

Package ‘score’

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/score>

BugReports <https://github.com/kharchenkolab/score/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:

adjacentVertices	3
adjacent_vertex_weights	3
appendSpecificityMetricsToDE	4
as_factor	5
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
getClusterGraph	20
get_nearest_neighbors	21
graphToAdjList	22
jsDist	22
mergeCountMatrices	23
multi2dend	24
plapply	24
propagateLabels	25
propagateLabelsDiffusion	26
propagateLabelsSolver	27
propagate_labels	28
setMinMax	29
smooth_count_matrix	29
sn	30
splitVectorByNodes	31
styleEmbeddingPlot	32
umapEmbedding	33
val2ggcol	33

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

edge_verts edge vertices of igraph graph object
edge_weights edge weights of igraph graph object

Value

list of adjacent vertices

Examples

```
## 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

```
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	<i>convert character vector into a factor with names "values" and "levels"</i>
-----------	--

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	<i>Conos cell annotations</i>
-----------------	-------------------------------

Description

Conos cell annotations

Usage

```
cellAnnotations
```

Format

An object of class character of length 3000.

collapseGraphPaga	<i>Collapse graph using PAGA 1.2 algorithm, Wolf et al 2019, Genome Biology (2019) <https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1663-x></i>
-------------------	--

Description

Collapse graph using PAGA 1.2 algorithm, Wolf et al 2019, Genome Biology (2019) <<https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1663-x>>

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	<i>Collapse Graph By Sum</i>
------------------	------------------------------

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)

Value

collapsed graph

Examples

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

colSumByFac

Calculates factor-stratified sums for each column

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

conosClusterList

Conos clusters list

Description

Conos clusters list

Usage

```
conosClusterList
```

Format

An object of class list of length 2.

conosGraph	<i>Conos graph</i>
------------	--------------------

Description

Conos graph

Usage

```
conosGraph
```

Format

An object of class igraph of length 10.

dotPlot	<i>Dot plot adapted from Seurat:::DotPlot, see ?Seurat:::DotPlot for details</i>
---------	--

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

<code>markers</code>	Vector of gene markers to plot
<code>count.matrix</code>	Merged count matrix
<code>cell.groups</code>	Named factor containing cell groups (clusters) and cell names
<code>marker.colour</code>	Character or numeric vector (default="black")
<code>cluster.colour</code>	Character or numeric vector (default="black")
<code>xlab</code>	string X-axis title (default="Marker")
<code>ylab</code>	string Y-axis title (default="Cluster")
<code>n.cores</code>	integer Number of cores (default=1)
<code>text.angle</code>	numeric Angle of text displayed (default=45)
<code>gene.order</code>	Either factor of genes passed to <code>dplyr::mutate(levels=gene.order)</code> , or a boolean. (default=NULL) If TRUE, <code>gene.order</code> is set to the unique markers. If FALSE, <code>gene.order</code> is set to NULL. If NULL, the argument is ignored.
<code>cols</code>	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).
<code>col.min</code>	numeric Minimum scaled average expression threshold (default=-2.5). Everything smaller will be set to this.
<code>col.max</code>	numeric Maximum scaled average expression threshold (default=2.5). Everything larger will be set to this.
<code>dot.min</code>	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.
<code>dot.scale</code>	numeric Scale the size of the points, similar to <code>cex</code> (default=6)
<code>scale.by</code>	string Scale the size of the points by 'size' or by 'radius' (default="radius")
<code>scale.min</code>	numeric Set lower limit for scaling, use NA for default (default=NA)
<code>scale.max</code>	numeric Set upper limit for scaling, use NA for default (default=NA)
<code>verbose</code>	boolean Verbose output (default=TRUE)
<code>...</code>	Additional inputs passed to <code>score::plapply()</code> , see man for description.

Value

ggplot2 object

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

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	<i>Plotting function for cluster labels, names contain cell names. Used primarily in embeddingPlot().</i>
--------------------	---

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***Description**

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

<code>plot.na</code>	boolean whether to plot points, for which groups / colors are missed (default=FALSE). This argument is FALSE if 'subgroups' is NULL
<code>min.cluster.size</code>	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
<code>mark.groups</code>	plot cluster labels above points (default=TRUE)
<code>show.legend</code>	show legend (default=FALSE)
<code>alpha</code>	opacity level [0, 1] (default=0.4)
<code>size</code>	point size (default=0.8)
<code>title</code>	plot title (default=NULL)
<code>plot.theme</code>	theme for the plot (default=NULL)
<code>palette</code>	function, which accepts number of colors and return list of colors (i.e. see 'colorRampPalette') (default=NULL)
<code>color.range</code>	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.
<code>font.size</code>	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
<code>show.ticks</code>	show ticks and tick labels (default=FALSE)
<code>show.labels</code>	show labels (default=FALSE)
<code>legend.position</code>	vector with (x, y) positions of the legend (default=NULL)
<code>legend.title</code>	legend title (default=NULL)
<code>gradient.range.quantile</code>	Winsorization quantile for the numeric colors and gene gradient (default=1)
<code>raster</code>	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
<code>raster.dpi</code>	dpi of the rasterized plot. (default=300). Ignored if raster == FALSE.
<code>shuffle.colors</code>	shuffle colors (default=FALSE)
<code>keep.limits</code>	Keep axis limits from original plot (default=!is.null(subgroups)). Useful when plotting subgroups, only meaningful if plot.na=FALSE
<code>...</code>	Arguments passed on to <code>ggrepel::geom_label_repel</code>
	mapping Set of aesthetic mappings created by <code>aes</code> or <code>aes_</code> . If specified and <code>inherit.aes = TRUE</code> (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.

`xlim` 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.

`ylim` 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(score)
embeddingPlot(umapEmbedding, show.ticks=TRUE, show.labels=TRUE, title="UMAP embedding")
```

embedGraphUmap	<i>Embed a graph into a UMAP, Uniform Manifold Approximation and Projection for Dimension Reduction, <https://github.com/lmcinnes/umap>, <doi:10.21105/joss.00861></i>
----------------	---

Description

Embed a graph into a UMAP, Uniform Manifold Approximation and Projection for Dimension Reduction, <<https://github.com/lmcinnes/umap>>, <doi:10.21105/joss.00861>

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,
  ...
)
```

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 visted 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

embedKnnGraph	<i>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.</i>
---------------	--

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

```
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

<code>commute.times</code>	graph commute times from <code>get_nearest_neighbors()</code> . The definition of <code>commute_time(u, v)</code> is the expected time starting at <code>u</code> = to reach <code>v</code> and then return to <code>u</code> .
<code>n.neighbors</code>	numeric Number of neighbors
<code>names</code>	vector of names for UMAP rownames (default=NULL)
<code>n.cores</code>	numeric Number of cores to use (except during stochastic gradient descent) (default=1). See 'n_threads' in <code>uwot::umap()</code>
<code>n.epochs</code>	numeric Number of epochs to use during the optimization of the embedded coordinates (default=1000). See 'n_epochs' in <code>uwot::umap()</code>
<code>spread</code>	numeric The effective scale of embedded points (numeric default=15). See 'spread' in <code>uwot::umap()</code>
<code>min.dist</code>	numeric The effective minimum distance between embedded points (default=0.001). See 'min.dist' in <code>uwot::umap()</code>
<code>n.sgd.cores</code>	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 <code>uwot::umap()</code>
<code>target.dims</code>	numeric Dimensions for 'n_components' in <code>uwot::umap(n_components=target.dims)</code> (default=2)
<code>verbose</code>	boolean Verbose output (default=TRUE)
<code>...</code>	arguments passed to <code>uwot::umap()</code>

Value

resulting kNN graph embedding within a UMAP

extendMatrix	<i>Extend matrix to include new columns in matrix</i>
--------------	---

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

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

fac2col	<i>Utility function to translate a factor into colors</i>
---------	---

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
)
```

Arguments

x	input factor
s	numeric The "saturation" to be used to complete the HSV color descriptions (default=1) See ?rainbow in Palettes, grDevices
v	numeric The "value" to be used to complete the HSV color descriptions (default=1) See ?rainbow in Palettes, grDevices
shuffle	boolean If TRUE, shuffles columns with shuffle(columns) (default=FALSE)
min.group.size	integer Exclude groups of size less than the min.group.size (default=1)
return.details	boolean If TRUE, returns a list list(colors=y, palette=col). Otherwise, just returns the factor (default=FALSE)
unclassified.cell.color	Color for unclassified cells (default='gray50')
level.colors	(default=NULL)

Value

vector or list of colors

Examples

```
genes = factor(c("BRAF", "NPC1", "PAX3", "BRCA2", "FMR1"))
fac2col(genes)
```

fac2palette	<i>Encodes logic of how to handle named-vector and functional palettes. Used primarily within embeddingGroupPlot()</i>
-------------	--

Description

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')

Value

vector or palette

getClusterGraph	<i>Collapse vertices belonging to each cluster in a graph</i>
-----------------	---

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

```
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 visited when calculating hitting time per neighbors (default=1000)
min_prob_lower	numeric Probability threshold to continue iteration in depth first search hitting time, <code>dfs_hitting_time()</code> (default=1e-5)
max_hitting_nn_num	numeric Maximum adjacencies for calculating hitting time per neighbor, <code>hitting_time_per_neighbors()</code> (default=0)
max_commute_nn_num	numeric Maximum adjacencies for calculating commute time per neighbor, <code>commute_time_per_node()</code> (default=0)
verbose	boolean Whether to have verbose output (default=TRUE)

Value

list of commute times based on adjacencies

graphToAdjList	<i>Convert igraph graph into an adjacency list</i>
----------------	--

Description

Convert igraph graph into an adjacency list

Usage

```
graphToAdjList(graph)
```

Arguments

graph input igraph object

Value

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

Examples

```
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	<i>Jensen–Shannon distance metric (i.e. the square root of the Jensen–Shannon divergence) between the columns of a dense matrix m</i>
--------	--

Description

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

Usage

```
jsDist(m, ncores = 1L)
```

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

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

mergeCountMatrices	<i>Merge list of count matrices into a common matrix, entering 0s for the missing entries</i>
--------------------	---

Description

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

Usage

```
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

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

multi2dend	<i>Translate multilevel segmentation into a dendrogram, with the lowest level of the dendrogram listing the cells</i>
------------	---

Description

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

Usage

```
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	<i>Parallel, optionally verbose lapply. See ?parallel::mclapply for more info.</i>
---------	--

Description

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

Usage

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


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 as 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	<i>Estimate labeling distribution for each vertex, based on provided labels.</i>
-----------------	--

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.

Examples

```
propagateLabels(conosGraph, labels=cellAnnotations)
```

```
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, $\exp(-\text{diffusion.fading} * (\text{edge_length} + \text{diffusion.fading.const}))$ (default=10.0)
diffusion.fading.const	numeric Another constant used for diffusion on the graph, $\exp(-\text{diffusion.fading} * (\text{edge_length} + \text{diffusion.fading.const}))$ (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

Examples

```
propagateLabelsDiffusion(conosGraph, labels=cellAnnotations)
```

propagateLabelsSolver *Propagate labels using Zhu, Ghahramani, Lafferty (2003) algorithm, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions" <<http://mlg.eng.cam.ac.uk/zoubin/papers/zgl.pdf>>*

Description

Propagate labels using Zhu, Ghahramani, Lafferty (2003) algorithm, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions" <<http://mlg.eng.cam.ac.uk/zoubin/papers/zgl.pdf>>

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, $\exp(-\text{diffusion.fading} * (\text{edge_length} + \text{diffusion.fading.const}))$ (default=10.0)
diffusion_fading_const	numeric Another constant used for diffusion on the graph, $\exp(-\text{diffusion.fading} * (\text{edge_length} + \text{diffusion.fading.const}))$ (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	<i>Set range for values in object. Changes values outside of range to min or max. Adapted from Seurat::MinMax</i>
-----------	---

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	<i>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 $dv = \exp(-a * (v + b))$</i>
---------------------	---

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 $dv = \exp(-a * (v + b))$

Usage

```
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, $\exp(-\text{diffusion.fading} * (\text{edge_length} + \text{diffusion.fading.const}))$ (default=1.0)
diffusion_fading_const	numeric Another constant used for diffusion on the graph, $\exp(-\text{diffusion.fading} * (\text{edge_length} + \text{diffusion.fading.const}))$ (default=0.1)
tol	numeric Absolute tolerance as a stopping criteria (default=1e-3)
verbose	boolean Verbose mode (default=TRUE)
normalize	boolean Whether to normalize values (default=FALSE)

Value

matrix from input graph, with labels propagated

sn	<i>Set names equal to values, a stats::setNames wrapper function</i>
----	--

Description

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

Usage

```
sn(x)
```

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` *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	<i>Set plot.theme, legend, ticks for embedding plot. Used primarily in embeddingPlot().</i>
--------------------	---

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

gg	ggplot2 object to plot
plot.theme	theme for the plot (default=NULL)
title	plot title (default=NULL)
legend.position	vector with (x, y) positions of the legend (default=NULL)
show.legend	show legend (default=TRUE)
show.ticks	show ticks and tick labels (default=TRUE)
show.labels	show labels (default=TRUE)
relabel.axis	boolean If TRUE, relabel axes with ggplot2::labs(x='Component 1', y='Component 2') (default=TRUE)

Value

ggplot2 object

umapEmbedding	<i>UMAP embedding</i>
---------------	-----------------------

Description

UMAP embedding

Usage

```
umapEmbedding
```

Format

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

val2ggcol	<i>Helper function to return a ggplot color gradient for a numeric vector ggplot(aes(color=x, ...), ...) + val2ggcol(x)</i>
-----------	---

Description

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

Usage

```
val2ggcol(
  values,
  gradient.range.quantile = 1,
  color.range = "symmetric",
  palette = NULL,
  midpoint = NULL,
  oob = scales::squish,
  return.fill = FALSE,
  ...
)
```

Arguments

<code>values</code>	values by which the color gradient is determined
<code>gradient.range.quantile</code>	numeric Trimming quantile (default=1). Either a single number or two numbers - for lower and upper quantile.

<code>color.range</code>	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 <code>gradient.range.quantile</code>)
<code>palette</code>	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
<code>midpoint</code>	optional midpoint (default=NULL). Set for the center of the resulting range by default
<code>oob</code>	function to determine what to do with the values outside of the range (default = <code>scales::squish</code>). Refer to 'oob' parameter in <code>ggplot</code>
<code>return.fill</code>	boolean Whether to return fill gradients instead of color (default=FALSE)
<code>...</code>	additional arguments are passed to <code>ggplot2::scale_color_gradient*</code> functions, i.e. <code>scale_color_gradient()</code> , <code>scale_color_gradient2()</code> , <code>scale_color_gradientn()</code>

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