Package ‘iCellR’

October 9, 2021

Type Package

Title Analyzing High-Throughput Single Cell Sequencing Data

Version 1.6.5

Maintainer Alireza Khodadadi-Jamayran <alireza.khodadadi.j@gmail.com>

Description A toolkit that allows scientists to work with data from single cell sequencing technologies such as scRNA-seq, scVDJ-seq, scATAC-seq, CITSeq and Spatial Transcriptomics (ST). Single (i) Cell R package (‘iCellR’) provides unprecedented flexibility at every step of the analysis pipeline, including normalization, clustering, dimensionality reduction, imputation, visualization, and so on. Users can design both unsupervised and supervised models to best suit their research. In addition, the toolkit provides 2D and 3D interactive visualizations, differential expression analysis, filters based on cells, genes and clusters, data merging, normalizing for dropouts, data imputation methods, correcting for batch differences, pathway analysis, tools to find marker genes for clusters and conditions, predict cell types and pseudotime analysis. See Khodadadi-Jamayran, et al (2020) <doi:10.1101/2020.05.05.078550> and Khodadadi-Jamayran, et al (2020) <doi:10.1101/2020.03.31.019109> for more details.

Depends R (>= 3.3.0), ggplot2, plotly

Imports Matrix, Rtsne, gridExtra, ggrepel, ggpubr, scatterplot3d, RColorBrewer, knitr, NbClust, shiny, pheatmap, ape, ggdendro, plyr, reshape, Hmisc, htmlwidgets, methods, uwot, hdf5r, progress, igraph, data.table, Repp, RANN, jsonlite, png

License GPL-2

LinkingTo Rcpp

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

BugReports https://github.com/rezakj/iCellR/issues

URL https://github.com/rezakj/iCellR

NeedsCompilation yes
**Author**  Alireza Khodadadi-Jamayran [aut, cre]
(<https://orcid.org/0000-0003-2495-7504>),
Joseph Pucella [aut, ctb] (<https://orcid.org/0000-0003-0875-8046>),
Hua Zhou [aut, ctb] (<https://orcid.org/0000-0003-1822-1306>),
Nicole Doudican [aut, ctb] (<https://orcid.org/0000-0003-3827-9644>),
John Carucci [aut, ctb] (<https://orcid.org/0000-0001-6817-9439>),
Adriana Heguy [aut, ctb],
Boris Reizis [aut, ctb] (<https://orcid.org/0000-0003-1140-7853>),
Aristotelis Tsirigos [aut, ctb]
(<https://orcid.org/0000-0002-7512-8477>)

**Repository**  CRAN

**Date/Publication**  2021-10-09 15:00:15 UTC

**R topics documented:**

- add.10x.image ........................................ 3
- add.adt .................................................. 4
- add.vdj .................................................. 5
- adt.rna.merge ........................................... 5
- capture.image.10x ..................................... 6
- cc .......................................................... 6
- cell.cycle ................................................ 7
- cell.filter .............................................. 7
- cell.gating ............................................. 8
- cell.type.pred ......................................... 9
- change.clust ........................................... 10
- clono.plot .............................................. 10
- clust.avg.exp .......................................... 11
- clust.cond.info ........................................ 12
- clust.ord ................................................ 13
- clust.rm ................................................ 13
- clust.stats.plot ....................................... 14
- cluster.plot ........................................... 15
- data.aggregation ...................................... 16
- data.scale .............................................. 17
- down.sample ............................................ 18
- find.dim.genes ........................................ 18
- findMarkers ............................................ 19
- find_neighbors .......................................... 20
- g2m.phase .............................................. 20
- gate.to.clust .......................................... 21
- gene.plot .............................................. 21
- gene.stats .............................................. 23
- gg.cor .................................................... 23
- heatmap.gg.plot ..................................... 24
- hto.anno ................................................ 26
- i.score ................................................... 26
Description

This function takes a list of image data and adds it to the iCellR object.
Usage

```r
add.10x.image(x = NULL, image.data.list = NULL, condition.names = NULL)
```  
Arguments

- **x**: An object of class iCellR.
- **image.data.list**: A character vector of list object names. Lists should be made using "image.capture.10x" function.
- **condition.names**: A character vector of condition names.

Value

An object of class iCellR

---

### add.adt

**Add CITE-seq antibody-derived tags (ADT)**

**Description**

This function takes a data frame of ADT values per cell and adds it to the iCellR object.

**Usage**

```r
add.adt(x = NULL, adt.data = "data.frame")
```  
Arguments

- **x**: An object of class iCellR.
- **adt.data**: A data frame containing ADT counts for cells.

Value

An object of class iCellR
add.vdj

Add V(D)J recombination data

Description
This function takes a data frame of VDJ information per cell and adds it to the iCellR object.

Usage
add.vdj(x = NULL, vdj.data = "data.frame")

Arguments
x
An object of class iCellR.
vdj.data
A data frame containing VDJ information for cells.

Value
An object of class iCellR

adt.rna.merge

Merge RNA and ADT data

Description
This function is to merge the RNA and ADT data to the main.data slot of the iCellR object.

Usage
adt.rna.merge(x = NULL, adt.data = "raw")

Arguments
x
An object of class iCellR.
adt.data
Choose from raw or main (normalized) ADT data, default = "raw".

Value
An object of class iCellR
**capture.image.10x**  
*Read 10X image data*

**Description**

This function takes 10X image data files and converts them to proper file format for iCellR.

**Usage**

```r
capture.image.10x(dir.10x = NULL)
```

**Arguments**

- `dir.10x` A directory that includes the 10X image files (scalefactors_json.json, tissue_lowres_image.png and tissue_positions_list.csv).

**Value**

A list object

---

**cc**  
*Calculate Cell cycle phase prediction*

**Description**

This function takes an object of class iCellR and assigns cell cycle stage for the cells.

**Usage**

```r
cc(object = NULL, s.genes = s.phase, g2m.genes = g2m.phase)
```

**Arguments**

- `object` A data frame containing gene counts for cells.
- `s.genes` Genes that are used as a marker for S phase.
- `g2m.genes` Genes that are used as a marker for G2 and M phase.

**Value**

The data frame object
**cell.cycle**  
*Cell cycle phase prediction*

**Description**
This function takes an object of class iCellR and assigns cell cycle stage for the cells.

**Usage**
```r
cell.cycle(
  object = NULL,
  scoring.List = NULL,
  return.stats = FALSE,
  scoring.method = "tirosh"
)
```

**Arguments**
- `object`: A data frame containing gene counts for cells.
- `scoring.List`: Genes that are used as a marker for phases.
- `return.stats`: Return the data or object. If FALSE the object would be returned.
- `scoring.method`: Choose from "coverage" or "tirosh" for scoring method.

**Value**
The data frame object

---

**cell.filter**  
*Filter cells*

**Description**
This function takes an object of class iCellR and filters the raw data based on the number of UMIs, genes per cell, percentage of mitochondrial genes per cell, genes, gene expression and cell ids.

**Usage**
```r
cell.filter(
  x = NULL,
  min.mito = 0,
  max.mito = 1,
  min.genes = 0,
  max.genes = Inf,
  min.umis = 0,
  max.umis = Inf,
)```
filter.by.cell.id = "character",
keep.cell.id = "character",
filter.by.gene = "character",
filter.by.gene.exp.min = 1
}

Arguments

x An object of class iCellR.
min.mito Min rate for mitochondrial gene expression per cell, default = 0.
max.mito Max rate for mitochondrial gene expression per cell, default = 1.
min.genes Min number genes per cell, default = 0.
max.genes Max number genes per cell, default = Inf.
min.umis Min number UMIs per cell, default = 0.
max.umis Max number UMIs per cell, default = Inf.
filter.by.cell.id A character vector of cell ids to be filtered out.
keep.cell.id A character vector of cell ids to keep.
filter.by.gene A character vector of gene names to be filtered by thier expression. If more then
one gene is defined it would be OR not AND.
filter.by.gene.exp.min Minimum gene expression to be filtered by the genes set in filter.by.gene, default = 1.

Value

An object of class iCellR.

cell.gating Cell gating

Description

This function takes an object of class iCellR and a 2D tSNE or UMAP plot and gates around cells
to get their ids.

Usage

cell.gating(x = NULL, my.plot = NULL, plot.type = "tsne")

Arguments

x An object of class iCellR.
my.plot The plot to use for gating. Must be a 2D plot.
plot.type Choose from knetl, umap and tsne, default = NULL.
cell.type.pred

Value

An object of class iCellR.

Description

This function takes an object of class iCellR and genes and provides a heatmap.

Usage

```r
cell.type.pred(
  immgen.data = "rna",
  gene = "NULL",
  top.cell.types = 50,
  plot.type = "heatmap",
  heat.colors = c("blue", "white", "red")
)
```

Arguments

- `immgen.data`: Choose from "GSE109125", "GSE122108", "GSE122597", "GSE124829", "GSE15907", "GSE37448", "rna", "uli.rna" or "mca", default = "rna"
- `gene`: A set of gene names to used to predict cell type.
- `top.cell.types`: Top cell types sorted by cumulative expression, default = 25.
- `plot.type`: Choose from "heatmap" od "point.plot", default = "heatmap"
- `heat.colors`: Colors for heatmap, default = c("blue", "white", "red")

Value

An object of class iCellR
change.clust | Change the cluster number or re-name them

Description
This function re-names the clusters in the best.clust slot of the iCellR object.

Usage
change.clust(x = NULL, change.clust = 0, to.clust = 0, clust.reset = FALSE)

Arguments
- x: An object of class iCellR.
- change.clust: The name of the cluster to be changed.
- to.clust: The new name for the cluster.
- clust.reset: Reset to the original clustering.

Value
An object of class iCellR.

clono.plot | Make 2D and 3D scatter plots for clonotypes.

Description
This function takes an object of class iCellR and provides plots for clonotypes.

Usage
clono.plot(
  x = NULL,
  plot.data.type = "tsne",
  clonotype.column = 1,
  barcode.column = 2,
  clono = NULL,
  conds.to.plot = NULL,
  clust.dim = 2,
  cell.size = 1,
  cell.colors = c("red", "gray"),
  box.cell.col = "black",
  back.col = "white",
  cell.transparency = 1,
  interactive = TRUE,
  out.name = "plot"
)
The function `clust.avg.exp` creates a data frame of mean expression of genes per cluster. It takes an object of class `iCellR` and creates an average gene expression for every cluster. The function has several arguments:

- `x`: An object of class `iCellR`.
- `plot.data.type`: Choose from "tsne" and "pca", default = "tsne".
- `clonotype.column`: The column which has the clonotype IDs, default = 2.
- `barcode.column`: The column which has the barcode IDs, default = 1.
- `clono`: A clonotype name to be plotted, default = NULL.
- `conds.to.plot`: Choose one condition you want to see in the plot, default = NULL (all conditions).
- `clust.dim`: 2 for 2D plots and 3 for 3D plots, default = 2.
- `cell.size`: A number for the size of the points in the plot, default = 1.
- `cell.colors`: Colors for heat mapping the points in "scatterplot", default = c("gray","red").
- `box.cell.col`: Choose a color for box default = "black".
- `back.col`: A color for the plot background, default = "black".
- `cell.transparency`: Color transparency for points, default = 0.5.
- `interactive`: If set to TRUE an intractive HTML file will be created, default = TRUE.
- `out.name`: If "interactive" is set to TRUE, the output name for HTML, default = "plot".

The function returns an object of class `iCellR`.

### Usage

```r
clust.avg.exp(  
  x = NULL,  
  data.type = "main",  
  conds.to.avg = NULL,  
  rounding.digits = 4,  
  round.num = FALSE  
)
```
Arguments

- **x** An object of class iCellR.
- **data.type** Choose from "main", "atac", "atac.imputed" and "imputed", default = "main"
- **conds.to.avg** Choose the conditions you want to average, default = NULL (all conditions).
- **rounding.digits** integer indicating the number of decimal places (round) or significant digits (significant) to be used.
- **round.num** Rounding of Numbers, default = FALSE.

Value

An object of class iCellR.

---

**clust.cond.info** Calculate cluster and conditions frequencies

Description

This function takes an object of class iCellR and calculates cluster and conditions frequencies.

Usage

```r
clust.cond.info(
  x = NULL,
  plot.type = "pie",
  my.out.put = "data",
  normalize.ncell = TRUE,
  normalize.by = "percentage"
)
```

Arguments

- **x** An object of class iCellR.
- **plot.type** Choose from pie/pie.cond or bar/bar.cond, default = pie.
- **my.out.put** Chose from "data" or "plot", default = "data".
- **normalize.ncell** If TRUE the values will be normalized to the number of cells by downsampling.
- **normalize.by** Chose from "sf" (size factor) or "percentage", default = "percentage".

Value

An object of class iCellR.
clust.ord

Sort and relabel the clusters randomly or based on pseudotime

Description
This function takes an object of class iCellR and re-orders the clusters based on pseudotime (distance).

Usage
clust.ord(
    x = NULL,
    top.rank = 500,
    dist.method = "euclidean",
    clust.method = "complete",
    how.to.order = "distance"
)

Arguments
- **x** An object of class iCellR.
- **top.rank** A number. Taking the top genes ranked by base mean, default = 500.
- **dist.method** Choose from "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski", default = "euclidean".
- **clust.method** Choose from "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median" or "centroid", default = "complete".
- **how.to.order** Choose from "distance" and "random".

Value
An object of class iCellR.

clust.rm
Remove the cells that are in a cluster

Description
This function removes the cells from a designated cluster. Notice the cells will be removed from the main data (raw data would still have the original data).

Usage
clust.rm(x = NULL, clust.to.rm = "numeric")
Arguments

- **x**: A data frame containing gene counts for cells.
- **clust.to.rm**: The name of the cluster to be removed.

Value

An object of class `iCellR`

Description

This function takes an object of class `iCellR` and creates QC plot.

Usage

```r
clust.stats.plot(
  x = NULL,
  plot.type = "box.mito",
  conds.to.plot = NULL,
  cell.color = "slategray3",
  cell.size = 1,
  cell.transparency = 0.5,
  box.color = "red",
  box.line.col = "green",
  back.col = "white",
  notch = FALSE,
  interactive = TRUE,
  out.name = "plot"
)
```

Arguments

- **x**: An object of class `iCellR`.
- **plot.type**: Choose from "bar.cc", "pie.cc", "box.umi", "box.mito", "box.gene", default = "box.mito".
- **conds.to.plot**: Choose the conditions you want to see in the plot, default = NULL (all conditions).
- **cell.color**: Choose a color for points in the plot.
- **cell.size**: A number for the size of the points in the plot, default = 1.
- **cell.transparency**: Color transparency for points in "scatterplot" and "boxplot", default = 0.5.
- **box.color**: A color for the boxes in the "boxplot", default = "red".
cluster.plot

box.line.col  A color for the lines around the "boxplot", default = "green".
back.col  Background color, default = "white"
notch  Notch the box plots, default = FALSE.
interactive  If set to TRUE an interactive HTML file will be created, default = TRUE.
out.name  If "interactive" is set to TRUE, the output name for HTML, default = "plot".

Value

An object of class iCellR.

Description

This function takes an object of class iCellR and creates plots to see the clusters.

Usage

cluster.plot(
  x = NULL,
  cell.size = 0.5,
  plot.type = "tsne",
  cell.color = "black",
  back.col = "white",
  col.by = "clusters",
  cond.facet = FALSE,
  cond.shape = FALSE,
  anno.clust = FALSE,
  anno.size = 4,
  cell.transparency = 1,
  clust.dim = 2,
  angle = 20,
  clonotype.max = 10,
  density = FALSE,
  interactive = TRUE,
  static3D = FALSE,
  out.name = "plot"
)

Arguments

x  An object of class iCellR.

 cell.size  A numeric value for the size of the cells, default = 1.

 plot.type  Choose between "tsne", "pca", "umap", "knetl", "diffusion", default = "tsne".
data.aggregation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell.color</td>
<td>Choose cell color if col.by = &quot;monochrome&quot;, default = &quot;black&quot;.</td>
</tr>
<tr>
<td>back.col</td>
<td>Choose background color, default = &quot;black&quot;.</td>
</tr>
<tr>
<td>col.by</td>
<td>Choose between &quot;clusters&quot;, &quot;conditions&quot;, &quot;cc&quot; (cell cycle) or &quot;monochrome&quot;,</td>
</tr>
<tr>
<td></td>
<td>default = &quot;clusters&quot;.</td>
</tr>
<tr>
<td>condfacet</td>
<td>Show the conditions in separate plots.</td>
</tr>
<tr>
<td>cond.shape</td>
<td>If TRUE the conditions will be shown in shapes.</td>
</tr>
<tr>
<td>anno.clust</td>
<td>Annotate cluster names on the plot, default = TRUE.</td>
</tr>
<tr>
<td>anno.size</td>
<td>If anno.clust is TRUE set font size, default = 3.</td>
</tr>
<tr>
<td>cell.transparency</td>
<td>A numeric value between 0 to 1, default = 0.5.</td>
</tr>
<tr>
<td>clust.dim</td>
<td>A numeric value for plot dimensions. Choose either 2 or 3, default = 2.</td>
</tr>
<tr>
<td>angle</td>
<td>A number to rotate the non-interactive 3D plot.</td>
</tr>
<tr>
<td>clonotype.max</td>
<td>Number of clonotype to plot, default = 10.</td>
</tr>
<tr>
<td>density</td>
<td>If TRUE the density plots for PCA/tSNE second dimension will be created,</td>
</tr>
<tr>
<td></td>
<td>default = FALSE.</td>
</tr>
<tr>
<td>interactive</td>
<td>If TRUE an html interactive file will be made, default = TRUE.</td>
</tr>
<tr>
<td>static3D</td>
<td>If TRUE a non-interactive 3D plot will be made.</td>
</tr>
<tr>
<td>out.name</td>
<td>Output name for html file if interactive = TRUE, default = &quot;plot&quot;.</td>
</tr>
</tbody>
</table>

Value

An object of class iCellR.

---

data.aggregation  Merge multiple data frames and add the condition names to their cell ids

Description

This function takes data frame and merges them while also adding condition names to cell ids.

Usage

data.aggregation(samples = NULL, condition.names = NULL)

Arguments

- samples       A character vector of data.frame object names.
- condition.names A character vector of data.frame condition names.

Value

An object of class iCellR
Examples

demo <- read.table(
  file = system.file('extdata', 'demo_data.txt', package = 'iCellR'),
  as.is = TRUE)

# Lets divide your sample in to 3 samples as if you have 3 samples and want to merge them.
sample1 <- demo[1:30]
sample2 <- demo[31:60]
sample3 <- demo[61:90]

# merge all 3 data and add condition names
demo <- data.aggregation(samples =
  c("sample1","sample2","sample3"),
  condition.names = c("WT","ctrl","KO"))
head(demo)[1:4]

# make iCellR object
myDemo.obj <- make.obj(demo)


---

data.scale  Scale data

Description

This function takes an object of class iCellR and scales the normalized data.

Usage

data.scale(x = NULL)

Arguments

x  An object of class iCellR.

Value

An object of class iCellR.
**down.sample**  
*Down sample conditions*

**Description**
This function takes an object of class iCellR and down samples the condition to have equal number of cells in each condition.

**Usage**
```r
down.sample(x = NULL)
```

**Arguments**
- `x`: An object of class iCellR.

**Value**
An object of class iCellR.

---

**find.dim.genes**  
*Find model genes from PCA data*

**Description**
This function takes an object of class iCellR finds the model genes to run a second round of PCA.

**Usage**
```r
find.dim.genes(x = NULL, dims = 1:10, top.pos = 15, top.neg = 5)
```

**Arguments**
- `x`: An object of class iCellR.
- `dims`: PC dimentions to be used.
- `top.pos`: Number of top positive marker genes to be taken from each PC, default = 15.
- `top.neg`: Number of top negative marker genes to be taken from each PC, default = 5.

**Value**
An object of class iCellR.
findMarkers

Find marker genes for each cluster

Description
This function takes an object of class iCellR and performs differential expression (DE) analysis to find marker genes for each cluster.

Usage
findMarkers(
  x = NULL,
  data.type = "main",
  pval.test = "t.test",
  p.adjust.method = "hochberg",
  fold.change = 2,
  padjval = 0.1,
  Inf.FCs = FALSE,
  uniq = FALSE,
  positive = TRUE
)

Arguments

x An object of class iCellR.
data.type Choose from "main", "atac", "atac.imputed" and "imputed", default = "main"
pval.test Choose from "t.test", "wilcox.test", default = "t.test".
fold.change A number that designates the minimum fold change for output, default = 2.
padjval Minimum adjusted p value for output, default = 0.1.
Inf.FCs If set to FALSE the infinite fold changes would be filtered from output, default = FALSE.
uniq If set to TRUE only genes that are a marker for only one cluster would be in the output, default = FALSE.
positive If set to FALSE both the up regulated (positive) and down regulated (negative) markers would be in the output, default = TRUE.

Value
An object of class iCellR
**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**

`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/](http://www.cs.umd.edu/~mount/ANN/).

---

**g2m.phase**  
*A dataset of G2 and M phase genes*

**Description**

A dataset containing the genes for G2 and M phase

**Usage**

`g2m.phase`

**Format**

A character with 54 genes

**Source**

gate.to.clust

Assign cluster number to cell ids

Description

This function takes an object of class iCellR and assigns cluster number to a vector of cell ids.

Usage

gate.to.clust(x = NULL, my.gate = NULL, to.clust = 0)

Arguments

x
An object of class iCellR.
my.gate
A vector of cell ids.
to.clust
A cluster id to be assigned to the provided cell ids.

Value

An object of class iCellR.

gene.plot

Make scatter, box and bar plots for genes

Description

This function takes an object of class iCellR and provides plots for genes.

Usage

gene.plot(
  x = NULL,
  gene = NULL,
  cond.shape = FALSE,
  conds.to.plot = NULL,
  data.type = "main",
  box.to.test = 0,
  box.pval = "sig.signs",
  plot.data.type = "tsne",
  scaleValue = TRUE,
  min.scale = 0,
  max.scale = 2.5,
  clust.dim = 2,
  col.by = "clusters",
  plot.type = "scatterplot"
cell.size = 1,
cell.colors = c("gray", "red"),
box.cell.col = "black",
box.color = "red",
box.line.col = "green",
back.col = "white",
cell.transparency = 1,
box.transparency = 0.5,
interactive = TRUE,
out.name = "plot",
write.data = FALSE
)

Arguments

x An object of class iCellR.
gene Gene name/names to be plotted.
cond.shape If TRUE the conditions will be shown in shapes.
conds.to.plot Choose the conditions you want to see in the plot, default = NULL (all conditions).
data.type Choose from "main", "atac", "atac.imputed" and "imputed", default = "main".
box.to.test A cluster number so that all the boxes in the box plot would be compared to. If set to "0" the cluster with the highest average would be chosen, default = 0.
box.pval Choose from "sig.values" and "sig.signs". If set to "sig.signs" p values would be replaced with signs ("na", "+", "+*", "+**", "+***"), default = "sig.signs".
scaleValue Scale the colors, default = FALSE.
min.scale If scaleValue = TRUE, set a number for min, default = -2.5.
max.scale If scaleValue = TRUE, set a number for max, default = 2.5.
clust.dim 2 for 2D plots and 3 for 3D plots, default = 2.
col.by Choose from "clusters" and "conditions", default = "clusters".
plot.type Choose from "scatterplot", "boxplot" and "barplot", default = "scatterplot".
cell.size A number for the size of the points in the plot, default = 1.
cell.colors Colors for heat mapping the points in "scatterplot", default = c("gray","red").
box.cell.col A color for the points in the box plot, default = "black".
box.color A color for the boxes in the "boxplot", default = "red".
box.line.col A color for the lines around the "boxplot", default = "green".
back.col A color for the plot background, default = "black".
cell.transparency Color transparency for points in "scatterplot" and "boxplot", default = 1.
box.transparency Color transparency for box in "boxplot", default = 0.5.
interactive  If set to TRUE an interactive HTML file will be created, default = TRUE.
out.name    If "interactive" is set to TRUE, the output name for HTML, default = "plot".
write.data Write export the data used for the plot plot, default = TFALSE.

Value
An object of class iCellR.

Gene statistics
Make statistical information for each gene across all the cells (SD, mean, expression, etc.)

Description
This function takes an object of class iCellR and provides some statistical information for the genes.

Usage
gene.stats(x = NULL, which.data = "raw.data", each.cond = FALSE)

Arguments
  x           An object of class iCellR.
  which.data  Choose from "raw.data" or "main.data", default = "raw.data".
  each.cond   If TRUE each condition will be calculated, default = FALSE.

Value
An object of class iCellR.

gene correlation
Gene-gene correlation. This function helps to visualize and calculate gene-gene correlations.

Description
Gene-gene correlation. This function helps to visualize and calculate gene-gene correlations.
Usage

```r
gg.cor(
  x = NULL,
  data.type = "imputed",
  gene1 = NULL,
  gene2 = NULL,
  conds = NULL,
  clusts = NULL,
  cell.size = 1,
  cell.transparency = 0.5,
  interactive = TRUE,
  out.name = "plot"
)
```

Arguments

- **x**: An object of class iCellR.
- **data.type**: Choose from imputed and main, default = "imputed".
- **gene1**: First gene name.
- **gene2**: Second gene name.
- **conds**: Filter only one condition (only one), default is all conditions.
- **clusts**: Choose clusters to plot.
- **cell.size**: A numeric value for the size of the cells, default = 1.
- **cell.transparency**: A numeric value between 0 to 1, default = 0.5.
- **interactive**: If TRUE an html interactive file will be made, default = TRUE.
- **out.name**: Output name for html file if interactive = TRUE, default = "plot".

Value

An object of class iCellR

---

**heatmap.gg.plot**

Create heatmaps for genes in clusters or conditions.

Description

This function takes an object of class iCellR and genes and provides a heatmap.
**Usage**

```r
heatmap.gg.plot(
  x = NULL,
  gene = "NULL",
  cell.sort = FALSE,
  data.type = "main",
  cluster.by = "clusters",
  conds.to.plot = NULL,
  min.scale = -2.5,
  max.scale = 2.5,
  interactive = TRUE,
  cex.col = 10,
  cex.row = 10,
  no.key = FALSE,
  out.name = "plot",
  heat.colors = c("blue", "white", "red")
)
```

**Arguments**

- **x**: A data frame containing gene counts for cells.
- **gene**: A set of gene names to be heatmapped.
- **cell.sort**: If FALSE the cells will not be sorted based on their distance, default = TRUE.
- **data.type**: Choose from "main", "atac", atac.imputed and "imputed", default = "main".
- **cluster.by**: Choose from "clusters" or "none", default = "clusters".
- **conds.to.plot**: Choose the conditions you want to see in the plot, default = NULL (all conditions).
- **min.scale**: Set a minimum color scale, default = -2.5.
- **max.scale**: Set a maximum color scale, default = 2.5.
- **interactive**: If TRUE an html interactive file will be made, default = TRUE.
- **cex.col**: Choose a size, default = 10.
- **cex.row**: Choose a size, default = 10.
- **no.key**: If you want a color legend key, default = FALSE.
- **out.name**: Output name for html file if interactive = TRUE, default = "plot".
- **heat.colors**: Colors for heatmap, default = c("blue", "white", "red").

**Value**

An object of class iCellIR
hto.anno  

**Description**

Demultiplexing HTOs

**Usage**

hto.anno(hto.data = "data.frame", cov.thr = 10, assignment.thr = 80)

**Arguments**

- **hto.data**: HTO raw data
- **cov.thr**: A number which average coverage is divided by to set a threshold for low coverage. For example 10 means it is 10 time less than the average. default = 10.
- **assignment.thr**: A percent above which you decide to set as a good sample assignment/HTO, default = 80.

**Value**

An object of class iCellR

---

i.score  

**Description**

This function takes an object of class iCellR and assignes cell cycle stage for the cells.

**Usage**

i.score(
  object = NULL,
  data.type = "main.data",
  scoring.List = NULL,
  return.stats = TRUE,
  scoring.method = "tirosh"
)

---

Demultiplexing HTOs

Cell cycle phase prediction
Arguments

object A data frame containing gene counts for cells.
data.type Choose from "raw.data" or "main.data", "imputed.data", default = "main.data".
score.List Genes that are used as a marker for phases.
return.stats Return the data or object. If FALSE the object would be returned.
score.method Choose from "tirosh" (Tirosh, et. al. 2016), mean, sum, gsva, ssgsea, zscore and plage., default = "tirosh".

Value

The data frame object

iba

iCellR Batch Alignment (IBA)

Description

This function takes an object of class iCellR and runs CCCA or CPCA batch alignment.

Usage

iba(
  x = NULL,
  dims = 1:30,
  k = 10,
  ba.method = "CPCA",
  method = "base.mean.rank",
  top.rank = 500,
  plus.log.value = 0.1,
  scale.data = TRUE,
  gene.list = "character"
)

Arguments

x An object of class iCellR.
dims PC dimentions to be used
k number of neighboring cells for KNN, default = 10.
ba.method Batch alignment method. Choose from "CCCA" and "CPCA", default = "CPCA".
method Choose from "base.mean.rank" or "gene.model", default is "base.mean.rank". If
  gene.model is chosen you need to provide gene.list.
top.rank A number. Taking the top genes ranked by base mean, default = 500.
plus.log.value A number to add to each value in the matrix before log transformation to avoid
  Inf numbers, default = 0.1.
scale.data  If TRUE the data will be scaled (log2 + plus.log.value), default = TRUE.
gene.list  A character vector of genes to be used for PCA. If "clust.method" is set to "gene.model", default = "my_model_genes.txt".

iclust  iCellR Clustering

Description

This function takes an object of class iCellR and finds optimal number of clusters and clusters the data.

Usage

iclust(
  x = NULL,
  dist.method = "euclidean",
  sensitivity = 100,
  data.type = "pca",
  dims = 1:10,
  return.graph = FALSE
)

Arguments

  x  An object of class iCellR.
  dist.method  the distance measure to be used to compute the dissimilarity matrix. This must be one of: "euclidean", "maximum", "mandatattan", "canberra", "binary", "minkowski" or "NULL". By default, distance="euclidean". If the distance is "NULL", the dissimilarity matrix (diss) should be given by the user. If distance is not "NULL", the dissimilarity matrix should be "NULL".
  sensitivity  The higher the number the less sensitivity, default = 100.
  data.type  Choose between "tsne", "pca", "umap", default = "pca".
  dims  PCA dimensions to be use for clustering, default = 1:10.
  return.graph  return igraph object, default = FALSE.

Value

An object of class iCellR.
load.h5  

**Load h5 data as data.frame**

**Description**
This function reads hdf5 files.

**Usage**
load.h5(filename, feature.names = TRUE, uniq.rows = TRUE)

**Arguments**
- `filename`: path to the input (h5) file
- `feature.names`: row names to be feature names or ID numbers.
- `uniq.rows`: make row names unique.

**Value**
The data frame object

load10x  

**Load 10X data as data.frame**

**Description**
This function takes 10X data files barcodes.tsv, genes.tsv and matrix.mtx and converts them to proper matrix file for iCellR.

**Usage**
load10x(dir.10x = NULL, gene.name = 2)

**Arguments**
- `dir.10x`: A directory that includes the 10X barcodes.tsv, genes.tsv and matrix.mtx files.
- `gene.name`: Gene names or ids column number, default = 2.

**Value**
The data frame object
Examples

```r
my.data <- load10x(system.file("extdata", "filtered_gene_bc_matrices", package = "iCellR"))

# See first few rows and columns
head(my.data)[1:5]
```

---

### make.bed

#### Make BED Files

**Description**

This function takes peak marker files and makes the bed files per cluster.

**Usage**

```r
make.bed(x = NULL)
```

**Arguments**

- `x`: Peak marker file.

**Value**

Bed files

---

### make.gene.model

#### Make a gene model for clustering

**Description**

This function takes an object of class iCellR and provides a gene list for clustering based on the parameters set in the model.

**Usage**

```r
make.gene.model(
  x = NULL,
  dispersion.limit = 1.5,
  base.mean.rank = 500,
  gene.num.max = 2000,
  non.sig.col = "darkgray",
  right.sig.col = "chartreuse3",
  left.sig.col = "cadetblue3",
  disp.line.col = "black",
  rank.line.col = "red",
```
make.gene.model

```r
cell.size = 1.75,
cell.transparency = 0.5,
no.mito.model = TRUE,
no.cell.cycle = TRUE,
mark.mito = TRUE,
interactive = TRUE,
out.name = "plot"
)
```

Arguments

- `x`: An object of class `iCellR`.
- `dispersion.limit`: A number for taking the genes that have dispersion above this number, default = 1.5.
- `base.mean.rank`: A number taking the top genes ranked by base mean, default = 500.
- `gene.num.max`: Maximum number of genes, default = 2000.
- `non.sig.col`: Color for the genes not used for the model, default = "darkgray".
- `right.sig.col`: Color for the genes above the dispersion limit, default = "chartreuse3".
- `left.sig.col`: Color for the genes above the rank limit, default = "cadetblue3".
- `disp.line.col`: Color of the line for dispersion limit, default = "black".
- `rank.line.col`: Color of the line for rank limit, default = "red".
- `my.out.put`: Chose from "data" or "plot", default = "data".
- `cell.size`: A number for the size of the points in the plot, default = 1.75.
- `cell.transparency`: Color transparency for the points in the plot, default = 0.5.
- `no.mito.model`: If set to TRUE, mitochondrial genes would be excluded from the gene list made for clustering, default = TRUE.
- `no.cell.cycle`: If TRUE the cell cycle genes will be removed (s.phase and g2m.phase), default = TRUE.
- `mark.mito`: Mark mitochondrial genes in the plot, default = TRUE.
- `interactive`: If set to TRUE an interactive HTML file will be created, default = TRUE.
- `out.name`: If "interactive" is set to TRUE, the output name for HTML, default = "plot".

Value

An object of class `iCellR`. 
make.obj

Create an object of class iCellR.

Description
This function takes data frame and makes an object of class iCellR.

Usage
make.obj(x = NULL)

Arguments
x
A data frame containing gene counts for cells.

Value
An object of class iCellR

Examples
demo <- read.table(
  file = system.file('extdata', 'demo_data.txt', package = 'iCellR'),
  as.is = TRUE)
myDemo.obj <- make.obj(demo)
myDemo.obj

myImp

Impute data

Description
This function imputes data.

Usage
myImp(x = NULL)

Arguments
x
An object of class iCellR.

Value
An object of class iCellR
**norm.adt**

Normalize ADT data. This function takes data frame and Normalizes ADT data.

**Description**

Normalize ADT data. This function takes data frame and Normalizes ADT data.

**Usage**

```r
norm.adt(x = NULL)
```

**Arguments**

- `x`:
  - An object of class iCellR.

**Value**

An object of class iCellR

---

**norm.data**

Normalize data

**Description**

This function takes an object of class iCellR and normalized the data based on "global.glsf", "ranked.glsf" or "spike.in" methods.

**Usage**

```r
norm.data(
  x = NULL,
  norm.method = "ranked.glsf",
  top.rank = 500,
  spike.in.factors = NULL,
  rpm.factor = 1000,
  rounding.digits = 3,
  round.num = TRUE,
  ATAC.data = FALSE,
  ATAC.filter = TRUE
)
```
Arguments

x                     An object of class iCellR.
norm.method           Choose a normalization method, there are three options currently. Choose from "global.glsf", "ranked.glsf","spike.in" or no.norm, default = "ranked.glsf".
top.rank              If the method is set to "ranked.glsf", you need to set top number of genes sorted based on global base mean, default = 500.
spike.in.factors      A numeric vector of spike-in values with the same cell id order as the main data.
rpm.factor            If the method is set to "rpm" the library sizes would be divided by this number, default = 1000 (higher numbers recommended for bulk RNA-Seq).
rounding.digits       Integer indicating the number of decimal places (round) or significant digits (signif) to be used.
round.num             Rounding of Numbers, default = FALSE.
ATAC.data             If TRUE, it would normalize ATAC-Seq data and not RNA-Seq, default = FALSE.
ATAC.filter           If TRUE, all the cells filtered in RNA-Seq will be filtered in ATAC-Seq. This needs to be done for both data to match, default = TRUE.

Value

An object of class iCellR.

---

opt.pcs.plot  
Find optimal number of PCs for clustering

Description

This function takes an object of class iCellR and finds optimal number of PCs for clustering.

Usage

opt.pcs.plot(x = NULL, pcs.in.plot = 50)

Arguments

x                     An object of class iCellR.
pcs.in.plot           Number of PCs to show in plot, default = 50.

Value

An object of class iCellR.
 prep.vdj

Prep.vdj

Prepare VDJ data

Description
This function takes a data frame of VDJ data per cell and prepares it to adds it to the iCellR object.

Usage
prep.vdj(vdj.data = "data.frame", cond.name = "NULL")

Arguments
vdj.data A data frame containing vdj information.
cond.name Conditions.

Value
An object of class iCellR

pseudotime

Pseudotime

Description
This function takes an object of class iCellR and marker genes for clusters and performs pseudotime analysis.

Usage
pseudotime(x = NULL, marker.genes = "NULL", dims = 1:10)

Arguments
x An object of class iCellR.
marker.genes A list of marker genes for clusters.
dims PC dimensions to be used, default = 1:10.

Value
An object of class iCellR.
pseudotime.knetl  
iCellR KNN Network

Description

This function takes an object of class iCellR and runs kNet for dimensionality reduction.

Usage

pseudotime.knetl(
  x = NULL,
  dist.method = "euclidean",
  k = 5,
  abstract = TRUE,
  data.type = "pca",
  dims = 1:20,
  conds.to.plot = NULL,
  my.layout = "layout_with_fr",
  node.size = 10,
  cluster.membership = FALSE,
  interactive = TRUE,
  node.colors = NULL,
  edge.color = "gray",
  out.name = "Pseudotime.Abstract.KNetL",
  my.seed = 1
)

Arguments

x  An object of class iCellR.

dist.method  the distance measure to be used to compute the dissimilarity matrix. This must be one of: "euclidean", "maximum", "mandattattan", "canberra", "binary", "minkowski" or "NULL". By default, distance="euclidean". If the distance is "NULL", the dissimilarity matrix (diss) should be given by the user. If distance is not "NULL", the dissimilarity matrix should be "NULL".

k  KNN the higher the number the less sensitivity, default = 5.

abstract  Draw all the cells or clusters, , default = TRUE.

data.type  Choose between "tsne", "pca", "umap", default = "pca". We highly recommend PCA.

dims  PCA dimentions to be use for clustering, default = 1:20.

conds.to.plot  Choose the conditions you want to see in the plot, default = NULL (all conditions).

my.layout  Choose a layout, default = "layout_with_fr".

node.size  Size of the nodes, , default = 10.
pseudotime.tree

cluster.membership
Calculate memberships based on distance.

interactive
If set to TRUE an interactive HTML file will be created, default = TRUE.

node.colors
Color of the nodes, default = random colors.

e边.color
Color of the edges, default = "gray".

dout.name
If "interactive" is set to TRUE, the output name for HTML, default = "Abstract.KNetL".

my.seed
Seed number, default = 1.

Value

A plot.

pseudotime.tree  Pseudotime Tree

Description

This function takes an object of class iCellR and marker genes for clusters and performs pseudotime for differentiation or time course analysis.

Usage

pseudotime.tree(
  x = NULL,
  marker.genes = "NULL",
  clust.names = "NULL",
  dist.method = "euclidean",
  clust.method = "complete",
  label.offset = 0.5,
  type = "classic",
  hang = 1,
  cex = 1
)

Arguments

x
An object of class iCellR.

marker.genes
A list of marker genes for clusters.

clust.names
A list of names for clusters.

dist.method
Choose from "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski", default = "euclidean".

clust.method
Choose from "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median" or "centroid", default = "complete".
label.offset  Space between names and tree, default = 0.5.
type      Choose from "classic", "jitter", "unrooted", "fan", "cladogram", "radial", default = "classic".
hang      Hang, default = 1.
cex       Text size, default = 1.

Value

An object of class iCellR.

qc.stats

Calculate the number of UMIs and genes per cell and percentage of mitochondrial genes per cell and cell cycle genes.

Description

This function takes data frame and calculates the number of UMIs, genes per cell and percentage of mitochondrial genes per cell and cell cycle genes.

Usage

qc.stats(
x = NULL,
which.data = "raw.data",
mito.genes = NULL,
s.phase.genes = s.phase,
g2m.phase.genes = g2m.phase
)

Arguments

x A data frame containing gene counts for cells.
which.data Choose from "raw.data" or "main.data", "imputed.data", default = "raw.data".
mito.genes A character vector of mitochondrial genes names , default is the genes starting with mt.
s.phase.genes A character vector of gene names for S phase, default = s.phase.
g2m.phase.genes A character vector of gene names for G2 and M phase, default = g2m.phase.

Value

The data frame object
Description

R implementation of the PhenoGraph algorithm

Usage

Rphenograph(data, k = 30)

Arguments

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

Details

A simple 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 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.
membership 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_hierarchial 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.

Source

https://github.com/JinmiaoChenLab/Rphenograph

References


run.anchor Run anchor alignment on the main data.

Description

This function takes an object of class iCellR and runs anchor alignment. It’s a wrapper for Seurat.

Usage

run.anchor(
  x = NULL,
  method = "base.mean.rank",
  top.rank = 500,
  gene.list = "character",
  data.type = "main",
  normalization.method = "LogNormalize",
  scale.factor = 10000,
  margin = 1,
  block.size = NULL,
  selection.method = "vst",
  nfeatures = 2000,
  anchor.features = 2000,
  scale = TRUE,
  sct.clip.range = NULL,
  reduction = c("cca", "rpca"),
  l2.norm = TRUE,
  dims = 1:30,
  k.anchor = 5,
  k.filter = 200,
  k.score = 30,
  max.features = 200,
  nn.method = "rann",
  eps = 0,
  k.weight = 100
)
Arguments

x
An object of class iCellR.

method
Choose from "base.mean.rank" or "gene.model", default is "base.mean.rank". If gene.model is chosen you need to provide gene.list.

top.rank
A number taking the top genes ranked by base mean, default = 500.

gene.list
A character vector of genes to be used for PCA. If "clust.method" is set to "gene.model", default = "my_model_genes.txt".

data.type
Choose from "main" and "imputed", default = "main"

normalization.method
Choose from "LogNormalize", "CLR" and "RC". LogNormalize: Feature counts for each cell are divided by the total counts for that cell and multiplied by the scale.factor. This is then natural-log transformed using log1p. CLR: Applies a centered log ratio transformation. RC: Relative counts. Feature counts for each cell are divided by the total counts for that cell and multiplied by the scale.factor. No log-transformation is applied. For counts per million (CPM) set `scale.factor = 1e6`

scale.factor
Sets the scale factor for cell-level normalization.

margin
If performing CLR normalization, normalize across features (1) or cells (2)

block.size
How many cells should be run in each chunk, will try to split evenly across threads

selection.method
Choose from "vst","mean.var.plot (mvp)","dispersion (disp)".

nfeatures
Number of features to select as top variable features; only used when 'selection.method' is set to "dispersion" or "vst"

anchor.features
A numeric value. This will call ‘SelectIntegrationFeatures’ to select the provided number of features to be used in anchor finding

scale
Whether or not to scale the features provided. Only set to FALSE if you have previously scaled the features you want to use for each object in the object.list

sct.clip.range
Numeric of length two specifying the min and max values the Pearson residual will be clipped to

reduction
cca: Canonical correlation analysis. r pca: Reciprocal PCA

dims
Perform L2 normalization on the CCA cell embeddings after dimensional reduction

k.anchor
How many neighbors (k) to use when picking anchors

k.filter
How many neighbors (k) to use when filtering anchors

k.score
How many neighbors (k) to use when scoring anchors

max.features
The maximum number of features to use when specifying the neighborhood search space in the anchor filtering

nn.method
Method for nearest neighbor finding. Options include: rann, annoy

eps
Error bound on the neighbor finding algorithm (from RANN)

k.weight
Number of neighbors to consider when weighting
run.cca

Value
An object of class iCellR.

Description
This function takes an object of class iCellR and runs CCA using Seurat.

Usage
run.cca(
  x = NULL,
  top.vari.genes = 1000,
  cc.number = 30,
  dims.align = 1:20,
  normalize.data = TRUE,
  scale.data = TRUE,
  normalization.method = "LogNormalize",
  scale.factor = 10000,
  display.progress = TRUE
)

Arguments

  x                        An object of class iCellR.
  top.vari.genes           Chose top genes to use for CCA, default = 1000.
  cc.number                Choose a number, default = 30.
  dims.align               Choose the CCA dimentions to align, default = 1:20.
  normalize.data           TRUE or FALSE, default = TRUE.
  scale.data               TRUE or FALSE, default = TRUE.
  normalization.method     Choose a method, default = "LogNormalize".
  scale.factor             Scaling factor, default = 10000.
  display.progress         Show progress, default = TRUE.

Value
An object of class iCellR.
Description

This function takes an object of class iCellR and finds optimal number of clusters and clusters the data.

Usage

```r
run.clustering(
  x = NULL,
  clust.method = "kmeans",
  dist.method = "euclidean",
  index.method = "silhouette",
  max.clust = 25,
  min.clust = 2,
  dims = 1:10
)
```

Arguments

- **x**: An object of class iCellR.
- **clust.method**: the cluster analysis method to be used. This should be one of: "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", "centroid", "kmeans".
- **dist.method**: the distance measure to be used to compute the dissimilarity matrix. This must be one of: "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski" or "NULL". By default, distance="euclidean". If the distance is "NULL", the dissimilarity matrix (diss) should be given by the user. If distance is not "NULL", the dissimilarity matrix should be "NULL".
- **index.method**: the index to be calculated. This should be one of: "kl", "ch", "hartigan", "ccc", "scott", "marriot", "trcovw", "tracew", "friedman", "rubin", "cindex", "db", "silhouette", "duda", "pseudot2", "beale", "ratkowsky", "ball", "ptbserial", "gap", "frey", "mcclain", "gamma", "gplus", "tau", "dunn", "hubert", "sdindex", "dindex", "sdbw", "all" (all indices except GAP, Gamma, Gplus and Tau), "alllong" (all indices with Gap, Gamma, Gplus and Tau included).
- **max.clust**: maximal number of clusters, between 2 and (number of objects - 1), greater or equal to min.clust.
- **min.clust**: minimum number of clusters, default = 2.
- **dims**: PCA dimentions to be use for clustering, default = 1:10.

Value

An object of class iCellR.
Description

This function takes an object of class iCellR and performs differential expression (DE) analysis for clusters and conditions.

Usage

```r
run.diff.exp(
  x = NULL,
  data.type = "main",
  pval.test = "t.test",
  p.adjust.method = "hochberg",
  de.by = "clusters",
  cond.1 = "array",
  cond.2 = "array",
  base.cond = 0
)
```

Arguments

- `x`: An object of class iCellR.
- `data.type`: Choose from "main" and "imputed", default = "main".
- `pval.test`: Choose from "t.test", "wilcox.test", default = "t.test".
- `de.by`: Choose from "clusters", "conditions", "clustBase.condComp" or "condBase.clustComp".
- `cond.1`: First condition to do DE analysis on.
- `cond.2`: Second condition to do DE analysis on.
- `base.cond`: A base condition or cluster if de.by is either cond.clust or clust.cond

Value

An object of class iCellR
run.diffusion.map

Run diffusion map on PCA data (PHATE - Potential of Heat-Diffusion for Affinity-Based Transition Embedding)

Description

This function takes an object of class iCellR and runs diffusion map on PCA data.

Usage

run.diffusion.map(
  x = NULL,
  dims = 1:10,
  method = "destiny",
  ndim = 3,
  k = 5,
  alpha = 40,
  n.landmark = 2000,
  gamma = 1,
  t = "auto",
  knn.dist.method = "euclidean",
  init = NULL,
  mds.method = "metric",
  mds.dist.method = "euclidean",
  t.max = 100,
  npca = 100,
  plot.optimal.t = FALSE,
  verbose = 1,
  n.jobs = 1,
  seed = NULL,
  potential.method = NULL,
  use.alpha = NULL,
  n.svd = NULL,
  pca.method = NULL,
  g.kernel = NULL,
  diff.op = NULL,
  landmark.transitions = NULL,
  diff.op.t = NULL,
  dist.method = NULL
)

Arguments

- **x**: An object of class iCellR.
- **dims**: PC dimensions to be used for UMAP analysis.
- **method**: Diffusion map method, default = "phate".
- `ndim` int, optional, default: 2 number of dimensions in which the data will be embedded
- `k` int, optional, default: 5 number of nearest neighbors on which to build kernel
- `alpha` int, optional, default: 40 sets decay rate of kernel tails. If NULL, alpha decaying kernel is not used
- `n_landmark` int, optional, default: 2000 number of landmarks to use in fast PHATE
- `gamma` float, optional, default: 1 Informational distance constant between -1 and 1. gamma=1 gives the PHATE log potential, gamma=0 gives a square root potential.
- `t` int, optional, default: 'auto' power to which the diffusion operator is powered sets the level of diffusion
- `knn.dist.method` string, optional, default: 'euclidean'. recommended values: 'euclidean', 'cosine', 'precomputed' Any metric from scipy.spatial.distance can be used distance metric for building kNN graph. If 'precomputed', data should be an n_samples x n_samples distance or affinity matrix. Distance matrices are assumed to have zeros down the diagonal, while affinity matrices are assumed to have non-zero values down the diagonal. This is detected automatically using data[0,0]. You can override this detection with knn.dist.method='precomputed_distance' or knn.dist.method='precomputed_affinity'.
- `init` phate object, optional object to use for initialization. Avoids recomputing intermediate steps if parameters are the same.
- `mds.method` string, optional, default: 'metric' choose from 'classic', 'metric', and 'non-metric' which MDS algorithm is used for dimensionality reduction
- `mds.dist.method` string, optional, default: 'euclidean' recommended values: 'euclidean' and 'cosine'
- `t.max` int, optional, default: 100. Maximum value of t to test for automatic t selection.
- `npca` int, optional, default: 100 Number of principal components to use for calculating neighborhoods. For extremely large datasets, using n_pca < 20 allows neighborhoods to be calculated in log(n_samples) time.
- `plot.optimal.t` boolean, optional, if TRUE, produce a plot showing the Von Neumann Entropy curve for automatic t selection.
- `verbose` int or boolean, optional (default : 1) If TRUE or > 0, message verbose updates.
- `n.jobs` int, optional (default: 1) The number of jobs to use for the computation. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all, which is useful for debugging. For n_jobs below -1, (n.jobs + 1 + n.jobs) are used. Thus for n_jobs = -2, all CPUs but one are used
- `seed` int or NULL, random state (default: NULL)
- `potential.method` Deprecated. For log potential, use gamma=1. For sqrt potential, use gamma=0.
- `use.alpha` Deprecated To disable alpha decay, use alpha=NULL
- `n.svd` Deprecated.
run.impute

Value

An object of class iCellR.

Description

This function takes an object of class iCellR and runs imputation on the main data.

Usage

run.impute(
  x = NULL,
  imp.method = "iCellR.imp",
  dims = 1:10,
  nn = 10,
  ATAC.data = FALSE,
  rounding.digits = 4,
  round.num = TRUE,
  data.type = "pca",
  genes = "all_genes",
  k = 10,
  alpha = 15,
  t = "auto",
  npca = 100,
  init = NULL,
  t.max = 20,
  knn.dist.method = "euclidean",
  verbose = 1,
  n.jobs = 1,
  seed = NULL
)

Arguments

x
An object of class iCellR.

imp.method
Choose between "iCellR.imp" and "magic", default = "iCellR.imp".

dims
PC dimentions to be used for the analysis, default = 10.

nn
Number of neighboring cells to find, default = 10.

ATAC.data
If TRUE, it would normalize ATAC-Seq data and not RNA-Seq, default = FALSE.

rounding.digits
integer indicating the number of decimal places (round) or significant digits (signif) to be used.

round.num
Rounding of Numbers, default = FALSE.

data.type
Choose between "tsne", "pca", "umap", "diffusion", "knetl", default = "pca".

genes
character or integer vector, default: NULL vector of column names or column indices for which to return smoothed data. If 'all_genes' or NULL, the entire smoothed matrix is returned.

k
if imp.method is magic; int, optional, default: 10 number of nearest neighbors on which to build kernel.

alpha
if imp.method is magic; int, optional, default: 15 sets decay rate of kernel tails. If NULL, alpha decaying kernel is not used.

t
if imp.method is magic; int, optional, default: 'auto' power to which the diffusion operator is powered sets the level of diffusion. If 'auto', t is selected according to the Procrustes disparity of the diffused data.'

npca
number of PCA components that should be used; default: 100.

init
magic object, optional object to use for initialization. Avoids recomputing intermediate steps if parameters are the same.

t.max
if imp.method is magic; int, optional, default: 20 Maximum value of t to test for automatic t selection.

knn.dist.method
string, optional, default: 'euclidean'. recommended values: 'euclidean', 'cosine' Any metric from 'scipy.spatial.distance' can be used distance metric for building kNN graph.

verbose
'int' or 'boolean', optional (default : 1) If 'TRUE' or '> 0', message verbose updates.

n.jobs
'int', optional (default: 1) The number of jobs to use for the computation. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all, which is useful for debugging. For n_jobs below -1, (n cpus + 1 + n.jobs) are used. Thus for n_jobs = -2, all CPUs but one are used.

seed
int or 'NULL', random state (default: 'NULL')

Value

An object of class iCellR.
run.knetl

### Description

This function takes an object of class iCellR and and runs kNet for dimensionality reduction.

### Usage

```r
run.knetl(
  x = NULL,
  dist.method = "euclidean",
  zoom = 300,
  data.type = "pca",
 _dims = 1:20,
  joint = FALSE,
  col.by = "clusters",
  my.seed = 1,
  layout.2d = "layout_nicely",
  layout.3d = "layout_with_fr",
  add.3d = FALSE,
  dim.redux = "umap",
  do.redux = TRUE,
  run.iclust = FALSE,
  return.graph = FALSE
)
```

### Arguments

- **x**: An object of class iCellR.
- **dist.method**: the distance measure to be used to compute the dissimilarity matrix. This must be one of: "euclidean", "maximum", "mandattan", "canberra", "binary", "minkowski" or "NULL". By default, distance="euclidean". If the distance is "NULL", the dissimilarity matrix (diss) should be given by the user. If distance is not "NULL", the dissimilarity matrix should be "NULL".
- **zoom**: Adjusting zoom the higher the number the less sensitivity, default = 400.
- **data.type**: Choose between "tsne", "pca", "umap", default = "pca".
- **dims**: PCA dimentions to be use for clustering. default = 1:20.
- **joint**: Run in Combined or joint fashion as in CCCA and CPCA, default = FALSE.
- **col.by**: If return.graph is TRUE the choose the cluster colors. Choose between "clusters", "conditions".
- **my.seed**: seed number, default = 1.
- **layout.2d**: Choose your 2D layout, default = "layout_nicely".
- **layout.3d**: Choose your 3D layout, default = "layout_with_fr".
Add 3D KNNetL as well, default = FALSE.

Choose between "tsne", "pca", "umap" to unpack the nodes, default = "umap".

Perform dim redux for unpacking the nodes, default = TRUE.

Perform clustering as well (not recommended), default = FALSE.

return igraph object, default = FALSE.

Value

An object of class iCellR.

run.mnn

Run MNN alignment on the main data.

Description

This function takes an object of class iCellR and runs MNN alignment. It’s a wrapper for scran.

Usage

run.mnn(
  x = NULL,
  method = "base.mean.rank",
  top.rank = 500,
  gene.list = "character",
  data.type = "main",
  k = 20,
  cos.norm = TRUE,
  ndist = 3,
  d = 50,
  approximate = FALSE,
  irlba.args = list(),
  subset.row = NULL,
  auto.order = FALSE,
  pc.input = FALSE,
  compute.variances = FALSE,
  assay.type = "logcounts",
  get.spikes = FALSE,
  BNPARAM = NULL,
  BPPARAM = SerialParam()
)

Arguments

x An object of class iCellR.

method Choose from "base.mean.rank" or "gene.model", default is "base.mean.rank". If gene.model is chosen you need to provide gene.list.
top.rank A number taking the top genes ranked by base mean, default = 500.
gene.list A character vector of genes to be used for PCA. If "clust.method" is set to "gene.model", default = "my_model_genes.txt".
data.type Choose from "main" and "imputed", default = "main"
k An integer scalar specifying the number of nearest neighbors to consider when identifying MNNs.
cos.norm A logical scalar indicating whether cosine normalization should be performed on the input data prior to calculating distances between cells.
ndist A numeric scalar specifying the threshold beyond which neighbours are to be ignored when computing correction vectors. Each threshold is defined in terms of the number of median distances.
d Number of dimensions to pass to ‘multiBatchPCA’.
approximate Further arguments to pass to ‘multiBatchPCA’. Setting ‘approximate=TRUE’ is recommended for large data sets with many cells.
irlba.args Further arguments to pass to ‘multiBatchPCA’. Setting ‘approximate=TRUE’ is recommended for large data sets with many cells.
subset.row See ‘?'scran-gene-selection'’.
auto.order Logical scalar indicating whether re-ordering of batches should be performed to maximize the number of MNN pairs at each step. Alternatively an integer vector containing a permutation of ‘1:N’ where ‘N’ is the number of batches.
pc.input Logical scalar indicating whether the values in ‘...’ are already low-dimensional, e.g., the output of ‘multiBatchPCA’.
compute.variances Logical scalar indicating whether the percentage of variance lost due to non-orthogonality should be computed.
assay.type A string or integer scalar specifying the assay containing the expression values, if SingleCellExperiment objects are present in ‘...’.
get.spikes See ‘?’scran-gene-selection'’. Only relevant if ‘...’ contains SingleCellExperiment objects.
BNPARAM A BiocNeighborParam object specifying the nearest neighbor algorithm. Defaults to an exact algorithm if ‘NULL’, see ‘?findKNN’ for more details.
BPPARAM A BiocParallelParam object specifying whether the PCA and nearest-neighbor searches should be parallelized.

Value
An object of class iCellR.
run.pc.tsne  

Run tSNE on PCA Data. Barnes-Hut implementation of t-Distributed Stochastic Neighbor Embedding

Description

This function takes an object of class iCellR and runs tSNE on PCA data. Wrapper for the C++ implementation of Barnes-Hut t-Distributed Stochastic Neighbor Embedding. t-SNE is a method for constructing a low dimensional embedding of high-dimensional data, distances or similarities. Exact t-SNE can be computed by setting theta=0.0.

Usage

run.pc.tsne(
  x = NULL,
  dims = 1:10,
  my.seed = 0,
  add.3d = TRUE,
  initial_dims = 50,
  perplexity = 30,
  theta = 0.5,
  check_duplicates = FALSE,
  pca = TRUE,
  max_iter = 1000,
  verbose = FALSE,
  is_distance = FALSE,
  Y_init = NULL,
  pca_center = TRUE,
  pca_scale = FALSE,
  stop_lying_iter = ifelse(is.null(Y_init), 250L, 0L),
  mom_switch_iter = ifelse(is.null(Y_init), 250L, 0L),
  momentum = 0.5,
  final_momentum = 0.8,
  eta = 200,
  exaggeration_factor = 12
)

Arguments

x  
An object of class iCellR.

dims  
PC dimensions to use for tSNE analysis.

my.seed  
seed number, default = 0.

add.3d  
Add 3D tSNE as well, default = TRUE.

initial_dims  
integer; the number of dimensions that should be retained in the initial PCA step (default: 50)
run.pca

perplexity numeric; Perplexity parameter
theta numeric; Speed/accuracy trade-off (increase for less accuracy), set to 0.0 for exact TSNE (default: 0.5)
check_duplicates logical; Checks whether duplicates are present. It is best to make sure there are no duplicates present and set this option to FALSE, especially for large datasets (default: TRUE)
pca logical; Whether an initial PCA step should be performed (default: TRUE)
max_iter integer; Number of iterations (default: 1000)
verbose logical; Whether progress updates should be messageed (default: FALSE)
is_distance logical; Indicate whether X is a distance matrix (experimental, default: FALSE)
Y_init matrix; Initial locations of the objects. If NULL, random initialization will be used (default: NULL). Note that when using this, the initial stage with exaggerated perplexity values and a larger momentum term will be skipped.
pca_center logical; Should data be centered before pca is applied? (default: TRUE)
pca_scale logical; Should data be scaled before pca is applied? (default: FALSE)
stop_lying_iter integer; Iteration after which the perplexities are no longer exaggerated (default: 250, except when Y_init is used, then 0)
mom_switch_iter integer; Iteration after which the final momentum is used (default: 250, except when Y_init is used, then 0)
momentum numeric; Momentum used in the first part of the optimization (default: 0.5)
final_momentum numeric; Momentum used in the final part of the optimization (default: 0.8)
eta numeric; Learning rate (default: 200.0)
exaggeration_factor numeric; Exaggeration factor used to multiply the P matrix in the first part of the optimization (default: 12.0)

Value

An object of class iCellR.

---

**Description**

This function takes an object of class iCellR and runs PCA on the main data.
run.phenograph

Usage

run.pca(
  x = NULL,
  data.type = "main",
  method = "base.mean.rank",
  top.rank = 500,
  plus.log.value = 0.1,
  scale.data = TRUE,
  gene.list = "character"
)

Arguments

x An object of class iCellR.
data.type Choose from "main" and "imputed", default = "main"
method Choose from "base.mean.rank" or "gene.model", default is "base.mean.rank". If
gene.model is chosen you need to provide gene.list.
top.rank A number. Taking the top genes ranked by base mean, default = 500.
plus.log.value A number to add to each value in the matrix before log transformasion to aviond
Inf numbers, default = 0.1.
scale.data If TRUE the data will be scaled (log2 + plus.log.value), default = TRUE.
gene.list A charactor vector of genes to be used for PCA. If "clust.method" is set to
"gene.model", default = "my_model_genes.txt".

Value

An object of class iCellR.

run.phenograph

Clustering the data

Description

This function takes an object of class iCellR and finds optimal number of clusters and clusters the
data.

Usage

run.phenograph(x = NULL, k = 100, data.type = "pca", dims = 1:10)

Arguments

x An object of class iCellR.
k integer; number of nearest neighbours (default:45)
data.type Choose between "tsne", "pca", "umap", default = "pca".
dims PCA dimentions to be use for clustering, default = 1:10.
**run.tsne**

**Value**

An object of class iCellR.

---

**run.tsne**  
*Run tSNE on the Main Data. Barnes-Hut implementation of t-Distributed Stochastic Neighbor Embedding*

**Description**

This function takes an object of class iCellR and runs tSNE on main data. Wrapper for the C++ implementation of Barnes-Hut t-Distributed Stochastic Neighbor Embedding. t-SNE is a method for constructing a low dimensional embedding of high-dimensional data, distances or similarities. Exact t-SNE can be computed by setting theta=0.0.

**Usage**

```r
run.tsne(
  x = NULL,
  clust.method = "base.mean.rank",
  top.rank = 500,
  gene.list = "character",
  add.3d = TRUE,
  initial_dims = 50,
  perplexity = 30,
  theta = 0.5,
  check_duplicates = TRUE,
  pca = TRUE,
  max_iter = 1000,
  verbose = FALSE,
  is_distance = FALSE,
  Y_init = NULL,
  pca_center = TRUE,
  pca_scale = FALSE,
  stop_lying_iter = ifelse(is.null(Y_init), 250L, 0L),
  mom_switch_iter = ifelse(is.null(Y_init), 250L, 0L),
  momentum = 0.5,
  final_momentum = 0.8,
  eta = 200,
  exaggeration_factor = 12
)
```

**Arguments**

- **x**  
  An object of class iCellR.
- **clust.method**  
  Choose from "base.mean.rank" or "gene.model", default is "base.mean.rank".
- **top.rank**  
  A number taking the top genes ranked by base mean, default = 500.
run.tsne

gene.list A list of genes to be used for tSNE analysis. If "clust.method" is set to "gene.model",
default = "my_model_genes.txt".
add.3d Add 3D tSNE as well, default = TRUE.
initial_dims integer; the number of dimensions that should be retained in the initial PCA step
(default: 50)
perplexity numeric; Perplexity parameter
theta numeric; Speed/accuracy trade-off (increase for less accuracy), set to 0.0 for
exact TSNE (default: 0.5)
check_duplicates logical; Checks whether duplicates are present. It is best to make sure there are
no duplicates present and set this option to FALSE, especially for large datasets
(default: TRUE)
pca logical; Whether an initial PCA step should be performed (default: TRUE)
max_iter integer; Number of iterations (default: 1000)
verbose logical; Whether progress updates should be messageed (default: FALSE)
is_distance logical; Indicate whether X is a distance matrix (experimental, default: FALSE)
Y_init matrix; Initial locations of the objects. If NULL, random initialization will be
used (default: NULL). Note that when using this, the initial stage with exagger-
ated perplexity values and a larger momentum term will be skipped.
pca_center logical; Should data be centered before pca is applied? (default: TRUE)
pca_scale logical; Should data be scaled before pca is applied? (default: FALSE)
stop_lying_iter integer; Iteration after which the perplexities are no longer exaggerated (default:
250, except when Y_init is used, then 0)
mom_switch_iter integer; Iteration after which the final momentum is used (default: 250, except
when Y_init is used, then 0)
momentum numeric; Momentum used in the first part of the optimization (default: 0.5)
final_momentum numeric; Momentum used in the final part of the optimization (default: 0.8)
etta numeric; Learning rate (default: 200.0)
exaggeration_factor numeric; Exaggeration factor used to multiply the P matrix in the first part of
the optimization (default: 12.0)

Value
An object of class iCellR.
run.umap

**Run UMAP on PCA Data (Computes a manifold approximation and projection)**

**Description**

This function takes an object of class iCellR and runs UMAP on PCA data.

**Usage**

```r
run.umap(
  x = NULL,
  my.seed = 0,
  dims = 1:10,
  n_neighbors = 15,
  n_components = 2,
  metric = "euclidean",
  n_epochs = NULL,
  learning_rate = 1,
  scale = FALSE,
  init = "spectral",
  init_sdev = NULL,
  spread = 1,
  min_dist = 0.01,
  set_op_mix_ratio = 1,
  local_connectivity = 1,
  bandwidth = 1,
  repulsion_strength = 1,
  negative_sample_rate = 5,
  a = NULL,
  b = NULL,
  nn_method = NULL,
  n_trees = 50,
  search_k = 2 * n_neighbors * n_trees,
  approx_pow = FALSE,
  y = NULL,
  target_n_neighbors = n_neighbors,
  target_metric = "euclidean",
  target_weight = 0.5,
  pca = NULL,
  pca_center = TRUE,
  pcg_rand = TRUE,
  fast_sgd = FALSE,
  ret_model = FALSE,
  ret_nn = FALSE,
  n_threads = 1,
  n_sgd_threads = 0,
)```
grain_size = 1,
    tmpdir = tempdir(),
    verbose = getOption("verbose", TRUE)
)

Arguments

x       An object of class iCellR.
my.seed  seed number, default = 0.
dims     PC dimensions to be used for UMAP analysis.
n_neighbors  The size of local neighborhood (in terms of number of neighboring sample points) used for manifold approximation. Larger values result in more global views of the manifold, while smaller values result in more local data being preserved. In general values should be in the range 2 to 100.
n_components  The dimension of the space to embed into. This defaults to 2 to provide easy visualization, but can reasonably be set to any integer value in the range 2 to 100.
metric     Type of distance metric to use to find nearest neighbors. One of:

    • "euclidean" (the default)
    • "cosine"
    • "manhattan"
    • "hamming"
    • "categorical" (see below)

Only applies if nn_method = "annoy" (for nn_method = "fnn", the distance metric is always "euclidean").

If X is a data frame or matrix, then multiple metrics can be specified, by passing a list to this argument, where the name of each item in the list is one of the metric names above. The value of each list item should be a vector giving the names or integer ids of the columns to be included in a calculation, e.g. metric = list(euclidean = 1:4, manhattan = 5:10).

Each metric calculation results in a separate fuzzy simplicial set, which are intersected together to produce the final set. Metric names can be repeated. Because non-numeric columns are removed from the data frame, it is safer to use column names than integer ids.

Factor columns can also be used by specifying the metric name "categorical". Factor columns are treated different from numeric columns and although multiple factor columns can be specified in a vector, each factor column specified is processed individually. If you specify a non-factor column, it will be coerced to a factor.

For a given data block, you may override the pca and pca_center arguments for that block, by providing a list with one unnamed item containing the column names or ids, and then any of the pca or pca_center overrides as named items, e.g. metric = list(euclidean = 1:4, manhattan = list(5:10, pca_center = FALSE)). This exists to allow mixed binary and real-valued data to be included and to have PCA applied to both, but with centering applied only to
the real-valued data (it is typical not to apply centering to binary data before PCA is applied).

n_epochs Number of epochs to use during the optimization of the embedded coordinates. By default, this value is set to 500 for datasets containing 10,000 vertices or less, and 200 otherwise.

learning_rate Initial learning rate used in optimization of the coordinates.

scale Scaling to apply to X if it is a data frame or matrix:
- "none" or FALSE or NULL: No scaling.
- "Z" or "scale" or TRUE: Scale each column to zero mean and variance 1.
- "maxabs": Center each column to mean 0, then divide each element by the maximum absolute value over the entire matrix.
- "range": Range scale the entire matrix, so the smallest element is 0 and the largest is 1.
- "colrange": Scale each column in the range (0,1).

For UMAP, the default is "none".

init Type of initialization for the coordinates. Options are:
- "spectral": Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, with Gaussian noise added.
- "normlaplacian": Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, without noise.
- "random": Coordinates assigned using a uniform random distribution between -10 and 10.
- "lvrandom": Coordinates assigned using a Gaussian distribution with standard deviation 1e-4, as used in LargeVis (Tang et al., 2016) and t-SNE.
- "laplacian": Spectral embedding using the Laplacian Eigenmap (Belkin and Niyogi, 2002).
- "pca": The first two principal components from PCA of X if X is a data frame, and from a 2-dimensional classical MDS if X is of class "dist".
- "spca": Like "pca", but each dimension is then scaled so the standard deviation is 1e-4, to give a distribution similar to that used in t-SNE. This is an alias for init = "pca", init_sdev = 1e-4.
- "agspectral": An "approximate global" modification of "spectral" which all edges in the graph to a value of 1, and then sets a random number of edges (negative_sample_rate edges per vertex) to 0.1, to approximate the effect of non-local affinities.
- A matrix of initial coordinates.

For spectral initializations, ("spectral", "normlaplacian", "laplacian"), if more than one connected component is identified, each connected component is initialized separately and the results are merged. If verbose = TRUE the number of connected components are logged to the console. The existence of multiple connected components implies that a global view of the data cannot be attained with this initialization. Either a PCA-based initialization or increasing the value of n_neighbors may be more appropriate.
**init_sdev**
If non-NULL, scales each dimension of the initialized coordinates (including any user-supplied matrix) to this standard deviation. By default no scaling is carried out, except when init = "spca", in which case the value is 0.0001. Scaling the input may help if the unscaled versions result in initial coordinates with large inter-point distances or outliers. This usually results in small gradients during optimization and very little progress being made to the layout. Shrinking the initial embedding by rescaling can help under these circumstances. Scaling the result of init = "pca" is usually recommended and init = "spca" as an alias for init = "pca", init_sdev = 1e-4 but for the spectral initializations the scaled versions usually aren’t necessary unless you are using a large value of n_neighbors (e.g. n_neighbors = 150 or higher).

**spread**
The effective scale of embedded points. In combination with min_dist, this determines how clustered/clumped the embedded points are.

**min_dist**
The effective minimum distance between embedded points. Smaller values will result in a more clustered/clumped embedding where nearby points on the manifold are drawn closer together, while larger values will result on a more even dispersal of points. The value should be set relative to the spread value, which determines the scale at which embedded points will be spread out.

**set_op_mix_ratio**
Interpolate between (fuzzy) union and intersection as the set operation used to combine local fuzzy simplicial sets to obtain a global fuzzy simplicial sets. Both fuzzy set operations use the product t-norm. The value of this parameter should be between 0.0 and 1.0; a value of 1.0 will use a pure fuzzy union, while 0.0 will use a pure fuzzy intersection.

**local_connectivity**
The local connectivity required – i.e. the number of nearest neighbors that should be assumed to be connected at a local level. The higher this value the more connected the manifold becomes locally. In practice this should be not more than the local intrinsic dimension of the manifold.

**bandwidth**
The effective bandwidth of the kernel if we view the algorithm as similar to Laplacian Eigenmaps. Larger values induce more connectivity and a more global view of the data, smaller values concentrate more locally.

**repulsion_strength**
Weighting applied to negative samples in low dimensional embedding optimization. Values higher than one will result in greater weight being given to negative samples.

**negative_sample_rate**
The number of negative edge/1-simplex samples to use per positive edge/1-simplex sample in optimizing the low dimensional embedding.

**a**
More specific parameters controlling the embedding. If NULL these values are set automatically as determined by min_dist and spread.

**b**
More specific parameters controlling the embedding. If NULL these values are set automatically as determined by min_dist and spread.

**nn_method**
Method for finding nearest neighbors. Options are:
- "fnn". Use exact nearest neighbors via the FNN package.
- "annoy" Use approximate nearest neighbors via the RcppAnnoy package.
By default, if \( X \) has less than 4,096 vertices, the exact nearest neighbors are found. Otherwise, approximate nearest neighbors are used. You may also pass precalculated nearest neighbor data to this argument. It must be a list consisting of two elements:

- "idx". A \( n_{\text{vertices}} \times n_{\text{neighbors}} \) matrix containing the integer indexes of the nearest neighbors in \( X \). Each vertex is considered to be its own nearest neighbor, i.e. \( \text{idx}[,1] == 1:n_{\text{vertices}} \).
- "dist". A \( n_{\text{vertices}} \times n_{\text{neighbors}} \) matrix containing the distances of the nearest neighbors.

Multiple nearest neighbor data (e.g. from two different precomputed metrics) can be passed by passing a list containing the nearest neighbor data lists as items. The \( n_{\text{neighbors}} \) parameter is ignored when using precomputed nearest neighbor data.

\( n_{\text{trees}} \) Number of trees to build when constructing the nearest neighbor index. The more trees specified, the larger the index, but the better the results. With \( \text{search}_k \), determines the accuracy of the Annoy nearest neighbor search. Only used if the \( \text{nn}_\text{method} \) is "annoy". Sensible values are between 10 to 100.

\( \text{search}_k \) Number of nodes to search during the neighbor retrieval. The larger \( k \), the more accurate results, but the longer the search takes. With \( n_{\text{trees}} \), determines the accuracy of the Annoy nearest neighbor search. Only used if the \( \text{nn}_\text{method} \) is "annoy".

\( \text{approx}_\text{pow} \) If TRUE, use an approximation to the power function in the UMAP gradient, from https://martin.ankerl.com/2012/01/25/optimized-approximative-pow-in-c-and-cpp/.

\( y \) Optional target data for supervised dimension reduction. Can be a vector, matrix or data frame. Use the \( \text{target}_\text{metric} \) parameter to specify the metrics to use, using the same syntax as \( \text{metric} \). Usually either a single numeric or factor column is used, but more complex formats are possible. The following types are allowed:

- Factor columns with the same length as \( X \). \( \text{NA} \) is allowed for any observation with an unknown level, in which case UMAP operates as a form of semi-supervised learning. Each column is treated separately.
- Numeric data. \( \text{NA} \) is not allowed in this case. Use the parameter \( \text{target}_\text{n_neighbors} \) to set the number of neighbors used with \( y \). If unset, \( n_{\text{neighbors}} \) is used. Unlike factors, numeric columns are grouped into one block unless \( \text{target}_\text{metric} \) specifies otherwise. For example, if you wish columns \( a \) and \( b \) to be treated separately, specify \( \text{target}_\text{metric} = \text{list(euclidean} = "a",euclidean = "b") \). Otherwise, the data will be effectively treated as a matrix with two columns.
- Nearest neighbor data, consisting of a list of two matrices, idx and dist. These represent the precalculated nearest neighbor indices and distances, respectively. This is the same format as that expected for precalculated data in \( \text{nn}_\text{method} \). This format assumes that the underlying data was a numeric vector. Any user-supplied value of the \( \text{target}_\text{n_neighbors} \) parameter is ignored in this case, because the the number of columns in the matrices is used for the value. Multiple nearest neighbor data using different metrics can be supplied by passing a list of these lists.
Unlike X, all factor columns included in y are automatically used.

**target_n_neighbors**

Number of nearest neighbors to use to construct the target simplicial set. Default value is n_neighbors. Applies only if y is non-NULL and numeric.

**target_metric**

The metric used to measure distance for y if using supervised dimension reduction. Used only if y is numeric.

**target_weight**

Weighting factor between data topology and target topology. A value of 0.0 weights entirely on data, a value of 1.0 weights entirely on target. The default of 0.5 balances the weighting equally between data and target. Only applies if y is non-NULL.

**pca**

If set to a positive integer value, reduce data to this number of columns using PCA. Doesn’t apply if the distance metric is “hamming”, or the dimensions of the data is larger than the number specified (i.e. number of rows and columns must be larger than the value of this parameter). If you have $>100$ columns in a data frame or matrix, reducing the number of columns in this way may substantially increase the performance of the nearest neighbor search at the cost of a potential decrease in accuracy. In many t-SNE applications, a value of 50 is recommended, although there’s no guarantee that this is appropriate for all settings.

**pca_center**

If TRUE, center the columns of X before carrying out PCA. For binary data, it’s recommended to set this to FALSE.

**pcg_rand**

If TRUE, use the PCG random number generator (O’Neill, 2014) during optimization. Otherwise, use the faster (but probably less statistically good) Tausworthe “taus88” generator. The default is TRUE.

**fast_sgd**

If TRUE, then the following combination of parameters is set: pcg_rand = TRUE, n_sgd_threads = “auto” and approx_pow = TRUE. The default is FALSE. Setting this to TRUE will speed up the stochastic optimization phase, but give a potentially less accurate embedding, and which will not be exactly reproducible even with a fixed seed. For visualization, fast_sgd = TRUE will give perfectly good results. For more generic dimensionality reduction, it’s safer to leave fast_sgd = FALSE. If fast_sgd = TRUE, then user-supplied values of pcg_rand, n_sgd_threads, and approx_pow are ignored.

**ret_model**

If TRUE, then return extra data that can be used to add new data to an existing embedding via `umap_transform`. The embedded coordinates are returned as the list item `embedding`. If FALSE, just return the coordinates. This parameter can be used in conjunction with ret_nn. Note that some settings are incompatible with the production of a UMAP model: external neighbor data (passed via a list to `nn_method`), and factor columns that were included via the metric parameter. In the latter case, the model produced is based only on the numeric data. A transformation using new data is possible, but the factor columns in the new data are ignored.

**ret_nn**

If TRUE, then in addition to the embedding, also return nearest neighbor data that can be used as input to `nn_method` to avoid the overhead of repeatedly calculating the nearest neighbors when manipulating unrelated parameters (e.g. `min_dist`, `n_epochs`, `init`). See the “Value” section for the names of the list items. If FALSE, just return the coordinates. Note that the nearest neighbors
could be sensitive to data scaling, so be wary of reusing nearest neighbor data if modifying the scale parameter. This parameter can be used in conjunction with ret_model.

**n_threads**  
Number of threads to use.

**n_sgd_threads**  
Number of threads to use during stochastic gradient descent. If set to > 1, then results will not be reproducible, even if 'set.seed' is called with a fixed seed before running. Set to "auto" go use the same value as n_threads.

**grain_size**  
Minimum batch size for multithreading. If the number of items to process in a thread falls below this number, then no threads will be used. Used in conjunction with n_threads and n_sgd_threads.

**tmpdir**  
Temporary directory to store nearest neighbor indexes during nearest neighbor search. Default is tempdir. The index is only written to disk if n_threads > 1 and nn_method = "annoy"; otherwise, this parameter is ignored.

**verbose**  
If TRUE, log details to the console.

Value

An object of class iCellR.

---

**s.phase**  
**A dataset of S phase genes**

Description

A dataset containing the genes for S phase

Usage

s.phase

Format

A character with 43 genes

Source

https://www.science.org/doi/abs/10.1126/science.aad0501
spatial.plot  

Plot nGenes, UMIs and percentage mito, genes, clusters and more on spatial image

Description

This function takes an object of class iCellR and creates spatial plots.

Usage

spatial.plot(
  x = NULL,
  cell.size = 1,
  cell.colors = c("gray", "red"),
  back.col = "black",
  col.by = "clusters",
  conds.to.plot = NULL,
  gene = NULL,
  data.type = "main",
  scaleValue = TRUE,
  min.scale = 0,
  max.scale = 2.5,
  anno.clust = FALSE,
  anno.size = 4,
  anno.col = "white",
  cell.transparency = 1,
  interactive = TRUE,
  out.name = "plot"
)

Arguments

x                         An object of class iCellR.
cell.size                 A numeric value for the size of the cells, default = 1.
cell.colors               Colors for heat mapping the points in "scatterplot", default = c("gray", "red").
back.col                  A color for the plot background, default = "black".
col.by                    Choose between "clusters", "mt", "UMIs", "nGenes", "cc" (cell cycle) or "gene". default = "clusters".
conds.to.plot             Choose the conditions you want to see in the plot, default = NULL (all conditions).
gene                      Gene name/names to be plotted, if col.by = "gene".
data.type                  Choose from "main" or "imputed", default = "main".
scaleValue                Scale the colors, default = FALSE.
min.scale                 If scaleValue = TRUE, set a number for min, default = -2.5.
`stats.plot`  

**max.scale**  
If `scaleValue = TRUE`, set a number for max, default = 2.5.

**anno.clust**  
Annotate cluster names on the plot, default = TRUE.

**anno.size**  
If `anno.clust` is TRUE set font size, default = 3.

**anno.col**  
If `anno.clust` is TRUE set color, default = "white".

**cell.transparency**  
Color transparency for points in "scatterplot" and "boxplot", default = 1.

**interactive**  
If set to TRUE an interactive HTML file will be created, default = TRUE.

**out.name**  
If "interactive" is set to TRUE, the output name for HTML, default = "plot".

**Value**

An object of class iCellR.

---

**stats.plot**  

*Plot nGenes, UMIs and percent mito*

**Description**

This function takes an object of class iCellR and creates QC plot.

**Usage**

```r
stats.plot(
  x = NULL,
  plot.type = "three.in.one",
  cell.color = "slategray3",
  cell.size = 1,
  cell.transparency = 0.5,
  box.color = "red",
  box.line.col = "green",
  back.col = "white",
  interactive = TRUE,
  out.name = "plot"
)
```

**Arguments**

- **x**  
An object of class iCellR.

- **plot.type**  
Choose from "box.umi", "box.mito", "box.gene", "box.s.phase", "box.g2m.phase", "all.in.one", "three.in.one", "point.mito.umi", "point.gene.umi".

- **cell.color**  
Choose a color for points in the plot.

- **cell.size**  
A number for the size of the points in the plot, default = 1.

- **cell.transparency**  
Color transparency for points in "scatterplot" and "boxplot", default = 0.5.
top.markers

box.color  A color for the boxes in the "boxplot", default = "red".
box.line.col  A color for the lines around the "boxplot", default = "green".
back.col  Background color, default = "white"
interactive  If set to TRUE an interactive HTML file will be created, default = TRUE.
out.name  If "interactive" is set to TRUE, the output name for HTML, default = "plot".

Value

An object of class iCellR.

Description

This function takes the marker genes info if chooses marker gene names for plots.

Usage

top.markers(
  x = NULL,
  topde = 10,
  min.base.mean = 0.2,
  filt.ambig = TRUE,
  cluster = 0
)

Arguments

x  An object of class iCellR.
topde  Number of top differentially expressed genes to be chosen from each cluster, default = 10.
min.base.mean  Minimum base mean of the genes to be chosen, default = 0.5.
filt.ambig  Filter markers that are seen for more than one cluster, default = TRUE.
cluster  Choose a cluster to find markers for. If 0, it would find markers for all clusters, default = 0.

Value

A set of gene names
vdj.stats  

**Description**

This function takes a data frame of VDJ info per cell and dose QC.

**Usage**

```r
vdj.stats(my.vdj = "data.frame")
```

**Arguments**

- **my.vdj**
  - A data frame containing VDJ data for cells.

**Value**

An object of class iCellR

volcano.ma.plot  

**Create MA and Volcano plots.**

**Description**

This function takes the result of differential expression (DE) analysis and provides MA and volcano plots.

**Usage**

```r
volcano.ma.plot(
  x = NULL,
  sig.value = "padj",
  sig.line = 0.1,
  plot.type = "volcano",
  x.limit = 2,
  y.limit = 2,
  limit.force = FALSE,
  scale.ax = TRUE,
  dot.size = 1.75,
  dot.transparency = 0.5,
  dot.col = c("#E64B35", ",#31B2bd", ",#636363"),
  interactive = TRUE,
  out.name = "plot"
)
```
Arguments

- **x**: A data frame containing differential expression (DE) analysis results.
- **sig.value**: Choose from "pval" or "padj", default = "padj".
- **sig.line**: A number to draw the line for the significant genes based on sig.value type, default = 0.1.
- **plot.type**: Choose from "ma" or "volcano", default = "volcano".
- **x.limit**: A number to set a limit for the x axis.
- **y.limit**: A number to set a limit for the y axis.
- **limit.force**: If set to TRUE the x.limit and y.limit will be forced, default = FALSE.
- **scale.ax**: If set to TRUE the y axis will be scaled to include all the points, default = TRUE.
- **dot.size**: A number for the size of the points in the plot, default = 1.75.
- **dot.transparency**: Color transparency for points in "scatterplot" and "boxplot", default = 0.5.
- **dot.col**: A set of three colors for the points in the volcano plot, default = c("#E64B35","#3182bd","#636363").
- **interactive**: If set to TRUE an interactive HTML file will be created, default = TRUE.
- **out.name**: If "interactive" is set to TRUE, the output name for HTML, default = "plot".

Value

Plots
Index

* datasets
  g2m.phase, 20
  s.phase, 63
  add.10x.image, 3
  add.adt, 4
  add.vdj, 5
  adt.rna.merge, 5
  capture.image.10x, 6
  cc, 6
  cell.cycle, 7
  cell.filter, 7
  cell.gating, 8
  cell.type.pred, 9
  change.clust, 10
  clono.plot, 10
  clust.avg.exp, 11
  clust.cond.info, 12
  clust.ord, 13
  clust.rm, 13
  clust.stats.plot, 14
  cluster.plot, 15
  data.aggregation, 16
  data.scale, 17
  down.sample, 18
  find.dim.genes, 18
  find.neighbors, 20
  findMarkers, 19
  g2m.phase, 20
  gate.to.clust, 21
  gene.plot, 21
  gene.stats, 23
  gg.cor, 23
  heatmap.gg.plot, 24
  hto.anno, 26
  i.score, 26
  iba, 27
  iclust, 28
  load.h5, 29
  load10x, 29
  make.bed, 30
  make.gene.model, 30
  make.obj, 32
  myImp, 32
  norm.adt, 33
  norm.data, 33
  opt.pcs.plot, 34
  prep.vdj, 35
  pseudotime, 35
  pseudotime.knetl, 36
  pseudotime.tree, 37
  qc.stats, 38
  Rphenograph, 39
  run.anchor, 40
  run.cca, 42
  run.clustering, 43
  run.diff.exp, 44
  run.diffusion.map, 45
  run.impute, 47
  run.knetl, 49
  run.mnn, 50
  run.pc.tsne, 52
  run.pca, 53
  run.phenograph, 54
  run.tsne, 55
  run.umap, 57
  s.phase, 63
  spatial.plot, 64
stats.plot, 65

tempdir, 63
top.markers, 66

umap_transform, 62

vdj.stats, 67
volcano.ma.plot, 67