Package: rliger (via r-universe)

September 2, 2024

Version 2.0.1

Date 2024-04-04

Type Package

Title Linked Inference of Genomic Experimental Relationships

Description Uses an extension of nonnegative matrix factorization to identify shared and dataset-specific factors. See Welch J, Kozareva V, et al (2019) <doi:10.1016/j.cell.2019.05.006>, and Liu J, Gao C, Sodicoff J, et al (2020) <doi:10.1038/s41596-020-0391-8> for more details.

Author Joshua Welch [aut], Yichen Wang [aut, cre], Chao Gao [aut], Jialin Liu [aut], Joshua Sodicoff [aut, ctb], Velina Kozareva [aut, ctb], Evan Macosko [aut, ctb], Paul Hoffman [ctb], Ilya Korsunsky [ctb], Robert Lee [ctb], Andrew Robbins [ctb]

Maintainer Yichen Wang <wayichen@umich.edu>

BugReports https://github.com/welch-lab/liger/issues

URL https://welch-lab.github.io/liger/

License GPL-3

LazyData true

RoxygenNote 7.3.1

VignetteBuilder knitr

Encoding UTF-8

Additional_repositories https://welch-lab.r-universe.dev, https://blaserlab.r-universe.dev

LinkingTo Rcpp, RcppArmadillo, RcppProgress

Depends methods, stats, utils, R (>= 3.5)

Imports circlize, cli, cowplot, ComplexHeatmap, dplyr, ggplot2, grid, hdf5r, leidenAlg (>= 1.1.1), lifecycle, magrittr, Matrix, RANN, RColorBrewer, Rcpp, rlang, Rtsne, S4Vectors, scales, uwot, viridis

Suggests AnnotationDbi, DESeq2, DoubletFinder (>= 2.0.4), EnhancedVolcano, fgsea, GenomicRanges, ggrepel, gprofiler2, IRanges, knitr, org.Hs.eg.db, plotly, psych, RcppPlanc, reactome.db, rmarkdown, sankey, scattermore (>= 0.7), Seurat, SeuratObject, SingleCellExperiment, SummarizedExperiment, testthat

Repository https://welch-lab.r-universe.dev

RemoteUrl https://github.com/welch-lab/liger

RemoteRef HEAD

RemoteSha 58e59572385688b5975f4d70c0537776a33edd74

Contents

`.complexHeatmapDotPlot` *Generate dot plot from input matrix with ComplexHeatmap*

Description

Generate dot plot from input matrix with ComplexHeatmap

Usage

```
.complexHeatmapDotPlot(
  colorMat,
  sizeMat,
  featureAnnDF = NULL,
  cellSplitVar = NULL,
  cellLabels = NULL,
  maxDotSize = 4,
  clusterFeature = FALSE,
  clusterCell = FALSE,
  legendColorTitle = "Matrix Value",
  legendSizeTitle = "Fraction Value",
  transpose = FALSE,
  baseSize = 8,
```

```
  cellTextSize = NULL,
  featureTextSize = NULL,
  cellTitleSize = NULL,
  featureTitleSize = NULL,
  legendTextSize = NULL,
  legendTitleSize = NULL,
  featureGrpRot = 0,
  viridisOption = "C",
  viridisDirection = -1,
  ...
)
```

Arguments

`colorMat`, `sizeMat`
: Matrix of the same size. Values in `colorMat` will be visualized with color while values in `sizeMat` will be reflected by dot size.

`featureAnnDF`
: Data frame of features containing feature names and grouping labels.

`cellSplitVar`
: Split the cell orientation (default columns) by this variable.

`cellLabels`
: Label to be shown on cell orientation.

`maxDotSize`
: The maximum dot size. Default `4`.

`clusterFeature`, `clusterCell`
: Whether the feature/cell orientation (default rows/column, respectively) should be clustered. Default `FALSE`.

`legendColorTitle`, `legendSizeTitle`
: The title for color bar and dot size legends, repectively. Default see `"Matrix Value"` and `"Fraction Value"`.

`transpose`
: Logical, whether to rotate the dot plot orientation. i.e. rows as cell aggregation and columns as features. Default `FALSE`.

`baseSize`
: One-parameter control of all text sizes. Individual text element sizes can be controlled by other size arguments. "Title" sizes are 2 points larger than "text" sizes when being controlled by this. Default `8`.

`cellTextSize`, `featureTextSize`, `legendTextSize`
: Size of cell labels, feature label and legend text. Default `NULL` controls by `baseSize`.

`cellTitleSize`, `featureTitleSize`, `legendTitleSize`
: Size of titles on cell and feature orientation and legend title. Default `NULL` controls by `baseSize + 2`.

`featureGrpRot`
: Number of degree to rotate the feature grouping label. Default `0`.

`viridisOption`, `viridisDirection`
: See argument `option` and `direction` of `viridis`. Default `"A"` and `-1`.

`...`
: Additional arguments passed to `Heatmap`.

Value

A `HeatmapList` object.

`.ggCellViolin` *Produce single violin plot with data frame passed from upstream*

Description

Produce single violin plot with data frame passed from upstream

Usage

```
.ggCellViolin(
  plotDF,
  y,
  groupBy = NULL,
  colorBy = NULL,
  violin = TRUE,
  violinAlpha = 0.8,
  violinWidth = 0.9,
  box = FALSE,
  boxAlpha = 0.6,
  boxWidth = 0.4,
  dot = FALSE,
  dotColor = "black",
  dotSize = getOption("ligerDotSize"),
  raster = NULL,
  seed = 1,
  ...
)
```

Arguments

`plotDF`	Data frame like object (fortifiable) that contains all necessary information to make the plot.
`y`, `groupBy`, `colorBy`	See `plotCellViolin`.
`violin`, `box`, `dot`	Logical, whether to add violin plot, box plot or dot (scatter) plot, respectively. Layers are added in the order of dot, violin, and violin on the top surface. By default, only violin plot is generated.
`violinAlpha`, `boxAlpha`	Numeric, controls the transparency of layers. Default `0.8`, `0.6`, respectively.
`violinWidth`, `boxWidth`	Numeric, controls the width of violin/box bounding box. Default `0.9` and `0.4`.
`dotColor`, `dotSize`	Numeric, globally controls the appearance of all dots. Default `"black"` and `getOption("ligerDotSize")` (1).
`raster`	Logical, whether to rasterize the dot plot. Default `NULL` automatically rasterizes the dot plot when number of total cells to be plotted exceeds 100,000.

seed	Random seed for reproducibility. Default 1.
...	More theme setting arguments passed to .ggplotLigerTheme.

Value

ggplot object by default. When plotly = TRUE, returns plotly (htmlwidget) object.

.ggplotLigerTheme *Generic ggplot theme setting for rliger package*

Description

Controls content and size of all peripheral texts.

Usage

```
.ggplotLigerTheme(
  plot,
  title = NULL,
  subtitle = NULL,
  xlab = TRUE,
  ylab = TRUE,
  legendColorTitle = NULL,
  legendFillTitle = NULL,
  legendShapeTitle = NULL,
  legendSizeTitle = NULL,
  showLegend = TRUE,
  legendPosition = "right",
  baseSize = getOption("ligerBaseSize"),
  titleSize = NULL,
  subtitleSize = NULL,
  xTextSize = NULL,
  xFacetSize = NULL,
  xTitleSize = NULL,
  yTextSize = NULL,
  yFacetSize = NULL,
  yTitleSize = NULL,
  legendTextSize = NULL,
  legendTitleSize = NULL,
  legendDotSize = 4,
  panelBorder = FALSE,
  legendNRow = NULL,
  legendNCol = NULL,
  colorLabels = NULL,
  colorValues = NULL,
  colorPalette = "magma",
  colorDirection = -1,
```

```
  naColor = "#DEDEDE",
  colorLow = NULL,
  colorMid = NULL,
  colorHigh = NULL,
  colorMidPoint = NULL,
  plotly = FALSE
)
```

Arguments

plot	ggplot object passed from wrapper plotting functions
title, subtitle, xlab, ylab	Main title, subtitle or X/Y axis title text. By default, no main title or subtitle will be set, and X/Y axis title will be the names of variables used for plotting. Use NULL to hide elements. TRUE for xlab or ylab shows default values.
legendColorTitle, legendFillTitle, legendShapeTitle, legendSizeTitle	Set alternative title text for legend on aes of color, fill, shape and size, respectively. Default NULL shows the original variable name.
showLegend	Whether to show the legend. Default TRUE.
legendPosition	Text indicating where to place the legend. Choose from "top", "bottom", "left" or "right". Default "right".
baseSize	One-parameter control of all text sizes. Individual text element sizes can be controlled by other size arguments. "Title" sizes are 2 points larger than "text" sizes when being controlled by this.
titleSize, xTitleSize, yTitleSize, legendTitleSize	Size of main title, axis titles and legend title. Default NULL controls by baseSize + 2.
subtitleSize, xTextSize, yTextSize, legendTextSize	Size of subtitle text, axis texts and legend text. Default NULL controls by baseSize.
xFacetSize, yFacetSize	Size of facet label text. Default NULL controls by baseSize - 2.
legendDotSize	Allow dots in legend region to be large enough to see the colors/shapes clearly. Default 4.
panelBorder	Whether to show rectangle border of the panel instead of using ggplot classic bottom and left axis lines. Default FALSE.
legendNRow, legendNCol	Integer, when too many categories in one variable, arranges number of rows or columns. Default NULL, automatically split to ceiling(levels(variable)/10) columns.
colorLabels, colorValues	Each a vector with as many values as the number of categories for the categorical coloring aesthetics. Labels will be the shown text and values will be the color code. These are passed to scale_color_manual. Default uses an internal selected palette if there are <= 26 colors needed, or ggplot hues otherwise, and plot original labels (levels of the factor).
colorPalette	For continuous coloring, an index or a palette name to select from available options from ggplot scale_brewer or viridis. Default "magma".

`colorDirection` Choose `1` or `-1`. Applied when `colorPalette` is from Viridis options. Default `-1` use darker color for higher value, while `1` reverses this direction.

`naColor` The color code for `NA` values. Default `"#DEDEDE"`. `scale_colour_gradient2`. Default `NULL`.

`colorLow`, `colorMid`, `colorHigh`, `colorMidPoint`
All four of these must be specified to customize palette with

`plotly` Whether to use plotly to enable web based interactive browsing for the plot. Requires installation of package "plotly". Default `FALSE`.

Value

Updated ggplot object by default. When `plotly = TRUE`, returns plotly (htmlwidget) object.

.ggScatter *Produce single scatter plot with data frame passed from upstream*

Description

Produce single scatter plot with data frame passed from upstream

Usage

```
.ggScatter(
  plotDF,
  x,
  y,
  colorBy = NULL,
  shapeBy = NULL,
  dotOrder = c("shuffle", "ascending", "descending"),
  dotSize = getOption("ligerDotSize"),
  dotAlpha = 0.9,
  trimHigh = NULL,
  trimLow = NULL,
  zeroAsNA = TRUE,
  raster = NULL,
  labelBy = colorBy,
  labelText = TRUE,
  labelTextSize = 4,
  seed = 1,
  ...
)
```

Arguments

`plotDF` Data frame like object (fortifiable) that contains all necessary information to make the plot.

x, y
: Available variable name in `cellMeta` slot to look for the dot coordinates. See details.

colorBy, shapeBy
: See `plotDimRed`.

dotOrder
: Controls the order that each dot is added to the plot. Choose from `"shuffle"`, `"ascending"`, or `"descending"`. Default `"shuffle"`, useful when coloring by categories that overlaps (e.g. "dataset"), `"ascending"` can be useful when coloring by a continuous variable (e.g. gene expression) where high values needs more highlight. `NULL` use default order.

dotSize, dotAlpha
: Numeric, controls the size or transparency of all dots. Default `getOption("ligerDotSize")` (1) and `0.9`.

trimHigh, trimLow
: Numeric, limit the largest or smallest value of continuous `colorBy` variable. Default `NULL`.

zeroAsNA
: Logical, whether to set zero values in continuous `colorBy` variable to `NA` so the color of these value.

raster
: Logical, whether to rasterize the plot. Default `NULL` automatically rasterize the plot when number of total cells to be plotted exceeds 100,000.

labelBy
: A variable name available in `plotDF`. If the variable is categorical (a factor), the label position will be the median coordinates of all dots within the same group. Unique labeling in character vector for each dot is also acceptable. Default `colorBy`.

labelText
: Logical, whether to show text label at the median position of each categorical group specified by `colorBy`. Default `TRUE`. Does not work when continuous coloring is specified.

labelTextSize
: Numeric, controls the size of label size when `labelText = TRUE`. Default `4`.

seed
: Random seed for reproducibility. Default `1`.

...
: More theme setting arguments passed to `.ggplotLigerTheme`.

Details

Having package "ggrepel" installed can help adding tidier text labels on the scatter plot.

Value

ggplot object by default. When `plotly = TRUE`, returns plotly (htmlwidget) object.

`.plotHeatmap` *General heatmap plotting with prepared matrix and data.frames*

Description

This is not an exported function. This documentation just serves for a manual of extra arguments that users can use when generating heatmaps with `plotGeneHeatmap` or `plotFactorHeatmap`.

Note that the following arguments are pre-occupied by upstream wrappers so users should not include them in a function call: `dataMatrix`, `dataName`, `cellDF`, `featureDF`, `cellSplitVar`, `featureSplitVar`.

The following arguments of `Heatmap` is occupied by this function, so users should include them in a function call as well: `matrix`, `name`, `col`, `heatmap_legend_param`, `top_annotation`, `column_title_gp`, `column_names_gp`, `show_column_names`, `column_split`, `column_gap`, `left_annotation`, `row_title_gp`, `row_names_gp`, `show_row_names`, `row_split`, `row_gap`.

Usage

```
.plotHeatmap(
  dataMatrix,
  dataName = "Value",
  cellDF = NULL,
  featureDF = NULL,
  transpose = FALSE,
  cellSplitVar = NULL,
  featureSplitVar = NULL,
  dataScaleFunc = NULL,
  showCellLabel = FALSE,
  showCellLegend = TRUE,
  showFeatureLabel = TRUE,
  showFeatureLegend = TRUE,
  cellAnnColList = NULL,
  featureAnnColList = NULL,
  scale = FALSE,
  trim = c(-2, 2),
  baseSize = 8,
  cellTextSize = NULL,
  featureTextSize = NULL,
  cellTitleSize = NULL,
  featureTitleSize = NULL,
  legendTextSize = NULL,
  legendTitleSize = NULL,
  viridisOption = "A",
  viridisDirection = -1,
  RColorBrewerOption = "RdBu",
  ...
)
```

Arguments

dataMatrix	Matrix object with features/factors as rows and cells as columns.
dataName	Text for heatmap color bar title. Default Value.
cellDF	data.frame object. Number of rows must match with number of columns of dataMatrix.
featureDF	data.frame object. Number of columns must match with number of rows of dataMatrix.
transpose	Logical, whether to "rotate" the heatmap by 90 degrees so that cell information is displayed by row. Default FALSE.
cellSplitVar, featureSplitVar	Subset columns of cellDF or featureDF, respectively.
dataScaleFunc	A function object, applied to dataMatrix.
showCellLabel, showFeatureLabel	Logical, whether to show cell barcodes, gene symbols or factor names. Default TRUE for gene/factors but FALSE for cells.
showCellLegend, showFeatureLegend	Logical, whether to show cell or feature legends. Default TRUE. Can be a scalar for overall control or a vector matching with each given annotation variable.
cellAnnColList, featureAnnColList	List object, with each element a named vector of R-interpretable color code. The names of the list elements are used for matching the annotation variable names. The names of the colors in the vectors are used for matching the levels of a variable (factor object, categorical). Default NULL generates ggplot-flavor categorical colors.
scale	Logical, whether to take z-score to scale and center gene expression. Applied after dataScaleFunc. Default FALSE.
trim	Numeric vector of two values. Limit the z-score value into this range when scale = TRUE. Default c(-2, 2).
baseSize	One-parameter control of all text sizes. Individual text element sizes can be controlled by other size arguments. "Title" sizes are 2 points larger than "text" sizes when being controlled by this.
cellTextSize, featureTextSize, legendTextSize	Size of cell barcode labels, gene/factor labels, or legend values. Default NULL.
cellTitleSize, featureTitleSize, legendTitleSize	Size of titles of the cell slices, gene/factor slices, or the legends. Default NULL.
viridisOption, viridisDirection	See argument option and direction of viridis. Default "A" and -1.
RColorBrewerOption	When scale = TRUE, heatmap color will be mapped with brewer.pal. This is passed to name. Default "RdBu".
...	Additional arguments to be passed to Heatmap.

Value

HeatmapList-class object

`as.liger.dgCMatrix` *Converting other classes of data to a liger object*

Description

This function converts data stored in SingleCellExperiment (SCE), Seurat object or a merged sparse matrix (dgCMatrix) into a liger object. This is designed for a container object or matrix that already contains multiple datasets to be integerated with LIGER. For individual datasets, please use `createLiger` instead.

Usage

```
## S3 method for class 'dgCMatrix'
as.liger(object, datasetVar = NULL, modal = NULL, ...)

## S3 method for class 'SingleCellExperiment'
as.liger(object, datasetVar = NULL, modal = NULL, ...)

## S3 method for class 'Seurat'
as.liger(object, datasetVar = NULL, modal = NULL, assay = NULL, ...)

seuratToLiger(object, datasetVar = NULL, modal = NULL, assay = NULL, ...)

as.liger(object, ...)
```

Arguments

`object`	Object.
`datasetVar`	Specify the dataset belonging by: 1. Select a variable from existing metadata in the object (e.g. colData column); 2. Specify a vector/factor that assign the dataset belonging. 3. Give a single character string which means that all data is from one dataset (must not be a metadata variable, otherwise it is understood as 1.). Default `NULL` gathers things into one dataset and names it "sample" for dgCMatrix, attempts to find variable "sample" from SCE or "orig.ident" from Seurat.
`modal`	Modality setting for each dataset. See `createLiger`.
`...`	Additional arguments passed to `createLiger`
`assay`	Name of assay to use. Default `NULL` uses current active assay.

Details

For Seurat V5 structure, it is highly recommended that users make use of its split layer feature, where things like "counts", "data", and "scale.data" can be held for each dataset in the same Seurat object, e.g. with "count.ctrl", "count.stim", not merged. If a Seurat object with split layers is given, `datasetVar` will be ignored and the layers will be directly used.

Value

a liger object.

Examples

```
# dgCMatrix (common sparse matrix class), usually obtained from other
# container object, and contains multiple samples merged in one.
matList <- rawData(pbmc)
multiSampleMatrix <- mergeSparseAll(matList)
# The `datasetVar` argument expects the variable assigning the sample source
pbmc2 <- as.liger(multiSampleMatrix, datasetVar = pbmc$dataset)
pbmc2


if (requireNamespace("SingleCellExperiment", quietly = TRUE)) {
    sce <- SingleCellExperiment::SingleCellExperiment(
        assays = list(counts = multiSampleMatrix)
    )
    sce$sample <- pbmc$dataset
    pbmc3 <- as.liger(sce, datasetVar = "sample")
    pbmc3
}

if (requireNamespace("Seurat", quietly = TRUE)) {
    seu <- SeuratObject::CreateSeuratObject(multiSampleMatrix)
    # Seurat creates variable "orig.ident" by identifying the cell barcode
    # prefixes, which is indeed what we need in this case. Users might need
    # to be careful and have it confirmed first.
    pbmc4 <- as.liger(seu, datasetVar = "orig.ident")
    pbmc4

    # As per Seurat V5 updates with layered data, specifically helpful udner the
    # scenario of dataset integration. "counts" and etc for each datasets can be
    # split into layers.
    seu5 <- seu
    seu5[["RNA"]] <- split(seu5[["RNA"]], pbmc$dataset)
    print(SeuratObject::Layers(seu5))
    pbmc5 <- as.liger(seu5)
    pbmc5
}
```

---

as.ligerDataset.ligerDataset

Converting other classes of data to a ligerDataset object

Description

Works for converting a matrix or container object to a single ligerDataset, and can also convert the modality preset of a ligerDataset. When used with a dense matrix object, it automatically converts

the matrix to sparse form (dgCMatrix-class). When used with container objects such as Seurat or SingleCellExperiment, it is highly recommended that the object contains only one dataset/sample which is going to be integrated with LIGER. For multi-sample objects, please use as.liger with dataset source variable specified.

Usage

```
## S3 method for class 'ligerDataset'
as.ligerDataset(
  object,
  modal = c("default", "rna", "atac", "spatial", "meth"),
  ...
)

## Default S3 method:
as.ligerDataset(
  object,
  modal = c("default", "rna", "atac", "spatial", "meth"),
  ...
)

## S3 method for class 'matrix'
as.ligerDataset(
  object,
  modal = c("default", "rna", "atac", "spatial", "meth"),
  ...
)

## S3 method for class 'Seurat'
as.ligerDataset(
  object,
  modal = c("default", "rna", "atac", "spatial", "meth"),
  assay = NULL,
  ...
)

## S3 method for class 'SingleCellExperiment'
as.ligerDataset(
  object,
  modal = c("default", "rna", "atac", "spatial", "meth"),
  ...
)

as.ligerDataset(object, ...)
```

Arguments

object	Object.

modal	Modality setting for each dataset. Choose from "default", "rna", "atac", "spatial", "meth".
...	Additional arguments passed to createLigerDataset
assay	Name of assay to use. Default NULL uses current active assay.

Value

a liger object.

Examples

```
ctrl <- dataset(pbmc, "ctrl")
ctrl
# Convert the modality preset
as.ligerDataset(ctrl, modal = "atac")
rawCounts <- rawData(ctrl)
class(rawCounts)
as.ligerDataset(rawCounts)
```

---

bmmc *liger object of bone marrow subsample data with RNA and ATAC modality*

Description

liger object of bone marrow subsample data with RNA and ATAC modality

Usage

```
bmmc
```

Format

liger object with two dataset named by "rna" and "atac"

Source

https://www.nature.com/articles/s41587-019-0332-7

References

Jeffrey M. Granja and et. al., Nature Biotechnology, 2019

`calcAgreement` *Calculate agreement metric after integration*

Description

This metric quantifies how much the factorization and alignment distorts the geometry of the original datasets. The greater the agreement, the less distortion of geometry there is. This is calculated by performing dimensionality reduction on the original and quantile aligned (or just factorized) datasets, and measuring similarity between the k nearest neighbors for each cell in original and aligned datasets. The Jaccard index is used to quantify similarity, and is the final metric averages across all cells.

Note that for most datasets, the greater the chosen `nNeighbor`, the greater the agreement in general. Although agreement can theoretically approach 1, in practice it is usually no higher than 0.2-0.3.

Usage

```
calcAgreement(
  object,
  ndims = 40,
  nNeighbors = 15,
  useRaw = FALSE,
  byDataset = FALSE,
  seed = 1,
  dr.method = NULL,
  k = nNeighbors,
  use.aligned = NULL,
  rand.seed = seed,
  by.dataset = byDataset
)
```

Arguments

`object`	`liger` object. Should call quantile_norm before calling.
`ndims`	Number of factors to produce in NMF. Default `40`.
`nNeighbors`	Number of nearest neighbors to use in calculating Jaccard index. Default `15`.
`useRaw`	Whether to evaluate just factorized H matrices instead of using quantile aligned $H.norm$ matrix. Default `FALSE` uses aligned matrix.
`byDataset`	Whether to return agreement calculated for each dataset instead of the average for all datasets. Default `FALSE`.
`seed`	Random seed to allow reproducible results. Default `1`.
`dr.method`	[defunct] We no longer support other methods but just NMF.
`k`, `rand.seed`, `by.dataset`	[Deprecated] See Usage for replacement.
`use.aligned`	[defunct] Use `useRaw` instead.

Value

A numeric vector of agreement metric. A single value if `byDataset = FALSE` or each dataset a value otherwise.

Examples

```
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    pbmc <- pbmc %>%
    normalize %>%
    selectGenes %>%
    scaleNotCenter %>%
    runINMF %>%
    quantileNorm
    calcAgreement(pbmc)
}
```

`calcAlignment` *Calculate alignment metric after integration*

Description

This metric quantifies how well-aligned two or more datasets are. We randomly downsample all datasets to have as many cells as the smallest one. We construct a nearest-neighbor graph and calculate for each cell how many of its neighbors are from the same dataset. We average across all cells and compare to the expected value for perfectly mixed datasets, and scale the value from 0 to 1. Note that in practice, alignment can be greater than 1 occasionally.

Usage

```
calcAlignment(
  object,
  clustersUse = NULL,
  clusterVar = NULL,
  nNeighbors = NULL,
  cellIdx = NULL,
  cellComp = NULL,
  resultBy = c("all", "dataset", "cell"),
  seed = 1,
  k = nNeighbors,
  rand.seed = seed,
  cells.use = cellIdx,
  cells.comp = cellComp,
  clusters.use = clustersUse,
  by.cell = NULL,
  by.dataset = NULL
)
```

Arguments

`object`	A liger object, with `quantileNorm` already run.
`clustersUse`	The clusters to consider for calculating the alignment. Should be a vector of existing levels in `clusterVar`. Default `NULL`. See Details.
`clusterVar`	The name of one variable in `cellMeta(object)`. Default `NULL` uses default clusters.
`nNeighbors`	Number of neighbors to use in calculating alignment. Default `NULL` uses `floor(0.01*ncol(object))`, with a lower bound of 10 in all cases except where the total number of sampled cells is less than 10.
`cellIdx`, `cellComp`	Character, logical or numeric index that can subscribe cells. Default `NULL`. See Details.
`resultBy`	Select from `"all"`, `"dataset"` or `"cell"`. On which level should the mean alignment be calculated. Default `"all"`.
`seed`	Random seed to allow reproducible results. Default `1`.
`k`, `rand.seed`, `cells.use`, `cells.comp`, `clusters.use`	[Deprecated] Please see Usage for replacement.
`by.cell`, `by.dataset`	[Defunct] Use `resultBy` instead.

Details

$\bar{x}$ is the average number of neighbors belonging to any cells' same dataset, N is the number of datasets, k is the number of neighbors in the KNN graph.

$$1 - \frac{\bar{x} - \frac{k}{N}}{k - \frac{k}{N}}$$

The selection on cells to be measured can be done in various way and represent different scenarios:

1. By default, all cells are considered and the alignment across all datasets will be calculated.
2. Select `clustersUse` from `clusterVar` to use cells from the clusters of interests. This measures the alignment across all covered datasets within the specified clusters.
3. Only Specify `cellIdx` for flexible selection. This measures the alignment across all covered datasets within the specified cells. A none-NULL `cellIdx` privileges over `clustersUse`.
4. Specify `cellIdx` and `cellComp` at the same time, so that the original dataset source will be ignored and cells specified by each argument will be regarded as from each a dataset. This measures the alignment between cells specified by the two arguments. `cellComp` can contain cells already specified in `cellIdx`.

Value

The alignment metric.

Examples

```
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    pbmc <- pbmc %>%
    normalize %>%
    selectGenes %>%
    scaleNotCenter %>%
    runINMF %>%
    quantileNorm
    calcAlignment(pbmc)
}
```

`calcARI` *Calculate adjusted Rand index (ARI) by comparing two cluster labeling variables*

Description

This function aims at calculating the adjusted Rand index for the clustering result obtained with LIGER and the external clustering (existing "true" annotation). ARI ranges from 0 to 1, with a score of 0 indicating no agreement between clusterings and 1 indicating perfect agreement.

The true clustering annotation must be specified as the base line. We suggest setting it to the object cellMeta so that it can be easily used for many other visualization and evaluation functions.

The ARI can be calculated for only specified datasets, since true annotation might not be available for all datasets. Evaluation for only one or a few datasets can be done by specifying `useDatasets`. If `useDatasets` is specified, the argument checking for `trueCluster` and `useCluster` will be enforced to match the cells in the specified datasets.

Usage

```
calcARI(
  object,
  trueCluster,
  useCluster = NULL,
  useDatasets = NULL,
  verbose = getOption("ligerVerbose", TRUE),
  classes.compare = trueCluster
)
```

Arguments

`object`	A liger object, with the clustering result present in cellMeta.
`trueCluster`	Either the name of one variable in `cellMeta(object)` or a factor object with annotation that matches with all cells being considered.
`useCluster`	The name of one variable in `cellMeta(object)`. Default `NULL` uses default clusters.

useDatasets — A character vector of the names, a numeric or logical vector of the index of the datasets to be considered for the purity calculation. Default NULL uses all datasets.

verbose — Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.

classes.compare — [Deprecated/Renamed]. Use trueCluster instead.

Value

A numeric scalar, the ARI of the clustering result indicated by useCluster compared to trueCluster.

References

L. Hubert and P. Arabie (1985) Comparing Partitions, Journal of the Classification, 2, pp. 193-218.

Examples

```
# Assume the true cluster in `pbmcPlot` is "leiden_cluster"
# generate fake new labeling
fake <- sample(1:7, ncol(pbmcPlot), replace = TRUE)
# Insert into cellMeta
pbmcPlot$new <- factor(fake)
calcARI(pbmcPlot, trueCluster = "leiden_cluster", useCluster = "new")

# Now assume we got existing base line annotation only for "stim" dataset
nStim <- ncol(dataset(pbmcPlot, "stim"))
stimTrueLabel <- factor(fake[1:nStim])
# Insert into cellMeta
cellMeta(pbmcPlot, "stim_true_label", useDatasets = "stim") <- stimTrueLabel
# Assume "leiden_cluster" is the clustering result we got and need to be
# evaluated
calcARI(pbmcPlot, trueCluster = "stim_true_label",
        useCluster = "leiden_cluster", useDatasets = "stim")
```

---

calcDatasetSpecificity

Calculate a dataset-specificity score for each factor

Description

This score represents the relative magnitude of the dataset-specific components of each factor's gene loadings compared to the shared components for two datasets. First, for each dataset we calculate the norm of the sum of each factor's shared loadings (W) and dataset-specific loadings (V). We then determine the ratio of these two values and subtract from 1... TODO: finish description.

Usage

```
calcDatasetSpecificity(
  object,
  dataset1,
  dataset2,
  doPlot = FALSE,
  do.plot = doPlot
)
```

Arguments

object	liger object with factorization results.
dataset1	Name of first dataset. Required.
dataset2	Name of second dataset. Required.
doPlot	Logical. Whether to display a barplot of dataset specificity scores (by factor). Default FALSE.
do.plot	**Deprecated**. Use doPlot instead.

Value

List containing three elements.

pct1	Vector of the norm of each metagene factor for dataset1.
pct2	Vector of the norm of each metagene factor for dataset2.
pctSpec	Vector of dataset specificity scores.

---

calcPurity *Calculate purity by comparing two cluster labeling variables*

Description

This function aims at calculating the purity for the clustering result obtained with LIGER and the external clustering (existing "true" annotation). Purity can sometimes be a more useful metric when the clustering to be tested contains more subgroups or clusters than the true clusters. Purity ranges from 0 to 1, with a score of 1 representing a pure, accurate clustering.

The true clustering annotation must be specified as the base line. We suggest setting it to the object cellMeta so that it can be easily used for many other visualization and evaluation functions.

The purity can be calculated for only specified datasets, since true annotation might not be available for all datasets. Evaluation for only one or a few datasets can be done by specifying useDatasets. If useDatasets is specified, the argument checking for trueCluster and useCluster will be enforced to match the cells in the specified datasets.

Usage

```
calcPurity(
  object,
  trueCluster,
  useCluster = NULL,
  useDatasets = NULL,
  verbose = getOption("ligerVerbose", TRUE),
  classes.compare = trueCluster
)
```

Arguments

object	A liger object, with the clustering result present in cellMeta.
trueCluster	Either the name of one variable in cellMeta(object) or a factor object with annotation that matches with all cells being considered.
useCluster	The name of one variable in cellMeta(object). Default NULL uses default clusters.
useDatasets	A character vector of the names, a numeric or logical vector of the index of the datasets to be considered for the purity calculation. Default NULL uses all datasets.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
classes.compare	[Deprecated/Renamed]. Use trueCluster instead.

Value

A numeric scalar, the purity of the clustering result indicated by useCluster compared to trueCluster.

Examples

```
# Assume the true cluster in `pbmcPlot` is "leiden_cluster"
# generate fake new labeling
fake <- sample(1:7, ncol(pbmcPlot), replace = TRUE)
# Insert into cellMeta
pbmcPlot$new <- factor(fake)
calcPurity(pbmcPlot, trueCluster = "leiden_cluster", useCluster = "new")

# Now assume we got existing base line annotation only for "stim" dataset
nStim <- ncol(dataset(pbmcPlot, "stim"))
stimTrueLabel <- factor(fake[1:nStim])
# Insert into cellMeta
cellMeta(pbmcPlot, "stim_true_label", useDatasets = "stim") <- stimTrueLabel
# Assume "leiden_cluster" is the clustering result we got and need to be
# evaluated
calcPurity(pbmcPlot, trueCluster = "stim_true_label",
           useCluster = "leiden_cluster", useDatasets = "stim")
```

closeAllH5 *Close all links (to HDF5 files) of a liger object*

Description

When need to interact with the data embedded in HDF5 files out of the currect R session, the HDF5 files has to be closed in order to be available to other processes.

Usage

```
closeAllH5(object)

## S3 method for class 'liger'
closeAllH5(object)

## S3 method for class 'ligerDataset'
closeAllH5(object)
```

Arguments

object	liger object.

Value

Nothing is returned.

commandDiff *Check difference of two liger command*

Description

Check difference of two liger command

Usage

```
commandDiff(object, cmd1, cmd2)
```

Arguments

object	liger object
cmd1, cmd2	Exact string of command labels. Available options could be viewed with running `commands(object)`.

Value

If any difference found, a character vector summarizing all differences

Examples

```
pbmc <- normalize(pbmc)
pbmc <- normalize(pbmc, log = TRUE, scaleFactor = 1e4)
cmds <- commands(pbmc)
commandDiff(pbmc, cmds[1], cmds[2])
```

`convertOldLiger` *Convert old liger object to latest version*

Description

Convert old liger object to latest version

Usage

```
convertOldLiger(
  object,
  dimredName,
  clusterName = "clusters",
  h5FilePath = NULL
)
```

Arguments

`object`	`liger` object from rliger version <1.99.0
`dimredName`	The name of variable in `cellMeta` slot to store the dimensionality reduction matrix, which originally located in `tsne.coords` slot. Default `"tsne.coords"`.
`clusterName`	The name of variable in `cellMeta` slot to store the clustering assignment, which originally located in `clusters` slot. Default `"clusters"`.
`h5FilePath`	Named list, to specify the path to the H5 file of each dataset if location has been changed. Default `NULL` looks at the file paths stored in object.

Examples

```
## Not run:
# Suppose you have a liger object of old version (<1.99.0)
newLig <- convertOldLiger(oldLig)

## End(Not run)
```

coordinate *Access ligerSpatialDataset coordinate data*

Description

Similar as how default ligerDataset data is accessed.

Usage

```
coordinate(x, dataset)

coordinate(x, dataset, check = TRUE) <- value

## S4 method for signature 'liger,character'
coordinate(x, dataset)

## S4 replacement method for signature 'liger,character'
coordinate(x, dataset, check = TRUE) <- value

## S4 method for signature 'ligerSpatialDataset,missing'
coordinate(x, dataset = NULL)

## S4 replacement method for signature 'ligerSpatialDataset,missing'
coordinate(x, dataset = NULL, check = TRUE) <- value
```

Arguments

x	ligerSpatialDataset object or a liger object.
dataset	Name or numeric index of an spatial dataset.
check	Logical, whether to perform object validity check on setting new value.
value	matrix.

Value

The retrieved coordinate matrix or the updated x object.

createH5LigerDataset *Create on-disk ligerDataset Object*

Description

For convenience, the default `formatType = "10x"` directly fits the structure of cellranger output. `formatType = "anndata"` works for current AnnData H5AD file specification (see Details). If a customized H5 file structure is presented, any of the `rawData`, `indicesName`, `indptrName`, `genesName`, `barcodesName` should be specified accordingly to override the `formatType` preset.

DO make a copy of the H5AD files because rliger functions write to the files and they will not be able to be read back to Python. This will be fixed in the future.

Usage

```
createH5LigerDataset(
  h5file,
  formatType = "10x",
  rawData = NULL,
  normData = NULL,
  scaleData = NULL,
  barcodesName = NULL,
  genesName = NULL,
  indicesName = NULL,
  indptrName = NULL,
  anndataX = "X",
  modal = c("default", "rna", "atac", "spatial", "meth"),
  featureMeta = NULL,
  ...
)
```

Arguments

h5file	Filename of an H5 file
formatType	Select preset of H5 file structure. Default "10X". Alternatively, we also support "anndata" for H5AD files.
rawData, indicesName, indptrName	The path in a H5 file for the raw sparse matrix data. These three types of data stands for the x, i, and p slots of a dgCMatrix-class object. Default NULL uses formatType preset.
normData	The path in a H5 file for the "x" vector of the normalized sparse matrix. Default NULL.
scaleData	The path in a H5 file for the Group that contains the sparse matrix constructing information for the scaled data. Default NULL.
genesName, barcodesName	The path in a H5 file for the gene names and cell barcodes. Default NULL uses formatType preset.
anndataX	The HDF5 path to the raw count data in an H5AD file. See Details. Default "X".
modal	Name of modality for this dataset. Currently options of "default", "rna", "atac", "spatial" and "meth" are supported. Default "default".
featureMeta	Data frame for feature metadata. Default NULL.
...	Additional slot data. See ligerDataset for detail. Given values will be directly placed at corresponding slots.

Details

For H5AD file written from an AnnData object, we allow using formatType = "anndata" for the function to infer the proper structure. However, while a typical AnnData-based analysis tends to in-place update the adata.X attribute and there is no standard/forced convention for where the raw count data, as needed from LIGER, is stored. Therefore, we expose argument anndataX for

specifying this information. The default value "X" looks for adata.X. If the raw data is stored in a layer, e.g. adata.layers['count'], then anndataX = "layers/count". If it is stored to adata.raw.X, then anndataX = "raw/X". If your AnnData object does not have the raw count retained, you will have to go back to the Python work flow to have it inserted at desired object space and re-write the H5AD file, or just go from upstream source files with which the AnnData was originally created.

Value

H5-based ligerDataset object

Examples

```
h5Path <- system.file("extdata/ctrl.h5", package = "rliger")
tempPath <- tempfile(fileext = ".h5")
file.copy(from = h5Path, to = tempPath)
ld <- createH5LigerDataset(tempPath)
```

---

createLiger *Create liger object*

Description

This function allows creating liger object from multiple datasets of various forms (See rawData).

DO make a copy of the H5AD files because rliger functions write to the files and they will not be able to be read back to Python. This will be fixed in the future.

Usage

```
createLiger(
  rawData,
  modal = NULL,
  cellMeta = NULL,
  removeMissing = TRUE,
  addPrefix = "auto",
  formatType = "10X",
  anndataX = "X",
  dataName = NULL,
  indicesName = NULL,
  indptrName = NULL,
  genesName = NULL,
  barcodesName = NULL,
  newH5 = TRUE,
  verbose = getOption("ligerVerbose", TRUE),
  ...,
  raw.data = rawData,
  take.gene.union = NULL,
```

```
  remove.missing = removeMissing,
  format.type = formatType,
  data.name = dataName,
  indices.name = indicesName,
  indptr.name = indptrName,
  genes.name = genesName,
  barcodes.name = barcodesName
)
```

Arguments

rawData	Named list of datasets. Required. Elements allowed include a matrix, a Seurat object, a SingleCellExperiment object, an AnnData object, a ligerDataset object or a filename to an HDF5 file. See detail for HDF5 reading.
modal	Character vector for modality setting. Use one string for all datasets, or the same number of strings as the number of datasets. Currently options of "default", "rna", "atac", "spatial" and "meth" are supported.
cellMeta	data.frame of metadata at single-cell level. Default NULL.
removeMissing	Logical. Whether to remove cells that do not have any counts and features not expressed in any cells from each dataset. Default TRUE.
addPrefix	Logical. Whether to add "<dataset name>_" as a prefix of cell identifiers (e.g. barcodes) to avoid duplicates in multiple libraries (common with 10X data). Default "auto" detects if matrix columns already has the exact prefix or not. Logical value forces the action.
formatType	Select preset of H5 file structure. Current available options are "10x" and "anndata". Can be either a single specification for all datasets or a character vector that match with each dataset.
anndataX	The HDF5 path to the raw count data in an H5AD file. See createH5LigerDataset Details. Default "X".
dataName, indicesName, indptrName	The path in a H5 file for the raw sparse matrix data. These three types of data stands for the x, i, and p slots of a dgCMatrix-class object. Default NULL uses formatType preset.
genesName, barcodesName	The path in a H5 file for the gene names and cell barcodes. Default NULL uses formatType preset.
newH5	When using HDF5 based data and subsets created after removing missing cells/features, whether to create new HDF5 files for the subset. Default TRUE. If FALSE, data will be subset into memory and can be dangerous for large scale analysis.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
...	Additional slot values that should be directly placed in object.
raw.data, remove.missing, format.type, data.name, indices.name, indptr.name, genes.name, barcodes.name	**Deprecated.** See Usage section for replacement.
take.gene.union	Defuncted. Will be ignored.

See Also

createLigerDataset, createH5LigerDataset

Examples

```
# Create from raw count matrices
ctrl.raw <- rawData(pbmc, "ctrl")
stim.raw <- rawData(pbmc, "stim")
pbmc1 <- createLiger(list(ctrl = ctrl.raw, stim = stim.raw))

# Create from H5 files
h5Path <- system.file("extdata/ctrl.h5", package = "rliger")
tempPath <- tempfile(fileext = ".h5")
file.copy(from = h5Path, to = tempPath)
lig <- createLiger(list(ctrl = tempPath))

# Create from other container object
if (requireNamespace("SeuratObject", quietly = TRUE)) {
    ctrl.seu <- SeuratObject::CreateSeuratObject(ctrl.raw)
    stim.seu <- SeuratObject::CreateSeuratObject(stim.raw)
    pbmc2 <- createLiger(list(ctrl = ctrl.seu, stim = stim.seu))
}
```

---

createLigerDataset *Create in-memory ligerDataset object*

Description

Create in-memory ligerDataset object

Usage

```
createLigerDataset(
  rawData = NULL,
  modal = c("default", "rna", "atac", "spatial", "meth"),
  normData = NULL,
  scaleData = NULL,
  featureMeta = NULL,
  ...
)
```

Arguments

rawData, normData, scaleData
: A dgCMatrix-class object for the raw or normalized expression count or a dense matrix of scaled variable gene expression, respectively. Default NULL for all three but at lease one has to be specified.

modal	Name of modality for this dataset. Currently options of "default", "rna", "atac", "spatial" and "meth" are supported. Default "default".
featureMeta	Data frame of feature metadata. Default NULL.
...	Additional slot data. See ligerDataset for detail. Given values will be directly placed at corresponding slots.

See Also

ligerDataset, ligerATACDataset, ligerSpatialDataset, ligerMethDataset

Examples

```
ctrl.raw <- rawData(pbmc, "ctrl")
ctrl.ld <- createLigerDataset(ctrl.raw)
```

---

downsample *Downsample datasets*

Description

This function mainly aims at downsampling datasets to a size suitable for plotting or expensive in-memmory calculation.

Users can balance the sample size of categories of interests with balance. Multi-variable specification to balance is supported, so that at most maxCells cells will be sampled from each combination of categories from the variables. For example, when two datasets are presented and three clusters labeled across them, there would then be at most $2 \times 3 \times maxCells$ cells being selected. Note that "dataset" will automatically be added as one variable when balancing the downsampling. However, if users want to balance the downsampling solely basing on dataset origin, users have to explicitly set balance = "dataset".

Usage

```
downsample(
  object,
  balance = NULL,
  maxCells = 1000,
  useDatasets = NULL,
  seed = 1,
  returnIndex = FALSE,
  ...
)
```

Arguments

object	liger object
balance	Character vector of categorical variable names in cellMeta slot, to subsample maxCells cells from each combination of all specified variables. Default NULL samples maxCells cells from the whole object.
maxCells	Max number of cells to sample from the grouping based on balance.
useDatasets	Index selection of datasets to include Default NULL for using all datasets.
seed	Random seed for reproducibility. Default 1.
returnIndex	Logical, whether to only return the numeric index that can subset the original object instead of a subset object. Default FALSE.
...	Arguments passed to subsetLiger, where cellIdx is occupied by internal implementation.

Value

By default, a subset of liger object. Alternatively when returnIndex = TRUE, a numeric vector to be used with the original object.

Examples

```
# Subsetting an object
pbmc <- downsample(pbmc)
# Creating a subsetting index
sampleIdx <- downsample(pbmcPlot, balance = "leiden_cluster",
                        maxCells = 10, returnIndex = TRUE)
plotClusterDimRed(pbmcPlot, cellIdx = sampleIdx)
```

---

exportInteractTrack *Export predicted gene-pair interaction*

Description

Export the predicted gene-pair interactions calculated by upstream function linkGenesAndPeaks into an Interact Track file which is compatible with UCSC Genome Browser.

Usage

```
exportInteractTrack(
  corrMat,
  pathToCoords,
  useGenes = NULL,
  outputPath = getwd()
)
```

Arguments

corrMat	A sparse matrix of correlation with peak names as rows and gene names as columns.
pathToCoords	Path to the gene coordinates file.
useGenes	Character vector of gene names to be exported. Default NULL uses all genes available in corrMat.
outputPath	Path of filename where the output file will be stored. If a folder, a file named "Interact_Track.bed" will be created. Default current working directory.

Value

No return value. A file located at outputPath will be created.

Examples

```
bmmc <- normalize(bmmc)
bmmc <- selectGenes(bmmc)
bmmc <- scaleNotCenter(bmmc)
if (requireNamespace("RcppPlanc", quietly = TRUE) &&
    requireNamespace("GenomicRanges", quietly = TRUE) &&
    requireNamespace("IRanges", quietly = TRUE) &&
    requireNamespace("psych", quietly = TRUE)) {
    bmmc <- runINMF(bmmc)
    bmmc <- quantileNorm(bmmc)
    bmmc <- normalizePeak(bmmc)
    bmmc <- imputeKNN(bmmc, reference = "atac", queries = "rna")
    corr <- linkGenesAndPeaks(
        bmmc, useDataset = "rna",
        pathToCoords = system.file("extdata/hg19_genes.bed", package = "rliger")
    )
    resultPath <- tempfile()
    exportInteractTrack(
        corrMat = corr,
        pathToCoords = system.file("extdata/hg19_genes.bed", package = "rliger"),
        outputPath = resultPath
    )
    head(read.table(resultPath, skip = 1))
}
```

---

getFactorMarkers *Find shared and dataset-specific markers*

Description

Applies various filters to genes on the shared (W) and dataset-specific (V) components of the factorization, before selecting those which load most significantly on each factor (in a shared or dataset-specific way).

Usage

```
getFactorMarkers(
  object,
  dataset1,
  dataset2,
  factorShareThresh = 10,
  datasetSpecificity = NULL,
  logFCThresh = 1,
  pvalThresh = 0.05,
  nGenes = 30,
  printGenes = FALSE,
  verbose = getOption("ligerVerbose", TRUE),
  factor.share.thresh = factorShareThresh,
  dataset.specificity = datasetSpecificity,
  log.fc.thresh = logFCThresh,
  pval.thresh = pvalThresh,
  num.genes = nGenes,
  print.genes = printGenes
)
```

Arguments

`object`	liger object with factorization results.
`dataset1`	Name of first dataset. Required.
`dataset2`	Name of second dataset. Required
`factorShareThresh`	Numeric. Only factors with a dataset specificity less than or equal to this threshold will be used. Default `10`.
`datasetSpecificity`	Numeric vector. Pre-calculated dataset specificity if available. Length should match number of all factors available. Default `NULL` automatically calculates with `calcDatasetSpecificity`.
`logFCThresh`	Numeric. Lower log-fold change threshold for differential expression in markers. Default `1`.
`pvalThresh`	Numeric. Upper p-value threshold for Wilcoxon rank test for gene expression. Default `0.05`.
`nGenes`	Integer. Max number of genes to report for each dataset. Default `30`.
`printGenes`	Logical. Whether to print ordered markers passing logFC, UMI and frac thresholds, when `verbose = TRUE`. Default `FALSE`.
`verbose`	Logical. Whether to show information of the progress. Default `getOption("ligerVerbose")` or `TRUE` if users have not set.
`factor.share.thresh`, `dataset.specificity`, `log.fc.thresh`, `pval.thresh`, `num.genes`, `print.genes`	**Deprecated**. See Usage section for replacement.

Value

A list object consisting of the following entries:

`[value of 'dataset1']` data.frame of dataset1-specific markers

`shared` data.frame of shared markers

`[value of 'dataset1']` data.frame of dataset2-specific markers

`num_factors_V1` A frequency table indicating the number of factors each marker appears, in dataset1

`num_factors_V2` A frequency table indicating the number of factors each marker appears, in dataset2

Examples

```
library(dplyr)
result <- getFactorMarkers(pbmcPlot, dataset1 = "ctrl", dataset2 = "stim")
print(class(result))
print(names(result))
result$shared %>% group_by(factor_num) %>% top_n(2, logFC)
```

---

getProportionMito *Calculate proportion mitochondrial contribution*

Description

Calculates proportion of mitochondrial contribution based on raw or normalized data.

Usage

```
getProportionMito(object, use.norm = FALSE, pattern = "^mt-")
```

Arguments

`object`	`liger` object.
`use.norm`	**Deprecated** Whether to use cell normalized data in calculating contribution. Default `FALSE`.
`pattern`	Regex pattern for identifying mitochondrial genes. Default `"^mt-"` for mouse.

Value

Named vector containing proportion of mitochondrial contribution for each cell.

Note

`getProportionMito` will be deprecated because `runGeneralQC` generally covers and expands its use case.

Examples

```
# Example dataset does not contain MT genes, expected to see a message
pbmc$mito <- getProportionMito(pbmc)
```

H5Apply *Apply function to chunks of H5 data in ligerDataset object*

Description

h5 calculation wrapper, that runs specified calculation with on-disk matrix in chunks

Usage

```
H5Apply(
  object,
  FUN,
  init = NULL,
  useData = c("rawData", "normData"),
  chunkSize = 1000,
  verbose = getOption("ligerVerbose"),
  ...
)
```

Arguments

object	A ligerDataset object.
FUN	A function that is applied to each chunk. See detail for restrictions.
init	Initialized result if it need to be updated iteratively. Default NULL.
useData	The slot name of the data to be processed. Choose from "rawData", "normData", "scaleData". Default "rawData".
chunkSize	Number if columns to be included in each chunk. Default 1000.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") which is TRUE if users have not set.
...	Other arguments to be passed to FUN.

Details

The FUN function has to have the first four arguments ordered by:

1. **chunk data:** A sparse matrix (dgCMatrix-class) containing maximum chunkSize columns.
2. **x-vector index:** The index that subscribes the vector of x slot of a dgCMatrix, which points to the values in each chunk. Mostly used when need to write a new sparse matrix to H5 file.
3. **cell index:** The column index of each chunk out of the whole original matrix

4. **Initialized result:** A customized object, the value passed to `H5Apply(init)` argument will be passed here in the first iteration. And the returned value of `FUN` will be iteratively passed here in next chunk iterations. So it is important to keep the object structure of the returned value consistent with `init`.

No default value to these four arguments should be pre-defined because `H5Apply` will automatically generate the input.

`importPBMC` *Import prepared dataset publically available*

Description

These are functions to download example datasets that are subset from public data.

- **PBMC** - Downsampled from GSE96583, Kang et al, Nature Biotechnology, 2018. Contains two scRNAseq datasets.
- **BMMC** - Downsampled from GSE139369, Granja et al, Nature Biotechnology, 2019. Contains two scRNAseq datasets and one scATAC data.
- **CGE** - Downsampled from GSE97179, Luo et al, Science, 2017. Contains one scRNAseq dataset and one DNA methylation data.

Usage

```
importPBMC(
  dir = getwd(),
  overwrite = FALSE,
  method = "libcurl",
  verbose = getOption("ligerVerbose", TRUE),
  ...
)

importBMMC(
  dir = getwd(),
  overwrite = FALSE,
  method = "libcurl",
  verbose = getOption("ligerVerbose", TRUE),
  ...
)

importCGE(
  dir = getwd(),
  overwrite = FALSE,
  method = "libcurl",
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

dir	Path to download datasets. Default current working directory getwd().
overwrite	Logical, if a file exists at corresponding download location, whether to re-download or directly use this file. Default FALSE.
method	method argument directly passed to download.file. Using "libcurl" while other options might not work depending on platform.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
...	Additional arguments passed to download.file

Value

Constructed liger object with QC performed and missing data removed.

Examples

```
pbmc <- importPBMC()
bmmc <- importBMMC()
cge <- importCGE()
```

---

imputeKNN *Impute the peak counts from gene expression data referring to an ATAC dataset after integration*

Description

This function is designed for creating peak data for a dataset with only gene expression. This function uses quantile normalized cell factor loading to find nearest neighbors between cells from the queried dataset (without peak) and cells from reference dataset (with peak). And then impute the peak for the former basing on the weight. Therefore, the reference dataset selected must be of "atac" modality setting.

Usage

```
imputeKNN(
  object,
  reference,
  queries = NULL,
  nNeighbors = 20,
  weight = TRUE,
  norm = TRUE,
  scale = FALSE,
  verbose = getOption("ligerVerbose", TRUE),
```

```
  ...,
  knn_k = nNeighbors
)
```

Arguments

object	liger object with aligned factor loading computed in advance.
reference	Name of a dataset containing peak data to impute into query dataset(s).
queries	Names of datasets to be augmented by imputation. Should not include reference. Default NULL uses all datasets except the reference.
nNeighbors	The maximum number of nearest neighbors to search. Default 20.
weight	Logical. Whether to use KNN distances as weight matrix. Default FALSE.
norm	Logical. Whether to normalize the imputed data. Default TRUE.
scale	Logical. Whether to scale but not center the imputed data. Default TRUE.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
...	Optional arguments to be passed to normalize when norm = TRUE.
knn_k	**Deprecated**. See Usage section for replacement.

Value

The input object where queried ligerDataset objects in datasets slot are replaced. These datasets will all be converted to ligerATACDataset class with an additional slot rawPeak to store the imputed peak counts, and normPeak for normalized imputed peak counts if norm = TRUE.

Examples

```
bmmc <- normalize(bmmc)
bmmc <- selectGenes(bmmc, datasets.use = "rna")
bmmc <- scaleNotCenter(bmmc)
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    bmmc <- runINMF(bmmc, k = 20)
    bmmc <- quantileNorm(bmmc)
    bmmc <- normalizePeak(bmmc)
    bmmc <- imputeKNN(bmmc, reference = "atac", queries = "rna")
}
```

---

is.newLiger *Check if given liger object if under new implementation*

Description

Check if given liger object if under new implementation

Usage

```
is.newLiger(object)
```

Arguments

object	A liger object

Value

TRUE if the version of object is later than or equal to 1.99.0. Otherwise FALSE. It raises an error if input object is not of liger class.

Examples

```
is.newLiger(pbmc) # TRUE
```

---

isH5Liger *Check if a liger or ligerDataset object is made of HDF5 file*

Description

Check if a liger or ligerDataset object is made of HDF5 file

Usage

```
isH5Liger(object, dataset = NULL)
```

Arguments

object	A liger or ligerDataset object.
dataset	If object is of liger class, check a specific dataset. If NULL, Check if all datasets are made of HDF5 file. Default NULL.

Value

TRUE or FALSE for the specified check.

Examples

```
isH5Liger(pbmc)
isH5Liger(pbmc, "ctrl")
ctrl <- dataset(pbmc, "ctrl")
isH5Liger(ctrl)
```

liger-class *liger class*

Description

`liger` object is the main data container for LIGER analysis in R. The slot `datasets` is a list where each element should be a ligerDataset object containing dataset specific information, such as the expression matrices. The other parts of liger object stores information that can be shared across the analysis, such as the cell metadata and factorization result matrices.

This manual provides explanation to the `liger` object structure as well as usage of class-specific methods. Please see detail sections for more information.

For `liger` objects created with older versions of rliger package, please try updating the objects individually with `convertOldLiger`.

Usage

```
datasets(x, check = NULL)

datasets(x, check = TRUE) <- value

dataset(x, dataset = NULL)

dataset(x, dataset, type = NULL, qc = TRUE) <- value

cellMeta(
  x,
  columns = NULL,
  useDatasets = NULL,
  cellIdx = NULL,
  as.data.frame = FALSE,
  ...
)

cellMeta(
  x,
  columns = NULL,
  useDatasets = NULL,
  cellIdx = NULL,
  inplace = FALSE,
  check = FALSE
) <- value

defaultCluster(x, useDatasets = NULL, ...)

defaultCluster(x, name = NULL, useDatasets = NULL, ...) <- value
```

```
dimReds(x)

dimReds(x) <- value

dimRed(x, name = NULL, useDatasets = NULL, cellIdx = NULL, ...)

dimRed(x, name = NULL, useDatasets = NULL, cellIdx = NULL, ...) <- value

defaultDimRed(x, useDatasets = NULL, cellIdx = NULL)

defaultDimRed(x) <- value

varFeatures(x)

varFeatures(x, check = TRUE) <- value

varUnsharedFeatures(x, dataset = NULL)

varUnsharedFeatures(x, dataset, check = TRUE) <- value

commands(x, funcName = NULL, arg = NULL)

## S4 method for signature 'liger'
show(object)

## S4 method for signature 'liger'
dim(x)

## S4 method for signature 'liger'
dimnames(x)

## S4 replacement method for signature 'liger,list'
dimnames(x) <- value

## S4 method for signature 'liger'
datasets(x, check = NULL)

## S4 replacement method for signature 'liger,logical'
datasets(x, check = TRUE) <- value

## S4 replacement method for signature 'liger,missing'
datasets(x, check = TRUE) <- value

## S4 method for signature 'liger,character_OR_NULL'
dataset(x, dataset = NULL)

## S4 method for signature 'liger,missing'
dataset(x, dataset = NULL)
```

```
## S4 method for signature 'liger,numeric'
dataset(x, dataset = NULL)

## S4 replacement method for signature 'liger,character,missing,ANY,ligerDataset'
dataset(x, dataset, type = NULL, qc = TRUE) <- value

## S4 replacement method for signature 'liger,character,ANY,ANY,matrixLike'
dataset(x, dataset, type = c("rawData", "normData"), qc = FALSE) <- value

## S4 replacement method for signature 'liger,character,missing,ANY,NULL'
dataset(x, dataset, type = NULL, qc = TRUE) <- value

## S3 method for class 'liger'
names(x)

## S3 replacement method for class 'liger'
names(x) <- value

## S3 method for class 'liger'
length(x)

## S3 method for class 'liger'
lengths(x, use.names = TRUE)

## S4 method for signature 'liger,NULL'
cellMeta(
  x,
  columns = NULL,
  useDatasets = NULL,
  cellIdx = NULL,
  as.data.frame = FALSE,
  ...
)

## S4 method for signature 'liger,character'
cellMeta(
  x,
  columns = NULL,
  useDatasets = NULL,
  cellIdx = NULL,
  as.data.frame = FALSE,
  ...
)

## S4 method for signature 'liger,missing'
cellMeta(
  x,
```

```
  columns = NULL,
  useDatasets = NULL,
  cellIdx = NULL,
  as.data.frame = FALSE,
  ...
)

## S4 replacement method for signature 'liger,missing'
cellMeta(x, columns = NULL, useDatasets = NULL, cellIdx = NULL, check = FALSE) <- value

## S4 replacement method for signature 'liger,character'
cellMeta(
  x,
  columns = NULL,
  useDatasets = NULL,
  cellIdx = NULL,
  inplace = TRUE,
  check = FALSE
) <- value

## S4 method for signature 'liger'
rawData(x, dataset = NULL)

## S4 replacement method for signature 'liger,ANY,ANY,matrixLike_OR_NULL'
rawData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'liger,ANY,ANY,H5D'
rawData(x, dataset = NULL, check = TRUE) <- value

## S4 method for signature 'liger'
normData(x, dataset = NULL)

## S4 replacement method for signature 'liger,ANY,ANY,matrixLike_OR_NULL'
normData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'liger,ANY,ANY,H5D'
normData(x, dataset = NULL, check = TRUE) <- value

## S4 method for signature 'liger,ANY'
scaleData(x, dataset = NULL)

## S4 replacement method for signature 'liger,ANY,ANY,matrixLike_OR_NULL'
scaleData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'liger,ANY,ANY,H5D'
scaleData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'liger,ANY,ANY,H5Group'
```

```
scaleData(x, dataset = NULL, check = TRUE) <- value

## S4 method for signature 'liger,character'
scaleUnsharedData(x, dataset = NULL)

## S4 method for signature 'liger,numeric'
scaleUnsharedData(x, dataset = NULL)

## S4 replacement method for signature 'liger,ANY,ANY,matrixLike_OR_NULL'
scaleUnsharedData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'liger,ANY,ANY,H5D'
scaleUnsharedData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'liger,ANY,ANY,H5Group'
scaleUnsharedData(x, dataset = NULL, check = TRUE) <- value

## S4 method for signature 'liger,ANY,ANY,ANY'
getMatrix(
  x,
 slot = c("rawData", "normData", "scaleData", "scaleUnsharedData", "H", "V", "U", "A",
    "B", "W", "H.norm"),
  dataset = NULL,
  returnList = FALSE
)

## S4 method for signature 'liger,ANY'
getH5File(x, dataset = NULL)

## S3 replacement method for class 'liger'
x[[i]] <- value

## S3 method for class 'liger'
x$name

## S3 replacement method for class 'liger'
x$name <- value

## S4 method for signature 'liger'
defaultCluster(x, useDatasets = NULL, droplevels = FALSE, ...)

## S4 replacement method for signature 'liger,ANY,ANY,character'
defaultCluster(x, name = NULL, useDatasets = NULL, ...) <- value

## S4 replacement method for signature 'liger,ANY,ANY,factor'
defaultCluster(x, name = NULL, useDatasets = NULL, droplevels = TRUE, ...) <- value

## S4 replacement method for signature 'liger,ANY,ANY,NULL'
```

```
defaultCluster(x, name = NULL, useDatasets = NULL, ...) <- value

## S4 method for signature 'liger'
dimReds(x)

## S4 replacement method for signature 'liger,list'
dimReds(x) <- value

## S4 method for signature 'liger,missing_OR_NULL'
dimRed(x, name = NULL, useDatasets = NULL, cellIdx = NULL, ...)

## S4 method for signature 'liger,index'
dimRed(x, name = NULL, useDatasets = NULL, cellIdx = NULL, ...)

## S4 replacement method for signature 'liger,index,ANY,ANY,NULL'
dimRed(x, name = NULL, useDatasets = NULL, cellIdx = NULL, ...) <- value

## S4 replacement method for signature 'liger,character,ANY,ANY,matrixLike'
dimRed(
  x,
  name = NULL,
  useDatasets = NULL,
  cellIdx = NULL,
  asDefault = NULL,
  inplace = FALSE,
  ...
) <- value

## S4 method for signature 'liger'
defaultDimRed(x, useDatasets = NULL, cellIdx = cellIdx)

## S4 replacement method for signature 'liger,character'
defaultDimRed(x) <- value

## S4 method for signature 'liger'
varFeatures(x)

## S4 replacement method for signature 'liger,ANY,character'
varFeatures(x, check = TRUE) <- value

## S4 method for signature 'liger,ANY'
varUnsharedFeatures(x, dataset = NULL)

## S4 replacement method for signature 'liger,ANY,ANY,character'
varUnsharedFeatures(x, dataset, check = TRUE) <- value

## S3 method for class 'liger'
fortify(model, data, ...)
```

```
## S3 method for class 'liger'
c(...)

## S4 method for signature 'liger'
commands(x, funcName = NULL, arg = NULL)

## S4 method for signature 'ligerDataset,missing'
varUnsharedFeatures(x, dataset = NULL)

## S4 replacement method for signature 'ligerDataset,missing,ANY,character'
varUnsharedFeatures(x, dataset = NULL, check = TRUE) <- value
```

Arguments

x, object, model	A liger object
check	Logical, whether to perform object validity check on setting new value. Users are not supposed to set FALSE here.
value	Metadata value to be inserted
dataset	Name or numeric index of a dataset
type	When using dataset<- with a matrix like value, specify what type the matrix is. Choose from "rawData", "normData" or "scaleData".
qc	Logical, whether to perform general qc on added new dataset.
columns	The names of available variables in cellMeta slot. When as.data.frame = TRUE, please use variable names after coercion.
useDatasets	Setter or getter method should only apply on cells in specified datasets. Any valid character, numeric or logical subscriber is acceptable. Default NULL works with all datasets.
cellIdx	Valid cell subscription to subset retrieved variables. Default NULL uses all cells.
as.data.frame	Logical, whether to apply as.data.frame on the subscription. Default FALSE.
...	See detailed sections for explanation.
inplace	For cellMeta<- method, when columns is for existing variable and useDatasets or cellIdx indicate partial insertion to the object, whether to by default (TRUE) in-place insert value into the variable for selected cells or to replace the whole variable with non-selected part left as NA.
name	The name of available variables in cellMeta slot or the name of a new variable to store.
funcName, arg	See Command records section.
use.names	Whether returned vector should be named with dataset names.
slot	Name of slot to retrieve matrix from. Options shown in Usage.
returnList	Logical, whether to force return a list even when only one dataset-specific matrix (i.e. expression matrices, H, V or U) is requested. Default FALSE.
i	Name or numeric index of cell meta variable to be replaced

droplevels	Whether to remove unused cluster levels from the factor object fetched by defaultCluster(). Default FALSE.
asDefault	Whether to set the inserted dimension reduction matrix as default for visualization methods. Default NULL sets it when no default has been set yet, otherwise does not change current default.
data	fortify method required argument. Not used.

Value

See detailed sections for explanetion.

Input liger object updated with replaced/new variable in cellMeta(x).

Slots

datasets list of ligerDataset objects. Use generic dataset, dataset<-, datasets or datasets<- to interact with. See detailed section accordingly.

cellMeta DFrame object for cell metadata. Pre-existing metadata, QC metrics, cluster labeling, low-dimensional embedding and etc. are all stored here. Use generic cellMeta, cellMeta<-, $, [[]] or [[]]<- to interact with. See detailed section accordingly.

varFeatures Character vector of feature names. Use generic varFeatures or varFeatures<- to interact with. See detailed section accordingly.

W Matrix of gene loading for each factor. See runIntegration.

H.norm Matrix of aligned factor loading for each cell. See quantileNorm and runIntegration.

commands List of ligerCommand objects. Record of analysis. Use commands to retrieve information. See detailed section accordingly.

uns List for unstructured meta-info of analyses or presets.

version Record of version of rliger package

Dataset access

datasets() method only accesses the datasets slot, the list of ligerDataset objects. dataset() method accesses a single dataset, with subsequent cell metadata updates and checks bonded when adding or modifying a dataset. Therefore, when users want to modify something inside a ligerDataset while no cell metadata change should happen, it is recommended to use: datasets(x)[[name]] <- ligerD for efficiency, though the result would be the same as dataset(x, name) <- ligerD.

length() and names() methods are implemented to access the number and names of datasets. names<- method is supported for modifying dataset names, with taking care of the "dataset" variable in cell metadata.

Matrix access

For liger object, rawData(), normData, scaleData() and scaleUnsharedData() methods are exported for users to access the corresponding feature expression matrix with specification of one dataset. For retrieving a type of matrix from multiple datasets, please use getMatrix() method.

When only one matrix is expected to be retrieved by getMatrix(), the matrix itself will be returned. A list will be returned if multiple matrices is requested (by querying multiple datasets) or returnList is set to TRUE.

Cell metadata access

Three approaches are provided for access of cell metadata. A generic function `cellMeta` is implemented with plenty of options and multi-variable accessibility. Besides, users can use double-bracket (e.g. `ligerObj[[varName]]`) or dollor-sign (e.g. `ligerObj$nUMI`) to access or modify single variables.

For users' convenience of generating a customized ggplot with available cell metadata, the S3 method `fortify.liger` is implemented. With this under the hook, users can create simple ggplots by directly starting with `ggplot(ligerObj, aes(...))` where cell metadata variables can be directly thrown into `aes()`.

Special partial metadata insertion is implemented specifically for mapping categorical annotation from sub-population (subset object) back to original experiment (full-size object). For example, when sub-clustering and annotation is done for a specific cell-type of cells (stored in `subobj`) subset from an experiment (stored as `obj`), users can do `cellMeta(obj, "sub_ann", cellIdx = colnames(subobj)) <- subobj$sub_ann` to map the value back, leaving other cells non-annotated with NAs. Plotting with this variable will then also show NA cells with default grey color. Furthermore, sub-clustering labels for other cell types can also be mapped to the same variable. For example, `cellMeta(obj, "sub_ann",cellIdx = colnames(subobj2)) <- subobj2$sub_ann`. As long as the labeling variables are stored as factor class (categorical), the levels (category names) will be properly handled and merged. Other situations follow the R default behavior (e.g. categories might be converted to integer numbers if mapped to numerical variable in the original object). Note that this feature is only available with using the generic function `cellMeta` but not with the `` `[[` `` or `` `$` `` accessing methods due to syntax reasons.

The generic `defaultCluster` works as both getter and setter. As a setter, users can do `defaultCluster(obj) <- "existingVariableName"` to set a categorical variable as default cluster used for visualization or downstream analysis. Users can also do `defaultCluster(obj,"newVarName") <- factorOfLabels` to push new labeling into the object and set as default. For getter method, the function returns a factor object of the default cluster labeling. Argument `useDatasets` can be used for requiring that given or retrieved labeling should match with cells in specified datasets. We generally don't recommend setting `"dataset"` as a default cluster because it is a preserved (always existing) field in metadata and can lead to meaningless result when running analysis that utilizes both clustering information and the dataset source information.

Dimension reduction access

Currently, low-dimensional representaion of cells, presented as dense matrices, are all stored in `dimReds` slot, and can totally be accessed with generics `dimRed` and `dimRed<-`. Adding a dimRed to the object looks as simple as `dimRed(obj, "name") <- matrixLike`. It can be retrieved back with `dimRed(obj, "name")`. Similar to having a default cluster labeling, we also constructed the feature of default dimRed. It can be set with `defaultDimRed(obj) <- "existingMatLikeVar"` and the matrix can be retrieved with `defaultDimRed(obj)`.

Variable feature access

The `varFeatures` slot allows for character vectors of gene names. `varFeatures(x)` returns this vector and `value` for `varFeatures<-` method has to be a character vector or `NULL`. The replacement method, when `check = TRUE` performs checks on gene name consistency check across the `scaleData`, `H`, `V` slots of inner `ligerDataset` objects as well as the `W` and `H.norm` slots of the input `liger` object.

Command records

rliger functions, that perform calculation and update the `liger` object, will be recorded in a `ligerCommand` object and stored in the `commands` slot, a list, of `liger` object. Method `commands()` is implemented to retrieve or show the log history. Running with `funcName = NULL` (default) returns all command labels. Specifying `funcName` allows partial matching to all command labels and returns a subset list (of `ligerCommand` object) of matches (or the `ligerCommand` object if only one match found). If `arg` is further specified, a subset list of parameters from the matches will be returned. For example, requesting a list of resolution values used in all louvain cluster attempts: `commands(ligerObj, "louvainCluster","resolution")`

Dimensionality

For a `liger` object, the column orientation is assigned for cells. Due to the data structure, it is hard to define a row index for the `liger` object, which might contain datasets that vary in number of genes.

Therefore, for `liger` objects, `dim` and `dimnames` returns `NA`/`NULL` for rows and total cell counts/barcodes for the columns.

For direct call of `dimnames<-` method, `value` should be a list with `NULL` as the first element and valid cell identifiers as the second element. For `colnames<-` method, the character vector of cell identifiers. `rownames<-` method is not applicable.

Subsetting

For more detail of subsetting a `liger` object or a ligerDataset object, please check out `subsetLiger` and `subsetLigerDataset`. Here, we set the S4 method "single-bracket" `[` as a quick wrapper to subset a `liger` object. Note that `j` serves as cell subscriptor which can be any valid index refering the collection of all cells (i.e. `rownames(cellMeta(obj))`). While `i`, the feature subscriptor can only be character vector because the features for each dataset can vary. `...` arugments are passed to `subsetLiger` so that advanced options are allowed.

Combining multiple liger object

The list of `datasets` slot, the rows of `cellMeta` slot and the list of `commands` slot will be simply concatenated. Variable features in `varFeatures` slot will be taken a union. The W and $H.norm$ matrices are not taken into account for now.

Examples

```
# Methods for base generics
pbmcPlot
print(pbmcPlot)
dim(pbmcPlot)
ncol(pbmcPlot)
colnames(pbmcPlot)[1:5]
pbmcPlot[varFeatures(pbmcPlot)[1:10], 1:10]
names(pbmcPlot)
length(pbmcPlot)

# rliger generics
```

```
## Retrieving dataset(s), replacement methods available
datasets(pbmcPlot)
dataset(pbmcPlot, "ctrl")
dataset(pbmcPlot, 2)

## Retrieving cell metadata, replacement methods available
cellMeta(pbmcPlot)
head(pbmcPlot[["nUMI"]])

## Retrieving dimemtion reduction matrix
head(dimRed(pbmcPlot, "UMAP"))

## Retrieving variable features, replacement methods available
varFeatures(pbmcPlot)

## Command record/history
pbmcPlot <- scaleNotCenter(pbmcPlot)
commands(pbmcPlot)
commands(pbmcPlot, funcName = "scaleNotCenter")

# S3 methods
pbmcPlot2 <- pbmcPlot
names(pbmcPlot2) <- paste0(names(pbmcPlot), 2)
c(pbmcPlot, pbmcPlot2)

library(ggplot2)
ggplot(pbmcPlot, aes(x = UMAP_1, y = UMAP_2)) + geom_point()
cellMeta(pbmc)
# Add new variable
pbmc[["newVar"]] <- 1
cellMeta(pbmc)
# Change existing variable
pbmc[["newVar"]][1:3] <- 1:3
cellMeta(pbmc)
```

`ligerATACDataset-class` *Subclass of ligerDataset for ATAC modality*

Description

Inherits from ligerDataset class. Contained slots can be referred with the link.

Slots

`rawPeak` sparse matrix

`normPeak` sparse matrix

`ligerCommand-class` *ligerCommand object: Record the input and time of a LIGER function call*

Description

ligerCommand object: Record the input and time of a LIGER function call

Usage

```
## S4 method for signature 'ligerCommand'
show(object)
```

Arguments

`object`	A `ligerCommand` object

Slots

`funcName` Name of the function

`time` A time stamp object

`call` A character string converted from system call

`parameters` List of all arguments except the liger object. Large object are summarized to short string.

`objSummary` List of attributes of the liger object as a snapshot when command is operated.

`ligerVersion` Character string converted from `packageVersion("rliger")`.

`dependencyVersion` Named character vector of version number, if any dependency library has a chance to be included by the function. A dependency might only be invoked under certain conditions, such as using an alternative algorithm, which a call does not actually reach to, but it would still be included for this call.

Examples

```
pbmc <- normalize(pbmc)
cmd <- commands(pbmc, "normalize")
cmd
```

ligerDataset-class *ligerDataset class*

Description

Object for storing dastaset specific information. Will be embedded within a higher level liger object

Usage

```
rawData(x, dataset = NULL)

rawData(x, dataset = NULL, check = TRUE) <- value

normData(x, dataset = NULL)

normData(x, dataset = NULL, check = TRUE) <- value

scaleData(x, dataset = NULL)

scaleData(x, dataset = NULL, check = TRUE) <- value

scaleUnsharedData(x, dataset = NULL)

scaleUnsharedData(x, dataset = NULL, check = TRUE) <- value

getMatrix(x, slot = "rawData", dataset = NULL, returnList = FALSE)

h5fileInfo(x, info = NULL)

h5fileInfo(x, info = NULL, check = TRUE) <- value

getH5File(x, dataset = NULL)

## S4 method for signature 'ligerDataset,missing'
getH5File(x, dataset = NULL)

featureMeta(x, check = NULL)

featureMeta(x, check = TRUE) <- value

## S4 method for signature 'ligerDataset'
show(object)

## S4 method for signature 'ligerDataset'
dim(x)

## S4 method for signature 'ligerDataset'
```

```
dimnames(x)

## S4 replacement method for signature 'ligerDataset,list'
dimnames(x) <- value

## S4 method for signature 'ligerDataset'
rawData(x, dataset = NULL)

## S4 replacement method for signature 'ligerDataset,ANY,ANY,matrixLike_OR_NULL'
rawData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'ligerDataset,ANY,ANY,H5D'
rawData(x, dataset = NULL, check = TRUE) <- value

## S4 method for signature 'ligerDataset'
normData(x, dataset = NULL)

## S4 replacement method for signature 'ligerDataset,ANY,ANY,matrixLike_OR_NULL'
normData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'ligerDataset,ANY,ANY,H5D'
normData(x, dataset = NULL, check = TRUE) <- value

## S4 method for signature 'ligerDataset,missing'
scaleData(x, dataset = NULL)

## S4 replacement method for signature 'ligerDataset,ANY,ANY,matrixLike_OR_NULL'
scaleData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'ligerDataset,ANY,ANY,H5D'
scaleData(x, dataset = NULL, check = TRUE) <- value

## S4 replacement method for signature 'ligerDataset,ANY,ANY,H5Group'
scaleData(x, dataset = NULL, check = TRUE) <- value

## S4 method for signature 'ligerDataset,missing'
scaleUnsharedData(x, dataset = NULL)

## S4 replacement method for signature 'ligerDataset,missing,ANY,matrixLike_OR_NULL'
scaleUnsharedData(x, check = TRUE) <- value

## S4 replacement method for signature 'ligerDataset,missing,ANY,H5D'
scaleUnsharedData(x, check = TRUE) <- value

## S4 replacement method for signature 'ligerDataset,missing,ANY,H5Group'
scaleUnsharedData(x, check = TRUE) <- value

## S4 method for signature 'ligerDataset,ANY,missing,missing'
```

```
getMatrix(
  x,
 slot = c("rawData", "normData", "scaleData", "scaleUnsharedData", "H", "V", "U", "A",
    "B"),
  dataset = NULL
)

## S4 method for signature 'ligerDataset'
h5fileInfo(x, info = NULL)

## S4 replacement method for signature 'ligerDataset'
h5fileInfo(x, info = NULL, check = TRUE) <- value

## S4 method for signature 'ligerDataset'
featureMeta(x, check = NULL)

## S4 replacement method for signature 'ligerDataset'
featureMeta(x, check = TRUE) <- value

## S3 method for class 'ligerDataset'
cbind(x, ..., deparse.level = 1)
```

Arguments

x, object	A ligerDataset object.
dataset	Not applicable for ligerDataset methods.
check	Whether to perform object validity check on setting new value.
value	See detail sections for requirements
slot	The slot name when using getMatrix.
returnList	Not applicable for ligerDataset methods.
info	Name of the entry in h5fileInfo slot.
...	See detailed sections for explanation.
deparse.level	Not used here.

Slots

rawData Raw data.

normData Normalized data

scaleData Scaled data, usually with subset variable features

scaleUnsharedData Scaled data of features not shared with other datasets

varUnsharedFeatures Variable features not shared with other datasets

V matrix

A matrix

B matrix

`H` matrix

`U` matrix

`h5fileInfo` list

`featureMeta` Feature metadata, DataFrame

`colnames` character

`rownames` character

Matrix access

For `ligerDataset` object, `rawData()`, `normData`, `scaleData()` and `scaleUnsharedData()` methods are exported for users to access the corresponding feature expression matrix. Replacement methods are also available to modify the slots.

For other matrices, such as the H and V, which are dataset specific, please use `getMatrix()` method with specifying slot name. Directly accessing slot with `@` is generally not recommended.

H5 file and information access

A `ligerDataset` object has a slot called `h5fileInfo`, which is a list object. The first element is called `$H5File`, which is an `H5File` class object and is the connection to the input file. The second element is `$filename` which stores the absolute path of the H5 file in the current machine. The third element `$formatType` stores the name of preset being used, if applicable. The other following keys pair with paths in the H5 file that point to specific data for constructing a feature expression matrix.

`h5fileInfo()` method access the list described above and simply retrieves the corresponding value. When `info = NULL`, returns the whole list. When `length(info) == 1`, returns the requested list value. When more info requested, returns a subset list.

The replacement method modifies the list elements and corresponding slot value (if applicable) at the same time. For example, running `h5fileInfo(obj, "rawData") <- newPath` not only updates the list, but also updates the `rawData` slot with the `H5D` class data at "newPath" in the `H5File` object.

`getH5File()` is a wrapper and is equivalent to `h5fileInfo(obj, "H5File")`.

Feature metadata access

A slot `featureMeta` is included for each `ligerDataset` object. This slot requires a `DataFrame-class` object, which is the same as `cellMeta` slot of a liger object. However, the associated S4 methods only include access to the whole table for now. Internal information access follows the same way as data.frame operation. For example, `featureMeta(ligerD)$nCell` or `featureMeta(ligerD)[varFeatures(ligerObj), "gene_var"]`.

Dimensionality

For a `ligerDataset` object, the column orientation is assigned for cells and rows are for features. Therefore, for `ligerDataset` objects, `dim()` returns a numeric vector of two numbers which are number of features and number of cells. `dimnames()` returns a list of two character vectors, which are the feature names and the cell barcodes.

For direct call of `dimnames<-` method, `value` should be a list with a character vector of feature names as the first element and cell identifiers as the second element. For `colnames<-` method, the character vector of cell identifiers. For `rownames<-` method, the character vector of feature names.

Subsetting

For more detail of subsetting a `liger` object or a ligerDataset object, please check out `subsetLiger` and `subsetLigerDataset`. Here, we set the S3 method "single-bracket" `[` as a quick wrapper to subset a `ligerDataset` object. `i` and `j` serves as feature and cell subscriptor, respectively, which can be any valid index refering the available features and cells in a dataset. `...` arugments are passed to `subsetLigerDataset` so that advanced options are allowed.

Concatenate ligerDataset

`cbind()` method is implemented for concatenating `ligerDataset` objects by cells. When applying, all feature expression matrix will be merged with taking a union of all features for the rows.

Examples

```
ctrl <- dataset(pbmc, "ctrl")

# Methods for base generics
ctrl
print(ctrl)
dim(ctrl)
ncol(ctrl)
nrow(ctrl)
colnames(ctrl)[1:5]
rownames(ctrl)[1:5]
ctrl[1:5, 1:5]

# rliger generics
## raw data
m <- rawData(ctrl)
class(m)
dim(m)
## normalized data
pbmc <- normalize(pbmc)
ctrl <- dataset(pbmc, "ctrl")
m <- normData(ctrl)
class(m)
dim(m)
## scaled data
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
ctrl <- dataset(pbmc, "ctrl")
m <- scaleData(ctrl)
class(m)
dim(m)
n <- scaleData(pbmc, "ctrl")
identical(m, n)
## Any other matrices
```

```
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    pbmc <- runOnlineINMF(pbmc, k = 20, minibatchSize = 100)
    ctrl <- dataset(pbmc, "ctrl")
    V <- getMatrix(ctrl, "V")
    V[1:5, 1:5]
    Vs <- getMatrix(pbmc, "V")
    length(Vs)
    names(Vs)
    identical(Vs$ctrl, V)
}
```

`ligerMethDataset-class`
Subclass of ligerDataset for Methylation modality

Description

Inherits from ligerDataset class. Contained slots can be referred with the link. `scaleNotCenter` applied on datasets of this class will automatically be taken by reversing the normalized data instead of scaling the variable features.

`ligerRNADataset-class` *Subclass of ligerDataset for RNA modality*

Description

Inherits from ligerDataset class. Contained slots can be referred with the link. This subclass does not have any different from the default `ligerDataset` class except the class name.

`ligerSpatialDataset-class`
Subclass of ligerDataset for Spatial modality

Description

Inherits from ligerDataset class. Contained slots can be referred with the link.

Slots

`coordinate` dense matrix

`ligerToSeurat` *Convert between liger and Seurat object*

Description

For converting a liger object to a Seurat object, the `rawData`, `normData`, and `scaleData` from each dataset, the `cellMeta`, `H.norm` and `varFeatures` slot will be included. Compatible with V4 and V5. It is not recommended to use this conversion if your liger object contains datasets from various modalities.

Usage

```
ligerToSeurat(
  object,
  assay = NULL,
  identByDataset = FALSE,
  merge = FALSE,
  nms = NULL,
  renormalize = NULL,
  use.liger.genes = NULL,
  by.dataset = identByDataset
)
```

Arguments

`object`	A liger object to be converted
`assay`	Name of assay to store the data. Default `NULL` detects by dataset modality. If the object contains various modality, default to `"LIGER"`. Default dataset modality setting is understood as `"RNA"`.
`identByDataset`	Logical, whether to combine dataset variable and default cluster labeling to set the Idents. Default `FALSE`.
`merge`	Logical, whether to merge layers of different datasets into one. Not recommended. Default `FALSE`.
`nms`	[Defunct] Will be ignored because new object structure does not have related problem.
`renormalize`	[Defunct] Will be ignored because since Seurat V5, layers of data can exist at the same time and it is better to left it for users to do it by themselves.
`use.liger.genes`	[Defunct] Will be ignored and will always set LIGER variable features to the place.
`by.dataset`	[Deprecated]. Use `identByDataset` instead.

Value

Always returns Seurat object(s) of the latest version. By default a Seurat object with split layers, e.g. with layers like "counts.ctrl" and "counts.stim". If `merge = TRUE`, return a single Seurat object with layers for all datasets merged.

Examples

```
seu <- ligerToSeurat(pbmc)
```

`linkGenesAndPeaks` *Linking genes to putative regulatory elements*

Description

Evaluate the relationships between pairs of genes and peaks based on specified distance metric. Usually used for inferring the correlation between gene expression and imputed peak counts for datasets without the modality originally (i.e. applied to `imputeKNN` result).

Usage

```
linkGenesAndPeaks(
  object,
  useDataset,
  pathToCoords,
  useGenes = NULL,
  method = c("spearman", "pearson", "kendall"),
  alpha = 0.05,
  verbose = getOption("ligerVerbose", TRUE),
  path_to_coords = pathToCoords,
  genes.list = useGenes,
  dist = method
)
```

Arguments

`object`	A liger object, with datasets that is of ligerATACDataset class in the `datasets` slot.
`useDataset`	Name of one dataset, with both normalized gene expression and normalized peak counts available.
`pathToCoords`	Path tothe gene coordinates file, usually a BED file.
`useGenes`	Character vector of gene names to be tested. Default `NULL` uses all genes available in `useDataset`.
`method`	Choose the type of correlation to calculate, from `"spearman"`, `"pearson"` and `"kendall"`. Default `"spearman"`
`alpha`	Numeric, significance threshold for correlation p-value. Peak-gene correlations with p-values below this threshold are considered significant. Default `0.05`.
`verbose`	Logical. Whether to show information of the progress. Default `getOption("ligerVerbose")` or `TRUE` if users have not set.
`path_to_coords, genes.list, dist`	**Deprecated**. See Usage section for replacement.

Value

A sparse matrix with peak names as rows and gene names as columns, with each element indicating the correlation between peak i and gene j, 0 if the gene and peak are not significantly linked.

See Also

imputeKNN

Examples

```
if (requireNamespace("RcppPlanc", quietly = TRUE) &&
    requireNamespace("GenomicRanges", quietly = TRUE) &&
    requireNamespace("IRanges", quietly = TRUE) &&
    requireNamespace("psych", quietly = TRUE)) {
    bmmc <- normalize(bmmc)
    bmmc <- selectGenes(bmmc)
    bmmc <- scaleNotCenter(bmmc)
    bmmc <- runINMF(bmmc, miniBatchSize = 100)
    bmmc <- quantileNorm(bmmc)
    bmmc <- normalizePeak(bmmc)
    bmmc <- imputeKNN(bmmc, reference = "atac", queries = "rna")
    corr <- linkGenesAndPeaks(
        bmmc, useDataset = "rna",
        pathToCoords = system.file("extdata/hg19_genes.bed", package = "rliger")
    )
}
```

---

louvainCluster-deprecated

[Deprecated] Louvain algorithm for community detection

Description

After quantile normalization, users can additionally run the Louvain algorithm for community detection, which is widely used in single-cell analysis and excels at merging small clusters into broad cell classes.

Arguments

object	liger object. Should run quantile_norm before calling.
k	The maximum number of nearest neighbours to compute. (default 20)
resolution	Value of the resolution parameter, use a value above (below) 1.0 if you want to obtain a larger (smaller) number of communities. (default 1.0)
prune	Sets the cutoff for acceptable Jaccard index when computing the neighborhood overlap for the SNN construction. Any edges with values less than or equal to this will be set to 0 and removed from the SNN graph. Essentially sets the strigency of pruning (0 — no pruning, 1 — prune everything). (default 1/15)

eps	The error bound of the nearest neighbor search. (default 0.1)
nRandomStarts	Number of random starts. (default 10)
nIterations	Maximal number of iterations per random start. (default 100)
random.seed	Seed of the random number generator. (default 1)
verbose	Print messages (TRUE by default)
dims.use	Indices of factors to use for clustering. Default NULL uses all available factors.

Value

object with refined cluster assignment updated in "louvain_cluster" variable in cellMeta slot. Can be fetched with object$louvain_cluster

See Also

rliger-deprecated

---

makeFeatureMatrix *Fast calculation of feature count matrix*

Description

Fast calculation of feature count matrix

Usage

```
makeFeatureMatrix(bedmat, barcodes)
```

Arguments

bedmat	A feature count list generated by bedmap
barcodes	A list of barcodes

Value

A feature count matrix with features as rows and barcodes as columns

Examples

```
## Not run:
gene.counts <- makeFeatureMatrix(genes.bc, barcodes)
promoter.counts <- makeFeatureMatrix(promoters.bc, barcodes)
samnple <- gene.counts + promoter.counts

## End(Not run)
```

makeInteractTrack-deprecated

[Deprecated] Export predicted gene-pair interaction

Description

Export the predicted gene-pair interactions calculated by upstream function `linkGenesAndPeaks` into an Interact Track file which is compatible with UCSC Genome Browser.

Arguments

`corr.mat`	A sparse matrix of correlation with peak names as rows and gene names as columns.
`path_to_coords`	Path to the gene coordinates file.
`genes.list`	Character vector of gene names to be exported. Default `NULL` uses all genes available in `corrMat`.
`output_path`	Path of filename where the output file will be stored. If a folder, a file named `"Interact_Track.bed"` will be created. Default current working directory.

Value

No return value. A file located at `outputPath` will be created.

See Also

`rliger-deprecated`, `exportInteractTrack`

makeRiverplot-deprecated

[Deprecated] Generate a river (Sankey) plot

Description

Creates a riverplot to show how separate cluster assignments from two datasets map onto a joint clustering. The joint clustering is by default the object clustering, but an external one can also be passed in. Uses the riverplot package to construct riverplot object and then plot.

Arguments

`object`	`liger` object. Should run quantileAlignSNF before calling.
`cluster1`	Cluster assignments for dataset 1. Note that cluster names should be distinct across datasets.
`cluster2`	Cluster assignments for dataset 2. Note that cluster names should be distinct across datasets.

cluster_consensus
: Optional external consensus clustering (to use instead of object clusters)

min.frac
: Minimum fraction of cluster for edge to be shown (default 0.05).

min.cells
: Minumum number of cells for edge to be shown (default 10).

river.yscale
: y-scale to pass to riverplot – scales the edge with values by this factor, can be used to squeeze vertically (default 1).

river.lty
: Line style to pass to riverplot (default 0).

river.node_margin
: Node_margin to pass to riverplot – how much vertical space to keep between the nodes (default 0.1).

label.cex
: Size of text labels (default 1).

label.col
: Color of text labels (defualt "black").

lab.srt
: Angle of text labels (default 0).

river.usr
: Coordinates at which to draw the plot in form (x0, x1, y0, y1).

node.order
: Order of clusters in each set (list with three vectors of ordinal numbers). By default will try to automatically order them appropriately.

Value

`object` with refined cluster assignment updated in `"louvain_cluster"` variable in `cellMeta` slot. Can be fetched with `object$louvain_cluster`

See Also

`rliger-deprecated`

`mapCellMeta` *Create new variable from categories in cellMeta*

Description

Designed for fast variable creation when a new variable is going to be created from existing variable. For example, multiple samples can be mapped to the same study design condition, clusters can be mapped to cell types.

Usage

```
mapCellMeta(object, from, newTo = NULL, ...)
```

Arguments

object
: A liger object.

from
: The name of the original variable to be mapped from.

newTo
: The name of the new variable to store the mapped result. Default `NULL` returns the new variable (factor class).

...
: Mapping criteria, argument names are original existing categories in the `from` and values are new categories in the new variable.

Value

When newTo = NULL, a factor object of the new variable. Otherwise, the input object with variable newTo updated in cellMeta(object).

Examples

```
pbmc <- mapCellMeta(pbmc, from = "dataset", newTo = "modal",
                    ctrl = "rna", stim = "rna")
```

---

mergeH5 *Merge hdf5 files*

Description

This function merges hdf5 files generated from different libraries (cell ranger by default) before they are preprocessed through Liger pipeline.

Usage

```
mergeH5(
  file.list,
  library.names,
  new.filename,
  format.type = "10X",
  data.name = NULL,
  indices.name = NULL,
  indptr.name = NULL,
  genes.name = NULL,
  barcodes.name = NULL
)
```

Arguments

file.list	List of path to hdf5 files.
library.names	Vector of library names (corresponding to file.list)
new.filename	String of new hdf5 file name after merging (default new.h5).
format.type	string of HDF5 format (10X CellRanger by default).
data.name	Path to the data values stored in HDF5 file.
indices.name	Path to the indices of data points stored in HDF5 file.
indptr.name	Path to the pointers stored in HDF5 file.
genes.name	Path to the gene names stored in HDF5 file.
barcodes.name	Path to the barcodes stored in HDF5 file.

Value

Directly generates newly merged hdf5 file.

Examples

```
## Not run:
# For instance, we want to merge two datasets saved in HDF5 files (10X
# CellRanger) paths to datasets: "library1.h5","library2.h5"
# dataset names: "lib1", "lib2"
# name for output HDF5 file: "merged.h5"
mergeH5(list("library1.h5","library2.h5"), c("lib1","lib2"), "merged.h5")

## End(Not run)
```

`mergeSparseAll` *Merge matrices while keeping the union of rows*

Description

`mergeSparseAll` takes in a list of DGEs, with genes as rows and cells as columns, and merges them into a single DGE. Also adds `libraryNames` to colnames from each DGE if expected to be overlap (common with 10X barcodes). Values in `rawData` or `normData` slot of a ligerDataset object can be processed with this.

For a list of dense matrices, usually the values in `scaleData` slot of a ligerDataset object, please use `mergeDenseAll` which works in the same way.

Usage

```
mergeSparseAll(
  datalist,
  libraryNames = NULL,
  mode = c("union", "intersection")
)

mergeDenseAll(datalist, libraryNames = NULL)
```

Arguments

`datalist`	List of dgCMatrix for `mergeSparseAll` or a list of matrix for `mergeDenseAll`.
`libraryNames`	Character vector to be added as the prefix for the barcodes in each matrix in `datalist`. Length should match with the number of matrices. Default `NULL` do not modify the barcodes.
`mode`	Whether to take the `"union"` or `"intersection"` of features when merging. Default `"union"`.

Value

dgCMatrix or matrix with all barcodes in `datalist` as columns and the union of genes in `datalist` as rows.

Examples

```
rawDataList <- getMatrix(pbmc, "rawData")
merged <- mergeSparseAll(rawDataList, libraryNames = names(pbmc))
```

`modalOf` *Return preset modality of a ligerDataset object or that of all datasets in a liger object*

Description

Return preset modality of a ligerDataset object or that of all datasets in a liger object

Usage

```
modalOf(object)
```

Arguments

`object`	a ligerDataset object or a liger object

Value

A single character of modality setting value for ligerDataset `object`, or a named vector for liger object, where the names are dataset names.

Examples

```
modalOf(pbmc)
ctrl <- dataset(pbmc, "ctrl")
modalOf(ctrl)
ctrl.atac <- as.ligerDataset(ctrl, modal = "atac")
modalOf(ctrl.atac)
```

`normalize` *Normalize raw counts data*

Description

Perform library size normalization on raw counts input. As for the preprocessing step of iNMF integration, by default we don't multiply the normalized values with a scale factor, nor do we take the log transformation. Applicable S3 methods can be found in Usage section.

`normalizePeak` is designed for datasets of "atac" modality, i.e. stored in ligerATACDataset. S3 method for various container object is not supported yet due to difference in architecture design.

Usage

```
normalize(object, ...)

## S3 method for class 'dgCMatrix'
normalize(object, log = FALSE, scaleFactor = NULL, ...)

## S3 method for class 'ligerDataset'
normalize(object, chunk = 1000, verbose = getOption("ligerVerbose", TRUE), ...)

## S3 method for class 'liger'
normalize(
  object,
  useDatasets = NULL,
  verbose = getOption("ligerVerbose", TRUE),
  format.type = NULL,
  remove.missing = NULL,
  ...
)

## S3 method for class 'Seurat'
normalize(object, assay = NULL, layer = "counts", save = "ligerNormData", ...)

normalizePeak(
  object,
  useDatasets = NULL,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

object	liger object
...	Arguments to be passed to S3 methods. The "liger" method calls the "ligerDataset" method, which then calls "dgCMatrix" method. normalizePeak directly calls normalize.dgCMatrix.
log	Logical. Whether to do a log(x + 1) transform on the normalized data. Default TRUE.
scaleFactor	Numeric. Scale the normalized expression value by this factor before transformation. NULL for not scaling. Default 1e4.
chunk	Integer. Number of maximum number of cells in each chunk when working on HDF5 file based ligerDataset. Default 1000.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
useDatasets	A character vector of the names, a numeric or logical vector of the index of the datasets to be normalized. Should specify ATACseq datasets when using normalizePeak. Default NULL normalizes all valid datasets.

format.type, remove.missing
: **Deprecated**. The functionality of these is covered through other parts of the whole workflow and is no long needed. Will be ignored if specified.

assay
: Name of assay to use. Default NULL uses current active assay.

layer
: Where the input raw counts should be from. Default "counts". For older Seurat, always retrieve from counts slot.

save
: For Seurat>=4.9.9, the name of layer to store normalized data. Default "ligerNormData". For older Seurat, stored to data slot.

Value

Updated object.

- dgCMatrix method - Returns processed dgCMatrix object
- ligerDataset method - Updates the normData slot of the object
- liger method - Updates the normData slot of chosen datasets
- Seurat method - Adds a named layer in chosen assay (V5), or update the data slot of the chosen assay (<=V4)
- normalizePeak - Updates the normPeak slot of chosen datasets.

Examples

```
pbmc <- normalize(pbmc)
```

---

online_iNMF-deprecated

[Deprecated] Perform online iNMF on scaled datasets

Description

Please turn to runOnlineINMF or runIntegration.

Perform online integrative non-negative matrix factorization to represent multiple single-cell datasets in terms of H, W, and V matrices. It optimizes the iNMF objective function using online learning (non-negative least squares for H matrix, hierarchical alternating least squares for W and V matrices), where the number of factors is set by k. The function allows online learning in 3 scenarios: (1) fully observed datasets; (2) iterative refinement using continually arriving datasets; and (3) projection of new datasets without updating the existing factorization. All three scenarios require fixed memory independent of the number of cells.

For each dataset, this factorization produces an H matrix (cells by k), a V matrix (k by genes), and a shared W matrix (k by genes). The H matrices represent the cell factor loadings. W is identical among all datasets, as it represents the shared components of the metagenes across datasets. The V matrices represent the dataset-specific components of the metagenes.

Arguments

object	liger object with data stored in HDF5 files. Should normalize, select genes, and scale before calling.
X_new	List of new datasets for scenario 2 or scenario 3. Each list element should be the name of an HDF5 file.
projection	Perform data integration by shared metagene (W) projection (scenario 3). (default FALSE)
W.init	Optional initialization for W. (default NULL)
V.init	Optional initialization for V (default NULL)
H.init	Optional initialization for H (default NULL)
A.init	Optional initialization for A (default NULL)
B.init	Optional initialization for B (default NULL)
k	Inner dimension of factorization–number of metagenes (default 20). A value in the range 20-50 works well for most analyses.
lambda	Regularization parameter. Larger values penalize dataset-specific effects more strongly (ie. alignment should increase as lambda increases). We recommend always using the default value except possibly for analyses with relatively small differences (biological replicates, male/female comparisons, etc.) in which case a lower value such as 1.0 may improve reconstruction quality. (default 5.0).
max.epochs	Maximum number of epochs (complete passes through the data). (default 5)
miniBatch_max_iters	Maximum number of block coordinate descent (HALS algorithm) iterations to perform for each update of W and V (default 1). Changing this parameter is not recommended.
miniBatch_size	Total number of cells in each minibatch (default 5000). This is a reasonable default, but a smaller value such as 1000 may be necessary for analyzing very small datasets. In general, minibatch size should be no larger than the number of cells in the smallest dataset.
h5_chunk_size	Chunk size of input hdf5 files (default 1000). The chunk size should be no larger than the batch size.
seed	Random seed to allow reproducible results (default 123).
verbose	Print progress bar/messages (TRUE by default)

Value

liger object with H, W, V, A and B slots set.

optimizeALS-deprecated

[Deprecated] Perform iNMF on scaled datasets

Description

Please turn to `runINMF` **or** `runIntegration`.

Perform integrative non-negative matrix factorization to return factorized H, W, and V matrices. It optimizes the iNMF objective function using block coordinate descent (alternating non-negative least squares), where the number of factors is set by k. TODO: include objective function equation here in documentation (using deqn)

For each dataset, this factorization produces an H matrix (cells by k), a V matrix (k by genes), and a shared W matrix (k by genes). The H matrices represent the cell factor loadings. W is held consistent among all datasets, as it represents the shared components of the metagenes across datasets. The V matrices represent the dataset-specific components of the metagenes.

Arguments

`object`	`liger` object. Should normalize, select genes, and scale before calling.
`k`	Inner dimension of factorization (number of factors). Run suggestK to determine appropriate value; a general rule of thumb is that a higher k will be needed for datasets with more sub-structure.
`lambda`	Regularization parameter. Larger values penalize dataset-specific effects more strongly (ie. alignment should increase as lambda increases). Run suggestLambda to determine most appropriate value for balancing dataset alignment and agreement (default 5.0).
`thresh`	Convergence threshold. Convergence occurs when |obj0-obj|/(mean(obj0,obj)) < thresh. (default 1e-6)
`max.iters`	Maximum number of block coordinate descent iterations to perform (default 30).
`nrep`	Number of restarts to perform (iNMF objective function is non-convex, so taking the best objective from multiple successive initializations is recommended). For easier reproducibility, this increments the random seed by 1 for each consecutive restart, so future factorizations of the same dataset can be run with one rep if necessary. (default 1)
`H.init`	Initial values to use for H matrices. (default NULL)
`W.init`	Initial values to use for W matrix (default NULL)
`V.init`	Initial values to use for V matrices (default NULL)
`rand.seed`	Random seed to allow reproducible results (default 1).
`print.obj`	Print objective function values after convergence (default FALSE).
`verbose`	Print progress bar/messages (TRUE by default)
`...`	Arguments passed to other methods

Value

`liger` object with H, W, and V slots set.

See Also

`rliger-deprecated`

`optimizeNewData` *Perform factorization for new data*

Description

Uses an efficient strategy for updating that takes advantage of the information in the existing factorization. Assumes that variable features are presented in the new datasets. Two modes are supported (controlled by `merge`):

- Append new data to existing datasets specified by `useDatasets`. Here the existing V matrices for the target datasets will directly be used as initialization, and new H matrices for the merged matrices will be initialized accordingly.
- Set new data as new datasets. Initial V matrices for them will be copied from datasets specified by `useDatasets`, and new H matrices will be initialized accordingly.

Usage

```
optimizeNewData(
  object,
  dataNew,
  useDatasets,
  merge = TRUE,
  lambda = NULL,
  nIteration = 30,
  seed = 1,
  verbose = getOption("ligerVerbose"),
  new.data = dataNew,
  which.datasets = useDatasets,
  add.to.existing = merge,
  max.iters = nIteration,
  thresh = NULL
)
```

Arguments

`object`	A liger object. Should have integrative factorization performed e.g. (`runINMF`) in advance.
`dataNew`	Named list of **raw count** matrices, genes by cells.

useDatasets — Selection of datasets to append new data to if merge = TRUE, or the datasets to inherit V matrices from and initialize the optimization when merge = FALSE. Should match the length and order of dataNew.

merge — Logical, whether to add the new data to existing datasets or treat as totally new datasets (i.e. calculate new V matrices). Default TRUE.

lambda — Numeric regularization parameter. By default NULL, this will use the lambda value used in the latest factorization.

nIteration — Number of block coordinate descent iterations to perform. Default 30.

seed — Random seed to allow reproducible results. Default 1. Used by runINMF factorization.

verbose — Logical. Whether to show information of the progress. Default getOption("ligerVerbose") which is TRUE if users have not set.

new.data, which.datasets, add.to.existing, max.iters — These arguments are now replaced by others and will be removed in the future. Please see usage for replacement.

thresh — **Deprecated**. New implementation of iNMF does not require a threshold for convergence detection. Setting a large enough nIteration will bring it to convergence.

Value

object with W slot updated with the new W matrix, and the H and V slots of each ligerDataset object in the datasets slot updated with the new dataset specific H and V matrix, respectively.

See Also

runINMF, optimizeNewK, optimizeNewLambda

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
# Only running a few iterations for fast examples
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    pbmc <- runINMF(pbmc, k = 20, nIteration = 2)
    # Create fake new data by increasing all non-zero count in "ctrl" by 1,
    # and make unique cell identifiers
    ctrl2 <- rawData(dataset(pbmc, "ctrl"))
    ctrl2@x <- ctrl2@x + 1
    colnames(ctrl2) <- paste0(colnames(ctrl2), 2)
    pbmcNew <- optimizeNewData(pbmc, dataNew = list(ctrl2 = ctrl2),
                               useDatasets = "ctrl", nIteration = 2)
}
```

`optimizeNewK` *Perform factorization for new value of k*

Description

This uses an efficient strategy for updating that takes advantage of the information in the existing factorization. It is most recommended for values of `kNew` smaller than current value (`k`, which is set when running `runINMF`), where it is more likely to speed up the factorization.

Usage

```
optimizeNewK(
  object,
  kNew,
  lambda = NULL,
  nIteration = 30,
  seed = 1,
  verbose = getOption("ligerVerbose"),
  k.new = kNew,
  max.iters = nIteration,
  rand.seed = seed,
  thresh = NULL
)
```

Arguments

`object`	A liger object. Should have integrative factorization performed e.g. (`runINMF`) in advance.
`kNew`	Number of factors of factorization.
`lambda`	Numeric regularization parameter. By default `NULL`, this will use the lambda value used in the latest factorization.
`nIteration`	Number of block coordinate descent iterations to perform. Default `30`.
`seed`	Random seed to allow reproducible results. Default `1`. Used by `runINMF` factorization and initialization only when if `kNew` is greater than `k`.
`verbose`	Logical. Whether to show information of the progress. Default `getOption("ligerVerbose")` which is `TRUE` if users have not set.
`k.new`, `max.iters`, `rand.seed`	These arguments are now replaced by others and will be removed in the future. Please see usage for replacement.
`thresh`	**Deprecated**. New implementation of iNMF does not require a threshold for convergence detection. Setting a large enough `nIteration` will bring it to convergence.

Value

`object` with `W` slot updated with the new W matrix, and the `H` and `V` slots of each ligerDataset object in the `datasets` slot updated with the new dataset specific H and V matrix, respectively.

See Also

runINMF, optimizeNewLambda, optimizeNewData

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
# Only running a few iterations for fast examples
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    pbmc <- runINMF(pbmc, k = 20, nIteration = 2)
    pbmc <- optimizeNewK(pbmc, kNew = 25, nIteration = 2)
}
```

---

optimizeNewLambda *Perform factorization for new lambda value*

Description

Uses an efficient strategy for updating that takes advantage of the information in the existing factorization; always uses previous k. Recommended mainly when re-optimizing for higher lambda and when new lambda value is significantly different; otherwise may not return optimal results.

Usage

```
optimizeNewLambda(
  object,
  lambdaNew,
  nIteration = 30,
  seed = 1,
  verbose = getOption("ligerVerbose"),
  new.lambda = lambdaNew,
  max.iters = nIteration,
  rand.seed = seed,
  thresh = NULL
)
```

Arguments

object	liger object. Should have integrative factorization (e.g. runINMF) performed in advance.
lambdaNew	Numeric regularization parameter. Larger values penalize dataset-specific effects more strongly.
nIteration	Number of block coordinate descent iterations to perform. Default 30.
seed	Random seed to allow reproducible results. Default 1. Used by runINMF factorization.

verbose — Logical. Whether to show information of the progress. Default `getOption("ligerVerbose")` which is `TRUE` if users have not set.

new.lambda, max.iters, rand.seed — These arguments are now replaced by others and will be removed in the future. Please see usage for replacement.

thresh — **Deprecated**. New implementation of iNMF does not require a threshold for convergence detection. Setting a large enough `nIteration` will bring it to convergence.

Value

Input `object` with optimized factorization values updated. including the `W` matrix in liger object, and `H` and `V` matrices in each ligerDataset object in the `datasets` slot.

See Also

`runINMF`, `optimizeNewK`, `optimizeNewData`

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    # Only running a few iterations for fast examples
    pbmc <- runINMF(pbmc, k = 20, nIteration = 2)
    pbmc <- optimizeNewLambda(pbmc, lambdaNew = 5.5, nIteration = 2)
}
```

---

optimizeSubset *Perform factorization for subset of data*

Description

Uses an efficient strategy for updating that takes advantage of the information in the existing factorization.

Usage

```
optimizeSubset(
  object,
  clusterVar = NULL,
  useClusters = NULL,
  lambda = NULL,
  nIteration = 30,
  cellIdx = NULL,
  scaleDatasets = NULL,
  seed = 1,
```

```
  verbose = getOption("ligerVerbose"),
  cell.subset = cellIdx,
  cluster.subset = useClusters,
  max.iters = nIteration,
  datasets.scale = scaleDatasets,
  thresh = NULL
)
```

Arguments

object	liger object. Should have integrative factorization (e.g. runINMF) performed in advance.
clusterVar, useClusters	Together select the clusters to subset the object conveniently. clusterVar is the name of variable in cellMeta(object) and useClusters should be vector of names of clusters in the variable. clusterVar is by default the default cluster (See runCluster, or defaultCluster at "Cell metadata access"). Users can otherwise select cells explicitly with cellIdx for complex conditions. useClusters overrides cellIdx.
lambda	Numeric regularization parameter. By default NULL, this will use the lambda value used in the latest factorization.
nIteration	Maximum number of block coordinate descent iterations to perform. Default 30.
cellIdx	Valid index vector that applies to the whole object. See subsetLiger for requirement. Default NULL.
scaleDatasets	Names of datasets to re-scale after subsetting. Default NULL does not re-scale.
seed	Random seed to allow reproducible results. Default 1. Used by runINMF factorization.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") which is TRUE if users have not set.
cell.subset, cluster.subset, max.iters, datasets.scale	These arguments are now replaced by others and will be removed in the future. Please see usage for replacement.
thresh	**Deprecated**. New implementation of iNMF does not require a threshold for convergence detection. Setting a large enough nIteration will bring it to convergence.

Value

Subset object with factorization matrices optimized, including the W matrix in liger object, and W and V matrices in each ligerDataset object in the datasets slot. scaleData in the ligerDataset objects of datasets specified by scaleDatasets will also be updated to reflect the subset.

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
```

```
pbmc <- scaleNotCenter(pbmc)
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    # Only running a few iterations for fast examples
    pbmc <- runINMF(pbmc, k = 20, nIteration = 2)
    pbmc <- optimizeSubset(pbmc, cellIdx = sort(sample(ncol(pbmc), 200)),
                           nIteration = 2)
}
```

---

pbmc *liger object of PBMC subsample data with Control and Stimulated datasets*

Description

liger object of PBMC subsample data with Control and Stimulated datasets

Usage

```
pbmc
```

Format

liger object with two datasets named by "ctrl" and "stim".

Source

https://www.nature.com/articles/nbt.4042

References

Hyun Min Kang and et. al., Nature Biotechnology, 2018

---

pbmcPlot *liger object of PBMC subsample data with plotting information available*

Description

This data was generated from data "pbmc" with default parameter integration pipeline: normalize, selectGenes, scaleNotCenter, runINMF, runCluster, runUMAP. To minimize the object size distributed with the package, rawData and scaleData were removed. Genes are downsampled to the top 50 variable genes, for smaller normData and W matrix.

Usage

```
pbmcPlot
```

Format

liger object with two datasets named by "ctrl" and "stim".

Source

https://www.nature.com/articles/nbt.4042

References

Hyun Min Kang and et. al., Nature Biotechnology, 2018

---

plotCellViolin *Generate violin/box plot(s) using liger object*

Description

This function allows for using available cell metadata, feature expression or factor loading to generate violin plot, and grouping the data with available categorical cell metadata. Available categorical cell metadata can be used to form the color annotation. When it is different from the grouping, it forms a nested grouping. Multiple y-axis variables are allowed from the same specification of `slot`, and this returns a list of violin plot for each. Users can further split the plot(s) by grouping on cells (e.g. datasets).

Usage

```
plotCellViolin(
  object,
  y,
  groupBy = NULL,
  slot = c("cellMeta", "rawData", "normData", "scaleData", "H.norm", "H"),
  yFunc = NULL,
  cellIdx = NULL,
  colorBy = NULL,
  splitBy = NULL,
  titles = NULL,
  ...
)
```

Arguments

`object`	liger object
`y`	Available variable name in `slot` to look for the value to visualize.
`groupBy, colorBy`	Available variable name in `cellMeta` slot to look for categorical grouping. See details. Default `NULL` produces no grouping and all-black graphic elements.
`slot`	Choose the slot to find the `y` variable. See Details. Default `"cellMeta"`.

yFunc	A function object that expects a vector/factor/data.frame retrieved by y as the only input, and returns an object of the same size, so that the y-axis is replaced by this output. Useful when, for example, users need to scale the gene expression shown on plot.
cellIdx	Character, logical or numeric index that can subscribe cells. Missing or NULL for all cells.
splitBy	Character vector of categorical variable names in cellMeta slot. Split all cells by groupings on this/these variable(s) to produce a violin plot containing only the cells in each group. Default NULL.
titles	Title text. A character scalar or a character vector with as many elements as multiple plots are supposed to be generated. Default NULL.
...	More plot setting arguments. See .ggCellViolin and .ggplotLigerTheme.

Details

Available option for slot include: "cellMeta", "rawData", "normData", "scaleData", "H.norm" and "H". When "rawData", "normData" or "scaleData", y has to be a character vector of feature names. When "H.norm" or "H", colorBy can be any valid index to select one factor of interests. Note that character index follows "Factor_[k]" format, with replacing [k] with an integer.

When "cellMeta", y has to be an available column name in the table. Note that, for y as well as groupBy, colorBy and splitBy since a matrix object is feasible in cellMeta table, using a column (e.g. named as "column1" in a certain matrix (e.g. named as "matrixVar") should follow the syntax of "matrixVar.column1". When the matrix does not have a "colname" attribute, the subscription goes with "matrixVar.V1", "matrixVar.V2" and etc. These are based on the nature of as.data.frame method on a DataFrame object.

groupBy is basically send to ggplot2::aes(x), while colorBy is for the "colour" aesthetics. Specifying colorBy without groupBy visually creates grouping but there will not be varying values on the x-axis, so boxWidth will be forced to the same value as violinWidth under this situation.

Value

A ggplot object when a single plot is intended. A list of ggplot objects, when multiple y variables and/or splitBy are set. When plotly = TRUE, all ggplot objects become plotly (htmlwidget) objects.

Examples

```
plotCellViolin(pbmcPlot, y = "nUMI", groupBy = "dataset", slot = "cellMeta")
plotCellViolin(pbmcPlot, y = "nUMI", groupBy = "leiden_cluster",
               slot = "cellMeta", splitBy = "dataset",
               colorBy = "leiden_cluster",
               box = TRUE, dot = TRUE,
               ylab = "Total counts per cell",
               colorValues = RColorBrewer::brewer.pal(8, "Set1"))
plotCellViolin(pbmcPlot, y = "S100A8", slot = "normData",
               yFunc = function(x) log2(10000*x + 1),
               groupBy = "dataset", colorBy = "leiden_cluster",
               box = TRUE, ylab = "S100A8 Expression")
```

`plotClusterFactorDot` *Make dot plot of factor loading in cell groups*

Description

This function produces dot plots. Each column represent a group of cells specified by `groupBy`, each row is a factor specified by `useDims`. The color of dots reflects mean of factor loading of specified factors in each cell group and sizes reflects the percentage of cells that have loadings of a factor in a group. We utilize ComplexHeatmap for simplified management of adding annotation and slicing subplots. This was inspired by the implementation in scCustomize.

Usage

```
plotClusterFactorDot(
  object,
  groupBy = NULL,
  useDims = NULL,
  useRaw = FALSE,
  splitBy = NULL,
  factorScaleFunc = NULL,
  cellIdx = NULL,
  legendColorTitle = "Mean Factor\nLoading",
  legendSizeTitle = "Percent\nLoaded",
  viridisOption = "viridis",
  verbose = FALSE,
  ...
)
```

Arguments

`object`	A liger object
`groupBy`	The names of the columns in `cellMeta` slot storing categorical variables. Loading data would be aggregated basing on these, together with `splitBy`. Default uses default clusters.
`useDims`	A Numeric vector to specify exact factors of interests. Default `NULL` uses all available factors.
`useRaw`	Whether to use un-aligned cell factor loadings (H matrices). Default `FALSE`.
`splitBy`	The names of the columns in `cellMeta` slot storing categorical variables. Dotplot panel splitting would be based on these. Default `NULL`.
`factorScaleFunc`	A function object applied to factor loading matrix for scaling the value for better visualization. Default `NULL`.
`cellIdx`	Valid cell subscription. See `subsetLiger`. Default `NULL` for using all cells.
`legendColorTitle`	Title for colorbar legend. Default `"Mean Factor\nLoading"`.

legendSizeTitle	Title for size legend. Default "Percent\nLoaded"
viridisOption	Name of available viridis palette. See viridis. Default "viridis".
verbose	Logical. Whether to show progress information. Mainly when subsetting data. Default FALSE.
...	Additional theme setting arguments passed to .complexHeatmapDotPlot and heatmap setting arguments passed to Heatmap. See Details.

Details

For ..., please notice that arguments colorMat, sizeMat, featureAnnDF, cellSplitVar, cellLabels and viridisOption from .complexHeatmapDotPlot are already occupied by this function internally. A lot of arguments from Heatmap have also been occupied: matrix,name, heatmap_legend_param, rect_gp, col, layer_fun, km, border, border_gp,column_gap, row_gap, cluster_row_slices, cluster_rows, row_title_gp,row_names_gp, row_split, row_labels, cluster_column_slices, cluster_columns,column_split, column_title_gp, column_title, column_labels, column_names_gp,top_annot

Value

HeatmapList object.

Examples

```
plotClusterFactorDot(pbmcPlot)
```

---

plotClusterGeneDot *Make dot plot of gene expression in cell groups*

Description

This function produces dot plots. Each column represent a group of cells specified by groupBy, each row is a gene specified by features. The color of dots reflects mean of normalized expression of specified genes in each cell group and sizes reflects the percentage of cells expressing each gene in a group. We utilize ComplexHeatmap for simplified management of adding annotation and slicing subplots. This was inspired by the implementation in scCustomize.

Usage

```
plotClusterGeneDot(
  object,
  features,
  groupBy = NULL,
  splitBy = NULL,
  featureScaleFunc = function(x) log2(10000 * x + 1),
  cellIdx = NULL,
  legendColorTitle = "Mean\nExpression",
  legendSizeTitle = "Percent\nExpressed",
```

```
  viridisOption = "magma",
  verbose = FALSE,
  ...
)
```

Arguments

object	A liger object
features	Use a character vector of gene names to make plain dot plot like a heatmap. Use a data.frame where the first column is gene names and second column is a grouping variable (e.g. subset runMarkerDEG output)
groupBy	The names of the columns in cellMeta slot storing categorical variables. Expression data would be aggregated basing on these, together with splitBy. Default uses default clusters.
splitBy	The names of the columns in cellMeta slot storing categorical variables. Dotplot panel splitting would be based on these. Default NULL.
featureScaleFunc	A function object applied to normalized data for scaling the value for better visualization. Default function(x) log2(10000*x + 1)
cellIdx	Valid cell subscription. See subsetLiger. Default NULL for using all cells.
legendColorTitle	Title for colorbar legend. Default "Mean\nExpression".
legendSizeTitle	Title for size legend. Default "Percent\nExpressed"
viridisOption	Name of available viridis palette. See viridis. Default "magma".
verbose	Logical. Whether to show progress information. Mainly when subsetting data. Default FALSE.
...	Additional theme setting arguments passed to .complexHeatmapDotPlot and heatmap setting arguments passed to Heatmap. See Details.

Details

For ..., please notice that arguments colorMat, sizeMat, featureAnnDF, cellSplitVar, cellLabels and viridisOption from .complexHeatmapDotPlot are already occupied by this function internally. A lot of arguments from Heatmap have also been occupied: matrix,name, heatmap_legend_param, rect_gp, col, layer_fun, km, border, border_gp,column_gap, row_gap, cluster_row_slices, cluster_rows, row_title_gp,row_names_gp, row_split, row_labels, cluster_column_slices, cluster_columns,column_split, column_title_gp, column_title, column_labels, column_names_gp,top_annot

Value

HeatmapList object.

Examples

```
# Use character vector of genes
features <- varFeatures(pbmcPlot)[1:10]
plotClusterGeneDot(pbmcPlot, features = features)

# Use data.frame with grouping information, with more tweak on plot
features <- data.frame(features, rep(letters[1:5], 2))
plotClusterGeneDot(pbmcPlot, features = features,
                   clusterFeature = TRUE, clusterCell = TRUE, maxDotSize = 6)
```

---

plotDensityDimRed *Create density plot basing on specified coordinates*

Description

This function shows the cell density presented in a 2D dimensionality reduction coordinates. Density is shown with coloring and contour lines. A scatter plot of the dimensionality reduction is added as well. The density plot can be splitted by categorical variables (e.g. `"dataset"`), while the scatter plot will always be shown for all cells in subplots as a reference of the global structure.

Usage

```
plotDensityDimRed(
  object,
  useDimRed = NULL,
  splitBy = NULL,
  combinePlot = TRUE,
  minDensity = 8,
  contour = TRUE,
  contourLineWidth = 0.3,
  contourBins = 5,
  dot = TRUE,
  dotColor = "grey",
  dotSize = 0.6,
  dotAlpha = 0.3,
  dotRaster = NULL,
  title = NULL,
  legendFillTitle = "Density",
  colorPalette = "magma",
  colorDirection = -1,
  ...
)
```

Arguments

`object`	A liger object

useDimRed — Name of the variable storing dimensionality reduction result in the cellMeta slot. Default uses default dimension reduction.

splitBy — Character vector of categorical variable names in cellMeta slot. Split all cells by groupings on this/these variable(s) to produce a density plot containing only the cells in each group. Default NULL.

combinePlot — Logical, whether to utilize plot_grid to combine multiple plots into one. Default TRUE returns combined ggplot. FALSE returns a list of ggplot or a single ggplot when only one plot is requested.

minDensity — A positive number to filter out low density region colored on plot. Default 8. Setting zero will show density on the whole panel.

contour — Logical, whether to draw the contour line. Default TRUE.

contourLineWidth — Numeric, the width of the contour line. Default 0.3.

contourBins — Number of contour bins. Higher value generates more contour lines. Default 5.

dot — Logical, whether to add scatter plot of all cells, even when density plot is splitted with splitBy. Default TRUE.

dotColor, dotSize, dotAlpha — Numeric, controls the appearance of all dots. Default "grey", 0.6 and 0.3, respectively.

dotRaster — Logical, whether to rasterize the scatter plot. Default NULL automatically rasterizes the dots when number of total cells to be plotted exceeds 100,000.

title — Text of main title of the plots. Default NULL. Length of character vector input should match with number of plots generated.

legendFillTitle — Text of legend title. Default "Density".

colorPalette — Name of the option for scale_fill_viridis_c. Default "magma".

colorDirection — Color gradient direction for scale_fill_viridis_c. Default -1.

... — More theme setting arguments passed to .ggplotLigerTheme.

Value

A ggplot object when only one plot is generated, A ggplot object combined with plot_grid when multiple plots and combinePlot = TRUE. A list of ggplot when multiple plots and combinePlot = FALSE.

Examples

```
# Example dataset has small number of cells, thus cutoff adjusted.
plotDensityDimRed(pbmcPlot, minDensity = 1)
```

plotDimRed *Generate scatter plot(s) using liger object*

Description

This function allows for using available cell metadata to build the x-/y-axis. Available per-cell data can be used to form the color/shape annotation, including cell metadata, raw or processed gene expression, and unnormalized or aligned factor loading. Multiple coloring variable is allowed from the same specification of `slot`, and this returns a list of plots with different coloring values. Users can further split the plot(s) by grouping on cells (e.g. datasets).

some text

Usage

```
plotDimRed(
  object,
  colorBy = NULL,
  useDimRed = NULL,
 slot = c("cellMeta", "rawData", "normData", "scaleData", "H.norm", "H", "normPeak",
    "rawPeak"),
  colorByFunc = NULL,
  cellIdx = NULL,
  splitBy = NULL,
  shapeBy = NULL,
  titles = NULL,
  ...
)

plotClusterDimRed(object, useCluster = NULL, useDimRed = NULL, ...)

plotDatasetDimRed(object, useDimRed = NULL, ...)

plotByDatasetAndCluster(
  object,
  useDimRed = NULL,
  useCluster = NULL,
  combinePlots = TRUE,
  ...
)

plotGeneDimRed(
  object,
  features,
  useDimRed = NULL,
  log = TRUE,
  scaleFactor = 10000,
```

```
  zeroAsNA = TRUE,
  colorPalette = "C",
  ...
)

plotPeakDimRed(
  object,
  features,
  useDimRed = NULL,
  log = TRUE,
  scaleFactor = 10000,
  zeroAsNA = TRUE,
  colorPalette = "C",
  ...
)

plotFactorDimRed(
  object,
  factors,
  useDimRed = NULL,
  trimHigh = 0.03,
  zeroAsNA = TRUE,
  colorPalette = "D",
  ...
)
```

Arguments

object	A liger object.
colorBy	Available variable name in specified slot to look for color annotation information. See details. Default NULL generates all-black dots.
useDimRed	Name of the variable storing dimensionality reduction result in the cellMeta(object). Default NULL use default dimRed.
slot	Choose the slot to find the colorBy variable. See details. Default "cellMeta".
colorByFunc	Default NULL. A function object that expects a vector/factor/data.frame retrieved by colorBy as the only input, and returns an object of the same size, so that the all color "aes" are replaced by this output. Useful when, for example, users need to scale the gene expression shown on plot.
cellIdx	Character, logical or numeric index that can subscribe cells. Missing or NULL for all cells.
splitBy	Character vector of categorical variable names in cellMeta slot. Split all cells by groupings on this/these variable(s) to produce a scatter plot containing only the cells in each group. Default NULL.
shapeBy	Available variable name in cellMeta slot to look for categorical annotation to be reflected by dot shapes. Default NULL.
titles	Title text. A character scalar or a character vector with as many elements as multiple plots are supposed to be generated. Default NULL.

...	More plot setting arguments. See .ggScatter and .ggplotLigerTheme.
useCluster	Name of variable in cellMeta(object). Default NULL uses default cluster.
combinePlots	Logical, whether to utilize plot_grid to combine multiple plots into one. Default TRUE returns combined ggplot. FALSE returns a list of ggplot.
features, factors	Name of genes or index of factors that need to be visualized.
log	Logical. Whether to log transform the normalized expression of genes. Default TRUE.
scaleFactor	Number to be multiplied with the normalized expression of genes before log transformation. Default 1e4. NULL for not scaling.
zeroAsNA	Logical, whether to swap all zero values to NA so naColor will be used to represent non-expressing features. Default TRUE.
colorPalette	Name of viridis palette. See viridis for options. Default "C" ("plasma") for gene expression and "D" ("viridis") for factor loading.
trimHigh	Number for highest cut-off to limit the outliers. Factor loading above this value will all be trimmed to this value. Default 0.03.

Details

Available option for slot include: "cellMeta", "rawData", "normData", "scaleData", "H.norm" and "H". When "rawData", "normData" or "scaleData", colorBy has to be a character vector of feature names. When "H.norm" or "H", colorBy can be any valid index to select one factor of interests. Note that character index follows "Factor_[k]" format, with replacing [k] with an integer.

When "cellMeta", colorBy has to be an available column name in the table. Note that, for colorBy as well as x, y, shapeBy and splitBy, since a matrix object is feasible in cellMeta table, using a column (e.g. named as "column1" in a certain matrix (e.g. named as "matrixVar") should follow the syntax of "matrixVar.column1". When the matrix does not have a "colname" attribute, the subscription goes with "matrixVar.V1", "matrixVar.V2" and etc. Use "UMAP.1", "UMAP.2", "TSNE.1" or "TSNE.2" for the 2D embeddings generated with rliger package. These are based on the nature of as.data.frame method on a DataFrame object.

Value

A ggplot object when a single plot is intended. A list of ggplot objects, when multiple colorBy variables and/or splitBy are set. When plotly = TRUE, all ggplot objects become plotly (htmlwidget) objects.

ggplot object when only one feature (e.g. cluster variable, gene, factor) is set. List object when multiple of those are specified.

See Also

Please refer to plotDimRed, .ggScatter, .ggplotLigerTheme for additional graphic setting

Examples

```
plotDimRed(pbmcPlot, colorBy = "dataset", slot = "cellMeta",
           labelText = FALSE)
plotDimRed(pbmcPlot, colorBy = "S100A8", slot = "normData",
           dotOrder = "ascending", dotSize = 2)
plotDimRed(pbmcPlot, colorBy = 2, slot = "H.norm",
           dotOrder = "ascending", dotSize = 2, colorPalette = "viridis")
plotClusterDimRed(pbmcPlot)
plotDatasetDimRed(pbmcPlot)
plotByDatasetAndCluster(pbmcPlot)
plotGeneDimRed(pbmcPlot, varFeatures(pbmcPlot)[1])
plotFactorDimRed(pbmcPlot, 2)
```

---

plotGeneHeatmap *Plot Heatmap of Gene Expression or Factor Loading*

Description

Plot Heatmap of Gene Expression or Factor Loading

Usage

```
plotGeneHeatmap(
  object,
  features,
  cellIdx = NULL,
  slot = c("normData", "rawData", "scaleData", "scaleUnsharedData"),
  useCellMeta = NULL,
  cellAnnotation = NULL,
  featureAnnotation = NULL,
  cellSplitBy = NULL,
  featureSplitBy = NULL,
  viridisOption = "C",
  ...
)

plotFactorHeatmap(
  object,
  factors = NULL,
  cellIdx = NULL,
  slot = c("H.norm", "H"),
  useCellMeta = NULL,
  cellAnnotation = NULL,
  factorAnnotation = NULL,
  cellSplitBy = NULL,
  factorSplitBy = NULL,
  trim = c(0, 0.03),
```

```
  viridisOption = "D",
  ...
)
```

Arguments

object	A liger object, with data to be plot available.
features, factors	Character vector of genes of interests or numeric index of factor to be involved. features is required, while factors is by default all the factors (reads object recorded k value in uns slot).
cellIdx	Valid index to subscribe cells to be included. See subsetLiger. Default NULL use all cells.
slot	Use the chosen matrix for heatmap. For plotGeneHeatmap, default "normData", alternatively "rawData", "scaleData" or "scaleUnsharedData". For plotFactorHeatmap, default "H.norm", alternatively "H".
useCellMeta	Character vector of available variable names in cellMeta, variables will be added as annotation to the heatmap. Default NULL.
cellAnnotation	data.frame object for using external annotation, with each column a variable and each row is a cell. Row names of this data.frame will be used for matching cells involved in heatmap. For cells not found in this data.frame, NAs will be added with warning. Default NULL.
featureAnnotation, factorAnnotation	Similar as cellAnnotation, while each row would be a gene or factor, respectively. Default NULL.
cellSplitBy	Character vector of variable names available in annotation given by useCellMeta and cellAnnotation. This slices the heatmap by specified variables. Default NULL.
featureSplitBy, factorSplitBy	Similar as cellSplitBy. Default NULL
viridisOption	See option argument of viridis. Default "C" (plasma) for plotGeneHeatmap and "D" (viridis) for plotFactorHeatmap.
...	Additional arguments passed to general function .plotHeatmap and Heatmap.
trim	Numeric vector of two numbers. Higher value limits the maximum value and lower value limits the minimum value. Default c(0, 0.03).

Value

HeatmapList-class object

Examples

```
plotGeneHeatmap(pbmcPlot, varFeatures(pbmcPlot))
plotGeneHeatmap(pbmcPlot, varFeatures(pbmcPlot),
                useCellMeta = c("leiden_cluster", "dataset"),
                cellSplitBy = "leiden_cluster")
```

```
plotFactorHeatmap(pbmcPlot)
plotFactorHeatmap(pbmcPlot, cellIdx = pbmcPlot$leiden_cluster %in% 1:3,
                  useCellMeta = c("leiden_cluster", "dataset"),
                  cellSplitBy = "leiden_cluster")
```

`plotGeneLoadings` *Visualize factor expression and gene loading*

Description

Visualize factor expression and gene loading

Usage

```
plotGeneLoadings(
  object,
  markerTable,
  useFactor,
  useDimRed = NULL,
  nLabel = 15,
  nPlot = 30,
  ...
)

plotGeneLoadingRank(
  object,
  markerTable,
  useFactor,
  nLabel = 15,
  nPlot = 30,
  ...
)
```

Arguments

`object`	A liger object with valid factorization result.
`markerTable`	Returned result of `getFactorMarkers`.
`useFactor`	Integer index for which factor to visualize.
`useDimRed`	Name of the variable storing dimensionality reduction result in the `cellMeta` slot. Default `"UMAP"`.
`nLabel`	Integer, number of top genes to be shown with text labels. Default `15`.
`nPlot`	Integer, number of top genes to be shown in the loading rank plot. Default `30`.
`...`	Additional plot theme setting arguments passed to `.ggScatter` and `.ggplotLigerTheme`.

Examples

```
result <- getFactorMarkers(pbmcPlot, "ctrl", "stim")
plotGeneLoadings(pbmcPlot, result, useFactor = 2)
```

`plotGeneViolin` *Visualize gene expression or cell metadata with violin plot*

Description

Visualize gene expression or cell metadata with violin plot

Usage

```
plotGeneViolin(object, gene, byDataset = TRUE, groupBy = NULL, ...)

plotTotalCountViolin(object, groupBy = "dataset", ...)

plotGeneDetectedViolin(object, groupBy = "dataset", ...)
```

Arguments

`object`	A liger object.
`gene`	Character gene names.
`byDataset`	Logical, whether the violin plot should be splitted by dataset. Default `TRUE`.
`groupBy`	Names of available categorical variable in `cellMeta` slot. Use `FALSE` for no grouping. Default `NULL` looks clustering result but will not group if no clustering found.
`...`	Additional arguments passed to `plotCellViolin`.

Value

ggplot if using a single gene and not splitting by dataset. Otherwise, list of ggplot.

Examples

```
plotGeneViolin(pbmcPlot, varFeatures(pbmcPlot)[1],
               groupBy = "leiden_cluster")
plotTotalCountViolin(pbmc)
plotGeneDetectedViolin(pbmc, dot = TRUE, box = TRUE, colorBy = "dataset")
```

plotGroupClusterDimRed *Comprehensive group splited cluster plot on dimension reduction with proportion*

Description

This function produces combined plot on group level (e.g. dataset, other metadata variable like biological conditions). Scatter plot of dimension reduction with cluster labeled is generated per group. Furthermore, a stacked barplot of cluster proportion within each group is also combined with the subplot of each group.

Usage

```
plotGroupClusterDimRed(
  object,
  useGroup = "dataset",
  useCluster = NULL,
  useDimRed = NULL,
  combinePlot = TRUE,
  droplevels = TRUE,
  relHeightMainLegend = c(5, 1),
  relHeightDRBar = c(10, 1),
  mainNRow = NULL,
  mainNCol = NULL,
  legendNRow = 1,
  ...
)
```

Arguments

object	A liger object with dimension reduction, grouping variable and cluster assignment in `cellMeta(object)`.
useGroup	Variable name of the group division in metadata. Default `"dataset"`.
useCluster	Name of variable in `cellMeta(object)`. Default `NULL` uses default cluster.
useDimRed	Name of the variable storing dimensionality reduction result in `cellMeta(object)`. Default `NULL` use default dimRed.
combinePlot	Whether to return combined plot. Default `TRUE`. If `FALSE`, will return a list containing only the scatter plots.
droplevels	Logical, whether to perform `droplevels()` on the selected grouping variable. Default `TRUE` will not show groups that are listed as categories but do not indeed have any cells.
relHeightMainLegend	Relative heights of the main combination panel and the legend at the bottom. Must be a numeric vector of 2 numbers. Default `c(5, 1)`.

relHeightDRBar Relative heights of the scatter plot and the barplot within each subpanel. Must be a numeric vector of 2 numbers. Default c(10, 1).

mainNRow, mainNCol
Arrangement of the main plotting region, for number of rows and columns. Default NULL will be automatically handled by plot_grid.

legendNRow Arrangement of the legend, number of rows. Default 1.

... Additional graphic setting arguments passed to plotDimRed.

Value

ggplot object when only one feature (e.g. cluster variable, gene, factor) is set. List object when multiple of those are specified.

See Also

Please refer to plotDimRed, .ggScatter, .ggplotLigerTheme for additional graphic setting

Examples

```
plotGroupClusterDimRed(pbmcPlot)
```

---

plotMarkerHeatmap *Create heatmap for showing top marker expression in conditions*

Description

Create heatmap for showing top marker expression in conditions

Usage

```
plotMarkerHeatmap(
  object,
  result,
  topN = 5,
  lfcThresh = 1,
  padjThresh = 0.05,
  pctInThresh = 50,
  pctOutThresh = 50,
  dedupBy = c("logFC", "padj"),
  groupBy = NULL,
  groupSize = 50,
  column_title = NULL,
  ...
)
```

Arguments

object	A liger object, with normalized data and metadata to annotate available.
result	The data.frame returned by runMarkerDEG.
topN	Number of top features to be plot for each group. Default 5.
lfcThresh	Hard threshold on logFC value. Default 1.
padjThresh	Hard threshold on adjusted P-value. Default 0.05.
pctInThresh, pctOutThresh	Threshold on expression percentage. These mean that a feature will only pass the filter if it is expressed in more than pctInThresh percent of cells in the corresponding cluster. Similarly for pctOutThresh. Default 50 percent for both.
dedupBy	When ranking by padj and logFC and a feature is ranked as top for multiple clusters, assign this feature as the marker of a cluster when it has the largest "logFC" in the cluster or has the lowest "padj". Default "logFC".
groupBy	Cell metadata variable names for cell grouping. Downsample balancing will also be aware of this. Default c("dataset","leiden_cluster").
groupSize	Maximum number of cells in each group to be downsampled for plotting. Default 50.
column_title	Title on the column. Default NULL.
...	Parameter passed to wrapped functions in the inheritance order: plotGeneHeatmap, .plotHeatmap, ComplexHeatmap::Heatmap

Examples

```
markerTable <- runMarkerDEG(pbmcPlot)
plotMarkerHeatmap(pbmcPlot, markerTable)
```

---

plotProportion *Visualize proportion across two categorical variables*

Description

plotProportionBar creates bar plots comparing the cross-category proportion. plotProportionDot creates dot plots. plotClusterProportions has variable pre-specified and calls the dot plot. plotProportion produces a combination of both bar plots and dot plot.

Having package "ggrepel" installed can help adding tidier percentage annotation on the pie chart.

Usage

```
plotProportion(
  object,
  class1 = NULL,
  class2 = "dataset",
  method = c("stack", "group", "pie"),
```

```
  ...
)

plotProportionDot(
  object,
  class1 = NULL,
  class2 = "dataset",
  showLegend = FALSE,
  panelBorder = TRUE,
  ...
)

plotProportionBar(
  object,
  class1 = NULL,
  class2 = "dataset",
  method = c("stack", "group"),
  inclRev = FALSE,
  panelBorder = TRUE,
  combinePlot = TRUE,
  ...
)

plotClusterProportions(object, useCluster = NULL, return.plot = FALSE, ...)

plotProportionPie(
  object,
  class1 = NULL,
  class2 = "dataset",
  labelSize = 4,
  labelColor = "white",
  ...
)
```

Arguments

`object`	A liger object.
`class1`, `class2`	Each should be a single name of a categorical variable available in `cellMeta` slot. Number of cells in each categories in `class2` will be served as the denominator when calculating proportions. By default `class1 = NULL` and uses default clusters and `class2 = "dataset"`.
`method`	For bar plot, choose whether to draw `"stack"` or `"group"` bar plot. Default `"stack"`.
`showLegend`, `panelBorder`, `...`	ggplot theme setting arguments passed to `.ggplotLigerTheme`.
`inclRev`	Logical, for barplot, whether to reverse the specification for `class1` and `class2` and produce two plots. Default `FALSE`.

combinePlot — Logical, whether to combine the two plots with plot_grid when two plots are created. Default TRUE.

useCluster — For plotClusterProportions. Same as class1 while class2 is hardcoded with "dataset".

return.plot — **defuncted**.

labelSize, labelColor — Settings on pie chart percentage label. Default 4 and "white".

Value

ggplot or list of ggplot

Examples

```
plotProportion(pbmcPlot)
plotProportionBar(pbmcPlot, method = "group")
plotProportionPie(pbmcPlot)
```

---

plotSankey — *Make Riverplot/Sankey diagram that shows label mapping across datasets*

Description

Creates a riverplot/Sankey diagram to show how independent cluster assignments from two datasets map onto a joint clustering. Prior knowledge of cell annotation for the given datasets is required to make sense from the visualization. Dataset original annotation can be added with the syntax shown in example code in this manual. The joint clustering could be generated with runCluster or set by any other metadata annotation.

Dataset original annotation can be inserted before running this function using cellMeta<- method. Please see example below.

This function depends on CRAN available package "sankey" and it has to be installed in order to make this function work.

Usage

```
plotSankey(
  object,
  cluster1,
  cluster2,
  clusterConsensus = NULL,
  minFrac = 0.01,
  minCell = 10,
  titles = NULL,
  prefixes = NULL,
  labelCex = 1,
```

```
  titleCex = 1.1,
  colorValues = scPalette,
  mar = c(2, 2, 4, 2)
)
```

Arguments

`object`	A liger object with all three clustering variables available.
`cluster1, cluster2`	Name of the variables in `cellMeta(object)` for the cluster assignments of dataset 1 and 2, respectively.
`clusterConsensus`	Name of the joint cluster variable to use. Default uses the default clustering of the object. Can select a variable name in `cellMeta(object)`.
`minFrac`	Numeric. Minimum fraction of cluster for an edge to be shown. Default `0.05`.
`minCell`	Numeric. Minimum number of cells for an edge to be shown. Default `10`.
`titles`	Character vector of three. Customizes the column title text shown. Default uses the variable names `cluster1`, `clusterConsensus` and `cluster2`.
`prefixes`	Character vector of three. Cluster names have to be unique across all three variables, so this is provided to deduplicate the clusters by adding `"prefixes[i]-"` before the actual label. This will not be applied when no duplicate is found. Default `NULL` uses variable names. An NA value or a string with no character (i.e. `""`) does not add the prefix to the corresponding variable.
`labelCex`	Numeric. Amount by which node label text should be magnified relative to the default. Default `1`.
`titleCex`	Numeric. Amount by which node label text should be magnified relative to the default. Default `1.1`.
`colorValues`	Character vector of color codes to set color for each level in the consensus clustering. Default `scPalette`.
`mar`	Numeric vector of the form `c(bottom, left, top, right)` which gives the number of lines of margin to be specified on the four sides of the plot. Increasing the 2nd and 4th values can be helpful when cluster labels are long and extend out side of the plotting region. Default `c(2, 2, 4, 2)`.

Value

No returned value. The sankey diagram will be displayed instead.

Note

This function works as a replacement of the function `makeRiverplot` in rliger <1.99. We decide to make a new function because the dependency adopted by the older version is archived on CRAN and will be no longer available.

Examples

```
# Make fake dataset specific labels from joint clustering result
cellMeta(pbmcPlot, "ctrl_cluster", "ctrl") <-
    cellMeta(pbmcPlot, "leiden_cluster", "ctrl")
cellMeta(pbmcPlot, "stim_cluster", "stim") <-
    cellMeta(pbmcPlot, "leiden_cluster", "stim")
if (requireNamespace("sankey", quietly = TRUE)) {
    plotSankey(pbmcPlot, "ctrl_cluster", "stim_cluster",
               titles = c("control", "LIGER", "stim"),
               prefixes = c("c", NA, "s"))
}
```

`plotSpatial2D` *Visualize a spatial dataset*

Description

Visualize a spatial dataset

Usage

```
plotSpatial2D(object, ...)

## S3 method for class 'liger'
plotSpatial2D(object, dataset, useCluster = NULL, legendColorTitle = NULL, ...)

## S3 method for class 'ligerSpatialDataset'
plotSpatial2D(
  object,
  useCluster = NULL,
  legendColorTitle = NULL,
  useDims = c(1, 2),
  xlab = NULL,
  ylab = NULL,
  labelText = FALSE,
  ...
)
```

Arguments

`object`	Either a liger object containing a spatial dataset or a ligerSpatialDataset object.
`...`	Arguments passed to other methods. `.liger` method passes everything to `.ligerSpatialDataset` method, and the latter passes everything to `.ggScatter` and then `.ggplotLigerTheme`.
`dataset`	Name of one spatial dataset.
`useCluster`	Either the name of one variable in `cellMeta(object)` or a factor object with annotation that matches with all cells in the specified dataset. Default `NULL` uses default clusters.

legendColorTitle
: Alternative title text in the legend. Default NULL uses the variable name set by useCluster, or "Annotation" is useCluster is a customized factor object.

useDims
: Numeric vector of two, choosing the coordinates to be drawn on 2D space. (STARmap data could have 3 dimensions.) Default c(1, 2).

xlab, ylab
: Text label on x-/y-axis. Default NULL does not show it.

labelText
: Logical, whether to label annotation onto the scatter plot. Default FALSE.

Value

A ggplot object

Examples

```
ctrl.fake.spatial <- as.ligerDataset(dataset(pbmc, "ctrl"), modal = "spatial")
fake.coords <- matrix(rnorm(2 * ncol(ctrl.fake.spatial)), ncol = 2)
dimnames(fake.coords) <- list(colnames(ctrl.fake.spatial), c("x", "y"))
coordinate(ctrl.fake.spatial) <- fake.coords
dataset(pbmc, "ctrl") <- ctrl.fake.spatial
plotSpatial2D(pbmc, dataset = "ctrl")
```

---

plotVarFeatures *Plot the variance vs mean of feature expression*

Description

For each dataset where the feature variablitity is calculated, a plot of log10 feature expression variance and log10 mean will be produced. Features that are considered as variable would be highlighted in red.

Usage

```
plotVarFeatures(object, combinePlot = TRUE, dotSize = 1, ...)
```

Arguments

object
: liger object. selectGenes needs to be run in advance.

combinePlot
: Logical. If TRUE, sub-figures for all datasets will be combined into one plot. if FALSE, a list of plots will be returned. Default TRUE.

dotSize
: Controls the size of dots in the main plot. Default 0.8.

...
: More theme setting parameters passed to .ggplotLigerTheme.

Value

ggplot object when combinePlot = TRUE, a list of ggplot objects when combinePlot = FALSE

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
plotVarFeatures(pbmc)
```

`plotVolcano` *Create volcano plot for Wilcoxon test result*

Description

`plotVolcano` is a simple implementation and shares most of arguments with other rliger plotting functions. `plotEnhancedVolcano` is a wrapper function of `EnhancedVolcano::EnhancedVolcano()`, which has provides substantial amount of arguments for graphical control. However, that requires the installation of package "EnhancedVolcano".

Usage

```
plotVolcano(
  result,
  group,
  logFCThresh = 1,
  padjThresh = 0.01,
  labelTopN = 20,
  dotSize = 2,
  dotAlpha = 0.8,
  legendPosition = "top",
  labelSize = 4,
  ...
)

plotEnhancedVolcano(result, group, ...)
```

Arguments

`result`	Data frame table returned by `runWilcoxon`
`group`	Selection of one group available from `result$group`
`logFCThresh`	Number for the threshold on the absolute value of the log2 fold change statistics. Default `1`.
`padjThresh`	Number for the threshold on the adjusted p-value statistics. Default `0.01`.
`labelTopN`	Number of top differential expressed features to be labeled on the top of the dots. Default `20`.
`dotSize`, `dotAlpha`	Numbers for universal aesthetics control of dots. Default `2` and `0.8`.
`legendPosition`	Text indicating where to place the legend. Choose from `"top"`, `"bottom"`, `"left"` or `"right"`. Default `"top"`.

labelSize	Size of labeled top features and line annotations. Default 4.
...	For plotVolcano, more theme setting arguments passed to .ggplotLigerTheme. For plotEnhancedVolcano, arguments passed to EnhancedVolcano::EnhancedVolcano().

Value

ggplot

Examples

```
result <- runMarkerDEG(pbmcPlot)
plotVolcano(result, 1)
```

---

quantileAlignSNF *Quantile align (normalize) factor loadings*

Description

This is a deprecated function. Calling 'quantileNorm' instead.

Usage

```
quantileAlignSNF(
  object,
  knn_k = 20,
  k2 = 500,
  prune.thresh = 0.2,
  ref_dataset = NULL,
  min_cells = 20,
  quantiles = 50,
  nstart = 10,
  resolution = 1,
  dims.use = 1:ncol(x = object@H[[1]]),
  dist.use = "CR",
  center = FALSE,
  small.clust.thresh = 0,
  id.number = NULL,
  print.mod = FALSE,
  print.align.summary = FALSE
)
```

Arguments

object	liger object. Should run optimizeALS before calling.
knn_k	Number of nearest neighbors for within-dataset knn graph (default 20).

k2 Horizon parameter for shared nearest factor graph. Distances to all but the k2 nearest neighbors are set to 0 (cuts down on memory usage for very large graphs). (default 500)

prune.thresh Minimum allowed edge weight. Any edges below this are removed (given weight 0) (default 0.2)

ref_dataset Name of dataset to use as a "reference" for normalization. By default, the dataset with the largest number of cells is used.

min_cells Minimum number of cells to consider a cluster shared across datasets (default 2)

quantiles Number of quantiles to use for quantile normalization (default 50).

nstart Number of times to perform Louvain community detection with different random starts (default 10).

resolution Controls the number of communities detected. Higher resolution -> more communities. (default 1)

dims.use Indices of factors to use for shared nearest factor determination (default 1:ncol(H[[1]])).

dist.use Distance metric to use in calculating nearest neighbors (default "CR").

center Centers the data when scaling factors (useful for less sparse modalities like methylation data). (default FALSE)

small.clust.thresh
Extracts small clusters loading highly on single factor with fewer cells than this before regular alignment (default 0 – no small cluster extraction).

id.number Number to use for identifying edge file (when running in parallel) (generates random value by default).

print.mod Print modularity output from clustering algorithm (default FALSE).

print.align.summary
Print summary of clusters which did not align normally (default FALSE).

Details

This process builds a shared factor neighborhood graph to jointly cluster cells, then quantile normalizes corresponding clusters.

The first step, building the shared factor neighborhood graph, is performed in SNF(), and produces a graph representation where edge weights between cells (across all datasets) correspond to their similarity in the shared factor neighborhood space. An important parameter here is knn_k, the number of neighbors used to build the shared factor space (see SNF()). Afterwards, modularity-based community detection is performed on this graph (Louvain clustering) in order to identify shared clusters across datasets. The method was first developed by Waltman and van Eck (2013) and source code is available at http://www.ludowaltman.nl/slm/. The most important parameter here is resolution, which corresponds to the number of communities detected.

Next we perform quantile alignment for each dataset, factor, and cluster (by stretching/compressing datasets' quantiles to better match those of the reference dataset). These aligned factor loadings are combined into a single matrix and returned as H.norm.

Value

liger object with H.norm and cluster slots set.

Examples

```
## Not run:
# liger object, factorization complete
ligerex
# do basic quantile alignment
ligerex <- quantileAlignSNF(ligerex)
# higher resolution for more clusters (note that SNF is conserved)
ligerex <- quantileAlignSNF(ligerex, resolution = 1.2)
# change knn_k for more fine-grained local clustering
ligerex <- quantileAlignSNF(ligerex, knn_k = 15, resolution = 1.2)

## End(Not run)
```

---

quantileNorm *Quantile Align (Normalize) Factor Loadings*

Description

This process builds a shared factor neighborhood graph to jointly cluster cells, then quantile normalizes corresponding clusters.

The first step, building the shared factor neighborhood graph, is performed in SNF(), and produces a graph representation where edge weights between cells (across all datasets) correspond to their similarity in the shared factor neighborhood space. An important parameter here is nNeighbors, the number of neighbors used to build the shared factor space.

Next we perform quantile alignment for each dataset, factor, and cluster (by stretching/compressing datasets' quantiles to better match those of the reference dataset).

Usage

```
quantileNorm(object, ...)

## S3 method for class 'liger'
quantileNorm(
  object,
  quantiles = 50,
  reference = NULL,
  minCells = 20,
  nNeighbors = 20,
  useDims = NULL,
  center = FALSE,
  maxSample = 1000,
  eps = 0.9,
  refineKNN = TRUE,
  clusterName = "quantileNorm_cluster",
  seed = 1,
  verbose = getOption("ligerVerbose", TRUE),
```

```
  ...
)

## S3 method for class 'Seurat'
quantileNorm(
  object,
  reduction = "inmf",
  quantiles = 50,
  reference = NULL,
  minCells = 20,
  nNeighbors = 20,
  useDims = NULL,
  center = FALSE,
  maxSample = 1000,
  eps = 0.9,
  refineKNN = TRUE,
  clusterName = "quantileNorm_cluster",
  seed = 1,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

object	A liger or Seurat object with valid factorization result available (i.e. runIntegration performed in advance).
...	Arguments passed to other S3 methods of this function.
quantiles	Number of quantiles to use for quantile normalization. Default 50.
reference	Character, numeric or logical selection of one dataset, out of all available datasets in object, to use as a "reference" for quantile normalization. Default NULL tries to find an RNA dataset with the largest number of cells; if no RNA dataset available, use the globally largest dataset.
minCells	Minimum number of cells to consider a cluster shared across datasets. Default 20.
nNeighbors	Number of nearest neighbors for within-dataset knn graph. Default 20.
useDims	Indices of factors to use for shared nearest factor determination. Default NULL uses all factors.
center	Whether to center the data when scaling factors. Could be useful for less sparse modalities like methylation data. Default FALSE.
maxSample	Maximum number of cells used for quantile normalization of each cluster and factor. Default 1000.
eps	The error bound of the nearest neighbor search. Lower values give more accurate nearest neighbor graphs but take much longer to compute. Default 0.9.
refineKNN	whether to increase robustness of cluster assignments using KNN graph. Default TRUE.

clusterName	Variable name that will store the clustering result in metadata of a liger object or a Seurat object. Default "quantileNorm_cluster"
seed	Random seed to allow reproducible results. Default 1.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
reduction	Name of the reduction where LIGER integration result is stored. Default "inmf".

Value

Updated input object

- liger method
  - Update the H.norm slot for the alignment cell factor loading, ready for running graph based community detection clustering or dimensionality reduction for visualization.
  - Update the cellMata slot with a cluster assignment basing on cell factor loading
- Seurat method
  - Update the reductions slot with a new DimReduc object containing the aligned cell factor loading.
  - Update the metadata with a cluster assignment basing on cell factor loading

Examples

```
pbmc <- quantileNorm(pbmcPlot)
```

---

quantile_norm-deprecated

[Deprecated] Quantile align (normalize) factor loading

Description

Please turn to quantileNorm.

This process builds a shared factor neighborhood graph to jointly cluster cells, then quantile normalizes corresponding clusters.

The first step, building the shared factor neighborhood graph, is performed in SNF(), and produces a graph representation where edge weights between cells (across all datasets) correspond to their similarity in the shared factor neighborhood space. An important parameter here is knn_k, the number of neighbors used to build the shared factor space.

Next we perform quantile alignment for each dataset, factor, and cluster (by stretching/compressing datasets' quantiles to better match those of the reference dataset). These aligned factor loadings are combined into a single matrix and returned as H.norm.

Arguments

object	liger object. Should run optimizeALS before calling.
knn_k	Number of nearest neighbors for within-dataset knn graph (default 20).
ref_dataset	Name of dataset to use as a "reference" for normalization. By default, the dataset with the largest number of cells is used.
min_cells	Minimum number of cells to consider a cluster shared across datasets (default 20)
quantiles	Number of quantiles to use for quantile normalization (default 50).
eps	The error bound of the nearest neighbor search. (default 0.9) Lower values give more accurate nearest neighbor graphs but take much longer to computer.
dims.use	Indices of factors to use for shared nearest factor determination (default 1:ncol(H[[1]])).
do.center	Centers the data when scaling factors (useful for less sparse modalities like methylation data). (default FALSE)
max_sample	Maximum number of cells used for quantile normalization of each cluster and factor. (default 1000)
refine.knn	whether to increase robustness of cluster assignments using KNN graph.(default TRUE)
rand.seed	Random seed to allow reproducible results (default 1)

Value

liger object with 'H.norm' and 'clusters' slot set.

See Also

rliger-deprecated

---

rawPeak *Access ligerATACDataset peak data*

Description

Similar as how default ligerDataset data is accessed.

Usage

```
rawPeak(x, dataset)

rawPeak(x, dataset, check = TRUE) <- value

normPeak(x, dataset)

normPeak(x, dataset, check = TRUE) <- value
```

```
## S4 method for signature 'liger,character'
rawPeak(x, dataset)

## S4 replacement method for signature 'liger,character'
rawPeak(x, dataset, check = TRUE) <- value

## S4 method for signature 'liger,character'
normPeak(x, dataset)

## S4 replacement method for signature 'liger,character'
normPeak(x, dataset, check = TRUE) <- value

## S4 method for signature 'ligerATACDataset,missing'
rawPeak(x, dataset = NULL)

## S4 replacement method for signature 'ligerATACDataset,missing'
rawPeak(x, dataset = NULL, check = TRUE) <- value

## S4 method for signature 'ligerATACDataset,missing'
normPeak(x, dataset = NULL)

## S4 replacement method for signature 'ligerATACDataset,missing'
normPeak(x, dataset = NULL, check = TRUE) <- value
```

Arguments

x	ligerATACDataset object or a liger object.
dataset	Name or numeric index of an ATAC dataset.
check	Logical, whether to perform object validity check on setting new value.
value	dgCMatrix-class matrix.

Value

The retrieved peak count matrix or the updated x object.

---

read10X *Load in data from 10X*

Description

Enables easy loading of sparse data matrices provided by 10X genomics.

read10X works generally for 10X cellranger pipelines including: CellRanger < 3.0 & >= 3.0 and CellRanger-ARC.

read10XRNA invokes read10X and takes the "Gene Expression" out, so that the result can directly be used to construct a liger object. See Examples for demonstration.

read10XATAC works for both cellRanger-ARC and cellRanger-ATAC pipelines but needs user arguments for correct recognition. Similarly, the returned value can directly be used for constructing a liger object.

Usage

```
read10X(
  path,
  sampleNames = NULL,
  useFiltered = NULL,
  reference = NULL,
  geneCol = 2,
  cellCol = 1,
  returnList = FALSE,
  verbose = getOption("ligerVerbose", TRUE),
  sample.dirs = path,
  sample.names = sampleNames,
  use.filtered = useFiltered,
  data.type = NULL,
  merge = NULL,
  num.cells = NULL,
  min.umis = NULL
)

read10XRNA(
  path,
  sampleNames = NULL,
  useFiltered = NULL,
  reference = NULL,
  returnList = FALSE,
  ...
)

read10XATAC(
  path,
  sampleNames = NULL,
  useFiltered = NULL,
  pipeline = c("atac", "arc"),
  arcFeatureType = "Peaks",
  returnList = FALSE,
  geneCol = 2,
  cellCol = 1,
  verbose = getOption("ligerVerbose", TRUE)
)
```

Arguments

path	[A.] A Directory containing the matrix.mtx, genes.tsv (or features.tsv), and barcodes.tsv files provided by 10X. A vector, a named vector, a list or a named list

can be given in order to load several data directories. [B.] The 10X root directory where subdirectories of per-sample output folders can be found. Sample names will by default take the name of the vector, list or subfolders.

`sampleNames` A vector of names to override the detected or set sample names for what is given to `path`. Default `NULL`. If no name detected at all and multiple samples are given, will name them by numbers.

`useFiltered` Logical, if `path` is given as case B, whether to use the filtered feature barcode matrix instead of raw (unfiltered). Default `TRUE`.

`reference` In case of specifying a CellRanger<3 root folder to `path`, import the matrix from the output using which reference. Only needed when multiple references present. Default `NULL`.

`geneCol` Specify which column of genes.tsv or features.tsv to use for gene names. Default `2`.

`cellCol` Specify which column of barcodes.tsv to use for cell names. Default `1`.

`returnList` Logical, whether to still return a structured list instead of a single matrix object, in the case where only one sample and only one feature type can be found. Otherwise will always return a list. Default `FALSE`.

`verbose` Logical. Whether to show information of the progress. Default `getOption("ligerVerbose")` or `TRUE` if users have not set.

`sample.dirs`, `sample.names`, `use.filtered`
These arguments are renamed and will be deprecated in the future. Please see usage for corresponding arguments.

`data.type`, `merge`, `num.cells`, `min.umis`
These arguments are defuncted because the functionality can/should be fulfilled with other functions.

`...` Arguments passed to `read10X`

`pipeline` Which cellRanger pipeline type to find the ATAC data. Choose `"atac"` to read the peak matrix from cellranger-atac pipeline output folder(s), or `"arc"` to split the ATAC feature subset out from the multiomic cellranger-arc pipeline output folder(s). Default `"atac"`.

`arcFeatureType` When `pipeline = "arc"`, which feature type is for the ATAC data of interests. Default `"Peaks"`. Other possible feature types can be `"Chromatin Accessibility"`. Error message will show available options if argument specification cannot be found.

Value

- When only one sample is given or detected, and only one feature type is detected or using CellRanger < 3.0, and `returnList = FALSE`, a sparse matrix object (dgCMatrix class) will be returned.
- When using `read10XRNA` or `read10XATAC`, which are modality specific, returns a list named by samples, and each element is the corresponding sparse matrix object (dgCMatrix class).
- `read10X` generally returns a list named by samples. Each sample element will be another list named by feature types even if only one feature type is detected (or using CellRanger < 3.0) for data structure consistency. The feature type "Gene Expression" always comes as the first type if available.

Examples

```
## Not run:
# For output from CellRanger < 3.0
dir <- 'path/to/data/directory'
list.files(dir) # Should show barcodes.tsv, genes.tsv, and matrix.mtx
mat <- read10X(dir)
class(mat) # Should show dgCMatrix

# For root directory from CellRanger < 3.0
dir <- 'path/to/root'
list.dirs(dir) # Should show sample names
matList <- read10X(dir)
names(matList) # Should show the sample names
class(matList[[1]][["Gene Expression"]]) # Should show dgCMatrix

# For output from CellRanger >= 3.0 with multiple data types
dir <- 'path/to/data/directory'
list.files(dir) # Should show barcodes.tsv.gz, features.tsv.gz, and matrix.mtx.gz
matList <- read10X(dir, sampleNames = "tissue1")
names(matList) # Shoud show "tissue1"
names(matList$tissue1) # Should show feature types, e.g. "Gene Expression" and etc.

# For root directory from CellRanger >= 3.0 with multiple data types
dir <- 'path/to/root'
list.dirs(dir) # Should show sample names, e.g. "rep1", "rep2", "rep3"
matList <- read10X(dir)
names(matList) # Should show the sample names: "rep1", "rep2", "rep3"
names(matList$rep1) # Should show the avalable feature types for rep1

## End(Not run)
## Not run:
# For creating LIGER object from root directory of CellRanger >= 3.0
dir <- 'path/to/root'
list.dirs(dir) # Should show sample names, e.g. "rep1", "rep2", "rep3"
matList <- read10XRNA(dir)
names(matList) # Should show the sample names: "rep1", "rep2", "rep3"
sapply(matList, class) # Should show matrix class all are "dgCMatrix"
lig <- createLigerObject(matList)

## End(Not run)
```

`readLiger` *Read liger object from RDS file*

Description

This file reads a liger object stored in RDS files under all kinds of types. 1. A liger object with in-memory data created from package version since 1.99. 2. A liger object with on-disk H5 data associated, where the link to H5 files will be automatically restored. 3. A liger object created with older package version, and can be updated to the latest data structure by default.

Usage

```
readLiger(
  filename,
  dimredName = "tsne_coords",
  clusterName = "clusters",
  h5FilePath = NULL,
  update = TRUE
)
```

Arguments

filename	Path to an RDS file of a liger object of old versions.
dimredName	The name of variable in cellMeta slot to store the dimensionality reduction matrix, which originally located in tsne.coords slot. Default "tsne.coords".
clusterName	The name of variable in cellMeta slot to store the clustering assignment, which originally located in clusters slot. Default "clusters".
h5FilePath	Named list, to specify the path to the H5 file of each dataset if location has been changed. Default NULL looks at the file paths stored in object.
update	Logical, whether to update an old (<=1.0.0) liger object to the currect version of structure. Default TRUE.

Value

New version of liger object

Examples

```
# Save and read regular current-version liger object
tempPath <- tempfile(fileext = ".rds")
saveRDS(pbmc, tempPath)
pbmc <- readLiger(tempPath, dimredName = NULL)

# Save and read H5-based liger object
h5Path <- system.file("extdata/ctrl.h5", package = "rliger")
h5tempPath <- tempfile(fileext = ".h5")
file.copy(from = h5Path, to = h5tempPath)
lig <- createLiger(list(ctrl = h5tempPath))
tempPath <- tempfile(fileext = ".rds")
saveRDS(lig, tempPath)
lig <- readLiger(tempPath)

## Not run:
# Read a old liger object <= 1.0.1
# Assume the dimensionality reduction method applied was UMAP
# Assume the clustering was derived with Louvain method
lig <- readLiger(
    filename = "path/to/oldLiger.rds",
    dimredName = "UMAP",
    clusterName = "louvain",
```

```
    update = TRUE
)

## End(Not run)
```

---

readSubset *[Deprecated] See* downsample

Description

This function mainly aims at downsampling datasets to a size suitable for plotting.

Usage

```
readSubset(
  object,
  slot.use = "normData",
  balance = NULL,
  max.cells = 1000,
  chunk = 1000,
  datasets.use = NULL,
  genes.use = NULL,
  rand.seed = 1,
  verbose = getOption("ligerVerbose", TRUE)
)
```

Arguments

object	liger object
slot.use	Only create subset from one or more of "rawData", "normData" and "scaleData". Default NULL subsets the whole object including downstream results.
balance	"all" for sampling maxCells cells from all datasets specified by useDatasets. "cluster" for sampling maxCells cells per cluster per dataset. "dataset" for maxCells cells per dataset.
max.cells	Max number of cells to sample from the grouping based on balance.
chunk	Integer. Number of maximum number of cells in each chunk, Default 1000.
datasets.use	Index selection of datasets to consider. Default NULL for using all datasets.
genes.use	Character vector. Subset features to this specified range. Default NULL does not subset features.
rand.seed	Random seed for reproducibility. Default 1.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.

Value

Subset of liger object.

See Also

downsample, subsetLiger, subsetLigerDataset

---

removeMissing *Remove missing cells or features from liger object*

Description

Remove missing cells or features from liger object

Usage

```
removeMissing(
  object,
  orient = c("both", "feature", "cell"),
  minCells = NULL,
  minFeatures = NULL,
  useDatasets = NULL,
  newH5 = TRUE,
  filenameSuffix = "removeMissing",
  verbose = getOption("ligerVerbose", TRUE),
  ...
)

removeMissingObs(
  object,
  slot.use = NULL,
  use.cols = TRUE,
  verbose = getOption("ligerVerbose", TRUE)
)
```

Arguments

object	liger object
orient	Choose to remove non-expressing features ("feature"), empty barcodes ("cell"), or both of them ("both"). Default "both".
minCells	Keep features that are expressed in at least this number of cells, calculated on a per-dataset base. A single value for all datasets or a vector for each dataset. Default NULL only removes none expressing features.
minFeatures	Keep cells that express at least this number of features, calculated on a per-dataset base. A single value for all datasets or a vector for each dataset. Default NULL only removes none expressing cells.
useDatasets	A character vector of the names, a numeric or logical vector of the index of the datasets to be processed. Default NULL removes empty entries from all datasets.

newH5	Logical, whether to create a new H5 file on disk for each H5-based dataset on subset. Default TRUE
filenameSuffix	When subsetting H5-based datasets to new H5 files, this suffix will be added to all the filenames. Default "removeMissing".
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
...	Arguments passed to subsetLigerDataset
slot.use	**Deprecated**. Always look at rawData slot of inner ligerDataset objects.
use.cols	**Deprecated**. Previously means "treating each column as a cell" when TRUE, now means orient="cell".

Value

Updated (subset) object.

Note

removeMissingObs will be deprecated. removeMissing covers and expands the use case and should be easier to understand.

Examples

```
# The example dataset does not contain non-expressing genes or empty barcodes
pbmc <- removeMissing(pbmc)
```

---

restoreH5Liger *Restore links (to HDF5 files) for reloaded liger/ligerDataset object*

Description

When loading the saved liger object with HDF5 data in a new R session, the links to HDF5 files would be closed. This function enables the restoration of those links so that new analyses can be carried out.

Usage

```
restoreH5Liger(object, filePath = NULL)

restoreOnlineLiger(object, file.path = NULL)
```

Arguments

object	liger or ligerDataset object.
filePath	Paths to HDF5 files. A single character path for ligerDataset input or a list of paths named by the datasets for liger object input. Default NULL looks for the path(s) of the last valid loading.
file.path	Will be deprecated with restoreOnlineLiger. The same as filePath.

Value

`object` with restored links.

Note

`restoreOnlineLiger` will be deprecated for clarifying the terms used for data structure.

Examples

```
h5Path <- system.file("extdata/ctrl.h5", package = "rliger")
tempPath <- tempfile(fileext = ".h5")
file.copy(from = h5Path, to = tempPath)
lig <- createLiger(list(ctrl = tempPath))
# Now it is actually an invalid object! which is equivalent to what users
# will get with `saveRDS(lig, "object.rds"); lig <- readRDS("object.rds")``
closeAllH5(lig)
lig <- restoreH5Liger(lig)
```

`retrieveCellFeature` *Retrieve a single matrix of cells from a slot*

Description

Only retrieve data from specific slot to reduce memory used by a whole liger object of the subset. Useful for plotting. Internally used by `plotDimRed` and `plotCellViolin`.

Usage

```
retrieveCellFeature(
  object,
  feature,
  slot = c("rawData", "normData", "scaleData", "H", "H.norm", "cellMeta", "rawPeak",
    "normPeak"),
  cellIdx = NULL,
  ...
)
```

Arguments

`object`	liger object
`feature`	Gene names, factor index or cell metadata variable names. Should be available in specified `slot`.
`slot`	Exactly choose from `"rawData"`, `"normData"`, `"scaleData"`, `"H"`, `"H.norm"` or `"cellMeta"`.
`cellIdx`	Any valid type of index that subset from all cells. Default `NULL` uses all cells.
`...`	Additional arguments passed to `subsetLiger` when `slot` is one of `"rawData"`, `"normData"` or `"scaleData"`.

Value

A matrix object where rows are cells and columns are specified features.

Examples

```
S100A8Exp <- retrieveCellFeature(pbmc, "S100A8")
qcMetrics <- retrieveCellFeature(pbmc, c("nUMI", "nGene", "mito"),
                                 slot = "cellMeta")
```

---

reverseMethData *Create "scaled data" for DNA methylation datasets*

Description

Because gene body mCH proportions are negatively correlated with gene expression level in neurons, we need to reverse the direction of the methylation data. We do this by simply subtracting all values from the maximum methylation value. The resulting values are positively correlated with gene expression. This will only be applied to variable genes detected in prior.

Usage

```
reverseMethData(object, useDatasets, verbose = getOption("ligerVerbose", TRUE))
```

Arguments

object	A liger object, with variable genes identified.
useDatasets	Required. A character vector of the names, a numeric or logical vector of the index of the datasets that should be identified as methylation data where the reversed data will be created.
verbose	Logical. Whether to show information of the progress. Default `getOption("ligerVerbose")` or TRUE if users have not set.

Value

The input liger object, where the `scaleData` slot of the specified datasets will be updated with value as described in Description.

Examples

```
# Assuming the second dataset in example data "pbmc" is methylation data
pbmc <- normalize(pbmc, useDatasets = 1)
pbmc <- selectGenes(pbmc, datasets.use = 1)
pbmc <- scaleNotCenter(pbmc, useDatasets = 1)
pbmc <- reverseMethData(pbmc, useDatasets = 2)
```

runCINMF *Perform consensus iNMF on scaled datasets*

Description

NOT STABLE - This is an experimental function and is subject to change.

Performs consensus integrative non-negative matrix factorization (c-iNMF) to return factorized H, W, and V matrices. In order to address the non-convex nature of NMF, we built on the cNMF method proposed by D. Kotliar, 2019. We run the regular iNMF multiple times with different random starts, and cluster the pool of all the factors in W and Vs and take the consensus of the clusters of the largest population. The cell factor loading H matrices are eventually solved with the consensus W and V matrices.

Please see runINMF for detailed introduction to the regular iNMF algorithm which is run multiple times in this function.

The consensus iNMF algorithm is developed basing on the consensus NMF (cNMF) method (D. Kotliar et al., 2019).

Usage

```
runCINMF(object, k = 20, lambda = 5, rho = 0.3, ...)

## S3 method for class 'liger'
runCINMF(
  object,
  k = 20,
  lambda = 5,
  rho = 0.3,
  nIteration = 30,
  nRandomStarts = 10,
  HInit = NULL,
  WInit = NULL,
  VInit = NULL,
  seed = 1,
  nCores = 2L,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)

## S3 method for class 'Seurat'
runCINMF(
  object,
  k = 20,
  lambda = 5,
  rho = 0.3,
  datasetVar = "orig.ident",
  layer = "ligerScaleData",
```

```
  assay = NULL,
  reduction = "cinmf",
  nIteration = 30,
  nRandomStarts = 10,
  HInit = NULL,
  WInit = NULL,
  VInit = NULL,
  seed = 1,
  nCores = 2L,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

object	A liger object or a Seurat object with non-negative scaled data of variable features (Done with scaleNotCenter).
k	Inner dimension of factorization (number of factors). Generally, a higher k will be needed for datasets with more sub-structure. Default 20.
lambda	Regularization parameter. Larger values penalize dataset-specific effects more strongly (i.e. alignment should increase as lambda increases). Default 5.
rho	Numeric number between 0 and 1. Fraction for determining the number of nearest neighbors to look at for consensus (by rho * nRandomStarts). Default 0.3.
...	Arguments passed to methods.
nIteration	Total number of block coordinate descent iterations to perform. Default 30.
nRandomStarts	Number of replicate runs for creating the pool of factorization results. Default 10.
HInit	Initial values to use for H matrices. A list object where each element is the initial H matrix of each dataset. Default NULL.
WInit	Initial values to use for W matrix. A matrix object. Default NULL.
VInit	Initial values to use for V matrices. A list object where each element is the initial V matrix of each dataset. Default NULL.
seed	Random seed to allow reproducible results. Default 1.
nCores	The number of parallel tasks to speed up the computation. Default 2L. Only supported for platform with OpenMP support.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
datasetVar	Metadata variable name that stores the dataset source annotation. Default "orig.ident".
layer	For Seurat>=4.9.9, the name of layer to retrieve input non-negative scaled data. Default "ligerScaleData". For older Seurat, always retrieve from scale.data slot.
assay	Name of assay to use. Default NULL uses current active assay.
reduction	Name of the reduction to store result. Also used as the feature key. Default "cinmf".

Value

- liger method - Returns updated input liger object
  - A list of all H matrices can be accessed with `getMatrix(object, "H")`
  - A list of all V matrices can be accessed with `getMatrix(object, "V")`
  - The W matrix can be accessed with `getMatrix(object, "W")`
- Seurat method - Returns updated input Seurat object
  - H matrices for all datasets will be concatenated and transposed (all cells by k), and form a DimReduc object in the `reductions` slot named by argument `reduction`.
  - W matrix will be presented as `feature.loadings` in the same DimReduc object.
  - V matrices, an objective error value and the dataset variable used for the factorization is currently stored in `misc` slot of the same DimReduc object.

References

Joshua D. Welch and et al., Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity, Cell, 2019

Dylan Kotliar and et al., Identifying gene expression programs of cell-type identity and cellular activity with single-cell RNA-Seq, eLife, 2019

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    pbmc <- runCINMF(pbmc)
}
```

`runCluster` *SNN Graph Based Community Detection*

Description

After quantile normalization, users can additionally run the Leiden or Louvain algorithm for community detection, which is widely used in single-cell analysis and excels at merging small clusters into broad cell classes.

While using quantile normalized factor loadings (result from `quantileNorm`) is recommended, this function looks for unnormalized factor loadings (result from `runIntegration`) when the former is not available.

Usage

```
runCluster(
  object,
  resolution = 1,
  nNeighbors = 20,
  prune = 1/15,
  eps = 0.1,
  nRandomStarts = 10,
  nIterations = 5,
  method = c("leiden", "louvain"),
  useRaw = NULL,
  useDims = NULL,
  groupSingletons = TRUE,
  saveSNN = FALSE,
  clusterName = paste0(method, "_cluster"),
  seed = 1,
  verbose = getOption("ligerVerbose", TRUE)
)
```

Arguments

object	A liger object. Should have valid factorization result available.
resolution	Numeric, value of the resolution parameter, a larger value results in a larger number of communities with smaller sizes. Default 1.0.
nNeighbors	Integer, the maximum number of nearest neighbors to compute. Default 20.
prune	Numeric. Sets the cutoff for acceptable Jaccard index when computing the neighborhood overlap for the SNN construction. Any edges with values less than or equal to this will be set to 0 and removed from the SNN graph. Essentially sets the stringency of pruning. 0 for no pruning, while 1 prunes everything. Default 1/15.
eps	Numeric, the error bound of the nearest neighbor search. Default 0.1.
nRandomStarts	Integer number of random starts. Will pick the membership with highest quality to return. Default 10.
nIterations	Integer, maximal number of iterations per random start. Default 5.
method	Community detection algorithm to use. Choose from "leiden" or "louvain". Default "leiden".
useRaw	Whether to use un-aligned cell factor loadings (H matrices). Default NULL search for quantile-normalized loadings first and un-aligned loadings then.
useDims	Indices of factors to use for clustering. Default NULL uses all available factors.
groupSingletons	Whether to group single cells that make up their own cluster in with the cluster they are most connected to. Default TRUE, if FALSE, assign all singletons to a "singleton" group.
saveSNN	Logical, whether to store the SNN graph, as a dgCMatrix object, in the object. Default FALSE.

clusterName	Name of the variable that will store the clustering result in cellMeta slot of object. Default "leiden_cluster" and "louvain_cluster".
seed	Seed of the random number generator. Default 1.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.

Value

object with cluster assignment updated in clusterName variable in cellMeta slot. Can be fetched with object[[clusterName]]. If saveSNN = TRUE, the SNN graph will be stored at object@uns$snn.

Examples

```
pbmcPlot <- runCluster(pbmcPlot)
head(pbmcPlot$leiden_cluster)
pbmcPlot <- runCluster(pbmcPlot, method = "louvain")
head(pbmcPlot$louvain_cluster)
```

---

runDoubletFinder *Doublet detection with DoubletFinder*

Description

Detect doublet with DoubletFinder. Package "Seurat" and "DoubletFinder" would be required to run this function.

This wrapper runs Seurat PCA workflow (NormalizeData, FindVariableFeatures, ScaleData, RunPCA) with all default settings on each dataset, and then calls DoubletFinder::doubletFinder. Users that prefer having more control on the preprocessing part might consider creating single-sample Seurat object with CreateSeuratObject(rawData(object, "datasetName")).

Usage

```
runDoubletFinder(
  object,
  useDatasets = NULL,
  PCs = 1:10,
  nNeighbors = 20,
  nExp = NULL,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

object	A liger object.
useDatasets	A character vector of the names, a numeric or logical vector of the index of the datasets to run DoubletFinder::doubletFinder with. Default NULL applies to all datasets.
PCs	Specific principal components to use. Default 1:10.
nNeighbors	Number of the PC neighborhood size used to compute pANN. See "See Also". Scalar for all used datasets or vector for each. Default 20.
nExp	The total number of doublet predictions produced. Scalar for all used datasets or vector for each. Default NULL sets a 0.15 proportion.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
...	Additional arguments passed to DoubletFinder::doubletFinder.

Value

Updated object with variables DoubletFinder_pANN and DoubletFinder_classification updated in cellMeta slot

Examples

```
if (requireNamespace("DoubletFinder", quietly = TRUE)) {
    pbmc <- runDoubletFinder(pbmc)
    print(cellMeta(pbmc))
}
```

---

runGeneralQC *General QC for liger object*

Description

Calculate number of UMIs, number of detected features and percentage of feature subset (e.g. mito) expression per cell.

Usage

```
runGeneralQC(
  object,
  mito = TRUE,
  ribo = TRUE,
  hemo = TRUE,
  features = NULL,
  pattern = NULL,
  useDatasets = NULL,
  chunkSize = 1000,
  verbose = getOption("ligerVerbose", TRUE)
)
```

Arguments

object	liger object with rawData available in each ligerDataset embedded
mito, ribo, hemo	Whether to calculate the expression percentage of mitochondrial, ribosomal or hemoglobin genes, respectively. Default TRUE.
features	Feature names matching the feature subsets that users want to calculate the expression percentage with. A vector for a single subset, or a named list for multiple subset. Default NULL.
pattern	Regex patterns for matching the feature subsets that users want to calculate the expression percentage with. A vector for a single subset, or a named list for multiple subset. Default NULL.
useDatasets	A character vector of the names, a numeric or logical vector of the index of the datasets to be included for QC. Default NULL performs QC on all datasets.
chunkSize	Integer number of cells to include in a chunk when working on HDF5 based dataset. Default 1000
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.

Value

Updated object with nUMI, nGene updated in cellMeta(object), as well as expression percentage value for each feature subset.

Examples

```
pbmc <- runGeneralQC(pbmc)
```

---

runGOEnrich *Run Gene Ontology enrichment analysis on differentially expressed genes.*

Description

This function forms genesets basing on the differential expression result, and calls gene ontology (GO) analysis method provided by gprofiler2.

Usage

```
runGOEnrich(
  result,
  group = NULL,
  useBg = TRUE,
  orderBy = "padj",
  logFCThresh = 1,
  padjThresh = 0.05,
  splitReg = FALSE,
  ...
)
```

Arguments

result	Data frame of unfiltered output from runMarkerDEG or runPairwiseDEG.
group	Selection of one group available from result$group. Default NULL uses all groups involved in DE result table.
useBg	Logical, whether to set all genes involved in DE analysis (before threshold filtering) as a domain background of GO analysis. Default TRUE.
orderBy	Name of DE statistics metric to order the gene list for each group. Choose from "logFC" (default), "pval" or "padj". Or set NULL to turn off ranked mode.
logFCThresh	The log2FC threshold above which the genes will be used. Default 1.
padjThresh	The adjusted p-value threshold less than which the genes will be used. Default 0.05.
splitReg	Whether to have queries of both up-regulated and down-regulated genes for each group. Default FALSE only queries up-regulated genes and should be preferred when result comes from marker detection test. When result comes from group-to-group DE test, it is recommended to set splitReg = TRUE.
...	Additional arguments passed to gprofiler2::gost(). Arguments query, custom_bg, domain_scope, and ordered_query are pre-specified by this wrapper function. Users must set organism = "mmusculus" when working on mouse data.

Value

A list object where each element is a result list for a group. Each result list contains two elements:

result	data.frame of main GO analysis result.
meta	Meta information for the query.

See gprofiler2::gost(). for detailed explanation.

References

Kolberg, L. et al, 2020 and Raudvere, U. et al, 2019

Examples

```
res <- runMarkerDEG(pbmcPlot)
# Setting `significant = FALSE` because it's hard for a gene list obtained
# from small test dataset to represent real-life biology.

if (requireNamespace("gprofiler2", quietly = TRUE)) {
    go <- runGOEnrich(res, group = 0, significant = FALSE)
}
```

runGSEA *Analyze biological interpretations of metagene*

Description

Identify the biological pathways (gene sets from Reactome) that each metagene (factor) might belongs to.

Usage

```
runGSEA(
  object,
  genesets = NULL,
  useW = TRUE,
  useV = NULL,
  customGenesets = NULL,
  gene_sets = genesets,
  mat_w = useW,
  mat_v = useV,
  custom_gene_sets = customGenesets
)
```

Arguments

object	A liger object with valid factorization result.
genesets	Character vector of the Reactome gene sets names to be tested. Default NULL uses all the gene sets from the Reactome.
useW	Logical, whether to use the shared factor loadings (W). Default TRUE.
useV	A character vector of the names, a numeric or logical vector of the index of the datasets where the V matrices will be included for analysis. Default NULL uses all datasets.
customGenesets	A named list of character vectors of entrez gene ids. Default NULL uses all the gene symbols from the input matrix.
gene_sets, mat_w, mat_v, custom_gene_sets	**Deprecated**. See Usage section for replacement.

Value

A list of matrices with GSEA analysis for each factor

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly = TRUE) &&
    requireNamespace("reactome.db", quietly = TRUE) &&
    requireNamespace("fgsea", quietly = TRUE) &&
    requireNamespace("AnnotationDbi", quietly = TRUE)) {
```

```
    runGSEA(pbmcPlot)
}
```

---

runINMF *Perform iNMF on scaled datasets*

Description

Performs integrative non-negative matrix factorization (iNMF) (J.D. Welch, 2019) using block coordinate descent (alternating non-negative least squares, ANLS) to return factorized H, W, and V matrices. The objective function is stated as

$$\arg \min_{H \geq 0, W \geq 0, V \geq 0} \sum_{i}^{d} ||E_i - (W + V_i)Hi||_F^2 + \lambda \sum_{i}^{d} ||V_i H_i||_F^2$$

where E_i is the input non-negative matrix of the i'th dataset, d is the total number of datasets. E_i is of size $m \times n_i$ for m variable genes and n_i cells, H_i is of size $n_i \times k$, V_i is of size $m \times k$, and W is of size $m \times k$.

The factorization produces a shared W matrix (genes by k), and for each dataset, an H matrix (k by cells) and a V matrix (genes by k). The H matrices represent the cell factor loadings. W is held consistent among all datasets, as it represents the shared components of the metagenes across datasets. The V matrices represent the dataset-specific components of the metagenes.

This function adopts highly optimized fast and memory efficient implementation extended from Planc (Kannan, 2016). Pre-installation of extension package `RcppPlanc` is required. The underlying algorithm adopts the identical ANLS strategy as `optimizeALS` in the old version of LIGER.

Usage

```
runINMF(object, k = 20, lambda = 5, ...)

## S3 method for class 'liger'
runINMF(
  object,
  k = 20,
  lambda = 5,
  nIteration = 30,
  nRandomStarts = 1,
  HInit = NULL,
  WInit = NULL,
  VInit = NULL,
  seed = 1,
  nCores = 2L,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

```
## S3 method for class 'Seurat'
runINMF(
  object,
  k = 20,
  lambda = 5,
  datasetVar = "orig.ident",
  layer = "ligerScaleData",
  assay = NULL,
  reduction = "inmf",
  nIteration = 30,
  nRandomStarts = 1,
  HInit = NULL,
  WInit = NULL,
  VInit = NULL,
  seed = 1,
  nCores = 2L,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

object	A liger object or a Seurat object with non-negative scaled data of variable features (Done with scaleNotCenter).
k	Inner dimension of factorization (number of factors). Generally, a higher k will be needed for datasets with more sub-structure. Default 20.
lambda	Regularization parameter. Larger values penalize dataset-specific effects more strongly (i.e. alignment should increase as lambda increases). Default 5.
...	Arguments passed to methods.
nIteration	Total number of block coordinate descent iterations to perform. Default 30.
nRandomStarts	Number of restarts to perform (iNMF objective function is non-convex, so taking the best objective from multiple successive initialization is recommended). For easier reproducibility, this increments the random seed by 1 for each consecutive restart, so future factorization of the same dataset can be run with one rep if necessary. Default 1.
HInit	Initial values to use for H matrices. A list object where each element is the initial H matrix of each dataset. Default NULL.
WInit	Initial values to use for W matrix. A matrix object. Default NULL.
VInit	Initial values to use for V matrices. A list object where each element is the initial V matrix of each dataset. Default NULL.
seed	Random seed to allow reproducible results. Default 1.
nCores	The number of parallel tasks to speed up the computation. Default 2L. Only supported for platform with OpenMP support.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.

datasetVar	Metadata variable name that stores the dataset source annotation. Default "orig.ident".
layer	For Seurat>=4.9.9, the name of layer to retrieve input non-negative scaled data. Default "ligerScaleData". For older Seurat, always retrieve from scale.data slot.
assay	Name of assay to use. Default NULL uses current active assay.
reduction	Name of the reduction to store result. Also used as the feature key. Default "inmf".

Value

- liger method - Returns updated input liger object
  - A list of all H matrices can be accessed with getMatrix(object, "H")
  - A list of all V matrices can be accessed with getMatrix(object, "V")
  - The W matrix can be accessed with getMatrix(object, "W")
- Seurat method - Returns updated input Seurat object
  - H matrices for all datasets will be concatenated and transposed (all cells by k), and form a DimReduc object in the reductions slot named by argument reduction.
  - W matrix will be presented as feature.loadings in the same DimReduc object.
  - V matrices, an objective error value and the dataset variable used for the factorization is currently stored in misc slot of the same DimReduc object.

Difference from optimizeALS()

In the old version implementation, we compute the objective error at the end of each iteration, and then compares if the algorithm is reaching a convergence, using an argument thresh. Now, since the computation of objective error is indeed expensive, we canceled this feature and directly runs a default of 30 (nIteration) iterations, which empirically leads to a convergence most of the time. Given that the new version is highly optimized, running this many iteration should be acceptable.

References

Joshua D. Welch and et al., Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity, Cell, 2019

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    pbmc <- runINMF(pbmc)
}
```

`runIntegration` *Integrate scaled datasets with iNMF or variant methods*

Description

LIGER provides dataset integration methods based on iNMF (integrative Non-negative Matrix Factorization [1]) and its variants (online iNMF [2] and UINMF [3]). This function wraps `runINMF`, `runOnlineINMF` and `runUINMF`, of which the help pages have more detailed description.

Usage

```
runIntegration(
  object,
  k = 20,
  lambda = 5,
  method = c("iNMF", "onlineINMF", "UINMF"),
  ...
)

## S3 method for class 'liger'
runIntegration(
  object,
  k = 20,
  lambda = 5,
  method = c("iNMF", "onlineINMF", "UINMF"),
  seed = 1,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)

## S3 method for class 'Seurat'
runIntegration(
  object,
  k = 20,
  lambda = 5,
  method = c("iNMF", "onlineINMF"),
  datasetVar = "orig.ident",
  useLayer = "ligerScaleData",
  assay = NULL,
  seed = 1,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

`object`	A liger object or a Seurat object with non-negative scaled data of variable features (Done with `scaleNotCenter`).

k	Inner dimension of factorization (number of factors). Generally, a higher k will be needed for datasets with more sub-structure. Default 20.
lambda	Regularization parameter. Larger values penalize dataset-specific effects more strongly (i.e. alignment should increase as lambda increases). Default 5.
method	iNMF variant algorithm to use for integration. Choose from "iNMF", "onlineINMF", "UINMF". Default "iNMF".
...	Arguments passed to other methods and wrapped functions.
seed	Random seed to allow reproducible results. Default 1.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
datasetVar	Metadata variable name that stores the dataset source annotation. Default "orig.ident".
useLayer	For Seurat>=4.9.9, the name of layer to retrieve input non-negative scaled data. Default "ligerScaleData". For older Seurat, always retrieve from scale.data slot.
assay	Name of assay to use. Default NULL uses current active assay.

Value

Updated input object. For detail, please refer to the refered method linked in Description.

References

1. Joshua D. Welch and et al., Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity, Cell, 2019
2. Chao Gao and et al., Iterative single-cell multi-omic integration using online learning, Nat Biotechnol., 2021
3. April R. Kriebel and Joshua D. Welch, UINMF performs mosaic integration of single-cell multi-omic datasets using nonnegative matrix factorization, Nat. Comm., 2022

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    pbmc <- runIntegration(pbmc)
}
```

`runOnlineINMF` *Perform online iNMF on scaled datasets*

Description

Perform online integrative non-negative matrix factorization to represent multiple single-cell datasets in terms of H, W, and V matrices. It optimizes the iNMF objective function (see `runINMF`) using online learning (non-negative least squares for H matrices, and hierarchical alternating least squares (HALS) for V matrices and W), where the number of factors is set by `k`. The function allows online learning in 3 scenarios:

1. Fully observed datasets;
2. Iterative refinement using continually arriving datasets;
3. Projection of new datasets without updating the existing factorization

All three scenarios require fixed memory independent of the number of cells.

For each dataset, this factorization produces an H matrix (k by cell), a V matrix (genes by k), and a shared W matrix (genes by k). The H matrices represent the cell factor loadings. W is identical among all datasets, as it represents the shared components of the metagenes across datasets. The V matrices represent the dataset-specific components of the metagenes.

Usage

```
runOnlineINMF(object, k = 20, lambda = 5, ...)

## S3 method for class 'liger'
runOnlineINMF(
  object,
  k = 20,
  lambda = 5,
  newDatasets = NULL,
  projection = FALSE,
  maxEpochs = 5,
  HALSiter = 1,
  minibatchSize = 5000,
  WInit = NULL,
  VInit = NULL,
  AInit = NULL,
  BInit = NULL,
  seed = 1,
  nCores = 2L,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)

## S3 method for class 'Seurat'
```

```
runOnlineINMF(
  object,
  k = 20,
  lambda = 5,
  datasetVar = "orig.ident",
  layer = "ligerScaleData",
  assay = NULL,
  reduction = "onlineINMF",
  maxEpochs = 5,
  HALSiter = 1,
  minibatchSize = 5000,
  seed = 1,
  nCores = 2L,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

`object`	liger object. Scaled data required.
`k`	Inner dimension of factorization–number of metagenes. A value in the range 20-50 works well for most analyses. Default `20`.
`lambda`	Regularization parameter. Larger values penalize dataset-specific effects more strongly (i.e. alignment should increase as lambda increases). We recommend always using the default value except possibly for analyses with relatively small differences (biological replicates, male/female comparisons, etc.) in which case a lower value such as 1.0 may improve reconstruction quality. Default `5.0`.
`...`	Arguments passed to other S3 methods of this function.
`newDatasets`	Named list of dgCMatrix. New datasets for scenario 2 or scenario 3. Default `NULL` triggers scenario 1.
`projection`	Whether to perform data integration with scenario 3 when `newDatasets` is specified. See description. Default `FALSE`.
`maxEpochs`	The number of epochs to iterate through. See detail. Default `5`.
`HALSiter`	Maximum number of block coordinate descent (HALS algorithm) iterations to perform for each update of W and V. Default `1`. Changing this parameter is not recommended.
`minibatchSize`	Total number of cells in each minibatch. See detail. Default `5000`.
`WInit, VInit, AInit, BInit`	Optional initialization for W, V, A, and B matrices, respectively. Must be presented all together. See detail. Default `NULL`.
`seed`	Random seed to allow reproducible results. Default `1`.
`nCores`	The number of parallel tasks to speed up the computation. Default `2L`. Only supported for platform with OpenMP support.
`verbose`	Logical. Whether to show information of the progress. Default `getOption("ligerVerbose")` or `TRUE` if users have not set.

datasetVar	Metadata variable name that stores the dataset source annotation. Default "orig.ident".
layer	For Seurat>=4.9.9, the name of layer to retrieve input non-negative scaled data. Default "ligerScaleData". For older Seurat, always retrieve from scale.data slot.
assay	Name of assay to use. Default NULL uses current active assay.
reduction	Name of the reduction to store result. Also used as the feature key. Default "onlineINMF".

Details

For performing scenario 2 or 3, a complete set of factorization result from a run of scenario 1 is required. Given the structure of a liger object, all of the required information can be retrieved automatically. Under the circumstance where users need customized information for existing factorization, arguments WInit, VInit, AInit and BInit are exposed. The requirements for these argument follows:

- WInit - A matrix object of size $m \times k$. (see runINMF for notation)
- VInit - A list object of matrices each of size $m \times k$. Number of matrices should match with newDatasets.
- AInit - A list object of matrices each of size $k \times k$. Number of matrices should match with newDatasets.
- BInit - A list object of matrices each of size $m \times k$. Number of matrices should match with newDatasets.

Minibatch iterations is performed on small subset of cells. The exact minibatch size applied on each dataset is minibatchSize multiplied by the proportion of cells in this dataset out of all cells. In general, minibatchSize should be no larger than the number of cells in the smallest dataset (considering both object and newDatasets). Therefore, a smaller value may be necessary for analyzing very small datasets.

An epoch is one completion of calculation on all cells after a number of iterations of minibatches. Therefore, the total number of iterations is determined by the setting of maxEpochs, total number of cells, and minibatchSize.

Currently, Seurat S3 method does not support working on Scenario 2 and 3, because there is no simple solution for organizing a number of miscellaneous matrices with a single Seurat object. We strongly recommend that users create a liger object which has the specific structure.

Value

- liger method - Returns updated input liger object.
  - A list of all H matrices can be accessed with getMatrix(object, "H")
  - A list of all V matrices can be accessed with getMatrix(object, "V")
  - The W matrix can be accessed with getMatrix(object, "W")
  - Meanwhile, intermediate matrices A and B produced in HALS update can also be accessed similarly.
- Seurat method - Returns updated input Seurat object.

- H matrices for all datasets will be concatenated and transposed (all cells by k), and form a DimReduc object in the `reductions` slot named by argument `reduction`.
- W matrix will be presented as `feature.loadings` in the same DimReduc object.
- V matrices, A matrices, B matricesm an objective error value and the dataset variable used for the factorization is currently stored in `misc` slot of the same DimReduc object.

References

Chao Gao and et al., Iterative single-cell multi-omic integration using online learning, Nat Biotechnol., 2021

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
if (requireNamespace("RcppPlanc", quietly = TRUE)) {
    # Scenario 1
    pbmc <- runOnlineINMF(pbmc, minibatchSize = 200)
    # Scenario 2
    # Fake new dataset by increasing all non-zero value in "ctrl" by 1
    ctrl2 <- rawData(dataset(pbmc, "ctrl"))
    ctrl2@x <- ctrl2@x + 1
    colnames(ctrl2) <- paste0(colnames(ctrl2), 2)
    pbmc2 <- runOnlineINMF(pbmc, k = 20, newDatasets = list(ctrl2 = ctrl2),
                           minibatchSize = 100)
    # Scenario 3
    pbmc3 <- runOnlineINMF(pbmc, k = 20, newDatasets = list(ctrl2 = ctrl2),
                           projection = TRUE)
}
```

---

runPairwiseDEG *Find DEG between two groups*

Description

Find DEG between two groups. Two methods are supported: `"wilcoxon"` and `"pseudoBulk"`. Wilcoxon rank sum test is performed on single-cell level, while pseudo-bulk method aggregates cells basing on biological replicates and calls bulk RNAseq DE methods, DESeq2 wald test. When real biological replicates are not available, pseudo replicates can be generated. Please see below for detailed scenario usage.

Usage

```
runPairwiseDEG(
  object,
  groupTest,
  groupCtrl,
```

```
  variable1 = NULL,
  variable2 = NULL,
  method = c("wilcoxon", "pseudoBulk"),
  usePeak = FALSE,
  useReplicate = NULL,
  nPsdRep = 5,
  minCellPerRep = 10,
  seed = 1,
  verbose = getOption("ligerVerbose", TRUE)
)

runMarkerDEG(
  object,
  conditionBy = NULL,
  splitBy = NULL,
  method = c("wilcoxon", "pseudoBulk"),
  useDatasets = NULL,
  usePeak = FALSE,
  useReplicate = NULL,
  nPsdRep = 5,
  minCellPerRep = 10,
  seed = 1,
  verbose = getOption("ligerVerbose", TRUE)
)

runWilcoxon(
  object,
  data.use = NULL,
  compare.method = c("clusters", "datasets")
)
```

Arguments

`object`	A liger object, with normalized data available
`groupTest`, `groupCtrl`, `variable1`, `variable2`	Condition specification. See `?runPairwiseDEG` section **Pairwise DEG Scenarios** for detail.
`method`	DEG test method to use. Choose from `"wilcoxon"` or `"pseudoBulk"`. Default `"wilcoxon"`
`usePeak`	Logical. Whether to use peak count instead of gene count. Only supported when ATAC datasets are involved. Default `FALSE`.
`useReplicate`	`cellMeta` variable of biological replicate annotation. Only used with `method = "pseudoBulk"`. Default `NULL` will create `nPsdRep` pseudo replicates per group.
`nPsdRep`	Number of pseudo replicates to create. Only used when `method = "pseudoBulk"`, `useReplicate = NULL`. Default `5`.
`minCellPerRep`	Numeric, will not make pseudo-bulk for replicate with less than this number of cells. Default `10`.

seed	Random seed to use for pseudo-replicate generation. Default 1.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
conditionBy	cellMeta variable(s). Marker detection will be performed for each level of this variable. Multiple variables will be combined. Default NULL uses default cluster.
splitBy	Split data by cellMeta variable(s) here and identify markers for conditionBy within each chunk. Default NULL.
useDatasets	Datasets to perform marker detection within. Default NULL will use all datasets.
data.use	Same as useDatasets.
compare.method	Choose from "clusters" (default) or "datasets". "clusters" compares each cluster against all other cells, while "datasets" run within each cluster and compare each dataset against all other datasets.

Value

A data.frame with DEG information

Pairwise DEG Scenarios

Users can select classes of cells from a variable in cellMeta. variable1 and variable2 are used to specify a column in cellMeta, and groupTest and groupCtrl are used to specify existing classes from variable1 and variable2, respectively. When variable2 is missing, groupCtrl will be considered from variable1.

For example, when variable1 = "celltype" and variable2 = NULL, groupTest and groupCtrl should be valid cell types in object$celltype.

When variable1 is "celltype" and variable2 is "gender", groupTest should be a valid cell type from object$celltype and groupCtrl should be a valid class from object$gender.

When both variable1 and variable2 are missing, groupTest and groupCtrl should be valid index of cells in object.

Marker Detection Scenarios

Marker detection is generally performed in a one vs. rest manner. The grouping of such condition is specified by conditionBy, which should be a column name in cellMeta. When splitBy is specified as another variable name in cellMeta, the marker detection will be iteratively done for within each level of splitBy variable.

For example, when conditionBy = "celltype" and splitBy = NULL, marker detection will be performed by comparing all cells of "celltype_i" against all other cells, and etc.

When conditionBy = "celltype" and splitBy = "gender", marker detection will be performed by comparing "celltype_i" cells from "gender_j" against other cells from "gender_j", and etc.

Examples

```
# Compare between cluster "0" and cluster "1"
degStats <- runPairwiseDEG(pbmcPlot, groupTest = 0, groupCtrl = 1,
                           variable1 = "leiden_cluster")
```

```
# Compare between all cells from cluster "5" and
# all cells from dataset "stim"
degStats <- runPairwiseDEG(pbmcPlot, groupTest = "5", groupCtrl = "stim",
                           variable1 = "leiden_cluster",
                           variable2 = "dataset")
# Identify markers for each cluster. Equivalent to old version
# `runWilcoxon(method = "cluster")`
markerStats <- runMarkerDEG(pbmcPlot, conditionBy = "leiden_cluster")
# Identify dataset markers within each cluster. Equivalent to old version
# `runWilcoxon(method = "dataset")`.
markerStatsList <- runMarkerDEG(pbmcPlot, conditionBy = "dataset",
                                splitBy = "leiden_cluster")
```

---

runTSNE *Perform t-SNE dimensionality reduction*

Description

Runs t-SNE on the quantile normalized cell factors (result from quantileNorm), or unnormalized cell factors (result from runIntegration)) to generate a 2D embedding for visualization. By default Rtsne (Barnes-Hut implementation of t-SNE) method is invoked, while alternative "fftRtsne" method (FFT-accelerated Interpolation-based t-SNE, using Kluger Lab implementation) is also supported. For very large datasets, it is recommended to use `method = "fftRtsne"` due to its efficiency and scalability.

Extra external installation steps are required for using "fftRtsne" method. Please consult detailed guide.

Usage

```
runTSNE(
  object,
  useRaw = NULL,
  useDims = NULL,
  nDims = 2,
  usePCA = FALSE,
  perplexity = 30,
  theta = 0.5,
  method = c("Rtsne", "fftRtsne"),
  dimredName = "TSNE",
  fitsnePath = NULL,
  seed = 42,
  verbose = getOption("ligerVerbose", TRUE),
  k = nDims,
  use.raw = useRaw,
  dims.use = useDims,
  use.pca = usePCA,
  fitsne.path = fitsnePath,
  rand.seed = seed
)
```

Arguments

object	liger object with factorization results.
useRaw	Whether to use un-aligned cell factor loadings (H matrices). Default NULL search for quantile-normalized loadings first and un-aligned loadings then.
useDims	Index of factors to use for computing UMAP embedding. Default NULL uses all factors.
nDims	Number of dimensions to reduce to. Default 2.
usePCA	Whether to perform initial PCA step for Rtsne. Default FALSE.
perplexity	Numeric parameter to pass to Rtsne (expected number of neighbors). Default 30.
theta	Speed/accuracy trade-off (increase for less accuracy), set to 0.0 for exact TSNE. Default 0.5.
method	Choose from "Rtsne" or "fftRtsne". See Description. Default "Rtsne".
dimredName	Name of the variable in cellMeta slot to store the result matrix. Default "TSNE".
fitsnePath	Path to the cloned FIt-SNE directory (i.e. '/path/to/dir/FIt-SNE'). Required only when first time using runTSNE with method = "fftRtsne". Default NULL.
seed	Random seed for reproducibility. Default 42.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
use.raw, dims.use, k, use.pca, fitsne.path, rand.seed	**Deprecated**. See Usage section for replacement.

Value

The object where a "TSNE" variable is updated in the cellMeta slot with the whole 2D embedding matrix.

See Also

runUMAP

Examples

```
pbmc <- runTSNE(pbmcPlot)
```

runUINMF *Perform Mosaic iNMF (UINMF) on scaled datasets with unshared features*

Description

Performs mosaic integrative non-negative matrix factorization (UINMF) (A.R. Kriebel, 2022) using block coordinate descent (alternating non-negative least squares, ANLS) to return factorized H, W, V and U matrices. The objective function is stated as

$$\arg \min_{H\geq 0, W\geq 0, V\geq 0, U\geq 0} \sum_{i}^{d} || \begin{bmatrix} E_i \\ P_i \end{bmatrix} - (\begin{bmatrix} W \\ 0 \end{bmatrix} + \begin{bmatrix} V_i \\ U_i \end{bmatrix}) Hi||_F^2 + \lambda_i \sum_{i}^{d} || \begin{bmatrix} V_i \\ U_i \end{bmatrix} H_i||_F^2$$

where E_i is the input non-negative matrix of the i'th dataset, P_i is the input non-negative matrix for the unshared features, d is the total number of datasets. E_i is of size $m \times n_i$ for m shared features and n_i cells, P_i is of size $u_i \times n_i$ for u_i unshared feaetures, H_i is of size $k \times n_i$, V_i is of size $m \times k$, W is of size $m \times k$ and U_i is of size $u_i \times k$.

The factorization produces a shared W matrix (genes by k). For each dataset, an H matrix (k by cells), a V matrix (genes by k) and a U matrix (unshared genes by k). The H matrices represent the cell factor loadings. W is held consistent among all datasets, as it represents the shared components of the metagenes across datasets. The V matrices represent the dataset-specific components of the metagenes, U matrices are similar to Vs but represents the loading contributed by unshared features.

This function adopts highly optimized fast and memory efficient implementation extended from Planc (Kannan, 2016). Pre-installation of extension package `RcppPlanc` is required. The underlying algorithm adopts the identical ANLS strategy as `optimizeALS(unshared = TRUE)` in the old version of LIGER.

Usage

```
runUINMF(object, k = 20, lambda = 5, ...)

## S3 method for class 'liger'
runUINMF(
  object,
  k = 20,
  lambda = 5,
  nIteration = 30,
  nRandomStarts = 1,
  seed = 1,
  nCores = 2L,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

object	liger object. Should run selectGenes with unshared = TRUE and then run scaleNotCenter in advance.
k	Inner dimension of factorization (number of factors). Generally, a higher k will be needed for datasets with more sub-structure. Default 20.
lambda	Regularization parameter. Larger values penalize dataset-specific effects more strongly (i.e. alignment should increase as lambda increases). Default 5.
...	Arguments passed to other methods and wrapped functions.
nIteration	Total number of block coordinate descent iterations to perform. Default 30.
nRandomStarts	Number of restarts to perform (iNMF objective function is non-convex, so taking the best objective from multiple successive initialization is recommended). For easier reproducibility, this increments the random seed by 1 for each consecutive restart, so future factorization of the same dataset can be run with one rep if necessary. Default 1.
seed	Random seed to allow reproducible results. Default 1.
nCores	The number of parallel tasks to speed up the computation. Default 2L. Only supported for platform with OpenMP support.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.

Value

- liger method - Returns updated input liger object.
  - A list of all H matrices can be accessed with getMatrix(object, "H")
  - A list of all V matrices can be accessed with getMatrix(object, "V")
  - The W matrix can be accessed with getMatrix(object, "W")
  - A list of all U matrices can be accessed with getMatrix(object, "U")

Note

Currently, Seurat S3 method is not supported for UINMF because there is no simple solution for organizing a number of miscellaneous matrices with a single Seurat object. We strongly recommend that users create a liger object which has the specific structure.

References

April R. Kriebel and Joshua D. Welch, UINMF performs mosaic integration of single-cell multi-omic datasets using nonnegative matrix factorization, Nat. Comm., 2022

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc, useUnsharedDatasets = c("ctrl", "stim"))
pbmc <- scaleNotCenter(pbmc)
if (!is.null(getMatrix(pbmc, "scaleUnsharedData", "ctrl")) &&
    !is.null(getMatrix(pbmc, "scaleUnsharedData", "stim"))) {
```

```
    # TODO: unshared variable features cannot be detected from this example
    pbmc <- runUINMF(pbmc)
}
```

---

runUMAP *Perform UMAP Dimensionality Reduction*

Description

Run UMAP on the quantile normalized cell factors (result from quantileNorm), or unnormalized cell factors (result from runIntegration)) to generate a 2D embedding for visualization (or general dimensionality reduction). Has option to run on subset of factors. It is generally recommended to use this method for dimensionality reduction with extremely large datasets. The underlying UMAP calculation imports uwot umap.

Usage

```
runUMAP(
  object,
  useRaw = NULL,
  useDims = NULL,
  nDims = 2,
  distance = c("cosine", "euclidean", "manhattan", "hamming"),
  nNeighbors = 20,
  minDist = 0.1,
  dimredName = "UMAP",
  seed = 42,
  verbose = getOption("ligerVerbose", TRUE),
  k = nDims,
  use.raw = useRaw,
  dims.use = useDims,
  n_neighbors = nNeighbors,
  min_dist = minDist,
  rand.seed = seed
)
```

Arguments

object	liger object with factorization results.
useRaw	Whether to use un-aligned cell factor loadings (H matrices). Default NULL search for quantile-normalized loadings first and un-aligned loadings then.
useDims	Index of factors to use for computing UMAP embedding. Default NULL uses all factors.
nDims	Number of dimensions to reduce to. Default 2.
distance	Character. Metric used to measure distance in the input space. Default "cosine", alternative options include: "euclidean", "manhattan" and "hamming".

nNeighbors	Number of neighboring points used in local approximations of manifold structure. Default 10.
minDist	Numeric. Controls how tightly the embedding is allowed compress points together. Default 0.1.
dimredName	Name of the variable in cellMeta slot to store the result matrix. Default "UMAP".
seed	Random seed for reproducibility. Default 42.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
k, use.raw, dims.use, n_neighbors, min_dist, rand.seed	**Deprecated**. See Usage section for replacement.

Details

For nNeighbors, larger values will result in more global structure being preserved at the loss of detailed local structure. In general this parameter should often be in the range 5 to 50, with a choice of 10 to 15 being a sensible default.

For minDist, larger values ensure embedded points are more evenly distributed, while smaller values allow the algorithm to optimize more accurately with regard to local structure. Sensible values are in the range 0.001 to 0.5, with 0.1 being a reasonable default.

Value

The object where a "UMAP" variable is updated in the cellMeta slot with the whole 2D embedding matrix.

See Also

runTSNE

Examples

```
pbmc <- runUMAP(pbmcPlot)
```

---

scaleNotCenter *Scale genes by root-mean-square across cells*

Description

This function scales normalized gene expression data after variable genes have been selected. We do not mean-center the data before scaling in order to address the non-negativity constraint of NMF. Computation applied to each normalized dataset matrix can form the following equation:

$$S_{i,j} = \frac{N_{i,j}}{\sqrt{\sum_p^n \frac{N_{i,p}^2}{n-1}}}$$

Where N denotes the normalized matrix for an individual dataset, S is the output scaled matrix for this dataset, and n is the number of cells in this dataset. i, j denotes the specific gene and cell index, and p is the cell iterator.

Please see detailed section below for explanation on methylation dataset.

Usage

```
scaleNotCenter(object, ...)

## S3 method for class 'dgCMatrix'
scaleNotCenter(object, ...)

## S3 method for class 'ligerDataset'
scaleNotCenter(
  object,
  features = NULL,
  chunk = 1000,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)

## S3 method for class 'ligerMethDataset'
scaleNotCenter(
  object,
  features = NULL,
  verbose = getOption("ligerVerbose", TRUE),
  ...
)

## S3 method for class 'liger'
scaleNotCenter(
  object,
  useDatasets = NULL,
  features = varFeatures(object),
  verbose = getOption("ligerVerbose", TRUE),
  remove.missing = NULL,
  ...
)

## S3 method for class 'Seurat'
scaleNotCenter(
  object,
  assay = NULL,
  layer = "ligerNormData",
  save = "ligerScaleData",
  datasetVar = "orig.ident",
  features = NULL,
  ...
```

)

Arguments

object	liger object, ligerDataset object, dgCMatrix, or a Seurat object.
...	Arguments passed to other methods. The order goes by: "liger" method calls "ligerDataset" method", which then calls "dgCMatrix" method. "Seurat" method directly calls "dgCMatrix" method.
features	Character, numeric or logical index that choose the variable feature to be scaled. "liger" method by default uses `varFeatures(object)`. "ligerDataset" method by default uses all features. "Seurat" method by default uses `Seurat::VariableFeatures(object)`.
chunk	Integer. Number of maximum number of cells in each chunk, when scaling is applied to any HDF5 based dataset. Default `1000`.
verbose	Logical. Whether to show information of the progress. Default `getOption("ligerVerbose")` or `TRUE` if users have not set.
useDatasets	A character vector of the names, a numeric or logical vector of the index of the datasets to be scaled but not centered. Default `NULL` applies to all datasets.
remove.missing	**Deprecated**. The functionality of this is covered through other parts of the whole workflow and is no long needed. Will be ignored if specified.
assay	Name of assay to use. Default `NULL` uses current active assay.
layer	For Seurat>=4.9.9, the name of layer to retrieve normalized data. Default `"ligerNormData"`. For older Seurat, always retrieve from `data` slot.
save	For Seurat>=4.9.9, the name of layer to store normalized data. Default `"ligerScaleData"`. For older Seurat, stored to `scale.data` slot.
datasetVar	Metadata variable name that stores the dataset source annotation. Default `"orig.ident"`.

Value

Updated `object`

- dgCMatrix method - Returns scaled dgCMatrix object
- ligerDataset method - Updates the `scaleData` and `scaledUnsharedData` (if unshared variable feature available) slot of the object
- liger method - Updates the `scaleData` and `scaledUnsharedData` (if unshared variable feature available) slot of chosen datasets
- Seurat method - Adds a named layer in chosen assay (V5), or update the `scale.data` slot of the chosen assay (<=V4)

Methylation dataset

Because gene body mCH proportions are negatively correlated with gene expression level in neurons, we need to reverse the direction of the methylation data before performing the integration. We do this by simply subtracting all values from the maximum methylation value. The resulting values are positively correlated with gene expression. This will only be applied to variable genes detected in prior. Please make sure that argument `modal` is set accordingly when running `createLiger`. In this way, this function can automatically detect it and take proper action. If it is not set, users can still manually have the equivalent processing done by doing `scaleNotCenter(lig, useDataset = c("other", "datasets"))`, and then `reverseMethData(lig, useDataset = c("meth", "datasets"))`.

Note

Since the scaling on genes is applied on a per dataset base, other scaling methods that apply to a whole concatenated matrix of multiple datasets might not be considered as equivalent alternatives, even if options like `center` are set to `FALSE`. Hence we implemented an efficient solution that works under such circumstance, provided with the Seurat S3 method.

Examples

```
pbmc <- normalize(pbmc)
pbmc <- selectGenes(pbmc)
pbmc <- scaleNotCenter(pbmc)
```

`selectGenes` *Select a subset of informative genes*

Description

This function identifies highly variable genes from each dataset and combines these gene sets (either by union or intersection) for use in downstream analysis. Assuming that gene expression approximately follows a Poisson distribution, this function identifies genes with gene expression variance above a given variance threshold (relative to mean gene expression). Alternatively, we allow selecting a desired number of genes for each dataset by ranking the relative variance, and then take the combination.

Usage

```
selectGenes(object, thresh = 0.1, nGenes = NULL, alpha = 0.99, ...)

## S3 method for class 'liger'
selectGenes(
  object,
  thresh = 0.1,
  nGenes = NULL,
  alpha = 0.99,
  useDatasets = NULL,
  useUnsharedDatasets = NULL,
  unsharedThresh = 0.1,
  combine = c("union", "intersection"),
  chunk = 1000,
  verbose = getOption("ligerVerbose", TRUE),
  var.thresh = thresh,
  alpha.thresh = alpha,
  num.genes = nGenes,
  datasets.use = useDatasets,
  unshared.datasets = useUnsharedDatasets,
  unshared.thresh = unsharedThresh,
  tol = NULL,
```

```
  do.plot = NULL,
  cex.use = NULL,
  unshared = NULL,
  ...
)

## S3 method for class 'Seurat'
selectGenes(
  object,
  thresh = 0.1,
  nGenes = NULL,
  alpha = 0.99,
  useDatasets = NULL,
  layer = "ligerNormData",
  assay = NULL,
  datasetVar = "orig.ident",
  combine = c("union", "intersection"),
  verbose = getOption("ligerVerbose", TRUE),
  ...
)
```

Arguments

object	A liger, ligerDataset or Seurat object, with normalized data available (no scale factor multipled nor log transformed).
thresh	Variance threshold used to identify variable genes. Higher threshold results in fewer selected genes. Liger and Seurat S3 methods accept a single value or a vector with specific threshold for each dataset in useDatasets.* Default 0.1.
nGenes	Number of genes to find for each dataset. By setting this, we optimize the threshold used for each dataset so that we get nGenes selected features for each dataset. Accepts single value or a vector for dataset specific setting matching useDataset.* Default NULL does not optimize.
alpha	Alpha threshold. Controls upper bound for expected mean gene expression. Lower threshold means higher upper bound. Default 0.99.
...	Arguments passed to other methods.
useDatasets	A character vector of the names, a numeric or logical vector of the index of the datasets to use for shared variable feature selection. Default NULL uses all datasets.
useUnsharedDatasets	A character vector of the names, a numeric or logical vector of the index of the datasets to use for finding unshared variable features. Default NULL does not attempt to find unshared features.
unsharedThresh	The same thing as thresh that is applied to test unshared features. A single value for all datasets in useUnsharedDatasets or a vector for dataset-specific setting.* Default 0.1.
combine	How to combine variable genes selected from all datasets. Choose from "union" or "intersection". Default "union".

chunk — Integer. Number of maximum number of cells in each chunk, when gene selection is applied to any HDF5 based dataset. Default 1000.

verbose — Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.

var.thresh, alpha.thresh, num.genes, datasets.use, unshared.datasets, unshared.thresh — **Deprecated**. These arguments are renamed and will be removed in the future. Please see function usage for replacement.

tol, do.plot, cex.use, unshared — **Deprecated**. Gene variability metric is now visualized with separated function plotVarFeatures. Users can now set none-NULL useUnsharedDatasets to select unshared genes, instead of having to switch unshared on.

layer — Where the input normalized counts should be from. Default "ligerNormData". For older Seurat, always retrieve from data slot.

assay — Name of assay to use. Default NULL uses current active assay.

datasetVar — Metadata variable name that stores the dataset source annotation. Default "orig.ident".

Value

Updated object

- liger method - Each involved dataset stored in ligerDataset is updated with its featureMeta slot and varUnsharedFeatures slot (if requested with useUnsharedDatasets), while varFeatures(object) will be updated with the final combined gene set.
- Seurat method - Final selection will be updated at Seurat::VariableFeatures(object). Per-dataset information is stored in the meta.features slot of the chosen Assay.

Examples

```
pbmc <- normalize(pbmc)
# Select basing on thresholding the relative variance
pbmc <- selectGenes(pbmc, thresh = .1)
# Select specified number for each dataset
pbmc <- selectGenes(pbmc, nGenes = c(60, 60))
```

---

selectGenesVST *Select variable genes from one dataset with Seurat VST method*

Description

Seurat FindVariableFeatures VST method. This allows the selection of a fixed number of variable features, but only applies to one dataset. No normalization is needed in advance.

Usage

```
selectGenesVST(
  object,
  useDataset,
  n = 2000,
  loessSpan = 0.3,
  clipMax = "auto",
  useShared = TRUE,
  verbose = getOption("ligerVerbose", TRUE)
)
```

Arguments

object	A liger object.
useDataset	The names, a numeric or logical index of the dataset to be considered for selection.
n	Number of variable features needed. Default 2000.
loessSpan	Loess span parameter used when fitting the variance-mean relationship. Default 0.3.
clipMax	After standardization values larger than clipMax will be set to clipMax. Default "auto" sets this value to the square root of the number of cells.
useShared	Logical. Whether to only select from genes shared by all dataset. Default TRUE.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.

References

Seurat::FindVariableFeatures.default(selection.method = "vst")

Examples

```
pbmc <- selectGenesVST(pbmc, "ctrl", n = 50)
```

---

sub-liger *Subset liger with brackets*

Description

Subset liger with brackets

Usage

```
## S3 method for class 'liger'
x[i, j, ...]
```

Arguments

x	A liger object
i	Feature subscriptor, passed to featureIdx of subsetLiger.
j	Cell subscriptor, passed to cellIdx of subsetLiger.
...	Additional arguments passed to subsetLiger.

Value

Subset of x with specified features and cells.

See Also

subsetLiger

Examples

```
pbmcPlot[varFeatures(pbmcPlot)[1:10], 1:10]
```

---

sub-ligerDataset *Subset ligerDataset object*

Description

Subset ligerDataset object

Usage

```
## S3 method for class 'ligerDataset'
x[i, j, ...]
```

Arguments

x	A ligerDataset object
i	Numeric, logical index or character vector of feature names to subscribe. Leave missing for all features.
j	Numeric, logical index or character vector of cell IDs to subscribe. Leave missing for all cells.
...	Additional arguments passed to subsetLigerDataset.

Value

If i is given, the selected metadata will be returned; if it is missing, the whole cell metadata table in S4Vectors::DataFrame class will be returned.

Examples

```
ctrl <- dataset(pbmc, "ctrl")
ctrl[1:5, 1:5]
```

sub-sub-liger *Get cell metadata variable*

Description

Get cell metadata variable

Usage

```
## S3 method for class 'liger'
x[[i, ...]]
```

Arguments

x	A liger object
i	Name or numeric index of cell meta data to fetch
...	Anything that S4Vectors::DataFrame method allows.

Value

If i is given, the selected metadata will be returned; if it is missing, the whole cell metadata table in S4Vectors::DataFrame class will be returned.

Examples

```
# Retrieve whole cellMeta
pbmc[[]]
# Retrieve a variable
pbmc[["dataset"]]
```

subsetLiger *Subset liger object*

Description

This function subsets a liger object with character feature index and any valid cell index. For datasets based on HDF5, the filenames of subset H5 files could only be automatically generated for now. Feature subsetting is based on the intersection of available features from datasets involved by cellIdx, while featureIdx = NULL does not take the intersection (i.e. nothing done on the feature axis).

a ligerDataset object is also allowed for now and meanwhile, setting filename is supported.

Usage

```
subsetLiger(
  object,
  featureIdx = NULL,
  cellIdx = NULL,
  useSlot = NULL,
  chunkSize = 1000,
  verbose = getOption("ligerVerbose", TRUE),
  newH5 = TRUE,
  returnObject = TRUE,
  ...
)
```

Arguments

object	A liger or ligerDataset object.
featureIdx	Character vector. Missing or NULL for all features.
cellIdx	Character, logical or numeric index that can subscribe cells. Missing or NULL for all cells.
useSlot	The slot(s) to only consider. Choose one or more from "rawData", "normData" and "scaleData". Default NULL subsets the whole object including analysis result matrices.
chunkSize	Integer. Number of maximum number of cells in each chunk, Default 1000.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
newH5	Whether to create new H5 files on disk for the subset datasets if involved datasets in the object is HDF5 based. TRUE writes a new ones, FALSE returns in memory data.
returnObject	Logical, whether to return a liger object for result. Default TRUE. FALSE returns a list containing requested values.
...	Arguments passed to subsetLigerDataset

Value

Subset object

See Also

subsetLigerDataset

Examples

```
pbmc.small <- subsetLiger(pbmc, cellIdx = pbmc$nUMI > 200)
pbmc.small <- pbmc[, pbmc$nGene > 50]
```

`subsetLigerDataset` *Subset ligerDataset object*

Description

This function subsets a ligerDataset object with valid feature and cell indices. For HDF5 based object, options are available for subsetting data into memory or a new on-disk H5 file. Feature and cell subscription is always based on the size of rawData. Therefore, the feature subsetting on scaled data, which usually contains already a subset of features, will select the intersection between the wanted features and the set available from scaled data.

Usage

```
subsetLigerDataset(
  object,
  featureIdx = NULL,
  cellIdx = NULL,
  useSlot = NULL,
  newH5 = TRUE,
  filename = NULL,
  filenameSuffix = NULL,
  chunkSize = 1000,
  verbose = getOption("ligerVerbose", TRUE),
  returnObject = TRUE,
  ...
)

subsetH5LigerDataset(
  object,
  featureIdx = NULL,
  cellIdx = NULL,
  useSlot = NULL,
  newH5 = TRUE,
  filename = NULL,
  filenameSuffix = NULL,
  chunkSize = 1000,
  verbose = getOption("ligerVerbose", TRUE),
  returnObject = TRUE
)

subsetMemLigerDataset(
  object,
  featureIdx = NULL,
  cellIdx = NULL,
  useSlot = NULL,
  returnObject = TRUE
)
```

Arguments

object	ligerDataset object. HDF5 based object if using subsetH5LigerDataset, in-memory data for subsetMemLigerDataset.
featureIdx	Character, logical or numeric index that can subscribe features. Missing or NULL for all features.
cellIdx	Character, logical or numeric index that can subscribe cells. Missing or NULL for all cells.
useSlot	The slot(s) to only consider. Choose one or more from "rawData", "normData" and "scaleData". Default NULL subsets the whole object including analysis result matrices.
newH5	Whether to create a new H5 file on disk for the subset dataset if object is HDF5 based. TRUE writes a new one, FALSE returns in memory data.
filename	Filename of the new H5 file if being created. Default NULL adds suffix ".subset_{yymmdd_HHMMSS}.h5" to the original name.
filenameSuffix	Instead of specifying the exact filename, set a suffix for the new files so the new filename looks like original.h5.[suffix].h5. Default NULL.
chunkSize	Integer. Number of maximum number of cells in each chunk, Default 1000.
verbose	Logical. Whether to show information of the progress. Default getOption("ligerVerbose") or TRUE if users have not set.
returnObject	Logical, whether to return a ligerDataset object for result. Default TRUE. FALSE returns a list containing requested values.
...	Arguments passed to subsetH5LigerDataset

Value

Subset object

Examples

```
ctrl <- dataset(pbmc, "ctrl")
ctrl.small <- subsetLigerDataset(ctrl, cellIdx = 1:5)
ctrl.tiny <- ctrl[1:5, 1:5]
```

---

writeH5 *Write in-memory data into H5 file*

Description

This function writes in-memory data into H5 file by default in 10x cellranger HDF5 output format. The main goal of this function is to allow users to integrate large H5-based dataset, that cannot be fully loaded into memory, with other data already loaded in memory using runOnlineINMF. In this case, users can write the smaller in-memory data to H5 file instead of loading subset of the large H5-based dataset into memory, where information might be lost.

Basing on the goal of the whole workflow, the data will always be written in a CSC matrix format and colnames/rownames are always required.

The default method coerces the input to a dgCMatrix. Methods for other container classes tries to extract proper data and calls the default method.

Usage

```
writeH5(x, file, ...)

## Default S3 method:
writeH5(x, file, ...)

## S3 method for class 'dgCMatrix'
writeH5(
  x,
  file,
  overwrite = FALSE,
  indicesPath = "matrix/indices",
  indptrPath = "matrix/indptr",
  dataPath = "matrix/data",
  shapePath = "matrix/shape",
  barcodesPath = "matrix/barcodes",
  featuresPath = "matrix/features/name",
  ...
)

## S3 method for class 'ligerDataset'
writeH5(x, file, ...)

## S3 method for class 'liger'
writeH5(x, file, useDatasets, ...)
```

Arguments

x	An object with in-memory data to be written into H5 file.
file	A character string of the file path to be written.
...	Arguments passed to other S3 methods.
overwrite	Logical, whether to overwrite the file if it already exists. Default FALSE.
indicesPath, indptrPath, dataPath	The paths inside the H5 file where the dgCMatrix constructor i, p, and x will be written to, respectively. Default using cellranger convention "matrix/indices", "matrix/indptr", and "matrix/data".
shapePath	The path inside the H5 file where the shape of the matrix will be written to. Default "matrix/shape".
barcodesPath	The path inside the H5 file where the barcodes/colnames will be written to. Default "matrix/barcodes". Skipped if the object does not have colnames.

featuresPath	The path inside the H5 file where the features/rownames will be written to. Default "matrix/features/name". Skipped if the object does not have rownames.
useDatasets	For liger method. Names or indices of datasets to be written to H5 files. Required.

Value

Nothing is returned. H5 file will be created on disk.

See Also

10X cellranger H5 matrix detail

Examples

```
raw <- rawData(pbmc, "ctrl")
writeH5(raw, tempfile(pattern = "ctrl_", fileext = ".h5"))
```

Index