Package: bayesmsm (via r-universe)

October 27, 2026

Type Package

Title Fitting Bayesian Marginal Structural Models for Longitudinal Observational Data

Version 1.0.0

Maintainer Kuan Liu <kuan.liu@utoronto.ca>

Description Implements Bayesian marginal structural models for causal effect estimation with time-varying treatment and confounding. It includes an extension to handle informative right censoring. The Bayesian importance sampling weights are estimated using JAGS. See Saarela (2015) <doi:10.1111/biom.12269> for methodological details.

License MIT + file LICENSE

URL https://github.com/Kuan-Liu-Lab/bayesmsm

BugReports https://github.com/Kuan-Liu-Lab/bayesmsm/issues

Depends R (>= 4.2.0)

Imports coda (>= 0.19-4), doParallel, foreach, ggplot2, graphics, grDevices, MCMCpack, parallel, R2jags, stats

Suggests devtools, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.2

Author Kuan Liu [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-5017-1276>), Xiao Yan [aut] (ORCID: <https://orcid.org/0000-0001-9712-1199>), Martin Urner [aut] (ORCID: <https://orcid.org/0000-0003-1133-4885>)

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

Date/Publication 2025-06-17 06:00:02 UTC

RemoteUrl https://github.com/cran/bayesmsm

RemoteRef HEAD

RemoteSha 10d5e447f497b682606178f9ea982d7d8f7546e4

Contents

bayesmsm 2
bayesweight 4
bayesweight_cen 6
calculate_effect 8
plot_APO 9
plot_ATE 10
plot_est_box 11
simData 13
summary_bayesmsm 14

Index **16**

`bayesmsm` *Bayesian Marginal Structural Model Bootstrap Estimation*

Description

This function performs Bayesian non-parametric bootstrap to estimate causal effects in Bayesian marginal structural models. It supports both continuous (Gaussian) and binary (binomial) outcome variables

Usage

```
bayesmsm(
  ymodel,
  nvisit,
  reference = c(rep(0, nvisit)),
  comparator = c(rep(1, nvisit)),
  treatment_effect_type = "sq",
  family = "gaussian",
  data,
  wmean = rep(1, nrow(data)),
  nboot = 1000,
  optim_method = "BFGS",
  seed = NULL,
  parallel = TRUE,
  ncore = 4
)
```

Arguments

ymodel	Model statement for the outcome variable.
nvisit	Number of visits or time points to simulate.
reference	Vector denoting the intervention to be used as the reference across all visits for calculating the risk ratio and risk difference. The default is a vector of all 0's with length nvisit (i.e. never treated).
comparator	Vector denoting the intervention to be used as the comparator across all visits for calculating the risk ratio and risk difference. The default is a vector of all 1's with length nvisit (i.e. always treated).
treatment_effect_type	Character string specifying the type of treatment effect to estimate. Options are "sq" for sequential treatment effects, which estimates effects for specific treatment sequences across visits, and "cum" for cumulative treatment effects, which assumes a single cumulative treatment variable representing the total exposure. The default is "sq".
family	Character string specifying the outcome distribution family. The possible distributions are: "Gaussian" (default) for continuous outcomes, and "binomial" for binary outcomes.
data	Data table containing the variable names in the outcome model.
wmean	Vector of treatment assignment weights. The default is rep(1, nrow(data)).
nboot	Integer specifying the number of bootstrap iterations. The default is 1000.
optim_method	Character string specifying the optimization method to be used. The default is "BFGS".
seed	Starting seed for simulations and bootstrapping. The default is NULL.
parallel	Logical scalar indicating whether to parallelize bootstrapping to multiple cores. The default is TRUE.
ncore	Integer specifying the number of CPU cores to use in parallel simulation. This argument is required when parallel is set to TRUE, and the default is 4.

Value

It returns an object of class "bayesmsm" that contains the information about the data, model, etc. An object of class "bayesmsm" is a list containing at least the following components: "mean", the mean of the bootstrap estimates; "sd", the standard deviation of the bootstrap estimates; "quantile", the 95% quantiles of the bootstrap estimates; "bootdata", a data frame of bootstrapped estimates; "reference", the reference intervention level and "comparator", the comparison intervention level

Examples

```
# 1) Specify simple treatment-assignment models
amodel <- list(
  c("(Intercept)" =  0, "L1_1" =  0.5, "L2_1" = -0.5),
  c("(Intercept)" =  0, "L1_2" =  0.5, "L2_2" = -0.5, "A_prev" = 0.3)
)
# 2) Specify a continuous-outcome model
```

```
ymodel <- c("(Intercept)" = 0,
            "A1"          = 0.2,
            "A2"          = 0.3,
            "L1_2"        = 0.1,
            "L2_2"        = -0.1)
# 3) Simulate without right-censoring
testdata <- simData(
  n                = 200,
  n_visits         = 2,
  covariate_counts = c(2, 2),
  amodel           = amodel,
  ymodel           = ymodel,
  y_type           = "continuous",
  right_censor     = FALSE,
  seed             = 123)
model <- bayesmsm(ymodel = Y ~ A1 + A2,
                  nvisit = 2,
                  reference = c(rep(0,2)),
                  comparator = c(rep(1,2)),
                  treatment_effect_type = "sq",
                  family = "binomial",
                  data = testdata,
                  wmean = rep(1,200),
                  nboot = 10,
                  optim_method = "BFGS",
                  seed = 890123,
                  parallel = FALSE)
```

---

bayesweight *Bayesian Treatment Effect Weight Estimation Using JAGS*

Description

This function estimates Bayesian importance sampling weights for time-varying treatment effects using specified models for each treatment time point via JAGS

Usage

```
bayesweight(
  trtmodel.list,
  data,
  n.chains = 2,
  n.iter = 25000,
  n.burnin = 15000,
  n.thin = 5,
  seed = NULL,
  parallel = TRUE
)
```

Arguments

trtmodel.list	A list of formulas corresponding to each time point with the time-specific treatment variable on the left hand side and pre-treatment covariates to be balanced on the right hand side. The formulas must be in temporal order, and must contain all covariates to be balanced at that time point. Interactions and functions of covariates are allowed.
data	A data set in the form of a data frame containing the variables in "trtmodel.list". This must be a wide data set with exactly one row per unit.
n.chains	Integer specifying the number of MCMC chains to run. Set to 1 for non-parallel computation. For parallel computation, it is required to use at least 2 chains. The default is 2.
n.iter	Integer specifying the total number of iterations for each chain (including burn-in). The default is 25000.
n.burnin	Integer specifying the number of burn-in iterations for each chain. The default is 15000.
n.thin	Integer specifying the thinning rate for the MCMC sampler. The default is 5.
seed	Starting seed for the JAGS model. The default is NULL.
parallel	Logical scalar indicating whether to run the MCMC chains in parallel. The default is TRUE.

Value

A list of the calculated weights and the JAGS model, where ‘weights‘ is a vector of posterior mean weights, computed by taking the average of the weights across all MCMC iterations and ‘model_string‘ is a character of the JAGS model based on the input of ‘trtmodel.list‘.

Examples

```
# 1) Specify simple treatment-assignment models
amodel <- list(
  c("(Intercept)" =  0, "L1_1" =  0.5, "L2_1" = -0.5),
  c("(Intercept)" =  0, "L1_2" =  0.5, "L2_2" = -0.5, "A_prev" = 0.3)
)
# 2) Specify a continuous-outcome model
ymodel <- c("(Intercept)" = 0,
            "A1"          = 0.2,
            "A2"          = 0.3,
            "L1_2"        = 0.1,
            "L2_2"        = -0.1)
# 3) Simulate without right-censoring
testdata <- simData(
  n               = 200,
  n_visits        = 2,
  covariate_counts = c(2, 2),
  amodel          = amodel,
  ymodel          = ymodel,
  y_type          = "continuous",
  right_censor    = FALSE,
```

```
  seed             = 123)
weights <- bayesweight(trtmodel.list = list(
                       A1 ~ L1_1 + L2_1,
                       A2 ~ L2_2 + L2_2 + A1),
                       data = testdata,
                       n.chains = 1,
                       n.iter = 20,
                       n.burnin = 10,
                       n.thin = 1,
                       seed = 890123,
                       parallel = FALSE)
summary(weights)
```

`bayesweight_cen` *Bayesian Treatment Effect Weight Estimation for Censored Data*

Description

This function estimates Bayesian importance sampling weights for treatment models and censoring models across multiple time points via JAGS

Usage

```
bayesweight_cen(
  trtmodel.list,
  cenmodel.list,
  data,
  n.chains = 2,
  n.iter = 25000,
  n.burnin = 15000,
  n.thin = 5,
  seed = NULL,
  parallel = TRUE
)
```

Arguments

`trtmodel.list`	A list of formulas corresponding to each time point with the time-specific treatment variable on the left-hand side and pre-treatment covariates to be balanced on the right-hand side. The formulas must be in temporal order, and must contain all covariates to be balanced at that time point. Interactions and functions of covariates are allowed.
`cenmodel.list`	A list of formulas for the censored data at each time point, with censoring indicators on the left-hand side and covariates on the right-hand side. The formulas must be in temporal order, and must contain all covariates to be balanced at that time point.
`data`	A data set in the form of a data frame containing the variables in "trtmodel.list" and "cenmodel.list". This must be a wide data set with exactly one row per unit.

n.chains	Integer specifying the number of MCMC chains to run. Set to 1 for non-parallel computation. For parallel computation, it is required to use at least 2 chains. The default is 2.
n.iter	Integer specifying the total number of iterations for each chain (including burn-in). The default is 25000.
n.burnin	Integer specifying the number of burn-in iterations for each chain. The default is 15000.
n.thin	Integer specifying the thinning rate for the MCMC sampler. The default is 5.
seed	Starting seed for the JAGS model. The default is NULL.
parallel	Logical scalar indicating whether to run the MCMC chains in parallel. The default is TRUE.

Value

A list of the calculated weights and the JAGS model where "weights" is a vector of posterior mean weights, computed by taking the average of the weights across all MCMC iterations and 'model_string' is a character of the JAGS model based on the input of "trtmodel.list".

Examples

```
amodel <- list(
 c("(Intercept)" = -0.3, "L1_1" = 0.4, "L2_1" = -0.2),
 c("(Intercept)" = -0.1, "L1_2" = 0.3, "L2_2" = -0.1, "A_prev" = 0.5))
ymodel <- c(
"(Intercept)" = -0.8,
"A1"          = 0.2,
"A2"          = 0.4,
"L1_2"        = 0.3,
"L2_2"        = -0.3)
cmodel <- list(
 c("(Intercept)" = -1.5, "L1_1" = 0.2, "L2_1" = -0.2, "A" = 0.2),
 c("(Intercept)" = -1.5, "L1_2" = 0.1, "L2_2" = -0.1, "A" = 0.3))
testdata <- simData(
 n                = 50,
 n_visits         = 2,
 covariate_counts = c(2, 2),
 amodel           = amodel,
 ymodel           = ymodel,
 y_type           = "binary",
 right_censor     = TRUE,
 cmodel           = cmodel,
 seed             = 123
)
weights_cen <- bayesweight_cen(
                trtmodel.list = list(
                       A1 ~ L1_1 + L2_1,
                       A2 ~ L2_2 + L2_2 + A1),
                cenmodel.list = list(
                C1 ~ L1_1 + L2_1 + A1,
                C2 ~ L1_2 + L2_2 + A2),
```

```
                data = testdata,
                n.chains = 1,
                n.iter = 20,
                n.burnin = 10,
                n.thin = 1,
                seed = 890123,
                parallel = FALSE)
summary(weights_cen)
```

`calculate_effect` *This function to calculate the causal effect of an intervention given the parameter estimates and intervention levels*

Description

This function to calculate the causal effect of an intervention given the parameter estimates and intervention levels

Usage

```
calculate_effect(
  intervention_levels,
  variables,
  param_estimates,
  treatment_effect_type
)
```

Arguments

`intervention_levels`	A numeric vector indicating the levels of intervention for each predictor variable.
`variables`	A list of the names of the response variable and predictor variables extracted from the model.
`param_estimates`	A vector of parameter estimates from the model.
`treatment_effect_type`	Character string specifying the type of treatment effect to estimate. Options are "sq" for sequential treatment effects, which estimates effects for specific treatment sequences across visits, and "cum" for cumulative treatment effects, which assumes a single cumulative treatment variable representing the total exposure. The default is "sq".

Value

A numeric value representing the calculated effect of the specified intervention.

plot_APO *Plot Average Potential Outcomes (APO)*

Description

This function plots the density of APO for a specified effect type from bayesmsm output.

Usage

```
plot_APO(input, effect_type, ...)
```

Arguments

input	A data frame or model object containing bootstrap results.
effect_type	A character string specifying which effect to plot (e.g., comparator or reference treatment sequences).
...	Additional arguments passed to the plotting function.

Value

A ggplot object representing density plot showing the distribution of the specified average potential outcome (reference or comparison).

Examples

```
# 1) Specify simple treatment-assignment models
amodel <- list(
  c("(Intercept)" =  0, "L1_1" =  0.5, "L2_1" = -0.5),
  c("(Intercept)" =  0, "L1_2" =  0.5, "L2_2" = -0.5, "A_prev" = 0.3)
)
# 2) Specify a continuous-outcome model
ymodel <- c("(Intercept)" = 0,
            "A1"          = 0.2,
            "A2"          = 0.3,
            "L1_2"        = 0.1,
            "L2_2"        = -0.1)
# 3) Simulate without right-censoring
testdata <- simData(
  n                = 200,
  n_visits         = 2,
  covariate_counts = c(2, 2),
  amodel           = amodel,
  ymodel           = ymodel,
  y_type           = "continuous",
  right_censor     = FALSE,
  seed             = 123)
model <- bayesmsm(ymodel = Y ~ A1 + A2,
                  nvisit = 2,
                  reference = c(rep(0,2)),
```

```
                    comparator = c(rep(1,2)),
                    treatment_effect_type = "sq",
                    family = "binomial",
                    data = testdata,
                    wmean = rep(1,200),
                    nboot = 10,
                    optim_method = "BFGS",
                    seed = 890123,
                    parallel = FALSE)
plot_APO(model$bootdata, effect_type = "effect_comparator")
plot_APO(model, effect_type = "effect_reference")
```

---

plot_ATE *Plot Average Treatment Effect Density from bayesmsm output*

Description

This function plots the density of ATE from bayesmsm output.

Usage

```
plot_ATE(
  input,
  ATE = "RD",
  col_density = "blue",
  fill_density = "lightblue",
  main = "Posterior Predictive Distribution of Average Treatment Effect",
  xlab = "ATE",
  ylab = "Posterior Predictive Distribution",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

input	A model object, data frame or vector containing the bootstrap estimates of ATE.
ATE	define causal estimand of interest from RD, OR, RR.
col_density	Color for the density plot (default is "blue").
fill_density	Fill color for the density plot (default is "lightblue").
main	Title of the plot (default is "Density of ATE Estimates").
xlab	X-axis label (default is "ATE").
ylab	Y-axis label (default is "Density").
xlim	Limits for the x-axis (default is NULL).
ylim	Limits for the y-axis (default is NULL).
...	Additional graphical parameters passed to the plot function.

Value

A ggplot object representing the density plot for the posterior predictive distribution of the Average Treatment Effect (ATE).

Examples

```
# 1) Specify simple treatment-assignment models
amodel <- list(
  c("(Intercept)" =  0, "L1_1" =  0.5, "L2_1" = -0.5),
  c("(Intercept)" =  0, "L1_2" =  0.5, "L2_2" = -0.5, "A_prev" = 0.3)
)
# 2) Specify a continuous-outcome model
ymodel <- c("(Intercept)" = 0,
            "A1"          = 0.2,
            "A2"          = 0.3,
            "L1_2"        = 0.1,
            "L2_2"        = -0.1)
# 3) Simulate without right-censoring
testdata <- simData(
  n                = 200,
  n_visits         = 2,
  covariate_counts = c(2, 2),
  amodel           = amodel,
  ymodel           = ymodel,
  y_type           = "continuous",
  right_censor     = FALSE,
  seed             = 123)
model <- bayesmsm(ymodel = Y ~ A1 + A2,
                  nvisit = 2,
                  reference = c(rep(0,2)),
                  comparator = c(rep(1,2)),
                  treatment_effect_type = "sq",
                  family = "binomial",
                  data = testdata,
                  wmean = rep(1,200),
                  nboot = 10,
                  optim_method = "BFGS",
                  seed = 890123,
                  parallel = FALSE)
plot_ATE(model)
```

`plot_est_box` *Error bar plots for causal treatment effects*

Description

This function plots the point estimates and 95% credible intervals of ATE and APO from bayesmsm output.

Usage

```
plot_est_box(input, ...)
```

Arguments

input	A data frame or model object containing bootstrap results.
...	Additional arguments passed to the plotting function.

Value

A ggplot object presenting error bar plot of the mean effects and their 95% credible intervals for comparator level, reference level, and ATE.

Examples

```
# 1) Specify simple treatment-assignment models
amodel <- list(
  c("(Intercept)" =  0, "L1_1" =  0.5, "L2_1" = -0.5),
  c("(Intercept)" =  0, "L1_2" =  0.5, "L2_2" = -0.5, "A_prev" = 0.3)
)
# 2) Specify a continuous-outcome model
ymodel <- c("(Intercept)" = 0,
            "A1"          = 0.2,
            "A2"          = 0.3,
            "L1_2"        = 0.1,
            "L2_2"        = -0.1)
# 3) Simulate without right-censoring
testdata <- simData(
  n                = 200,
  n_visits         = 2,
  covariate_counts = c(2, 2),
  amodel           = amodel,
  ymodel           = ymodel,
  y_type           = "continuous",
  right_censor     = FALSE,
  seed             = 123)
model <- bayesmsm(ymodel = Y ~ A1 + A2,
                  nvisit = 2,
                  reference = c(rep(0,2)),
                  comparator = c(rep(1,2)),
                  treatment_effect_type = "sq",
                  family = "binomial",
                  data = testdata,
                  wmean = rep(1,200),
                  nboot = 10,
                  optim_method = "BFGS",
                  seed = 890123,
                  parallel = FALSE)
plot_est_box(model)
```

simData *Generate synthetic longitudinal data with optional right-censoring*

Description

This function simulates repeated measurements of normally-distributed covariates, binary treatments, and an end-of-study outcome for longitudinal causal analyses. When ‘right_censor = TRUE‘, a right-censoring indicator ‘Cj‘ is generated at each visit: if ‘Cj = 1‘, all subsequent ‘L‘, ‘A‘, and ‘Y‘ values are set to ‘NA‘.

Usage

```
simData(
  n,
  n_visits,
  covariate_counts = rep(2, n_visits),
  amodel,
  ymodel,
  y_type = c("binary", "continuous"),
  right_censor = FALSE,
  cmodel = NULL,
  seed = NULL
)
```

Arguments

n	Integer. Sample size.
n_visits	Integer. Number of visits (including baseline as visit 1).
covariate_counts	Integer vector of length ‘n_visits‘. Number of covariates per visit (default: rep(2, n_visits)).
amodel	List of length ‘n_visits‘. Each element is a named numeric vector of coefficients for the logistic model of treatment ‘Aj‘ on covariates (and ‘A_prev‘ for j > 1).
ymodel	Named numeric vector. Coefficients for the end-of-study outcome model. If ‘y_type = "binary"‘, a logistic model is used; if ‘"continuous"‘, a linear model with Gaussian noise.
y_type	Character. One of "binary" or "continuous".
right_censor	Logical. If TRUE, generates ‘Cj‘ using ‘cmodel‘ at each visit.
cmodel	List of length ‘n_visits‘. Named numeric vectors for logistic censoring models at each visit, regressing ‘Cj‘ on covariates and current ‘Aj‘.
seed	Integer. Optional random seed.

Value

A ‘data.frame‘ with columns ‘Lk_j‘, ‘Aj‘, optional ‘Cj‘, and ‘Y‘.

summary_bayesmsm *Summary function to generate result table from bayesmsm*

Description

This function generates a ready to use result table that contents the estimated APO and ATE and their 95% credible intervals

Usage

```
summary_bayesmsm(model)
```

Arguments

model	A model object from bayesmsm

Value

A summary table of the results from bayesmsm.

Examples

```
# 1) Specify simple treatment-assignment models
amodel <- list(
  c("(Intercept)" =  0, "L1_1" =  0.5, "L2_1" = -0.5),
  c("(Intercept)" =  0, "L1_2" =  0.5, "L2_2" = -0.5, "A_prev" = 0.3)
)
# 2) Specify a continuous-outcome model
ymodel <- c("(Intercept)" = 0,
            "A1"          = 0.2,
            "A2"          = 0.3,
            "L1_2"        = 0.1,
            "L2_2"        = -0.1)
# 3) Simulate without right-censoring
testdata <- simData(
  n                = 200,
  n_visits         = 2,
  covariate_counts = c(2, 2),
  amodel           = amodel,
  ymodel           = ymodel,
  y_type           = "continuous",
  right_censor     = FALSE,
  seed             = 123)
model <- bayesmsm(ymodel = Y ~ A1 + A2,
                  nvisit = 2,
                  reference = c(rep(0,2)),
                  comparator = c(rep(1,2)),
                  treatment_effect_type = "sq",
                  family = "binomial",
                  data = testdata,
```

```
                    wmean = rep(1,200),
                    nboot = 10,
                    optim_method = "BFGS",
                    seed = 890123,
                    parallel = FALSE)
summary_bayesmsm(model)
```

Index

bayesmsm, 2
bayesweight, 4
bayesweight_cen, 6

calculate_effect, 8

plot_APO, 9
plot_ATE, 10
plot_est_box, 11

simData, 13
summary_bayesmsm, 14