Package ‘bmscstan’

October 12, 2022

Type Package

Title Bayesian Multilevel Single Case Models using 'Stan'

Version 1.2.1.0

Description Analyse single case analyses against a control group.
Its purpose is to provide a flexible, with good power and
low first type error
approach that can manage at the same time controls' and patient's data.
The use of Bayesian statistics allows to test both the alternative and
null hypothesis.
Scandola, M., & Romano, D. (2020, August 3). <doi:10.31234/osf.io/sajdq>
Scandola, M., & Romano, D. (2021). <doi:10.1016/j.neuropsychologia.2021.107834>.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0), rstan, ggplot2, bayesplot

Imports loo, logspline, LaplacesDemon

VignetteBuilder knitr

Suggests reshape2, gridExtra, bridgesampling, testthat, knitr,
rmarkdown, covr

URL https://github.com/michelescandola/bmscstan

BugReports https://github.com/michelescandola/bmscstan

RoxygenNote 7.2.1

NeedsCompilation no

Author Michele Scandola [aut, cre] (<https://orcid.org/0000-0003-0853-8975>)

Maintainer Michele Scandola <michele.scandola@univr.it>

Repository CRAN

Date/Publication 2022-09-04 18:00:02 UTC

R topics documented:

BMSC *Fit Bayesian Multilevel Single Case models*

Description

BMSC fits the Bayesian Multilevel Single Case models.

Usage

```
BMSC(
  formula,
  data_ctrl,
  data_sc,
  cores = 1,
  chains = 4,
  iter = 4000,
  warmup,
  seed = NA,
  typeprior = "normal",
  s,
  family = "gaussian",
  ...
)
```

Arguments

`formula` An object of class `formula`: a symbolic description of the model to be fitted.

`data_ctrl` An object of class `data.frame` (or one that can be coerced to that class) containing data of all variables used in the model for the control group.

`data_sc` An object of class `data.frame` (or one that can be coerced to that class) containing data of all variables used in the model for the Single Case

`cores` The number of cores to use when executing the Markov chains in parallel. The default is 1.

`chains` Number of Markov chains (defaults to 4).

`iter` Number of total iterations per chain (including warmup; defaults to 4000).

`warmup` A positive integer specifying number of warmup (aka burnin) iterations. This also specifies the number of iterations used for stepsize adaptation, so warmup samples should not be used for inference. The number of warmup should not be larger than iter and the default is 2000.

`seed` The seed for random number generation to make results reproducible. If NA (the default), Stan will set the seed randomly.

`typeprior` Set the desired prior distribution for the fixed effects.

- **normal** a normal distribution with $\mu = 0$ and $\sigma = 10$
- **cauchy** a cauchy distribution with $\mu = 0$ and scale $\sqrt{2}/2$
- **student** a Student's T distribution, with $\mu = 0$, $\nu = 3$ and $\sigma = 10$

The normal distribution is the default.

The σ or scale parameters of the prior distributions can be modified by setting the dispersion parameter `s`.

`s` is the dispersion parameter (standard deviation or scale) for the prior distribution.

If NULL (the default) and `typeprior = "normal"` or `typeprior = "student"` `s = 10`, otherwise, if `typeprior = "cauchy"` `s = sqrt(2)/2`.

`family` a description of the response distribution to be used in this model. This is a character string naming the family. By default, a linear gaussian model is applied.

- **gaussian** the dependent variable is distributed along a Gaussian distribution, with `identity` link function
- **binomial** the dependent variable is distributed along a Binomial distribution, with `logit` link function

`...` further arguments to be passed to **stan** function.

Value

a `BMSC` object

Examples

```
# simulation of healthy controls data

Sigma.ctrl <- matrix(cbind(1, .7,  .7, 1) ,nrow=2)

U <- t(chol(Sigma.ctrl))

numobs <- 100

set.seed(123)

random.normal <- matrix( rnorm( n = ncol(U) * numobs, mean = 3, sd = 1),
                         nrow = ncol(U), ncol = numobs)

X = U %*% random.normal

dat.ctrl <- as.data.frame(t(X))

names(dat.ctrl) <- c("y","x")

cor(dat.ctrl)

# simulation of patient data

Sigma.pt <- matrix(cbind(1, 0,  0, 1) ,nrow=2)

U <- t(chol(Sigma.pt))

numobs <- 20

set.seed(0)

random.normal <- matrix( rnorm( n = ncol(U) * numobs, mean = 3, sd = 1),
                 nrow = ncol(U), ncol = numobs)

X = U %*% random.normal

dat.pt <- as.data.frame(t(X))

names(dat.pt) <- c("y","x")

cor(dat.pt)

# fit the single case model

mdl.reg <- BMSC(y ~ x, data_ctrl = dat.ctrl, data_sc = dat.pt, seed = 10)

# posterior-predictive check of the model

pp_check(mdl.reg)
```

```
# summarize the results

summary(mdl.reg)

# plot the results

plot(mdl.reg)
```

---

bmscstan *Bayesian Multilevel Single Case models using 'Stan'*

Description

The **bmscstan** package provides an interface to fit Bayesian Multilevel Single Case models. These models compare the performance of a Single Case against a control group, combining the flexibility of multilevel models and the potentiality of Bayesian Statistics.

Details

The package is now limited to gaussian data only, but we will further expand it to cover binomial and ordinal (Likert scales) data.

By means of **bmscstan** the effects of the control group and the effects of the deviance between the Single Case and the group will be estimated.

The model to estimate the controls parameters is:

$y\ N(\beta X + bZ, \sigma^2)$

where y is the controls' dependent variable, X the contrast matrix for Population-level (or Fixed) Effects, and β are the unknown coefficients to be estimate. Z is the contrast matrix for the Varying (or Random, or Group-level) effects, and b are the unknown estimates for the varying effects. σ^2 is the variance.

In order to estimate the coefficients of the Single Case, the formula is the following:

$y_{pt}\ N(\phi X_{pt}, \sigma^2_{pt})$

where $\phi = \beta + \delta$.

The validation of the approach can be found here: https://www.doi.org/10.31234/osf.io/sajdq

Details

The main function of **bmscstan** is BMSC, which uses formula syntax to specify your model.

BMSC_loo *loo and waic.*

Description

bmscstan wrapper for computing approximate leave-one-out cross-validation (loo) and Watanabe-Akaike Information Criterion or Widely Applicable Information Criterion (WAIC) using PSIS-LOO for the single case and the control group

Usage

```
BMSC_loo(x, cores = 1, ...)

BMSC_waic(x, ...)

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

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

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

Arguments

x	An object of class BMSC, resulting from the BMSC function.
cores	The number of cores for the ‘loo::relative_eff‘ function
...	for ‘BMSC_loo‘ and ‘BMSC_waic‘ further arguments passed to the ‘loo::extract_log_lik‘ function. for ‘print‘ and ‘plot‘ methods further arguments to be passed to the ‘print‘ or ‘plot‘ functions

Value

for ‘BMSC_loo‘ a list with the log likelihood of the single case and the control group, the MCMC effective sample size divided by the total sample size, and the leave-one-out cross-validation. For ‘BMSC_waic‘ a list with the log likelihood of the single case and the control group, and the waic scores.

`BMSC_loo_compare` *bmscstan wrapper for model comparison.*

Description

bmscstan wrapper for model comparison.

Usage

```
BMSC_loo_compare(x, ...)

## S3 method for class 'loo_compare_BMSC'
print(x, simplify = TRUE, ...)

## S3 method for class 'waic_compare_BMSC'
print(x, simplify = TRUE, ...)
```

Arguments

`x`	A list of `loo_BMSC` or `waic_BMSC` objects.
`...`	further arguments passed to the function.
`simplify`	For the print method only, should only the essential columns of the summary matrix be printed? The entire matrix is always returned, but by default only the most important columns are printed.

Value

a list with the log likelihood of the single case and the control group, the MCMC effective sample size divided by the total sample size, and the leave-one-out cross-validation.

`BMSC_pareto_k_table` *bmscstan wrapper for diagnostics for Pareto smoothed importance sampling (PSIS)*

Description

bmscstan wrapper for diagnostics for Pareto smoothed importance sampling (PSIS)

Usage

```
BMSC_pareto_k_table(x)

BMSC_pareto_k_ids(x, threshold = 0.5)

BMSC_mcse_loo(x, threshold = 0.7)

BMSC_pareto_k_values(x)

BMSC_pareto_k_influence_values(x)

BMSC_psis_n_eff_values(x)

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

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

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

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

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

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

Arguments

`x`	An object of class `loo_BMSC`
`threshold`	for the ‘pareto_k_ids‘ method is the minimum k value to flag. for the ‘mcse_loo‘ method all the k values greater than the ‘threshold‘ will be returned as NA.
`...`	further arguments passed to the ‘print‘ function.

Value

- pareto_k_table returns an object of class "pareto_k_table_BMSC", which is a matrix with columns "Count", "Proportion", and "Min. n_eff"
- pareto_k_ids returns an integer vector indicating which observations have Pareto k estimates above threshold
- mcse_loo returns the Monte Carlo standard error (MCSE) estimate for PSIS-LOO. MCSE will be NA if any Pareto kk values are above threshold.

- pareto_k_values returns a vector of the estimated Pareto k parameters. These represent the reliability of sampling.
- pareto_k_influence_values returns a vector of the estimated Pareto k parameters. These represent influence of the observations on the model posterior distribution.
- psis_k_influence_table returns a vector of the estimated PSIS effective sample sizes.

`data.ctrl` *Data from a control group of 16 participants*

Description

A dataset containing the results from the Body Sidednedd Task from a control group of 16 participants

Usage

```
data.ctrl
```

Format

A data frame with 4049 rows and 5 variables

RT Reaction times, in milliseconds

Body.District Body district, categorial factor of Body Sidedness Task: FOOT or HAND

Congruency The trail was Congruent or Incongruent?

Side The trial showed a left or right limb

ID The participant ID

`data.pt` *Data from a Single Case with brachial plexious lesion*

Description

A dataset containing the results from the Body Sidedness Task from a single Single Case

Usage

```
data.pt
```

Format

A data frame with 467 rows and 4 variables

RT Reaction times, in milliseconds

Body.District Body district, categorial factor of Body Sidedness Task: FOOT or HAND

Congruency The trail was Congruent or Incongruent?

Side The trial showed a left or right limb pairwise.BMSC *Pairwise contrasts*

Description

Calculate pairwise comparisons between marginal posterior distributions divided by group levels

Usage

```
pairwise.BMSC(mdl, contrast, covariate = NULL, who = "delta")
```

Arguments

mdl	An object of class BMSC.
contrast	Character value giving the name of the coefficient whose levels need to be compared.
covariate	at the moment is silent
who	parameter to choose the estimates to contrast **control** only the controls **singlecase** only the single case $(\beta + \delta)$ **delta** only the difference between the single case and controls

Value

a pairwise.BMSC object

Examples

```
######################################
# simulation of controls' group data
######################################

# Number of levels for each condition and trials
NCond1  <- 2
NCond2  <- 2
Ntrials <- 8
NSubjs  <- 30

betas <- c( 0 , 0 , 0 ,  0.2)

data.sim <- expand.grid(
  trial      = 1:Ntrials,
  ID         = factor(1:NSubjs),
  Cond1      = factor(1:NCond1),
  Cond2      = factor(1:NCond2)
```

```
)

contrasts(data.sim$Cond1) <- contr.sum(2)
contrasts(data.sim$Cond2) <- contr.sum(2)

### d.v. generation
y <- rep( times = nrow(data.sim) , NA )

# cheap simulation of individual random intercepts
set.seed(1)
rsubj <- rnorm(NSubjs , sd = 0.1)

for( i in 1:length( levels( data.sim$ID ) ) ){

  sel <- which( data.sim$ID == as.character(i) )

  mm  <- model.matrix(~ Cond1 * Cond2 , data = data.sim[ sel , ] )

  set.seed(1 + i)
  y[sel] <- mm %*% as.matrix(betas + rsubj[i]) +
    rnorm( n = Ntrials * NCond1 * NCond2 )

}

data.sim$y <- y

# just checking the simulated data...
boxplot(y~Cond1*Cond2, data = data.sim)

######################################
# simulation of patient data
######################################

betas.pt <- c( 0 , 0.8 , 0 ,  0)

data.pt <- expand.grid(
  trial      = 1:Ntrials,
  Cond1      = factor(1:NCond1),
  Cond2      = factor(1:NCond2)
)

contrasts(data.pt$Cond1) <- contr.sum(2)
contrasts(data.pt$Cond2) <- contr.sum(2)

### d.v. generation
mm  <- model.matrix(~ Cond1 * Cond2 , data = data.pt )

set.seed(5)
data.pt$y <- (mm %*% as.matrix(betas.pt) +
                rnorm( n = Ntrials * NCond1 * NCond2 ))[,1]

# just checking the simulated data...
boxplot(y~Cond1*Cond2, data = data.pt)
```

```
mdl <- BMSC(y ~ Cond1 * Cond2 + ( 1 | ID ),
            data_ctrl = data.sim, data_sc = data.pt, seed = 77,
            typeprior = "cauchy", s = 1)

summary(mdl)

pp_check(mdl)


pairwise.BMSC( mdl, contrast = "Cond11:Cond21")
```

`plot.BMSC` *Plot estimates from a* `BMSC` *object.*

Description

Plot estimates from a `BMSC` object.

Usage

```
## S3 method for class 'BMSC'
plot(x, who = "both", type = "interval", CI = 0.95, ...)
```

Arguments

`x` An object of class BMSC.

`who` parameter to choose the estimates to plot

- **both** plot in the same graph both controls and the Single Case
- **control** only the controls
- **single** only the Single Case $(\beta + \delta)$
- **delta** only the difference between the Single Case and controls

`type` a parameter to select the typology of graph

- **interval** the estimates will be represented by means of pointrange, with median and the boundaries of the credible interval
- **area** a density plot
- **hist** a density histogram

`CI` the dimension of the Credible Interval (or Equally Tailed Interval). Default 0.95.

`...` other arguments are ignored.

Value

a plot, a ggplot2 object, or a bayesplot object

Examples

```
# simulation of healthy controls data

Sigma.ctrl <- matrix(cbind(1, .7,  .7, 1) ,nrow=2)

U <- t(chol(Sigma.ctrl))

numobs <- 100

set.seed(123)

random.normal <- matrix( rnorm( n = ncol(U) * numobs, mean = 3, sd = 1),
                         nrow = ncol(U), ncol = numobs)

X = U %*% random.normal

dat.ctrl <- as.data.frame(t(X))

names(dat.ctrl) <- c("y","x")

cor(dat.ctrl)

# simulation of patient data

Sigma.pt <- matrix(cbind(1, 0,  0, 1) ,nrow=2)

U <- t(chol(Sigma.pt))

numobs <- 20

set.seed(0)

random.normal <- matrix( rnorm( n = ncol(U) * numobs, mean = 3, sd = 1),
                 nrow = ncol(U), ncol = numobs)

X = U %*% random.normal

dat.pt <- as.data.frame(t(X))

names(dat.pt) <- c("y","x")

cor(dat.pt)

# fit the single case model

mdl.reg <- BMSC(y ~ x, data_ctrl = dat.ctrl, data_sc = dat.pt, seed = 10)

# summarize the data

summary(mdl.reg)
```

```
# plot the results of both patient and control group

plot(mdl.reg)

# plot the results of the patient

plot(mdl.reg, who = "single")

# plot the results of the difference between the control group and the patient

plot(mdl.reg, who = "delta")

# density plots

plot(mdl.reg, type = "area")

# histograms

plot(mdl.reg, type = "hist")
```

`plot.pairwise.BMSC` *Plot estimates from a* `pairwise.BMSC` *object.*

Description

Plot estimates from a `pairwise.BMSC` object.

Usage

```
## S3 method for class 'pairwise.BMSC'
plot(x, type = "interval", CI = 0.95, ...)
```

Arguments

`x`	An object of class pairwise.BMSC.
`type`	a parameter to select the typology of graph **interval** the estimates will be represented by means of pointrange, with median and the boundaries of the credible interval **area** a density plot **hist** a density histogram
`CI`	the dimension of the Credible Interval (or Equally Tailed Interval). Default 0.95.
`...`	other arguments are ignored.

Value

a list of two ggplot2 objects

Examples

```
######################################
# simulation of controls' group data
######################################

# Number of levels for each condition and trials
NCond1  <- 2
NCond2  <- 2
Ntrials <- 8
NSubjs  <- 30

betas <- c( 0 , 0 , 0 ,  0.2)

data.sim <- expand.grid(
  trial      = 1:Ntrials,
  ID         = factor(1:NSubjs),
  Cond1      = factor(1:NCond1),
  Cond2      = factor(1:NCond2)
)

contrasts(data.sim$Cond1) <- contr.sum(2)
contrasts(data.sim$Cond2) <- contr.sum(2)

### d.v. generation
y <- rep( times = nrow(data.sim) , NA )

# cheap simulation of individual random intercepts
set.seed(1)
rsubj <- rnorm(NSubjs , sd = 0.1)

for( i in 1:length( levels( data.sim$ID ) ) ){

  sel <- which( data.sim$ID == as.character(i) )

  mm  <- model.matrix(~ Cond1 * Cond2 , data = data.sim[ sel , ] )

  set.seed(1 + i)
  y[sel] <- mm %*% as.matrix(betas + rsubj[i]) +
    rnorm( n = Ntrials * NCond1 * NCond2 )

}

data.sim$y <- y

# just checking the simulated data...
boxplot(y~Cond1*Cond2, data = data.sim)

######################################
# simulation of patient data
######################################
```

```
betas.pt <- c( 0 , 0.8 , 0 ,  0)

data.pt <- expand.grid(
  trial      = 1:Ntrials,
  Cond1      = factor(1:NCond1),
  Cond2      = factor(1:NCond2)
)

contrasts(data.pt$Cond1) <- contr.sum(2)
contrasts(data.pt$Cond2) <- contr.sum(2)

### d.v. generation
mm  <- model.matrix(~ Cond1 * Cond2 , data = data.pt )

set.seed(5)
data.pt$y <- (mm %*% as.matrix(betas.pt) +
                rnorm( n = Ntrials * NCond1 * NCond2 ))[,1]

# just checking the simulated data...
boxplot(y~Cond1*Cond2, data = data.pt)

mdl <- BMSC(y ~ Cond1 * Cond2 + ( 1 | ID ),
            data_ctrl = data.sim, data_sc = data.pt, seed = 77,
            typeprior = "cauchy", s = 1)

summary(mdl)

pp_check(mdl)

# compute pairwise contrasts
ph <- pairwise.BMSC( mdl, contrast = "Cond11:Cond21")

ph

# plot pairwise comparisons

plot(ph)

plot(ph , type = "area")

# customization of pairiwse comparisons plot

plot(ph)[[1]]+theme_bw(base_size = 18)

plot(ph , type = "area")[[1]]+theme_bw(base_size = 18)+
   theme(strip.text.y = element_text( angle = 0))
```

---

pp_check.BMSC *Posterior predictive check for BMSC objects*

Description

`pp_check()` plots the posterior predictive check for BMSC objects.

Usage

```
## S3 method for class 'BMSC'
pp_check(object, type = "dens", limited = FALSE, ...)
```

Arguments

`object`	a BMSC object
`type`	a parameter to select the typology of graph **dens** density overlay plot **hist** histogram plot **mode** the distribution of the mode statistic, over the simulated datasets, compared to the mode of the real data
`limited`	logical. TRUE if the output should be limited within the 95% credible interval, FALSE it should not. Default FALSE.
`...`	other arguments are ignored.

Value

a ggplot2 object

Examples

```
# simulation of healthy controls data

Sigma.ctrl <- matrix(cbind(1, .7,  .7, 1) ,nrow=2)

U <- t(chol(Sigma.ctrl))

numobs <- 100

set.seed(123)

random.normal <- matrix( rnorm( n = ncol(U) * numobs, mean = 3, sd = 1),
                         nrow = ncol(U), ncol = numobs)

X = U %*% random.normal

dat.ctrl <- as.data.frame(t(X))

names(dat.ctrl) <- c("y","x")

cor(dat.ctrl)

# simulation of patient data
```

```
Sigma.pt <- matrix(cbind(1, 0,  0, 1) ,nrow=2)

U <- t(chol(Sigma.pt))

numobs <- 20

set.seed(0)

random.normal <- matrix( rnorm( n = ncol(U) * numobs, mean = 3, sd = 1),
                 nrow = ncol(U), ncol = numobs)

X = U %*% random.normal

dat.pt <- as.data.frame(t(X))

names(dat.pt) <- c("y","x")

cor(dat.pt)

# fit the single case model

mdl.reg <- BMSC(y ~ x, data_ctrl = dat.ctrl, data_sc = dat.pt, seed = 10)

# summarize the data

summary(mdl.reg)

# plot the posterior predictive checks

pp_check(mdl.reg, limited = FALSE)

pp_check(mdl.reg, limited = TRUE)

pp_check(mdl.reg, type = "mode", limited = FALSE)

pp_check(mdl.reg, type = "hist", limited = FALSE)
```

`print.pairwise.BMSC` *Print summaries of Pairwise Bayesian Multilevel Single Case objects*

Description

Print summaries of Pairwise Bayesian Multilevel Single Case objects

Usage

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

Arguments

`x`	An object of class `pairwise.BMSC`, resulting from the pairwise.BMSC function.
`...`	further arguments passed to or from other methods.

`print.summary.BMSC` *Print summaries of Bayesian Multilevel Single Case objects*

Description

Print summaries of Bayesian Multilevel Single Case objects

Usage

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

Arguments

`x`	An object of class `summary.BMSC`, resulting from the summary.BMSC function.
`...`	further arguments passed to or from other methods.

`randomeffects` *Random Effects specification on Bayesian Multilevel Single Case models using 'Stan'*

Description

The **BMSC** function allows the flexibility of multilevel (generalised) linear models on single case analysis.

In particular, it is possible to specify the population-level (a.k.a. mixed effects) and the group-level (a.k.a. random effects) coefficients.

The specification of the population- and group-level effects can be done using the well-known **lme4** notation with specific limitations:

- it is no possible to estimate uncorrelated group-level effects
- it is no possible to directly estimate nested effects. You need to use a trick that is specified in the **Details** section.

Details

lmer formulation	BMSC availability
`(1 | grouping_factor)`	Yes
`(1 + slope | grouping_factor)`	Yes
`(0 + slope | grouping_factor)`	No
`(1 | grouping_factor1 : grouping_factor2)`	Yes[^1]
`(1 | grouping_factor1 / grouping_factor2)`	Yes[^2]

[^1]: The **BMSC** function dose not allow the use of the interaction symbol ":", but this problem is easily solved by creating a new variable within your dataframe given by the interaction of the two factors.

[^2]: The `(1 | grouping_factor1 / grouping_factor2)` syntax is the equivalent of the explicit version `(1 \| grouping_factor1:grouping_factor2) + (1 | grouping_factor1)`.

Therefore, you need to create a new grouping factor representing the interaction between `grouping_factor1` and `grouping_factor2`, and use this in the explicit version `(1 | grouping_factor_interaction) + (1 | grouping_factor1)`.

`summary.BMSC` *Summarizing Bayesian Multilevel Single Case objects*

Description

summary method for class "BMSC".

Usage

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

Arguments

`object`	An object of class `BMSC`, resulting from the BMSC function.
`...`	other arguments are ignored.

Value

a `summary.BMSC` object

Index