Package ‘mlts’

June 27, 2024

Title Multilevel Latent Time Series Models with 'R' and 'Stan'

Version 1.0.0

Description Fit multilevel manifest or latent time-
series models, including popular Dynamic Structural Equation Models (DSEM).
The models can be set up and modified with user-
friendly functions and are fit to the data using 'Stan' for Bayesian inference.
Path models and formulas for user-
defined models can be easily created with functions using 'knitr'.
Asparouhov, Hamaker, & Muthen (2018) <doi:10.1080/10705511.2017.1406803>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.1

URL `https://github.com/munchfab/mlts`

BugReports `https://github.com/munchfab/mlts/issues`

Imports cowplot, dplyr (>= 1.1.3), ggplot2, methods, mvtnorm,
pdftools, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang,
rmarkdown, rstan (>= 2.32.3), rstantools (>= 2.4.0), stats

Suggests knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

Biarch true

Depends R (>= 3.5.0)

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0),
RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>=
2.18.0)

SystemRequirements GNU make

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

Author Kenneth Koslowski [aut, cre, cph]
(<https://orcid.org/0000-0001-5296-5267>),
Fabian Münch [aut] (<https://orcid.org/0000-0001-5591-9901>),
Tobias Koch [aut] (<https://orcid.org/0000-0002-8143-3566>),
Jana Holtmann [aut] (<https://orcid.org/0000-0002-7949-0772>)

Maintainer Kenneth Koslowski <kenneth.koslowski@uni-leipzig.de>

Repository CRAN

Date/Publication 2024-06-27 14:20:02 UTC

Contents

ar1_data 2
create_missings 3
mlts_fit 4
mlts_model 7
mlts_model_formula 10
mlts_model_paths 12
mlts_plot 13
mlts_sim 15
mlts_standardized 16
summary.mltsfit 18
ts_data 19

Index **20**

`ar1_data` *Simple Time-Series Data*

Description

Simulated Data (from `mlts_sim`) for one time-series variable.

Usage

```
ar1_data
```

Format

`ar1_data`**:**

A data frame with 2,500 rows and 3 columns:

ID Unit identifier

time Time point

Y1 The time-series variable

Source

`mlts_sim`

`create_missings` *Create Missings for Approximation of Continuous Time Dynamic Models (new version)*

Description

Create Missings for Approximation of Continuous Time Dynamic Models (new version)

Usage

```
create_missings(data, tinterval, id, time, btw_vars = NULL)
```

Arguments

`data`	An object of class `data.frame` (or one that can be coerced to that class) containing data of all variables used in the model.
`tinterval`	The step interval for approximation for a continuous time DSEM. The smaller the step interval, the better the approximation.
`id`	The variable in `data` that identifies the person or observational unit (as character).
`time`	The variable in `data` that contains the (continuous) time (as string).
`btw_vars`	The names of between-level variables in the data to be added in newly created rows with NAs.

Value

A `data.frame` with missings imputed for use in `mlts_fit`.

Examples

```
# create some data for example
data <- data.frame(
  id = rep(c(1, 2), each = 4),
  time = c(0, 3, 4, 6,
           1, 4, 5, 7)
)

# create missings to approximate continuous time process
create_missings(
  data = data, id = "id", time = "time",
  tinterval = 1 # use time interval of 1 minute
)
```

mlts_fit *Fit Bayesian Multilevel Manifest or Latent Time-Series Models*

Description

Fit Bayesian Multilevel Manifest or Latent Time-Series Models

Usage

```
mlts_fit(
  model,
  data = NULL,
  id,
  ts,
  covariates = NULL,
  outcomes = NULL,
  outcome_pred_btw = NULL,
  center_covs = TRUE,
  time = NULL,
  tinterval = NULL,
  beep = NULL,
  days = NULL,
  n_overnight_NAs,
  na.rm = FALSE,
  iter = 500,
  chains = 2,
  cores = 2,
  monitor_person_pars = FALSE,
  get_SD_latent = FALSE,
  fit_model = TRUE,
  print_message = TRUE,
  print_warning = TRUE,
  ...
)
```

Arguments

`model`	`data.frame`. Output of `mlts_model` and related functions.
`data`	An object of class `data.frame` (or one that can be coerced to that class) containing data of all variables used in the model. Alternatively, a list object with simulated data created by `mlts_sim` can be entered directly and allows for comparison of estimates and true population paramter values used in the data generation.
`id`	Character. The variable in `data` that identifies the observational cluster unit. Not necessary when `data` is a list object of simulated data generated with `mlts_sim`.

`ts`	Character. The variable(s) in `data` that contain the time-series construct(s) or their indicator variable(s). If multiple constructs are provided in the `model`, multiple entries are necessary. Note that the order of variable names provided in `ts` has to match the specification made in the `model`. E.g., if multiple constructs (e.g., `mlts_model(q = 2)`) are provided the order of variables names provided in `ts` determines which construct is referred to as mu_1, phi(1)_11, etc..
`covariates`	Named character vector. An optional named vector of characters to refer to predictors of random effects as specified in the `model`. Note that specifying `covariates` is only necessary if the respective variable name(s) in `data` differ from the variables names specified in `model`.
`outcomes`	Named character vector. Similar to `covariates`, an optional named vector of characters to refer to outcome predicted by random effects as specified in the `model`. Note that specifying `outcomes` is only necessary if the respective variable name(s) in `data` differ from the outcome variable name(s) specified in `model`.
`outcome_pred_btw`	Named character vector. Similar to `covariates`, an optional named vector of characters to refer to additional between-level variables entered as outcome predictor(s) as specified in the `model`. Note that specifying `outcome_pred_btw` is only necessary if the respective variable name(s) in `data` differ from the variable name(s) specified in `model`.
`center_covs`	Logical. Between-level covariates used as predictors of random effects will be grand-mean centered before model fitting by default. Set `center_covs` to `FALSE` when including categorical predictors into the set of `covariates`. Note that in this case, additional continuous covariates should be grand-mean centered prior to using `mlts_fit`.
`time`	Character. The variable in `data` that contains the (continuous) time of observation.
`tinterval`	The step interval for approximating equally spaced observations in time by insertion of missing values, to be specified with respect to the time stamp variable provided in time. Procedure for inserting missing values resembles the procedure for time shift transformation as described in Asparouhov, Hamaker, & Muthén (2018).
`beep`	Character. The variable in `data` that contains the running beep number starting with 1 for each person.
`days`	Optional. If a running beep identifier is provided via the `beep` argument and observations are nested within days (or similar grouping unit), the variable in `data` that contains the day identifier can be added to correct for overnight lags (see Details).
`n_overnight_NAs`	Optional. The number of `NA` rows to add after the last observation of each day (if `days` is provided).
`na.rm`	logical. Per default, missing values remain in the data and will be imputed during model estimation. Set to `TRUE` to remove all rows with missing values in variables given in `ts`.

`iter`	A positive integer specifying the number of iterations for each chain (including 50% used as warmup). The default is 500.
`chains`	A positive integer specifying the number of Markov chains. The default is 2.
`cores`	The number of cores to use when executing the Markov chains in parallel. The default is 2 (see `stan`).
`monitor_person_pars`	Logical. Should person parameters (i.e., values of the latent variables) be stored? Default is FALSE.
`get_SD_latent`	Logical. Set to `TRUE` to obtain standardized estimates in multiple-indicator models.
`fit_model`	Logical. Set to FALSE to avoid fitting the model which may be helpful to inspect prepared data used for model estimation (default = TRUE).
`print_message`	Logical. Print messages based on defined inputs (default = TRUE).
`print_warning`	Logical. Print warnings based on defined inputs (default = TRUE).
`...`	Additional arguments passed to `sampling`.

Value

An object of class `mltsfit`. The object is a list containing the following components:

`model`	the model object passed to `mlts_fit`
`data`	the preprocessed data used for fitting the model
`param.labels`	a `data.frame` that provides the names of parameters used in the stan model. These parameter names are necessary when running standard post-processing functions using `mlts_fit$stanfit`
`pop.pars.summary`	a `data.frame` that contains summary statistics for all parameter in `model`
`person.pars.summary`	if `monitor_person_pars = TRUE`, a `data.frame` containing summary statistics for cluster-specific parameters is provided
`standata`	a `list` with the data as passed to `sampling`
`stanfit`	an object of class `stanfit` with the raw output created by `sampling`
`posteriors`	an `array` of the MCMC chain results for all parameters in `model` created by `rstan::extract` with `dimnames` adapted to match the parameter names provided in `model`

References

Asparouhov, T., Hamaker, E. L., & Muthén, B. (2018). Dynamic Structural Equation Models. Structural Equation Modeling: *A Multidisciplinary Journal*, *25*(3), 359–388. doi:10.1080/10705511.2017.1406803

Examples

```
# build simple vector-autoregressive mlts model for two time-series variables
var_model <- mlts_model(q = 2)

# fit model with (artificial) dataset ts_data
fit <- mlts_fit(
  model = var_model,
  data = ts_data,
  ts = c("Y1", "Y2"), # time-series variables
  id = "ID", # cluster identifier variable
  time = "time", # time variable
  tinterval = 1 # interval for approximation of equidistant measurements,
)

# inspect model summary
summary(fit)
```

`mlts_model` *Build a multilevel latent time series model*

Description

Build a multilevel latent time series model

Usage

```
mlts_model(
  class = c("VAR"),
  q,
  p = NULL,
  max_lag = c(1, 2, 3),
  btw_factor = TRUE,
  btw_model = NULL,
  fix_dynamics = FALSE,
  fix_inno_vars = FALSE,
  fix_inno_covs = TRUE,
  inno_covs_zero = FALSE,
  inno_covs_dir = NULL,
  fixef_zero = NULL,
  ranef_zero = NULL,
  ranef_pred = NULL,
  out_pred = NULL,
  out_pred_add_btw = NULL
)
```

Arguments

`class`	Character. Indicating the model type to be specified. For now restricted to `VAR`, the default. Future package releases might include additional model types.
`q`	Integer. The number of time-varying constructs.
`p`	Integer. For multiple-indicator models, specify a vector of length `q` with the number of manifest indicators per construct. If all constructs are measured with the same number of indicators, a single value is sufficient.
`max_lag`	Integer. The maximum lag of the autoregressive effect to be included in the model. The maximum is 3. Defaults to 1.
`btw_factor`	Logical. If `TRUE` (the default), a common between-level factor is modeled across all indicator variables per construct `q`. If `FALSE`, instead of a between-level factor, indicator mean levels will be included as individual (random) effects drawn from a joint multivariate normal distribution.
`btw_model`	A list to indicate for which manifest indicator variables a common between-level factor should be modeled (see Details for detailed instructions). At this point restricted to one factor per latent construct.
`fix_dynamics`	Logical. Fix all random effect variances of autoregressive and cross-lagged effects to zero (constraining parameters to be equal across clusters).
`fix_inno_vars`	Logical. Fix all random effect variances of innovation variances to zero (constraining parameters to be equal across clusters).
`fix_inno_covs`	Logical. Fix all random effect variances of innovation covariances to zero (constraining parameters to be equal across clusters).
`inno_covs_zero`	Logical. Set to `TRUE` to treat all innovations as independent.
`inno_covs_dir`	For bivariate VAR models with person-specific innovation covariances, a latent variable approach is applied (for a detailed description, see Hamaker et al., 2018). by specifying an additional factor that loads onto the contemporaneous innovations of both constructs, capturing the shared variance of innovations, that is not predicted by the previous time points. The loading parameters of this latent factor, however, have to be restricted in accordance with researchers assumptions about the sign of the association between innovations across construct. Hence, if innovations at time \$t\$ are assumed to be positively correlated across clusters, set the argument to `pos`, or `neg` respectively.
`fixef_zero`	Character. A character vector to index which fixed effects (referring to the parameter labels in `model$Param`) should be constrained to zero (Note: this also results in removing the random effect variance of the respective parameter).
`ranef_zero`	Character. A character vector to index which random effect variances (referring to the parameter labels in `model$Param`) should be constrained to zero.
`ranef_pred`	A character vector or a named list. Include between-level covariate(s) as predictor(s) of all random effects in `model` by entering a vector of unique variable names. Alternatively, to include between-level covariates or differing sets of between-level covariates as predictors of specific random effects, a named list (using the labels in `model$Param`) can be entered (see examples). Note that if a named list is provided, all names that do not match random parameters in `model` will be ignored. Note that variables entered in `ranef_pred` will be grand-mean centered by default when fitting the model with `mlts_fit`.

`out_pred`	A character vector or a named list. Include between-level outcome(s) to be regressed on all random effects in `model` by entering a vector of unique variable names. Alternatively, to include multiple between-level outcomes regressed differing sets of specific random effects, a named list (using the labels in `model$Param`) can be entered (see examples). Note that if a named list is provided, all character strings in the vector of each list (with independent variables) element that do not match random effect parameter names in `model$Param` will be treated as additional between-level predictors.
`out_pred_add_btw`	A character vector. If `out_pred` is a character (vector), all inputs will be treated as between-level covariates to be used as additional predictors of all outcomes specified in `out_pred`.

Value

An object of class `data.frame` with the following columns:

`Model`	Indicates if the parameter in the respective row is part of the structural, or the measurement model (if multiple indicators per construct are provided)
`Level`	Parameter on the between- or within-level.
`Type`	Describes the parameter type.
`Param`	Parameter names to be referred to in arguments of `mlts_model`.
`Param_Label`	Parameter labels (additional option to address specific parameters).
`isRandom`	Indicates which within-level parameters are modeled as random (1) or a constant across clusters (0).
`Constraint`	Optional. Included if multiple-indicators per construct ($p > 1$) are provided. Constraints on measurement model parameters can be changed by overwriting the respective value in `model`. Possible inputs are "free", "= 0" (for SDs of measurement error variances), and "= 1" (for loading parameters).
`prior_type`	Contains the parameters' prior distribution used in `mlts_fit` (prior classes can not be changed at this point).
`prior_location`	Location values of the parameters' prior distribution used in `mlts_fit` (can be changed to any real value by overwriting the respective value in `model`).
`prior_scale`	Scale values of the parameters' prior distribution used in `mlts_fit` (can be changed to any real value by overwriting the respective value in `model`).

References

Hamaker, E. L., Asparouhov, T., Brose, A., Schmiedek, F., & Muthén, B. (2018). At the frontiers of modeling intensive longitudinal data: Dynamic structural equation models for the affective measurements from the COGITO study. *Multivariate behavioral research*, *53*(6), 820-841. doi:10.1080/00273171.2018.1446819

Examples

```
# To illustrate the general model building procedure, starting with a simple
# two-level AR(1) model with person-specific individual means, AR effects,
# and innovation variances (the default option when using mlts_model() and q = 1).
model <- mlts_model(q = 1)

# All model parameters (with their labels stored in model$Param) can be inspected by calling:
model

# Possible model extensions/restrictions:
# 1. Introducing additional parameter constraints, such as fixing specific
#    parameters to a constant value by setting the respective random effect
#    variances to zero, such as e.g. (log) innovation variances
model <- mlts_model(q = 1, ranef_zero = "ln.sigma2_1")
#    Note that setting the argument `fix_inno_vars` to `TRUE` provides
#    a shortcut to fixing the innovation variances of all constructs
#    (if q >= 1) to a constant.

# 2. Including a multiple indicator model, where the construct is measured by
#    multiple indicators (here, p = 3 indicators)
model <- mlts_model(
         q = 1, # the number of time-varying constructs
         p = 3, # the number of manifest indicators
         # assuming a common between-level factor (the default)
         btw_factor = TRUE
       )

# 3. Incorporating between-level variables. For example, inclusion of
#    an additional between-level variable ("cov1") as predictor of all
#    (ranef_pred = "cov1") or a specific set of random effects
#    (ranef_pred = list("phi(1)_11") = "cov1"), an external outcome (e.g., "out1")
#    to be predicted by all (out_pred = "out1") or specific random effects
#    (out_pred = list("out1" = c("etaB_1", "phi(1)_11")), using the latent
#    between-level factor trait scores (etaB_1) and individual first-order
#    autoregressive effects (phi(1)_11) as joint predictors of outcome "out1".
model <- mlts_model(
           q = 1,
           p = 3,
           fix_inno_vars = TRUE,
           ranef_pred = "cov1",
           out_pred = list("out1" = c("etaB_1", "phi(1)_11"))
          )
#    Note that the names of the random effect parameters must match the
#    parameter labels provided in model$Param, the result of the
#    mlts_model()-functions.
```

mlts_model_formula *Create TeX Model Formula from mlts model object*

Description

Create TeX Model Formula from mlts model object

Usage

```
mlts_model_formula(
  model,
  file = NULL,
  keep_tex = FALSE,
  ts = NULL,
  covariates = NULL,
  outcomes = NULL
)
```

Arguments

`model`	A model built with `mlts_model`.
`file`	An optional string containing the name of the file and file path. Has to end with .pdf file format.
`keep_tex`	Logical. Should the TeX file be kept (additional to the Rmd file)? Defaults to `FALSE`.
`ts`	To be included in future releases. An optional character vector containing the names of the time-series variables or indicators.
`covariates`	To be included in future releases. An optional character vector containing the names of the between-level covariates.
`outcomes`	To be included in future releases. An optional character vector containing the names of the between-level outcomes.

Value

An RMarkdown file that is automatically rendered to a pdf document.

Examples

```
# build a simple vector-autoregressive mlts model with two time-series variables
var_model <- mlts_model(q = 2)

# create formula from the specified model
mlts_model_formula(model = var_model)
```

`mlts_model_paths` *Create Path Diagrams from mlts model object*

Description

Create Path Diagrams from mlts model object

Usage

```
mlts_model_paths(
  model,
  file = NULL,
  add_png = FALSE,
  keep_tex = FALSE,
  ts = NULL,
  covariates = NULL,
  outcomes = NULL
)
```

Arguments

`model`	A model built with `mlts_model`.
`file`	An optional string containing the name of the file and file path. Has to end with .pdf file format.
`add_png`	Logical. Set to `TRUE` to transform created PDF to .png file using `pdftools::pdf_convert`.
`keep_tex`	Logical. Should the TeX file be kept (additional to the Rmd file)? Defaults to `FALSE`.
`ts`	To be included in future releases. An optional character vector containing the names of the time-series variables or indicators.
`covariates`	To be included in future releases. An optional character vector containing the names of the between-level covariates.
`outcomes`	To be included in future releases. An optional character vector containing the names of the between-level outcomes.

Value

An RMarkdown file that is automatically rendered to a pdf document.

Examples

```
# build a simple vector-autoregressive mlts model with two time-series variables
var_model <- mlts_model(q = 2)

# create a pathmodel from the specified model
mlts_model_paths(model = var_model)
```

mlts_plot — *Plot results of mlts*

Description

Plot results of mlts

Usage

```
mlts_plot(
  fit,
  type = c("fe", "re", "re.cor"),
  bpe = c("median", "mean"),
  what = c("all", "Fixed effect", "Random effect SD", "RE correlation",
    "Outcome prediction", "RE prediction", "Item intercepts", "Loading",
    "Measurement Error SD"),
  sort_est = NULL,
  xlab = NULL,
  ylab = NULL,
  facet_ncol = 1,
  dot_size = 1,
  dot_color = "black",
  dot_shape = 1,
  errorbar_color = "black",
  errorbar_width = 0.3,
  add_true = FALSE,
  true_color = "red",
  true_shape = 22,
  true_size = 1,
  hide_xaxis_text = TRUE,
  par_labels = NULL,
  labels_as_expressions = FALSE
)
```

Arguments

`fit`	An object of class `mlts.fit`
`type`	Type of plot. type = "fe" (Default) Forest-plot of model coefficients. type = "re" Plot of individual (random) effects type = "re.cor" Combined plot depicting the distribution of individual parameter estimates (posterior summary statistics as provided by bpe), as well as bivariate scatter plots.
`bpe`	The Bayesian point estimate is, by default, the median of the posterior distribution (`bpe = "median"`). Set `bpe = "mean"` to use the mean of the posterior distribution as point estimates.
`what`	Character. For `type = "fe"`, indicate which parameters should be included in the plot by setting `what` to "all" (the default), or one (or multiple) of "Fixed effect",

	"Random effect SD", "RE correlation", "Outcome prediction", "RE prediction", "Item intercepts", "Loading", or "Measurement Error SD".
`sort_est`	Add parameter label for sorting of random effects.
`xlab`	Title for the x axis.
`ylab`	Title for the y axis.
`facet_ncol`	Number of facet columns (see `ggplot2::facet_grid`).
`dot_size`	numeric, size of the dots that indicate the point estimates.
`dot_color`	character. indicating the color of the point estimates.
`dot_shape`	numeric. shape of the dots that indicate the point estimates.
`errorbar_color`	character. Color of error bars.
`errorbar_width`	integer. Width of error bars.
`add_true`	logical. If model was fitted with simulated data using `mlts_sim`, true population parameter values can be plotted as reference by setting the argument ot `TRUE`.
`true_color`	character. Color of points depicting true population parameter used in the data generation.
`true_shape`	integer. Shape of points depicting true population parameter used in the data generation.
`true_size`	integer. Size of points depicting true population parameter used in the data generation.
`hide_xaxis_text`	logical. Hide x-axis text if set to `TRUE`.
`par_labels`	character vector. User-specified labels for random effect parameters can be specified.
`labels_as_expressions`	logical. Should parameter names on plot labels be printed as mathematical expressions? Defaults to `FALSE`. Still experimental.

Value

Returns a `ggplot`-object .

Examples

```
# build simple vector-autoregressive mlts model for two time-series variables
var_model <- mlts_model(q = 2)

# fit model with (artificial) dataset ts_data
fit <- mlts_fit(
  model = var_model,
  data = ts_data,
  ts = c("Y1", "Y2"), # time-series variables
  id = "ID", # identifier variable
  time = "time",
  tinterval = 1 # interval for approximation of continuous-time dynamic model,
)
```

```
# inspect model summary
mlts_plot(fit, type = "fe", what = "Fixed effect")
```

---

mlts_sim *Simulate data from mlts model*

Description

Simulate data from mlts model

Usage

```
mlts_sim(
  model,
  default = FALSE,
  N,
  TP,
  burn.in = 50,
  seed = NULL,
  seed.true = 1,
  btw.var.sds = NULL
)
```

Arguments

`model`	`data.frame`. Output of `mlts_model`.
`default`	logical. If set to `TRUE`, default prior specifications are added.
`N`	integer Number of observational units.
`TP`	integer. Number of measurements per observational unit.
`burn.in`	integer. Length of 'burn-in' period.
`seed`	integer. Seed used for data generation.
`seed.true`	integer. Separate seed used for sampling of true population parameters values from plausible ranges for stationary time series.
`btw.var.sds`	named numeric vector. Provide standard deviation(s) for all exogenous between-level variable(s) specified in `model`, e.g. (`btw.var.sds = c("covariate1" = 1)`, to set the SD of the variable "covariate1" to 1). Mean values of the respective variable(s) will be set to 0 per default.

Details

A function to generate data from an output of `mlts_model`.

Value

An object of class "`mlts_simdata`". The object is a list containing the following components:

`model`	the model object passed to `mlts_sim` with true parameter values used in the data generation added in the column `true.val`
`data`	a long format `data.frame` of the generated time series data
`RE.pars`	a `matrix` of cluster-specific true values used in the data generation

Examples

```
# build a simple vector-autoregressive mlts model with two time-series variables
var_model <- mlts_model(q = 2)

# simulate data from this model with default true values
# (true values are randomly drawn from normal distribution)
var_data <- mlts_sim(
  model = var_model,
  N = 50, TP = 30, # number of units and number of measurements per unit
  default = TRUE # use default parameter values
)

# the data set is stored in .$data
head(var_data$data)

# individual parameter values are stored in .$RE.pars
head(var_data$RE.pars)

# if the mltssim-object is used in mlts_fit(), true values
# are added to the fitted object
fit <- mlts_fit(
  model = var_model,
  data = var_data,
  id = "ID", ts = c("Y1", "Y2"), time = "time"
)

# inspect model with true values
head(fit$pop.pars.summary)
```

`mlts_standardized` *Get Standardized Estimates for an mlts Model*

Description

Get Standardized Estimates for an mlts Model

Usage

```
mlts_standardized(
  object,
  what = c("between", "within", "both"),
  digits = 3,
  prob = 0.95,
  add_cluster_std = FALSE
)
```

Arguments

`object`	`mltsfit`. Output of `mlts_model` and related functions.
`what`	character. Get between-level standardized estimates (`what = "between"`, the default), within-level standardized estimates averaged over clusters (`what = "within"`), or both (`what = "both"`).
`digits`	Number of digits. Default is 3.
`prob`	A value between 0 and 1 to indicate the width of the credible interval. Default is .95.
`add_cluster_std`	logical. If `what = "within"`, within-level standardized effects for each cluster are included in the output (defaults to `FALSE`).

Value

A `list` containing between- and within-level standardized parameters.

Examples

```
# build simple vector-autoregressive mlts model for two time-series variables
var_model <- mlts_model(q = 2)

# fit model with (artificial) dataset ts_data
fit <- mlts_fit(
  model = var_model,
  data = ts_data,
  ts = c("Y1", "Y2"), # time-series variables
  id = "ID", # identifier variable
  time = "time", # time variable
  tinterval = 1, # interval for approximation of continuous-time dynamic model,
  monitor_person_pars = TRUE # person parameters need to be sampled for standardization
)

# inspect standardized parameter estimates
mlts_standardized(fit)
```

summary.mltsfit *Create a summary of a fitted model with class* `mltsfit`

Description

Create a summary of a fitted model with class `mltsfit`

Usage

```
## S3 method for class 'mltsfit'
summary(
  object,
  priors = FALSE,
  se = FALSE,
  prob = 0.95,
  bpe = c("mean", "median"),
  digits = 3,
  flag_signif = FALSE,
  ...
)
```

Arguments

`object`	An object of class `mltsfit`.
`priors`	Add prior information (default = FALSE).
`se`	Logical. Should the Monte Carlo Standard Error be included in the summary? Defaults to `FALSE`.
`prob`	A value between 0 and 1 to indicate the width of the credible interval. Default is .95.
`bpe`	Bayesian posterior estimate can be either "mean" (the default) or the "median" of the posterior distribution.
`digits`	Number of digits.
`flag_signif`	Add significance flags based on `prob` (default = FALSE).
`...`	Additional arguments affecting the summary produced.

Value

A summary of model parameters.

Examples

```
# build simple vector-autoregressive mlts model for two time-series variables
var_model <- mlts_model(q = 2)

# fit model with (artificial) dataset ts_data
fit <- mlts_fit(
  model = var_model,
  data = ts_data,
  ts = c("Y1", "Y2"), # time-series variables
  id = "ID", # identifier variable
  time = "time",
  tinterval = 1 # interval for approximation of continuous-time dynamic model,
)

# inspect model summary
summary(fit)
```

ts_data *Simple Time-Series Data*

Description

Simulated Time-Series Data (from mlts_sim) for two time-series variables.

Usage

```
ts_data
```

Format

ts_data:
A data frame with 1,100 rows and 4 columns:

ID Unit identifier

time Time point

Y1, Y2 The two time-series variables

Source

mlts_sim

Index

∗ **datasets**
  ar1_data, 2
  ts_data, 19

ar1_data, 2

create_missings, 3

mlts_fit, *3*, 4
mlts_model, *4*, 7, *11, 12*, *15*, *17*
mlts_model_formula, 10
mlts_model_paths, 12
mlts_plot, 13
mlts_sim, *2*, 15, *19*
mlts_standardized, 16

sampling, *6*
stan, *6*
summary.mltsfit, 18

ts_data, 19