Population forecasting via microsimulation: the software design of the MicMac-project

Gampe, J.; Zinn, S.; Willekens, F.J.C.; van der Gaag, N.L.

published in
Work session on demographic projections, Bucharest, 10-12 October 2007
2007

Joint EUROSTAT/UNECE Work Session on Demographic Projections
Bucarest, 10-12 October 2007

POPULATION FORECASTING VIA MICROSIMULATION: THE SOFTWARE DESIGN OF THE MICMAC-PROJECT[1]

Jutta Gampe[2], Sabine Zinn
Max Planck Institute for Demographic Research
Rostock, Germany

Frans Willekens, Nicole van den Gaag
Netherlands Interdisciplinary Demographic Institute,
The Hague, The Netherlands

1. Introduction

This paper describes design considerations and the general layout of the microsimulation software of the project 'MicMac - Bridging the micro-macro gap in population forecasting', which is funded by the European Commission under the 6th Framework Programme.

In microsimulation life-courses of individuals are projected by randomly drawing their trajectories from a stochastic model, which portrays the propensity for individual transitions between relevant demographic states during life (Willekens, 2005). These simulated life-courses are collected in a virtual population and inference on future population development can be made by analysing this virtual population. If the underlying model realistically describes individual behaviour, then rich and detailed future population characteristics can be derived from the analysis of the aggregated simulated life-courses.

This procedure has several key ingredients: A stochastic model that is able to characterize individual behaviour over the life-course in settings that can be rather complex. Data sources, statistical models, and corresponding estimating procedures that allow to derive the empirical input for the microsimulation, that is, the estimated transition rates. And software that combines the input, allows to incorporate assumptions about future behavioural and institutional changes easily, performs the actual life-course simulations, and provides the simulation results in a format that will allow detailed further analysis.

The microsimulation software that is developed as part of the MicMac-project shall serve all these purposes. It will contain a pre-processor to facilitate the estimation of relevant transition rates from data. Then the so called Mic-core will perform the simulation according to the underlying multistate model. Finally, a so called postprocessor will provide tools for presentation of results.

---

[1] This is joint work with J. Himmelspach and A. Uhrmacher from the Department of Computer Science, University Rostock.

[2] Corresponding author: Max Planck Institute for Demographic Research, Konrad-Zuse-Str. 1, 1087 Rostock, Germany. Fon: +49 381 2081 243, Fax: +49 381 081 543. email : gampe@demogr.mpg.de

The following section will first summarize some general considerations for the software design. Then we will briefly describe the underlying multistate model, followed by a description of the Mic-core. A summary of the current state of development, which is still in progress, and an outlook on features still to be implemented will conclude the paper.

2. General considerations

When designing software, several decisions have to be made that will have far-reaching impact on its potential usefulness, on how easily it will be accessible, and consequently on how widely, or not, it can serve the purposes of its users. As the MicMac-project is to provide methodology that is not linked to a specific problem, but that can be used for applications in a very general multistate framework, this flexibility needs to be anticipated in the software design. The development of the MicMac-software is guided by the following principles:

- No additional expenses should be linked to the use of the Mic-software. That is, we will use free software with no license restrictions. Furthermore, the software should be available for as many current operating systems as possible. No particular commercial product will be needed to run the Mic-software.
- Transparency of the product is mandatory. Therefore we will use open-source software so that the users can follow our steps and are put in the position of easily making adequate changes according to their needs. Additionally, open-source software will allow to extend the current features of Mic to settings we have not (yet) anticipated.
- The software shall be computationally efficient to handle the potentially large number of individual life-courses with state-spaces of considerable size.
- The MicMac-project intends to provide tools for the users, without forcing them to the solutions we suggest. This implies that we will use well-defined interfaces between the Mic-core, which provides the actual simulation results, and the software that creates input rates as well as the software employed to analyse the created output.

As a consequence of these considerations we decided to use R as software tool for the pre-processor as well as the post-processor, which is available as Free Software under the terms of the Free Software Foundation's GNU General Public License (R Development Core Team, 2007). R provides an environment for statistical analyses and graphics, and it runs on UNIX platforms, Windows and MacOS. The combination of up-to-date statistical methodology and well-designed publication-quality plots make it an ideal candidate for the pre- and post- processor. Nevertheless, we are aware that R may not be familiar to many prospective users. Therefore, even though we encourage its use, we design our software so that users can continue with their preferred tools and will only have to provide the input rates in a specified format. The same applies to the simulated life-courses, which are simply stored in a database for further evaluation.

The Mic-core, which will be described in section 4, is the only part that will be closed to the user and it will be programmed in JAVA. The Mic-core will request no other interaction by the user than the provision of the transition rates in a well-defined format and a simulation frame, basically providing the starting population and the time horizon for the microsimulation. This is done so that the simulation is performed in a computationally efficient way with a little overhead as possible. If necessary, JAMES also allows distributed computation.

3. Model of the individual life-course

The MicMac microsimulation is based on a continuous-time multistate model (Willekens, 2006). Each individual occupies one of I potential states, which are collected in the state-space $S = \{1,...,I\}$. This state-space is determined by the problem to be studied and will contain the relevant demographic states that need to be considered. In Mic we usually include death as one of the potential transitions, hence include 'dead' as one (absorbing) state in the model. Furthermore, there are two key time scales, which both are considered as continuous, namely the age x of an individual and calendar time t. (A third time-scale, the time since entry into the current state, is not yet considered in MicMac's multistate-model. An extension to this setting poses no particular problem to the simulation approach though.)

An individual enters the population in state $i_0 \in S$ at age x_0 and calendar time t_0. We allow individuals to enter either by birth, in which case $x_0 = 0$, or by immigration at age $x_0 \geq 0$. When and to which state the next transition will happen is determined by the age- and time-dependent transition rates $\lambda_{ij}(x,t)$. These rates define the instantaneous risks of an individual, who is currently in state *i*, aged *x*, at calendar time *t* to make a transition to state *j*. All states $j \neq i$ for which $\lambda_{ij}(x,t) > 0$ compete to be the state visited next. For the actual performance of the simulation these transition rates $\lambda_{ij}(x,t)$ will have to be provided by the user.

While the age-dependence will usually be estimated from appropriate surveys or population data, the dependence on calendar time will be determined by assumptions on how future development in these age-specific rates is expected to be. These can be derived from formal forecasts or expert judgement, and the Mic-core will not be restricted to any specific kind of projection model. In the simplest case current rates will be assumed to prevail in the future, hence removing the dependence on t, i.e. assuming that $\lambda_{ij}(x,t) = \lambda_{ij}(x)$.

To each rate $\lambda_{ij}(x,t)$ corresponds a random sojourn time S_{ij}, whose distribution is uniquely defined by the transition rate. Therefore random numbers from these sojourn-time distributions can be simulated. Which of the (usually several) competing states will be visited next is determined in the following way: Random sojourn times are simulated for all competing states, for which $\lambda_{ij}(x,t) > 0$. The 'winning' state is the state j^*, for which the actual random time $s_{ij^*} < s_{ij}$, for $j \neq j^*$. Once the next state is determined, we know that the individual entered this state j^* at age $x_0 + s_{ij^*}$ at calendar time $t_0 + s_{ij^*}$. The corresponding rates for a transition out of j^* can again be determined from the provided $\lambda_{ij}(x,t)$. The procedure is repeated until either the individual enters the absorbing state 'dead' or until the pre-defined time-frame of the simulation is reached.

The simulation of the random sojourn times is achieved by the relation that links the distribution function of a positive random variable to its hazard rate. This relation is

$$F(s) = P(S \leq s) = 1 - \exp\{-\Lambda(s)\},$$

where $\Lambda(s) = \int_0^s \lambda(u)\,du$ is the so called integrated or cumulative hazard (Kalbfleisch & Prentice, 2002). In this way we can simulate random numbers from the distribution of *S* by drawing random numbers from a standard uniform distribution $U[0,1]$ and inverting the distribution function $F(s)$, see Robert & Casella (1999). In short this leads to

$$s = \Lambda^{-1}(-\ln u),$$

where *u* is a random number from the standard uniform distribution. The inversion of the integrated hazard Λ^{-1} is possible if the corresponding rate λ is positive, i.e. the respective transition is possible. The actual inversion need not have a closed-form solution, but rather may have to be obtained numerically. It is, however, particularly simple if the integrated hazard is piecewise linear. In the Mic-software we use piecewise linear approximations to integrated hazard rates.

4. Simulating the virtual population: the Mic-core

The actual microsimulation of the MicMac-project is implemented in JAVA, based on *JAMES, a Java-based Agent Modelling Environment for Simulation*, developed by the Research Group of Modelling and Simulation at the Computer Science Department, University of Rostock (Himmelspach & Uhrmacher, 2007; Himmelspach et al., 2007). This environment allows flexible definitions of simulation environments, and the multistate model described in the previous section has been incorporated in the framework of *JAMES*.

The virtual population is combined from the individual life-courses that are generated during the run of the microsimulation. To perform such a run the following parameters have to be set:

- A starting time t_s, an ending time t_e,[3] and
- a starting population of size N_s, where for each individual the age and the occupied state at time t_s must be given.

After the frame for the simulation has been defined, the simulation is started. This is done by first determining the next transition times and the next states visited for all individuals in the initial population. All individuals, whose next state is not 'dead', are then lined up in the so called event-queue according to their times for next transition. Figure 1 shows an example of such an event-queue.

The actual run of the experiment is performed by repeating the following steps, which are solely depending on the event queue:

- The first element of the event-queue, the so called 'head', is dequeued.
- The next event-time and the next state is determined for the head of the queue.
- If the new state for the head is 'dead', then the individual is not enqueued again.
- *Otherwise the individual is enqueued at the correct position in the event-queue.*
- *The previous new state becomes the current state, and the newly determined next state is stored.*
- The new head of the event-queue is picked and the procedure is continued until either the end-time t_e is reached or the event-queue is empty.

Figure 1 Example of an event-queue

Individual: 56	Individual: 257	Individual: 669	….
s.current: 5	s.current: 1	s.current: 1	….
trans.time: 12.6.2007	trans.time: 1.9.2007	trans.time: 4.12.2007	….
s.next: 6	s.next: 8	s.next: 2	….

To give an example, consider the situation depicted in Figure 1. In the first step, individual No. 56 would be picked and its next transition time would be determined. If the next transition would be assigned for, let's say, October 1, 2007, and if the scheduled transition would not be death, then individual No. 56 would be enqueued between individual 257 and 669. Individual No. 257 would be the new head of the event-queue and would be dealt with next. If, however, individual No. 56 would be schedule to have the next transition on, e.g., August 25, 2007 (and again the target state would not be absorbing), then it would be enqueued at position number one again, and would be dealt with in the next step. If the transition would have been death of the individual, it would disappear from the event-queue. All transitions of all individuals are stored in a database for further analysis.

5. Conclusion and Outlook

At present the Mic-core allows the simulation of life-courses of independent individuals according to rates that are age-dependent, but not time-dependent yet. The inclusion of transition rates that vary over calendar time is a simple extension of the current design and poses no technical challenge whatsoever. New individuals are added to the population when childbirth occurs to female individuals in the virtual population.

[3] The ending time of a simulation run alternatively can be defined implicitly by the last individual of the starting population entering the absorbing state 'dead'.

Another feature that is not yet implemented is the creation of 'linked lives', that is the formation of partnerships where the attributes, i.e. the state, of the partner changes the transition rates of an individual. Currently individuals are considered independently, and no marriage market is implemented. However, the design of a mate-matching algorithm, which assorts partners according to some key characteristics is already in progress.

In this context one must bear in mind though, that many of these extensions will also ask for the corresponding data so that these diverse rates and attributes can be estimated. Besides the technical aspects of the software implementation these data requirements, which can be quite extensive, pose one of the major challenges to realistic microsimulation approaches.

6. References

Jan Himmelspach, Roland Ewald, Stefan Leye, and Adelinde M. Uhrmacher. Parallel and distributed simulation of parallel DEVS models. In *Proceedings of the SpringSim '07, EVS Integrative M&S Symposium*, pages 249-256. SCS, 2007.

Jan Himmelspach and Adelinde M. Uhrmacher. Plug'n simulate. In *Proceedings of the 40th Annual Simulation Symposium*, pages 137-143. IEEE Computer Society, 2007.

John D. Kalbfleisch and Ross L. Prentice (2002). *The Statistical Analysis of Failure Time Data.* Second Edition. John Wiley & Sons.

R Development Core Team (2007). *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing.
ISBN 3-900051-07-0. (http://www-.r-project.org)

Christian P. Robert and George Casella (1999). *Monte Carlo Statistical Methods.*
Springer-Verlag.

Frans Willekens (2005). Biographic forecasting: Bridging the micro-macro gap in population forecasting. *New Zealand Population Review*, 3:77-124.

Frans Willekens (2006). Multistate model for biographic analysis and projection. *Technical report, NIDI.* Available at: www.nidi.knaw.nl/en/micmac/publications/d01/micmac-d01r.pdf