Conclusion

In the course of this book, we have developed a dynamic population microsimulation model in successive stages. We have progressively increased the number of state variables and events associated with actors in a simulation. The added characteristics enabled us to differentiate the risks of undergoing one or another of the events in the model. The final model enabled us to make projections of a national population by age, sex and region of residence, as well as to calculate the components of a classic multi-regional life table.

The first chapter paid particular attention to explaining the structure of a Modgen program and to exploring the interface used to generate scenarios. At this stage, the model generated by the Modgen Wizard was taken as is, and a single event, death, was modelled using a constant risk. In the following chapter we added a *classification* to attribute a sex to the actors, and also a *range* to measure their ages in completed years. These two new characteristics of the *Person* actor, contained in the two state variables (*sex* and *age_int*) allowed us to specify more accurately the risk of death. We were able to construct an abridged life table and appreciate how easy it is to add new dimensions to the model.

Adding an internal migration event in Chap. 3 demonstrated the other advantages of Modgen for programming microsimulation models. We saw the ease with which a new event module could be added, a useful feature given the advantages of modular programming. We also introduced and explained a new type of parameter (*cumrate*) and a function (*Lookup*) specific to Modgen, which together enable us to change the value of a state variable from a probability distribution matrix. In the mobility example, this matrix was solely a function of the province of origin, but *cumrate* and *Lookup* may be used with an arbitrary number of dimensions.

Another of Modgen's major advantage is its ability to automatically manage competing risks. Unlike discrete-time models, Modgen's continuous-time modelling allows us to add a new event without having to re-estimate the parameters of the

A. Bélanger, P. Sabourin, *Microsimulation and Population Dynamics*,
The Springer Series on Demographic Methods and Population Analysis 43,
DOI 10.1007/978-3-319-44663-9

other model events, and without the problem of having to explicitely decide in what order the events are to take place.

In the following chapters we saw how a cohort model could be transformed to become a full projection model allowing for the addition of new actors to the population through fertility and immigration. The fertility module enabled new actors to be created and linked to their mother allowing for the transfer of information between actors using pointers and the Modgen *link* instruction.

Gradually complexifying a simple model is a useful didactical approach for a tutorial course and it has enabled us to show that it is relatively simple to complexify or modify an existing model in Modgen; however this is not the best way to proceed in reality. Microsimulation models should be thought through as much as possible before you start programming. Before generating a new model from the Modgen Wizard, the designer of a microsimulation should have a clear conceptual outlook of the final model and should define each of the state variables and their sub-categories using *classification, range,* or other types of variables. He or she should also have a clear idea of the equations which will define the waiting time of the different events in the model, as well as the consequences of each of these events for the whole set of characteristics of the simulated actor.

The full model presented in chapter six could easily be developed further by adding relevant dimensions. Let's take the level of education, for instance, which is a characteristic of analytical interest in itself, as well as an important determinant of the other events we want to model, such as fertility, migration or death. To add the level of education in the model, we would create a module determining the age at which each actor reaches each educational milestone. This would involve a new *classification* describing the educational level of the actor, as well as the corresponding state variables. The modeling of education itself could be done in several ways. We could first determine the highest educational attainment of an actor, and then determine the timing of the event, as well as the timing of intermediate educational achievements. We could also calculate transition rates between the different educational levels (this would be the preferred solution for multistate models). In any case, the result of this modeling would be implemented in the event function and its associated time function. Education could then be added as an additional dimension in other demographic events, such as fertility or mortality.

The model developed in this book is a dynamic case-based microsimulation model in which each actor is simulated separately up to the time of death or the end of the projection. As we saw in the chapter on the fertility module, it is possible to create links between actors of the same case, such as between children and their mother. However, this type of model does not allow for links or interactions between different cases.

Modgen can also be used to create microsimulation models with interactions between the simulated actors. To do this, the model builder has to choose a time-based rather than a case-based model when creating the microsimulation program. The code described in this book can be used without major changes to define a similar corresponding time-based model. In such a model, all the actors are simulated simultaneously, thus enabling interactions between simulated actors. Time-based models are useful if one wants, for example, to simulate the evolution of a contagious disease in a population, or to create families by explicitly modelling the formation of unions between two actors in the simulation. To do this, Modgen provides tools to create actor sets, so that simulated actors can be grouped and accessed dynamically according to some of their characteristics. To model union formation, for example, we could create actor sets by sex, marital status and other characteristics associated with marriage practices and partner selection – age group, level of education, place of residence, etc. In this way we could create a union-formation event which would match unmarried candidates from the different mate pools according to pre-determined probabilities based on mate preferences. Union types having a generally low probability of formation, for instance, could still have good chances of happening in a marriage market where suitable candidates are rare.

So the Modgen language has a lot to offer and allows for many different types of microsimulation models to be created. In this book we have developed a relatively simple dynamic microsimulation model, but one which is sufficiently general to serve as the basis for a set of models with different objectives. This concluding section has shown the different development paths the user could follow, using the model created here, to meet some specific research needs.

Appendix: Coding Standards

The coding convention we propose is a set of simple generic guidelines for making a Modgen microsimulation program easier to write and to read. A model builder should follow these guidelines as closely as possible, but they are not hard and fast rules; when you build a model you will naturally adapt them to your particular set of problems while respecting their essential spirit.

- A microsimulation model should be divided as far as possible into thematic or functional modules. Results tables are grouped together in a separate module.
- The order of sections should be the same for each module. In order of appearance, starting from the top, this should be: classifications and ranges; partitions; parameters; the *Person* class and its functions, if possible in their order of appearance in the class.
- As many comments as possible should be added next to the code. Clearly written and detailed comments are of immense value, not just for other users and developers but as reminders for oneself.
- No numbers should ever be inserted into the code; instead, state variables or parameters should be created.
- Give meaningful names to states, state variables, parameters, classifications, ranges and other types of variable. Names of classifications and ranges are written in CAPITALS. Classification modalities are also in upper case letters. They start with the first letter of the classification name, followed by an underline and the name of the modality.
- The first letter of each word in the name of a parameter is capitalised, as in ParameterNameExample.
- State variable names are written in lower case letters. This makes it easy to distinguish, for example, the province state variable from the PROVINCE classification.
- Names of links begin with the letter l (for link). So a link to the mother has the form *lMother.*

A. Bélanger, P. Sabourin, *Microsimulation and Population Dynamics*,
The Springer Series on Demographic Methods and Population Analysis 43,
DOI 10.1007/978-3-319-44663-9

Glossary

Actor Generally corresponds to an instance of the *Person* object. Analog to an individual in a real population.

Case A case is a unit of simulation in Modgen. A case starts by launching the simulation of an actor. Events taking place during the simulation of this case may in turn create other actors within the same case (for example when there is a birth).

Derived State A derived state is information generated by Modgen on the state of an actor at a given moment. For example, a derived state can detect a change in state or compile the value of a state variable at a particular moment (when there is a change in the state of another variable, for example).

Event An actor's state variables are modified in events. Each event function is paired with its own time function, which determines the exact time at which the event takes place.

Main file This is the file which drives a Modgen simulation and usually bears the model name. It contains the *Simulation* function (containing the loop that generates the cases) and the *CaseSimulation* function.

User interface This is the graphic interface of a Modgen compiled program.

Precompilation/Compilation The action of transforming the model code into an independently executable program (which does not need additional software to be executed). In Modgen, compilation takes place in two phases. In the first, Modgen code is transformed into C++ code by the Modgen compiler: this is precompilation. In the second phase, the Visual Studio compiler transforms the C++ code into an executable program. In Modgen 12, the two phases occur simultaneously when the project is compiled from the Visual Studio build menu.

A. Bélanger, P. Sabourin, *Microsimulation and Population Dynamics*,
The Springer Series on Demographic Methods and Population Analysis 43,
DOI 10.1007/978-3-319-44663-9

Waiting time Waiting time is the duration before the occurrence of an event. Modgen automatically orders and manages waiting time for all the events. It recalculates this time whenever there is a change in a state variable on which waiting times depend. For example if mortality risk varies with age, Modgen recalculates the time before death on each birthday.

References

Bélanger, A., & Caron Malenfant, E. (2005). *Population projections of visible minority groups, Canada, provinces and regions, 2001–2017*. s.l. Ottawa: Statistics Canada, Demography Division.

Bélanger, A., & Larrivée, D. (1992). New approach for constructing Canadian working life tables, 1986–1987. *Statistical Journal of the United Nations Economic Commission for Europe, 9*(1, IOS Press), 27–49.

Bélanger, A., Martel, L., Berthelot, J.-M., & Wilkins, R. (2002). Gender differences in disability-free life expectancy for selected risk factors and chronic conditions in Canada. *Journal of Women & Aging, 14*(1–2), 61–83.

Orcutt, G. H. (1957). A new type of socio-economic system. *The Review of Economics and Statistics, 39*(2, JSTOR), 116–123.

Rogers, A. (1975). *Introduction to multiregional mathematical demography*. New York: Wiley.

Rogers, A., Rogers, R. G., & Bélanger, A. (1990). Longer life but worse health? Measurement and dynamics. *The Gerontologist, 30*(5, Oxford University Press), 640–649.

Rowland, D. T. (2003). *Demographic methods and concepts*. Oxford: Oxford University Press, 546 pages.

Spielauer, M. (2009). *Modgen and the application RiskPaths from the model developer's view*. Ottawa: Statistics Canada.

Van Imhoff, E., & Post, W. (1998). Microsimulation methods for population projection. *Population: An English Selection, 10*(JSTOR), 97–138.

Willekens, F. J. (1980). Multistate analysis: Tables of working life. *Environment and Planning A, 12*(5, SAGE), 563–588.

Willekens, F. J., Shah, I., Shah, J. M., & Ramachandran, P. (1982). Multi-state analysis of marital status life tables: Theory and application. *Population Studies, 36*(1, Taylor & Francis), 129–144.

Wolf, D. A. (2001). The role of microsimulation in longitudinal data analysis. *Canadian Studies in Population, 28*(2), 313–339.

Zeng, Yi., Morgan, S. P., Wang, Z., Gu, D., & Yang, C. (2012). A multistate life table analysis of union regimes in the United States: Trends and racial differentials, 1970–2002. *Population research and policy review, 31*(2, Springer), 207–234.

A. Bélanger, P. Sabourin, *Microsimulation and Population Dynamics*,
The Springer Series on Demographic Methods and Population Analysis 43,
DOI 10.1007/978-3-319-44663-9

Index

A. Bélanger, P. Sabourin, *Microsimulation and Population Dynamics*,
The Springer Series on Demographic Methods and Population Analysis 43,
DOI 10.1007/978-3-319-44663-9