# Package: CDI (via r-universe)

November 25, 2026

**Version** 1.12.0

**Date** 2026-03-15

**Title** Clustering Deviation Index (CDI)

**Description** Single-cell RNA-sequencing (scRNA-seq) is widely used to explore cellular variation. The analysis of scRNA-seq data often starts from clustering cells into subpopulations. This initial step has a high impact on downstream analyses, and hence it is important to be accurate. However, there have not been unsupervised metric designed for scRNA-seq to evaluate clustering performance. Hence, we propose clustering deviation index (CDI), an unsupervised metric based on the modeling of scRNA-seq UMI counts to evaluate clustering of cells.

**biocViews** SingleCell, Software, Clustering, Visualization, Sequencing, RNASeq, CellBasedAssays

**URL** https://github.com/jichunxie/CDI

**BugReports** https://github.com/jichunxie/CDI/issues

**Imports** matrixStats, SeuratObject, Seurat, stats, BiocParallel, ggplot2, reshape2, grDevices, ggsci, SingleCellExperiment, SummarizedExperiment, methods

**RoxygenNote** 7.2.3

**Depends** R(>= 3.6)

**Suggests** knitr, rmarkdown, RUnit, BiocGenerics, magick, BiocStyle

**VignetteBuilder** knitr

**License** GPL-3 + file LICENSE

**Encoding** UTF-8

**Config/pak/sysreqs** libglpk-dev make libicu-dev libpng-dev libxml2-dev libssl-dev zlib1g-dev

**Repository** Bioconductor 3.24

**Date/Publication** 2026-11-23 22:50:58 UTC

**RemoteUrl** https://github.com/bioc/CDI

**RemoteRef** RELEASE_3_24

**RemoteSha** a249415ba009418363aa1e4ba10d914085b5d950

# Contents

calculate_CDI . . . 2
CDI . . . 6
CDI_lineplot . . . 6
contingency_heatmap . . . 7
feature_gene_selection . . . 9
one_batch_matrix . . . 11
one_batch_matrix_celltype . . . 12
one_batch_matrix_label_df . . . 12
size_factor . . . 13
two_batch_matrix . . . 14
two_batch_matrix_batch . . . 15
two_batch_matrix_celltype . . . 15
two_batch_matrix_label_df . . . 16

**Index** **17**

---

`calculate_CDI` *Clustering Deviance Index (CDI)*

---

## Description

This function calculates CDI-AIC and CDI-BIC for each candidate set of cell labels. CDI calculates AIC and BIC of cell-type-specific gene-specific NB model for UMI counts, where the cell types are based on each candidate label set, and only the selected subset of genes are considered. Whether to use CDI-AIC or CDI-BIC depend on the goals. We suggest using CDI-BIC to select optimal main cell types and using CDI-AIC to select optimal subtypes, because BIC puts more penalty on the complexity of models (number of clusters).

## Usage

```
calculate_CDI(
  X,
  feature_gene_index = NULL,
  cand_lab_df,
  cell_size_factor,
  batch_label = NULL,
  count_slot = NULL,
  batch_slot = NULL,
  lrt_pval_threshold = 0.01,
  clustering_method = NULL,
  BPPARAM = SerialParam()
)
```

## Arguments

| | |
|---|---|
| X | The class of X can be "matrix", "Seurat" object, or "SingleCellExperiment" object. If X is a matrix, it should be a UMI count matrix where each row represents a gene, and each column represents a cell. If X is a Seurat object or SingleCellExperiment object, users need to specify where the count matrix and batch labels are stored in count_slot and batch_slot, respectively. If feature_gene_index is NULL, genes in X should only included feature genes (that are selected by feature_gene_selection function); if feature_gene_index is not NULL, this function will extract a subset of X with genes indexed by feature_gene_index. |
| feature_gene_index | A vector of unique integers indicating the indices of feature genes. The default value if NULL, which means all genes in X will be used to calculate CDI. The integers in feature_gene_index need to be no greater than the number of genes in X. |
| cand_lab_df | A vector of cluster labels of the cells or a data frame where each column corresponds to one set of cluster labels of the cells. This (these) label sets can be clustering results obtained by any clustering methods. The length (number of rows) of cand_lab_df should be the same as the number of columns in the count matrix. If the column names of label set data frame are provided with the format "[ClusteringMethod]_k[NumberOfClusters]" such as "KMeans_K5, 'calculate_CDI' will extract the "[ClusteringMethod]" as the Cluster_method. The clustering method can also be provided in the argument "clustering_method" for each label set. |
| cell_size_factor | A numeric vector indicating the size factor of the cells. This should be the output of function size_factor. The length of cell_size_factor should be the same as the number of columns in the count matrix. |
| batch_label | A vector of characters indicating the batch labels of the cells. The length of batch_label should be the same as the number of columns in the count matrix. |
| count_slot | A string indicating the location of raw UMI count. For Seurat object, it is a slot in "RNA" of "assays"; For SingleCellExperiment object, it is a slot in "assays". Each row represents a gene, and each column represents a cell. The genes should be those before feature gene selection. |
| batch_slot | A string indicating the location of batch labels of cells. For Seurat object, it is a slot in meta.data; For SingleCellExperiment object, it is a slot in "colData". The default value is NULL indicating that there is no batch information available. |
| lrt_pval_threshold | A numeric value within (0, 1) indicating the p-value threshold for the likelihood ratio test (LRT). If multiple batches exist, within each cluster and each gene, CDI will test whether a batch-common NB model or a batch-specific NB model should be fitted with the LRT. If the p-value is less than this threshold, a batch-specific NB model will be fitted. Otherwise, a batch-common NB model will be fitted. |
| clustering_method | A vector of characters indicating the corresponding clustering method for each label set. The length of the vector needs to be the same as the number of columns in cand_lab_df. |

BPPARAM A BiocParallelParam object from the BiocParallel package. By specifying this argument, users can control over how to perform the parallel computing. Default is SerialParam which uses a single core.

## Value

calculate_CDI returns a data frame with 5 columns. The columns are Label_name (name of each label set), Cluster_method (clustering method), CDI-AIC, CDI-BIC, and N_cluster (number of clusters). Each row corresponds to one set of cell labels.

## References

SMartin Morgan, Valerie Obenchain, Michel Lang, Ryan Thompson and Nitesh Turaga (2021). doi:https://github.com/Bioconductor/BiocParallel

## Examples

```
ng <- 100; nc <- 100
set.seed(1)

# count matrix
X <- cbind(
 matrix(
  c(rnbinom(ng*nc/4, size = 1, mu = 0.1),
   rnbinom(ng*nc/4, size = 1, mu = 0.5)),
  nrow = ng,
  byrow = TRUE),
 matrix(
  c(rnbinom(ng*nc/4, size = 1, mu = 1),
   rnbinom(ng*nc/4, size = 1, mu = 0.5)),
  nrow = ng,
  byrow = TRUE))
colnames(X) <- paste0('c', seq_len(nc))
rownames(X) <- paste0('g', seq_len(ng))

# batch label
Batches <- rep(seq_len(2), nc/2)

# cell clustering labels
Method1_k2 <- rep(seq_len(2), c(nc/2,nc/2))
Method1_k3 <- sample(seq_len(3), nc, replace = TRUE)
label_df <- data.frame(
 Method1_k2 = Method1_k2,
 Method1_k3 = Method1_k3)

## select feature genes (see feature_gene_selection function)
selected_genes <- seq_len(30)

## calculate size factor (see size_factor function)
size_factor_vec <- rep(1, nc)

calculate_CDI(
```

```
 X = X[selected_genes, ],
 cand_lab_df = label_df,
 cell_size_factor = size_factor_vec,
 batch_label = Batches)

## Input: SingleCellExperiment object
library(SingleCellExperiment)
sim_sce <- SingleCellExperiment(
  list(count = X),
  colData = data.frame(
    Cell_name = colnames(X),
   batch = Batches),
  rowData = data.frame(
    Gene_name = rownames(X)))

calculate_CDI(
  X = sim_sce,
  feature_gene_index = selected_genes,
  cand_lab_df = label_df,
  cell_size_factor = size_factor_vec,
  count_slot = "count",
  batch_slot = "batch")

## Input: Seurat object
library(Seurat)
library(SeuratObject)
sim_seurat <- CreateSeuratObject(counts = as.data.frame(X))
sim_seurat <- AddMetaData(sim_seurat, colnames(X), "Cell_name")
sim_seurat <- AddMetaData(sim_seurat, Batches, "batch")

calculate_CDI(
 X = sim_seurat,
 feature_gene_index = selected_genes,
 cand_lab_df = label_df,
 cell_size_factor = size_factor_vec,
 count_slot = "counts",
 batch_slot = "batch")

## parallel computing
library(BiocParallel)
## single core
bp_object <- SerialParam()
## multi-cores
## bp_object  <- MulticoreParam(workers = 2)
calculate_CDI(
 X = X[selected_genes, ],
 cand_lab_df = label_df,
 cell_size_factor = size_factor_vec,
 batch_label = Batches,
 lrt_pval_threshold = 0.01,
 clustering_method = NULL,
 BPPARAM = bp_object)
```

## CDI *Clustering Deviation Index.*

### Description

**CDI** is a package for evaluating clustering results of single cell RNA-seq datasets.

## CDI_lineplot *Visualize CDI values via a lineplot*

### Description

This function visualize CDI outputs with a lineplot. The x-axis is for the number of clusters. The y-axis is for the CDI values. Different colors represent different clustering methods. The red triangle represents the optimal clustering result corresponding to the smallest CDI value. The purple star represents the CDI value for the benchmark (main) cell type label set. he brown star represents the CDI value for the benchmark sub-type label set.

### Usage

```
CDI_lineplot(
  cdi_dataframe,
  cdi_type,
  benchmark_celltype_cdi = NULL,
  benchmark_celltype_ncluster = NULL,
  benchmark_maintype_cdi = NULL,
  benchmark_maintype_ncluster = NULL,
  clustering_method = NULL,
  show_axis_names = TRUE,
  show_method_legend = TRUE
)
```

### Arguments

| | |
|---|---|
| `cdi_dataframe` | A data frame of CDI values. Each row represents one clustering method and the number of clusters combination. The columns include "Cluster_method", "CDI_AIC", "CDI_BIC", and "N_cluster". |
| `cdi_type` | A string indication the type of CDI. It can be either "CDI_AIC" or "CDI_BIC". |
| `benchmark_celltype_cdi` | A list of the output from the CalculateCDI function on the benchmark cell type label set. Default is null. |
| `benchmark_celltype_ncluster` | A number indicating the number of cell types in the benchmark cell type label set. Default is null. |

benchmark_maintype_cdi
: A list of the output from the CalculateCDI function on the benchmark main type label set. Default is null.

benchmark_maintype_ncluster
: A number indicating the number of cell types in the benchmark main type label set. Default is null.

clustering_method
: A vector of characters indicating the corresponding clustering method for each label set. If this is provided, the lineplot will be colored differently by different clustering methods. This color scheme can also be obtained if the column names in candidate_label_dataframe of CalculateCDI are provided with the form "[ClusteringMethod]_k[NumberOfClusters]" such as "KMeans_k5".

show_axis_names
: A bool value indicating whether the axis names should be shown or not in the plot.

show_method_legend
: A bool value indicating whether the legend of methods should be shown or not in the plot.

## Value

A ggplot object.

## Examples

```
## This data frame is generated here for demonstration only:
CDI_return <- data.frame(
Cluster_method = c("HC", "HC",  "HC", "KMeans", "KMeans", "KMeans"),
N_cluster = c(2,4,6,2,4,6),
CDI_AIC = c(150, 200, 250, 220, 160, 180),
CDI_BIC = c(170, 210, 280, 250, 180, 200))
CDI_lineplot(CDI_return, cdi_type = "CDI_AIC")
benchmark_cdi_return <- list(CDI_AIC = 150, CDI_BIC = 170)
benchmark_ncelltype <- 3
CDI_lineplot(CDI_return,
            cdi_type = "CDI_AIC",
            benchmark_celltype_cdi = benchmark_cdi_return,
            benchmark_celltype_ncluster = benchmark_ncelltype)
```

---

# contingency_heatmap *Heatmap of contingency table*

---

## Description

If the benchmark cell type label set is available, we can also compare one candidate label set (e.g. the optimal clustering label set selected by CDI) with the benchmark cell type labels. Here we provide the heatmap of contingency table for comparison. Each row represents a cell type in the benchmark label set, and each column represents a cluster in the clustering label set. Each rectangle is color-scaled by the proportion of the cells in the given cluster coming from the benchmark types. Each column sums to 1.

## Usage

```
contingency_heatmap(
  benchmark_label,
  candidate_label,
  proportion_size = 4,
  show_axis_names = TRUE,
  show_legend = TRUE,
  rename_candidate_clusters = FALSE,
  candidate_cluster_names = NULL
)
```

## Arguments

| | |
|---|---|
| benchmark_label | A vector of characters indicating the benchmark cell type label set of cells. |
| candidate_label | A vector of characters indicating the candidate clustering label set of cells. |
| proportion_size | A number indicating the label size of proportion values inside each rectangle. The label will not be shown if this parameter is set to be FALSE. |
| show_axis_names | A bool value indicating whether the axis names should be shown or not in the plot. |
| show_legend | A bool value indicating whether the legend of methods should be shown or not in the plot. |
| rename_candidate_clusters | A bool value indicating whether the candidate cluster names will be changed or not. If true, the candidate cluster names will be changed to the candidate_cluster_names. |
| candidate_cluster_names | A vector of characters indicating the candidate cluster names. The order of names will be the same as the order without customizing the candidate cluster names. For example, when rename_candidate_clusters is FALSE, and the output figure has x axis label (cluster1, cluster0). If rename_candidate_clusters is TRUE, and candidate_cluster_names is (c1, c2). That means cluster1 -> c1, and cluster0 -> c0. |

## Value

A ggplot object.

## Examples

```
contingency_heatmap(
 benchmark_label = c(rep("type_b", 45), rep("type_a", 145), rep("type_c", 10)),
 candidate_label = paste0('cluster', c(rep(0,10), rep(1, 180), rep(2, 10))))
contingency_heatmap(
 benchmark_label = c(rep("type_b", 45), rep("type_a", 145), rep("type_c", 10)),
 candidate_label = paste0('cluster', c(rep(0,10), rep(1, 180), rep(2, 10))),
```

```
 rename_candidate_clusters = TRUE,
 candidate_cluster_names = c('1', '2', '3'))
```

---

`feature_gene_selection` *Select feature genes*

---

## Description

This function selects a subset of feature genes that are expected to express differently across cell types before calculating CDI.

## Usage

```
feature_gene_selection(
  X,
  batch_label = NULL,
  count_slot = NULL,
  batch_slot = NULL,
  method = "wds",
  nfeature = 500,
  zp_threshold = 0.95
)
```

## Arguments

| | |
|---|---|
| `X` | The class of X can be "matrix", "Seurat" object, or "SingleCellExperiment" object. If X is a matrix, it should be a UMI count matrix where each row represents a gene, and each column represents a cell. If X is a Seurat object or SingleCellExperiment object, users need to specify where the count matrix, and batch labels are stored in count_slot and batch_slot, respectively. |
| `batch_label` | A vector of characters indicating the batch labels of the cells. The length of batch_label should be the same as the number of columns in the count matrix. The default value is NULL indicating that there is no batch information available. |
| `count_slot` | A string indicating the location of raw UMI count. For Seurat object, it is a slot in "RNA" of "assays"; For SingleCellExperiment object, it is a slot in "assays". Each row represents a gene, and each column represents a cell. The genes should be those before feature gene selection. |
| `batch_slot` | A string indicating the location of batch labels of cells. For Seurat object, it is a slot in meta.data; For SingleCellExperiment object, it is a slot in "colData". The default value is NULL indicating that there is no batch information available. |
| `method` | A character indicating the method used to select feature genes. "wds" (default) represent the working dispersion score proposed in our study; "vst" is the default method for feature gene selection in FindVariableFeatures function of Seurat package. |

| | |
|---|---|
| nfeature | An integer indicating the number of features to select. The default value is 500. |
| zp_threshold | A number of zero proportion threshold. The range should be (0,1). Genes with zero proportion greater than this value will be excluded before feature selection. |

## Value

A vector of indices of selected feature genes corresponding to the row indices of input count matrix.

## References

Stuart and Butler et al. Comprehensive Integration of Single-Cell Data. Cell (2019) [Seurat V3]

## Examples

```
## Simulate a matrix of 100 rows (genes), where the first 50 genes have
## different mean expression levels.
## Apply feature_gene_selection to select genes
ng <- 100; nc <- 100
set.seed(1)
X <- cbind(
 matrix(
  c(rnbinom(ng*nc/4, size = 1, mu = 0.1),
   rnbinom(ng*nc/4, size = 1, mu = 0.5)),
  nrow = ng,
  byrow = TRUE),
 matrix(
  c(rnbinom(ng*nc/4, size = 1, mu = 1),
   rnbinom(ng*nc/4, size = 1, mu = 0.5)),
  nrow = ng,
  byrow = TRUE))
colnames(X) <- paste0('c', seq_len(nc))
rownames(X) <- paste0('g', seq_len(ng))
Batches <- rep(seq_len(2), nc/2)

## Input: matrix
feature_gene_selection(
 X = X,
 batch_label = Batches,
 nfeature = 20,
 zp_threshold = 0.95)


## Input: SingleCellExperiment object
library(SingleCellExperiment)
sim_sce <- SingleCellExperiment(
  list(count = X),
  colData = data.frame(
   Cell_name = colnames(X),
  batch = Batches),
 rowData = data.frame(gene_name = rownames(X)))
selected_genes <- feature_gene_selection(
  X = sim_sce,
```

```
  count_slot = "count",
  batch_slot = "batch",
  nfeature = 20,
  zp_threshold = 0.95)

## Input: Seurat object
library(Seurat)
library(SeuratObject)
sim_seurat <- CreateSeuratObject(counts = as.data.frame(X))
sim_seurat <- AddMetaData(sim_seurat, colnames(X), "Cell_name")
sim_seurat <- AddMetaData(sim_seurat, Batches, "batch")
selected_genes <- feature_gene_selection(
 X = sim_seurat,
 count_slot = "counts",
 batch_slot = "batch",
 nfeature = 20,
 zp_threshold = 0.95)
```

---

one_batch_matrix *Simulated count matrix from one batch*

---

## Description

This matrix contains raw UMI count matrix simulated from the Negative Binomial(NB) mixture distribution. The code simulating this matrix can be found in data-raw/one_batch_simulation.R.

## Usage

```
data(one_batch_matrix)
```

## Format

one_batch_matrix is a matrix with 3,000 genes (rows) and 2,000 cells (columns). The rows are named with g1, g2, . . . , g3000 representing gene 1 to gene 3000; the columns are named with rc1, rc2, . . . , rc2000 representing raw count 1 to raw count 2000.

## Note

We simulated 3,000 genes and five group of cells with 400 cells each. Different cell groups have different mean and dispersion parameters in the NB distribution. One group of cells was treated as the baseline group with mean parameters generated from truncated Normal with mean and standard deviation 0.2, 0.1 and dispersion generated from truncated with mean and standard deviation 0.5, 0.1. For each of the other four groups, 30 genes had mean parameters shifted from the baseline group with log2 fold change 2.5. Dispersion parameters of feature genes were shifted by a normalized factor with mean 0 and standard deviation 0.05. The true cell type labels are in 'one_batch_matrix_celltype", and the cell clustering results generated from two clustering methods are in "one_batch_matrix_label_df".

## one_batch_matrix_celltype

*Cell type labels of simulated count matrix from one batch*

### Description

This is the true label set corresponds to the cells in the one_batch_matrix. The code simulating this vector can be found in data-raw/one_batch_simulation.R.

### Usage

```
data(one_batch_matrix_celltype)
```

### Format

A vector of 2,000 characters indicating the cell types of the reference cells. For each cell type, there are 400 cells with the name "type1-5".

## one_batch_matrix_label_df

*Clustering labels for simulated one-batch single-cell count matrix*

### Description

This is the label sets correspond to the cells in the one_batch_matrix. Cells were clustered by K-Means on first 200 PCs and Seurat v3.1.5 with 2000 (default) feature genes. The number of clusters are set to be 2 to 7. The code simulating this vector can be found in data-raw/one_batch_simulation.R.

### Usage

```
data(one_batch_matrix_label_df)
```

### Format

A data frame of 2,000 rows and 12 columns. Each row represents a cell. Each column represents a clustering method and the corresponding number of clusters.

`size_factor` *Size factor of each cell*

## Description

Different cells have different library sizes. This function calculates the size factor of each cell in the UMI count matrix to capture the variation in cell library size.

## Usage

```
size_factor(X, count_slot = NULL)
```

## Arguments

| | |
|---|---|
| `X` | The class of X can be "matrix", "Seurat" object, or "SingleCellExperiment" object. If X is a matrix, it should be a raw UMI count matrix where each row represents a gene, and each column represents a cell. The genes should be those before feature gene selection. If X is a Seurat object or SingleCellExperiment object, users need to specify where the count matrix is stored in count_slot. |
| `count_slot` | A string indicating the location of raw UMI count. For Seurat object, it is a slot in "RNA" of "assays"; For SingleCellExperiment object, it is a slot in "assays". Each row represents a gene, and each column represents a cell. The genes should be those before feature gene selection. |

## Value

A numeric vector indicating the size factor of the cells. This should be one of the inputs of the function calculate_CDI.

## Examples

```
ng <- 100; nc <- 100
set.seed(1)
X <- cbind(
 matrix(
  c(rnbinom(ng*nc/4, size = 1, mu = 0.1),
   rnbinom(ng*nc/4, size = 1, mu = 0.5)),
  nrow = ng,
  byrow = TRUE),
 matrix(
  c(rnbinom(ng*nc/4, size = 1, mu = 1),
   rnbinom(ng*nc/4, size = 1, mu = 0.5)),
  nrow = ng,
  byrow = TRUE))
colnames(X) <- paste0('c', seq_len(nc))
rownames(X) <- paste0('g', seq_len(ng))

## Input: matrix
```

```
cell_size <- size_factor(X = X)

## Input: SingleCellExperiment object
library(SingleCellExperiment)
sim_sce <- SingleCellExperiment(
  list(count = X),
  colData = data.frame(Cell_name = colnames(X)),
  rowData = data.frame(Gene_name = rownames(X)))
cell_size <- size_factor(X = sim_sce, count_slot = "count")

## Input: Seurat object
library(Seurat)
library(SeuratObject)
sim_seurat <- CreateSeuratObject(counts = as.data.frame(X))
sim_seurat <- AddMetaData(sim_seurat, colnames(X), "Cell_name")
cell_size <- size_factor(X = sim_seurat, count_slot = "counts")
```

---

`two_batch_matrix` *Simulated count matrix from two batches*

---

## Description

This matrix contains raw UMI count matrix simulated from the Negative Binomial(NB) mixture distribution. The code simulating this vector can be found in data-raw/two_batch_simulation.R.

## Usage

```
data(two_batch_matrix)
```

## Format

A matrix with 3,000 genes (rows) and 2,000 cells (columns). The cells are from 5 cell type, and each cell type contains 400 cells. Within each cell type, 200 of cells with odd indices are from batch 1, and the other cells with even indices are from batch 2. See Note for details of simulation setting.

## Note

We first simulated one batch of cells with 1,000 cells in total. Within the batch, we simulated 3,000 genes and five group of cells with 400 cells each. One group of cells was treated as the baseline group. The mean parameters are generated from truncated Normal with mean and standard deviation 0.2, 0.1 and dispersion generated from truncated with mean and standard deviation 0.5, 0.1. For each of the other four groups, 30 genes had mean parameters shifted from the baseline group with log2 fold change 2.5. Dispersion parameters of feature genes were shifted by a normalized factor with mean 0 and standard deviation 0.05. To generate the second batch of cells, we shifted the mean parameters of randomly selected 10

## two_batch_matrix_batch

*Batch labels of simulated count matrix from two batches*

### Description

This is the batch label set corresponds to the cells in the two_batch_matrix. The code simulating this vector can be found in data-raw/two_batch_simulation.R.

### Usage

```
data(two_batch_matrix_batch)
```

### Format

A vector of 2,000 characters indicating the batch labels of cells in two_batch_matrix. For each batch, there are 1,000 cells.

## two_batch_matrix_celltype

*Cell type labels of simulated count matrix from two batches*

### Description

This is the true label set corresponds to the cells in the two_batch_matrix.

### Usage

```
data(two_batch_matrix_celltype)
```

### Format

A vector of 2,000 characters indicating the cell types of cells in two_batch_matrix. For each cell type, there are 400 cells with the name "type1-5".

two_batch_matrix_label_df *Clustering labels for simulated two-batch single-cell count matrix*

## Description

his is the label sets correspond to the cells in the two_batch_matrix. Cells in two batches of the two_batch_matrix was first integrated by Seurat v4. The dataset after integration was then clustered by K-Means on first 200 PCs and Seurat with 2000 (default) feature genes. The number of clusters are set to be 2 to 10. The code simulating this vector can be found in data-raw/two_batch_simulation.R.

## Usage

```
data(two_batch_matrix_label_df)
```

## Format

A data frame of 2,000 rows and 18 columns. Each row represents a cell. Each column represents a clustering method and the corresponding number of clusters.

# Index

∗ **datasets**
one_batch_matrix, 11
one_batch_matrix_celltype, 12
one_batch_matrix_label_df, 12
two_batch_matrix, 14
two_batch_matrix_batch, 15
two_batch_matrix_celltype, 15
two_batch_matrix_label_df, 16

BiocParallelParam, *4*

calculate_CDI, 2
CDI, 6
CDI_lineplot, 6
contingency_heatmap, 7

feature_gene_selection, 9

one_batch_matrix, 11
one_batch_matrix_celltype, 12
one_batch_matrix_label_df, 12

SerialParam, *4*
size_factor, 13

two_batch_matrix, 14
two_batch_matrix_batch, 15
two_batch_matrix_celltype, 15
two_batch_matrix_label_df, 16