

# Package ‘MetaNeighbor’

March 2, 2021

**Type** Package

**Title** Single cell replicability analysis

**Version** 1.10.0

**Description** MetaNeighbor allows users to quantify cell type replicability across datasets using neighbor voting.

**biocViews** ImmunoOncology, GeneExpression, GO, MultipleComparison, SingleCell, Transcriptomics

**License** MIT + file LICENSE

**Depends** R(>= 3.5)

**Imports** grDevices, graphics, methods, stats (>= 3.4), utils (>= 3.4), Matrix (>= 1.2), matrixStats (>= 0.54), beanplot (>= 1.2), gplots (>= 3.0.1), RColorBrewer (>= 1.1.2), SummarizedExperiment (>= 1.12), SingleCellExperiment, igraph, dplyr, tidyr, tibble, ggplot2

**Suggests** knitr (>= 1.17), rmarkdown (>= 1.6), testthat (>= 1.0.2), UpSetR

**LazyData** true

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**Maintainer** Stephan Fischer <fischer@cshl.edu>

**git\_url** <https://git.bioconductor.org/packages/MetaNeighbor>

**git\_branch** RELEASE\_3\_12

**git\_last\_commit** c4000c0

**git\_last\_commit\_date** 2020-10-27

**Date/Publication** 2021-03-01

**Author** Megan Crow [aut, cre],  
Sara Ballouz [ctb],  
Manthan Shah [ctb],  
Stephan Fischer [ctb],  
Jesse Gillis [aut]

**R topics documented:**

extendClusterSet	2
extractMetaClusters	3
getCellType	3
getStudyId	4
GOhuman	4
GOMouse	5
makeClusterGraph	5
makeClusterName	6
mergeSCE	6
MetaNeighbor	7
MetaNeighborUS	8
mn_data	10
neighborVoting	10
plotBPlot	11
plotClusterGraph	12
plotDotPlot	13
plotHeatmap	13
plotHeatmapPretrained	14
plotMetaClusters	15
plotUpset	16
scoreMetaClusters	16
splitClusters	17
splitTestClusters	17
splitTrainClusters	18
standardizeLabel	19
subsetClusterGraph	19
topHits	20
trainModel	21
variableGenes	22
<b>Index</b>	<b>23</b>

---

extendClusterSet	<i>Extend cluster set to nearest neighbors on cluster graph.</i>
------------------	--

---

**Description**

Note that the graph is directed, i.e. neighbors are retrieved by following arrows that start from the initial clusters.

**Usage**

```
extendClusterSet(graph, initial_set, max_neighbor_distance = 2)
```

**Arguments**

graph	Graph in igraph format generated by makeClusterGraph.
initial_set	Vector of cluster labels
max_neighbor_distance	Include more distantly related nodes by performing neighbor extension max_neighbor_distance rounds.

**Value**

Character vector including initial cluster set and all neighboring clusters (if any).

---

extractMetaClusters     *Extracts groups of reciprocal top hits from a 1-vs-best AUROC matrix.*

---

**Description**

Note that meta-clusters are *not* cliques, but connected components, e.g., if 1<->2 and 1<->3 are reciprocal top hits, 1, 2, 3 is a meta-cluster, independently from the relationship between 2 and 3.

**Usage**

```
extractMetaClusters(best_hits, threshold = 0)
```

**Arguments**

**best\_hits**     Matrix of AUROCs produced by MetaNeighborUS.

**threshold**     AUROC threshold. Two clusters belong to the same meta-cluster if they are reciprocal top hits and their similarity exceeds the threshold *both* ways (AUROC(1->2) > threshold *AND* AUROC(2->1) > threshold).

**Value**

A named list, where names are default meta-cluster names, and values are vectors of cluster names, one vector per meta-cluster. The last element of the list is called "outliers" and contains all clusters that had no match in any other dataset.

---

getCellType     *Return cell type from a label in format 'study\_id|cell\_type'*

---

**Description**

Return cell type from a label in format 'study\_id|cell\_type'

**Usage**

```
getCellType(cluster_name)
```

**Arguments**

**cluster\_name**     Character vector containing cluster names in the format study\_id|cell\_type.

**Value**

Character vector containing all cell type names.

getStudyId                      *Return study ID from a label in format 'study\_id|cell\_type'*

---

**Description**

Return study ID from a label in format 'study\_id|cell\_type'

**Usage**

```
getStudyId(cluster_name)
```

**Arguments**

cluster\_name      Character vector containing cluster names in the format study\_id|cell\_type.

**Value**

Character vector containing all study ids.

---

GOhuman                      *GOhuman*

---

**Description**

List containing gene symbols for 71 GO function

**Usage**

```
GOhuman
```

**Format**

**genesets** List containing gene symbols for 71 GO function (GO slim terms containing between 50 and 1,000 genes) downloaded from the Gene Ontology Consortium August 2015 <http://www.geneontology.org/page/download-annotations>

**Source**

Dataset: <https://github.com/mm-shah/MetaNeighbor/tree/master/data> | Paper: <https://www.biorxiv.org/content/early/2017/06/16/150524>

GOMouse

*GOMouse***Description**

List containing gene symbols for 10 GO function

**Usage**

GOMouse

**Format**

**genesets** List containing gene symbols for 10 GO function (GO:0016853 , GO:0005615, GO:0005768, GO:0007067, GO:0065003, GO:0042592, GO:0005929, GO:0008565, GO:0016829, GO:0022857) downloaded from the Gene Ontology Consortium August 2015 <http://www.geneontology.org/page/download-annotations>

**Source**

Dataset: <https://github.com/mm-shah/MetaNeighbor/tree/master/data> | Paper: <https://www.biorxiv.org/content/early/2017/06/16/150524>

makeClusterGraph

*Convert AUROC matrix into a graph.***Description**

This representation is a useful alternative for heatmaps for large datasets and sparse AUROC matrices (MetaNeighborUS with one\_vs\_best = TRUE)

**Usage**

```
makeClusterGraph(best_hits, low_threshold = 0, high_threshold = 1)
```

**Arguments**

**best\_hits** Matrix of AUROCs produced by MetaNeighborUS.

**low\_threshold** AUROC threshold value. An edge is drawn between two clusters only if their similarity exceeds low\_threshold.

**high\_threshold** AUROC threshold value. An edge is drawn between two clusters only if their similarity is lower than high\_threshold (enables focusing on close calls).

**Value**

A graph in igraph format, where nodes are clusters and edges are AUROC similarities.

---

makeClusterName	<i>Make cluster names in format 'study_id cell_type'</i>
-----------------	--

---

**Description**

Make cluster names in format 'study\_id|cell\_type'

**Usage**

```
makeClusterName(study_id, cell_type)
```

**Arguments**

study_id	Character vector containing study ids.
cell_type	Character vector containing cell type names

**Value**

Character vector containing cluster names in the format study\_id|cell\_type.

---

mergeSCE	<i>Merge multiple SingleCellExperiment objects.</i>
----------	---

---

**Description**

Merge multiple SingleCellExperiment objects.

**Usage**

```
mergeSCE(sce_list)
```

**Arguments**

sce_list	A <i>*named*</i> list, where values are SingleCellExperiment objects and names are SingleCellExperiment objects.
----------	--

**Value**

A SingleCellExperiment object containing the input datasets with the following limitations: (i) only genes common to all datasets are kept, (ii) only colData columns common to all datasets are kept, (iii) only assays common to all datasets (i.e., having the same name) are kept, (iv) all other slots (e.g., reducedDims or rowData) will be ignored and left empty. The SingleCellExperiment object contains a "study\_id" column, mapping each cell to its original dataset (names in "sce\_list").

---

MetaNeighbor	<i>Runs MetaNeighbor</i>
--------------	--------------------------

---

### Description

For each gene set of interest, the function builds a network of rank correlations between all cells. Next, it builds a network of rank correlations between all cells for a gene set. Next, the neighbor voting predictor produces a weighted matrix of predicted labels by performing matrix multiplication between the network and the binary vector indicating cell type membership, then dividing each element by the null predictor (i.e., node degree). That is, each cell is given a score equal to the fraction of its neighbors (including itself), which are part of a given cell type. For cross-validation, we permute through all possible combinations of leave-one-dataset-out cross-validation, and we report how well we can recover cells of the same type as area under the receiver operator characteristic curve (AUROC). This is repeated for all folds of cross-validation, and the mean AUROC across folds is reported. Calls [neighborVoting](#).

### Usage

```
MetaNeighbor(
  dat,
  i = 1,
  experiment_labels,
  celltype_labels,
  genesets,
  bplot = TRUE,
  fast_version = FALSE,
  node_degree_normalization = TRUE,
  batch_size = 10,
  detailed_results = FALSE
)
```

### Arguments

<code>dat</code>	A SummarizedExperiment object containing gene-by-sample expression matrix.
<code>i</code>	default value 1; non-zero index value of assay containing the matrix data
<code>experiment_labels</code>	A vector that indicates the source/dataset of each sample.
<code>celltype_labels</code>	A character vector or one-hot encoded matrix (cells x cell type) that indicates the cell type of each sample.
<code>genesets</code>	Gene sets of interest provided as a list of vectors.
<code>bplot</code>	default true, beanplot is generated
<code>fast_version</code>	default value FALSE; a boolean flag indicating whether to use the fast and low memory version of MetaNeighbor
<code>node_degree_normalization</code>	default value TRUE; a boolean flag indicating whether to normalize votes by dividing through total node degree.
<code>batch_size</code>	Optimization parameter. Gene sets are processed in groups of size <code>batch_size</code> . The count matrix is first subset to all genes from these groups, then to each gene set individually.

detailed\_results

Should the function return the average AUROC across all test datasets (default) or a detailed table with the AUROC for each test dataset?

### Value

A matrix of AUROC scores representing the mean for each gene set tested for each celltype is returned directly (see [neighborVoting](#)). If detailed\_results is set to TRUE, the function returns a table of AUROC scores in each test dataset for each gene set.

### See Also

[neighborVoting](#)

### Examples

```
data("mn_data")
data("G0mouse")
library(SummarizedExperiment)
AUROC_scores = MetaNeighbor(dat = mn_data,
                           experiment_labels = as.numeric(factor(mn_data$study_id)),
                           celltype_labels = metadata(colData(mn_data))["cell_labels"],
                           genesets = G0mouse,
                           bplot = TRUE)
```

---

MetaNeighborUS

*Runs unsupervised version of MetaNeighbor*

---

### Description

When it is difficult to know how cell type labels compare across datasets this function helps users to make an educated guess about the overlaps without requiring in-depth knowledge of marker genes

### Usage

```
MetaNeighborUS(
  var_genes = c(),
  dat,
  i = 1,
  study_id,
  cell_type,
  trained_model = NULL,
  fast_version = FALSE,
  node_degree_normalization = TRUE,
  one_vs_best = FALSE,
  symmetric_output = TRUE
)
```



**Arguments**

<code>var_genes</code>	vector of high variance genes.
<code>dat</code>	SummarizedExperiment object containing gene-by-sample expression matrix.
<code>i</code>	default value 1; non-zero index value of assay containing the matrix data
<code>study_id</code>	a vector that lists the Study (dataset) ID for each sample
<code>cell_type</code>	a vector that lists the cell type of each sample
<code>trained_model</code>	default value NULL; a matrix containing a trained model generated from <code>MetaNeighbor::trainModel</code> . If not NULL, the trained model is treated as training data and <code>dat</code> is treated as testing data. If a trained model is provided, <code>fast_version</code> will automatically be set to TRUE and <code>var_genes</code> will be overridden with genes used to generate the <code>trained_model</code>
<code>fast_version</code>	default value FALSE; a boolean flag indicating whether to use the fast and low memory version of <code>MetaNeighbor</code>
<code>node_degree_normalization</code>	default value TRUE; a boolean flag indicating whether to use normalize votes by dividing through total node degree.
<code>one_vs_best</code>	default value FALSE; a boolean flag indicating whether to compute AUROCs based on a best match against second best match setting (default version is one-vs-rest). This option is currently only relevant when <code>fast_version = TRUE</code> .
<code>symmetric_output</code>	default value TRUE; a boolean flag indicating whether to average AUROCs in the output matrix.

**Value**

The output is a cell type-by-cell type mean AUROC matrix, which is built by treating each pair of cell types as testing and training data for `MetaNeighbor`, then taking the average AUROC for each pair (NB scores will not be identical because each test cell type is scored out of its own dataset, and the differential heterogeneity of datasets will influence scores). If `symmetric_output` is set to FALSE, the training cell types are displayed as columns and the test cell types are displayed as rows. If `trained_model` was provided, the output will be a cell type-by-cell type AUROC matrix with training cell types as columns and test cell types as rows (no swapping of test and train, no averaging).

**Examples**

```
data(mn_data)
var_genes = variableGenes(dat = mn_data, exp_labels = mn_data$study_id)
celltype_NV = MetaNeighborUS(var_genes = var_genes,
                             dat = mn_data,
                             study_id = mn_data$study_id,
                             cell_type = mn_data$cell_type)
celltype_NV
```

---

mn_data	<i>mn_data</i>
---------	----------------

---

### Description

A SummarizedExperiment object containing: a gene matrix, cell type labels, experiment labels, sets of genes, sample ID, study id and cell types.

### Usage

```
mn_data
```

### Format

**Gene matrix** A gene-by-sample expression matrix consisting of 3157 rows (genes) and 1051 columns (cell types)

**cell\_labels** 1051x1 binary matrix that indicates whether a cell belongs to the SstNos cell type (1=yes, 0 = no)

**sample\_id** A character vector of length 1051 that indicates the sample\_id of each sample

**study\_id** A character vector of length 1051 that indicates the study\_id of each sample ("GSE60361" = Zeisel et al, "GSE71585" = Tasic et al)

**cell\_type** A character vector of length 1051 that indicates the cell-type of each sample

### Source

Dataset:<https://github.com/mm-shah/MetaNeighbor/tree/master/data> 1. Zeisel et al. <http://science.sciencemag.org/content/347/6226/1138> 2. Tasic et al. <http://www.nature.com/neuro/journal/v19/n2/full/nn.4216.html>

---

neighborVoting	<i>Runs the neighbor voting algorithm.</i>
----------------	--

---

### Description

The function performs cell type identity prediction based on 'guilt by association' using cross validation. Performance is evaluated by calculating the AUROC for each cell type.

### Usage

```
neighborVoting(
  exp_labels,
  cell_labels,
  network,
  means = TRUE,
  node_degree_normalization = TRUE
)
```

**Arguments**

exp_labels	A vector that indicates the dataset source of each sample
cell_labels	sample by cell type matrix that indicates the cell type of each sample (0-absent; 1-present)
network	sample by sample adjacency matrix, ranked and standardized between 0-1
means	default TRUE, determines output formatting
node_degree_normalization	default TRUE, should predictions be divided by node degree?

**Value**

If means = TRUE (default) a vector containing the mean of AUROC values across cross-validation folds will be returned. If FALSE a list is returned containing a cell type by dataset matrix of AUROC scores, for each fold of cross-validation. Default is over-ridden when more than one cell type is assessed.

**See Also**

[MetaNeighbor](#)

**Examples**

```
data("mn_data")
data("G0mouse")
library(SummarizedExperiment)
AUROC_scores = MetaNeighbor(dat = mn_data,
                           experiment_labels = as.numeric(factor(mn_data$study_id)),
                           celltype_labels = metadata(colData(mn_data))["cell_labels"],
                           genesets = G0mouse,
                           bplot = TRUE)

AUROC_scores
```

---

plotBPlot	<i>Plot Bean Plot, showing how replicability of cell types depends on gene sets.</i>
-----------	--

---

**Description**

Plot Bean Plot, showing how replicability of cell types depends on gene sets.

**Usage**

```
plotBPlot(nv_mat, hvg_score = NULL, cex = 1)
```

**Arguments**

nv_mat	A rectangular AUROC matrix as returned by MetaNeighbor, where each row is a gene set and each column is a cell type.
hvg_score	Named vector with AUROCs obtained from a set of Highly Variable Genes (HVGs). The names must correspond to cell types from nv_mat. If specified, the HVG score is highlighted in red.
cex	Size factor for row and column labels.

**Examples**

```

data("mn_data")
data("G0mouse")
library(SummarizedExperiment)
AUROC_scores = MetaNeighbor(dat = mn_data,
                             experiment_labels = as.numeric(factor(mn_data$study_id)),
                             celltype_labels = metadata(colData(mn_data))["cell_labels"],
                             genesets = G0mouse,
                             bplot = FALSE)

plotBPlot(AUROC_scores)

```

---

plotClusterGraph      *Plot cluster graph generated with makeClusterGraph.*

---

**Description**

In this visualization, edges are colored in black when AUROC > 0.5 and orange when AUROC < 0.5, edge width scales linearly with AUROC. Edges are oriented from training cluster towards test cluster. A black bidirectional edge indicates that two clusters are reciprocal top matches. Node radius reflects cluster size (small: up to 10 cells, medium: up to 100 cells, large: all other clusters).

**Usage**

```

plotClusterGraph(
  graph,
  study_id = NULL,
  cell_type = NULL,
  size_factor = 1,
  label_cex = 0.2 * size_factor,
  legend_cex = 2,
  study_cols = NULL
)

```

**Arguments**

graph	Graph in igraph format generated by makeClusterGraph.
study_id	Vector with study IDs provided to MetaNeighborUS to compute AUROCs stored in graph (used to compute cluster size). If NULL, all nodes have medium size.
cell_type	Vector with cell type labels provided to MetaNeighborUS to compute AUROCs stored in graph (used to compute cluster size). If NULL, all nodes have medium size.
size_factor	Numeric value controlling the size of nodes and edges.
label_cex	Numeric value controlling the size of cell type labels.
legend_cex	Numeric value controlling the size of the legend.
study_cols	Named vector where values are RGB colors and names are unique study identifiers. If NULL, a default color palette is used.

---

plotDotPlot	<i>Plot dot plot showing expression of a gene set across cell types.</i>
-------------	--

---

### Description

The size of each dot reflects the number of cell that express a gene, the color reflects the average expression. Expression of genes is first average and scaled in each dataset independently. The final value is obtained by averaging across datasets.

### Usage

```
plotDotPlot(
  dat,
  experiment_labels,
  celltype_labels,
  gene_set,
  i = 1,
  normalize_library_size = TRUE,
  alpha_row = 10,
  average_expressing_only = TRUE
)
```

### Arguments

dat	A SummarizedExperiment object containing gene-by-sample expression matrix.
experiment_labels	A vector that indicates the source/dataset of each sample.
celltype_labels	A character vector that indicates the cell type of each sample.
gene_set	Gene set of interest provided as a vector of genes.
i	Default value 1; non-zero index value of assay containing the matrix data.
normalize_library_size	Whether to apply library size normalization before computing average expression (set this value to FALSE if data are already normalized).
alpha_row	Parameter controlling row ordering: a higher value of alpha_row gives more weight to extreme AUROC values (close to 1).
average_expressing_only	Whether average expression should be computed based only on expressing cells (Seurat default) or taking into account zeros.

---

plotHeatmap	<i>Plots symmetric AUROC heatmap, clustering cell types by similarity.</i>
-------------	--

---

### Description

Plots symmetric AUROC heatmap, clustering cell types by similarity.

**Usage**

```
plotHeatmap(aurocs, cex = 1, margins = c(8, 8))
```

**Arguments**

aurocs	A square AUROC matrix as returned by MetaNeighborUS.
cex	Size factor for row and column labels.
margins	Size of margins (for row and column labels).

**Examples**

```
data(mn_data)
var_genes = variableGenes(dat = mn_data, exp_labels = mn_data$study_id)
celltype_NV = MetaNeighborUS(var_genes = var_genes,
                             dat = mn_data,
                             study_id = mn_data$study_id,
                             cell_type = mn_data$cell_type)

plotHeatmap(celltype_NV)
```

---

`plotHeatmapPretrained` *Plots rectangular AUROC heatmap, clustering train cell types (columns) by similarity, and ordering test cell types (rows) according to similarity to train cell types..*

---

**Description**

Plots rectangular AUROC heatmap, clustering train cell types (columns) by similarity, and ordering test cell types (rows) according to similarity to train cell types..

**Usage**

```
plotHeatmapPretrained(
  aurocs,
  alpha_col = 1,
  alpha_row = 10,
  cex = 1,
  margins = c(8, 8)
)
```

**Arguments**

aurocs	A rectangular AUROC matrix as returned by MetaNeighborUS,
alpha_col	Parameter controlling column clustering: a higher value of alpha_col gives more weight to extreme AUROC values (close to 1).
alpha_row	Parameter controlling row ordering: a higher value of alpha_row gives more weight to extreme AUROC values (close to 1).
cex	Size factor for row and column labels.
margins	Size of margins (for row and column labels).

**Examples**

```

data(mn_data)
var_genes = variableGenes(dat = mn_data, exp_labels = mn_data$study_id)
celltype_NV = MetaNeighborUS(var_genes = var_genes,
                             dat = mn_data,
                             study_id = mn_data$study_id,
                             cell_type = mn_data$cell_type,
                             symmetric_output = FALSE)
keep_col = getStudyId(colnames(celltype_NV)) == "GSE71585"
keep_row = getStudyId(rownames(celltype_NV)) != "GSE71585"
plotHeatmapPretrained(celltype_NV[keep_row, keep_col])

```

---

plotMetaClusters	<i>Plot meta-cluster badges, each badge is a small AUROC heatmap restricted to a specific meta-cluster.</i>
------------------	---

---

**Description**

Plot meta-cluster badges, each badge is a small AUROC heatmap restricted to a specific meta-cluster.

**Usage**

```

plotMetaClusters(
  meta_clusters,
  best_hits,
  reorder = FALSE,
  cex = 1,
  study_cols = NULL,
  auroc_breaks = c(0, 0.5, 0.7, 0.9, 0.95, 0.99, 1),
  auroc_cols = (grDevices::colorRampPalette(c("white", "blue")))(length(auroc_breaks) -
    1)
)

```

**Arguments**

meta_clusters	Meta-cluster list generated by extractMetaClusters.
best_hits	Matrix of AUROCs used to extract meta-clusters.
reorder	Reorder datasets by similarity for each badge? By default, the same dataset ordering is used for each badge.
cex	Size factor controlling label size.
study_cols	Named vector where values are RGB colors and names are unique study identifiers (corresponding to study_id). If NULL, a default color palette is used.
auroc_breaks	Numeric vector used to bin AUROC values for color coding.
auroc_cols	Vector containing RGB colors used to encode AUROC levels. The length of auroc_cols must correspond to the length of auroc_breaks - 1.

---

plotUpset	<i>Plot Upset plot showing how replicability depends on input dataset.</i>
-----------	--

---

### Description

Plot Upset plot showing how replicability depends on input dataset.

### Usage

```
plotUpset(metaclusters, min_recurrence = 2, outlier_name = "outliers")
```

### Arguments

metaclusters	Metaclusters extracted from MetaNeighborUS analysis.
min_recurrence	Only show replicability structure for metaclusters that are replicable across at least min_recurrence datasets.
outlier_name	In metaclusters, name assigned to outliers (clusters that did not match with any other cluster)

### Examples

```
data(mn_data)
var_genes = variableGenes(dat = mn_data, exp_labels = mn_data$study_id)
celltype_NV = MetaNeighborUS(var_genes = var_genes,
                             dat = mn_data,
                             study_id = mn_data$study_id,
                             cell_type = mn_data$cell_type,
                             fast_version = TRUE, one_vs_best = TRUE)
mclusters = extractMetaClusters(celltype_NV)
plotUpset(mclusters)
```

---

scoreMetaClusters	<i>Summarize meta-cluster information in a table.</i>
-------------------	---

---

### Description

Summarize meta-cluster information in a table.

### Usage

```
scoreMetaClusters(meta_clusters, best_hits, outlier_label = "outliers")
```

### Arguments

meta_clusters	Meta-cluster list generated by extractMetaClusters.
best_hits	Matrix of AUROCs used to extract meta-clusters.
outlier_label	Element of meta-cluster list containing outlier clusters.



**Value**

A data.frame. Column "meta\_cluster" contains meta-cluster names, "clusters" lists the clusters belonging to each meta-cluster, "n\_studies" is the number of studies spanned by the meta-cluster, "score" is the average similarity between meta-cluster members (average AUROC, NAs are treated as 0).

---

splitClusters	<i>Split clusters according to symmetric AUROC similarity.</i>
---------------	--

---

**Description**

This function computes hierarchical clustering to group similar clusters, interpreting the AUROC matrix as a similarity matrix, then uses a standard tree cutting algorithm to obtain groups of similar clusters. Note that the cluster hierarchy corresponds exactly to the dendrogram shown when using the plotHeatmap function.

**Usage**

```
splitClusters(mn_scores, k)
```

**Arguments**

mn_scores	A symmetric AUROC matrix as generated by MetaNeighborUS.
k	The number of desired cluster sets.

**Value**

A list of cluster sets, each cluster set is a character vector containing cluster labels.

**See Also**

[plotHeatmap](#)

---

splitTestClusters	<i>Split test clusters according to AUROC similarity to train clusters.</i>
-------------------	---

---

**Description**

This function computes hierarchical clustering to group similar test clusters, using similarity to train clusters as features, then uses a standard tree cutting algorithm to obtain groups of similar clusters. Note that the cluster hierarchy does *not* correspond to the row ordering of plotHeatmapPretrained function, which uses a different heuristic.

**Usage**

```
splitTestClusters(mn_scores, k)
```

**Arguments**

mn_scores	An AUROC matrix as generated by MetaNeighborUS, usually with the "trained_model" option.
k	The number of desired cluster sets.

**Value**

A list of cluster sets, each cluster set is a character vector containing cluster labels.

**See Also**

[plotHeatmapPretrained](#)

---

splitTrainClusters	<i>Split train clusters according to AUROC similarity to test clusters.</i>
--------------------	---

---

**Description**

This function computes hierarchical clustering to group similar train clusters, using similarity to test clusters as features, then uses a standard tree cutting algorithm to obtain groups of similar clusters. Note that the cluster hierarchy corresponds exactly to the column dendrogram shown when using the plotHeatmapPretrained function.

**Usage**

```
splitTrainClusters(mn_scores, k)
```

**Arguments**

mn_scores	An AUROC matrix as generated by MetaNeighborUS, usually with the "trained_model" option.
k	The number of desired cluster sets.

**Value**

A list of cluster sets, each cluster set is a character vector containing cluster labels.

**See Also**

[plotHeatmapPretrained](#)

---

standardizeLabel	<i>Remove special characters (" ") from labels to avoid later conflicts</i>
------------------	---

---

**Description**

Remove special characters ("|") from labels to avoid later conflicts

**Usage**

```
standardizeLabel(labels, replace = "|", with = ".")
```

**Arguments**

labels	Character vector containing study ids or cell type names.
replace	Special character to replace
with	Character to use instead of special character

**Value**

Character vector with replaced special characters.

---

subsetClusterGraph	<i>Subset cluster graph to clusters of interest.</i>
--------------------	--

---

**Description**

Subset cluster graph to clusters of interest.

**Usage**

```
subsetClusterGraph(graph, vertices)
```

**Arguments**

graph	Graph in igraph format generated by makeClusterGraph.
vertices	Vector of cluster labels

**Value**

Graph in igraph format, where nodes have been restricted to clusters of interests.

**See Also**

[extendClusterSet](#)

---

topHits	<i>Find reciprocal top hits</i>
---------	---------------------------------

---

### Description

Identifies reciprocal top hits and high scoring cell type pairs

### Usage

```
topHits(cell_NV, dat, i = 1, study_id, cell_type, threshold = 0.95)
```

### Arguments

cell_NV	matrix of celltype-to-celltype AUROC scores (output from <a href="#">MetaNeighborUS</a> )
dat	a SummarizedExperiment object containing gene-by-sample expression matrix.
i	default value 1; non-zero index value of assay containing the matrix data
study_id	a vector that lists the Study (dataset) ID for each sample
cell_type	a vector that lists the cell type of each sample
threshold	default value 0.95. Must be between [0,1]

### Value

Function returns a dataframe with cell types that are either reciprocal best matches, and/or those with AUROC values greater than or equal to threshold value

### Examples

```
data(mn_data)
var_genes = variableGenes(dat = mn_data, exp_labels = mn_data$study_id)
celltype_NV = MetaNeighborUS(var_genes = var_genes,
                             dat = mn_data,
                             study_id = mn_data$study_id,
                             cell_type = mn_data$cell_type)
top_hits = topHits(cell_NV = celltype_NV,
                  dat = mn_data,
                  study_id = mn_data$study_id,
                  cell_type = mn_data$cell_type,
                  threshold = 0.9)

top_hits
```

---

trainModel	<i>Pretrains model for the unsupervised version of MetaNeighbor</i>
------------	---

---

### Description

When comparing clusters to a large reference dataset, this function summarizes the gene-by-cell matrix into a much smaller highly variable gene-by-cluster matrix which can be fed as training data into MetaNeighborUS, resulting in substantial time and memory savings.

### Usage

```
trainModel(var_genes, dat, i = 1, study_id, cell_type)
```

### Arguments

var_genes	vector of high variance genes.
dat	SummarizedExperiment object containing gene-by-sample expression matrix.
i	default value 1; non-zero index value of assay containing the matrix data
study_id	a vector that lists the Study (dataset) ID for each sample
cell_type	a vector that lists the cell type of each sample

### Value

The output is a gene-by-cluster matrix that contains all the information necessary to run MetaNeighborUS from a pre-trained model.

### Examples

```
data(mn_data)
var_genes = variableGenes(dat = mn_data, exp_labels = mn_data$study_id)
trained_model = trainModel(var_genes = var_genes,
                           dat = mn_data,
                           study_id = mn_data$study_id,
                           cell_type = mn_data$cell_type)
celltype_NV = MetaNeighborUS(trained_model = trained_model,
                             dat = mn_data,
                             study_id = mn_data$study_id,
                             cell_type = mn_data$cell_type)

celltype_NV
```

---

variableGenes	<i>Identify a highly variable gene set</i>
---------------	--

---

**Description**

Identifies genes with high variance compared to their median expression (top quartile) within each experiment. Certain function

**Usage**

```
variableGenes(
  dat,
  i = 1,
  exp_labels,
  min_recurrence = length(unique(exp_labels)),
  downsampling_size = 10000
)
```

**Arguments**

dat	SummarizedExperiment object containing gene-by-sample expression matrix.
i	default value 1; non-zero index value of assay containing the matrix data
exp_labels	character vector that denotes the source (Study ID) of each sample.
min_recurrence	Number of studies across which a gene must be detected as highly variable to be kept. By default, only genes that are variable across all studies are kept (intersection).
downsampling_size	Downsample each study to downsampling_size samples without replacement. If set to 0 or value exceeds dataset size, no downsampling is applied.

**Value**

The output is a vector of gene names that are highly variable in every experiment (intersect)

**Examples**

```
data(mn_data)
var_genes = variableGenes(dat = mn_data, exp_labels = mn_data$study_id)
var_genes
```

# Index

## \* datasets

- GOhuman, 4
- GOMouse, 5
- mn\_data, 10

extendClusterSet, 2, 19  
extractMetaClusters, 3

getCellType, 3  
getStudyId, 4  
GOhuman, 4  
GOMouse, 5

makeClusterGraph, 5  
makeClusterName, 6  
mergeSCE, 6  
MetaNeighbor, 7, 11  
MetaNeighborUS, 8, 20  
mn\_data, 10

neighborVoting, 7, 8, 10

plotBPlot, 11  
plotClusterGraph, 12  
plotDotPlot, 13  
plotHeatmap, 13, 17  
plotHeatmapPretrained, 14, 18  
plotMetaClusters, 15  
plotUpset, 16

scoreMetaClusters, 16  
splitClusters, 17  
splitTestClusters, 17  
splitTrainClusters, 18  
standardizeLabel, 19  
subsetClusterGraph, 19

topHits, 20  
trainModel, 21

variableGenes, 22