Package ‘simglm’

May 31, 2019

Type Package

Version 0.7.4

License MIT + file LICENSE

Title Simulate Models Based on the Generalized Linear Model

Description Simulates regression models,
including both simple regression and generalized linear mixed
models with up to three level of nesting. Power simulations that are
flexible allowing the specification of missing data, unbalanced designs,
and different random error distributions are built into the package.

Depends R (>= 3.3.0)

Imports stats, methods, Matrix, rlang, dplyr, purrr, broom,
future.apply

Suggests knitr, lme4, nlme, testthat, shiny, e1071, ggplot2, tidyr,
geepack, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 6.1.1

Author Brandon LeBeau [aut, cre]

Maintainer Brandon LeBeau <lebebr01+simglm@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2019-05-31 17:10:03 UTC

R topics documented:

`compute_statistics` *Compute Power, Type I Error, or Precision Statistics*

Description

Compute Power, Type I Error, or Precision Statistics

Usage

```
compute_statistics(data, sim_args, power = TRUE, type_1_error = TRUE,
  precision = TRUE)
```

Arguments

`data` A list of model results generated by `replicate_simulation` function.

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`power` TRUE/FALSE flag indicating whether power should be computed. Defaults to TRUE.

`type_1_error` TRUE/FALSE flag indicating whether type I error rate should be computed. Defaults to TRUE.

`precision` TRUE/FALSE flag indicating whether precision should be computed. Defaults to TRUE.

`corr_variables` *Function to correlate variables*

Description

Inputs a matrix and other parameters and outputs a correlated matrix

Usage

```
corr_variables(mat, cor_vars, cov_param, standardize = TRUE)
```

Arguments

`mat` A matrix of variables to correlate

`cor_vars` A vector of correlations to specify, must be specified by row where the first element is the correlation between variable 1 and variable 2, second correlation is between variable 1 and variable 3, and so on.

`cov_param` Variable specification similar to specifying fixed effects. See `sim_reg` for more details.

`standardize` TRUE/FALSE flag indicating whether variables should be standardized prior to correlating (this is needed for accurate correlated variables)

`cross_class` *Cross Classified Generation*

Description

Input cross classified simulation parameters, output cross classified structure as a function of the original id variables. This function currently only supports a single (intercept) cross classified random effect.

Usage

```
cross_class(num_ids, samp_size, random_param)
```

Arguments

`num_ids` Number of cross classified ids to generate.

`samp_size` Sample size to generate, this is used to pass to the `sample` function.

`random_param` A list of data generating characteristics used to generate the cross classified random effect. This function needs to include:

- random_var The variance of the cross classified random effect.
- rand_gen The random generating function used.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

See `sim_rand_eff` for additional parameters that can be passed.

`data_glm_nested` *Generate logistic regression outcome*

Description

Takes simulation parameter arguments and returns simulated data for two different probability distributions. One is logistic (0/1) outcome and the second being poisson (count) outcomes.

Usage

```
data_glm_nested(Xmat, Zmat, beta, rand_eff, n, p, outcome_type)
```

Arguments

`Xmat`	A matrix of covariates.
`Zmat`	Design matrix for random effects.
`beta`	A vector of regression parameters.
`rand_eff`	A vector of random effects, must be stacked.
`n`	Number of clusters.
`p`	Number of units within each cluster.
`outcome_type`	A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`data_glm_nested3` *Simulates three level nested data with a single third level random effect*

Description

Takes simulation parameter arguments and returns simulated data for two different probability distributions. One is logistic (0/1) outcome and the second being poisson (count) outcomes.

Usage

```
data_glm_nested3(Xmat, Zmat, Zmat3, beta, rand_eff, rand_eff3, k, n, p,
  outcome_type)
```

Arguments

`Xmat` A matrix of covariates.

`Zmat` Design matrix for random effects.

`Zmat3` Design matrix for level 3 random effects.

`beta` A vector of regression parameters.

`rand_eff` A vector of random effects, must be stacked.

`rand_eff3` A vector of level 3 random effects, must be stacked.

`k` Number of third level clusters.

`n` Number of clusters.

`p` Number of units within each cluster.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`data_glm_single` *Generate logistic regression outcome*

Description

Takes simulation parameter arguments and returns simulated data for two different probability distributions. One is logistic (0/1) outcome and the second being poisson (count) outcomes.

Usage

```
data_glm_single(Xmat, beta, n, outcome_type)
```

Arguments

`Xmat` A matrix of covariates.

`beta` A vector of regression parameters.

`n` Number of clusters.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

data_reg_nested *Simulates two level nested data*

Description

Takes simulation parameter arguments and returns simulated data.

Usage

```
data_reg_nested(Xmat, Zmat, beta, rand_eff, n, p, err)
```

Arguments

Xmat	A matrix of covariates.
Zmat	Design matrix for random effects.
beta	A vector of regression parameters.
rand_eff	A vector of random effects, must be stacked.
n	Number of clusters.
p	Number of units within each cluster.
err	A vector of within cluster errors.

data_reg_nested3 *Simulates three level nested data with a single third level random effect*

Description

Takes simulation parameter arguments and returns simulated data.

Usage

```
data_reg_nested3(Xmat, Zmat, Zmat3, beta, rand_eff, rand_eff3, k, n, p,
  err)
```

Arguments

Xmat	A matrix of covariates.
Zmat	Design matrix for random effects.
Zmat3	Design matrix for level 3 random effects.
beta	A vector of regression parameters.
rand_eff	A vector of random effects, must be stacked.
rand_eff3	A vector of level 3 random effects, must be stacked.
k	Number of third level clusters.
n	Number of clusters.
p	Number of units within each cluster.
err	A vector of within cluster errors.

`data_reg_single` *Simulates single level data*

Description

Takes simulation parameter arguments and returns simulated data.

Usage

```
data_reg_single(Xmat, beta, n, err)
```

Arguments

`Xmat`	A matrix of covariates.
`beta`	A vector of regression parameters.
`n`	Number of clusters.
`err`	A vector of within cluster errors.

Details

This is a helper function to the master function `sim_reg`, this function does the actual simulation to return the data for single level models.

`desireVar` *Computes mixture normal variance*

Description

Input the desired variance, number of distributions, and mean of the distributions, returns a value of the variance of each mixture distribution.

Usage

```
desireVar(desVar, num_dist, means, equalWeight = TRUE)
```

Arguments

`desVar`	Desired overall variance of mixture normal distribution.
`num_dist`	Number of normal distributions.
`means`	Vector of means for each normal distribution. Must equal num_dist.
`equalWeight`	Should equal weights be used, only TRUE is currently supported.

Details

This function can be used to generate the inputs for the `rbimod` variances when a specific variance is desired. Especially useful when attempting to simulate a mixture normal/bimodal distribution.

Examples

```
# calculating variance to be 2.5 with 2 distributions
desireVar(2.5, 2, means = c(-1, 1), equalWeight = TRUE)
```

`extract_coefficients` *Extract Coefficients*

Description

Extract Coefficients

Usage

```
extract_coefficients(model, extract_function = NULL)
```

Arguments

`model` A returned model object from a fitted model.

`extract_function`
A function that extracts model results. The function must take the model object as the only argument.

`generate_missing` *Tidy Missing Data Function*

Description

Tidy Missing Data Function

Usage

```
generate_missing(data, sim_args)
```

Arguments

`data` Data simulated from other functions to pass to this function.

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`generate_response` *Simulate response variable*

Description

Simulate response variable

Usage

```
generate_response(data, sim_args, keep_intermediate = TRUE, ...)
```

Arguments

`data` Data simulated from other functions to pass to this function.

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`keep_intermediate` TRUE/FALSE flag indicating whether intermediate steps should be kept. This would include fixed effects times regression weights, random effect summations, etc. Default is TRUE.

`...` Other arguments to pass to error simulation functions.

`missing_data` *Missing Data Functions*

Description

Function that inputs simulated data and returns data frame with new response variable that includes missing data. Missing data types incorporated include dropout missing data, missing at random, and random missing data.

Usage

```
missing_data(sim_data, resp_var = "sim_data",
  new_outcome = "sim_data2", clust_var = NULL, within_id = NULL,
  miss_prop = NULL, dropout_location = NULL, type = c("dropout",
  "random", "mar"), miss_cov, mar_prop)

dropout_missing(sim_data, resp_var = "sim_data",
  new_outcome = "sim_data2", clust_var = "clustID",
  within_id = "withinID", miss_prop = NULL, dropout_location = NULL)
```

```
random_missing(sim_data, resp_var = "sim_data",
  new_outcome = "sim_data2", miss_prop, clust_var = NULL,
  within_id = "withinID")

mar_missing(sim_data, resp_var = "sim_data", new_outcome = "sim_data2",
  miss_cov, mar_prop)
```

Arguments

`sim_data` Simulated data frame

`resp_var` Character string of response variable with complete data.

`new_outcome` Character string of new outcome variable name that includes the missing data.

`clust_var` Cluster variable used for the grouping, set to NULL by default which means no clustering.

`within_id` ID variable within each cluster.

`miss_prop` Proportion of missing data overall

`dropout_location`
A vector the same length as the number of clusters representing the number of data observations for each individual.

`type` The type of missing data to generate, currently supports dropout, random, or missing at random (mar) missing data.

`miss_cov` Covariate that the missing values are based on.

`mar_prop` Proportion of missing data for each unique value specified in the miss_cov argument.

`model_fit` *Tidy Model Fitting Function*

Description

Tidy Model Fitting Function

Usage

```
model_fit(data, sim_args, ...)
```

Arguments

`data` A data object, most likely generated from within simglm

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)

- error: This is the error (i.e. residual term).
- model_fit: These are arguments passed to the model_fit function.

... Currently not used.

parse_crossclass *Parse Cross-classified Random Effects*

Description

Parse Cross-classified Random Effects

Usage

```
parse_crossclass(sim_args, random_formula_parsed)
```

Arguments

sim_args Simulation arguments

random_formula_parsed
This is the output from parse_randomeffect.

parse_formula *Parses tidy formula simulation syntax*

Description

A function that parses the formula simulation syntax in order to simulate data.

Usage

```
parse_formula(sim_args)
```

Arguments

sim_args A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`parse_power` *Parse power specifications*

Description

Parse power specifications

Usage

```
parse_power(sim_args)
```

Arguments

`sim_args`	A named list with special model formula syntax. See details and examples for more information. The named list may contain the following: • fixed: This is the fixed portion of the model (i.e. covariates) • random: This is the random portion of the model (i.e. random effects) • error: This is the error (i.e. residual term).

`parse_randomeffect` *Parses random effect specification*

Description

Parses random effect specification

Usage

```
parse_randomeffect(formula)
```

Arguments

`formula`	Random effect formula already parsed by `parse_formula`

parse_varyarguments *Parse varying arguments*

Description

Parse varying arguments

Usage

```
parse_varyarguments(sim_args)
```

Arguments

sim_args	A named list with special model formula syntax. See details and examples for more information. The named list may contain the following: • fixed: This is the fixed portion of the model (i.e. covariates) • random: This is the random portion of the model (i.e. random effects) • error: This is the error (i.e. residual term).

rbimod *Simulating mixture normal distributions*

Description

Input simulation metrics returns mixture normal random variable.

Usage

```
rbimod(n, mean, var, num_dist)
```

Arguments

n	Number of random draws. Optionally can be a vector with number in each simulated normal distribution.
mean	Vector of mean values for each normal distribution. Must be the same length as num_dist.
var	Vector of variance values for each normal distribution. Must be the same length as num_dist.
num_dist	Number of normal distributions to use when simulating mixture normal distribution.

Details

Function to simulate mixture normal distributions. The function computes adds the specified number of normal distributions into a single vector.

Use of the function desireVar can be used to generate a mixture normal distribution with a specific global variance.

Examples

```
## mix normal with two normal distributions (bimodal)
simData <- rbimod(100, mean = c(-2, 3), var = c(1.5, 1.5), num_dist = 2)
plot(density(simData))

## mixt normal with four distributions (multimodal)
simData <- rbimod(400, mean = c(-14, -4, 6, 20), var = c(rep(1.2, 4)),
  num_dist = 4)
plot(density(simData))
```

replicate_simulation *Replicate Simulation*

Description

Replicate Simulation

Usage

```
replicate_simulation(sim_args, return_list = FALSE, future.seed = TRUE,
  ...)
```

Arguments

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`return_list` TRUE/FALSE indicating whether a full list output should be returned. If TRUE, the nested list is returned. If FALSE, replications are combined with a replication id appended.

`future.seed` TRUE/FALSE or numeric. Default value is true, see future_replicate.

`...` Currently not used.

`run_shiny` *Run Shiny Application Demo*

Description

Function runs Shiny Application Demo

Usage

```
run_shiny()
```

Details

This function does not take any arguments and will run the Shiny Application. If running from RStudio, will open the application in the viewer, otherwise will use the default internet browser.

`simglm` *simglm: A package to simulate and perform power by simulation for models based on the generalized linear model.*

Description

The simglm package provides two categories of important functions: simulation functions (`sim_reg` and `sim_glm`) and power functions (`sim_pow` and `sim_pow_glm`). #'

This function is most useful to pass to `replicate_simulation`. The function attempts to determine automatically which aspects to add to the simulation/power generation based on the elements found in the sim_args argument.

Usage

```
simglm(sim_args)
```

Arguments

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`simulate_error` *Tidy error simulation*

Description

Tidy error simulation

Usage

```
simulate_error(data, sim_args, ...)
```

Arguments

`data` Data simulated from other functions to pass to this function.

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`...` Other arguments to pass to error simulation functions.

`simulate_fixed` *Tidy fixed effect formula simulation*

Description

This function simulates the fixed portion of the model using a formula syntax.

Usage

```
simulate_fixed(data, sim_args, ...)
```

Arguments

`data` Data simulated from other functions to pass to this function. Can pass NULL if first in simulation string.

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`...` Other arguments to pass to error simulation functions.

simulate_heterogeneity *Tidy heterogeneity of variance simulation*

Description

This function simulates heterogeneity of level one error variance.

Usage

```
simulate_heterogeneity(data, sim_args, ...)
```

Arguments

`data` Data simulated from other functions to pass to this function. This function needs to be specified after ‘simulate_fixed‘ and ‘simulate_error‘.

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`...` Other arguments to pass to error simulation functions.

simulate_randomeffect *Tidy random effect formula simulation*

Description

This function simulates the random portion of the model using a formula syntax.

Usage

```
simulate_randomeffect(data, sim_args, ...)
```

Arguments

`data` Data simulated from other functions to pass to this function. Can pass NULL if first in simulation string.

`sim_args` A named list with special model formula syntax. See details and examples for more information. The named list may contain the following:

- fixed: This is the fixed portion of the model (i.e. covariates)
- random: This is the random portion of the model (i.e. random effects)
- error: This is the error (i.e. residual term).

`...` Other arguments to pass to error simulation functions.

sim_continuous *Simulate continuous variables*

Description

Function that simulates continuous variables. Any distribution function in R is supported.

Usage

```
sim_continuous(k = NULL, n, p, dist_fun, var_type = c("level1",
  "level2", "level3", "single"), ...)
```

Arguments

k	Number of third level clusters.
n	Number of clusters or number of observations for single level
p	Number of within cluster observations for multilevel
dist_fun	A distribution function. This argument takes a quoted R distribution function (e.g. 'rnorm').
var_type	Variable type for the variable, must be either "level1", "level2", "level3", or "single"
...	Additional parameters to pass to the dist_fun argument.

sim_continuous2 *Simulate continuous variables*

Description

Function that simulates continuous variables. Any distribution function in R is supported.

Usage

```
sim_continuous2(n, dist = "rnorm", var_level = 1, variance = NULL,
  ther_sim = FALSE, ther_val = NULL, ...)
```

Arguments

n	A list of sample sizes.
dist	A distribution function. This argument takes a quoted R distribution function (e.g. 'rnorm'). Default is 'rnorm'.
var_level	The level the variable should be simulated at. This can either be 1, 2, or 3 specifying a level 1, level 2, or level 3 variable respectively.
variance	The variance for random effect simulation.

`ther_sim` A TRUE/FALSE flag indicating whether the error simulation function should be simulated, that is should the mean and standard deviation used for standardization be simulated.

`ther_val` A vector of 2 that should include the theoretical mean and standard deviation of the generating function.

`...` Additional parameters to pass to the dist_fun argument.

`sim_err_nested` *Function that simulates errors.*

Description

Input error simulation parameters and outputs simulated errors.

Usage

```
sim_err_nested(error_var, n, p, with_err_gen, arima = FALSE,
  lvl1_err_params = NULL, arima_mod = list(NULL), ther = c(0, 1),
  ther_sim = FALSE, homogeneity = TRUE, fixef = NULL,
  heterogeneity_var = NULL, ...)
```

Arguments

`error_var` Scalar of error variance

`n` Cluster sample size.

`p` Within cluster sample size.

`with_err_gen` The generating function used as a character, (e.g. 'rnorm').

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`lvl1_err_params` Additional values that need to be passed to the function called from with_err_gen.

`arima_mod` A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`ther` A vector of length two that specifies the theoretical mean and standard deviation of the with_err_gen. This would commonly be used to standardize the generating variable to have a mean of 0 and standard deviation of 1 to meet model assumptions. The variable is then rescaled to have the variance specified by error_var.

`ther_sim` A TRUE/FALSE flag indicating whether the error simulation function should be simulated, that is should the mean and standard deviation used for standardization be simulated.

`homogeneity` Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`fixef` The design matrix, this is passed internally and used for heterogeneity of variance simulation.

`heterogeneity_var` Variable name as a character string to use for heterogeneity of variance simulation.

`...` Not currently used.

`sim_err_single` *Function that simulates errors.*

Description

Input error simulation parameters and outputs simulated errors.

Usage

```
sim_err_single(error_var, n, with_err_gen, arima = FALSE,
  lvl1_err_params = NULL, arima_mod = list(NULL), ther = c(0, 1),
  ther_sim = FALSE, homogeneity = TRUE, fixef = NULL,
  heterogeneity_var = NULL, ...)
```

Arguments

`error_var` Numeric scalar of error variance or vector used when simulating heterogeneity of variance.

`n` Cluster sample size.

`with_err_gen` The generating function used.

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`lvl1_err_params` Additional values that need to be passed to the function called from with_err_gen.

`arima_mod` A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`ther` A vector of length two that specifies the theoretical mean and standard deviation of the with_err_gen. This would commonly be used to standardize the generating variable to have a mean of 0 and standard deviation of 1 to meet model assumptions. The variable is then rescaled to have the variance specified by error_var.

`ther_sim` A TRUE/FALSE flag indicating whether the error simulation function should be simulated, that is should the mean and standard deviation used for standardization be simulated.

`homogeneity` Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`fixef` The design matrix, this is passed internally and used for heterogeneity of variance simulation.

`heterogeneity_var` Variable name as a character string to use for heterogeneity of variance simulation.

`...` Not currently used.

Details

Simulates error term for single level regression models.

`sim_factor` *Simulate categorical, factor, or discrete variables*

Description

Function that simulates discrete, factor, or categorical variables. Is essentially a wrapper around the sample function from base R.

Usage

```
sim_factor(k = NULL, n, p, numlevels, var_type = c("level1", "level2",
  "level3", "single"), ...)
```

Arguments

`k` Number of third level clusters.

`n` Number of clusters or number of observations for single level

`p` Number of within cluster observations for multilevel

`numlevels` Scalar indicating the number of levels for categorical, factor, or discrete variable

`var_type` Variable type for the variable, must be either "level1", "level2", "level3", or "single"

`...` Additional parameters passed to the sample function.

sim_factor2 *Simulate categorical, factor, or discrete variables*

Description

Function that simulates discrete, factor, or categorical variables. Is essentially a wrapper around the sample function from base R.

Usage

```
sim_factor2(n, levels, var_level = 1, replace = TRUE, ...)
```

Arguments

`n`	A list of sample sizes.
`levels`	Scalar indicating the number of levels for categorical, factor, or discrete variable. Can also specify levels as a character vector.
`var_level`	The level the variable should be simulated at. This can either be 1, 2, or 3 specifying a level 1, level 2, or level 3 variable respectively.
`replace`	TRUE/FALSE indicating whether levels should be sampled with replacement. Default is TRUE.
`...`	Additional parameters passed to the sample function.

sim_fixef_nested *Simulates design matrix.*

Description

Input fixed variables, sample size, and number of within variables, returns design matrix.

Usage

```
sim_fixef_nested(fixed, fixed_vars, cov_param, n, p, data_str,
  cor_vars = NULL, fact_vars = list(NULL), contrasts = NULL,
  knot_args = list(NULL))
```

Arguments

`fixed`	One sided formula for fixed effects in the simulation.
`fixed_vars`	Character vector of covariates for design matrix.
`cov_param`	List of arguments to pass to the continuous generating function. Required arguments include: • dist_fun: This is a quoted R distribution function.

- var_type: This is the level of variable to generate. Must be either ’level1’ or ’level2’. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples for example code for this. Does not include intercept, time, factors, or interactions.

`n` Number of clusters.

`p` Number of within cluster units.

`data_str` Type of data. Must be "cross", or "long".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels: Number of levels for ordinal or factor variables.
- var_type: Must be ’level1’ or ’level2’.

Optional arguments passed on to sample in a nested list. These include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

Details

Simulates the fixed effects for the `sim_reg` function when a linear mixed model is specified. This function assumes a time variable when longitudinal data is specified and does include any interactions that are specified.

`sim_fixef_nested3` *Simulates design matrix.*

Description

Input fixed variables, sample size, and number of within variables, returns design matrix.

Usage

```
sim_fixef_nested3(fixed, fixed_vars, cov_param, k, n, p, data_str,
  cor_vars = NULL, fact_vars = list(NULL), contrasts = NULL,
  knot_args = list(NULL))
```

Arguments

`fixed` One sided formula for fixed effects in the simulation.

`fixed_vars` Character vector of covariates for design matrix.

`cov_param` List of arguments. Required arguments are:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be either 'level1', 'level2', or 'level3'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples for example code for this. Does not include intercept, time, factors, or interactions.

`k` Number of third level clusters.

`n` Number of clusters.

`p` Number of within cluster units.

`data_str` Type of data. Must be "cross", or "long".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'level1', 'level2', or 'level3'.

Optional arguments passed on to sample in a nested list. These include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

Details

Simulates the fixed effects for the `sim_reg` function when a linear mixed model is specified. This function assumes a time variable when longitudinal data is specified and does include any interactions that are specified.

sim_fixef_single *Simulates design matrix for single level model.*

Description

Input fixed variables, sample size, and number of within variables, returns design matrix.

Usage

```
sim_fixef_single(fixed, fixed_vars, n, cov_param, cor_vars = NULL,
  fact_vars = list(NULL), contrasts = NULL, knot_args = list(NULL))
```

Arguments

`fixed` One sided formula for fixed effects in the simulation.

`fixed_vars` Character vector of covariates for design matrix.

`n` Number of clusters.

`cov_param` List of arguments to pass to the continuous generating function. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'single'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples for example code for this. Does not include intercept, time, factors, or interactions.

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'single'.

Optional arguments passed on to sample in a nested list. These include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

Details

Simulates the fixed effects for the sim_reg function when simulating a simple regression model.

sim_glm *Master generalized simulation function.*

Description

Takes simulation parameters as inputs and returns simulated data.

Usage

```
sim_glm(fixed, random, random3, fixed_param, random_param = list(),
  random_param3 = list(), cov_param, k, n, p, data_str,
  cor_vars = NULL, fact_vars = list(NULL), unbal = list(level2 =
  FALSE, level3 = FALSE), unbal_design = list(level2 = NULL, level3 =
  NULL), contrasts = NULL, outcome_type, cross_class_params = NULL,
  knot_args = list(NULL), ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`random3` One sided formula for random effects at third level in the simulation. Must be a subset of fixed (and likely of random).

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`random_param3` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,

- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be either ’single’, ’level1’, ’level2’, or ’level3’. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`k` Number of third level clusters.

`n` Cluster sample size.

`p` Within cluster sample size.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be ’single’, ’level1’, ’level2’, or ’level3’.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`cross_class_params`

A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`...` Not currently used.

Details

Simulated data is useful for classroom demonstrations and to study the impacts of assumption violations on parameter estimates, statistical power, or empirical type I error rates.

This function allows researchers a flexible approach to simulate regression models, including single level models and cross sectional or longitudinal linear mixed models (aka. hierarchical linear models or multilevel models).

Examples

```
# generating parameters for single level regression
set.seed(2)
fixed <- ~1 + act + diff + numCourse + act:numCourse
fixed_param <- c(0.1, -0.2, 0.15, 0.5, -0.02)
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
   var_type = c("single", "single", "single"),
   opts = list(list(mean = 0, sd = 4),
   list(mean = 0, sd = 3),
   list(mean = 0, sd = 3)))
n <- 150
temp_single <- sim_glm(fixed = fixed, fixed_param = fixed_param,
  cov_param = cov_param, n = n, data_str = "single", outcome_type = 'logistic')

  # counts
temp_single <- sim_glm(fixed = fixed, fixed_param = fixed_param,
  cov_param = cov_param, n = n, data_str = "single", outcome_type = 'poisson')

# Longitudinal linear mixed model example
fixed <- ~1 + time + diff + act + time:act
random <- ~1 + time + diff
```

```
fixed_param <- c(0.1, -0.2, 0.15, 0.5, -0.02)
random_param <- list(random_var = c(7, 4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm'),
   var_type = c(”level1”, ”level2”),
   opts = list(list(mean = 0, sd = 1.5),
   list(mean = 0, sd = 4)))
n <- 150
p <- 30
data_str <- ”long”
temp_long <- sim_glm(fixed, random, random3 = NULL, fixed_param,
random_param, random_param3 = NULL,
 cov_param, k = NULL, n, p, data_str = data_str, outcome_type = 'logistic')

 # counts
temp_long <- sim_glm(fixed, random, random3 = NULL, fixed_param,
random_param, random_param3 = NULL,
 cov_param, k = NULL, n, p, data_str = data_str, outcome_type = 'poisson')

# Three level example
fixed <- ~1 + time + diff + act + actClust + time:act
random <- ~1 + time + diff
random3 <- ~ 1 + time
fixed_param <- c(0.1, -0.2, 0.15, 0.5, -0.02, 0.03)
random_param <- list(random_var = c(7, 4, 2), rand_gen = 'rnorm')
random_param3 <- list(random_var = c(4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
   var_type = c(”level1”, ”level2”, ”level3”),
   opts = list(list(mean = 0, sd = 1.5),
   list(mean = 0, sd = 4),
   list(mean = 0, sd = 2)))
k <- 10
n <- 15
p <- 10
data_str <- ”long”
temp_three <- sim_glm(fixed, random, random3, fixed_param, random_param,
  random_param3, cov_param, k,n, p, data_str = data_str, outcome_type = 'logistic')

  # count data sim
  temp_three <- sim_glm(fixed, random, random3, fixed_param, random_param,
  random_param3, cov_param, k,n, p, data_str = data_str, outcome_type = 'poisson')
```

`sim_glm_nested` *Simulate two level logistic regression model*

Description

Takes simulation parameters as inputs and returns simulated data.

Usage

```
sim_glm_nested(fixed, random, fixed_param, random_param = list(),
  cov_param, n, p, data_str, cor_vars = NULL, fact_vars = list(NULL),
  unbal = FALSE, unbal_design = NULL, contrasts = NULL, outcome_type,
  cross_class_params = NULL, knot_args = list(NULL), ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'level1' or 'level2'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`n` Cluster sample size.

`p` Within cluster sample size.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'level1' or 'level2'.

Optional arguments include:

- replace
- prob
- value.labels

See also sample for use of these optional arguments.

unbal
: A vector of sample sizes for the number of observations for each level 2 cluster. Must have same length as level two sample size n. Alternative specification can be TRUE, which uses additional argument, unbal_design.

unbal_design
: When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two sample size.

contrasts
: An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See contrasts for more detail.

outcome_type
: A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

cross_class_params
: A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to sim_rand_eff. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

knot_args
: A nested list of named knot arguments. See sim_knot for more details. Arguments must include:

- var
- knot_locations

...
: Not currently used.

Details

Simulates data for the nested logistic regression models. Returns a data frame with ID variables, fixed effects, random effects, and many other variables to help when running simulation studies.

Examples

```
# Longitudinal linear mixed model example
fixed <- ~1 + time + diff + act + time:act
random <- ~1 + time + diff
```

```
fixed_param <- c(0.1, -0.2, 0.15, 0.5, -0.02)
random_param <- list(random_var = c(7, 4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm'),
   var_type = c(”level1”, ”level2”),
   opts = list(list(mean = 0, sd = 1.5),
   list(mean = 0, sd = 4)))
n <- 150
p <- 30
data_str <- ”long”
temp_long <- sim_glm(fixed, random, random3 = NULL, fixed_param,
random_param, random_param3 = NULL,
 cov_param, k = NULL, n, p, data_str = data_str, outcome_type = 'logistic')
```

`sim_glm_nested3` *Function to simulate three level nested data*

Description

Takes simulation parameters as inputs and returns simulated data.

Usage

```
sim_glm_nested3(fixed, random, random3, fixed_param,
  random_param = list(), random_param3 = list(), cov_param, k, n, p,
  data_str, cor_vars = NULL, fact_vars = list(NULL),
  unbal = list(level2 = FALSE, level3 = FALSE),
  unbal_design = list(level2 = NULL, level3 = NULL), contrasts = NULL,
  outcome_type, cross_class_params = NULL, knot_args = list(NULL), ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`random3` One sided formula for random effects at third level in the simulation. Must be a subset of fixed (and likely of random).

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,

- ...: Additional parameters needed for rand_gen function.

`random_param3` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'level1', 'level2', or 'level3'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`k` Number of third level clusters.

`n` Level two sample size within each level three cluster.

`p` Within cluster sample size within each level two cluster.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'single', 'level1', 'level2', or 'level3'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`cross_class_params`

A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`...` Not currently used.

Details

Simulates data for the linear mixed model, both cross sectional and longitudinal data. Returns a data frame with ID variables, fixed effects, and many other variables useful to help when running simulation studies.

See Also

`sim_reg` for a convenient wrapper for all data conditions.

Examples

```
# Three level example
fixed <- ~1 + time + diff + act + actClust + time:act
random <- ~1 + time + diff
random3 <- ~ 1 + time
fixed_param <- c(0.1, -0.2, 0.15, 0.5, -0.02, 0.04)
random_param <- list(random_var = c(7, 4, 2), rand_gen = 'rnorm')
random_param3 <- list(random_var = c(4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
   var_type = c("level1", "level2", "level3"),
   opts = list(list(mean = 0, sd = 1.5),
    list(mean = 0, sd = 4),
    list(mean = 0, sd = 2)))
k <- 10
n <- 15
p <- 10
data_str <- "long"
temp_three <- sim_glm(fixed, random, random3, fixed_param, random_param,
  random_param3, cov_param, k,n, p, data_str = data_str,
  outcome_type = 'logistic')
```

sim_glm_single *Simulation single level logistic regression model*

Description

Takes simulation parameters as inputs and returns simulated data.

Usage

```
sim_glm_single(fixed, fixed_param, cov_param, n, data_str,
  cor_vars = NULL, fact_vars = list(NULL), contrasts = NULL,
  outcome_type, knot_args = list(NULL), ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'single'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`n` Cluster sample size.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'single', 'lvl1', 'lvl2', or 'lvl3'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`...` Not currently used.

Details

Simulates data for the simple logistic regression models. Returns a data frame with ID variables, fixed effects, and many other variables to help when running simulation studies.

Examples

```
# generating parameters for single level regression
set.seed(2)
fixed <- ~1 + act + diff + numCourse + act:numCourse
fixed_param <- c(0.1, -0.2, 0.15, 0.5, -0.02)
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
   var_type = c("single", "single", "single"),
   opts = list(list(mean = 0, sd = 4),
   list(mean = 0, sd = 3),
   list(mean = 0, sd = 3)))
n <- 150
temp_single <- sim_glm(fixed = fixed, fixed_param = fixed_param,
  cov_param = cov_param, n = n, data_str = "single",
  outcome_type = 'logistic')
```

`sim_knot` *Simulate knot locations*

Description

Function that generates knot locations. An example of usefulness of this funciton would be with generation of interrupted time series data. Another application may be with simulation of piecewise linear data structures.

Usage

```
sim_knot(var, knot_locations, right = FALSE)
```

Arguments

`var`	Variable used to create knots in the data.
`knot_locations`	The locations to create knots. These need to be specified with the scale of the variable in mind. See examples.
`right`	logical, indicating if the intervals should be closed on the right (and open on the left) or vice versa. See `cut` for more details. Defaults to FALSE, which is likely most desirable behavior in this context.

Examples

```
sim_knot(0:10, knot_locations = c(4, 9))
sim_knot(rnorm(100), knot_locations = c(-1, 1.5))
sim_knot(0:8, knot_locations = 5)
sim_knot(0:8, knot_locations = 5, right = TRUE)
```

`sim_pow` *Master power simulation function.*

Description

Input simulation conditions, returns power for term.

Usage

```
sim_pow(fixed, random = NULL, random3 = NULL, fixed_param,
  random_param = list(NULL), random_param3 = list(NULL), cov_param,
  k = NULL, n, p = NULL, error_var, with_err_gen, arima = FALSE,
  data_str, cor_vars = NULL, fact_vars = list(NULL),
  unbal = list(level2 = FALSE, level3 = FALSE),
  unbal_design = list(level2 = NULL, level3 = NULL),
  lvl1_err_params = NULL, arima_mod = list(NULL), contrasts = NULL,
  homogeneity = TRUE, heterogeneity_var = NULL,
  cross_class_params = NULL, knot_args = list(NULL), missing = FALSE,
  missing_args = list(NULL), pow_param, alpha, pow_dist = c("z", "t"),
  pow_tail = c(1, 2), replicates, terms_vary = NULL,
  raw_power = TRUE, lm_fit_mod = NULL, lme4_fit_mod = NULL,
  nlme_fit_mod = NULL, arima_fit_mod = NULL, general_mod = NULL,
  general_extract = NULL, ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`random3` One sided formula for random effects at third level in the simulation. Must be a subset of fixed (and likely of random).

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`random_param3` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be either 'single', 'level1', 'level2', or 'level3'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`k` Number of third level clusters.

`n` Cluster sample size.

`p` Within cluster sample size.

`error_var` Scalar of error variance.

`with_err_gen` Distribution function to pass on to the level one simulation of errors.

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'single', 'level1', 'level2', or 'level3'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`lvl1_err_params`
Additional parameters passed as a list on to the level one error generating function

`arima_mod` A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`homogeneity` Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`heterogeneity_var`
Variable name as a character string to use for heterogeneity of variance simulation.

`cross_class_params`
A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect

  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:

  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`missing` TRUE/FALSE flag indicating whether missing data should be simulated.

`missing_args` Additional missing arguments to pass to the missing_data function. See `missing_data` for examples.

`pow_param` Number of parameter to calculate power includes intercept where applicable.

`alpha` What should the per test alpha rate be used for the hypothesis testing.

`pow_dist` Which distribution should be used when testing hypothesis test, z or t?

`pow_tail` One-tailed or two-tailed test?

`replicates` How many replications should be done (i.e. the denominator in power calculation).

`terms_vary` A named list of terms that should vary as a function for the power simulation. The names must match arguments to the simulation function, see `sim_reg` for examples. Values specified here should not be included as arguments in the function call.

`raw_power` TRUE/FALSE indicating whether raw power output should be returned. Default is TRUE, which will create a new nested column with raw data by variable(s) manipulated in power analysis.

`lm_fit_mod` Valid lm syntax to be used for model fitting.

`lme4_fit_mod` Valid lme4 syntax to be used for model fitting.

`nlme_fit_mod` Valid nlme syntax to be used for model fitting. This should be specified as a named list with fixed and random components.

`arima_fit_mod` Valid nlme syntax for fitting serial correlation structures. See `corStruct` for help. This must be specified to include serial correlation.

`general_mod` Valid model syntax. This syntax can be from any R package. By default, broom is used to extract model result information. Note, package must be defined or loaded prior to running the sim_pow function.

`general_extract`

A valid function to extract model results if general_mod argument is used. This argument is primarily used if extracting model results is not possibly using the broom package. If this is left NULL (default), broom is used to collect model results.

`...` Currently not used.

Details

This function is a wrapper that replicates the simulation functions for simple regression and the linear mixed model power functions. This function replicates the power call a specified number of times and prints outs a matrix with the results.

Examples

```
# single level example
fixed <- ~ 1 + act + diff + numCourse + act:numCourse
fixed_param <- c(0.5, 1.1, 0.6, 0.9, 1.1)
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
                  var_type = c(”single”, ”single”, ”single”),
                  opts = list(list(mean = 0, sd = 2),
                              list(mean = 0, sd = 2),
                              list(mean = 0, sd = 1)))
n <- 150
error_var <- 20
with_err_gen <- 'rnorm'
pow_param <- c('(Intercept)', 'act', 'diff', 'numCourse')
alpha <- .01
pow_dist <- ”t”
pow_tail <- 2
replicates <- 2
power_out <- sim_pow(fixed = fixed, fixed_param = fixed_param, cov_param = cov_param,
                     n = n, error_var = error_var, with_err_gen = with_err_gen,
                     data_str = ”single”, pow_param = pow_param, alpha = alpha,
                     pow_dist = pow_dist, pow_tail = pow_tail,
                     replicates = replicates, raw_power = FALSE)

# Vary terms example
fixed <- ~ 1 + act + diff + numCourse + act:numCourse
fixed_param <- c(0.5, 1.1, 0.6, 0.9, 1.1)
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
                  var_type = c(”single”, ”single”, ”single”),
                  opts = list(list(mean = 0, sd = 2),
                              list(mean = 0, sd = 2),
                              list(mean = 0, sd = 1)))
n <- NULL
error_var <- NULL
with_err_gen <- 'rnorm'
pow_param <- c('(Intercept)', 'act', 'diff', 'numCourse')
alpha <- .01
pow_dist <- ”t”
pow_tail <- 2
replicates <- 1
terms_vary <- list(n = c(20, 40, 60, 80, 100), error_var = c(5, 10, 20))
power_out <- sim_pow(fixed = fixed, fixed_param = fixed_param, cov_param = cov_param,
                     n = n, error_var = error_var, with_err_gen = with_err_gen,
                     data_str = ”single”, pow_param = pow_param, alpha = alpha,
                     pow_dist = pow_dist, pow_tail = pow_tail,
                     replicates = replicates, terms_vary = terms_vary,
                     raw_power = FALSE)


# Three level example
fixed <- ~1 + time + diff + act + actClust + time:act
random <- ~1 + time
random3 <- ~ 1 + time
fixed_param <- c(4, 2, 6, 2.3, 7, 0)
random_param <- list(random_var = c(7, 4), rand_gen = 'rnorm')
random_param3 <- list(random_var = c(4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
                  var_type = c("level1", "level2", "level3"),
                  opts = list(list(mean = 0, sd = 1.5),
                              list(mean = 0, sd = 4),
                              list(mean = 0, sd = 2)))
k <- 10
n <- 15
p <- 5
error_var <- 4
with_err_gen <- 'rnorm'
data_str <- "long"
pow_param <- c('time', 'diff', 'act', 'actClust')
alpha <- .01
pow_dist <- "z"
pow_tail <- 2
replicates <- 1
power_out <- sim_pow(fixed = fixed, random = random, random3 = random3,
                     fixed_param = fixed_param,
                     random_param = random_param,
                     random_param3 = random_param3,
                     cov_param = cov_param,
                     k = k, n = n, p = p,
                     error_var = error_var, with_err_gen = "rnorm",
                     data_str = data_str,
                     unbal = list(level3 = FALSE, level2 = FALSE),
                     pow_param = pow_param, alpha = alpha,
                     pow_dist = pow_dist, pow_tail = pow_tail,
                     replicates = replicates, raw_power = FALSE)
```

sim_pow_glm *Master power simulation function for glm models.*

Description

Input simulation conditions, returns power for term.

Usage

```
sim_pow_glm(fixed, random = NULL, random3 = NULL, fixed_param,
random_param = list(NULL), random_param3 = list(NULL), cov_param,
k = NULL, n, p = NULL, data_str, cor_vars = NULL,
fact_vars = list(NULL), unbal = list(level2 = FALSE, level3 = FALSE),
unbal_design = list(level2 = NULL, level3 = NULL), contrasts = NULL,
outcome_type, cross_class_params = NULL, knot_args = list(NULL),
missing = FALSE, missing_args = list(NULL), pow_param, alpha,
pow_dist = c(”z”, ”t”), pow_tail = c(1, 2), replicates,
terms_vary = NULL, raw_power = TRUE, glm_fit_mod = NULL,
lme4_fit_mod = NULL, glm_fit_family = NULL, lme4_fit_family = NULL,
general_mod = NULL, general_extract = NULL, ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`random3` One sided formula for random effects at third level in the simulation. Must be a subset of fixed(and likely of random).

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`random_param3` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be either 'single', 'level1', 'level2', or 'level3'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`k` Number of third level clusters.

`n` Cluster sample size.

`p` Within cluster sample size.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'single', 'level1', 'level2', or 'level3'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`cross_class_params`

A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,

  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`missing` TRUE/FALSE flag indicating whether missing data should be simulated.

`missing_args` Additional missing arguments to pass to the missing_data function. See `missing_data` for examples.

`pow_param` Number of parameter to calculate power includes intercept where applicable.

`alpha` What should the per test alpha rate be used for the hypothesis testing.

`pow_dist` Which distribution should be used when testing hypothesis test, z or t?

`pow_tail` One-tailed or two-tailed test?

`replicates` How many replications should be done (i.e. the denominator in power calculation).

`terms_vary` A named list of terms that should vary as a function for the power simulation. The names must match arguments to the simulation function, see `sim_glm` for examples. Values specified here should not be included as arguments in the function call.

`raw_power` TRUE/FALSE indicating whether raw power output should be returned. Default is TRUE, which will create a new nested column with raw data by variable(s) manipulated in power analysis.

`glm_fit_mod` Valid glm syntax to be used for model fitting.

`lme4_fit_mod` Valid lme4 syntax to be used for model fitting.

`glm_fit_family` Valid family syntax to pass to the glm function.

`lme4_fit_family`
Valid lme4 family specification passed to glmer.

`general_mod` Valid model syntax. This syntax can be from any R package. By default, broom is used to extract model result information. Note, package must be defined or loaded prior to running the sim_pow function.

`general_extract`
A valid function to extract model results if general_mod argument is used. This argument is primarily used if extracting model results is not possibly using the broom package. If this is left NULL (default), broom is used to collect model results.

`...` Current not used.

Details

This function is a wrapper that replicates the simulation functions for simple generalized regression and the generalized linear mixed model power functions. This function replicates the power call a specified number of times and prints outs a matrix with the results.

Examples

```
# single level dichotomous (glm) example
fixed <- ~ 1 + act + diff
fixed_param <- c(0.1, 0.5, 0.3)
cov_param <- list(dist_fun = c('rnorm', 'rnorm'),
                  var_type = c("single", "single"),
                  opts = list(list(mean = 0, sd = 2),
                              list(mean = 0, sd = 4)))
n <- 50
pow_param <- c('(Intercept)', 'act', 'diff')
alpha <- .01
pow_dist <- "z"
pow_tail <- 2
replicates <- 2

power_out <- sim_pow_glm(fixed = fixed, fixed_param = fixed_param,
                         cov_param = cov_param,
                         n = n, data_str = "single",
                         outcome_type = 'logistic',
                         pow_param = pow_param, alpha = alpha,
                         pow_dist = pow_dist, pow_tail = pow_tail,
                         replicates = replicates, raw_power = FALSE)
```

---

sim_pow_glm_nested *Power simulation for nested designs*

Description

Takes simulation conditions as input, exports power.

Usage

```
sim_pow_glm_nested(fixed, random, fixed_param, random_param = list(),
  cov_param, n, p, data_str, cor_vars = NULL, fact_vars = list(NULL),
  unbal = list(level2 = FALSE, level3 = FALSE),
  unbal_design = list(level2 = NULL, level3 = NULL), contrasts = NULL,
  outcome_type, cross_class_params = NULL, knot_args = list(NULL),
  missing = FALSE, missing_args = list(NULL), pow_param = NULL,
  alpha, pow_dist = c("z", "t"), pow_tail = c(1, 2),
  lme4_fit_mod = NULL, lme4_fit_family, general_mod = NULL,
  general_extract = NULL, ...)
```

Arguments

`fixed`	One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be ’level1’ or ’level2’. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`n` Cluster sample size.

`p` Within cluster sample size.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels: Number of levels for ordinal or factor variables.
- var_type: Must be ’level1’ or ’level2’.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`cross_class_params`

A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`missing` TRUE/FALSE flag indicating whether missing data should be simulated.

`missing_args` Additional missing arguments to pass to the missing_data function. See `missing_data` for examples.

`pow_param` Name of variable to calculate power for, must be a name from fixed.

`alpha` What should the per test alpha rate be used for the hypothesis testing.

`pow_dist` Which distribution should be used when testing hypothesis test, z or t?

`pow_tail` One-tailed or two-tailed test?

`lme4_fit_mod` Valid lme4 formula syntax to be used for model fitting.

`lme4_fit_family`

Valid lme4 family specification passed to glmer.

`general_mod` Valid model syntax. This syntax can be from any R package. By default, broom is used to extract model result information. Note, package must be defined or loaded prior to running the sim_pow function.

`general_extract`

A valid function to extract model results if general_mod argument is used. This argument is primarily used if extracting model results is not possibly using the broom package. If this is left NULL (default), broom is used to collect model results.

`...` Not currently used.

Details

Power function to compute power for a regression term for the generalized linear mixed model. This function would need to be replicated to make any statement about power. Use `sim_pow_glm` as a convenient wrapper for this.

See Also

`sim_pow_glm` for a wrapper to replicate.

`sim_pow_glm_nested3` *Power simulation for nested designs*

Description

Takes simulation conditions as input, exports power.

Usage

```
sim_pow_glm_nested3(fixed, random, random3, fixed_param,
  random_param = list(), random_param3 = list(), cov_param, k, n, p,
  data_str, cor_vars = NULL, fact_vars = list(NULL),
  unbal = list(level2 = FALSE, level3 = FALSE),
  unbal_design = list(level2 = NULL, level3 = NULL), contrasts = NULL,
  outcome_type, cross_class_params = NULL, knot_args = list(NULL),
  missing = FALSE, missing_args = list(NULL), pow_param = NULL,
  alpha, pow_dist = c("z", "t"), pow_tail = c(1, 2),
  lme4_fit_mod = NULL, lme4_fit_family, general_mod = NULL,
  general_extract = NULL, ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`random3` One sided formula for random effects at third level in the simulation. Must be a subset of fixed (and likely of random).

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,

- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`random_param3` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be ’level1’, ’level2’, or ’level3’. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`k` Number of third level clusters.

`n` Cluster sample size.

`p` Within cluster sample size.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be ’level1’, ’level2’, or ’level3’.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`cross_class_params`

A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`missing` TRUE/FALSE flag indicating whether missing data should be simulated.

`missing_args` Additional missing arguments to pass to the missing_data function. See `missing_data` for examples.

`pow_param` Name of variable to calculate power for, must be a name from fixed.

`alpha` What should the per test alpha rate be used for the hypothesis testing.

`pow_dist` Which distribution should be used when testing hypothesis test, z or t?

`pow_tail` One-tailed or two-tailed test?

`lme4_fit_mod` Valid lme4 formula syntax to be used for model fitting.

`lme4_fit_family`

Valid lme4 family specification passed to glmer.

`general_mod` Valid model syntax. This syntax can be from any R package. By default, broom is used to extract model result information. Note, package must be defined or loaded prior to running the sim_pow function.

`general_extract`

A valid function to extract model results if general_mod argument is used. This argument is primarily used if extracting model results is not possibly using the broom package. If this is left NULL (default), broom is used to collect model results.

`...` Not currently used.

Details

Power function to compute power for a regression term for the generalized linear mixed model. This function would need to be replicated to make any statement about power. Use `sim_pow_glm` as a convenient wrapper for this.

See Also

`sim_pow_glm` for a wrapper to replicate.

`sim_pow_glm_single` *Function to simulate power.*

Description

Input simulation conditions and which term to compute power for, export reported power.

Usage

```
sim_pow_glm_single(fixed, fixed_param, cov_param, n, data_str,
  cor_vars = NULL, fact_vars = list(NULL), contrasts = NULL,
  outcome_type, knot_args = list(NULL), missing = FALSE,
  missing_args = list(NULL), pow_param = NULL, alpha,
  pow_dist = c("z", "t"), pow_tail = c(1, 2), glm_fit_mod = NULL,
  glm_fit_family, general_mod = NULL, general_extract = NULL, ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'single'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`n` Cluster sample size.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels: Number of levels for ordinal or factor variables.

- var_type: Must be 'single'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`outcome_type` A vector specifying the type of outcome, must be either logistic or poisson. Logitstic outcome will be 0/1 and poisson outcome will be counts.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`missing` TRUE/FALSE flag indicating whether missing data should be simulated.

`missing_args` Additional missing arguments to pass to the missing_data function. See `missing_data` for examples.

`pow_param` Name of variable to calculate power for, must be a name from fixed.

`alpha` What should the per test alpha rate be used for the hypothesis testing.

`pow_dist` Which distribution should be used when testing hypothesis test, z or t?

`pow_tail` One-tailed or two-tailed test?

`glm_fit_mod` Valid glm syntax to be used for model fitting.

`glm_fit_family` Valid family syntax to pass to the glm function.

`general_mod` Valid model syntax. This syntax can be from any R package. By default, broom is used to extract model result information. Note, package must be defined or loaded prior to running the sim_pow function.

`general_extract`

A valid function to extract model results if general_mod argument is used. This argument is primarily used if extracting model results is not possibly using the broom package. If this is left NULL (default), broom is used to collect model results.

`...` Additional specification needed to pass to the random generating function defined by with_err_gen.

Details

Power function to compute power for a regression term for simple generalized regression models. This function would need to be replicated to make any statement about power. Use `sim_pow_glm` as a convenient wrapper for this.

See Also

`sim_pow_glm` for a wrapper to replicate.

`sim_pow_nested` *Power simulation for nested designs*

Description

Takes simulation conditions as input, exports power.

Usage

```
sim_pow_nested(fixed, random, fixed_param, random_param = list(),
  cov_param, n, p, error_var, with_err_gen, arima = FALSE, data_str,
  cor_vars = NULL, fact_vars = list(NULL), unbal = FALSE,
  unbal_design = NULL, lvl1_err_params = NULL,
  arima_mod = list(NULL), contrasts = NULL, homogeneity = TRUE,
  heterogeneity_var = NULL, cross_class_params = NULL,
  knot_args = list(NULL), missing = FALSE, missing_args = list(NULL),
  pow_param = NULL, alpha, pow_dist = c("z", "t"), pow_tail = c(1,
  2), lme4_fit_mod = NULL, nlme_fit_mod = NULL, arima_fit_mod = NULL,
  general_mod = NULL, general_extract = NULL, ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'level1' or 'level2'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

n Cluster sample size.

p Within cluster sample size.

error_var Scalar of error variance.

with_err_gen Simulated within cluster error distribution. Must be a quoted 'r' distribution function.

arima TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See arima.sim for examples.

data_str Type of data. Must be "cross", "long", or "single".

cor_vars A vector of correlations between variables.

fact_vars A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels: Number of levels for ordinal or factor variables.
- var_type: Must be 'level1' or 'level2'.

Optional arguments include:

- replace
- prob
- value.labels

See also sample for use of these optional arguments.

unbal A vector of sample sizes for the number of observations for each level 2 cluster. Must have same length as level two sample size n. Alternative specification can be TRUE, which uses additional argument, unbal_design.

unbal_design When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two sample size.

lvl1_err_params

Additional parameters passed as a list on to the level one error generating function

arima_mod A list indicating the ARIMA model to pass to arima.sim. See arima.sim for examples.

contrasts An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See contrasts for more detail.

homogeneity Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

heterogeneity_var

Variable name as a character string to use for heterogeneity of variance simulation.

cross_class_params

A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`missing` TRUE/FALSE flag indicating whether missing data should be simulated.

`missing_args` Additional missing arguments to pass to the missing_data function. See `missing_data` for examples.

`pow_param` Name of variable to calculate power for, must be a name from fixed.

`alpha` What should the per test alpha rate be used for the hypothesis testing.

`pow_dist` Which distribution should be used when testing hypothesis test, z or t?

`pow_tail` One-tailed or two-tailed test?

`lme4_fit_mod` Valid lme4 syntax to be used for model fitting.

`nlme_fit_mod` Valid nlme syntax to be used for model fitting. This should be specified as a named list with fixed and random components.

`arima_fit_mod` Valid nlme syntax for fitting serial correlation structures. See `corStruct` for help. This must be specified to include serial correlation.

`general_mod` Valid model syntax. This syntax can be from any R package. By default, broom is used to extract model result information. Note, package must be defined or loaded prior to running the sim_pow function.

`general_extract`

A valid function to extract model results if general_mod argument is used. This argument is primarily used if extracting model results is not possibly using the broom package. If this is left NULL (default), broom is used to collect model results.

`...` Not currently used.

Details

Power function to compute power for a regression term for the linear mixed model. This function would need to be replicated to make any statement about power. Use `sim_pow` as a convenient wrapper for this.

See Also

sim_pow for a wrapper to replicate.

sim_pow_nested3 *Power simulation for nested designs*

Description

Takes simulation conditions as input, exports power.

Usage

```
sim_pow_nested3(fixed, random, random3, fixed_param,
  random_param = list(), random_param3 = list(), cov_param, k, n, p,
  error_var, with_err_gen, arima = FALSE, data_str, cor_vars = NULL,
  fact_vars = list(NULL), unbal = list(level2 = FALSE, level3 = FALSE),
  unbal_design = list(level2 = NULL, level3 = NULL),
  lvl1_err_params = NULL, arima_mod = list(NULL), contrasts = NULL,
  homogeneity = TRUE, heterogeneity_var = NULL,
  cross_class_params = NULL, knot_args = list(NULL), missing = FALSE,
  missing_args = list(NULL), pow_param = NULL, alpha,
  pow_dist = c("z", "t"), pow_tail = c(1, 2), lme4_fit_mod = NULL,
  nlme_fit_mod = NULL, arima_fit_mod = NULL, general_mod = NULL,
  general_extract = NULL, ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`random3` One sided formula for random effects at third level in the simulation. Must be a subset of fixed (and likely of random).

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`random_param3` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'level1', 'level2', or 'level3'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`k` Number of third level clusters.

`n` Cluster sample size.

`p` Within cluster sample size.

`error_var` Scalar of error variance.

`with_err_gen` Simulated within cluster error distribution. Must be a quoted 'r' distribution function.

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels: Number of levels for ordinal or factor variables.
- var_type: Must be 'level1', 'level2', or 'level3'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`lvl1_err_params`
Additional parameters passed as a list on to the level one error generating function

`arima_mod` A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`homogeneity` Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`heterogeneity_var`
Variable name as a character string to use for heterogeneity of variance simulation.

`cross_class_params`
A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`missing` TRUE/FALSE flag indicating whether missing data should be simulated.

`missing_args` Additional missing arguments to pass to the missing_data function. See `missing_data` for examples.

`pow_param` Name of variable to calculate power for, must be a name from fixed.

`alpha` What should the per test alpha rate be used for the hypothesis testing.

`pow_dist` Which distribution should be used when testing hypothesis test, z or t?

`pow_tail` One-tailed or two-tailed test?

`lme4_fit_mod` Valid lme4 syntax to be used for model fitting.

`nlme_fit_mod` Valid nlme syntax to be used for model fitting. This should be specified as a named list with fixed and random components.

`arima_fit_mod` Valid nlme syntax for fitting serial correlation structures. See `corStruct` for help. This must be specified to include serial correlation.

`general_mod` Valid model syntax. This syntax can be from any R package. By default, broom is used to extract model result information. Note, package must be defined or loaded prior to running the sim_pow function.

`general_extract`
A valid function to extract model results if general_mod argument is used. This argument is primarily used if extracting model results is not possibly using the broom package. If this is left NULL (default), broom is used to collect model results.

`...` Not currently used.

Details

Power function to compute power for a regression term for the linear mixed model. This function would need to be replicated to make any statement about power. Use `sim_pow` as a convenient wrapper for this.

See Also

`sim_pow` for a wrapper to replicate.

`sim_pow_single` *Function to simulate power.*

Description

Input simulation conditions and which term to compute power for, export reported power.

Usage

```
sim_pow_single(fixed, fixed_param, cov_param, n, error_var, with_err_gen,
  arima = FALSE, data_str, cor_vars = NULL, fact_vars = list(NULL),
  lvl1_err_params = NULL, arima_mod = list(NULL), contrasts = NULL,
  homogeneity = TRUE, heterogeneity_var = NULL,
  knot_args = list(NULL), missing = FALSE, missing_args = list(NULL),
  pow_param = NULL, alpha, pow_dist = c("z", "t"), pow_tail = c(1,
  2), lm_fit_mod = NULL, general_mod = NULL, general_extract = NULL,
  ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'single'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`n` Cluster sample size.

`error_var` Scalar of error variance.

`with_err_gen` Simulated within cluster error distribution. Must be a quoted 'r' distribution function.

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels: Number of levels for ordinal or factor variables.
- var_type: Must be 'single'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`lvl1_err_params` Additional parameters passed as a list on to the level one error generating function

`arima_mod` A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`homogeneity` Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`heterogeneity_var` Variable name as a character string to use for heterogeneity of variance simulation.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`missing` TRUE/FALSE flag indicating whether missing data should be simulated.

`missing_args` Additional missing arguments to pass to the missing_data function. See `missing_data` for examples.

`pow_param` Name of variable to calculate power for, must be a name from fixed.

`alpha` What should the per test alpha rate be used for the hypothesis testing.

`pow_dist` Which distribution should be used when testing hypothesis test, z or t?

`pow_tail` One-tailed or two-tailed test?

`lm_fit_mod` Valid lm syntax to be used for model fitting.

`general_mod` Valid model syntax. This syntax can be from any R package. By default, broom is used to extract model result information. Note, package must be defined or loaded prior to running the sim_pow function.

`general_extract`
A valid function to extract model results if general_mod argument is used. This argument is primarily used if extracting model results is not possibly using the broom package. If this is left NULL (default), broom is used to collect model results.

`...` Additional specification needed to pass to the random generating function defined by with_err_gen.

Details

Power function to compute power for a regression term for simple regression models. This function would need to be replicated to make any statement about power. Use `sim_pow` as a convenient wrapper for this.

See Also

`sim_pow` for a wrapper to replicate.

`sim_rand_eff` *Function to simulate random effects.*

Description

Input simulation parameters and returns random effects.

Usage

```
sim_rand_eff(random_var, n, rand_gen, ther = c(0, 1), ther_sim = FALSE,
  cor_vars = NULL, ...)
```

Arguments

`random_var` Variance of random effects. Must be same length as random.

`n` Cluster sample size.

`rand_gen` The generating function used (e.g. rnorm).

`ther` A vector of length two that specifies the theoretical mean and standard deviation of the rand_gen. This would commonly be used to standardize the generating variable to have a mean of 0 and standard deviation of 1 to meet model assumptions. The variable is then rescaled to have the variance specified by random_var.

`ther_sim` A TRUE/FALSE flag indicating whether the error simulation function should be simulated, that is should the mean and standard deviation used for standardization be simulated.

`cor_vars` A vector of correlations between random effects.

`...` Additional values that need to be passed to the function called from rand_gen.

Details

Simulates random effects for the master function `sim_reg` when simulating a linear mixed model, both cross sectional and longitudinal. Allows the ability to simulate random effects from a Laplace, chi-square (1), mixture normal, or normal distribution.

`sim_reg` *Master continuous simulation function.*

Description

Takes simulation parameters as inputs and returns simulated data.

Usage

```
sim_reg(fixed, random, random3, fixed_param, random_param = list(),
  random_param3 = list(), cov_param, k, n, p, error_var, with_err_gen,
  arima = FALSE, data_str, cor_vars = NULL, fact_vars = list(NULL),
  unbal = list(level2 = FALSE, level3 = FALSE),
  unbal_design = list(level2 = NULL, level3 = NULL),
  lvl1_err_params = NULL, arima_mod = list(NULL), contrasts = NULL,
  homogeneity = TRUE, heterogeneity_var = NULL,
  cross_class_params = NULL, knot_args = list(NULL), ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`random3` One sided formula for random effects at third level in the simulation. Must be a subset of fixed (and likely of random).

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`random_param3` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be either 'single', 'level1', 'level2', or 'level3'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`k` Number of third level clusters.

`n` Cluster sample size.

`p` Within cluster sample size.

`error_var` Scalar of error variance.

`with_err_gen` Distribution function to pass on to the level one simulation of errors.

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`data_str` Type of data. Must be "cross", "long", or "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.

- var_type = Must be 'single', 'level1', 'level2', or 'level3'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`lvl1_err_params`

Additional parameters passed as a list on to the level one error generating function

`arima_mod` A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`homogeneity` Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`heterogeneity_var`

Variable name as a character string to use for heterogeneity of variance simulation.

`cross_class_params`

A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,

- ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`...` Not currently used.

Details

Simulated data is useful for classroom demonstrations and to study the impacts of assumption violations on parameter estimates, statistical power, or empirical type I error rates.

This function allows researchers a flexible approach to simulate regression models, including single level models and cross sectional or longitudinal linear mixed models (aka. hierarchical linear models or multilevel models).

Examples

```
# generating parameters for single level regression
fixed <- ~1 + act + diff + numCourse + act:numCourse
fixed_param <- c(2, 4, 1, 3.5, 2)
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
   var_type = c(”single”, ”single”, ”single”),
   opts = list(list(mean = 0, sd = 4),
   list(mean = 0, sd = 3),
   list(mean = 0, sd = 3)))
n <- 150
error_var <- 3
with_err_gen <- 'rnorm'
temp_single <- sim_reg(fixed = fixed, fixed_param = fixed_param,
   cov_param = cov_param,
   n = n, error_var = error_var, with_err_gen = with_err_gen,
   data_str = ”single”)
# Fitting regression to obtain parameter estimates
summary(lm(sim_data ~ 1 + act + diff + numCourse + act:numCourse,
   data = temp_single))

# Longitudinal linear mixed model example
fixed <- ~1 + time + diff + act + time:act
random <- ~1 + time + diff
fixed_param <- c(4, 2, 6, 2.3, 7)
random_param <- list(random_var = c(7, 4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm'),
  var_type = c(”level1”, ”level2”),
  opts = list(list(mean = 0, sd = 1.5),
  list(mean = 0, sd = 4)))
n <- 150
p <- 30
error_var <- 4
with_err_gen <- 'rnorm'
```

```
data_str <- "long"
temp_long <- sim_reg(fixed, random, random3 = NULL, fixed_param,
   random_param, random_param3 = NULL,
   cov_param, k = NULL, n, p, error_var, with_err_gen, data_str = data_str)

## fitting lmer model
library(lme4)
lmer(sim_data ~ 1 + time + diff + act + time:act +
  (1 + time + diff | clustID),
  data = temp_long)

# Three level example
fixed <- ~1 + time + diff + act + actClust + time:act
random <- ~1 + time + diff
random3 <- ~ 1 + time
fixed_param <- c(4, 2, 6, 2.3, 7, 0)
random_param <- list(random_var = c(7, 4, 2), rand_gen = 'rnorm')
random_param3 <- list(random_var = c(4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
     var_type = c("level1", "level2", "level3"),
     opts = list(list(mean = 0, sd = 1.5),
     list(mean = 0, sd = 4),
     list(mean = 0, sd = 2)))
k <- 10
n <- 15
p <- 10
error_var <- 4
with_err_gen <- 'rnorm'
data_str <- "long"
temp_three <- sim_reg(fixed, random, random3, fixed_param, random_param,
random_param3, cov_param, k,n, p, error_var, with_err_gen,
   data_str = data_str)

library(lme4)
lmer(sim_data ~ 1 + time + diff + act + actClust + time:act +
   (1 + time + diff | clustID) +
   (1 | clust3ID), data = temp_three)
```

sim_reg_nested *Function to simulate nested data*

Description

Takes simulation parameters as inputs and returns simulated data.

Usage

```
sim_reg_nested(fixed, random, fixed_param, random_param = list(),
  cov_param, n, p, error_var, with_err_gen, arima = FALSE, data_str,
  cor_vars = NULL, fact_vars = list(NULL), unbal = FALSE,
  unbal_design = NULL, lvl1_err_params = NULL,
  arima_mod = list(NULL), contrasts = NULL, homogeneity = TRUE,
  heterogeneity_var = NULL, cross_class_params = NULL,
  knot_args = list(NULL), ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'level1' or 'level2'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`n` Cluster sample size.

`p` Within cluster sample size.

`error_var` Scalar of error variance.

`with_err_gen` Simulated within cluster error distribution. Must be a quoted 'r' distribution function.

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`data_str` Type of data. Must be "cross" or "long".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.

- var_type = Must be 'level1' or 'level2'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A vector of sample sizes for the number of observations for each level 2 cluster. Must have same length as level two sample size n. Alternative specification can be TRUE, which uses additional argument, unbal_design.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two sample size.

`lvl1_err_params` Additional parameters passed as a list on to the level one error generating function

`arima_mod` A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`homogeneity` Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`heterogeneity_var` Variable name as a character string to use for heterogeneity of variance simulation.

`cross_class_params` A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

... Not currently used.

Details

Simulates data for the linear mixed model, both cross sectional and longitudinal data. Returns a data frame with ID variables, fixed effects, and many other variables useful to help when running simulation studies.

See Also

sim_reg for a convenient wrapper for all data conditions.

Examples

```
#' # Longitudinal linear mixed model example
fixed <- ~1 + time + diff + act + time:act
random <- ~1 + time + diff
fixed_param <- c(4, 2, 6, 2.3, 7)
random_param <- list(random_var = c(7, 4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm'),
  var_type = c("level1", "level2"),
  opts = list(list(mean = 0, sd = 1.5),
  list(mean = 0, sd = 4)))
n <- 150
p <- 30
error_var <- 4
with_err_gen <- 'rnorm'
data_str <- "long"
temp_long <- sim_reg(fixed, random, random3 = NULL, fixed_param,
   random_param, random_param3 = NULL,
   cov_param, k = NULL, n, p, error_var, with_err_gen, data_str = data_str)
```

---

sim_reg_nested3 *Function to simulate three level nested data*

Description

Takes simulation parameters as inputs and returns simulated data.

Usage

```
sim_reg_nested3(fixed, random, random3, fixed_param,
  random_param = list(), random_param3 = list(), cov_param, k, n, p,
  error_var, with_err_gen, arima = FALSE, data_str, cor_vars = NULL,
  fact_vars = list(NULL), unbal = list(level2 = FALSE, level3 = FALSE),
  unbal_design = list(level2 = NULL, level3 = NULL),
  lvl1_err_params = NULL, arima_mod = list(NULL), contrasts = NULL,
  homogeneity = TRUE, heterogeneity_var = NULL,
  cross_class_params = NULL, knot_args = list(NULL), ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`random` One sided formula for random effects in the simulation. Must be a subset of fixed.

`random3` One sided formula for random effects at third level in the simulation. Must be a subset of fixed (and likely of random).

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`random_param` A list of named elements that must contain:

- random_var: variance of random parameters,
- rand_gen: Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`random_param3` A list of named elements that must contain:

- random_var = variance of random parameters,
- rand_gen = Name of simulation function for random effects.

Optional elements are:

- ther: Theorectial mean and variance from rand_gen,
- ther_sim: Simulate mean/variance for standardization purposes,
- cor_vars: Correlation between random effects,
- ...: Additional parameters needed for rand_gen function.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'level1', 'level2', or 'level3'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`k` Number of third level clusters.

`n` Level two cluster sample size within each level three cluster.

`p` Within cluster sample size within each level two cluster.

`error_var` Scalar of error variance.

`with_err_gen` Simulated within cluster error distribution. Must be a quoted 'r' distribution function.

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`data_str` Type of data. Must be "cross" or "long".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'level1', 'level2', or 'level3'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`unbal` A named TRUE/FALSE list specifying whether unbalanced simulation design is desired. The named elements must be: "level2" or "level3" representing unbalanced simulation for level two and three respectively. Default is FALSE, indicating balanced sample sizes at both levels.

`unbal_design` When unbal = TRUE, this specifies the design for unbalanced simulation in one of two ways. It can represent the minimum and maximum sample size within a cluster via a named list. This will be drawn from a random uniform distribution with min and max specified. Secondly, the actual sample sizes within each cluster can be specified. This takes the form of a vector that must have the same length as the level two or three sample size. These are specified as a named list in which level two sample size is controlled via "level2" and level three sample size is controlled via "level3".

`lvl1_err_params`

Additional parameters passed as a list on to the level one error generating function

`arima_mod` A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`contrasts` An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`homogeneity` Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`heterogeneity_var`

Variable name as a character string to use for heterogeneity of variance simulation.

`cross_class_params`

A list of named parameters when cross classified data structures are desired. Must include the following arguments:

- num_ids: The number of cross classified clusters. These are in addition to the typical cluster ids
- random_param: This argument is a list of arguments passed to `sim_rand_eff`. These must include:
  - random_var: The variance of the cross classified random effect
  - rand_gen: The random generating function used to generate the cross classified random effect.

  Optional elements are:
  - ther: Theorectial mean and variance from rand_gen,
  - ther_sim: Simulate mean/variance for standardization purposes,
  - cor_vars: Correlation between random effects,
  - ...: Additional parameters needed for rand_gen function.

`knot_args` A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`...` Not currently used.

Details

Simulates data for the linear mixed model, both cross sectional and longitudinal data. Returns a data frame with ID variables, fixed effects, and many other variables useful to help when running simulation studies.

See Also

`sim_reg` for a convenient wrapper for all data conditions.

Examples

```
#' # Three level example
fixed <- ~1 + time + diff + act + actClust + time:act
random <- ~1 + time + diff
random3 <- ~ 1 + time
fixed_param <- c(4, 2, 6, 2.3, 7, 0)
random_param <- list(random_var = c(7, 4, 2), rand_gen = 'rnorm')
random_param3 <- list(random_var = c(4, 2), rand_gen = 'rnorm')
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
     var_type = c(”level1”, ”level2”, ”level3”),
     opts = list(list(mean = 0, sd = 1.5),
     list(mean = 0, sd = 4),
     list(mean = 0, sd = 2)))
k <- 10
n <- 15
p <- 10
error_var <- 4
with_err_gen <- 'rnorm'
data_str <- ”long”
```

```
temp_three <- sim_reg(fixed, random, random3, fixed_param, random_param,
   random_param3, cov_param, k,n, p, error_var, with_err_gen,
   data_str = data_str)
```

sim_reg_single *Master function to simulate single level data.*

Description

Takes simulation parameters as inputs and returns simulated data.

Usage

```
sim_reg_single(fixed, fixed_param, cov_param, n, error_var, with_err_gen,
  arima = FALSE, data_str, cor_vars = NULL, fact_vars = list(NULL),
  lvl1_err_params = NULL, arima_mod = list(NULL), contrasts = NULL,
  homogeneity = TRUE, heterogeneity_var = NULL,
  knot_args = list(NULL), ...)
```

Arguments

`fixed` One sided formula for fixed effects in the simulation. To suppress intercept add -1 to formula.

`fixed_param` Fixed effect parameter values (i.e. beta weights). Must be same length as fixed.

`cov_param` List of arguments to pass to the continuous generating function, must be the same order as the variables specified in fixed. This list does not include intercept, time, factors, or interactions. Required arguments include:

- dist_fun: This is a quoted R distribution function.
- var_type: This is the level of variable to generate. Must be 'single'. Must be same order as fixed formula above.

Optional arguments to the distribution functions are in a nested list, see the examples or vignettes for example code.

`n` Cluster sample size.

`error_var` Scalar of error variance.

`with_err_gen` Simulated within cluster error distribution. Must be a quoted 'r' distribution function.

`arima` TRUE/FALSE flag indicating whether residuals should be correlated. If TRUE, must specify a valid model to pass to arima.sim via the arima_mod argument. See `arima.sim` for examples.

`data_str` Type of data. Must be "single".

`cor_vars` A vector of correlations between variables.

`fact_vars` A nested list of factor, categorical, or ordinal variable specification, each list must include:

- numlevels = Number of levels for ordinal or factor variables.
- var_type = Must be 'single'.

Optional arguments include:

- replace
- prob
- value.labels

See also `sample` for use of these optional arguments.

`lvl1_err_params`
: Additional parameters passed as a list on to the level one error generating function

`arima_mod`
: A list indicating the ARIMA model to pass to arima.sim. See `arima.sim` for examples.

`contrasts`
: An optional list that specifies the contrasts to be used for factor variables (i.e. those variables with .f or .c). See `contrasts` for more detail.

`homogeneity`
: Either TRUE (default) indicating homogeneity of variance assumption is assumed or FALSE to indicate desire to generate heterogeneity of variance.

`heterogeneity_var`
: Variable name as a character string to use for heterogeneity of variance simulation.

`knot_args`
: A nested list of named knot arguments. See `sim_knot` for more details. Arguments must include:

- var
- knot_locations

`...`
: Not currently used.

Details

Simulates data for the simple regression models. Returns a data frame with ID variables, fixed effects, and many other variables useful to help when running simulation studies.

See Also

`sim_reg` for a convenient wrapper for all data conditions.

Examples

```
#' # generating parameters for single level regression
fixed <- ~1 + act + diff + numCourse + act:numCourse
fixed_param <- c(2, 4, 1, 3.5, 2)
cov_param <- list(dist_fun = c('rnorm', 'rnorm', 'rnorm'),
   var_type = c("single", "single", "single"),
   opts = list(list(mean = 0, sd = 4),
   list(mean = 0, sd = 3),
   list(mean = 0, sd = 3)))
n <- 150
error_var <- 3
```

```
with_err_gen <- 'rnorm'
temp_single <- sim_reg(fixed = fixed, fixed_param = fixed_param,
   cov_param = cov_param,
   n = n, error_var = error_var, with_err_gen = with_err_gen,
   data_str = ”single”)
```

sim_time *Simulate Time*

Description

This function simulates data for the time variable of longitudinal data.

Usage

```
sim_time(n, time_levels = NULL, ...)
```

Arguments

`n` Sample size of the levels.

`time_levels` The values the time variable should take. If NULL (default), the time values are discrete integers starting at 0 and going to n - 1.

`...` Currently not used.

transform_outcome *Transform response variable*

Description

Transform response variable

Usage

```
transform_outcome(outcome, type, ...)
```

Arguments

`outcome` The outcome variable to transform.

`type` Type of transformation to apply.

`...` Additional arguments passed to distribution functions.

varcov_randeff *Function to create random effect variance-covariance matrices*

Description

Input variances of random effects and correlation between random effects, returns variance-covariance matrix of random effects.

Usage

```
varcov_randeff(random_var, cor_re)
```

Arguments

random_var	Variance of random effects.
cor_re	Correlation between random effects, currently only a constant supported.

Index