Package ‘CytoTree’

April 10, 2023

Type Package

Title A Toolkit for Flow And Mass Cytometry Data

Version 1.8.0

Date 2020-06-19

Description A trajectory inference toolkit for flow and mass
cytometry data. CytoTree is a valuable tool to build a
tree-shaped trajectory using flow and mass cytometry data.
The application of CytoTree ranges from clustering
and dimensionality reduction to trajectory reconstruction
and pseudotime estimation. It offers complete analyzing
workflow for flow and mass cytometry data.

Depends R (>= 4.0), igraph

Imports FlowSOM, Rtsne, ggplot2, destiny, gmodels, flowUtils, Biobase,
Matrix, flowCore, sva, matrixStats, methods, mclust, prettydoc,
RANN(>= 2.5), Rcpp (>= 0.12.0), BiocNeighbors, cluster,
pheatmap, scatterpie, umap, scatterplot3d, limma, stringr,
grDevices, grid, stats

Suggests BiocGenerics, knitr, RColorBrewer, rmarkdown, testthat,
BiocStyle

biocViews CellBiology, Clustering, Visualization, Software,
CellBasedAssays, FlowCytometry, NetworkInference, Network

VignetteBuilder knitr

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

BugReports `https://github.com/JhuangLab/CytoTree/issues`

URL `http://www.r-project.org`, `https://github.com/JhuangLab/CytoTree`

LinkingTo Rcpp

PackageStatus Deprecated

git_url https://git.bioconductor.org/packages/CytoTree

git_branch RELEASE_3_16

git_last_commit 1802ae8

git_last_commit_date 2022-11-01

Date/Publication 2023-04-10

Author Yuting Dai [aut, cre]

Maintainer Yuting Dai <forlynna@sjtu.edu.cn>

R topics documented:

CytoTree-package . . . 3
buildTree . . . 4
constraintMatrix . . . 5
correctBatchCYT . . . 6
createCYT . . . 7
CYT-class . . . 8
defLeafCells . . . 9
defRootCells . . . 10
fetchCell . . . 11
fetchClustMeta . . . 12
fetchPlotMeta . . . 12
find_neighbors . . . 13
gatingMatrix . . . 14
plot2D . . . 15
plot3D . . . 17
plotBranchHeatmap . . . 18
plotCluster . . . 19
plotClusterHeatmap . . . 20
plotHeatmap . . . 21
plotMarkerDensity . . . 22
plotPieCluster . . . 23
plotPieTree . . . 24
plotPseudotimeDensity . . . 25
plotPseudotimeTraj . . . 26
plotTrajHeatmap . . . 27
plotTree . . . 28
plotViolin . . . 29
processingCluster . . . 30
Rphenograph . . . 31
runClara . . . 33
runCluster . . . 34
runDiff . . . 35
runDiffusionMap . . . 36
runExprsExtract . . . 37
runExprsMerge . . . 39
runFastPCA . . . 40

runHclust 41
runKmeans 42
runKNN 43
runMclust 44
runPhenograph 45
runPseudotime 46
runSOM 47
runTSNE 48
runUMAP 50
runWalk 51
subsetCYT 52
updateClustMeta 53
updatePlotMeta 53

Index **55**

`CytoTree-package` *Visualization and analyzation for flow cytometry data*

Description

Functions and methods to visualize and analyze flow cytometry data.

Details

Package:	CytoTree
Type:	Package
Version:	0.99.6
Date:	2020-06-19
License:	GPL-3.0

While high-dimensional single-cell based flow and mass cytometry data has demonstrated increased applications in microenvironment composition and stem-cell research, integrated analyzing workflow design for experimental cytometry data has been challenging. Here, we present CytoTree, an R package designed for the analysis and interpretation of flow and mass cytometry data. We have applied CytoTree to mass cytometry and time course flow cytometry data to validate the usage and practical utility of its computational modules. These use cases introduce CytoTree as a reliable tool for high-dimensional cytometry data workflow and reveal good performance on trajectory reconstruction and pseudotime estimation.

Author(s)

Maintainer: Yuting Dai <forlynna@sjtu.edu.cn> Authors: Yuting Dai

Examples

```
if (FALSE) {
```

```
## examples go here
## See vignette tutorials
vignette(package = "CytoTree")
}
```

---

buildTree *buildTree*

Description

buildTree

Usage

```
buildTree(
  object,
  method = "euclidean",
  dim.type = c("raw", "pca", "tsne", "dc", "umap"),
  dim.use = seq_len(2),
  verbose = FALSE
)
```

Arguments

`object`	an CYT object
`method`	character. Mehtod to build MST.
`dim.type`	character. Type of dimensions that will be used to build the tree. Five `dim.type` are provided, 'raw', 'pca', 'tsne', 'dc' and 'umap'. By default is 'raw'.
`dim.use`	numeric. Number of dimensions that will be used to build the tree. For example. If `dim.use` is 'raw', there is no limit for `dim.type`. And if the `dim.use` is 'tsne' or 'umap', the default `dim.use` is seq_len(2).
`verbose`	logical. Whether to print calculation progress.

Value

An CYT object with tree

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- buildTree(cyt, dim.type = "raw")

# build minimum spanning tree (MST) based on tsne
cyt <- buildTree(cyt, dim.type = "tsne", dim.use = seq_len(2))
```

```
# Using PCA
cyt <- buildTree(cyt, dim.type = "pca", dim.use =seq_len(4))

# Using UMAP
cyt <- buildTree(cyt, dim.type = "umap", dim.use = seq_len(2))

# Using Diffusion Maps
cyt <- buildTree(cyt, dim.type = "dc", dim.use = seq_len(3))
```

`constraintMatrix` *constraintMatrix*

Description

constraint FCS data by a provid cutoff

Usage

```
constraintMatrix(x, cutoff = 0.99, markers = NULL, method = "euclidean")
```

Arguments

`x`	matrix
`cutoff`	numeric. Cutoff of the constraint value
`markers`	character. Markers used in the calculation of constraint model.
`method`	character. the distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".

Value

a matrix

Examples

```
mat <- matrix(runif(10000), nrow = 1000, ncol = 10)
colnames(mat) <- LETTERS[ seq_len(10)]
dim(mat)

mat <- constraintMatrix(mat)
dim(mat)
```

correctBatchCYT *correctBatchCYT*

Description

Remove batch effect in CYT object

Usage

```
correctBatchCYT(
  object,
  batch = NULL,
  par.prior = TRUE,
  mean.only = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	An CYT object
`batch`	vector. Batch covariate (only one batch allowed)
`par.prior`	logical. TRUE indicates parametric adjustments will be used, FALSE indicates non-parametric adjustments will be used.
`mean.only`	logical. FALSE If TRUE ComBat only corrects the mean of the batch effect (no scale adjustment)
`verbose`	logical. Whether to show log information
`...`	Parameters passing to `ComBat` function

Value

An CYT object after removing batch effect

An CYT object with corrected batch effects

See Also

`findKNN`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)
plot.meta <- fetchPlotMeta(cyt)
batch <- as.numeric(plot.meta$stage)
cyt <- correctBatchCYT(cyt, batch = batch)
```

`createCYT` *create an CYT object*

Description

This function is about how to build an CYT object. An CYT object is the base for the whole analysizing workflow of flow and mass cytometry data.

Usage

```
createCYT(
  raw.data,
  markers,
  meta.data,
  batch = NULL,
  batch.correct = FALSE,
  normalization.method = "none",
  verbose = FALSE,
  ...
)
```

Arguments

`raw.data`	matrix. Raw data read from FCS file after perform preprocessing.
`markers`	vector. Detailed marker information in the gate of flow cytometer.
`meta.data`	data.frame. Raw metadata of each cell. Columns "cell" and "stage" are required.
`batch`	vector. Batch covariate (only one batch allowed). Method to correct batch effect function is refered to `ComBat`.
`batch.correct`	logical. Whether to correct batch effect. If TRUE, batch must be provided.
`normalization.method`	character. Normalization and transformation method. Whether to normalize and log transformed of raw.data. In CytoTree workflow, it's better to perform transformation of FCS data using `runExprsExtract` or `runExprsMerge` before creating an CYT object. `CytoTree` only provide log transforma method. If you need to using truncateTransform, scaleTransform, linearTransform, quadraticTransform and lnTransform, see `flowCore` for more information. And `runExprsExtract` in `CytoTree`, autoLgcl, cytofAsinh, logicle, arcsinh, and logAbs can be used to perform transformation of FCS data.
`verbose`	logical. Whether to print calculation progress.
`...`	paramters pass to `correctBatchCYT` function.

Value

An CYT object with raw.data and markers and meta.data

Examples

```
# Read fcs files
fcs.path <- system.file("extdata", package = "CytoTree")
fcs.files <- list.files(fcs.path, pattern = '.FCS$', full = TRUE)

fcs.data <- runExprsMerge(fcs.files, comp = FALSE, transformMethod = "none")

# Refine colnames of fcs data
recol <- c(`FITC-A<CD43>` = "CD43", `APC-A<CD34>` = "CD34",
           `BV421-A<CD90>` = "CD90", `BV510-A<CD45RA>` = "CD45RA",
           `BV605-A<CD31>` = "CD31", `BV650-A<CD49f>` = "CD49f",
           `BV 735-A<CD73>` = "CD73", `BV786-A<CD45>` = "CD45",
           `PE-A<FLK1>` = "FLK1", `PE-Cy7-A<CD38>` = "CD38")
colnames(fcs.data)[match(names(recol), colnames(fcs.data))] = recol
fcs.data <- fcs.data[, recol]

day.list <- c("D0", "D2", "D4", "D6", "D8", "D10")
meta.data <- data.frame(cell = rownames(fcs.data),
                        stage = gsub(".FCS.+", "", rownames(fcs.data) ) )
meta.data$stage <- factor(as.character(meta.data$stage), levels = day.list)

markers <- c("CD43","CD34","CD90","CD45RA","CD31","CD49f","CD73","CD45","FLK1","CD38")

# Build the CYT object
cyt <- createCYT(raw.data = fcs.data, markers = markers,
                 meta.data = meta.data,
                 normalization.method = "log",
                 verbose = TRUE)

# See information
cyt
```

`CYT-class` *Class* `CYT`

Description

All information stored in CYT object. You can use `creatCYT` to create an CYT object. In this package, most of the functions will use CYT object as input, and return a modified CYT obejct as well.

Slots

`raw.data` matrix. Raw signal data captured in flow or mass cytometry.

`log.data` matrix. Log-transfromed dataset of raw.data.

`meta.data` data.frame. Meta data information, and colnames of "stage" and "cell" are required.

`markers` vector. Markers used in the calculation of PCA, tSNE, diffusion map and UMAP.

`markers.idx` vector. Index of markers used in the calculation of PCA, tSNE, destiny and umap.

`cell.name` vector. Cell names after performing downsampling.

`knn` numeric. Numbers of nearest neighbors

`knn.index,knn.distance` matrix. Each row of the `knn.index` matrix corresponds to a point in `log.data` and contains the row indices in `log.data` that are its nearest neighbors. And each row of the `knn.distance` contains the distance of its nearest neighbors.

`som` list. Store som network information calculated using `FlowSOM`.

`cluster` data.frame. Cluster information

`pca.sdev,pca.value,pca.scores` PCA information of CYT object which are generated from `fast.prcomp`.

`tsne.value` matrix. tSNE coordinates information. See `Rtsne`.

`dm` DiffusionMap object. Diffusion map calculated by package `destiny`

`umap.value` matrix umap coordinates information calculated using `umap`.

`root.cells` vector, Names of root cells, which can be modified by `defRootCells`. An root cell is manually set to be the origin of all cells. Pseudotime in root cells are the lowest.

`leaf.cells` vector. Names of leaf cells, which can be modified by `defLeafCells`. An leaf cell is manually set to be the terminal state of all cells. Pseuodtime in leaf cells are the largest.

`network` list. Network stored in the calculation of trajectory and pseudotime.

`walk` list. Random forward and backward walk between `root.cells` and `leaf.cells`.

`diff.traj` list. Differentiation trajectory all cells.

`plot.meta` data.frame. Plot meta information for `plot2D` or `plot3D`.

`tree.meta` data.frame. Tree meta information of CYT object.

`defLeafCells` *definition of leaf cells*

Description

definition of root cells

Usage

```
defLeafCells(object, leaf.cells = NULL, pseudotime.cutoff = 0, verbose = FALSE)
```

Arguments

`object`	an CYT object
`leaf.cells`	character or numeric. Cell name of the root cells or cluster.id of root.cells
`pseudotime.cutoff`	numeric. Cutoff of pseudotime. Cells with pseudotime over pseudotime.cutoff will be set to be leaf cells
`verbose`	logical. Whether to print calculation progress.

Value

An CYT object

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

# Define leaf cells by cluster
cyt <- defLeafCells(cyt, leaf.cells = 1, verbose = TRUE)
cyt <- defLeafCells(cyt, leaf.cells = c(1,3), verbose = TRUE)

# Define root cells by cell names
meta.data <- fetchPlotMeta(cyt)
cells <- meta.data$cell[which(meta.data$stage == "D10")]
cells <- as.character(cells)
cyt <- defLeafCells(cyt, leaf.cells = cells, verbose = TRUE)
```

`defRootCells` *definition of root cells*

Description

definition of root cells

Usage

```
defRootCells(object, root.cells = NULL, verbose = FALSE)
```

Arguments

`object`	an CYT object
`root.cells`	vector. Cell name of the root cells
`verbose`	logical. Whether to print calculation progress.

Value

An CYT object

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

# Define root cells by cluster
cyt <- defRootCells(cyt, root.cells = 6, verbose = TRUE)
cyt <- defRootCells(cyt, root.cells = c(6,8), verbose = TRUE)

# Define root cells by cell names
meta.data <- fetchPlotMeta(cyt)
cells <- meta.data$cell[which(meta.data$stage == "D0")]
cells <- as.character(cells)
cyt <- defRootCells(cyt, root.cells = cells, verbose = TRUE)
```

`fetchCell` *Fetching cellls of CYT*

Description

Fetching cellls of CYT

Usage

```
fetchCell(object, logical.connect = "or", verbose = FALSE, ...)
```

Arguments

`object`	An CYT object
`logical.connect`	character. "and" or "or"
`verbose`	logical. Whether to print calculation progress.
`...`	Paramters to pass to limitation

Value

a vector containing cell names

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cell.fetch <- fetchCell(cyt, traj.value.log = 0.01)
cell.fetch <- fetchCell(cyt, stage = c("D0", "D10"))
cell.fetch <- fetchCell(cyt, stage = c("D0", "D10"), traj.value.log = 0.01,
                    logical.connect = "or")
```

fetchClustMeta   *Fetching clusters' metadata of CYT*

Description

Fetching clusters' metadata of CYT

Usage

```
fetchClustMeta(object, verbose = FALSE)
```

Arguments

`object`	An CYT object
`verbose`	logical. Whether to print calculation progress.

Value

a data.frame containing clustering information for visualization

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

clust.data <- fetchClustMeta(cyt)
head(clust.data)
```

fetchPlotMeta   *Fetching plot metadata of CYT*

Description

Fetching plot metadata of CYT

Usage

```
fetchPlotMeta(object, markers = NULL, verbose = FALSE)
```

Arguments

`object`	An CYT object
`markers`	vector. Makers fetched from expression matrix
`verbose`	logical. Whether to print calculation progress.

Value

a data.frame containing meta information for visualization

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plot.data <- fetchPlotMeta(cyt)
head(plot.data)

plot.data <- fetchPlotMeta(cyt, markers = c("CD43", "CD34"))
head(plot.data)
```

`find_neighbors` *K Nearest Neighbour Search*

Description

Uses a kd-tree to find the p number of near neighbours for each point in an input/output dataset. Use the nn2 function from the RANN package, utilizes the Approximate Near Neighbor (ANN) C++ library, which can give the exact near neighbours or (as the name suggests) approximate near neighbours to within a specified error bound. For more information on the ANN library please visit http://www.cs.umd.edu/~mount/ANN/.

Usage

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

Arguments

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

Value

a n-by-k matrix of neighbor indices

Author(s)

Hao Chen <chen_hao@immunol.a-star.edu.sg>

Examples

```
iris_unique <- unique(iris) # Remove duplicates
data <- as.matrix(iris_unique[, seq_len(4)])
neighbors <- find_neighbors(data, k=10)
```

`gatingMatrix` *Apply gating on the matrix data*

Description

Apply gating on the matrix data

Usage

```
gatingMatrix(x, lower.gate = NULL, upper.gate = NULL)
```

Arguments

`x`	matrix
`lower.gate`	vector. Gating parameter, the name of the vector is the marker name, and the value of the vector is the lower bound of gating cutoff.
`upper.gate`	vector. Gating parameter, the name of the vector is the marker name, and the value of the vector is the upper bound of gating cutoff.

Value

a matrix

Examples

```
par(mfrow=c(1,2))
x <- matrix(rnorm(200, 3, 1), nrow = 100, ncol = 2)
colnames(x) <- c("CD34", "CD43")
plot(x[, "CD34"], x[, "CD43"], main = "Before gating")

lower.gate = c(CD34 = 2, CD43 = 3)
upper.gate = c(CD34 = 4, CD43 = 5)

x <- gatingMatrix(x, lower.gate = lower.gate, upper.gate = upper.gate)
plot(x[, "CD34"], x[, "CD43"], main = "After gating")

par(mfrow=c(1,1))
```

plot2D *Visualization of 2D data of CYT*

Description

Visualization of 2D data of CYT

Usage

```
plot2D(
  object,
  item.use = c("PC_1", "PC_2"),
  color.by = "stage",
  order.by = NULL,
  size = 1,
  alpha = 1,
  category = "categorical",
  show.cluser.id = FALSE,
  show.cluser.id.size = 4,
  main = "2D plot of CYT",
  plot.theme = theme_bw()
)
```

Arguments

`object`	An CYT object
`item.use`	character. Items use to 2D plot, axes x and y must be numeric.
`color.by`	character. Dot or mesh color by which character. It can be one of the column of plot.meta, or it can be just "density" (the default value).
`order.by`	vector. Order of color theme.
`size`	numeric. Size of the dot
`alpha`	numberic. Transparency (0-1) of the dot, default is 1.
`category`	character. numeric or categorical
`show.cluser.id`	logical. Whether to show cluster id in the plot.
`show.cluser.id.size`	numeric. Size of the cluster id.
`main`	character. Title of the plot.
`plot.theme`	themes from `ggplot2`

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

# Default plot
plot2D(cyt)

# PCA plot
plot2D(cyt, item.use = c("PC_1", "PC_2"))
plot2D(cyt, item.use = c("PC_1", "PC_2"), color.by = "cluster.id")
plot2D(cyt, item.use = c("PC_1", "PC_2"), color.by = "stage")
plot2D(cyt, item.use = c("PC_2", "PC_3"), color.by = "stage")
plot2D(cyt, item.use = c("PC_2", "PC_3"), color.by = "CD43",
       category = "numeric")
plot2D(cyt, item.use = c("PC_2", "PC_3"), color.by = "CD43",
       category = "numeric")

# tSNE plot
plot2D(cyt, item.use = c("tSNE_1", "tSNE_2"))
plot2D(cyt, item.use = c("tSNE_1", "tSNE_2"), color.by = "stage")
plot2D(cyt, item.use = c("tSNE_1", "tSNE_2"), color.by = "cluster.id",
       alpha = 0.5, main = "tSNE Plot")
plot2D(cyt, item.use = c("tSNE_1", "tSNE_2"), color.by = "cluster.id",
       alpha = 1, main = "tSNE Plot", show.cluser.id = TRUE)
plot2D(cyt, item.use = c("tSNE_1", "tSNE_2"), color.by = "CD43",
       category = "numeric", size = 3)
plot2D(cyt, item.use = c("tSNE_1", "tSNE_2"), color.by = "stage")

# Diffusion Map plot
plot2D(cyt, item.use = c("DC_1", "DC_2"))
plot2D(cyt, item.use = c("DC_1", "DC_2"), color.by = "stage")
plot2D(cyt, item.use = c("DC_2", "DC_3"), color.by = "cluster.id",
       alpha = 0.5, main = "Diffusion Map Plot")
plot2D(cyt, item.use = c("DC_2", "DC_3"), color.by = "cluster.id",
       alpha = 1, main = "Diffusion Map Plot", show.cluser.id = TRUE)
plot2D(cyt, item.use = c("DC_1", "DC_2"), color.by = "CD43",
       category = "numeric", size = 3)

# UMAP plot
plot2D(cyt, item.use = c("UMAP_1", "UMAP_2"))
plot2D(cyt, item.use = c("UMAP_1", "UMAP_2"), color.by = "stage")
plot2D(cyt, item.use = c("UMAP_1", "UMAP_2"), color.by = "cluster.id",
       alpha = 0.5, main = "UMAP Plot")
plot2D(cyt, item.use = c("UMAP_1", "UMAP_2"), color.by = "cluster.id",
       alpha = 1, main = "UMAP Plot", show.cluser.id = TRUE)
plot2D(cyt, item.use = c("UMAP_1", "UMAP_2"), color.by = "CD43",
       category = "numeric", size = 3)
plot2D(cyt, item.use = c("UMAP_1", "UMAP_2"), color.by = "stage")

# Marker Plot
plot2D(cyt, item.use = c("CD43", "CD90"), color.by = "cluster.id")
plot2D(cyt, item.use = c("CD34", "CD90"), color.by = "CD43",
        category = "numeric", size = 3)

# Pseudotime
plot2D(cyt, item.use = c("pseudotime", "CD43"), color.by = "stage")
```

---

plot3D *Visualization of 3D data of CYT*

Description

Visualization of 3D data of CYT

Usage

```
plot3D(
  object,
  item.use = c("PC1", "PC2", "PC3"),
  color.by = "stage",
  order.by = NULL,
  size = 1,
  angle = 60,
  scale.y = 0.8,
  category = "categorical",
  main = "3D plot of CYT",
  color.theme = NULL,
  ...
)
```

Arguments

`object`	An CYT object
`item.use`	character. Items use to 3D plot, axes x and y and z must be numeric.
`color.by`	character. Dot or mesh color by which character. It can be one of the column of plot.meta, or it can be just "density" (the default value).
`order.by`	character. Order of color theme.
`size`	numeric. size of the dot
`angle`	numberic. angle of the plot
`scale.y`	numeric. scale of y axis related to x- and z axis
`category`	character. numeric or categorical
`main`	character. title of the plot
`color.theme`	vector. Color themes use in the plot.
`...`	options to pass on to the `scatterplot3d` function.

Value

gplots figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plot3D(cyt, item.use = c("DC_2","DC_1","DC_3"), color.by = "stage",
       size = 0.5, angle = 60, color.theme = c("#FF99FF","#7A06A0","#FF3222"))
```

`plotBranchHeatmap` *Visualization heatmap of branch data of CYT*

Description

Visualization heatmap of branch data of CYT

Usage

```
plotBranchHeatmap(
  object,
  color = colorRampPalette(c("blue", "white", "red"))(100),
  scale = "row",
  ...
)
```

Arguments

`object`	An CYT object
`color`	vector. Colors used in heatmap.
`scale`	character. Whether the values should be centered and scaled in either the row direction or the column direction, or none. Corresponding values are "row", "column" and "none"
`...`	options to pass on to the `pheatmap` function.

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plotBranchHeatmap(cyt)
plotBranchHeatmap(cyt, color = colorRampPalette(c("purple","white","yellow"))(100))
plotBranchHeatmap(cyt, cluster_row = FALSE)
plotBranchHeatmap(cyt, cluster_row = FALSE, cluster_col = FALSE)
```

---

plotCluster *Visualization of cluster data of CYT*

Description

Visualization of cluster data of CYT

Usage

```
plotCluster(
  object,
  item.use = c("PC_1", "PC_2"),
  color.by = "cluster",
  size.by = "cell.number.percent",
  order.by = NULL,
  size = 1,
  alpha = 1,
  category = "categorical",
  show.cluser.id = FALSE,
  show.cluser.id.size = 4,
  main = "2D plot of cluster in CYT",
  plot.theme = theme_bw()
)
```

Arguments

`object`	An CYT object
`item.use`	character. Items use to 2D plot, axes x and y must be numeric.
`color.by`	character. Dot or mesh color by which character. It can be one of the column of plot.meta, or it can be just "density" (the default value).
`size.by`	character. Size of the dot
`order.by`	vector. Order of color theme.
`size`	numeric. Size of the dot
`alpha`	numberic. Transparency (0-1) of the dot, default is 1.

`category`	character. numeric or categorical
`show.cluser.id`	logical. Whether to show cluster id in the plot.
`show.cluser.id.size`	numeric. Size of the cluster id.
`main`	character. Title of the plot.
`plot.theme`	themes from `ggplot2`

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plotCluster(cyt)

plotCluster(cyt, item.use = c("PC_1", "PC_2"))
plotCluster(cyt, item.use = c("PC_2", "PC_3"))
plotCluster(cyt, item.use = c("PC_2", "PC_3"), color.by = "CD43", category = "numeric")
plotCluster(cyt, item.use = c("PC_2", "PC_3"), color.by = "CD43", category = "numeric")

plotCluster(cyt, item.use = c("tSNE_1", "tSNE_2"))
plotCluster(cyt, item.use = c("tSNE_1", "tSNE_2"), show.cluser.id = TRUE)

plotCluster(cyt, item.use = c("DC_1", "DC_2"))

plotCluster(cyt, item.use = c("UMAP_1", "UMAP_2"))
```

`plotClusterHeatmap` *Visualization heatmap of cluster data of CYT*

Description

Visualization heatmap of cluster data of CYT

Usage

```
plotClusterHeatmap(
  object,
  color = colorRampPalette(c("blue", "white", "red"))(100),
  scale = "row",
  ...
)
```

Arguments

`object`	An CYT object
`color`	vector. Colors used in heatmap.
`scale`	character. Whether the values should be centered and scaled in either the row direction or the column direction, or none. Corresponding values are "row", "column" and "none"
`...`	options to pass on to the `pheatmap` function.

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plotClusterHeatmap(cyt)
plotClusterHeatmap(cyt, color = colorRampPalette(c("purple","white","yellow"))(100))
plotClusterHeatmap(cyt, cluster_row = FALSE)
plotClusterHeatmap(cyt, cluster_row = FALSE, cluster_col = FALSE)
```

`plotHeatmap` *Visualization heatmap of data of CYT*

Description

Visualization heatmap of data of CYT

Usage

```
plotHeatmap(
  object,
  markers = NULL,
  color = colorRampPalette(c("blue", "white", "red"))(100),
  scale = "row",
  downsize = 1000,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  ...
)
```

Arguments

`object`	An CYT object
`markers`	vector. markers to plot on the heatmap
`color`	vector. Colors used in heatmap.
`scale`	character. Whether the values should be centered and scaled in either the row direction or the column direction, or none. Corresponding values are "row", "column" and "none"
`downsize`	numeric. Cells size used to plot heatmap
`cluster_rows`	logical. Whether rows should be clustered
`cluster_cols`	logical. Whether columns should be clustered
`...`	options to pass on to the `pheatmap` function.

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plotHeatmap(cyt)
plotHeatmap(cyt, cluster_rows = TRUE)
plotHeatmap(cyt, cluster_rows = TRUE, clustering_method = "ward.D")
plotHeatmap(cyt, cluster_rows = TRUE, cluster_cols = TRUE)
```

`plotMarkerDensity` *plotMarkerDensity*

Description

plotMarkerDensity

Usage

```
plotMarkerDensity(
  object,
  cutoff = -1,
  markers = NULL,
  adjust = 0.5,
  plot.theme = theme_bw()
)
```

Arguments

`object`	An CYT object
`cutoff`	numeric. Cutoff of trajectory value
`markers`	character. Markers used in the calculation progress
`adjust`	numeric. Transparency (0-1) of the dot, default is 1.
`plot.theme`	themes from `ggplot2`

Value

ggplot2 figure

Examples

```
if (FALSE) {

plotMarkerDensity(cyt)
plotMarkerDensity(cyt, adjust = 1)

}
```

`plotPieCluster` *Visualization pie plot of cluster data of CYT*

Description

Visualization pie plot of cluster data of CYT

Usage

```
plotPieCluster(
  object,
  item.use = c("PC_1", "PC_2"),
  cex.size = 1,
  size.by.cell.number = TRUE,
  main = "2D pie plot of CYT",
  plot.theme = theme_bw()
)
```

Arguments

`object`	An CYT object
`item.use`	character. Items use to 2D plot, axes x and y must be numeric.
`cex.size`	numeric. Size of the dot
`size.by.cell.number`	logical. Whether to show size of cell number.
`main`	character. Title of the plot.
`plot.theme`	themes from `ggplot2`

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

# Runs only have more than two stages
plotPieCluster(cyt, cex.size = 0.5)

plotPieCluster(cyt, item.use = c("PC_1", "PC_2"), cex.size = 0.5)
plotPieCluster(cyt, item.use = c("PC_2", "PC_3"), cex.size = 0.5)

plotPieCluster(cyt, item.use = c("tSNE_1", "tSNE_2"), cex.size = 20)

plotPieCluster(cyt, item.use = c("DC_1", "DC_2"), cex.size = 0.5)

plotPieCluster(cyt, item.use = c("UMAP_1", "UMAP_2"), cex.size = 1)
plotPieCluster(cyt, item.use = c("UMAP_1", "UMAP_2"), cex.size = 1)
```

`plotPieTree` *plot MST pie of CYT*

Description

plot MST pie of CYT

Usage

```
plotPieTree(
  object,
  cex.size = 2,
  size.by.cell.number = TRUE,
  as.tree = FALSE,
  root.id = NULL,
  show.node.name = FALSE
)
```

Arguments

`object`	an CYT object
`cex.size`	numeric. size cex of the dot
`size.by.cell.number`	logical. Whether to size node by cell number
`as.tree`	logical. Whether to show node as tree

`root.id`	numeric. Root id of the tree, if as.tree is TRUE
`show.node.name`	logical. whether to show node name

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

# Runs only have two or more stages
plotPieTree(cyt, cex.size = 1, size.by.cell.number = TRUE)
```

`plotPseudotimeDensity` *plot Pseudotime density of CYT*

Description

plot Pseudotime density of CYT

Usage

```
plotPseudotimeDensity(
  object,
  color.by = "stage",
  main = "Density of pseudotime",
  adjust = 0.5,
  plot.theme = theme_bw()
)
```

Arguments

`object`	an CYT object
`color.by`	character.
`main`	character. Title of the plot
`adjust`	numeric. A multiplicate bandwidth adjustment.
`plot.theme`	themes from `ggplot2`

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plotPseudotimeDensity(cyt)

plotPseudotimeDensity(cyt, adjust = 1)
plotPseudotimeDensity(cyt, adjust = 2)
```

`plotPseudotimeTraj` *plotPseudotimeTraj*

Description

plotPseudotimeTraj

Usage

```
plotPseudotimeTraj(
  object,
  cutoff = -1,
  markers = NULL,
  size = 0.5,
  alpha = 0.6,
  print.curve = TRUE,
  var.cols = FALSE,
  plot.theme = theme_bw()
)
```

Arguments

`object`	An CYT object
`cutoff`	numeric. Cutoff of trajectory value
`markers`	character. Markers used in the calculation progress
`size`	numeric. Size of the dot
`alpha`	numeric. Transparency (0-1) of the dot, default is 1.
`print.curve`	logical. Whether to perform curve fitting
`var.cols`	logical. Whether to plot stage
`plot.theme`	themes from `ggplot2`

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plotPseudotimeTraj(cyt)
plotPseudotimeTraj(cyt, print.curve = FALSE)
plotPseudotimeTraj(cyt, var.cols = TRUE)

plotPseudotimeTraj(cyt, markers = c("CD43", "CD34"))
```

`plotTrajHeatmap` *Visualization heatmap of intermediate cells of CYT*

Description

Visualization heatmap of intermediate cells of CYT

Usage

```
plotTrajHeatmap(
  object,
  cutoff = 0,
  markers = NULL,
  color = colorRampPalette(c("blue", "white", "red"))(100),
  scale = "row",
  ...
)
```

Arguments

`object`	An CYT object
`cutoff`	numeric. value to identify intermediate state cells
`markers`	markers to plot on the heatmap
`color`	vector. Colors used in heatmap.
`scale`	character. Whether the values should be centered and scaled in either the row direction or the column direction, or none. Corresponding values are "row", "column" and "none"
`...`	options to pass on to the `pheatmap` function.

Value

ggplot2 figure

`plotTree` *plot MST of CYT*

Description

plot MST of CYT

Usage

```
plotTree(
  object,
  cex.size = 1,
  color.by = "cell.number",
  size.by = "cell.number",
  as.tree = FALSE,
  root.id = NULL,
  show.node.name = FALSE
)
```

Arguments

`object`	an CYT object
`cex.size`	numeric. size cex of the dot
`color.by`	numeric. size color theme of the dot
`size.by`	numeric. size theme of the dot
`as.tree`	logical. Whether to show node as tree
`root.id`	numeric. Root id of the tree, if as.tree is TRUE
`show.node.name`	logical. whether to show node name

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plotTree(cyt)

plotTree(cyt, show.node.name = TRUE)

plotTree(cyt, color.by = "CD43", show.node.name = TRUE, cex.size = 1)

plotTree(cyt, color.by = "D0.percent", show.node.name = TRUE, cex.size = 1)
plotTree(cyt, color.by = "D2.percent", show.node.name = TRUE, cex.size = 1)

plotTree(cyt, color.by = "pseudotime", cex.size = 1)
```

---

plotViolin *Visualization violin plot of CYT*

Description

Visualization violin plot of CYT

Usage

```
plotViolin(
  object,
  marker,
  color.by = "cluster.id",
  order.by = NULL,
  size = 1,
  text.angle = 0,
  main = "Violin plot CYT",
  plot.theme = theme_bw()
)
```

Arguments

`object`	An CYT object
`marker`	character. Markers used to plot
`color.by`	character. Dot or mesh color by which character. It can be one of the column of plot.meta, or it can be just "density" (the default value).
`order.by`	vector. Order of color theme.
`size`	numeric. Size of the dot
`text.angle`	numberic. Text angle of the violin plot
`main`	character. Title of the plot.
`plot.theme`	themes from `ggplot2`

Value

ggplot2 figure

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

plotViolin(cyt, marker = "CD34")
plotViolin(cyt, marker = "CD34", order.by = "pseudotime")
```

processingCluster *processingCluster*

Description

Calculate Principal Components Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (tSNE), Diffusion Map and Uniform Manifold Approximation and Projection (UMAP) of clusters calculated by runCluster.

Usage

```
processingCluster(
  object,
  perplexity = 5,
  k = 5,
  downsampling.size = 1,
  force.resample = TRUE,
  random.cluster = FALSE,
  umap.config = umap.defaults,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	an CYT object
`perplexity`	numeric. Perplexity parameter (should not be bigger than 3 * perplexity < nrow(X) - 1, see details for interpretation). See `Rtsne` for more information.
`k`	numeric. The parameter k in k-Nearest Neighbor.
`downsampling.size`	numeric. Percentage of sample size of downsampling. This parameter is from 0 to 1. by default is 1.
`force.resample`	logical. Whether to do resample if downsampling.size < 1
`random.cluster`	logical. Whether to perfrom random downsampling. If FALSE, an uniform downsampling will be processed.
`umap.config`	object of class umap.config. See `umap`.
`verbose`	logic. Whether to print calculation progress.
`...`	options to pass on to the dimensionality reduction functions.

Value

An CYT object with cluster.id in meta.data

An CYT object with dimensionality reduction of clusters

See Also

`umap`, `fast.prcomp`, `Rtsne`, `destiny`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

# After running clustering
set.seed(1)
cyt <- runCluster(cyt, cluster.method = "som", xdim = 3, ydim = 3, verbose = TRUE)

# Do not perfrom downsampling
cyt <- processingCluster(cyt, perplexity = 2)

# Perform cluster based downsampling
# Only keep 50% cells
cyt <- processingCluster(cyt, perplexity = 2, downsampling.size = 0.5)

# Processing clusters without downsampling step
cyt <- processingCluster(cyt, perplexity = 2, force.resample = FALSE)
```

`Rphenograph` *RphenoGraph clustering*

Description

R implementation of the PhenoGraph algorithm

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

This function is developed by Hao Chen and updated by Yuting Dai.

Usage

```
Rphenograph(data, k = 30)
```

Arguments

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

Value

a list contains an igraph graph object for `graph_from_data_frame` and a communities object, the operations of this class contains:

`print`	returns the communities object itself, invisibly.
`length`	returns an integer scalar.
`sizes`	returns a numeric vector.
`membership`	returns a numeric vector, one number for each vertex in the graph that was the input of the community detection.
`modularity`	returns a numeric scalar.
`algorithm`	returns a character scalar.
`crossing`	returns a logical vector.
`is_hierarchical`	returns a logical scalar.
`merges`	returns a two-column numeric matrix.
`cut_at`	returns a numeric vector, the membership vector of the vertices.
`as.dendrogram`	returns a dendrogram object.
`show_trace`	returns a character vector.
`code_len`	returns a numeric scalar for communities found with the InfoMAP method and NULL for other methods.
`plot`	for communities objects returns NULL, invisibly.

cluster information

Author(s)

Hao Chen <chen_hao@immunol.a-star.edu.sg>

References

Jacob H. Levine and et.al. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. Cell, 2015.

Examples

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

runClara *runClara*

Description

Clustering a data matrix into k clusters

Usage

```
runClara(
  object,
  k = 25,
  metric = c("euclidean", "manhattan", "jaccard"),
  stand = FALSE,
  samples = 5,
  scale = TRUE,
  trace = 0,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	an CYT object
`k`	numeric. The number of clusters. It is required that 0 < k < n where n is the number of observations (i.e., n = nrow(x)).
`metric`	character. string specifying the metric to be used for calculating dissimilarities between observations.
`stand`	logical. Indicating if the measurements in x are standardized before calculating the dissimilarities.
`samples`	numeric. Say N, the number of samples to be drawn from the dataset. The default is N = 5,
`scale`	logical. Whether to use scaled data in kmeans.
`trace`	numberic. Indicating a trace level for diagnostic output during the algorithm
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to `clara` function

Value

an CYT object with clara.id in meta.data

See Also

`clara`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runClara(cyt, k = 25, verbose = TRUE)
```

---

runCluster *Specific Clustering Method Toolkits*

Description

Compute a specific clustering using the combined flow cytometry data. "som" `SOM`, "hclust" `hclust`, "clara" `clara`, "phenograph", "kmeans" `kmeans` are provided.

Usage

```
runCluster(
  object,
  cluster.method = c("som", "kmeans", "clara", "phenograph", "hclust", "mclust"),
  verbose = FALSE,
  ...
)
```

Arguments

`object`	an CYT object
`cluster.method`	character. Four clustering method are provided: som, clara, kmeans and phenograph. Clustering method "hclust" and "mclust" are not recommended because of long computing time.
`verbose`	logic. Whether to print calculation progress.
`...`	options to pass on to the clustering functions.

Value

An CYT object with cluster

See Also

`SOM`, `hclust`, `clara`, `kmeans`. You can use `runSOM`, `runClara`, `runPhenotype`, `runKmeans`, `runMclust` and `runHclust` to run clustering respectively.

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

# After building an CYT object
# Set random seed to make results reproducible

set.seed(1)
cyt <- runCluster(cyt, cluster.method = "som", xdim = 3, ydim = 3, verbose = TRUE)

# K-means clustering
cyt <- runCluster(cyt, cluster.method = "kmeans", k = 9, verbose = TRUE)

# Clara clustering
cyt <- runCluster(cyt, cluster.method = "clara", k = 9, verbose = TRUE)

# phenoGraph clustering
cyt <- runCluster(cyt, cluster.method = "phenograph", verbose = TRUE)

# hclust clustering
# not recommended for large cell size
cyt <- runCluster(cyt, cluster.method = "hclust", k = 9, verbose = TRUE)

# mclust clustering
# not recommended for large cell size
cyt <- runCluster(cyt, cluster.method = "mclust", verbose = TRUE)
```

`runDiff` *Calculate differential expression markers*

Description

Calculating differentially expressed markers

Usage

```
runDiff(object, branch.id = NULL, branch.id.2 = NULL, verbose = FALSE)
```

Arguments

`object`	an CYT object
`branch.id`	vector. Branch ids use to run differentially expressed markers
`branch.id.2`	vector. Branch ids use to run differentially expressed markers in compare with branch.id
`verbose`	logic. Whether to print calculation progress.

Value

An CYT object with cluster.id in meta.data

a data.frame with differential expressed markers

See Also

`bulidTree`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

DEG.table <- runDiff(cyt)
```

`runDiffusionMap` *Calculate diffusion map in CYT*

Description

Calculate diffusion map in CYT

Usage

```
runDiffusionMap(
  object,
  sigma.use = NULL,
  distance = c("euclidean", "cosine", "rankcor"),
  k = 30,
  density.norm = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	an CYT object
`sigma.use`	numeric. Diffusion scale parameter of the Gaussian kernel. One of '`local`', '`global`', a `numeric` global sigma or a Sigmas object. When choosing '`global`', a global sigma will be calculated using find_sigmas (See `destiny`). A larger sigma might be necessary if the eigenvalues can not be found because of a singularity in the matrix. See `destiny`.

`distance`	Distance measurement method applied to data or a distance matrix/dist. For the allowed values, see `destiny`
`k`	numeric. By default is 30. `destiny` can be used to specify k.
`density.norm`	logical. If TRUE, use density normalisation. See `destiny`
`verbose`	logical. Whether to print calculation progress.
`...`	options to pass on to the `destiny`.

Value

An CYT object

See Also

`destiny`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runDiffusionMap(cyt, verbose = TRUE)
```

`runExprsExtract` *Extract the expression data from a FCS file with preprocessing*

Description

Extract the FCS expresssion data with preprocessing of compensation (for FCM data only) and transformation. Transformtion methods includes `autoLgcl`, `cytofAsinh`, `logicle` (customizable) and `arcsinh` (customizable).

Usage

```
runExprsExtract(
  fcsFile,
  verbose = FALSE,
  comp = FALSE,
 transformMethod = c("autoLgcl", "cytofAsinh", "logicle", "arcsinh", "logAbs", "none"),
  scaleTo = NULL,
  showDesc = TRUE,
  keepRaw = TRUE,
  q = 0.05,
  l_w = 0.1,
  l_t = 4000,
  l_m = 4.5,
  l_a = 0,
  a_a = 1,
  a_b = 1,
  a_c = 0
)
```

Arguments

`fcsFile`	The name of the FCS file.
`verbose`	If `TRUE`, print the message details of FCS loading.
`comp`	If `TRUE`, does compensation by compensation matrix contained in FCS. Agrument also accepts a compensation matrix to be applied. Otherwise `FALSE`.
`transformMethod`	Data Transformation method, including `autoLgcl`, `cytofAsinh`, `logicle` and `arcsinh`, or `none` to avoid transformation.
`scaleTo`	Scale the expression to a specified range c(a, b), default is NULL.
`showDesc`	logical. Whether to show `desc` name in the output matrix.
`keepRaw`	logical. Whether to keep raw data for FSC and SSC.
`q`	Quantile of negative values removed for auto w estimation, default is 0.05, parameter for autoLgcl transformation.
`l_w`	Linearization width in asymptotic decades, parameter for logicle transformation.
`l_t`	Top of the scale data value, parameter for logicle transformation.
`l_m`	Full width of the transformed display in asymptotic decades, parameter for logicle transformation.
`l_a`	Additional negative range to be included in the display in asymptotic decades, parameter for logicle transformation.
`a_a`	Positive double that corresponds to the base of the arcsinh transformation, `arcsinh` = asinh(a + b * x) + c).
`a_b`	Positive double that corresponds to a scale factor of the arcsinh transformation, `arcsinh` = asinh(a + b * x) + c).
`a_c`	Positive double that corresponds to another scale factor of the arcsinh transformation, `arcsinh` = asinh(a + b * x) + c).

Value

A transformed expression data matrix

Author(s)

Chen Hao

References

Hao Chen, Mai Chan Lau, Michael Thomas Wong, Evan W. Newell, Michael Poidinger, Jinmiao Chen. Cytofkit: A Bioconductor Package for an Integrated Mass Cytometry Data Analysis Pipeline. PLoS Comput Biol, 2016.

Examples

```
# Read fcs files
fcs.file <- system.file("extdata/D0.FCS", package = "CytoTree")

# Read FCS files
exp.data <- runExprsExtract(fcs.file, showDesc = FALSE, transformMethod = "none")
```

---

runExprsMerge — *Merge the expression matrix from multiple FCS files with preprocessing*

Description

Apply preprocessing on each FCS file including compensation (for FCM data only) and transformation with selected markers, then expression matrix are extracted and merged using one of the methods, `all`, `min`, `fixed` or `ceil`

Usage

```
runExprsMerge(
  fcsFiles,
  comp = FALSE,
 transformMethod = c("autoLgcl", "cytofAsinh", "logicle", "arcsinh", "logAbs", "none"),
  scaleTo = NULL,
  mergeMethod = c("ceil", "all", "fixed", "min"),
  fixedNum = 2000,
  ...
)
```

Arguments

`fcsFiles`	A vector of FCS file names.
`comp`	If `TRUE`, does compensation by compensation matrix contained in FCS. Agrument also accepts a compensation matrix to be applied. Otherwise `FALSE`.
`transformMethod`	Data Transformation method, including `autoLgcl`, `cytofAsinh`, `logicle` and `arcsinh`, or `none` to avoid transformation.
`scaleTo`	Scale the expression to a specified range c(a, b), default is NULL.
`mergeMethod`	Merge method for mutiple FCS expression data. cells can be combined using one of the four different methods including `ceil`, `all`, `min`, `fixed`. The default option is `ceil`, up to a fixed number (specified by `fixedNum`) of cells are sampled without replacement from each fcs file and combined for analysis. `all`:

	all cells from each fcs file are combined for analysis. `min`: The minimum number of cells among all the selected fcs files are sampled from each fcs file and combined for analysis. `fixed`: a fixed num (specified by fixedNum) of cells are sampled (with replacement when the total number of cell is less than fixedNum) from each fcs file and combined for analysis.
`fixedNum`	The fixed number of cells to be extracted from each FCS file.
`...`	Other arguments passed to `runExprsExtract`

Value

A matrix containing the merged expression data, with selected markers.

Author(s)

Chen Hao

References

Hao Chen, Mai Chan Lau, Michael Thomas Wong, Evan W. Newell, Michael Poidinger, Jinmiao Chen. Cytofkit: A Bioconductor Package for an Integrated Mass Cytometry Data Analysis Pipeline. PLoS Comput Biol, 2016.

See Also

`runExprsExtract`

Examples

```
# Read fcs files
fcs.path <- system.file("extdata", package = "CytoTree")
fcs.files <- list.files(fcs.path, pattern = '.FCS$', full = TRUE)

fcs.data <- runExprsMerge(fcs.files, comp = FALSE, transformMethod = "none")
```

`runFastPCA` *Calculate principal components in CYT*

Description

Calculate principal components in CYT

Usage

```
runFastPCA(object, center = FALSE, scale. = TRUE, verbose = FALSE, ...)
```

Arguments

`object`	an CYT object
`center`	logical, a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of x can be supplied. The value is passed to scale. See `fast.prcomp`
`scale.`	logical, a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is FALSE for consistency with S, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of x can be supplied. The value is passed to scale. See `fast.prcomp`
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to `fast.prcomp` function

Value

An CYT object with PCA

See Also

`fast.prcomp`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runFastPCA(cyt, verbose = TRUE)
```

`runHclust` *runHclust*

Description

Hierarchical cluster analysis on a set of dissimilarities and methods for analyzing it.

Usage

```
runHclust(
  object,
  k = 25,
  hclust.method = "complete",
  dist.method = "euclidean",
  verbose = FALSE
)
```

Arguments

`object`	an CYT object
`k`	numeric. The number of clusters.
`hclust.method`	character or a function. The agglomeration method to be used. This should be one of "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median" or "centroid". Or you can specify an equation as input, for example `function(x) hclust(x,method = 'ward.D2')`.
`dist.method`	character or a function. The distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Or you can specify an equation as input, for example `function(x) as.dist((1-cor(t(x)))/2)`.
`verbose`	logical. Whether to print calculation progress.

Value

An CYT object with cluster

cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree") cyt <- readRDS(file = cyt.file)

cyt <- runHclust(cyt, k = 9, verbose = TRUE)

See Also

`hclust`, `dist`

---

runKmeans *runKmeans*

Description

Perform k-means clustering on a data matrix.

Usage

```
runKmeans(
  object,
  k = 25,
  iter.max = 10,
  nstart = 1,
  algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"),
  trace = FALSE,
  scale = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	an CYT object
`k`	numeric. The number of clusters.
`iter.max`	numeric. The maximum number of iterations allowed.
`nstart`	numeric. If k is a number, how many random sets should be chosen.
`algorithm`	character. Type of algorithm that will be choosen to calculate kmeans. Four algoritms are provided: Hartigan-Wong, Lloyd, Forgy, MacQueen.
`trace`	logical or integer number.
`scale`	logical. Whether to use scaled data in kmeans.
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to `kmeans` function

Value

an CYT object with kmeans.id in meta.data

See Also

`kmeans`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runKmeans(cyt, k = 25, verbose = TRUE)
```

`runKNN` *Calculate k-nearest neighbors of CYT*

Description

Calculates and stores a k-nearest neighbor graph based on Euclidean distance with (KMKNN) algorithm using log-transformed signaling matrix of flow cytometry data. The base function are base on `findKNN`.

Usage

```
runKNN(
  object,
  given.mat = NULL,
  knn = 30,
  knn.replace = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	an CYT object
`given.mat`	matrix. Given matrix to run knn
`knn`	numeric. Number of k-nearest neighbors.
`knn.replace`	logic. Whether to replace knn in CYT object
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to `findKNN` function

Value

An CYT object with knn, knn.index and knn.distance information.

See Also

`findKNN`

`runMclust` *runMclust*

Description

Model-based clustering based on parameterized finite Gaussian mixture models. This function is based on `Mclust`.

Usage

```
runMclust(object, scale = FALSE, verbose = FALSE, ...)
```

Arguments

`object`	an CYT object
`scale`	logical. Whether to use scaled data in Mclust.
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to `Mclust` function

Value

an CYT object with mclust.id in meta.data

See Also

`Mclust`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runMclust(cyt, verbose = TRUE)
```

---

runPhenograph *RphenoGraph clustering*

Description

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

Usage

```
runPhenograph(object, knn = 30, scale = FALSE, verbose = FALSE, ...)
```

Arguments

`object`	an CYT object.
`knn`	numeric. Number of nearest neighbours, default is 30.
`scale`	logical. Whether to scale the expression matrix
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to `igraph` function

Value

An CYT object with cluster

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runPhenograph(cyt, knn = 30, verbose = TRUE)
```

`runPseudotime` *Calculation of Pseudotime*

Description

calculation of Pseudotime based on KNN

Usage

```
runPseudotime(
  object,
  mode = "undirected",
  dim.type = c("raw", "pca", "tsne", "dc", "umap"),
  dim.use = seq_len(2),
  verbose = FALSE,
  ...
)
```

Arguments

`object`	An CYT object
`mode`	character. Specifies how igraph should interpret the supplied matrix. Possible values are: directed, undirected, upper, lower, max, min, plus.
`dim.type`	character. Type of dimensionality reduction method used to calculate pseudotime: raw, umap, tsne, dc and pca. By default is raw.
`dim.use`	numeric. Dimensions used to calculate pseudotime
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to calculation function.

Value

An CYT object

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runPseudotime(cyt, verbose = TRUE, dim.type = "raw")
cyt <- runPseudotime(cyt, verbose = TRUE, dim.type = "umap", dim.use = seq_len(2))
cyt <- runPseudotime(cyt, verbose = TRUE, dim.type = "tsne", dim.use = seq_len(2))
cyt <- runPseudotime(cyt, verbose = TRUE, dim.type = "dc", dim.use = seq_len(3))
cyt <- runPseudotime(cyt, verbose = TRUE, dim.type = "pca", dim.use = seq_len(3))

# tSNE plot colored by pseudotime
plot2D(cyt, item.use = c("tSNE_1", "tSNE_2"), category = "numeric",
       size = 1, color.by = "pseudotime")
```

```
# UMAP plot colored by pseudotime
plot2D(cyt, item.use = c("UMAP_1", "UMAP_2"), category = "numeric",
       size = 1, color.by = "pseudotime")
```

---

runSOM *calculation SOM in CYT object*

Description

Build a self-organizing map

Usage

```
runSOM(
  object,
  xdim = 6,
  ydim = 6,
  rlen = 8,
  mst = 1,
  alpha = c(0.05, 0.01),
  radius = 1,
  init = FALSE,
  distf = 2,
  codes = NULL,
  importance = NULL,
  method = "euclidean",
  verbose = FALSE,
  ...
)
```

Arguments

`object`	an CYT object
`xdim`	Width of the grid.
`ydim`	Hight of the grid.
`rlen`	Number of times to loop over the training data for each MST
`mst`	Number of times to build an MST
`alpha`	Start and end learning rate
`radius`	Start and end radius
`init`	Initialize cluster centers in a non-random way
`distf`	Distance function (1=manhattan, 2=euclidean, 3=chebyshev, 4=cosine)
`codes`	Cluster centers to start with

`importance`	array with numeric values. Parameters will be scaled according to importance
`method`	the distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given. See `dist`
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to `SOM` function

Value

an CYT object with som.id in CYT object

References

This code is strongly based on the `SOM` function. Which is developed by Sofie Van Gassen, Britt Callebaut and Yvan Saeys (2018).

See Also

`BuildSOM`

`SOM`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runSOM(cyt, xdim = 10, ydim = 10, verbose = TRUE)
```

`runTSNE` *Calculate t-Distributed Stochastic Neighbor Embedding in CYT*

Description

Calculate t-Distributed Stochastic Neighbor Embedding in CYT

Usage

```
runTSNE(
  object,
  dims = 2,
  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,
  ...
)
```

Arguments

`object`	an CYT object
`dims`	integer, Output dimensionality (default: 2)
`initial_dims`	integer. the number of dimensions that should be retained in the initial PCA step (default: 50). See `Rtsne`
`perplexity`	numeric. Perplexity parameter. See `Rtsne`
`theta`	numeric. Speed/accuracy trade-off (increase for less accuracy), set to 0.0 for exact TSNE (default: 0.5). See `Rtsne`
`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). See `Rtsne`
`pca, max_iter, is_distance, Y_init, pca_center, pca_scale`	See `Rtsne`
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to `Rtsne` function

Value

An CYT object

References

Maaten, L. Van Der, 2014. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research, 15, p.3221-3245.

van der Maaten, L.J.P. & Hinton, G.E., 2008. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research, 9, pp.2579-2605.

See Also

`Rtsne`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)
cyt <- runTSNE(cyt, dims = 2, verbose = TRUE)
cyt <- runTSNE(cyt, dims = 2, perplexity = 20, verbose = TRUE)
```

runUMAP *Calculating UMAP*

Description

Calculate Uniform Manifold Approximation and Projection in CYT

Usage

```
runUMAP(
  object,
  umap.config = umap.defaults,
  n_neighbors = 30,
  dims = 2,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	an CYT object
`umap.config`	object of class umap.config. See `umap`.
`n_neighbors`	numeric. Number of neighbors
`dims`	numeric. Dim of umap, you can also change it in umap.config.
`verbose`	logical. Whether to print calculation progress.
`...`	Options to pass on to the `umap` function

Value

An CYT object

See Also

`umap`

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runUMAP(cyt, verbose = TRUE)
cyt <- runUMAP(cyt, n_neighbors = 20, verbose = TRUE)
```

runWalk *Walk between root cells and leaf cells*

Description

Walk between root cells and leaf cells

Usage

```
runWalk(
  object,
  mode = c("undirected", "directed", "max", "min", "upper", "lower", "plus"),
  max.run.forward = 20,
  backward.walk = FALSE,
  max.run.backward = 20,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	An CYT object
`mode`	character. Specifies how igraph should interpret the supplied matrix. Possible values are: undirected, directed, upper, lower, max, min, plus. By default is undirected.
`max.run.forward`	numeric. Maximum cycles of forward walk.
`backward.walk`	logical. Whether to run backward walk.
`max.run.backward`	numeric. Maximum cycles of backward walk.
`verbose`	logical. Whether to print calculation progress.
`...`	Parameters passing to calculation function.

Value

An CYT object

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- runWalk(cyt, verbose = TRUE)
cyt <- runWalk(cyt, backward.walk = FALSE, verbose = TRUE)
```

`subsetCYT` *subset CYT object*

Description

This subsets an CYT object by given a list of cells or cluster id. This function will subset all results without recalculating them, such as knn, PCA, tSNE, umap and pseudotime. For instance, you can choose recalculate PCA and tSNE and destiny scores by paramter recalculate.

Usage

```
subsetCYT(object, cells = NULL, knn = NA, verbose = FALSE)
```

Arguments

`object`	An CYT object
`cells`	vector, Names of the cells to retain.
`knn`	numeric. If is NA, the KNN will be equal to the knn number in the input CYT object.
`verbose`	logic. Whether to print calculation progress.

Value

An CYT object

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

meta.data <- fetchPlotMeta(cyt)
cells <- meta.data$cell[which(meta.data$stage == "D0")]
sub.cyt <- subsetCYT(cyt, cells = cells)
sub.cyt
```

updateClustMeta *Update clusters' meta information of CYT*

Description

Update clusters' meta information of CYT

Usage

```
updateClustMeta(object, verbose = TRUE)
```

Arguments

`object`	An CYT object
`verbose`	logical. Whether to print calculation progress.

Value

An CYT object

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- updateClustMeta(cyt)
```

updatePlotMeta *Update plot meta information of CYT*

Description

Update plot meta information of CYT

Usage

```
updatePlotMeta(object, verbose = TRUE)
```

Arguments

`object`	An CYT object
`verbose`	logical. Whether to print calculation progress.

Value

An CYT object

Examples

```
cyt.file <- system.file("extdata/cyt.rds", package = "CytoTree")
cyt <- readRDS(file = cyt.file)

cyt <- updatePlotMeta(cyt)
```

Index

∗ **package**
    CytoTree-package, 3

BuildSOM, *48*
buildTree, 4

clara, *33, 34*
ComBat, *6, 7*
constraintMatrix, 5
correctBatchCYT, 6
createCYT, 7
CYT *(*CYT-class*)*, 8
CYT-class, 8
CYT-class, *(*CYT-class*)*, 8
CYTclass, *(*CYT-class*)*, 8
CytoTree *(*CytoTree-package*)*, 3
CytoTree-package, 3

defLeafCells, 9
defRootCells, 10
dist, *42*, *48*

fast.prcomp, *9*, *31*, *41*
fetchCell, 11
fetchClustMeta, 12
fetchPlotMeta, 12
find_neighbors, 13
findKNN, *6*, *43, 44*
FlowSOM, *9*

gatingMatrix, 14

hclust, *34*, *42*

kmeans, *34*, *43*

Mclust, *44*

numeric, *36*

pheatmap, *18*, *21, 22*, *27*

plot2D, 15
plot3D, 17
plotBranchHeatmap, 18
plotCluster, 19
plotClusterHeatmap, 20
plotHeatmap, 21
plotMarkerDensity, 22
plotPieCluster, 23
plotPieTree, 24
plotPseudotimeDensity, 25
plotPseudotimeTraj, 26
plotTrajHeatmap, 27
plotTree, 28
plotViolin, 29
processingCluster, 30

Rphenograph, 31
Rtsne, *9*, *30, 31*, *49*
runClara, 33
runCluster, 34
runDiff, 35
runDiffusionMap, 36
runExprsExtract, 37, *40*
runExprsMerge, 39
runFastPCA, 40
runHclust, 41
runKmeans, 42
runKNN, 43
runMclust, 44
runPhenograph, 45
runPseudotime, 46
runSOM, 47
runTSNE, 48
runUMAP, 50
runWalk, 51

scatterplot3d, *17*
SOM, *34*, *48*
subsetCYT, 52

umap, *9*, *30, 31*, *50*
updateClustMeta, 53
updatePlotMeta, 53