Package 'STdeconvolve'

May 25, 2024

Type Package

Title Reference-free Cell-Type Deconvolution of Multi-Cellular Spatially Resolved Transcriptomics Data

Version 1.9.0

URL https://jef.works/STdeconvolve/

BugReports https://github.com/JEFworks-Lab/STdeconvolve/issues

Description STdeconvolve as an unsupervised, reference-free approach to infer latent cell-type proportions and transcriptional profiles within multi-cellular spatially-resolved pixels from spatial transcriptomics (ST) datasets. STdeconvolve builds on latent Dirichlet allocation (LDA), a generative statistical model commonly used in natural language processing for discovering latent topics in collections of documents. In the context of natural language processing, given a count matrix of words in documents, LDA infers the distribution of words for each topic and the distribution of topics in each document. In the context of ST data, given a count matrix of gene expression in multi-cellular ST pixels, STdeconvolve applies LDA to infer the putative transcriptional profile for each cell-type and the proportional representation of each cell-type in each multi-cellular ST pixel.

biocViews Transcriptomics, Visualization, RNASeq, Bayesian, Spatial, Software, GeneExpression

License GPL-3

Encoding UTF-8

LazyData FALSE

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports topicmodels, BiocParallel, Matrix, methods, mgcv, ggplot2, scatterpie, viridis, slam, stats, clue, liger, reshape2, graphics, grDevices, utils

Depends R (>= 4.1)

Suggests knitr, BiocStyle, rmarkdown, testthat, rcmdcheck, gplots, gridExtra, hash, dplyr, parallel

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/STdeconvolve

git_branch devel

git_last_commit 2b5edb8

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-05-24

Author Brendan Miller [aut, cre] (<https://orcid.org/0000-0002-9559-4045>), Jean Fan [aut] (<https://orcid.org/0000-0002-0212-5451>)

Maintainer Brendan Miller <bmill3r@gmail.com>

Contents

`annotateCellTypesGSEA` *Match deconvolved cell-types to ground truth cell-types based on transcriptional profiles*

Description

Match deconvolved cell-types to ground truth cell-types by testing for enrichment of ground truth marker gene sets in the deconvolved transcriptional profiles. Uses liger::iterative.bulk.gsea.

Usage

```
annotateCellTypesGSEA(beta, gset, qval = 0.05)
```

Arguments

beta	cell-type (rows) x gene (columns) matrix of deconvolved cell-type transcriptional profiles
gset	named list where each entry is a vector of marker genes for a given ground truth cell-type.
qval	adjusted p-value threshold (default: 0.05)

Value

A list that contains

- results: A named list that contains sorted matrices for each deconvolved cell-type. The matrix rows are the ground truth cell-types ordered by significance, edge-score, and enrichment score of their gene sets in the deconvolved transcriptional profile of a given deconvolved cell-type.
- predictions: a named vector where the names are the deconvolved cell-types and the values are the best matched ground truth cell-type that is also positively enriched.

---

cleanCounts *Filter a counts matrix*

Description

Filter a counts matrix based on gene (row) and cell (column) requirements.

Usage

```
cleanCounts(
  counts,
  min.lib.size = 1,
  max.lib.size = Inf,
  min.reads = 1,
  min.detected = 1,
  verbose = FALSE,
  plot = FALSE
)
```

Arguments

counts	A sparse read count matrix. The rows correspond to genes, columns correspond to individual cells
min.lib.size	Minimum number of genes detected in a cell. Cells with fewer genes will be removed (default: 1)
max.lib.size	Maximum number of genes detected in a cell. Cells with more genes will be removed (default: Inf)
min.reads	Minimum number of reads per gene. Genes with fewer reads will be removed (default: 1)

min.detected	Minimum number of cells a gene must be seen in. Genes not seen in a sufficient number of cells will be removed (default: 1)
verbose	Verbosity (default: FALSE)
plot	Whether to plot (default: FALSE)

Value

a filtered read count matrix

Examples

```
data(mOB)
counts <- cleanCounts(mOB$counts, min.lib.size = 100)
```

---

correlationPlot *Generate heatmap of correlations*

Description

Visualize the correlations between topics stored in a matrix, typically one returned via `getCorrMtx()`. This function uses ggplot2::geom_tile.

Usage

```
correlationPlot(
  mat,
  colLabs = NA,
  rowLabs = NA,
  title = NA,
  annotation = FALSE
)
```

Arguments

mat	matrix with correlation values from -1 to 1
colLabs	x-axis label for plot. These are the columns of the matrix, or specifically m2 from getCorrMtx. (default: NULL)
rowLabs	y-axis label for plot. These are the rows of the matrix, or specifically m1 from getCorrMtx. (default: NULL)
title	title of the plot. (default: NULL)
annotation	Boolean to show the correlation values in the squares of the heatmap (default; FALSE)

Value

a heatmap of the values in the input mat

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = 3)
optLDA <- optimalModel(models = ldas, opt = 3)
results <- getBetaTheta(optLDA, perc.filt = 0.05, betaScale = 1000)
deconProp <- results$theta
corMtx <- getCorrMtx(m1 = as.matrix(deconProp), m2 = as.matrix(deconProp), type = "t")
rownames(corMtx) <- paste0("X", seq(nrow(corMtx)))
colnames(corMtx) <- paste0("X", seq(ncol(corMtx)))
correlationPlot(mat = corMtx, title = "Proportional correlation", annotation = TRUE) +
    ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 90, vjust = 0))
```

`fitLDA` *Find the optimal number of cell-types K for the LDA model*

Description

The input for topicmodels::LDA needs to be a slam::as.simple_triplet_matrix (docs x words). Access a given model in the returned list via: lda$models$k. The models are objects from the R package "topicmodels". The LDA models have slots with additional information.

Usage

```
fitLDA(
  counts,
  Ks = seq(2, 10, by = 2),
  seed = 0,
  perc.rare.thresh = 0.05,
  ncores = 1,
  plot = TRUE,
  verbose = TRUE
)
```

Arguments

`counts`	Gene expression counts with pixels as rows and genes as columns
`Ks`	vector of K parameters, or number of cell-types, to fit models with
`seed`	Random seed
`perc.rare.thresh`	the number of deconvolved cell-types with mean pixel proportion below this fraction used to assess performance of fitted models for each K. Recorded for each K. (default: 0.05)

ncores	Number of cores for parallelization (default: 1). Suggest: parallel::detectCores()
plot	Boolean for plotting (default: TRUE)
verbose	Boolean for verbosity (default: TRUE)

Value

A list that contains

- models: each fitted LDA model for a given K
- kneedOptK: the optimal K based on Kneed algorithm
- minOptK: the optimal K based on minimum
- ctPropOptK: Suggested upper bound on K. K in which number of returned cell-types with mean proportion < perc.rare.thresh starts to increase steadily.
- numRare: number of cell-types with mean pixel proportion < perc.rare.thresh for each K
- perplexities: perplexity scores for each model
- fitCorpus: the corpus that was used to fit each model
- testCorpus: the corpus used to compute model perplexity.

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = 3, ncores=7)
```

---

getBetaTheta *Pull out cell-type proportions across pixels (theta) and cell-type gene probabilities (beta) matrices from fitted LDA models from fitLDA*

Description

Pull out cell-type proportions across pixels (theta) and cell-type gene probabilities (beta) matrices from fitted LDA models from fitLDA

Usage

```
getBetaTheta(
  lda,
  corpus = NULL,
  perc.filt = 0.05,
  betaScale = 1,
  verbose = TRUE
)
```

Arguments

lda	an LDA model from "topicmodels" R package. From list of models returned by fitLDA
corpus	If corpus is NULL, then it will use the original corpus that the model was fitted to. Otherwise, compute deconvolved topics from this new corpus. Needs to be pixels x genes and nonnegative integer counts. Each row needs at least 1 nonzero entry (default: NULL)
perc.filt	proportion threshold to remove cell-types in pixels (default: 0.05)
betaScale	factor to scale the predicted cell-type gene expression profiles (default: 1)
verbose	Boolean for verbosity (default: TRUE)

Value

A list that contains

- beta: cell-type (rows) by gene (columns) distribution matrix. Each row is a probability distribution of a cell-type expressing each gene in the corpus
- theta: pixel (rows) by cell-types (columns) distribution matrix. Each row is the cell-type composition for a given pixel

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = 3, ncores=7)
optLDA <- optimalModel(models = ldas, opt = 3)
results <- getBetaTheta(optLDA, perc.filt = 0.05, betaScale = 1000)
head(results$theta)
head(results$beta)
```

---

getCorrMtx *Find Pearson's correlations between topics (cell-types) with respect to their proportions across documents (pixels), i.e. thetas, or gene probabilities, i.e. betas.*

Description

Find Pearson's correlations between topics (cell-types) with respect to their proportions across documents (pixels), i.e. thetas, or gene probabilities, i.e. betas.

Usage

```
getCorrMtx(m1, m2, type = c("t", "b"), thresh = NULL, verbose = TRUE)
```

Arguments

m1	first matrix
m2	second matrix
type	must be either "t" (theta; cell-type proportions across pixels) or "b" (beta; cell-type gene expression profiles)
thresh	if comparing betas, use to compare genes above this probability (e.g., expression level). NULL or 0 < numeric < 1.0 (default: NULL)
verbose	control the verbosity (default: TRUE)

Value

matrix of Pearson's correlations; m1 (rows) by m2 (cols)

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = 3)
optLDA <- optimalModel(models = ldas, opt = 3)
results <- getBetaTheta(optLDA, perc.filt = 0.05, betaScale = 1000)
deconProp <- results$theta
corMtx <- getCorrMtx(m1 = as.matrix(deconProp), m2 = as.matrix(deconProp), type = "t")
rownames(corMtx) <- paste0("X", seq(nrow(corMtx)))
colnames(corMtx) <- paste0("X", seq(ncol(corMtx)))
head(corMtx)
```

---

getOverdispersedGenes *Normalize gene expression variance relative to transcriptome-wide expectations (Modified from SCDE/PAGODA2 code)*

Description

Normalizes gene expression magnitudes to with respect to its ratio to the transcriptome-wide expectation as determined by local regression on expression magnitude

Usage

```
getOverdispersedGenes(
  counts,
  gam.k = 5,
  alpha = 0.05,
  plot = FALSE,
  use.unadjusted.pvals = FALSE,
```

```
  do.par = TRUE,
  max.adjusted.variance = 1000,
  min.adjusted.variance = 0.001,
  verbose = TRUE,
  details = FALSE
)
```

Arguments

counts	Read count matrix. The rows correspond to genes, columns correspond to individual cells
gam.k	Generalized additive model parameter; the dimension of the basis used to represent the smooth term (default: 5)
alpha	Significance threshold (default: 0.05)
plot	Whether to plot the results (default: FALSE)
use.unadjusted.pvals	If true, will apply BH correction (default: FALSE)
do.par	Whether to adjust par for plotting if plotting (default: TRUE)
max.adjusted.variance	Ceiling on maximum variance after normalization to prevent infinites (default: 1e3)
min.adjusted.variance	Floor on minimum variance after normalization (default: 1e-3)
verbose	Verbosity (default: TRUE)
details	If true, will return data frame of normalization parameters. Else will return list of overdispersed genes. (default: FALSE)

Value

If details is true, will return data frame of normalization parameters. Else will return list of overdispersed genes.

Examples

```
data(mOB)
od <- getOverdispersedGenes(counts = mOB$counts, gam.k = 5, alpha = 0.05, details = FALSE)
head(od)

od <- getOverdispersedGenes(counts = mOB$counts, gam.k = 5, alpha = 0.05, details = TRUE)
head(od$mat)
head(od$ods)
head(od$df)
```

`lsatPairs` *Function to get Hungarian sort pairs via clue::lsat*

Description

Finds best matches between cell-types that correlate between beta or theta matrices that have been compared via getCorrMtx(). Each row is paired with a column in the output matrix from getCorrMtx(). If there are less rows than columns, then some columns will not be matched and not part of the output.

Usage

```
lsatPairs(mtx)
```

Arguments

`mtx`	output correlation matrix from getCorrMtx(). Must not have more rows than columns

Value

A list that contains

- pairs: output of clue::solve_LSAP. A vectorized object where for each position the first element is a row and the second is the paired column.
- rowix: the indices of the rows. Essentially seq_along(pairing)
- colsix: the indices of each column paired to each row

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = 3)
optLDA <- optimalModel(models = ldas, opt = 3)
results <- getBetaTheta(optLDA, perc.filt = 0.05, betaScale = 1000)
deconProp <- results$theta
corMtx <- getCorrMtx(m1 = as.matrix(deconProp), m2 = as.matrix(deconProp), type = "t")
pairs <- lsatPairs(corMtx)
pairs
```

mOB *Spatial transcriptomics of the mouse olfactory bulb*

Description

Spatial transcriptomics of the mouse olfactory bulb

Usage

```
data(mOB)
```

Format

List where 'counts' is a sparse matrix with columns as voxels and rows as genes and 'pos' is a data frame of x and y position values per voxel

Source

https://science.sciencemag.org/content/353/6294/78

optimalModel *Get the optimal LDA model*

Description

Get the optimal LDA model

Usage

```
optimalModel(models, opt)
```

Arguments

models	list returned from fitLDA
opt	either "kneed" (kOpt1) or "min" (kOpt2), or designate a specific K. "kneed" = K vs perplexity inflection point. "min" = K corresponding to minimum perplexity "proportion" = K vs number of cell-type with mean proportion < 5% inflection point

Value

optimal LDA model fitted to the K based on opt parameter

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = seq(2,4), ncores=7)
optLDA <- optimalModel(models = ldas, opt = "min")
```

`perplexityPlot` *Plot the perplexity and rare cell-types versus fitted Ks*

Description

the same plot returned by fitLDA() but now callable as a separate function.

Usage

```
perplexityPlot(models, corpus = NULL, perc.rare.thresh = 0.05)
```

Arguments

`models`	list returned from fitLDA
`corpus`	If corpus is NULL, then it will use the original corpus that the model was fitted to. Otherwise, compute deconvolved topics from this new corpus. Needs to be pixels x genes and nonnegative integer counts. Each row needs at least 1 nonzero entry (default: NULL)
`perc.rare.thresh`	the number of deconvolved cell-types with mean pixel proportion below this fraction used to assess performance of fitted models for each K. Recorded for each K. (default: 0.05)

Value

a plot indicating the perplexity and number of rare cell-types of a list of fitted LDA models

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = seq(2,5))
perplexityPlot(models = ldas, corpus = corpus)
```

`preprocess` *Pre-process ST pixel gene count matrices to construct corpus for input into LDA*

Description

Takes pixel (row) x gene (columns) matrix and filters out poor genes and pixels. Then selects for genes to be included in final corpus for input into LDA. If the pixel IDs are made up of their positions in "XxY" these can be extracted as the pixel position coordinates (a characteristic of Stahl datasets).

```
Order of filtering options:
1. Selection to use specific genes only
2. `cleanCounts` to remove poor pixels and genes
3. Remove top expressed genes in matrix
4. Remove specific genes based on grepl pattern matching
5. Remove genes that appear in more/less than a percentage of pixels
6. Use the over dispersed genes computed from the remaining genes
   after filtering steps 1-5 (if selected)
7. Choice to use the top over dispersed genes based on -log10(p.adj)
```

Usage

```
preprocess(
  dat,
  extractPos = FALSE,
  selected.genes = NA,
  nTopGenes = NA,
  genes.to.remove = NA,
  removeAbove = NA,
  removeBelow = NA,
  min.reads = 1,
  min.lib.size = 1,
  min.detected = 1,
  ODgenes = TRUE,
  nTopOD = 1000,
  od.genes.alpha = 0.05,
  gam.k = 5,
  verbose = TRUE,
  plot = TRUE
)
```

Arguments

`dat`	pixel (row) x gene (columns) mtx with gene counts OR path to it
`extractPos`	Boolean to extract pixel positional coordinates from pixel name names (default: FALSE)

selected.genes	vector of gene names to use specifically for the corpus (default: NA)
nTopGenes	integer for number of top expressed genes to remove (default: NA)
genes.to.remove	vector of gene names or patterns for matching to genes to remove (default: NA). ex: c("^mt-") or c("^MT", "^RPL", "^MRPL")
removeAbove	non-negative numeric <=1 to use as a percentage. Removes genes present in this fraction or more of pixels (default: NA)
removeBelow	non-negative numeric <=1 to use as a percentage. Removes genes present in this fraction or less of pixels (default: NA)
min.reads	cleanCounts() param; minimum number of reads to keep a gene (default: 1)
min.lib.size	cleanCounts() param; minimum number of counts a pixel needs to keep (default: 1)
min.detected	cleanCounts() param; minimum number of pixels a gene needs to have been detected in to keep (default: 1)
ODgenes	Boolean to use getOverdispersedGenes() for the corpus genes (default: TRUE)
nTopOD	number of top over dispersed genes to use. int (default: 1000). If the number of overdispersed genes is less then this number will use all of them, or set to NA to use all overdispersed genes.
od.genes.alpha	alpha parameter for getOverdispersedGenes(). Higher = less stringent and more over dispersed genes returned (default: 0.05)
gam.k	gam.k parameter for getOverdispersedGenes(). Dimension of the "basis" functions in the GAM used to fit, higher = "smoother" (default: 5)
verbose	control verbosity (default: TRUE)
plot	control if plots are returned (default: TRUE)

Value

A list that contains

- corpus: (pixels x genes) matrix of the counts of the selected genes
- slm: slam::as.simple_triplet_matrix(corpus); required format for topicmodels::LDA input
- positions: matrix of x and y coordinates of pixels rownames = pixels, colnames = "x", "y"

Examples

```
data(mOB)
cd <- mOB$counts
corpus <- preprocess(t(cd), removeAbove = 0.95, removeBelow = 0.05)
```

`restrictCorpus` *Restrict to informative words (genes) for topic modeling*

Description

identifies over dispersed genes across pixels to use as informative words (genes) in topic modeling. Also allows ability to restrict over dispersed genes to those that occur in more than and/or less than selected fractions of pixels in corpus. Limits to the top 1000 overdispersed genes in order to keep the corpus to a reasonable size.

Usage

```
restrictCorpus(
  counts,
  removeAbove = 1,
  removeBelow = 0.05,
  alpha = 0.05,
  nTopOD = 1000,
  plot = FALSE,
  verbose = TRUE
)
```

Arguments

`counts`	genes x pixels gene count matrix
`removeAbove`	remove over dispersed genes that are present in more than this fraction of pixels (default: 1.0)
`removeBelow`	remove over dispersed genes that are present in less than this fraction of pixels (default: 0.05)
`alpha`	alpha parameter for getOverdispersedGenes(). Higher = less stringent and more overdispersed genes returned (default: 0.05)
`nTopOD`	number of top over dispersed genes to use. int (default: 1000). If the number of overdispersed genes is less then this number will use all of them, or set to NA to use all overdispersed genes.
`plot`	return histogram plots of genes per pixel and pixels per genes for over dispersed genes and after corpus restriction. (default: FALSE)
`verbose`	(default: TRUE)

Value

a gene by pixel matrix where the remaining genes have been filtered

Examples

```
data(mOB)
corpus <- restrictCorpus(counts = mOB$counts)
corpus
```

`topGenes` *Returns top n genes of each deconvolved cell-type for a given beta matrix*

Description

For a given beta matrix (cell-type gene distribution matrix), returns the top n genes based on their probability.

Usage

```
topGenes(beta, n = 10)
```

Arguments

`beta`	beta matrix (cell-type gene distribution matrix)
`n`	number of top genes for each deconvolved cell-type to return (default: 10)

Value

a list where each item is a vector of the top genes and their associated probabilities for a given deconvolved cell-type

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = 3, ncores=7)
optLDA <- optimalModel(models = ldas, opt = 3)
results <- getBetaTheta(optLDA, perc.filt = 0.05, betaScale = 1000)
deconGexp <- results$beta
genes <- topGenes(deconGexp)
```

`vizAllTopics` *Visualize all topic proportions across pixels with* `scatterpie`

Description

Note: visualizes all cell-types in theta at once (could be individual cell-types or cell-type-clusters) so for accuracy of the proportions of each cell-type in a pixel, the row (pixel) should sum to 1.

Usage

```
vizAllTopics(
  theta,
  pos,
  topicOrder = seq(ncol(theta)),
  topicCols = rainbow(ncol(theta)),
  groups = NA,
  group_cols = NA,
  r = max(0.4, max(pos)/nrow(pos) * 4),
  lwd = 0.5,
  showLegend = TRUE,
  plotTitle = NA,
  overlay = NA
)
```

Arguments

`theta`	document (pixel) x cell-type proportion matrix
`pos`	position of pixels, as data.frame with x and y columns
`topicOrder`	order of topics in theta to visualize as a numeric vector and same length as topicCols (default: seq(ncol(theta)))
`topicCols`	Vector of colors for each of the cell-types to be visualized. Same length and order as topicOrder (default: rainbow(ncol(theta)))
`groups`	Indicates color of the scatterpie strokes (borders) with the goal of coloring them by their assigned group. This can be a vector or factor indicating the group of each pixel. Needs to be in the same order as the pixel rows in "theta" (default: NA)
`group_cols`	Color labels for the groups. Can be a vector or factor. (default: NA)
`r`	Radius of the scatterpie circles. Adjust based on positions of pixels (default: max(0.4, max(pos)/nrow(pos)*4))
`lwd`	Width of lines of the pie charts. Increasing helps visualize group_cols if being used.
`showLegend`	Boolean to show the legend indicating cell-types and their color
`plotTitle`	add title to the resulting plot (default: NA)
`overlay`	raster image of an H&E tissue (for example) to plot the scatterpies on top of (default: NA)

Value

a plot of scatterpies, where each scatterpie represents a pixel in space based on the x,y coordinates and the components represent the proportion of each cell-type at that pixel.

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
annot <- mOB$annot
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = 3)
optLDA <- optimalModel(models = ldas, opt = 3)
results <- getBetaTheta(optLDA, perc.filt = 0.05, betaScale = 1000)
deconProp <- results$theta
vizAllTopics(deconProp,pos, groups = annot, group_cols = rainbow(length(levels(annot))), r=0.4)
```

---

vizGeneCounts *Visualize gene counts for a given gene in the pixels. Can also see group assignment of spots.*

Description

Visualize one gene at a time.

Usage

```
vizGeneCounts(
  df,
  gene,
  groups = NA,
  group_cols = NA,
  winsorize = 0,
  size = 7,
  stroke = 0.5,
  alpha = 1,
  plotTitle = NA,
  showLegend = TRUE
)
```

Arguments

df	data.frame where rows are spots and columns must be at least: "x", "y" for spot positions in space and "gene" column that is counts of a gene for each spot.
gene	column name of the gene counts in df to be visualized

groups	colors the spots lines based on a group or cell layer they belong to. Needs to be a character vector in the order of the spot rows in df. Ex: c("0", "1", "0", ...)
group_cols	color labels for the groups. Ex: c("0" = "gray", "1" = "red")
winsorize	Winsorization quantile
size	size of the geom_points to plot (default: 7)
stroke	thickness of the geom_point lines to help in emphasizing groups (default: 0.5)
alpha	alpha value of colored pixels (default: 1)
plotTitle	option to add a title to the plot
showLegend	Boolean to show the plot legend

Value

a plot where each point is a pixel colored by the expression level of the selected gene

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
df <- merge(as.data.frame(pos), as.data.frame(t(as.matrix(counts))), by = 0)
vizGeneCounts(df = df, gene = "Sox11",
    size = 3, stroke = 0.1, plotTitle = "Sox11",
    winsorize = 0.05, showLegend = TRUE)
```

---

vizTopic *Visualize pixel proportions of a single cell-type.*

Description

Visualize the pixel proportions of a single topic.

Usage

```
vizTopic(
  theta,
  pos,
  topic,
  groups = NA,
  group_cols = NA,
  size = 2,
  stroke = 0.3,
  alpha = 1,
  low = "white",
  high = "red",
```

```
  plotTitle = NA,
  showLegend = TRUE
)
```

Arguments

theta	document (pixel) x cell-type proportion matrix
pos	position of pixels, as data.frame with x and y columns
topic	the index of the topic
groups	colors the pixel border lines based on a group or cell layer they belong to. Needs to be a character or named vector of assigned groups for each pixel Ex: c("0", "1", "0", ...)
group_cols	color labels for the groups. Ex: c("0" = "gray", "1" = "red")
size	size of the geom_points to plot (default: 2)
stroke	thickness of the geom_point lines to help in emphasizing groups (default: 0.5)
alpha	alpha value of colored pixels (default: 1)
low	sets the color for the low end of the topic proportion color scale (default: "white")
high	sets the color the the high end of the topic proportion color scale (default: "red")
plotTitle	option to add a title to the plot (character)
showLegend	Boolean to show the plot legend

Value

a plot where each point is a pixel colored by the proportion of the selected cell-type

Examples

```
data(mOB)
pos <- mOB$pos
cd <- mOB$counts
counts <- cleanCounts(cd, min.lib.size = 100)
corpus <- restrictCorpus(counts, removeAbove=1.0, removeBelow = 0.05)
ldas <- fitLDA(t(as.matrix(corpus)), Ks = 3)
optLDA <- optimalModel(models = ldas, opt = 3)
results <- getBetaTheta(optLDA, perc.filt = 0.05, betaScale = 1000)
deconProp <- results$theta
vizTopic(theta = deconProp, pos = pos, topic = "3", plotTitle = "X3",
    size = 5, stroke = 1, alpha = 0.5, low = "white", high = "red")
```

Index