Package 'BatChef'

May 12, 2026

Type Package

Title Single-cell RNA-seq batch effects correction methods interface

Version 1.1.0

Description This package implements a variety of methods for batch correction in single-cell RNA sequencing (scRNA-seq) data. It incorporates quantitative metrics (e.g. Wasserstein distance, Adjusted Rand Index) to evaluate their performance.
Furthermore, the package assists users in identifying and applying the optimal method for specific datasets.

Depends R (>= 4.5.0)

License GPL (>= 3)

Encoding UTF-8

biocViews BatchEffect, SingleCell, Sequencing

RoxygenNote 7.3.3

LinkingTo Rcpp, RcppArmadillo

Imports anndata, aricode, batchelor, bluster, cluster, e1071, fitdistrplus, ggplot2, harmony, leidenAlg, limma, Matrix, mclust, methods, parallel, purrr, RANN, Rcpp, reticulate, rliger, S4Vectors, scCustomize, scMerge, scrapper, Seurat, SeuratObject, sf, SingleCellExperiment, SparseArray, splatter, stats, SummarizedExperiment, sva, transport, zellkonverter

BugReports https://github.com/zuinelena3/BatChef/issues

URL https://github.com/zuinelena3/BatChef

VignetteBuilder knitr

Suggests BiocStyle, knitr, R.devices, rmarkdown, scater, testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData false

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

git_branch devel

git_last_commit b4ef68e

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-12

Author Elena Zuin [aut, cre] (ORCID: <https://orcid.org/0009-0006-3060-4835>, affiliation: Department of Biology, University of Padova),
Chiara Romualdi [ctb] (affiliation: Department of Biology, University of Padova),
Davide Risso [ctb] (affiliation: Department of Statistical Sciences, University of Padova),
Gabriele Sales [ctb] (affiliation: Department of Biology, University of Padova)

Maintainer Elena Zuin <elena.zuin.3@phd.unipd.it>

Contents

`BatChef-package` *BatChef: Single-cell RNA-seq batch effects correction methods interface*

Description

This package implements a variety of methods for batch correction in single-cell RNA sequencing (scRNA-seq) data. It incorporates quantitative metrics (e.g. Wasserstein distance, Adjusted Rand Index) to evaluate their performance. Furthermore, the package assists users in identifying and applying the optimal method for specific datasets.

Author(s)

Maintainer: Elena Zuin <elena.zuin.3@phd.unipd.it> (ORCID) (Department of Biology, University of Padova)

Other contributors:

- Chiara Romualdi (Department of Biology, University of Padova) [contributor]
- Davide Risso (Department of Statistical Sciences, University of Padova) [contributor]
- Gabriele Sales (Department of Biology, University of Padova) [contributor]

See Also

Useful links:

- https://github.com/zuinelena3/BatChef
- Report bugs at https://github.com/zuinelena3/BatChef/issues

adjusted_rand_index *Adjusted Rand Index (ARI)*

Description

The Adjusted Rand Index (ARI) is a metric used to measure the similarity between the clustering and ground truth labels.

Usage

```
adjusted_rand_index(
  input,
  label_true,
  reduction,
  nmi_compute = FALSE,
  resolution,
  k = 10
)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
label_true	A string specifying the ground truth label.
reduction	A string specifying the dimensional reduction on which the clustering analysis will be performed.
nmi_compute	A Boolean value indicating NMI metric calculation to identify the optimal clustering is to be performed.
resolution	A numeric value specifying the resolution parameter.
k	An integer scalar specifying the number of nearest neighbors.

Details

After computing Leiden clustering algorithm, the ARI metric is computed.

Value

A numeric value

Examples

```
sim <- simulate_data(
  n_genes = 500, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
ari <- adjusted_rand_index(
  input = sim, label_true = "Group", reduction = "PCA",
  nmi_compute = FALSE, resolution = 0.5
)
```

`average_silhouette_width` *Average Silhouette Width (ASW)*

Description

The average silhouette width for each cell is defined as the difference between the average of within-cluster distances to all cells and the average between-cluster distances of that cell to the closest cluster divided by their maximum.

Usage

```
average_silhouette_width(input, label_true, reduction, metric = "euclidean")
```

Arguments

`input`	A SingleCellExperiment object.
`label_true`	A string specifying the ground truth label.
`reduction`	A string specifying the dimensional reduction.
`metric`	The metric to use when calculating distance between instances.

Value

A numeric value

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
asw <- average_silhouette_width(
  input = sim, label_true = "Group",
  reduction = "PCA"
)
```

`batchCorrect` *batchCorrect*

Description

A common interface for single-cell batch correction methods.

Usage

```
batchCorrect(input, batch, params)

## S4 method for signature 'LimmaParams'
batchCorrect(input, batch, params)

## S4 method for signature 'CombatParams'
batchCorrect(input, batch, params)

## S4 method for signature 'SeuratV3Params'
batchCorrect(input, batch, params)

## S4 method for signature 'SeuratV5Params'
batchCorrect(input, batch, params)

## S4 method for signature 'FastMNNParams'
batchCorrect(input, batch, params)

## S4 method for signature 'HarmonyParams'
batchCorrect(input, batch, params)

## S4 method for signature 'ScMerge2Params'
batchCorrect(input, batch, params)

## S4 method for signature 'LigerParams'
batchCorrect(input, batch, params)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
batch	A string specifying the batch variable.
params	A BatChefParams object specifying the batch correction method to use and the parameters for its execution.

Details

Users can pass parameters to each method via the constructors for params.

Value

A SingleCellExperiment Seurat or 'AnnData' object, where the output of the method (such as the corrected gene expression matrix and/or the corrected dimensional reduction space) is stored within the original input object.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
out <- batchCorrect(input = sim, batch = "Batch", params = HarmonyParams())
```

batch_params *Batch effects strength*

Description

Batch effects strength

Usage

```
batch_params(norm_counts, batch)
```

Arguments

norm_counts	Normalized counts matrix.
batch	A string specifying the batch variable.

Value

Median of symmetric Kullback-Leibler (KL) divergence

capture_params *Capture parameters of the correction methods*

Description

Capture parameters of the correction methods

Usage

```
capture_params(target, params)
```

Arguments

target	Target parameters
params	Parameters

Value

Parameters

clustInput *Convert into a SingleCellExperiment object for clustering*

Description

Convert into a SingleCellExperiment object for clustering

Usage

```
clustInput(input, reduction)

## S4 method for signature 'Seurat'
clustInput(input, reduction)

## S4 method for signature 'SingleCellExperiment'
clustInput(input, reduction)

## S4 method for signature 'AnnDataR6'
clustInput(input, reduction)
```

Arguments

input	A SingleCellExperiment Seurat or ‘AnnData‘ object can be supplied.
reduction	A string specifying the batch for each cell.

Value

A SingleCellExperiment object.

combatRun *ComBat method*

Description

ComBat allows users to adjust for batch effects in datasets using an empirical Bayes framework.

Usage

```
combatRun(input, batch, assay_type = "counts", ...)
```

Arguments

input	A SingleCellExperiment object.
batch	A string specifying the batch for each cell.
assay_type	A string specifying the assay.
...	Named arguments to pass to individual methods upon dispatch.

Value

A corrected matrix

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = FALSE,
  output_format = "SingleCellExperiment"
)
combat <- combatRun(input = sim, batch = "Batch")
```

---

compute_lisi *Local Inverse Simpson Index (LISI)*

Description

Local Inverse Simpson Index (LISI) scores are computed for each cell.

Usage

```
compute_lisi(X, meta_data, label_colnames, perplexity = 30, nn_eps = 0)
```

Arguments

X	A matrix with cells (rows) and features (columns).
meta_data	A data frame with one row per cell.
label_colnames	Which variables to compute LISI for.
perplexity	The effective number of each cell's neighbors.
nn_eps	Error bound for nearest neighbor search with RANN:nn2(). Default of 0.0 implies exact nearest neighbor search.

Value

A data frame of LISI values. Each row is a cell and each column is a different label variable.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
red <- SingleCellExperiment::reducedDim(sim, "PCA")
lisi <- compute_lisi(
  X = red, meta_data = SingleCellExperiment::colData(sim),
  label_colnames = "Group"
)
```

compute_simpson_index *Compute the Local Inverse Simpson Index (LISI)*

Description

Compute the Local Inverse Simpson Index (LISI)

Usage

```
compute_simpson_index(
  D,
  knn_idx,
  batch_labels,
  n_batches,
  perplexity = 15,
  tol = 1e-05
)
```

Arguments

D	Distance matrix of K nearest neighbors.
knn_idx	Adjacency matrix of K nearest neighbors.
batch_labels	A categorical variable.
n_batches	The number of categories in the categorical variable.
perplexity	The effective number of neighbors around each cell.
tol	Stop when the score converges to this tolerance.

Value

A vector of float values

extract_features *Extract data characteristics*

Description

Extract data characteristics

Usage

```
extract_features(input, batch)
```

Arguments

input	A SingleCellExperiment object can be supplied.
batch	A string specifying the batch variable.

Value

A data.frame that contains the features of input data.

fastMNNPost *Convert fastMNN output*

Description

Convert fastMNN output into a SingleCellExperiment Seurat or 'AnnData' object

Usage

```
fastMNNPost(input, output, method)

## S4 method for signature 'Seurat'
fastMNNPost(input, output, method)

## S4 method for signature 'SingleCellExperiment'
fastMNNPost(input, output, method)

## S4 method for signature 'AnnDataR6'
fastMNNPost(input, output, method)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
output	fastMNN output: a SingleCellExperiment containing the reconstructed expression matrix and the corrected dimensionality reduction space.
method	A string specifying the correction method.

Value

A SingleCellExperiment Seurat or 'AnnData' object.

fastMNNRun *fastMNN method*

Description

Correct for batch effects in single-cell expression data using a fast version of the mutual nearest neighbors (MNN) method.

Usage

```
fastMNNRun(input, batch, ...)
```

Arguments

input	SingleCellExperiment object.
batch	A string specifying the batch variable.
...	Named arguments to pass to individual methods upon dispatch

Value

SingleCellExperiment object is returned where each row is a gene and each column is a cell. This contains a corrected low-dimensional coordinates and a reconstructed matrix in the assays slot.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
fastmnn <- fastMNNRun(input = sim, batch = "Batch")
```

---

harmonyPost *Convert Harmony output*

Description

Convert Harmony output into a SingleCellExperiment Seurat or 'AnnData' object.

Usage

```
harmonyPost(input, output, method)

## S4 method for signature 'Seurat'
harmonyPost(input, output, method)

## S4 method for signature 'SingleCellExperiment'
harmonyPost(input, output, method)

## S4 method for signature 'AnnDataR6'
harmonyPost(input, output, method)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
output	Harmony output: a SingleCellExperiment containing the corrected dimensionality reduction space.
method	A string specifying the correction method.

Value

A SingleCellExperiment Seurat or 'AnnData' object.

harmonyRun *Harmony method*

Description

Harmony is a mixture-model based method.

Usage

```
harmonyRun(input, batch, ...)
```

Arguments

input	A SingleCellExperiment object.
batch	A string specifying the batch variable.
...	Named arguments to pass to individual methods upon dispatch

Value

A SingleCellExperiment object which contains a corrected low-dimensional space.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
harmony <- harmonyRun(input = sim, batch = "Batch")
```

kl_gamma *Kullback-Leibler (KL) divergence) between two Gamma distributions*

Description

Kullback-Leibler (KL) divergence) between two Gamma distributions

Usage

```
kl_gamma(p, q)
```

Arguments

p	Gamma parameters (shape and rate) of probability distribution P
q	Gamma parameters (shape and rate) of probability distribution Q

Value

Kullback-Leibler (KL) divergence) between two Gamma distributions

`leiden_clustering` *Leiden clustering*

Description

Leiden clustering algorithm.

Usage

```
leiden_clustering(
  input,
  label_true = NULL,
  reduction,
  nmi_compute = TRUE,
  resolution = NULL,
  k = 15,
  store = FALSE,
  n_iter = -1
)
```

Arguments

`input`	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
`label_true`	A string specifying the ground truth label.
`reduction`	A string specifying the dimensional reduction on which the clustering analysis will be performed.
`nmi_compute`	A Boolean value indicating NMI metric calculation to identify the optimal clustering is to be performed (Default: TRUE).
`resolution`	A numeric value specifying the resolution parameter.
`k`	An integer scalar specifying the number of nearest neighbors.
`store`	A Boolean value indicating whether cluster labels are stored within the input object (Default: FALSE).
`n_iter`	Number of iterations of the Leiden clustering algorithm to perform. Positive values above 2 define the total number of iterations to perform, -1 has the algorithm run until it reaches its optimal clustering

Details

The clustering algorithm can be executed by specifying either a single resolution parameter or range of resolution parameters (from 0.1 to 2). In the case of multiple resolutions, the clustering outcome that corresponds to the highest Normalized Mutual Information (NMI) score is selected. Finding the optimal clustering can be useful to compute the performance evaluation of batch correction methods.

Value

A A SingleCellExperiment Seurat or 'AnnData' object that contains the cluster labels.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
clust <- leiden_clustering(
  input = sim, reduction = "PCA",
  nmi_compute = FALSE, resolution = 1, n_iter = 2
)
```

---

lib_size_params *Library size parameters*

Description

Library size parameters

Usage

```
lib_size_params(counts)
```

Arguments

counts	Raw counts matrix.

Value

Median of sequencing depth

---

ligerInput *Convert to a LIGER compatible object*

Description

Convert to a LIGER compatible object

Usage

```
ligerInput(input, batch, features, ...)

## S4 method for signature 'Seurat'
ligerInput(input, batch, features, ...)

## S4 method for signature 'SingleCellExperiment'
ligerInput(input, batch, features, ...)

## S4 method for signature 'AnnDataR6'
ligerInput(input, batch, features, ...)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
batch	A string specifying the batch variable.
features	Vector of features to use.
...	Named arguments to pass to individual methods upon dispatch.

Value

A LIGER compatible object

---

ligerPost *Convert the LIGER output*

Description

Convert the LIGER output into a SingleCellExperiment Seurat or 'AnnData' object.

Usage

```
ligerPost(input, output, method)

## S4 method for signature 'Seurat'
ligerPost(input, output, method)

## S4 method for signature 'SingleCellExperiment'
ligerPost(input, output, method)

## S4 method for signature 'AnnDataR6'
ligerPost(input, output, method)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
output	LIGER output: liger object
method	A string specifying the correction method

Value

A SingleCellExperiment Seurat or 'AnnData' object.

ligerRun *LIGER method*

Description

LIGER is a integrative non-negative matrix factorization method

Usage

```
ligerRun(
  input,
  method = "iNMF",
  quantiles = 50,
  reference = NULL,
  min_cells = 20,
  n_neighbors = 20,
  use_dims = NULL,
  center = FALSE,
  max_sample = 1000,
  eps = 0.9,
  refine_knn = TRUE,
  cluster_name = "quantileNorm_cluster",
  seed = 1,
  verbose = FALSE,
  ...
)
```

Arguments

input	A liger object.
method	A string specifying the batch correction method. Choose from "iNMF", "onlineINMF", "UINMF". Default "iNMF".
quantiles	Number of quantiles to use for quantile normalization.
reference	Character, numeric or logical selection of one dataset, out of all available datasets in object, to use as a "reference" for quantile normalization.
min_cells	Minimum number of cells to consider a cluster shared across datasets
n_neighbors	Number of nearest neighbors for within-dataset knn graph.
use_dims	Indices of factors to use for shared nearest factor determination.
center	A logical to center data.
max_sample	Maximum number of cells used for quantile normalization of each cluster and factor
eps	The error bound of the nearest neighbor search.
refine_knn	A logical to increase robustness of cluster assignments using KNN graph.
cluster_name	Variable name that will store the clustering result in metadata of a liger object or a Seurat object.
seed	Random seed
verbose	Print progress bar/messages
...	Named arguments to pass to individual methods upon dispatch.

Value

A liger object, which contains the quantile align factor loadings.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)

ll <- lapply(
  unique(SingleCellExperiment::colData(sim)[, "Batch"]),
  function(i) sim[, SingleCellExperiment::colData(sim)[, "Batch"] == i]
)
names(ll) <- unique(SingleCellExperiment::colData(sim)[, "Batch"])
# Create a liger object
lo <- rliger::createLiger(rawData = ll)
lo <- rliger::normalize(object = lo)
lo <- rliger::scaleNotCenter(object = lo, features = rownames(sim))
liger <- ligerRun(input = lo)
```

LimmaParams *BatChefParams methods*

Description

Constructors and methods for the params parameter classes. BatChefParams objects contain method specific parameters to pass to the batchCorrect generic.

Usage

```
LimmaParams(assay_type = "logcounts", ...)

CombatParams(assay_type = "counts", ...)

SeuratV3Params(
  features,
  pca_name = NULL,
  assay = NULL,
  reference = NULL,
  anchor_features = 2000,
  scale = TRUE,
  normalization_method = "LogNormalize",
  sct_clip_range = NULL,
  reduction = "cca",
  l2_norm = TRUE,
  dims = 1:30,
  k_anchor = 5,
  k_filter = 200,
  k_score = 30,
```

```
  max_features = 200,
  nn_method = "annoy",
  n_trees = 50,
  eps = 0,
  verbose = FALSE,
  new_assay_name = "integrated",
  features_to_integrate = NULL,
  k_weight = 100,
  weight_reduction = NULL,
  sd_weight = 1,
  sample_tree = NULL,
  preserve_order = FALSE
)

SeuratV5Params(
  pca_name = NULL,
  method = "CCAIntegration",
  orig_reduction = "pca",
  assay = NULL,
  features = NULL,
  layers = NULL,
  scale_layer = "scale.data",
  new_reduction = "integrated.dr",
  reference = NULL,
  normalization_method = "LogNormalize",
  dims = 1:30,
  k_filter = NA,
  dims_to_integrate = NULL,
  k_weight = 100,
  weight_reduction = NULL,
  sd_weight = 1,
  sample_tree = NULL,
  preserve_order = FALSE,
  verbose = FALSE,
  l2_norm = TRUE,
  k_anchor = 5,
  k_score = 30,
  max_features = 200,
  nn_method = "annoy",
  n_trees = 50,
  eps = 0
)

FastMNNParams(...)

HarmonyParams(...)

ScMerge2Params(assay_type = "logcounts", ...)

LigerParams(features, method = "iNMF", ...)
```

Arguments

assay_type	A string specifying the assay to use for correction.
...	Named arguments to pass to individual methods upon dispatch.
features	Vector of features to use.
pca_name	A string specifying the PCA name.
assay	Name of assay for integration.
reference	A reference Seurat object.
anchor_features	Number of features to be used in anchor finding.
scale	A logical to scale the features provided.
normalization_method	Name of normalization method used: LogNormalize or SCT.
sct_clip_range	Numeric of length two specifying the min and max values the Pearson residual will be clipped to.
reduction	Dimensional reduction to perform when finding anchors.
l2_norm	Perform L2 normalization on the CCA cell embeddings after dimensional reduction.
dims	Number of dimensions of dimensional reduction.
k_anchor	Number of neighbors (k) to use when picking anchors.
k_filter	Number of anchors to filter.
k_score	Number of neighbors (k) to use when scoring anchors.
max_features	The maximum number of features to use when specifying the neighborhood search space in the anchor filtering.
nn_method	Method for nearest neighbor finding.
n_trees	More trees gives higher precision when using annoy approximate nearest neighbor search.
eps	Error bound on the neighbor finding algorithm.
verbose	Print progress bars and output.
new_assay_name	Name for the new assay containing the integrated data.
features_to_integrate	Vector of features to integrate.
k_weight	Number of neighbors to consider when weighting anchors.
weight_reduction	Dimension reduction to use when calculating anchor weights.
sd_weight	Controls the bandwidth of the Gaussian kernel for weighting.
sample_tree	Specify the order of integration.
preserve_order	Do not reorder objects based on size for each pairwise integration.
method	iNMF variant algorithm to use for integration.
orig_reduction	Name of dimensional reduction for correction.
layers	Names of normalized layers in assay.
scale_layer	Name(s) of scaled layer(s) in assay.
new_reduction	Name of new integrated dimensional reduction.
dims_to_integrate	Number of dimensions to return integrated values for.

Value

A BatChefParams object of the specified subclass, containing parameter settings for the corresponding batch correction method.

`limmaRun` *limma method*

Description

limma method

Usage

```
limmaRun(input, batch, assay_type = "logcounts", ...)
```

Arguments

`input`	A SingleCellExperiment object.
`batch`	A string specifying the batch variable.
`assay_type`	A string specifying the assay.
`...`	Named arguments to pass to individual methods upon dispatch.

Value

A corrected matrix

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = FALSE,
  output_format = "SingleCellExperiment"
)
limma <- limmaRun(input = sim, batch = "Batch")
```

`linearInput` *Convert into a SingleCellExperiment object for linear model based methods*

Description

Convert into a SingleCellExperiment object for linear model based methods

Usage

```
linearInput(input, batch)

## S4 method for signature 'Seurat'
linearInput(input, batch)

## S4 method for signature 'SingleCellExperiment'
linearInput(input, batch)

## S4 method for signature 'AnnDataR6'
linearInput(input, batch)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
batch	A string specifying the batch for each cell.

Value

A SingleCellExperiment object.

`linearPost` *Convert the output of linear models*

Description

Convert a corrected matrix output into a SingleCellExperiment Seurat or 'AnnData' object.

Usage

```
linearPost(input, output, method)

## S4 method for signature 'Seurat'
linearPost(input, output, method)

## S4 method for signature 'SingleCellExperiment'
linearPost(input, output, method)

## S4 method for signature 'AnnDataR6'
linearPost(input, output, method)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
output	A corrected matrix.
method	A string specifying the correction method.

Value

A SingleCellExperiment Seurat or 'AnnData' object.

`local_inverse_simpson_index` *Median of Local Inverse Simposon Index (LISI)*

Description

Local Inverse Simpson's Index (LISI) is a local level metric based on the kNN algorithm. This metric defines the effective number of datasets in a neighborhood. In other words, LISI determines the number of neighbor cells necessary before one batch is observed twice.

Usage

```
local_inverse_simpson_index(
  input,
  label_true,
  reduction,
  meta_data = colData(input),
  perplexity = 30,
  nn_eps = 0
)
```

Arguments

`input`	A SingleCellExperiment object.
`label_true`	A string specifying cell types.
`reduction`	A string specifying the dimensional reduction.
`meta_data`	A data frame with one row per cell.
`perplexity`	The effective number of each cell's neighbors.
`nn_eps`	Error bound for nearest neighbor search with 'RANN:nn2()'.

Value

A numeric value.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
lisi <- local_inverse_simpson_index(
  input = sim, label_true = "Group",
  reduction = "PCA"
)
```

log_transf *A shifted log transformation*

Description

A shifted log transformation

Usage

```
log_transf(mat, base = exp(1))
```

Arguments

mat	A matrix of data characteristics
base	logarithm base

Value

A shifted log transformation matrix.

merge_params *Parameters merging*

Description

Parameters merging

Usage

```
merge_params(base, extra, class)
```

Arguments

base	base params
extra	extra params
class	class

Value

Vector of strings of base and extra parameters.

`metrics` *Performance evaluation metrics.*

Description

Performance evaluation metrics: Wasserstein distance, Local Inverse Simpson's Index, Average Silhouette Width, Adjusted Rand Index, and Normalized Mutual Information.

Usage

```
metrics(
  input,
  batch,
  group,
  reduction,
  rep = 10,
  mc_cores = 1,
  nmi_compute = TRUE,
  resolution = NULL,
  k = 10,
  metric = "euclidean",
  variant = "sum",
  meta_data = colData(input),
  perplexity = 30,
  nn_eps = 0
)
```

Arguments

`input`	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
`batch`	A string specifying the batch variable.
`group`	A string specifying the ground truth labels.
`reduction`	A string specifying the dimensional reduction. on which the clustering analysis will be performed.
`rep`	Number of times the Wasserstein distance is calculated.
`mc_cores`	The number of cores to use.
`nmi_compute`	A Boolean value indicating NMI metric calculation to identify the optimal clustering is to be performed (Default: TRUE).
`resolution`	A numeric value specifying the resolution parameter.
`k`	An integer scalar specifying the number of nearest neighbors.
`metric`	The metric to use when calculating distance between instances.
`variant`	How to compute the normalizer in the denominator.
`meta_data`	A data frame with one row per cell.
`perplexity`	The effective number of each cell's neighbors.
`nn_eps`	Error bound for nearest neighbor search with 'RANN:nn2()'.

Details

This function performs Leiden clustering before computing the evaluation metrics.

Value

A data.frame object

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 110),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
metrics <- metrics(
  input = sim, batch = "Batch", group = "Group",
  reduction = "PCA", rep = 5
)
```

---

normalized *Normalization*

Description

Normalization

Usage

```
normalized(counts)
```

Arguments

counts	Raw counts matrix.

Value

A normalized matrix.

---

normalized_mutual_info *Normalized Mutual Information (NMI)*

Description

The Normalized Mutual Information (NMI) metric is used to compare the overlap between the true cell-type and clustering labels computed after batch correction.

Usage

```
normalized_mutual_info(
  input,
  label_true,
  reduction,
  nmi_compute = FALSE,
  resolution = 1,
  k = 10,
  variant = "sum"
)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
label_true	A string specifying the ground truth labels.
reduction	A string specifying the dimensional reduction on which the clustering analysis will be performed.
nmi_compute	A Boolean value indicating NMI metric calculation to identify the optimal clustering is to be performed (Default: FALSE).
resolution	A numeric value specifying the resolution parameter.
k	An integer scalar specifying the number of nearest neighbors.
variant	How to compute the normalizer in the denominator.

Value

A numeric value.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
nmi <- normalized_mutual_info(
  input = sim, label_true = "Group", reduction = "PCA",
  nmi_compute = FALSE, resolution = 0.5
)
```

---

outlier_params *Highly expressed genes probability*

Description

Highly expressed genes probability

Usage

```
outlier_params(norm_counts)
```

Arguments

norm_counts	A normalized matrix

Value

Highly expressed genes probability

---

params_btc_factors *Gamma parameters estimation of batch factors*

Description

Gamma parameters estimation of batch factors

Usage

```
params_btc_factors(factors)
```

Arguments

factors	Batch factors

Value

a data.frame with estimated shape and rate parameters

---

prediction_plot *Prediction plot*

Description

Prediction plot

Usage

```
prediction_plot(params)
```

Arguments

params	A data.frame of data characteristics

Value

A ggplot object

sceInput *Convert into a SingleCellExperiment object*

Description

Convert into a SingleCellExperiment object

Usage

```
sceInput(input, batch)

## S4 method for signature 'Seurat'
sceInput(input, batch)

## S4 method for signature 'SingleCellExperiment'
sceInput(input, batch)

## S4 method for signature 'AnnDataR6'
sceInput(input, batch)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
batch	A string specifying the batch for each cell.

Value

A SingleCellExperiment object.

scMerge2Post *Convert the scMerge2 output*

Description

Convert the scMerge2 output into a SingleCellExperiment Seurat or 'AnnData' object.

Usage

```
scMerge2Post(input, output, method)

## S4 method for signature 'Seurat'
scMerge2Post(input, output, method)

## S4 method for signature 'SingleCellExperiment'
scMerge2Post(input, output, method)

## S4 method for signature 'AnnDataR6'
scMerge2Post(input, output, method)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
output	scMerge2 output: a list that contains the corrected gene expression matrix
method	A string specifying the correction method

Value

A SingleCellExperiment Seurat or 'AnnData' object.

---

scMerge2Run *scMerge2 method*

Description

scMerge2 is based on pseudo-replicates to remove unwanted batch effects.

Usage

```
scMerge2Run(input, batch, assay_type = "logcounts", ...)
```

Arguments

input	A SingleCellExperiment object.
batch	A string specifying the batch variable.
assay_type	A string specifying the assay to use for correction.
...	Named arguments to pass to individual methods upon dispatch.

Value

A list that contains the corrected gene expression matrix

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(250, 200),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
scmerge2 <- scMerge2Run(input = sim, batch = "Batch", assay_type = "logcounts")
```

seuratv3Input *Convert to a SeuratV3 compatible object*

Description

Convert to a SeuratV3 compatible object

Usage

```
seuratv3Input(input, batch, features, pca_name = NULL)

## S4 method for signature 'Seurat'
seuratv3Input(input, batch, features, pca_name = NULL)

## S4 method for signature 'SingleCellExperiment'
seuratv3Input(input, batch, features, pca_name = NULL)

## S4 method for signature 'AnnDataR6'
seuratv3Input(input, batch, features, pca_name = NULL)

## S4 method for signature 'AnnDataR6'
seuratv3Input(input, batch, features, pca_name = NULL)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
batch	A string specifying the batch variable.
features	Vector of features to use.
pca_name	A string specifying the PCA.

Value

A list of Seurat objects.

seuratv3Post *Convert the SeuratV3 output*

Description

Convert the SeuratV3 output into a SingleCellExperiment Seurat or 'AnnData' object.

Usage

```
seuratv3Post(input, output, method)

## S4 method for signature 'Seurat'
seuratv3Post(input, output, method)

## S4 method for signature 'SingleCellExperiment'
```

```
seuratv3Post(input, output, method)

## S4 method for signature 'AnnDataR6'
seuratv3Post(input, output, method)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
output	Seurat V3 output: a Seurat object
method	A string specifying the correction method

Value

A SingleCellExperiment Seurat or 'AnnData' object.

`seuratV3Run` *Seurat V3 method*

Description

SeuratV3 is an anchor-based method.

Usage

```
seuratV3Run(
  input,
  assay = NULL,
  reference = NULL,
  anchor_features = 2000,
  scale = TRUE,
  normalization_method = "LogNormalize",
  sct_clip_range = NULL,
  reduction = "cca",
  l2_norm = TRUE,
  dims = 1:30,
  k_anchor = 5,
  k_filter = 200,
  k_score = 30,
  max_features = 200,
  nn_method = "annoy",
  n_trees = 50,
  eps = 0,
  verbose = FALSE,
  new_assay_name = "integrated",
  features = NULL,
  features_to_integrate = NULL,
  k_weight = 100,
  weight_reduction = NULL,
  sd_weight = 1,
  sample_tree = NULL,
  preserve_order = FALSE
)
```

Arguments

input	A list of Seurat objects.
assay	A vector of assay names specifying which assay to use when constructing anchors.
reference	A vector specifying the object/s to be used as a reference during integration.
anchor_features	Number of features to be used in anchor finding.
scale	A logical to scale the features provided.
normalization_method	Name of normalization method used: LogNormalize (default) or SCT.
sct_clip_range	Numeric of length two specifying the min and max values the Pearson residual will be clipped to.
reduction	Dimensional reduction to perform when finding anchors.
l2_norm	Perform L2 normalization on the CCA cell embeddings after dimensional reduction.
dims	Number of dimensions.
k_anchor	Number of neighbors (k) to use when picking anchors.
k_filter	Number of neighbors (k) to use when filtering anchors.
k_score	Number of neighbors (k) to use when scoring anchors.
max_features	The maximum number of features to use when specifying the neighborhood search space in the anchor filtering.
nn_method	Method for nearest neighbor finding.
n_trees	More trees gives higher precision when using annoy approximate nearest neighbor search.
eps	Error bound on the neighbor finding algorithm.
verbose	Print progress bars and output.
new_assay_name	Name for the new assay containing the integrated data.
features	Vector of features to use.
features_to_integrate	Vector of features to integrate.
k_weight	Number of neighbors to consider when weighting anchors.
weight_reduction	Dimension reduction to use when calculating anchor weights.
sd_weight	Controls the bandwidth of the Gaussian kernel for weighting.
sample_tree	Specify the order of integration.
preserve_order	Do not reorder objects based on size for each pairwise integration.

Value

A Seurat object that contains the corrected matrix.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(200, 200),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = FALSE,
  output_format = "Seurat"
)
feat <- Seurat::VariableFeatures(sim)
sim <- Seurat::SplitObject(sim, split.by = "Batch")
seuv3 <- seuratV3Run(
  input = sim, reduction = "cca",
  features = feat
)
```

---

seuratv5Input *Convert to a SeuratV5 compatible object*

Description

Convert to a SeuratV5 compatible object

Usage

```
seuratv5Input(input, batch, features = NULL, pca_name = NULL)

## S4 method for signature 'Seurat'
seuratv5Input(input, batch, features = NULL, pca_name = NULL)

## S4 method for signature 'SingleCellExperiment'
seuratv5Input(input, batch, features = NULL, pca_name = NULL)

## S4 method for signature 'AnnDataR6'
seuratv5Input(input, batch, features = NULL, pca_name = NULL)
```

Arguments

input	A SingleCellExperiment Seurat or ‘AnnData‘ object can be supplied.
batch	A string specifying the batch variable.
features	Vector of features to use.
pca_name	A string specifying the PCA.

Value

A Seurat object.

`seuratv5Post` *Convert the SeuratV5 output*

Description

Convert the SeuratV5 output into a SingleCellExperiment Seurat or 'AnnData' object.

Usage

```
seuratv5Post(input, output, method, name)

## S4 method for signature 'Seurat'
seuratv5Post(input, output, method, name)

## S4 method for signature 'SingleCellExperiment'
seuratv5Post(input, output, method, name)

## S4 method for signature 'AnnDataR6'
seuratv5Post(input, output, method, name)
```

Arguments

`input`	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
`output`	Seurat V5 output: a Seurat object
`method`	A string specifying the correction method.
`name`	A string specifying the corrected reduce space name.

Value

SingleCellExperiment Seurat or 'AnnData' object.

`seuratV5Run` *Seurat V5 method*

Description

SeuratV5 is an anchor-based method.

Usage

```
seuratV5Run(
  input,
  method = "CCAIntegration",
  orig_reduction = "pca",
  assay = NULL,
  features = NULL,
  layers = NULL,
  scale_layer = "scale.data",
  new_reduction = "integrated.dr",
```

```
  reference = NULL,
  normalization_method = "LogNormalize",
  dims = 1:30,
  k_filter = NA,
  dims_to_integrate = NULL,
  k_weight = 100,
  weight_reduction = NULL,
  sd_weight = 1,
  sample_tree = NULL,
  preserve_order = FALSE,
  verbose = FALSE,
  l2_norm = TRUE,
  k_anchor = 5,
  k_score = 30,
  max_features = 200,
  nn_method = "annoy",
  n_trees = 50,
  eps = 0
)
```

Arguments

input	A Seurat object.
method	Integration method function.
orig_reduction	Name of dimensional reduction for correction.
assay	Name of assay for integration.
features	A vector of features to use for integration.
layers	Names of normalized layers in assay.
scale_layer	Name(s) of scaled layer(s) in assay.
new_reduction	Name of new integrated dimensional reduction.
reference	A reference Seurat object.
normalization_method	Name of normalization method used: LogNormalize or SCT.
dims	Number of dimensions of dimensional reduction.
k_filter	Number of anchors to filter.
dims_to_integrate	Number of dimensions to return integrated values for.
k_weight	Number of neighbors to consider when weighting anchors.
weight_reduction	Dimension reduction to use when calculating anchor weights.
sd_weight	Controls the bandwidth of the Gaussian kernel for weighting.
sample_tree	Specify the order of integration.
preserve_order	Do not reorder objects based on size for each pairwise integration.
verbose	Print progress bars and output.
l2_norm	Perform L2 normalization on the CCA cell embeddings after dimensional reduction.
k_anchor	Number of neighbors (k) to use when picking anchors.

k_score	Number of neighbors (k) to use when scoring anchors.
max_features	The maximum number of features to use when specifying the neighborhood search space in the anchor filtering.
nn_method	Method for nearest neighbor finding.
n_trees	More trees gives higher precision when using annoy approximate nearest neighbor search.
eps	Error bound on the neighbor finding algorithm.

Value

A Seurat object.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(250, 200),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE,
  output_format = "Seurat"
)
sim[[SeuratObject::DefaultAssay(sim)]] <- split(
  x = sim[[SeuratObject::DefaultAssay(sim)]],
  f = sim[["Batch"]][, 1]
)
sim <- Seurat::ScaleData(sim, verbose = FALSE)
seuv5 <- seuratV5Run(
  input = sim, method = "CCAIntegration",
  features = rownames(sim)
)
```

---

simulate_data *Function to simulate data*

Description

The function allows to simulated single-cell RNA-seq data using Splatter package, normalize data, select highly variable genes and compute Principal Component Analysis.

Usage

```
simulate_data(
  n_genes = 10000,
  batch_cells = 100,
  batch_fac_loc = 0.1,
  batch_fac_scale = 0.1,
  batch_rm_effect = FALSE,
  mean_rate = 0.3,
  mean_shape = 0.6,
  lib_loc = 11,
  lib_scale = 0.2,
  lib_norm = FALSE,
```

```
  out_prob = 0.05,
  out_fac_loc = 4,
  out_fac_scale = 0.5,
  group_prob = 1,
  de_prob = 0.1,
  de_down_prob = 0.5,
  de_fac_loc = 0.1,
  de_fac_scale = 0.4,
  bcv_common = 0.1,
  bcv_df = 60,
  dropout_type = "none",
  dropout_mid = 0,
  dropout_shape = -1,
  path_from = 0,
  path_n_steps = 100,
  path_skew = 0.5,
  path_nonlinear_prob = 0.1,
  path_sigma_fac = 0.8,
  compute_hvgs = FALSE,
  n_hvgs = 1000,
  num_threads = 1,
  compute_pca = FALSE,
  pca_ncomp = 10,
  output_format = "SingleCellExperiment",
  seed = 333
)
```

Arguments

n_genes	Number of genes.
batch_cells	Number of cells per batch.
batch_fac_loc	Batch factor location parameter.
batch_fac_scale	Batch factor scale parameter.
batch_rm_effect	Remove batch effect.
mean_rate	Mean rate.
mean_shape	Mean shape.
lib_loc	Library size location parameter.
lib_scale	Library size scale parameter.
lib_norm	Library size distribution.
out_prob	Expression outlier probability.
out_fac_loc	Expression outlier factor location.
out_fac_scale	Expression outlier factor scale.
group_prob	Group probabilities.
de_prob	Differential expression probability.
de_down_prob	Down-regulation probability.
de_fac_loc	DE factor location.

de_fac_scale	DE factor scale.
bcv_common	Common biological coefficient of variation.
bcv_df	BCV degrees of freedom.
dropout_type	Dropout type.
dropout_mid	Dropout mid point.
dropout_shape	Dropout shape.
path_from	Path origin.
path_n_steps	Number of steps.
path_skew	Path skew.
path_nonlinear_prob	Non-linear probability.
path_sigma_fac	Path skew.
compute_hvgs	Boolean value. If TRUE, highly variable genes will be selected Default is FALSE.
n_hvgs	Number of highly variable genes.
num_threads	Integer scalar specifying the number of threads to use.
compute_pca	Boolean value. If TRUE, Principal Component Analysis (PCA) will be computed. Default is FALSE.
pca_ncomp	Number of principal component.
output_format	A SingleCellExperiment Seurat or 'AnnData' object.
seed	Random seed.

Value

A SingleCellExperiment Seurat or 'AnnData' object.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 1000,
  compute_pca = FALSE
)
```

---

suggested_method *Suggested method prediction*

Description

Suggested method prediction

Usage

```
suggested_method(input, batch)
```

Arguments

input	A SingleCellExperiment Seurat or 'AnnData' object can be supplied.
batch	A string specifying the batch variable.

Value

A list containing two elements:a string specifying the recommended method; a ggplot object that visualizes the data points in a two-dimensional space derived from the characteristics of 130 datasets.

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(150, 50),
  group_prob = c(0.5, 0.5), n_hvgs = 1000,
  compute_pca = FALSE, output_format = "SingleCellExperiment"
)
pred <- suggested_method(input = sim, batch = "Batch")
```

---

wasserstein_distance *Wasserstein distance*

Description

The Wasserstein distance measures the minimal transport needed to shift one distribution to match another.

Usage

```
wasserstein_distance(input, batch, reduction, rep, mc_cores = 1)
```

Arguments

input	A SingleCellExperiment object.
batch	A string specifying batch variable.
reduction	A string specifying the dimensional reduction.
rep	Number of times the Wasserstein distance is calculated.
mc_cores	The number of cores to use.

Value

A numeric value

Examples

```
sim <- simulate_data(
  n_genes = 1000, batch_cells = c(130, 110),
  group_prob = c(0.5, 0.5), n_hvgs = 500,
  compute_pca = TRUE, output_format = "SingleCellExperiment"
)
wass <- wasserstein_distance(
  input = sim, batch = "Batch",
  reduction = "PCA", rep = 1, mc_cores = 1
)
```

`winsorization` *Winsorization*

Description

It's a transformation by limiting extreme values in data to reduce the effect of outliers.

Usage

```
winsorization(norm_counts)
```

Arguments

`norm_counts`	A normalized matrix

Value

Winsorized numeric vector

Index