

# Package: BCFM (via r-universe)

May 12, 2026

**Type** Package

**Title** Bayesian Clustering Factor Models

**Version** 1.0.0

**Description** Implements the Bayesian Clustering Factor Models (BCFM) for simultaneous clustering and latent factor analysis of multivariate longitudinal data. The model accounts for within-cluster dependence through shared latent factors while allowing heterogeneity across clusters, enabling flexible covariance modeling in high-dimensional settings. Inference is performed using Markov chain Monte Carlo (MCMC) methods with computationally intensive steps implemented via 'Rcpp'. Model selection and visualization tools are provided. The methodology is described in Shin, Ferreira, and Tegge (2018) [doi:10.1002/sim.70350](https://doi.org/10.1002/sim.70350).

**License** GPL (>= 3)

**SystemRequirements** pandoc (>= 1.12.3) for building vignettes

**Imports** Rcpp, RcppArmadillo, dplyr, fastmatrix, ggplot2, grDevices, gridExtra, LaplacesDemon, mvtnorm, psych, RColorBrewer, stats, tidyr, ggpubr, tibble

**LinkingTo** Rcpp, RcppArmadillo

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**Depends** R (>= 3.5)

**LazyData** true

**URL** <https://github.com/ategge/BCFM>

**BugReports** <https://github.com/ategge/BCFM/issues>

**Config/pak/sysreqs** cmake make libicu-dev

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

**Date/Publication** 2026-02-07 03:36:25 UTC

**RemoteUrl** <https://github.com/ategge/bcfm>

**RemoteRef** HEAD

**RemoteSha** e53869492cce50f7c4b4fbc829b2d494f89abc2c

## Contents

BCFM.fit . . . . .	2
BCFM.model.selection . . . . .	4
BCFMcpp . . . . .	6
getmode . . . . .	7
ggplot_B.CI . . . . .	7
ggplot_B.trace . . . . .	8
ggplot_IC . . . . .	9
ggplot_latent.profiles . . . . .	10
ggplot_mu.density . . . . .	11
ggplot_omega.density . . . . .	12
ggplot_probs.density . . . . .	13
ggplot_probs.trace . . . . .	13
ggplot_sigma2.CI . . . . .	14
ggplot_tau.CI . . . . .	15
ggplot_variability . . . . .	16
ggplot_Zit.heatmap . . . . .	17
IC . . . . .	17
init.data . . . . .	18
initialize.cluster.hyperparms . . . . .	19
initialize.hyp.parm . . . . .	20
initialize.model.attributes . . . . .	21
permutation.order . . . . .	21
permutation.scale . . . . .	22
sim.data . . . . .	23
<b>Index</b>	<b>24</b>

---

BCFM.fit

*Fit BCFM Model*

---

## Description

Fits a single Bayesian Clustering Factor Models (BCFM) using C++ implementation. This function serves as a wrapper for the BCFMcpp function, handling timing and output formatting.

**Usage**

```
BCFM.fit(
  data,
  model.attributes,
  hyp.parm,
  n.iter = 50000,
  vague.mu = FALSE,
  covariance = TRUE,
  p.exponent = 2,
  every = 10
)
```

**Arguments**

<code>data</code>	An array of dimensions (observations, variables, time points)
<code>model.attributes</code>	Model attributes generated by <code>initialize.model.attributes</code>
<code>hyp.parm</code>	Hyperparameters generated by <code>initialize.hyp.parm</code>
<code>n.iter</code>	Number of MCMC iterations. Default is 50000.
<code>vague.mu</code>	Logical indicating whether to use vague priors for mu. Default is FALSE.
<code>covariance</code>	Logical indicating whether to model covariance structure. Default is TRUE.
<code>p.exponent</code>	The Dirichlet priors exponent for probabilities. Default is 2.
<code>every</code>	Integer specifying the frequency of progress updates during MCMC. Default is 10.

**Value**

A list containing:

**Result** Output from the BCFMcpp C++ function

**model.attributes** The model attributes used

**hyp.parm** The hyperparameters used

**start.time** POSIXct timestamp when model started

**end.time** POSIXct timestamp when model completed

**run.time** Total elapsed time for model execution

**See Also**

[BCFM.model.selection](#) for fitting multiple models, [initialize.model.attributes](#), [initialize.hyp.parm](#)

**Examples**

```
# Prepare data using the included simulated dataset
data(sim.data)
data.pre <- init.data(sim.data, paste0("V", 1:5))
```

```

# Initialize model components
model.attributes <- initialize.model.attributes(S = nrow(sim.data), times = 1,
                                              R = 5, L = 2, G = 2)
cluster.hyperparms <- initialize.cluster.hyperparms(data.pre, model.attributes)
hyp.parm <- initialize.hyp.parm(model.attributes, cluster.hyperparms)

# Fit model
result <- BCFM.fit(data.pre, model.attributes, hyp.parm,
                  n.iter = 100, every = 10)
result$run.time

```

---

BCFM.model.selection *BCFM Model Selection Over Multiple Groups and Factors*

---

## Description

Performs Bayesian Clustering Factor Models analysis across a grid of group numbers and factor numbers. For each combination, the function fits the BCFM model, calculates IC, and saves results. This is the primary function for model selection to determine the optimal number of clusters and latent factors.

## Usage

```

BCFM.model.selection(
  data,
  cluster.vars,
  grouplist,
  factorlist,
  n.iter = 50000,
  vague.mu = FALSE,
  covariance = TRUE,
  p.exponent = 2,
  every = 10,
  cluster.size = 0.05,
  burnin = NA,
  output_dir = tempdir(),
  seed = NULL
)

```

## Arguments

<code>data</code>	A data frame containing the variables to be analyzed
<code>cluster.vars</code>	A character vector specifying the column names of variables to be used for clustering
<code>grouplist</code>	A numeric vector specifying the numbers of groups to test (e.g., <code>c(2, 3, 4, 5)</code> )

<code>factorlist</code>	A numeric vector specifying the numbers of latent factors to test (e.g., <code>c(1, 2, 3)</code> )
<code>n.iter</code>	Number of MCMC iterations for each model. Default is 50000.
<code>vague.mu</code>	Logical indicating whether to use vague priors for mu. Default is FALSE.
<code>covariance</code>	Logical indicating whether to model covariance structure. Default is TRUE.
<code>p.exponent</code>	The Dirichlet priors exponent for probabilities. Default is 2.
<code>every</code>	Integer specifying the frequency of progress updates during MCMC. Default is 10.
<code>cluster.size</code>	Minimum proportion required for each cluster. Default is 0.05.
<code>burnin</code>	Number of initial MCMC iterations to discard when calculating IC. If NA, an appropriate burnin is determined automatically.
<code>output_dir</code>	Directory where results will be saved. Defaults to <code>tempdir()</code> . The function will create this directory if it doesn't exist.
<code>seed</code>	Optional integer seed for reproducibility.

### Details

The function performs the following steps for each group-factor combination:

1. Preprocesses data using `init.data`
2. Determines optimal variable ordering using `permutation.order`
3. Initializes model attributes and hyperparameters
4. Fits BCFM model using `BCFM.fit`
5. Calculates IC for model comparison
6. Saves individual results and cumulative IC matrix

The IC matrix can be used to identify the optimal model configuration by selecting the combination of groups and factors with the lowest IC value.

### Value

Invisibly returns NULL. Results are saved to disk:

**results-covarianceF-gX-fY.Rdata** Individual model results for each group-factor combination (where X is the number of groups and Y is the number of factors), containing `SDresult` and variable order

**IC.Rdata** Contains `IC.matrix`, timing information, and data. Load this file to compare models and identify the optimal configuration.

### Note

This function can be computationally intensive as it fits multiple models. Consider running on high-performance computing resources for large datasets or extensive model grids. The function includes error handling to continue execution even if individual models fail to converge.

### See Also

[BCFM.fit](#) for fitting a single model, [init.data](#), [initialize.model.attributes](#), [initialize.hyp.parm](#)

**Examples**

```
# Run model selection using the included simulated dataset
data(sim.data)
BCFM.model.selection(
  data = sim.data,
  cluster.vars = paste0("V", 1:5),
  grouplist = 2:3,
  factorlist = 1:2,
  n.iter = 100,
  every = 10,
  burnin = 10
)

# Load and examine IC results
load(file.path(tempdir(), "IC.Rdata"))
print(IC.matrix)
```

---

BCFMcpp

*Gibbs sampler of BCFM*


---

**Description**

It runs a Gibbs sampler for common factors X, factor loadings B, group mean mu, group covariate Omega, idiosyncratic variances  $\sigma^2$ , group assignment Z and group probabilities probs. This function uses Rcpp.

**Usage**

```
BCFMcpp(data, model.attributes, hyp.parm, n.iter, every = 1, verbose = FALSE)
```

**Arguments**

data	The dataset
model.attributes	Model attributes generated by initialize.model.attributes
hyp.parm	Hyperparameters generated by initialize.hyp.parm
n.iter	Total number of iterations
every	Save every every iterations
verbose	Print the results by every 10th step

**Value**

List of Gibbs sampler of parameters described in description

---

getmode	<i>Get the mode of a vector</i>
---------	---------------------------------

---

**Description**

The function returns the mode of a vector.

**Usage**

```
getmode(v)
```

**Arguments**

v                      The vector to find the mode.

**Value**

The mode of v.

---

ggplot_B.CI	<i>Build factor loadings plot</i>
-------------	-----------------------------------

---

**Description**

The function builds a column-wise plots of factor loadings. The parameters fixed at 1 are displayed with red dashed vertical lines.

**Usage**

```
ggplot_B.CI(
  Gibbs,
  true.val = NA,
  burnin = NA,
  permutation = NA,
  main.bool = TRUE,
  var_labels = NULL
)
```

**Arguments**

Gibbs	Result of Gibbs sampler from BCFM function
true.val	True values of factor loadings. If not available, NA.
burnin	Number of burn-in. If not set, it uses the first tenths as burn-in period.
permutation	Permutation of variables. If not set, no permutation.
main.bool	Add title of the plots. Default is TRUE.
var_labels	Character vector of variable names. If NULL, defaults to Variable 1, Variable 2, etc.

**Value**

A ggplot object

---

ggplot_B.trace	<i>Trace plot for posterior of factor loadings</i>
----------------	--

---

**Description**

It returns a trace plot of factor loadings, B, showing MCMC convergence

**Usage**

```
ggplot_B.trace(  
  Gibbs,  
  burnin = NA,  
  permutation = NA,  
  true.val = NA,  
  factor.num = 1,  
  var_labels = NULL  
)
```

**Arguments**

Gibbs	MCMC sample simulated from BCFMR or BCFMcpp function
burnin	Number of burn-in period. If not specified, no burn-in is removed.
permutation	Permutation order vector, if applicable
true.val	True values of factor loadings. If not available, NA.
factor.num	The index of variable to plot (which variable's loadings across all factors to display). Use NULL to plot all variables.
var_labels	Character vector of variable names. If NULL, uses Variable 1, Variable 2, etc.

**Value**

A ggplot object showing trace plots

ggplot\_IC

*Plot IC Matrix from Model Selection***Description**

Creates two plots showing IC values across different numbers of groups and factors to help identify the optimal model configuration.

**Usage**

```
ggplot_IC(matrix, factor_list = 2:4, group_list = 2:4, combine = TRUE)
```

**Arguments**

<code>matrix</code>	An IC matrix from <code>BCFM.model.selection</code> , with rows representing groups and columns representing factors.
<code>factor_list</code>	Numeric vector of factor values corresponding to matrix columns. Default is 2:6.
<code>group_list</code>	Numeric vector of group values corresponding to matrix rows. Default is 5:11.
<code>combine</code>	Logical. If TRUE (default), returns a combined plot using <code>ggarrange</code> . If FALSE, returns a list of two separate ggplot objects.

**Details**

The function creates two complementary visualizations:

- Plot 1: IC vs. number of groups, with lines for each number of factors
- Plot 2: IC vs. number of factors, with lines for each number of groups

Lower IC values indicate better model fit.

**Value**

If `combine = TRUE`, a combined ggplot object. If `combine = FALSE`, a list with two elements:

**by\_groups** ggplot showing IC vs. number of groups

**by\_factors** ggplot showing IC vs. number of factors

**Examples**

```
# Create a toy IC matrix for demonstration
IC.matrix <- matrix(c(100, 95, 90, 92, 88, 85), nrow = 2, ncol = 3)
rownames(IC.matrix) <- paste0("G", 2:3)
colnames(IC.matrix) <- paste0("F", 1:3)

# Combined plot (default)
ggplot_IC(IC.matrix, factor_list = 1:3, group_list = 2:3)
```

```
# Separate ggplot objects
plots <- ggplot_IC(IC.matrix, factor_list = 1:3, group_list = 2:3,
                  combine = FALSE)
plots$by_groups
plots$by_factors
```

---

```
ggplot_latent.profiles
```

*Plot Latent Factor Profiles by Cluster*

---

### Description

Visualizes the latent factor profile means ( $\mu$ ) for each cluster, similar to Latent Profile Analysis (LPA) plots

### Usage

```
ggplot_latent.profiles(
  Gibbs,
  burnin = NA,
  factor_labels = NULL,
  cluster_names = NULL,
  colors = NULL,
  title = "Latent Factor Profiles by Cluster",
  x_label = "Factor",
  y_label = "Posterior Mean"
)
```

### Arguments

Gibbs	Gibbs sample derived from BCFM function
burnin	Number of burn-in period. If not specified, it uses the first tenth as burn-in period
factor_labels	Character vector of factor names. If NULL, defaults to Factor 1, Factor 2, etc.
cluster_names	Character vector of cluster names. If NULL, defaults to Cluster 1, Cluster 2, etc.
colors	Named vector of colors for each cluster. If NULL, uses default color palette
title	Plot title. Default is "Latent Factor Profiles by Cluster"
x_label	X-axis label. Default is "Factor"
y_label	Y-axis label. Default is "Posterior Mean"

### Value

A ggplot object

**Examples**

```
# Fit a model first using the included simulated dataset
data(sim.data)
data.pre <- init.data(sim.data, paste0("V", 1:5))
model.attributes <- initialize.model.attributes(S = nrow(sim.data), times = 1,
                                              R = 5, L = 2, G = 2)
cluster.hyperparms <- initialize.cluster.hyperparms(data.pre, model.attributes)
hyp.parm <- initialize.hyp.parm(model.attributes, cluster.hyperparms)
result <- BCFM.fit(data.pre, model.attributes, hyp.parm,
                  n.iter = 100, every = 10)

# Plot latent profiles
ggplot_latent.profiles(Gibbs = result$Result)
```

---

ggplot\_mu.density      *Density of group means mu using ggplot2*

---

**Description**

The function returns multiple density plots of group mean parameter mu.

**Usage**

```
ggplot_mu.density(
  Gibbs,
  true.val = NA,
  add.legend = FALSE,
  burnin = NA,
  layout.dim = NA,
  main.title = "Posterior Densities of Group Means (mu)",
  x.label = "mu",
  y.label = "Density"
)
```

**Arguments**

Gibbs	The Gibbs sample from BCFM function
true.val	The true value of mu, if applicable
add.legend	Add legend on extra pane
burnin	Number of burn-in period. If not specified, it uses the first tenths as burn-in.
layout.dim	Dimension of panes. If not specified, the plots are in one column.
main.title	Main title for the entire plot. Default is "Posterior Densities of Group Means (mu)"
x.label	X-axis label. Default is "mu"
y.label	Y-axis label. Default is "Density"

**Value**

A ggplot object (grob from grid.arrange)

---

`ggplot_omega.density` *The density plot of the diagonal of group covariance, Omega, with ggplot2*

---

**Description**

It returns multiple plots of the diagonal of group covariance, Omega using ggplot2 and gridExtra. It returns the result by each factor, different colors representing different factors

**Usage**

```
ggplot_omega.density(  
  Gibbs,  
  group.select = 1,  
  true.val = NA,  
  burnin = NA,  
  main.title = "Posterior Densities of Omega",  
  x.label = "Value",  
  y.label = "Density",  
  factor_labels = NULL,  
  show.offdiag = TRUE  
)
```

**Arguments**

<code>Gibbs</code>	Gibbs sample from BCFM
<code>group.select</code>	Group/cluster to plot. If not specified, the first group will be used.
<code>true.val</code>	True values of Omega, if applicable.
<code>burnin</code>	Number of burn-in period. If not specified, the first tenths is used as burn-in.
<code>main.title</code>	Main title for the plot. Default is "Posterior Densities of Omega"
<code>x.label</code>	X-axis label. Default is "Value"
<code>y.label</code>	Y-axis label. Default is "Density"
<code>factor_labels</code>	Character vector of factor names. If NULL, defaults to Factor 1, Factor 2, etc.
<code>show.offdiag</code>	Show off-diagonal elements. Default is TRUE for any k.

**Value**

A ggplot object showing densities of Omega elements

---

`ggplot_probs.density`    *Density plot for posterior of probabilities*

---

### Description

The function returns a density plot of the cluster assignment probability,  $p$ . Different color represent different Clusters.

### Usage

```
ggplot_probs.density(  
  Gibbs,  
  burnin = NA,  
  truep = NA,  
  main.title = "Posterior Densities of Cluster Probabilities",  
  x.label = "Probability",  
  y.label = "Density",  
  cluster_names = NULL  
)
```

### Arguments

<code>Gibbs</code>	MCMC sample simulated from BCFMR or BCFMcpp function
<code>burnin</code>	Number of burn-in period. If not specified, it uses the first tenths as burn-in.
<code>truep</code>	True values of probabilities. If not available, NA.
<code>main.title</code>	Title of the plot. Default is "Posterior Densities of Cluster Probabilities"
<code>x.label</code>	X-axis label. Default is "Probability"
<code>y.label</code>	Y-axis label. Default is "Density"
<code>cluster_names</code>	Character vector of cluster names. If NULL, defaults to Cluster 1, Cluster 2, etc.

### Value

A ggplot object (grob from `grid.arrange`) with plot and legend

---

`ggplot_probs.trace`    *Trace plot of probabilities parameter*

---

### Description

It returns a trace plot of probabilities parameter after burn-in. Different colors represent different groups.

**Usage**

```
ggplot_probs.trace(
  Gibbs,
  burnin = NA,
  main.title = "Trace Plot: Cluster Probabilities",
  x.label = "BCFM Iteration (post burn-in)",
  y.label = "Probability",
  cluster_names = NULL
)
```

**Arguments**

Gibbs	Gibbs sample from BCFM function.
burnin	Number of burn-in period. If not specified, it uses the first tenths sample as burn-in.
main.title	Title of the plot. Default is "Trace Plot: Cluster Probabilities"
x.label	X-axis label. Default is "BCFM Iteration (post burn-in)"
y.label	Y-axis label. Default is "Probability"
cluster_names	Character vector of cluster names. If NULL, defaults to Cluster 1, Cluster 2, etc.

**Value**

A ggplot object showing trace plots

---

ggplot_sigma2.CI	<i>A credible interval plot of posterior of sigma squared</i>
------------------	---

---

**Description**

It returns a credible interval plot of idiosyncratic variance, sigma squared. The lines are 95% intervals, while the circles are posterior mean.

**Usage**

```
ggplot_sigma2.CI(
  Gibbs,
  burnin = NA,
  permutation = NA,
  main.bool = TRUE,
  main.title = NULL,
  x.label = "Variables",
  y.label = "Variance",
  var_labels = NULL
)
```

**Arguments**

Gibbs	Gibbs sample from BCFM function
burnin	Number of burn-in period. If not specified, it uses the first tenths sample as burn-in period.
permutation	Permutation vector, if applicable
main.bool	Return main title. Default is TRUE.
main.title	Main title for the plot. Default is expression for sigma squared.
x.label	X-axis label. Default is "Variables"
y.label	Y-axis label. Default is "Variance"
var_labels	Character vector of variable names. If NULL, defaults to Variable 1, Variable 2, etc.

**Value**

A ggplot object showing the 95\ posterior means (points) of the idiosyncratic variances sigma squared for each variable.

---

ggplot_tau.CI	<i>A credible interval plot of posterior of factor loadings covariance, tau</i>
---------------	---

---

**Description**

It returns a credible interval plot of factor loadings covariance, tau. The lines are 95% intervals, while the circles are posterior mean.

**Usage**

```
ggplot_tau.CI(
  Gibbs,
  burnin = NA,
  true.val = NA,
  main.bool = TRUE,
  main.title = NULL,
  x.label = "Factor",
  y.label = "Tau",
  factor_labels = NULL
)
```

**Arguments**

Gibbs	Gibbs sample from BCFM function
burnin	Number of burn-in period. If not specified, it uses the first tenths sample as burn-in period.
true.val	True values of the taus

<code>main.bool</code>	Return main title. Default is TRUE.
<code>main.title</code>	Main title for the plot. Default is expression for tau.
<code>x.label</code>	X-axis label. Default is "Factor"
<code>y.label</code>	Y-axis label. Default is "Tau"
<code>factor_labels</code>	Character vector of factor names. If NULL, defaults to Factor 1, Factor 2, etc.

**Value**

A ggplot object

---

`ggplot_variability`     *Variability explained by factors*

---

**Description**

Plots the proportion of variability explained by each factor based on eigenvalues of the correlation matrix. Useful for determining the number of factors.

**Usage**

```
ggplot_variability(
  data,
  nfactors = 5,
  main.title = "Proportion of Variability Explained by Factors",
  x.label = "Number of Factors",
  y.label = "Proportion of Variability"
)
```

**Arguments**

<code>data</code>	The data matrix
<code>nfactors</code>	Number of factors to display. Default is 5.
<code>main.title</code>	Main title for the plot. Default is "Proportion of Variability Explained by Factors"
<code>x.label</code>	X-axis label. Default is "Number of Factors"
<code>y.label</code>	Y-axis label. Default is "Proportion of Variability"

**Value**

A ggplot object

---

ggplot\_Zit.heatmap      *A heatmap of group assignments, Z using ggplot2*

---

### Description

A heatmap of group assignments, Z using ggplot2. It first sorts by the largest group with the most assigned.

### Usage

```
ggplot_Zit.heatmap(  
  Gibbs,  
  true.val = NA,  
  burnin = NA,  
  main.title = "Cluster Assignment Heatmap",  
  x.label = "Cluster",  
  y.label = "Subject-Time"  
)
```

### Arguments

Gibbs	Gibbs sample from BCFM function
true.val	Table of true group assignments, if applicable
burnin	Number of burn-in period. If not specified, it uses the first tenths sample as burn-in.
main.title	Title of the plot. Default is "Cluster Assignment Heatmap"
x.label	X-axis label. Default is "Cluster"
y.label	Y-axis label. Default is "Subject-Time"

### Value

A ggplot object

---

IC	<i>Information Criterion. Very close to the original BIC method, but this uses the integrated likelihood instead.</i>
----	---

---

### Description

It finds a Laplace-Metropolis marginal density of likelihood using posterior mean. It also uses Woodbury lemma for fast calculation

### Usage

```
IC(data.row, Gibbs, model.attributes, cluster.size = 0.05, burnin = NA)
```

**Arguments**

<code>data.row</code>	The dataset
<code>Gibbs</code>	Gibbs sample derived form BCFM function
<code>model.attributes</code>	Model attributes generated by <code>initialize.model.attributes</code>
<code>cluster.size</code>	Minimum proportion required for each cluster (default 0.05)
<code>burnin</code>	Number of burn-in period. If not specified, it uses the first tenths as burn-in period

**Value**

The value of Laplace-Metropolis marginal density

---

<code>init.data</code>	<i>Initialize Data Array for BCFM Model</i>
------------------------	---

---

**Description**

Prepares the input data by converting it into a 3D array format required by the BCFM model. If data is already in the correct 3D array format, it returns the data as-is. Takes selected clustering variables and creates an array with dimensions (observations, variables, time points).

**Usage**

```
init.data(data, cluster.vars = NULL)
```

**Arguments**

<code>data</code>	A data frame, matrix, or 3D array containing the data to be used for clustering. If a 3D array with appropriate dimensions is provided and <code>cluster.vars</code> is <code>NULL</code> , the function returns the data unchanged.
<code>cluster.vars</code>	A character vector specifying the column names of variables to be used for clustering (required for data frames). If <code>NULL</code> and data is already a 3D array, the function returns data as-is. If <code>NULL</code> and data is a matrix, all columns are used.

**Value**

A 3D array with dimensions (n, p, t) where n is the number of observations, p is the number of clustering variables, and t is the number of time points (defaults to 1 for cross-sectional data).

**Examples**

```
# Example 1: Data frame with variable selection
data <- data.frame(x1 = rnorm(100), x2 = rnorm(100), x3 = rnorm(100))
cluster.vars <- c("x1", "x2", "x3")
data.pre <- init.data(data, cluster.vars)

# Example 2: Matrix (uses all columns)
data_matrix <- matrix(rnorm(300), nrow = 100, ncol = 3)
data.pre <- init.data(data_matrix)

# Example 3: 3D array (returns as-is)
data_3d <- array(rnorm(1500), dim = c(100, 3, 5))
data.pre <- init.data(data_3d) # Returns unchanged
```

---

```
initialize.cluster.hyperparms
```

*Initialize cluster hyperparameters*

---

**Description**

The function returns list of hyperparameters for Omegas and mus.

**Usage**

```
initialize.cluster.hyperparms(
  data,
  model.attributes,
  covariance = FALSE,
  diag.Psi = FALSE,
  vague.mu = FALSE,
  zero.mu = FALSE,
  seed = NULL
)
```

**Arguments**

data	The dataset.
model.attributes	Model attributes generated by initialize.model.attributes
covariance	Use of covariance matrix of common factors. If FALSE, it uses the correlation matrix.
diag.Psi	Diagonal matrix for cluster covariance. If FALSE, it uses the sample covariance.
vague.mu	Use of large cluster covariance prior.
zero.mu	Set the cluster mean prior at 0. If FALSE, the cluster mean prior are the sample means of the clusters.
seed	Optional integer seed for reproducibility.

**Value**

A list of mean and variance hyperparameter of mu, and scale hyperparameter of Omega

---

initialize.hyp.parm     *Initialize hyperparameters for BCFM model*

---

**Description**

The function returns a list of hyperparameters of Omega,  $\sigma^2$  and mu. It also calls the results from cluster.hyperparms and information from model.attributes.

**Usage**

```
initialize.hyp.parm(
  model.attributes,
  cluster.hyperparms,
  n.sigma = 2.2,
  n.s2.sigma = 0.1,
  n.tau = 1,
  n.s2.tau = 1,
  omega.diag.nu = 2,
  p.exponent = NA
)
```

**Arguments**

model.attributes	Model attributes generated by initialize.model.attributes
cluster.hyperparms	Cluster related hyperparameters generated by initialize.cluster.hyperparms
n.sigma	The shape parameter of $\sigma^2$ . If not specified, 6.
n.s2.sigma	The rate parameter of $\sigma^2$ . If not specified, 4.
n.tau	The shape parameter of $\tau^2$ . If not specified, 6.
n.s2.tau	The rate parameter of $\tau^2$ . If not specified, 4.
omega.diag.nu	The shape parameter for the first cluster covariance
p.exponent	The Dirichlet priors of probabilities.

**Value**

A list of fixed hyperparameters of mu, Omega and sigma squared.

---

```
initialize.model.attributes
```

*Build model attributes from the dataset*

---

**Description**

Basic setting and information of the dataset: number of subjects, timepoints, variables, factors and groups.

**Usage**

```
initialize.model.attributes(S = 216, times = 5, R = 9, L = 3, G = 4)
```

**Arguments**

S	Number of subjects
times	Number of timepoints
R	Number of covariates
L	Number of factors
G	Number of groups

**Value**

A list of model attributes

---

```
permutation.order
```

*Order of permutation by the largest absolute value in each eigenvector*

---

**Description**

It finds the vector of permutation to permute data by its largest absolute value in each eigenvector. It sets the order by specified number of factors, and the rest is ordered as they were.

**Usage**

```
permutation.order(data, covariance = FALSE, L = 1, fa = TRUE)
```

**Arguments**

data	The dataset
covariance	Logic variable indicating whether the analysis uses covariance or correlation matrix
L	Number of latent factors
fa	Use factor analysis to sort the variables

**Value**

The vector of permutation

---

permutation.scale	<i>Permute the dataset by the largest absolute value in each eigenvector, and scale</i>
-------------------	---

---

**Description**

It finds the vector of permutation to permute data by its largest absolute value in each eigenvector. It sets the order by specified number of factors, and the rest is ordered as they were. The data is permuted, and if needed, scaled.

**Usage**

```
permutation.scale(  
  data,  
  permutation = NA,  
  covariance = FALSE,  
  return.array = TRUE,  
  num.layers = 1  
)
```

**Arguments**

data	The dataset
permutation	Vector with the permutation of the data
covariance	Logic variable indicating whether the analysis uses covariance or correlation matrix
return.array	Return the data as 3-dimensional array
num.layers	Number of timepoints

**Value**

The dataset that is permuted, either in matrix or array

---

sim.data	<i>Simulated data for BCFM model</i>
----------	--------------------------------------

---

**Description**

A simulated dataset for demonstrating the Bayesian Consensus Factor Model.

**Usage**

```
sim.data
```

**Format**

A data frame with 200 rows and 20 variables (V1 through V20): All variables are simulated numeric values.

**Source**

Simulated data generated for package examples

**Examples**

```
data(sim.data)
dim(sim.data)
head(sim.data)
```

# Index

## \* datasets

sim.data, [23](#)

BCFM.fit, [2](#), [5](#)

BCFM.model.selection, [3](#), [4](#)

BCFMcpp, [6](#)

getmode, [7](#)

ggplot\_B.CI, [7](#)

ggplot\_B.trace, [8](#)

ggplot\_IC, [9](#)

ggplot\_latent.profiles, [10](#)

ggplot\_mu.density, [11](#)

ggplot\_omega.density, [12](#)

ggplot\_probs.density, [13](#)

ggplot\_probs.trace, [13](#)

ggplot\_sigma2.CI, [14](#)

ggplot\_tau.CI, [15](#)

ggplot\_variability, [16](#)

ggplot\_Zit.heatmap, [17](#)

IC, [17](#)

init.data, [5](#), [18](#)

initialize.cluster.hyperparms, [19](#)

initialize.hyp.parm, [3](#), [5](#), [20](#)

initialize.model.attributes, [3](#), [5](#), [21](#)

permutation.order, [21](#)

permutation.scale, [22](#)

sim.data, [23](#)