Matta *et al. Large-scale Assess Educ* (2018) 6:15
https://doi.org/10.1186/s40536-018-0068-8

Large-scale Assessments in Education

SOFTWARE ARTICLE Open Access

lsasim: an R package for simulating large-scale assessment data

Tyler H. Matta[1,2*], Leslie Rutkowski[2,3], David Rutkowski[2,3] and Yuan-Ling Liaw[2]

*Correspondence:
tyler.matta@pearson.com
[2] Centre for Educational Measurement, University of Oslo, Oslo, Norway
Full list of author information is available at the end of the article

Abstract

This article provides an overview of the R package **lsasim**, designed to facilitate the generation of data that mimics a large scale assessment context. The package features functions for simulating achievement data according to a number of common IRT models with known parameters. A clear advantage of **lsasim** over other simulation software is that the achievement data, in the form of item responses, can arise from multiple-matrix sampled test designs. Furthermore, **lsasim** offers the possibility of simulating data that adhere to general properties found in the background questionnaire (mostly ordinal, correlated variables that are also related to varying degrees with some latent trait). Although the background questionnaire data can be linked to the test responses, all aspects of **lsasim** can function independently, affording researchers a high degree of flexibility in terms of possible research questions and the part of an assessment that is of most interest.

Introduction

Large-scale assessments in education

An important tool for monitoring educational systems around the world, international large-scale assessments (ILSAs) are cross-national, comparative studies of achievement. ILSAs are used to measure educational achievement in select content domains for representative samples of students enrolled in primary and secondary educational systems. The achievement tests that students take are intended to be an adequate representation of what students know and can do in the relevant content areas (e.g., math, science, and reading). And the results of these assessments are used to compare educational systems (usually, but not exclusively, countries) and to inform policy, practice, and educational research both nationally and internationally. In terms of numbers of participants, these studies have grown tremendously over the past few decades. Today, two-thirds of all countries with populations greater than 30,000 have participated in one or more international or regional large-scale assessments (Lockheed et al. 2015). Among the most well-known ILSAs are the Trends in International Mathematics and Science Study (TIMSS) and the Programme for International Student Assessment (PISA). On a 4 year cycle, beginning in 1995, TIMSS measures mathematics and science in a representative sample of fourth and eighth grade students. Starting in 2000 and every 3 years afterward, PISA assesses 15-year-olds enrolled in

school in math, science, and reading. Besides the subject assessment (e.g. math, science, and reading tests), these studies also solicit information from students, their teachers, principals, and their parents regarding beliefs, attitudes, experiences, and the context of schooling. With over half a million students from 70 educational systems taking part, PISA is now the largest such study (OECD 2017). New versions of PISA, such as PISA for Development, targeting developing economies, and PISA for schools, focused on providing participating schools with internationally comparable results, will only increase these numbers and bring international assessments to new contexts and audiences. The quantitative nature, scale, and scope of these and related modern educational surveys necessitates a fairly sophisticated approach to the survey design, sampling, analysis, and reporting. We elaborate subsequently.

The first such study, the Pilot Twelve-Country Study (Foshay et al. 1962) was revolutionary at the time—before email, the Internet, and fast, desktop computers. By modern standards of educational and psychological measurement AERA, APA, and NCME (2014), however, the study was primitive, using classical test theory methods to uncover data of relatively low quality on a single test form. Beginning with the 1995 cycle of TIMSS, international assessments adopted an item response theory-based approach to scaling and a sophisticated booklet design, referred to as multiple matrix sampling (MMS) (Shoemaker 1973). Essentially, MMS is a method that divides test items into non-overlapping blocks that are then assembled into booklets according to, typically, a variant of a balanced incomplete block design (Gonzalez and Rutkowski 2010; Rutkowski et al. 2014). The result is, in the case of most international large-scale assessments (ILSAs), 10 or more hours of testing content delivered in two-hour booklets (Martin et al. 2016; OECD 2017). A concrete example, located in Table 1, is the 2011 Trends in International Mathematics and Science Study (TIMSS) design that distributed 429 total mathematics and science items across 14 non-overlapping mathematics blocks and 14 non-overlapping science blocks.

Table 1 2011 TIMSS booklet design

Booklet	Part 1		Part 2	
	Block 1	Block 2	Block 3	Block 4
1	M01	M02	S01	S02
2	S02	S03	M02	M03
3	M03	M04	S03	S04
4	S04	S05	M04	M05
5	M05	M06	S05	S06
6	S06	S07	M06	M07
7	M07	M08	S07	S08
8	S08	S09	M08	M09
9	M09	M10	S09	S10
10	S10	S11	M10	M11
11	M11	M12	S11	S12
12	S12	S13	M12	M13
13	M13	M14	S13	S14
14	S14	S01	M14	M01

Only a fraction of the students in the sample take any one item, and any selected student takes only a fraction of the total available items. As a result, the actual distribution of student proficiency cannot be approximated by its empirical estimate (Mislevy et al. 1992b). Further, traditional methods of estimating individual achievement introduce an unacceptable level of uncertainty and the possibility of serious aggregate-level bias (Little and Rubin 1983; Mislevy et al. 1992a). As one means for overcoming the methodological challenges associated with multiple-matrix sampling, large-scale assessment programs adopted a population or latent regression modeling approach that uses marginal estimation techniques to generate population- and subpopulation-level achievement estimates (Mislevy 1991; Mislevy et al. 1992a, b)

More specifically, using information from background questionnaires, other demographic variables of interest and responses to the cognitive portion of the test, student achievement is estimated via a latent regression model, where achievement (θ) is treated as a latent or unobserved variable for all examinees. Essentially, the limited achievement test responses, complete student background questionnaires responses, and select demographic information are used in conjunction with a measurement model-based extension of Rubin's (1987) multiple imputation approach to generate a proficiency distribution for the population (or sub-population) of interest (Beaton and Johnson 1992; Mislevy et al. 1992a, b; von Davier et al. 2006). A short, slightly more technical description follows.

As in multiple imputation methods, an imputation model (called a "conditioning model") is used to derive posterior distributions of student achievement. This model uses all available student data (cognitive as well as background information) to generate a conditional proficiency distribution from which to draw a number of plausible values (usually five) for each student on each latent trait (e.g. mathematics, science, and associated sub-domains).

Because θ is a latent, unobserved variable for every examinee, it is reasonable to treat it as a missing value and to approximate statistics involving θ by its expectation. That is, for any statistic, t, $\hat{t}(\mathbf{X},\mathbf{Y}) = E[t(\theta),\mathbf{Y}|\mathbf{X},\mathbf{Y}] = \int t(\theta,\mathbf{Y})p(\theta|\mathbf{X},\mathbf{Y})d\theta$ where $\mathbf{X}$ is a matrix of achievement item responses for all examinees and $\mathbf{Y}$ is the matrix of responses of all examinees to the set of administered background questions. Because closed-form solutions are typically not available, random draws from the conditional distributions are drawn for each sampled examinee j (Mislevy et al. 1992b). In line with missing data practices (Rubin 1976, 1987), values for each examinee are drawn multiple times. These are typically referred to as plausible values in LSA terminology or multiple imputations in missing data literature. Using Bayes' theorem and the IRT assumption of conditional independence,

$$p(\theta|\mathbf{x}_j,\mathbf{y}_j) \propto P(\mathbf{x}_j|\theta,\mathbf{y}_j)p(\theta|\mathbf{y}_j) = P(\mathbf{x}_j|\theta)p(\theta|\mathbf{y}_j)\,, \tag{1}$$

where $P(\mathbf{x}_j|\theta)$ is the likelihood function for θ and $p(\theta|\mathbf{y}_j)$ is the distribution of θ for a given vector of response variables. Usually, it is assumed that θ is normally distributed according to the following model

$$\theta_j = \mathbf{\Gamma}'\mathbf{y}_j + \epsilon_j \tag{2}$$

where $\epsilon_j \sim \mathcal{N}(0,\mathbf{\Sigma})$ and $\mathbf{\Gamma}$ and $\mathbf{\Sigma}$ have to be estimated.

The role of simulations in large-scale assessment research and development

In the past 20 years, national and international assessments have expanded significantly in terms of the number of national- or system-level participants, platforms (computerized in addition to paper and pencil), content domains (e.g., collaborative problem solving), and the degree to which participating countries differ in economic, cultural, linguistic, geographic, and other terms. To that end, two areas of research in large-scale assessment are evaluating the performance of currently used methods given the changing nature of LSAs and developing new methods. In both cases, areas of emphasis can conceivably include test design, administration, data collection, sampling, or other relevant areas. As study administrators are naturally cautious about implementing new designs and methods without evidence of their merit and worth, a viable option for testing new methods is through simulation. Further, using empirical data to examine the performance of current and new methods is limited by the fact that we can never know the true, underlying population values of item- or person-parameters. Simulation is a low-cost powerful means for conducting methodological research in the area of large-scale assessment. Examples include Adams et al. (2013), Rutkowski and Zhou (2015).

Traditionally, the mandate of large-scale assessments surrounds measuring and reporting achievement across populations of interest. As such, large-scale assessment developers prioritize achievement measures, in terms of framework development, psychometric quality, analysis, and reporting (OECD 2014; Martin et al. 2016). Nevertheless, background questionnaires serve to contextualize educational achievement and provide opportunities to understand correlates of learning. To that end, the background questionnaire and achievement measures have distinct frameworks, and different teams work to develop and innovate in each respective area. This (in some cases arbitrary) distinction between the achievement test and background questionnaires frequently leads researchers to regard each component separately for many methodological investigations. Therefore, **lsasim** (Matta et al. 2017) simulates data in a way that treats background questionnaire responses as separate from but related to the achievement test.

Software for generating large-scale assessment data

The goal of **lsasim** is to provide a set of functions that enable users to design and modify test designs that are commonly utilized in large-scale educational surveys. Such goals are similar to the goals of **catR** (Magis and Raiche 2012) for generating item response patterns from computer adaptive tests and **mstR** (Magis et al. 2017) for generating item response patterns from multi-stage tests. The difference, however, is that multi-matrix sampling designs utilized in large-scale assessments are not (yet) adaptive, and can thus, depend on other packages to estimate item parameters and achievement estimates.

Although generation of item responses, given a set of item parameters and a true score, for a fixed test is not unique, none of the existing **R** (R Core Team 2017) IRT packages provide a means to establish multi-matrix sampling designs. Two of the most commons IRT packages used are **TAM** (Robitzsch et al. 2017) and **mirt** (Chalmers 2012). Both packages include functions to simulate item response patterns, but every "observation" will be given a generated response to every item. Users would have to delete item responses post-hoc to arrive at data that resembles a matrix sampling design.

In addition to the inability to generate item responses under a multi-matrix sampling designs, none of the IRT packages reviewed provide a means for generating responses to "background questionnaires," data that are commonly used in the estimation of achievement. To include responses to background variables, one would need to develop functions on their own, or utilized an alternative package to generate mixed normal, bivariate, and ordinal data such as **GenOrd** (Barbiero and Ferrari 2015).

With **lsasim** designed to generate item responses, it has no functionality to estimate item parameters or achievement estimates. For this, users should turn to existing packages, for example, **TAM**, as is demonstrated later in this article. The data output from **lsasim** is formatted to be used with **TAM** or **mirt** without any further data manipulation. Furthermore, the **ibd** package (Mandal 2018) can be used in tandem with **lsasim** to generate balanced incomplete designs.

Simulation methodology

Generating correlated questionnaire data

Let $X = \{X_1, X_2, \ldots, X_p, \ldots, X_P\}$ be a set of continuous random variables and $W = \{W_1, W_2, \ldots, W_q, \ldots, W_Q\}$ be a set of ordinal (possibly dichotomous) random variables. For any W_q, let there be $1, \ldots, k_q, \ldots, K_q$ ordered response categories where $p(W_q = k_q) = \pi_{q,k}$ such that $\sum_{1 \le k \le K} \pi_{q,k} = 1$. Furthermore, let $\mathbf{R}$, be a $(P+Q) \times (P+Q)$ possibly heterogeneous correlation matrix which includes (a) Pearson product-moment correlations for $\rho(X_p, X_{p'})$; (b) polychoric correlations for any $\rho(W_q, W_{q'})$; and (c) polyserial correlation for any $\rho(X_p, W_q)$.

Often in the development of psychological tests, items are designed with the assumption that ordinal item responses map to a continuous latent trait. Thus, we can specify an underlying continuous variable, $W_q^\star$, for any ordinal variable, W_q. The relationship between W_q and $W_q^\star$ is

$$W_q = \begin{cases} 1 & \text{if}, \quad -\infty < W_q^\star \le \alpha_{q,1} \\ 2 & \text{if}, \quad \alpha_{q,1} < W_q^\star \le \alpha_{q,2} \\ \vdots & \\ k & \text{if}, \quad \alpha_{q,k-1} < W_q^\star \le \alpha_{q,k} \\ \vdots & \\ K & \text{if}, \quad \alpha_{q,K-1} < W_q^\star \le \infty\,. \end{cases} \tag{3}$$

where $\alpha_{q,k}$ is the kth threshold for $W_q^\star$, delineating responses $k-1$ and k on the scale of $W_q^\star$.

To simulate correlated mixed-type data, we need only a $(P+Q) \times (P+Q)$ data-generating correlation matrix, $\mathbf{R}$ and the K_q marginal probabilities π_q corresponding to each ordinal variable W_q. First, generate N replicates from $P+Q$ independent standard normal random variables $\mathbf{Z} = \{Z(X_1), \ldots, Z(X_P), \ldots, Z(W_1^\star), \ldots, Z(W_Q^\star)\}$, such that an $N \times (P+Q)$ data matrix, $\mathbf{Z}$, is obtained. Second, let $\mathbf{L}$ be the lower triangle matrix of the Cholesky factorization of $\mathbf{R}$ where $\mathbf{R} = \mathbf{L}\mathbf{L}'$. We can transform $\mathbf{Z}$ to $\{\mathbf{X}, \mathbf{W}^\star\}$ using $\mathbf{L}$ such that $\{\mathbf{X}, \mathbf{W}^\star\} = \mathbf{Z}\mathbf{L}$. Finally, we transform the latent variables $\mathbf{W}^\star$ to $\mathbf{W}$ by coarsening based on Eq. 3.

Generating IRT-based data

In order to generate data from many models in the IRT-family, we specify an overly general model accompanied by explicit constraints, given the generating information. The general model combines the three-parameter model for dichotomous item responses and the generalized partial credit model for ordered responses,

$$p(u_{ij} = k|\theta_j) = c_i + (1 - c_i)\frac{\exp\left[\sum_{u=1}^{k} Da_i(\theta_j - b_i + d_{iu})\right]}{\sum_{v=1}^{K_i}\exp\left[\sum_{u=1}^{v} Da_i(\theta_j - b_i + d_{iu})\right]} \tag{4}$$

where k is the response to item i by respondent j, θ_j is the respondent's true score, and K_i is the maximum score on item i. Furthermore, b_i is the average difficulty for item i, d_{iu} is the threshold parameter between scores u and $u-1$ for item i, a_i is the item's discrimination parameter, c_i is the item's pseudo-guessing parameter, and D is a scaling constant for the item.

Because the partial credit model does not, generally, include a guessing parameter, we place constraints on parts of the model given the known parameters. Namely, when multiple thresholds are specified for a given item, $u > 1$, the pseudo-guessing parameter is constrained to zero, $c = 0$, resulting in the generalized partial credit model. Further constraining $a = 1$ results in the partial credit model. When only one threshold is specified, Eq. 4 reduces to

$$p(y_{ij} = k|\theta_j) = c_i + (1 - c_i)\frac{\exp\left[Da_i(\theta_j - b_i)\right]}{1 + \exp\left[Da_i(\theta_j - b_i)\right]}. \tag{5}$$

From here, constraining $c = 0$ results in the two-parameter item response model and constraining $c = 0, a = 1$ results in the Rasch model.

The lsasim package

The **lsasim** package contains a set of functions that facilitate the generation of large-scale assessment data. The package can be divided into two interrelated sets of functions: one set for generating background questionnaire data and another set for generating the cognitive data. This section provides a description of each function within the package and demonstrates how they can be used. To start, we set a seed for replicability purposes and load the **lsasim** package and the **polycor** package (Fox 2016), both available on CRAN.

```
set.seed(48523)

library(lsasim)

library(polycor)
```

Background questionnaire data

The main function for generating background questionnaire data is aptly named `questionnaire_gen`. The function facilitates the generation of correlate continuous, binary, and ordinal data by specifying the cumulative proportions for each background item and a correlation matrix.

```
questionnaire_gen(n_obs, cat_prop, cor_matrix, c_mean = NULL, c_sd = NULL,
  theta = FALSE)
```

The `n_obs` argument specifies the number of observations (examinees) to generate. The argument `cat_prop` takes a list of vectors, each of which contain the cumulative proportions for a given background item. The length of the list, that is, the number of vectors within the list, indicates the number of background items to be generated. Each vector in the list should end with 1 such that the length of each vector specifies the number of response categories for that background item. For continuous items, the vector should contain one element, 1.

```
ex_prop <- list(1, c(0.23, 0.54, 0.81, 1))
print(ex_prop)

## [[1]]
## [1] 1
##
## [[2]]
## [1] 0.23 0.54 0.81 1.00
```

The code above provides an example of cumulative proportions for two background items. The first background variable has one category, indicating a continuous response. The second background item has four response categories with marginal population proportions of 0.23, 0.31, 0.27, and 0.19, respectively.

The argument `cor_matrix` takes a possibly heterogeneous correlation matrix, consisting of Pearson correlations between numeric variables, polyserial correlations between numeric and ordinal variables, and polychoric correlations between ordinal variables. That is, all discrete variables are assumed to have an underlying continuous latent variable.

```
ex_cor <- matrix(c(1, 0.7, 0.7, 1), nrow = 2, byrow = TRUE)
print(ex_cor)

##      [,1] [,2]
## [1,]  1.0  0.7
## [2,]  0.7  1.0
```

In the above example, we specify a polyserial correlation of .7 between the discrete background item and the continuous item. Notice that the size of `ex_cor` is equal to the length of the `ex_prop` as the size and order of `cor_matrix` corresponds to `cat_prop`.

Using `ex_prop` and `ex_cor`, we simulate one dataset with 1000 observations. By default, continuous variables are generated from standard normals, $\mathcal{N}(0, 1)$.

```
ex_background_data <- lsasim::questionnaire_gen(n_obs = 1000,
                                                cat_prop = ex_prop,
                                                cor_matrix = ex_cor)


str(ex_background_data)

## 'data.frame': 1000 obs. of  3 variables:
##  $ subject: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ q1     : num  0.833 -0.842 0.115 0.167 -0.83 ...
##  $ q2     : Factor w/ 4 levels "1","2","3","4": 3 1 3 1 1 3 1 4 3 4 ...
```

Notice the continuous variable `q1` is a numeric variable and the discrete variable `q2` is a factor with four levels. A third variable, `subject`, is the unique identifier for each observation. With our simulated data set, we see the first two moments of `q1` and the marginal proportions for `q2` are both well-recovered.

```
##      mean   sd
## [1,]    0 1.03
```

```
##    1    2    3    4
## 0.25 0.53 0.79 1.00
```

Additionally, the polyserial correlation is also well-recovered using the `hetcor` function from the **polycor** package.

```
## [1] 0.73
```

It is important to note that converting the factor variables to numeric and estimating a Pearson correlation will not recover the generating correlation matrix.

The `questionnaire_gen` function includes three optional arguments, `c_mean`, `c_sd`, and `theta`. The arguments `c_mean` and `c_sd` are used to scale the continuous variables where `c_mean` takes a vector of means for those continuous variables

and `c_sd` takes a vector of standard deviations. The lengths of `c_mean` and `c_sd` are equal to the number of continuous items to be generated. Specification of `c_mean` and/or `c_sd` results in a continuous variables i to be distributed $\mathcal{N}$(`c_mean[i]`, `c_sd[i]`). Finally, `theta` is a logical argument where `theta = TRUE` results in the first continuous background item to be named "theta" in the resulting data frame. This optional argument is only for convenience when generating both background questionnaire data and cognitive data.

```
ex_background_data2 <- lsasim::questionnaire_gen(n_obs = 1000,
                                                 cat_prop = ex_prop,
                                                 cor_matrix = ex_cor,
                                                 c_mean = 3,
                                                 c_sd = 1.5,
                                                 theta = TRUE)


str(ex_background_data2)

## 'data.frame': 1000 obs. of  3 variables:
##  $ subject: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ theta  : num  4.78 3.61 5.29 5.37 3.45 ...
##  $ q1     : Factor w/ 4 levels "1","2","3","4": 3 4 2 2 2 2 2 1 2 2 ...
```

```
##      mean   sd
## [1,] 2.98 1.52
##    1    2    3    4
## 0.24 0.52 0.81 1.00
## [1] 0.71
```

Notice in the example above, the continuous variable is now named `theta` and has a mean and standard deviation close to that specified by `c_mean` and `c_sd`.

When the background questionnaires are not of substantive importance, **lsasim** provides two functions for generating the correlation matrix and list of marginal cumulative proportions. The `cor_gen` function generates a random correlation matrix by specifying the number of variables via the `n_var` argument.

```
ex_cor_gen <- lsasim::cor_gen(n_var = 6)
round(ex_cor_gen, 2)

##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
## [1,]  1.00 -0.40  0.67  0.69  0.73  0.19
## [2,] -0.40  1.00 -0.25 -0.58 -0.42 -0.07
## [3,]  0.67 -0.25  1.00  0.30  0.88  0.19
## [4,]  0.69 -0.58  0.30  1.00  0.68  0.02
## [5,]  0.73 -0.42  0.88  0.68  1.00  0.13
## [6,]  0.19 -0.07  0.19  0.02  0.13  1.00
```

The `proportion_gen` function generates a list of random cumulative proportions using two arguments. The first argument `cat_options` takes a vector whose entries specify the types of items to be generated. In the code below, `cat_options = c(1, 2, 3)` specifies continuous, two-category, and three-category item types to be generated. The second argument, `n_cat_options` is a vector of equal length to `cat_options`, which specifies the number of each item type to be generated. Below, `n_cat_options = c(3, 2, 1)` indicates that there will be three continuous items, two two-category items, and one three-category item generated (six items in all).

```
ex_prop_gen <- lsasim::proportion_gen(cat_options = c(1, 2, 3),
                                      n_cat_options = c(3, 2, 1))

str(ex_prop_gen)

## List of 6
##  $ : num 1
##  $ : num 1
##  $ : num 1
##  $ : num [1:2] 0.33 1
##  $ : num [1:2] 0.95 1
##  $ : num [1:3] 0.13 0.82 1
```

Both the random correlation matrix, `ex_cor_gen` and the random marginal proportions, `ex_prop_gen`, can then be used by `questionnaire_gen`. We will generate background data for 40 examinees to be used later in this section.

```
ex_background_data3 <- lsasim::questionnaire_gen(n_obs = 40,
                                                 cat_prop = ex_prop_gen,
                                                 cor_matrix = ex_cor_gen)

str(ex_background_data3)

## 'data.frame': 40 obs. of  7 variables:
##  $ subject: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ q1     : num  0.511 1.793 0.98 -0.712 -0.596 ...
##  $ q2     : num  -0.231 -0.474 -0.738 -0.309 0.872 ...
##  $ q3     : num  0.0745 2.2196 0.0956 -2.0868 -0.9192 ...
##  $ q4     : Factor w/ 2 levels "1","2": 2 2 2 2 1 1 2 2 2 2 ...
##  $ q5     : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ q6     : Factor w/ 3 levels "1","2","3": 2 2 2 2 2 2 2 2 2 2 ...
```

Cognitive data

The cognitive assessments for LSAs are much more involved than the background questionnaire. As mentioned above, the cognitive assessments use an IRT measurement model administered using a multi-matrix sampling scheme. The package **lsasim** has been designed to provide researchers with extensive flexibility in varying these design features while providing valid default solutions as an option. There are five functions that make up the cognitive data generation, which we organize here under three categories (a) item parameter generation: `item_gen`; (b) test assembly and administration: `block_design`, `booklet_design`, and `booklet_sample`; and (c) item response generation: `response_gen`.

Item parameter generation

Although researchers may wish to use pre-determined item parameters, the `item_gen` function enables the flexible generation of item parameters from item response models.

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

The arguments `n_1pl`, `n_2pl`, and `n_3pl` specify how many one-, two-, and three-parameter items will be included in the item pool. For this example, we will generate five two-parameter items and ten three-parameter items. The argument `thresholds` specifies the number of thresholds for the one- and two-parameter. Specifying `thresholds = 2` will generate two threshold parameters for each of the five two-parameter items. Finally, the arguments `b_bounds`, `a_bounds`, and

`c_bounds` specify the bounds of the uniform distributions used to generate the b, a, and c parameters, respectively. Note that `a_bounds` are only applied to the two- and three-parameter items and `c_bounds` are applied to three-parameter items only.

```
item_pool <- lsasim::item_gen(n_2pl = 5, n_3pl = 10, thresholds = 2,
                              b_bounds = c(-2, 2), a_bounds = c(.75, 1.25),
                              c_bounds = c(0, .25))

print(item_pool)
```

```
##    item     b    d1   d2    a    c k p
## 1     1 -0.70 -0.65 0.66 0.78 0.00 2 2
## 2     2  0.01 -0.44 0.44 0.80 0.00 2 2
## 3     3 -1.45 -0.33 0.34 1.04 0.00 2 2
## 4     4 -0.80 -0.63 0.62 0.99 0.00 2 2
## 5     5 -0.20 -0.38 0.37 0.89 0.00 2 2
## 6     6  1.48  0.00 0.00 1.05 0.03 1 3
## 7     7 -1.78  0.00 0.00 0.86 0.21 1 3
## 8     8  1.63  0.00 0.00 1.17 0.14 1 3
## 9     9  1.63  0.00 0.00 1.04 0.18 1 3
## 10   10 -1.42  0.00 0.00 1.17 0.12 1 3
## 11   11  1.86  0.00 0.00 0.88 0.16 1 3
## 12   12 -1.63  0.00 0.00 0.78 0.15 1 3
## 13   13  2.14  0.00 0.00 1.01 0.21 1 3
## 14   14  1.96  0.00 0.00 0.78 0.22 1 3
## 15   15  0.42  0.00 0.00 0.78 0.05 1 3
```

The above example shows the item information for the 15 items in `item_pool`. All 15 items have a b_i parameter, which is the average difficulty for the item. The five two-parameter items were specified as generalized partial credit items with two thresholds. Thus, item 1 though item 5 have two d parameters, `d1` and `d2` such that $b_i + d_{ik}$ is the kth threshold for item i. All 15 items have a discrimination parameter, a_i, while only item 6 through item 15 have a c parameter (pseudo-guessing). The last two variables in `item_pool`, `k` and `p`, are indicators to identify the number of thresholds and whether the item is from a 1PL, 2PL, or 3PL model, respectively.

Test assembly and administration

Large scale assessments are typically designed such that items are compiled into blocks or clusters, which are then assembled into test booklets. The goal is to develop

non-overlapping blocks of items that can be assembled into test booklets. Two functions, `block_design` and `booklet_design`, facilitate the assembly of the test booklets while one function, `booklet_sample` facilitates the administration of the booklets to subjects. The test assembly process was split into two functions to provide users ample flexibility of how tests are constructed.

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


booklet_design(item_block_assignment, book_design = NULL)


booklet_sample(n_subj, book_item_design, book_prob = NULL, resample = FALSE,
  e = .1, iter = 20)
```

Block design

The first step in the test assembly is to determine the number of blocks and the assignment of items to those blocks. The function `block_design` facilitates this process with two required arguments and one optional argument. The `n_blocks` argument specifies the number of blocks while the `item_parameters` argument takes a data frame of item parameters. The default allocation of items to blocks is a spiraling design. For $1, 2, \ldots, H$ item blocks, the first item is assigned to block 1, item 2 is assigned to the block 2, and item H is assigned to block H. The process is continued such that item $H + 1$ is assigned to block 1, item $H + 2$ is assigned to block 2 and item $H + H$ is assigned to block H until all items are assigned to a block.

The default functionality of the block design is illustrated by administering the 15 items in `item_pool` across 4 blocks.

```
block_ex <- lsasim::block_design(n_blocks = 4, item_parameters = item_pool,
                                 item_block_matrix = NULL)


str(block_ex)

## List of 2
##  $ block_assignment  : num [1:4, 1:4] 1 5 9 13 2 6 10 14 3 7 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "i1" "i2" "i3" "i4"
##   .. ..$ : chr [1:4] "b1" "b2" "b3" "b4"
##  $ block_descriptives: num [1:4, 1:2] 4 4 4 3 0.718 0.507 -0.238 -0.267
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "b1" "b2" "b3" "b4"
##   .. ..$ : chr [1:2] "block length" "average difficulty"
```

The function `block_design` produces a list that contains two elements. The first element, `block_assignment`, is a matrix that that identifies which items from the item pool correspond to which block.

```
print(block_ex$block_assignment)

##    b1 b2 b3 b4
## i1  1  2  3  4
## i2  5  6  7  8
## i3  9 10 11 12
## i4 13 14 15  0
```

The column names of `block_assignment` begin with `b` to indicate block while the rows begin with `i` to indicate item. For block `b1`, the first item, `i1`, is the first item from `item_pool`, the second item, `i2`, is the fifth item from `item_pool`, the third item, `i3`, is the ninth item from `item_pool`, and the fourth item, `i4`, is the 13th item from `item_pool`. Because the 15 items do not evenly distribute across 4 blocks, the fourth block only contains three items. To avoid dealing with ragged matrices, all shorter blocks are filled with zeros.

The second element in `block_ex` is a table of descriptive statistics for each block.

```
print(block_ex$block_descriptives)

##    block length average difficulty
## b1            4              0.718
## b2            4              0.507
## b3            4             -0.238
## b4            3             -0.267
```

This table indicates the number of items in each block and the average difficulty for each block. Again, notice blocks 1 though 3 each have four items while block 4 only has three items. Furthermore, the easiest block is `b4` with an average difficulty of -0.267 while the most difficult block is `b1` with an average difficulty of 0.718. Note that for partial credit items, b_i is used in the calculation of the average difficulty.

For user-specified block construction, we can specify an indicator matrix where the number of columns equals the number of blocks and the number of rows equals the number of items in the item pool.

```
item_block_design <- matrix(c(1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                              0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
                              0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
                              0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1),
                            ncol = 4, nrow = 15)
print(item_block_design)

##       [,1] [,2] [,3] [,4]
##  [1,]    1    0    0    0
##  [2,]    1    0    0    0
##  [3,]    1    0    0    0
##  [4,]    1    0    0    0
##  [5,]    0    1    0    0
##  [6,]    0    1    0    0
##  [7,]    0    1    0    0
##  [8,]    0    1    0    0
##  [9,]    0    0    1    0
## [10,]    0    0    1    0
## [11,]    0    0    1    0
## [12,]    0    0    1    0
## [13,]    0    0    0    1
```

The 1s indicate which items belong to which blocks. Given the example above, we will assign items 1 through 4 to block 1, items 5 though 8 to block 2, items 9 through 12 to block 3 and items 13 though 15 to block 4. Below, `block_ex2` demonstrates how the matrix is used with the `item_block_matrix` argument and the items in `item_pool`.

```
block_ex2 <- lsasim::block_design(n_blocks = 4, item_parameters = item_pool,
                                  item_block_matrix = item_block_design)
print(block_ex2)

## $block_assignment
##    b1 b2 b3 b4
## i1  1  5  9 13
## i2  2  6 10 14
## i3  3  7 11 15
## i4  4  8 12  0
##
## $block_descriptives
##    block length average difficulty
## b1            4             -0.735
## b2            4              0.282
## b3            4              0.110
## b4            3              1.507
```

Booklet design

After the items have been assigned to blocks, the blocks are then assigned to test booklets. The `booklet_design` function facilitates this process with one required argument and one optional argument. The `item_block_assignment` argument takes the `block_assignment` matrix from `block_design`. Like `block_design`, `booklet_design` provides a default spiraling booklet assembly method which is illustrated in Table 2.

```
book_ex <- lsasim::booklet_design(item_block_assignment = block_ex$block_assignment)

print(book_ex)

##    B1 B2 B3 B4
## i1  1  2  3  1
## i2  5  6  7  5
## i3  9 10 11  9
## i4 13 14 15 13
## i5  2  3  4  4
## i6  6  7  8  8
## i7 10 11 12 12
## i8 14 15  0  0
```

Table 2 Default booklet design

Booklet	Item blocks							
	b_1	b_2	b_3	b_4	$\cdots$	b_{H-2}	b_{H-1}	b_H
B_1	1	1	0	0	...	0	0	0
B_2	0	1	1	0	...	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
B_{N-1}	0	0	0	0	...	0	1	1
B_N	1	0	0	0	...	0	0	1

`book_ex` uses the default item-block assembly of `block_ex`. In the above output, book `B1` contains the items from block 1 and block 2, booklet `B2` contains the items from block 2 and block 3, booklet `B3` contains the items from block 3 and block 4, and book `B4` contains the items from block 1 and block 4. Notice that the first two test booklets contain eight items while the last two books contain seven items. This is because block 4 only has three items whereas blocks 1, 2, and 3 have four times. Like `block_design`, `booklet_design` avoids ragged matrices by filling shorter booklets with zeros.

Users can also explicitly specify the booklet assembly design using the optional argument `book_design`. By specifying a booklet design matrix, users can control which item blocks go to which booklets. The `book_design` argument takes an indicator matrix where the columns indicate the item blocks and the rows indicate the booklets. In the code below, we create a matrix, `block_book_design`, that will create six test booklets. Booklet 1 (row 1) will include items from blocks 1, 3 and 4 while booklet 6 (row 6) will include items from block 1 only.

```
block_book_design <- matrix(c(1, 0, 1, 1,
                              0, 1, 0, 0,
                              0, 0, 1, 1,
                              0, 0, 0, 1,
                              0, 1, 1, 1,
                              1, 0, 0, 0), ncol = 4, byrow = TRUE)
print(block_book_design)

##      [,1] [,2] [,3] [,4]
## [1,]    1    0    1    1
## [2,]    0    1    0    0
## [3,]    0    0    1    1
## [4,]    0    0    0    1
## [5,]    0    1    1    1
## [6,]    1    0    0    0
```

```
book_ex2 <- lsasim::booklet_design(
                      item_block_assignment = block_ex$block_assignment,
                      book_design = block_book_design)


print(book_ex2)
```

```
##     B1 B2 B3 B4 B5 B6
## i1   1  2  3  4  2  1
## i2   5  6  7  8  6  5
## i3   9 10 11 12 10  9
## i4  13 14 15  0 14 13
## i5   3  0  4  0  3  0
## i6   7  0  8  0  7  0
## i7  11  0 12  0 11  0
## i8  15  0  0  0 15  0
## i9   4  0  0  0  4  0
## i10  8  0  0  0  8  0
## i11 12  0  0  0 12  0
```

Notice how booklets `B1` and `B5`, contain 11 items each while booklets `B2` and `B6` contain four items each.

Booklet administration

The final component of the test assembly and administration is the administration of test booklets to examinees. The function `booklet_sample` facilitates the distribution of test booklets to examinees. The two required arguments are `n_subj`, the number of examinees, and `book_item_design`, the output from `booklet_design`. The default sampling scheme makes all books equally likely but the optional argument `book_prob` takes a vector of probabilities to make some books more or less likely to be administered. The logical argument `resample` will resample booklets until the difference in booklets sampled is less than `e` or `iter` attempts. The resampling functionality may be useful when `n_subj` is small and only one dataset is being generated.

Using `book_ex` from above, we administer the four books to 40 examinees.

```
book_admin <- lsasim::booklet_sample(n_subj = 40, book_item_design = book_ex)


head(book_admin, 10)
##    subject book item
## 1        1    2    2
## 2        1    2    6
## 3        1    2   10
## 4        1    2   14
## 5        1    2    3
## 6        1    2    7
## 7        1    2   11
## 8        1    2   15
## 9        2    4    1
## 10       2    4    5
```

The result is a data frame with three columns: subject, book, and item. The data frame is organized in a long (univariate) format where there is one row for each subject-item combination. The long format is required for generating item responses with the `response_gen` function. As can be seen in the output above, subject 1 has been administered booklet 2 while subject 2 has been administered booklet 4.

Item response generation

Recall that the `item_gen` function enables the user to specify different combinations of item response models for a given item pool. The `response_gen` function will generate item responses for all models given four required arguments and five optional arguments.

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

Both arguments `subject` and `item` take length-N vectors that provide the subject by item information where $N = \sum_{1:J} n_j$ where n_j is the number of items in the test for examinee j. The arguments `theta` takes a J-length vector of true latent proficiency values for the examinees, where J is the total number of examinees. Finally, because the simplest model is the one-parameter model, only the `b_par` argument is required for any items. The `b_par` argument takes an I-length vector of item difficulties where I is

the total number of items in the item pool. The optional arguments `a_par` and `c_par` also take I-length vectors for the corresponding item parameters, while `d_par` takes an I-length list where each element in the list is an $(K_i - 1)$-length vector containing the thresholds for each item. The argument `item_no` is used only when a subset of items are used from a given item pool. Finally, the argument `ogive` allows the user to omit the scaling constant for a logistic ogive (default) or to include a normal ogive, `ogive = "Normal"`.

Using the generated background data in `ex_background_data3`, generated item parameters in `item_pool`, and the booklet sampling information in `book_admin`, we generate item responses.

```
ex_cognative_data <- lsasim::response_gen(subject = book_admin$subject,
                                          item    = book_admin$item,
                                          theta   = ex_background_data3$q1,
                                          a_par   = item_pool$a,
                                          b_par   = item_pool$b,
                                          c_par   = item_pool$c,
                                          d_par   = list(item_pool$d1,
                                                         item_pool$d2))


str(ex_cognative_data)

## 'data.frame': 40 obs. of  16 variables:
##  $ i001   : num  NA 1 2 0 NA 1 NA NA 0 2 ...
##  $ i002   : num  0 NA NA 0 0 NA 0 NA NA NA ...
##  $ i003   : num  2 NA NA NA 1 NA 2 2 NA NA ...
##  $ i004   : num  NA 2 2 NA NA 2 NA 2 0 1 ...
##  $ i005   : num  NA 2 2 0 NA 2 NA NA 0 2 ...
##  $ i006   : num  0 NA NA 0 0 NA 0 NA NA NA ...
##  $ i007   : num  1 NA NA NA 1 NA 0 1 NA NA ...
##  $ i008   : num  NA 0 0 NA NA 0 NA 0 0 0 ...
##  $ i009   : num  NA 1 0 0 NA 0 NA NA 0 0 ...
##  $ i010   : num  1 NA NA 1 1 NA 1 NA NA NA ...
##  $ i011   : num  1 NA NA NA 0 NA 1 1 NA NA ...
##  $ i012   : num  NA 1 1 NA NA 1 NA 1 1 0 ...
##  $ i013   : num  NA 1 0 1 NA 0 NA NA 0 0 ...
##  $ i014   : num  0 NA NA 0 1 NA 1 NA NA NA ...
##  $ i015   : num  0 NA NA NA 1 NA 0 0 NA NA ...
##  $ subject: int  1 2 3 4 5 6 7 8 9 10 ...
```

The result of `response_gen` is a wide (multivariate) data frame where there is one row per examinee and each item is a column with the final variable indicating the subject ID. Because not every examinee sees every item, items not administered are considered missing.

Combining cognitive and background questionnaire data

At this point, we've generated correlated background data that includes a continuous true score (e.g., θ) and cognitive data. The **lsasim** package is designed such that both generated datasets have equal rows for easy merging using the `subject` variable in each data frame. The resulting dataset, `ex_full_data` is the generated large-scale assessment data.

```
ex_full_data <- merge(ex_background_data3, ex_cognative_data, by = "subject")
str(ex_full_data)

## 'data.frame': 40 obs. of  22 variables:
##  $ subject: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ q1     : num  0.511 1.793 0.98 -0.712 -0.596 ...
##  $ q2     : num  -0.231 -0.474 -0.738 -0.309 0.872 ...
##  $ q3     : num  0.0745 2.2196 0.0956 -2.0868 -0.9192 ...
##  $ q4     : Factor w/ 2 levels "1","2": 2 2 2 2 1 1 2 2 2 2 ...
##  $ q5     : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ q6     : Factor w/ 3 levels "1","2","3": 2 2 2 2 2 2 2 2 2 2 ...
##  $ i001   : num  NA 1 2 0 NA 1 NA NA 0 2 ...
##  $ i002   : num  0 NA NA 0 0 NA 0 NA NA NA ...
##  $ i003   : num  2 NA NA NA 1 NA 2 2 NA NA ...
##  $ i004   : num  NA 2 2 NA NA 2 NA 2 0 1 ...
##  $ i005   : num  NA 2 2 0 NA 2 NA NA 0 2 ...
##  $ i006   : num  0 NA NA 0 0 NA 0 NA NA NA ...
##  $ i007   : num  1 NA NA NA 1 NA 0 1 NA NA ...
##  $ i008   : num  NA 0 0 NA NA 0 NA 0 0 0 ...
##  $ i009   : num  NA 1 0 0 NA 0 NA NA 0 0 ...
##  $ i010   : num  1 NA NA 1 1 NA 1 NA NA NA ...
##  $ i011   : num  1 NA NA NA 0 NA 1 1 NA NA ...
##  $ i012   : num  NA 1 1 NA NA 1 NA 1 1 0 ...
##  $ i013   : num  NA 1 0 1 NA 0 NA NA 0 0 ...
##  $ i014   : num  0 NA NA 0 1 NA 1 NA NA NA ...
##  $ i015   : num  0 NA NA NA 1 NA 0 0 NA NA ...
```

Note that this section provided a general overview of the **lsasim** package. Interested readers can turn to the **lsasim** wiki for further documentation and testing results. In particular, one can find vignettes that further illustrate item parameter generation, test assembly, and test adminstration.

Generating data from PISA 2012

In this section, we demonstrate how existing background information, item parameters, and booklet designs can be used to generate data. The **lsasim** package includes prepared data from the 2012 administration of Programme for International Student Assessment.

PISA 2012 background questionnaire

The **lsasim** package includes the cumulative probabilities and heterogeneous correlation matrix for 18 background questionnaire items and a single mathematics plausible value. The 18 items comprise three scales, perseverance, openness to problem solving, and attitudes toward school. The cumulative proportions and correlation matrix were estimated using Switzerland student questionnaire data available from the PISA 2012 data base. It is important to note, however, that this background information is included for demonstration purposes only and is not suitable for valid inferences.

`pisa2012_q_marginal` is a list of cumulative proportions for all 19 variables. There are 10 five-category items, eight four-category items, and 1 continuous variable. Notice that each vector is given a name that corresponds to the item name in the PISA 2012 student codebook (also available from the PISA 2012 data base).

```
str(pisa2012_q_marginal)

## List of 19
##  $ ST93Q01: num [1:5] 0.053 0.167 0.429 0.826 1
##  $ ST93Q03: num [1:5] 0.132 0.346 0.679 0.903 1
##  $ ST93Q04: num [1:5] 0.144 0.471 0.778 0.956 1
##  $ ST93Q06: num [1:5] 0.13 0.382 0.692 0.932 1
##  $ ST93Q07: num [1:5] 0.078 0.228 0.507 0.846 1
##  $ ST94Q05: num [1:5] 0.176 0.565 0.892 0.984 1
##  $ ST94Q06: num [1:5] 0.204 0.598 0.894 0.985 1
##  $ ST94Q09: num [1:5] 0.201 0.563 0.881 0.984 1
##  $ ST94Q10: num [1:5] 0.179 0.539 0.867 0.983 1
##  $ ST94Q14: num [1:5] 0.113 0.3 0.559 0.83 1
##  $ ST88Q01: num [1:4] 0.065 0.311 0.79 1
##  $ ST88Q02: num [1:4] 0.029 0.102 0.548 1
##  $ ST88Q03: num [1:4] 0.197 0.752 0.945 1
##  $ ST88Q04: num [1:4] 0.452 0.895 0.974 1
##  $ ST89Q02: num [1:4] 0.433 0.908 0.988 1
##  $ ST89Q03: num [1:4] 0.544 0.891 0.973 1
##  $ ST89Q04: num [1:4] 0.735 0.981 0.995 1
##  $ ST89Q05: num [1:4] 0.437 0.935 0.988 1
##  $ PV1MATH: num 1
```

The 19 × 19 heterogeneous correlation matrix is `pisa2012_q_cormat`. Printing the sub-matrix for the first five items reveals that the matrix also includes the item names.

```
round(pisa2012_q_cormat[1:5, 1:5], 2)

##         ST93Q01 ST93Q03 ST93Q04 ST93Q06 ST93Q07
## ST93Q01    1.00    0.59   -0.33   -0.30   -0.23
## ST93Q03    0.59    1.00   -0.22   -0.23   -0.20
## ST93Q04   -0.33   -0.22    1.00    0.61    0.47
## ST93Q06   -0.30   -0.23    0.61    1.00    0.60
## ST93Q07   -0.23   -0.20    0.47    0.60    1.00
```

With these two pieces of information, we can generate background questionnaire responses and mathematics scores for 1000 test takers. Notice that the variable names in `pisa_background` take the name of the PISA items. When a named list is used with the `questionnaire_gen` function, the resulting variables in the data frame will adopt those names.

```
n_examinees <- 1000


pisa_background <- lsasim::questionnaire_gen(n          = n_examinees,
                                             cat_prop   = pisa2012_q_marginal,
                                             cor_matrix = pisa2012_q_cormat)


str(pisa_background)

## 'data.frame': 1000 obs. of  20 variables:
##  $ subject: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ST93Q01: Factor w/ 5 levels "1","2","3","4",..: 5 4 3 5 5 4 4 4 1 3 ...
##  $ ST93Q03: Factor w/ 5 levels "1","2","3","4",..: 3 2 1 5 4 4 5 4 2 1 ...
##  $ ST93Q04: Factor w/ 5 levels "1","2","3","4",..: 2 3 4 2 3 3 3 1 5 3 ...
##  $ ST93Q06: Factor w/ 5 levels "1","2","3","4",..: 3 4 2 3 3 2 2 1 5 4 ...
##  $ ST93Q07: Factor w/ 5 levels "1","2","3","4",..: 2 3 3 3 4 5 2 2 5 4 ...
##  $ ST94Q05: Factor w/ 5 levels "1","2","3","4",..: 2 2 3 1 3 2 4 2 3 3 ...
##  $ ST94Q06: Factor w/ 5 levels "1","2","3","4",..: 1 2 4 1 4 2 2 1 3 3 ...
##  $ ST94Q09: Factor w/ 5 levels "1","2","3","4",..: 2 1 4 1 4 1 3 1 4 2 ...
##  $ ST94Q10: Factor w/ 5 levels "1","2","3","4",..: 1 1 4 2 3 2 2 2 4 3 ...
##  $ ST94Q14: Factor w/ 5 levels "1","2","3","4",..: 3 3 5 2 4 2 1 4 4 3 ...
##  $ ST88Q01: Factor w/ 4 levels "1","2","3","4": 3 3 2 3 2 3 4 3 3 2 ...
##  $ ST88Q02: Factor w/ 4 levels "1","2","3","4": 3 3 2 4 4 4 4 3 4 4 ...
##  $ ST88Q03: Factor w/ 4 levels "1","2","3","4": 3 2 2 2 1 2 1 1 2 2 ...
##  $ ST88Q04: Factor w/ 4 levels "1","2","3","4": 1 2 3 1 1 1 1 1 1 2 ...
##  $ ST89Q02: Factor w/ 4 levels "1","2","3","4": 1 2 4 1 1 2 1 1 1 2 ...
##  $ ST89Q03: Factor w/ 4 levels "1","2","3","4": 1 2 3 1 1 3 1 1 2 1 ...
##  $ ST89Q04: Factor w/ 4 levels "1","2","3","4": 1 2 2 1 1 1 1 2 3 1 ...
##  $ ST89Q05: Factor w/ 4 levels "1","2","3","4": 2 2 2 1 1 1 1 2 2 1 ...
##  $ PV1MATH: num  1.108 -0.237 -2.918 0.827 0.264 ...
```

PISA 2012 mathematics assessment

The measurement model used in the 2012 administration of PISA was a partial credit model (OECD 2014). The technical manual provides the item parameters for all 109 mathematics items (OECD 2014, pp. 406–409), which are stored in `pisa2012_math_item`. Notice that each item has an item name, an item number, a b parameter, and, for those partial credit items, two d parameters.

```
pisa2012_math_item[1:10, ]

##    item_name item        b       d1       d2
## 1   PM00FQ01    1  0.39027  0.00000  0.00000
## 2   PM00GQ01    2  2.75209  0.00000  0.00000
## 3   PM00KQ02    3  1.97967  0.00000  0.00000
## 4   PM033Q01    4 -1.44130  0.00000  0.00000
## 5  PM034Q01T    5  0.42603  0.00000  0.00000
## 6   PM155Q01    6 -0.84340  0.00000  0.00000
## 7  PM155Q02D    7 -0.44941  0.74491 -0.74491
## 8  PM155Q03D    8  1.56865 -1.56865  1.56865
## 9  PM155Q04T    9 -0.38438  0.00000  0.00000
## 10 PM192Q01T   10  0.44066  0.00000  0.00000
```

The PISA technical manual also provides information regarding the allocation of the 109 items to ten item blocks (OECD 2014, pp. 406–409). `pisa2012_math_block` is the PISA 2012 block design matrix suitable for the `block_design` function. The design is stored as a data frame object, as it also includes the item names for consistency.

```
pisa2012_math_block[1:10, 1:7]

##    item_name item_no block1 block2 block3 block4 block5
## 1   PM00FQ01       1      0      0      0      0      0
## 2   PM00GQ01       2      0      0      0      0      1
## 3   PM00KQ02       3      0      0      0      1      0
## 4   PM033Q01       4      1      0      0      0      0
## 5  PM034Q01T       5      1      0      0      0      0
## 6   PM155Q01       6      1      0      0      0      0
## 7  PM155Q02D       7      1      0      0      0      0
## 8  PM155Q03D       8      1      0      0      0      0
## 9  PM155Q04T       9      1      0      0      0      0
## 10 PM192Q01T      10      0      1      0      0      0
```

Because only the indicator component of `pisa2012_math_block` is needed, we subset it and coerce it to a matrix. The PISA 2012 item parameters and corresponding block design matrix are used to create the block assignment matrix. Printing the `block_descriptives` provides a quick check of the block lengths and average difficulties.

```
pisa2012_math_block_mat <- as.matrix(pisa2012_math_block[, -c(1:2)])


pisa_blocks <- lsasim::block_design(item_parameters   = pisa2012_math_item,
                                    item_block_matrix = pisa2012_math_block_mat)


print(pisa_blocks$block_descriptives)

##     block length average difficulty
## b1            12              0.096
## b2            11             -0.007
## b3            12              0.012
## b4            12              0.025
## b5            12              0.067
## b6            13              0.092
## b7            13             -0.199
## b8            12              0.158
## b9            12             -0.238
## b10           12             -0.306
```

The booklet design information for the PISA mathematics assessment was also translated to a book design matrix, `pisa2012_math_booklet` (OECD 2014, p. 31). Like the block design, the book design is also stored as a data frame object to include a variable for booklet IDs. Note that this matrix was designed to construct the 13 *standard* booklets and does not include the *easy* booklets. Each row indicates a book while each column indicates an item block.

```
print(pisa2012_math_booklet)

##    booklet b1 b2 b3 b4 b5 b6 b7 b8 b9
## 1       B1  0  0  0  0  1  1  0  0  0
## 2       B2  0  0  0  0  0  0  0  1  0
## 3       B3  0  0  1  0  0  1  0  0  0
## 4       B4  0  0  0  1  0  1  0  1  0
## 5       B5  1  0  0  0  1  0  0  1  0
## 6       B6  1  1  0  0  0  1  0  0  0
## 7       B7  0  1  1  0  0  0  0  1  0
## 8       B8  0  0  0  1  0  0  0  0  0
## 9       B9  0  0  1  0  1  0  0  0  0
## 10     B10  1  0  1  1  0  0  0  0  0
## 11     B11  0  1  0  1  1  0  0  0  0
## 12     B12  0  1  0  0  0  0  0  0  0
## 13     B13  1  0  0  0  0  0  0  0  0
```

Again, we subset the data frame and coerce it to a matrix. From here, we can use it with `block_assignment` from `pisa_blocks` to create the 13 booklets.

```
pisa2012_math_book_mat <- as.matrix(pisa2012_math_booklet[, -1])


pisa_books <- lsasim::booklet_design(
                        item_block_assignment = pisa_blocks$block_assignment,
                        book_design           = pisa2012_math_book_mat)


print(pisa_books[1:5, ])

##    B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13
## i1  2 41 11  3  4  4 10  3 11   4  10  10   4
## i2 45 42 15 43  5  5 12 43 15   5  12  12   5
## i3 46 53 18 44  6  6 13 44 18   6  13  13   6
## i4 47 54 21 48  7  7 14 48 21   7  14  14   7
## i5 73 68 22 49  8  8 19 49 22   8  19  19   8
```

```
subj_booklets <- lsasim::booklet_sample(n_subj           = n_examinees,
                                        book_item_design = pisa_books)
```

The 13 standard books can now be administered to the 1000 test takers.

Because we have excluded the easy booklets, not all 109 items will be used to generate responses. Because the `response_gen` function matches item information based on a unique item number, we must subset the item bank to exclude those items in `pisa2012_math_item` that were not administered in the standard test books. To accomplish this, first obtain a sorted vector of unique items administered to the test takers, `subitems`. This vector can be used to subset the item bank, `pisa2012_math_item`.

```
subitems <- sort(unique(subj_booklets$item))
pisa_items <- pisa2012_math_item[subitems, ]
```

The resulting object, `pisa_items`, contains item information for the 84 administered items. Because we are using a subset of items whose item numbers are not sequential, we use the optional argument `item_no` when using the `response_gen` function. The variables of the resulting data frame are renamed to the PISA 2012 items names.

```
pisa_ir <- lsasim::response_gen(subject = subj_booklets$subject,
                                item    = subj_booklets$item,
                                theta   = pisa_background$PV1MATH,
                                item_no = pisa_items$item,
                                b_par   = pisa_items$b,
                                d_par   = list(pisa_items$d1,
                                               pisa_items$d2))

colnames(pisa_ir)[1:(ncol(pisa_ir)-1)] <- pisa2012_math_item$item_name[subitems]
```

The final procedure to obtain the generated PISA 2012 data is to merge the background data with the mathematics data. The final dataset, `pisa_data` contains 104 columns and 1000 rows. Printing the first 10 observations for select background questionnaire items and mathematics assessment items provides a feel for the resulting data.

```
pisa_data <- merge(pisa_background, pisa_ir, by = "subject")


print(pisa_data[1:10, c(1:4, 21:24)])

##    subject ST93Q01 ST93Q03 ST93Q04 PM00FQ01 PM00GQ01 PM00KQ02 PM033Q01
## 1        1       5       3       2       NA        0       NA        NA
## 2        2       4       2       3       NA       NA       NA         1
## 3        3       3       1       4       NA       NA        0         0
## 4        4       5       5       2       NA       NA        0        NA
## 5        5       5       4       3       NA       NA       NA        NA
## 6        6       4       4       3       NA       NA       NA        NA
## 7        7       4       5       3       NA        0       NA        NA
## 8        8       4       4       1       NA       NA        0        NA
## 9        9       1       2       5       NA       NA       NA         1
## 10      10       3       1       3       NA       NA       NA        NA
```

Test design simulation study

We now conduct a simple simulation to demonstrate that the default test generating functions operate as intended.

One hundred one-parameter items were generated using the `item_gen` function. Those items were distributed across five item blocks, which were assembled into 5 booklets. Each booklet contained 40 items. Both the item block assembly and booklet assembly use the default spiraling design described above.

```
library(TAM)
item_pool <- lsasim::item_gen(n_1pl = 100, b_bounds = c(-2, 2))
blocks <- lsasim::block_design(n_blocks = 5, item_parameters = item_pool)
books <- lsasim::booklet_design(item_block_assignment = blocks$block_assignment)
```

A true score, θ, was generated for 10,000 test takers in two countries (5000 test takers in each country). For the first country, $\theta \sim \mathcal{N}(0, 1)$ while for the second country, $\theta \sim \mathcal{N}(0.25, 1)$. The five booklets were distributed evenly across the two countries and country-specific means and variances were estimated using **TAM** (Robitzsch et al. 2017). The model is such that the intercept for country 1 is constrained to zero so that we are estimating the group difference. As seen in Table 3, the country means and variances were recovered.

```
m1_out <- matrix(NA, nrow = 100, ncol = 4)


for (i in 1:100){
  theta_c1 <- data.frame(theta = rnorm(5000, 0, 1))
  theta_c1$country <- 1
  theta_c2 <- data.frame(theta = rnorm(5000, 0.25, 1))
  theta_c2$country <- 2
  theta <- rbind(theta_c1, theta_c2)


  book_samp <- lsasim::booklet_sample(n_subj = nrow(theta),
                                      book_item_design = books)


  cog <- lsasim::response_gen(subject = book_samp$subject,
                              item    = book_samp$item,
                              theta   = theta$theta,
                              b_par = item_pool$b)


  m1 <- TAM::tam(cog[, -ncol(cog)], group = theta[, "country"])


  c_means <- m1$beta
  c1_var <- mean(m1$variance[theta$country == 1])
  c2_var <- mean(m1$variance[theta$country == 2])


  m1_out[i, ] <- c(c_means, c1_var, c2_var)
}
```

```
mu <- sprintf("%.4f", colMeans(m1_out))
sd <- sprintf("%.4f", apply(m1_out, 2, FUN=sd))
```

Discussion

ILSAs are tasked with obtaining sound measures of what students from around the world know and can do in the relevant content areas, as well as obtaining a host of background variables to aid in the contextualization of those measures. These tasks place a

Table 3 Simulation results, means and standard deviations of country-specific parameters based on 100 replications

	Country 1		Country 2	
	$\bar{\theta}$	var(θ)	$\bar{\theta}$	var(θ)
Generating value	0	1	0.25	1
Mean	0.0000	1.0000	0.2514	1.0014
Std. dev	0.0000	0.0255	0.0205	0.0253

unique set of requirements on the test design and psychometric modeling. Although innovations in ILSAs can come from the re-analysis of past assessments, those data are fundamentally constricted to a particular design and by extant data. Due to the scope of ILSAs, pilot testing is highly restricted, leaving simulation studies as the primary means for understanding issues and possible solutions within the ILSA arena. The intention for **lsasim** was to develop a minimal number of functions to facilitate the generation of data that mimics the large scale assessment context to the extent possible. To that end, **lsasim** offers the possibility of simulating data that adhere to general properties found in the background questionnaire (mostly ordinal, correlated variables that are also related to varying degrees with some latent trait). The package also features functions for simulating achievement data according to a number of common IRT models with known parameters. A clear advantage of **lsasim** over other simulation software is that the *achievement* data, in the form of item responses, can arise from multiple-matrix sampled test designs. Although the background questionnaire data can be linked to the test responses, all aspects of **lsasim** can function independently, affording researchers a high degree of flexibility in terms of possible research questions and the part of an assessment that is of most interest. Built in default functionality also allows researchers to opt for randomly chosen population parameters. Alternatively, users can specify their own test design specifications and population parameters, offering the possibility of full control over the research design and data generation process.

By way of introduction, the paper described and briefly illustrated the eight functions that make up the package. Because researchers will in many circumstances use information from previous assessments for simulation purposes, the paper went on to demonstrate how LSA data can be generated from parameter estimates and design features of PISA 2012. Finally, a small simulation showed that using the default test assembly functions recovered known population proficiency parameters for two groups. Although we demonstrated the soundness of the package's default settings, we expect users to rely on those default settings only for aspects of a given simulation design that are considered to be nuisances. For example, a study designed to examine the efficiency of various item block designs could rely on the default background questionnaire functions without loss of generality. Alternatively, a researcher interested in studying background questionnaire invariance across heterogeneousness populations might utilize many of the default test assembly functions. Otherwise, we generally assume that users will bring a set of known or plausible population parameters that will provide the basis for further investigations. We believe that **lsasim** can be a useful tool for operational test developers and basic and applied measurement researchers. As national and international assessments branch

into new platforms and populations, it is important that researchers with a solid background in measurement and large-scale test design have a ready means for evaluating the performance of new and existing methods. Finally, as a reminder, the default PISA parameters that are included with **lsasim** are not intended to be used to infer anything about the 2012 PISA administration. Rather, they provide an illustrative example of a test design and associated parameters that approximates an operational setting.

Authors' contributions
The concept for **lsasim** was fostered by LR and DR. THM carried out the development of **lsasim**. YL led the testing of **lsasim**. THM, LR and DR contributed to the writing of the manuscript. All authors read and approved the final manuscript.

Author details
[1] Pearson, San Antonio, TX, USA. [2] Centre for Educational Measurement, University of Oslo, Oslo, Norway. [3] Indiana University, Bloomington, IN, USA.

Acknowledgements
The authors acknowledge Dr. Eugenio Gonzalez for his feedback during development as well as the Norwegian Research Council for supporting this research.

Competing interests
None of the authors have any competing interests that would be interpreted as influencing the research and ethical standards were followed in the conduct of **lsasim** and the writing of this manuscript.

Availability of data and materials
The package **lsasim**, including data used in this manuscript, can be found on Comprehensive R Archive Network, CRAN.

Funding
This manuscript was partially funded by the Norwegian Research Council, FINNUT program, Grant 255246.

Publisher's Note

Received: 9 May 2018 Accepted: 10 November 2018
Published online: 19 November 2018

References

Adams, R. J., Lietz, P., & Berezner, A. (2013). On the use of rotated context questionnaires in conjunction with multilevel item response models. *Large-scale Assessments in Education*, *1*(1), 5.
American Educational Research Association, American Psychological Association, & National Council on Measurement in Education. (2014). *Standards for educational and psychological testing 2014*. AERA.
Barbiero, A., & Ferrari, P. A. (2015). *GenOrd: Simulation of discrete random variables with given correlation matrix and marginal distributions [Computer software manual]*. Retrieved from https://CRAN.R-project.org/package=GenOrd (R package version 1.4.0)
Beaton, A. E., & Johnson, E. G. (1992). Overview of the scaling methodology used in the national assessment. *Journal of Educational Measurement*, *29*(2), 163–175.
Chalmers, R. P. (2012). mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, *48*(6), 1–29. Retrieved from http://www.jstatsoft.org/v48/i06/
Foshay, A. W., Thorndike, R., Hotyat, F., Pidgeon, D., & Walker, D. (1962). *Educational achievement of thirteen-year-olds in twelve countries (Tech. Rep.)*. Hamburg: UNESCO Institute for Education. Retrieved from http://unesdoc.unesco.org/images/0013/001314/131437eo.pdf
Fox, J. (2016). *polycor: Polychoric and polyserial correlations [Computer software manual]*. Retrieved from https://CRAN.R-project.org/package=polycor (R package version 0.7-9)
Gonzalez, E., & Rutkowski, L. (2010). Principles of matrix booklet designs and parameter recovery in large-scale assessments. *IERI Monograph Series*, *3*, 125–156.
Little, R. J. A., & Rubin, D. B. (1983). On jointly estimating parameters and missing data by maximizing the complete-data likelihood. *The American Statistician*, *37*(3), 218–220.
Lockheed, M., Prokic-Breuer, T., & Shadrova, A. (2015). *The experience of middle-income countries participating in PISA 2000–2015*. Washington, DC: World Bank Publications. https://doi.org/10.1787/9789264246195-en.
Magis, D., & Raiche, G. (2012). Random generation of response patterns under computerized adaptive testing with the R package catR. *Journal of Statistical Software*, *48*(8), 1–31. https://doi.org/10.18637/jss.v048.i08.
Magis, D., Yan, D., & von Davier, A. (2017). *mstR: Procedures to generate patterns under multistage testing [Computer software manual]*. Retrieved from https://CRAN.R-project.org/package=mstR (R package version 1.0)
Mandal, B. N. (2018). *ibd: Incomplete block designs [Computer software manual]*. Retrieved from https://CRAN.R-project.org/package=ibd (R package version 1.4)

Martin, M. O., Mullis, I. V. S., & Hooper, M. (Eds.). (2016). *Methods and procedures in TIMSS 2015*. Boston: TIMSS & PIRLS International Study Center, Boston College. Retrieved from http://timssandpirls.bc.edu/publications/timss/2015-methods.html

Matta, T., Rutkowski, L., Rutkowski, D., & Liaw, Y. (2017). *Isasim: Simulate large scale assessment data [Computer software manual]*. Retrieved from https://CRAN.R-project.org/package=lsasim (R package version 1.0.0)

Mislevy, R. J. (1991). Randomization-based inference about latent variables from complex samples. *Psychometrika*, *56*(2), 177–196.

Mislevy, R. J., Beaton, A. E., Kaplan, B., & Sheehan, K. M. (1992a). Estimating population characteristics from sparse matrix samples of item responses. *Journal of Educational Measurement*, *29*(2), 133–161.

Mislevy, R. J., Johnson, E. G., & Muraki, E. (1992b). Scaling procedures in NAEP. *Journal of Educational and Behavioral Statistics*, *17*(2), 131–154.

OECD. (2014). *PISA 2012 technical report (Tech. Rep.)*. Paris: OECD Publishing. Retrieved from https://www.oecd.org/pisa/pisaproducts/PISA-2012-technical-report-final.pdf

OECD. (2017). *PISA 2015 technical report (draft)*. Paris: OECD Publishing.

R Core Team. (2017). *R: A language and environment for statistical computing [Computer software manual]*. Vienna: R Core Team. Retrieved from https://www.R-project.org/

Robitzsch, A., Kiefer, T., & Wu, M. (2017). *TAM: Test analysis modules [Computer software manual]*. Retrieved from https://CRAN.R-project.org/package=TAM (R package version 2.0-37)

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, *63*(3), 581–592.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. Hoboken, NJ: Wiley.

Rutkowski, L., von Davier, M., Gonzalez, E., & Zhou, Y. (2014). Assessment design for international large-scale assessment. In L. Rutkowski, M. von Davier, & D. Rutkowski (Eds.), *Handbook of international large-scale assessment: Background, technical issues, and methods of data analysis*. Boca Raton: Chapman & Hall/CRC Press.

Rutkowski, L., & Zhou, Y. (2015). The impact of missing and error-prone auxiliary information on sparse-matrix sub-population parameter estimates. *Methodology*, *11*(3), 89–99.

Shoemaker, D. M. (1973). *Principles and procedures of multiple matrix sampling*. Oxford: Ballinger.

von Davier, M., Sinharay, S., Oranje, A., & Beaton, A. (2006). The statistical procedures used in national assessment of educational progress: Recent developments and future directions. In C. R. Rao & S. Sinharay (Eds.), *Handbook of statistics* (pp. 1039–1055). Amsterdam: Elsevier.