Package: BCClong (via r-universe)

August 25, 2024

Type Package

Title Bayesian Consensus Clustering for Multiple Longitudinal Features

Version 1.0.3

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, abind, gridExtra

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

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://zhiwent.r-universe.dev

RemoteUrl https://github.com/zhiwent/bcclong

RemoteRef HEAD

RemoteSha 1b50efcdc7c54a120180b9f03d44192a936e98c7

Contents

`BayesT` *Goodness of fit.*

Description

This function assess the model goodness of fit by calculate the discrepancy measure T(bm(y), 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 dataframe with length equals to 2 that contains observed and predict value

Examples

```
#import data
data(example)
fit.BCC <- example
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 BCC class model contains clustering information

Examples

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

---

conRes *conRes dataset*

Description

This data sets contains the result that run from `BayesT` function using epil1 BCC object. The epil1 object was obtained using `BCC.multi` function

Usage

```
data(conRes)
```

Format

This is a dataframe with two columns and twenty observations

Examples

```
data(conRes)
conRes
```

`epil` *epil dataset*

Description

This is epileptic.qol data set from `joinrRML`

Usage

```
data(epil)
```

Format

This is a dataframe with 4 varaibles and 1852 observations

Examples

```
data(epil)
epil
```

`epil1` *epil1 model*

Description

This model contains the result that run from `BCC.multi` function using epileptic.qol dataset in `joinrRML` package. This model has formula of `formula =list(y ~ time + (1|id))`

Usage

```
data(epil1)
```

Format

This is a BCC model with thirty elements

Examples

```
data(epil1)
epil1
```

epil2 *epil2 model*

Description

This model contains the result that run from `BCC.multi` function using epileptic.qol dataset in `joinrRML` package. This model has formula of `formula =list(y ~ time + (1 + time|id))`

Usage

```
data(epil2)
```

Format

This is a BCC model with thirty elements

Examples

```
data(epil2)
epil2
```

epil3 *epil3 model*

Description

This model contains the result that run from `BCC.multi` function using epileptic.qol dataset in `joinrRML` package. This model has formula of `formula =list(y ~ time + time2 + (1 + time|id))`

Usage

```
data(epil3)
```

Format

This is a BCC model with thirty elements

Examples

```
data(epil3)
epil3
```

example *example model*

Description

This is an example model which contains the result that run from `BCC.multi` function using epileptic.qol dataset in `joinrRML` package. Only used in documented example and tests. Since small number of iterations were used, this model can may not represent the true performance for this method.

Usage

```
data(example)
```

Format

This is a BCC model with thirty elements

Examples

```
data(example)
example
```

example1 *example1 model*

Description

This is an example model which contains the result that run from `BCC.multi` function using epileptic.qol dataset in `joinrRML` package. Only used the tests. Since small number of iterations were used, this model can may not represent the true performance for this method.

Usage

```
data(example1)
```

Format

This is a BCC model with thirty elements

Examples

```
data(example1)
example1
```

model.selection.criteria *Model selection*

Description

A function that calculates DIC and WAIC for model selection

Usage

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

Arguments

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

Value

Returns the calculated score

Examples

```
#import data
data(example1)
fit.BCC <- example1
res <- model.selection.criteria(fit.BCC, fast_version=TRUE)
res
```

PBCseqfit *PBCseqfit model*

Description

This model contains the result that run from BCC.multi function using PBC910 dataset in mixAK package

Usage

```
data(PBCseqfit)
```

Format

This is a BCC model with thirty elements

Examples

```
data(PBCseqfit)
PBCseqfit
```

`plot.BCC` *Generic plot method for BCC objects*

Description

Generic plot method for BCC objects

Usage

```
## S3 method for class 'BCC'
plot(x, ...)
```

Arguments

`x`	An object of class BCC.
`...`	further arguments passed to or from other methods.

Value

Void function plot model object, no object return

Examples

```
# get data from the package
data(epil1)
fit.BCC <- epil1
plot(fit.BCC)
```

print.BCC *Generic print method for BCC objects*

Description

Generic print method for BCC objects

Usage

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

Arguments

x	An object of class BCC.
...	further arguments passed to or from other methods.

Value

Void function prints model information, no object return

Examples

```
# get data from the package
data(epil2)
fit.BCC <- epil2
print(fit.BCC)
```

summary.BCC *Generic summary method for BCC objects*

Description

Generic summary method for BCC objects

Usage

```
## S3 method for class 'BCC'
summary(object, ...)
```

Arguments

object	An object of class BCC.
...	further arguments passed to or from other methods.

Value

Void function summarize model information, no object return

Examples

```
# get data from the package
data(epil2)
fit.BCC <- epil2
summary(fit.BCC)
```

`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
data(epil1)
fit.BCC <- epil1
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

A plot object

Examples

```
# get data from the package
data(epil1)
fit.BCC <- epil1
# 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