Chapter 2

Getting Started

Abstract This chapter sets the stage before building the microsimulation model. First, we describe proprieties of the microsimulation model that will be built. The model is time-based, discrete-time and stochastic. We then describe properties of a multistate model that will be converted into a microsimulation model and we show how building a synthetic base population that consists of the individuals that will be projected. We finally explain how to set up the workspace in SAS.

Keywords Microsimulation · Population projection · Demography · Method · SAS

2.1 Properties of the Microsimulation Model

Various microsimulation methods exist. The one presented in this guide is as follows:

- *Time-based.* This means that we simulate the life of all individuals from time t to $t + a$, then repeat from $t + a$ to $t + 2 * a$ (in our example, from 2010 to 2015, then from 2015 to 2020, and so on until 2060). In contrast, a person-based model would simulate the life of the first individual until his death (or until the end of the projection), then simulate the life of the second one, and so on for all the individuals of the base population. The advantage of a time-based model is the possibility of using aggregated outcomes as predictors of individual events. For instance, both the size and the sociocultural composition of a municipality impact migration dynamics (Marois and Bélanger 2015). A time-based model could implement this effect.
- *Discrete-time.* This means we consider only the population at specific points in time (by 5-year steps, for instance), without considering what could happen between those points. If the projection uses 5-year steps, all probabilities of an

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-79111-7_2) contains supplementary material, which is available to authorized users.

event need to be applied for a 5-year period. This also requires ordering the occurrence of events. For instance, we might decide that changes in education happen before fertility events. Thus, we would use the educational attainment at time $t + a$ as a predictor of the births between t and $t + a$.

- *Stochastic*. All events occur stochastically using random experiments, which means we compare the probability of occurrence with a random linear number from 0 to 1 to determine whether or not the event occurs. It is also possible to perform microsimulation using a deterministic approach. This consists of multiplying the weight of individuals by the probability. We will show a quick example of this other approach later in this guide (Chap. 6).

The model described in this guide is represented schematically in Fig. 2.1 and works as follows: we start from a microdata set, which is a sample of individuals representing the starting population at time t , for instance, the population of India and its characteristics in 2010. To get the population of 2015, the dataset is then submitted to different modules that modify the characteristics of the population according to different rules and predetermined assumptions. Each module corresponds to one event (dying, giving birth, change in education, moving to a different region, etc.). Once all modules have been applied, the dataset will correspond to the population of 2015. From the resulting dataset for 2015, we then repeat the process to get the population in 2020, and so on, until the end of the projection.

The order in which modules occur matters, as they change the exposure and some events can be conditional on others. This relies in large part on the way assumptions are calculated. For instance, if migration assumptions are based on the previous place of residence as recorded in surveys or a census, then by definition, only the mobility of surviving individuals should be assessed. Therefore, in the microsimulation, the migration events should occur after the mortality event.

2.2 The Multistate Model for India

The demographic assumptions of the microsimulation model presented in Chaps. 3, 4 and 5 of this book are taken from the multistate projection by education produced by KC et al. (2018) for India and its regions. We thus replicate the demographic events (birth, mortality, migration, and change in education) from a multistate model into a microsimulation framework. More specifically, we use assumptions from the baseline scenario, taking into account differentials by education, state, and type of residence (urban/rural) for fertility and mortality. The multistate projection used as our source has the following properties:

- The projection time-span is from 2010 to 2100 by 5-year steps, though in our example, we project only until 2060.
- It includes 35 states of India, all classified into rural and urban areas (for a total of 70 regions).

Fig. 2.1 Framework of the microsimulation model

- It includes educational attainment in 6 categories. From 0 to 14, the education variable is not applied. At the age of 15–19, the cohort is broken down according to the education level reached at this age. Transition rates are then applied and the final educational attainment is reached at the age of 30–34.
- Educational attainment matters for fertility and mortality, but not for migration.
- Before the age of 15, the education of the mother is used for mortality. However, in the base population, the education of the mother is not known, so no differentials by education are used for the populations aged 0–14 in 2010, 5–14 in 2015 and 10–14 in 2020.
- Internal mobility is modelled with an age- and sex-specific origin–destination matrix.
- In addition to internal mobility, the model allows for the reclassification of rural areas as urban areas.
- The model is closed, which means there is no international migration.

Assumptions have been formulated according to past trends, expert judgments, and statistical modeling. The detailed methodology for the assumptions is available in the supplementary information for KC et al. (2018). In this multistate projection, events are ordered as follows:

1. The mortality is applied with survival ratios by age, sex, education and region;
2. Education transition rates by age, sex, region and education are then used for educational shifts;
3. For those who survive, the domestic migration is then applied using age- and sex-specific rates from an origin–destination matrix;
4. Births are generated with fertility rates by age, education, and region applied to the exposed population;
5. Finally, region-specific reclassification rates from rural to urban areas are applied.

2.3 The Base Population

Any microsimulation model for demographic projection requires a comprehensive microdata set representing the base population. When available, the best option is to use a public microdata file of the most recent census, since it has a large sample, good coverage and good accuracy for the most relevant variables for a multidimensional population projection (age, sex, region of residence, education, etc.). Many of these files are available for free upon registration on IPUMS-International.¹ The microsimulation models for Canada and the USA developed by the *Laboratoire de Simulation Démographique* are both based on public files of recent censuses (Bélanger et al. 2019). However, variables in censuses rarely go beyond age, sex, education, and place of residence. If other variables are required in the microsimulation model, various

¹ <https://international.ipums.org/international/>.

imputation methods using other sources of data may be used (see for instance the MICE package in R (van Buuren and Groothuis-Oudshoorn 2011)).

When no public files of a census are available, or when they are outdated, a second option is to use a public file from a survey with a large sample (such as the *Labour Force Survey* in the European Union or *Demographic and Health Surveys* for African countries). A calibration of the microdata on aggregated data of census or population estimates may be required for optimal accuracy since surveys are not necessarily designed to be representative of the population for all variables used. This is what has been done for CEPAM-Mic, a microsimulation model projection of the population of European countries developed by the International Institute for Applied Systems Analysis (Sabourin et al. 2017). Again, variables from secondary surveys may be imputed.

Finally, when no microdata sets are available, a last option is to build one's own synthetic base population from population estimates or aggregated tables from censuses. The synthetic base population may have to follow certain rules, depending on the purpose of the projection.

For the microsimulation model showed in the example in this book, no census data files are available. The IPUMS-International provides a public file for the National Sample Survey (NSS) on Employment and Unemployment 2009, which is close to the starting year of the multistate projection (as a reminder, we want to replicate the multistate projection from KC et al. (2018) that starts in 2010), but its sample size is low for some regions, and the survey doesn't allow us to split the population by rural and urban areas. We will thus build our base population from scratch, using the aggregated population by age, sex, region, and education in 2010 from KC et al. (2018). This aggregated population can be found in the file `AggregatedPop2010.csv`. This input file, the complete code to generate the base population (In this Chapter —`BasePop.sas`) and the resulting base population (`POP_2010.csv`) can be found in this Chapter.

As shown in Fig. 2.2, the dataset of the input file is structured such that each line represents a possible combination of subgroups and all population counts are in one column.

Starting from this file, we build the base population in different steps. First, we import the file in SAS with `proc import`, which is used to import a CSV file and transform it into a SAS dataset. The `datafile` statement specifies the location of the CSV file (you might change this part according to your setting). The `out` statement specifies where we want to store the SAS dataset resulting from the importation. In our example, we store it in a file called `aggregatedpop2010` located in the “work” library (the “work” library stores temporary SAS files, which means the file will be deleted after closing the session). The `dbms` option specifies the type of file that is imported, in our case, a CSV. The `replace` option indicates that an existing file will be overwritten. By giving the value “yes” to the `getnames` option, we finally indicate that the first row of the file is the labels of variables rather than data.

	A	B	C	D	E
1	agegr	edu	sex	region	Pop
2		0 e1		0 AN_urban	5178
3		0 e1		0 AD_urban	1084307
4		0 e1		0 AR_urban	13887
5		0 e1		0 AS_urban	160455
6		0 e1		0 BR_urban	609958
7		0 e1		0 CH_urban	41751
8		0 e1		0 CT_urban	264598
9		0 e1		0 DN_urban	8500
10		0 e1		0 DD_urban	7341
11		0 e1		0 DL_urban	713579
12		0 e1		0 GA_urban	32628
13		0 e1		0 GJ_urban	1108371
14		0 e1		0 HR_urban	413680
15		0 e1		0 HP_urban	23296
16		0 e1		0 JK_urban	162862

Fig. 2.2 Screenshot of the file aggregatedPop2010.csv (opened with Excel)

```

/*Importing the aggregated population by age, sex, education, and region*/
proc import
datafile="C:\Users\Guillaume\Desktop\Microsimulation\Chapter2\aggregatedpop2010.csv"
  out=work.aggregatedpop2010
  dbms=csv
  replace;
  getnames=yes;
run;

```

In microsimulation, we simulate individuals, so we need to disaggregate the dataset in a way that each row represents one observation. However, since the population of India is very large (more than 1 billion inhabitants), the resulting dataset of such a file would be way too large, and normal computing power is unlikely to be enough to run the simulation. We will thus apply sampling rules and weights.

The choice of the number of cases (individuals) to simulate depends on the purposes of the projection and the computer power. The larger the number of cases, the lower the Monte Carlo error. However, the computing time needed to run the simulation increases accordingly. At the national and subnational levels, a sample size higher than 500,000 cases is generally large enough to have a marginal Monte Carlo error and accurate simulation outcomes with only one single run (Bélanger et al. 2019; Caron-Malenfant et al. 2017; Marois et al. 2020; Van Hook et al. 2020). When large samples are not possible for the base population (for instance, because

of limited computing power or because the base population is built from a survey), multiple runs of the microsimulation may be required (Van Imhoff and Post 1998). In this book, since the base population we build is synthetic, we can decide a priori the number of cases to simulate. Therefore, we choose a number that will be large enough to produce accurate national and subnational outcomes with a single microsimulation run. If we want to analyse a more disaggregated group further (for instance, women with a high level of education in a specific region), we can either increase the sample size for the group of interest or perform multiple runs of the simulation.

The first thing to do is to remove from the dataset the groups with 0 population (such as the children with a high level of education), which reduces the size of the dataset. We create a new temporary dataset called `pop` from the imported dataset `aggregatedpop2010`. When the variable `pop` (for population size) is 0, we delete the row.

```
/*Removing categories with 0 population*/
data work.pop;
set work.aggregatedpop2010;
if pop=0 then delete;
run;
```

For large enough population counts, for example, larger than 10,000, individuals can be used to represent a percentage of the population. We decided in our example that the number of simulated individuals would represent 0.05% of the population size. Using a `do` loop, the principle of the code to do so is to replicate rows that meet this condition ($10,000 \leq \text{pop}$, that is, when the population is higher than 10,000). The number of replications corresponds to the multiplication of the population size by 0.0005 (0.05%) minus 1 (the initial row of the replication is already there).

```
/*Generating individuals for the base population*/
data work.pop;
set work.pop;

if 10000<=pop then do i=1 to (pop*0.0005-1);
weight=pop/(pop*0.0005)+(pop-floor(pop*0.0005)*pop/(pop*0.0005))/floor(pop*0.0005);
output;
end;
output;

run;
```

In the replication process, a weight variable is created at the same time. Since we set the sample size to 0.05%, the weight of each replication is about 2000 ($\text{pop}/(\text{pop} \times 0.0005)$). However, because the population size is in general not a perfect fraction of 0.0005, we need to adjust the weight. The “do loop” only uses integers, so when the population size is for instance 13,800, there are 5 loops and the sum of weights of those 6 observations (the 5 replications plus the initial observation) is 12,000. The difference between 13,800 and 12,000 divided by the number of observations (that we get using the floor, which rounds down a value) is added to the weight of each observation.

For groups with smaller populations, using the same sample rule might generate too few observations, which would lead to less accurate forecasting results by increasing the Monte Carlo error. At the same time, the dataset would be too loaded

if we included too many observations for groups that are very marginal (such as women 90–95 with a postsecondary education living in a rural area of Dadra & Nagar Haveli). We, therefore, decided to generate a specific number of observations that decreases with the population size of the group: 40 observations for populations between 1000 and 10,000, 30 observations for populations between 100 and 1000, 10 for populations between 30 and 100 and 2 for populations lower than 30.

To alleviate the codes, we create a macro function that will repeat the codes to generate individuals under different parameters for the thresholds in population size and the number of observations to generate. The macro function starts with `%macro` followed by the name of the macro and the parameters in bracket. The code to repeat is then stated, while parameters are preceded by the symbol `&`. For our purpose, the name of the macro is “sample”. The parameters are `minpop` (for the minimum population threshold), `maxpop` (for the maximum population threshold) and `size` (for the number of individuals to generate). The code embedded in the function is then the one shown above in the example for populations larger than 10,000, adding however an upper limit to the condition (`&maxpop`) and replacing 10,000 with `&minpop` and “`pop*0.0005`” with `&size`.

```
/*Generating individuals for the base population*/
%macro sample(minpop, maxpop, size);
data work.pop;
set work.pop;
if &minpop<=pop<&maxpop then do i=1 to (&size-1);
weight=pop/(&size)+(pop-floor(&size)*pop/(&size))/floor(&size);
output;
end;
output;
run;
%mend;
```

We can then call the macro with different parameters to generate the desired number of observations in the dataset.

```
%sample(0,30,2);
%sample(30,100,10);
%sample(100,1000,30);
%sample(1000,10000,40);
```

If not generated already, we can also use the macro for populations larger than 10,000.

```
%sample(10000,100000000,(pop*0.0005));
```

The resulting dataset now has 846,024 observations with an average population weight of 1431.2 (maximum 2398.2) and a standard variation of 890.5. Now that the dataset is disaggregated, we can prepare the dataset for the microsimulation in the new temporary dataset `pop2010`. We add a variable `year` with the value 2010, which is the starting year of the projection we replicate. This variable will be updated further in the microsimulation. From the age group variable, we create the cohort of birth. We add a variable for the education of the mother. This variable will only be used as a determinant of the mortality of children. For those already born in 2010, there are no differentials, so the value doesn’t matter in the base population. By default, all the individuals in the base population are alive, so we set the variable `death = 0`. This will

change further during the simulation. Finally, we drop variables `pop` and `i` that were used to generate the observations, as they will not be used in the microsimulation.

```
/*Preparing the dataset for the microsimulation*/
data work.pop2010;
set work.pop;

/*Starting year*/
year=2010;

/*Cohort*/
cohort=year-5-agegr;

/*Education of the mother, set the same as own education temporarily*/
eduM=edu;

/*All individuals are alive*/
death=0;

drop pop i;
run;
```

The last step in the creation of the base population is to export the dataset in a CSV file. We use the export procedure. In the outfile statement, we specify the location and name of the exported file (POP_2010.csv). With `dbms`, we specify the type of file (CSV). Finally, the `replace` option overwrites any existing file with the same name.

```
/*Exporting the base population in a csv file*/
proc export data=work.pop2010
outfile=C:\Users\Guillaume\Desktop\Microsimulation\Chapter\POP_2010.csv'
dbms=csv replace;
run;
```

The resulting CSV file has 9 columns (Fig. 2.3). The complete base population can be downloaded online.

The values of variables are as follows:

agegr—Age group of.

- 0. 0-4;
- 5. 5-9;
- 10. 10-14;
- ...

	A	B	C	D	E	F	G	H	I
1	agegr	edu	sex	region	weight	year	cohort	eduM	death
2	0	e1	0	AN_urban	129.4587	2010	2005	e1	0
3	0	e1	0	AN_urban	129.4587	2010	2005	e1	0
4	0	e1	0	AN_urban	129.4587	2010	2005	e1	0
5	0	e1	0	AN_urban	129.4587	2010	2005	e1	0
6	0	e1	0	AN_urban	129.4587	2010	2005	e1	0
7	0	e1	0	AN_urban	129.4587	2010	2005	e1	0
8	0	e1	0	AN_urban	129.4587	2010	2005	e1	0

Fig. 2.3 Screenshot of the file POP_2010.csv (opened with Excel)

100. 100+;

edu—Educational attainment/***eduM***—Education of the mother.

- e1. No education;
- e2. Incomplete primary;
- e3. Complete primary;
- e4. Lower secondary;
- e5. Upper secondary;
- e6. Postsecondary;

sex—Sex.

- 0. Male;
- 1. Female;

region—Region of residence.

(followed by *_rural* for rural parts and *_urban* for urban parts)

- AD. Andhra Pradesh;
- AN. Andaman & Nicobar Islands;
- AR. Arunachal Pradesh;
- AS. Assam;
- BR. Bihar;
- CH. Chandigarh;
- CT. Chhattisgarh;
- DD. Daman & Diu;
- DL. Nct Of Delhi;
- DN. Dadra & Nagar Haveli;
- GA. Goa;
- GJ. Gujarat;
- HP. Himachal Pradesh;
- HR. Haryana;
- JH. Jharkhand;
- JK. Jammu & Kashmir;
- KA. Karnataka;
- KL. Kerala;
- LD. Lakshadweep;
- MH. Maharashtra;
- ML. Meghalaya;
- MN. Manipur;
- MP. Madhya Pradesh;
- MZ. Mizoram;
- NL. Nagaland;
- OR. Odisha;
- PB. Punjab;
- PY. Puducherry;
- RJ. Rajasthan;
- SK. Sikkim;

TN. Tamil Nadu;
 TR. Tripura;
 UP. Uttar Pradesh;
 UT. Uttarakhand;
 WB. West Bengal;

weight—Sample weight (individual).

year—Year of observation.

cohort—Year of birth.

1905. 1905-1909;
 1910. 1910-1914;
 ...
 2005. 2005-2009;

death—Death status.

0. Alive;
 1. Death;

2.4 Setting up the Workspace and Importing Parameters

In Chaps. 3 and 4, we will project the population for India and its regions from 2010 to 2060. In the support documents provided with this book (Chapter ESM), the folders for Chaps. 3 to 6 are divided into three subfolders. The subfolder “Population” includes the disaggregated population file of the projection year by year. Before running the simulation, the only file is the base population in CSV generated in the previous step of this chapter (BasePop_2010.csv). The subfolder “Parameters” contains the parameter files in CSV (fertility rates, survival ratios, etc.) that will be used for the projection. Finally, the subfolder “Outputs”, empty before running the simulation, will contain the projection aggregated outcomes (population counts and components of growth).

Starting from the base population of 2010 we described above, we want to generate the projection population in 2015, then 2020, and so on until 2060. Every step follows exactly the same equations, though with different parameters. Thus, we only need to do the coding for one step, such as from 2010 to 2015 in the example that follows. Further, we will translate the code into a macro function that repeats the process for every step until 2060.

Before coding the microsimulation model, we need to set up the workspace. The complete code for this purpose can be found in this Chapter—BasePop.sas and is also replicated in the code of the complete microsimulation provided for other chapters.

First, using the %let statement, we define the name of the scenario in a variable stored as “scenario_name”. Further in the code, we can then use “&scenario_name” at every place where the name of the scenario has to be recalled. This allows us to simplify the creation of alternative scenarios, as the name of the scenario would just

need to be changed once (with the %let statement). In our example, the name of the scenario will first be Chap. 3. This will be changed in other chapters.

```
%let scenario_name=Chapter3;
```

Second, we define libraries, each of which is a collection of files located in a single folder.² Here, we define three libraries. The first one, called “pop”, is the folder where the population files are (including the base population). The “param” library is the one where the parameters files are located (e.g. fertility rates, mortality rates, regression parameters, etc.). Finally, the “results” library is the folder where projection outcomes are stored. All of these folders are subfolders of the folder “Chap. 3” which is recalled using &scenario_name. The datasets procedure is used to erase results from a previous preliminary run of the projection (if you didn’t run anything yet, this statement won’t do anything).

```
libname pop "C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\population ";
libname param "C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters";
libname results "C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\outputs";
proc datasets lib=results kill nolist memtype=data; quit;
```

Third, we import the CSV files that will be required and convert them into SAS files (.sas7bbat). Here, rather than repeating the code to import every file, we create a macro “import” (using %macro < code > %mend;) and call it with %import. The macro parameter “source” stands for the path of the CSV file, while the parameter “destination” is the name and location of the converted file. For instance, in the first call of the macro, we import the CSV file of the base population (base_pop2010.csv) and convert it into a SAS file stored in the library pop. The other imported files are the parameters for the events:

- survival_a.csv are survival ratios (sx) for adults (15+) are.
- survival_k.csv are survival ratios (sx) for kids (less than 15 yo). Kids and adults are split into two files, because predictors are different (the education is used for adults, while the education of the mother is used for kids).
- fertility.csv are fertility rates.
- srb.csv are the sex ratios at birth.
- education.csv are the transition rates for the progression of education.
- dom_mig.csv are the mobility rates between regions.
- ur.csv are the reclassification rates of rural to urban area.
- lfp.csv are logit regression parameters for the modeling of the labour force participation (used in Chap. 4).
- formal.csv are logit regression parameters for the modeling of the sector of activity (used in Chap. 4).
- lfp_imput.csv are logit regression parameters for the imputation of the labour force participation in the base population (used in Chap. 4).

² If you use SAS University Edition, the folder needs to be in the shared folder specified when configuring the server. For the standard SAS, the folder can be located anywhere.

- `formal_imput.csv` are logit regression parameters for the imputation of the sector of activity in the base population (used in Chap. 4).

```

/*Importing parameters and data*/
%MACRO import(source,destination);
PROC IMPORT OUT=%destination
           DATAFILE=%source
           DBMS=csv REPLACE;
           GETNAMES=YES;
           guessingrows=1000;

RUN;
%MEND;

%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\Population\POP_2010.csv",pop.pop_2010);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\survival_a.csv",param.survival_a);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\survival_k.csv",param.survival_k);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\fertility.csv",param.fertility);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\srb.csv",param.srb);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\education.csv",param.education);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\dom_mig.csv",param.dom_mig);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\ur.csv",param.ur);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\lfp.csv",param.lfp);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\lfp_imput.csv",param.lfp_imput);
%import
("C:\Users\Guillaume\Desktop\Microsimulation\%scenario_name\parameters
\parameters\formal_imput.csv",param.formal_imput);

```

The structure of each of these files will be explained in the section of the corresponding event. At this step, we also sort the imported files (the `proc sort` statement). When datasets are merged (which will be done later), they need to be sorted by the same variables.

```

proc sort data=param.survival_a; by region agegr edu sex year;run;
proc sort data=param.survival_k; by region agegr eduM sex year;run;
proc sort data=param.fertility; by region agegr edu sex year;run;
proc sort data=param.education; by region agegr sex year;run;
proc sort data=param.dom_mig; by region agegr edu sex;run;
proc sort data=param.ur; by region;run;
proc sort data=param.srb; by region year;run;

```

References

- Marois, G., & Bélanger, A. (2015). Analyzing the impact of urban planning on population distribution in the Montreal metropolitan area using a small-area microsimulation projection model. *Population and Environment*, 37, 131–156. <https://doi.org/10.1007/s11111-015-0234-7>
- Kc, S., Wurzer, M., Speringer, M., & Lutz, W. (2018). Future population and human capital in heterogeneous India. *Proceedings of the National Academy of Sciences of the United States of America*, 115, 8328. <https://doi.org/10.1073/pnas.1722359115>
- Bélanger, A., Sabourin, P., Marois, G., et al. (2019). A framework for the prospective analysis of ethno-cultural super-diversity. *Demographic Research*, 41, 293–330. <https://doi.org/10.4054/DemRes.2019.41.11>
- van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 1(3), 2011.
- Sabourin, P., Marois, G., & Bélanger, A. (2017). The base population of the CEPAM microsimulation model.
- Caron-Malenfant, É., Coulombe, S., & Grenier, D. (2017). Demosim: An overview of methods and data sources.
- Marois, G., Bélanger, A., & Lutz, W. (2020). Population aging, migration, and productivity in Europe. *Proceedings of the National Academy of Sciences of the United States of America*, 117, 7690. <https://doi.org/10.1073/pnas.1918988117>
- Van Hook, J., Bélanger, A., Sabourin, P., & Morse, A. (2020). Immigration selection and the educational composition of the US labor force. *Population and Development Review*, 46, 321–346. <https://doi.org/10.1111/padr.12315>
- Van Imhoff, E., & Post, W. (1998). Microsimulation methods for population projection. *Population*, 10, 97–138.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.