

# Package ‘iCellR’

October 15, 2019

**Type** Package

**Title** Analyzing High-Throughput Single Cell Sequencing Data

**Version** 1.2.0

**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

**Repository** CRAN

**Date/Publication** 2019-10-15 05:20:02 UTC

## R topics documented:

|                            |    |
|----------------------------|----|
| add.adt . . . . .          | 3  |
| add.vdj . . . . .          | 3  |
| adt.rna.merge . . . . .    | 4  |
| cc . . . . .               | 5  |
| cell.filter . . . . .      | 5  |
| cell.gating . . . . .      | 6  |
| cell.type.pred . . . . .   | 7  |
| change.clust . . . . .     | 7  |
| clono.plot . . . . .       | 8  |
| clust.avg.exp . . . . .    | 9  |
| clust.cond.info . . . . .  | 9  |
| clust.rm . . . . .         | 10 |
| clust.stats.plot . . . . . | 11 |
| cluster.plot . . . . .     | 12 |
| data.aggregation . . . . . | 13 |
| data.scale . . . . .       | 14 |
| demo.obj . . . . .         | 14 |
| down.sample . . . . .      | 15 |
| find.dim.genes . . . . .   | 15 |
| findMarkers . . . . .      | 16 |
| g2m.phase . . . . .        | 17 |
| gate.to.clust . . . . .    | 17 |
| gene.plot . . . . .        | 18 |
| gene.stats . . . . .       | 19 |
| gg.cor . . . . .           | 20 |
| heatmap.gg.plot . . . . .  | 21 |
| hto.anno . . . . .         | 22 |
| load10x . . . . .          | 22 |
| make.gene.model . . . . .  | 23 |
| make.obj . . . . .         | 24 |
| myImp . . . . .            | 25 |
| norm.adt . . . . .         | 26 |
| norm.data . . . . .        | 26 |
| opt.pcs.plot . . . . .     | 27 |
| prep.vdj . . . . .         | 27 |
| pseudotime . . . . .       | 28 |
| pseudotime.tree . . . . .  | 29 |
| qc.stats . . . . .         | 30 |
| run.cca . . . . .          | 30 |
| run.clustering . . . . .   | 31 |
| run.diff.exp . . . . .     | 32 |

|                             |    |
|-----------------------------|----|
| run.diffusion.map . . . . . | 33 |
| run.impute . . . . .        | 35 |
| run.pc.tsne . . . . .       | 36 |
| run.pca . . . . .           | 37 |
| run.phenograph . . . . .    | 38 |
| run.tsne . . . . .          | 39 |
| run.umap . . . . .          | 40 |
| s.phase . . . . .           | 41 |
| stats.plot . . . . .        | 41 |
| top.markers . . . . .       | 42 |
| vdj.stats . . . . .         | 43 |
| volcano.ma.plot . . . . .   | 44 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>46</b> |
|--------------|-----------|

---

|         |   |
|---------|---|
| add.adt | <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

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

### Value

An object of class iCellR

---

|         |                                     |
|---------|-------------------------------------|
| 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.  
vdj.data            A data frame containing VDJ information for cells.

**Value**

An object of class iCellR

**Examples**

```
my.vdj <- read.csv(file = system.file('extdata', 'all_contig_annotations.csv',  
  package = 'iCellR'),  
  as.is = TRUE)  
head(my.vdj)  
dim(my.vdj)  
  
My.VDJ <- prep.vdj(vdj.data = my.vdj, cond.name = "NULL")  
head(My.VDJ)  
dim(My.VDJ)  
  
my.obj <- add.vdj(demo.obj, vdj.data = My.VDJ)  
my.obj
```

---

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

---

cc *Calculate Cell cycle phase prediction*

---

### 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

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

### Value

The data frame object

---

`cell.filter` *Filter cells*

---

### Description

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

### Usage

```
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

|                        |   |
|------------------------|---|
| <code>x</code>         | An object of class <code>iCellR</code> .                          |
| <code>min.mito</code>  | Min rate for mitochondrial gene expression per cell, default = 0. |
| <code>max.mito</code>  | Max rate for mitochondrial gene expression per cell, default = 1. |
| <code>min.genes</code> | Min number genes per cell, default = 0.                           |
| <code>max.genes</code> | Max number genes per cell, default = Inf.                         |
| <code>min.umis</code>  | 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`.

---

|                |   |
|----------------|---|
| cell.type.pred | <i>Create heatmaps or dot plots for genes in clusters to find their 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", "uli.rna" or "mca", default = "rna"        |
| gene           | A set of gene names to used to predict cell type.             |
| top.cell.types | Top cell types sorted by cumulative expression, default = 25. |
| plot.type      | Choose from "heatmap" od "point.plot", default = "heatmap"    |
| heat.colors    | Colors for heatmap, default = c("blue", "white", "red").      |

**Value**

An object of class iCellR

---

|              |  |
|--------------|--|
| change.clust | <i>Change the cluster number or re-name them</i> |
|--------------|--|

---

**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.



---

|               |  |
|---------------|--|
| 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**

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

---

|                 |   |
|-----------------|---|
| 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", my.out.put = "data",
  normalize.ncell = TRUE)
```

**Arguments**

x                    An object of class iCellR.  
plot.type            Choose from pie or bar, default = pie.  
my.out.put           Chose from "data" or "plot", default = "data".  
normalize.ncell      If TRUE the values will be normalized to the number of cells by downsampling.

**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 percent 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 | <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**

```

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 | <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

```
my.obj <- data.scale(demo.obj)
head(my.obj@scaled.data)[1:5]
```

---

|          |   |
|----------|---|
| demo.obj | <i>An object of class iCellR for demo</i> |
|----------|---|

---

### 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](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(demo.obj)
```

```
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 | <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 = 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 = FALSE)
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**

|                       |   |
|-----------------------|---|
| <code>x</code>        | An object of class iCellR.                            |
| <code>my.gate</code>  | A vector of cell ids.                                 |
| <code>to.clust</code> | A cluster id to be assigned to the provided cell ids. |

**Value**

An object of class iCellR.

gene.plot

*Make scatter, box and bar plots for genes***Description**

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

**Usage**

```
gene.plot(x = NULL, gene = "NULL", cond.shape = FALSE,
  conds.to.plot = NULL, data.type = "main", box.to.test = 0,
  box.pval = "sig.signs", plot.data.type = "tsne",
  scaleValue = 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.   |
| conds.to.plot  | Choose the conditions you want to see in the plot, default = NULL (all conditions).   |
| 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 average would be chosen, default = 0. |
| box.pval       | Choose from "sig.values" and "sig.signs". If set to "sig.signs" p values would be replaced with signs ("na", "*", "**", "***"), default = "sig.signs".        |
| 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 | <i>Make statistical information for each gene across all the cells (SD, mean, expression, etc.)</i> |
|------------|---|

---

**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 visualize and calculate gene-gene correlations.*

---

**Description**

Gene-gene correlation. This function helps to visualize 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

---

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", cell.sort = TRUE,
  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.  |
| cell.sort   | If FALSE the cells will not be sorted based on their distance, default = TRUE. |
| data.type   | Choose from "main" and "imputed", default = "main".                            |
| cluster.by  | Choose from "clusters", "conditions" or "none", 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     | Choose a size, default = 10.   |
| cex.row     | Choose a size, default = 10.   |
| no.key      | If you want a color legend key, default = FALSE.                               |
| out.name    | Output name for html file if interactive = TRUE, default = "plot".             |
| heat.colors | Colors for heatmap, default = c("blue", "white", "red").                       |

### Value

An object of class 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)
```

---

|          |                            |
|----------|----------------------------|
| hto.anno | <i>Demultiplexing HTOs</i> |
|----------|----------------------------|

---

**Description**

Demultiplexing HTOs

**Usage**

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

**Arguments**

|                |  |
|----------------|--|
| hto.data       | HTO raw data   |
| cov.thr        | A number which average coverage is divided by to set a threshold for low coverage, default = 10. |
| assignment.thr | A percent above which you decide to set as a good sample assignment/HTO, default = 80.           |

**Value**

An object of class iCellR

**Examples**

```
my.hto <- read.table(file = system.file('extdata', 'dense_umis.tsv',
  package = 'iCellR'),
  as.is = TRUE)
head(my.hto)[1:5]

htos <- hto.anno(hto.data = my.hto)
head(htos)

boxplot(htos$percent.match)
```

---

|         |                                    |
|---------|------------------------------------|
| load10x | <i>Load 10X data as data.frame</i> |
|---------|------------------------------------|

---

**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**

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

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

---

`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 | <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

---

|           |                       |
|-----------|-----------------------|
| 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", "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 divided 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)
```

---

|              |  |
|--------------|--|
| opt.pcs.plot | <i>Find optimal number of PCs for clustering</i> |
|--------------|--|

---

**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**

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

---

|          |                         |
|----------|-------------------------|
| prep.vdj | <i>Prepare VDJ data</i> |
|----------|-------------------------|

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

An object of class iCellR

**Examples**

```
my.vdj <- read.csv(file = system.file('extdata', 'all_contig_annotations.csv',  
  package = 'iCellR'),  
  as.is = TRUE)  
head(my.vdj)  
dim(my.vdj)  
  
My.VDJ <- prep.vdj(vdj.data = my.vdj, cond.name = "NULL")  
head(My.VDJ)  
dim(My.VDJ)
```

---

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.

---

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

---

### 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 | <i>Run CCA on the main data</i> |
|---------|---------------------------------|

---

### 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 dimention 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 | <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", "ptbserial", "gap", "frey", "mcclain", "gamma", "gplus", "tau", "dunn", "hubert", "sindex", "dindex", "sdbw", "all" (all indices except GAP, Gamma, Gplus and Tau), "allong" (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)
```

---

|                   |  |
|-------------------|--|
| run.diffusion.map | <i>Run diffusion map 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 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 dimention 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   |

|                                   |  |
|-----------------------------------|--|
| <code>knn.dist.method</code>      | string, optional, default: 'euclidean'. recommended values: 'euclidean', 'cosine', 'precomputed' Any metric from <code>scipy.spatial.distance</code> can be used distance metric for building kNN graph. If 'precomputed', data should be an <code>n_samples x n_samples</code> 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 <code>data[0,0]</code> . You can override this detection with <code>knn.dist.method='precomputed_distance'</code> or <code>knn.dist.method='precomputed_affinity'</code> . |
| <code>init</code>                 | phate object, optional object to use for initialization. Avoids recomputing intermediate steps if parameters are the same.   |
| <code>mds.method</code>           | string, optional, default: 'metric' choose from 'classic', 'metric', and 'non-metric' which MDS algorithm is used for dimensionality reduction   |
| <code>mds.dist.method</code>      | string, optional, default: 'euclidean' recommended values: 'euclidean' and 'cosine'  |
| <code>t.max</code>                | int, optional, default: 100. Maximum value of t to test for automatic t selection.   |
| <code>npca</code>                 | int, optional, default: 100 Number of principal components to use for calculating neighborhoods. For extremely large datasets, using <code>n_pca &lt; 20</code> allows neighborhoods to be calculated in <code>log(n_samples)</code> time.   |
| <code>plot.optimal.t</code>       | boolean, optional, if TRUE, produce a plot showing the Von Neumann Entropy curve for automatic t selection.  |
| <code>verbose</code>              | int or boolean, optional (default : 1) If TRUE or > 0, message verbose updates.  |
| <code>n.jobs</code>               | 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 <code>n_jobs</code> below -1, <code>(n.cpus + 1 + n.jobs)</code> are used. Thus for <code>n_jobs = -2</code> , all CPUs but one are used   |
| <code>seed</code>                 | int or NULL, random state (default: NULL)  |
| <code>potential.method</code>     | Deprecated. For log potential, use <code>gamma=1</code> . For sqrt potential, use <code>gamma=0</code> .   |
| <code>use.alpha</code>            | Deprecated To disable alpha decay, use <code>alpha=NULL</code>   |
| <code>n.svd</code>                | Deprecated.  |
| <code>pca.method</code>           | Deprecated.  |
| <code>g.kernel</code>             | Deprecated.  |
| <code>diff.op</code>              | Deprecated.  |
| <code>landmark.transitions</code> | Deprecated.  |
| <code>diff.op.t</code>            | Deprecated.  |
| <code>dist.method</code>          | Deprecated.  |

**Value**

An object of class `iCellR`.

run.impute

*Impute the main data***Description**

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

**Usage**

```
run.impute(x = NULL, imp.method = "iCellR.imp", dims = 1:10,
  cell.ratio = 2, data.type = "pca", genes = "all_genes", k = 10,
  alpha = 15, t = "auto", npca = 100, init = NULL, t.max = 20,
  knn.dist.method = "euclidean", verbose = 1, n.jobs = 1,
  seed = NULL)
```

**Arguments**

|                 |  |
|-----------------|--|
| x               | An object of class iCellR.   |
| imp.method      | Choose between "iCellR.imp" and "magic", default = "iCellR.imp".   |
| dims            | PC dimentions to be used for the analysis, default = 10.   |
| cell.ratio      | Percent of cells to use to find neighboring cells, default = 2.  |
| data.type       | Choose between "tsne", "pca", "umap", "diffusion", default = "pca".  |
| genes           | character or integer vector, default: NULL vector of column names or column indices for which to return smoothed data If 'all_genes' or NULL, the entire smoothed matrix is returned               |
| k               | 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', message verbose updates.  |

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

**Value**

An object of class iCellR.

---

|             |   |
|-------------|---|
| run.pc.tsne | <i>Run tSNE on PCA Data. Barnes-Hut implementation of t-Distributed Stochastic Neighbor Embedding</i> |
|-------------|---|

---

**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 dimentionns 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 messageed (default: FALSE)   |
| is_distance         | logical; Indicate whether X is a distance matrix (experimental, default: FALSE)  |
| Y_init              | matrix; Initial locations of the objects. If NULL, random initialization will be used (default: NULL). Note that when using this, the initial stage with exaggerated perplexity values and a larger momentum term will be skipped. |
| pca_center          | logical; Should data be centered before pca is applied? (default: TRUE)  |
| pca_scale           | logical; Should data be scaled before pca is applied? (default: FALSE)   |
| stop_lying_iter     | integer; Iteration after which the perplexities are no longer exaggerated (default: 250, except when Y_init is used, then 0)   |
| mom_switch_iter     | integer; Iteration after which the final momentum is used (default: 250, except when Y_init is used, then 0)   |
| momentum            | numeric; Momentum used in the first part of the optimization (default: 0.5)  |
| final_momentum      | numeric; Momentum used in the final part of the optimization (default: 0.8)  |
| eta                 | numeric; Learning rate (default: 200.0)  |
| exaggeration_factor | numeric; Exaggeration factor used to multiply the P matrix in the first part of the optimization (default: 12.0)   |

**Value**

An object of class iCellR.

**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.phenograph      *Clustering the data*

---

**Description**

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

**Usage**

```
run.phenograph(x = NULL, k = 45, dims = 1:10)
```

**Arguments**

|      |  |
|------|--|
| x    | An object of class iCellR.                               |
| k    | integer; number of nearest neighbours (default:45)       |
| dims | PCA dimentions to be use for clustering, default = 1:10. |

**Value**

An object of class iCellR.

---

|          |  |
|----------|--|
| run.tsne | <i>Run tSNE on the Main Data. Barnes-Hut implementation of t-Distributed Stochastic Neighbor Embedding</i> |
|----------|--|

---

### 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", 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 messageed (default: FALSE)  |
| is_distance      | logical; Indicate whether X is a distance matrix (experimental, default: FALSE)   |

|                     |  |
|---------------------|--|
| Y_init              | matrix; Initial locations of the objects. If NULL, random initialization will be used (default: NULL). Note that when using this, the initial stage with exaggerated perplexity values and a larger momentum term will be skipped. |
| pca_center          | logical; Should data be centered before pca is applied? (default: TRUE)  |
| pca_scale           | logical; Should data be scaled before pca is applied? (default: FALSE)   |
| stop_lying_iter     | integer; Iteration after which the perplexities are no longer exaggerated (default: 250, except when Y_init is used, then 0)   |
| mom_switch_iter     | integer; Iteration after which the final momentum is used (default: 250, except when Y_init is used, then 0)   |
| momentum            | numeric; Momentum used in the first part of the optimization (default: 0.5)  |
| final_momentum      | numeric; Momentum used in the final part of the optimization (default: 0.8)  |
| eta                 | numeric; Learning rate (default: 200.0)  |
| exaggeration_factor | numeric; Exaggeration factor used to multiply the P matrix in the first part of the optimization (default: 12.0)   |

**Value**

An object of class iCellR.

**Examples**

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

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

---

|          |  |
|----------|--|
| 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 dimenitions 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 | <i>A dataset of S phase genes</i> |
|---------|-----------------------------------|

---

**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 | <i>Plot nGenes, UMIs and percent mito</i> |
|------------|---|

---

**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**

|                            |  |
|----------------------------|--|
| <code>x</code>             | An object of class <code>iCellR</code> .   |
| <code>topde</code>         | Number of top differentially expressed genes to be chosen from each cluster, default = 10.         |
| <code>min.base.mean</code> | Minimum base mean of the genes to be chosen, default = 0.5.  |
| <code>filt.ambig</code>    | Filter markers that are seen for more than one cluster, default = TRUE.                            |
| <code>cluster</code>       | 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(my.vdj = "data.frame")
```

**Arguments**

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

**Value**

An object of class `iCellR`

**Examples**

```
my.vdj <- read.csv(file = system.file('extdata', 'all_contig_annotations.csv',
  package = 'iCellR'),
  as.is = TRUE)
head(my.vdj)
dim(my.vdj)

My.VDJ <- prep.vdj(vdj.data = my.vdj, cond.name = "NULL")
head(My.VDJ)
```

```
dim(My.VDJ)
vdj.stats(My.VDJ)
```

---

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

## \*Topic **datasets**

- demo.obj, [14](#)
- g2m.phase, [17](#)
- s.phase, [41](#)

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

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

data.aggregation, [13](#)  
data.scale, [14](#)  
demo.obj, [14](#)  
down.sample, [15](#)

find.dim.genes, [15](#)  
findMarkers, [16](#)

g2m.phase, [17](#)  
gate.to.clust, [17](#)  
gene.plot, [18](#)  
gene.stats, [19](#)  
gg.cor, [20](#)

heatmap.gg.plot, [21](#)  
hto.anno, [22](#)

load10x, [22](#)

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

make.obj, [24](#)  
myImp, [25](#)

norm.adt, [26](#)  
norm.data, [26](#)

opt.pcs.plot, [27](#)

prep.vdj, [27](#)  
pseudotime, [28](#)  
pseudotime.tree, [29](#)

qc.stats, [30](#)

run.cca, [30](#)  
run.clustering, [31](#)  
run.diff.exp, [32](#)  
run.diffusion.map, [33](#)  
run.impute, [35](#)  
run.pc.tsne, [36](#)  
run.pca, [37](#)  
run.phenograph, [38](#)  
run.tsne, [39](#)  
run.umap, [40](#)

s.phase, [41](#)  
stats.plot, [41](#)

top.markers, [42](#)

vdj.stats, [43](#)  
volcano.ma.plot, [44](#)