Package 'BEND'

March 31, 2026

Title Bayesian Estimation of Nonlinear Data (BEND)

Version 2.0.0

Description Provides a set of models to estimate nonlinear longitudinal data using Bayesian estimation methods. These models include the: 1) Bayesian Piecewise Random Effects Model (Bayes_PREM()) which estimates a piecewise random effects (mixture) model for a given number of latent classes and a latent number of possible changepoints in each class, and can incorporate class and outcome predictive covariates (see Lamm (2022) <https://hdl.handle.net/11299/252533> and Lock et al., (2018) <doi:10.1007/s11336-017-9594-5>), 2) Bayesian Crossed Random Effects Model (Bayes_CREM()) which estimates a linear, quadratic, exponential, or piecewise crossed random effects models where individuals are changing groups over time (e.g., students and schools; see Rohloff et al., (2024) <doi:10.1111/bmsp.12334>), and 3) Bayesian Bivariate Piecewise Random Effects Model (Bayes_BPREM()) which estimates a bivariate piecewise random effects model to jointly model two related outcomes (e.g., reading and math achievement; see Peralta et al., (2022) <doi:10.1037/met0000358>).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/crohlo/BEND

BugReports https://github.com/crohlo/BEND/issues

Depends R (>= 3.6.3)

Imports coda (>= 0.19.4.1), label.switching (>= 1.8), rjags (>= 4.17)

LazyData true

NeedsCompilation no

Author Corissa T. Rohloff [aut, cre] (ORCID: <https://orcid.org/0000-0003-3228-4653>), Rik Lamm [aut] (ORCID: <https://orcid.org/0000-0002-3317-6243>), Yadira Peralta [aut] (ORCID: <https://orcid.org/0000-0003-4823-6939>), Nidhi Kohli [aut] (ORCID: <https://orcid.org/0000-0003-4690-2854>), Eric F. Lock [aut] (ORCID: <https://orcid.org/0000-0003-4663-2356>)

Maintainer Corissa T. Rohloff <corissa.wurth@gmail.com>

Repository CRAN

Date/Publication 2026-03-31 03:30:02 UTC

Contents

`Bayes_BPREM` *Bayesian Bivariate Piecewise Random Effects Model (BPREM)*

Description

Estimates a Bayesian bivariate piecewise random effects models (BPREM) for longitudinal data with two interrelated outcomes. See Peralta et al. (2022) for more details.

Usage

```
Bayes_BPREM(
  data,
  id_var,
  time_var,
  y1_var,
  y2_var,
  iters_adapt = 5000,
  iters_burn_in = 1e+05,
  iters_sampling = 50000,
  thin = 15,
  save_full_chains = FALSE,
  save_conv_chains = FALSE,
  verbose = TRUE
)
```

Arguments

`data`	Data frame in long format, where each row describes a measurement occasion for a given individual. It is assumed that each individual has the same number of assigned timepoints (a.k.a., rows). There can be missingness in the outcomes (`y1_var` and `y2_var`), but there cannot be missingness in time (`time_var`).
`id_var`	Name of column that contains ids for individuals with repeated measures in a longitudinal dataset.
`time_var`	Name of column that contains the time variable. This column cannot contain any missing values.
`y1_var`	Name of column that contains the first outcome variable. Missing values should be denoted by NA.
`y2_var`	Name of column that contains the second outcome variable. Missing values should be denoted by NA.
`iters_adapt`	(optional) Number of iterations for adaptation of jags model (default = 5000).
`iters_burn_in`	(optional) Number of iterations for burn-in (default = 100000).
`iters_sampling`	(optional) Number of iterations for posterior sampling (default = 50000).
`thin`	(optional) Thinning interval for posterior sampling (default = 15).
`save_full_chains`	Logical indicating whether the MCMC chains from rjags should be saved (default = FALSE). Note, this should not be used regularly as it will result in an object with a large file size.
`save_conv_chains`	Logical indicating whether the MCMC chains from rjags should be saved but only for the parameters monitored for convergence (default = FALSE). This would be useful for plotting traceplots for relevant model parameters to evaluate convergence behavior. Note, this should not be used regularly as it will result in an object with a large file size.
`verbose`	Logical controlling whether progress messages/bars are generated (default = TRUE).

Details

For more information on the model equation and priors implemented in this function, see Peralta et al. (2022).

Value

A list (an object of class BPREM) with elements:

Call	A list of all specified function arguments.
Sample Size	Number of subjects (n_subj) and number of timepoints (n_time) in the dataset.
Data	Dataset used for estimation.
Convergence	Potential scale reduction factor (PSRF) for each parameter (parameter_psrf), Gelman multivariate scale reduction factor (multivariate_psrf), and mean PSRF (mean_psrf) to assess model convergence.
Model_Fit	Deviance (deviance), effective number of parameters (pD), and Deviance information criterion (dic) to assess model fit.
Fitted_Values	Data frame with the fitted values for y1 and y2 at each timepoint for each individual.
Parameter_Estimates	Data frame with posterior mean and 95% credible intervals for each model parameter.
Random_Coefficients	List object with data frames providing random effects and random coefficients for each individual.
Run_Time	Total run time for model fitting.
Full_MCMC_Chains	If save_full_chains=TRUE, raw MCMC chains from rjags.
Convergence_MCMC_Chains	If save_conv_chains=TRUE, raw MCMC chains from rjags but only for the parameters monitored for convergence.

Author(s)

Corissa T. Rohloff, Yadira Peralta

References

Peralta, Y., Kohli, N., Lock, E. F., & Davison, M. L. (2022). Bayesian modeling of associations in bivariate piecewise linear mixed-effects models. Psychological Methods, 27(1), 44–64. https://doi.org/10.1037/met0000358

Examples

```
# load simulated data
data(SimData_BPREM)
# fit Bayes_BPREM()
results_bprem <- Bayes_BPREM(data = SimData_BPREM,
```

```
                            id_var = "id",
                            time_var = "time",
                            y1_var = "y1",
                            y2_var = "y2")
# result summary
summary(results_bprem)
# plot fitted results
plot(results_bprem)
```

`Bayes_CREM` *Bayesian Crossed Random Effects Model (CREM)*

Description

Estimates a Bayesian crossed random effects models (CREM) for longitudinal data with dynamic group membership. Four different choices for functional forms are provided: linear, quadratic, exponential, and piecewise. See Rohloff et al. (2024) for more details.

Usage

```
Bayes_CREM(
  data,
  ind_id_var,
  cross_id_var,
  time_var,
  y_var,
  form = "linear",
  fixed_effects = NULL,
  iters_adapt = 5000,
  iters_burn_in = 50000,
  iters_sampling = 50000,
  thin = 15,
  save_full_chains = FALSE,
  save_conv_chains = FALSE,
  verbose = TRUE
)
```

Arguments

`data`	Data frame in long format, where each row describes a measurement occasion for a given individual. It is assumed that each individual has the same number of assigned timepoints (a.k.a., rows). There can be missingness in the outcome (`y_var`), but there cannot be missingness in time (`time_var`).
`ind_id_var`	Name of column that contains ids for individuals with repeated measures in a longitudinal dataset (e.g., students).

cross_id_var	Name of column that contains ids for the crossed factor (e.g., teachers).
time_var	Name of column that contains the time variable. This column cannot contain any missing values.
y_var	Name of column that contains the outcome variable. Missing values should be denoted by NA.
form	Name of the functional form. Options include: 'linear' (default), 'quadratic', 'exponential', 'piecewise'.
fixed_effects	(optional) Starting values for the fixed effects parameters.
iters_adapt	(optional) Number of iterations for adaptation of jags model (default = 5000).
iters_burn_in	(optional) Number of iterations for burn-in (default = 50000).
iters_sampling	(optional) Number of iterations for posterior sampling (default = 50000).
thin	(optional) Thinning interval for posterior sampling (default = 15).
save_full_chains	Logical indicating whether the MCMC chains from rjags should be saved (default = FALSE). Note, this should not be used regularly as it will result in an object with a large file size.
save_conv_chains	Logical indicating whether the MCMC chains from rjags should be saved but only for the parameters monitored for convergence (default = FALSE). This would be useful for plotting traceplots for relevant model parameters to evaluate convergence behavior. Note, this should not be used regularly as it will result in an object with a large file size.
verbose	Logical controlling whether progress messages/bars are generated (default = TRUE).

Details

For more information on the model equation and priors implemented in this function, see Rohloff et al. (2024).

Note, this function differs from the above reference by estimating the covariances between the random effects parameters. The variance-covariance matrices of the individual and group random effects have a scaled inverse-Wishart prior (see Peralta et al., 2022).

Value

A list (an object of class CREM) with elements:

Call	A list of all specified function arguments.
Sample Size	Number of subjects (n_subj), number of groups (n_group) and number of time-points (n_time) in the dataset.
Data	Dataset used for estimation.
Convergence	Potential scale reduction factor (PSRF) for each parameter (parameter_psrf), Gelman multivariate scale reduction factor (multivariate_psrf), and mean PSRF (mean_psrf) to assess model convergence.

`Model_Fit` Deviance (`deviance`), effective number of parameters (`pD`), and Deviance information criterion (`dic`) to assess model fit.

`Fitted_Values` Data frame with the fitted values at each timepoint for each individual.

`Functional_Form`
Functional form fitted.

`Parameter_Estimates`
Data frame with posterior mean and 95% credible intervals for each model parameter.

`Random_Coefficients`
List object with data frames providing individual random effects, group random effects, and random coefficients for each individual and group.

`Run_Time` Total run time for model fitting.

`Full_MCMC_Chains`
If save_full_chains=TRUE, raw MCMC chains from rjags.

`Convergence_MCMC_Chains`
If save_conv_chains=TRUE, raw MCMC chains from rjags but only for the parameters monitored for convergence.

Author(s)

Corissa T. Rohloff

References

Peralta, Y., Kohli, N., Lock, E. F., & Davison, M. L. (2022). Bayesian modeling of associations in bivariate piecewise linear mixed-effects models. Psychological Methods, 27(1), 44–64. https://doi.org/10.1037/met0000358

Rohloff, C. T., Kohli, N., & Lock, E. F. (2024). Identifiability and estimability of Bayesian linear and nonlinear crossed random effects models. British Journal of Mathematical and Statistical Psychology. https://doi.org/10.1111/bmsp.12334

Examples

```
# load simulated data
data(SimData_PCREM)
# fit Bayes_CREM()
results_pcrem <- Bayes_CREM(data = SimData_PCREM,
                            ind_id_var = "id",
                            cross_id_var = "teacherid",
                            time_var = "time",
                            y_var = "y",
                            form="piecewise")
# result summary
summary(results_pcrem)
# plot fitted results
plot(results_pcrem)
```

Bayes_PREM *Bayesian Piecewise Random Effects Model (PREM) + Extensions*

Description

Estimates a Bayesian piecewise random effects model (PREM), with some useful extensions. There are three model options included in this function:

- `PREM` estimates a Bayesian piecewise random effects model with a latent number of changepoints (default). Allows the inclusion of outcome-predictive covariates (`CI-PREM`).
- `PREMM` estimates a piecewise random effects mixture model for a given number of latent classes and a latent number of possible changepoints in each class.
- `CI-PREMM` estimates a covariate influenced piecewise random effects mixture model for a given number of latent classes and a latent number of possible changepoints in each class. Allows the inclusion of outcome- and/or class-predictive covariates. See Lock et al. (2018) and Lamm (2022) for more details.

Usage

```
Bayes_PREM(
  data,
  id_var,
  time_var,
  y_var,
  n_class = 1,
  max_cp = 2,
  class_predictive_vars = NULL,
  outcome_predictive_vars = NULL,
  scale_prior = "uniform",
  alpha = 1,
  cp_prior = "binomial",
  binom_prob = 0.5,
  iters_adapt = 1000,
  iters_burn_in = 20000,
  iters_sampling = 30000,
  thin = 15,
  save_full_chains = FALSE,
  save_conv_chains = FALSE,
  verbose = TRUE
)
```

Arguments

`data`	Data frame in long format, where each row describes a measurement occasion for a given individual. It is assumed that each individual has the same number of assigned timepoints (a.k.a., rows). There can be missingness in the outcome (`y_var`), but there cannot be missingness in time (`time_var`).

`id_var` Name of column that contains ids for individuals with repeated measures in a longitudinal dataset.

`time_var` Name of column that contains the time variable. This column cannot contain any missing values.

`y_var` Name of column that contains the outcome variable. Missing values should be denoted by NA.

`n_class` Number of latent classes (default = 1). Note, CI-PREMM only allows for two classes.

`max_cp` Maximum number of changepoints in each latent class (default = 2).

`class_predictive_vars`
Name(s) of column(s) that contain class-predictive covariates (time-invariant only). Give a vector of names if multiple covariates. Note, there cannot be any missingness in the covariates.

`outcome_predictive_vars`
Name(s) of column(s) that contain outcome-predictive covariates (time-varying or -invariant). Give a vector of names if multiple covariates. Note, there cannot be any missingness in the covariates.

`scale_prior` Prior for the scale parameter for the hierarchical random effects. Options include: 'uniform' (scaled uniform prior; default) or 'hc' (scaled half-cauchy prior).

`alpha` Concentration parameter for Dirichlet prior for latent classes (default = 1). This can be a vector of values corresponding to the number of classes (specified by n_class). Note, this is not used for CI-PGMM.

`cp_prior` Prior for the number of changepoints in each class. Options include: 'binomial' (default) or 'uniform'.

`binom_prob` Probability for binomial prior, if specified (default = 0.5).

`iters_adapt` (optional) Number of iterations for adaptation of jags model (default = 1000).

`iters_burn_in` (optional) Number of iterations for burn-in (default = 20000).

`iters_sampling` (optional) Number of iterations for posterior sampling (default = 30000).

`thin` (optional) Thinning interval for posterior sampling (default = 15).

`save_full_chains`
Logical indicating whether the MCMC chains from rjags should be saved (default = FALSE). Note, this should not be used regularly as it will result in an object with a large file size.

`save_conv_chains`
Logical indicating whether the MCMC chains from rjags should be saved but only for the parameters monitored for convergence (default = FALSE). This would be useful for plotting traceplots for relevant model parameters to evaluate convergence behavior. Note, this should not be used regularly as it will result in an object with a large file size.

`verbose` Logical controlling whether progress messages/bars are generated (default = TRUE).

Details

For more information on the model equation and priors implemented in this function, see Lamm et al. (2022; CI-PREMM) and Lock et al. (2018; PREMM).

Value

A list (an object of class `PREM`) with elements:

`Call` A list of all specified function arguments.

`Sample Size` Number of subjects (`n_subj`) and number of timepoints (`n_time`) in the dataset.

`Data` Dataset used for estimation.

`Convergence` Potential scale reduction factor (PSRF) for each parameter (`parameter_psrf`), Gelman multivariate scale reduction factor (`multivariate_psrf`), and mean PSRF (`mean_psrf`) to assess model convergence.

`Model_Fit` Deviance (`deviance`), effective number of parameters (`pD`), and Deviance information criterion (`dic`) to assess model fit.

`Fitted_Values` Data frame with the fitted values at each timepoint for each individual.

`Parameter_Estimates`
Data frame with posterior mean and 95% credible intervals for each model parameter.

`Random_Coefficients`
List object with data frames providing random coefficients for each individual.

`Run_Time` Total run time for model fitting.

`Full_MCMC_Chains`
If save_full_chains=TRUE, raw MCMC chains from rjags.

`Convergence_MCMC_Chains`
If save_conv_chains=TRUE, raw MCMC chains from rjags but only for the parameters monitored for convergence.

`Class_Information` contains a list with elements:

`class_membership`
Vector of length n with class membership assignments for each individual.

`individ_class_probability`
nxC matrix with each individual's probabilities of belonging to each class conditional on their class-predictive covariates (when applicable) and growth curve.

`unconditional_class_probability`
This output will differ based on which model was fit. For a PREM or CI-PREM, this will equal 1 as there is only one class. For a PREMM or CI-PREMM with only outcome-predictive covariates, this will be a vector of length C denoting the population probability of belonging to each class. For a CI-PREMM with class-predictive covariates, this will be a vector of length n denoting the probability of each individual belonging to the non-reference class (Class 2) based on their class-predictive covariates only.

Author(s)

Corissa T. Rohloff, Rik Lamm, Eric F. Lock

References

Lamm, R. (2022). Incorporation of covariates in Bayesian piecewise growth mixture models. https://hdl.handle.net/11299/252533

Lock, E. F., Kohli, N., & Bose, M. (2018). Detecting multiple random changepoints in Bayesian piecewise growth mixture models. Psychometrika, 83(3), 733–750. https://doi.org/10.1007/s11336-017-9594-5

Examples

```
# load simulated data
data(SimData_PREM)

# PREM ------------------------------------------------------------------------------
# fit Bayes_PREM()
results_prem <- Bayes_PREM(data = SimData_PREM,
                           id_var = "id",
                           time_var = "time",
                           y_var = "y")
# result summary
summary(results_prem)
# plot fitted results
plot(results_prem)

# CI-PREM ---------------------------------------------------------------------------
# fit Bayes_PREM()
results_ciprem <- Bayes_PREM(data = SimData_PREM,
                             id_var = "id",
                             time_var = "time",
                             y_var = "y",
                             outcome_predictive_vars = "outcome_pred_1")
# result summary
summary(results_ciprem)
# plot fitted results
plot(results_ciprem)

# PREMM -----------------------------------------------------------------------------
# fit Bayes_PREM()
results_premm <- Bayes_PREM(data = SimData_PREM,
                            id_var = "id",
                            time_var = "time",
                            y_var = "y",
                            n_class = 2)
# result summary
summary(results_premm)
# plot fitted results
plot(results_premm)


# CI-PREMM --------------------------------------------------------------------------
# fit Bayes_PREM()
results_cipremm <- Bayes_PREM(data = SimData_PREM,
```

```
                                id_var = "id",
                                time_var = "time",
                                y_var = "y",
                                n_class = 2,
                                class_predictive_vars = c("class_pred_1", "class_pred_2"),
                                outcome_predictive_vars = "outcome_pred_1")
# result summary
summary(results_cipremm)
# plot fitted results
plot(results_cipremm)
```

`getClassProb` *Extract class probabilities*

Description

Extracts the posterior class probabilities for each individual from a fitted model of class "PREM".

Usage

```
getClassProb(x, ...)

## S3 method for class 'PREM'
getClassProb(x, ...)

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

Arguments

`x`	An object of class "PREM".
`...`	Additional arguments.

Value

Returns a list of the posterior class probabilities for each individual.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# get class probabilities
getClassProb(results_prem)
```

getCoef *Extract random coefficients*

Description

Extracts the random coefficients from a fitted model of class "BPREM", "CREM", or "PREM".

Usage

```
getCoef(x, ...)

## S3 method for class 'BPREM'
getCoef(x, ...)

## S3 method for class 'CREM'
getCoef(x, ...)

## S3 method for class 'PREM'
getCoef(x, ...)

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

Arguments

x	An object of class "BPREM", "CREM", or "PREM".
...	Additional arguments.

Value

Returns a data frame of the random coefficients for each individual/group.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# get random coefficients
getCoef(results_prem)
```

getFitted *Extract fitted values*

Description

Extracts the individual fitted values from a fitted model of class "BPREM", "CREM", or "PREM".

Usage

```
getFitted(x, ...)

## S3 method for class 'BPREM'
getFitted(x, ...)

## S3 method for class 'CREM'
getFitted(x, ...)

## S3 method for class 'PREM'
getFitted(x, ...)

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

Arguments

x	An object of class "BPREM", "CREM", or "PREM".
...	Additional arguments.

Value

Returns a data frame of the fitted values for each individual and timepoint.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# get fitted values
getFitted(results_prem)
```

getFixEf *Extract fixed effects parameter estimates*

Description

Extracts the fixed effects parameter estimates from a fitted model of class "BPREM", "CREM", or "PREM".

Usage

```
getFixEf(x, ...)

## S3 method for class 'BPREM'
getFixEf(x, ...)

## S3 method for class 'CREM'
getFixEf(x, ...)

## S3 method for class 'PREM'
getFixEf(x, ...)

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

Arguments

x	An object of class "BPREM", "CREM", or "PREM".
...	Additional arguments.

Value

Returns a vector or data frame of fixed effects parameter values.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_bprem)
# get fixed effects
getFixEf(results_bprem)
```

getKProb *Extract changepoint probabilities*

Description

Extracts the K (number of changepoints) probabilities from a fitted model of class "PREM".

Usage

```
getKProb(x, ...)

## S3 method for class 'PREM'
getKProb(x, ...)

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

Arguments

x	An object of class "PREM".
...	Additional arguments.

Value

Returns a list of the posterior probabilities for each possible number of changepoints, for each class.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# get changepoint probabilities
getKProb(results_prem)
```

getModelFit *Extract model fit*

Description

Extracts the model fit information from a fitted model of class "BPREM", "CREM", or "PREM".

Usage

```
getModelFit(x, ...)

## S3 method for class 'BPREM'
getModelFit(x, ...)

## S3 method for class 'CREM'
getModelFit(x, ...)

## S3 method for class 'PREM'
getModelFit(x, ...)

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

Arguments

x	An object of class "BPREM", "CREM", or "PREM".
...	Additional arguments.

Value

Returns a vector of the model fit information (deviance, pD, DIC).

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# get model fit
getModelFit(results_prem)
```

getRanEf *Extract random effects*

Description

Extracts the random effects from a fitted model of class "BPREM" or "CREM".

Usage

```
getRanEf(x, ...)

## S3 method for class 'BPREM'
getRanEf(x, ...)

## S3 method for class 'CREM'
getRanEf(x, ...)

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

Arguments

x	An object of class "BPREM" or "CREM".
...	Additional arguments.

Value

Returns a list of the random effects for each individual/group.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_bprem)
# get random effects
getRanEf(results_bprem)
```

getVarCov *Extract random effects variance-covariance matrix*

Description

Extracts the random effects variance-covariance matrix from a fitted model of class "BPREM", "CREM", or "PREM".

Usage

```
getVarCov(x, ...)

## S3 method for class 'BPREM'
getVarCov(x, ...)

## S3 method for class 'CREM'
getVarCov(x, ...)

## S3 method for class 'PREM'
getVarCov(x, ...)

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

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

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

Arguments

x	An object of class "BPREM", "CREM", or "PREM".
...	Additional arguments.

Value

Returns a list of the random effects variance-covariance matrices.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# get random effects variance-covariance matrices
getVarCov(results_prem)
```

`plot.BPREM` *Plot the results of a bivariate piecewise random effects model (BPREM)*

Description

Provides a fitted plot of a BPREM model, as returned by `Bayes_BPREM()`.

Usage

```
## S3 method for class 'BPREM'
plot(
  x,
  xlab = "X",
  ylab = "Y",
  colors = NULL,
  mean_colors = NULL,
  legend_pos = "topright",
  ...
)
```

Arguments

`x`	An object of class "BPREM" (returned by `Bayes_BPREM(...)`).
`xlab`	X-axis label for the generated plot.
`ylab`	Y-axis label for the generated plot.
`colors`	Colors for each class outcome. By default, up to 2 colors are provided in the following order: "blue" (outcome 1), "red" (outcome 2).
`mean_colors`	Colors for the trajectory defined by the mean parameters for each outcome. By default, up to 2 colors are provided in the following order: "darkblue" (outcome 1), "darkred" (outcome 2).
`legend_pos`	(optional) Option to change legend position (default = "topright").
`...`	(optional) Other parameters to pass to the `plot()` function.

Value

No return value.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_bprem)
# plot fitted results
plot(results_bprem)
```

`plot.CREM` *Plot the results of a crossed random effects model (CREM)*

Description

Provides a fitted plot of a CREM model, as returned by `Bayes_CREM()`.

Usage

```
## S3 method for class 'CREM'
plot(
  x,
  xlab = "X",
  ylab = "Y",
  colors = NULL,
  mean_colors = NULL,
  legend_pos = "topright",
  ...
)
```

Arguments

`x`	An object of class "CREM" (returned by `Bayes_CREM(...)`).
`xlab`	X-axis label for the generated plot.
`ylab`	Y-axis label for the generated plot.
`colors`	Color for observed trajectories (optional). Default is "grey".
`mean_colors`	Colors for the trajectory defined by the mean parameters for each outcome (optional). Default is "black".
`legend_pos`	(optional) Option to change legend position (default = "topright").
`...`	(optional) Other parameters to pass to the `plot()` function.

Value

No return value.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_pcrem)
# plot fitted results
plot(results_pcrem)
```

`plot.PREM` *Plot the results of a piecewise random effects model (PREM)*

Description

Provides a fitted plot of a PREM model, as returned by `Bayes_PREM()`.

Usage

```
## S3 method for class 'PREM'
plot(
  x,
  xlab = "X",
  ylab = "Y",
  colors = NULL,
  mean_colors = NULL,
  legend_pos = "topright",
  ...
)
```

Arguments

`x`	An object of class "PREM" (returned by `Bayes_PREM(...)`).
`xlab`	X-axis label for the generated plot.
`ylab`	Y-axis label for the generated plot.
`colors`	Colors for each class (`PREMM` or `CI-PREMM`). By default, up to 5 colors are provided in the following order: "blue" (class 1), "red" (class 2), "green" (class 3), "gold" (class 4), "gray" (class 5).
`mean_colors`	Colors for the trajectory defined by the mean parameters for each class (`PREMM` or `CI-PREMM`). By default, up to 5 colors are provided in the following order: "darkblue" (class 1), "darkred" (class 2), "darkgreen" (class 3), "gold4" (class 4), "darkgray" (class 5).
`legend_pos`	(optional) Option to change legend position (default = "topright").
`...`	(optional) Other parameters to pass to the `plot()` function.

Value

No return value.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# plot fitted results
plot(results_prem)
```

`plot_BEND` *Plot a BEND Model (PREM, CREM, BPREM)*

Description

Generates a "spaghetti plot" of observed longitudinal trajectories for each individual. If the results from a `BEND` function are supplied, the trajectory defined by the mean parameters is shown in bold. If fitting a mixture (`PREMM` or `CI-PREMM`) or bivariate model (`BPREM`), the mean trajectories for classes or outcomes will be distinguished by color.

Usage

```
plot_BEND(
  data,
  id_var,
  time_var,
  y_var,
  y2_var = NULL,
  results = NULL,
  xlab = "X",
  ylab = "Y",
  colors = NULL,
  mean_colors = NULL,
  legend_pos = "topright",
  ...
)
```

Arguments

`data`	Data frame in long format, where each row describes a measurement occasion for a given individual. It is assumed that each individual has the same number of assigned timepoints (a.k.a., rows).

id_var	Name of column that contains ids for individuals with repeated measures in a longitudinal dataset.
time_var	Name of column that contains the time variable.
y_var	Name of column that contains the outcome variable.
y2_var	(for BPREM only) Name of column that contains the second outcome variable.
results	The output of BEND model to the data. If results=NULL, only a spaghetti plot of the data will be generated.
xlab	X-axis label for the generated plot.
ylab	Y-axis label for the generated plot.
colors	Colors for each class (PREMM or CI-PREMM) or outcome (BPREM). By default, up to 5 colors are provided in the following order: "blue" (class 1 and outcome 1), "red" (class 2 and outcome 2), "green" (class 3), "gold" (class 4), "gray" (class 5).
mean_colors	Colors for the trajectory defined by the mean parameters for each class (PREMM or CI-PREMM) or outcome (BPREM). By default, up to 5 colors are provided in the following order: "darkblue" (class 1 and outcome 1), "darkred" (class 2 and outcome 2), "darkgreen" (class 3), "gold4" (class 4), "darkgray" (class 5).
legend_pos	(optional) Option to change legend position (default = "topright").
...	(optional) Other parameters to pass to the plot() function.

Value

No return value, called to generate plot.

Author(s)

Corissa T. Rohloff

Examples

```
# load simulated data
data(SimData_PREM)
# plot observed data
plot_BEND(data = SimData_PREM,
          id_var = "id",
          time_var = "time",
          y_var = "y")
# load fitted model results
data(results_prem)
# plot fitted results
plot_BEND(data = SimData_PREM,
          id_var = "id",
          time_var = "time",
          y_var = "y",
          results = results_prem)
```

print.BPREM — *Print the results of a bivariate piecewise random effects model (BPREM)*

Description

Provides a summary of a BPREM model, as returned by `Bayes_BPREM()`.

Usage

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

Arguments

`x`	An object of class "BPREM" (returned by `Bayes_BPREM(...)`).
`...`	Additional arguments.

Value

Returns a list of key parameter estimates.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_bprem)
# print results
print(results_bprem)
```

print.CREM — *Print the results of a crossed random effects model (CREM)*

Description

Provides a summary of a CREM model, as returned by `Bayes_CREM()`.

Usage

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

Arguments

x	An object of class "CREM" (returned by `Bayes_CREM(...)`).
...	Additional arguments.

Value

Returns a list of key parameter estimates.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_pcrem)
# print results
print(results_pcrem)
```

---

print.PREM *Print the results of a piecewise random effects model (PREM)*

Description

Provides a summary of a PREM model, as returned by `Bayes_PREM()`.

Usage

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

Arguments

x	An object of class "PREM" (returned by `Bayes_PREM(...)`).
...	Additional arguments.

Value

Returns a list of key parameter estimates.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# print results
print(results_prem)
```

`results_bprem` *Fitted results for a BPREM*

Description

Simulated data for a bivariate piecewise random effects model (BPREM) using `SimData_BPREM`. Included to demonstrate the use of `summary.BPREM()`.

Results from fitting a bivariate piecewise random effects model (BPREM) to `SimData_BPREM` (returned by `Bayes_BPREM(...)`).

Usage

```
data(results_bprem)

results_bprem
```

Format

A list (an object of class `BPREM`) with fitted model results.

`results_bprem`**:**

A list object with elements:

Call Function arguments

Sample_Size Data set sample size

Data Data used for model fitting

Convergence Convergence status

Model_Fit Model fit information

Fitted_Values Fitted outcome values

Parameter_Estimates Parameter estimates

Random_Coefficients Random coefficient estimates

Run_Time Model run time

results_pcrem *Fitted results for a PCREM*

Description

Simulated data for a piecewise crossed random effects model (PCREM) using `SimData_CREM`. Included to demonstrate the use of `summary.CREM()`.

Results from fitting a piecewise crossed random effects model (PCREM) to `SimData_PCREM` (returned by `Bayes_CREM(...)`).

Usage

```
data(results_pcrem)

results_pcrem
```

Format

A list (an object of class `CREM`) with fitted model results.

`results_pcrem`**:**
A list object with elements:

Call Function arguments
Sample_Size Data set sample size
Data Data used for model fitting
Convergence Convergence status
Model_Fit Model fit information
Fitted_Values Fitted outcome values
Parameter_Estimates Parameter estimates
Random_Coefficients Random coefficient estimates
Run_Time Model run time

results_prem *Fitted results for a PREM*

Description

Fitted results for a piecewise random effects model (PREM) using `SimData_PREM`. Included to demonstrate the use of `plot_BEND()` and `summary.PREM()`.

Results from fitting a piecewise random effects model (PREM) to `SimData_PREM` (returned by `Bayes_PREM(...)`).

Usage

```
data(results_prem)

results_prem
```

Format

A list (an object of class PREM) with fitted model results.

results_prem:
A list object with elements:

Call Function arguments
Sample_Size Data set sample size
Data Data used for model fitting
Convergence Convergence status
Model_Fit Model fit information
Fitted_Values Fitted outcome values
Parameter_Estimates Parameter estimates
Random_Coefficients Random coefficient estimates
Run_Time Model run time

SimData_BPREM *Simulated data for a BPREM*

Description

Simulated data for a bivariate piecewise random effects model (BPREM) with 7 timepoints collected on 30 individuals.

A simulated dataset for the bivariate piecewise random effects model (BPREM).

Usage

```
data(SimData_BPREM)

SimData_BPREM
```

Format

A data frame with 210 rows and 4 variables.

SimData_BPREM:
A data frame with 210 rows and 4 columns:

id Individual ids
time Time
y1 Outcome 1
y2 Outcome 2

Details

- `id` ID for each individual.
- `time` Timepoints for each individual.
- `y1` Outcome 1.
- `y2` Outcome 2.

`SimData_PCREM` *Simulated data for a PCREM*

Description

Simulated data for a piecewise crossed random effects model (PCREM) with 7 timepoints collected on 30 individuals.

A simulated dataset for the piecewise crossed random effects model (PCREM).

Usage

```
data(SimData_PCREM)

SimData_PCREM
```

Format

A data frame with 210 rows and 4 variables.

`SimData_PCREM`**:**
A data frame with 210 rows and 4 columns:

id Individual ids

teacherid Teacher ids

time Time

y Outcome

Details

- `id` ID for each individual.
- `teacherid` ID for each teacher.
- `time` Timepoints for each individual.
- `y` Outcome.

SimData_PREM *Simulated data for a PREM + Extensions*

Description

Simulated data for a piecewise random effects model (PREM) and useful extensions (CI-PREM, PREMM, CI-PREMM) with 18 timepoints collected on 30 individuals.

A simulated dataset for the piecewise random effects model (PREM).

Usage

```
data(SimData_PREM)

SimData_PREM
```

Format

A data frame with 540 rows and 6 variables.

SimData_PREM**:**

A data frame with 540 rows and 6 columns:

id Individual ids

time Time

y Outcome

class_pred_1 First Class-Predictive Covariate

class_pred_2 Second Class-Predictive Covariate

outcome_pred_1 First Outcome-Predictive Covariate

Details

- id ID for each individual.
- time Timepoints for each individual.
- y Outcome.
- class_pred_1 First class predictive covariate (time-invariant).
- class_pred_2 Second class predictive covariate (time-invariant).
- outcome_pred_1 Outcome predictive covariate (time-varying).

summary.BPREM *Summarize the results of a bivariate piecewise random effects model (BPREM)*

Description

Provides a summary of a BPREM model, as returned by `Bayes_BPREM()`.

Usage

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

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

Arguments

`object`	An object of class "BPREM" (returned by `Bayes_BPREM(...)`).
`...`	Additional arguments.
`x`	An object of class "summary.BPREM".

Value

Returns a list of key parameter estimates.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_bprem)
# result summary
summary(results_bprem)
```

summary.CREM *Summarize the results of a crossed random effects model (CREM)*

Description

Provides a summary of a CREM model, as returned by `Bayes_CREM()`.

Usage

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

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

Arguments

`object`	An object of class "CREM" (returned by `Bayes_CREM(...)`).
`...`	Additional arguments.
`x`	An object of class "summary.CREM".

Value

Returns a list of key parameter estimates.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_pcrem)
# result summary
summary(results_pcrem)
```

summary.PREM *Summarize the results of a piecewise random effects model (PREM)*

Description

Provides a summary of a PREM model, as returned by `Bayes_PREM()`.

Usage

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

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

Arguments

`object`	An object of class "PREM" (returned by `Bayes_PREM(...)`).
`...`	Additional arguments.
`x`	An object of class "summary.PREM".

Value

Returns a list of key parameter estimates.

Author(s)

Corissa T. Rohloff

Examples

```
# load fitted model results
data(results_prem)
# result summary
summary(results_prem)
```

Index