

# Package: biclustermd (via r-universe)

August 28, 2024

**Type** Package

**Title** Biclustering with Missing Data

**Version** 0.2.3

**Maintainer** John Reisner <johntreisner@gmail.com>

**Description** Biclustering is a statistical learning technique that simultaneously partitions and clusters rows and columns of a data matrix. Since the solution space of biclustering is infeasible to completely search with current computational mechanisms, this package uses a greedy heuristic. The algorithm featured in this package is, to the best of our knowledge, the first biclustering algorithm to work on data with missing values. Li, J., Reisner, J., Pham, H., Olafsson, S., and Vardeman, S. (2020) Biclustering with Missing Data. Information Sciences, 510, 304–316.

**URL** <https://github.com/jreisner/biclustermd>

**BugReports** <https://github.com/jreisner/biclustermd/issues>

**Depends** ggplot2 (>= 3.0.0), R (>= 3.5.0), tidyr (>= 0.8.1)

**Imports** biclust (>= 2.0.1), doParallel (>= 1.0.14), dplyr (>= 0.7.6),  
foreach (>= 1.4.4), magrittr (>= 1.5), nycflights13 (>= 1.0.0),  
phyclust (>= 0.1-24)

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Roxygen** list(markdown = TRUE)

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**Repository** <https://jreisner.r-universe.dev>

**RemoteUrl** <https://github.com/jreisner/biclustermd>

**RemoteRef** HEAD

**RemoteSha** 975af747fce1adf46feba4ec65c05debf6cc8b17

## Contents

biclustermd-package	2
as.Biclust	3
autoplot.biclustermd	4
autoplot.biclustermd_sim	5
autoplot.biclustermd_sse	6
biclustermd	7
binary_vector_gen	9
cell_heatmap	10
cell_mse	11
cluster_iteration_sum_sse	11
col.names	12
col.names.biclustermd	12
col_cluster_names	13
compare_biclusters	14
fill_empties_P	14
fill_empties_Q	15
format_partition	16
gather.biclustermd	16
jaccard_similarity	17
mse_heatmap	18
partition_gen	19
partition_gen_by_p	19
part_matrix_to_vector	20
position_finder	20
print.biclustermd	21
reorder_biclust	21
rep_biclustermd	22
results_heatmap	24
row.names.biclustermd	25
row_cluster_names	25
runtimes	26
synthetic	27
tune_biclustermd	27
<b>Index</b>	<b>30</b>

---

biclustermd-package    *biclustermd: A package to bicluster data with missing values*

---

## Description

The main function is `biclustermd()`. Results can be plotted with `autoplot()` and `as.Biclust()` converts results to Biclust objects.

---

as.Biclust	<i>Convert a biclustermd object to a Biclust object</i>
------------	---

---

**Description**

Convert a biclustermd object to a Biclust object

**Usage**

```
as.Biclust(object)
```

**Arguments**

object            The biclustermd object to convert to a Biclust object

**Value**

Returns an object of class Biclust.

**Examples**

```
data("synthetic")

bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,
  miss_val = mean(synthetic, na.rm = TRUE),
  miss_val_sd = sd(synthetic, na.rm = TRUE),
  col_min_num = 2, row_min_num = 2,
  col_num_to_move = 1, row_num_to_move = 1,
  max.iter = 10)

bc

as.Biclust(bc)

# biclust::drawHeatmap won't work since it doesn't exclude NAs
## Not run: biclust::drawHeatmap(synthetic, as.Biclust(bc), 6)

# bicluster 6 is in the top right-hand corner here:
autoplot(bc)
# compare with biclust::drawHeatmap2:
biclust::drawHeatmap2(synthetic, as.Biclust(bc), 6)

# bicluster 3 is in the bottom right-hand corner here:
autoplot(bc)
# compare with biclust::drawHeatmap2:
biclust::drawHeatmap2(synthetic, as.Biclust(bc), 3)
```

---

autoplot.biclustermd *Make a heatmap of sparse biclustering results*

---

## Description

Make a heatmap of sparse biclustering results

## Usage

```
## S3 method for class 'biclustermd'
autoplot(
  object,
  axis.text = NULL,
  reorder = FALSE,
  transform_colors = FALSE,
  c = 1/6,
  cell_alpha = 1/5,
  col_clusts = NULL,
  row_clusts = NULL,
  ...
)
```

## Arguments

object	An object of class "biclustermd".
axis.text	A character vector specifying for which axes text should be drawn. Can be any of "x", "col" for columns, "y", "row" for rows, or any combination of the four. By default this is NULL; no axis text is drawn.
reorder	A logical. If TRUE, heatmap will be sorted according to the cell-average matrix, A.
transform_colors	If equals TRUE then the data is scaled by c and run through a standard normal cdf before plotting. If FALSE (default), raw data values are used in the heat map.
c	Value to scale the data by before running it through a standard normal CDF. Default is 1/6.
cell_alpha	A scalar defining the transparency of shading over a cell and by default this equals 1/5. The color corresponds to the cell mean.
col_clusts	A vector of column cluster indices to display. If NULL (default), all are displayed.
row_clusts	A vector of row cluster indices to display. If NULL (default), all are displayed.
...	Arguments to be passed to geom_vline() and geom_hline().

## Value

An object of class ggplot.

**Examples**

```

data("synthetic")

bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,
  miss_val = mean(synthetic, na.rm = TRUE),
  miss_val_sd = sd(synthetic, na.rm = TRUE),
  col_min_num = 2, row_min_num = 2,
  col_num_to_move = 1, row_num_to_move = 1,
  max.iter = 10)

bc
autoplot(bc)

autoplot(bc, axis.text = c('x', 'row')) +
  ggplot2::scale_fill_distiller(palette = "Spectral", na.value = "white")

# Complete shading
autoplot(bc, axis.text = c('col', 'row'), cell_alpha = 1)

# Transformed values and no shading
autoplot(bc, transform_colors = TRUE, c = 1/20, cell_alpha = 0)

# Focus on row cluster 1 and column cluster 2
autoplot(bc, col_clusts = 2, row_clusts = 1)

```

---

```
autoplot.biclustermd_sim
```

*Plot similarity measures between two consecutive biclusterings.*

---

**Description**

Creates a ggplot of the three similarity measures used in `biclustermd::bicluster()` for both row and column dimensions.

**Usage**

```

## S3 method for class 'biclustermd_sim'
autoplot(object, similarity = NULL, facet = TRUE, ncol = NULL, ...)

```

**Arguments**

<code>object</code>	Object of class "biclustermd_sim"
<code>similarity</code>	A character vector indicating which similarity measure to plot. Can be any of "Rand", "HA", "Jaccard", or "used". If "used", plot only the measure used as the stopping condition in the algorithm). By default (NULL) all three are plotted. When plotted, the used measure will have an asterisk.
<code>facet</code>	If TRUE (default), each similarity measure will be in its own plot. if FALSE, all three similarity measures for rows and columns are given in one plot.

ncol            If faceting, the number of columns to arrange the plots in.  
 ...            Arguments to pass to `ggplot2::geom_point()`

**Value**

A ggplot object.

**Examples**

```
data("synthetic")

bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,
  miss_val = mean(synthetic, na.rm = TRUE),
  miss_val_sd = sd(synthetic, na.rm = TRUE),
  col_min_num = 2, row_min_num = 2,
  col_num_to_move = 1, row_num_to_move = 1,
  max.iter = 10)

bc
autoplot(bc$Similarities, ncol = 1)
```

---

autoplot.biclustermd\_sse

*Plot sums of squared errors (SSEs) consecutive biclustering iterations.*

---

**Description**

Creates a ggplot of the decrease in SSE recorded in `biclustermd::bicluster()`.

**Usage**

```
## S3 method for class 'biclustermd_sse'
autoplot(object, ...)
```

**Arguments**

object            Object of class "biclustermd\_sse" with columns "Iteration" and "SSE"  
 ...            Arguments to pass to `ggplot2::geom_point()`

**Value**

A ggplot object.

**Examples**

```

data("synthetic")

bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,
  miss_val = mean(synthetic, na.rm = TRUE),
  miss_val_sd = sd(synthetic, na.rm = TRUE),
  col_min_num = 2, row_min_num = 2,
  col_num_to_move = 1, row_num_to_move = 1,
  max.iter = 10)

bc
autoplot(bc$$SSE)

```

---

biclustermd

*Bicluster data with non-random missing values*


---

**Description**

Bicluster data with non-random missing values

**Usage**

```

biclustermd(
  data,
  row_clusters = floor(sqrt(nrow(data))),
  col_clusters = floor(sqrt(ncol(data))),
  miss_val = mean(data, na.rm = TRUE),
  miss_val_sd = 1,
  similarity = "Rand",
  row_min_num = floor(nrow(data)/row_clusters),
  col_min_num = floor(ncol(data)/col_clusters),
  row_num_to_move = 1,
  col_num_to_move = 1,
  row_shuffles = 1,
  col_shuffles = 1,
  max.iter = 100,
  verbose = FALSE
)

```

**Arguments**

data	Dataset to bicluster. Must to be a data matrix with only numbers and missing values in the data set. It should have row names and column names.
row_clusters	The number of clusters to partition the rows into. The default is <code>floor(sqrt(nrow(data)))</code> .
col_clusters	The number of clusters to partition the columns into. The default is <code>floor(sqrt(ncol(data)))</code> .
miss_val	Value or function to put in empty cells of the prototype matrix. If a value, a random normal variable with <code>sd = miss_val_sd</code> is used each iteration. By default, this equals the mean of data.

miss_val_sd	Standard deviation of the normal distribution miss_val follows if miss_val is a number. By default this equals 1.
similarity	The metric used to compare two successive clusterings. Can be "Rand" (default), "HA" for the Hubert and Arabie adjusted Rand index or "Jaccard". See <a href="#">RRand</a> for details.
row_min_num	Minimum row prototype size in order to be eligible to be chosen when filling an empty row prototype. Default is $\text{floor}(\text{nrow}(\text{data}) / \text{row\_clusters})$ .
col_min_num	Minimum column prototype size in order to be eligible to be chosen when filling an empty row prototype. Default is $\text{floor}(\text{ncol}(\text{data}) / \text{col\_clusters})$ .
row_num_to_move	Number of rows to remove from the sampled prototype to put in the empty row prototype. Default is 1.
col_num_to_move	Number of columns to remove from the sampled prototype to put in the empty column prototype. Default is 1.
row_shuffles	Number of times to shuffle rows in each iteration. Default is 1.
col_shuffles	Number of times to shuffle columns in each iteration. Default is 1.
max.iter	Maximum number of iterations to let the algorithm run for.
verbose	Logical. If TRUE, will report progress.

### Value

A list of class `biclustermd`:

params	a list of all arguments passed to the function, including defaults.
data	the inputted two way table of data.
P0	the initial column partition matrix.
Q0	the initial row partition matrix.
InitialSSE	the SSE of the original partitioning.
P	the final column partition matrix.
Q	the final row partition matrix.
SSE	a matrix of class <code>biclustermd_sse</code> detailing the SSE recorded at the end of each iteration.
Similarities	a data frame of class <code>biclustermd_sim</code> detailing the value of row and column similarity measures recorded at the end of each iteration. Contains information for all three similarity measures. This carries an attribute "used" which provides the similarity measure used as the stopping condition for the algorithm.
iteration	the number of iterations the algorithm ran for, whether <code>max.iter</code> was reached or convergence was achieved.
A	the final prototype matrix which gives the average of each bicluster.

### References

Li, J., Reisner, J., Pham, H., Olafsson, S., and Vardeman, S. (2020) *Biclustering with Missing Data. Information Sciences*, 510, 304–316.



**See Also**[rep\\_biclustermd](#), [tune\\_biclustermd](#)**Examples**

```
data("synthetic")
# default parameters
bc <- biclustermd(synthetic)
bc
autoplot(bc)

# providing the true number of row and column clusters
bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2)
bc
autoplot(bc)

# an example with the nycflights13::flights dataset
library(nycflights13)
data("flights")

library(dplyr)
flights_bcd <- flights %>%
  select(month, dest, arr_delay)

flights_bcd <- flights_bcd %>%
  group_by(month, dest) %>%
  summarise(mean_arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  spread(dest, mean_arr_delay) %>%
  as.data.frame()

rownames(flights_bcd) <- flights_bcd$month
flights_bcd <- as.matrix(flights_bcd[, -1])

flights_bc <- biclustermd(data = flights_bcd, col_clusters = 6, row_clusters = 4,
  row_min_num = 3, col_min_num = 5,
  max.iter = 20, verbose = TRUE)

flights_bc
```

---

binary_vector_gen	<i>Make a binary vector with all values equal to zero except for one</i>
-------------------	--

---

**Description**

Make a binary vector with all values equal to zero except for one

**Usage**

```
binary_vector_gen(n, i)
```

**Arguments**

n                    Desired vector length.  
i                    Index whose value is one.

**Value**

A vector

---

cell\_heatmap                    *Make a heat map of bicluster cell sizes.*

---

**Description**

Make a heat map of bicluster cell sizes.

**Usage**

```
cell_heatmap(x, ...)
```

**Arguments**

x                    An object of class biclustermd.  
...                   Arguments to pass to geom\_tile()

**Examples**

```
data("synthetic")  
  
bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,  
                  miss_val = mean(synthetic, na.rm = TRUE),  
                  miss_val_sd = sd(synthetic, na.rm = TRUE),  
                  col_min_num = 2, row_min_num = 2,  
                  col_num_to_move = 1, row_num_to_move = 1,  
                  max.iter = 10)  
  
cell_heatmap(bc)  
  
cell_heatmap(bc) + ggplot2::scale_fill_viridis_c()
```

---

cell_mse	<i>Make a data frame containing the MSE for each bicluster cell</i>
----------	---

---

**Description**

Make a data frame containing the MSE for each bicluster cell

**Usage**

```
cell_mse(x)
```

**Arguments**

x                    An object of class biclustermd.

**Value**

A data frame giving the row cluster, column cluster, the number of data points in each row and column cluster, the number of data points missing in the cell, and the cell MSE.

**Examples**

```
data("synthetic")
bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,
                 miss_val = mean(synthetic, na.rm = TRUE),
                 miss_val_sd = sd(synthetic, na.rm = TRUE),
                 col_min_num = 2, row_min_num = 2,
                 col_num_to_move = 1, row_num_to_move = 1,
                 max.iter = 10)
cell_mse(bc)
```

---

cluster_iteration_sum_sse	<i>Calculate the sum cluster SSE in each iteration</i>
---------------------------	--

---

**Description**

Calculate the sum cluster SSE in each iteration

**Usage**

```
cluster_iteration_sum_sse(data, P, Q)
```

**Arguments**

data	The data being biclustered. Must to be a data matrix with only numbers and missing values in the data set. It should have row names and column names.
P	Matrix for column prototypes.
Q	Matrix for row prototypes.

**Value**

The SSE for the parameters specified.

---

col.names	<i>A generic to gather column names</i>
-----------	---

---

**Description**

A generic to gather column names

**Usage**

```
col.names(x)
```

**Arguments**

x	an object to retrieve column names from
---	---

---

col.names.biclustermd	<i>Get data matrix column names and their corresponding column cluster membership</i>
-----------------------	---

---

**Description**

Get data matrix column names and their corresponding column cluster membership

**Usage**

```
## S3 method for class 'biclustermd'
col.names(x)
```

**Arguments**

x	and object of class biclustermd
---	---------------------------------

**Value**

a data frame with column names of the shuffled matrix and corresponding column cluster names.

## Examples

```
data("synthetic")
# default parameters
bc <- biclustermd(synthetic)
bc
col.names(bc)
# this is a simplified version of the output for gather(bc):
library(dplyr)
gather(bc) %>% distinct(col_cluster, col_name)
```

---

col_cluster_names	<i>Get column names in each column cluster</i>
-------------------	--

---

## Description

Get column names in each column cluster

## Usage

```
col_cluster_names(x, data)
```

## Arguments

x	Biclustering object to extract column cluster designation from
data	Data that contains the column names

## Value

A data frame with two columns: cluster corresponds to the column cluster and name gives the column names in each cluster.

## Examples

```
data("synthetic")
rownames(synthetic) <- letters[1:nrow(synthetic)]
colnames(synthetic) <- letters[1:ncol(synthetic)]
bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,
  miss_val = mean(synthetic, na.rm = TRUE),
  miss_val_sd = sd(synthetic, na.rm = TRUE),
  col_min_num = 2, row_min_num = 2,
  col_num_to_move = 1, row_num_to_move = 1,
  max.iter = 10)
bc
```

---

compare_biclusters	<i>Compare two biclusterings or a pair of partition matrices</i>
--------------------	--

---

**Description**

Compare two biclusterings or a pair of partition matrices

**Usage**

```
compare_biclusters(bc1, bc2)
```

**Arguments**

bc1	the first biclustering or partition matrix. Must be either of class <code>biclustermd</code> or <code>matrix</code> .
bc2	the second biclustering or partition matrix. Must be either of class <code>biclustermd</code> or <code>matrix</code> .

**Value**

If comparing a pair of biclusterings, a list containing the column similarity indices and the row similarity indices, in that order. If a pair of matrices, a vector of similarity indices.

**Examples**

```
data("synthetic")
bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2)
bc2 <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2)

# compare the two biclusterings
compare_biclusters(bc, bc2)

# determine the similarity between initial and final row clusterings
compare_biclusters(bc$Q0, bc$Q)
```

---

fill_empties_P	<i>Randomly select a column prototype to fill an empty column prototype with</i>
----------------	--

---

**Description**

Randomly select a column prototype to fill an empty column prototype with

**Usage**

```
fill_empties_P(data, obj, col_min_num = 10, col_num_to_move = 5)
```

**Arguments**

data	The data being biclustered. Must to be a data matrix with only numbers and missing values in the data set. It should have row names and column names.
obj	A matrix for column clusters, typically named P.
col_min_num	Minimum column prototype size in order to be eligible to be chosen when filling an empty column prototype. Default is 10.
col_num_to_move	Number of columns to remove from the sampled prototype to put in the empty column prototype. Default is 5.

**Value**

A matrix for column clusters, i.e., a P matrix.

---

fill_empties_Q	<i>Randomly select a row prototype to fill an empty row prototype with</i>
----------------	--

---

**Description**

Randomly select a row prototype to fill an empty row prototype with

**Usage**

```
fill_empties_Q(data, obj, row_min_num = 10, row_num_to_move = 5)
```

**Arguments**

data	The data being biclustered. Must to be a data matrix with only numbers and missing values in the data set. It should have row names and column names.
obj	A matrix for row clusters, typically named Q
row_min_num	Minimum row prototype size in order to be eligible to be chosen when filling an empty row prototype. Default is 10.
row_num_to_move	Number of rows to remove from the sampled prototype to put in the empty row prototype. Default is 5.

**Value**

A matrix for row clusters, i.e., a Q matrix.

---

<code>format_partition</code>	<i>Format a partition matrix</i>
-------------------------------	----------------------------------

---

**Description**

Formats a partition matrix so that subsets in a partition will be ordered by the value of the smallest in each subset

**Usage**

```
format_partition(P1)
```

**Arguments**

P1                    A partition matrix.

**Value**

A formatted partition matrix.

---

<code>gather.biclustermd</code>	<i>Gather a biclustermd object</i>
---------------------------------	------------------------------------

---

**Description**

Gather a biclustermd object

**Usage**

```
## S3 method for class 'biclustermd'  
gather(  
  data,  
  key = NULL,  
  value = NULL,  
  ...,  
  na.rm = FALSE,  
  convert = FALSE,  
  factor_key = FALSE  
)
```



**Arguments**

data	a biclustermmd object to gather.
key	unused; included for consistency with tidyr generic
value	unused; included for consistency with tidyr generic
...	unused; included for consistency with tidyr generic
na.rm	unused; included for consistency with tidyr generic
convert	unused; included for consistency with tidyr generic
factor_key	unused; included for consistency with tidyr generic

**Value**

A data frame containing the row names and column names of both the two-way table of data biclustered and the cell-average matrix.

**Examples**

```
data("synthetic")

bc <- biclustermmd(synthetic, col_clusters = 3, row_clusters = 2,
  miss_val = mean(synthetic, na.rm = TRUE),
  miss_val_sd = sd(synthetic, na.rm = TRUE),
  col_min_num = 2, row_min_num = 2,
  col_num_to_move = 1, row_num_to_move = 1,
  max.iter = 10)

gather(bc)

# bicluster 6 is in the top right-hand corner here:
autoplot(bc)

# bicluster 3 is in the bottom right-hand corner here:
autoplot(bc)
```

---

jaccard\_similarity      *Compute the Jaccard similarity coefficient for two clusterings*

---

**Description**

Compute the Jaccard similarity coefficient for two clusterings

**Usage**

```
jaccard_similarity(clus1, clus2)
```

**Arguments**

clus1	vector giving the first set of clusters
clus2	vector giving the second set of clusters

**Value**

a numeric

**References**

Milligan, G.W. and Cooper, M. C. (1986) *A study of the comparability of external criteria for hierarchical cluster analysis. Multivariate Behavioral Research, 21, 441-458.*

---

mse\_heatmap

*Make a heatmap of cell MSEs*

---

**Description**

Make a heatmap of cell MSEs

**Usage**

```
mse_heatmap(x, ...)
```

**Arguments**

x                    An object of class biclustermd.  
...                   Arguments to pass to geom\_tile()

**Value**

A ggplot object.

**Examples**

```
data("synthetic")
bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,
  miss_val = mean(synthetic, na.rm = TRUE),
  miss_val_sd = sd(synthetic, na.rm = TRUE),
  col_min_num = 2, row_min_num = 2,
  col_num_to_move = 1, row_num_to_move = 1,
  max.iter = 10)

mse_heatmap(bc)

mse_heatmap(bc) + ggplot2::scale_fill_viridis_c()
```

---

partition_gen	<i>Generate an initial, random partition matrix with <math>N</math> objects into <math>K</math> subsets/groups.</i>
---------------	---

---

**Description**

This function is used to randomly generate a partition matrix and assign rows or columns to prototypes. Must be the case that  $N > K$ .

**Usage**

```
partition_gen(N, K)
```

**Arguments**

N	Number of objects/rows in a partition matrix
K	Desired number of partitions

**Value**

A partition matrix.

---

partition_gen_by_p	<i>Create a partition matrix with a partition vector <math>p</math></i>
--------------------	---

---

**Description**

Create a partition matrix with a partition vector  $p$

**Usage**

```
partition_gen_by_p(N, K, p)
```

**Arguments**

N	Rows in a partition matrix
K	Number of prototypes to create
p	Integer vector containing the cluster each row in a partition matrix is to be assigned to.

**Value**

A partition matrix.

---

part\_matrix\_to\_vector *Convert a partition matrix to a vector*

---

**Description**

For each row in a partition matrix, this function gets the column index for which the row is equal to one. That is, for row  $i$ , this function returns the index of the row entry that is equal to one.

**Usage**

```
part_matrix_to_vector(P0)
```

**Arguments**

P0                    A partition matrix

**Value**

An integer vector

---

position\_finder        *Find the index of the first nonzero value in a vector*

---

**Description**

Find the index of the first nonzero value in a vector

**Usage**

```
position_finder(vec)
```

**Arguments**

vec                    A binary vector

**Value**

Position of the first nonzero value in a vector.

---

`print.biclustermd`      *Print an object of class biclustermd*

---

### **Description**

Print an object of class biclustermd

### **Usage**

```
## S3 method for class 'biclustermd'  
print(x, ...)
```

### **Arguments**

`x`                    a biclustermd object.  
`...`                arguments passed to or from other methods

---

`reorder_biclust`      *Reorder a bicluster object for making a heat map*

---

### **Description**

Reorder a bicluster object for making a heat map

### **Usage**

```
reorder_biclust(x)
```

### **Arguments**

`x`                    A bicluster object.

### **Value**

A list containing the two partition matrices used by `gg_bicluster`.

---

rep\_biclustermd      *Repeat a biclustering to achieve a minimum SSE solution*

---

### Description

Repeat a biclustering to achieve a minimum SSE solution

### Usage

```
rep_biclustermd(
  data,
  nrep = 10,
  parallel = FALSE,
  ncores = 2,
  col_clusters = floor(sqrt(ncol(data))),
  row_clusters = floor(sqrt(nrow(data))),
  miss_val = mean(data, na.rm = TRUE),
  miss_val_sd = 1,
  similarity = "Rand",
  row_min_num = 5,
  col_min_num = 5,
  row_num_to_move = 1,
  col_num_to_move = 1,
  row_shuffles = 1,
  col_shuffles = 1,
  max.iter = 100
)
```

### Arguments

data	Dataset to bicluster. Must to be a data matrix with only numbers and missing values in the data set. It should have row names and column names.
nrep	The number of times to repeat the biclustering. Default 10.
parallel	Logical indicating if the user would like to utilize the foreach parallel backend. Default is FALSE.
ncores	The number of cores to use if parallel computing. Default 2.
col_clusters	The number of clusters to partition the columns into.
row_clusters	The number of clusters to partition the rows into.
miss_val	Value or function to put in empty cells of the prototype matrix. If a value, a random normal variable with sd = miss_val_sd is used each iteration.
miss_val_sd	Standard deviation of the normal distribution miss_val follows if miss_val is a number. By default this equals 1.
similarity	The metric used to compare two successive clusterings. Can be "Rand" (default), "HA" for the Hubert and Arabie adjusted Rand index or "Jaccard". See <a href="#">RRand</a> and for details.

row_min_num	Minimum row prototype size in order to be eligible to be chosen when filling an empty row prototype. Default is 5.
col_min_num	Minimum column prototype size in order to be eligible to be chosen when filling an empty row prototype. Default is 5.
row_num_to_move	Number of rows to remove from the sampled prototype to put in the empty row prototype. Default is 1.
col_num_to_move	Number of columns to remove from the sampled prototype to put in the empty column prototype. Default is 1.
row_shuffles	Number of times to shuffle rows in each iteration. Default is 1.
col_shuffles	Number of times to shuffle columns in each iteration. Default is 1.
max.iter	Maximum number of iterations to let the algorithm run for.

**Value**

A list of the minimum SSE biclustering, a vector containing the final SSE of each repeat, and the time it took the function to run.

**References**

Li, J., Reisner, J., Pham, H., Olafsson, S., and Vardeman, S. (2019) *Biclustering for Missing Data. Information Sciences, Submitted*

**See Also**

[biclustermd](#), [tune\\_biclustermd](#)

**Examples**

```
data("synthetic")

# 20 repeats without parallelization
repeat_bc <- rep_biclustermd(synthetic, nrep = 20,
                             col_clusters = 3, row_clusters = 2,
                             miss_val = mean(synthetic, na.rm = TRUE),
                             miss_val_sd = sd(synthetic, na.rm = TRUE),
                             col_min_num = 2, row_min_num = 2,
                             col_num_to_move = 1, row_num_to_move = 1,
                             max.iter = 10)

repeat_bc
autoplot(repeat_bc$best_bc)
plot(repeat_bc$rep_sse, type = 'b', pch = 20)
repeat_bc$runtime

# 20 repeats with parallelization over 2 cores
repeat_bc <- rep_biclustermd(synthetic, nrep = 20, parallel = TRUE, ncores = 2,
                             col_clusters = 3, row_clusters = 2,
                             miss_val = mean(synthetic, na.rm = TRUE),
```

```

miss_val_sd = sd(synthetic, na.rm = TRUE),
col_min_num = 2, row_min_num = 2,
col_num_to_move = 1, row_num_to_move = 1,
max.iter = 10)
repeat_bc$runtime

```

---

results\_heatmap      *Make a heatmap of sparse biclustering results*

---

### Description

Make a heatmap of sparse biclustering results

### Usage

```

results_heatmap(
  x,
  reorder = FALSE,
  transform_colors = FALSE,
  c = 1/6,
  cell_alpha = 1/5,
  col_clusts = NULL,
  row_clusts = NULL,
  ...
)

```

### Arguments

x	A biclustermmd object.
reorder	A logical. If TRUE, heatmap will be sorted according to the cell-average matrix, A.
transform_colors	If equals TRUE then the data is scaled by c and run through a standard normal cdf before plotting. If FALSE (default), raw data values are used in the heat map.
c	Value to scale the data by before running it through a standard normal CDF. Default is 1/6.
cell_alpha	A scalar defining the transparency of shading over a cell and by default this equals 1/5. The color corresponds to the cell mean.
col_clusts	A vector of column cluster indices to display. If NULL (default), all are displayed.
row_clusts	A vector of row cluster indices to display. If NULL (default), all are displayed.
...	Arguments to be passed to geom_vline() and geom_hline().

### Value

An object of class ggplot.



---

row.names.biclustermd *Get data matrix row names and their corresponding row cluster membership*

---

**Description**

Get data matrix row names and their corresponding row cluster membership

**Usage**

```
## S3 method for class 'biclustermd'  
row.names(x)
```

**Arguments**

x                      and object of class biclustermd

**Value**

a data frame with row names of the shuffled matrix and corresponding row cluster names.

**Examples**

```
data("synthetic")  
# default parameters  
bc <- biclustermd(synthetic)  
bc  
row.names(bc)  
# this is a simplified version of the output for gather(bc):  
library(dplyr)  
gather(bc) %>% distinct(row_cluster, row_name)
```

---

row\_cluster\_names      *Get row names in each row cluster*

---

**Description**

Get row names in each row cluster

**Usage**

```
row_cluster_names(x, data)
```

**Arguments**

x                      Biclustering object to extract row cluster designation from  
data                    Data that contains the row names

**Value**

A data frame with two columns: `cluster` corresponds to the row cluster and `name` gives the row names in each cluster.

**Examples**

```
data("synthetic")
rownames(synthetic) <- letters[1:nrow(synthetic)]
colnames(synthetic) <- letters[1:ncol(synthetic)]
bc <- biclustermd(synthetic, col_clusters = 3, row_clusters = 2,
                 miss_val = mean(synthetic, na.rm = TRUE),
                 miss_val_sd = sd(synthetic, na.rm = TRUE),
                 col_min_num = 2, row_min_num = 2,
                 col_num_to_move = 1, row_num_to_move = 1,
                 max.iter = 10)
bc
```

---

runtimes

*Algorithm run time data*


---

**Description**

This dataset stems from the R journal article introducing `biclustermd` to R users. It describes the data attributes and run time for varying data sizes and structures.

**Usage**

```
runtimes
```

**Format**

An object of class `data.frame` with 2400 rows and 13 columns.

**Details**

A data frame of 2400 rows and 13 variables (defined range, inclusive):

**combination\_no** Unique identifier of a combination of parameters.

**rows** Number of rows in the data matrix. (50, 1500)

**cols** Number of columns in the data matrix. (50, 1500)

**N** Product of the dimensions of the data. (2500, 2250000)

**row\_clusts** Number of clusters to partition the rows into. (4, 300)

**col\_clusts** Number of clusters to partition the columns into. (4, 300)

**avg\_row\_clust\_size** Average row cluster size. `rows / row_clusts`

**avg\_col\_clust\_size** Average column cluster size. `cols / col_clusts`

**sparsity** Percent of data values which are missing.

**user.self** CPU time used executing instructions to calls (from ?proc.time.

**sys.self** CPU time used executing calls (from ?proc.time.

**elapsed** Amount of time in seconds it took the algorithm to converge.

**iterations** Number of iterations to convergence.

---

synthetic

*Synthetic data for examples.*

---

### Description

This simple dataset allows users to use data that are easy to understand while learning `biclustermd`. This is a matrix with 6 rows and 12 columns. 50% of values are missing.

### Usage

```
synthetic
```

### Format

An object of class `matrix` with 6 rows and 12 columns.

---

tune\_biclustermd

*Bicluster data over a grid of tuning parameters*

---

### Description

Bicluster data over a grid of tuning parameters

### Usage

```
tune_biclustermd(  
  data,  
  nrep = 10,  
  parallel = FALSE,  
  ncores = 2,  
  tune_grid = NULL  
)
```

**Arguments**

data	Dataset to bicluster. Must to be a data matrix with only numbers and missing values in the data set. It should have row names and column names.
nrep	The number of times to repeat the biclustering for each set of parameters. Default 10.
parallel	Logical indicating if the user would like to utilize the foreach parallel backend. Default is FALSE.
ncores	The number of cores to use if parallel computing. Default 2.
tune_grid	A data frame of parameters to tune over. The column names of this must match the arguments passed to biclustermd().

**Value**

A list of:

best_combn	The best combination of parameters,
best_bc	The minimum SSE biclustering using the parameters in best_combn,
grid	tune_grid with columns giving the minimum, mean, and standard deviation of the final SSE for each parameter combination, and
runtime	CPU runtime & elapsed time.

**References**

Li, J., Reisner, J., Pham, H., Olafsson, S., and Vardeman, S. (2019) *Biclustering for Missing Data. Information Sciences, Submitted*

**See Also**

[biclustermd](#), [rep\\_biclustermd](#)

**Examples**

```
library(dplyr)
library(ggplot2)
data("synthetic")
tg <- expand.grid(
  miss_val = fivenum(synthetic),
  similarity = c("Rand", "HA", "Jaccard"),
  col_min_num = 2,
  row_min_num = 2,
  col_clusters = 3:5,
  row_clusters = 2
)
tg

# in parallel: two cores:
tbc <- tune_biclustermd(synthetic, nrep = 2, parallel = TRUE, ncores = 2, tune_grid = tg)
tbc
```

```
tbc$grid %>%
  group_by(miss_val, col_clusters) %>%
  summarise(avg_sd = mean(sd_sse)) %>%
  ggplot(aes(miss_val, avg_sd, color = col_clusters, group = col_clusters)) +
  geom_line() +
  geom_point()

tbc <- tune_biclustermd(synthetic, nrep = 2, tune_grid = tg)
tbc

boxplot(tbc$grid$mean_sse ~ tbc$grid$similarity)
boxplot(tbc$grid$sd_sse ~ tbc$grid$similarity)

# nycflights13::flights dataset

library(nycflights13)
data("flights")

library(dplyr)
flights_bcd <- flights %>%
  select(month, dest, arr_delay)

flights_bcd <- flights_bcd %>%
  group_by(month, dest) %>%
  summarise(mean_arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  spread(dest, mean_arr_delay) %>%
  as.data.frame()

# months as rows
rownames(flights_bcd) <- flights_bcd$month
flights_bcd <- as.matrix(flights_bcd[, -1])

flights_grid <- expand_grid(
  row_clusters = 4,
  col_clusters = c(6, 9, 12),
  miss_val = fivenum(flights_bcd),
  similarity = c("Rand", "Jaccard")
)

# RUN TIME: approximately 40 seconds across two cores.
flights_tune <- tune_biclustermd(
  flights_bcd,
  nrep = 10,
  parallel = TRUE,
  ncores = 2,
  tune_grid = flights_grid
)
flights_tune
```

# Index

- \* **datasets**
  - runtimes, [26](#)
  - synthetic, [27](#)
  
- as.Biclust, [3](#)
- autoplot.biclustermd, [4](#)
- autoplot.biclustermd\_sim, [5](#)
- autoplot.biclustermd\_sse, [6](#)
  
- biclustermd, [7](#), [23](#), [28](#)
- biclustermd-package, [2](#)
- binary\_vector\_gen, [9](#)
  
- cell\_heatmap, [10](#)
- cell\_mse, [11](#)
- cluster\_iteration\_sum\_sse, [11](#)
- col.names, [12](#)
- col.names.biclustermd, [12](#)
- col\_cluster\_names, [13](#)
- compare\_biclusters, [14](#)
  
- fill\_emptyies\_P, [14](#)
- fill\_emptyies\_Q, [15](#)
- format\_partition, [16](#)
  
- gather.biclustermd, [16](#)
  
- jaccard\_similarity, [17](#)
  
- mse\_heatmap, [18](#)
  
- part\_matrix\_to\_vector, [20](#)
- partition\_gen, [19](#)
- partition\_gen\_by\_p, [19](#)
- position\_finder, [20](#)
- print.biclustermd, [21](#)
  
- reorder\_biclust, [21](#)
- rep\_biclustermd, [9](#), [22](#), [28](#)
- results\_heatmap, [24](#)
- row.names.biclustermd, [25](#)
  
- row\_cluster\_names, [25](#)
- RRand, [8](#), [22](#)
- runtimes, [26](#)
  
- synthetic, [27](#)
  
- tune\_biclustermd, [9](#), [23](#), [27](#)