Package ‘mfishtools’

May 6, 2022

Type Package

Title Building Gene Sets and Mapping mFISH Data

Version 0.0.2

Author Jeremy Miller <jeremym@alleninstitute.org>

Maintainer Jeremy Miller <jeremym@alleninstitute.org>

Description This repository includes code for gene selection for spatial transcriptomics methods and for mapping of spatial transcriptomics (or RNA-Seq data) onto a RNA-Seq reference. Specific topics include:
1) Correlation-based mapping of cells to reference cell types
2) Iterative building of gene panels a greedy algorithm with pre-defined constraints
3) Visualizations related to gene mapping a gene panel selection

License What license is it under?

Encoding UTF-8

LazyData true

Imports dendextend (>= 1.7.0),
gplots (>= 3.0.1),
pdist (>= 1.2),
matrixStats (>= 0.53.1),
Rtsne (>= 0.13),
ggplot2 (>= 2.2.1),
scrattch.vis (>= 0.0),
tasic2016data (>= 0.0),
dplyr (>= 0.3.4),
WGCNA (>= 1.0)

RoxygenNote 7.1.1

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

buildMappingBasedMarkerPanel

Greedy algorithm for building marker gene panel

Description

This is the primary function that iteratively builds a marker gene panel, one gene at a time by iteratively adding the most informative gene to the existing gene panel.

Usage

```
buildMappingBasedMarkerPanel(
  mapDat,
  medianDat = NA,
  clustersF = NA,
  panelSize = 50,
  subSamp = 20,
  maxFcGene = 1000,
  qMin = 0.75,
  seed = 10,
  currentPanel = NULL,
  panelMin = 5,
  writeText = TRUE,
  corMapping = TRUE,
  optimize = "FractionCorrect",
  clusterDistance = NULL,
  clusterGenes = NULL,
  dend = NULL,
  percentSubset = 100
)
```

Arguments

`mapDat`	normalized data of the mapping (=reference) data set.
`medianDat`	representative value for each leaf. If not entered, it is calculated
`clustersF`	cluster calls for each cell.
`panelSize`	number of genes to include in the marker gene panel
`subSamp`	number of random nuclei to select from each cluster (to increase speed); set as NA to not subsample
`maxFcGene`	maximum number of genes to consider at each iteration (to increase speed)
`qMin`	minimum quantile for fold change comparison (between 0 and 1, higher = more specific marker genes are included)
`seed`	for reproducibility
`currentPanel`	starting panel. Default is NULL.
`panelMin`	if there are fewer genes than this, the top number of these genes by fc rank are set as the starting panel. Cannot be less than 2.
`writeText`	should gene names and marker scores be output (default TRUE)

`corMapping`	if TRUE (default) map by correlation; otherwise, map by Euclidean distance (not recommended)
`optimize`	if 'FractionCorrect' (default) will seek to maximize the fraction of cells correctly mapping to final clusters if 'CorrelationDistance' will seek to minimize the total distance between actual cluster calls and mapped clusters if 'DendrogramHeight' will seek to minimize the total dendrogram height between actual cluster calls and mapped clusters
`clusterDistance`	only used if optimize='CorrelationDistance'; a matrix (or vector) of cluster distances. Will be calculated if NULL and if clusterGenes provided. (NOTE: order must be the same as medianDat and/or have column and row names corresponding to clusters in clustersF)
`clusterGenes`	a vector of genes used to calculate the cluster distance. Only used if optimize='CorrelationDistance' and clusterDistance=NULL.
`dend`	only used if optimize='DendrogramHeight' dendrogram; will error out of not provided
`percentSubset`	for each iteration the function can subset the set of possible genes to speed up the calculation.

Value

an ordered character vector corresponding to the marker gene panel

---

`buildPanel_oneCluster` *Build panel for one cluster (beta)*

Description

This UNTESTED function finds the best small marker panel for marking a single cluster, using proportion difference as the metric for determining the starting panel.

Usage

```
buildPanel_oneCluster(
  mapDat,
  clustersF,
  medianDat = NA,
  propIn = NA,
  clust = as.character(clustersF[1]),
  subSamp = NA,
  seed = 10,
  maxSize = 20,
  dexCutoff = 0.001,
  topGeneCount = 100
)
```

Arguments

`mapDat`	normalized data of the mapping (=reference) data set.
`clustersF`	cluster calls for each cell.
`medianDat`	median value for each leaf
`propIn`	proportions of cells with expression > 1 in each leaf
`clust`	which cluster to target?
`subSamp`	number of random nuclei to select from each cluster, EXCEPT the target cluster; set as NA to not subsample
`seed`	for reproducibility
`maxSize`	maximum size of marker gene panel
`dexCutoff`	criteria for stopping: when improvement in fraction of cells properly mapped dips below this value
`topGeneCount`	number of top genes by proportion to consider

Value

a matrix of the top marker genes for each cluster. Output matrix includes five columns: clust = cluster; panel = ordered genes in the panel for that cluster; onCorrect = fraction of correctly assigned cells in cluster; offCorrect = fraction of cells correctly assigned outside of cluster; dexTotal = additional dex explained by last gene added.

`buildQualityTable` *Correct mapping at different tree heights*

Description

This function takes as input an ordered set of marker genes (e.g., from at iterative algorithm, and returns an table showing the fraction of cells correctly mapped to a similar cell type (as defined by the heights parameter). A height of 1 indicates correct mapping to the leaf.

Usage

```
buildQualityTable(
  orderedGenes,
  dend,
  mapDat,
  medianDat,
  clustersF,
  minVal = 2,
  heights = c((0:100)/100),
  verbose = FALSE
)
```

Arguments

`orderedGenes`	an ordered list of input genes (e.g. from an iterative algorithm)
`dend`	dendrogram for mapping.
`mapDat`	normalized data of the mapping (=reference) data set.
`medianDat`	median value for each leaf
`clustersF`	cluster calls for each cell
`minVal`	minimum number of genes to consider from the list in the mapping
`heights`	height in the tree to look at
`verbose`	whether or not to show progress in the function

Value

a matrix of fractions of cells correctly mapped for different tree heights (columns) and different gene panels (rows)

`buildTreeFromGenePanel`
Build and plot dendrogram from gene panel

Description

Build and plot a dendrogram using correlation-based average linkage hierarchical clustering and only using a specified set of genes. The output is the expected accuracy of mapping to each node in the tree, which gives an idea of the best-case expected results for mFISH analysis.

Usage

```
buildTreeFromGenePanel(
  dend = NA,
  refDat = NA,
  mapDat = refDat,
  medianDat = NA,
  requiredGenes = 2,
  clusters = NA,
  mappedAsReference = FALSE,
  genesToMap = rownames(mapDat),
  plotdendro = TRUE,
  returndendro = TRUE,
  mar = c(12, 5, 5, 5),
  main = NULL,
  ylab = NULL,
  use = "p",
  ...
)
```

Arguments

`dend`	dendrogram for mapping. Ignored if medianDat is passed
`refDat`	normalized data of the REFERENCE data set. Ignored if medianDat is passed
`mapDat`	normalized data of the MAPPING data set. Default is to map the data onto itself.
`medianDat`	representative value for each leaf and node. If not entered, it is calculated
`requiredGenes`	minimum number of genes required to be expressed in a cluster (column of medianDat) for the cluster to be included (default=2)
`clusters`	cluster calls for each cell
`mappedAsReference`	if TRUE, returns the fraction of cells mapped to a node which are were orginally clustered from that node; if FALSE (default) returns the fraction of cells clustered under a node which are mapped to the correct node.
`genesToMap`	which genes to include in the correlation mapping
`plotdendro`	should the dendrogram be plotted (default = TRUE)
`mar`	margins (for use with par)
`main, ylab`	add title and labels to plot (default is NULL)
`use, ...`	additional parameters for cor
`returnDendro`	should the dendrogram be returned (default = TRUE)

Value

a list where the first entry is the resulting tree and the second entry is the fraction of cells correctly mapping to each node using the inputted gene panel.

`cellToClusterMapping_byCor`

Return top mapped correlation-based cluster and confidence

Description

Primary function for doing correlation-based mapping to cluster medians and also reporting the correlations and confidences. This is wrapper for getTopMatch and corTreeMapping.

Usage

```
cellToClusterMapping_byCor(
  medianDat,
  mapDat,
  refDat = NA,
  clusters = NA,
  genesToMap = rownames(mapDat),
  use = "p",
  method = "p",
  ...
)
```

Arguments

medianDat	representative value for each leaf and node. If not entered, it is calculated
mapDat	normalized data of the MAPPING data set. Default is to map the data onto itself.
refDat	normalized data of the REFERENCE data set. Ignored if medianDat is passed
clusters	cluster calls for each cell. Ignored if medianDat is passed
genesToMap	which genes to include in the correlation mapping
use	additional parameter for cor (use='p' as default)
method	additional parameter for cor (method='p' as default)
...	not used

Value

data frame with the top match and associated correlation

---

cellToClusterMapping_byRank

Cell-based cluster mapping

Description

Maps cells to clusters by correlating every mapped cell with every reference cell, ranking the cells by correlation, and the reporting the cluster with the lowest average rank.

Usage

```
cellToClusterMapping_byRank(
  mapDat,
  refDat,
  clustersF,
  genesToMap = rownames(mapDat),
  mergeFunction = rowMedians,
  useRank = TRUE,
  use = "p",
  method = "p"
)
```

Arguments

mapDat	normalized data of the MAPPING data set.
refDat	normalized data of the REFERENCE data set
clustersF	factor indicating which cluster each cell type is actually assigned to in the reference data set
genesToMap	character vector of which genes to include in the correlation mapping
mergeFunction	function for combining ranks; the tested choices are rowMeans or rowMedians (default)
useRank	use the rank of the correlation (default) or the correlation itself to determine the top cluster
use	additional parameter for cor (use='p' as default)
method	additional parameter for cor (method='p' as default)

Value

a two column data matrix where the first column is the mapped cluster and the second column is a confidence call indicating how close to the top of the ranked list cells of the assigned cluster were located relative to their best possible location in the ranked list. This confidence score seems to be a bit more reliable than correlation at determining how likely a cell in a training set is to being correctly assigned to the training cluster.

`corTreeMapping` *Correlation-based cluster mapping*

Description

Primary function for doing correlation-based mapping to cluster medians. This is wrapper for cor and returns a correlation matrix.

Usage

```
corTreeMapping(
  mapDat,
  medianDat,
  dend = NULL,
  refDat = NA,
  clusters = NA,
  genesToMap = rownames(mapDat),
  use = "p",
  method = "p"
)
```

Arguments

`mapDat`	normalized data of the MAPPING data set. Default is to map the data onto itself.
`medianDat`	representative value for each leaf and node. If not entered, it is calculated
`dend`	dendrogram for mapping. If provided, correlations to nodes are also returned
`refDat`	normalized data of the REFERENCE data set. Ignored if medianDat is passed
`clusters`	cluster calls for each cell. Ignored if medianDat is passed
`genesToMap`	which genes to include in the correlation mapping
`use`	additional parameter for cor (use='p' as default)
`method`	additional parameter for cor (method='p' as default)

Value

matrix with the correlation between expression of each cell and representative value for each leaf and node

corTreeMapping_withFilter

Correlation between nodes and leafs (deprecated)

Description

Returns the correlation between expression of each cell and representative value for each node and leaf. NOTE: this function is unstable and will eventually be merged with corTreeMapping.

Usage

```
corTreeMapping_withFilter(
  dend = NA,
  refDat = NA,
  mapDat = refDat,
  medianExpr = NA,
  propExpr = NA,
  filterMatrix = NA,
  clusters = NA,
  numberOfGenes = 1200,
  outerLimitGenes = 7200,
  rankGeneFunction = function(x) getBetaScore(x, returnScore = FALSE),
  use = "p",
  ...
)
```

Arguments

`dend`	dendrogram for mapping. Ignored if medianDat is passed
`refDat`	normalized data of the REFERENCE data set. Ignored if medianExpr and propExpr are passed
`mapDat`	normalized data of the MAPPING data set. Default is to map the data onto itself.
`medianExpr`	representative value for each leaf. If not entered, it is calculated
`propExpr`	proportion of cells in each type expressing a given gene. If not entered, it is calculated
`filterMatrix`	a matrix of TRUE/FALSE values to indicate whether a given cluster is possible
`clusters`	cluster calls for each cell. Ignored if medianExpr and propExpr are passed
`numberOfGenes`	how many variables genes
`outerLimitGenes`	choose different numberOfGenes per cell from the top overall outerLimitGenes (to speed up function)
`use, ...`	additional parameters for cor
`genesToMap`	which genes to include in the correlation mapping

Value

a matrix of correlation values with rows as mapped cells and columns as clusters

`distTreeMapping` *(Euclidean) distance mapping*

Description

Returns the distance between expression of each cell and representative value for each node and leaf (default is based on euclidean distance). In our hands this is does not work very well.

Usage

```
distTreeMapping(
  dend = NA,
  refDat = NA,
  mapDat = refDat,
  medianDat = NA,
  clusters = NA,
  genesToMap = rownames(mapDat),
  returnSimilarity = TRUE,
  use = "p",
  ...
)
```

Arguments

`dend`	dendrogram for mapping. Ignored if medianDat is passed
`refDat`	normalized data of the REFERENCE data set. Ignored if medianDat is passed
`mapDat`	normalized data of the MAPPING data set. Default is to map the data onto itself.
`medianDat`	representative value for each leaf and node. If not entered, it is calculated
`clusters`	cluster calls for each cell. Ignored if medianDat is passed
`genesToMap`	which genes to include in the correlation mapping
`returnSimilarity`	FALSE to return distance, TRUE to return something like a similarity
`use, ...`	additional parameters for dist (for back-compatiblity; doesn't work)

Value

matrix of Euclidean distances between cells (rows) and clusters (columns)

`filterByClass` *Filter by meta-data*

Description

Return a filter of TRUE/FALSE values for a given piece of meta-data (e.g., broad class).

Usage

```
filterByClass(
  classVector,
  sampleInfo,
  classColumn = "cluster_type_label",
  clusterColumn = "cluster_label",
  threshold = 0.1
)
```

Arguments

`classVector`	vector corresponding to the class information for filtering (e.g., vector of label calls)
`sampleInfo`	matrix of sample information with rows corresponding to cells and columns corresponding to meta-data
`classColumn`	column name of class information
`clusterColumn`	column name of cluster information
`threshold`	minimum fraction of cluster cells from a given class to be considered present

Value

a matrix of filters with rows as clusters and columns as classes with entries of TRUE or FALSE indicating whether cells from a given class can assigned to that cluster, given threshold.

`filterCells` *Filter (subset) fishScaleAndMap object*

Description

Subsets all components in a fishScaleAndMap object

Usage

```
filterCells(datFish, subset)
```

Arguments

`datFish`	a fishScaleAndMap output list
`subset`	a boolean or numeric vector of the elements to retain

Value

a fishScaleAndMap output subsetted to the requested elements

filterPanelGenes *Filter genes for spatial transcriptomics panel*

Description

Returns a set of genes for inclusion in a spatial transcriptomics panel based on a series of hard-coded and user-defined constraints

Usage

```
filterPanelGenes(
  summaryExpr,
  propExpr = summaryExpr,
  onClusters = 1:dim(summaryExpr)[2],
  offClusters = NULL,
  geneLengths = NULL,
  startingGenes = c("GAD1", "SLC17A7"),
  numBinaryGenes = 500,
  minOn = 10,
  maxOn = 250,
  maxOff = 50,
  minLength = 960,
  fractionOnClusters = 0.5,
  onThreshold = 0.5,
  excludeGenes = NULL,
 excludeFamilies = c("LOC", "LINC", "FAM", "ORF", "KIAA", "FLJ", "DKFZ", "RIK", "RPS",
    "RPL", "\\-")
)
```

Arguments

`summaryExpr`	Matrix of summarized expression levels for a given cluster. Typically the median or mean should be used. Rows are genes and columns are samples. ROW NAMES MUST BE GENE SYMBOLS!
`propExpr`	Proportion of cells expressed in each cluster for use with binary score calculation (default = summaryExpr, which is not recommended)
`onClusters`	Vector indicating which clusters should be included in the gene panel (default is all clusters. Can be logical or numeric, or a character string of cluster names)
`offClusters`	Vector indidicating from which clusters expression should be avoided
`numBinaryGenes`	Number of genes to include in the final panel. Genes are sorted by binary score using ‘getBetaScore‘ and this number of genes are chosen (default = 500)
`minOn`	Minimum summary expression level in most highly expressed "on" cluster (default = 10)
`maxOn`	Maximum summary expression level in most highly expressed "on" cluster (default = 250)
`maxOff`	Maximum summary expression level in most highly expressed "off" cluster (default = 50)
`minLength`	Minimum gene length for marker gene selection. Ignored if geneLength is not provided (default = 960)

`fractionOnClusters`	What is the maximum fraction of clusters in which a gene can be expressed (as defined by propExpr>onThreshold; default = 0.5). This prevents nearly ubiquitous genes from selection
`onThreshold`	What fraction of cells need to have expression for a gene to be defined as expressed (default = 0.5)
`excludeGenes`	Which genes should be excluded from the analysis (default is none)
`excludeFamilies`	Which gene classes or families should be excluded from the analysis? More specifically, any gene that contain these strings of characters anywhere in the symbol will be excluded (default is "LOC","LINC","FAM","ORF","KIAA","FLJ","DKFZ","RIK","R ").
`geneLength`	Optional vector of gene lengths in same order as summaryExpr. Default is NULL

Value

A character vector of genes meeting all constraints

`fishScaleAndMap` *Scale mFISH data and map to RNA-seq reference*

Description

This function is a wrapper for several other functions which aim to scale mFISH data to more closely match RNA-seq data and then map the mFISH data to the closest reference classes. There are several parameters allowing flexability in filtering and analysis.

Usage

```
fishScaleAndMap(
  mapDat,
  refSummaryDat,
  genesToMap = NULL,
  mappingFunction = cellToClusterMapping_byCor,
  transform = function(x) x,
  noiselevel = 0,
  scaleFunction = quantileTruncate,
  omitGenes = NULL,
  metadata = data.frame(experiment = rep("all", dim(mapDat)[2])),
  integerWeights = NULL,
  binarize = FALSE,
  binMin = 0.5,
  ...
)
```

Arguments

`mapDat`	normalized data of the MAPPING data set. Default is to map the data onto itself.
`refSummaryDat`	normalized summary data of the REFERENCE data set (e.g., what to map against)
`genesToMap`	which genes to include in the mapping (calculated in not entered)
`mappingFunction`	which function to use for mapping (default is cellToClusterMapping_byCor) The function must include at least two parameters with the first one being mapped data and the second data the reference. Additional parameters are okay. Output must be a data frame where the first value is a mapped class. Additional columns are okay and will be returned)
`transform`	function for transformation of the data (default in none)
`noiselevel`	scalar value at or below which all values are set to 0 (default is 0)
`scaleFunction`	which function to use for scaling mapDat to refSummaryDat (default is setting 90th quantile of mapDat to max of refSummaryDat and truncating higher mapDat values)
`omitGenes`	genes to be included in the data frames but excluded from the mapping
`metadata`	a data frame of possible metadata (additional columns are okay and ignored): **area** a vector of cell areas for normalization; **experiment** a vector indicating if multiple experiments should be scaled separately; **x,y** x (e.g., parallel to layer) and y (e.g., across cortical layers) coordinates in tissue
`integerWeights`	if not NULL (default) a vector of integers corresponding to how many times each gene should be counted as part of the correlation. This is equivalent to calculating a weighted correlation, but only allows for integer weight values (for use with cor).
`binarize`	should the data be binarized? (default=FALSE)
`binMin`	minimum ON value for the binarized matrix (ignored if binarize=FALSE)
`...`	additional parameters for passthrough into other functions

Value

a list with the following entrees:

mapDat mapDat data matrix is passed through

scaleDat scaled mapDat data matrix

mappingResults Results of the mapping and associated confidence values (if any)

metadata=metadata metadata is passed through unchanged

scaledX/Y scaled x and y coordinates (or unscaled if scaling was not performed)

fractionCorrectPerNode

Fraction of correct calls per node

Description

This function returns the fraction correctly assigned to each node (as defined that the actual and predicted cluster are both in the same node)

Usage

```
fractionCorrectPerNode(
  dendIn,
  clActual,
  clPredict,
  minCount = 0.1,
  defaultSum = -1,
  out = NULL
)
```

Arguments

dendIn	dendrogram for mapping. Ignored if minimizeHeight=FALSE
clActual	character vector of actual cluster assignments
clPredict	character vector of predicted cluster assignments
minCount	set to 0 results from clusters with fewer than this number of cells (default is to consider all clusters)
defaultSum	value to return in cases where there are fewer than minCount cells in the actual cluster (e.g., cases that aren't considered at all)
out	required for recursive function. Do not set!

Value

matrix of two columns: (1) node name and (2) the fraction of cells in that node that are correctly assigned

fractionCorrectWithGenes

Fraction of cells correctly assigned

Description

This function takes as input an ordered set of marker genes (e.g., from at iterative algorithm), and returns a vector showing the fraction of cells correctly mapped.

Usage

```
fractionCorrectWithGenes(
  orderedGenes,
  mapDat,
  medianDat,
  clustersF,
  verbose = FALSE,
  plot = TRUE,
  return = TRUE,
  ...
)
```

Arguments

`orderedGenes`	an ordered list of input genes (e.g. from an iterative algorithm)
`mapDat`	normalized data of the mapping (=reference) data set.
`medianDat`	median value for each leaf
`clustersF`	cluster calls for each cell
`verbose`	whether or not to show progress in the function
`plot`	if TRUE, plotCorrectWithGenes is run
`return`	if TRUE, the value is returned
`...`	parameters passed to plotCorrectWithGenes (if plot=TRUE)

Value

a vector showing the fraction of cells correctly mapped to each cluster

---

`generateMultipleCellReferenceSet` *Generate reference set of pseudo-cells*

Description

Creates a new reference set as input for cellToClusterMapping_byRank, where each 'cell' is the combiniation of several cells and this is run several times using different subsets of cells.

Usage

```
generateMultipleCellReferenceSet(
  refDat,
  clustersF,
  genesToUse = rownames(refDat),
  cellsPerMerge = 5,
  numberOfMerges = 10,
  mergeFunction = rowMedians,
  seed = 1
)
```

Arguments

`refDat`	normalized data of the REFERENCE data set
`clustersF`	factor indicating which cluster each cell type is actually assigned to in the reference data set
`cellsPerMerge`	Number of cells to include in each combo cell
`numberOfMerges`	Number of combo cells to include per cell type
`mergeFunction`	function for combining cells into combo cells (use rowMeans or rowMedians)
`seed`	for resproducibility
`genesToMap`	which genes to include in the correlation mapping

Value

list where first element is data matrix of multi-cells by genes and second element is a vector of corresponding clusters

---

`getBetaScore` *Get binary (aka beta) score*

Description

Returns a beta score which indicates the binaryness of a gene across clusters. High scores (near 1) indicate that a gene is either on or off in nearly all cells of every cluster. Scores near 0 indicate a cells is non-binary (e.g., not expressed, ubiquitous, or randomly expressed). This value is used for gene filtering prior to defining clustering.

Usage

```
getBetaScore(propExpr, returnScore = TRUE, spec.exp = 2)
```

Arguments

`propExpr`	a matrix of proportions of cells (rows) in a given cluster (columns) with CPM/FPKM > 1 (or 0, HCT uses 1)
`returnScore`	if TRUE returns the score, if FALSE returns the ranks
`spec.exp`	scaling factor (recommended to leave as default)

Value

returns a numeric vector of beta score (or ranks)

getBranchList *Branch list*

Description

Returns branches of a dendrogram in a specific format

Usage

```
getBranchList(
  dend,
  branches = list(),
  allTips = as.character(dend %>% labels())
)
```

Arguments

`dend`	dendrogram for mapping. Ignored if medianDat is passed
`branches`	do not change from default
`allTips`	do not change from default

Value

a list of branch information for use with leafToNodeMedians

getConfusionMatrix *Confusion matrix*

Description

Returns a confusion matrix of the found (mapped) vs. real (assigned) clusters.

Usage

```
getConfusionMatrix(realCluster, foundCluster, proportions = TRUE)
```

Arguments

`realCluster`	character vector of assigned clusters
`foundCluster`	character vector of mapped clusters
`proportions`	FALSE if the counts are to be returned and TRUE if the proportions are to be returned

`getDend` *Build a dendrogram from gene panel*

Description

Build a dendrogram from an inputted data matrix.

Usage

```
getDend(dat, distFun = function(x) return(as.dist(1 - WGCNA::cor(x))), ...)
```

Arguments

`dat`	matrix of values (e.g., genes x clusters) for calculating the dendrogram
`distFun`	function for calculating distance matrix (default is correlation-based)
`...`	additional variables for distFun

Value

dendrogram

`getNodeHeight` *Get node height*

Description

Returns the heights of each node, scaled from 0 (top) to 1 (leafs); this is a wrapper for dendextend functions

Usage

```
getNodeHeight(tree)
```

Arguments

`tree`	a dendrogram object

Value

a vector of node heights

`getTopMatch` *Get top leaf match*

Description

Returns the top leaf match for each cell and the corresponding fraction mapping there.

Usage

```
getTopMatch(memb.cl)
```

Arguments

`memb.cl`	membership scores for each leaf

Value

a matrix where first column is found cluster and second column is confidence score

`get_subtree_label` *Gets subtree labels for lca function.*

Description

Gets subtree labels for lca function.

Usage

```
get_subtree_label(dend)
```

Arguments

`dend`	a cluster dendrogram

Value

vector of subtree labels

labelDend *Label dendrogram nodes*

Description

Add numeric node labels to a dendrogram.

Usage

```
labelDend(dend, n = 1)
```

Arguments

`dend`	dendrogram object
`distFun`	starting numeric node value (default=1)

Value

a list where the first item is the new dendrogram object and the second item is the final numeric node value.

layerFraction *Layer weights per cell*

Description

Returns a numeric vector saying how to weight a particular cell for each layer. This is a wrapper for smartLayerAllocation

Usage

```
layerFraction(layerIn, useLayer = "L1", cluster = NA, ...)
```

Arguments

`layerIn`	a list corresponding to all layers of dissection for a given sample
`useLayer`	target layer
`cluster`	if passed the weights are smartly allocated based on laminar distributions by cluster
`...`	additional variables for smartLayerAllocation

Value

numeric vector with weights for cells in input layer

`layerScale` *Fraction of cells per layer*

Description

Determines the expected proportions in each layer based on input

Usage

```
layerScale(layerIn, layerNm = c("L1", "L2/3", "L4", "L5", "L6"), scale = TRUE)
```

Arguments

`layerIn`	a list corresponding to all layers of dissection for a given sample
`layerNm`	names of all layers. set to NULL to have this calculated
`scale`	if TRUE (default), scale to the total number of cells

Value

vector indicating the fraction of cells in each layerNm layer

`lca` *Get lowest common ancestor (defined cluster pairs)*

Description

Maps a cluster back up the tree to the first node where the mapped and correct clusters agree.

Usage

```
lca(dend, l1, l2, l = rep(attr(dend, "label"), length(l1)))
```

Arguments

`dend`	a cluster dendrogram
`l1`	a vector of node labels
`l2`	a second fector of node labels (of the same length as l1)
`l`	do not adjust; required for recursive function

Value

The function will return a vector for lowest common ancestor for every pair of nodes in l1 and l2 `leafToNodeMedians` *Return mean node expression*

Description

Define expression at a node as the MEAN expression for each leaf as default (using the median removes all specific marker genes!)

Usage

```
leafToNodeMedians(dend, medianDat, branches = getBranchList(dend), fnIn = mean)
```

Arguments

`dend`	dendrogram for mapping. Ignored if medianDat is passed
`medianDat`	median expression data at each node
`branches`	a particular format of branch information from the dendrogram structure
`fnIn`	function to use to wrap up to the node level (default = mean)

Value

a matrix of mean node expression (rows=genes, columns=nodes)

`makeLCAtable` *Get lowest common ancestor (all cluster pairs in tree)*

Description

Calculates the vector for lowest common ancestor for every pair of leaves in a tree and returns a vector in a specific format for faster look-up.

Usage

```
makeLCAtable(dend, includeInternalNodes = FALSE, verbose = FALSE)
```

Arguments

`dend`	a cluster dendrogram
`includeInternalNodes`	should internal nodes be included in the output?
`verbose`	if TRUE, status will be printed to the screen, since function is relatively slow for large trees (default FALSE)

Value

The function will return a vector for lowest common ancestor for every pair of leaves in dend. Vector names are l1||||l2 for string parsing in other functions.

map_dend *Tree-based mapping*

Description

Returns the mapping membership of each cell to each node and leaf using a tree-based method. This is a wrapper function for map_dend.

Usage

```
map_dend(
  dend,
  cl,
  dat,
  map.dat,
  select.cells,
  p = 0.8,
  low.th = 0.2,
  default.markers = NULL
)
```

Arguments

`dend`	dendrogram for mapping
`cl`	factor indicating which cluster each cell type is actually assigned to in the reference data set
`dat`	normalized data of the REFERENCE data set
`map.dat`	normalized data of the MAPPING data set. Default is to map the data onto itself.
`p`	proportion of marker genes to include in each iteration of the mapping algorithm.
`low.th`	the minimum difference in Pearson correlation required to decide on which branch to map to. otherwise, a random branch is chosen.
`default.markers`	not used

Value

a matrix of confidence scores (from 0 to 100) with rows as cells and columns as tree node/leafs. Values indicate the fraction of permutations in which the cell mapped to that node/leaf using the subset of cells/genes in map_dend

`mergeFish` *Merge two fishScaleAndMap objects*

Description

Merges all components of two fishScaleAndMap objects to create a new one. Note: only meta-data and mappingResults that is present in BOTH objects will be returned.

Usage

```
mergeFish(datFish1, datFish2)
```

Arguments

`datFish1`	a fishScaleAndMap output list
`datFish2`	a second fishScaleAndMap output list.

Value

a new fishScaleAndMap output list with the two original ones merged

`mfishtools` *mfishtools: Building Gene Sets and Mapping mFISH Data.*

Description

This repository includes code for gene selection for spatial transcriptomics methods and for mapping of spatial transcriptomics (or RNA-Seq data) onto a RNA-Seq reference. Specific topics include: 1) Correlation-based mapping of cells to reference cell types 2) Iterative building of gene panels a greedy algorithm with pre-defined constraints 3) Visualizations related to gene mapping a gene panel selection

`outputTopConfused` *Table of confused clusters*

Description

This function returns a table of the top confused clusters (assigned clusters incorrectly mapped)

Usage

```
outputTopConfused(confusionProp, count = 10)
```

Arguments

`confusionProp`	confusion matrix (e.g., output from getConfusionMatrix).
`count`	number of top confusions to show

Value

a 3 x count matrix of the top confused pairs of clusters with the three columns corresponding to mapped cluster, assigned cluster, and fraction of cells incorrectly mapped, respectively.

`plotConfusionVsConfidence` *Confusion plot vs. confidence*

Description

Produces line plots showing the percent of correctly mapped cells above a certain confidence value (or score). This is a wrapper for plot.

Usage

```
plotConfusionVsConfidence(
  foundClusterAndScore,
  realCluster,
  RI = (31:100)/100,
  main = "% mapping (blue) / correct (orange)",
  ylab = "Percent",
  xlab = "Fraction correctly mapped to leaf",
  type = "l",
  xlim = range(RI),
  ...
)
```

Arguments

`foundClusterAndScore`	matrix where first column is found cluster and second column is confidence score (e.g., output from getTopMatch)
`realCluster`	character vector of assigned clusters
`...`	additional parameters for the plot function

`plotCorrectWithGenes` *Plot fraction correct*

Description

This function is a wrapper for plot designd for plotting the fraction correctly mapped for a given gene set. If geneN is the Nth gene, the plotted value indicates correct mapping using genes 1:N.

Usage

```
plotCorrectWithGenes(
  frac,
  genes = names(frac),
  xlab = "Number of genes in panel",
  main = "All clusters gene panel",
  ylim = c(-10, 100),
  lwd = 5,
  ylab = "Percent of nuclei correctly mapping",
  colLine = "grey",
  ...
)
```

Arguments

`frac`	a numeric vector indicating the fraction of cells correctly mapped for a given gene panel
`genes`	ordered character vector (e.g., of genes) to be plotted; default is names(frac)
`...`	additional parameters for plot.

`plotDistributions` *Plot distributions*

Description

Plot the distributions of cells across the tissue with overlaying color information. This is a wrapper function for plot

Usage

```
plotDistributions(
  datIn,
  group,
  groups = NULL,
  colors = rep("black", dim(datIn$mapDat)[2]),
  colormap = gray.colors,
  maxrow = 12,
  pch = 19,
  cex = 1.5,
  xlim = NULL,
  ylim = NULL,
  main = "",
  xlab = "",
  ylab = "",
  ...
)
```

Arguments

`datIn`	a fishScaleAndMap output list
`group`	a character vector (or factor) indicating how to split the data (e.g., cluster call) or a metadata/mappingResults column name
`groups`	a character vector of groups to show (default is levels of group)
`colors`	a character vector (or factor) indicating how to color the plots (e.g., layer or gene expression) or a metadata/mappingResults column name (default is all black)
`colormap`	function to use for the colormap for the data (default gray.colors)
`maxrow`	maximum number of plots to show in one row (default=12)
`pch, cex`	for plot. Can be single values or vectors
`xlim, ylim`	for plot, but will be calculated if not entered
`main, xlab, ylab, ...`	other parameters for plot (must be single values)

Value

Only returns if there is an error

`plotHeatmap` *Plot heatmap*

Description

Plot the heatmap of cells ordering by a specified order. This is a wrapper for heatmap.2

Usage

```
plotHeatmap(
  datIn,
  group,
  groups = NULL,
  grouplab = "Grouping",
  useScaled = FALSE,
  capValue = Inf,
  colormap = grey.colors(1000),
  pch = 19,
  xlim = NULL,
  ylim = NULL,
  Rowv = FALSE,
  Colv = FALSE,
  dendrogram = "none",
  trace = "none",
  margins = c(6, 10),
  rowsep = NULL,
  sepwidth = c(0.4, 0.4),
  key = FALSE,
  ...
)
```

Arguments

`datIn`	a fishScaleAndMap output list
`group`	a character vector (or factor) indicating how to order the heatmap (e.g., cluster call) or a metadata/mappingResults column name
`groups`	a character vector of groups to show (default is levels of group)
`grouplab`	label for the grouping in the heatmap (default is 'Grouping' or the value for group)
`useScaled`	plot the scaled (TRUE) or unscaled (FALSE; default) values
`capValue`	values above capValue will be capped at capValue (default is none)
`colormap`	set of values to use for the colormap for the data (default heat_colors)
`Rowv, Colv, dendrogram, trace, margins, rowsep, colsep, key, ...`	other parameters for heatmap.2 (some default values are different)

Value

Only returns if there is an error

`plotNodes` *Plot dendrogram*

Description

Plots a dendrogram with set not colors, shapes, sizes and labels. This is a wrapper for plot.

Usage

```
plotNodes(
  tree,
  value = rep(1, length(labels(tree))),
  cexScale = 2,
  margins = c(10, 5, 2, 2),
  cols = "black",
  pch = 19,
  ...
)
```

Arguments

`tree`	a dendrogram object
`value`	numeric vector corresponding to the size of each node
`cexScale`	a global cex multiplier for node sizes
`margins`	set the margins using par(mar=margins)
`cols`	vector of node colors (or a single value)
`pch`	vector of node pch shapes (or a single value)
`...`	additional parameters for the plot function

`plotTsne` *Plot TSNE*

Description

Plot a TSNE of the data, with assigned colors and labels from provided variables. Note that this function is a modification of code from Pabloc (https://www.r-bloggers.com/author/pabloc/) from https://www.r-bloggers.com/playing-with-dimensions-from-clustering-pca-t-sne-to-carl-sagan/

Usage

```
plotTsne(
  datIn,
  colorGroup = "none",
  labelGroup = "none",
  useScaled = FALSE,
  capValue = Inf,
  perplexity = 10,
  theta = 0.5,
  main = "TSNE plot",
  maxNchar = 1000,
  seed = 10
)
```

Arguments

`datIn`	a fishScaleAndMap output list
`colorGroup`	a character vector (or factor) indicating how to color the Tsne (e.g., cluster call) or a metadata/mappingResults column name (default=NULL)
`labelGroup`	a character vector (or factor) indicating how to label the Tsne (e.g., cluster call) or a metadata/mappingResults column name (default=NULL)
`useScaled`	plot the scaled (TRUE) or unscaled (FALSE; default) values
`capValue`	values above capValue will be capped at capValue (default is none)
`perplexity, theta`	other parameters for Rtsne
`main`	title of the plot
`maxNchar`	what is the maximum number of characters to display in the plot for each entry?
`seed`	for reproducibility

Value

Only returns if there is an error

possibleClustersByPriors

Filter possible cluster calls using priors

Description

This function will return a vector of possible clusters for cells that meet a set of priors for each layer

Usage

```
possibleClustersByPriors(
  cluster,
  layer,
  subsetVector = rep(TRUE, length(cluster)),
  useClusters = sort(unique(cluster)),
  rareLimit = 0.005,
  layerNm = c("L1", "L2/3", "L4", "L5", "L6"),
  scaleByLayer = TRUE,
  scaleByFn = max,
  smartWeight = TRUE,
  spillFactor = 0.15,
  weightCutoff = 0.02
)
```

Arguments

`cluster`	vector of all clusters
`layer`	list of layers for each cluster entry (for data sets with only laminar dissections, each list entry will be of length 1)
`subsetVector`	a vector of TRUE/FALSE values indicated whether the entry is in the subset of interest (e.g., Cre lines); default is all
`useClusters`	a set of clusters to be considered a priori (e.g., GABA vs. glut); default is all
`rareLimit`	define any values less than this as 0. The idea is to exclude rare cells
`layerNm`	names of all layers. set to NULL to have this calculated
`scaleByLayer`	if TRUE, scales to the proportion of cells in each layer
`scaleByFn`	what function should be used for the layer scaling (default=max, ignored if scaleByLayer=FALSE)
`smartWeight`	if TRUE, multilayer dissections are weighted smartly by cluster, rather than evenly by cluster (FALSE)
`spillFactor`	fractional amount of cells in a layer below which it is assumed no cells are from that layer in multilayer dissection
`weightCutoff`	anything less than this is set to 0 for convenience

Value

a vector of possible clusters for cells that meet a set of priors for each layer

`quantileTruncate` *Quantile normalize, truncate, and scale*

Description

Quantile normalize, truncate, and scale a numeric vector (e.g. mFISH data from one gene)

Usage

```
quantileTruncate(x, qprob = 0.9, maxVal = 1, truncate = TRUE, ...)
```

Arguments

`x`	input data vector
`qprob`	probs value to result from quantile (default=0.9)
`maxVal`	max value for scaling (default=1)
`truncate`	should data above the qprob threshold be truncated (default=yes)
`...`	not used

Value

scaled vector

`resolve_cl` *Tree-based mapping (internal)*

Description

Returns the mapped cluster call of each cell to each leaf. This function is called by map_dend

Usage

```
resolve_cl(
  cl.g,
  cl.med,
  markers,
  dat,
  map.dat,
  select.cells,
  p = 0.7,
  low.th = 0.2
)
```

Arguments

`cl.g`	all clusters
`cl.med`	cluster medians
`markers`	gene markers
`dat`	normalized data of the REFERENCE data set
`map.dat`	normalized data of the MAPPING data set. Default is to map the data onto itself.
`select.cells`	which cells to use?
`p`	proportion of marker genes to include in each iteration of the mapping algorithm.
`low.th`	the minimum difference in Pearson correlation required to decide on which branch to map to. otherwise, a random branch is chosen.

Value

a vector of the mapped cluster

`rfTreeMapping` *Tree-based mapping*

Description

Returns the mapping membership of each cell to each node and leaf using a tree-based method. This is a wrapper function for map_dend.

Usage

```
rfTreeMapping(
  dend,
  refDat,
  clustersF,
  mapDat = refDat,
  p = 0.7,
  low.th = 0.15,
  seed = 1
)
```

Arguments

`dend`	dendrogram for mapping
`refDat`	normalized data of the REFERENCE data set
`clustersF`	factor indicating which cluster each cell type is actually assigned to in the reference data set
`mapDat`	normalized data of the MAPPING data set. Default is to map the data onto itself.
`p`	proportion of marker genes to include in each iteration of the mapping algorithm.
`low.th`	the minimum difference in Pearson correlation required to decide on which branch to map to. otherwise, a random branch is chosen.
`seed`	added for reproducibility

Value

a matrix of confidence scores (from 0 to 100) with rows as cells and columns as tree node/leafs. Values indicate the fraction of permutations in which the cell mapped to that node/leaf using the subset of cells/genes in map_dend

---

`rotateXY` *Rotate coordinates*

Description

Rotates the scaledX and scaledY elements of a fishScaleAndMap output list so that the axis of interest (e.g., cortical layer) is paralled with the x cooridate plan. Rotation code is from https://stackoverflow.com/questions/15 graph-by-angle

Usage

```
rotateXY(datFish, flatVector = NULL, flipVector = NULL, subset = NULL)
```

Arguments

`datFish`	a fishScaleAndMap output list
`flatVector`	a TRUE/FALSE vector ordred in the same way as the elements (e.g., cells) in datIn where all TRUE values correspond to cells who should have the same Y coordinate (e.g., be in the same layer). Alternatively a numeric vector of cell indices to include
`flipVector`	a numeric vector of values to ensure proper reflection on Y-axes (e.g., layer; default=NULL)
`subset`	a boolean or numeric vector of the elements to retain

Value

a fishScaleAndMap output list with updated scaledX and scaleY coordinates

---

`smartLayerAllocation` *Layer weights per cell*

Description

Returns a numeric vector saying how to weight a particular cell for each layer, using a smart weighting strategy

Usage

```
smartLayerAllocation(
  layerIn,
  useLayer = "L1",
  spillFactor = 0.15,
  weightCutoff = 0.02,
  layerNm = c("L1", "L2/3", "L4", "L5", "L6")
)
```

Arguments

`layerIn`	a list corresponding to all layers of dissection for a given sample
`useLayer`	target layer
`spillFactor`	fractional amount of cells in a layer below which it is assumed no cells are from that layer in multilayer dissection
`weightCutoff`	anything less than this is set to 0 for convenience and to avoid rare types
`layerNm`	names of all layers. set to NULL to have this calculated

Value

numeric vector saying how to weight a particular cell for each layer, using a smart weighting strategy

---

`subsampleCells` *Subsample cells*

Description

Subsets a categorical vector to include up to a maximum number of values for each category.

Usage

```
subsampleCells(clusters, subSamp = 25, seed = 5)
```

Arguments

`clusters`	vector of cluster labels (or any category) in factor or character format
`subSamp`	maximum number of values for each category to subsample. Can be single integer for global subsampling, or a *named* vector corresponding to how many values to take from each category in clusters.
`seed`	for reproducibility

Value

returns a vector of TRUE / FALSE with a maximum of subSamp TRUE calls per category

`summarizeMatrix` *Summarize matrix*

Description

Groups columns in a matrix by a specified group vector and summarizes using a specificed function. Optionally binarizes the matrix using a specified cutoff parameter. This is a wrapper for tapply.

Usage

```
summarizeMatrix(
  mat,
  group,
  scale = "none",
  scaleQuantile = 1,
  binarize = FALSE,
  binMin = 0.5,
  summaryFunction = median,
  ...
)
```

Arguments

`mat`	matrix where the columns (e.g., samples) are going to be grouped
`group`	vector of length dim(mat)[2] corresponding to the groups
`scale`	either 'none' (default),'row', or 'column'
`scaleQuantile`	what quantile of value should be set as 1 (default=1)
`binarize`	should the data be binarized? (default=FALSE)
`binMin`	minimum ON value for the binarized matrix (ignored if binarize=FALSE)
`summaryFunction`	function (or function name) to be used for summarization
`...`	additional parameters for summaryFunction

Value

matrix of summarized values

`update_mfishtools` *Update the mfishtools library*

Description

Update the mfishtools library

Usage

```
update_mfishtools()
```

Index