Generating a longitudinal synthetic population for educational and research purposes

Jack G. Gambino*

Abstract

We create a synthetic population with enough structure for survey statisticians and students to use to (1) illustrate existing sampling and estimation methods and (2) try out new methods in a realistic setting. This paper describes how the population is generated using `R` and how the result can be used. The population includes some basic static variables (age, sex, education level) and both discrete (labour force status) and continuous (income) analysis variables that change over time. The population evolves over many months in a way that is sufficiently realistic for our goals. The population has enough structure (strata, clusters) to make it useful for both teaching, learning and illustrating sample survey methods and also as a testbed for trying out new sampling methods and estimators. The default variables are sufficient for the purposes we have envisaged, but the user can easily add new variables, using the existing `R` code as a guide.

Keywords: Simulated population; survey sampling education; survey sampling research.

1 Introduction

Our goal in creating `poptimesimul` is to simulate a population with enough structure, across both space and time, to make it possible for survey statisticians and students to use the population in two ways: (1) to illustrate

*Jack G. Gambino, Statistics Canada (retired) and Statistical Society of Canada, Ottawa, Canada; jack.gambino@gmail.com

existing sampling and estimation methods and (2) to try out new methods in a realistic setting. Thus the synthetic population should have value as both a pedagogical tool and a research tool. To be useful in these ways, the population must have some features that allow for more than just simple random sampling of ultimate population units ("people" in our case). Thus our population units are grouped into clusters which are themselves grouped into strata that manifest socioeconomic differences.

To achieve our goal, we produce a *census*, i.e., the whole population (not just a sample), and then make it change over time. Our approach to generating the initial population, explained below, is quite simple but suitable for our specific goals. The `R` source code is described in Appendix 1 below. More elaborate approaches are available. For example, the `R` package SimCorrMix (Fialkowski and Tiwari (2019)) uses a more sophisticated approach to generate both discrete and continuous mixture variables that are correlated.

The key characteristics of our population units will be labour force status and income, along with auxiliary information: age, sex and education. For our limited goals, only the former variables will evolve over time.

As we noted, our goal is to create a synthetic population that has both pedagogical and research value. We briefly elaborate on what we mean by each of these. Examples of the population's pedagogical uses include the following:

- The population can be used to teach the theory of survey sampling, both basic (simple random sampling) and complex (sampling in two stages, PPS (probability proportional to size) sampling), using a realistic sampling frame.

- It can then be used to teach, illustrate and compare estimation methods for surveys.

- It can be used to illustrate time series methods. Because we have the complete population, we can draw many samples from it to show how sampling variability impacts time series methods. In practice, many time series methods treat the sequence of estimates as true values, not taking into account the fact that they are, in fact, estimates subject to sampling error (see Bell and Kramer (1999); see also the next item).

Some examples of research uses include:

- The population can be used to conduct an empirical study of new sampling methods and associated estimation methods. This may be particularly useful for variance estimation methods. Note that, in effect, if, say, 120 months of the population are generated, we have 120 censuses—a luxury that does not exist for most real populations. As a result we know the "truth" for all months, making it possible to compute true sampling errors and variances.

- By introducing nonresponse into the sampling process, the previous point can be extended to include studies of the impact of nonresponse (and imputation methods) on estimates.

Readers interested in other efforts related to ours can use the brief review in Appendix 3 below as a starting point. In Section 2, the initial synthetic population is described. Then Section 3 explains how the initial population changes over time. In Section 4, we give examples that illustrate how the population, both for one month and longitudinally, can be used. We end with some concluding remarks and possible extensions in Section 5. There are three appendices. Appendix 1 gives some details on the `R` source code. Users who want to dive into using the programs to produce a synthetic population can go straight to Appendix 1, which describes the necessary steps, but we recommend also looking at Section 3 to understand the structure of the population and the variables that will be created. Appendix 2 describes how transition probabilities are manipulated to produce realistic results. Finally, Appendix 3 briefly describes some related work.

WHERE TO GET THE PROGRAMS: The latest version of the programs is available in the GitHub repository `https://github.com/Jack-Gambino/poptimesimul`. There is extensive documentation in the `doc` directory, including a *Quick Start* document that corresponds to Appendix 1 below. The `doc` directory also includes the file `00what_to_do_first.txt` for the really impatient.

2 Description of the synthetic population

It is not difficult to generate a basic labour force population. The challenge is to "age" the population month by month in a realistic fashion. The transitions between employment and unemployment and the transitions into and

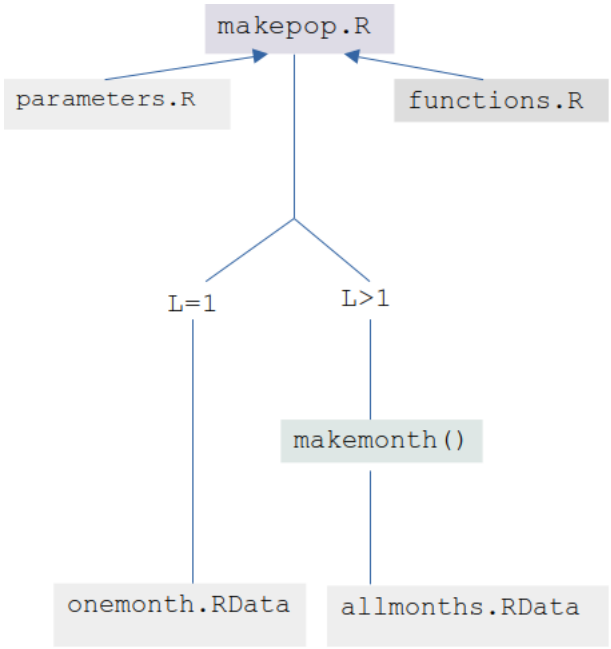

Figure 1: Functions, parameters and output files

out of the labour force should evolve in such a way that the micro-level transitions produce population units that roll up to realistic time series at the macro (population) level. In this section, we describe the features of the initial population.

For the following description, it is useful to refer to Figure 1 to understand the relationship between `R` programs, functions and parameters[1] and the output that will contain the population.

We can think of our population as the people living in a geographical area (a city or a region within a province, state or country). By default the population will consist of a little over 300,000 people. As explained later, when we refer to people here, we mean people aged 15 and over. These people live in geographical clusters which, in the urban case, can be thought of as small neighbourhoods of about M_0 people on average (by default, $M_0 = 400$). The clusters are grouped into H strata ($H = 10$ by default). By design, stratum 1 will have the lowest average income, the lowest average education

[1]By parameters we mean both function parameters and other values such as average cluster size, number of strata, and so on.

level and the highest unemployment rate. Socioeconomic status increases gradually from stratum 1 to stratum H. Therefore stratum H will have the highest average income, and so on.

Each person will have several associated variables: *age*, *sex*, *education level*, *employed* (0 or 1), *unemployed* (0 or 1), *income* and an income-like variable called *y_icc*. We now describe each of these variables.

The age variable is an integer between 15 and 100, with 100 denoting age greater than or equal to 100. The relative frequencies for the age and sex variables are based on those from the 2010 United States census. For our purposes, we do not generate people with age less than 15 but they can be easily added.

The education variable has four levels, denoting a person's highest level of education: no high school diploma, high school diploma, community college or equivalent, university degree. Their relative frequencies are based on those observed in Canada (based on recent estimates from the Canadian Labour Force Survey). The mean education level increases by stratum, with people in stratum 1 having the lowest mean educational attainment and those in stratum H having the highest.

The labour force variables are *employed* and *unemployed*. A person who is neither employed nor unemployed (i.e., employed = unemployed = 0) is said to be *inactive* or *not in the labour force*. The initial employment and unemployment probabilities are based on estimates from the 2021 Canadian Labour Force Survey (LFS) for age groups 15–19, 20–24, 25–54, 55–64, 65+, for males and females. These probabilities are adjusted to make them vary by stratum: by default, for employment, they go from 90 percent of the population average for stratum 1 to 110 percent of the average for stratum H. For unemployment they go in the opposite direction.

The mean annual income is set arbitrarily at 50 thousand by default. It is made to vary by stratum: by default, the mean for stratum 1 is 80 percent of the overall mean, for stratum H it is 120 percent, with a linear trend for strata 2 to $H-1$. To make the generated income distribution look somewhat realistic, it was generated in steps. For the clusters in each stratum, a cluster mean income was generated using the normal distribution. Therefore the clusters in each stratum have the same overall (stratum-level) mean income but the actual means vary by cluster. Then, within each cluster, the income for each individual is generated using the gamma distribution.

The income variable is first generated as an annual value and later converted into a monthly one. This is done so that an individual's income can

change immediately if his or her labour force status changes. In addition, at the outset, a random subset of people who are not employed (`emp=0`) have their income set to zero.

To make the synthetic population useful for the study of complex survey designs, we also generate an income-like variable, *y_icc*, that has a positive intracluster correlation. The steps to generate it are similar to those in the previous paragraph except for the last step. Instead of using a gamma distribution, we use a multivariate normal distribution with a positive correlation value in the off-diagonal elements of the correlation matrix. Doing this for each cluster induces a positive intracluster correlation, i.e., people in the same cluster tend to be more similar than across clusters.

Once the initial population is generated, each person will have the following variables.

Variable	Description
stratum	Stratum code: $h = 1,\ 2,\ \ldots,\ H$
cluster	Cluster code: $1,\ 2,\ \ldots,\ C(h)$ for the $C(h)$ clusters in stratum h
age	Age: $15,\ 16,\ \ldots,\ 100$
agegroup	1: 15–19, 2: 20–24, 3: 25–54, 4: 55–64, 5: 65+
sex	0 = female, 1 = male
educ	Education level 1 to 4
emp	emp = 1 if employed, 0 otherwise
unemp	unemp = 1 if unemployed, 0 otherwise
inc	Personal income
y_icc	Income-like variable with intracluster correlation

Socioeconomic variables vary by stratum, as described above. If the default parameters are used, then the employment rate in month 1 varies from 53.5% in stratum 1 to 65.0% in stratum 10 ($H = 10$ by default). Similarly, the unemployment to population ratio varies from 5.1% to 4.4%. As level of education increases, so does mean income: there is a big jump in mean income between people without a high school diploma and those who have one. There is another increase, but somewhat smaller, for people who have a university degree.

3 Aging the population

The next step towards achieving our goal is to make the population evolve over time (month by month in our case). We want employment, unemployment and income to change gradually in a realistic way. Because we can obtain actual transition probabilities for labour force status, we can achieve our goal for those variables. For income, the necessary information for monthly changes in income (at the person level) is harder to come by. Therefore we have developed an approach that changes income based on changes in employment status.

For monthly changes in labour force status we start with basic transition probabilities based on data from the Canadian LFS (see Singh and Rao (1995)). These basic probabilities are adjusted slightly by stratum (similar to the adjustments for income discussed above). For example, after adjustment, employed people in the highest income stratum (stratum H) are more likely to stay employed compared to people in the lowest income stratum (stratum 1).

Next, month by month, we adjust the transition probabilities using adjustment factors f_E and f_U in such a way that when we roll up the monthly *employment* and *unemployment* indicators to the population we get a time series that follows a real time series. By default, we have used seasonally unadjusted estimates of total employment and total unemployment from the Canadian LFS for the period January 2000 to December 2009[2]. However, any similar time series can be used as "benchmarks". Appendix 2 explains how the monthly adjustments f_E and f_U are computed.

Each month, a person who is employed (`emp=1`) either stays employed, becomes unemployed (`emp` goes to 0 and `unemp` goes from 0 to 1) or leaves the labour force (`emp` becomes 0 and `unemp` stays at 0), using the above transition probabilities. The process for people who are currently unemployed or not in the labour force is the same (but with different probabilities, of course).

For the income variable, the process is different. We have already noted that the annual income that was generated initially was converted into a monthly income. Each month, this income may change based on a number of factors.

First, a small percentage of the people classified as employed (one percent

[2]This period is interesting because it includes the impact of the 2007–2008 financial crisis on labour force estimates.

by default) will have their monthly income increase due to a "promotion" (or other reason). By default, the increase for a person getting one is a random value between 5 and 20 percent.

Second, people who change employment status (e.g., an unemployed person becomes employed or an employed person leaves the labour force) will have their income changed. For employed people who become "not employed" (i.e., they become unemployed or leave the labour force), the income is decreased by a random value in a reasonable range (20 to 60 percent by default). For people who become employed, a new income is imputed based on the original income that was generated for all population units *before* some were set to 0 (we do not want to impute a zero income for someone who just got a job).

Finally, the `R` program includes an inflation factor which, by default, is set to 1 (no inflation). Note that this is a *monthly* inflation factor. Therefore, to introduce inflation into the process, one must remember to take the twelfth root of an annual inflation rate (e.g., $1.05^{1/12}$ if the annual inflation rate is 5 percent). We have not found it necessary to introduce an inflation factor greater than 1 because the other adjustments produce reasonable looking time series without it.

The final population, or more precisely, the sequence of L monthly snapshots of the population ($L = 120$ months by default) looks quite similar to a real population. For each of the variables that evolve over time (employment, unemployment and income), the population-level time series look like real ones. Figure 2 shows the *employment* time series.

Figure 2 was created by summing the employment (`emp`) indicator variable each month (producing the true employment time series) and feeding the result to `R`'s `decompose()` function. The top graph shows the original time series. The graph below it shows the trend line. Below that are the (very obvious) seasonal factors and finally the irregular component. By design, the original series is virtually the same as the employment series observed in Canada in 2000–2009.

4 Using the synthetic population

In this section, we give examples of how the population can be used to study or illustrate sample survey methods. First we describe how the initial population (month 1) can be used on its own and then discuss using the

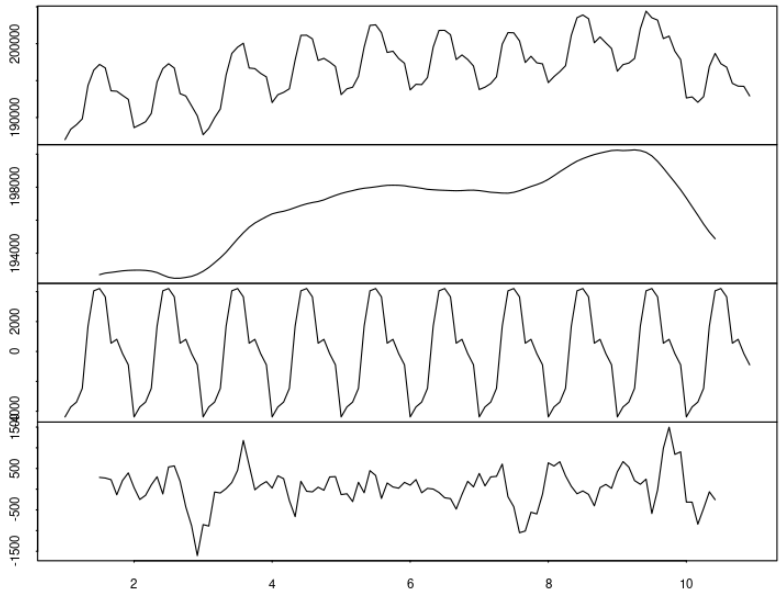

Figure 2: Time series decomposition of *Employment*

whole series over many months.

4.1 Using the initial population

The initial population (or, in fact, the population for any single month) can be used to illustrate many of the survey sampling and estimation methods covered in a course in survey statistics (for a modern treatment see Lohr (2022)). The user is not limited to the information immediately available in the output file. For example, the *stratum* variable can be ignored, and users can create their own strata based on, say, the *income* variable (in fact this is what we do in the second example below). Because the ultimate units in the population are grouped into clusters, it can also be used to study two-stage sample designs and related estimation methods. To study two-stage unequal probability sampling, such as PPS (probability proportional to size) sampling, instead of using the number of people in each cluster as the measure of size, it may be more interesting to use a different size measure, such as mean cluster income.

4.1.1 Example: Over-estimation of variance assuming sampling was done with replacement

In multistage designs, the variance estimation methods used in practice often make the assumption (perhaps implicitly) that the first-stage sampling units (e.g., clusters) are selected using sampling *with* replacement even though the units are actually selected *without* replacement. This is a reasonable approximation if the first-stage sampling fraction is small. It is also a conservative approach because it tends to overestimate the variance. We can use our synthetic population to illustrate this. We do so in the simplest possible case: a sample of clusters is selected in each stratum using simple random sampling without replacement (SRSWOR) and then a sample of people is selected in each cluster, also using SRSWOR. We compute the variance assuming clusters were selected using simple random sampling *with* replacement (SRSWR). Because we have the whole population, we know the true variance; therefore, we can evaluate the degree of overestimation introduced by the WR assumption.

We ran two sets[3] of sampling simulations. In the first, we used a single

[3]The `R` programs for the two sets of simulations have file names beginning with `example_var_with_replacement`.

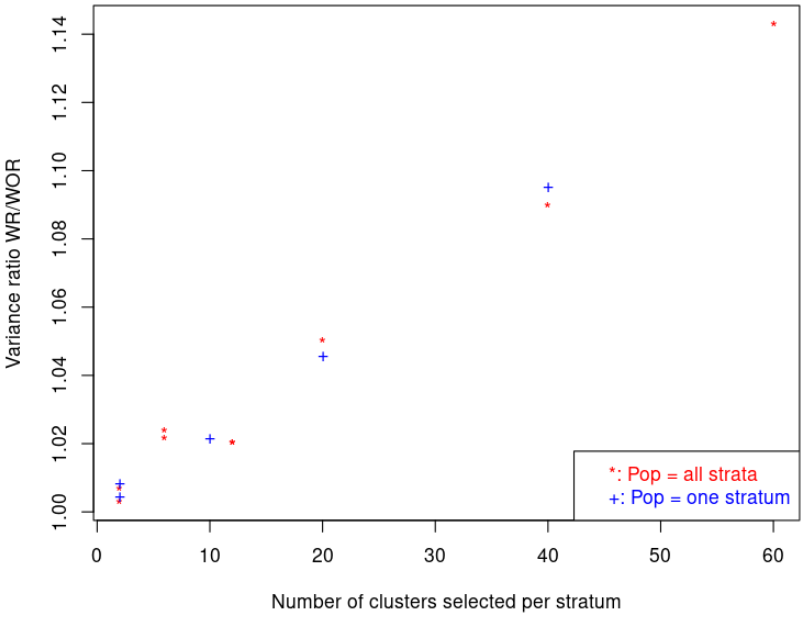

Figure 3: Overestimation of variance due to WR assumption as sampling fraction increases

stratum as our population (in blue in the graph) and in the second, we used the whole population (all H strata, in red). The variable of interest was mean income.

The results are presented in Figure 3. The y axis is the ratio of the two variances. We see clearly that the degree of overestimation increases as the sampling fraction (number of clusters selected in each stratum) increases (the number of clusters in most strata is 75 or 80). We observe that for this population and sampling method, the degree of overestimation is noticeable but not severe, even for big sampling fractions.

This is still an active area of research (see Beaumont and Émond (2022) and Bessonneau et al. (2022)); there are many ways of selecting first-stage units, and the results may depend on both the sampling method and the variables of interest.

4.2 Using the longitudinal population

As we mentioned in Section 2, it is not difficult to generate a synthetic population at one point in time. The challenge is to make that population evolve in a realistic way. Once we achieve that, the opportunities to use the population for both teaching and research increase substantially. In this section, we give an example of this.

4.2.1 Example: Stratification deteriorates over time

One of the key steps in designing a survey is stratifying the population into relatively homogeneous strata. This is usually done using historical data such as the results of a recent census. A good stratification will result in improvement in the quality (reduced variance) of key estimates. However, a population changes over time and, as a result, what started out as a good stratification is likely to become less efficient over time. This is particularly true in business surveys because businesses grow and shrink, thrive and go bankrupt over fairly short time frames. While our synthetic population is one of people, not businesses, it can still be used to illustrate the deterioration in efficiency.

To do this, we create a very good stratification for income using the whole month 1 population. The sample is allocated among strata using Neyman allocation[4]. We then look at the quality of income estimates at months 1,

[4]For the student, it is interesting to see how dramatically the sample size allocated to

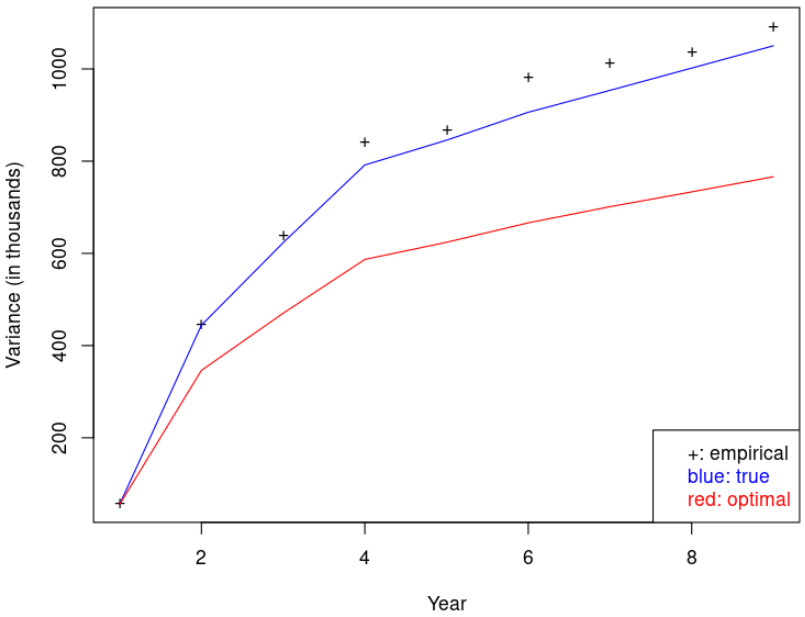

Figure 4: Deterioration of stratification efficiency over time

13, 25, etc., keeping the month 1 stratification and allocation throughout. For comparison, for each of these months we compute the variance using a "fresh" sample allocation using population data for the particular month—a luxury that rarely exists in the real world.

Because we have the whole population for each month, we know the "truth" and can compute the actual variance of the estimated mean income. In Figure 4, the blue line is the true variance and the `+` signs show the approximate (Monte Carlo) variances. The red line shows the variances we would have observed in the unrealistic situation where we could redo the allocation each year. The source code for this example is in the file `example_stratification_deterioration.R` and is replicated in the corresponding `.Rmd` file.

strata differs even though the strata are designed to have the same number of people.

5 Conclusion and extensions

We have produced `R` code that generates a population that includes basic variables (age, sex, education level) and variables that change over time, both discrete (labour force status) and continuous (income). The population evolves over many months in a realistic way (realistic for our goals; more on this below). The population has enough structure (strata, clusters) to make it useful for both teaching, learning and illustrating sample survey methods and also as a testbed for trying out new sampling methods and estimators. The default variables are sufficient for the purposes we have envisaged, but the user can easily add new variables, using the existing `R` code as a guide.

Some extensions of the population are easy to implement. Because we focus on income and labour force status, we have not included children in our output. However, the `R` code already includes the probabilities to generate children, and so they can be added easily. It is also fairly straightforward to add a nonresponse propensity variable if one wishes to study the impact of nonresponse on estimation methods. One could model such a nonresponse propensity as a function of income, education, labour force status or some combination of these.

If labour force variables are of interest then adding children to the population introduces a complication: when children turn a certain age (15 in Canada, 16 in the United States), they should be assigned a labour force status (typically *not in the labour force*, of course). This leads us into a discussion of other changes that would make the population more like a real population.

Perhaps the least realistic aspects of our population are the absence of births and deaths, and of immigrants and emigrants. Adding these elements in a realistic way is a challenge we did not face because of our limited goals. If the user wants to introduce these population changes, some of the references mentioned in Appendix 3 may provide useful insights.

Another missing element in our population is households. We have a population of "isolated" individuals, but to make the population resemble a real one, some individuals should live in the same household. Taking this one step further, one would add relationships such as marriage and parent-child. Doing the latter in a realistic way would be quite a challenge because relationships change over time and would involve having households come together and break up.

In summary, adding some of these elements to the synthetic population

would make it more similar to a real population but they are not needed for our specific goals.

Acknowledgements

The inspiration for this work was an early attempt at creating a synthetic population, begun by the author while he was working as a post-retirement consultant for Statistics Canada. The agency decided to use a different approach based on census microdata (which is not publicly available due to confidentiality), and so the author continued the early work independently, in the hope that a useful publicly-available product would be the outcome.

References

Beaumont, J.-F. and Émond, N. (2022). A bootstrap variance estimation method for multistage sampling and two-phase sampling when poisson sampling is used at the second phase. *Stats*, 5:339–357.

Bell, W. and Kramer, M. (1999). Toward variances for X-11 seasonal adjustments. *Survey Methodology*, 25:13–29.

Bessonneau, P., Brilhaut, G., Chauvet, G., and Garcia, C. (2022). With-replacement bootstrap variance estimation for household surveys: Principles, examples and implementation. *Survey Methodology*, 47:313–347.

Fialkowski, A. and Tiwari, H. (2019). SimCorrMix: Simulation of correlated data with multiple variable types including continuous and count mixture distributions. *The R Journal*, 11.

Lohr, S. L. (2022). *Sampling: Design and Analysis*. CRC Press.

Nowok, B., Raab, G., and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, 74:1–26.

Regular, P., Robertson, G., Lewis, K., Babyn, J., Healey, B., and Mowbray, F. (2020). SimSurvey: An R package for comparing the design and analysis of surveys by simulating spatially-correlated populations. *PLoS ONE*, 15.

Schofield, D., Zeppel, M. J. B., Tan, O., Lymer, S., Cunich, M. M., and Shrestha, R. N. (2018). A brief, global history of microsimulation models in health: Past applications, lessons learned and future directions. *International Journal of Microsimulation*, 11:97–142.

Singh, A. and Rao, J. (1995). On the adjustment of gross flow estimates for classification error with application to data from the Canadian Labour Force Survey. *Journal of the American Statistical Association*, 90:478–488.

Templ, M., Meindl, B., Kowarik, A., and Dupriez, O. (2017). Simulation of synthetic complex data: The R package simPop. *Journal of Statistical Software*, 79:1–38.

Urban Institute (2015). The dynamic simulation of income model (`DYNASIM3`): A brief overview.

Zinn, S. (2014). The micsim package of r: An entry-level toolkit for continuous-time microsimulation. *International Journal of Microsimulation*, 7:3–32.

Appendix 1: Notes on the `R` programs

In this Appendix, we describe the `R` programs that make up `poptimesimul` and how they should be run. We also briefly describe the `R` code for the examples that are given in the paper. The source code uses base `R` only. Some of the code can be improved by importing functions from other packages (such as `summarize()` from `dplyr`), but we wanted to avoid requiring users to install additional software. The example programs have extensive comments to help the user understand the steps.

While reading this Appendix, it is useful to refer to Figure 1 in Section 2 above. Users who are only interested in creating a population, without changing any of the parameters, need to run just one file and change a single value depending on whether they want (1) just one month of data or (2) many months of data:

1. *One month of data*: To create an output file (or an environment) containing values for a population at one point in time, simply run the program `makepop.R` with the value `L=1`. By default, it will call

other files automatically, as described below. This will create a file `onemonth.RData` containing the important variables. For more information, see the `poptimesimul` and `makepop` help files in the `man` or `html` directories.

2. *Many months of data*: To create an output file containing many months of data, run `makepop.R` with the value `L=120` (120 is the default number of months but other values can be used). It will call `makemonth()`, as described below. This will create a file `allmonths.RData`. For more information, see the `poptimesimul` and `makepop` help files in the `man` or `html` directories.

The basic parameters for the synthetic population are set in the file `parameters.R`. These include population values such as the number of strata, the number of clusters in each stratum and the average cluster size (number of people). The initial employment and unemployment probabilities (by age-sex group) are set here, as are the relative frequencies for education level (which vary by stratum). The parameters for income and `y_icc`, which are similar, are also set here. For the latter variable, a correlation parameter, which leads to an intracluster correlation, is set. Finally, basic transition probabilities for labour force status are defined.

The file `functions.R` defines a few small utility functions. These include functions that may be of general use (such as one to generate equi-correlated multivariate normal deviates) and functions that may only be useful for this particular project.

We describe the other programs in their logical sequence:

- The main program is `makepop.R`. The number of months L determines what happens next. If $L = 1$ then `onemonth.R` is sourced. If $L > 1$ then `manymonths.R` is sourced. Much of the work to prepare the first month of data is done within `makepop.R`.

- $L = 1$: `onemonth.R` is sourced by `makepop.R`. It simply saves the key population variables in `onemonth.RData`.

- `analyze.R` is an optional program that can be run (or better still, stepped through) to become familiar with the one-month version of the population. It can be run once the output file `onemonth.RData` has been created in the previous step.

- $L > 1$: `manymonths.R` is sourced by `makepop.R`. Its main job is to adjust transition probabilities month by month and call `makemonth()` each time. At the end, it saves the key population variables in `manymonths.RData`.

- `makemonth()` (in `makemonth.R`) is called by `makepop.R` via `manymonths.R`. For each month, once its parameters (transition probabilities) for that month are set in `makepop.R`, the function `makemonth()` is called with those parameters. `makemonth()` produces new population values for the current month. In this way, the initial population is "aged" month by month for L months. The main idea is to adjust the transition probabilities month by month to make the population evolve in a realistic fashion (see Appendix 2). Once the probabilities are adjusted, the "aging" is done by calling `makemonth()`. Once all months have been generated, the user will have the full L months of population data as described in Sections 2 and 3.

- `analyze_time_series.R` is an optional program that can be stepped through to become familiar with the full longitudinal population.

Example programs

There are several example programs that illustrate the use of the synthetic population. The programs were designed to have pedagogical value by implementing methods that are useful to students of sample survey theory. They also suggest how the population can be used by researchers. The package also includes two `.Rmd` files (vignettes), which correspond to the first and third example below.

- `example_stratification_inc.R` is the most elementary example (and not discussed further in this paper—but see its vignette or `example_stratification.pdf`). It simply illustrates the benefits of stratified random sampling. It produces both simulation-based results and exact ones from sampling theory (which are available because we have the whole population). In the stratified case, the sample is allocated using Neyman allocation.

- `example_var_with_replacement.R` uses the initial (month 1) population to illustrate that, in multistage sampling, the assumption that first stage units (clusters in our case) are selected *with* replacement when they are, in fact, selected *without* replacement is conservative, i.e., it

overestimates the variance—slightly if the sampling fraction is low and more noticeably as the sampling fraction increases. There are two versions of the program: the one mentioned here and another one that uses just one stratum. For additional details about this example, see Section 4.

- `example_stratification_deterioration.R` is used to show how a good stratification of the population deteriorates over time as the characteristics of the members of the population evolve over the years. In addition to showing the deterioration of a fixed stratification-allocation combination it also shows what would happen if we could (unrealistically) reallocate the sample to strata each time. See Section 4 or the corresponding vignette (or example_stratification_deterioration.pdf) for additional details.

Appendix 2: Note on transition probabilities for labour force status in *poptimesimul*

Let E, U and I denote the population count of people whose status is employed, unemployed and inactive (not in the labour force), respectively. Note that $E + U + I = N$, the population size. We will use a prime to denote the subsequent month, so E' is the number of employed people in the second month. Dividing $E + U + I = N$ by N, we get $p_E + p_U + p_I = 1$, where $p_E = E/N$, etc. Because we are assuming that N is fixed, we also have $p_{E'} + p_{U'} + p_{I'} = 1$. We can interpret p_E and $p_{E'}$ as the probability of being employed in months 1 and 2. The transition probabilities will be denoted by p_{EE}, p_{EU}, and so on. For example, p_{EU} is the probability that someone goes from being employed in month 1 to being unemployed in month 2. Note that there are six free transition probabilities since, $p_{EI} = 1 - p_{EE} - p_{EU}$ and similarly for p_{UI} and p_{II}.

Remark: For a real population at two points in time, the p_{ij} are not really probabilities but proportions. For example, p_{EU} is simply the proportion of people who started out as employed and became unemployed. Of course, we can reintroduce the probability concept by noting that if we select one person at random from the population, then p_{EU} is indeed the probability that the person we selected went from being employed in month 1 to being

unemployed in month 2.

We have the following equations relating counts in the two months.

$$\begin{aligned} E' &= p_{EE}E + p_{UE}U + p_{IE}I \\ U' &= p_{EU}E + p_{UU}U + p_{IU}I \\ I' &= N - E' - U'. \end{aligned}$$

Substituting $I = N - E - U$ as well as $p_{EI} = 1 - p_{EE} - p_{EU}$, and similarly for p_{UI} and p_{II}, we get

$$\begin{aligned} E' &= (p_{EE} - p_{IE})E + (p_{UE} - p_{IE})U + p_{IE}N \\ U' &= (p_{EU} - p_{IU})E + (p_{UU} - p_{IU}))U + p_{IU}N. \end{aligned}$$

To implement transition probabilities that themselves evolve over time, we will first define some basic transition probabilities and develop adjustment factors that will be used to meet monthly population targets (based, for example, on estimates from a real survey).

Let the basic transition probabilities be denoted by p_{EE0}, p_{EU0}, etc. Consider the target E' for employment in month 2. Then, from the above, $E' = p_{EE}E+p_{UE}U+p_{IE}I$, which we will write as $E' = f_E(p_{EE0}E+p_{UE0}U+p_{IE0}I)$. Hence the factor f_E for employment is

$$f_E = \frac{E'}{p_{EE0}E + p_{UE0}U + p_{IE0}I}.$$

Similarly,

$$f_U = \frac{U'}{p_{EU0}E + p_{UU0}U + p_{IU0}I}.$$

The factors in the `R` code are used in this form. But by dividing both the numerator and denominator by N, both f_E and f_U can be expressed in terms of probabilities (instead of counts):

$$f_E = \frac{p_{E'}}{p_{EE0}p_E + p_{UE0}p_U + p_{IE0}p_I}$$

and

$$f_U = \frac{p_{U'}}{p_{EU0}p_E + p_{UU0}p_U + p_{IU0}p_I}.$$

In the `R` code, the basic transition probabilities (i.e, p_{EE0}, etc.) are defined in the file `parameters.R`. The adjustment factors, which change every month, are defined in `manymonths.R`.

Remark: The default basic transition probabilities in the package are based on values from the Canadian Labour Force Survey. These will vary among countries, of course, and within a country over time (e.g., they depend on the phase of the economic cycle—recession versus booming economy). By using values from a real survey over many months for E, E', etc., the adjustment factors f_E and f_U produce a simulated population whose time series mimic those of real time series.

Appendix 3: Related work

Many of the related projects have a very different goal from ours, namely to generate synthetic units (persons) to protect confidentiality. The methods are meant to produce microdata which, when aggregated or modelled (using regression, for example), produce results close to those that would be produced using real data from a survey. Another key difference is that most projects are designed to produce a snapshot of a population, i.e., the population at one point in time. Our goal is to create a population and make it evolve over many months.

The Official Statistics and Survey Statistics task view on CRAN mentions a few related packages:

`simPop` (Templ et al. (2017)) presents a good overview of some of the methods commonly used. These methods typically start with an existing sample from a real population along with macro-level estimates that the synthetic population should respect. This is achieved by repeatedly sampling from the existing sample in an intelligent way. Different ways of doing this are summarized in the paper. The authors apply their method to produce a synthetic population for Austria using survey microdata and aggregated census data.

The goal of `synthpop` (Nowok et al. (2016)) is "a more modest one of providing test data for users of confidential datasets." The generated data can, for example, be used to test programs that will eventually be run on the (confidential) real data.

Unlike most of the other packages, `SimSurvey` (Regular et al. (2020)) was specifically designed for the study of sampling strategies by testing them

using synthetic data. The paper focuses on fish populations but the methods can be modified for other populations. It does not appear that it can be used as is to create a longitudinal population (although it can be used to create a sequence of populations).

Of the packages available in CRAN, `MicSim` (Zinn (2014)) is closest in spirit to our approach. It too takes a population of individuals and makes them evolve over time. Unlike our package, an initial population is needed as an input. `MicSim` is more flexible than our package, which has hard-coded variables such as labour force status. Because of its ability to easily introduce a variety of variables, we recommend it over our package if the goal is to have a more general tool for simulating populations over time. Note, however, that `MicSim` uses a continuous time approach and so, its output is not a series of snapshots (e.g., monthly or annual) of the population.

There is a rich literature on the broader topic of microsimulation—in fact there is a journal devoted to the topic: the *International Journal of Microsimulation*. As with the `R` packages mentioned above, much of this literature is only tangentially related to our specific goals but is interesting to study as a source of ideas. For a recent survey of recent developments in the microsimulation field, see the recent paper by Schofield et al. (2018).

We note that when a national statistical organization has a microsimulation model, it is often based on data from a sample survey, perhaps augmented with other data, and then aged based on assumptions about transitions. For example, the Urban Institute's DYNASIM3 model starts "with a representative sample of individuals and families, the model 'ages' the data year by year, simulating such demographic events as births, deaths, marriages and divorces, and such economic events as labor force participation, earnings, hours of work, disability onset, and retirement" (see Urban Institute (2015)).