Package 'iCellR'

July 23, 2019

Type Package

Title Analyzing High-Throughput Single Cell Sequencing Data

Version 1.0.0

Author Alireza Khodadadi-Jamayran,
Joseph Pucella,
Hua Zhou,
Nicole Doudican,
John Carucci,
Adriana Heguy,
Boris Reizis,
Aristotelis Tsirigos

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 and CITE-Seq. 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.

Depends R (>= 3.3.0), ggplot2, plotly

Imports Matrix, Rtsne, gridExtra, ggrepel, ggpubr, scatterplot3d, RColorBrewer, knitr, NbClust, shiny, umap, pheatmap, ape, ggdendro, plyr, reshape, Hmisc, htmlwidgets, methods

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

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

Suggests phateR, Rmagic, Seurat

NeedsCompilation no

R topics documented:

`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

```
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

Examples

```
## Not run:
# Read your ADT data (in this case in ADT.tsv file) and add to your object as below
my.adt.data <- read.table("ADT.tsv")

my.obj <- add.adt(my.obj, adt.data = my.adt.data)

head(my.obj@adt.raw)[1:5]


## End(Not run)
```

`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

Examples

```
## Not run:
Read your VDJ data (in this case in VDJ.tsv file) and add to your object as below

my.vdj.data <- read.table("VDJ.tsv")

VDJ <- prep.vdj(my.obj, adt.data = my.vdj.data)

head(VDJ)

my.obj <- add.vdj(my.obj, vdj.data = VDJ)

head(my.obj@vdj.data)

## End(Not run)
```

`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

Examples

```
## Not run:
my.obj <- adt.rna.merge(my.obj, adt.data = "raw")

## End(Not run)
```

cc *Calculate Cell cycle phase prediction*

Description

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

Usage

```
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

Examples

```
## Not run:
my.obj <- cc(my.obj, s.genes = s.phase, g2m.genes = g2m.phase)

## End(Not run)
```

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

```
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.

Examples

```
demo.obj <- cell.filter(demo.obj,
                        min.mito = 0,
                        max.mito = 0.05 ,
                        min.genes = 100,
                        max.genes = 2500,
                        min.umis = 0,
                        max.umis = Inf)

message(demo.obj@my.filters)
```

---

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 = NULL)
```

Arguments

x	An object of class iCellR.
my.plot	The plot to use for gating. Must be a 2D plot.
plot.type	Choose from UMAP and tSNE, default = NULL.

Value

An object of class iCellR.

Examples

```
## Not run:
cell.gating(my.obj, my.plot = PLOT, plot.type = "tsne")

## End(Not run)
```

---

cell.type.pred *Create heatmaps or dot plots for genes in clusters to find thier cell types using ImmGen data.*

Description

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

Usage

```
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 "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

Examples

```
## Not run:
imm.gen(immgen.data = "uli.rna", gene = MyGenes, plot.type = "heatmap")
imm.gen(immgen.data = "rna", gene = MyGenes, plot.type = "point.plot")

## End(Not run)
```

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.

Examples

```
demo.obj <- change.clust(demo.obj, change.clust = 1, to.clust = 3)
cluster.plot(demo.obj,plot.type = "umap",interactive = FALSE)

demo.obj <- change.clust(demo.obj, change.clust = 3, to.clust = "B Cell")
cluster.plot(demo.obj,plot.type = "umap",interactive = FALSE)

demo.obj <- change.clust(demo.obj, clust.reset = TRUE)
cluster.plot(demo.obj,plot.type = "umap",interactive = FALSE)
```

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", clono = 1,
  clust.dim = 2, cell.size = 1, cell.colors = c("red", "gray"),
  box.cell.col = "black", back.col = "white",
  cell.transparency = 0.5, interactive = TRUE, out.name = "plot")
```

Arguments

x	An object of class iCellR.
plot.data.type	Choose from "tsne" and "pca", default = "tsne".
clono	A clonotype name to be plotted, default = 1.
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 out put name for HTML, default = "plot".

Value

An object of class iCellR.

Examples

```
## Not run:
clono.plot(my.obj,
            plot.data.type = "tsne",
            clono = 1,
            clust.dim = 2,
            cell.size = 1,
            cell.colors = c("red","gray"),
            cbox.cell.col = "black",
            back.col = "white",
            interactive = TRUE,
            cell.transparency = 0.5,
            out.name = "tSNE_3D_clusters")

## End(Not run)
```

---

clust.avg.exp *Create a data frame of mean expression of genes per cluster*

Description

This function takes an object of class iCellR and creates an average gene expression for every cluster.

Usage

```
clust.avg.exp(x = NULL)
```

Arguments

x	An object of class iCellR.

Value

An object of class iCellR.

Examples

```
demo.obj <- clust.avg.exp(demo.obj)

head(demo.obj@clust.avg)
```

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

Description

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

Usage

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

Arguments

`x`	An object of class iCellR.
`plot.type`	Choose from pie or bar, defult = 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.

Value

An object of class iCellR.

Examples

```
clust.cond.info(demo.obj, plot.type = "pie", normalize.ncell = TRUE, my.out.put = "data")

head(demo.obj@my.freq)

clust.cond.info(demo.obj, plot.type = "pie", normalize.ncell = TRUE, my.out.put = "plot")
```

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

Examples

```
demo.obj <- clust.rm(demo.obj, clust.to.rm = 1)
```

clust.stats.plot *QC on clusters (nGenes, UMIs and percent mito)*

Description

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

Usage

```
clust.stats.plot(x = NULL, plot.type = "box.mito",
  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 "box.umi", "box.mito", "box.gene", default = "box.mito".
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".

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 out put name for HTML, default = "plot".

Value

An object of class iCellR.

Examples

```
clust.stats.plot(demo.obj,
                  plot.type = "box.mito",
                  interactive = FALSE,
                  out.name = "box.mito.clusters")
```

---

cluster.plot *Plot nGenes, UMIs and perecent mito*

Description

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

Usage

```
cluster.plot(x = NULL, cell.size = 1, plot.type = "tsne",
  cell.color = "black", back.col = "white", col.by = "clusters",
  cond.shape = FALSE, cell.transparency = 0.5, 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", "diffusion", "pseudo.A" and "pseudo.B", default = "tsne".
cell.color	Choose cell color if col.by = "monochrome", default = "black".
back.col	Choose background color, default = "black".
col.by	Choose between "clusters", "conditions", "cc" (cell cycle) or "monochrome", default = "clusters".
cond.shape	If TRUE the conditions will be shown in shapes.
cell.transparency	A numeric value between 0 to 1, default = 0.5.
clust.dim	A numeric value for plot dimensions. Choose either 2 or 3, default = 2.
angle	A number to rotate the non-interactive 3D plot.
clonotype.max	Number of clonotype to plot, default = 10.

density	If TRUE the density plots for PCA/tSNE second dimension will be created, default = FALSE.
interactive	If TRUE an html interactive file will be made, default = TRUE.
static3D	If TRUE a non-interactive 3D plot will be made.
out.name	Output name for html file if interactive = TRUE, default = "plot".

Value

An object of class iCellR.

Examples

```
cluster.plot(demo.obj,plot.type = "umap",interactive = FALSE)

cluster.plot(demo.obj,plot.type = "tsne",interactive = FALSE)

cluster.plot(demo.obj,plot.type = "pca",interactive = FALSE)

cluster.plot(demo.obj,plot.type = "pca",col.by = "conditions",interactive = FALSE)

cluster.plot(demo.obj,plot.type = "umap",col.by = "conditions",interactive = FALSE)

cluster.plot(demo.obj,plot.type = "tsne",col.by = "conditions",interactive = FALSE)
```

---

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)
myDemo.obj
```

`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.

Examples

```
my.obj <- data.scale(my.obj)
```

demo.obj *An object of class iCellR for demo*

Description

A demo object

Usage

```
demo.obj
```

Format

Subset of the data with 200 genes and 90 cells

Source

https://s3-us-west-2.amazonaws.com/10x.files/samples/cell/pbmc3k/pbmc3k_filtered_gene_bc_matrices.tar.gz

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

```
down.sample(x = NULL)
```

Arguments

x	An object of class iCellR.

Value

An object of class iCellR.

Examples

```
my.obj <- down.sample(my.obj)
```

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

```
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.

Examples

```
demo.obj <- find.dim.genes(demo.obj, dims = 1:10,top.pos = 20, top.neg = 20)

head(demo.obj@gene.model)
```

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, fold.change = 2, padjval = 0.1,
  Inf.FCs = FALSE, uniq = FALSE, positive = TRUE)
```

Arguments

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

Value

An object of class iCellR

Examples

```
marker.genes <- findMarkers(demo.obj,fold.change = 2,padjval = 0.1,uniq = TRUE)

head(marker.genes)
```

---

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

https://science.sciencemag.org/content/352/6282/189

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,
  data.type = "main", box.to.test = 0, box.pval = "sig.signs",
  plot.data.type = "tsne", scaleValue = FALSE, min.scale = -2.5,
  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 = 0.5, interactive = TRUE, out.name = "plot")
```

Arguments

x	An object of class iCellR.
gene	A gene name to be plotted.
cond.shape	If TRUE the conditions will be shown in shapes.
data.type	Choose from "main" or "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 avrage would be choosen, 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".
plot.data.type	Choose between "tsne", "pca", "umap", "diffusion", "pseudo.A" and "pseudo.B", default = "tsne".
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 = 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 out put name for HTML, default = "plot".

Value

An object of class iCellR.

Examples

```
gene.plot(demo.obj, gene = "CD74",interactive = FALSE)

gene.plot(demo.obj, gene = "CD74",plot.data.type = "umap",interactive = FALSE)

gene.plot(demo.obj, gene = "CD74",
          plot.data.type = "umap",
          interactive = FALSE,
          plot.type = "barplot")

gene.plot(demo.obj, gene = "CD74",
          plot.data.type = "umap",
          interactive = FALSE,
          plot.type = "boxplot")
```

`gene.stats` *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.

Examples

```
demo.obj <- gene.stats(demo.obj, which.data = "main.data")
head(demo.obj@gene.data)
```

`gg.cor` *Gene-gene correlation. This function helps to visulaize and calculate gene-gene correlations.*

Description

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

Usage

```
gg.cor(x = NULL, data.type = "imputed", gene1 = NULL, gene2 = NULL,
  conds = 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.
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

Examples

```
gg.cor(my.obj, interactive = F, gene1 = "NKG7",gene2 = "GNLY", conds=c("WT"))
```

---

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

```
heatmap.gg.plot(x = NULL, gene = "NULL", data.type = "main",
  cluster.by = "clusters", 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.
data.type	Choose from "main" and "imputed", default = "main".
cluster.by	Choose from "clusters" or "conditions", default = "clusters".
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	Chhose 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 iCellR

Examples

```
marker.genes <- findMarkers(demo.obj,fold.change = 2,padjval = 0.1,uniq = TRUE)

MyGenes <- top.markers(marker.genes, topde = 10, min.base.mean = 0.8)

heatmap.gg.plot(demo.obj,
                gene = MyGenes,
                out.name = "plot",
                cluster.by = "clusters",
                interactive = FALSE)
```

---

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

```
## Not run:
# The directory should have barcodes.tsv, genes.tsv, and matrix.mtx files.
load10x('path/to/data/directory')

## End(Not run)
```

`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

```
make.gene.model(x = NULL, dispersion.limit = 1.5,
  base.mean.rank = 500, non.sig.col = "darkgray",
  right.sig.col = "chartreuse3", left.sig.col = "cadetblue3",
  disp.line.col = "black", rank.line.col = "red",
  my.out.put = "data", 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.
`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 out put name for HTML, default = "plot".

Value

An object of class iCellR.

Examples

```
make.gene.model(demo.obj,
                dispersion.limit = 1.5,
                base.mean.rank = 500,
                no.mito.model = TRUE,
                mark.mito = TRUE,
                interactive = FALSE,
                my.out.put = "plot",
                out.name = "gene.model")

demo.obj <- make.gene.model(demo.obj,
                            dispersion.limit = 1.5,
                            base.mean.rank = 500,
                            no.mito.model = TRUE,
                            mark.mito = TRUE,
                            interactive = FALSE,
                            out.name = "gene.model")

head(demo.obj@gene.model)
```

---

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

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

Arguments

x	An object of class iCellR.

Value

An object of class iCellR

Examples

```
## Not run:
my.obj <- norm.adt(my.obj)

## End(Not run)
```

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

```
norm.data(x = NULL, norm.method = "ranked.glsf", top.rank = 500,
  spike.in.factors = NULL, rpm.factor = 1000)
```

Arguments

x	An object of class iCellR.
norm.method	Choose a normalization method, there are three option currently. Choose from "global.glsf", "ranked.glsf", "ranked.deseq", "deseq", "rpm","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 norm.method is set to "rpm" the library sizes would be diveded by this number, default = 1000 (higher numbers recomanded for bulk RNA-Seq).

Value

An object of class iCellR.

Examples

```
demo.obj <- norm.data(demo.obj, norm.method = "ranked.glsf", top.rank = 500)


my.obj <- norm.data(my.obj,
                    norm.method = "ranked.glsf",
                    top.rank = 500) # best for scRNA-Seq

my.obj <- norm.data(my.obj, norm.method = "global.glsf") # best for bulk RNA-Seq
my.obj <- norm.data(my.obj, norm.method = "rpm", rpm.factor = 100000) # best for bulk RNA-Seq
my.obj <- norm.data(my.obj, norm.method = "spike.in", spike.in.factors = NULL)
my.obj <- norm.data(my.obj, norm.method = "no.norm") # if the data is already normalized
```

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, defult = 50.

Value

An object of class iCellR.

Examples

```
opt.pcs.plot(demo.obj)
```

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 = "all_contig_annotations.csv", cond.name = "NULL")
```

Arguments

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

Value

An object of class iCellR

Examples

```
## Not run:
Read your VDJ data (in this case in VDJ.tsv file) and add to your object as below

my.vdj.data <- read.table("VDJ.tsv")

VDJ <- prep.vdj(my.obj, adt.data = my.vdj.data)

head(VDJ)

## End(Not run)
```

`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 dimentions to be used, , default = 1:10.

Value

An object of class iCellR.

Examples

```
## Not run:
my.obj <- pseudotime(my.obj, marker.genes = MyGenes, dims = 1:10)

## End(Not run)
```

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.

Examples

```
marker.genes <- findMarkers(demo.obj,fold.change = 2,padjval = 0.1,uniq = TRUE)

MyGenes <- top.markers(marker.genes, topde = 10, min.base.mean = 0.8)

pseudotime.tree(demo.obj,
                marker.genes = MyGenes,
                type = "unrooted",
                clust.method = "complete")
```

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 = "default.genes", 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, 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

Examples

```
New.demo.obj <- qc.stats(demo.obj)
head(New.demo.obj@stats)
```

run.cca *Run CCA on the main data*

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.

---

run.clustering *Clustering the data*

Description

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

Usage

```
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", "ptbiserial", "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.nc.
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.

Examples

```
demo.obj <- run.clustering(demo.obj,
                           clust.method = "kmeans",
                           dist.method = "euclidean",
                           index.method = "silhouette",
                           max.clust = 2,
                           min.clust = 2,
                           dims = 1:10)

 head(demo.obj@best.clust)
```

---

run.diff.exp *Differential expression (DE) analysis*

Description

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

Usage

```
run.diff.exp(x = NULL, de.by = "clusters", cond.1 = "array",
  cond.2 = "array", base.cond = 0)
```

Arguments

x	An object of class iCellR.
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

Examples

```
diff.res <- run.diff.exp(demo.obj, de.by = "clusters", cond.1 = c(1), cond.2 = c(2))

head(diff.res)

## Not run:
diff.res <- run.diff.exp(my.obj, de.by = "clusters", cond.1 = c(1,4), cond.2 = c(2))
diff.res <- run.diff.exp(my.obj, de.by = "conditions", cond.1 = c("WT"), cond.2 = c("KO"))
diff.res <- run.diff.exp(my.obj, de.by = "clustBase.condComp",
cond.1 = c("WT"), cond.2 = c("KO"), base.cond = 1)
diff.res <- run.diff.exp(my.obj, de.by = "condBase.clustComp",
cond.1 = c(1), cond.2 = c(2), base.cond = "WT")

## End(Not run)
```

`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 = "phate",
  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 dimentions 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 'nonmetric' 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, print 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)
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.
pca.method	Deprecated.
g.kernel	Deprecated.
diff.op	Deprecated.
landmark.transitions	Deprecated.
diff.op.t	Deprecated.
dist.method	Deprecated.

Value

An object of class iCellR.

Examples

```
demo.obj <- run.diffusion.map(demo.obj, dims = 1:10, method = "phate")
head(demo.obj@diffusion.data)
```

`run.impute` *Impute the main data*

Description

This function takes an object of class iCellR and runs imputation on the main data. MAGIC as one of the methods: Markov Affinity-based Graph Imputation of Cells (MAGIC) is an algorithm for denoising and transcript recover of single cells applied to single-cell RNA sequencing data, as described in van Dijk et al, 2018.

Usage

```
run.impute(x = NULL, 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.
`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`	int, optional, default: 10 number of nearest neighbors on which to build kernel
`alpha`	int, optional, default: 15 sets decay rate of kernel tails. If NULL, alpha decaying kernel is not used
`t`	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`	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', print 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.

Examples

```
demo.obj <- run.impute(demo.obj)
```

`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, 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 dimentions to be used for tSNE analysis.
`my.seed`	seed number, default = 0.
`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 printed (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.

Examples

```
demo.obj <- run.pc.tsne(demo.obj, dims = 1:10,perplexity = 20)

head(demo.obj@pca.data)[1:5]
```

---

run.pca *Run PCA on the main data*

Description

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

Usage

```
run.pca(x = NULL, data.type = "main", method = "base.mean.rank",
  top.rank = 500, plus.log.value = 0.1, batch.norm = FALSE,
  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.
batch.norm	If TRUE the data will be normalized based on the genes in gene.list or top ranked genes.
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.

Examples

```
demo.obj <- run.pca(demo.obj, method = "gene.model", gene.list = demo.obj@gene.model)

head(demo.obj@pca.data)[1:5]
```

---

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

```
run.tsne(x = NULL, clust.method = "base.mean.rank", top.rank = 500,
  gene.list = "character", 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", defult is "base.mean.rank".
top.rank	A number taking the top genes ranked by base mean, defult = 500.
gene.list	A list of genes to be used for tSNE analysis. If "clust.method" is set to "gene.model", defult = "my_model_genes.txt".
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 printed (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.

Examples

```
demo.obj <- run.tsne(demo.obj, perplexity = 20)

head(demo.obj@tsne.data)
```

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

```
run.umap(x = NULL, dims = 1:10, method = "naive")
```

Arguments

x	An object of class iCellR.
dims	PC dimentions to be used for UMAP analysis.
method	Character, implementation. Available methods are 'naive' (an implementation written in pure R) and 'umap-learn' (requires python package 'umap-learn'). Choose from "naive" and "umap-learn", default = "naive".

Value

An object of class iCellR.

Examples

```
demo.obj <- run.umap(demo.obj, dims = 1:10)
head(demo.obj@umap.data)
```

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://science.sciencemag.org/content/352/6282/189

stats.plot *Plot nGenes, UMIs and percent mito*

Description

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

Usage

```
stats.plot(x = NULL, plot.type = "box.umi",
  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", "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.
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 out put name for HTML, default = "plot".

Value

An object of class iCellR.

Examples

```
stats.plot(demo.obj,
          plot.type = "all.in.one",
          out.name = "UMI-plot",
          interactive = FALSE,
          cell.color = "slategray3",
          cell.size = 1,
          cell.transparency = 0.5,
          box.color = "red",
          box.line.col = "green")
```

top.markers *Choose top marker genes*

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 choosen 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

Examples

```
marker.genes <- findMarkers(demo.obj,fold.change = 2,padjval = 0.1,uniq = TRUE)
top.markers(marker.genes, topde = 10, min.base.mean = 0.8)
```

vdj.stats *VDJ stats*

Description

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

Usage

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

Arguments

vdj.data	A data frame containing VDJ data for cells.

Value

An object of class iCellR

Examples

```
## Not run:
Read your VDJ data (in this case in VDJ.tsv file) and add to your object as below

my.vdj.data <- read.table("VDJ.tsv")

VDJ <- prep.vdj(my.obj, adt.data = my.vdj.data)

head(VDJ)

vdj.stats(vdj.data = VDJ)

## End(Not run)
```

`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

```
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", "#3182bd", "#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 out put name for HTML, default = "plot".

Value

Plots

Examples

```
diff.res <- run.diff.exp(demo.obj, de.by = "clusters", cond.1 = c(1), cond.2 = c(2))

volcano.ma.plot(diff.res,
              sig.value = "pval",
              sig.line = 0.05,
              plot.type = "volcano",
              interactive = FALSE)

volcano.ma.plot(diff.res,
             sig.value = "pval",
             sig.line = 0.05,
             plot.type = "ma",
             interactive = FALSE)
```

Index