Package 'mlts'

December 8, 2025

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

Version 2.0.0

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,
shape, diagram, grDevices, graphics

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

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

Repository CRAN

Date/Publication 2025-12-08 16:30:02 UTC

Contents

`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*

Description

Create Missings for Approximation of Continuous Time Dynamic Models

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,
  group = NULL,
  ts,
  covariates = NULL,
  outcomes = NULL,
  outcome_pred_btw = NULL,
  center_covs = TRUE,
  time = NULL,
  tinterval = NULL,
  beep = NULL,
  days = NULL,
  n_overnight_NAs,
  max_NA_seq = NULL,
  na.rm = FALSE,
  iter = 500,
  chains = 2,
  cores = 2,
  monitor_person_pars = FALSE,
  monitor_all_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`.
- `group` — Character. The variable in `data` that identifies the grouping variable across cluster units.
- `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`. If `group` is specified, covariates will be centered on their respective group mean.
- `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).

max_NA_seq
: Integer. Specify a maximum number of consecutive missing values. Can decrease estimation times drastically in the presence of very long sequences of missing values (e.g., when setting tinterval to values of small time steps).

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.

monitor_all_pars
: Logical. Should all parameters 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,
  equal_loads_levels = FALSE,
  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,
  group = NULL,
  is_exogenous = NULL,
  incl_t0_effects = NULL,
  incl_interaction_effects = NULL,
  censor_left = NULL,
  censor_right = NULL,
  silent = FALSE
)
```

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.
- `equal_loads_levels` Logical. For multiple-indicator model with `btw_factor = TRUE`, if `TRUE`, factor loadings of the same indicators are assumed to be equal across levels. Note, that the first indicator loading parameters remain fixed to `1`.
- `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`.

`group`
: An integer specifying the number of groups (not yet supported). Add a binary coded (0 vs. 1) variable to include group differences in fixed effects (intercepts). When dynamic or variance parameters are allowed to vary by cluster, you can enter the grouping variable to `re_pred`.

`is_exogenous`
: Integer or a vector of integers. Indicate if any of the constructs should be treated as exogenous (i.e., no latent mean centering will be performed). Probable use case: Adding a dichotomous time-varying predictor variable.

`incl_t0_effects`
: A character vector. Experimental: Add contemporaneous effects to the model. For example, to include an effect of the first construct on the second construct at time t, following the general pattern for naming of dynamic parameters in the mlts framework, can be included by specifying `phi(0)_21` where the `0` indicates the lag, the first subscript letter (`2`) the dependent, and the latter subscript (`1`) the independent construct. The respective within-level correlation/covariance of innovations between involved constructs will be excluded from the model accordingly.

incl_interaction_effects
: A character vector. Add interaction terms on the dynamic within-level. For example, to add an interaction term between the first construct at time t (lag of 0) and the second construct at $t-1$ (lag of 1) to the prediction of the second construct at time t specify `incl_interaction_effects = phi(i)_2.2(1)1(0)`. where the `i` indicates an interaction effect, the first subscript letter (`2`) the dependent, and the latter subscripts after the dot (i.e., `2(1)` and `1(0)`) the independent constructs involved in the interaction each followed by the respective lag in brackets. Note, that in this case the respective lag 0 effects need to be included separately using `incl_t0_effects`.

censor_left
: Numeric. If an input is provided (i.e., a single numeric value) a left-censored version of the model will be estimated by treating all observations (of manifest indicators) at the censoring threshold (i.e., usually the lower bound of the scale) to be treated as missing during model estimation. These missing values (observations at the value of `censor_left`) are replaced with imputed values (declared as parameters in the stan model) with an upper limit of `censor_left` (see https://mc-stan.org/docs/stan-users-guide/truncation-censoring.html). Note that all manifest variables are affected by the censoring. To prevent individual variables from being treated as censored you could change the scale of the respective variable(s) so that all values exceed the censoring threshold.

censor_right
: Numeric. Developmental. Similar to `censor_left` but assumes variables to be censored on the upper bound of the scale. Can be combined with `censor_left`.

silent
: logical. Set to `TRUE` to suppress warnings and messages.

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

Deprecated. Please use mlts_paths.

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_paths *Plot Paths for Two-Level VAR Model*

Description

The mlts_paths function depcits models specified using mlts_model as a path diagram.

Usage

```
mlts_paths(
  model,
  asp_decomp = 0.25,
  asp_w_b = 0.5,
  fig_margins.x = c(0, 8),
  fig_margins.y = c(0, 8),
  width = 7,
  height = 5,
  file = NULL,
  asp = height/width,
  family = "serif",
  cex_b = 0.8,
  cex_w = 1,
  cex_decomp = 1,
  cex_loads = 0.8,
  b_style = "h",
  w_y_offset = 0,
  decomp_F_y_offset = 4,
  arrHead_w = 0.16,
  arrHead_b = 0.16,
  scale_decomp_ind = 0.35,
  scale_decomp_F = 0.45,
  scale_within = 0.3,
  scale_within_inno = 0.2,
  scale_between = 0.3,
  scale_int = 0.25,
  lwd_nodes = 1.7,
  rand_dot_pos = 0.4,
  units = "in",
  res = 700,
  pointsize = 10,
  type = "cairo",
  y_ind_labs = NULL,
  y_fac_labs = NULL,
  y_fac_lab_sep = ",",
  remove_lag_lab = FALSE,
  adj_load_x = 1.25,
  ...
```

```
)
```

Arguments

`model`	`data.frame`. Output of `mlts_model` and related functions.
`asp_decomp`	A numeric value specifying the aspect ratio for the decomposition plot region. Defaults to 0.25.
`asp_w_b`	A numeric value specifying the aspect ratio between the within-level and between-level sections. Defaults to 0.5.
`fig_margins.x`	A numeric vector of length 2 defining the horizontal margins of the plot. Defaults to `c(0, 8)`.
`fig_margins.y`	A numeric vector of length 2 defining the vertical margins of the plot. Defaults to `c(0, 8)`.
`width`	Width of the plot in inches. Defaults to 7.
`height`	Height of the plot in inches. Defaults to 5.
`file`	A character string specifying the path to save the plot. If `NULL`, the plot will not be saved. Defaults to `NULL`.
`asp`	The overall aspect ratio of the plot, computed as `height` / `width`. Defaults to `height` / `width`.
`family`	Font family used in the plot. Defaults to `"serif"`.
`cex_b`	Numeric value specifying the scaling of text in the between-level section. Defaults to 0.8.
`cex_w`	Numeric value specifying the scaling of text in the within-level section. Defaults to 0.8.
`cex_decomp`	Numeric value specifying the scaling of text in the decomposition section. Defaults to 0.8.
`cex_loads`	Numeric value specifying the scaling of text of loading parameters. Defaults to 0.8.
`b_style`	A character string specifying the style of the between-level plot ("h" for horizontal). Defaults to `"h"`.
`w_y_offset`	Numeric value specifying the vertical width of the within-level part. Defaults to 0.
`decomp_F_y_offset`	Numeric value to control the vertical space between manifest indicators and latent factors in the decomposition part of the path model. Defaults to 4.
`arrHead_w`	Numeric values controlling the arrowhead size for within-level paths. Defaults to 0.16.
`arrHead_b`	Numeric values controlling the arrowhead size for between-level paths. Defaults to 0.16.
`scale_decomp_ind`	Numeric. Specify the scaling factor for manifest indicators in the decomposition section.
`scale_decomp_F`	Numeric. Specify the scaling factor for latent factors in the decomposition section.

`scale_within`	Numeric. Specify the scaling factor for latent factors in the within-level section.
`scale_within_inno`	Numeric. Specify the scaling factor for innovations in the within-level section.
`scale_between`	Numeric. Specify the scaling factor for factors in the between-level section.
`scale_int`	Numeric. Specify the scaling factor for interaction factors in the within-level section.
`lwd_nodes`	Line width for node borders in the plot. Defaults to 1.7.
`rand_dot_pos`	Numeric value controlling the random dot position in the plot. Defaults to 0.5.
`units`	A character string specifying the units for saving the plot. Defaults to "in".
`res`	The nominal resolution in ppi. Defaults to 320.
`pointsize`	Numeric value specifying the font point size for the plot. Defaults to 10.
`type`	A character string specifying the file type for the saved plot (e.g., "cairo"). Defaults to "cairo".
`y_ind_labs`	A vector of character strings with names of observed variables.
`y_fac_labs`	A vector of character strings with factor labels to replace numeric indices in parameter names.
`y_fac_lab_sep`	A character string to separate multiple factor labels. Defaults to ",".
`remove_lag_lab`	Logical. Remove lag index from phi-parameter labels. Defaults to `FALSE`.
`adj_load_x`	Numeric value specifying the x-axis offset loading parameter labels. Defaults to 1.25.
`...`	Additional arguments passed to internal plotting functions.

Details

This function calculates positions, radii, and labels for nodes and arrows based on the model structure and its parameters. It divides the plot into sections:

- **Decomposition**: Shows the breakdown of observed variables into within- and between-level components.
- **Within-Level Dynamics**: Illustrates autoregressive and cross-lagged paths between variables at the within level.
- **Between-Level Dynamics**: Depicts random effects, covariates, and their interrelations at the between level.

Depending on the model structure (e.g., maximum lag, number of random effects, presence of interaction terms), the function dynamically adjusts the visualization.

Value

A graphical object representing the path diagram of the model.

Examples

```
# A two-level second-order autoregressive model
model <- mlts_model(q = 1, max_lag = 2)

# Plot the paths
mlts_paths(model)
```

---

mlts_plot *Plot results of mlts*

Description

Plot results of mlts

Usage

```
mlts_plot(
  fit,
  type = c("fe", "re", "re.cor", "int"),
  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 = "int" Experimental: Plot within-level interactions. 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_posterior_sample *Generate Posterior Predictive Samples for Multilevel Latent Time Series Models*

Description

The `mlts_posterior_sample()` function generates replicated datasets from a fitted `mlts` model using draws from the posterior distribution. The function can simulate data under the population model or based on individual-specific (random effect) parameters.

Usage

```
mlts_posterior_sample(
  fit,
  draw_person_pars = FALSE,
  n_draws = 10,
  draws = NULL,
  as_matrix = TRUE
)
```

Arguments

`fit` An object of class `mlts.fit`, as returned by a fitted model using `mlts()`.

`draw_person_pars`
Logical. If `TRUE`, samples are generated using person-specific parameters (random effects). If `FALSE`, only population-level parameters are used. Defaults to `FALSE`.

`n_draws` Integer. Number of posterior draws to use for simulating replicated datasets. Ignored if `draws` is provided. Defaults to 10.

draws	Optional integer vector indicating specific posterior draw indices to use. If NULL, n_draws draws are chosen with the maximum distance between posterior samples.

as_matrix	Logical. Return replications of each variable as a matrix with n_draw rows, ready to run graphical posterior predictive checks using the bayesplot package.

Details

The function extracts posterior samples of population-level (and optionally individual-level) parameters from a fitted mlts model and simulates replicated datasets from the posterior predictive distribution. Each replication corresponds to a different posterior draw and reflects uncertainty in the model's parameters. See PPC for an overview on graphical posterior predictive checks and how they can be performed.

If draw_person_pars = TRUE, the function uses sampled person-specific random effects and covariate effects from the posterior to generate new data at the individual level. This requires that the model was fitted with monitor_person_pars = TRUE in mlts_fit. If this condition is not met, the function will throw an error.

Posterior draws are either selected with the maximum distance between posterior samples (n_draws) or specified manually using the draws argument. Optionally, left or right censoring is respected in the simulated data if such constraints were present in the model.

Value

A list of replicated datasets, each as a data.frame with columns:

Y_rep Replication number.

ID Subject/cluster ID.

time Time point.

... One column per time-series variable defined in the model.

See Also

mlts_pp_check for plotting posterior predictive checks.

Examples

```
## Not run:
# 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
)

# fit model
```

```
fit <- mlts_fit(
  model = var_model,
  data = var_data,
  id = "ID", ts = c("Y1", "Y2"),
  time = "time",
  monitor_person_pars = TRUE
)

# Simulate 20 replications from the posterior
yreps <- mlts_posterior_sample(fit = fit, n_draws = 20)

# Include person-specific parameters in simulation
yreps <- mlts_posterior_sample(fit = fit, draw_person_pars = TRUE)

# Use specific posterior draws
yreps <- mlts_posterior_sample(fit = fit, draws = c(10, 50, 100))

## End(Not run)
```

`mlts_pp_check` *Posterior Predictive Checks for Multilevel Latent Time Series Models*

Description

Developemental This function plots posterior predictive distributions of one or multiple fitted `mlts.fit` models. Simulated data from posterior draws are compared to the observed data to visually assess model fit.

Usage

```
mlts_pp_check(
  fit,
  fit_list = NULL,
  ts = NULL,
  y_reps = NULL,
  by_cluster = FALSE,
  by_group = FALSE,
  cluster_ids = NULL,
  draw_person_pars = FALSE,
  n_draws = 10,
  draws = NULL,
  add_y_obs = TRUE,
  model_lab = NULL,
  y_rep_col = NULL,
  y_obs_col = "#009E73",
  y_obs_lw = 1.1,
  y_rep_lw = 0.5,
  y_rep_alpha = 0.5
)
```

Arguments

fit	A fitted model object of class mlts.fit. Only used if fit_list is NULL.
fit_list	An optional list of fitted mlts.fit objects for model comparison. If provided, fit is ignored.
ts	Optional vector of variable names to include in the plot.
y_reps	Optional. A list of posterior predictive samples (as returned by mlts_posterior_sample) for a single model. If NULL, samples are generated within the function.
by_cluster	Logical. If TRUE, density plots are faceted by individual and time-series variable. If FALSE, only time-series variables are used for faceting. Default is FALSE.
by_group	Logical. If TRUE, density plots are faceted by grouping and time-series variable. If FALSE, only time-series variables are used for faceting. Default is FALSE.
cluster_ids	Optional vector of cluster IDs to include in the plot. If NULL, all IDs are shown.
draw_person_pars	Logical. If TRUE, samples are generated using person-specific parameters (random effects). If FALSE, only population-level parameters are used. Defaults to FALSE.
n_draws	Integer. Number of posterior draws to use for generating replicated datasets. Defaults to 20. Ignored if draws is specified.
draws	Optional vector of indices specifying which posterior draws to use. If NULL, n_draws samples are drawn randomly.
add_y_obs	Logical. Whether to include the observed data distribution in the plot. Defaults to TRUE.
model_lab	Optional character vector with labels for each model in fit_list. If NULL, defaults to "Model 1", "Model 2", etc.
y_rep_col	Optional vector of colors for the posterior predictive densities of each model. If NULL, a default color palette is used.
y_obs_col	Color for the observed data distribution. Default is "#009E73".
y_obs_lw	Line width of the observed data density curve. Default is 1.1.
y_rep_lw	Line width of the observed data density curve. Default is 0.5.
y_rep_alpha	Alpha transparency for the predictive density curves. Default is 0.5.

Details

This function performs graphical posterior predictive checks by overlaying kernel density estimates of replicated data from the posterior with the observed data. This can be used to visually assess how well a fitted model captures key distributional aspects of the observed time series. If fit_list is specified, multiple models can be compared side-by-side in the same plot.

If draw_person_pars = TRUE, simulated datasets incorporate subject-specific effects (random effects). This requires that monitor_person_pars = TRUE was set during model fitting.

Value

A ggplot object showing density curves of observed and replicated data across time-series variables (and optionally across individuals).

See Also

mlts_posterior_sample for generating replicated data samples.

Examples

```
## Not run:
# Set up AR(1) model
ar1 <- mlts_model(q = 1, censor_left = -1)

# Simulate data under the AR(1) model
simData <- mlts_sim(model = ar1, N = 50, TP =100, default = TRUE)

# Fit the model
fit_AR <- mlts_fit(model = ar1, data = simData$data,
                   id = "ID", ts = "Y1", monitor_person_pars = TRUE)

# Run posterior predictive check
mlts_pp_check(fit = fit_AR,
              model_lab = "AR(1)",
              y_rep_col = "steelblue")


## End(Not run)
```

---

mlts_sim *Simulate data from mlts model*

Description

Simulate data from mlts model

Usage

```
mlts_sim(
  model,
  default = FALSE,
  N = NULL,
  N_G = NULL,
  TP,
  burn.in = 50,
  seed = NULL,
  seed.true = 1,
  btw.var.sds = NULL,
  exogenous = 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.
N_G	vector of integers. Number of observational units per group.
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.
exogenous	Matrix of numeric values of exogenous variables with N*(TP+burn.in) rows and separate columns for each variable.

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,
  get_samples = 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).
get_samples	logical. For internal use.

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"),
  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