

Package ‘iCellR’

May 3, 2019

Type Package

Title iCellR; A Feature-rich Interactive R Package to Work with High-Throughput Single Cell Sequencing Technologies.

Version 0.99.0

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

Maintainer The package maintainer;
Alireza Khodadadi-Jamayran,
Applied Bioinformatics Laboratories (ABL),
Genome Technology Center (GTC),
Division of Advanced Research Technologies (DART),
NYU Langone Medical Center,
550 1st Ave, SB-604/MSB-304, New York, NY 10016,
<alireza.khodadadi-jamayran@nyumc.org>

Description

Single (i) Cell R package (iCellR) is an interactive R package to work with high-throughput single cell sequencing technologies (i.e scRNA-seq, scVDJ-seq and CITE-seq). iCellR is an R package that allows scientists unprecedented flexibility at every step of the analysis pipeline, including normalization, clustering, dimensionality reduction, imputation, visualization, and so on. iCellR allows users to design both unsupervised and supervised models to best suit their research. In addition, iCellR provides 2D and 3D interactive visualizations, differential expression analysis, filters based on cells, genes and clusters, data merging, normalizing for dropouts and filling them with imputation methods, batch differences, pathway analysis and tools to find marker genes for clusters and conditions, predict cell types and pseudotime analysis.

Keywords PCA, UMAP, Diffusion Map, tSNE, high throughput, single cell, differential gene expression, analysis, clustering, PseudoTime, dimension reduction, normalization, spike-in, scRNA-Seq, CITE-Seq, scVDJ-Seq, data visualization, interactive plots, 3D plots, cell type prediction, pathway analysis, batch correction, cell cycle, cell gating, ADT, filtering, Clustering methods: ward.D, ward.D2, single, complete, average, mcquitty, median, centroid, kmeans, distance calculation methods: euclidean, maximum, "manhattan, canberra, binary, minkowski, indexing methods: kl, ch, hartigan, ccc, scott, marriot, trcovw, tracew, fried-

man, rubin, cindex, db, silhouette, duda, pseudot2, beale, ratkowsky, ball, ptbise-
rial, gap, frey, mcclain, gamma, gplus, tau, dunn, hubert, sdindex, dindex, sdbw, normaliza-
tion methods: ranked.glsf, global.glsf, deseq, rpm, spike.in.

Depends R (>= 3.3.0), ggplot2, plotly

Imports Matrix,
Rtsne,
gridExtra,
ggpubr,
scatterplot3d,
RColorBrewer,
knitr,
NbClust,
reshape,
shiny,
umap,
pheatmap

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

VignetteBuilder knitr

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

Suggests rmarkdown,
Rmagic,
phateR

R topics documented:

add.adt	3
add.vdj	4
adt.rna.merge	4
cc	5
cell.filter	6
cell.gating	7
cell.type.pred	7
change.clust	8
clono.plot	9
clust.avg.exp	10
clust.cond.info	10
clust.rm	11
clust.stats.plot	11
cluster.plot	12
data.aggregation	13
data.scale	14
diff.exp	14
down.sample	15
find.dim.genes	16
findMarkers	16
gate.to.clust	17

<i>add.adt</i>	3
gene.plot	18
gene.stats	19
gg.cor	20
heatmap.gg.plot	21
load10x	21
make.gene.model	22
make.obj	23
norm.adt	24
norm.data	24
opt.pcs.plot	25
prep.vdj	26
pseudotime.tree	26
qc.stats	27
run.cca	28
run.clustering	28
run.diffusion.map	29
run.impute	31
run.pc.tsne	32
run.pca	34
run.tsne	35
run.umap	36
stats.plot	37
top.markers	38
vdj.stats	38
volcano.ma.plot	39
Index	41

<code>add.adt</code>	<i>Add CITE-seq antibody-derived tags (ADT)</i>
----------------------	---

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

<code>x</code>	An object of class iCellR.
<code>adt.data</code>	A data frame containing ADT counts for cells.

Value

An object of class iCellR

Examples

```
## Not run:
my.obj <- add.adt(my.obj, adt.data = adt.data)

## End(Not run)
```

add.vdj	<i>Add V(D)J recombination data</i>
---------	-------------------------------------

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.
adt.data	A data frame containing VDJ information for cells.

Value

An object of class iCellR

Examples

```
## Not run:
my.obj <- add.adt(my.obj, adt.data = adt.data)

## End(Not run)
```

adt.rna.merge	<i>Merge RNA and ADT data</i>
---------------	-------------------------------

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	<i>Calculate Cell cycle phases</i>
----	------------------------------------

Description

This function takes an object of class iCellR and assigns 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

```
## Not run:
cell.filter(my.obj,
  min.mito = 0,
  max.mito = 1,
  min.genes = 0,
  max.genes = Inf,
  min.umis = 0,
  max.umis = Inf)

## End(Not run)
```

cell.gating	<i>Cell gating</i>
-------------	--------------------

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	<i>Create heatmaps or dot plots for genes in clusters to find thier cell types using ImmGen data.</i>
----------------	---

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" od "uli.rna", 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 = F)
```

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

```
## Not run:
my.obj <- change.clust(my.obj, change.clust = 3, to.clust = 1)
my.obj <- change.clust(my.obj, change.clust = 2, to.clust = "B Cell")
my.obj <- change.clust(my.obj, clust.reset = T)

## End(Not run)
```

clono.plot	<i>Make 2D and 3D scatter plots for clonotypes.</i>
------------	---

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").
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 = T,
  cell.transparency = 0.5,
  out.name = "tSNE_3D_clusters")

## End(Not run)
```

clust.avg.exp	<i>Create a data frame of mean expression of genes per cluster</i>
---------------	--

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

```
## Not run:
my.obj <- clust.avg.exp(my.obj)

## End(Not run)
```

clust.cond.info	<i>Calculate cluster and conditions frequencies</i>
-----------------	---

Description

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

Usage

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

Arguments

x An object of class iCellR.

plot.type Choose from pie or bar, default = pie.

Value

An object of class iCellR.

Examples

```
## Not run:
clust.cond.info(my.obj, plot.type = "pie", normalize.ncell = TRUE)
clust.cond.info(my.obj, plot.type = "bar")

## End(Not run)
```

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

```
## Not run:
my.obj <- clust.rm(my.obj, clust.to.rm = 5)

## End(Not run)
```

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 = F, 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".
notch	Notch the box plots, default = F.
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

```
## Not run:
clust.stats.plot(my.obj, plot.type = "box.mito", interactive = F, out.name = "box.mito.clusters")

## End(Not run)
```

cluster.plot

Plot nGenes, UMIs and perecent mito

Description

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

Usage

```
cluster.plot(x = NULL, cell.size = 1, plot.type = "tsne",
  cell.color = "black", back.col = "white", col.by = "clusters",
  cond.shape = F, cell.transparency = 0.5, clust.dim = 2,
  angle = 20, clonotype.max = 10, density = F, interactive = TRUE,
  static3D = F, 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" and "pca", 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".
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.
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.
out.name	Output name for html file if interactive = TRUE, default = "plot".

Value

An object of class iCellR.

Examples

```
## Not run:
tsne.plot(my.obj)

## End(Not run)
```

data.aggregation	<i>Merge multiple data frames and add the condition names to their cell ids</i>
------------------	---

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

```
## Not run:
my.data <- data.aggregation(samples = c("sample1", "sample2", "sample3"),
                             condition.names = c("WT", "KO", "Ctrl"))

## End(Not run)
```

data.scale	<i>Scale data</i>
------------	-------------------

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

```
## Not run:
my.obj <- data.scale(my.obj)

## End(Not run)
```

diff.exp	<i>Differential expression (DE) analysis</i>
----------	--

Description

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

Usage

```
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

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

## End(Not run)
```

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

```
## Not run:
my.obj <- down.sample(my.obj)

## End(Not run)
```

find.dim.genes	<i>Find best model genes from PCA data</i>
----------------	--

Description

This function takes an object of class iCellR finds the best 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

```
## Not run:
my.obj <- run.pca(my.obj, clust.method = "gene.model", gene.list = "my_model_genes.txt")

## End(Not run)
```

findMarkers	<i>Find marker genes for each cluster</i>
-------------	---

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 = F,
  uniq = F, 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

```
## Not run:
marker.genes <- findMarkers(my.obj, fold.change = 2, padjval = 0.1, uniq = T)

## End(Not run)
```

gate.to.clust	<i>Assign cluster number to cell ids</i>
---------------	--

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.

Examples

```
## Not run:
my.obj <- gate.to.clust(my.obj, my.gate = readLines("ids.txt"), to.clust = 1)

## End(Not run)
```

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", data.type = "main",
  box.to.test = 0, box.pval = "sig.signs", plot.data.type = "tsne",
  min.scale = -1, max.scale = 1, 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.
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 from "tsne" and "pca", default = "tsne".
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

```

## Not run:
cluster.plot(my.obj,
             cell.size = 1,
             plot.type = "tsne",
             cell.color = "black",
             back.col = "white",
             col.by = "clusters",
             cell.transparency = 0.5,
             clust.dim = 3,
             interactive = T,
             density = F,
             out.name = "tSNE_3D_clusters")

cluster.plot(my.obj, cell.size = 1,
             plot.type = "tsne",
             col.by = "clusters",
             cell.transparency = 0.5,
             clust.dim = 2,
             interactive = T,
             density = F,
             out.name = "tSNE_2D_clusters")

cluster.plot(my.obj,
             cell.size = 2,
             plot.type = "tsne",
             clust.dim = 2,
             interactive = F)

cluster.plot(my.obj,
             cell.size = 1,
             plot.type = "tsne",
             col.by = "clusters",
             clust.dim = 3,
             interactive = F,
             angle = 45)

cluster.plot(my.obj,
             cell.size = 1,
             plot.type = "pca",
             cell.color = "black",
             back.col = "white",
             col.by = "conditions",
             cell.transparency = 0.5,
             clust.dim = 3,
             interactive = T,
             density = F,
             out.name = "PCA_3D_conditions")

## End(Not run)

```

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 = F)
```

Arguments

x An object of class iCellR.
which.data Choose from "raw.data" or "main.data", default = "raw.data".

Value

An object of class iCellR.

Examples

```
## Not run:
my.obj <- gene.stats(my.obj, which.data = "main.data")

## End(Not run)
```

gg.cor	<i>Normalize ADT data. This function takes data frame and Normalizes ADT data.</i>
--------	--

Description

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

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.

Value

An object of class iCellR

Examples

```
## Not run:
gg.cor(my.obj, interactive = F, gene1 = "NKG7", gene2 = "GNLY", conds=c("WT"))

## End(Not run)
```

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 = T, 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.
cluster.by	Choose from "clusters" or "conditions", default = "clusters".
interactive	If TRUE an html interactive file will be made, default = TRUE.
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

```
## Not run:
heatmap.gg.plot(my.obj, gene = MyGenes, interactive = T, out.name = "plot", cluster.by = "clusters")

## End(Not run)
```

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 = "geneSymbol")
```

Arguments

<code>dir.10x</code>	A directory that includes the 10X barcodes.tsv, genes.tsv and matrix.mtx files.
<code>gene.name</code>	Should be either geneSymbol or ensembleID.

Value

The data frame object

Examples

```
## Not run:
load10x("/hg19", gene.name = "geneSymbol")

## End(Not run)
```

<code>make.gene.model</code>	<i>Make a gene model for clustering</i>
------------------------------	---

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

Arguments

<code>x</code>	An object of class iCellR.
<code>dispersion.limit</code>	A number for taking the genes that have dispersion above this number, default = 1.5.
<code>base.mean.rank</code>	A number taking the top genes ranked by base mean, default = 500.
<code>non.sig.col</code>	Color for the genes not used for the model, default = "darkgray".
<code>right.sig.col</code>	Color for the genes above the dispersion limit, default = "chartreuse3".
<code>left.sig.col</code>	Color for the genes above the rank limit, default = "cadetblue3".
<code>disp.line.col</code>	Color of the line for dispersion limit, default = "black".
<code>rank.line.col</code>	Color of the line for rank limit, default = "red".
<code>cell.size</code>	A number for the size of the points in the plot, default = 1.75.
<code>cell.transparency</code>	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.

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

```
## Not run:
make.gene.model(my.obj,
  dispersion.limit = 1.5,
  base.mean.rank = 500,
  no.mito.model = T,
  mark.mito = T,
  interactive = T,
  out.name = "gene.model")

make.gene.model(my.obj,
  dispersion.limit = 1.5,
  base.mean.rank = 500,
  no.mito.model = T,
  mark.mito = T,
  interactive = F,
  out.name = "gene.model")

## End(Not run)
```

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

```
## Not run:
my.obj <- make.obj(my.data)

## End(Not run)
```

norm.adt	<i>Normalize ADT data. This function takes data frame and Normalizes ADT data.</i>
----------	--

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	<i>Normalize data</i>
-----------	-----------------------

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", "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.
rpm.factor	If the norm.method is set to "rpm" the library sizes would be divided by this number, default = 1000 (higher numbers recomanded for bulk RNA-Seq).

Value

An object of class iCellR.

Examples

```
## Not run:
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

## End(Not run)
```

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.

Examples

```
## Not run:
find.opt.pcs(my.obj)

## End(Not run)
```

```
prep.vdj
```

```
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

```
prep.vdj(vdj.data = "all_contig_annotations.csv", cond.name = "NULL")
```

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

## 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".
cex	Text size, default = 1.

Value

An object of class iCellR.

Examples

```
## Not run:
my.obj <- run.pca(my.obj, clust.method = "gene.model", gene.list = "my_model_genes.txt")

## End(Not run)
```

qc.stats	<i>Calculate the number of UMIs and genes per cell and percentage of mitochondrial genes per cell.</i>
----------	--

Description

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

Usage

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

Arguments

x	A data frame containing gene counts for cells.
---	--

Value

The data frame object

Examples

```
## Not run:
UMIs.genes.mit(my.data)

## End(Not run)
```

run.cca	<i>Run PCA on the main data</i>
---------	---------------------------------

Description

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

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.
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.
plus.log.value	A number to add to each value in the matrix before log transformation to avoid Inf numbers, default = 0.1.
gene.list	A list 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

```
## Not run:
my.obj <- run.pca(my.obj, clust.method = "gene.model", gene.list = "my_model_genes.txt")

## End(Not run)
```

run.clustering	<i>Clustering the data</i>
----------------	----------------------------

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

```
## Not run:
my.obj <- run.clustering(my.obj,
  clust.method = "kmeans",
  dist.method = "euclidean",
  index.method = "silhouette",
  max.clust = 25,
  min.clust = 2,
  dims = 1:10)

## End(Not run)
```

run.diffusion.map	<i>Run PHATE on PCA data (PHATE - Potential of Heat-Diffusion for Affinity-Based Transition Embedding)</i>
-------------------	--

Description

This function takes an object of class iCellR and runs PHATE 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.
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, default: FALSE 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

```
## Not run:
my.obj <- run.diffusion.map(my.obj, dims = 1:10, method = "phate")

## End(Not run)
```

run.impute	<i>Run MAGIC on Main Data.</i>
------------	--------------------------------

Description

This function takes an object of class iCellR and runs MAGIC on main data. 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', 'co-sine'. 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

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

## End(Not run)
```

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.
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

```
## Not run:
my.obj <- run.pc.tsne(my.obj, dims = 1:10)

## End(Not run)
```

run.pca	<i>Run PCA on the main data</i>
---------	---------------------------------

Description

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

Usage

```
run.pca(x = NULL, data.type = "main",
        clust.method = "base.mean.rank", top.rank = 500,
        plus.log.value = 0.1, batch.norm = F, gene.list = "character")
```

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.
plus.log.value	A number to add to each value in the matrix before log transformation to avoid Inf numbers, default = 0.1.
gene.list	A list 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

```
## Not run:
my.obj <- run.pca(my.obj, clust.method = "gene.model", gene.list = "my_model_genes.txt")

## End(Not run)
```

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

```

## Not run:
my.obj <- run.tsne(my.obj, clust.method = "gene.model", gene.list = "my_model_genes.txt")

## End(Not run)

```

run.umap	<i>Run UMAP on PCA Data (Computes a manifold approximation and projection)</i>
----------	--

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

```
## Not run:
my.obj <- run.umap(my.obj, dims = 1:10)

## End(Not run)
```

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".
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

```
## Not run:
stats.plot(my.obj,
  plot.type = "box.gene.umi.mito",
  out.name = "UMI-plot",
  interactive = F,
  cell.color = "slategray3",
  cell.size = 1,
  cell.transparency = 0.5,
  box.color = "red",
  box.line.col = "green",
```

```

        back.col = "white")

stats.plot(my.obj, plot.type = "point.gene.umi", interactive = T, out.name = "scatter.gene.umi")

stats.plot(my.obj, plot.type = "point.mito.umi", interactive = T, out.name = "scatter.mito.umi")

## End(Not run)

```

top.markers	<i>Choose top marker genes</i>
-------------	--------------------------------

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 = T, 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.

Value

A set of gene names

Examples

```

## Not run:
MyGenes <- top.markers(marker.genes, topde = 10, min.base.mean = 0.8)

## End(Not run)

```

vdj.stats	<i>Add CITE-seq antibody-derived tags (ADT)</i>
-----------	---

Description

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

Usage

```

vdj.stats(vdj.data = "VDJ_analysis_ready.tsv")

```

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

## End(Not run)
```

volcano.ma.plot	<i>Create MA and Volcano plots.</i>
-----------------	-------------------------------------

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 = "pval", sig.line = 0.1,
  plot.type = "volcano", x.limit = 2, y.limit = 2, limit.force = F,
  scale.ax = T, 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

```
## Not run:
volcano.ma.plot(diff.res,
  sig.value = "pval",
  sig.line = 0.05,
  plot.type = "volcano",
  interactive = T,
  out.name = "Volcano.plot-Control2_vs_Treated3")

volcano.ma.plot(diff.res,
  sig.value = "pval",
  sig.line = 0.05,
  plot.type = "ma",
  interactive = T,
  out.name = "MA.plot-Control2_vs_Treated3")

## End(Not run)
```


Index

add.adt, [3](#)
add.vdj, [4](#)
adt.rna.merge, [4](#)

cc, [5](#)
cell.filter, [6](#)
cell.gating, [7](#)
cell.type.pred, [7](#)
change.clust, [8](#)
clono.plot, [9](#)
clust.avg.exp, [10](#)
clust.cond.info, [10](#)
clust.rm, [11](#)
clust.stats.plot, [11](#)
cluster.plot, [12](#)

data.aggregation, [13](#)
data.scale, [14](#)
diff.exp, [14](#)
down.sample, [15](#)

find.dim.genes, [16](#)
findMarkers, [16](#)

gate.to.clust, [17](#)
gene.plot, [18](#)
gene.stats, [19](#)
gg.cor, [20](#)

heatmap.gg.plot, [21](#)

load10x, [21](#)

make.gene.model, [22](#)
make.obj, [23](#)

norm.adt, [24](#)
norm.data, [24](#)

opt.pcs.plot, [25](#)

prep.vdj, [26](#)
pseudotime.tree, [26](#)

qc.stats, [27](#)

run.cca, [28](#)
run.clustering, [28](#)
run.diffusion.map, [29](#)
run.impute, [31](#)
run.pc.tsne, [32](#)
run.pca, [34](#)
run.tsne, [35](#)
run.umap, [36](#)

stats.plot, [37](#)

top.markers, [38](#)

vdj.stats, [38](#)
volcano.ma.plot, [39](#)