Package 'simglm'

July 24, 2017

Type Package

Version 0.6.0

License MIT + file LICENSE

Title Simulate Models Based on the Generalized Linear Model

Description Easily 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), stats, Matrix, dplyr, purrr, tidyr, tibble

Suggests knitr, lme4, nlme, testthat, shiny, e1071, ggplot2

VignetteBuilder knitr

RoxygenNote 6.0.1

Author Brandon LeBeau [aut, cre]

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

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-24 15:48:32 UTC

R topics documented:

`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)

---

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)
```

dropout_missing *Dropout Missing Data*

Description

Function that inputs simulated data and returns data frame with new response variable that includes missing data. This function does dropout missing data, that is missing data dependent on time. Likely most appropriate for longitudinal data.

Usage

```
dropout_missing(sim_data, resp_var = "sim_data", clust_var = "clustID",
  within_id = "withinID", miss_prop)
```

Arguments

sim_data	Simulated data frame
resp_var	Response variable to add missing data to
clust_var	Cluster variable used for the grouping.
within_id	ID variable within each cluster.
miss_prop	Proportion of missing data overall or a vector the same length as the number of clusters representing the percentage of missing data for each cluster

Details

The function returns two new variables to the original data frame. The first is a dichotomous variable representing whether the response variable would be marked as NA (1) or not (0). The second is a re-representation of the response variable with the values coded as NA named 'sim_data2'.

---

mar_missing *Missing at Random*

Description

This type of missing data structure will be simulated based on values of a third variable. For example, the likelihood of a missing value is a function of gender, socioeconomic status, or age. Note, this function is similar to dropout missing data, but instead of missing due to time, this is missing due to another covariate.

Usage

```
mar_missing(sim_data, resp_var, miss_cov, miss_prop)
```

Arguments

sim_data	Simulated data frame
resp_var	Response variable to add missing data to
miss_cov	Covariate that the missing values are based on.
miss_prop	A vector the same length as the number of unique values from miss_cov variable.

missing_data *Master Missing Data Function*

Description

This function is a wrapper to easily call the specific types of missing data mechanisms.

Usage

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

Arguments

sim_data	Simulated data frame
resp_var	Response variable to add missing data to
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 or a vector the same length as the number of clusters representing the percentage of missing data for each cluster
type	The type of missing data to generate, currently supports droput, random, or missing at random (mar) missing data.
miss_cov	Covariate that the missing values are based on.

random_missing *Random Missing Data*

Description

Function that inputs simulated data and returns data frame with new response variable that includes missing data. This function simulates data that is randomly missing or missing completely at random.

Usage

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

Arguments

sim_data	Simulated data frame
resp_var	Response variable to add missing data to
miss_prop	Proportion of missing data overall or a vector the same length as the number of clusters representing the percentage of missing data for each cluster
clust_var	Cluster variable used for the grouping.
within_id	ID variable within each cluster.

Details

The function returns two new variables to the original data frame. The first is a dichotomous variable representing whether the response variable would be marked as NA (1) or not (0). The second is a re-representation of the response variable with the values coded as NA named 'sim_data2'.

---

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))
```

`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`). #'

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_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, replace = TRUE, prob = NULL,
  var_type = c("level1", "level2", "level3", "single"), value_labels = NULL)
```

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
replace	Whether to replace levels of categorical variable, TRUE/FALSE
prob	Probability of levels for variable, must be same length as numlevels
var_type	Variable type for the variable, must be either "level1", "level2", "level3", or "single"
value_labels	Optional argument with value labels for variable, converts variable to factor.

---

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)
```

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 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.

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)
```

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 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.

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)
```

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 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.

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, ...)
```

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.
...	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, ...)
```

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.
...	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, ...)
```

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.

`...` 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, ...)
```

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.
...	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_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), 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, ...)
```

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.

`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.
...	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 <- 2
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 <- 2
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), outcome_type,
  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, ...)
```

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".

`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.

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.
...	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), outcome_type, 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, ...)
```

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".

`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.

`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.

`...` 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),
  outcome_type, 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, ...)
```

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".

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.
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.
...	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), outcome_type, 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, ...)
```

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.
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.
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.
...	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), 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, ...)
```

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.
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.
...	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),
  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, ...)
```

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.

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.

... — 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), missing = FALSE,
  missing_args = list(NULL), pow_param = NULL, alpha, pow_dist = c("z",
  "t"), pow_tail = c(1, 2), lm_fit_mod = 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.

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.

...
: 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, ...)
```

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.

`...` 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, ...)
```

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.

...
: 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, ...)
```

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.

`...` 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, ...)
```

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.

`...` 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")
```

---

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