

Package ‘BayesRegDTR’

October 26, 2025

Type Package

Title Bayesian Regression for Dynamic Treatment Regimes

Version 1.1.1

Description Methods to estimate optimal dynamic treatment regimes using Bayesian likelihood-based regression approach as described in
Yu, W., & Bondell, H. D. (2023) <[doi:10.1093/jrsssb/qkad016](https://doi.org/10.1093/jrsssb/qkad016)>
Uses backward induction and dynamic programming theory for computing expected values. Offers options for future parallel computing.

License GPL (>= 3)

Imports Rcpp (>= 1.0.13-1), mvtnorm, foreach, progressr, stats, future

Depends doRNG

Suggests cli, testthat (>= 3.0.0), doFuture

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.3.3

URL <https://github.com/jlimrasc/BayesRegDTR>

BugReports <https://github.com/jlimrasc/BayesRegDTR/issues>

Config/testthat/edition 3

NeedsCompilation yes

Author Jeremy Lim [aut, cre],
Weichang Yu [aut] (ORCID: <<https://orcid.org/0000-0002-0399-3779>>)

Maintainer Jeremy Lim <jeremylim23@gmail.com>

Repository CRAN

Date/Publication 2025-10-26 18:30:09 UTC

Contents

BayesRegDTR-package	2
BayesLinRegDTR.model.fit	2
generate_dataset	6

Index

7

BayesRegDTR-package *BayesRegDTR: Bayesian Regression for Dynamic Treatment Regimes*

Description

Methods to estimate optimal dynamic treatment regimes using Bayesian likelihood-based regression approach as described in Yu, W., & Bondell, H. D. (2023) doi:10.1093/jrsssb/qkad016 Uses backward induction and dynamic programming theory for computing expected values. Offers options for future parallel computing.

Author(s)

Maintainer: Jeremy Lim <jeremylim23@gmail.com>

Authors:

- Weichang Yu <weichang.yu@unimelb.edu.au> (ORCID)

References

Yu, W., & Bondell, H. D. (2023), “Bayesian Likelihood-Based Regression for Estimation of Optimal Dynamic Treatment Regimes”, Journal of the Royal Statistical Society Series B: Statistical Methodology, 85(3), 551-574. doi:10.1093/jrsssb/qkad016

See Also

[generate_dataset\(\)](#) for generating a toy dataset to test the model fitting on

[BayesLinRegDTR.model.fit\(\)](#) for obtaining an estimated posterior distribution of the optimal treatment option at a user-specified prediction stage

Useful links:

- <https://github.com/jlimrasc/BayesRegDTR>
- Report bugs at <https://github.com/jlimrasc/BayesRegDTR/issues>

BayesLinRegDTR.model.fit

Main function for fitting a Bayesian likelihood-based linear regression model

Description

Fits the Bayesian likelihood-based linear model to obtain an estimated posterior distribution of the optimal treatment option at a user-specified prediction stage. Uses backward induction and dynamic programming theory for computing expected values.

Usage

```

BayesLinRegDTR.model.fit(
  Dat.train,
  Dat.pred,
  n.train,
  n.pred,
  num_stages,
  num_treats,
  p_list,
  t,
  R = 30,
  tau = 0.01,
  B = 10000,
  nu0 = 3,
  V0 = mapply(diag, p_list, SIMPLIFY = FALSE),
  alph = 1,
  gam = 1,
  showBar = TRUE
)

```

Arguments

Dat.train	Training data in format returned by generate_dataset: organised as a list of $\{y, X_1, X_2, \dots, X_{num_stages}, A\}$ where y is a vector of the final outcomes, $X_1, X_2, \dots, X_{num_stages}$ is a list of matrices of the intermediate covariates and A is an $n.train \times num_stages$ matrix of the assigned treatments, where num_stages is the total number of stages
Dat.pred	Prediction data in format returned by generate_dataset: organised as a list of $\{X_1, X_2, \dots, X_t, A\}$ where $X_1, X_2, \dots, X_t$ is a list of matrices of the intermediate covariates and A is an $n.pred \times (t - 1)$ matrix of the assigned treatments, where t is the prediction stage
n.train	Number of samples/individuals in the training data
n.pred	Number of samples/individuals in the prediction data
num_stages	Total number of stages
num_treats	Vector of number of treatment options at each stage
p_list	Vector of intermediate covariate dimensions for each stage
t	Prediction stage t , where $t \leq num_stages$
R	Draw size from distribution of intermediate covariates. default: 30
tau	Normal prior scale parameter for regression coefficients. Should be specified with a small value. default: 0.01
B	Number of MC draws from posterior of regression parameters. default 10000
nu0	Inverse-Wishart prior degrees of freedom for regression error Vcov matrix. Ignored if using a univariate dataset. default: 3
V0	List of Inverse-Wishart prior scale matrix for regression error Vcov matrix. Ignored if using a univariate dataset. default: list of identity matrices

alph	Inverse-Gamma prior shape parameter for regression error variance of y. default: 1
gam	Inverse-Gamma prior rate parameter for regression error variance of y. default: 1
showBar	Whether to show a progress bar. Uses API from progressr and future for parallel integration default: TRUE

Details

Utilises a [future](#) framework, so to enable parallel processing and register a parallel backend, [plan](#) and [registerDoFuture](#) must be called first.

Additionally, progress bars use [progressr](#) API, and a non-default progress bar (e.g. cli) is recommended. See below or [registerDoFuture](#) and [handlers](#) for examples.

Note that to have a progress bar for the parallel sections, future must be used. To turn off the immediate warnings, use `options(BRDTR_warn_imm = FALSE)`.

Value

GCV_results	An array of dimension $n.pred \times num_treats[t] \times B$, indicating the expected value under each treatment option at stage t.
post.prob	An $n.pred \times num_treats[t]$ matrix of the posterior probability that each treatment type at stage t is optimal
MC_draws.train	A list of Monte Carlo draws containing: • <i>sigmat_B_list</i> - A list of length num_stages with each element a vector of size $B \times p_list[t]$ • <i>Wt_B_list</i> - A list of length num_stages with each element a matrix of size $B \times p_list[t]$ • <i>beta_B</i> - A list of length B • <i>sigmay_2B</i> - A list of length B

Examples

```
# Code does not run within 10 seconds, so don't run

# -----
# Set Up Parallelism & Progress Bar
# -----
progressr::handlers("cli")      # Set handler to something with title/text
numCores <- parallel::detectCores() # Detect number of cores, use max
future::plan(future::multisession, # Or plan(multicore, workers) on Unix
             workers = numCores)   # Set number of cores to use
doFuture::registerDoFuture()      # Or doParallel::registerDoParallel()
                                # if no progress bar is needed and future
                                # is unwanted

## UVT
# -----
# Initialise Inputs
```

```
# -----
num_stages <- 5
t <- 3
p_list <- rep(1, num_stages)
num_treats <- rep(2, num_stages)
n.train <- 5000
n.pred <- 10

# -----
# Generate Dataset
# -----
Dat.train <- generate_dataset(n.train, num_stages, p_list, num_treats)
Dat.pred <- generate_dataset(n.pred, num_stages, p_list, num_treats)
Dat.pred <- Dat.pred[-1]
Dat.pred[[num_stages+1]] <- Dat.pred[[num_stages+1]][1:n.pred, 1:(t-1), drop = FALSE]

# -----
# Main
# -----
gcv_uvt <- BayesLinRegDTR.model.fit(Dat.train, Dat.pred, n.train, n.pred,
                                   num_stages, num_treats,
                                   p_list, t, R = 30,
                                   tau = 0.01, B = 500, nu0 = NULL,
                                   V0 = NULL, alph = 3, gam = 4)

## MVT
# -----
# Initialise Inputs
# -----
num_stages <- 3
t <- 2
p_list <- rep(2, num_stages)
num_treats <- rep(2, num_stages)
n.train <- 5000
n.pred <- 10

# -----
# Generate Dataset
# -----
Dat.train <- generate_dataset(n.train, num_stages, p_list, num_treats)
Dat.pred <- generate_dataset(n.pred, num_stages, p_list, num_treats)
Dat.pred <- Dat.pred[-1]
Dat.pred[[num_stages+1]] <- Dat.pred[[num_stages+1]][1:n.pred, 1:(t-1), drop = FALSE]

# -----
# Main
# -----
gcv_res <- BayesLinRegDTR.model.fit(Dat.train, Dat.pred, n.train, n.pred,
                                   num_stages, num_treats,
                                   p_list, t, R = 30,
                                   tau = 0.01, B = 500, nu0 = 3,
                                   V0 = mapply(diag, p_list, SIMPLIFY = FALSE),
                                   alph = 3, gam = 4)
```

generate_dataset	<i>Generate a toy dataset in the right format for testing BayesLin-RegDTR.model.fit</i>
------------------	---

Description

Generates a toy dataset simulating observed data of treatments over time with final outcomes and intermediate covariates. Follows the method outlined in [Toy-Datagen on Github](#)

Usage

```
generate_dataset(n, num_stages, p_list, num_treats)
```

Arguments

n	Number of samples/individuals to generate
num_stages	Total number of stages per individual
p_list	Vector of dimension for each stage
num_treats	Vector of number of treatment options at each stage

Value

Observed data organised as a list of $\{y, X_1, X_2, \dots, X_{num_stages}, A\}$ where y is a vector of the final outcomes, $X_1, X_2, \dots, X_{num_stages}$ is a list of matrices of the intermediate covariates and A is an $n \times num_stages$ matrix of the assigned treatments

Examples

```
# -----
# Initialise Inputs
# -----
n          <- 5000
num_stages <- 3
p_list_uvt <- rep(1, num_stages)
p_list_mvt <- c(1, 3, 3)
num_treats <- rep(3, num_stages)

# -----
# Main
# -----
Data_uvt <- generate_dataset(n, num_stages, p_list_uvt, num_treats)
Data_mvt <- generate_dataset(n, num_stages, p_list_mvt, num_treats)
```

Index

BayesLinRegDTR.model.fit, [2](#)
BayesLinRegDTR.model.fit(), [2](#)
BayesRegDTR (BayesRegDTR-package), [2](#)
BayesRegDTR-package, [2](#)

future, [4](#)

generate_dataset, [6](#)
generate_dataset(), [2](#)

handlers, [4](#)

plan, [4](#)
progressr, [4](#)

registerDoFuture, [4](#)