synthpop: Bespoke Creation of Synthetic Data in R

Beata Nowok
University of Edinburgh

Gillian M Raab
University of Edinburgh

Chris Dibben
University of Edinburgh

Abstract

In many contexts, confidentiality constraints severely restrict access to unique and valuable microdata. Synthetic data which mimic the original observed data and preserve the relationships between variables but do not contain any disclosive records are one possible solution to this problem. The **synthpop** package for R, introduced in this paper, provides routines to generate synthetic versions of original data sets. We describe the methodology and its consequences for the data characteristics. We illustrate the package features using a survey data example.

Keywords: synthetic data, disclosure control, CART, R, UK Longitudinal Studies.

This introduction to the R package **synthpop** is a slightly amended version of Nowok B, Raab GM, Dibben C (2016). **synthpop**: Bespoke Creation of Synthetic Data in R. *Journal of Statistical Software*, **74**(11), 1-26. doi:10.18637/jss.v074.i11. URL https://www.jstatsoft.org/article/view/v074i11.

1. Introduction and background

1.1. Synthetic data for disclosure control

National statistical agencies and other institutions gather large amounts of information about individuals and organisations. Such data can be used to understand population processes so as to inform policy and planning. The cost of such data can be considerable, both for the collectors and the subjects who provide their data. Because of confidentiality constraints and guarantees issued to data subjects the full access to such data is often restricted to the staff of the collection agencies. Traditionally, data collectors have used anonymisation along with simple perturbation methods such as aggregation, recoding, record-swapping, suppression of sensitive values or adding random noise to prevent the identification of data subjects. Advances in computer technology have shown that such measures may not prevent disclosure (Ohm 2010) and in addition they may compromise the conclusions one can draw from such data (Elliot and Purdam 2007; Winkler 2007).

In response to these limitations there have been several initiatives, most of them centred around the U.S. Census Bureau, to generate synthetic data which can be released to users outside the setting where the original data are held. The basic idea of synthetic data is to replace some or all of the observed values by sampling from appropriate probability distributions so that the essential statistical features of the original data are preserved. The approach has been developed along similar lines to recent practical experience with multiple imputation methods although synthesis is not the same as imputation. Imputation replaces data which are missing with modelled values and adjusts the inference for the additional uncertainty due to this process. For synthesis, in the circumstances when some data are missing two approaches are possible, one being to impute missing values prior to synthesis and the other to synthesise the observed patterns of missing data without estimating the missing values. In both cases all data to be synthesised are treated as known and they are used to create the synthetic data which are then used for inference. The data collection agency generates multiple synthetic data sets and inferences are obtained by combining the results of models fitted to each of them. The formulae for the variance of estimates from synthetic data are different from those used for imputed data.

The synthetic data methods were first proposed by Rubin (1993) and Little (1993) and have been developed by Raghunathan, Reiter, and Rubin (2003), Reiter (2003) and Reiter and Raghunathan (2007). They have been discussed and exemplified in a further series of papers (Abowd and Lane 2004; Abowd and Woodcock 2004; Reiter 2002, 2005a; Drechsler and Reiter 2010; Kinney, Reiter, and Berger 2010; Kinney, Reiter, Reznek, Miranda, Jarmin, and Abowd 2011). Non-parametric synthesising methods were introduced by Reiter (2005b) who first suggested to use classification and regression trees (CART; Breiman, Friedman, Olshen, and Stone 1984) to generate synthetic data. CART was then compared with more powerful machine learning procedures such as random forests, bagging and support vector machines (Caiola and Reiter 2010; Drechsler and Reiter 2011). The monograph by Drechsler (2011) summarises some of the theoretical, practical and policy developments and provides an excellent introduction to synthetic data for those new to the field.

The original aim of producing synthetic data has been to provide publicly available datasets that can be used for inference in place of the actual data. However, such inferences will only be valid if the model used to construct the synthetic data is the true mechanism that has generated the observed data, which is very difficult, if at all possible, to achieve. Our aim in writing the **synthpop** package (Nowok, Raab, Snoke, and Dibben 2016) for R (R Core Team 2016) is a more modest one of providing test data for users of confidential datasets. Note that currently all values of variables chosen for synthesis are replaced but this will be relaxed in future versions of the package. These test data should resemble the actual data as closely as possible, but would never be used in any final analyses. The users carry out exploratory analyses and test models on the synthetic data, but they, or perhaps staff of the data collection agencies, would use the code developed on the synthetic data to run their final analyses on the original data. This approach recognises the limitations of synthetic data produced by these methods. It is interesting to note that a similar approach is currently being used for both of the synthetic products made available by the U.S. Census Bureau (see `https://www.census.gov/ces/dataproducts/synlbd/` and `http://www.census.gov/programs-surveys/sipp/guidance/sipp-synthetic-beta-data-product.html`), where results obtained from the synthetic data are validated on the original data (“gold standard files").

1.2. Motivation for the development of synthpop

The England and Wales Longitudinal Study (ONS LS; Hattersley and Cresser 1995), the Scottish Longitudinal Study (SLS; Boyle, Feijten, Feng, Hattersley, Huang, Nolan, and Raab 2012) and the Northern Ireland Longitudinal Study (NILS; O’Reilly, Rosato, Catney, Johnston, and Brolly 2011) are rich micro-datasets linking samples from the national census in each country to administrative data (births, deaths, marriages, cancer registrations and other sources) for individuals and their immediate families across several decades. Whilst unique and valuable resources, the sensitive nature of the information they contain means that access to the microdata is restricted to approved researchers and longitudinal study (LS) support staff, who can only view and work with the data in safe settings controlled by the national statistical agencies. Consequently, compared to other census data products such as the aggregate statistics or samples of anonymised records, the three longitudinal studies (LSs) are used by a small number of researchers, a situation which limits their potential impact. Given that confidentiality constraints and legal restrictions mean that open access is not possible with the original microdata, alternative options are needed to allow academics and other users to carry out their research more freely. To address this the SYLLS (Synthetic Data Estimation for UK Longitudinal Studies) project (see `http://www.lscs.ac.uk/projects/synthetic-data-estimation-for-uk-longitudinal-studies/`) has been funded by the Economic and Social Research Council to develop techniques to produce synthetic data which mimics the observed data and preserves the relationships between variables and transitions of individuals over time, but can be made available to accredited researchers to analyse on their own computers. The **synthpop** package for R has been written as part of the SYLLS project to allow LS support staff to produce synthetic data for users of the LSs, that are tailored to the needs of each individual user. Hereinafter, we will use the term “synthesiser" for someone like an LS support officer who is producing the synthetic data from the observed data and hence has access to both. The term “analyst" will refer to someone like an LS user who has no access to the observed data and will be using the synthetic data for exploratory analyses. After the exploratory analysis the analyst will develop confirmatory models and can send the code to a synthesiser to run the gold standard analyses. As well as providing routines to generate the synthetic data the **synthpop** package contains routines that can be used by the analyst to summarise synthetic data and fitted models from synthetic data and those that can be used by the synthesiser to compare gold standard analyses with those from the synthetic data.

Although primarily targeted to the data from the LSs, the **synthpop** package is written in a form that makes it applicable to other confidential data where the resource of synthetic data would be valuable. By providing a comprehensive and flexible framework with parametric and non-parametric methods it fills a gap in tools for generating synthetic versions of original data sets. The R package **simPop** (Meindl, Templ, Alfons, and Kowarik 2016) which is a successor to the **simPopulation** package (Alfons, Kraft, Templ, and Filzmoser 2011; Alfons and Kraft 2013) implements model-based methods to simulate synthetic populations based on household survey data and auxiliary information. The approach used concentrates on simulation of close-to-reality population and is similar to microsimulation rather than multiple imputation. The software **IVEware** for SAS (SAS Institute Inc. 2013) and its stand-alone version **SRCware** (Raghunathan, Solenberger, and Van Hoewyk 2002; Survey Methodology Program 2011), originally developed for multiple imputation, include the `SYNTHESIZE` module that allows to produce synthetic data. **IVEware** uses conditionally specified parametric models with proper imputation and these can be adjusted for clustered, weighted or stratified samples. All item missing values are imputed when generating synthetic data sets. No analysis methods are available in this software because only the formulae for imputation are available which are not appropriate for synthetic data.

1.3. Structure of this paper

The structure of this paper is as follows. The next section introduces the notation, terminology and the main theoretical results needed for the simplest and, we expect, the most common use of the package. More details of the theoretical results for the general case can be found in Raab, Nowok, and Dibben (2016). Readers not interested in the theoretical details can now proceed directly to Section 3 which presents the package and its basic functionality. Section 4 that follows provides some illustrative examples. The concluding Section 5 indicates directions for future developments.

2. Overview of method

Observed data from a survey or a sample from a census or population register are available to the synthesiser. They consist of a sample of n units consisting of (x_{obs}, y_{obs}) where x_{obs}, which may be null, is a matrix of data that can be released unchanged to the analyst and y_{obs} is an $n \times p$ matrix of p variables that require to be synthesised. We consider here the simple case when the synthetic data sets (syntheses) will each have the same number of records as the original data and the method of generating the synthetic sample (e.g., simple random sampling or a complex sample design) matches that of the observed data. This condition allows to make inferences from synthetic data generated from distributions with parameters fitted to the observed data without sampling the parameters from their posterior distributions. We refer to such synthesis as “simple synthesis". When synthetic data are generated from distributions with parameters sampled from their posterior distributions we refer to this as “proper synthesis".

2.1. Generating synthetic data

The observed data are assumed to be a sample from a population with parameters that can be estimated by the synthesiser, specifically y_{obs} is assumed to be a sample from $f(Y|x_{obs}, \theta)$ where θ is a vector of parameters. This could be a hypothetical infinite super-population or a finite population which is large enough for finite population corrections to be ignored. The synthesiser fits the data to the assumed distribution and obtains estimates of its parameters. In most implementations of synthetic data generation, including **synthpop**, the joint distribution is defined in terms of a series of conditional distributions. A column of y_{obs} is selected and the distribution of this variable, conditional on x_{obs} is estimated. Then the next column is selected and its distribution is estimated conditional on x_{obs} and the column of y_{obs} already selected. The distribution of subsequent columns of y_{obs} are estimated conditional on x_{obs} and all previous columns of y_{obs}.

The generation of the synthetic data sets proceeds in parallel to the fitting of each conditional distribution. Each column of the synthetic data is generated from the assumed distribution, conditional on x_{obs}, the fitted parameters of the conditional distribution (simple synthesis) and the synthesised values of all the previous columns of y_{obs}. Alternatively the synthetic values can be generated from the posterior distribution of the parameters (proper synthesis). In both cases, a total of m synthetic data sets are generated.

2.2. Inference from the synthetic data

An analyst who wants to estimate a model from the synthetic data will fit the model to each of the m synthetic data sets and obtain an estimate of its vector of parameters β from each synthetic data set as $(\hat{\beta}_1, \cdots, \hat{\beta}_i, \cdots, \hat{\beta}_m)$. If the model for the data is correct the m estimates from the synthetic data will be centred around the estimate $\hat{\beta}$ that would have been obtained from the observed data. We are assuming that it is the goal of the analyst to use the synthetic data to estimate $\hat{\beta}$ and its variance-covariance matrix $V_{\hat{\beta}}$. If the method of inference used to fit the model provides consistent estimates of the parameters and the same is true for analyses of the synthetic data then the mean of m synthetic estimates, $\bar{\hat{\beta}} = \sum \hat{\beta}_i/m$ provides a consistent estimate of $\hat{\beta}$. Provided the observed and synthetic data are generated by the common sampling scheme then $V_{\bar{\hat{\beta}}} = \sum V_{\hat{\beta}_i}/m$ will be a consistent estimate of $V_{\hat{\beta}}$. The variance-covariance matrix of $\bar{\hat{\beta}}$, conditional on $\hat{\beta}$ and $V_{\hat{\beta}}$, becomes $V_{\hat{\beta}}/m$ which can be estimated from $V_{\bar{\hat{\beta}}}/m$. Thus the stochastic error in the mean of the synthetic estimates about the values from the observed data can be reduced to a negligible quantity by increasing m. It must be remembered, however that the consistency of $\bar{\hat{\beta}}$ only applies when observed data are a sample from the distribution used for synthesis. In practical applications differences between the analyses on the observed data and those from the mean of the syntheses will be found because the data do not conform to the model used for synthesis. Such differences will not be reduced by increasing m. The synthesiser, with access to the observed data, can estimate $\bar{\hat{\beta}} - \hat{\beta}$ and compare it to its standard error in order to judge the extent that this model mismatch affects the estimates.

Note that this result is different from the literature cited above which aims to use the results of the synthetic data to make inference about the population from which the original gold standard data have been generated. But our aim, in the simplest case we describe above, is only to make inferences to the results that would have been obtained by the gold standard analysis, with the expectation that the analyst will run final models on the observed data. Also, unlike most of the literature above, in the simplest case we do not sample from the predictive distribution of the parameters to create the synthetic data but an option to do so is available in **synthpop**. This approach has been proposed recently by Reiter and Kinney (2012) for partially synthetic data. The justification for this approach for completely synthesised data is in Raab *et al.* (2016) along with the details of how the **synthpop** package can be used to make inferences to the population.

3. The synthpop package in practice

3.1. Obtaining the software

The **synthpop** package is an add-on package to the statistical software R. It is freely available from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=synthpop. It utilises the structure and some functions of the **mice** multiple imputation package (Van Buuren and Groothuis-Oudshoorn 2011) but adopts and extends it for the specific purpose of generating and analysing synthetic data.

3.2. Basic functionality

The **synthpop** package aims to provide a user with an easy way of generating synthetic versions of original observed data sets. Via the function `syn()` a synthetic data set is produced using a single command. The only required argument is `data` which is a data frame or a matrix containing the data to be synthesised. By default, a single synthetic data set is produced using simple synthesis, i.e., without sampling from the posterior distribution of the parameters of the synthesising models. Multiple data sets can be obtained by setting parameter `m` to a desired number. Proper synthesis with synthetic data sampled from the posterior predictive distribution of the observed data is conducted when argument `proper` is set to `TRUE`. Data synthesis can be further customized with other optional parameters. Below, we only present the salient features of the `syn()` function. See examples in Section 4 and the R documentation for the function `syn()` for more details (command `?syn` at the R console).

Choice of synthesising method

The synthesising models are defined by a parameter `method` which can be a single string or a vector of strings. Providing a single method name assumes the same synthesising method for each variable, unless a variable's data type precludes it. Note that a variable to be synthesised first that has no predictors is a special case and its synthetic values are by default generated by random sampling with replacement from the original data (`"sample"` method). In general, a user can choose between parametric and non-parametric methods. The latter are based on classification and regression trees (CART) that can handle any type of data. By default `"cart"` method is used for all variables that have predictors. It utilizes function `rpart()` available in package **rpart** (Therneau, Atkinson, and Ripley 2015). An alternative implementation of the CART technique from package **party** (Hothorn, Hornik, and Zeileis 2006) can be used by selecting `"ctree"` method. Setting the parameter `method` to `"parametric"` assigns default parametric methods to variables to be synthesised based on their types. The default parametric methods for numeric, binary, unordered factor and ordered factor data type are specified in vector `default.method` which may be customised if desired. Alternatively a method can be chosen out of the available methods for each variable separately. The methods currently implemented are listed in Table 1. Their default settings can be modified via additional parameters of the `syn()` function that have to be named using period-separated method and parameter name (`method.parameter`). For instance, in order to set a `minbucket` (minimum number of observations in any terminal node of a CART model) for a `"cart"` synthesising method, `cart.minbucket` has to be specified. Those arguments are method-specific and are used for all variables to be synthesised using that method. For variables to be left unchanged an empty `method` (`""`) should be used. A new synthesising method can be easily introduced by writing a function named `syn.newmethod()` and then specifying `method` parameter of `syn()` function as `"newmethod"`.

Controlling the predictions

The synthetic values of the variables are generated sequentially from their conditional distributions given variables already synthesised with parameters from the same distributions fitted with the observed data. Next to choosing model types, a user may determine the order in which variables should be synthesised (`visit.sequence` parameter) and also the set of vari-

Method	Description	Data type
Non-parametric		
`ctree, cart`	Classification and regression trees	Any
`surv.ctree`	Classification and regression trees	Duration
Parametric		
`norm`	Normal linear regression	Numeric
`normrank`*	Normal linear regression preserving the marginal distribution	Numeric
`logreg`*	Logistic regression	Binary
`polyreg`*	Polytomous logistic regression	Factor, > 2 levels
`polr`*	Ordered polytomous logistic regression	Ordered factor, > 2 levels
`pmm`	Predictive mean matching	Numeric
Other		
`sample`	Random sample from the observed data	Any
`passive`	Function of other synthesised data	Any

Table 1: Built-in synthesising methods. * Indicates default parametric methods.

ables to include as predictors in the synthesising model (`predictor.matrix` parameter). As mentioned above, the choice of explanatory variables is restricted by the synthesis sequence and variables that are not synthesised yet cannot be used in prediction models. It is possible, however, to include predictor variables in the synthesis that will not be synthesized themselves.

Handling data with missing or restricted values

The aim of producing a synthetic version of observed data here is to mimic their characteristics in all possible ways, which may include missing and restricted values data. Values representing missing data in categorical variables are treated as additional categories and reproducing them is straightforward. Continuous variables with missing data are modelled in two steps. In the first step, we synthesise an auxiliary binary variable specifying whether a value is missing or not. Depending on the method specified by a user for the original variable a logit or CART model is used for synthesis. If there are different types of missing values an auxiliary categorical variable is created to reflect this and an appropriate model is used for synthesis (a polytomous or CART model). In the second step, a synthesising model is fitted to the non-missing values in the original variable and then used to generate synthetic values for the non-missing category records in our auxiliary variable. The auxiliary variable and a variable with non-missing values and zeros for remaining records are used instead of the original variable for prediction of other variables. The missing data codes have to be specified by a user in `cont.na` parameter of the `syn()` function if they differ from the `R` missing data code `NA`. The `cont.na` argument has to be provided as a named list with names of its elements corresponding to the variables names for which the missing data codes need to be specified.

Restricted values are those where the values for some cases are determined explicitly by those of other variables. In such cases the rules and the corresponding values should be specified using `rules` and `rvalues` parameters. They are supplied in the form of named lists in the same manner as the missing data codes parameter. The variables used in `rules` have to be synthesised prior to the variable they refer to. In the synthesis process the restricted values are assigned first and then only the records with unrestricted values are synthesised.

3.3. Additional functionality

Disclosure control

Completely synthesised data such as those generated by the `syn()` function with default settings do not by definition include real units, so disclosure of a real person is acknowledged to be unlikely. It has been confirmed by Elliot (2015) in his report on the disclosure risk associated with the synthetic data produced using **synthpop** package. Nonetheless, there are some options that are designed to further protect data and limit the perceived disclosure. For the CART model (`"ctree"` or `"cart"` method), the final leaves to be sampled from may include only a very small number of individuals, which elevates risk of replicating real persons. To avoid this, a user can specify, for instance, a minimum size of a final node that a CART model can produce. It can be done using the `cart.minbucket` and the `ctree.minbucket` parameter for the `"cart"` and `"ctree"` methods respectively. However, the right balance needs to be found between disclosure risk and synthetic data quality. For the `"ctree"`, `"cart"`, `"normrank"` and `"sample"` methods there is also the risk of releasing real unusual values for continuous variables and therefore use of a smoothing option is essential for protecting confidentiality. If the `smoothing` parameter is set to `"density"` a Gaussian kernel density smoothing is applied to the synthesised values.

There are also additional precautionary options built into the package, which can be applied using `sdc()` function (`sdc` stands for statistical disclosure control). The function allows top and bottom coding, adding labels to the synthetic data sets to make it clear that the data are fake so no one mistakenly believes them to be real and removing from the synthetic data set any unique cases with variable sequences that are identical to unique individuals in the real dataset. The last tool reduces the chances of a person who is in the real data believing that their actual data is in the synthetic data.

4. Illustrative examples

4.1. Data

The **synthpop** package includes a data frame `SD2011` with individual microdata that will be used for illustration. The data set is a subset of survey data collected in 2011 within the Social Diagnosis project (Council for Social Monitoring 2011) which aims to investigate objective and subjective quality of life in Poland. The complete data set is freely available at `http://www.diagnoza.com/index-en.html` along with a detailed documentation. The `SD2011` subset contains 35 selected variables of various type for a sample of 5,000 individuals aged 16 and over.

4.2. Simple example

To get access to **synthpop** functions and the `SD2011` data set we need to load the package via

Variable name	Description	Data type
`sex`	Sex	Binary
`age`	Age	Numeric
`edu`	Highest educational qualification	Factor, > 2 levels
`marital`	Marital status	Factor, > 2 levels
`income`	Personal monthly net income	Numeric
`ls`	Overall life satisfaction	Factor, > 2 levels
`wkabint`	Plans to go abroad to work in the next two years	Factor, > 2 levels

Table 2: Variables to be synthesised.

```
R> library("synthpop")
```

For our illustrative examples of the `syn()` function we use seven variables of various data types which are listed in Table 2.

Although function `syn()` allows synthesis of a subset of variables (see Section 4.3), for ease of presentation here we extract variables of interest from the `SD2011` data set and store them in a data frame called `ods` which stands for "observed data set". The structure of the `ods` data can be investigated using the `head()` function which prints the first rows of a data frame.

```
R> vars <- c("sex", "age", "edu", "marital", "income", "ls", "wkabint")
R> ods <- SD2011[, vars]
R> head(ods)

     sex age                  edu marital income               ls wkabint
1 FEMALE  57   VOCATIONAL/GRAMMAR MARRIED    800          PLEASED      NO
2   MALE  20   VOCATIONAL/GRAMMAR  SINGLE    350 MOSTLY SATISFIED      NO
3 FEMALE  18   VOCATIONAL/GRAMMAR  SINGLE     NA          PLEASED      NO
4 FEMALE  78 PRIMARY/NO EDUCATION WIDOWED    900            MIXED      NO
5 FEMALE  54   VOCATIONAL/GRAMMAR MARRIED   1500 MOSTLY SATISFIED      NO
6   MALE  20            SECONDARY  SINGLE     -8          PLEASED      NO
```

To run a default synthesis only the data to be synthesised have to be provided as a function argument. Here, an additional parameter `seed` is used to fix the pseudo random number generator seed and make the results reproducible. To monitor the progress of the synthesising process the function `syn()` prints to the console the current synthesis number and the name of a variable that is being synthesised. This output can be suppressed by setting an argument `print.flag` to `FALSE`.

```
R> my.seed <- 17914709
R> sds.default <- syn(ods, seed = my.seed)
```

The resulting object of class '`synds`' called here `sds.default`, where `sds` stands for "synthesised data set", is a list. The `print` method displays its selected components (see below). An element `syn` contains a synthesised data set which can be accessed using a standard list referencing (`sds.default$syn`).

```
R> sds.default


Call:
($call) syn(data = ods, seed = my.seed)

Number of synthesised data sets:
($m)  1

First rows of synthesised data set:
($syn)
     sex age                        edu marital income                  ls
1   MALE  72      PRIMARY/NO EDUCATION MARRIED     NA    MOSTLY SATISFIED
2   MALE  82      PRIMARY/NO EDUCATION MARRIED    861             PLEASED
3   MALE  25                 SECONDARY  SINGLE   1050 MOSTLY DISSATISFIED
4   MALE  69      PRIMARY/NO EDUCATION MARRIED    960    MOSTLY SATISFIED
5 FEMALE  34 POST-SECONDARY OR HIGHER WIDOWED   1000               MIXED
6 FEMALE  79                 SECONDARY MARRIED   1400             PLEASED
  wkabint
1      NO
2      NO
3      NO
4      NO
5      NO
6      NO
...

Synthesising methods:
($method)
     sex      age      edu  marital   income       ls  wkabint
"sample"   "cart"   "cart"   "cart"   "cart"   "cart"   "cart"

Order of synthesis:
($visit.sequence)
    sex     age     edu marital  income      ls wkabint
      1       2       3       4       5       6       7

Matrix of predictors:
($predictor.matrix)
        sex age edu marital income ls wkabint
sex       0   0   0       0      0  0       0
age       1   0   0       0      0  0       0
edu       1   1   0       0      0  0       0
marital   1   1   1       0      0  0       0
income    1   1   1       1      0  0       0
ls        1   1   1       1      1  0       0
wkabint   1   1   1       1      1  1       0
```

The remaining (undisplayed) list elements include other `syn()` function parameters used in the synthesis. Their names can be listed via the `names()` function. For a complete description see the `syn()` function help page (`?syn`).

```
R> names(sds.default)
```

```
 [1] "call"             "m"                "syn"
 [4] "method"           "visit.sequence"   "predictor.matrix"
 [7] "smoothing"        "event"            "denom"
[10] "proper"           "n"                "k"
[13] "rules"            "rvalues"          "cont.na"
[16] "semicont"         "drop.not.used"    "drop.pred.only"
[19] "models"           "seed"             "var.lab"
[22] "val.lab"          "obs.vars"         "numtocat"
[25] "catgroups"
```

By default, all variables except for the first one in the visit sequence (`visit.sequence`) are synthesised using the `"cart"` method. The first variable to be synthesised cannot have predictors that are to be synthesised later on and therefore a random sample (with replacement) is drawn from its observed values. The default visit sequence reflects the order of variables in the original data set - columns are synthesised from left to right.

The default predictor selection matrix (`predictor.matrix`) is defined by the visit sequence. All variables that are earlier in the visit sequence are used as predictors. A value of `1` in a predictor selection matrix means that the column variable is used as a predictor for the target variable in the row. Since the order of variables is exactly the same as in the original data, for the default visit sequence the default predictor selection matrix has values of `1` in the lower triangle.

Synthesising data with default parametric methods is run with the methods listed below. Values of the other `syn()` arguments remain the same as for the default synthesis.

```
R> sds.parametric <- syn(ods, method = "parametric", seed = my.seed)
```

```
R> sds.parametric$method
```

```
     sex        age        edu    marital     income          ls    wkabint
"sample" "normrank"  "polyreg"  "polyreg" "normrank"   "polyreg"  "polyreg"
```

4.3. Extended example

To extend the simple example presented in Section 4.2 we change the order of synthesis, synthesise only selected variables, customise selection of predictors, handle missing values in a continuous variable and apply some rules that a variable has to follow.

Sequence and scope of synthesis

The default algorithm of synthesising variables in columns from left to right can be changed via the `visit.sequence` argument. The vector `visit.sequence` should include indices of columns in an order desired by a user. Alternatively, names of variables can be used. If we do not want to synthesise some variables we can exclude them from visit sequence. By default those variables are not used to predict other variables but they are saved in the synthesised data. In order to remove their original values from the resulting synthetic data sets an argument `drop.not.used` has to be set to `TRUE`. To synthesize variables `sex`, `age`, `ls`, `marital` and `edu` in this order we run the `syn()` function with the following specification

```
R> sds.selection <- syn(ods, visit.sequence = c(1, 2, 6, 4, 3),
+   seed = my.seed, drop.not.used = TRUE)
```

An appropriate prediction matrix is created automatically. To avoid having to alter other parameters when the visit sequence is changed and to ensure the synthetic data have the same structure as the original ones, the variables in `sds.selection$predictor.matrix` are arranged in the same order as in the original data. The same applies to `sds.selection$method` and synthesised data set `sds.selection$syn`. As noted above, if the parameter `drop.not.used` is set to `TRUE` and there are variables that are not used in synthesis, they are not included in the output. In this case the column indices in visit sequence, which align to the synthetic data columns, may not be the same as in the original data.

```
R> sds.selection

Call:
($call) syn(data = ods, visit.sequence = c(1, 2, 6, 4, 3), drop.not.used = TRUE,
    seed = my.seed)

Number of synthesised data sets:
($m)  1

First rows of synthesised data set:
($syn)
     sex age                      edu marital               ls
1   MALE  72       VOCATIONAL/GRAMMAR MARRIED          PLEASED
2   MALE  82                SECONDARY MARRIED          PLEASED
3   MALE  25 POST-SECONDARY OR HIGHER  SINGLE          PLEASED
4   MALE  69     PRIMARY/NO EDUCATION MARRIED MOSTLY SATISFIED
5 FEMALE  34 POST-SECONDARY OR HIGHER WIDOWED MOSTLY SATISFIED
6 FEMALE  79     PRIMARY/NO EDUCATION WIDOWED            MIXED
...

Synthesising methods:
($method)
     sex      age      edu  marital       ls
"sample"   "cart"   "cart"   "cart"   "cart"

Order of synthesis:
($visit.sequence)
    sex     age      ls marital     edu
```

```
       1       2       5       4       3

Matrix of predictors:
($predictor.matrix)
        sex age edu marital ls
sex       0   0   0       0  0
age       1   0   0       0  0
edu       1   1   0       1  1
marital   1   1   0       0  1
ls        1   1   0       0  0
```

Note that a user-defined `method` vector (setting method for each variable separately) and a specified `predictor.matrix` both have to include information for all variables present in the original observed data set regardless of whether they are in `visit.sequence` or not. This allows changes in `visit.sequence` without adjustments to other arguments. For variables not to be synthesised but still to be used as predictors, which needs to be reflected in a `predictor.matrix`, an empty `method` (`""`) should be set. By default the original observed values of those variables are included in the synthesised data sets but it can be changed using an argument `drop.pred.only`.

Selection of predictors

The most important rule when selecting predictors is that independent variables in a prediction model have to be already synthesised. The only exception is when a variable is used only as a predictor and is not going to be synthesised at all. Assume we want to synthesise all variables except `wkabint` and:

- exclude life satisfaction (`ls`) from the predictors of marital status (`marital`);
- use monthly income (`income`) as a predictor of life satisfaction (`ls`), education (`edu`) and marital status (`marital`) but do not synthesise income variable itself;
- use polytomous logistic regression (`polyreg`) to generate marital status (`marital`) instead of a default `ctree` method.

In order to build an adequate predictor selection matrix, instead of doing it from scratch we can define an initial `visit.sequence` and corresponding `method` vector and run `syn()` function with parameter `drop.not.used` set to `FALSE` (otherwise `method` and `predictor.matrix` will miss information on `wkabint`), parameter `m` indicating number of synthesis set to zero and other arguments left as defaults. Then we can adjust the predictor selection matrix used in this synthesis and rerun the function with new parameters. The R code for this is given below.

```
R> visit.sequence.ini <- c(1, 2, 5, 6, 4, 3)
R> method.ini <- c("sample", "ctree", "ctree", "polyreg", "", "ctree", "")
R> sds.ini <- syn(data = ods, visit.sequence = visit.sequence.ini,
+   method = method.ini, m = 0, drop.not.used = FALSE)

R> sds.ini$predictor.matrix
```

```
        sex age edu marital income ls wkabint
sex       0   0   0       0      0  0       0
age       1   0   0       0      0  0       0
edu       1   1   0       1      1  1       0
marital   1   1   0       0      1  1       0
income    0   0   0       0      0  0       0
ls        1   1   0       0      1  0       0
wkabint   0   0   0       0      0  0       0
```

```
R> predictor.matrix.corrected <- sds.ini$predictor.matrix
R> predictor.matrix.corrected["marital", "ls"] <- 0
R> predictor.matrix.corrected
```

```
        sex age edu marital income ls wkabint
sex       0   0   0       0      0  0       0
age       1   0   0       0      0  0       0
edu       1   1   0       1      1  1       0
marital   1   1   0       0      1  0       0
income    0   0   0       0      0  0       0
ls        1   1   0       0      1  0       0
wkabint   0   0   0       0      0  0       0
```

```
R> sds.corrected <- syn(data = ods, visit.sequence = visit.sequence.ini,
+   method = method.ini, predictor.matrix = predictor.matrix.corrected,
+   seed = my.seed)
```

Handling missing values in continuous variables

Data can be missing for a number of reasons (e.g. refusal, inapplicability, lack of knowledge) and multiple missing data codes are used to represent this variety. By default, numeric missing data codes for a continuous variable are treated as non-missing values. This may lead to erroneous synthetic values, especially when standard parametric models are used or when synthetic values are smoothed to decrease disclosure risk. The problem refers not only to the variable in question, but also to variables predicted from it. The parameter `cont.na` of the `syn()` function allows to define missing-data codes for continuous variables in order to model them separately (see Section 3.2). In our simple example a continuous variable `income` has two types of missing values (`NA` and `-8`) and they should be provided in a list element named `"income"`. The following code shows the recommended settings for synthesis of income variable, which includes smoothing and separate synthesis of missing values

```
R> sds.income <- syn(ods, cont.na = list(income = c(NA, -8)),
+   smoothing = list(income = "density"), seed = NA)
```

Rules for restricted values

To illustrate application of rules for restricted values consider marital status. According to Polish law males have to be at least 18 to get married. Thus, in our synthesised data set all male individuals younger than 18 should have marital status `SINGLE` which is the case in the observed data set. Running without rules gives incorrect results with some of the males under 18 classified as `MARRIED` (see summary output table below).

```
R> M18.ods <- table(subset(ods,
+   age < 18 & sex == "MALE", marital))
R> M18.default <- table(subset(sds.default$syn,
+   age < 18 & sex == "MALE", marital))
R> M18.parametric <- table(subset(sds.parametric$syn,
+   age < 18 & sex == "MALE", marital))
R> cbind("Observed data" = M18.ods, CART = M18.default,
+   Parametric = M18.parametric)
```

```
                   Observed data CART Parametric
SINGLE                        57   62         54
MARRIED                        0    1         13
WIDOWED                        0    0          0
DIVORCED                       0    0          0
LEGALLY SEPARATED              0    0          0
DE FACTO SEPARATED             0    0          0
```

Application of a rule, as specified below using named lists, leads to the correct results

```
R> rules.marital <- list(marital = "age < 18 & sex == 'MALE'")
R> rvalues.marital <- list(marital = "SINGLE")
R> sds.rmarital <- syn(ods, rules = rules.marital,
+   rvalues = rvalues.marital, seed = my.seed)
R> sds.rmarital.param <- syn(ods, rules = rules.marital,
+   rvalues = rvalues.marital, method = "parametric", seed = my.seed)
```

A summary table can be produced using the following code

```
R> rM18.default <- table(subset(sds.rmarital$syn,
+   age < 18 & sex == "MALE", marital))
R> rM18.parametric <- table(subset(sds.rmarital.param$syn,
+   age < 18 & sex == "MALE", marital))
R> cbind("Observed data" = M18.ods, CART = rM18.default,
+   Parametric = rM18.parametric)
```

```
                   Observed data CART Parametric
SINGLE                        57   64         68
MARRIED                        0    0          0
WIDOWED                        0    0          0
DIVORCED                       0    0          0
LEGALLY SEPARATED              0    0          0
DE FACTO SEPARATED             0    0          0
```

4.4. Synthetic data analysis

Ideally, if the models used for synthesis truly represents the process that generated the original observed data, an analysis based on the synthesised data should lead to the same statistical inferences as an analysis based on the actual data. For illustration we estimate here a simple logistic regression model where our dependent variable is a probability of intention to work abroad. We use the `wkabint` variable which specifies the intentions of work migration but we adjust it to disregard the destination country group. Besides we recode the current missing data code of variable `income` (`-8`) into the R missing data code `NA`.

```
R> ods$wkabint <- as.character(ods$wkabint)
R> ods$wkabint[ods$wkabint == "YES, TO EU COUNTRY" |
+   ods$wkabint == "YES, TO NON-EU COUNTRY"] <- "YES"
R> ods$wkabint <- factor(ods$wkabint)
R> ods$income[ods$income == -8] <- NA
```

We generate five synthetic data sets.

```
R> sds <- syn(ods, method = "ctree", m = 5, seed = my.seed)
```

Before running the models let us compare some descriptive statistics of the observed and synthetic data sets. A very useful function in R for this purpose is `summary()`. When a data frame is provided as an argument, here our original data set `ods`, it produces summary statistics of each variable.

```
R> summary(ods)
```

```
     sex            age                                  edu
 MALE  :2182   Min.   :16.0   PRIMARY/NO EDUCATION    : 962
 FEMALE:2818   1st Qu.:32.0   VOCATIONAL/GRAMMAR      :1613
               Median :49.0   SECONDARY               :1482
               Mean   :47.7   POST-SECONDARY OR HIGHER: 936
               3rd Qu.:61.0   NA's                    :   7
               Max.   :97.0

               marital          income                      ls
 SINGLE             :1253   Min.   :  100   PLEASED            :1947
 MARRIED            :2979   1st Qu.:  970   MOSTLY SATISFIED   :1692
 WIDOWED            : 531   Median : 1350   MIXED              : 827
 DIVORCED           : 199   Mean   : 1641   MOSTLY DISSATISFIED: 274
 LEGALLY SEPARATED :   7   3rd Qu.: 2000   DELIGHTED          : 191
 DE FACTO SEPARATED:  22   Max.   :16000   (Other)            :  61
 NA's               :   9   NA's   :1286    NA's               :   8
 wkabint
 NO  :4646
 YES : 318
 NA's:  36
```

The `summary()` function with the `synds` object as an argument gives summary statistics of the variables in the synthesised data set. If more than one synthetic data set has been generated, as default summaries are calculated by averaging summary values for all synthetic data copies.

```
R> summary(sds)

Synthetic object with 5 syntheses using methods:
     sex      age      edu  marital   income       ls  wkabint
"sample"  "ctree"  "ctree"  "ctree"  "ctree"  "ctree"  "ctree"

Summary (average) for all synthetic data sets:
     sex           age                              edu
 MALE  :2168   Min.   :16.0   PRIMARY/NO EDUCATION    : 965
 FEMALE:2832   1st Qu.:31.8   VOCATIONAL/GRAMMAR      :1620
               Median :48.6   SECONDARY               :1491
               Mean   :47.6   POST-SECONDARY OR HIGHER: 919
               3rd Qu.:61.6   NA's                    :   6
               Max.   :96.4

               marital          income                        ls
 SINGLE             :1275.2   Min.   :  100   PLEASED            :1935.8
 MARRIED            :2957.2   1st Qu.:  952   MOSTLY SATISFIED   :1703.2
 WIDOWED            : 531.6   Median : 1334   MIXED              : 829.6
 DIVORCED           : 201.2   Mean   : 1632   MOSTLY DISSATISFIED: 273.6
 LEGALLY SEPARATED :   7.0   3rd Qu.: 1980   DELIGHTED          : 186.4
 DE FACTO SEPARATED:  22.0   Max.   :15800   (Other)            :  64.0
 NA's               :   5.8   NA's   : 1285   NA's               :   7.4
 wkabint
 NO  :4656.6
 YES : 309.8
 NA's:  33.6
```

Summary of individual data sets can be displayed by supplying the `msel` parameter, which can be a single number or a vector with selected synthesis numbers. An example code is presented below but the corresponding output is suppressed for space reasons.

```
R> summary(sds, msel = 2)
R> summary(sds, msel = 1:5)
```

To compare the synthesised variables with the original ones more easily, the synthesiser can use a `compare()` function. It is a generic function for comparison of various aspects of synthesised and observed data. The function invokes particular methods depending on the class of the first argument. If a synthetic data object and a data frame with original data are provided it compares relative frequency distributions of each variable in tabular and graphic form. The number of plots per page can be specified via `nrow` and `ncol` arguments. Alternatively, the

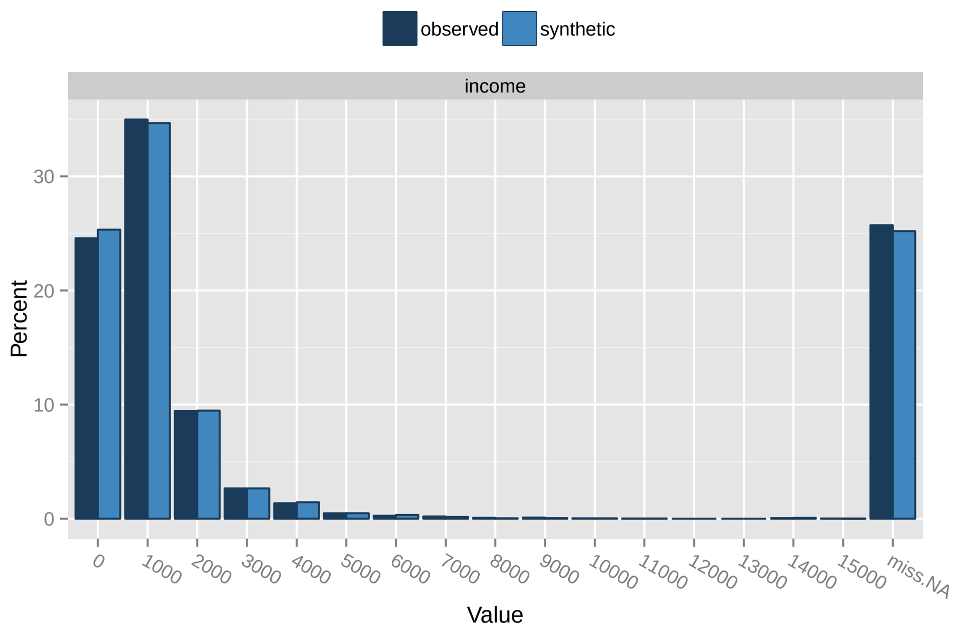

Figure 1: Relative frequency distribution of non-missing values and missing data categories for income variable for observed and synthetic data.

function can be used for a subset of variables specified by a `vars` argument. Output for `income` is presented below and in Figure 1. For quantitative variables, such as `income`, missing data categories are plotted on the same plot as non-missing values and they are indicated by `miss.` suffix. If a synthetic data object contains multiple data sets by default pooled synthetic data are used for comparison.

```
R> compare(sds, ods, vars = "income")

Comparing percentages observed with synthetic

Selected utility measures:
          pMSE S_pMSE df
income 8.8e-05  1.416  5
```

An argument `msel` can be used to compare the observed data with a single or multiple individual synthetic data sets, which is illustrated below and in Figure 2 for a life satisfaction factor variable (`ls`).

```
R> compare(sds, ods, vars = "ls", msel = 1:3)

Comparing percentages observed with synthetic

Selected utility measures:
      pMSE S_pMSE df
ls 7.5e-05 0.8612  7
```

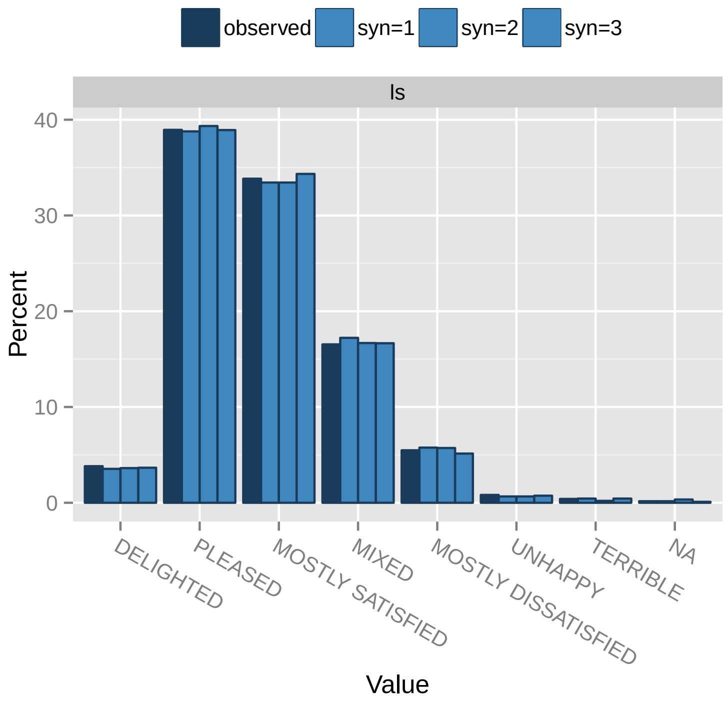

Figure 2: Relative frequency distribution of life satisfaction (`ls`) for observed and synthetic data.

Returning to the logistic regression model for `wkabint`, we estimate the original data model using generalised linear models implemented in R `glm()` function. A **synthpop** package function `glm.synds()` is an equivalent function for estimating models for each of the `m` synthesised data sets. A similar function called `lm.synds()` is available for a standard linear regression model. An outcome of `glm.synds()` and `lm.synds()` function is an object of class '`fit.synds`'. If `m > 1`, printing a '`fit.synds`' object gives the combined (average) coefficient estimates. Results for coefficient estimates based on individual synthetic data sets can be displayed using an `msel` argument of a `print` method.

```
R> model.ods <- glm(wkabint ~ sex + age + edu + log(income),
+   family = "binomial", data = ods)
R> model.ods

Call:  glm(formula = wkabint ~ sex + age + edu + log(income), family = "binomial",
    data = ods)

Coefficients:
                (Intercept)                          sexFEMALE
                -0.2105                        -0.4739
                    age          eduVOCATIONAL/GRAMMAR
                -0.0538                         0.6275
           eduSECONDARY  eduPOST-SECONDARY OR HIGHER
                 0.3684                        -0.1870
            log(income)
                -0.0461

Degrees of Freedom: 3690 Total (i.e. Null);  3684 Residual
  (1309 observations deleted due to missingness)
Null Deviance:          1540
Residual Deviance: 1370        AIC: 1390

R> model.sds <- glm.synds(wkabint ~ sex + age + edu + log(income),
+   family = "binomial", data = sds)
R> model.sds

Note: To get more details of the fit see vignette on inference.

Call:
glm.synds(formula = wkabint ~ sex + age + edu + log(income),
    family = "binomial", data = sds)

Average coefficient estimates from 5 syntheses:
            (Intercept)                     sexFEMALE
               -1.06343                      -0.66178
                    age         eduVOCATIONAL/GRAMMAR
               -0.05138                       0.17467
           eduSECONDARY eduPOST-SECONDARY OR HIGHER
                0.13112                      -0.01006
            log(income)
                0.09112
```

The `summary()` function of a `fit.synds` object can be used by the analyst to combine estimates based on all the synthesised data sets. By default inference is made to original data quantities. In order to make inference to population quantities the parameter `population.inference` has to be set to `TRUE`. The function's result provides point estimates of coefficients (`B.syn`), their standard errors (`se(B.syn)`) and Z scores (`Z.syn`) for population and observed data quantities respectively. For inference to original data quantities it contains in addition estimates of the actual standard errors based on synthetic data (`se(Beta).syn`) and standard errors of Z scores (`se(Z.syn)`). Note that not all these quantities are printed automatically.

The mean of the estimates from each of the `m` synthetic data sets yields unbiased estimates of the coefficients if the data conform to the model used for synthesis. The variance is estimated differently depending whether inference is made to the original data quantities or the population parameters and whether synthetic data were produced using simple or proper synthesis (for details see Raab *et al.* 2016; expressions used to calculate variance for different cases are presented in Table 1). By default a simple synthesis is conducted and inference is made to original data quantities.

```
R> summary(model.sds)

Fit to synthetic data set with 5 syntheses. Inference to coefficients
and standard errors that would be obtained from the original data.

Call:
glm.synds(formula = wkabint ~ sex + age + edu + log(income),
    family = "binomial", data = sds)

Combined estimates:
                            xpct(Beta) xpct(se.Beta) xpct(z) Pr(>|xpct(z)|)
(Intercept)                   -1.06343       0.98644   -1.08           0.28
sexFEMALE                     -0.66178       0.16326   -4.05          5e-05
age                           -0.05138       0.00547   -9.40         <2e-16
eduVOCATIONAL/GRAMMAR          0.17467       0.29220    0.60           0.55
eduSECONDARY                   0.13112       0.30020    0.44           0.66
eduPOST-SECONDARY OR HIGHER   -0.01006       0.33359   -0.03           0.98
log(income)                    0.09112       0.13231    0.69           0.49

(Intercept)
sexFEMALE                   ***
age                         ***
eduVOCATIONAL/GRAMMAR
eduSECONDARY
eduPOST-SECONDARY OR HIGHER
log(income)
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Function `compare()` allows the synthesiser to compare the estimates based on the synthesised data sets with those based on the original data and presents the results in both tabular and graphical form (see Figure 3).

```
R> compare(model.sds, ods)

Call used to fit models to the data:
glm.synds(formula = wkabint ~ sex + age + edu + log(income),
    family = "binomial", data = sds)

Differences between results based on synthetic and observed data:
                            Synthetic Observed      Diff Std. coef diff
(Intercept)                  -1.06343 -0.21052 -0.852916        -0.9570
```

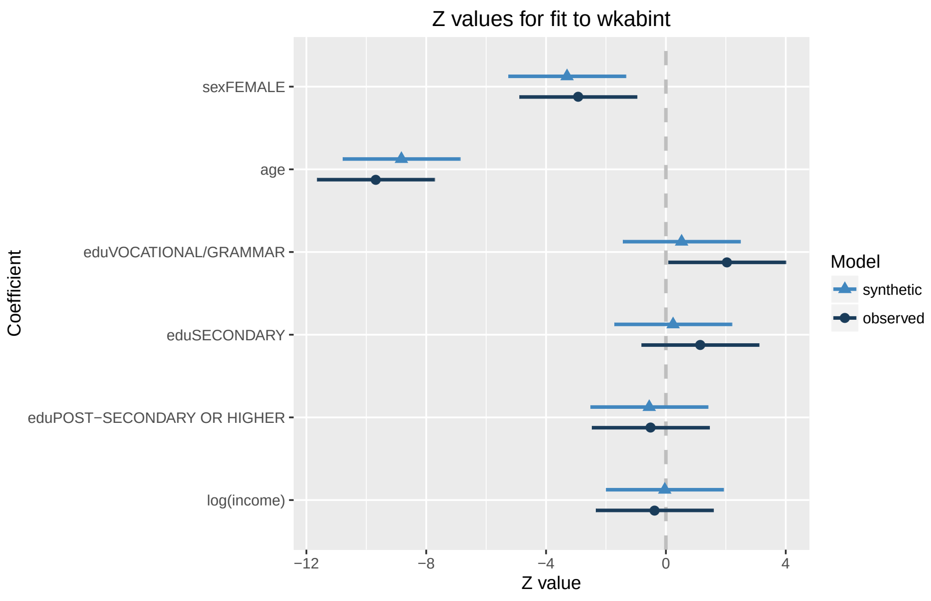

Figure 3: Estimates and 95% confidence intervals for Z statistics from a logistic regression of intention to go abroad to work for observed and synthetic data.

```
sexFEMALE                    -0.66178 -0.47387 -0.187908        -1.1612
age                          -0.05138 -0.05384  0.002455         0.4416
eduVOCATIONAL/GRAMMAR         0.17467  0.62753 -0.452859        -1.4723
eduSECONDARY                  0.13112  0.36839 -0.237265        -0.7386
eduPOST-SECONDARY OR HIGHER  -0.01006 -0.18697  0.176902         0.4821
log(income)                   0.09112 -0.04610  0.137224         1.1226
                            CI overlap
(Intercept)                     0.7559
sexFEMALE                       0.7038
age                             0.8873
eduVOCATIONAL/GRAMMAR           0.6244
eduSECONDARY                    0.8116
eduPOST-SECONDARY OR HIGHER     0.8770
log(income)                     0.7136

Measures for 5 syntheses and 7 coefficients
Mean confidence interval overlap:  0.7677
Mean absolute std. coef diff:  0.9108

Mahalanobis distance ratio for lack-of-fit (target 1.0): 10.21
Lack-of-fit test: 71.5; p-value 0 for test that synthesis model is compatible
with a chi-squared test with 7 degrees of freedom.
```

From both original and synthetic data we conclude that men are more likely to declare intention to work abroad as are those who are young. The fact that the results from synthetic data can have a similar pattern to the results from the real data is encouraging for further developments of synthetic data tools.

5. Concluding remarks

In this paper we presented the basic functionality of the R package **synthpop** for generating synthetic versions of microdata containing confidential information so that they are safe to be released to users for exploratory analysis. Interested readers can consult the package documentation for additional features currently implemented which can be used to influence the disclosure risk and the utility of the synthesised data. Note that **synthpop** is under continual development and future versions will include, among others, appropriate procedures for synthesising multiple event data, conducting stratified synthesis and replacing only selected cases from selected variables. The ultimate aim of **synthpop** is to provide a comprehensive, flexible and easy to use tool for generating bespoke synthetic data that can be safely released to interested data users. Since there are many different options to synthesise data, developing general guidelines for best practice remains an open issue to be addressed in our future research.

References

Abowd JM, Lane J (2004). "New Approaches to Confidentiality Protection: Synthetic Data, Remote Access and Research Data Centers." In J Domingo-Ferrer, V Torra (eds.), *Privacy in Statistical Databases, 2004*, pp. 282–289. Springer-Verlag.

Abowd JM, Woodcock SD (2004). "Multiply-Imputing Confidential Characteristics and File Links in Longitudinal Linked Data." In J Domingo-Ferrer, V Torra (eds.), *Privacy in Statistical Databases, 2004*, pp. 290–297. Springer-Verlag.

Alfons A, Kraft S (2013). ***simPopulation**: Simulation of Synthetic Populations for Survey Surveys Based on Sample Data.* R package version 0.4.1, URL `https://CRAN.R-project.org/src/contrib/Archive/simPopulation/`.

Alfons A, Kraft S, Templ M, Filzmoser P (2011). "Simulation of Close-to-Reality Population Data for Household Surveys with Application to EU-SILC." *Statistical Methods & Applications*, **20**(3), 383–407. `doi:10.1007/s10260-011-0163-2`.

Boyle P, Feijten P, Feng Z, Hattersley L, Huang Z, Nolan J, Raab GM (2012). "Cohort Profile: The Scottish Longitudinal Study (SLS)." *International Journal of Epidemiology*, **38**(2), 385–392.

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984). *Classification and Regression Trees.* Belmont, Wadsworth.

Caiola G, Reiter JP (2010). "Random Forests for Generating Partially Synthetic, Categorical Data." *Transactions on Data Privacy*, **3**(1), 27–42.

Council for Social Monitoring (2011). "Social Diagnosis 2000–2011: Integrated Database." URL `http://www.diagnoza.com/index-en.html`.

Drechsler J (2011). *Synthetic Data Sets for Statistical Disclosure Control.* Springer-Verlag. doi:10.1007/978-1-4614-0326-5.

Drechsler J, Reiter JP (2010). "Sampling with Synthesis: A New Approach for Releasing Public Use Census Microdata." *Journal of the American Statistical Association*, **105**(492), 1347–1357. doi:10.1198/jasa.2010.ap09480.

Drechsler J, Reiter JP (2011). "An Empirical Evaluation of Easily Implemented, Nonparametric Methods for Generating Synthetic Datasets." *Computational Statistics & Data Analysis*, **55**(12), 3232–3243. doi:10.1016/j.csda.2011.06.006.

Elliot M (2015). "Final Report on the Disclosure Risk Associated with the Synthetic Data Produced by the SYLLS Team." *Report 2015-2*, Cathie Marsh Institute for Social Research (CMIST), University of Manchester. URL http://www.cmist.manchester.ac.uk/research/publications/reports.

Elliot M, Purdam K (2007). "A Case Study of the Impact of Statistical Disclosure Control on Data Quality in the Individual UK Samples of Anonymized Records." *Environment and Planning A*, **39**(5), 1101–1118. doi:10.1068/a38335.

Hattersley L, Cresser R (1995). "Longitudinal Study, 1971–1991: History, Organisation and Quality of Data." *LS Series 7*, HMSO, London. URL http://celsius.lshtm.ac.uk/documents/LS%20No.7%20Hattersley%20&%20Creeser%201995.pdf.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi:10.1198/106186006x133933.

Kinney SK, Reiter JP, Berger JO (2010). "Model Selection When Multiple Imputation is Used to Protect Confidentiality in Public Use Data." *Journal of Privacy and Confidentiality*, **2**(2), 3–19.

Kinney SK, Reiter JP, Reznek AP, Miranda J, Jarmin RS, Abowd JM (2011). "Towards Unrestricted Public Use Business Microdata: The Synthetic Longitudinal Business Database." *International Statistical Review*, **79**(3), 362–384. doi:10.1111/j.1751-5823.2011.00153.x.

Little RJA (1993). "Statistical Analysis of Masked Data." *Journal of Official Statistics*, **9**(2), 407–26.

Meindl B, Templ M, Alfons A, Kowarik A (2016). ***simPop**: Simulation of Synthetic Populations for Survey Data Considering Auxiliary Information.* R package version 0.3.0, URL https://CRAN.R-project.org/package=simPop.

Nowok B, Raab GM, Snoke J, Dibben C (2016). ***synthpop**: Generating Synthetic Versions of Sensitive Microdata for Statistical Disclosure Control.* R package version 1.3-0, URL https://CRAN.R-project.org/package=synthpop.

Ohm P (2010). "Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization." *UCLA Law Review*, **57**(6), 1701–1775.

O'Reilly D, Rosato M, Catney G, Johnston F, Brolly M (2011). "Cohort Description: The Northern Ireland Longitudinal Study (NILS)." *International Journal of Epidemiology*, **41**(3), 634–641.

Raab GM, Nowok B, Dibben C (2016). "Practical Data Synthesis for Large Samples." arXiv:1409.0217 [stat.ME], URL http://arxiv.org/abs/1409.0217.

Raghunathan TE, Reiter JP, Rubin DB (2003). "Multiple Imputation for Statistical Disclosure Limitation." *Journal of Official Statistics*, **19**(1), 1–17.

Raghunathan TE, Solenberger PW, Van Hoewyk J (2002). "**IVEware**: Imputation and Variance Estimation Software User Guide." *Technical report*, Survey Methodology Program, Survey Research Center, Institute for Social Research, University of Michigan. URL http://www.isr.umich.edu/src/smp/ive/.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Reiter JP (2002). "Satisfying Disclosure Restrictions with Synthetic Data Sets." *Journal of Official Statistics*, **18**(4), 531–544.

Reiter JP (2003). "Inference for Partially Synthetic, Public Use Microdata Sets." *Survey Methodology*, **29**(2), 181–188.

Reiter JP (2005a). "Releasing Multiply Imputed, Synthetic Public Use Microdata: An Illustration and Empirical Study." *Journal of the Royal Statistical Society A*, **168**(1), 185–205. doi:10.1111/j.1467-985x.2004.00343.x.

Reiter JP (2005b). "Using CART to Generate Partially Synthetic, Public Use Microdata." *Journal of Official Statistics*, **21**(3), 441–462.

Reiter JP, Kinney SK (2012). "Inferentially Valid, Partially Synthetic Data: Generating from Posterior Predictive Distributions Not Necessary." *Journal of Official Statistics*, **28**(4), 583–590.

Reiter JP, Raghunathan E (2007). "The Multiple Adaptations of Multiple Imputation." *Journal of the American Statistical Society*, **102**(480), 1462–1471. doi:10.1198/016214507000000932.

Rubin DB (1993). "Discussion: Statistical Disclosure Limitation." *Journal of Official Statistics*, **9**(2), 461–468.

SAS Institute Inc (2013). *SAS Software, Version 9.4*. Cary. URL http://www.sas.com/.

Survey Methodology Program (2011). "**IVEware**: Imputation and Variance Estimation Version 0.2 Users Guide (Supplement)." *Technical report*, Survey Research Center, Institute for Social Research, University of Michigan. URL http://www.isr.umich.edu/src/smp/ive/.

Therneau T, Atkinson B, Ripley B (2015). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-10, URL https://CRAN.R-project.org/package=rpart.

Van Buuren S, Groothuis-Oudshoorn K (2011). "**mice**: Multivariate Imputation by Chained Equations in R." *Journal of Statistical Software*, **45**(3), 1–67. doi:10.18637/jss.v045.i03.

Winkler WE (2007). "Examples of Easy-to-Implement, Widely Used Methods of Masking for Which Analytical Properties Are Not Justified." *Technical Report Series 21*, Statistical Research Division, U.S. Census Bureau, Washington. URL http://www.census.gov.edgekey.net/srd/papers/pdf/rrs2007-21.pdf.

Affiliation:

Beata Nowok
Institute of Geography
School of GeoSciences
University of Edinburgh
Drummond Street
Edinburgh EH8 9XP, United Kingdom
E-mail: beata.nowok@ed.ac.uk