Package ‘genesorteR’

June 2, 2019

Type Package

Title Feature Ranking in Clustered Single Cell Data

Version 0.1.4

Author Mahmoud M Ibrahim

Maintainer Mahmoud M Ibrahim <mmibrahim@pm.me>

Description The main purpose of this R extension is to select features in (possibly very large) single cell data including scRNA-Seq and scATAC-Seq.
The main idea is that the dropout rate of a gene is a good measure of its expression, and that empirical statistics calculated based on binarized expression matrices are sufficient to select marker genes in a way that is consistent with the expected definition of ``marker gene'' in experimental biology research.
It can also provide a ranking of genes (or in fact whatever features present in the data) specificity to each cell cluster, as well as select large or small sets of marker genes by a permutation test or using entropy-based feature selection.
To assess cell clustering quality, some functions can also compute cell cluster quality metrics.

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

URL http://github.com/mahmoudibrahim/genesorteR

BugReports http://github.com/mahmoudibrahim/genesorteR/issues

Imports mclust,
parallel,
pheatmap,
methods

Depends R (>= 2.10),
Matrix

R topics documented:

`getClassAUC` *getClassAUC*

Description

getClassAUC implements one way to investigate clustering quality. It processes the output of `sortGenes` to obtain a curve for each cell cluster for all gene specificity scores against their ranking in the cluster. The Area Under the Curve (AUC) can be used as a measure of clustering quality in terms of the possibility to identify cell clusters using a few marker genes. See Details.

Usage

```
getClassAUC(gs, markers = NULL, plotCurves = TRUE,
  colors = c("#d64e3c7F", "#7dd5547F", "#8641c67F", "#cfca457F",
  "#7778cb7F", "#59803d7F", "#d04d9c7F", "#73d6a87F", "#492f607F",
  "#ccc4977F", "#7f343b7F", "#72acc07F", "#b97d407F", "#c796b57F",
  "#45483a7F", "#A020F07F", "#00FF007F", "#FFFF007F"))
```

Arguments

`gs`	A list containing `$specScore` sparse matrix. Typically the output of `sortGenes()`.
`markers`	A character vector of gene names to restrict this analysis to. See Details.
`plotCurves`	Should a plot be drawn? default value is TRUE.
`colors`	Color palette for the plot.

Details

Given the specificity score for all genes in a certain cell cluster, we can assume that a well-separated easily-identified cell cluster will have a relatively small number of genes that have a very high specificity score. Top marker genes for a cluster that is poorly separated from other cell clusters will have average or low specificity scores. Sorting the genes for each cell cluster by their specificity score and plotting the scaled scores in order creates a curve that should be far from the diagonal for well-separated clusters but close to the diagonal for poorly-separated clusters. The AUC of this curve can be used to quantify this intuition and estimate a clustering quality metric.

Value

`getClassAUC` returns a numeric vector of length `ncol($specScore)` that contains the AUC for each cell cluster.

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

See Also

`getMarkers` returns a cell cluster Shannon index that tends to correlate well with the AUC metric returned by `getClassAUC`.

Examples

```
#randomly generated cell clusters
set.seed(1234)
exp = matrix(sample(0:20,1000,replace=TRUE), ncol = 20)
rownames(exp) = sapply(1:50, function(x) paste0("g", x))
cellType = sample(c("cell type 1","cell type 2"),20,replace=TRUE)
sg = sortGenes(exp, cellType)
classAUC = getClassAUC(sg)
mean(classAUC) #overall clustering quality

#"reasonably" separated clusters, with a few clear markers
data(sim)
sg = sortGenes(sim$exp, sim$cellType)
classAUC = getClassAUC(sg)
mean(classAUC)

#real data with three well separated clusters
data(kidneyTabulaMuris)
sg = sortGenes(kidneyTabulaMuris$exp, kidneyTabulaMuris$cellType)
classAUC = getClassAUC(sg)
mean(classAUC)
```

`getMarkers` *getMarkers*

Description

getMarkers processes the output of `sortGenes` to select a relatively small set of marker genes.

Usage

```
getMarkers(gs, quant = 0.99, mutualInfo = FALSE, classEnt = FALSE)
```

Arguments

`gs`	A list containing `$specScore` sparse matrix, `$binary` expression sparse matrix and `$inputClass` vector. Typically the output of `sortGenes()`.
`quant`	A number greater than zero and smaller than one. 0.95 by default. See Details.
`mutualInfo`	Logical. If `TRUE`, the mutual information between gene expression and cell class will be returned for each gene. `FALSE` by default.
`classEnt`	Logical. If `TRUE`, a "Cluster Shannon Index" is returned for each cluster. See Details. `FALSE` by default.

Details

getMarkers relies on calculating an entropy-like metric (called here Shannon Index) calculated on the scaled gene-cluster specificity score. The intuition is that genes with low entropy are either lowly expressed or highly specific to one or few cell clusters. Therefore, we select the top n% of genes according to the scaled specificity score and then cluster those genes based on their Shannon Index. `n` is controlled by `quant`, the default value 0.99 means top 1 lower values than 0.95 are not recommended nor necessary. It can be increased to 0.999 for example to obtain a smaller set of genes. Scaling the specificity score for each cluster is done to try to guarantee that markers for each cluster will eventually be selected even if the cluster is not absolutely well separated.

It can also return the mutual information between each gene and cell clusters, as well as a Cluster Shannon Index, indicating how well separated cell clusters are from each other. Cluster Shannon Index is calculated on the scaled specificity score of selected marker genes for each cluster. The intuition is that, when restricted to top marker genes, well-defined clusters will have few high scoring genes (lower Shannon Index).

Value

`getMarkers` returns a list with the following components:

`markers`	A character vector containing the names of selected marker features.
`maxScaledSpecScore`	A numeric vector of length `nrow(specScore)`, including the maximum scaled specificity score for each gene across all cell clusters.
`gene_entropy`	A numeric vector of the same length as `maxScaledSpecScore`, including the Gene Shannon Index for each gene. See Details.
`mutInfo`	A numeric vector of the same length as `maxScaledSpecScore`, including the mutual information between the expression of each gene and the cell clustering.
`classEntropy`	A numeric vector of the same length as `ncol(specScore)`, including the Class Shannon Index for each cluster based on the selected marker genes.

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

Examples

```
#randomly generated cell clusters, almost no markers are found
set.seed(1234)
exp = matrix(sample(0:20,1000,replace=TRUE), ncol = 20)
rownames(exp) = sapply(1:50, function(x) paste0("g", x))
cellType = sample(c("cell type 1","cell type 2"),20,replace=TRUE)
sg = sortGenes(exp, cellType)
mm = getMarkers(sg,quant=0.95)
length(mm$markers) #only one marker gene was found


#"reasonably" separated clusters, with a few clear markers
data(sim)
sg = sortGenes(sim$exp, sim$cellType)
mm = getMarkers(sg,quant=0.95)
length(mm$markers)

#real data with three well separated clusters
```

```
data(kidneyTabulaMuris)
sg = sortGenes(kidneyTabulaMuris$exp, kidneyTabulaMuris$cellType)
mm = getMarkers(sg)
length(mm$markers) #we found 450 candidate markers
#we want to get a more focused list:
mm = getMarkers(sg, quant = 0.999)
length(mm$markers) #11 genes that can alone descriminate between the cell types
```

`getPValues` *getPValues*

Description

getPValues performs a permutation test on the gene-cell type specificity score to obtain a p-value value on each gene-cell type specificity score. The permutation keeps everything the same except that cell type assignments are permuted between the cells (but note that cell type proportions are also kept the same). This function can be a way to select all differentially expressed genes between all cell classes globally in a dataset in one go, but it can also perform "differential expression" locally between two cell types.

Usage

```
getPValues(gs, numPerm = 5, correctMethod = "BH", testGenes = NULL,
  subsetCells = NULL, cores = 1, seed = 111)
```

Arguments

`gs`	A list, typically the output of `sortGenes()`.
`numPerm`	The number of permutations to do. The default value 5 is to ensure reasonable running time for large datasets but might be advisable to increase it in many cases.
`correctMethod`	The method used to correct p-values for multiple hypothesis testing. Any valid input to "method" in `p.adjust` is allowed. p-value correction is done on a gene-by-gene basis.
`testGenes`	A character vector of gene names to restrict the output p-values to.
`subsetCells`	A numeric vector of cell indeces to restrict the permutation to. Note that the selected cells should still contain at least one cell from each of the cell clusters contained in `$specScore`. Note this option is still experimental and might not give consistent results.
`cores`	A number greater than zero (1 by default) that indicates how many cores to use for parallelization using mclapply.
`seed`	The seed for random permutations.

Value

`getPValues` returns a list with the following components:

`permuteVal`	A sparse matrix with as many rows as genes and as many columns as `ncol($specScore) * numPerm`, containing the specificity score matrices for all permutations concatenated after each other column-wise.

`startIndeces`	A numeric vector of length `numPerm` that indicates the starting column for each performed permutation in `permuteVal`
`pval`	A matrix with as many rows as genes and columns as `ncol($specScore)`, containing the p-values for the null hypothesis that the gene is equally specific to all cell clusters is true.
`adjpval`	A matrix with the same size as `pval`, containing the corrected p-values using the method specified in `correctMethod`

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

Examples

```
data(sim)
sg = sortGenes(sim$exp, sim$cellType)
pp = getPValues(sg)

#obtain genes that are "differentially expressed" in at least one cluster
markers = names(which(apply(pp$adjpval, 1, function(x) any(x < 0.01))))
```

`kidneyTabulaMuris` *A subset of the Tabula Muris SmartSeq data, restricted to three cell types from Kidney.*

Description

A subset of the Tabula Muris data with 215 cells and 23341 genes. There are three cell types, endothelial cells: 122 cells, kidney collecting duct epithelial cell: 78 cells and leukoocytes 15 cells.

Usage

```
kidneyTabulaMuris
```

Format

A list with two components:

exp Gene expression sparse matrix of class dgCMatrix, with normalized log expression values. Rows are genes, columns are cells.

cellType A character vector containing the cell types.

References

Schaum et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. Nature, 562:7727, 2018

plotCorrelationHeat *plotCorrelationHeat*

Description

`plotCorrelationHeat` uses the specificity scores to correlate cell clusters with each other and plot a heatmap.

Usage

```
plotCorrelationHeat(gs, markers = NULL, corMethod = "pearson",
  colors = colorRampPalette(rev(c("orangered4", "orangered", "gray90",
  "dodgerblue", "dodgerblue4")))(n = 100), outs = FALSE,
  displayNumbers = TRUE)
```

Arguments

`gs`	A list containing `$specScore` sparse matrix. Typically, the output of `sortGenes`.
`markers`	Restrict correlation analysis to those genes. A character vector.
`corMethod`	Correlation method, will passed to `method` in the function `cor`.
`colors`	Color palette for drawing the heatmap
`outs`	Should the `pheatmap` object be returned? FALSE by default.
`displayNumbers`	Should correlation values be displayed on the heatmap? TRUE by default.

Value

If `outs` is TRUE, the pheatmap object will be returned.

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

Examples

```
data(kidneyTabulaMuris)
sg = sortGenes(kidneyTabulaMuris$exp, kidneyTabulaMuris$cellType)
plotCorrelationHeat(sg)

#user only marker genes and spearman correlation
mm = getMarkers(sg, quant = 0.95)
plotCorrelationHeat(sg, markers = mm$markers, corMethod = "spearman")

#do not plot correlation values
plotCorrelationHeat(sg, markers = mm$markers, corMethod = "spearman", displayNumbers = FALSE)
```

`plotMarkerHeat` *plotMarkerHeat*

Description

Plot a heatmap of expression values for a select set of genes. Optionally also cluster those genes.

Usage

```
plotMarkerHeat(exp, classes, markers,
  colors = colorRampPalette(rev(c("orangered4", "orangered", "gray90",
  "dodgerblue", "dodgerblue4")))(n = 100),
  newOrder = 1:length(unique(classes)), clusterGenes = FALSE,
  clusterGenesK = length(unique(classes)), averageCells = 0,
  outs = FALSE, plotheat = TRUE, gaps = TRUE, seed = 10)
```

Arguments

`exp`	A matrix of expression values. Typically the one supplied to `sortGenes`.
`classes`	A vector or factor of cell classes, whose length is equal to `ncol(exp)`.
`markers`	A character vector of gene names to plot in the heatmap.
`colors`	Color palette used for the heatmap.
`newOrder`	Reorder the clusters in the heatmap? See Examples.
`clusterGenes`	Cluster genes before plotting?
`clusterGenesK`	How many clusters should genes by clustered into? See Details.
`averageCells`	Plot averages of cells instead of individual cells. You use this when you have a large number of cells. See Details.
`outs`	Should gene cluster output and pheatmap object be returned? FALSE by default.
`plotheat`	Should the heatmap be drawn? TRUE by default.
`gaps`	Should the heatmap have gaps between cell types and gene clusters? TRUE by default.
`seed`	Randomization seed used for gene clustering initialization.

Details

`clusterGenesK` parameter controls the number of gene clusters when `clusterGenes` is `TRUE`. By default, it is equal to the number of cell types/clusters. This is usually a good initial guess, the optimal number of clusters (what does that even mean?) is typically somewhere around this value in my subjective experience.

By default, the heatmap plots every single cell in one column, this might take forever if you have a lot of cells (would say >10k) or it can crash when you do not have enough RAM. If so, it might be good to set `averageCells` to `n` where `n` is the number of cells you want to average. For example, if `averageCells = 10` every 10 cells will be averaged (without averaging across cell clusters) before plotting the heatmap. if `averageCells =< 1`, no averaging happens. Hints: (1) If you want one column per cell cluster, set `averageCells` to a very high number (larger than the number of cells in the biggest cluster). (2) Gene clustering occurs after cell averaging, so it might be useful if the gene clusters you get do not make much sense without averaging (averaging will smooth out zeros).

Value

If `outs` is TRUE, `plotMarkerHeat` returns a list containing:

`p`	The pheatmap object corresponding to the plot.
`gene_class_info`	Gene cluster assignments if gene clustering was requested. See Details.
`new_class_info`	The new order of cell clusters. See Examples.

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

Examples

```
data(kidneyTabulaMuris)
sg = sortGenes(kidneyTabulaMuris$exp, kidneyTabulaMuris$cellType)
mm = getMarkers(sg, quant = 0.999)

#this plots a heatmap without reordering genes
plotMarkerHeat(sg$inputMat, sg$inputClass, mm$markers)

#so now cluster genes and return the clustering: a better looking plot
pp = plotMarkerHeat(sg$inputMat, sg$inputClass, mm$markers, clusterGenes=TRUE, outs=TRUE)
pp$gene_class_info #cell clusters

#reorder cell clusters in the heatmap
plotMarkerHeat(sg$inputMat, sg$inputClass, mm$markers, clusterGenes=TRUE, newOrder = c(3,1,2))

#average every 4 cells to make a less intense heatmap
plotMarkerHeat(sg$inputMat, sg$inputClass, mm$markers, clusterGenes=TRUE, newOrder = c(3,1,2),
averageCells = 4)

#only cluster genes, do not make plots
pp = plotMarkerHeat(sg$inputMat, sg$inputClass, mm$markers, clusterGenes=TRUE,
outs=TRUE, plotheat=FALSE)
```

`plotMarkerScores` *plotMarkerScores*

Description

plotMarkerScores plots scatter plots of the specificity score and the gene specificity Shannon index to investigate `getMarkers` results.

Usage

```
plotMarkerScores(mm, gs, markers = mm$markers, colors = c("#99999920",
  "orangered4"))
```

Arguments

`mm`	A list. The output of `getMarkers`
`gs`	A list. The output of `sortGenes`
`markers`	A character vector containing the genes to highlight in the scatter plot.
`colors`	A color palette to use for the scatter plot.

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

Examples

```
data(kidneyTabulaMuris)
sg = sortGenes(kidneyTabulaMuris$exp, kidneyTabulaMuris$cellType)
mm = getMarkers(sg,quant=0.99)
plotMarkerScores(mm, sg)
```

`plotTopMarkerHeat` *plotTopMarkerHeat*

Description

Plot a heatmap of expression values for the top genes in each cluster, defined by `sortGenes()`.

Usage

```
plotTopMarkerHeat(sg, top_n = 10,
  colors = colorRampPalette(rev(c("orangered4", "orangered", "gray90",
  "dodgerblue", "dodgerblue4")))(n = 100),
  newOrder = 1:length(unique(sg$inputClass)), averageCells = 0,
  gaps = TRUE, outs = FALSE, plotheat = TRUE)
```

Arguments

`sg`	A list, typically the output of `sortGenes`.
`top_n`	The number of top genes to plot for each cluster.
`colors`	Color palette used for the heatmap.
`newOrder`	Reorder the clusters in the heatmap? See Examples.
`averageCells`	Plot averages of cells instead of individual cells. You can use this when you have a large number of cells. See Details.
`gaps`	Should the heatmap have gaps between cell types and gene clusters? TRUE by default.
`outs`	Should the top genes names be returned? FALSE by default.
`plotheat`	Should the heatmap be drawn? TRUE by default.

Details

`plotTopMarkerHeat` is a convenience wrapper around `plotMarkerHeat` that plots a heatmap of the top `top_n` (10 by default) genes in each cell cluster. Unlike `plotMarkerHeat`, `plotTopMarkerHeat` takes the output of `sortGenes` as the only required input.

Value

If `outs` is TRUE, `plotMarkerHeat` returns a list

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

Examples

```
data(kidneyTabulaMuris)
sg = sortGenes(kidneyTabulaMuris$exp, kidneyTabulaMuris$cellType)
plotTopMarkerHeat(sg)
```

`sim` *Simulated single cell RNA-seq data with 500 cells.*

Description

A simulated dataset containing 500 cells and 3999 genes. There are two cell types, one with 245 cells and the other with 255 cells.

Usage

```
sim
```

Format

A list with two components:

exp Gene expression matrix, with raw counts. Rows are genes, columns are cells

cellType A numeric vector containing the cell types

`sortGenes` *sortGenes*

Description

sortGenes is the main function of the genesorteR package. It takes a gene expression matrix and cell cluster assignments. It binarizes the expression matrix and calculates empirical statistics on gene expression in each cluster.

Usage

```
sortGenes(x, classLabels, binarizeMethod = "median",
  returnInput = TRUE, cores = 1)
```

Arguments

`x`	A numeric (sparse) matrix. It will be coerced to a dgCMatrix sparse matrix. Rows represent genes, columns represent cells.
`classLabels`	A numeric or character vector or a factor of the same length as ncol(x) that represents cell cluster assignments. It will be coerced to a factor whose levels are the cell cluster names.
`binarizeMethod`	Either "median" (default) or "naive". See Details.
`returnInput`	Return the input matrix and cell classes? `TRUE` by default.
`cores`	A number greater than zero (1 by default) that indicates how many cores to use for parallelization using mclapply.

Details

When `binarizeMethod` is "median", expression matrix binarization is done by estimating a cutoff value uniformly on all values in the matrix. This is equal to the median of all non-zero entries in the matrix and is returned in `cutoff`. When binarizeMethod is "naive", all non-zero entries are kept and the minimum value of non-zero entries is returned in `cutoff`. When the input matrix `x` has already been binarized, set `binarizeMethod` to "naive".

The specificity score matrix `specScore` is calculated as `condGeneProb * postClustProb`. Therefore, it simply balances the posterior probability of observing a cell cluster given the gene (gene-cluster specificity) with its conditional probability given the cluster (a measure of gene expression). This ensures that highly specific genes are also highly expressed. The `specScore` matrix is considered the main output of this function, and on which many of the remaining calculations by other functions in the `genesorteR` package are performed. The values in this matrix can be used to rank features (genes in scRNA-Seq) in clusters.

`sortGenes` can in principle be applied to both a raw count matrix or a normalized log-count expression matrix.

Value

`sortGenes` returns a list with the following components:

`binary`	The binarized gene expression matrix. A sparse matrix of class dgCMatrix of the same size as "`x`".
`cutoff`	The cutoff value used to binarize the gene expression matrix. Anything lower than this value was set to zero.
`removed`	A numeric vector containing the row indeces of genes that were removed because they were not expressed in any cells. If none were removed, this will be `NULL`.
`geneProb`	A numeric vector whose length is equal to nrow(binary) that lists the fraction of cells in which a gene was detected.
`condGeneProb`	A sparse matrix of class dgCMatrix with as many rows as nrow(binary) and as many columns as the number of cell clusters. It includes the conditional probability of observing a gene in a cluster.
`postClustProb`	A sparse matrix of class dgCMatrix with the same size as `condGeneProb`, containing the posterior probability that a cell belongs to a certain cluster or type given that the gene was observed.
`specScore`	A sparse matrix of class dgCMatrix with the same size as `condGeneProb`, containing a specificity score for each gene in each cell cluster. See Details.

`classProb`	A numeric vector whose length is equal to the number of cell clusters, containing the fraction of cells belonging to each cluster.
`inputMat`	the input `x` matrix, after being coerced to a sparse matrix of class dgCMatrix.
`inputClass`	the input `classLabels` after being coerced to a factor.

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

Examples

```
#basic functionality
set.seed(1234)
exp = matrix(sample(0:20,1000,replace=TRUE), ncol = 20)
rownames(exp) = sapply(1:50, function(x) paste0("g", x))
classLab = sample(c("cell type 1","cell type 2"),20,replace=TRUE)
sg = sortGenes(exp, classLab)

#naive binarization keeps any non-zero input in the input matrix
sg_naive = sortGenes(exp, classLab, binarizeMethod = "naive")
```

`write.files` *write.files*

Description

write.files saves gene-cluster sorting information obtained from `sortGenes` to disk.

Usage

```
write.files(gs, prefix = "genesorteROuts", markers = NULL,
  eachCluster = FALSE)
```

Arguments

`gs`	A list containing `$specScore`, `$postClustProb` and `$specScore` sparse matrices. Typically the output of `sortGenes()`.
`prefix`	The prefix for saving the files.
`markers`	Additionally, output files restricted to these genes. A character vector.
`eachCluster`	Also write individual files for each cluster?

Details

Three files will be saved to disk, including the specificity score matrix, the posterior cluster probability given the gene and the conditional probability of observing the gene in a cluster. If `markers` is not NULL, three additional files will be created and indicated with "_markers" in their name. If `eachCluster` is `TRUE`, additional files (one per cluster will be written which includes all genes and their scaled specificity score for each cluster, with genes sorted by their score.

Author(s)

Mahmoud M Ibrahim <mmibrahim@pm.me>

Examples

```
data(sim)
sg = sortGenes(sim$exp, sim$cellType)
## Not run: write.files(sg) #write all files for all genes.
## Not run: write.files(sg, markers = c("g1","g2")) #additionally write files that are restricted to genes g1
```

Index