Package 'SBMTrees'

December 11, 2024

Type Package

Title Sequential Imputation with Bayesian Trees Mixed-Effects Models for Longitudinal Data

Version 1.2

Date 2024-11-20

Author Jungang Zou [aut, cre],
Liangyuan Hu [aut],
Robert McCulloch [ctb],
Rodney Sparapani [ctb],
Charles Spanbauer [ctb]

Maintainer Jungang Zou <jungang.zou@gmail.com>

Description Implements a sequential imputation framework using Bayesian Mixed-Effects Trees ('SBMTrees') for handling missing data in longitudinal studies. The package supports a variety of models, including non-linear relationships and non-normal random effects and residuals, leveraging Dirichlet Process priors for increased flexibility. Key features include handling Missing at Random (MAR) longitudinal data, imputation of both covariates and outcomes, and generating posterior predictive samples for further analysis. The methodology is designed for applications in epidemiology, biostatistics, and other fields requiring robust handling of missing data in longitudinal settings.

License GPL-2

Encoding UTF-8

Depends R (>= 4.1.0)

Imports Rcpp, lme4, Matrix, arm, dplyr, mvtnorm, sn, tidyr, mice, nnet

LinkingTo Rcpp, RcppArmadillo, RcppDist, RcppProgress

RoxygenNote 7.3.2

SystemRequirements GNU make

Suggests knitr, rmarkdown, mitml

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-12-11 07:50:10 UTC

Contents

`SBMTrees-package` *Sequential Imputation with Bayesian Trees Mixed-Effects Models*

Description

The SBMTrees package implements a Bayesian non-parametric framework for imputing missing covariates and outcomes in longitudinal data under the Missing at Random (MAR) assumption. Its core model, the Bayesian Trees Mixed-Effects Model (BMTrees), extends Mixed-Effects BART by employing centralized Dirichlet Process (CDP) Normal Mixture priors. This allows handling non-normal random effects and errors, addressing model misspecification, and capturing complex relationships.

Details

SBMTrees offers tools for predicting and imputing missing values in longitudinal data using Bayesian Trees Mixed-Effects Models. The package supports various semiparametric variants, including BMTrees_R and BMTrees_RE, and integrates mixedBART as a baseline model. Key functionalities include:

- `BMTrees_prediction`: Predicts longitudinal outcomes based on mixed-effects models.

- `sequential_imputation`: Imputes missing covariates and outcomes sequentially in longitudinal datasets.

The package supports flexibility in specifying priors for random effects and errors, making it suitable for diverse longitudinal data settings. Core computations leverage efficient Gibbs samplers implemented in C++.

This package modifies and extends C++ code originally derived from the BART3 package, developed by Rodney Sparapani, which is licensed under the GNU General Public License version 2 (GPL-2).

The modified code is redistributed in accordance with the GPL-2 license. For more details on the modifications, see the package's documentation.

Note

This package and all associated documentation are licensed under the GNU General Public License version 2 (GPL-2). See the `LICENSE` file for the full text of the license.

Author(s)

Jungang Zou <jungang.zou@gmail.com>

References

BART3 package: https://github.com/rsparapa/bnptools/tree/master, originally developed by Rodney Sparapani.

See Also

BMTrees_prediction, sequential_imputation

Examples

```
## Example of predicting longitudinal outcomes
## To make it faster to compile and check, we only run 30 iterations for
## burn-in and 40 for posterior sampling phases.
## Please increase to 3000 and 4000 iterations, respectively, when running the models.

data <- simulation_prediction(n_subject = 100, seed = 1234, nonlinear = TRUE,
        nonrandeff = TRUE, nonresidual = TRUE)
X_train <- data$X_train
Y_train <- data$Y_train
Z_train <- data$Z_train
subject_id_train <- data$subject_id_train

X_test <- data$X_test
Z_test <- data$Z_test
subject_id_test <- data$subject_id_test

model <- BMTrees_prediction(X_train, Y_train, Z_train, subject_id_train,
         X_test, Z_test, subject_id_test, model = "BMTrees", seed = 1234)
model$post_predictive_y_test



data2 = simulation_imputation(n_subject = 100, seed = 1234, nonrandeff = TRUE,
        nonresidual = TRUE, alligned = FALSE)
X_mis = data2$X_mis # get missing covariates
Y_mis = data2$Y_mis # get missing outcomes
Z = data2$Z # get random predictors
subject_id = data2$subject_id  # get subject id

model2 = sequential_imputation(X_mis, Y_mis, Z, subject_id, rep(0, 9), FALSE,
        model = "BMTrees", nburn = 30L, npost = 40L, skip = 2L,
        verbose = TRUE, seed = 1234)
model2$imputed_data
model2$imputed_data[,,10]
```

apply_locf_nocb *Impute Missing Values Using LOCF and NOCB*

Description

Imputes missing values in a matrix by applying Last Observation Carried Forward (LOCF) followed by Next Observation Carried Backward (NOCB) for each subject.

Usage

```
apply_locf_nocb(X, subject_id)
```

Arguments

X	A matrix where rows represent observations and columns represent variables.
subject_id	A vector of subject IDs corresponding to the rows of X.

Value

A matrix with missing values imputed using LOCF and NOCB.

Examples

```
X <- matrix(c(NA, 2, NA, 4, 5, NA, 7, 8, NA, NA), nrow = 5, byrow = TRUE)
subject_id <- c(1, 1, 1, 2, 2)
apply_locf_nocb(X, subject_id)
```

BMTrees_prediction *Bayesian Trees Mixed-Effects Models for Predicting Longitudinal Outcomes*

Description

Provides predictions for outcomes in longitudinal data using Bayesian Trees Mixed-Effects Models (BMTrees) and its semiparametric variants. The function predicts values for test data while accounting for random effects, complex relationships, and potential model misspecification.

Usage

```
BMTrees_prediction(
  X_train,
  Y_train,
  Z_train,
  subject_id_train,
  X_test,
```

```
  Z_test,
  subject_id_test,
  model = c("BMTrees", "BMTrees_R", "BMTrees_RE", "mixedBART"),
  binary = FALSE,
  nburn = 3000L,
  npost = 4000L,
  skip = 1L,
  verbose = TRUE,
  seed = NULL,
  tol = 1e-20,
  resample = 5,
  ntrees = 200,
  pi_CDP = 0.99
)
```

Arguments

X_train	A matrix of covariates in the training set.
Y_train	A numeric or logical vector of outcomes in the training set.
Z_train	A matrix of random predictors in the training set.
subject_id_train	A character vector of subject IDs in the training set.
X_test	A matrix of covariates in the testing set.
Z_test	A matrix of random predictors in the testing set.
subject_id_test	A character vector of subject IDs in the testing set.
model	A character string specifying the predictive model. Options are "BMTrees", "BMTrees_R", "BMTrees_RE", and "mixedBART". Default: "BMTrees".
binary	Logical. Indicates whether the outcome is binary (TRUE) or continuous (FALSE). Default: FALSE.
nburn	An integer specifying the number of burn-in iterations for Gibbs sampler. Default: 3000L.
npost	An integer specifying the number of posterior samples to collect. Default: 4000L.
skip	An integer indicating the thinning interval for MCMC samples. Default: 1L.
verbose	Logical. If TRUE, displays MCMC progress. If FALSE, shows a progress bar. Default: TRUE.
seed	An optional integer for setting the random seed to ensure reproducibility. Default: NULL.
tol	A numeric tolerance value to prevent numerical overflow and underflow in the model. Default: 1e-20.
resample	An integer specifying the number of resampling steps for the CDP prior. Default: 5. This parameter is only valid for "BMTrees" and "BMTrees_R".
ntrees	An integer specifying the number of trees in BART. Default: 200.
pi_CDP	A value between 0 and 1 for calculating the empirical prior in the CDP prior. Default: 0.99.

Value

A list containing posterior samples and predictions:

post_tree_train Posterior samples of the fixed-effects from BART on training data.

post_Sigma Posterior samples of covariance matrices in random effects.

post_lambda_F Posterior samples of lambda parameter in CDP normal mixture on random errors.

post_lambda_G Posterior samples of lambda parameter in CDP normal mixture on random-effects.

post_B Posterior samples of the coefficients in random effects.

post_random_effect_train Posterior samples of random effects for training data.

post_sigma Posterior samples of error deviation.

post_expectation_y_train Posterior expectations of training data outcomes, equal to fixed-effects + random effects.

post_expectation_y_test Posterior expectations of testing data outcomes, equal to fixed-effects + random effects.

post_predictive_y_train Posterior predictive distributions for training outcomes, equal to fixed-effects + random effects + predictive residual.

post_predictive_y_test Posterior predictive distributions for testing outcomes, equal to fixed-effects + random effects + predictive residual.

post_eta Posterior samples of location parameters in CDP normal mixture on random errors.

post_mu Posterior samples of location parameters in CDP normal mixture on random effects.

Note

This function utilizes modified C++ code originally derived from the BART3 package (Bayesian Additive Regression Trees). The original package was developed by Rodney Sparapani and is licensed under GPL-2. Modifications were made by Jungang Zou, 2024.

References

For more information about the original BART3 package, see: https://github.com/rsparapa/bnptools/tree/master/BART3

Examples

```
data = simulation_prediction(n_subject = 100, seed = 1234, nonlinear = TRUE,
nonrandeff = TRUE, nonresidual = TRUE)

# To make it faster to compile and check, we only run 30 iterations for burn-in
# and 40 for posterior sampling phases.
# Please increase to 3000 and 4000 iterations, respectively, when running the model.
model = BMTrees_prediction(data$X_train, data$Y_train, data$Z_train,
data$subject_id_train, data$X_test, data$Z_test, data$subject_id_test, model = "BMTrees",
binary = FALSE, nburn = 30L, npost = 40L, skip = 1L, verbose = TRUE, seed = 1234)
model$post_predictive_y_test
model$post_sigma
```

`sequential_imputation` *Sequential Imputation for Missing Data*

Description

Implements sequential imputation for missing covariates and outcomes in longitudinal data. The function uses a Bayesian non-parametric framework with mixed-effects models to handle both normal and non-normal random effects and errors. It sequentially imputes missing values by constructing univariate models in a fixed order, ensuring simplicity and consistency with a valid joint distribution.

Usage

```
sequential_imputation(
  X,
  Y,
  Z = NULL,
  subject_id,
  type,
  binary_outcome = FALSE,
  model = c("BMTrees", "BMTrees_R", "BMTrees_RE", "mixedBART"),
  nburn = 0L,
  npost = 3L,
  skip = 1L,
  verbose = TRUE,
  seed = NULL,
  tol = 1e-20,
  resample = 5,
  ntrees = 200,
  reordering = TRUE,
  pi_CDP = 0.99
)
```

Arguments

`X`	A matrix of missing covariates.
`Y`	A vector of missing outcomes (numeric or logical).
`Z`	A matrix of complete random predictors.
`subject_id`	A vector of subject IDs corresponding to the rows of `X` and `Y`. Can be both integer or character
`type`	A logical vector indicating whether each covariate in `X` is binary (1) or continuous (0).
`binary_outcome`	A logical value indicating whether the outcome `Y` is binary (1) or continuous (0). Default: `0`.
`model`	A character vector specifying the imputation model. Options are `"BMTrees"`, `"BMTrees_R"`, `"BMTrees_RE"`, and `"mixedBART"`. Default: `"BMTrees"`.

nburn	An integer specifying the number of burn-in iterations. Default: 0.
npost	An integer specifying the number of sampling iterations. Default: 3.
skip	An integer specifying the interval for keeping samples in the sampling phase. Default: 1.
verbose	A logical value indicating whether to display progress and MCMC information. Default: TRUE.
seed	A random seed for reproducibility. Default: NULL.
tol	A small numerical tolerance to prevent numerical overflow or underflow in the model. Default: 1e-20.
resample	An integer specifying the number of resampling steps for the CDP prior. Default: 5. This parameter is only valid for "BMTrees" and "BMTrees_R".
ntrees	An integer specifying the number of trees in BART. Default: 200.
reordering	A logical value indicating whether to apply a reordering strategy for sorting covariates. Default: TRUE.
pi_CDP	A value between 0 and 1 for calculating the empirical prior in the CDP prior. Default: 0.99.

Details

The function builds on the Bayesian Trees Mixed-Effects Model (BMTrees), which extends Mixed-Effects BART by using centralized Dirichlet Process (CDP) Normal Mixture priors. This framework handles non-normal random effects and errors, addresses model misspecification, and captures complex relationships. The function employs a Metropolis-Hastings MCMC method to sequentially impute missing values.

Value

A three-dimensional array of imputed data with dimensions (npost / skip, N, p + 1), where:

- N is the number of observations.
- p is the number of covariates in X. The array includes imputed covariates and outcomes.

Note

This function utilizes modified C++ code originally derived from the BART3 package (Bayesian Additive Regression Trees). The original package was developed by Rodney Sparapani and is licensed under GPL-2. Modifications were made by Jungang Zou, 2024.

References

For more information about the original BART3 package, see: https://github.com/rsparapa/bnptools/tree/master/BART3

Examples

```
data <- simulation_imputation(n_subject = 100, seed = 1234, nonrandeff = TRUE,
        nonresidual = TRUE, alligned = FALSE)

# To make it faster to compile and check, we only run 30 iterations for burn-in
# and 40 for posterior sampling phases.
# Please increase to 3000 and 4000 iterations, respectively, when running the model.
model <- sequential_imputation(data$X_mis, data$Y_mis, data$Z, data$subject_id,
        rep(0, 9), binary_outcome = FALSE, model = "BMTrees", nburn = 30L,
        npost = 40L, skip = 2L, verbose = TRUE, seed = 1234)
model$imputed_data
```

`simulation_imputation` *Simulate Longitudinal Data with Missingness*

Description

Generates a dataset with longitudinal data containing missing covariates and outcomes. The function allows customization of random effects, residuals, and the alignment of covariates to simulate data under different conditions.

Usage

```
simulation_imputation(
  n_subject = 800,
  seed = NULL,
  nonrandeff = FALSE,
  nonresidual = FALSE,
  alligned = FALSE
)
```

Arguments

`n_subject`	Number of subjects in the dataset. Each subject has multiple observations. Default: `800`.
`seed`	Random seed for reproducibility. Default: `123`.
`nonrandeff`	Logical value indicating whether the random effects are non-normal. Default: `FALSE`.
`nonresidual`	Logical value indicating whether the residuals are non-normal. Default: `FALSE`.
`alligned`	Logical value indicating whether the covariates should be aligned (`TRUE`) or shuffled (`FALSE`). Default: `FALSE`. If it is shuffled, we will return covariate order as X1, X2, X3, X4, X5, X6, X9, X8, X7. If it is alligned, we will return covariate order as X1, X2, X3, X4, X5, X6, X7, X8, X9.

Details

This function creates longitudinal data for multiple subjects, each observed across 6 time points. Non-normal or normal random effects and residual conditions can be specified. Missing values are introduced based MAR assumption. The alignment of covariates can be customized to test different imputation scenarios.

Value

A list containing:

`X_mis` Matrix of missing covariates.

`Y_mis` Vector of missing outcomes.

`Z` Matrix of complete random predictors.

`subject_id` Vector of subject IDs.

`time` Time points for each observation.

`X_O` Matrix of original complete covariates (for evaluation).

`Y_O` Vector of original complete outcomes (for evaluation).

See Also

`Normal`, `Uniform`, `Binomial`, `Chisquare`, `GammaDist` `Mvnorm` `dmst` `reexports`, `mutate`, `select` `pivot_wider`, `pivot_longer`, `reexports`, `separate` `invlogit`

Examples

```
simulated_data <- simulation_imputation(
  n_subject = 800,
  seed = 123,
  nonrandeff = TRUE,
  nonresidual = TRUE,
  alligned = FALSE
)
```

`simulation_prediction` *Simulate Longitudinal Data for Prediction*

Description

Generates a fixed population longitudinal dataset, with random seeds to generate different training and testing sets. The function supports customization of linear/nonlinear associations, normal/non-normal random effects, and random errors. It splits the data into training and testing sets, with the testing set comprising approximately 40% of the data.

Usage

```
simulation_prediction(
  n_subject = 800,
  seed = NULL,
  nonlinear = FALSE,
  nonrandeff = FALSE,
  nonresidual = FALSE
)
```

Arguments

n_subject	Number of subjects in the dataset. Each subject has multiple observations across 6 follow-up time points. Default: 800.
seed	Random seed for reproducibility. Ensures different training-testing splits. Default: 123.
nonlinear	Logical value indicating whether the outcome model includes nonlinear associations. Default: FALSE.
nonrandeff	Logical value indicating whether the random effects are non-normal. Default: FALSE.
nonresidual	Logical value indicating whether the residuals are non-normal. Default: FALSE.

Details

The function creates a dataset with individuals observed at 6 follow-up time points. It allows users to specify whether the associations are linear or nonlinear and whether random effects and residuals follow normal or non-normal distributions. Approximately 40% of the data is randomly chosen to form the testing set, while the remaining 60% constitutes the training set.

Value

A list containing:

Y_test_true True values of the vector of outcomes in the testing set.

X_train Matrix of covariates in the training set.

Y_train Vector of outcomes in the training set.

Z_train Matrix of random predictors in the training set.

subject_id_train Vector of subject IDs in the training set.

time_train Vector of time point in the training set.

X_test Matrix of covariates in the testing set.

Y_test Vector of outcomes in the testing set.

Z_test Matrix of random predictors in the testing set.

subject_id_test Vector of subject IDs in the testing set.

time_test Vector of time point in the testing set.

See Also

Mvnorm Chisquare ampute

Examples

```
# Generate data with nonlinear associations and non-normal random effects and residuals
data <- simulation_prediction(
  n_subject = 800,
  seed = 123,
  nonlinear = TRUE,
  nonrandeff = TRUE,
  nonresidual = TRUE
)
# Access training and testing data
X_train <- data$X_train
Y_train <- data$Y_train
Z_train <- data$Z_train
subject_id_train <- data$subject_id_train

X_test <- data$X_test
Y_test <- data$Y_test
Z_test <- data$Z_test
subject_id_test <- data$subject_id_test

Y_test_true = data$Y_test_true
```

Index