Package 'rliger'

October 14, 2022

Version 1.0.0

Date 2021-04-18

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, ctb],
Chao Gao [aut, ctb, cre],
Jialin Liu [aut, ctb],
Joshua Sodicoff [aut, ctb],
Velina Kozareva [aut, ctb],
Evan Macosko [aut, ctb],
Paul Hoffman [ctb],
Ilya Korsunsky [ctb],
Robert Lee [ctb]

Maintainer Chao Gao <gchao@umich.edu>

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

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

License GPL-3

Imports Rcpp (>= 0.12.10), plyr, FNN, dplyr, grid, ggrepel, irlba, ica, Rtsne, ggplot2, riverplot, foreach, parallel, doParallel, mclust, stats, psych, RANN, uwot, rlang, utils, hdf5r

biocViews

LinkingTo Rcpp, RcppArmadillo, RcppEigen, RcppProgress

Depends cowplot, Matrix, methods, patchwork

RoxygenNote 7.1.1

Encoding UTF-8

Suggests Seurat, knitr, reticulate, rmarkdown, testthat, GenomicRanges, S4Vectors, IRanges, org.Hs.eg.db, reactome.db, fgsea, AnnotationDbi

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-04-19 22:00:12 UTC

R topics documented:

`calcAgreement` *Calculate agreement metric*

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 k, the greater the agreement in general. There are several options for dimensionality reduction, with the default being 'NMF' as it is expected to be most similar to iNMF. Although agreement can theoretically approach 1, in practice it is usually no higher than 0.2-0.3 (particularly for non-deterministic approaches like NMF).

Usage

```
calcAgreement(
  object,
  dr.method = "NMF",
  ndims = 40,
```

```
  k = 15,
  use.aligned = TRUE,
  rand.seed = 42,
  by.dataset = FALSE
)
```

Arguments

object	liger object. Should call quantile_norm before calling.
dr.method	Dimensionality reduction method to use for assessing pre-alignment geometry (either "PCA", "NMF", or "ICA"). (default "NMF")
ndims	Number of dimensions to use in dimensionality reduction (recommended to use the same as number of factors) (default 40).
k	Number of nearest neighbors to use in calculating Jaccard index (default 15).
use.aligned	Whether to use quantile aligned or unaligned cell factor loadings (default TRUE).
rand.seed	Random seed for reproducibility (default 42).
by.dataset	Return agreement calculated for each dataset (default FALSE).

Value

Agreement metric (or vector of agreement per dataset).

Examples

```
## Not run:
# ligerex (liger object based on in-memory datasets), factorization complete
# generate H.norm by quantile normalizig factor loadings
ligerex <- quantile_norm(ligerex)
agreement <- calcAgreement(ligerex, dr.method = "NMF")
# ligerex (liger object based on datasets in HDF5 format), factorization complete
ligerex <- quantile_norm(ligerex)
ligerex <- readSubset(ligerex, slot.use = "scale.data", max.cells = 5000)
agreement <- calcAgreement(ligerex, dr.method = "NMF")

## End(Not run)
```

---

calcAlignment *Calculate alignment metric*

Description

This metric quantifies how well-aligned two or more datasets are. Alignment is defined as in the documentation for Seurat. 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,
  k = NULL,
  rand.seed = 1,
  cells.use = NULL,
  cells.comp = NULL,
  clusters.use = NULL,
  by.cell = FALSE,
  by.dataset = FALSE
)
```

Arguments

object	liger object. Should call quantile_norm before calling.
k	Number of nearest neighbors to use in calculating alignment. By default, this will be floor(0.01 * total number of cells), with a lower bound of 10 in all cases except where the total number of sampled cells is less than 10.
rand.seed	Random seed for reproducibility (default 1).
cells.use	Vector of cells across all datasets to use in calculating alignment
cells.comp	Vector of cells across all datasets to compare to cells.use when calculating alignment (instead of dataset designations). These can be from the same dataset as cells.use. (default NULL)
clusters.use	Names of clusters to use in calculating alignment (default NULL).
by.cell	Return alignment calculated individually for each cell (default FALSE).
by.dataset	Return alignment calculated for each dataset (default FALSE).

Value

Alignment metric.

Examples

```
## Not run:
# ligerex (liger object ), factorization complete
ligerex <- quantile_norm(ligerex)
alignment <- calcAlignment(ligerex)

## End(Not run)
```

calcAlignmentPerCluster *Calculate alignment for each cluster*

Description

Returns alignment for each cluster in analysiss (see documentation for calcAlignment).

Usage

```
calcAlignmentPerCluster(object, rand.seed = 1, k = NULL, by.dataset = FALSE)
```

Arguments

object	liger object. Should call quantileAlignSNF before calling.
rand.seed	Random seed for reproducibility (default 1).
k	Number of nearest neighbors in calculating alignment (see calcAlignment for default). Can pass in single value or vector with same length as number of clusters.
by.dataset	Return alignment calculated for each dataset in cluster (default FALSE).

Value

Vector of alignment statistics (with names of clusters).

Examples

```
## Not run:
# ligerex (liger object), factorization complete
ligerex <- quantile_norm(ligerex)
# get alignment for each cluster
alignment_per_cluster <- calcAlignmentPerCluster(ligerex)

## End(Not run)
```

calcARI *Calculate adjusted Rand index*

Description

Computes adjusted Rand index for liger clustering and external clustering. The Rand index ranges from 0 to 1, with 0 indicating no agreement between clusterings and 1 indicating perfect agreement.

Usage

```
calcARI(object, clusters.compare, verbose = TRUE)
```

Arguments

object	liger object. Should run quantileAlignSNF before calling.
clusters.compare	Clustering with which to compare (named vector).
verbose	Print messages (TRUE by default)

Value

Adjusted Rand index value.

Examples

```
## Not run:
# ligerex (liger object), factorization complete
ligerex <- quantile_norm(ligerex)
# toy clusters
cluster1 <- sample(c('type1', 'type2', 'type3'), ncol(ligerex@raw.data[[1]]), replace = TRUE)
names(cluster1) <- colnames(ligerex@raw.data[[1]])
cluster2 <- sample(c('type4', 'type5', 'type6'), ncol(ligerex@raw.data[[2]]), replace = TRUE)
names(cluster2) <- colnames(ligerex@raw.data[[2]])
# get ARI for first clustering
ari1 <- calcARI(ligerex, cluster1)
# get ARI for second clustering
ari2 <- calcARI(ligerex, cluster2)

## End(Not run)
```

---

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 = NULL,
  dataset2 = NULL,
  do.plot = TRUE
)
```

Arguments

object	liger object. Should run optimizeALS before calling.
dataset1	Name of first dataset (by default takes first two datasets for dataset1 and 2)
dataset2	Name of second dataset
do.plot	Display barplot of dataset specificity scores (by factor) (default TRUE).

Value

List containing three elements. First two elements are the norm of each metagene factor for each dataset. Last element is the vector of dataset specificity scores.

Examples

```
## Not run:
# ligerex (liger object), factorization complete
# generate H.norm by quantile normalizig factor loadings
ligerex <- quantile_norm(ligerex)
dataset_spec <- calcDatasetSpecificity(ligerex, do.plot = F)

## End(Not run)
```

---

calcGeneVars *Calculate variance of gene expression across cells in an online fashion*

Description

This function calculates the variance of gene expression values across cells for hdf5 files.

Usage

```
calcGeneVars(object, chunk = 1000, verbose = TRUE)
```

Arguments

object	liger object. The input raw.data should be a list of hdf5 files. Should call normalize and selectGenes before calling.
chunk	size of chunks in hdf5 file. (default 1000)
verbose	Print progress bar/messages (TRUE by default)

Value

liger object with scale.data slot set.

calcPurity *Calculate purity*

Description

Calculates purity for `liger` clustering and external clustering (true clusters/classes). Purity can sometimes be a more useful metric when the clustering to be tested contains more subgroups or clusters than the true clusters (or classes). Purity also ranges from 0 to 1, with a score of 1 representing a pure, or accurate, clustering.

Usage

```
calcPurity(object, classes.compare, verbose = TRUE)
```

Arguments

`object`	`liger` object. Should run quantileAlignSNF before calling.
`classes.compare`	Clustering with which to compare (named vector).
`verbose`	Print messages (TRUE by default)

Value

Purity value.

Examples

```
## Not run:
# ligerex (liger object), factorization complete
ligerex <- quantile_norm(ligerex)
# toy clusters
cluster1 <- sample(c('type1', 'type2', 'type3'), ncol(ligerex@raw.data[[1]]), replace = TRUE)
names(cluster1) <- colnames(ligerex@raw.data[[1]])
cluster2 <- sample(c('type4', 'type5', 'type6'), ncol(ligerex@raw.data[[2]]), replace = TRUE)
names(cluster2) <- colnames(ligerex@raw.data[[2]])
# get ARI for first clustering
ari1 <- calcPurity(ligerex, cluster1)
# get ARI for second clustering
ari2 <- calcPurity(ligerex, cluster2)

## End(Not run)
```

convertOldLiger *Convert older liger object into most current version (based on class definition)*

Description

Also works for Analogizer objects (but must have both liger and Analogizer loaded). Transfers data in slots with same names from old class object to new, leaving slots defined only in new class NULL.

Usage

```
convertOldLiger(object, override.raw = FALSE, verbose = TRUE)
```

Arguments

object	liger object.
override.raw	Keep original raw.data without any modifications (removing missing cells etc.) (defualt FALSE).
verbose	Print progress bar/messages (TRUE by default)

Value

Updated liger object.

Examples

```
## Not run:
# analogy (old Analogizer object)
# convert to latest class definition
ligerex <- convertOldLiger(analogy)

## End(Not run)
```

createLiger *Create a liger object.*

Description

This function initializes a liger object with the raw data passed in. It requires a list of expression (or another single-cell modality) matrices (gene by cell) for at least two datasets. By default, it converts all passed data into sparse matrices (dgCMatrix) to reduce object size. It initializes cell.data with nUMI and nGene calculated for every cell.

Usage

```
createLiger(
  raw.data,
  take.gene.union = FALSE,
  remove.missing = TRUE,
  format.type = "10X",
  data.name = NULL,
  indices.name = NULL,
  indptr.name = NULL,
  genes.name = NULL,
  barcodes.name = NULL,
  verbose = TRUE
)
```

Arguments

raw.data	List of expression matrices (gene by cell). Should be named by dataset.
take.gene.union	Whether to fill out raw.data matrices with union of genes across all datasets (filling in 0 for missing data) (requires make.sparse = TRUE) (default FALSE).
remove.missing	Whether to remove cells not expressing any measured genes, and genes not expressed in any cells (if take.gene.union = TRUE, removes only genes not expressed in any dataset) (default TRUE).
format.type	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.
verbose	Print messages (TRUE by default)

Value

liger object with raw.data slot set.

Examples

```
# Demonstration using matrices with randomly generated numbers
Y <- matrix(runif(5000,0,2), 10,500)
Z <- matrix(runif(5000,0,2), 10,500)
ligerex <- createLiger(list(y_set = Y, z_set = Z))
```

`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 = NULL,
  dataset2 = NULL,
  factor.share.thresh = 10,
  dataset.specificity = NULL,
  log.fc.thresh = 1,
  pval.thresh = 0.05,
  num.genes = 30,
  print.genes = FALSE,
  verbose = TRUE
)
```

Arguments

`object`	`liger` object. Should call optimizeALS before calling.
`dataset1`	Name of first dataset (default first dataset by order)
`dataset2`	Name of second dataset (default second dataset by order)
`factor.share.thresh`	Use only factors with a dataset specificity less than or equalt to threshold (default 10).
`dataset.specificity`	Pre-calculated dataset specificity if available. Will calculate if not available.
`log.fc.thresh`	Lower log-fold change threshold for differential expression in markers (default 1).
`pval.thresh`	Upper p-value threshold for Wilcoxon rank test for gene expression (default 0.05).
`num.genes`	Max number of genes to report for each dataset (default 30).
`print.genes`	Print ordered markers passing logfc, umi and frac thresholds (default FALSE).
`verbose`	Print messages (TRUE by default)

Value

List of shared and specific factors. First three elements are dataframes of dataset1- specific, shared, and dataset2-specific markers. Last two elements are tables indicating the number of factors in which marker appears.

Examples

```
## Not run:
# ligerex (liger object), factorization complete input
markers <- getFactorMarkers(ligerex, num.genes = 10)
# look at shared markers
head(markers[[2]])

## End(Not run)
```

`getGeneValues` *Get gene expression values from list of expression matrices.*

Description

Returns single vector of gene values across all datasets in list provided. Data can be in raw, normalized or scaled form. If matrices are in cell x gene format, set use.cols = TRUE.

Usage

```
getGeneValues(
  list,
  gene,
  use.cols = FALSE,
  methylation.indices = NULL,
  log2scale = FALSE,
  scale.factor = 10000
)
```

Arguments

`list`	List of gene x cell (or cell x gene) matrices
`gene`	Gene for which to return values (if gene is not found in appropriate dimnames will return vector of NA).
`use.cols`	Whether to query columns for desired gene (set to TRUE if matrices are cell x gene) (default FALSE).
`methylation.indices`	Indices of datasets with methylation data (never log2scaled) (default NULL).
`log2scale`	Whether to log2+1 scale (with multiplicative factor) values (default FALSE).
`scale.factor`	Scale factor to use with log2 scaling (default 10000).

Value

Plots to console (1-2 pages per factor)

Examples

```
## Not run:
# liger object with factorization complete
# ligerex
gene_values <- getGeneValues(ligerex@raw.data, 'MALAT1')

## End(Not run)
```

`getProportionMito` *Calculate proportion mitochondrial contribution*

Description

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

Usage

```
getProportionMito(object, use.norm = FALSE)
```

Arguments

`object`	`liger` object.
`use.norm`	Whether to use cell normalized data in calculating contribution (default FALSE).

Value

Named vector containing proportion of mitochondrial contribution for each cell.

Examples

```
## Not run:
# ligerex (liger object), factorization complete
ligerex@cell.data[["percent_mito"]] <- getProportionMito(ligerex)

## End(Not run)
```

imputeKNN *Impute the query cell expression matrix*

Description

Impute query features from a reference dataset using KNN.

Usage

```
imputeKNN(
  object,
  reference,
  queries,
  knn_k = 20,
  weight = TRUE,
  norm = TRUE,
  scale = FALSE,
  verbose = TRUE
)
```

Arguments

object	liger object.
reference	Dataset containing values to impute into query dataset(s).
queries	Dataset to be augmented by imputation. If not specified, will pass in all datasets.
knn_k	The maximum number of nearest neighbors to search. (default 20)
weight	Whether to use KNN distances as weight matrix (default FALSE).
norm	Whether normalize the imputed data with default parameters (default TRUE).
scale	Whether scale but not center the imputed data with default parameters (default TRUE).
verbose	Print progress bar/messages (TRUE by default)

Value

liger object with raw data in raw.data slot replaced by imputed data (genes by cells)

Examples

```
## Not run:
# ligerex (liger object), factorization complete
# impute every dataset other than the reference dataset
ligerex <- imputeKNN(ligerex, reference = "y_set", weight = FALSE)
# impute only z_set dataset
ligerex <- imputeKNN(ligerex, reference = "y_set", queries = list("z_set"), knn_k = 50)

## End(Not run)
```

liger-class *The LIGER Class*

Description

The liger object is created from two or more single cell datasets. To construct a liger object, the user needs to provide at least two expression (or another single-cell modality) matrices. The class provides functions for data preprocessing, integrative analysis, and visualization.

Details

The key slots used in the liger object are described below.

Slots

`raw.data` List of raw data matrices, one per experiment/dataset (genes by cells)

`norm.data` List of normalized matrices (genes by cells)

`scale.data` List of scaled matrices (cells by genes)

`sample.data` List of sampled matrices (gene by cells)

`h5file.info` List of HDF5-related information for each input dataset. Paths to raw data, indices, indptr, barcodes, genes and the pipeline through which the HDF5 file is formated (10X, AnnData, etc), type of sampled data (raw, normalized or scaled).

`cell.data` Dataframe of cell attributes across all datasets (nrows equal to total number cells across all datasets)

`var.genes` Subset of informative genes shared across datasets to be used in matrix factorization

`H` Cell loading factors (one matrix per dataset, dimensions cells by k)

`H.norm` Normalized cell loading factors (cells across all datasets combined into single matrix)

`W` Shared gene loading factors (k by genes)

`V` Dataset-specific gene loading factors (one matrix per dataset, dimensions k by genes)

`A` Matrices used for online learning (XH)

`B` Matrices used for online learning (HTH)

`tsne.coords` Matrix of 2D coordinates obtained from running t-SNE on H.norm or H matrices

`alignment.clusters` Initial joint cluster assignments from shared factor alignment

`clusters` Joint cluster assignments for cells

`snf` List of values associated with shared nearest factor matrix for use in clustering and alignment (out.summary contains edge weight information between cell combinations)

`agg.data` Data aggregated within clusters

`parameters` List of parameters used throughout analysis

`version` Version of package used to create object

`ligerToSeurat` *Create a Seurat object containing the data from a liger object*

Description

Merges raw.data and scale.data of object, and creates Seurat object with these values along with tsne.coords, iNMF factorization, and cluster assignments. Supports Seurat V2 and V3.

Usage

```
ligerToSeurat(
  object,
  nms = names(object@H),
  renormalize = TRUE,
  use.liger.genes = TRUE,
  by.dataset = FALSE
)
```

Arguments

`object`	`liger` object.
`nms`	By default, labels cell names with dataset of origin (this is to account for cells in different datasets which may have same name). Other names can be passed here as vector, must have same length as the number of datasets. (default names(H))
`renormalize`	Whether to log-normalize raw data using Seurat defaults (default TRUE).
`use.liger.genes`	Whether to carry over variable genes (default TRUE).
`by.dataset`	Include dataset of origin in cluster identity in Seurat object (default FALSE).

Details

Stores original dataset identity by default in new object metadata if dataset names are passed in nms. iNMF factorization is stored in dim.reduction object with key "iNMF".

Value

Seurat object with raw.data, scale.data, dr$tsne, dr$inmf, and ident slots set.

Examples

```
## Not run:
# ligerex (liger object based on in-memory datasets ONLY), factorization complete input
s.object <- ligerToSeurat(ligerex)

## End(Not run)
```

`linkGenesAndPeaks` *Linking genes to putative regulatory elements*

Description

Evaluate the relationships between pairs of genes and peaks based on specified distance metric.

Usage

```
linkGenesAndPeaks(
  gene_counts,
  peak_counts,
  genes.list = NULL,
  dist = "spearman",
  alpha = 0.05,
  path_to_coords,
  verbose = TRUE
)
```

Arguments

`gene_counts`	A gene expression matrix (genes by cells) of normalized counts. This matrix has to share the same column names (cell barcodes) as the matrix passed to peak_counts
`peak_counts`	A peak-level matrix (peaks by cells) of normalized accessibility values, such as the one resulting from imputeKNN. This matrix must share the same column names (cell barcodes) as the matrix passed to gene_counts.
`genes.list`	A list of the genes symbols to be tested. If not specified, this function will use all the gene symbols from the matrix passed to gmat by default.
`dist`	This indicates the type of correlation to calculate – one of "spearman" (default), "pearson", or "kendall".
`alpha`	Significance threshold for correlation p-value. Peak-gene correlations with p-values below this threshold are considered significant. The default is 0.05.
`path_to_coords`	Path to the gene coordinates file.
`verbose`	Print messages (TRUE by default)

Value

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

Examples

```
## Not run:
# some gene counts matrix: gmat.small
# some peak counts matrix: pmat.small
regnet <- linkGenesAndPeaks(gmat.small, pmat.small, dist = "spearman",
alpha = 0.05, path_to_coords = 'some_path')

## End(Not run)
```

---

louvainCluster *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.

Usage

```
louvainCluster(
  object,
  resolution = 1,
  k = 20,
  prune = 1/15,
  eps = 0.1,
  nRandomStarts = 10,
  nIterations = 100,
  random.seed = 1,
  verbose = TRUE
)
```

Arguments

object	liger object. Should run quantile_norm before calling.
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)
k	The maximum number of nearest neighbours to compute. (default 20)
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)

Value

liger object with refined 'clusters' slot set.

Examples

```
## Not run:
# ligerex (liger object), factorization complete
ligerex <- louvainCluster(ligerex, resulotion = 0.3)

## End(Not run)
```

---

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 *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

```
makeInteractTrack(corr.mat, genes.list, output_path, path_to_coords)
```

Arguments

corr.mat	A sparse matrix with peak names as rows and gene symbols as columns.
genes.list	A list of the genes symbols to be tested. If not specified, this function will use all the gene symbols from the matrix passed to gmat by default.
output_path	Path in which the output file will be stored.
path_to_coords	Path to the gene coordinates file.

Value

An Interact Track file stored in the specified path.

Examples

```
## Not run:
# some gene-peak correlation matrix: regent
makeInteractTrack(regnet, path_to_coords = 'some_path_to_gene_coordinates/hg19_genes.bed')

## End(Not run)
```

makeRiverplot *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.

Usage

```
makeRiverplot(
  object,
  cluster1,
  cluster2,
  cluster_consensus = NULL,
  min.frac = 0.05,
  min.cells = 10,
  river.yscale = 1,
  river.lty = 0,
  river.node_margin = 0.1,
  label.cex = 1,
  label.col = "black",
  lab.srt = 0,
  river.usr = NULL,
  node.order = "auto"
)
```

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

A riverplot object

Examples

```
## Not run:
# ligerex (liger object), factorization complete input
# toy clusters
cluster1 <- sample(c('type1', 'type2', 'type3'), ncol(ligerex@raw.data[[1]]), replace = TRUE)
names(cluster1) <- colnames(ligerex@raw.data[[1]])
cluster2 <- sample(c('type4', 'type5', 'type6'), ncol(ligerex@raw.data[[2]]), replace = TRUE)
names(cluster2) <- colnames(ligerex@raw.data[[2]])
# create riverplot
makeRiverplot(ligerex, cluster1, cluster2)

## End(Not run)
```

`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)
```

---

nnzeroGroups *nnzeroGroups*

Description

Utility function to compute number of zeros-per-feature within group

Usage

```
nnzeroGroups(X, y, MARGIN = 2)

## S3 method for class 'dgCMatrix'
nnzeroGroups(X, y, MARGIN = 2)

## S3 method for class 'matrix'
nnzeroGroups(X, y, MARGIN = 2)
```

Arguments

X	matrix
y	group labels
MARGIN	whether observations are rows (=2) or columns (=1)

Value

Matrix of groups by features

nonneg *Perform thresholding on dense matrix*

Description

Perform thresholding on the input dense matrix. Remove any values samller than eps by eps. Helper function for online_iNMF

Usage

```
nonneg(x, eps = 1e-16)
```

Arguments

x	Dense matrix.
eps	Threshold. Should be a small positive value. (default 1e-16)

Value

Dense matrix with smallest values equal to eps.

normalize *Normalize raw datasets to column sums*

Description

This function normalizes data to account for total gene expression across a cell.

Usage

```
normalize(
  object,
  chunk = 1000,
  format.type = "10X",
  remove.missing = TRUE,
  verbose = TRUE
)
```

Arguments

object	liger object.
chunk	size of chunks in hdf5 file. (default 1000)
format.type	string of HDF5 format (10X CellRanger by default).
remove.missing	Whether to remove cells not expressing any measured genes, and genes not expressed in any cells (if take.gene.union = TRUE, removes only genes not expressed in any dataset) (default TRUE).
verbose	Print progress bar/messages (TRUE by default)

Value

liger object with norm.data slot set.

Examples

```
# Demonstration using matrices with randomly generated numbers
Y <- matrix(runif(5000,0,2), 10,500)
Z <- matrix(runif(5000,0,2), 10,500)
ligerex <- createLiger(list(y_set = Y, z_set = Z))
ligerex <- normalize(ligerex)
```

---

online_iNMF *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 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.

Usage

```
online_iNMF(
  object,
  X_new = NULL,
  projection = FALSE,
  W.init = NULL,
  V.init = NULL,
  H.init = NULL,
  A.init = NULL,
  B.init = NULL,
  k = 20,
  lambda = 5,
  max.epochs = 5,
  miniBatch_max_iters = 1,
  miniBatch_size = 5000,
  h5_chunk_size = 1000,
  seed = 123,
  verbose = TRUE
)
```

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.

Examples

```
## Not run:
# Requires preprocessed liger object
# Get factorization using 20 factors and mini-batch of 5000 cells
# (default setting, can be adjusted for ideal results)
ligerex <- online_iNMF(ligerex, k = 20, lambda = 5, miniBatch_size = 5000)

## End(Not run)
```

optimizeALS *Perform iNMF on scaled datasets*

Description

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.

Usage

```
optimizeALS(object, ...)

## S3 method for class 'list'
optimizeALS(
  object,
  k,
  lambda = 5,
  thresh = 1e-06,
  max.iters = 30,
  nrep = 1,
  H.init = NULL,
  W.init = NULL,
  V.init = NULL,
  rand.seed = 1,
  print.obj = FALSE,
  verbose = TRUE,
  ...
)

## S3 method for class 'liger'
optimizeALS(
  object,
  k,
  lambda = 5,
  thresh = 1e-06,
  max.iters = 30,
  nrep = 1,
  H.init = NULL,
  W.init = NULL,
  V.init = NULL,
```

```
  rand.seed = 1,
  print.obj = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

object	liger object. Should normalize, select genes, and scale before calling.
...	Arguments passed to other methods
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)

Value

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

Examples

```
## Not run:
# Requires preprocessed liger object (only for objected not based on HDF5 files)
# Get factorization using 20 factors and mini-batch of 5000 cells
# (default setting, can be adjusted for ideal results)
ligerex <- optimizeALS(ligerex, k = 20, lambda = 5, nrep = 1)

## End(Not run)
```

`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 selected genes (var.genes) are represented in the new datasets.

Usage

```
optimizeNewData(
  object,
  new.data,
  which.datasets,
  add.to.existing = TRUE,
  lambda = NULL,
  thresh = 1e-04,
  max.iters = 100,
  verbose = TRUE
)
```

Arguments

`object`	`liger` object. Should call optimizeALS before calling.
`new.data`	List of raw data matrices (one or more). Each list entry should be named.
`which.datasets`	List of datasets to append new.data to if add.to.existing is true. Otherwise, the most similar existing datasets for each entry in new.data.
`add.to.existing`	Add the new data to existing datasets or treat as totally new datasets (calculate new Vs?) (default TRUE)
`lambda`	Regularization parameter. By default, this will use the lambda last used with optimizeALS.
`thresh`	Convergence threshold. Convergence occurs when |obj0-obj|/(mean(obj0,obj)) < thresh (default 1e-4).
`max.iters`	Maximum number of block coordinate descent iterations to perform (default 100).
`verbose`	Print progress bar/messages (TRUE by default)

Value

`liger` object with H, W, and V slots reset. Raw.data, norm.data, and scale.data will also be updated to include the new data.

Examples

```
## Not run:
# Given preprocessed liger object: ligerex (contains two datasets Y and Z)
# get factorization using three restarts and 20 factors
ligerex <- optimizeALS(ligerex, k = 20, lambda = 5, nrep = 3)
# acquire new data (Y_new, Z_new) from the same cell type, let's add it to existing datasets
new_data <- list(Y_set = Y_new, Z_set = Z_new)
ligerex2 <- optimizeNewData(ligerex, new.data = new_data, which.datasets = list('y_set', 'z_set'))
# acquire new data from different cell type (X), we'll just add another dataset
# it's probably most similar to y_set
ligerex <- optimizeNewData(ligerex, new.data = list(x_set = X), which.datasets = list('y_set'),
                           add.to.existing = FALSE)

## End(Not run)
```

---

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 k smaller than current value, where it is more likely to speed up the factorization.

Usage

```
optimizeNewK(
  object,
  k.new,
  lambda = NULL,
  thresh = 1e-04,
  max.iters = 100,
  rand.seed = 1,
  verbose = TRUE
)
```

Arguments

object	liger object. Should call optimizeALS before calling.
k.new	Inner dimension of factorization (number of factors)
lambda	Regularization parameter. By default, this will use the lambda last used with optimizeALS.
thresh	Convergence threshold. Convergence occurs when |obj0-obj|/(mean(obj0,obj)) < thresh (default 1e-4).
max.iters	Maximum number of block coordinate descent iterations to perform (default 100).
rand.seed	Random seed to set. Only relevant if k.new > k. (default 1)
verbose	Print progress bar/messages (TRUE by default)

Value

liger object with H, W, and V slots reset.

Examples

```
## Not run:
# decide to run with k = 15 instead (keeping old lambda the same)
ligerex <- optimizeNewK(ligerex, k.new = 15)

## End(Not run)
```

---

optimizeNewLambda *Perform factorization for new lambda value*

Description

Uses an efficient strategy for updating that takes advantage of the information in the existing factorization; 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,
  new.lambda,
  thresh = 1e-04,
  max.iters = 100,
  rand.seed = 1,
  verbose = TRUE
)
```

Arguments

object	liger object. Should call optimizeALS before calling.
new.lambda	Regularization parameter. Larger values penalize dataset-specific effects more strongly.
thresh	Convergence threshold. Convergence occurs when |obj0-objl/(mean(obj0,obj)) < thresh
max.iters	Maximum number of block coordinate descent iterations to perform (default 100).
rand.seed	Random seed for reproducibility (default 1).
verbose	Print progress bar/messages (TRUE by default)

Value

liger object with optimized factorization values

Examples

```
## Not run:
# decide to run with lambda = 15 instead (keeping k the same)
ligerex <- optimizeNewLambda(ligerex, new.lambda = 15)

## End(Not run)
```

`optimizeSubset` *Perform factorization for subset of data*

Description

Uses an efficient strategy for updating that takes advantage of the information in the existing factorization. Can use either cell names or cluster names to subset. For more basic subsetting functionality (without automatic optimization), see subsetLiger.

Usage

```
optimizeSubset(
  object,
  cell.subset = NULL,
  cluster.subset = NULL,
  lambda = NULL,
  thresh = 1e-04,
  max.iters = 100,
  datasets.scale = NULL
)
```

Arguments

`object`	`liger` object. Should call optimizeALS before calling.
`cell.subset`	List of cell names to retain from each dataset (same length as number of datasets).
`cluster.subset`	Clusters for which to keep cells (ie. c(1, 5, 6)). Should pass in either cell.subset or cluster.subset but not both.
`lambda`	Regularization parameter. By default, uses last used lambda.
`thresh`	Convergence threshold. Convergence occurs when lobj0-objl/(mean(obj0,obj)) < thresh (default 1e-4).
`max.iters`	Maximum number of block coordinate descent iterations to perform (default 100).
`datasets.scale`	Names of datasets to rescale after subsetting (default NULL).

Value

`liger` object with H, W, and V slots reset. Scale.data (if desired) will also be updated to reflect the subset.

Examples

```
## Not run:
# now want to look at only subset of data
# Requires a vector of cell names from data 1 and a vector of cell names from data 2
ligerex2 <- optimizeSubset(ligerex, cell.subset = list(cell_names_1, cell_names_2))

## End(Not run)
```

`plotByDatasetAndCluster` *Plot t-SNE coordinates of cells across datasets*

Description

Generates two plots of all cells across datasets, one colored by dataset and one colored by cluster. These are useful for visually examining the alignment and cluster distributions, respectively. If clusters have not been set yet (quantileAlignSNF not called), will plot by single color for second plot. It is also possible to pass in another clustering (as long as names match those of cells).

Usage

```
plotByDatasetAndCluster(
  object,
  clusters = NULL,
  title = NULL,
  pt.size = 0.3,
  text.size = 3,
  do.shuffle = TRUE,
  rand.seed = 1,
  axis.labels = NULL,
  do.legend = TRUE,
  legend.size = 5,
  reorder.idents = FALSE,
  new.order = NULL,
  return.plots = FALSE,
  legend.fonts.size = 12
)
```

Arguments

`object`	`liger` object. Should call runTSNE or runUMAP before calling.
`clusters`	Another clustering to use for coloring second plot (must have same names as clusters slot) (default NULL).
`title`	Plot titles (list or vector of length 2) (default NULL).
`pt.size`	Controls size of points representing cells (default 0.3).
`text.size`	Controls size of plot text (cluster center labels) (default 3).

do.shuffle	Randomly shuffle points so that points from same dataset are not plotted one after the other (default TRUE).
rand.seed	Random seed for reproducibility of point shuffling (default 1).
axis.labels	Vector of two strings to use as x and y labels respectively.
do.legend	Display legend on plots (default TRUE).
legend.size	Size of legend on plots (default 5).
reorder.idents	logical whether to reorder the datasets from default order before plotting (default FALSE).
new.order	new dataset factor order for plotting. must set reorder.idents = TRUE.
return.plots	Return ggplot plot objects instead of printing directly (default FALSE).
legend.fonts.size	Controls the font size of the legend.

Value

List of ggplot plot objects (only if return.plots TRUE, otherwise prints plots to console).

Examples

```
## Not run:
# ligerex (liger object), factorization complete
# get tsne.coords for normalized data
ligerex <- runTSNE(ligerex)
# plot to console
plotByDatasetAndCluster(ligerex)
# return list of plots
plots <- plotByDatasetAndCluster(ligerex, return.plots = TRUE)

## End(Not run)
```

---

plotClusterFactors *Plot heatmap of cluster/factor correspondence*

Description

Generates matrix of cluster/factor correspondence, using sum of row-normalized factor loadings for every cell in each cluster. Plots heatmap of matrix, with red representing high total loadings for a factor, black low. Optionally can also include dendrograms and sorting for factors and clusters.

Usage

```
plotClusterFactors(
  object,
  use.aligned = FALSE,
  Rowv = NA,
  Colv = "Rowv",
```

```
  col = NULL,
  return.data = FALSE,
  ...
)
```

Arguments

object	liger object.
use.aligned	Use quantile normalized factor loadings to generate matrix (default FALSE).
Rowv	Determines if and how the row dendrogram should be computed and reordered. Either a dendrogram or a vector of values used to reorder the row dendrogram or NA to suppress any row dendrogram (and reordering) (default NA for no dendrogram).
Colv	Determines if and how the column dendrogram should be reordered. Has the same options as the Rowv argument (default 'Rowv' to match Rowv).
col	Color map to use (defaults to red and black)
return.data	Return matrix of total factor loadings for each cluster (default FALSE).
...	Additional parameters to pass on to heatmap()

Value

If requested, matrix of size num_cluster x num_factor

Examples

```
## Not run:
# ligerex (liger object), factorization complete input
# plot expression for CD4 and return plots
loading.matrix <- plotClusterFactors(ligerex, return.data = TRUE)

## End(Not run)
```

---

plotClusterProportions

Plot cluster proportions by dataset

Description

Generates plot of clusters sized by the proportion of total cells

Usage

```
plotClusterProportions(object, return.plot = FALSE)
```

Arguments

object	liger object. Should call quantileAlignSNF before calling.
return.plot	Return ggplot object (default FALSE)

Value

print plot to console (return.plot = FALSE); ggplot object (return.plot = TRUE) list of ggplot objects.

Examples

```
## Not run:
# ligerex (liger object), factorization complete input
ligerex <- quantile_norm(ligerex)
# plot cluster proportions
plotClusterProportions(ligerex)

## End(Not run)
```

---

plotFactors *Plot scatter plots of unaligned and aligned factor loadings*

Description

Generates scatter plots of factor loadings vs cells for both unaligned and aligned (normalized) factor loadings. This allows for easier visualization of the changes made to the factor loadings during the alignment step. Lists a subset of highly loading genes for each factor. Also provides an option to plot t-SNE coordinates of the cells colored by aligned factor loadings.

It is recommended to call this function into a PDF due to the large number of plots produced.

Usage

```
plotFactors(
  object,
  num.genes = 10,
  cells.highlight = NULL,
  plot.tsne = FALSE,
  verbose = TRUE
)
```

Arguments

object	liger object. Should call quantileAlignSNF before calling.
num.genes	Number of genes to display for each factor (default 10).
cells.highlight	Names of specific cells to highlight in plot (black) (default NULL).
plot.tsne	Plot t-SNE coordinates for each factor (default FALSE).
verbose	Print messages (TRUE by default)

Value

Plots to console (1-2 pages per factor)

Examples

```
## Not run:
# ligerex (liger object), factorization complete
ligerex <- quantile_norm(ligerex)
# get tsne.coords for normalized data
ligerex <- runTSNE(ligerex)
# factor plots into pdf file
# pdf("plot_factors.pdf")
plotFactors(ligerex)
# dev.off()

## End(Not run)
```

---

plotFeature *Plot specific feature on t-SNE coordinates*

Description

Generates one plot for each dataset, colored by chosen feature (column) from cell.data slot. Feature can be categorical (factor) or continuous. Can also plot all datasets combined with by.dataset = FALSE.

Usage

```
plotFeature(
  object,
  feature,
  by.dataset = TRUE,
  discrete = NULL,
  title = NULL,
  pt.size = 0.3,
  text.size = 3,
  do.shuffle = TRUE,
  rand.seed = 1,
  do.labels = FALSE,
  axis.labels = NULL,
  do.legend = TRUE,
  legend.size = 5,
  option = "plasma",
  cols.use = NULL,
  zero.color = "#F5F5F5",
  return.plots = FALSE
)
```

Arguments

object	liger object. Should call runTSNE or runUMAP before calling.
feature	Feature to plot (should be column from cell.data slot).
by.dataset	Whether to generate separate plot for each dataset (default TRUE).
discrete	Whether to treat feature as discrete; if left NULL will infer from column class in cell.data (if factor, treated like discrete) (default NULL).
title	Plot title (default NULL).
pt.size	Controls size of points representing cells (default 0.3).
text.size	Controls size of plot text (cluster center labels) (default 3).
do.shuffle	Randomly shuffle points so that points from same dataset are not plotted one after the other (default TRUE).
rand.seed	Random seed for reproducibility of point shuffling (default 1).
do.labels	Print centroid labels for categorical features (default FALSE).
axis.labels	Vector of two strings to use as x and y labels respectively.
do.legend	Display legend on plots (default TRUE).
legend.size	Size of legend spots for discrete data (default 5).
option	Colormap option to use for ggplot2's scale_color_viridis (default 'plasma').
cols.use	Vector of colors to form gradient over instead of viridis colormap (low to high). Only applies to continuous features (default NULL).
zero.color	Color to use for zero values (no expression) (default '#F5F5F5').
return.plots	Return ggplot plot objects instead of printing directly (default FALSE).

Value

List of ggplot plot objects (only if return.plots TRUE, otherwise prints plots to console).

Examples

```
## Not run:
# ligerex (liger object), factorization complete
# get tsne.coords for normalized data
ligerex <- runTSNE(ligerex)
# plot nUMI to console
plotFeature(ligerex, feature = 'nUMI')

## End(Not run)
```

plotGene *Plot gene expression on dimensional reduction (t-SNE) coordinates*

Description

Generates plot of dimensional reduction coordinates (default t-SNE) colored by expression of specified gene. Data can be scaled by dataset or selected feature column from cell.data (or across all cells). Data plots can be split by feature.

Usage

```
plotGene(
  object,
  gene,
  use.raw = FALSE,
  use.scaled = FALSE,
  scale.by = "dataset",
  log2scale = NULL,
  methylation.indices = NULL,
  plot.by = "dataset",
  set.dr.lims = FALSE,
  pt.size = 0.1,
  min.clip = NULL,
  max.clip = NULL,
  clip.absolute = FALSE,
  points.only = FALSE,
  option = "plasma",
  cols.use = NULL,
  zero.color = "#F5F5F5",
  axis.labels = NULL,
  do.legend = TRUE,
  return.plots = FALSE,
  keep.scale = FALSE
)
```

Arguments

object	liger object. Should call runTSNE before calling.
gene	Gene for which to plot expression.
use.raw	Plot raw UMI values instead of normalized, log-transformed data (default FALSE).
use.scaled	Plot values scaled across specified groups of cells (with log transformation) (default FALSE).
scale.by	Grouping of cells by which to scale gene (can be any factor column in cell.data or 'none' for scaling across all cells) (default 'dataset').

log2scale — Whether to show log2 transformed values or original normalized, raw, or scaled values (as stored in object). Default value is FALSE if use.raw = TRUE, otherwise TRUE.

methylation.indices — Indices of datasets in object with methylation data (this data is not log transformed and must use normalized values). (default NULL)

plot.by — How to group cells for plotting (can be any factor column in cell.data or 'none' for plotting all cells in a single plot). Note that this can result in large number of plots. Users are encouraged to use same value as for scale.by (default 'dataset').

set.dr.lims — Whether to keep dimensional reduction coordinates consistent when multiple plots created (default FALSE).

pt.size — Point size for plots (default 0.1).

min.clip — Minimum value for expression values plotted. Can pass in quantile (0-1) or absolute cutoff (set clip.absolute = TRUE). Can also pass in vector if expecting multiple plots; users are encouraged to pass in named vector (from levels of desired feature) to avoid mismatches in order (default NULL).

max.clip — Maximum value for expression values plotted. Can pass in quantile (0-1) or absolute cutoff (set clip.absolute = TRUE). Can also pass in vector if expecting multiple plots; users are encouraged to pass in named vector (from levels of desired feature) to avoid mismatches in order (default NULL).

clip.absolute — Whether to treat clip values as absolute cutoffs instead of quantiles (default FALSE).

points.only — Remove axes, background, and legend when plotting coordinates (default FALSE).

option — Colormap option to use for ggplot2's scale_color_viridis (default 'plasma').

cols.use — Vector of colors to form gradient over instead of viridis colormap (low to high). (default NULL).

zero.color — Color to use for zero values (no expression) (default '#F5F5F5').

axis.labels — Vector of two strings to use as x and y labels respectively. (default NULL)

do.legend — Display legend on plots (default TRUE).

return.plots — Return ggplot objects instead of printing directly (default FALSE).

keep.scale — Maintain min/max color scale across all plots when using plot.by (default FALSE)

Value

If returning single plot, returns ggplot object; if returning multiple plots; returns list of ggplot objects.

Examples

```
## Not run:
# ligerex (liger object based on in-memory datasets), factorization complete
ligerex
ligerex <- runTSNE(ligerex)
# plot expression for CD4 and return plots
gene_plots <- plotGene(ligerex, "CD4", return.plots = TRUE)
```

```
# ligerex (liger object based on datasets in HDF5 format), factorization complete input
ligerex <- readSubset(ligerex, slot.use = "norm.data", max.cells = 5000)
gene_plots <- plotGene(ligerex, "CD4", return.plots = TRUE)

## End(Not run)
```

`plotGeneLoadings` *Generate t-SNE plots and gene loading plots*

Description

Plots t-SNE coordinates of all cells by their loadings on each factor. Underneath it displays the most highly loading shared and dataset-specific genes, along with the overall gene loadings for each dataset.

It is recommended to call this function into a PDF due to the large number of plots produced.

Usage

```
plotGeneLoadings(
  object,
  dataset1 = NULL,
  dataset2 = NULL,
  num.genes.show = 12,
  num.genes = 30,
  mark.top.genes = TRUE,
  factor.share.thresh = 10,
  log.fc.thresh = 1,
  umi.thresh = 30,
  frac.thresh = 0,
  pval.thresh = 0.05,
  do.spec.plot = TRUE,
  max.val = 0.1,
  pt.size = 0.1,
  option = "plasma",
  zero.color = "#F5F5F5",
  return.plots = FALSE,
  axis.labels = NULL,
  do.title = FALSE,
  verbose = TRUE
)
```

Arguments

`object`	`liger` object. Should call runTSNE before calling.
`dataset1`	Name of first dataset (by default takes first two datasets for dataset1 and 2)
`dataset2`	Name of second dataset

num.genes.show	Number of genes displayed as y-axis labels in the gene loading plots at the bottom (default 12)
num.genes	Number of genes to show in word clouds (default 30).
mark.top.genes	Plot points corresponding to top loading genes in different color (default TRUE).
factor.share.thresh	Use only factors with a dataset specificity less than or equal to threshold (default 10).
log.fc.thresh	Lower log-fold change threshold for differential expression in markers (default 1).
umi.thresh	Lower UMI threshold for markers (default 30).
frac.thresh	Lower threshold for fraction of cells expressing marker (default 0).
pval.thresh	Upper p-value threshold for Wilcoxon rank test for gene expression (default 0.05).
do.spec.plot	Include dataset specificity plot in printout (default TRUE).
max.val	Value between 0 and 1 at which color gradient should saturate to max color. Set to NULL to revert to default gradient scaling. (default 0.1)
pt.size	Point size for plots (default 0.1).
option	Colormap option to use for ggplot2's scale_color_viridis (default 'plasma').
zero.color	Color to use for zero values (no expression) (default '#F5F5F5').
return.plots	Return ggplot objects instead of printing directly (default FALSE).
axis.labels	Vector of two strings to use as x and y labels respectively (default NULL).
do.title	Include top title with cluster and Dataset Specificity (default FALSE).
verbose	Print progress bar/messages (TRUE by default)

Value

List of ggplot plot objects (only if return.plots TRUE, otherwise prints plots to console).

Examples

```
## Not run:
# ligerex (liger object based on in-memory datasets), factorization complete
ligerex <- quantile_norm(ligerex)
ligerex <- runUMAP(ligerex)
# pdf("gene_loadings.pdf")
plotGeneLoadings(ligerex, num.genes = 20)
# dev.off()
# ligerex (liger object based on datasets in HDF5 format), factorization complete input
ligerex <- readSubset(ligerex, slot.use = "norm.data", max.cells = 5000)
plotGeneLoadings(ligerex, num.genes = 20)

## End(Not run)
```

plotGenes *Plot expression of multiple genes*

Description

Uses plotGene to plot each gene (and dataset) on a separate page. It is recommended to call this function into a PDF due to the large number of plots produced.

Usage

```
plotGenes(object, genes, ...)
```

Arguments

object	liger object. Should call runTSNE before calling.
genes	Vector of gene names.
...	arguments passed from plotGene

Value

If returning single plot, returns ggplot object; if returning multiple plots; returns list of ggplot objects.

Examples

```
## Not run:
# ligerex (liger object), factorization complete input
ligerex <- runTSNE(ligerex)
# plot expression for CD4 and FCGR3A
# pdf("gene_plots.pdf")
plotGenes(ligerex, c("CD4", "FCGR3A"))
# dev.off()

## End(Not run)
```

plotGeneViolin *Plot violin plots for gene expression*

Description

Generates violin plots of expression of specified gene for each dataset.

Usage

```
plotGeneViolin(
  object,
  gene,
  methylation.indices = NULL,
  by.dataset = TRUE,
  return.plots = FALSE
)
```

Arguments

object	liger object.
gene	Gene for which to plot relative expression.
methylation.indices	Indices of datasets in object with methylation data (this data is not magnified and put on log scale).
by.dataset	Plots gene expression for each dataset separately (default TRUE).
return.plots	Return ggplot objects instead of printing directly to console (default FALSE).

Value

List of ggplot plot objects (only if return.plots TRUE, otherwise prints plots to console).

Examples

```
## Not run:
# ligerex (liger object based on in-memory datasets), factorization complete
# plot expression for CD4 and return plots
violin_plots <- plotGeneViolin(ligerex, "CD4", return.plots = TRUE)
# ligerex (liger object based on datasets in HDF5 format), factorization complete input
ligerex <- readSubset(ligerex, slot.use = "norm.data", max.cells = 5000)
violin_plots <- plotGeneViolin(ligerex, "CD4", return.plots = TRUE)

## End(Not run)
```

---

plotWordClouds *Generate word clouds and t-SNE plots*

Description

Plots t-SNE coordinates of all cells by their loadings on each factor. Underneath it displays the most highly loading shared and dataset-specific genes, with the size of the marker indicating the magnitude of the loading.

It is recommended to call this function into a PDF due to the large number of plots produced.

Usage

```
plotWordClouds(
  object,
  dataset1 = NULL,
  dataset2 = NULL,
  num.genes = 30,
  min.size = 1,
  max.size = 4,
  factor.share.thresh = 10,
  log.fc.thresh = 1,
  pval.thresh = 0.05,
  do.spec.plot = TRUE,
  return.plots = FALSE,
  verbose = TRUE
)
```

Arguments

object	liger object. Should call runTSNE before calling.
dataset1	Name of first dataset (by default takes first two datasets for dataset1 and 2)
dataset2	Name of second dataset
num.genes	Number of genes to show in word clouds (default 30).
min.size	Size of smallest gene symbol in word cloud (default 1).
max.size	Size of largest gene symbol in word cloud (default 4).
factor.share.thresh	Use only factors with a dataset specificity less than or equalt to threshold (default 10).
log.fc.thresh	Lower log-fold change threshold for differential expression in markers (default 1).
pval.thresh	Upper p-value threshold for Wilcoxon rank test for gene expression (default 0.05).
do.spec.plot	Include dataset specificity plot in printout (default TRUE).
return.plots	Return ggplot objects instead of printing directly (default FALSE).
verbose	Print progress bar/messages (TRUE by default)

Value

List of ggplot plot objects (only if return.plots TRUE, otherwise prints plots to console).

Examples

```
## Not run:
# ligerex (liger object based on in-memory datasets), factorization complete
ligerex <- quantile_norm(ligerex)
ligerex <- runTSNE(ligerex)
# pdf('word_clouds.pdf')
```

```
plotWordClouds(ligerex, num.genes = 20)
# dev.off()
# ligerex (liger object based on datasets in HDF5 format), factorization complete input
ligerex <- readSubset(ligerex, slot.use = "norm.data", max.cells = 5000)
plotWordClouds(ligerex, num.genes = 20)

## End(Not run)
```

---

quantileAlignSNF *Quantile align (normalize) factor loadings*

Description

This is a deprecated function. Calling 'quantile_norm' 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)
```

---

quantile_norm *Quantile align (normalize) factor loadings*

Description

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

Usage

```
quantile_norm(object, ...)

## S3 method for class 'list'
quantile_norm(
  object,
  quantiles = 50,
  ref_dataset = NULL,
  min_cells = 20,
  knn_k = 20,
  dims.use = NULL,
  do.center = FALSE,
  max_sample = 1000,
  eps = 0.9,
  refine.knn = TRUE,
  rand.seed = 1,
  ...
)

## S3 method for class 'liger'
quantile_norm(
  object,
  quantiles = 50,
  ref_dataset = NULL,
  min_cells = 20,
  knn_k = 20,
  dims.use = NULL,
  do.center = FALSE,
  max_sample = 1000,
  eps = 0.9,
```

```
  refine.knn = TRUE,
  rand.seed = 1,
  ...
)
```

Arguments

object	liger object. Should run optimizeALS before calling.
...	Arguments passed to other methods
quantiles	Number of quantiles to use for quantile normalization (default 50).
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)
knn_k	Number of nearest neighbors for within-dataset knn graph (default 20).
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)
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.
refine.knn	whether to increase robustness of cluster assignments using KNN graph.(default TRUE)
rand.seed	Random seed to allow reproducible results (default 1)

Details

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.

Value

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

Examples

```
## Not run:
# ligerex (liger object), factorization complete
# do basic quantile alignment
ligerex <- quantile_norm(ligerex)
```

```
# higher resolution for more clusters (note that SNF is conserved)
ligerex <- quantile_norm(ligerex, resolution = 1.2)
# change knn_k for more fine-grained local clustering
ligerex <- quantile_norm(ligerex, knn_k = 15, resolution = 1.2)

## End(Not run)
```

`rank_matrix` *rank_matrix*

Description

Utility function to rank columns of matrix

Usage

```
rank_matrix(X)

## S3 method for class 'dgCMatrix'
rank_matrix(X)

## S3 method for class 'matrix'
rank_matrix(X)
```

Arguments

`X`	feature by observation matrix.

Value

List with 2 items

`read10X` *Read 10X alignment data (including V3)*

Description

This function generates a sparse matrix (genes x cells) from the data generated by 10X's cellranger count pipeline. It can process V2 and V3 data together, producing either a single merged matrix or list of matrices. Also handles multiple data types produced by 10X V3 (Gene Expression, Antibody Capture, CRISPR, CUSTOM).

Usage

```
read10X(
  sample.dirs,
  sample.names,
  merge = TRUE,
  num.cells = NULL,
  min.umis = 0,
  use.filtered = FALSE,
  reference = NULL,
  data.type = "rna",
  verbose = TRUE
)
```

Arguments

sample.dirs	List of directories containing either matrix.mtx(.gz) file along with genes.tsv, (features.tsv), and barcodes.tsv, or outer level 10X output directory (containing outs directory).
sample.names	Vector of names to use for samples (corresponding to sample.dirs)
merge	Whether to merge all matrices of the same data type across samples or leave as list of matrices (default TRUE).
num.cells	Optional limit on number of cells returned for each sample (only for Gene Expression data). Retains the cells with the highest numbers of transcripts (default NULL).
min.umis	Minimum UMI threshold for cells (default 0).
use.filtered	Whether to use 10X's filtered data (as opposed to raw). Only relevant for sample.dirs containing 10X outs directory (default FALSE).
reference	For 10X V<3, specify which reference directory to use if sample.dir is outer level 10X directory (only necessary if more than one reference used for sequencing). (default NULL)
data.type	Indicates the protocol of the input data. If not specified, input data will be considered scRNA-seq data (default 'rna', alternatives: 'atac').
verbose	Print messages (TRUE by default)

Value

List of merged matrices across data types (returns sparse matrix if only one data type detected), or nested list of matrices organized by sample if merge=F.

Examples

```
## Not run:
# 10X output directory V2 -- contains outs/raw_gene_bc_matrices/<reference>/...
sample.dir1 <- "path/to/outer/dir1"
# 10X output directory V3 -- for two data types, Gene Expression and CUSTOM
sample.dir2 <- "path/to/outer/dir2"
dges1 <- read10X(list(sample.dir1, sample.dir2), c("sample1", "sample2"), min.umis = 50)
```

```
ligerex <- createLiger(expr = dges1[["Gene Expression"]], custom = dges1[["CUSTOM"]])

## End(Not run)
```

`readSubset` *Sample data for plotting*

Description

This function samples raw/normalized/scaled data from on-disk HDF5 files for plotting. This function assumes that the cell barcodes are unique across all datasets.

Usage

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

Arguments

`object`	`liger` object. Should call normalize and selectGenes before calling.
`slot.use`	Type of data for sampling (raw.data, norm.data(default), scale.data).
`balance`	Type of sampling. NULL means that max_cells are sampled from among all cells; balance="dataset" samples up to max_cells from each dataset; balance="cluster" samples up to max_cells from each cluster.
`max.cells`	Total number of cell to sample (default 5000).
`chunk`	is the max number of cells at a time to read from disk (default 1000).
`datasets.use`	uses only the specified datasets for sampling. Default is NULL (all datasets)
`genes.use`	samples from only the specified genes. Default is NULL (all genes)
`rand.seed`	for reproducibility (default 1).
`verbose`	Print progress bar/messages (TRUE by default)

Value

`liger` object with sample.data slot set.

Examples

```
## Not run:
# Only for online liger object (based on HDF5 files)
# Example: sample a total amount of 5000 cells from norm.data for downstream analysis
ligerex <- readSubset(ligerex, slot.use = "norm.data", max.cells = 5000)

## End(Not run)
```

`removeMissingObs` *Remove cells/genes with no expression across any genes/cells*

Description

Removes cells/genes from chosen slot with no expression in any genes or cells respectively.

Usage

```
removeMissingObs(
  object,
  slot.use = "raw.data",
  use.cols = TRUE,
  verbose = TRUE
)
```

Arguments

`object`	`liger` object (scale.data or norm.data must be set).
`slot.use`	The data slot to filter (takes "raw.data" and "scale.data") (default "raw.data").
`use.cols`	Treat each column as a cell (default TRUE).
`verbose`	Print messages (TRUE by default)

Value

`liger` object with modified raw.data (or chosen slot) (dataset names preserved).

Examples

```
## Not run:
# liger object: ligerex
ligerex <- removeMissingObs(ligerex)

## End(Not run)
```

`reorganizeLiger` *Construct a liger object organized by another feature*

Description

Using the same data, rearrange functional datasets using another discrete feature in cell.data. This removes most computed data slots, though cell.data and current clustering can be retained.

Usage

```
reorganizeLiger(
  object,
  by.feature,
  keep.meta = TRUE,
  new.label = "orig.dataset",
  ...
)
```

Arguments

`object`	`liger` object.
`by.feature`	Column in cell.data to use in reorganizing raw data.
`keep.meta`	Whether to carry over all existing data in cell.data slot (default TRUE).
`new.label`	If cell.data is to be retained, new column name for original organizing feature (previously labeled as dataset) (default "orig.dataset")
`...`	Additional parameters passed on to createLiger.

Value

`liger` object with rearranged raw.data slot.

Examples

```
## Not run:
# ligerex (liger object based on in-memory objects) organized by species
# with column designating sex in cell.data
# rearrange by sex
ligerex_new <- reorganizeLiger(ligerex, by.feature = "sex", new.label = "species")

## End(Not run)
```

restoreOnlineLiger *Restore links (to hdf5 files) for reloaded online Liger object*

Description

When loading the saved online Liger object in a new R session, the links to hdf5 files may be corrupted. This functions enables the restoration of those links so that new analyses can be carried out.

Usage

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

Arguments

object	liger object.
file.path	List of paths to hdf5 files.

Value

liger object with restored links.

Examples

```
## Not run:
# We want to restore the ligerex (liger object based on HDF5 files)
# It has broken connections to HDF5 files
# Call the following function and provide the paths to the correspoinding files
ligerex = restoreOnlineLiger(ligerex, file.path = list("path1/library1.h5", "path2/library2.h5"))

## End(Not run)
```

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,
  gene_sets = c(),
  mat_w = TRUE,
  mat_v = 0,
  custom_gene_sets = c()
)
```

Arguments

object	liger object.
gene_sets	A list of the Reactome gene sets names to be tested. If not specified, this function will use all the gene sets from the Reactome by default
mat_w	This indicates whether to use the shared factor loadings 'W' (default TRUE)
mat_v	This indicates which V matrix to be added to the analysis. It can be a numeric number or a list of the numerics.
custom_gene_sets	A named list of character vectors of entrez gene ids. If not specified, this function will use all the gene symbols from the input matrix by default

Value

A list of matrices with GSEA analysis for each factor

Examples

```
## Not run:
# ligerex (liger object), factorization complete
wilcox.results <- runGSEA(ligerex)
wilcox.results <- runGSEA(ligerex, mat_v = c(1, 2))

## End(Not run)
```

---

runTSNE *Perform t-SNE dimensionality reduction*

Description

Runs t-SNE on the normalized cell factors (or raw cell factors) to generate a 2D embedding for visualization. Has option to run on subset of factors. Note that running multiple times will reset tsne.coords values.

Usage

```
runTSNE(
  object,
  use.raw = FALSE,
  dims.use = 1:ncol(object@H.norm),
  use.pca = FALSE,
  perplexity = 30,
  theta = 0.5,
  method = "Rtsne",
  fitsne.path = NULL,
  rand.seed = 42
)
```

Arguments

object	liger object. Should run quantile_norm before calling with defaults.
use.raw	Whether to use un-aligned cell factor loadings (H matrices) (default FALSE).
dims.use	Factors to use for computing tSNE embedding (default 1:ncol(H.norm)).
use.pca	Whether to perform initial PCA step for Rtsne (default FALSE).
perplexity	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	Supports two methods for estimating tSNE values: Rtsne (Barnes-Hut implementation of t-SNE) and fftRtsne (FFT-accelerated Interpolation-based t-SNE) (using Kluger Lab implementation). (default Rtsne)
fitsne.path	Path to the cloned FIt-SNE directory (ie. '/path/to/dir/FIt-SNE') (required for using fftRtsne – only first time runTSNE is called) (default NULL).
rand.seed	Random seed for reproducibility (default 42).

Details

In order to run fftRtsne (recommended for large datasets), you must first install FIt-SNE as detailed here. Include the path to the cloned FIt-SNE directory as the fitsne.path parameter, though this is only necessary for the first call to runTSNE. For more detailed FIt-SNE installation instructions, see the liger repo README.

Value

liger object with tsne.coords slot set.

Examples

```
## Not run:
# ligerex (liger object), factorization complete
# generate H.norm by quantile normalizig factor loadings
ligerex <- quantile_norm(ligerex)
# get tsne.coords for normalized data
```

```
ligerex <- runTSNE(ligerex)
# get tsne.coords for raw factor loadings
ligerex <- runTSNE(ligerex, use.raw = TRUE)

## End(Not run)
```

---

runUMAP *Perform UMAP dimensionality reduction*

Description

Run UMAP on the normalized cell factors (or raw cell factors) to generate a 2D embedding for visualization (or general dimensionality reduction). Has option to run on subset of factors. Note that running multiple times will overwrite tsne.coords values. It is generally recommended to use this method for dimensionality reduction with extremely large datasets.

Note that this method requires that the package uwot is installed. It does not depend on reticulate or python umap-learn.

Usage

```
runUMAP(
  object,
  use.raw = FALSE,
  dims.use = 1:ncol(object@H.norm),
  k = 2,
  distance = "euclidean",
  n_neighbors = 10,
  min_dist = 0.1,
  rand.seed = 42
)
```

Arguments

object	liger object. Should run quantile_norm before calling with defaults.
use.raw	Whether to use un-aligned cell factor loadings (H matrices) (default FALSE).
dims.use	Factors to use for computing tSNE embedding (default 1:ncol(H.norm)).
k	Number of dimensions to reduce to (default 2).
distance	Mtric used to measure distance in the input space. A wide variety of metrics are already coded, and a user defined function can be passed as long as it has been JITd by numba. (default "euclidean", alternatives: "cosine", "manhattan", "hamming")
n_neighbors	Number of neighboring points used in local approximations of manifold structure. 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. (default 10)

min_dist	Controls how tightly the embedding is allowed compress points together. Larger values ensure embedded points are more evenly distributed, while smaller values allow the algorithm to optimise 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. (default 0.1)
rand.seed	Random seed for reproducibility (default 42).

Value

liger object with tsne.coords slot set.

Examples

```
## Not run:
# ligerex (liger object), factorization complete
# generate H.norm by quantile normalizig factor loadings
ligerex <- quantileAlignSNF(ligerex)
# get tsne.coords for normalized data
ligerex <- runUMAP(ligerex)
# get tsne.coords for raw factor loadings
ligerex <- runUMAP(ligerex, use.raw = TRUE)

## End(Not run)
```

---

runWilcoxon *Perform Wilcoxon rank-sum test*

Description

Perform Wilcoxon rank-sum tests on specified dataset using given method.

Usage

```
runWilcoxon(object, data.use = "all", compare.method)
```

Arguments

object	liger object.
data.use	This selects which dataset(s) to use. (default 'all')
compare.method	This indicates the metric of the test. Either 'clusters' or 'datasets'.

Value

A 10-columns data.frame with test results.

Examples

```
## Not run:
# ligerex (liger object based on in-memory datasets), factorization complete
wilcox.results <- runWilcoxon(ligerex, compare.method = "cluster")
wilcox.results <- runWilcoxon(ligerex, compare.method = "datastes", data.use = c(1, 2))
# HDF5 input
# ligerex (liger object based on datasets in HDF5 format), factorization complete
# Need to sample cells before implementing Wilcoxon test
ligerex <- readSubset(ligerex, slot.use = "norm.data", max.cells = 1000)
de_genes <- runWilcoxon(ligerex, compare.method = "clusters")

## End(Not run)
```

---

scaleNotCenter *Scale genes by root-mean-square across cells*

Description

This function scales normalized gene expression data after variable genes have been selected. Note that the data is not mean-centered before scaling because expression values must remain positive (NMF only accepts positive values). It also removes cells which do not have any expression across the genes selected, by default.

Usage

```
scaleNotCenter(object, remove.missing = TRUE, chunk = 1000, verbose = TRUE)
```

Arguments

object	liger object. Should call normalize and selectGenes before calling.
remove.missing	Whether to remove cells from scale.data with no gene expression (default TRUE).
chunk	size of chunks in hdf5 file. (default 1000)
verbose	Print progress bar/messages (TRUE by default)

Value

liger object with scale.data slot set.

Examples

```
## Not run:
# Given datasets Y and Z
ligerex <- createLiger(list(y_set = Y, z_set = Z))
ligerex <- normalize(ligerex)
# use default selectGenes settings (var.thresh = 0.1)
ligerex <- selectGenes(ligerex)
ligerex <- scaleNotCenter(ligerex)

## End(Not run)
```

`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). It also provides a log plot of gene variance vs gene expression (with a line indicating expected expression across genes and cells). Selected genes are plotted in green.

Usage

```
selectGenes(
  object,
  var.thresh = 0.1,
  alpha.thresh = 0.99,
  num.genes = NULL,
  tol = 1e-04,
  datasets.use = 1:length(object@raw.data),
  combine = "union",
  capitalize = FALSE,
  do.plot = FALSE,
  cex.use = 0.3,
  chunk = 1000
)
```

Arguments

`object`	`liger` object. Should have already called normalize.
`var.thresh`	Variance threshold. Main threshold used to identify variable genes. Genes with expression variance greater than threshold (relative to mean) are selected. (higher threshold -> fewer selected genes). Accepts single value or vector with separate var.thresh for each dataset. (default 0.1)
`alpha.thresh`	Alpha threshold. Controls upper bound for expected mean gene expression (lower threshold -> higher upper bound). (default 0.99)
`num.genes`	Number of genes to find for each dataset. Optimises the value of var.thresh for each dataset to get this number of genes. Accepts single value or vector with same length as number of datasets (optional, default=NULL).
`tol`	Tolerance to use for optimization if num.genes values passed in (default 0.0001).
`datasets.use`	List of datasets to include for discovery of highly variable genes. (default 1:length(object@raw.data))
`combine`	How to combine variable genes across experiments. Either "union" or "intersection". (default "union")

capitalize	Capitalize gene names to match homologous genes (ie. across species) (default FALSE)
do.plot	Display log plot of gene variance vs. gene expression for each dataset. Selected genes are plotted in green. (default FALSE)
cex.use	Point size for plot.
chunk	size of chunks in hdf5 file. (default 1000)

Value

liger object with var.genes slot set.

Examples

```
## Not run:
# Given datasets Y and Z
ligerex <- createLiger(list(y_set = Y, z_set = Z))
ligerex <- normalize(ligerex)
# use default selectGenes settings (var.thresh = 0.1)
ligerex <- selectGenes(ligerex)
# select a smaller subset of genes
ligerex <- selectGenes(ligerex, var.thresh = 0.3)

## End(Not run)
```

---

seuratToLiger *Create liger object from one or more Seurat objects*

Description

This function creates a liger object from multiple (disjoint) Seurat objects or a single (combined-analysis) Seurat object. It includes options for keeping the variable genes and cluster identities from the original Seurat objects. Seurat V2 and V3 supported (though all objects should share the same major version).

Usage

```
seuratToLiger(
  objects,
  combined.seurat = FALSE,
  names = "use-projects",
  meta.var = NULL,
  assays.use = NULL,
  raw.assay = "RNA",
  remove.missing = TRUE,
  renormalize = TRUE,
  use.seurat.genes = TRUE,
  num.hvg.info = NULL,
```

```
  use.idents = TRUE,
  use.tsne = TRUE,
  cca.to.H = FALSE
)
```

Arguments

objects	One or more Seurat v2 objects. If passing multiple objects, should be in list.
combined.seurat	Whether Seurat object (single) already contains multiple datasets (default FALSE).
names	Names to use for datasets in new liger object. If use-projects, takes project names from individual Seurat objects; if use-meta, takes value of object meta.data in meta.var column for each dataset; otherwise, user can pass in vector of names with same length as number of datasets. If combined.seurat, infers project names based on whether meta.var or assays.use is present (at least one required).
meta.var	Seurat meta.data column name to use in naming datasets. Either meta.var or assays.use required if combined.seurat is TRUE (default NULL).
assays.use	Names of Seurat v3 assays to use as separate datasets in conversion (e.g. RNA, ADT) (default NULL).
raw.assay	Name of Seurat v3 assay to use for raw data if meta.var used to split combined Seurat object – in case integrated assay has been set as default (default "RNA").
remove.missing	Whether to remove missing genes/cells when converting raw.data to liger object (default TRUE).
renormalize	Whether to automatically normalize raw.data once `liger` object is created (default TRUE).
use.seurat.genes	Carry over variable genes from Seurat objects. If num.hvg.info is set, uses that value to get top most highly variable genes from hvg.info slot in Seurat objects. Otherwise uses var.genes slot in Seurat objects. For multiple datasets, takes the union of the variable genes. (default TRUE)
num.hvg.info	Number of highly variable genes to include from each object's hvg.info slot. Only available for Seurat v2 objects. If set, recommended value is 2000 (default NULL).
use.idents	Carry over cluster identities from Seurat objects. If multiple objects with overlapping cluster names, will preface cluster names by dataset names to distinguish. (default TRUE).
use.tsne	Carry over t-SNE coordinates from Seurat object (only meaningful for combined analysis Seurat object). Useful for plotting directly afterwards. (default TRUE)
cca.to.H	Carry over CCA (and aligned) loadings and insert them into H (and H.norm) slot in liger object (only meaningful for combined analysis Seurat object). Useful for plotting directly afterwards. (default FALSE)

Value

`liger` object.

Examples

```
## Not run:
# Seurat objects for two pbmc datasets
tenx <- readRDS('tenx.RDS')
seqwell <- readRDS('seqwell.RDS')
# create liger object, using project names
ligerex <- seuratToLiger(list(tenx, seqwell))
# create liger object, passing in names explicitly, using hvg.info genes
ligerex2 <- seuratToLiger(list(tenx, seqwell), names = c('tenx', 'seqwell'), num.hvg.info = 2000)
# Seurat object for joint analysis
pbmc <- readRDS('pbmc.RDS')
# create liger object, using 'protocol' for dataset names
ligerex3 <- seuratToLiger(pbmc, combined.seurat = TRUE, meta.var = 'protocol', num.hvg.info = 2000)

## End(Not run)
```

---

show *show method for liger*

Description

show method for liger

Usage

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

Arguments

object	liger object

---

subsetLiger *Construct a liger object with a specified subset*

Description

The subset can be based on cell names or clusters. This function applies the subsetting to raw.data, norm.data, scale.data, cell.data, H, W, V, H.norm, tsne.coords, and clusters. Note that it does NOT reoptimize the factorization. See optimizeSubset for this functionality.

Usage

```
subsetLiger(
  object,
  clusters.use = NULL,
  cells.use = NULL,
  remove.missing = TRUE
)
```

Arguments

object	liger object. Should run quantileAlignSNF and runTSNE before calling.
clusters.use	Clusters to use for subset.
cells.use	Vector of cell names to keep from any dataset.
remove.missing	Whether to remove genes/cells with no expression when creating new object (default TRUE).

Value

liger object with subsetting applied to raw.data, norm.data, scale.data, H, W, V, H.norm, tsne.coords, and clusters.

Examples

```
## Not run:
# ligerex (liger object based on in-memory datasets), with clusters 0:10
# factorization, alignment, and t-SNE calculation have been performed
# subset by clusters
ligerex_subset <- subsetLiger(ligerex, clusters.use = c(1, 4, 5))

## End(Not run)
```

---

suggestK *Visually suggest appropiate k value*

Description

This can be used to select appropriate value of k for factorization of particular dataset. Plots median (across cells in all datasets) K-L divergence from uniform for cell factor loadings as a function of k. This should increase as k increases but is expected to level off above sufficiently high number of factors (k). This is because cells should have factor loadings which are not uniformly distributed when an appropriate number of factors is reached.

Depending on number of cores used, this process can take 10-20 minutes.

Usage

```
suggestK(
  object,
  k.test = seq(5, 50, 5),
  lambda = 5,
  thresh = 1e-04,
  max.iters = 100,
  num.cores = 1,
  rand.seed = 1,
  gen.new = FALSE,
  nrep = 1,
```

```
  plot.log2 = TRUE,
  return.data = FALSE,
  return.raw = FALSE,
  verbose = TRUE
)
```

Arguments

object	liger object. Should normalize, select genes, and scale before calling.
k.test	Set of factor numbers to test (default seq(5, 50, 5)).
lambda	Lambda to use for all foctorizations (default 5).
thresh	Convergence threshold. Convergence occurs when lobj0-objl/(mean(obj0,obj)) < thresh
max.iters	Maximum number of block coordinate descent iterations to perform
num.cores	Number of cores to use for optimizing factorizations in parallel (default 1)
rand.seed	Random seed for reproducibility (default 1).
gen.new	Do not use optimizeNewK in factorizations. Results in slower factorizations. (default FALSE).
nrep	Number restarts to perform at each k value tested (increase to produce smoother curve if results unclear) (default 1).
plot.log2	Plot log2 curve for reference on K-L plot (log2 is upper bound and con sometimes help in identifying "elbow" of plot). (default TRUE)
return.data	Whether to return list of data matrices (raw) or dataframe (processed) instead of ggplot object (default FALSE).
return.raw	If return.results TRUE, whether to return raw data (in format described below), or dataframe used to produce ggplot object. Raw data is list of matrices of K-L divergences (length(k.test) by n_cells). Length of list corresponds to nrep. (default FALSE)
verbose	Print progress bar/messages (TRUE by default)

Value

Matrix of results if indicated or ggplot object. Plots K-L divergence vs. k to console.

Examples

```
## Not run:
# Requires preprocessed liger object
# examine plot for most appropriate k, use multiple cores for faster results
suggestK(ligerex, num.cores = 4)

## End(Not run)
```

suggestLambda *Visually suggest appropriate lambda value*

Description

Can be used to select appropriate value of lambda for factorization of particular dataset. Plot alignment and agreement for various test values of lambda. Most appropriate lambda is likely around the "elbow" of the alignment plot (when alignment stops increasing). This will likely also correspond to slower decrease in agreement. Depending on number of cores used, this process can take 10-20 minutes.

Usage

```
suggestLambda(
  object,
  k,
  lambda.test = NULL,
  rand.seed = 1,
  num.cores = 1,
  thresh = 1e-04,
  max.iters = 100,
  knn_k = 20,
  k2 = 500,
  ref_dataset = NULL,
  resolution = 1,
  gen.new = FALSE,
  nrep = 1,
  return.data = FALSE,
  return.raw = FALSE,
  verbose = TRUE
)
```

Arguments

object	liger object. Should normalize, select genes, and scale before calling.
k	Number of factors to use in test factorizations. See optimizeALS documentation.
lambda.test	Vector of lambda values to test. If not given, use default set spanning 0.25 to 60
rand.seed	Random seed for reproducibility (default 1).
num.cores	Number of cores to use for optimizing factorizations in parallel (default 1).
thresh	Convergence threshold. Convergence occurs when |obj0-obj|/(mean(obj0,obj)) < thresh
max.iters	Maximum number of block coordinate descent iterations to perform
knn_k	Number of nearest neighbors for within-dataset knn in quantileAlignSNF (default 20).

k2	Horizon parameter for quantileAlignSNF (default 500).
ref_dataset	Reference dataset for quantileAlignSNF (defaults to larger dataset).
resolution	Resolution for quantileAlignSNF (default 1).
gen.new	Do not use optimizeNewLambda in factorizations. Recommended to set TRUE when looking at only a small range of lambdas (ie. 1:7) (default FALSE)
nrep	Number restarts to perform at each lambda value tested (increase to produce smoother curve if results unclear) (default 1).
return.data	Whether to return list of data matrices (raw) or dataframe (processed) instead of ggplot object (default FALSE).
return.raw	If return.results TRUE, whether to return raw data (in format described below), or dataframe used to produce ggplot object. Raw data is matrix of alignment values for each lambda value tested (each column represents a different rep for nrep).(default FALSE)
verbose	Print progress bar/messages (TRUE by default)

Value

Matrix of results if indicated or ggplot object. Plots alignment vs. lambda to console.

Examples

```
## Not run:
# Requires preprocessed liger object
# examine plot for most appropriate lambda, use multiple cores for faster results
suggestLambda(ligerex, k = 20, num.cores = 4)

## End(Not run)
```

---

sumGroups *sumGroups*

Description

Utility function to sum over group labels

Usage

```
sumGroups(X, y, MARGIN = 2)

## S3 method for class 'dgCMatrix'
sumGroups(X, y, MARGIN = 2)

## S3 method for class 'matrix'
sumGroups(X, y, MARGIN = 2)
```

Arguments

X	matrix
y	group labels
MARGIN	whether observations are rows (=2) or columns (=1)

Value

Matrix of groups by features

Index