

Package ‘lsasim’

January 22, 2025

Title Functions to Facilitate the Simulation of Large Scale Assessment Data

Version 2.1.6

BugReports <https://github.com/tmatta/lsasim/issues>

Description Provides functions to simulate data from large-scale educational assessments, including background questionnaire data and cognitive item responses that adhere to a multiple-matrix sampled design. The theoretical foundation can be found on
Matta, T.H., Rutkowski, L., Rutkowski, D. et al. (2018)
<[doi:10.1186/s40536-018-0068-8](https://doi.org/10.1186/s40536-018-0068-8)>.

Imports mvtnorm, cli, methods, polycor

Depends R (>= 3.6.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests testthat, knitr, formatR, rmarkdown, NAEPirtparams

VignetteBuilder knitr

Date 2025-01-22

Language en-US

NeedsCompilation no

Author Tyler Matta [aut],
Leslie Rutkowski [aut],
David Rutkowski [aut],
Yuan-Ling Linda Liaw [aut],
Kondwani Kajera Mughogho [ctb],
Waldir Leoncio [aut, cre],
Sinan Yavuz [ctb],
Paul Bailey [ctb]

Maintainer Waldir Leoncio <w.l.netto@medisin.uio.no>

Repository CRAN

Date/Publication 2025-01-22 09:50:03 UTC

Contents

.onAttach	3
anova.lsasimcluster	4
attribute_cluster_labels	5
beta_gen	5
block_design	7
booklet_design	8
booklet_sample	9
brr	10
calc_n_tilde	11
calc_replicate_weights	12
calc_se_rho	13
calc_var_between	13
calc_var_tot	14
calc_var_within	15
check_condition	15
check_ignored_parameters	16
check_n_N_class	16
check_valid_structure	17
cluster_gen	17
cluster_gen_separate	20
cluster_gen_together	22
cluster_message	24
convert_vector_to_list	24
cor_gen	25
cov_gen	26
cov_yfz_gen	26
cov_yxw_gen	27
cov_yxz_gen	27
customize_summary	28
draw_cluster_structure	29
gen_cat_prop	29
gen_variable_n	30
gen_X_W_cluster	30
intraclass_cor	31
irt_gen	31
item_gen	32
jackknife	33
label_respondents	34
lambda_gen	34
lsasim	35
pisa2012_math_block	36
pisa2012_math_booklet	37
pisa2012_math_item	37
pisa2012_q_cormat	38
pisa2012_q_marginal	39
pluralize	40

print_anova 41
 proportion_gen 42
 pt_bis_conversion 42
 questionnaire_gen 43
 questionnaire_gen_family 47
 questionnaire_gen_polychoric 48
 ranges 48
 recalc_final_weights 49
 replicate_var 49
 response_gen 50
 rzeropois 52
 sample_from 52
 sample_within_range 53
 select 53
 split_cat_prop 54
 summary.Isasimcluster 54
 summary_2 55
 trim_sample 56
 validate_questionnaire_gen 56
 weight_responses 57
 whitelist_message 58

Index **59**

.onAttach	<i>Prints welcome message on package load</i>
-----------	---

Description

Prints welcome message with package version on load

Usage

`.onAttach(libname, pkgname)`

Arguments

libname	no idea, but will break devtools::document() if removed
pkgname	name of the package

Note

This function was adapted from the lavaan package, so credit for it goes to lavaan’s creator, Yves Rosseel

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL <http://www.jstatsoft.org/v48/i02/>.

anova.lsasimcluster *Generate an ANOVA table for LSASIM clusters*

Description

Prints Analysis of Variance table for cluster_gen output.

Usage

```
## S3 method for class 'lsasimcluster'  
anova(object, print = TRUE, calc.se = TRUE, ...)
```

Arguments

object	list output of cluster_gen
print	if TRUE, output will be a list containing estimators; if FALSE (default), output are formatted tables of this information
calc.se	if TRUE, will try to calculate the standard error of the intraclass correlation
...	additional objects of the same type (see help("anova") for details)

Value

Printed ANOVA table or list of parameters

Note

If the rhos for different levels are varied in scale, the generated rho will be less accurate.

References

Snijders, T. A. B., & Bosker, R. J. (1999). Multilevel Analysis. Sage Publications.

See Also

[summary.lsasimcluster\(\)](#)

`attribute_cluster_labels`*Attribute Labels in Hierarchical Structure*

Description

Attributes cluster and respondent labels in the context of `cluster_gen`.

Usage

```
attribute_cluster_labels(n)
```

Arguments

`n` numeric vector or list

Value

list containing appropriate labels for the clusters and their respondents

See Also

[cluster_gen\(\)](#)

`beta_gen`*Generate regression coefficients*

Description

Uses the output from `questionnaire_gen` to generate linear regression coefficients.

Usage

```
beta_gen(  
  data,  
  MC = FALSE,  
  MC_replications = 100,  
  CI = c(0.005, 0.995),  
  output_cov = FALSE,  
  rename_to_q = FALSE,  
  verbose = TRUE  
)
```

Arguments

data	output from the questionnaire_gen function with full_output = TRUE and theta = TRUE
MC	if TRUE, performs Monte Carlo simulation to estimate regression coefficients
MC_replications	for MC = TRUE, this represents the number of Monte Carlo subsamples calculated
CI	confidence interval for Monte Carlo simulations
output_cov	if TRUE, will also output the covariance matrix of YXW
rename_to_q	if TRUE, renames the variables from "x" and "w" to "q"
verbose	if FALSE, output messages will be suppressed (useful for simulations). Defaults to TRUE

Details

This function was primarily conceived as a sub-function of questionnaire_gen, when family = "gaussian", theta = TRUE, and full_output = TRUE. However, it can also be directly called by the user so they can perform further analysis.

This function primarily calculates the true regression coefficients (β) for the linear influence of the background questionnaire variables in θ . From a statistical perspective, this relationship can be modeled as follows, where $E(\theta|\mathbf{X}, \mathbf{W})$ is the expectation of θ given $\mathbf{X} = \{X_1, \dots, X_P\}$ and $\mathbf{W} = \{W_1, \dots, W_Q\}$:

$$E(\theta|\mathbf{X}, \mathbf{W}) = \beta_0 + \sum_{p=1}^P \beta_p X_p + \sum_{q=1}^Q \beta_{P+q} W_q$$

The regression coefficients are calculated using the true covariance matrix either provided by the user upon calling of questionnaire_gen or randomly generated by that function if none was provided. In any case, that matrix is not sample-dependent, though it should be similar to the one observed in the generated data (especially for larger samples). One convenient way to check for this similarity is by running the function with MC = TRUE, which will generate a numeric estimate; the MC_replications argument can be then increased to improve the estimates at a often-noticeable cost in processing time. If MC = FALSE, the MC_replications will have no effect on the results. In any case, each subsample will always have the same size as the original sample.

If the background questionnaire contains categorical variables (W), the original covariance matrix cannot be used because it contains the covariances involving $Z \sim N(0, 1)$, which is the random variable that gets categorized into W . The case where W is always binomial is trivial, but if at least one W has more than two categories, the structure of the covariance matrix changes drastically. In this case, this function recalculates all covariances between θ , X and each category of W using some auxiliary internal functions which rely on the appropriate distribution (either multivariate normal or truncated normal). To avoid multicollinearity, the first categories of each W are dropped before the regression coefficients are calculated.

Value

By default, this function will output a vector of the regression coefficients, including intercept. If MC == TRUE, the output will instead be a matrix comparing the true regression coefficients obtained

from the covariance matrix with expected values obtained from a Monte Carlo simulation, complete with 99\

If `output_cov = TRUE`, the output will be a list with two elements: the first one, `betas`, will contain the same output described in the previous paragraph. The second one, called `vcov_YXW`, contains the covariance matrix of the regression coefficients.

Note

The equation in this page is best formatted in PDF. We recommend issuing `help("beta_gen", help_type = "PDF")` in your terminal and opening the `beta_gen.pdf` file generated in your working directory. You may also set `help_type = "HTML"`, but the equations will have degraded formatting.

See Also

[questionnaire_gen\(\)](#)

Examples

```
data <- questionnaire_gen(100, family="gaussian", theta = TRUE,
                          full_output = TRUE, n_X = 2, n_W = list(2, 2, 4))
beta_gen(data, MC = TRUE)
```

block_design

Assignment of test items to blocks

Description

`block_design` creates a length-2 list containing:

- a matrix that identifies which items correspond to which blocks and
- a table of block descriptive statistics.

Usage

```
block_design(n_blocks = NULL, item_parameters, item_block_matrix = NULL)
```

Arguments

`n_blocks` an integer indicating how many blocks to create.

`item_parameters` a data frame of item parameters.

`item_block_matrix` a matrix of indicators to assign items to blocks.

The default `book_design` assigns two blocks to every booklet in a spiral design. The number of default booklets is equal to the number of blocks and must be ≥ 3 . If `ncol(item_block_assignment) < 3`, `book_design` must be specified.

Examples

```
i_blk_mat <- matrix(seq(1:40), ncol = 5)
blk_book <- matrix(c(1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1,
                    0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0),
                  ncol = 5, byrow = TRUE)
booklet_design(item_block_assignment = i_blk_mat, book_design = blk_book)
booklet_design(item_block_assignment = i_blk_mat)
```

booklet_sample	<i>Assignment of test booklets to test takers</i>
----------------	---

Description

`booklet_sample` randomly assigns test booklets to test takers.

Usage

```
booklet_sample(
  n_subj,
  book_item_design,
  book_prob = NULL,
  resample = FALSE,
  e = 0.1,
  iter = 20
)
```

Arguments

<code>n_subj</code>	an integer, the number of subjects (test takers).
<code>book_item_design</code>	a data frame containing the items that belong to each booklet with booklets as columns and booklet item numbers as rows. See 'Details'.
<code>book_prob</code>	a vector of probability weights for obtaining the booklets being sampled. The default equally weights all books.
<code>resample</code>	logical indicating if booklets should be re-sampled to minimize differences. Can only be used when <code>book_prob = NULL</code> .
<code>e</code>	a number between 0 and 1 exclusive, re-sampling stopping criteria, the difference between the most sampled and least sampled booklets.
<code>iter</code>	an integer defining the number of iterations to reach <code>e</code> .

Details

If using `booklet_sample` in tandem with `booklet_design`, `book_item_design` is the the first element of the returned list of `block_design`.

Examples

```
it_bk <- matrix(c(1, 2, 1, 4, 5, 4, 7, 8, 7, 10, 3, 10, 2, 6, 3, 5, 9, 6, 8, 0, 9),
               ncol = 3, byrow = TRUE)
booklet_sample(n_subj = 10, book_item_design = it_bk, book_prob = c(.2, .5, .3))
```

 brr

Generate replicates of a dataset using Balanced Repeated Replication

Description

Generate replicates of a dataset using Balanced Repeated Replication

Usage

```
brr(
  data,
  k = 0,
  pseudo_strata = ceiling(nrow(data)/2),
  reps = NULL,
  max_reps = 80,
  weight_cols = "none",
  id_col = 1,
  drop = TRUE
)
```

Arguments

<code>data</code>	dataset
<code>k</code>	deflating weight factor. $0 \leq k \leq 1$.
<code>pseudo_strata</code>	number of pseudo-strata
<code>reps</code>	number of replicates
<code>max_reps</code>	maximum number of replicates (only functional if <code>reps = NULL</code>)
<code>weight_cols</code>	vector of weight columns
<code>id_col</code>	number of column in dataset containing subject IDs. Set 0 to use the row names as ID
<code>drop</code>	if TRUE, the observation that will not be part of the subsample is dropped from the dataset. Otherwise, it stays in the dataset but a new weight column is created to differentiate the selected observations

Value

a list containing all the BRR replicates of data

Note

PISA uses the BRR Fay method with $k = 0.5$.

References

OECD (2015). Pisa Data Analysis Manual. Adams, R., & Wu, M. (2002). PISA 2000 Technical Report. Paris: Organization for Economic Co-operation and Development (OECD). Rust, K. F., & Rao, J. N. K. (1996). Variance estimation for complex surveys using replication techniques. *Statistical methods in medical research*, 5(3), 283-310.

See Also

[jackknife\(\)](#)

calc_n_tilde

Calculate \tilde{n}

Description

Calculates n tilde

Usage

```
calc_n_tilde(M, N, n_j)
```

Arguments

M	total number of population (i.e., sum of n_j over all j)
N	number of each class j
n_j	vector with size of each class j

References

Snijders, T. A. B., & Bosker, R. J. (1999). *Multilevel Analysis*. Sage Publications.

See Also

[summary.lsasimcluster\(\)](#)

`calc_replicate_weights`*Calculate replicate weights and summary statistics*

Description

Takes the output of `cluster_gen` to calculate the replicate weights as well as some summary statistics

Usage

```
calc_replicate_weights(data, method, k = 0)
```

Arguments

<code>data</code>	list of background questionnaire data (typically generated by <code>cluster_gen</code>)
<code>method</code>	replication method. Can be "Jackknife", "BRR" or "BRR Fay"
<code>k</code>	deflating weight factor (used only when 'method = "BRR Fay"')

Details

Replicate weights can be calculated using the Jackknife for unstratified two-stage sample designs or Balanced Repeated Replication (BRR) with or without Fay's modification. According to OECD (2015), PISA uses the Fay method with a factor of 0.5. This is why `k = .5` by default.

Value

list with data and, if requested, some statistics

Note

This function is essentially a big wrapper for `replicate_var`, applying that function on each element of an output of `cluster_gen`.

References

OECD (2015). Pisa Data Analysis Manual. Rust, K. F., & Rao, J. N. K. (1996). Variance estimation for complex surveys using replication techniques. *Statistical methods in medical research*, 5(3), 283-310.

See Also

[cluster_gen\(\)](#) [jackknife\(\)](#)

Examples

```
data <- cluster_gen(c(3, 50))
calc_replicate_weights(data, "Jackknife")
calc_replicate_weights(data, "BRR")
calc_replicate_weights(data, "BRR Fay")
```

calc_se_rho *Calculate Standard Error of Intraclass Correlation*

Description

Calculate Standard Error of Intraclass Correlation

Usage

```
calc_se_rho(rho, n_j, N)
```

Arguments

rho	intraclass correlation
n_j	number of elements in class j
N	number of classes j

References

Snijders, T. A. B., & Bosker, R. J. (1999). *Multilevel Analysis*. Sage Publications.

See Also

[anova.lsasimcluster\(\)](#)

calc_var_between *Calculate variance between classes*

Description

Calculate variance between classes

Usage

```
calc_var_between(n_j, y_bar_j, y_bar, n_tilde, N)
```

Arguments

n_j	number of elements in class j
y_bar_j	mean of variable of interest per class j
y_bar	mean of variable of interest across classes
n_tilde	function of the variance of n_N, M and N. See documentation and code of <code>lsasim::summary.lsasimcluster</code> for details
N	number of classes j

References

Snijders, T. A. B., & Bosker, R. J. (1999). *Multilevel Analysis*. Sage Publications.

See Also

[anova.lsasimcluster\(\)](#)

calc_var_tot	<i>Calculate the total variance</i>
--------------	-------------------------------------

Description

Calculate the total variance

Usage

```
calc_var_tot(M, N, n_tilde, s2_within, s2_between)
```

Arguments

M	total sample size
N	number of classes j
n_tilde	function of the variance of n_N, M and N. See documentation and code of <code>lsasim::summary.lsasimcluster</code> for details
s2_within	Within-class variance
s2_between	Between-class variance

References

Snijders, T. A. B., & Bosker, R. J. (1999). *Multilevel Analysis*. Sage Publications.

See Also

[anova.lsasimcluster\(\)](#)

calc_var_within *Calculate variance within classes*

Description

Calculate variance within classes

Usage

```
calc_var_within(n_j, s2_j, M, N)
```

Arguments

n_j	number of elements in class j
s2_j	variance of all elements in class j
M	total sample size
N	number of classes j

References

Snijders, T. A. B., & Bosker, R. J. (1999). *Multilevel Analysis*. Sage Publications.

See Also

[anova.lsasimcluster\(\)](#)

check_condition *Check if an error condition is satisfied*

Description

Check if an error condition is satisfied

Usage

```
check_condition(condition, message, fatal = TRUE)
```

Arguments

condition	logical test which if TRUE will cause the function to return an error message
message	error message to be displayed if condition is met.
fatal	if TRUE, error message is fatal, i.e., it will abort the parent function which called check_condition.

check_ignored_parameters

Checks if provided parameters are ignored

Description

Internal function to match non-null parameters with a vector of ignored parameters

Usage

```
check_ignored_parameters(provided_parameters, ignored_parameters)
```

Arguments

provided_parameters

vector of provided parameters

ignored_parameters

vector of ignored parameters

Value

Warning message listing ignored parameters

check_n_N_class

Check class of n or N

Description

Check the class of an object (usually n and N from cluster_gen)

Usage

```
check_n_N_class(x)
```

Arguments

x

either n or N from cluster_gen

Note

This function is primarily used as a way to simplify the classification of n and N in the cluster_gen function.

See Also

[cluster_gen\(\)](#)

check_valid_structure *Check if List is Valid*

Description

Checks if a list has a proper structure to be transformed into a hierarchical structure

Usage

```
check_valid_structure(n)
```

Arguments

n list

Value

Error if the structure is improper. Otherwise, there's no output.

See Also

[check_condition\(\)](#)

cluster_gen *Generate cluster sample*

Description

Generate cluster sample

Usage

```
cluster_gen(  
  n,  
  N = 1,  
  cluster_labels = NULL,  
  resp_labels = NULL,  
  cat_prop = NULL,  
  n_X = NULL,  
  n_W = NULL,  
  c_mean = NULL,  
  sigma = NULL,  
  cor_matrix = NULL,  
  separate_questionnaires = TRUE,  
  collapse = "none",  
  sum_pop = sapply(N, sum),
```

```

    calc_weights = TRUE,
    sampling_method = "mixed",
    rho = NULL,
    theta = FALSE,
    verbose = TRUE,
    print_pop_structure = verbose,
    ...
)

```

Arguments

n	numeric vector or list with the number of sampled observations (clusters or subjects) on each level
N	population size of each <i>sampled</i> cluster element on each level. Either a numeric vector or a list of numeric vectors. If N is a list, it must have the same length as n and each element of N must have the same length as the corresponding element of n
cluster_labels	character vector with the names of each cluster level
resp_labels	character vector with the names of the questionnaire respondents on each level
cat_prop	list of cumulative proportions for each item. If theta = TRUE, the first element of cat_prop must be a scalar 1, which corresponds to the theta.
n_X	list of n_X per cluster level
n_W	list of n_W per cluster level
c_mean	vector of means for the continuous variables or list of vectors for the continuous variables for each level. Defaults to 0, but may change if rho is set.
sigma	vector of standard deviations for the continuous variables or list of vectors for the continuous variables for each level. Defaults to 1, but may change if rho is set.
cor_matrix	Correlation matrix between all variables (except weights). By default, correlations are randomly generated.
separate_questionnaires	if TRUE, each level will have its own questionnaire
collapse	if TRUE, function output contains only one data frame with all answers. It can also be "none", "partial" and "full" for finer control on 3+ levels
sum_pop	total population at each level (sampled or not)
calc_weights	if TRUE, sampling weights are calculated
sampling_method	can be "SRS" for Simple Random Sampling, "PPS" for Probabilities Proportional to Size, "mixed" to use PPS for schools and SRS otherwise, or a vector with the sampling method for each level
rho	intraclass correlation (scalar, vector or list, as appropriate)
theta	if TRUE, the first continuous variable will be labeled 'theta'. Otherwise, it will be labeled 'q1'.
verbose	if TRUE, prints output messages

```

print_pop_structure
    if TRUE, prints the population hierarchical structure (as long as it differs from the
    sample structure)
...
    Additional parameters to be passed to questionnaire_gen()

```

Details

This function relies heavily in two sub-functions—`cluster_gen_separate` and `cluster_gen_together`—which can be called independently. This does not make `cluster_gen` a simple wrapper function, as it performs several operations prior to calling its sub-functions, such as randomly generating `n_X` and `n_W` if they are not determined by user. `n` can have unitary length, in which case all clusters will have the same size. `N` is *not* the population size across all elements of a level, but the population size for each element of one level. Regarding the additional parameters to be passed to `questionnaire_gen()`, they can be passed either in the same format as `questionnaire_gen()` or as more complex objects that contain information for each cluster level.

Value

list with background questionnaire data, grouped by level or not

Note

For the purpose of this function, levels are counted starting from the top nesting/clustering level. This means that, for example, schools are the first cluster level, classes are the second, and students are the third and final level. This behavior can be customized by naming the `n` argument or providing custom labels in `cluster_labels` and `resp_labels`.

Manually setting both `c_mean` and `rho`, while possible, may yield unexpected results due to how those parameters work together. A high intraclass correlation (`rho`) theoretically means that each group will end up with different means so they can be better separated. If `c_mean` is left untouched (i.e., at the default value of zero), then `c_mean` will freely change between clusters in order to result in the expected intraclass correlation. For large samples, `c_mean` will in practice correspond to the grand mean across that level, as the means of each element will be different no matter the sample size.

Moreover, if `c_mean`, `sigma` and `rho` are passed to the function, the means will be recalculated as a function of the other two parameters. The three are interdependent and cannot be passed simultaneously.

If in addition to `rho` the user also determine different means for each level, the only way the math can check out is if the variance in each group becomes very high. For examples of this scenario and the one described in the previous paragraph, check out the final section of this page.

The `ranges()` function should always be put inside a `list()`, as putting it inside a vector (`c()`) will cancel its effect. For more details, please read the documentation of the `ranges()` function.

The only arguments that can be used to label each level are `n`, `N`, `cluster_labels` and `resp_labels`. Labeling other arguments such as `c_mean` and `cat_prop` has no effect on the final results, but it is a recommended way for users to keep track of which value corresponds to which element in a complex hierarchical structure.

One of the extra arguments that can be passed by this function is `family`. If `family == "gaussian"`, the questionnaire will be generated assuming that all the variables are jointly-distributed as a multivariate normal. The default behavior is `family == NULL`, where the data is generated using the polychoric correlation matrix, with no distributional assumptions.

See Also

[cluster_gen_separate\(\)](#) [cluster_gen_together\(\)](#) [questionnaire_gen\(\)](#)

Examples

```
# Simple structure of 3 schools with 5 students each
cluster_gen(c(3, 5))

# Complex structure of 2 schools with different number of students,
# sampling weights and custom number of questions
n <- list(3, c(20, 15, 25))
N <- list(5, c(200, 500, 400, 100, 100))
cluster_gen(n, N, n_X = 5, n_W = 2)

# Condensing the output
set.seed(0); cluster_gen(c(2, 4))
set.seed(0); cluster_gen(c(2, 4), collapse=TRUE) # same, but in one dataset

# Condensing the output: 3 levels
str(cluster_gen(c(2, 2, 1), collapse="none"))
str(cluster_gen(c(2, 2, 1), collapse="partial"))
str(cluster_gen(c(2, 2, 1), collapse="full"))

# Controlling the intra-class correlation and the grand mean
x <- cluster_gen(c(5, 1000), rho = .9, n_X = 2, n_W = 0, c_mean = 10)
sapply(1:5, function(s) mean(x$school[[s]]$q1)) # means per school != 10
mean(sapply(1:5, function(s) mean(x$school[[s]]$q1))) # closer to c_mean

# Making the intraclass variance explode by forcing "incompatible" rho and c_mean
x <- cluster_gen(c(5, 1000), rho = .5, n_X = 2, n_W = 0, c_mean = 1:5)
anova(x)
```

cluster_gen_separate *Generate cluster samples with individual questionnaires*

Description

This is a sub-function of `cluster_gen` that performs cluster sampling, with the twist that each cluster level has its own questionnaire.

Usage

```

cluster_gen_separate(
  n_levels,
  n,
  N,
  sum_pop,
  calc_weights,
  sampling_method,
  cluster_labels,
  resp_labels,
  collapse,
  n_X,
  n_W,
  cat_prop,
  c_mean,
  sigma,
  cor_matrix,
  rho,
  theta,
  whitelist,
  verbose,
  ...
)

```

Arguments

n_levels	number of cluster levels
n	numeric vector or list with the number of sampled observations (clusters or subjects) on each level
N	population size of each <i>sampled</i> cluster element on each level. Either a numeric vector or a list of numeric vectors. If N is a list, it must have the same length as n and each element of N must have the same length as the corresponding element of n
sum_pop	total population at each level (sampled or not)
calc_weights	if TRUE, sampling weights are calculated
sampling_method	can be "SRS" for Simple Random Sampling, "PPS" for Probabilities Proportional to Size, "mixed" to use PPS for schools and SRS otherwise, or a vector with the sampling method for each level
cluster_labels	character vector with the names of each cluster level
resp_labels	character vector with the names of the questionnaire respondents on each level
collapse	if TRUE, function output contains only one data frame with all answers. It can also be "none", "partial" and "full" for finer control on 3+ levels
n_X	list of n_X per cluster level
n_W	list of n_W per cluster level

cat_prop	list of cumulative proportions for each item. If theta = TRUE, the first element of cat_prop must be a scalar 1, which corresponds to the theta.
c_mean	vector of means for the continuous variables or list of vectors for the continuous variables for each level
sigma	vector of standard deviations for the continuous variables or list of vectors for the continuous variables for each level
cor_matrix	Correlation matrix between all variables (except weights)
rho	intraclass correlation (scalar, vector or list, as appropriate)
theta	if TRUE, the first continuous variable will be labeled 'theta'. Otherwise, it will be labeled 'q1'.
whitelist	used when n = select(...), determines which PSUs get to generate questionnaires
verbose	if TRUE, prints output messages
...	Additional parameters to be passed to questionnaire_gen()

See Also

[cluster_gen\(\)](#) [cluster_gen_together\(\)](#)

cluster_gen_together *Generate cluster samples with lowest-level questionnaires*

Description

This is a sub-function of cluster_gen that performs cluster sampling where only the lowest-level individuals (e.g. students) fill out questionnaires.

Usage

```
cluster_gen_together(
  n_levels,
  n,
  N,
  sum_pop,
  calc_weights,
  sampling_method,
  cluster_labels,
  resp_labels,
  collapse,
  n_X,
  n_W,
  cat_prop,
  c_mean,
  sigma,
  cor_matrix,
```

```

    rho,
    verbose,
    ...
)

```

Arguments

n_levels	number of cluster levels
n	numeric vector or list with the number of sampled observations (clusters or subjects) on each level
N	population size of each <i>sampled</i> cluster element on each level. Either a numeric vector or a list of numeric vectors. If N is a list, it must have the same length as n and each element of N must have the same length as the corresponding element of n
sum_pop	total population at each level (sampled or not)
calc_weights	if TRUE, sampling weights are calculated
sampling_method	can be "SRS" for Simple Random Sampling, "PPS" for Probabilities Proportional to Size, "mixed" to use PPS for schools and SRS otherwise, or a vector with the sampling method for each level
cluster_labels	character vector with the names of each cluster level
resp_labels	character vector with the names of the questionnaire respondents on each level
collapse	if TRUE, function output contains only one data frame with all answers. It can also be "none", "partial" and "full" for finer control on 3+ levels
n_X	list of n_X per cluster level
n_W	list of n_W per cluster level
cat_prop	list of cumulative proportions for each item. If theta = TRUE, the first element of cat_prop must be a scalar 1, which corresponds to the theta.
c_mean	vector of means for the continuous variables or list of vectors for the continuous variables for each level
sigma	vector of standard deviations for the continuous variables or list of vectors for the continuous variables for each level
cor_matrix	correlation matrix or list of correlation matrices per PSU
rho	intraclass correlation (scalar, vector or list, as appropriate)
verbose	if TRUE, prints output messages
...	Additional parameters to be passed to questionnaire_gen()

See Also

[cluster_gen\(\)](#) [cluster_gen_separate\(\)](#)

cluster_message	<i>Print messages about clusters</i>
-----------------	--------------------------------------

Description

Prints messages about the cluster scheme before generating questionnaire responses.

Usage

```
cluster_message(
  n_obs,
  resp_labels,
  cluster_labels,
  n_levels,
  separate_questionnaires,
  type,
  detail = FALSE
)
```

Arguments

n_obs	list with the number of elements per level
resp_labels	character vector with the names of the questionnaire respondents on each level
cluster_labels	character vector with the names of each cluster level
n_levels	number of cluster levels
separate_questionnaires	if TRUE, each level will have its own questionnaire
type	Type of top-level message
detail	if TRUE, prints further details about each level composition

Value

Messages.

convert_vector_to_list	<i>Convert Vector to Expanded List</i>
------------------------	--

Description

Converts a vector to list where each element is replicated a certain number of times depending on the previous vector. Also work for ranged lists

Usage

```
convert_vector_to_list(x, x_max = x, verbose = TRUE)
```

Arguments

x	vector or ranged list to be converted
x_max	reference vector or ranged list with max values for x
verbose	if TRUE, sends messages to user about what's being done

Value

expanded/replicated version of x

cor_gen	<i>Generation of random correlation matrix</i>
---------	--

Description

Creates a random correlation matrix.

Usage

```
cor_gen(n_var, cov_bounds = c(-1, 1))
```

Arguments

n_var	integer number of variables.
cov_bounds	a vector containing the bounds of the covariance matrix.

Details

The result from cor_gen can be used directly with the cor_matrix argument of questionnaire_gen.

Examples

```
cor_gen(n_var = 10)
```

 cov_gen

Generation of covariance matrices

Description

Construct covariance matrices for the generation of simulated test data.

Usage

```
cov_gen(pr_grp_1, n_fac, n_ind, Lambda = 0:1)
```

Arguments

pr_grp_1	proportion of observations in group 1. Can be a scalar or a vector
n_fac	number of factors
n_ind	number of indicators per factor
Lambda	either a matrix containing the factor loadings or a vector containing the lower and upper limits for a randomly-generated Lambda matrix

Value

A list containing three covariance matrices: vcov_yxw, vcov_yxz and vcov_yfz

Examples

```
vcov <- cov_gen(pr_grp_1 = .5, n_fac = 3, n_ind = 2)
str(vcov)
```

 cov_yfz_gen

Generate latent regression covariance matrix

Description

Generates covariance matrix between Y, F and Z

Usage

```
cov_yfz_gen(n_ind, n_fac, Phi, n_z, sd_z, w_names, pr_grp_1)
```

Arguments

n_ind	number of indicator variables
n_fac	number of factors
Phi	latent regression correlation matrix
n_z	number of background variables
sd_z	standard deviation of background variables
w_names	names of W variables
pr_grp_1	scalar or list of proportions of the first group

cov_yxw_gen	<i>Setup full YXW covariance matrix</i>
-------------	---

Description

Setup full YXW covariance matrix

Usage

```
cov_yxw_gen(n_ind, n_z, Phi, n_fac, Lambda)
```

Arguments

n_ind	number of indicator variables
n_z	number of background variables
Phi	latent regression correlation matrix
n_fac	number of factor variables
Lambda	matrix containing the factor loadings

cov_yxz_gen	<i>Generate analytical covariance matrix</i>
-------------	--

Description

Generate analytical covariance matrix

Usage

```
cov_yxz_gen(vcov_yxw, w_names, Phi, pr_grp_1, n_ind, n_fac, Lambda, var_z)
```

Arguments

vcov_yxw	covariance matrix between Y, X and W
w_names	name of the W variables
Phi	latent regression correlation matrix
pr_grp_1	scalar or list of proportions of the first group
n_ind	number of indicator variables
n_fac	number of factors
Lambda	matrix containing the factor loadings
var_z	vector of variances of the background variables

customize_summary	<i>Customize Summary</i>
-------------------	--------------------------

Description

Adds standard deviations and removes quantiles from a `summary()` output

Usage

```
customize_summary(df_summary, df, numeric_cols, factor_cols, digits = 3)
```

Arguments

df_summary	dataframe containing summary statistics
df	original data frame
numeric_cols	indices of the numeric columns
factor_cols	indices of the factor columns
digits	controls the number of digits in the output

See Also

[summary\(\)](#) [summary.lsasimcluster\(\)](#)

`draw_cluster_structure`*Draw Cluster Structure*

Description

This function creates a visual representation of the hierarchical structure

Usage

```
draw_cluster_structure(n, labels = NULL, resp = NULL, output = "tree")
```

Arguments

<code>n</code>	same from <code>cluster_gen</code>
<code>labels</code>	corresponds to <code>cluster_labels</code> from <code>cluster_gen</code>
<code>resp</code>	corresponds to <code>resp_labels</code> from <code>cluster_gen</code>
<code>output</code>	"tree" draws a tree-like structure on the console, "text" prints the structure as a character vector

Value

Prints structure to console.

Note

This function is useful for checking how a `list()` object looks as a hierarchical structure, usually to be used as the `n` and/or `N` arguments of the `cluster_gen` function.

Examples

```
n <- c(2, 4, 3)
draw_cluster_structure(n)
draw_cluster_structure(n, output="text")
```

`gen_cat_prop`*Generates cat_prop for questionnaire_gen*

Description

Generates `cat_prop` for `questionnaire_gen`

Usage

```
gen_cat_prop(n_X, n_W, n_cat_W)
```

Arguments

n_X	number of continuous variables
n_W	number of categorical variables
n_cat_W	number of categories per categorical variable

gen_variable_n	<i>Randomly generate the quantity of background variables</i>
----------------	---

Description

Randomly generate the quantity of background variables

Usage

```
gen_variable_n(n_vars, n_X, n_W, theta = FALSE)
```

Arguments

n_vars	number of variables in total (n_X + n_W + theta)
n_X	number of continuous variables
n_W	number of categorical variables
theta	number of latent variables

Value

vector with n_vars, n_X and n_W

gen_X_W_cluster	<i>Generate n_X and n_W for clusters</i>
-----------------	--

Description

Generates n_X and n_W for cluster_gen based on a correlation matrix

Usage

```
gen_X_W_cluster(n_levels, separate, class_cor)
```

Arguments

n_levels	number of levels
separate	to the separate_questionnaires argument of cluster_gen
class_cor	corresponds to the class_cor argument of cluster_gen

intraclass_cor	<i>Intraclass correlation</i>
----------------	-------------------------------

Description

Calculates the intraclass correlation of clustered data

Usage

```
intraclass_cor(tau2_hat, sigma2_hat)
```

Arguments

tau2_hat	estimate of the true between-class correlation
sigma2_hat	estimate of the true within-class correlation

References

Snijders, T. A. B., & Bosker, R. J. (1999). *Multilevel Analysis*. Sage Publications.

See Also

[cluster_gen\(\)](#) [summary.lsasimcluster\(\)](#)

irt_gen	<i>Simulate item responses from an item response model</i>
---------	--

Description

Creates a data frame of item parameters.

Usage

```
irt_gen(theta, a_par = 1, b_par, c_par = 0, D = 1)
```

Arguments

theta	numeric ability estimate.
a_par	numeric discrimination parameter.
b_par	numeric or vector of numerics difficulty parameter(s).
c_par	numeric guessing parameter.
D	numeric parameter to specify logistic (1) or normal (1.7).

Examples

```
irt_gen(theta = 0.2, b_par = 0.6)
irt_gen(theta = 0.2, a_par = 1.15, b_par = 0.6)
irt_gen(theta = 0.2, a_par = 1.15, b_par = 0.6, c_par = 0.2)
```

 item_gen

Generation of item parameters from uniform distributions

Description

Creates a data frame of item parameters.

Usage

```
item_gen(
  b_bounds,
  a_bounds = NULL,
  c_bounds = NULL,
  thresholds = 1,
  n_1pl = NULL,
  n_2pl = NULL,
  n_3pl = NULL
)
```

Arguments

b_bounds	a vector containing the bounds of the the uniform distribution for sampling the difficulty parameters.
a_bounds	a vector containing the bounds of the the uniform distribution for sampling the discrimination parameters.
c_bounds	a vector containing the bounds of the the uniform distribution for sampling the guessing parameters.
thresholds	if numeric, number of thresholds for 1- and/or 2- parameter dichotomous items, if vector, each element is the number of thresholds corresponding to the vector of n_1pl and/or n_2pl.
n_1pl	if integer, number of 1-parameter dichotomous items, if vector, each element is the number of partial credit items corresponding to thresholds number.
n_2pl	if integer, number of 2-parameter dichotomous items, if vector, each element is the number of generalized partial credit items corresponding to thresholds number.
n_3pl	integer, number of 3-parameter items.

Details

The data frame includes two variables p and k which indicate the number of parameters and the number of thresholds, respectively

Examples

```

item_gen(b_bounds = c(-2, 2), a_bounds = c(.75, 1.25),
  thresholds = c(1, 2, 3), n_1p1 = c(5, 5, 5), n_2p1 = c(0, 0, 5))
item_gen(b_bounds = c(-2, 2), a_bounds = c(.75, 1.25), c_bounds = c(0, .25),
  n_2p1 = 5, n_3p1 = 5)

```

jackknife

Generate replicates of a dataset using Jackknife

Description

Generate replicates of a dataset using Jackknife

Usage

```
jackknife(data, weight_cols = "none", drop = TRUE)
```

Arguments

data	dataset
weight_cols	vector of weight columns
drop	if TRUE, the observation that will not be part of the subsample is dropped from the dataset. Otherwise, it stays in the dataset but a new weight column is created to differentiate the selected observations

Value

a list containing all the Jackknife replicates of data

See Also

[brr\(\)](#)

Examples

```

x <- data.frame(
  number = 1:5,
  letter = LETTERS[1:5],
  stringsAsFactors = FALSE
)
jackknife(x)
jackknife(x, drop = FALSE)

```

label_respondents	<i>Label respondents</i>
-------------------	--------------------------

Description

Generate level label combinations for each respondent

Usage

```
label_respondents(
  n_obs,
  cluster_labels = names(n_obs),
  add_last_level = FALSE,
  apply_labels = TRUE
)
```

Arguments

`n_obs` list with the number of elements per level
`cluster_labels` character vector with the names of each cluster level
`add_last_level` if TRUE (not default), adds the last level to the output table
`apply_labels` if TRUE, applies labels (column names) to data cells

Value

Data frame with the combinations of IDs from all levels

lambda_gen	<i>Randomly generate a matrix of factor loadings</i>
------------	--

Description

Randomly generate a matrix of factor loadings

Usage

```
lambda_gen(n_ind, n_fac, limits, row_names, col_names)
```

Arguments

`n_ind` number of indicators per factor
`n_fac` number of factors
`limits` vector with lower and upper limits for the uniformly-generated Lambdas
`row_names` vector with row names
`col_names` vector with col names

`lsasim`*lsasim: A package for simulating large scale assessment data*

Description

Isasim simulates data that mimics large-scale assessments (LSAs), including background questionnaire data and cognitive item responses that adhere to a multiple-matrix sampled design

Functions to Facilitate the Simulation of Large Scale Assessment Data

Core functions

- `block_design` Assignment of test items to blocks.
- `booklet_design` Assignment of item blocks to test booklets.
- `booklet_sample` Assignment of test booklets to test takers.
- `item_gen` Generation of random correlation matrix.
- `proportion_gen` Generation of random cumulative proportions.
- `questionnaire_gen` Generation of ordinal and continuous variables.
- `response_gen` Generation of item response data using a rotated block design.
- `cluster_gen` Generation of background questionnaires from a cluster sampling scheme.

Useful ancillary functions

- `irt_gen` Generate item responses from an IRT model. Used by `response_gen`.
- `beta_gen` Calculates analytical and numeric regression coefficients for the background questionnaire responses as functions of the latent variable. Used by `questionnaire_gen`

Note

This package contains vignettes. If you are installing `lsasim` from GitHub, remember to use `build_vignettes=TRUE` in your `remotes::install_github()` call. Afterwards, you can browse the vignettes by issuing `browseVignettes("lsasim")` in your R terminal.

Author(s)

Maintainer: Waldir Leoncio <w.l.netto@medisin.uio.no>

Authors:

- Tyler Matta <tyler.matta@gmail.com>
- Leslie Rutkowski <leslie.rutkowski@cemo.uio.no>
- David Rutkowski <david.rutkowski@cemo.uio.no>
- Yuan-Ling Linda Liaw <y.l.liaw@cemo.uio.no>

Other contributors:

- Kondwani Kajera Mughogho <k.k.mughogho@cemo.uio.no> [contributor]
- Sinan Yavuz [contributor]
- Paul Bailey [contributor]

See Also

Useful links:

- Report bugs at <https://github.com/tmatta/lsasim/issues>

pisa2012_math_block *PISA 2012 mathematics item - item block indicator matrix*

Description

A dataset containing indicators associating those PISA 2012 mathematics items to the PISA 2012 mathematics item blocks.

Usage

```
pisa2012_math_block
```

Format

A data frame with 109 rows and 12 variables:

item_name Item name.

item_no Item numbers.

block1 Indicator specifying those items in block 1.

block2 Indicator specifying those items in block 2.

block3 Indicator specifying those items in block 3.

block4 Indicator specifying those items in block 4.

block5 Indicator specifying those items in block 5.

block6 Indicator specifying those items in block 6.

block7 Indicator specifying those items in block 7.

block8 Indicator specifying those items in block 8.

block9 Indicator specifying those items in block 9.

block10 Indicator specifying those items in block 10.

Source

PISA 2012 Technical Report, ANNEX A. Table A.1: PISA 2012 Main Survey mathematics item classification. Pages 406 - 409. <https://www.oecd.org/en/about/programmes/pisa.html>

pisa2012_math_booklet *PISA 2012 mathematics item block - test booklet indicator matrix*

Description

A dataset containing indicators associating those PISA 2012 mathematics item blocks to the PISA 2012 mathematics standard test booklet set.

Usage

pisa2012_math_booklet

Format

A data frame with 13 rows and 10 variables:

booklet Booklet name.

b1 Indicator specifying those test booklets that use item block 1.

b2 Indicator specifying those test booklets that use item block 2.

b3 Indicator specifying those test booklets that use item block 3.

b4 Indicator specifying those test booklets that use item block 4.

b5 Indicator specifying those test booklets that use item block 5.

b6 Indicator specifying those test booklets that use item block 6.

b7 Indicator specifying those test booklets that use item block 7.

b8 Indicator specifying those test booklets that use item block 8.

b9 Indicator specifying those test booklets that use item block 9.

Source

PISA 2012 Technical Report, Chapter 2: Test Design and Test Development. Figure 2.1: Cluster rotation design used to form standard test booklets for PISA 2012. Page 31. <https://www.oecd.org/en/about/programmes/pisa.html>

pisa2012_math_item *Item parameter estimates for 2012 PISA mathematics assessment*

Description

A dataset containing the estimated item parameters for the PISA 2012 mathematics assessment.

Usage

pisa2012_math_item

Format

A data frame with 109 rows and 5 variables:

item_name Item name.

item Item number.

b b parameter estimate.

d1 d1 parameter estimate (for partial credit items).

d2 d2 parameter estimate (for partial credit items).

Source

PISA 2012 Technical Report, ANNEX A. Table A.1: PISA 2012 Main Survey mathematics item classification. Pages 406 - 409. <https://www.oecd.org/en/about/programmes/pisa.html>

pisa2012_q_cormat

Correlation matrix from the PISA 2012 background questionnaire

Description

A correlation matrix for the selected background questionnaires and mathematics plausible value.

Usage

pisa2012_q_cormat

Format

An 19 by 19 matrix.

Details

A heterogeneous correlation matrix, consisting of polyserial correlations between numeric and ordinal variables, and polychoric correlations between ordinal variables.

Row/Col	Name	Label	Type
1	ST93Q01	Perseverance	Ordinal
2	ST93Q03	Perseverance	Ordinal
3	ST93Q04	Perseverance	Ordinal
4	ST93Q06	Perseverance	Ordinal
5	ST93Q07	Perseverance	Ordinal
6	ST94Q05	Openness for Problem Solving	Ordinal
7	ST94Q06	Openness for Problem Solving	Ordinal
8	ST94Q09	Openness for Problem Solving	Ordinal
9	ST94Q10	Openness for Problem Solving	Ordinal
10	ST94Q14	Openness for Problem Solving	Ordinal
11	ST88Q01	Attitude toward School	Ordinal

12	ST88Q02	Attitude toward School	Ordinal
13	ST88Q03	Attitude toward School	Ordinal
14	ST88Q04	Attitude toward School	Ordinal
15	ST89Q02	Attitude toward School	Ordinal
16	ST89Q03	Attitude toward School	Ordinal
17	ST89Q04	Attitude toward School	Ordinal
18	ST89Q05	Attitude toward School	Ordinal
19	1PV1MATH	Mathematics Plausible Value 1	Continuous

Warning

These data are for illustration purposes only. Handling of missing data may not be suitable for valid inferences.

Source

Raw data can be found at <https://www.oecd.org/en/data/datasets/pisa-2012-database.html> Codebook can be found at <https://www.oecd.org/en/about/programmes/pisa.html>

pisa2012_q_marginal *Marginal proportions from the PISA 2012 background questionnaire*

Description

Marginal proportions from the PISA 2012 background questionnaire

Usage

pisa2012_q_marginal

Format

A list of 19 named numeric vectors.

Details

A list containing the marginal cumulative proportions for each response category from the PISA 2012 background questionnaire. Elements 1 - 18 are the marginal proportions for the selected items from the background questionnaire. Element 19 is the marginal proportion for the selected mathematics plausible value.

Row/Col	Name	Label	Length
1	ST93Q01	Perseverance	5
2	ST93Q03	Perseverance	5
3	ST93Q04	Perseverance	5
4	ST93Q06	Perseverance	5

5	ST93Q07	Perseverance	5
6	ST94Q05	Openness for Problem Solving	5
7	ST94Q06	Openness for Problem Solving	5
8	ST94Q09	Openness for Problem Solving	5
9	ST94Q10	Openness for Problem Solving	5
10	ST94Q14	Openness for Problem Solving	5
11	ST88Q01	Attitude toward School	4
12	ST88Q02	Attitude toward School	4
13	ST88Q03	Attitude toward School	4
14	ST88Q04	Attitude toward School	4
15	ST89Q02	Attitude toward School	4
16	ST89Q03	Attitude toward School	4
17	ST89Q04	Attitude toward School	4
18	ST89Q05	Attitude toward School	4
19	1PV1MATH	Mathematics Plausible Value 1	1

Warning

These data are for illustration purposes only. Handling of missing data may not be suitable for valid inferences.

Source

Raw data can be found at <https://www.oecd.org/en/data/datasets/pisa-2012-database.html> Codebook can be found at <https://www.oecd.org/en/about/programmes/pisa.html>

pluralize

Pluralize words

Description

Pluralize a word

Usage

```
pluralize(word, n = rep(2, length(word)))
```

Arguments

word	vector of characters to be pluralized
n	vector of number of times each word appears (to determine if the plural or single form will be returned)

Value

word, either pluralized or not (depending on n)

print_anova	<i>Print the ANOVA table</i>
-------------	------------------------------

Description

Print the ANOVA table

Usage

```
print_anova(  
  s2_within,  
  s2_between,  
  s2_total,  
  sigma2_hat,  
  tau2_hat,  
  rho_hat,  
  se_rho,  
  n_tilde,  
  M,  
  N  
)
```

Arguments

s2_within	Within-class variance
s2_between	Between-class variance
s2_total	Total variance
sigma2_hat	estimate of the true within-class correlation
tau2_hat	estimate of the true between-class correlation
rho_hat	estimated intraclass correlation
se_rho	standard errors of rho_hat
n_tilde	function of the variance of n_N, M and N. See documentation and code of <code>lsasim:::summary.lsasimcluster</code> for details
M	total sample size
N	number of classes j

References

Snijders, T. A. B., & Bosker, R. J. (1999). *Multilevel Analysis*. Sage Publications.

See Also

[stats::anova\(\)](#)

proportion_gen *Generation of random cumulative proportions*

Description

Creates a list of vectors, each containing the randomly generated cumulative proportions of a discrete variable.

Usage

```
proportion_gen(cat_options, n_cat_options)
```

Arguments

cat_options vector of response types.
n_cat_options vector of number of items of the corresponding response type.

Details

cat_options and n_cat_options must be the same length. cat_options = 1 is a continuous variable.

The result from proportion_gen can be used directly with the cat_prop argument of questionnaire_gen.

Examples

```
proportion_gen(cat_options = c(1, 2, 3), n_cat_options = c(2, 2, 2))
proportion_gen(cat_options = c(1, 3), n_cat_options = c(4, 5))
```

pt_bis_conversion *Analytical point-biserial conversion*

Description

Analytical point-biserial conversion

Usage

```
pt_bis_conversion(bis_cor, pr_group1)
```

Arguments

bis_cor biserial correlations
pr_group1 probability of group 1

questionnaire_gen *Generation of ordinal and continuous variables*

Description

Creates a data frame of discrete and continuous variables based on several arguments.

Usage

```
questionnaire_gen(
  n_obs,
  cat_prop = NULL,
  n_vars = NULL,
  n_X = NULL,
  n_W = NULL,
  cor_matrix = NULL,
  cov_matrix = NULL,
  c_mean = NULL,
  c_sd = NULL,
  theta = FALSE,
  family = NULL,
  full_output = FALSE,
  verbose = TRUE
)
```

Arguments

n_obs	number of observations to generate.
cat_prop	list of cumulative proportions for each item. If theta = TRUE, the first element of cat_prop must be a scalar 1, which corresponds to the theta.
n_vars	total number of variables in the questionnaire, including the continuous and the discrete covariates (X and W , respectively), as well as the latent trait (Y , which is equivalent to θ).
n_X	number of continuous background variables. If not provided, a random number of continuous variables will be generated.
n_W	either a scalar corresponding to the number of categorical background variables or a list of scalars representing the number of categories for each categorical variable. If not provided, a random number of categorical variables will be generated.
cor_matrix	latent correlation matrix. The first row/column corresponds to the latent trait (Y). The other rows/columns correspond to the continuous (X or Z) or the discrete (W) background variables, in the same order as cat_prop.
cov_matrix	latent covariance matrix, formatted as cor_matrix.
c_mean	is a vector of population means for each continuous variable (Y and X). Defaults to 0.

<code>c_sd</code>	is a vector of population standard deviations for each continuous variable (Y and X). Defaults to 1.
<code>theta</code>	if TRUE, the first continuous variable will be labeled 'theta'. Otherwise, it will be labeled 'q1'.
<code>family</code>	distribution of the background variables. Can be NULL (default) or 'gaussian'.
<code>full_output</code>	if TRUE, output will be a list containing the questionnaire data as well as several objects that might be of interest for further analysis of the data.
<code>verbose</code>	if FALSE, output messages will be suppressed (useful for simulations). Defaults to TRUE

Details

In essence, this function begins by checking the validity of the arguments provided and randomly generating those that are not. Then, it will call one of two internal functions, `questionnaire_gen_polychoric` or `questionnaire_gen_family`. The former corresponds to the exact functionality of `questionnaire_gen` on `lsasim 1.0.1`, where the polychoric correlations are used to generate the background questionnaire data. If `family != NULL`, however, `questionnaire_gen_family` is called to generate data based on a joint probability distribution. Additionally, if `full_output == TRUE`, the external function `beta_gen` is called to generate the correlation coefficients based on the true covariance matrix. The latter argument also changes the class of the output of this function.

What follows are some notes on the input parameters.

`cat_prop` is a list where `length(cat_prop)` is the number of items to be generated. Each element of the list is a vector containing the marginal cumulative proportions for each category, summing to 1. For continuous items, the associated element in the list should be 1.

`cor_matrix` and `cov_matrix` are the correlation and covariance matrices that are the same size as `length(cat_prop)`. The correlations related to the correlation between variables on the latent scale.

`c_mean` and `c_sd` are each vectors whose length is equal to the number of continuous variables as specified by `cat_prop`. The default is to keep the continuous variables with mean zero and standard deviation of one.

`theta` is a logical indicator that determines if the first continuous item should be labeled *theta*. If `theta == TRUE` but there are no continuous variables generated, a random number of background variables will be generated.

If `cat_prop` is a named list, those names will be used as variable names for the returned `data.frame`. Generic names will be provided to the variables if `cat_prop` is not named.

As an alternative to providing `cat_prop`, the user can call this function by specifying the total number of variables using `n_vars` or the specific number of continuous and categorical variables through `n_X` and `n_W`. All three arguments should be provided as scalars; `n_W` may also be provided as a list, where each element contains the number of categories for one background variable. Alternatively, `n_W` may be provided as a one-element list, in which case it will be interpreted as all the categorical variables having the same number of categories.

If `family == "gaussian"`, the questionnaire will be generated assuming that all the variables are jointly-distributed as a multivariate normal. The default behavior is `family == NULL`, where the data is generated using the polychoric correlation matrix, with no distributional assumptions.

When data is generated using the Gaussian distribution, the matrices provided correspond to the relations between the latent variable θ , the continuous covariates X and the continuous covariates— $Z \sim N(0, 1)$ —that will later be discretized into categorical covariates W . That is why there will be a difference in labels and lengths between `cov_matrix` and `vcov_YXW`. For more information, check the references cited later in this document.

Value

By default, the function returns a `data.frame` object where the first column ("subject") is a $1, \dots, n$ ordered list of the n observations and the other columns correspond to the questionnaire answers. If `theta = TRUE`, the first column after "subject" will be the latent variable θ ; in any case, the continuous variables always come before the categorical ones.

If `full_output = TRUE`, the output will be a list containing the following objects:

<code>bg</code>	a data frame containing the background questionnaire answers (i.e., the same object as described above).
<code>c_mean</code>	identical to the input argument of the same name. Read the Details section for more information.
<code>c_sd</code>	identical to the input argument of the same name. Read the Details section for more information.
<code>cat_prop</code>	identical to the input argument of the same name. Read the Details section for more information.
<code>cat_prop_W_p</code>	a list containing the probabilities for each category of the categorical variables (<code>cat_prop_W</code> contains the cumulative probabilities).
<code>cor_matrix</code>	identical to the input argument of the same name. Read the Details section for more information.
<code>cov_matrix</code>	identical to the input argument of the same name. Read the Details section for more information.
<code>family</code>	identical to the input argument of the same name.
<code>n_obs</code>	identical to the input argument of the same name.
<code>n_tot</code>	named vector containing the number of total variables, the number of continuous background variables (i.e., the total number of background variables except θ) and the number of categorical variables.
<code>n_W</code>	vector containing the number of categorical variables.
<code>n_X</code>	vector containing the number of continuous variables (except θ).
<code>sd_YXW</code>	vector with the standard deviations of all the variables
<code>sd_YXZ</code>	vector containing the standard deviations of θ , the background continuous variables (X) and the Normally-distributed variables Z which will generate the background categorical variables (W).
<code>theta</code>	identical to the input argument of the same name.
<code>var_W</code>	list containing the variances of the categorical variables.
<code>var_YX</code>	list containing the variances of the continuous variables (including θ)

linear_regression

This list is printed only if `theta = TRUE`, `family = "gaussian"` and `full_output = TRUE`. It contains one vector named `betas` and one table named `cov_YXW`. The former displays the true linear regression coefficients of *theta* on the background questionnaire answers; the latter contains the covariance matrix between all these variables.

Note

If `family == NULL`, the number of levels for each categorical variables will be determined by the number of categories observed in the generated data. This means it might be smaller than the number of categories determined by `cat_prop`, which is more likely to happen with small values of `n_obs`. If `family == "gaussian"`, however, the number of levels for the categorical variables will always be equivalent to the number of possible categories, even if they are not observed in the data.

It is important to note that all arguments directly related to variable parameters (e.g. `cat_prop`, `cov_matrix`, `cor_matrix`, `c_mean`, `c_sd`) have the following order: Y, X, W (missing variables are skipped). This must be kept in mind when using real-life data as input to `questionnaire_gen`, as the input might need to be reordered to fit the expectations of the function.

By definition, the expected order of the variables is *theta*, followed by *X* and then *W*. The reference category of the categorical variables *W* is always the first one.

For very small means/sigmas (e.g. 0.005) and multiple levels, estimates may have differing levels of accuracy (e.g. school level estimates will not be as accurate as the student levels ones). In general, one should expect naturally worse estimation on higher hierarchical setups.

References

Matta, T. H., Rutkowski, L., Rutkowski, D., & Liaw, Y. L. (2018). *Isasim*: an R package for simulating large-scale assessment data. *Large-scale Assessments in Education*, 6(1), 15.

See Also

[beta_gen\(\)](#)

Examples

```
# Using polychoric correlations
props <- list(c(1), c(.25, .6, 1)) # one continuous, one with 3 categories
questionnaire_gen(n_obs = 10, cat_prop = props,
                  cor_matrix = matrix(c(1, .6, .6, 1), nrow = 2),
                  c_mean = 2, c_sd = 1.5, theta = TRUE)

# Using the multinomial distribution
# two categorical variables W: one has 2 categories, the other has 3
props <- list(1, c(.25, 1), c(.2, .8, 1))
yw_cov <- matrix(c(1, .5, .5, .5, 1, .8, .5, .8, 1), nrow = 3)
questionnaire_gen(n_obs = 10, cat_prop = props, cov_matrix = yw_cov,
                  family = "gaussian")

# Not providing covariance matrix
questionnaire_gen(n_obs = 10,
```

```
cat_prop = list(c(.25, 1), c(.6, 1), c(.2, 1)),  
family = "gaussian")
```

questionnaire_gen_family

Generation of ordinal and continuous variables

Description

Creates a data frame of discrete and continuous variables based on a latent correlation matrix and marginal proportions.

Usage

```
questionnaire_gen_family(  
  n_obs,  
  cat_prop,  
  cov_matrix,  
  family = "gaussian",  
  theta = FALSE,  
  mean_yx = NULL,  
  n_cats  
)
```

Arguments

n_obs	number of observations to generate.
cat_prop	list of cumulative proportions for each item.
cov_matrix	covariance matrix. between the latent trait (Y) and the background variables (X and Z).
family	distribution of the background variables. Can be NULL or 'gaussian'.
theta	if TRUE will label the first continuous variable 'theta'.
mean_yx	vector with the means of the latent trait (Y) and the continuous background variables with flexible variance (X).
n_cats	vector with number of categories for each W.

`questionnaire_gen_polychoric`*Generation of ordinal and continuous variables*

Description

Creates a data frame of discrete and continuous variables based on a latent correlation matrix and marginal proportions.

Usage

```
questionnaire_gen_polychoric(n_obs, cat_prop, cor_matrix, c_mean, c_sd, theta)
```

Arguments

<code>n_obs</code>	number of observations to generate.
<code>cat_prop</code>	list of cumulative proportions for each item.
<code>cor_matrix</code>	latent correlation matrix.
<code>c_mean</code>	is a vector of population means for each continuous variable.
<code>c_sd</code>	is a vector of population standard deviations for each continuous variable.
<code>theta</code>	if TRUE will label the first continuous variable 'theta'.

`ranges`*Defines vector as range*

Description

Redefines the class of a vector as "range"

Usage

```
ranges(x, y)
```

Arguments

<code>x</code>	first element
<code>y</code>	second element

Value

`c(x, y)`, but with the "range" class

Note

This function was created to be used as an element in the `N` argument of `cluster_gen`. The name was chosen to avoid conflict with `base::range()`.

`ranges()` should always be used within a `list()`. Inserting a "range" vector inside a common vector (`c()`) will result in a common vector. For example, `c(3, ranges(8, 10))` is the same as `c(3, 8, 10)`, because when faced with conflicting classes in the same element, R will resolve to the simpler case ("numeric", in this case). An easier way to understand this concept is by checking `class(c(3, "a"))` is "character", meaning the number 3 was devolved into a character "3".

`recalc_final_weights` *Recalculate final weights*

Description

Recalculate final weights given the replicate weights

Usage

```
recalc_final_weights(data, w_cols, replicate_weight = 1, reorder = TRUE)
```

Arguments

<code>data</code>	dataset
<code>w_cols</code>	columns containing the weights
<code>replicate_weight</code>	scalar with the replicate weights
<code>reorder</code>	if TRUE, reorders the dataset so that the replicate weights appear before the final weights

Value

input data with recalculated final weights, incorporating the replicate weights

`replicate_var` *Sampling variance of the mean for replications*

Description

Estimates the mean variance for Jackknife, BRR and BRR Fay replication methods

Usage

```
replicate_var(  
  data_whole,  
  data_rep,  
  method,  
  k = 0,  
  weight_var = NULL,  
  stat = weighted.mean,  
  vars = NULL,  
  full_output = FALSE  
)
```

Arguments

data_whole	full, original dataset (the one that generated the replications)
data_rep	list with replications of data_whole
method	replication method. Can be "Jackknife", "BRR" or "BRR Fay"
k	deflating weight factor (used only when 'method = "BRR Fay"')
weight_var	variables containing the weights
stat	statistic of interest to calculate (must be a base R function)
vars	vector containing the variables of interest
full_output	if TRUE, returns all intermediate objects created

Details

data_rep can be obtained from

See Also

[jackknife\(\)](#) [brr\(\)](#)

response_gen

Generation of item response data using a rotated block design

Description

Creates a data frame of discrete item responses based on.

Usage

```
response_gen(
  subject,
  item,
  theta,
  a_par = NULL,
  b_par,
  c_par = NULL,
  d_par = NULL,
  item_no = NULL,
  ogive = "Logistic"
)
```

Arguments

subject	integer vector of test taker IDs.
item	integer vector of item IDs.
theta	numeric vector of latent test taker abilities.
a_par	numeric vector of item a parameters for each item.
b_par	numeric vector of item b parameters for each item.
c_par	numeric vector of item c parameters for each item.
d_par	list of numeric vectors of item threshold parameters for each item.
item_no	vector of item numbers the correspond the item parameters
ogive	can be "Normal" or "Logistic".

Details

subject and item must be equal lengths.

Generalized partial credit models (!is.null(d_par)) uses threshold parameterization.

Examples

```
set.seed(1234)
s_id <- c(1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4,
  4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7,
  7, 7, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10,
  10, 11, 11, 11, 11, 11, 11, 12,12, 12, 12, 12, 12, 12, 13, 13, 13, 13,
  13, 13, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 16,16, 16, 16,
  16, 16, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 19, 19,
  19, 19, 19,19, 20, 20, 20, 20, 20, 20, 20)
i_id<- c(1, 4, 7, 10, 3, 6, 9, 1, 4, 7, 10, 2, 5, 8, 1, 4, 7, 10, 3, 6, 9, 1, 4,
  7, 10, 3, 6, 9, 1, 4, 7, 10, 3, 6, 9, 2, 5, 8, 3, 6, 9, 1, 4, 7, 10, 2,
  5, 8, 2, 5, 8, 3, 6, 9, 1, 4, 7, 10, 2, 5, 8, 1, 4, 7, 10, 3, 6, 9, 2,
  5, 8, 3, 6, 9, 1, 4, 7, 10, 3, 6, 9, 2, 5, 8, 3, 6, 9, 2, 5, 8, 3, 6, 9,
  2, 5, 8, 3, 6, 9, 2, 5, 8, 3, 6, 9, 1, 4, 7, 10, 2, 5, 8, 1, 4, 7, 10,
  2, 5, 8, 1, 4, 7, 10, 2, 5, 8, 1, 4, 7, 10, 3, 6, 9)
bb <- c(-1.72, -1.85, 0.98, 0.07, 1.00, 0.13, -0.43, -0.29, 0.86, 1.26)
```

```
aa <- c(1.28, 0.78, 0.98, 1.21, 0.83, 1.01, 0.92, 0.76, 0.88, 1.11)
cc <- rep(0, 10)
dd <- list(c(0, 0, -0.13, 0, -0.19, 0, 0, 0, 0, 0),
           c(0, 0, 0.13, 0, 0.19, 0, 0, 0, 0, 0))
response_gen(subject = s_id, item = i_id, theta = rnorm(20, 0, 1),
             b_par = bb, a_par = aa, c_par = cc, d_par = dd)
```

rzeropois

Generate data from a Zero-truncated Poisson

Description

Random generation of one observation of a random variable distributed as a Zero-truncated Poisson

Usage

```
rzeropois(lambda)
```

Arguments

lambda corresponds to the lambda parameter of a Poisson

Details

The zero-truncated Poisson (a.k.a. conditional Poisson or positive Poisson) distribution is a discrete probability distribution whose support is the set of positive integers.

sample_from

Sample from population structure

Description

Generates a sample from a population structure

Usage

```
sample_from(N, n, labels = names(N), verbose = TRUE)
```

Arguments

N list containing the population sampling structure

n numeric vector with the number of sampled observations (clusters or subjects) on each level

labels character vector with the names of the questionnaire respondents on each level

verbose if TRUE, prints output messages

sample_within_range	<i>Sample from range</i>
---------------------	--------------------------

Description

Creates a uniformly-distributed sample from a 2-length vector

Usage

```
sample_within_range(rg, sample_size = NULL, seed = NULL)
```

Arguments

rg	a "range"-class vector
sample_size	the size of the sample to be generated
seed	pseudo-random number generator seed

Value

A vector containing the generated sample

Note

This function was created primarily to be used to expand an object with the "range" class.

select	<i>Transform regular vector into selection vector</i>
--------	---

Description

Attaches a "select" class to a vector

Usage

```
select(...)
```

Arguments

...	parameters to be passed to c()
-----	--------------------------------

Value

same as x, but with a class attribute that classifies x as "select"

Note

This function was created to be used instead of c() in the n argument of cluster_gen.

split_cat_prop	<i>Split variables in cat_prop</i>
----------------	------------------------------------

Description

Split variables in cat_prop

Usage

```
split_cat_prop(cat_prop, keepYX = FALSE)
```

Arguments

cat_prop	list corresponding to cat_prop from questionnaire_gen
keepYX	if TRUE, output will be a list separating cat_prop_YX and cat_prop_W. IF FALSE, it will be a list with these objects combined (just like cat_prop)

summary.lsasimcluster	<i>Summarizes clusters</i>
-----------------------	----------------------------

Description

Takes the output of cluster_gen and creates summary statistics of the questionnaire variables

Usage

```
## S3 method for class 'lsasimcluster'
summary(
  object,
  digits = 4,
  print = "partial",
  print_hetcor = TRUE,
  force_matrix = FALSE,
  ...
)
```

Arguments

object	output of cluster_gen
digits	loosely controls the number of digits (significant or not) in the output (for print = TRUE)
print	"all" will pretty-print a summary of statistics, "partial" will only print cluster-level summaries; "none" outputs statistics as a list
print_hetcor	if TRUE (default), prints the heterogeneous correlation matrix
force_matrix	if TRUE, prints the heterogeneous correlation matrix even if warnings are generated
...	additional arguments (unused; added for compatibility with generic)

Value

list of summaries

Note

Setting `print="none"` allows for saving the results as an R object (list). Otherwise, the results will be simply printed and not savable.

Changing `digits` may yield unexpected results for the estimates of continuous variables, given how most of them are printed using the number of significant digits (for more information, see `help("summary")`).

Please note that datasets containing large values for the coefficient of variation (σ / μ) should yield imprecise results.

See Also

[anova.lsasimcluster\(\)](#)

Examples

```
n <- c(3, 30)
cls <- cluster_gen(n, n_X = 3, n_W = 5)
summary(cls)
summary(cls, print="none") # allows saving results
```

summary_2

Dataset summary statistics

Description

Creates summary statistics of a dataset

Usage

```
summary_2(data, digits = 3)
```

Arguments

<code>data</code>	Data frame
<code>digits</code>	number of digits for the output

Note

This function is inspired by `base::summary()`, but outputs content more relevant to the context of `cluster_gen()` and `summary()`

See Also

[base::summary\(\)](#)

trim_sample	<i>Trim sample</i>
-------------	--------------------

Description

Makes sure $n \leq N$

Usage

```
trim_sample(n, N)
```

Arguments

n	vector or non-ranged list corresponding to sample structure
N	vector or non-ranged list corresponding to population structure

See Also

[cluster_gen\(\)](#)

validate_questionnaire_gen	<i>Wrapper-functions for check_condition</i>
----------------------------	--

Description

functions to save space in their parent functions by moving the validation checks here

Usage

```
validate_questionnaire_gen(  
  n_cats,  
  n_vars,  
  n_X,  
  n_W,  
  theta,  
  cat_prop,  
  cor_matrix,  
  cov_matrix,  
  c_mean,  
  c_sd  
)
```


Arguments

n_cats	vector with number of categories for each categorical variable (W)
n_vars	number of variables (Y, X and W)
n_X	number of continuous background variables (X)
n_W	number of categorical variables (W)
theta	is there a latent variable (Y)?
cat_prop	list of vectors with the cumulative proportions of the background variables
cor_matrix	correlation matrix of YXW
cov_matrix	covariance matrix of YXW
c_mean	vector of means of all variables (YXW)
c_sd	vector of standard deviations of all variables (YXW)

weight_responses	<i>Weight responses</i>
------------------	-------------------------

Description

calculates sampling weights for the questionnaire responses

Usage

```
weight_responses(
  cluster_bg,
  n_obs,
  N,
  lvl,
  sublvl,
  previous_sublvl,
  sampling_method,
  cluster_labels,
  resp_labels,
  sum_pop,
  verbose
)
```

Arguments

cluster_bg	dataset with background questionnaire
n_obs	list with the number of elements per level
N	list of numeric vector with the population size of each <i>sampled</i> cluster element on each level
lvl	number of the current level
sublvl	number of the current sub-level (element within level)

previous_sublvl	number of the sub-level of the parent level
sampling_method	can be "SRS" for Simple Random Sampling or "PPS" for Probabilities Proportional to Size
cluster_labels	character vector with the names of each cluster level
resp_labels	character vector with the names of the questionnaire respondents on each level
sum_pop	total population at each level (sampled or not)
verbose	if TRUE, prints output messages

Value

Input data frame (cluster_bg) with three new columns for the sampling weights.

whitelist_message	<i>Whitelist message</i>
-------------------	--------------------------

Description

Prints out the sampled elements when cluster_gen is called with select. This function is analogous to cluster_message, but is more proper for random sampling.

Usage

```
whitelist_message(w)
```

Arguments

w whitelist

Index

* datasets

- pisa2012_math_block, 36
- pisa2012_math_booklet, 37
- pisa2012_math_item, 37
- pisa2012_q_cormat, 38
- pisa2012_q_marginal, 39
- .onAttach, 3

- anova.lsasimcluster, 4
- anova.lsasimcluster(), 13–15, 55
- attribute_cluster_labels, 5

- base::summary(), 55
- beta_gen, 5
- beta_gen(), 46
- block_design, 7
- booklet_design, 8
- booklet_sample, 9
- brr, 10
- brr(), 33, 50

- calc_n_tilde, 11
- calc_replicate_weights, 12
- calc_se_rho, 13
- calc_var_between, 13
- calc_var_tot, 14
- calc_var_within, 15
- check_condition, 15
- check_condition(), 17
- check_ignored_parameters, 16
- check_n_N_class, 16
- check_valid_structure, 17
- cluster_gen, 17
- cluster_gen(), 5, 12, 16, 22, 23, 31, 56
- cluster_gen_separate, 20
- cluster_gen_separate(), 20, 23
- cluster_gen_together, 22
- cluster_gen_together(), 20, 22
- cluster_message, 24
- convert_vector_to_list, 24

- cor_gen, 25
- cov_gen, 26
- cov_yfz_gen, 26
- cov_yxw_gen, 27
- cov_yxz_gen, 27
- customize_summary, 28

- draw_cluster_structure, 29

- gen_cat_prop, 29
- gen_variable_n, 30
- gen_X_W_cluster, 30

- intraclass_cor, 31
- irt_gen, 31
- item_gen, 32

- jackknife, 33
- jackknife(), 11, 12, 50

- label_respondents, 34
- lambda_gen, 34
- lsasim, 35
- lsasim-package (lsasim), 35

- pisa2012_math_block, 36
- pisa2012_math_booklet, 37
- pisa2012_math_item, 37
- pisa2012_q_cormat, 38
- pisa2012_q_marginal, 39
- pluralize, 40
- print_anova, 41
- proportion_gen, 42
- pt_bis_conversion, 42

- questionnaire_gen, 43
- questionnaire_gen(), 7, 20
- questionnaire_gen_family, 47
- questionnaire_gen_polychoric, 48

- ranges, 48

recalc_final_weights, 49
replicate_var, 49
response_gen, 50
rzeropois, 52

sample_from, 52
sample_within_range, 53
select, 53
split_cat_prop, 54
stats::anova(), 41
summary(), 28
summary.lsasimcluster, 54
summary.lsasimcluster(), 4, 11, 28, 31
summary_2, 55

trim_sample, 56

validate_questionnaire_gen, 56

weight_responses, 57
whitelist_message, 58