE)

Value

Data frame of differential expression values with metrics attached
as_factor

**Description**

convert character vector into a factor with names "values" and "levels"

**Usage**

as_factor(vals)

**Arguments**

vals vector of values to evaluate

**Value**

factor with names "values" and "levels"

cellAnnotations

**Description**

Conos cell annotations

**Usage**

cellAnnotations

**Format**

An object of class character of length 3000.
### collapseGraphPaga

**Description**


**Usage**

```
collapseGraphPaga(graph, groups, linearize = TRUE, winsorize = FALSE)
```

**Arguments**

- **graph**
  - igraph graph object Graph to be collapsed
- **groups**
  - factor on vertices describing cluster assignment (can specify integer vertex ids, or character vertex names which will be matched)
- **linearize**
  - should normally be always TRUE (default=TRUE)
- **winsorize**
  - winsorize final connectivity statistics value (default=FALSE) Note: Original PAGA has it as always TRUE, but in this case there is no way to distinguish level of connectivity for highly connected groups.

**Value**

collapsed graph

### collapseGraphSum

**Description**

Collapse Graph By Sum

**Usage**

```
collapseGraphSum(graph, groups, normalize = TRUE)
```

**Arguments**

- **graph**
  - igraph graph object Graph to be collapsed
- **groups**
  - factor on vertices describing cluster assignment (can specify integer vertex ids, or character vertex names which will be matched)
- **normalize**
  - boolean Whether to recalculate edge weight as observed/expected (default=TRUE)
colSumByFac

Value

Collapsed graph

Examples

collapsed = collapseGraphPaga(conosGraph, igraph::V(conosGraph), linearize=TRUE, winsorize=FALSE)

<table>
<thead>
<tr>
<th>colSumByFac</th>
<th>Calculates factor-stratified sums for each column</th>
</tr>
</thead>
</table>

Description

Calculates factor-stratified sums for each column

Usage

colSumByFac(sY, rowSel)

Arguments

- sY: sparse matrix (dgCmatrix)
- rowSel: integer factor. Note that the 0-th column will return sums for any NA values; 0 or negative values will be omitted

Value

Matrix

<table>
<thead>
<tr>
<th>conosClusterList</th>
<th>Conos clusters list</th>
</tr>
</thead>
</table>

Description

Conos clusters list

Usage

conosClusterList

Format

An object of class list of length 2.
Conos graph

Description

Conos graph

Usage

conosGraph

Format

An object of class igraph of length 10.

dotPlot

Dot plot adapted from Seurat:::DotPlot, see ?Seurat:::DotPlot for details

Description

Dot plot adapted from Seurat:::DotPlot, see ?Seurat:::DotPlot for details

Usage

dotPlot(
  markers,
  count.matrix,
  cell.groups,
  marker.colour = "black",
  cluster.colour = "black",
  xlab = "Marker",
  ylab = "Cluster",
  n.cores = 1,
  text.angle = 45,
  gene.order = NULL,
  cols = c("blue", "red"),
  col.min = -2.5,
  col.max = 2.5,
  dot.min = 0,
  dot.scale = 6,
  scale.by = "radius",
  scale.min = NA,
  scale.max = NA,
  verbose = TRUE,
  ...
)
## Arguments

- **markers**: Vector of gene markers to plot.
- **count.matrix**: Merged count matrix.
- **cell.groups**: Named factor containing cell groups (clusters) and cell names.
- **marker.colour**: Character or numeric vector (default="black").
- **cluster.colour**: Character or numeric vector (default="black").
- **xlab**: string X-axis title (default="Marker").
- **ylab**: string Y-axis title (default="Cluster").
- **n.cores**: integer Number of cores (default=1).
- **text.angle**: numeric Angle of text displayed (default=45).
- **gene.order**: Either factor of genes passed to dplyr::mutate(levels=gene.order), or a boolean. (default=NULL) If TRUE, gene.order is set to the unique markers. If FALSE, gene.order is set to NULL. If NULL, the argument is ignored.
- **cols**: Colors to plot (default=c("blue", "red")). The name of a palette from 'RColorBrewer::brewer.pal.info', a pair of colors defining a gradient, or 3+ colors defining multiple gradients (if 'split.by' is set).
- **col.min**: numeric Minimum scaled average expression threshold (default=-2.5). Everything smaller will be set to this.
- **col.max**: numeric Maximum scaled average expression threshold (default=2.5). Everything larger will be set to this.
- **dot.min**: numeric The fraction of cells at which to draw the smallest dot (default=0). All cell groups with less than this expressing the given gene will have no dot drawn.
- **dot.scale**: numeric Scale the size of the points, similar to cex (default=6)
- **scale.by**: string Scale the size of the points by 'size' or by 'radius' (default="radius")
- **scale.min**: numeric Set lower limit for scaling, use NA for default (default=NA)
- **scale.max**: numeric Set upper limit for scaling, use NA for default (default=NA)
- **verbose**: boolean Verbose output (default=TRUE)
- **...**: Additional inputs passed to sccore::plapply(), see man for description.

## Value

- **ggplot2 object**
embeddingColorsPlot

Description

Set colors for embedding plot. Used primarily in embeddingPlot().

Usage

embeddingColorsPlot(
  plot.df,
  colors,
  groups = NULL,
  geom_point_w = ggplot2::geom_point,
  gradient.range.quantile = 1,
  color.range = "symmetric",
  legend.title = NULL,
  palette = NULL
)

Arguments

plot.df data.frame for plotting. In embeddingPlot(), this is a tibble from tibble::rownames_to_column().

colors vector of numbers, which must be shown with point colors, names contain cell names (default=NULL). This argument is ignored if groups are provided.

groups vector of cluster labels, names contain cell names (default=NULL)

geom_point_w function to work with geom_point layer from ggplot2 (default=ggplot2::geom_point)

gradient.range.quantile Winsorization quantile for the numeric colors and gene gradient (default=1)

color.range controls range, in which colors are estimated (default="symmetric"). Pass "all" to estimate range based on all values of "colors", pass "data" to estimate it only based on colors, presented in the embedding. Alternatively you can pass vector of length 2 with (min, max) values.

legend.title legend title (default=NULL)

palette function, which accepts number of colors and return list of colors (i.e. see 'colorRampPalette') (default=NULL)

Value

ggplot2 object
embeddingGroupPlot

Plotting function for cluster labels, names contain cell names. Used primarily in embeddingPlot().

Description

Plotting function for cluster labels, names contain cell names. Used primarily in embeddingPlot().

Usage

`embeddingGroupPlot(
  plot.df,
  groups,
  geom_point_w,
  min.cluster.size,
  mark.groups,
  font.size,
  legend.title,
  shuffle.colors,
  palette,
  ...
)

Arguments

- `plot.df`: data.frame for plotting. In embeddingPlot(), this is a tibble from tibble::rownames_to_column().
- `groups`: vector of cluster labels, names contain cell names (default=NULL)
- `geom_point_w`: function to work with geom_point layer from ggplot2 (default=ggplot2::geom_point)
- `min.cluster.size`: labels for all groups with number of cells fewer than this parameter are considered as missed (default=0). This argument is ignored if groups aren’t provided
- `mark.groups`: plot cluster labels above points (default=TRUE)
- `font.size`: font size for cluster labels (default=c(3, 7)). It can either be single number for constant font size or pair (min, max) for font size depending on cluster size
- `legend.title`: legend title (default=NULL)
- `shuffle.colors`: shuffle colors (default=FALSE)
- `palette`: function, which accepts number of colors and return list of colors (i.e. see 'colorRampPalette’) (default=NULL)
- `...`: Additional arguments passed to ggplot2::geom_label_repel()

Value

ggplot2 object
embeddingPlot

Plot embedding with provided labels / colors using ggplot2

Usage

embeddingPlot(
  embedding,
  groups = NULL,
  colors = NULL,
  subgroups = NULL,
  plot.na = is.null(subgroups),
  min.cluster.size = 0,
  mark.groups = TRUE,
  show.legend = FALSE,
  alpha = 0.4,
  size = 0.8,
  title = NULL,
  plot.theme = NULL,
  palette = NULL,
  color.range = "symmetric",
  font.size = c(3, 7),
  show.ticks = FALSE,
  show.labels = FALSE,
  legend.position = NULL,
  legend.title = NULL,
  gradient.range.quantile = 1,
  raster = FALSE,
  raster.dpi = 300,
  shuffle.colors = FALSE,
  keep.limits = !is.null(subgroups),
  ...
)

Arguments

embedding two-column matrix with x and y coordinates of the embedding, rownames contain cell names and are used to match coordinates with groups or colors

groups vector of cluster labels, names contain cell names (default=NULL)

colors vector of numbers, which must be shown with point colors, names contain cell names (default=NULL). This argument is ignored if groups are provided.

subgroups subset of ‘groups’, selecting the cells for plot (default=NULL). Ignored if ‘groups’ is NULL.
embeddingPlot

plot.na boolean whether to plot points, for which groups / colors are missed (default=FALSE). This argument is FALSE if 'subgroups' is NULL

min.cluster.size labels for all groups with number of cells fewer than this parameter are considered as missed (default=0). This argument is ignored if groups aren’t provided

mark.groups plot cluster labels above points (default=TRUE)

show.legend show legend (default=FALSE)

alpha opacity level [0, 1] (default=0.4)

size point size (default=0.8)

title plot title (default=NULL)

plot.theme theme for the plot (default=NULL)

palette function, which accepts number of colors and return list of colors (i.e. see 'colorRampPalette') (default=NULL)

color.range controls range, in which colors are estimated (default="symmetric"). Pass "all" to estimate range based on all values of "colors", pass "data" to estimate it only based on colors, presented in the embedding. Alternatively you can pass vector of length 2 with (min, max) values.

font.size font size for cluster labels (default=c(3, 7)). It can either be single number for constant font size or pair (min, max) for font size depending on cluster size

show.ticks show ticks and tick labels (default=FALSE)

show.labels show labels (default=FALSE)

legend.position vector with (x, y) positions of the legend (default=NULL)

legend.title legend title (default=NULL)

gradient.range.quantile Winsorization quantile for the numeric colors and gene gradient (default=1)

raster boolean whether layer with the points be rasterized (default=FALSE). Setting of this argument to TRUE is useful when you need to export a plot with large number of points

raster.dpi dpi of the rasterized plot. (default=300). Ignored if raster == FALSE.

shuffle.colors shuffle colors (default=FALSE)

keep.limits Keep axis limits from original plot (default=!is.null(subgroups)). Useful when plotting subgroups, only meaningful it plot.na=FALSE

... Arguments passed on to ggrepel::geom_label_repel

mapping Set of aesthetic mappings created by aes or aes_. If specified and inherit.aes = TRUE (the default), is combined with the default mapping at the top level of the plot. You only need to supply mapping if there isn’t a mapping defined for the plot.

data A data frame. If specified, overrides the default data frame defined at the top level of the plot.

stat The statistical transformation to use on the data for this layer, as a string.
position Position adjustment, either as a string, or the result of a call to a position adjustment function.

parse If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath

box.padding Amount of padding around bounding box, as unit or number. Defaults to 0.25. (Default unit is lines, but other units can be specified by passing unit(x,"units").)

label.padding Amount of padding around label, as unit or number. Defaults to 0.25. (Default unit is lines, but other units can be specified by passing unit(x,"units").)

point.padding Amount of padding around labeled point, as unit or number. Defaults to 0. (Default unit is lines, but other units can be specified by passing unit(x,"units").)

label.r Radius of rounded corners, as unit or number. Defaults to 0.15. (Default unit is lines, but other units can be specified by passing unit(x,"units").)

label.size Size of label border, in mm.

segment.colour Colour of the line segment. Defaults to the same colour as the text. In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

segment.color Colour of the line segment. Defaults to the same colour as the text. In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

segment.size Width of line segment connecting the data point to the text label, in mm.

segment.alpha Transparency of the line segment. Defaults to the same transparency as the text.

min.segment.length Skip drawing segments shorter than this, as unit or number. Defaults to 0.5. (Default unit is lines, but other units can be specified by passing unit(x,"units").)

arrow specification for arrow heads, as created by arrow

force Force of repulsion between overlapping text labels. Defaults to 1.

max.iter Maximum number of iterations to try to resolve overlaps. Defaults to 2000.

nudge_x Horizontal and vertical adjustments to nudge the starting position of each text label.

nudge_y Horizontal and vertical adjustments to nudge the starting position of each text label.

xlab Limits for the x and y axes. Text labels will be constrained to these limits.

By default, text labels are constrained to the entire plot area.

ylab Limits for the x and y axes. Text labels will be constrained to these limits.

By default, text labels are constrained to the entire plot area.

na.rm If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.

direction "both", "x", or "y" – direction in which to adjust position of labels

seed Random seed passed to set.seed. Defaults to NA, which means that set.seed will not be called.
inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

Value

ggplot2 object

Examples

library(sccore)
embeddingPlot(umapEmbedding, show.ticks=TRUE, show.labels=TRUE, title="UMAP embedding")

embedGraphUmap


Description


Usage

embedGraphUmap(
  graph,
  min.prob = 0.001,
  min.visited.verts = 1000,
  n.cores = 1,
  max.hitting.nn.num = 0,
  max.commute.nn.num = 0,
  min.prob.lower = 1e-07,
  n.neighbors = 40,
  n.epochs = 1000,
  spread = 15,
  min.dist = 0.001,
  return.all = FALSE,
  n.sgd.cores = n.cores,
  verbose = TRUE,
  ...
)
embedKnnGraph

Embed a k-nearest neighbor (kNN) graph within a UMAP. Used within embedGraphUmap(). Please see McInnes et al <doi:10.21105/joss.00861> for the UMAP description and implementation.

**Arguments**

- `graph`: input igraph object
- `min.prob`: numeric Minimum probability for proximity when calculating hitting time per neighbors (default=1e-3)
- `min.visited.verts`: numeric Minimum number of vertices visited when calculating hitting time per neighbors (default=1000)
- `n.cores`: numeric Number of cores to use (default=1)
- `max.hitting.nn.num`: numeric Maximum adjacencies for calculating hitting time per neighbor, hitting_time_per_neighbors() (default=0)
- `max.commute.nn.num`: numeric Maximum adjacencies for calculating commute time per neighbor, commute_time_per_node() (default=0)
- `min.prob.lower`: numeric Probability threshold to continue iteration in depth first search hitting time, dfs_hitting_time() (default=1e-7)
- `n.neighbors`: numeric Number of neighbors (default=40)
- `n.epochs`: numeric Number of epochs to use during the optimization of the embedded coordinates (default=1000). See ‘n_epochs’ in uwot::umap()
- `spread`: numeric The effective scale of embedded points (numeric default=15). See ‘spread’ in uwot::umap()
- `min.dist`: numeric The effective minimum distance between embedded points (default=0.001). See ‘min.dist’ in uwot::umap()
- `return.all`: boolean If TRUE, return list(adj.info=adj.info, commute.times=commute.times, umap=umap). Otherwise, just return UMAP(default=FALSE)
- `n.sgd.cores`: numeric Number of cores to use during stochastic gradient descent. If set to >1, then results will not be reproducible, even if ’set.seed’ is called with a fixed seed before running (default=n_threads) See ‘n_sgd_threads’ in uwot::umap()
- `verbose`: boolean Verbose output (default=TRUE)
- `...`: Additional arguments passed to embedKnnGraph()

**Value**

resulting UMAP embedding
**Description**

Embed a k-nearest neighbor (kNN) graph within a UMAP. Used within embedGraphUmap(). Please see McInnes et al <doi:10.21105/joss.00861> for the UMAP description and implementation.

**Usage**

```r
embedKnnGraph(
  commute.times,
  n.neighbors,
  names = NULL,
  n.cores = 1,
  n.epochs = 1000,
  spread = 15,
  min.dist = 0.001,
  n.sgd.cores = n.cores,
  target.dims = 2,
  verbose = TRUE,
  ...
)
```

**Arguments**

- `commute.times` graph commute times from get_nearest_neighbors(). The definition of commute_time(u, v) is the expected time starting at u = to reach v and then return to u.
- `n.neighbors` numeric Number of neighbors
- `names` vector of names for UMAP rownames (default=NULL)
- `n.cores` numeric Number of cores to use (except during stochastic gradient descent) (default=1). See 'n_threads' in uwot::umap()
- `n.epochs` numeric Number of epochs to use during the optimization of the embedded coordinates (default=1000). See 'n_epochs' in uwot::umap()
- `spread` numeric The effective scale of embedded points (numeric default=15). See 'spread' in uwot::umap()
- `min.dist` numeric The effective minimum distance between embedded points (default=0.001). See 'min.dist' in uwot::umap()
- `n.sgd.cores` numeric Number of cores to use during stochastic gradient descent. If set to > 1, then results will not be reproducible, even if 'set.seed' is called with a fixed seed before running (default=n.cores) See 'n_sgd_threads' in uwot::umap()
- `target.dims` numeric Dimensions for 'n_components' in uwot::umap(n_components=target.dims) (default=2)
- `verbose` boolean Verbose output (default=TRUE)
- ... arguments passed to uwot::umap()

**Value**

resulting kNN graph embedding within a UMAP
**extendMatrix**

*Extend matrix to include new columns in matrix*

**Description**

Extend matrix to include new columns in matrix

**Usage**

```
extendMatrix(mtx, col.names)
```

**Arguments**

- `mtx`: Matrix
- `col.names`: Columns that should be included in matrix

**Value**

Matrix with new columns but rows retained

**Examples**

```r
library(dplyr)
geneUnion <- lapply(conosClusterList, colnames) %>% Reduce(union, .)
extendMatrix(conosClusterList[[1]], col.names=geneUnion)
```

---

**fac2col**

*Utility function to translate a factor into colors*

**Description**

Utility function to translate a factor into colors

**Usage**

```
fac2col(
  x,
  s = 1,
  v = 1,
  shuffle = FALSE,
  min.group.size = 1,
  return.details = FALSE,
  unclassified.cell.color = "gray50",
  level.colors = NULL
)
```
Encodes logic of how to handle named-vector and functional palettes. Used primarily within embeddingGroupPlot()

Usage

fac2palette(groups, palette, unclassified.cell.color = "gray50")

Arguments

groups vector of cluster labels, names contain cell names
palette function, which accepts number of colors and return list of colors (i.e. see 'colorRampPalette')
unclassified.cell.color Color for unclassified cells (default='gray50')
getClusterGraph

Description

Collapse vertices belonging to each cluster in a graph

Usage

getClusterGraph(
  graph,
  groups,
  method = "sum",
  plot = FALSE,
  node.scale = 50,
  edge.scale = 50,
  edge.alpha = 0.3,
  seed = 1,
  ...
)

Arguments

- graph: igraph graph object Graph to be collapsed
- groups: factor on vertices describing cluster assignment (can specify integer vertex ids, or character vertex names which will be matched)
- method: string Method to be used, either "sum" or "paga" (default="sum")
- plot: boolean Whether to show collapsed graph plot (default=FALSE)
- node.scale: numeric Scaling to control value of 'vertex.size' in plot.igraph() (default=50)
- edge.scale: numeric Scaling to control value of 'edge.width' in plot.igraph() (default=50)
- edge.alpha: numeric Scaling to control value of 'alpha.f' in adjustcolor() within plot.igraph() (default=0.3)
- seed: numeric Set seed via set.seed() for plotting (default=1)
- ... arguments passed to collapseGraphSum()

Value

collapsed graph

Examples

cluster.graph = getClusterGraph(conosGraph, igraph::V(conosGraph))
**get_nearest_neighbors**  
*Get nearest neighbors method on graph*

### Description

Get nearest neighbors method on graph

### Usage

```r
get_nearest_neighbors(
  adjacency_list,
  transition_probabilities,
  n_verts = 0L,
  n_cores = 1L,
  min_prob = 0.001,
  min_visited_verts = 1000L,
  min_prob_lower = 1e-05,
  max_hitting_nn_num = 0L,
  max_commute_nn_num = 0L,
  verbose = TRUE
)
```

### Arguments

- `adjacency_list`  
  igraph adjacency list

- `transition_probabilities`  
  vector of transition probabilities

- `n_verts`  
  numeric Number of vertices (default=0)

- `n_cores`  
  numeric Number of cores to use (default=1)

- `min_prob`  
  numeric Minimum probability for proximity when calculating hitting time per neighbors (default=1e-3)

- `min_visited_verts`  
  numeric Minimum number of vertices visted when calculating hitting time per neighbors (default=1000)

- `min_prob_lower`  
  numeric Probability threshold to continue iteration in depth first search hitting time, dfs_hitting_time() (default=1e-5)

- `max_hitting_nn_num`  
  numeric Maximum adjacencies for calculating hitting time per neighbor, hitting_time_per_neighbors() (default=0)

- `max_commute_nn_num`  
  numeric Maximum adjacencies for calculating commute time per neighbor, commute_time_per_node() (default=0)

- `verbose`  
  boolean Whether to have verbose output (default=TRUE)
Value
list of commute times based on adjacencies

---

**graphToAdjList**
*Convert igraph graph into an adjacency list*

### Description
Convert igraph graph into an adjacency list

### Usage
```r
graphToAdjList(graph)
```

### Arguments
- `graph`: input igraph object

### Value
adjacency list, defined by list(idx=adj.list, probabilities=probs, names=edge.list.fact$levels)

### Examples
```r
library(dplyr)
edge.list.fact <- igraph::as_edgelist(conosGraph) %>% as_factor()
edge.list <- matrix(edge.list.fact$values, ncol=2)
n.nodes <- length(igraph::V(conosGraph))
splitVectorByNodes(edge.list[,1], edge.list[,2], n.nodes)
```

---

**jsDist**
*_Jensen–Shannon distance metric (i.e. the square root of the Jensen–Shannon divergence)* between the columns of a dense matrix `m`

### Description
Jensen–Shannon distance metric (i.e. the square root of the Jensen–Shannon divergence) between the columns of a dense matrix `m`

### Usage
```r
jsDist(m, ncores = 1L)
```
mergeCountMatrices

Arguments

- **m**: Input matrix
- **ncores**: integer Number of cores (default=1)

Value

Vectorized version of the lower triangle as an R distance object, stats::dist()

Examples

```r
ex = matrix(1:9, nrow = 3, ncol = 3)
j_dist(ex)
```

mergeCountMatrices

Merge list of count matrices into a common matrix, entering 0s for the missing entries

Description

Merge list of count matrices into a common matrix, entering 0s for the missing entries

Usage

```r
mergeCountMatrices(cms, transposed = FALSE, ...)
```

Arguments

- **cms**: List of count matrices
- **transposed**: boolean Indicate whether ‘cms’ is transposed, e.g. cells in rows and genes in columns (default=FALSE)
- **...**: Parameters for ‘plapply’ function

Value

A merged extended matrix, with 0s for missing entries

Examples

```r
mergeCountMatrices(conosClusterList, n.cores=1)
## 12 x 67388 sparse Matrix of class "dgCMatrix"
```
multi2dend

*Translate multilevel segmentation into a dendrogram, with the lowest level of the dendrogram listing the cells*

**Description**

Translate multilevel segmentation into a dendrogram, with the lowest level of the dendrogram listing the cells

**Usage**

```r
multi2dend(cl, counts, deep = FALSE, dist = "cor")
```

**Arguments**

- `cl`: igraph communities object, returned from igraph community detection functions
- `counts`: dgCmatrix of counts
- `deep`: boolean If TRUE, take (cl$memberships[,1]). Otherwise, uses as.integer(membership(cl)) (default=FALSE)
- `dist`: Distance metric used (default='cor'). Either 'cor' for the correlation distance in log10 space, or 'JS' for the Jensen–Shannon distance metric (i.e. the square root of the Jensen–Shannon divergence)

**Value**

resulting dendrogram

---

plapply

*Parallel, optionally verbose lapply. See ?parallel::mclapply for more info.*

**Description**

Parallel, optionally verbose lapply. See ?parallel::mclapply for more info.

**Usage**

```r
plapply(
  ..., 
  progress = FALSE, 
  n.cores = parallel::detectCores(), 
  mc.preschedule = FALSE 
)
```
propagateLabels

Arguments

... Additional arguments passed to mclapply(), lapply(), or pbapply::pblapply()
progress Show progress bar via pbapply (default=FALSE)
n.cores Number of cores to use (default=parallel::detectCores())
mc.preschedule See ?parallel::mclapply (default=FALSE) If TRUE then the computation is first divided to (at most) as many jobs are there are cores and then the jobs are started, each job possibly covering more than one value. If FALSE, then one job is forked for each value of X. The former is better for short computations or large number of values in X, the latter is better for jobs that have high variance of completion time and not too many values of X compared to mc.cores.

Value

list, as returned by lapply

Examples

square = function(x){ x**2 }
plapply(1:10, square, n.cores=1, progress=TRUE)

propagateLabels 

Estimate labeling distribution for each vertex, based on provided labels.

Description

Estimate labeling distribution for each vertex, based on provided labels.

Usage

propagateLabels(graph, labels, method = "diffusion", ...)

Arguments

graph igraph graph object
labels vector of factor or character labels, named by cell names, used in propagateLabelsSolver() and propagateLabelsDiffusion()
method string Type of propagation. Either 'diffusion' or 'solver'. (default='diffusion') 'solver' gives better result but has bad asymptotics, so it is inappropriate for datasets > 20k cells.
... additional arguments passed to either propagateLabelsSolver() or propagateLabelsDiffusion()

Value

matrix with distribution of label probabilities for each vertex by rows.
propagateLabelsDiffusion

Estimate labeling distribution for each vertex, based on provided labels using a Random Walk on graph

Description

Estimate labeling distribution for each vertex, based on provided labels using a Random Walk on graph

Usage

propagateLabelsDiffusion(
    graph,
    labels,
    max.iters = 100,
    diffusion.fading = 10,
    diffusion.fading.const = 0.1,
    tol = 0.025,
    fixed.initial.labels = TRUE,
    verbose = TRUE
)

Arguments

graph    igraph graph object Graph input
labels   vector of factor or character labels, named by cell names
max.iters integer Maximal number of iterations (default=100)
diffusion.fading numeric Constant used for diffusion on the graph, exp(-diffusion.fading * (edge_length + diffusion.fading.const)) (default=10.0)
diffusion.fading.const numeric Another constant used for diffusion on the graph, exp(-diffusion.fading * (edge_length + diffusion.fading.const)) (default=0.1)
tol      numeric Absolute tolerance as a stopping criteria (default=0.025)
fixed.initial.labels boolean Prohibit changes of initial labels during diffusion (default=TRUE)
verbose  boolean Verbose mode (default=TRUE)

Value

matrix from input graph, with labels propagated
propagateLabelsSolver

Examples

propagateLabelsDiffusion(conosGraph, labels=cellAnnotations)

Description


Usage

propagateLabelsSolver(graph, labels, solver = "mumps")

Arguments

graph igraph graph object Graph input
labels vector of factor or character labels, named by cell names
solver Method of solver to use (default="mumps"), either "Matrix" or "mumps" (i.e. "rmumps::Rmumps")

Value

result from Matrix::solve() or rmumps::Rmumps

Examples

propagateLabelsSolver(conosGraph, labels=cellAnnotations)
propagate_labels  

Label propagation

Description

Label propagation

Usage

propagate_labels(
  edge_verts,
  edge_weights,
  vert_labels,
  max_n_iters = 10L,
  verbose = TRUE,
  diffusion_fading = 10,
  diffusion_fading_const = 0.5,
  tol = 0.005,
  fixed_initial_labels = FALSE
)

Arguments

edge_verts  edge vertices of igraph graph object
edge_weights edge weights of igraph graph object
vert_labels vector of factor or character labels, named by cell names
max_n_iters integer Maximal number of iterations (default=10)
verbose boolean Verbose mode (default=TRUE)
diffusion_fading numeric Constant used for diffusion on the graph, exp(-diffusion.fading * (edge_length + diffusion.fading.const)) (default=10.0)
diffusion_fading_const numeric Another constant used for diffusion on the graph, exp(-diffusion.fading * (edge_length + diffusion.fading.const)) (default=0.5)
tol numeric Absolute tolerance as a stopping criteria (default=5e-3)
fixed_initial_labels boolean Prohibit changes of initial labels during diffusion (default=FALSE)

Value

matrix from input graph, with labels propagated
setMinMax

Set range for values in object. Changes values outside of range to min or max. Adapted from Seurat::MinMax

Description

Set range for values in object. Changes values outside of range to min or max. Adapted from Seurat::MinMax

Usage

setMinMax(obj, min, max)

Arguments

obj
Object to manipulate
min
Minimum value
max
Maximum value

Value

An object with the same dimensions as input but with altered range in values

Examples

example_matrix = matrix(rep(c(1:5), 3), 5)
setMinMax(example_matrix, 2, 4)

smooth_count_matrix

Smooth gene expression, used primarily within conos::correctGenes. Used to smooth gene expression values in order to better represent the graph structure. Use diffusion of expression on graph with the equation \( dv = \exp(-a \times (v + b)) \)

Description

Smooth gene expression, used primarily within conos::correctGenes. Used to smooth gene expression values in order to better represent the graph structure. Use diffusion of expression on graph with the equation \( dv = \exp(-a \times (v + b)) \)
Usage

```r
smooth_count_matrix(
  edge_verts,
  edge_weights,
  count_matrix,
  is_label_fixed,
  max_n_iters = 10L,
  diffusion_fading = 1,
  diffusion_fading_const = 0.1,
  tol = 0.001,
  verbose = TRUE,
  normalize = FALSE
)
```

Arguments

- `edge_verts`: edge vertices of igraph graph object
- `edge_weights`: edge weights of igraph graph object
- `count_matrix`: gene count matrix
- `is_label_fixed`: boolean Whether label is fixed
- `max_n_iters`: integer Maximal number of iterations (default=10)
- `diffusion_fading`: numeric Constant used for diffusion on the graph, \( \exp(-\text{diffusion.fading} \times (\text{edge.length} + \text{diffusion.fading.const})) \) (default=1.0)
- `diffusion_fading_const`: numeric Another constant used for diffusion on the graph, \( \exp(-\text{diffusion.fading} \times (\text{edge.length} + \text{diffusion.fading.const})) \) (default=0.1)
- `tol`: numeric Absolute tolerance as a stopping criteria (default=1e-3)
- `verbose`: boolean Verbose mode (default=FALSE)
- `normalize`: boolean Whether to normalize values (default=FALSE)

Value

matrix from input graph, with labels propagated

---

### `sn`

Set names equal to values, a `stats::setNames` wrapper function

Description

Set names equal to values, a `stats::setNames` wrapper function

Usage

```r
sn(x)
```
splitVectorByNodes

Arguments

x an object for which names attribute will be meaningful

Value

An object with names assigned equal to values

Examples

vec = c(1, 2, 3, 4)
sn(vec)

splitVectorByNodes

Description

splitVectorByNodes

Usage

splitVectorByNodes(vec, nodes, n.nodes)

Arguments

vec input vector to be divided
nodes nodes used to divide the vector 'vec' via split()
n.nodes numeric The number of nodes for splitting

Value

list from vec with names given by the nodes

Examples

adjList = graphToAdjList(conosGraph)
print(names(adjList))
## [1] "idx" "probabilities" "names"
length(adjList$names)
## [1] 12000
styleEmbeddingPlot

Set plot.theme, legend, ticks for embedding plot. Used primarily in embeddingPlot().

Description

Set plot.theme, legend, ticks for embedding plot. Used primarily in embeddingPlot().

Usage

styleEmbeddingPlot(
  gg,
  plot.theme = NULL,
  title = NULL,
  legend.position = NULL,
  show.legend = TRUE,
  show.ticks = TRUE,
  show.labels = TRUE,
  relabel.axis = TRUE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gg</td>
<td>ggplot2 object to plot</td>
</tr>
<tr>
<td>plot.theme</td>
<td>theme for the plot (default=NULL)</td>
</tr>
<tr>
<td>title</td>
<td>plot title (default=NULL)</td>
</tr>
<tr>
<td>legend.position</td>
<td>vector with (x, y) positions of the legend (default=NULL)</td>
</tr>
<tr>
<td>show.legend</td>
<td>show legend (default=TRUE)</td>
</tr>
<tr>
<td>show.ticks</td>
<td>show ticks and tick labels (default=TRUE)</td>
</tr>
<tr>
<td>show.labels</td>
<td>show labels (default=TRUE)</td>
</tr>
<tr>
<td>relabel.axis</td>
<td>boolean If TRUE, relabel axes with ggplot2::labs(x='Component 1', y='Component 2') (default=TRUE)</td>
</tr>
</tbody>
</table>

Value

ggplot2 object
**umapEmbedding**  
**UMAP embedding**

**Description**

UMAP embedding

**Usage**

`umapEmbedding`

**Format**

An object of class `matrix` (inherits from `array`) with 12000 rows and 2 columns.

---

**val2ggcol**

*Helper function to return a ggplot color gradient for a numeric vector*

`ggplot(aes(color=x, ...), ...) + val2ggcol(x)`

**Description**

Helper function to return a ggplot color gradient for a numeric vector `ggplot(aes(color=x, ...), ...) + val2ggcol(x)`

**Usage**

`val2ggcol(`

```r
values,
    gradient.range.quantile = 1,
    color.range = "symmetric",
    palette = NULL,
    midpoint = NULL,
    oob = scales::squish,
    return.fill = FALSE,
    ...
)
```

**Arguments**

- `values` values by which the color gradient is determined
- `gradient.range.quantile` numeric trimming quantile (default=1). Either a single number or two numbers - for lower and upper quantile.
color.range  either a vector of two values explicitly specifying the values corresponding to the start/end of the gradient, or string "symmetric" or "all" (default="symmetric"). "symmetric": range will fit data, but will be symmetrized around zeros, "all": gradient will match the span of the range of the data (after gradient.range.quantile)

palette an optional palette function (default=NULL). The default becomes blue-gray90-red; if the values do not straddle 0, then truncated gradients (blue-gray90 or gray90-red) will be used

midpoint optional midpoint (default=NULL). Set for the center of the resulting range by default

oob function to determine what to do with the values outside of the range (default = scales::squish). Refer to 'oob' parameter in ggplot

return.fill boolean Whether to return fill gradients instead of color (default=FALSE)

... additional arguments are passed to ggplot2::scale_color_gradient* functions, i.e. scale_color_gradient(), scale_color_gradient2(), scale_color_gradientn()

Value

 ggplot2::scale_colour_gradient* object
## Index

### *datasets*
- cellAnnotations, 5
- conosClusterList, 7
- conosGraph, 8
- umapEmbedding, 33
- adjacent_vertex_weights, 3
- adjacentVertices, 3
- aes, 13
- aes_, 13
- appendSpecificityMetricsToDE, 4
- arrow, 14
- as_factor, 5
- borders, 15
- cellAnnotations, 5
- collapseGraphPaga, 6
- collapseGraphSum, 6
- colSumByFac, 7
- conosClusterList, 7
- conosGraph, 8
- dotPlot, 8
- embeddingColorsPlot, 10
- embeddingGroupPlot, 11
- embeddingPlot, 12
- embedGraphUmap, 15
- embedKnnGraph, 16
- extendMatrix, 18
- fac2col, 18
- fac2palette, 19
- get_nearest_neighbors, 21
- getClusterGraph, 20
- ggrepel::geom_label_repel, 13
- graphToAdjList, 22
- jsDist, 22
- mergeCountMatrices, 23
- multi2dend, 24
- plapply, 24
- propagate_labels, 28
- propagateLabels, 25
- propagateLabelsDiffusion, 26
- propagateLabelsSolver, 27
- set.seed, 14
- setMinMax, 29
- smooth_count_matrix, 29
- sn, 30
- splitVectorByNodes, 31
- styleEmbeddingPlot, 32
- umapEmbedding, 33
- val2ggcol, 33