# Package 'BCClong'

August 8, 2023

**Type** Package

**Title** Bayesian Consensus Clustering for Multiple Longitudinal Features

**Version** 1.0.1

**Maintainer** Zhiwen Tan <21zt9@queensu.ca>

**Description** It is very common nowadays for a study to collect multiple features and appropriately integrating multiple longitudinal features simultaneously for defining individual clusters becomes increasingly crucial to understanding population heterogeneity and predicting future outcomes. 'BCClong' implements a Bayesian consensus clustering (BCC) model for multiple longitudinal features via a generalized linear mixed model. Compared to existing packages, several key features make the 'BCClong' package appealing: (a) it allows simultaneous clustering of mixed-type (e.g., continuous, discrete and categorical) longitudinal features, (b) it allows each longitudinal feature to be collected from different sources with measurements taken at distinct sets of time points (known as irregularly sampled longitudinal data), (c) it relaxes the assumption that all features have the same clustering structure by estimating the feature-specific (local) clusterings and consensus (global) clustering.

**License** MIT + file LICENSE

**Depends** R (>= 3.5.0)

**Imports** cluster, coda, ggplot2, graphics, label.switching, LaplacesDemon, lme4, MASS, mclust, MCMCpack, mixAK, mvtnorm, nnet, Rcpp (>= 1.0.9), Rmpfr, stats, truncdist

**Suggests** cowplot, joineRML, knitr, rmarkdown, survival, survminer, testthat (>= 3.0.0)

**LinkingTo** Rcpp, RcppArmadillo

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**NeedsCompilation** yes

**Author** Zhiwen Tan [aut, cre],
Zihang Lu [ctb],
Chang Shen [ctb]

**Repository** CRAN

**Date/Publication** 2023-08-08 10:10:08 UTC

# R topics documented:



---

`BayesT` *Goodness of fit.*

---

## Description

This function assess the model goodness of fit by calculate the discrepancy measure T(bmy, bm-Theta) with following steps (a) Generate T.obs based on the MCMC samples (b) Generate T.rep based on the posterior distribution of the parameters (c) Compare T.obs and T.rep, and calculate the P values.

## Usage

```
BayesT(fit)
```

## Arguments

| | |
|---|---|
| `fit` | an objective output from BCC.multi() function |

## Value

Returns a list with length equals to 2 that contains observed and predict value

## Examples

```
#import data
filePath <- system.file("extdata", "example.rds", package = "BCClong")
fit.BCC <- readRDS(filePath)
set.seed(20220929)
BayesT(fit.BCC)
```

BCC.multi *Compute a Bayesian Consensus Clustering model for mixed-type longitudinal data*

## Description

This function performs clustering on mixed-type (continuous, discrete and categorical) longitudinal markers using Bayesian consensus clustering method with MCMC sampling

## Usage

```
BCC.multi(
  mydat,
  id,
  time,
  center = 1,
  num.cluster,
  formula,
  dist,
  alpha.common = 0,
  initials = NULL,
  sigma.sq.e.common = 1,
 hyper.par = list(delta = 1, a.star = 1, b.star = 1, aa0 = 0.001, bb0 = 0.001, cc0 =
    0.001, ww0 = 0, vv0 = 1000, dd0 = 0.001, rr0 = 4, RR0 = 3),
  c.ga.tunning = NULL,
  c.theta.tunning = NULL,
  adaptive.tunning = 0,
  tunning.freq = 20,
  initial.cluster.membership = "random",
  input.initial.local.cluster.membership = NULL,
  input.initial.global.cluster.membership = NULL,
  seed.initial = 2080,
  burn.in,
  thin,
  per,
  max.iter
)
```

## Arguments

| | |
|---|---|
| mydat | list of R longitudinal features (i.e., with a length of R), where R is the number of features. The data should be prepared in a long-format (each row is one time point per individual). |
| id | a list (with a length of R) of vectors of the study id of individuals for each feature. Single value (i.e., a length of 1) is recycled if necessary |
| time | a list (with a length of R) of vectors of time (or age) at which the feature measurements are recorded |

`center` 1: center the time variable before clustering, 0: no centering

`num.cluster` number of clusters K

`formula` a list (with a length of R) of formula for each feature. Each formula is a twosided linear formula object describing both the fixed-effects and random effects part of the model, with the response (i.e., longitudinal feature) on the left of a ~ operator and the terms, separated by + operations, or the right. Random-effects terms are distinguished by vertical bars (|) separating expressions for design matrices from grouping factors. See formula argument from the lme4 package

`dist` a character vector (with a length of R) that determines the distribution for each feature. Possible values are "gaussian" for a continuous feature, "poisson" for a discrete feature (e.g., count data) using a log link and "binomial" for a dichotomous feature (0/1) using a logit link. Single value (i.e., a length of 1) is recycled if necessary

`alpha.common` 1 - common alpha, 0 - separate alphas for each outcome

`initials` List of initials for: zz, zz.local ga, sigma.sq.u, sigma.sq.e, Default is NULL

`sigma.sq.e.common`
1 - estimate common residual variance across all groups, 0 - estimate distinct residual variance, default is 1

`hyper.par` hyper-parameters of the prior distributions for the model parameters. The default hyper-parameters values will result in weakly informative prior distributions.

`c.ga.tunning` tuning parameter for MH algorithm (fixed effect parameters), each parameter corresponds to an outcome/marker, default value equals NULL

`c.theta.tunning`
tuning parameter for MH algorithm (random effect), each parameter corresponds to an outcome/marker, default value equals NULL

`adaptive.tunning`
adaptive tuning parameters, 1 - yes, 0 - no, default is 1

`tunning.freq` tuning frequency, default is 20

`initial.cluster.membership`
"mixAK" or "random" or "PAM" or "input" - input initial cluster membership for local clustering, default is "random"

`input.initial.local.cluster.membership`
if use "input", option input.initial.cluster.membership must not be empty, default is NULL

`input.initial.global.cluster.membership`
input initial cluster membership for global clustering default is NULL

`seed.initial` seed for initial clustering (for initial.cluster.membership = "mixAK") default is 2080

`burn.in` the number of samples disgarded. This value must be smaller than max.iter.

`thin` the number of thinning. For example, if thin = 10, then the MCMC chain will keep one sample every 10 iterations

`per` specify how often the MCMC chain will print the iteration number

`max.iter` the number of MCMC iterations.

## Value

Returns a model contains clustering information

## Examples

```
# import dataframe
filePath <- system.file("extdata", "epil.rds", package = "BCClong")
dat <- readRDS(filePath)
set.seed(20220929)
# example only, larger number of iteration required for accurate result
fit.BCC <-  BCC.multi (
       mydat = list(dat$anxiety_scale,dat$depress_scale),
       dist = c("gaussian"),
       id = list(dat$id),
       time = list(dat$time),
       formula =list(y ~ time + (1|id)),
       num.cluster = 2,
       burn.in = 3,
       thin = 1,
       per =1,
       max.iter = 8)
```

---

# model.selection.criteria

*Model selection*

---

## Description

A function that calculates DIC and WAIC for model selection

## Usage

```
model.selection.criteria(fit, fast_version = 1)
```

## Arguments

| | |
|---|---|
| `fit` | an objective output from BCC.multi() function |
| `fast_version` | if fast_verion=1 (default), then compute the DIC and WAIC using the first 100 MCMC samples (after burn-in and thinning) . If fast_version=0, then compute the DIC and WAIC using all MCMC samples (after burn-in and thinning) |

## Value

Returns the calculated score

## Examples

```
#import data
filePath <- system.file("extdata", "example1.rds", package = "BCClong")
fit.BCC <- readRDS(filePath)
res <- model.selection.criteria(fit.BCC, fast_version=1)
res
```

---

`traceplot` *Trace plot function*

---

## Description

To visualize the MCMC chain for model parameters

## Usage

```
traceplot(
  fit,
  cluster.indx = 1,
  feature.indx = 1,
  parameter = "PPI",
  xlab = NULL,
  ylab = NULL,
  ylim = NULL,
  xlim = NULL,
  title = NULL
)
```

## Arguments

| | |
|---|---|
| `fit` | an objective output from BCC.multi() function. |
| `cluster.indx` | a numeric value. For cluster-specific parameters, specifying cluster.indx will generate the trace plot for the corresponding cluster. |
| `feature.indx` | a numeric value. For cluster-specific parameters, specifying feature.indx will generate the trace plot for the corresponding cluster. |
| `parameter` | a character value. Specify which parameter for which the trace plot will be generated. The value can be "PPI" for pi, alpha for alpha, "GA" for gamma, "SIGMA.SQ.U" for Sigma and "SIGMA.SQ.E" for sigma. |
| `xlab` | Label for x axis |
| `ylab` | Label for y axis |
| `ylim` | The range for y axis |
| `xlim` | The range for x axis |
| `title` | Title for the trace plot |

## Value

void function with no return value, only show plots

## Examples

```
# get data from the package
filePath <- system.file("extdata", "epil1.rds", package = "BCClong")
fit.BCC <- readRDS(filePath)
traceplot(fit=fit.BCC, parameter="PPI",ylab="pi",xlab="MCMC samples")
```

---

# trajplot *Trajplot for fitted model*

---

## Description

plot the longitudinal trajectory of features by local and global clusterings

## Usage

```
trajplot(
  fit,
  feature.ind = 1,
  which.cluster = "global.cluster",
  title = NULL,
  ylab = NULL,
  xlab = NULL,
  color = NULL
)
```

## Arguments

| | |
|---|---|
| fit | an objective output from BCC.multi() function |
| feature.ind | a numeric value indicating which feature to plot. The number indicates the order of the feature specified in mydat argument of the BCC.multi()() function |
| which.cluster | a character value: "global" or "local", indicating whether to plot the trajectory by global cluster or local cluster indices |
| title | Title for the trace plot |
| ylab | Label for y axis |
| xlab | Label for x axis |
| color | Color for the trajplot |

## Value

void function with no return value, only show plots

## Examples

```
# get data from the package
filePath <- system.file("extdata", "epil1.rds", package = "BCClong")
fit.BCC <- readRDS(filePath)
# for local cluster
trajplot(fit=fit.BCC,feature.ind=1, which.cluster = "local.cluster",
         title= "Local Clustering",xlab="time (months)",
         ylab="anxiety",color=c("#00BA38", "#619CFF"))

# for global cluster
trajplot(fit=fit.BCC,feature.ind=1,
         which.cluster = "global.cluster",
         title="Global Clustering",xlab="time (months)",
         ylab="anxiety",color=c("#00BA38", "#619CFF"))
```

# Index