Package ‘robustSingleCell’

April 24, 2019

Type Package
Title Robust Clustering of Single Cell RNA-Seq Data
Version 0.1.1
Description Robust single cell clustering and comparison of population compositions across tissues and experimental models via similarity analysis from Magen 2019 <doi:10.1101/543199>.
Depends R (>= 3.2.0)
Imports utils, grDevices, graphics, Matrix, limma, biomaRt, dplyr, ggplot2, reshape2, GGally, ggrepel, RColorBrewer, gplots, ggpubr, cccd, rslurm, Rtsne, igraph, scales, RANN, Rcpp
LinkingTo Rcpp
License Artistic-2.0
URL https://github.com/asmagen/robustSingleCell
BugReports https://github.com/asmagen/robustSingleCell/issues
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
Suggests GEOquery, knitr, rmarkdown, testthat
VignetteBuilder knitr
NeedsCompilation yes
Author Assaf Magen [aut, cph] (<https://orcid.org/0000-0001-5473-274X>), Meng Wang [aut, cre] (<https://orcid.org/0000-0002-3453-7805>), Hao Chen [ctb]
Maintainer Meng Wang <szmamie@live.com>
Repository CRAN
Date/Publication 2019-04-23 22:00:02 UTC
Add Confounder Variables

Add confounder variables’ activation level per cell to environment object.

Usage

add.confounder.variables(environment, ...)

Index

add.confounder.variables

Add Confounder Variables

Description

Add confounder variables’ activation level per cell to environment object.

Usage

add.confounder.variables(environment, ...)
assess.cluster.similarity

Assess Cluster Similarity

Description
Assess similarity between pairs of clusters.

Usage
assess.cluster.similarity(environment, diff.exp.file = "main.datasets.diff.exp.rds", cluster.similarity.function = pearson.correlation, label = "pearson", rerun = F)

Arguments
environment environment object
diff.exp.file name of differential expression results file
cluster.similarity.function which similarity function to use (either 'pearson.correlation' or '?') Mamie - there was another similarity function using euclidean distance. Do you know where did it go to? Can you replace the '?' with the name of this other function?
label name of the similarity measure to use for the results folder
rerun whether to rerun the analysis or load from cache

Value
pairwise cluster similarity measures
Examples

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0,
              min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
LCMV1 <- cluster.analysis(LCMV1)
types = rbind(
    data.frame(type='Tfh', gene=c('Tcf7', 'Cxcr5', 'Bcl6'))),
    data.frame(type='Th1', gene=c('Cxcr6', 'Ifng', 'Tbx2'))),
    data.frame(type='Tcmp', gene=c('Ccr7', 'Bcl2', 'Tcf7'))),
    data.frame(type='Treg', gene=c('Foxp3', 'Il2ra'))),
    data.frame(type='Tmem', gene=c('Il7r', 'Ccr7'))),
    data.frame(type='CD8', gene=c('Cd8a'))),
    data.frame(type='CD4', gene = c('Cd4'))),
    data.frame(type='Cycle', gene=c('Mki67', 'Top2a', 'Birc5'))) )
summarize(LCMV1)
cluster_names <- get.cluster.names(LCMV1, types, min.fold = 1.0, max.Qval = 0.01)
LCMV1 <- set.cluster.names(LCMV1, names = cluster_names)
LCMV2 <- setup_LCMV_example("LCMV2")
LCMV2 <- get.variable.genes(LCMV2, min.mean = 0.1, min.frac.cells = 0,
              min.dispersion.scaled = 0.1)
LCMV2 <- PCA(LCMV2)
LCMV2 <- cluster.analysis(LCMV2)
summarize(LCMV2)
cluster_names <- get.cluster.names(LCMV2, types, min.fold = 1.0, max.Qval = 0.01)
LCMV2 <- set.cluster.names(LCMV2, names = cluster_names)
pooled_env <- setup_pooled_env()
pooled_env <- read.preclustered.datasets(pooled_env)
pooled_env <- PCA(pooled_env, clear.previously.calculated.clustering = F)
summarize(pooled_env, contrast = "datasets")
cluster.similarity <- assess.cluster.similarity(pooled_env)
```

cell.cycle.score  

**Compute Cell Cycle Score**

Description

Compute the activation of cell cycle genes defined in Kowalczyk, M. S. et al.

Usage

cell.cycle.score(environment, knn = 10, cc.genes.path = NA)
cluster.analysis

Arguments

- **environment**: environment object
- **knn**: number of nearest neighbor

Value

a matrix of cell cycle genes activation scores (S, G2M and aggregated S/G2M scores, separately)

Examples

```r
LCMV1 <- setup_LCMV_example() cell.cycle.score <- cell.cycle.score(LCMV1)
```

Description

Perform clustering analysis for a range of hyperparameter (KNN Ratios) values and assess clustering quality relative to simulation analysis of shuffled data.

Usage

```r
cluster.analysis(environment, knn.ratios = c(0.01, 0.05, 0.1), nShuffleRuns = 10, shuffledKNN = 10, loadPreviousKnn = T, rerun = F, deleteCache = F, mem = "4GB", time = "0:15:00", plot = T, local = F)
```

Arguments

- **environment**: environment object
- **knn.ratios**: range of KNN parameters to scan (corresponding to different resolutions)
- **nShuffleRuns**: number of shuffled clustering analyses to perform per KNN threshold
- **shuffledKNN**: number of closest KNN shuffled analyses to include in background clustering quality computation
- **loadPreviousKnn**: whether to load previous analysis results
- **rerun**: whether to rerun the analysis rather than load from cache
- **deleteCache**: whether to delete cache files
controlled.mean.score

controlled.mean.score  Compute Controlled Activation Score

Description
Compute mean gene signatures activation scores while controlling for technically similar genes.

Usage
controlled.mean.score(environment, genes, knn = 10, exclude.missing.genes = T, constrain.cell.universe = NA)

Arguments
environment   environment object
genes         gene list upon which to calculate gene signature activate
knn           number of nearest neighbors
exclude.missing.genes whether to exclude genes with missing values
constrain.cell.universe   binary vector indicating in which subset of cells to calculate the gene signature activation. Default is all cells.

Value
gene signature activation scores per cell
**download_LCMV**

**Examples**

```r
LCMV1 <- setup_LCMV_example()
exhaustion_markers <- c('Pdcd1', 'Cd244', 'Havcr2', 'Ctl4a4', 'Cd160', 'Lag3', 'Tigit', 'Cd96')
Exhaustion <- controlled.mean.score(LCMV1, exhaustion_markers)
```

---

**Description**

Download two replicates of CD44+ T cell 10X scRNAseq data sets (Ciucci 2018).

**Usage**

```r
download_LCMV(base_dir = NULL)
```

**Arguments**

- `base_dir` Full path to a directory where data and analysis will be stored

**Value**

1 if download fails and 0 if succeeds

**Examples**

```r
download_LCMV()
```

---

**filter_cluster_data**

**Remove selected clusters**

**Description**

Remove selected clusters from the environment object.

**Usage**

```r
filter_cluster_data(environment, remove.clusters)
```
find_neighbors

Arguments

- environment: The environment object
- remove.clusters: A character vector of the clusters to be removed

Value

An environment object with selected clusters removed

Examples

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- filter_cluster_data(LCMV1, "1")
```

---

find_neighbors  
K Nearest Neighbour Search

Description

Uses a kd-tree to find the p number of near neighbours for each point in an input/output dataset.

Usage

```r
find_neighbors(data, k)
```

Arguments

- data: matrix; input data matrix
- `k`: integer; number of nearest neighbours

Details

Use the `nn2` function from the RANN package, utilizes the Approximate Near Neighbor (ANN) C++ library, which can give the exact near neighbours or (as the name suggests) approximate near neighbours to within a specified error bound. For more information on the ANN library please visit http://www.cs.umd.edu/~mount/ANN/.

Value

a n-by-k matrix of neighbor indices

Examples

```r
iris_unique <- unique(iris)  # Remove duplicates
data <- as.matrix(iris_unique[,1:4])
neighbors <- find_neighbors(data, k=10)
```
**get.cluster.names**  
*Get/Set Cluster Names by Marker Gene Expression*

**Description**

`get.cluster.names` uses predefined marker genes to assign clusters with putative cell type or state labels.

`set.cluster.names` saves the cluster names in storage and in the environment object.

**Usage**

```r
get.cluster.names(environment, types, min.fold = 1.25, max.Qval = 0.1, print = T)
set.cluster.names(environment, names)
```

**Arguments**

- `environment` environment object
- `types` data frame associating cell type or state with marker genes
- `min.fold` minimum fold change to consider a marker as overexpressed
- `max.Qval` maximum FDR q value to consider a marker as overexpressed
- `print` whether to print output calculations
- `names` cluster names defined in `get.cluster.names`

**Value**

- `get.cluster.names` returns a vector containing assigned cluster name labels
- `set.cluster.names` returns an environment object coded with cluster names

**Functions**

- `set.cluster.names`: set annotations to clusters

**Examples**

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0, min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
LCMV1 <- cluster.analysis(LCMV1)
types = rbind(
data.frame(type='Tfh',gene=c('Tcf7','Cxcr5','Bcl6')),
data.frame(type='Th1',gene=c('CxcR6','Ifng','Tbx21')),
data.frame(type='Tcmp',gene=c('Ccr7','Bcl2','Tcf7')),
```
get.robust.cluster.similarity

Description

Use cross-replicate experiment cluster similarity to remove irreproducible clusters.

Usage

get.robust.cluster.similarity(environment, similarity,
  min.sd = stats::qnorm(0.95), max.q.val = 0.01, rerun = F)

Arguments

  environment       environment object
  similarity        pearson correlation between clusters’ FC vectors defined in assess.cluster.similarity
  min.sd            minimum standard deviation for cluster reproducibility assessment
  max.q.val         maximum q value for cluster correlation cutoff
  rerun             whether to rerun the analysis or load from cache

Value

filtered cluster similarity matrix

Examples

LCM1 <- setup.LCMV_example()
LCM1 <- get.variable.genes(LCM1, min.mean = 0.1, min.frac.cells = 0,
  min.dispersion.scaled = 0.1)
LCM1 <- PCA(LCM1)
LCM1 <- cluster.analysis(LCM1)
types = rbind(
  data.frame(type='Tfh',gene=c('Tcf7','Cxc5','Bcl6')),
  data.frame(type='Th1',gene=c('Cxc5','Ifng','Tbx21')),
  data.frame(type='Tcmp',gene=c('Ccr7','Bcl2','Tcf7')))
get.robust.markers

```r
library(LCMV)
cluster_names <- get.cluster.names(LCMV1, types, min.fold = 1.0, max.Qval = 0.01)
LCMV1 <- set.cluster.names(LCMV1, names = cluster_names)
LCMV2 <- setup_LCMV_example("LCMV2")
LCMV2 <- get.variable.genes(LCMV2, min.mean = 0.1, min.fraction.cells = 0,
min.dispersion.scaled = 0.1)
LCMV2 <- PCA(LCMV2)
LCMV2 <- cluster.analysis(LCMV2)
summarize(LCMV2)
cluster_names <- get.cluster.names(LCMV2, types, min.fold = 1.0, max.Qval = 0.01)
LCMV2 <- set.cluster.names(LCMV2, names = cluster_names)
pooled_env <- setup_pooled_env()
pooled_env <- read.preclustered.datasets(pooled_env)
pooled_env <- PCA(pooled_env, clear.previously.calculated.clustering = F)
summarize(pooled_env, contrast = "datasets")
cluster.similarity <- assess.cluster.similarity(pooled_env)
similarity <- cluster.similarity$similarity
map <- cluster.similarity$map
filtered.similarity <- get.robust.cluster.similarity(
  pooled_env, similarity, min.sd = qnorm(.9), max.q.val = 0.01, rerun = F)
```

---

**get.robust.markers**  
*Get Robust Marker*

**Description**
Analysis of robust subpopulation marker prioritization

**Usage**

```r
get.robust.markers(environment, cluster_group1, cluster_group2,
  group1_label, group2_label, annotate.genes = NA, min.fold.diff = 1.5,
  min.ratio.diff = 3, QValue = 0.05)
```

**Arguments**

- **environment** environment object
- **cluster_group1** cluster group 1 to be used as a foreground
- **cluster_group2** cluster group 2 to be used as a background
- **group1_label** label for group 1
- **group2_label** label for group 2
get.variable.genes

annotate.genes  specific gene names to annotate in figure in addition to novel markers
min.fold.diff  average expression fold change cutoff
min.ratio.diff  detection ratio fold change cutoff
QValue  Qvalue cutoff

Examples

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0,
min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
LCMV1 <- cluster.analysis(LCMV1)
diff_exp <- get.robust.markers(LCMV1,
cluster_group1 = c('1','2'),
cluster_group2 = c('3','4'),
group1_label = 'CD4 T Cells',
group2_label = 'CD8 T Cells')
```

get.variable.genes  Identify Highly Variable Genes

Description

Get highly variable genes by Heteroscedasticity controlled binning of gene expression measurements within each dataset separately.

Usage

```r
get.variable.genes(environment, min.mean = 0.05, min.frac.cells = 0,
min.dispersion.scaled = 1, rerun = F)
```

Arguments

- `environment`: environment object
- `min.mean`: minimum mean expression per gene
- `min.frac.cells`: minimum fraction of cells expressing each gene
- `min.dispersion.scaled`: minimum dispersion value
- `rerun`: whether to rerun the analysis or load from cache

Value

`environment` parameter containing highly variable genes selection
initialize.project

Examples

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1)
```

---

**initialize.project**  
**Initialize the Project Environment**

**Description**

Set up a project environment variable mapped to project results folder.

**Usage**

```r
initialize.project(datasets, origins, experiments, data.path,
work.path = NULL, marker.genes = NULL, clear.history = F,
analysis.label = NULL, convert.to.mouse.gene.symbols = NULL)
```

**Arguments**

- **datasets**  
  list of dataset code names

- **origins**  
  list of dataset tissue origin/condition full name

- **experiments**  
  list of experiment design annotations

- **data.path**  
  path to where the data is located

- **work.path**  
  path to where the analysis results are stored; optional, by default, a temporary directory

- **marker.genes**  
  set of genes of interest for visualization purposes

- **clear.history**  
  whether you would like to remove any previous project by this name

- **analysis.label**  
  whether you would like to add a specific label to the analysis folder

- **convert.to.mouse.gene.symbols**  
  whether you are using human gene symbols and would like to convert them to mouse gene symbols

**Value**

- environment parameter containing file paths and experiment parameters

**Examples**

```r
data.path <- system.file("extdata", package = "robustSingleCell")
LCMV1_proj <- initialize.project(datasets = "LCMV1",
                                origins = "CD44+ cells",
                                experiments = "Rep1",
                                data.path = data.path,
                                work.path = tempdir())
```
**mitochondrial.score**  
*Compute Mitochondrial Score*

**Description**
Compute the activation level of mitochondrial genes.

**Usage**
```
mitochondrial.score(environment, control = F, knn = 10)
```

**Arguments**
- `environment`: environment object
- `control`: whether to subtract the score defined by technically similar genes
- `knn`: number of nearest neighbor

**Value**
a vector of mitochondrial genes activation score

**Examples**
```
LCMV1 <- setup_LCMV_example()
mitochondrial.score <- mitochondrial.score(LCMV1)
```

---

**PCA**  
*Parallelized PCA Analysis*

**Description**
Run PCA analysis with a simulation analysis of shuffled data to determine the appropriate number of PCs.

**Usage**
```
PCA(environment, regress = NA, groups = NA, nShuffleRuns = 10,
    threshold = 0.1, maxPCs = 100, label = NA, mem = "2GB",
    time = "0:10:00", rerun = F,
    clear.previously.calculated.clustering = T, local = F)
```
plot_contour_overlay_tSNE

Arguments

- **environment**: environment object
- **regress**: gene signature activation scores to regress
- **groups**: experimental design annotation to guide dataset-specific regression
- **nShuffleRuns**: number of shuffled analyses
- **threshold**: FDR threshold
- **maxPCs**: maximum number of possible PCs
- **label**: optional analyses label folder
- **mem**: HPC memory
- **time**: HPC time
- **rerun**: whether to rerun the analysis rather than load from cache
- **clear.previously.calculated.clustering**: whether to clear previous clustering analysis
- **local**: whether to run jobs locally on slurm instead of submitting the job

Value

environment parameter containing PC coordinates

Examples

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0,
min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
```

plot_contour_overlay_tSNE

*Plot Gene Expression on tSNE*

Description

Visualize normalized expression of selected genes on tSNE plot with color-code and contour annotation.

Usage

```r
plot_contour_overlay_tSNE(environment, genes, perplexity = 30,
max_iter = 10000, width = 10, height = 10)
```
plot_pair_scatter

Arguments

environment  environment object
genes  selected genes to visualize
perplexity  tSNE perplexity parameter
max_iter  tSNE max_iter parameter
width  pdf file canvas width
height  pdf file canvas height

Examples

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0,
min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
LCMV1 <- cluster.analysis(LCMV1)
plot_contour_overlay_tSNE(LCMV1, genes = c('Cd4', 'Cd8a'))
```

---

**plot_pair_scatter**  *Plot Pairwise Gene Scatter Plot*

**Description**

Visualize normalized expression contours of a selected gene pair across selected cluster groups.

**Usage**

```r
plot_pair_scatter(environment, gene1, gene2, cluster_group1,
cluster_group2, group1_label, group2_label, width = 10, height = 10)
```

**Arguments**

environment  environment object
gene1  selected gene number 1
gene2  selected gene number 2
cluster_group1  cluster group 1 to be visualized (one or more clusters)
cluster_group2  cluster group 2 to be visualized (one or more clusters)
group1_label  label for group 1 legend and file name
group2_label  label for group 2 legend and file name
width  pdf file canvas width
height  pdf file canvas height
plot_pca

Examples

```
LCMV1 <- setup_LCMV_example()
LCMV1 <- get_variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0,
min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
LCMV1 <- cluster.analysis(LCMV1)
cluster_names <- get.cluster.names(LCMV1, types, min.fold = 1.0, max.Qval = 0.01)
LCMV1 <- set.cluster.names(LCMV1, names = cluster_names)
plot_pair_scatter(LCMV1,
gene1 = 'Cd4',
gen2 = 'Cd8',
cluster_group1 = cluster_names[1:2],
cluster_group2 = cluster_names[3:4],
group1_label = 'CD4 T Cells',
group2_label = 'CD8 T Cells')
```

plot_pca  Plot PCA results

Description

Plot the results obtained from PCA analysis as cell embedding in 2D space and annotation of gene loadings.

Usage

```
plot_pca(environment, quantile, order)
```

Arguments

- `environment` environment object
- `quantile` quantile of PCA loadings for which to define top genes driving PCs
- `order` ordering by which to plot the in heatmap of top genes driving PCs

plot_simple_heatmap  Plot heatmap

Description

Plot heatmap given a set of markers.
Usage

plot_simple_heatmap(environment, name, markers, path = NA,
  membership = NA, normalized = NA, order = NA, width = 5,
  height = 5, scale = "row", RowSideColors = NA, counts = F,
  filter.diff.exp = F, cellnote = F, key = F, save = NA,
  sort.rows = T, sort.cols = T, Colv = F, Rowv = F,
  dendrogram = "none", main = NA)

Arguments

environment The environment object
name The file name of the figure
markers The markers to be plotted
path The path where the plot is saved; by default in TMPDIR
membership The cluster membership
normalized The normalized data matrix
order The ordering of markers
width The width of the pdf figure
height The height of the pdf figure
scale character indicating if the values should be centered and scaled in either the row
direction or the column direction, or none. The default is “none”.
RowSideColors (optional) character vector of length nrow(x) containing the color names for a
vertical side bar that may be used to annotate the rows of x.
counts Plot count matrix or not
filter.diff.exp Whether to filter for differentially expressed genes
cellnote (optional) matrix of character strings which will be placed within each color
cell, e.g. p-value symbols.
key logical indicating whether a color-key should be shown.
save The path where the plot is saved
sort.rows Whether to sort rows
sort.cols Whether to sort columns
Colv determines if and how the column dendrogram should be reordered. Has the
options as the Rowv argument above and additionally when x is a square matrix,
Colv="Rowv" means that columns should be treated identically to the rows.
Rowv determines if and how the row dendrogram should be reordered. By default, it
is TRUE, which implies dendrogram is computed and reordered based on row
means. If NULL or FALSE, then no dendrogram is computed and no reordering
is done. If a dendrogram, then it is used "as-is", ie without any reordering. If
a vector of integers, then dendrogram is computed and reordered based on the
order of the vector.
dendrogram  character string indicating whether to draw 'none', 'row', 'column' or 'both' dendrograms. Defaults to 'both'. However, if Rowv (or Colv) is FALSE or NULL and dendrogram is 'both', then a warning is issued and Rowv (or Colv) arguments are honoured.

main  main, x- and y-axis titles; defaults to none.

**Description**

Read 10X data files or a raw data matrix and perform normalization, QC filtering and duplicates removal.

**Usage**

```r
read.data(environment, genome = "mm10", min.genes.per.cell = 500, 
max.genes.per.cell.quantile = 0.98, 
max.UMIs.per.cell.quantile = 0.98, min.cells.per.gene = 1, 
max.mitochondrial.frac = 0.1, max.ribosomal.frac = NA, 
cell.filters = NA, raw.data.matrices = NA, rerun = F, 
subsample = NULL, seed = 0)
```

**Arguments**

- environment: environment object
- genome: genome annotation
- min.genes.per.cell: minimum required number of genes per cell
- max.genes.per.cell.quantile: upper quantile for number of genes per cell
- max.UMIs.per.cell.quantile: upper quantile for number of UMIs per cell
- min.cells.per.gene: minimum required number of cells per gene
- max.mitochondrial.frac: maximum fraction of reads mapped to mitochondrial genes per cell
- max.ribosomal.frac: maximum fraction of reads mapped to ribosomal genes per cell
- cell.filters: filtering option for cells based on marker genes
- raw.data.matrices: logical indicating if data matrices is provided instead of 10X dataset
- rerun: whether to rerun loading the dataset or load from cache
- subsample: number of cells to subsample
- seed: seed for subsampling of cells
Examples

```r
LCMV1 <- setup_LCMV_example()
data.path <- system.file("extdata/LCMV1_small.txt", package = "robustSingleCell")# name of list should be the same as LCMV1$datasetsraw_LCMV1 <- as.matrix(read.table(data.path, check.names = FALSE))LCMV1 <- read.data(LCMV1,
raw.data.matrices = list(LCMV1 = raw_LCMV1),
min.genes.per.cell = 100,
max.genes.per.cell.quantile = 1,
max.UMIs.per.cell.quantile = 1,
min.cells.per.gene = 1)
```

**Description**

Read previous analysis of multiple datasets to perform integrated analysis.

**Usage**

```r
read.preclustered.datasets(environment, path = NA, recursive = T, rerun = F)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>environment</td>
<td>environment object</td>
</tr>
<tr>
<td>path</td>
<td>search path for previous projects</td>
</tr>
<tr>
<td>recursive</td>
<td>recursive path search</td>
</tr>
<tr>
<td>rerun</td>
<td>whether to rerun the reading process or load from cache</td>
</tr>
</tbody>
</table>

**Examples**

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0,
min.dispersion.scaled = 0.1)LCMV1 <- PCA(LCMV1)LCMV1 <- cluster.analysis(LCMV1)
types = rbind(
data.frame(type='Tfh', gene=c('Tcf7','Ccr5','Bcl6')),
data.frame(type='Th1', gene=c('Cxcr6','Ifng','Tbx21')),
data.frame(type='Tcmp', gene=c('Ccr7','Bcl2','Tcf7')),
data.frame(type='Treg', gene=c('Foxp3','Il2ra')),
data.frame(type='Tmem', gene=c('Il7r','Ccr7')),
data.frame(type='CD8', gene=c('Cd8a')),
data.frame(type='CD4', gene = c("Cd4")),
```
read_10x_data  

**Description**
Load sparse data matrices from 10X genomics.

**Usage**
```
read_10x_data(path)
```

**Arguments**
- `path` Path to directory containing matrix.mtx, genes.tsv, and barcodes.tsv

**Value**
a matrix of genes by cells

ribosomal.score  

**Description**
Compute the activation level of ribosomal genes.

**Usage**
```
ribosomal.score(environment, control = T, knn = 10)
```
Arguments

environment  environment object
control  whether to subtract the score defined by technically similar genes
knn  number of nearest neighbor

Value

a vector of ribosomal genes activation score

Examples

LCMV1 <- setup_LCMV_example()
ribosomal.score <- ribosomal.score(LCMV1)

Rphenograph  \textit{R}phe\textit{n}o\textit{G}raph clustering

Description

\textit{R} implementation of the PhenoGraph algorithm

Usage

\texttt{Rphenograph(data, k = 30)}

Arguments

data  matrix; input data matrix
k  integer; number of nearest neighbours (default: 30)

Details

A simple \textit{R} implementation of the [PhenoGraph](http://www.cell.com/cell/abstract/S0092-8674(15)00637-6) algorithm, which is a clustering method designed for high-dimensional single-cell data analysis. It works by creating a graph (‘network’) representing phenotypic similarities between cells by calculating the Jaccard coefficient between nearest-neighbor sets, and then identifying communities using the well known [Louvain method](https://sites.google.com/site/findcommunities/) in this graph.

Value

a list contains an igraph graph object for \texttt{graph_from_data_frame} and a communities object, the operations of this class contains:

| print  | returns the communities object itself, invisibly. |
| length | returns an integer scalar. |
| sizes | returns a numeric vector. |
run_tSNE

member
ship returns a numeric vector, one number for each vertex in the graph that was the input of the community detection.

modularity returns a numeric scalar.

algorithm returns a character scalar.

crossing returns a logical vector.

is_hierarchical returns a logical scalar.

merges returns a two-column numeric matrix.

cut_at returns a numeric vector, the membership vector of the vertices.

as.dendrogram returns a dendrogram object.

show_trace returns a character vector.

code_len returns a numeric scalar for communities found with the InfoMAP method and NULL for other methods.

plot for communities objects returns NULL, invisibly.

References


Examples

library(ggplot2)
iris_unique <- unique(iris) # Remove duplicates
data <- as.matrix(iris_unique[,1:4])
Rphenograph_out <- Rphenograph(data, k = 45)
igraph::modularity(Rphenograph_out[[2]])
igraph::membership(Rphenograph_out[[2]])
iris_unique$phenograph_cluster <- factor(igraph::membership(Rphenograph_out[[2]]))

ggplot(iris_unique, aes(x=Sepal.Length, y=Sepal.Width, col=Species, shape=phenograph_cluster)) + geom_point(size = 3)+theme_bw()

---

run_tSNE

Perform tSNE Analyses

Description

Run distributed tSNE analysis for multiple input hyperparameters

Usage

run_tSNE(environment, perplexity, max_iter, rerun, local = F, mem = "4GB", time = "0:15:00")
Arguments

- **environment**: environment object
- **perplexity**: perplexity parameter of tSNE
- **max_iter**: maximum number of iterations to run the tSNE
- **rerun**: whether to rerun or load from cache
- **local**: whether to run tSNE locally
- **mem**: Memory for each job; default 4 GB
- **time**: Time for each job; default 15 minutes

Value

Distributed job identified object

Description

Set the path in the LCMV object

Usage

```r
setup_LCMV_example(dataset = "LCMV1")
```

Arguments

- **dataset**: The name of dataset (LCMV1 or LCMV2)

Value

An environment object containing the LCMV1_small data

Examples

```r
LCMV1 <- setup_LCMV_example("LCMV1")
```
**setup_pooled_env**

Set up the pooled environment

**Description**

Set the path in the LCMV object

**Usage**

```r
setup_pooled_env()
```

**Value**

An environment object containing the two LCMV datasets

**Examples**

```r
pooled_env <- setup_pooled_env()
```

**summarize**

Differential Expression and Figure Generation

**Description**

Summarize the clustering results by conducting differential expression analysis and plotting figures.

**Usage**

```r
summarize(environment, perplexity = seq(10, 30, 10), max_iter = 10000, rerun = F, order = NA, contrast = "all", min.fold = 1.5, quantile = 0.95, local = F, mem = "4GB", time = "0:15:00")
```

**Arguments**

- `environment`: environment object
- `perplexity`: perplexity parameters for tSNE analyses
- `max_iter`: maximum iterations for tSNE
- `rerun`: whether to rerun
- `order`: order in which to plot the clusters
- `contrast`: either 'all' indicating differential expression between one cluster against all others or 'datasets' indicating differential expression analysis comparing one cluster to all other within each dataset separately ('datasets' should be used in pooled analysis for optimal results)
- `min.fold`: minimum fold change for filtering final differentially expressed gene lists
### visualize.cluster.cors.heatmaps

#### Description

Plot correlation heatmaps for each pair of datasets.

#### Usage

```r
visualize.cluster.cors.heatmaps(environment, work.path, similarity, margins = c(17, 17))
```

#### Arguments

- `environment`: environment object
- `work.path`: where to locate the figures
- `similarity`: similarity matrix defined in `compare.cluster.similarity` or `get.robust.cluster.similarity`
- `margins`: The margins to the plot

#### Examples

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0, min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
LCMV1 <- cluster.analysis(LCMV1)
types = rbind(data.frame(type='Tfh',gene=c('Tcf7','Cxcr5','Bcl6')),
            data.frame(type='foamy',gene=c('Not1','Cav2a','Cav2b')))
```

---

### Examples

```r
# after running cluster.analysis()
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0, min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
LCMV1 <- cluster.analysis(LCMV1)
summarize(LCMV1)
```
data.frame(type='Th1', gene=c('Cxcr6', 'Ifng', 'Tbx21')),
data.frame(type='Tcmp', gene=c('Ccr7', 'Bcl2', 'Tcf7')),
data.frame(type='Treg', gene=c('Foxp3', 'Il2ra')),
data.frame(type='Tmem', gene=c('Il17r', 'Ccr7')),
data.frame(type='CD8', gene=c('Cd8a')),
data.frame(type='CD4', gene = c('Cd4')),
data.frame(type='Cycle', gene=c('Mki67', 'Top2a', 'Birc5'))
)
summarize(LCMV1)
cluster_names <- get.cluster.names(LCMV1, types, min.fold = 1.0, max.Qval = 0.01)
LCMV1 <- set.cluster.names(LCMV1, names = cluster_names)
LCMV2 <- setup_LCMV_example("LCMV2")
LCMV2 <- get.variable.genes(LCMV2, min.mean = 0.1, min.fraction.cells = 0, min.dispersion.scaled = 0.1)
LCMV2 <- PCA(LCMV2)
LCMV2 <- cluster.analysis(LCMV2)
summarize(LCMV2)
cluster_names <- get.cluster.names(LCMV2, types, min.fold = 1.0, max.Qval = 0.01)
LCMV2 <- set.cluster.names(LCMV2, names = cluster_names)
pooled_env <- setup_pooled_env()
pooled_env <- read.preclustered.datasets(pooled_env)
pooled_env <- PCA(pooled_env, clear.previously.calculated.clustering = F)
summarize(pooled_env, contrast = "datasets")
cluster.similarity <- assess.cluster.similarity(pooled_env)
similarity <- cluster.similarity$similarity
map <- cluster.similarity$map
filtered.similarity <- get.robust.cluster.similarity(
    pooled_env, similarity, min.sd = qnorm(0.9), max.q.val = 0.01, rerun = F)
robust.clusters <- sort(unique(c(filtered.similarity$cluster1, filtered.similarity$cluster2)))
visualize.cluster.cors.heatmaps(pooled_env, pooled_env$work.path, filtered.similarity)

visualize.cluster.similarity.stats

Plot Similarity Results

Description

Perform hierarchical clustering and plot cluster similarities according to dendrogram.

Usage

visualize.cluster.similarity.stats(environment, similarity,
hclust.resolution = 8, margins = c(40, 40))
Arguments

- `environment` environment object
- `similarity` similarity matrix defined in `compare.cluster.similarity` or `get.robust.cluster.similarity`
- `hclust.resolution` clustering resolution to impose on `hclust.cutree` function
- `margins` The margins to the plot

Examples

```r
LCMV1 <- setup_LCMV_example()
LCMV1 <- get.variable.genes(LCMV1, min.mean = 0.1, min.frac.cells = 0, min.dispersion.scaled = 0.1)
LCMV1 <- PCA(LCMV1)
LCMV1 <- cluster.analysis(LCMV1)
types = rbind(  
data.frame(type='Th1',gene=c('Tcf7','Cxcr5','Bcl6')),  
data.frame(type='Th1',gene=c('Cxcr6','Ifng','Tbx21')),  
data.frame(type='Tcm',gene=c('Ccr7','Bcl2','Tcf7')),  
data.frame(type='Tre',gene=c('Foxp3','Il2ra')),  
data.frame(type='Tmem',gene=c('Il17r','Ccr7')),  
data.frame(type='CD8',gene=c('Cd8a')),  
data.frame(type='CD4', gene = c('Cd4'))  
)
summarize(LCMV1)
cluster_names <- get.cluster.names(LCMV1, types, min.fold = 1.0, max.Qval = 0.01)
LCMV1 <- set.cluster.names(LCMV1, names = cluster_names)
LCMV2 <- setup_LCMV_example("LCMV2")
LCMV2 <- get.variable.genes(LCMV2, min.mean = 0.1, min.frac.cells = 0, min.dispersion.scaled = 0.1)
LCMV2 <- PCA(LCMV2)
LCMV2 <- cluster.analysis(LCMV2)
summarize(LCMV2)
cluster_names <- get.cluster.names(LCMV2, types, min.fold = 1.0, max.Qval = 0.01)
LCMV2 <- set.cluster.names(LCMV2, names = cluster_names)
pooled_env <- setup_pooled_env()
pooled_env <- read.preclustered.datasets(pooled_env)
pooled_env <- PCA(pooled_env, clear.previously.calculated.clustering = F)
summarize(pooled_env, contrast = "datasets")
cluster.similarity <- assess.cluster.similarity(pooled_env)
similarity <- cluster.similarity$similarity
map <- cluster.similarity$map
filtered.similarity <- get.robust.cluster.similarity(pooled_env, similarity, min.sd = qnorm(0.9), max.q.val = 0.01, rerun = F)
visualize.cluster.similarity.stats(pooled_env, filtered.similarity)
```
Index

add.confounder.variables, 2
assess.cluster.similarity, 3

cell.cycle.score, 4
cluster.analysis, 5
controlled.mean.score, 6

dendrogram, 18
download_LCMV, 7

filter_cluster_data, 7
find_neighbors, 8

get.cluster.names, 9
get.robust.cluster.similarity, 10
get.robust.markers, 11
get.variable.genes, 12
initialize.project, 13

mitochondrial.score, 14

PCA, 14
plot_contour_overlay_tsNE, 15
plot_pair_scatter, 16
plot_PCA, 17
plot_simple_heatmap, 17

read.data, 19
read.preclustered.datasets, 20
read_10x_data, 21
ribosomal.score, 21
Rphenograph, 22
run_tSNE, 23

set.cluster.names(get.cluster.names), 9
setup_LCMV_example, 24
setup_pooled_env, 25
summarize, 25

visualize.cluster.cors.heatmaps, 26
visualize.cluster.similarity.stats, 27