Package 'CellAnnotatoR'

November 25, 2019

Title Automated marker-based cell type annotation

Version 0.0.0.9000

Description Automated marker-based cell type annotation

Depends R (>= 3.5.1)

License What license is it under?

Encoding UTF-8

LazyData true

Imports conos (>= 1.1.2),
garnett (>= 0.1.13),
dplyr,
ggplot2,
igraph,
Matrix,
tibble,
magrittr,
cowplot,
Rcpp

Suggests ggtree (>= 1.99.1),
knitr,
rmarkdown,
ggpubr

Remotes hms-dbmi/conos,
cole-trapnell-lab/garnett,
YuLab-SMU/ggtree

RoxygenNote 6.1.1

VignetteBuilder knitr

LinkingTo Rcpp

R topics documented:

`arrangePlots` *Arrange Plots*

Description

Arrange Plots

Usage

```
arrangePlots(plot.list, build.panel, n.col = NULL, n.row = NULL)
```

Arguments

`plot.list`	list of plots
`build.panel`	combine individual plots to a single panel
`n.col`	number of columns in the panel
`n.row`	number of rows in the panel

Value

panel of plots if 'build.panel==T' or 'plot.list' otherwise

`assignCellsByScores` *Assign Cells By Scores*

Description

Assign cell types for each cell based on type scores. Optionally uses 'clusters' to expand annotation.

Usage

```
assignCellsByScores(graph, clf.data, score.info = NULL,
  clusters = NULL, verbose = 0, uncertainty.thresholds = c(coverage =
  0.5, negative = 0.5, positive = 0.75), ...)
```

Arguments

`graph`	cell graph from Seurat, Pagoda2 or some other tool. Can be either in igraph or adjacency matrix format. Use 'graph=NULL' to skip graph diffusion step and get raw score annotation (useful when debug marker genes).
`clf.data`	classification data from 'getClassificationData'
`score.info`	cell type scores from 'getMarkerScoreInfo' function. Re-estimated if NULL
`clusters`	vector with cluster labels named by cell ids. Used to expand annotation on these clusters.
`verbose`	verbosity level (from 0 to 2)
`...`	Arguments passed on to `diffuseScorePerType`

Value

list with parameters:

- annotation: annotation per level
- scores: assignment scores per level
- annotation.filt: the same as annotation, but cells, which don't pass QC are assigned to NA class

Examples

```
clf_data <- getClassificationData(cm, marker_path)
ann_by_level <- assignCellsByScores(graph, clf_data, clusters=clusters)
```

`classificationTreeToDf` *Classification Tree to DataFrame*

Description

Classification Tree to DataFrame

Usage

```
classificationTreeToDf(classification.tree)
```

Arguments

`classification.tree`	cell type hierarchy represented by graph. Part of 'clf_data' object from 'getClassificationData'

`createClassificationTree` *Create Classification Tree*

Description

creates graph object for cell type hierarchy

Usage

```
createClassificationTree(marker.list)
```

Arguments

`marker.list`	list of markers per cell type. Can be obtained with 'parseMarkerFile'

Value

igraph graph with the type hierarhcy

Examples

```
markers <- parseMarkerFile(marker_path)
clf_tree <- createClassificationTree(markers)
```

deriveHierarchy *Derive Hierarchy*

Description

derive hierarchy from the data using hclust

Usage

```
deriveHierarchy(feature.matrix, annotation, dist.method = "cor",
  max.depth = 2)
```

Arguments

feature.matrix	matrix where rows represent cells and columns represent either genes or some embedded space (e.g. PCA)
annotation	vector with cell type label per cell
dist.method	method for pairwise distance estimation. Either "cor" for correlation distance or any method supported by ‘dist‘ function
max.depth	maximal depth of the hierarchy

Value

list with cell type hierarchy

Examples

```
hierarchy <- deriveHierarchy(pca_mtx, annotation, max.depth=3)
clf_tree <- hierarchyToClassificationTree(hierarchy)
plotTypeHierarchy(clf_tree)
```

diffuseGraph *Diffuse Graph*

Description

Run diffusion on graph

Usage

```
diffuseGraph(graph, scores, fading = 10, fading.const = 0.5,
  score.fixing.threshold = 0.8, verbose = FALSE, max.iters = 1000,
  tol = 0.001)
```

Arguments

graph	graph to diffuse on
scores	table of scores
fading	fading level for graph diffusion
fading.const	constant in exponent for graph diffusion
score.fixing.threshold	threshold for a label to be considered certain. Such labels can't be changed during diffusion.
verbose	print progress bar
tol	tolerance for diffusion stopping

---

extractMarkersFromSubtypes

Extract Markers From Subtypes

Description

Extract Markers From Subtypes

Usage

```
extractMarkersFromSubtypes(parent.type = "root", clf.data = NULL,
  clf.tree = NULL, marker.list = NULL, count.matrix = NULL,
  max.depth = NULL, drop.missing = T, marker.type = c("expressed",
  "not_expressed"))
```

Arguments

parent.type	cell type for which the markers of the subtypes must be plotted

---

getAnnotationPerParent

Get Annotation Per Parent

Description

for each cell type get annotation of its subtypes on the next hierarchy level

Usage

```
getAnnotationPerParent(classification.tree, annotation)
```

Arguments

classification.tree	cell type hierarchy represented by graph. Part of 'clf_data' object from 'getClassificationData'
annotation	annotation for high-resolution. Cell type names must correspond to nodes in the 'classification.tree'

Value

list of sub-annotations named by parent types

Examples

```
ann_by_parent <- getAnnotationPerParent(clf_data$classification.tree, annotation)
```

---

getCellTypeScoreInfo *Get Cell Type Score Info*

Description

estimate info, neccessary for scoring of cells by cell types for the specified cell type

Usage

```
getCellTypeScoreInfo(markers, tf.idf, aggr = T)
```

Arguments

markers	element of marker list. List with fields 'expressed' and 'not_expressed'
tf.idf	TF-IDF normalized matrix. Can be obtained with 'normalizeTfIdfWithFeatures'
aggr	if scores must be aggregated for the whole cell type or returned for each marker separately

Value

list with score info:

- scores.raw: scores from positive markers
- mult: score multiplier, estimated with negative markers
- max.positive: maximal expression of positive markers. Used for estimation of negative scores
- scores: final scores. Equal to 'scores * score.mult'

getClassificationData *Get Classification data*

Description

prepare information neccessary for cell type classification

Usage

```
getClassificationData(cm, markers, prenormalized = F,
  data.gene.id.type = "SYMBOL", marker.gene.id.type = "SYMBOL",
  db = NULL, verbose = F)
```

Arguments

cm	gene count matrix with cells by columns and genes by rows. May be in raw, TC-normalized or tf-idf-normalized format (in case of tf-idf, 'prenormalized' must be set to 'T')
markers	path to the file with marker genes or parsed marker list from 'parseMarkerFile' function
prenormalized	is 'cm' in tf-idf-normalized format? Default: FALSE.

getMarkerScoreInfo *Get Marker Score Info*

Description

estimate info, neccessary for scoring of cells by cell types

Usage

```
getMarkerScoreInfo(clf.data, ...)
```

Arguments

clf.data	classification data from 'getClassificationData'
...	Arguments passed on to getCellTypeScoreInfo
	aggr if scores must be aggregated for the whole cell type or returned for each marker separately

Value

List of score info for each cell type. See 'CellAnnotatoR:::getCellTypeScoreInfo' for more info

`getMarkerScoresPerCellType` *Return initial scores of each cell type for each cell*

Description

Return initial scores of each cell type for each cell

Usage

```
getMarkerScoresPerCellType(clf.data, score.info = NULL, aggr = T)
```

Arguments

`clf.data`	classification data from 'getClassificationData'
`score.info`	pre-calculated score info from 'getMarkerScoreInfo'
`aggr`	should individual gene scores be aggregated per cell type? If 'FALSE', returns list of data.frames, showing scores of each gene for each cell. Useful for debugging list of markers.

Value

data.frame with rows corresponding to cells and columns corresponding to cell types. Values are cell type scores, normalized per level of hierarchy

`hierarchyToClassificationTree` *Hierarchy to Classification Tree*

Description

Convert list with cell type hierarchy to classification tree in igraph format

Usage

```
hierarchyToClassificationTree(hierarchy)
```

Arguments

`hierarchy`	list of cell types, where each element represent a cell type. If cell type has some subtypes it's represented as another list, otherwise it's just a string

Value

igraph graph with the type hierarhcy

Examples

```
hierarchy <- list(
  Alveolar=c("AT1 Cell", "AT2 Cell"),
  B=c("B Cell", "Ig-producing B cell"),
  NK_T=list(`T`=c("Dividing T cells", "T Cell_Cd8b1 high", "Nuocyte"), "NK Cell"),
  "Endothelial"
)

clf_tree <- hierarchyToClassificationTree(hierarchy)
```

---

markerListToMarkup *Marker List to Markup*

Description

Convert marker list to the markup language and optionally save it to a file

Usage

```
markerListToMarkup(marker.list, file = "", group.by.parent = T)
```

Arguments

`marker.list`	list of markers per cell type. Can be obtained with 'parseMarkerFile'
`file`	file to save to. If empty string, returns markup text instead of saving
`group.by.parent`	group cell types in the markup by the parent type

Value

path to the file if 'file == ""' or markupt text otherwise

Examples

```
markerListToMarkup(clf_data$marker.list, file="markers.txt")
```

`mergeAnnotationByLevels` *Merge Annotation By Levels*

Description

merge provided annotation using ‘classification.tree‘ to get annotations for different levels of hierarchy

Usage

```
mergeAnnotationByLevels(annotation, classification.tree)
```

Arguments

`annotation`	annotation for high-resolution. Cell type names must correspond to nodes in the ‘classification.tree‘
`classification.tree`	cell type hierarchy represented by graph. Part of ‘clf_data‘ object from ‘getClassificationData‘

Value

list of annotations where each element correspond to some hierarchy level

Examples

```
ann_by_level <- mergeAnnotationByLevels(annotation, clf_data$classification.tree)
```

`mergeAnnotationToLevel` *Merge Annotation to Level*

Description

Merge Annotation to Level

Usage

```
mergeAnnotationToLevel(level, annotation, classification.tree)
```

Arguments

`annotation`	annotation for high-resolution. Cell type names must correspond to nodes in the ‘classification.tree‘

mergeScoreInfos *Merge Score Infos*

Description

merge score information from multiple datasets

Usage

```
mergeScoreInfos(score.infos, verbose = F)
```

Arguments

score.infos	scoring info per cell per cell type
verbose	show progress bar

Value

List of score info for each cell type. See 'CellAnnotatoR:::getCellTypeScoreInfo' for more info

Examples

```
clf_datas <- lapply(cms, getClassificationData, marker_path)
score_infos <- lapply(clf_datas, getMarkerScoreInfo)
all_score_info <- mergeScoreInfos(score_infos, verbose=T)
```

mergeScores *Merge Scores*

Description

Merge Scores

Usage

```
mergeScores(score.name, score.infos, aggr.func = c)
```

Arguments

score.name	type of score to merge
score.infos	scoring info per cell per cell type

normalizeTfIdfWithFeatures *Normalize TF-IDF with Features*

Description

Normalize ‘cm‘ matrix using TF-IDF and then column-wise min-max scaling

Usage

```
normalizeTfIdfWithFeatures(cm, max.quantile = 0.95, max.smooth = 1e-10)
```

Arguments

cm	matrix to normalize. Rows are observations (e.g. cells) and columns are features (e.g. genes)
max.quantile	quantile to be used for max estimation during scaling

Value

Normalized matrix of the same shape as ‘cm‘

parseMarkerFile *Parse Marker File*

Description

read markers from the markup file

Usage

```
parseMarkerFile(path)
```

Arguments

path	path to the markup file

Value

list of markers for each cell type. List elements have the following info:

- expressed: vector of positive markers
- not_expressed: vector of negative markers
- parent: parent cell type (‘"root"‘ if there is no parent)

`plapply` *Parallel Lapply*

Description

parallel, optionally verbose lapply

Usage

```
plapply(..., n.cores = 1, verbose = F)
```

Arguments

`n.cores`	number of cores to use
`verbose`	show progress bar

`plotAssignmentScores` *Plot Assignment Scores*

Description

plot assignment scores on a separate scatterplot for each cell type

Usage

```
plotAssignmentScores(embedding, scores, classification.tree,
  parent.node = "root", build.panel = T, n.col = NULL,
  n.row = NULL, ...)
```

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
`scores`	assignment scores. Can be obtained with 'assignCellsByScores'
`classification.tree`	cell type hierarchy represented by graph. Part of 'clf_data' object from 'getClassificationData'
`parent.node`	cell type, which subtypes' scores must be plotted
`build.panel`	combine individual plots to a single panel
`n.col`	number of columns in the panel
`n.row`	number of rows in the panel
`...`	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.

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.

show.legend logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.

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`.

`...` Arguments passed on to `conos::embeddingPlot`

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

colors vector of numbers, which must be shouwn with point colors, names contain cell names. This argument is ignored if groups are provided.

subgroups subset of 'groups', selecting the cells for plot. Ignored if 'groups' is NULL

plot.na plot points, for which groups / colors are missed (TRUE / FALSE)

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

mark.groups plot cluster labels above points

show.legend show legend

alpha opacity level [0; 1]

size point size

title plot title

plot.theme theme for the plot

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

color.range controls range, in which colors are estimated. 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. 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

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

legend.title legend title

gradient.range.quantile Winsorization quantile for the numeric colors and gene gradient

raster should layer with the points be rasterized (TRUE/ FALSE)? Setting of this argument to TRUE is useful when you need to export a plot with large number of points

raster.width width of the plot in inches. Ignored if raster == FALSE.

raster.height height of the plot in inches. Ignored if raster == FALSE.

raster.dpi dpi of the rasterized plot. Ignored if raster == FALSE.

shuffle.colors shuffle colors

keep.limits Keep axis limits from original plot, useful when plotting subgroups, only meaningful it plot.na=F

Value

panel of plots if 'build.panel==T' or 'plot.list' otherwise

Examples

```
clf_data <- getClassificationData(cm, marker_path)
ann_by_level <- assignCellsByScores(graph, clf_data)
plotAssignmentScores(t_sne, ann_by_level$scores$l1, clf_data$classification.tree)
```

---

plotGeneExpression *Plot gene expression on cell embedding*

Description

Plot gene expression on cell embedding

Usage

```
plotGeneExpression(genes, embedding, cm, build.panel = T, n.col = NULL,
  n.row = NULL, ...)
```

Arguments

genes	vector of gene names
embedding	cell embedding
cm	count matrix with genes as columns
build.panel	combine individual plots to a single panel
n.col	number of columns in the panel
n.row	number of rows in the panel
...	Arguments passed on to conos::embeddingPlot

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

colors vector of numbers, which must be shouwn with point colors, names contain cell names. This argument is ignored if groups are provided.

subgroups subset of 'groups', selecting the cells for plot. Ignored if 'groups' is NULL

plot.na plot points, for which groups / colors are missed (TRUE / FALSE)

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

mark.groups plot cluster labels above points

show.legend show legend

alpha opacity level [0; 1]

size point size

title plot title

plot.theme theme for the plot

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

color.range controls range, in which colors are estimated. 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. 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

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

legend.title legend title

gradient.range.quantile Winsorization quantile for the numeric colors and gene gradient

raster should layer with the points be rasterized (TRUE/ FALSE)? Setting of this argument to TRUE is useful when you need to export a plot with large number of points

raster.width width of the plot in inches. Ignored if raster == FALSE.

raster.height height of the plot in inches. Ignored if raster == FALSE.

raster.dpi dpi of the rasterized plot. Ignored if raster == FALSE.

shuffle.colors shuffle colors

keep.limits Keep axis limits from original plot, useful when plotting subgroups, only meaningful it plot.na=F

`plotMarkerListViolinMap` *Plot expression violin map for given marker list*

Description

Plot expression violin map for given marker list

Usage

```
plotMarkerListViolinMap(count.matrix, annotation, parent.type = "root",
  clf.data = NULL, clf.tree = NULL, marker.list = NULL,
  max.depth = NULL, marker.type = "expressed", text.angle = 45,
  gene.order = NULL, ...)
```

Arguments

`count.matrix`	gene expression matrix with genes by columns and cells by rows
`parent.type`	cell type for which the markers of the subtypes must be plotted
`marker.list`	list of markers per cell type. Can be obtained with 'parseMarkerFile'
`...`	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.

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.

show.legend logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.

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`.

`plotSubtypeMarkers` *Plot Subtype Markers*

Description

plot markers for all subtypes of the specified 'parent.type'

Usage

```
plotSubtypeMarkers(embedding, count.matrix, parent.type = "root",
  clf.data = NULL, clf.tree = NULL, marker.list = NULL,
  show.legend = F, max.depth = NULL, build.panel = T, n.col = NULL,
  n.row = NULL, marker.type = c("expressed", "not_expressed"), ...)
```

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
`count.matrix`	gene expression matrix with genes by columns and cells by rows
`parent.type`	cell type for which the markers of the subtypes must be plotted
`marker.list`	list of markers per cell type. Can be obtained with 'parseMarkerFile'
`show.legend`	show legend
`build.panel`	combine individual plots to a single panel
`n.col`	number of columns in the panel
`n.row`	number of rows in the panel
`...`	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.

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.

show.legend logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.

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`.

`...` Arguments passed on to `conos::embeddingPlot`

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

colors vector of numbers, which must be shouwn with point colors, names contain cell names. This argument is ignored if groups are provided.

subgroups subset of 'groups', selecting the cells for plot. Ignored if 'groups' is NULL

plot.na plot points, for which groups / colors are missed (TRUE / FALSE)

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

mark.groups plot cluster labels above points

show.legend show legend

alpha opacity level [0; 1]

size point size

title plot title

plot.theme theme for the plot

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

color.range controls range, in which colors are estimated. 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. 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

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

legend.title legend title

gradient.range.quantile Winsorization quantile for the numeric colors and gene gradient

raster should layer with the points be rasterized (TRUE/ FALSE)? Setting of this argument to TRUE is useful when you need to export a plot with large number of points

raster.width width of the plot in inches. Ignored if raster == FALSE.

raster.height height of the plot in inches. Ignored if raster == FALSE.

raster.dpi dpi of the rasterized plot. Ignored if raster == FALSE.

shuffle.colors shuffle colors

keep.limits Keep axis limits from original plot, useful when plotting subgroups, only meaningful it plot.na=F

`plotTypeMarkers` *Plot Type Markers*

Description

plot markers for the specified 'cell.type'

Usage

```
plotTypeMarkers(embedding, count.matrix, cell.type, marker.list,
  show.legend = T, build.panel = T, n.col = NULL, n.row = NULL,
  ...)
```

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
`count.matrix`	gene expression matrix with genes by columns and cells by rows
`cell.type`	cell type for which the markers must be plotted
`marker.list`	list of markers per cell type. Can be obtained with 'parseMarkerFile'
`show.legend`	show legend
`build.panel`	combine individual plots to a single panel
`n.col`	number of columns in the panel
`n.row`	number of rows in the panel
`...`	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.

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.

show.legend logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.

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`.

`...` Arguments passed on to `conos::embeddingPlot`

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

colors vector of numbers, which must be shouwn with point colors, names contain cell names. This argument is ignored if groups are provided.

subgroups subset of 'groups', selecting the cells for plot. Ignored if 'groups' is NULL

plot.na plot points, for which groups / colors are missed (TRUE / FALSE)

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

mark.groups plot cluster labels above points

show.legend show legend

alpha opacity level [0; 1]

size point size

title plot title

plot.theme theme for the plot

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

color.range controls range, in which colors are estimated. 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. 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

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

legend.title legend title

gradient.range.quantile Winsorization quantile for the numeric colors and gene gradient

raster should layer with the points be rasterized (TRUE/ FALSE)? Setting of this argument to TRUE is useful when you need to export a plot with large number of points

raster.width width of the plot in inches. Ignored if raster == FALSE.

raster.height height of the plot in inches. Ignored if raster == FALSE.

raster.dpi dpi of the rasterized plot. Ignored if raster == FALSE.

shuffle.colors shuffle colors

keep.limits Keep axis limits from original plot, useful when plotting subgroups, only meaningful it plot.na=F

`plotUncertaintyPerCell` *Uncertainty scatterplots per cell*

Description

Uncertainty scatterplots per cell

Usage

```
plotUncertaintyPerCell(embedding, uncertainty.info,
  palette = colorRampPalette(c("gray", "#ffeda0", "#fec44f", "#f03b20")),
  alpha = 0.3, build.panel = T, n.col = length(uncertainty.info),
  n.row = NULL, ...)
```

Arguments

`build.panel`	join plots to single panel
`...`	Arguments passed on to `conos::embeddingPlot`

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

colors vector of numbers, which must be shouwn with point colors, names contain cell names. This argument is ignored if groups are provided.

subgroups subset of 'groups', selecting the cells for plot. Ignored if 'groups' is NULL

plot.na plot points, for which groups / colors are missed (TRUE / FALSE)

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

mark.groups plot cluster labels above points

show.legend show legend

alpha opacity level [0; 1]

size point size

title plot title

plot.theme theme for the plot

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

color.range controls range, in which colors are estimated. 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. 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

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

legend.title legend title

gradient.range.quantile Winsorization quantile for the numeric colors and gene gradient

raster should layer with the points be rasterized (TRUE/ FALSE)? Setting of this argument to TRUE is useful when you need to export a plot with large number of points

raster.width width of the plot in inches. Ignored if raster == FALSE.

raster.height height of the plot in inches. Ignored if raster == FALSE.

raster.dpi dpi of the rasterized plot. Ignored if raster == FALSE.

shuffle.colors shuffle colors

keep.limits Keep axis limits from original plot, useful when plotting subgroups, only meaningful it plot.na=F

plotUncertaintyPerClust

Uncertainty barplots per cluster

Description

Uncertainty barplots per cluster

Usage

```
plotUncertaintyPerClust(uncertainty.per.clust, clusters,
  annotation = NULL, ann.per.clust = NULL, thresholds = c(coverage =
  0.5, negative = 0.5, positive = 0.75), build.panel = T, n.col = 1,
  n.row = NULL, adjust.legend = T, rel.legend.width = 0.25, ...)
```

Arguments

`...`	Arguments passed on to `plotOneUncertaintyPerClust`
	text.angle angle of x-axis labels

scoreCellUncertaintyPerLevel

Score Cell Uncertainty Per Level

Description

Estimate uncertainty scores per annotation level

Usage

```
scoreCellUncertaintyPerLevel(ann.info.per.level, score.info, verbose = F,
  ...)
```

Arguments

`ann.info.per.level`	annotation info per level. Result of 'assignCellsByScores'.
`score.info`	list of marker scores per cell type. Can be obtained with 'getMarkerScoreInfo'
`verbose`	show progress bar for estimation
`...`	Arguments passed on to `scorePerCellUncertainty`
	cur.types subset of types used to measure uncertainty. Default: all values presented in annotation
	coverage.max.quantile all coverage values above this quantille are winsorized

Value

Uncertainty per level for each cell

Examples

```
score_info <- getMarkerScoreInfo(clf_data)
unc_info <- scoreCellUncertaintyPerLevel(ann_by_level, score_info)
```

`scoreClusterUncertaintyPerLevel`
Score Cluster Uncertainty Per Level

Description

Score uncertainty for each cluster per level

Usage

```
scoreClusterUncertaintyPerLevel(cell.uncertainty.per.level, clusters)
```

Arguments

`cell.uncertainty.per.level`	uncertainty per cell per level. Can be obtained from 'scoreCellUncertaintyPerLevel'
`clusters`	vector with cluster label for each cell

Value

Uncertainty per level for each cluster

Examples

```
score_info <- getMarkerScoreInfo(clf_data)
unc_info <- scoreCellUncertaintyPerLevel(ann_by_level, score_info)
unc_per_clust <- scoreClusterUncertaintyPerLevel(unc_info, clusters)
```

`scorePerCellUncertainty` *Score Per Cell Uncertainty*

Description

Score uncertainty per each cell

Usage

```
scorePerCellUncertainty(annotation, scores.norm, score.info,
  cur.types = unique(annotation), coverage.max.quantile = 0.75)
```

Arguments

`annotation`	assigned annotation labels. Can be obtrained with 'assignCellsByScores'.
`scores.norm`	assignment scores. Can be obtrained with 'assignCellsByScores'.
`score.info`	list of marker scores per cell type. Can be obtained with 'getMarkerScoreInfo'
`cur.types`	subset of types used to measure uncertainty. Default: all values presented in annotation
`coverage.max.quantile`	all coverage values above this quantille are winsorized

Value

list of scores with elements:

- positive: uncertainty based on conflicting positive markers
- negative: uncertainty based on large expression of negative markers
- coverage: uncertainty based on low coverage

Index