Package 'plsmmLasso'

June 7, 2024

Title Variable Selection and Inference for Partial Semiparametric Linear Mixed-Effects Model

Version 1.1.0

Description Implements a partial linear semiparametric mixed-effects model (PLSMM) featuring a random intercept and applies a lasso penalty to both the fixed effects and the coefficients associated with the nonlinear function.
The model also accommodates interactions between the nonlinear function and a grouping variable, allowing for the capture of group-specific nonlinearities. Nonlinear functions are modeled using a set of bases functions. Estimation is conducted using a penalized Expectation-Maximization algorithm, and the package offers flexibility in choosing between various information criteria for model selection.
Post-selection inference is carried out using a debiasing method, while inference on the nonlinear functions employs a bootstrap approach.

License GPL (>= 3)

Imports dplyr, ggplot2, glmnet, hdi, MASS, mvtnorm, rlang, scalreg, stats

Encoding UTF-8

RoxygenNote 7.3.1

URL https://github.com/Sami-Leon/plsmmLasso

BugReports https://github.com/Sami-Leon/plsmmLasso/issues

NeedsCompilation no

Author Sami Leon [aut, cre, cph] (<https://orcid.org/0000-0001-9138-9450>), Tong Tong Wu [ths] (<https://orcid.org/0000-0002-1175-9923>)

Maintainer Sami Leon <samileon@hotmail.fr>

Repository CRAN

Date/Publication 2024-06-04 09:45:47 UTC

Contents

`create_bases` *Generate bases function matrix*

Description

The `create_bases` function generates a matrix of bases functions for modeling time. The bases function matrix include Fourier basis functions and polynomial basis functions.

Usage

```
create_bases(t, keep = NULL)
```

Arguments

`t`	A vector representing the timepoints for which basis functions are generated.
`keep`	(Optional) A vector specifying the indices of the basis functions to retain. If not provided, all generated basis functions are retained.

Details

The function constructs two types of basis functions: Fourier basis functions and polynomial basis functions. Fourier basis functions are constructed based on the maximum timepoint (`max_t`) and the input timepoints `t`. Polynomial basis functions are constructed with degrees ranging from 0.1 to 2, incrementing by 0.02.

Value

A list with the following components:

`bases`	A matrix containing the generated bases function.
`selected_bases`	A vector containing the indices of the selected basis functions after applying `filter_nonzero_bases`. If the keep argument is provided, `filter_nonzero_bases` is applied exclusively to the functions specified in keep.

See Also

`filter_nonzero_bases`

Examples

```
t <- seq(0, 10, by = 0.5)
bases <- create_bases(t)
selected_bases <- create_bases(t)
selected_bases[[1]]
selected_bases[[2]]
```

`debias_plsmm` *Post-selection inference for PLSMM*

Description

This function debias the lasso coefficients estimated from the `plsmm_lasso` function and computes p-values.

Usage

```
debias_plsmm(x, y, series, plsmm_output, a = 1, Z = NULL)
```

Arguments

`x`	A matrix of predictor variables.
`y`	A continuous vector of response variable.
`series`	A variable representing different series or groups in the data modeled as a random intercept.
`plsmm_output`	Output object obtained from the `plsmm_lasso` function.
`a`	A scalar that adjusts the variance of the random intercept ϕ by $a \times \sigma_\phi$, default is 1.
`Z`	(Optional) Pre-computed correction score matrix. If provided, it will be used directly for debiasing.

Details

The original data is decorrelated, and a correction score matrix is computed. The correction scores are a measure of correlation between the predictors in the data. The debiasing process utilizes these scores to compute debiased estimates of the coefficients, along with associated p-values.

Value

A data frame containing debiased coefficients, standard errors, confidence intervals, and p-values.

Examples

```
set.seed(123)
data_sim = simulate_group_inter(N = 50, n_mvnorm = 3, grouped = TRUE,
                                timepoints = 3:5, nonpara_inter = TRUE,
                                sample_from = seq(0,52,13),
                                cos = FALSE, A_vec = c(1, 1.5))
sim = data_sim$sim
x = as.matrix(sim[,-1:-3])
y = sim$y
series = sim$series
t = sim$t
bases = create_bases(t)
lambda <- 0.0046
gamma <- 0.00000001
plsmm_output <- plsmm_lasso(x, y, series, t,
  name_group_var = "group", bases$bases,
  gamma = gamma, lambda = lambda, timexgroup = TRUE,
  criterion = "BIC"
)
debias_plsmm(x, y, series, plsmm_output)
```

`filter_nonzero_bases` *Filter bases functions*

Description

The `filter_nonzero_bases` function filters out bases functions that are essentially zero. Bases functions with a sum of absolute values less than a threshold (10^{-10}) are considered as essentially zero and are filtered out.

Usage

```
filter_nonzero_bases(bases)
```

Arguments

`bases`	A matrix containing the bases functions.

Value

A list with the following components:

`filtered_bases`	A matrix containing the filtered bases functions, removing those that are essentially zero.
`selected_bases`	A vector containing the indices of the selected bases functions after filtering.

Examples

```
bases <- matrix(c(0, 0.1, 0.2, 0, 0, 0.3, 0, 0, 0), nrow = 3)
filtered_bases <- filter_nonzero_bases(bases)
```

`plot_fit` *Visualization of estimated mean trajectories and nonlinear functions from a PLSMM*

Description

This function plots the observed data, the estimated mean trajectories, and the estimated nonlinear functions from the output of `plsmm_lasso`.

Usage

```
plot_fit(
  x,
  y,
  series,
  t,
  name_group_var,
  plsmm_output,
  predicted = FALSE,
  show_obs = FALSE
)
```

Arguments

`x`	A matrix of predictors.
`y`	A continuous vector of response variable.
`series`	A variable representing different series or groups in the data modeled as a random intercept.
`t`	A numeric vector indicating the time points.
`name_group_var`	A character string specifying the name of the grouping variable.
`plsmm_output`	Output object obtained from the `plsmm_lasso` function.
`predicted`	Logical indicating whether to plot predicted values. If `FALSE` only the observed time points are used.
`show_obs`	Logical. If `TRUE` the observed time points are used for the position scale of the x-axis.

Details

If `predicted` is `TRUE` the function uses the model from `plsmm_output` to predict unobserved time points on a continuous grid of time.

Value

Two plots:

- The first plot shows the observed data and the estimated mean trajectories.
- The second plot shows the estimated nonlinear functions.

Examples

```
set.seed(123)
data_sim <- simulate_group_inter(
  N = 50, n_mvnorm = 3, grouped = TRUE,
  timepoints = 3:5, nonpara_inter = TRUE,
  sample_from = seq(0, 52, 13),
  cos = FALSE, A_vec = c(1, 1.5)
)
sim <- data_sim$sim
x <- as.matrix(sim[, -1:-3])
y <- sim$y
series <- sim$series
t <- sim$t
bases <- create_bases(t)
lambda <- 0.0046
gamma <- 0.00000001
plsmm_output <- plsmm_lasso(x, y, series, t,
  name_group_var = "group", bases$bases,
  gamma = gamma, lambda = lambda, timexgroup = TRUE,
  criterion = "BIC"
)
plot_fit(x, y, series, t, name_group_var = "group", plsmm_output)
```

---

plsmm_lasso *Fit a high-dimensional PLSMM*

Description

Fits a partial linear semiparametric mixed effects model (PLSMM) via penalized maximum likelihood.

Usage

```
plsmm_lasso(
  x,
  y,
  series,
  t,
  name_group_var = NULL,
  bases,
```

```
  gamma,
  lambda,
  timexgroup,
  criterion,
  nonpara = FALSE,
  cvg_tol = 0.001,
  max_iter = 100,
  verbose = FALSE
)
```

Arguments

x	A matrix of predictor variables.
y	A continuous vector of response variable.
series	A variable representing different series or groups in the data modeled as a random intercept.
t	A numeric vector indicating the timepoints.
name_group_var	A character string specifying the name of the grouping variable in the x matrix.
bases	A matrix of bases functions.
gamma	The regularization parameter for the nonlinear effect of time.
lambda	The regularization parameter for the fixed effects.
timexgroup	Logical indicating whether to use a time-by-group interaction. If TRUE, each group in name_group_var will have its own estimate of the time effect.
criterion	The information criterion to be computed. Options are "BIC", "BICC", or "EBIC".
nonpara	Logical. If TRUE, the criterion is computed using both the coefficients of the fixed-effects and the coefficients of the nonlinear function. If FALSE, only the coefficients of the fixed-effects are used.
cvg_tol	Convergence tolerance for the algorithm.
max_iter	Maximum number of iterations allowed for convergence.
verbose	Logical indicating whether to print convergence details at each iteration. Default is FALSE.

Details

This function fits a PLSMM with a lasso penalty on the fixed effects and the coefficient associated with the bases functions. It uses the Expectation-Maximization (EM) algorithm for estimation. The bases functions represent a nonlinear effect of time.

The model includes a random intercept for each level of the variable specified by series. Additionally, if timexgroup is set to TRUE, the model includes a time-by-group interaction, allowing each group of name_group_var to have its own estimate of the nonlinear function, which can capture group-specific nonlinearities over time. If name_group_var is set to NULL only one nonlinear function for the whole data is being used

The algorithm iteratively updates the estimates until convergence or until the maximum number of iterations is reached.

Value

A list containing the following components:

lasso_output	A list with the fitted values for the fixed effect and nonlinear effect. The estimated coeffcients for the fixed effects and nonlinear effect. The indices of the used bases functions.
se	Estimated standard deviation of the residuals.
su	Estimated standard deviation of the random intercept.
out_phi	Data frame containing the estimated individual random intercept.
ni	Number of timepoitns per observations.
hyperparameters	Data frame with lambda and gamma values.
converged	Logical indicating if the algorithm converged.
crit	Value of the selected information criterion.

Examples

```
set.seed(123)
data_sim <- simulate_group_inter(
  N = 50, n_mvnorm = 3, grouped = TRUE,
  timepoints = 3:5, nonpara_inter = TRUE,
  sample_from = seq(0, 52, 13),
  cos = FALSE, A_vec = c(1, 1.5)
)
sim <- data_sim$sim
x <- as.matrix(sim[, -1:-3])
y <- sim$y
series <- sim$series
t <- sim$t
bases <- create_bases(t)
lambda <- 0.0046
gamma <- 0.00000001
plsmm_output <- plsmm_lasso(x, y, series, t,
  name_group_var = "group", bases$bases,
  gamma = gamma, lambda = lambda, timexgroup = TRUE,
  criterion = "BIC"
)
# fixed effect coefficients
plsmm_output$lasso_output$theta

# fixed effect fitted values
plsmm_output$lasso_output$x_fit

# nonlinear functions coefficients
plsmm_output$lasso_output$alpha

# nonlinear functions fitted values
plsmm_output$lasso_output$out_f
```

```
# standard deviation of residuals
plsmm_output$se

# standard deviation of random intercept
plsmm_output$su

# series specific random intercept
plsmm_output$out_phi
```

`simulate_group_inter` *Simulate PLSMM*

Description

Simulate a partial linear semiparametric mixed-effects model.

Usage

```
simulate_group_inter(
  N = 50,
  n_mvnorm = 100,
  grouped = TRUE,
  timepoints = 3:5,
  nonpara_inter = TRUE,
  sample_from,
  cos = FALSE,
  A_vec = c(1, 1.5)
)
```

Arguments

`N`	Number of subjects.
`n_mvnorm`	Number of covariate generates from a multivariate normal distribution.
`grouped`	Logical indicating whether to include grouping effect.
`timepoints`	Vector specifying timepoints for each subject.
`nonpara_inter`	Logical indicating whether the nonparametric function is specific to each group.
`sample_from`	Vector of time points to sample from.
`cos`	Logical indicating whether to use cosine function for second group.
`A_vec`	Vector of amplitudes for the nonlinear functions for each group.

Value

A list with three components:

`sim`	Simulated data frame.
`phi`	Individual random intercepts.
`f_val`	Values of the nonlinear functions.

Examples

```
simulate_group_inter(
  N = 50, n_mvnorm = 100, grouped = TRUE,
  timepoints = 3:5, nonpara_inter = TRUE,
  sample_from = seq(0, 10, by = 0.1), cos = FALSE, A_vec = c(1, 1.5)
)
```

`test_f` *Bootstrap joint confidence bands and L2-norm based test on nonlinear functions*

Description

This function conducts a test of overall equality of two nonlinear functions and generates confidence bands for the estimated difference of the nonlinear functions using a bootstrap method.

Usage

```
test_f(
  x,
  y,
  series,
  t,
  name_group_var,
  plsmm_output,
  n_boot = 1000,
  predicted = FALSE,
  show_obs = FALSE,
  verbose = TRUE
)
```

Arguments

`x`	A matrix of predictors.
`y`	A continuous vector of response variable.
`series`	A variable representing different series or groups in the data modeled as a random intercept.
`t`	A numeric vector indicating the time points.
`name_group_var`	A character string specifying the name of the grouping variable.
`plsmm_output`	Output object obtained from the `plsmm_lasso` function.
`n_boot`	Numeric specifying the number of bootstrap samples (default is 1000).
`predicted`	Logical indicating whether to plot predicted values. If `FALSE` only the observed time points are used.
`show_obs`	Logical. If `TRUE` the observed time points are used for the position scale of the x-axis.
`verbose`	Logical indicating whether to display bootstrap progress. Default is `TRUE`.

Details

The function generate bootstrap samples and estimate the nonlinear functions for each `n_boot` sample. These bootstrap estimates are then used to compute the L2-norm test of equality and the joint confidence bands.

Value

A plot showing the estimated difference and confidence bands of the nonlinear functions.

A list containing:

`overall_test_results`
: Results from the L2-norm test of equality.

`CI_f`
: Confidence intervals values for the difference of the estimated functions used for plotting.

Examples

```
set.seed(123)
data_sim <- simulate_group_inter(
  N = 50, n_mvnorm = 3, grouped = TRUE,
  timepoints = 3:5, nonpara_inter = TRUE,
  sample_from = seq(0, 52, 13),
  cos = FALSE, A_vec = c(1, 1.5)
)
sim <- data_sim$sim
x <- as.matrix(sim[, -1:-3])
y <- sim$y
series <- sim$series
t <- sim$t
bases <- create_bases(t)
lambda <- 0.0046
gamma <- 0.00000001
plsmm_output <- plsmm_lasso(x, y, series, t,
  name_group_var = "group", bases$bases,
  gamma = gamma, lambda = lambda, timexgroup = TRUE,
  criterion = "BIC"
)
test_f_results <- test_f(x, y, series, t,
 name_group_var = "group", plsmm_output,
 n_boot = 10
)
test_f_results[[1]]
test_f_results[[2]]
```

tune_plsmm *Tune Penalized PLSMM*

Description

This function tunes a penalized partial linear semiparametric mixed-model (PLSMM) by performing a grid search over a set of hyperparameters to find the best model based on a given criterion.

Usage

```
tune_plsmm(
  x,
  y,
  series,
  t,
  name_group_var,
  bases,
  gamma_vec,
  lambda_vec,
  timexgroup,
  criterion,
  ...
)
```

Arguments

x	A matrix of predictors.
y	A continuous vector of response variable.
series	A variable representing different series or groups in the data modeled as a random intercept.
t	A numeric vector indicating the time points.
name_group_var	A character string specifying the name of the grouping variable.
bases	A matrix of bases functions.
gamma_vec	A vector of values for the regularization parameter for the coefficients of the nonlinear functions.
lambda_vec	A vector of values for the regularization parameter for the coefficients of the fixed effects.
timexgroup	Logical indicating whether to use a time-by-group interaction. If TRUE, each group in name_group_var will have its own estimate of the time effect.
criterion	A character string specifying the criterion to be optimized ('BIC', 'BICC', 'EBIC').
...	Additional arguments to be passed to the plsmm_lasso function.

Details

This function performs a grid search over the hyperparameters specified by `lambda_vec` and `gamma_vec` to find the best-fitted PLSMM based on the given criterion. It fits PLSMMs using the `plsmm_lasso` function for each combination of hyperparameters and retains only the models that have converged. The best model is selected based on the minimum value of the specified criterion.

Value

A PLSMM object representing the best-tuned model based on the specified criterion.

See Also

`plsmm_lasso`

Examples

```
set.seed(123)
data_sim <- simulate_group_inter(
 N = 50, n_mvnorm = 3, grouped = TRUE,
 timepoints = 3:5, nonpara_inter = TRUE,
 sample_from = seq(0, 52, 13),
 cos = FALSE, A_vec = c(1, 1.5)
)
sim = data_sim$sim
x = as.matrix(sim[,-1:-3])
y = sim$y
series = sim$series
t = sim$t
bases = create_bases(t)
lambdas <- c(0.0046, 0.0001)
gammas <- 0.00000001
tuned_plsmm <- tune_plsmm(x, y, series, t,
                      name_group_var = "group", bases$bases,
                      gamma_vec = gammas, lambda_vec = lambdas, timexgroup = TRUE,
                      criterion = "BIC"
)
```

Index