Chapter 3

Converting a Cohort Component Model into a Microsimulation Model

Abstract In this chapter, we show and explain the code that reproduces the multi-state projection of India described in Chap. 2 into a microsimulation model. The microsimulation code is divided into modules for each demographic event, namely the mortality, the education, the fertility, the domestic migration, and the reclassification of rural to urban areas. Section by section, we explain the code for the simulation and the production of outputs. We also a basic validation of the mode. The code file “Chapter 3—Replicating multistate.sas” contains the final complete code that generates the simulation for 2010–2060, including the setting up of the workspace (see Chap. 2).

Keywords Microsimulation · Population projection · Demography · Method · SAS

3.1 Mortality Event

The parameter files for the mortality module are `survival_a` (for adults’ mortality) and `survival_k` (for children’s mortality). As a reminder, parameters for adults and children are split into two files because they don’t have the same predictors. However, both files have the same structure. In this example, survival ratios (`sx_a` or `sx_k`) by year, age, sex, region and education (for adults)/education of the mother (for kids) are used for the modeling of mortality. Files are structured such that all parameters are in one single column, and each line corresponds to one possible combination of education category, age group, sex and region, as shown below in the image of the first lines of the dataset `survival_a`. The value 0.990324 for `sx_a` represents the probability of surviving the next 5 years (up to 2015) for a male (`sex = 0`) in the age group 15–19 (`agegr = 15`) with no education (`edu = e1`) living in a rural area of

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-79111-7_3) contains supplementary material, which is available to authorized users.

	A	B	C	D	E	F
1	year	sex	agegr	edu	region	sx_a
2	2010	0	15	e1	AD_rural	0.990324
3	2010	0	20	e1	AD_rural	0.987227
4	2010	0	25	e1	AD_rural	0.984276
5	2010	0	30	e1	AD_rural	0.979242
6	2010	0	35	e1	AD_rural	0.972255
7	2010	0	40	e1	AD_rural	0.961296
8	2010	0	45	e1	AD_rural	0.945148
9	2010	0	50	e1	AD_rural	0.922248
10	2010	0	55	e1	AD_rural	0.885773

Fig. 3.1 Screenshot of the parameter file survival_a.csv (opened with Excel)

Andhra Pradesh (region = AD_rural) in 2010. Note that in the dataset survival_k, the agegr = -5 corresponds to the survival ratios for children that will be born during the period. Those will be used in the fertility module (Fig. 3.1).

We need to allocate to all individuals of the base population of 2010 their corresponding survival ratio. To do so, we first merge the parameters file survival_a to the population of 2010 in a temporary dataset (pop_survival1) located in the temporary library “work”. We specify variables to take into account in the merging with the “by” statement, in this case: region, agegr, edu, sex and year. The option “in = in1” followed by “if in1” is used to keep only observations from the population file (in other words, if there is no match in the population file for a specific survival ratio, the line will be deleted).

```

/*Mortality module*/
/*For children age<15, we use the education of the mother.
Therefore, we pick parameters in a different dataset*/
data work.pop_survival1;
merge pop.pop_2010(in=in1) param.survival_a;
by region agegr edu sex year;
if in1;
run;

```

Similarly, after sorting pop_survival1, we then create a second temporary dataset, pop_survival2, in which we add survival ratios for kids (using eduM—the education of the mother—in the merge statement rather than edu). In the base population, we do not know what the education of the mother. Thus, for those cohorts aged 0–14 in 2010, no differentials by mother’s education are implemented. Those differentials will only be applied for new cohorts generated throughout the projection.

```
proc sort data= work.pop_survival1; by region agegr eduM sex year;run;
data work.pop_survival2;
merge work.pop_survival1(in=in1) param.survival_k;
by region agegr eduM sex year;
if in1;
(...)
```

Depending on the age of the individual (below or above 15), we attribute the appropriate *sx* (either from *survival_a*, which are *sx_a*, or from *survival_k*, which are *sx_k*).

```
(...)
if agegr<=15 then sx=sx_k;
if agegr>15 then sx=sx_a;
(...)
```

Now that each individual has their own survival ratio, the temporary dataset is ready to simulate the mortality event with a random experiment. For this, we compare *sx* to a random number (from 0 to 1) following a uniform distribution generated with “*rand(‘uniform’)*”. If *sx* is lower than the random number, then the individual will die during the period (death = 1). At this point, we don’t yet delete him from the dataset, because we still need to consider him in the exposed population of other events that will be modelled in further steps such as fertility.

```
(...)
if rand('uniform')>sx then death=1;
(...)
```

Once we know who will survive and who will die during the period, we can then remove the survival ratios with the *drop* statement, since they will not be used anymore (different *sx* will be used for further periods).

```
drop sx sx_a sx_k;
run;
```

3.2 Education Module

The education variable has 6 categories. Transitions in education may only occur in one direction (from a lower level to the next upper level) and only at certain ages. Children below the age of 15 are classified in a specific category in the outputs. The education variable thus starts at the age of 15. Before that age, the education of the mother is used in events such as mortality. The parameters for the education module are stored in the file *education.csv*, the structure of which is shown in Fig. 3.2.

Transitions vary by sex, age, region, and year. There are two types of parameters, and they start with the prefixes “*e*” and “*pe*”, respectively. Parameters starting with “*e*” are only applied to the population age 10–14 at time *t* and correspond to educational attainment at age 15–19 (time *t* + 5). For instance, among males (sex = 0) who were age 10 (agegr = 10) in 2010 and living in a rural part of Andhra Pradesh (region = AD_rural), 13.9% will have a complete primary education (e3) as a higher level

	A	B	C	D	E	F	G	H	I	J	K	L
1	sex	agegr	year	region	e2	e3	e4	e5	e6	pe4	pe5	pe6
2	0	10	2010	AD_rural	0.019676	0.139038	0.17938	0.575099	0.018768	0	0	0
3	0	15	2010	AD_rural	0	0	0	0	0	0	0.343771	0.361183
4	0	20	2010	AD_rural	0	0	0	0	0	0.012108	0.023131	0.085319
5	0	25	2010	AD_rural	0	0	0	0	0	0.018848	0.048247	0.010175
6	0	10	2015	AD_rural	0.014472	0.143229	0.185251	0.594921	0.019408	0	0	0
7	0	15	2015	AD_rural	0	0	0	0	0	0	0.473058	0.376329
8	0	20	2015	AD_rural	0	0	0	0	0	0.008798	0.072366	0.09179
9	0	25	2015	AD_rural	0	0	0	0	0	0.015036	0.031584	0.007449
10	0	10	2020	AD_rural	0.010552	0.145952	0.189266	0.608558	0.019853	0	0	0
11	0	15	2020	AD_rural	0	0	0	0	0	0.143779	0.584226	0.389758
12	0	20	2020	AD_rural	0	0	0	0	0	0.00583	0.158749	0.101361
13	0	25	2020	AD_rural	0	0	0	0	0	0.011746	0.023993	0.006875
14	0	10	2025	AD_rural	0.007655	0.147451	0.191666	0.617543	0.020136	0	0	0
15	0	15	2025	AD_rural	0	0	0	0	0	0.285765	0.678805	0.401433
16	0	20	2025	AD_rural	0	0	0	0	0	0.003337	0.239352	0.110323
17	0	25	2025	AD_rural	0	0	0	0	0	0.008562	0.015157	0.005805

Fig. 3.2 Screenshot of the parameter file education.csv (opened with Excel)

of education at age 15–19 in 2015. The category e1 is omitted in parameters, as it corresponds to 1 minus the sum e2 to e6.

The second type (pe) of parameter is the probabilities of change from level e3 to e4 (pe4), from e4 to e5 (pe5) and from e5 to e6 (pe6) between t and $t + 5$. These parameters are applied from the age group 15–19 to 25–29 at time t (so 20–24 to 30–34 at time $t + 5$). In our example, a man in the age group 15–19 living in a rural part of Andhra Pradesh and having an upper secondary education in 2010 has a 36.1183% chance of completing postsecondary education by 2015.

There are no transitions between other education categories, such as between incomplete primary (e2) and complete primary (e3), as at the age of 15–19, most of those individuals who do not have at least a complete primary education (e3) are already out of the education system, and their current educational attainment is the one they will have for the rest of their life.

The first step in the code of the education module is the merging of the parameter file (education) to the last population file resulting from the mortality module (work.pop_survival2) by the relevant variables (in our example, region, agegr, sex and year) into a new temporary population file (pop_edu) in which the education event will occur.

```
/*Education module*/
proc sort data=work.pop_survival2; by region agegr sex year;run;
data work.pop_edu;
merge work.pop_survival2(in=in1) param.education;
by region agegr sex year;
if in1;
(…)
```

We then generate a linear random variable that we store in the temporary variable “a”, and we create another temporary variable “edu_new” for which the initial value is the same as the current level of education (edu). We then simulate changes in education. First, by comparing the random variable “a” to parameters e2 to e6, we assign the educational attainment at the age 15–19 at time $t + 5$ (so to the population

in the age group 10–14 at time t). Since the education variable has many categories, we need to use cumulative probabilities in order to have only one alternative for each value of “ a ”. The default value of `new_edu` is “e1”, so if the random variable “ a ” is higher than the cumulative proportions of “e2” to “e6”, the category “e1” is kept.

```
(...)
a=rand('uniform');
edu_new=edu;
if agegr=10 then do;
    if a<e2 then edu_new='e2';
    if e2<=a<(e2+e3) then edu_new='e3';
    if (e2+e3)<=a<(e2+e3+e4) then edu_new='e4';
    if (e2+e3+e4)<=a<(e2+e3+e4+e5) then edu_new='e5';
    if (e2+e3+e4+e5)<=a<(e2+e3+e4+e5+e6) then edu_new='e6';
end;
(...)
```

For the population age 15–19 to 25–29, we then simulate transitions from the category “e3” to “e4”, from “e4” to “e5” and from “e5” to “e6” using parameters “pe4”, “pe5” and “pe6” respectively, that we compare with the random variable “ a ”.

```
(...)
if 15<=agegr<=25 then do;
    if edu in ('e3') and a<pe4 then edu_new='e4';
    if edu in ('e4') and a<pe5 then edu_new='e5';
    if edu in ('e5') and a<pe6 then edu_new='e6';
end;
(...)
```

The projection uses fertility rates by education as an input. To have good exposure for those rates, we create a variable “`edu_fert`” that is for half of the population the education at the beginning of the period (at this point, it is still the variable `edu`) and for the other half, the education at the end of the period (the variable `edu_new`).

```
edu_fert=edu;
if rand('uniform')<0.5 then edu_fert=edu_new;
```

Finally, we replace the value of `edu` with the value of `edu_new`. The variable `edu` is now the education level that will be reached at $t + 5$. We drop parameters (“e2–e6”, “pe5” and “pe6”) and temporary variables (“ a ” and “`edu_new`”).

```
(...)
edu=edu_new;
drop a edu_new e2-e6 pe4-pe6;
run;
```

3.3 Domestic Migration Module

Interregional (or domestic) migration can be modelled in different ways. In our example, we use rates in an origin–destination (O–D) matrix. For other dimensions we have modelled up to now, the number of states was small and few transitions were possible. For the mortality module, there are only two states and only one transition is possible, from alive to dead. In the education module, there are 6 states

and transitions are only possible in one direction (from a lower level to the next upper level). In the case of domestic mobility, we have in this example 70 regions and it is possible for an individual to move to any of them in any stage of life.

The original parameters from the multistate projection are an OD matrix in which each possible combination of age, sex, education, region of origin and region of destination has its own rate (Fig. 3.3). The probability of a male (sex = 0) between 30 and 34 years of age (agegr = 30) with no education (edu = e1) living in an urban area of Andaman & Nicobar Islands (region = AN_urban) will move to an urban area of Andhra Pradesh (AD_urban) is thus 0.7042%. Values in the diagonal are very high, as they represent the probability of staying in the origin region (89.6198% in this example). The matrix includes the education dimension, but in the reference scenario no differentials have been implemented. This dimension might however be used in a further extension of the assumptions.

In the case of a multistate event such as this one, we need to rearrange the file in order to have cumulative probabilities for the different alternatives, as shown in Fig. 3.4. Otherwise, if we use the same type of structure as used for the mortality module, a random number might be lower than more than one destination region. For the case presented above, the cumulative probability corresponding to the mobility from AN_urban to AD_urban would thus be 90.324%, either the probability of staying in AN_urban (89.6198%) + the probability of moving to AD_urban (0.7042%). For all cases, the value of the last option would thus be 1. Once structured in this way, only one region will correspond to a random number between 0 and 1.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	sex	agegr	edu	region	AN_urban	AD_urban	AR_urban	AS_urban	BR_urban	CH_urban	CT_urban	DN_urban	DD_urban
2	0	30 e1	AN_urban		0.896198	0.007042	0.000041	0.000285	0.000102	0.000265	0.000020	0.000020	0.000000
3	0	30 e1	AD_urban		0.000034	0.974024	0.000002	0.000025	0.000018	0.000031	0.000163	0.000009	0.000004
4	0	30 e1	AR_urban		0.000014	0.000153	0.944825	0.002429	0.000201	0.000076	0.000035	0.000000	0.000000
5	0	30 e1	AS_urban		0.000006	0.000219	0.000945	0.972931	0.000280	0.000178	0.000076	0.000005	0.000005
6	0	30 e1	BR_urban		0.000011	0.000493	0.000059	0.000373	0.956394	0.000140	0.000583	0.000064	0.000015
7	0	30 e1	CH_urban		0.000017	0.000239	0.000004	0.000067	0.000137	0.941175	0.000096	0.000015	0.000022
8	0	30 e1	CT_urban		0.000001	0.000183	0.000003	0.000015	0.000074	0.000052	0.972335	0.000004	0.000002
9	0	30 e1	DN_urban		0	0.000211	0.000000	0.000106	0.000106	0.000053	0.000106	0.961594	0.002009
10	0	30 e1	DD_urban		0.000116	0.000232	0.000039	0.000000	0.000425	0.000039	0.000077	0.001391	0.952345
11	0	30 e1	DL_urban		0.000019	0.000451	0.000007	0.000050	0.000114	0.000491	0.000073	0.000008	0.000006
12	0	30 e1	GA_urban		0.000251	0.001261	0.000006	0.000017	0.000026	0.000037	0.000023	0.000006	0.000167
13	0	30 e1	GJ_urban		0.000003	0.000201	0.000001	0.000017	0.000038	0.000038	0.000037	0.000061	0.000066
14	0	30 e1	HR_urban		0.000005	0.000181	0.000007	0.000075	0.000084	0.001341	0.000137	0.000009	0.000005
15	0	30 e1	HP_urban		0.000015	0.000328	0.000065	0.000198	0.000148	0.005537	0.000136	0.000025	0.000012
16	0	30 e1	JK_urban		0.000007	0.000283	0.000017	0.000099	0.000146	0.000509	0.000081	0.000007	0.000001
17	0	30 e1	JH_urban		0.000006	0.000149	0.000005	0.000038	0.000204	0.000060	0.000767	0.000011	0.000003
18	0	30 e1	KA_urban		0.000007	0.000936	0.000002	0.000012	0.000012	0.000041	0.000019	0.000010	0.000007
19	0	30 e1	KL_urban		0.000079	0.001018	0.000024	0.000043	0.000022	0.000045	0.000084	0.000027	0.000023
20	0	30 e1	LD_urban		0.000094	0.000377	0.000000	0.000000	0.000189	0.000000	0.000094	0.000004	0.000094
21	0	30 e1	MP_urban		0.000001	0.000204	0.000002	0.000017	0.000031	0.000040	0.001760	0.000011	0.000004
22	0	30 e1	MH_urban		0.000014	0.000521	0.000001	0.000014	0.000023	0.000039	0.000151	0.000063	0.000032
23	0	30 e1	MN_urban		0	0.000120	0.000171	0.000585	0.000060	0.000687	0.000029	0.000003	0.000000
24	0	30 e1	ML_urban		0.000028	0.000220	0.000312	0.002852	0.000101	0.000202	0.000046	0.000005	0.000000
25	0	30 e1	MZ_urban		0.000007	0.000029	0.000038	0.000237	0.000050	0.000058	0.000004	0.000000	0.000000
26	0	30 e1	NL_urban		0	0.000096	0.000191	0.001907	0.001109	0.000100	0.000048	0.000000	0.000000
27	0	30 e1	OR_urban		0.000014	0.001187	0.000008	0.000056	0.000140	0.000036	0.000972	0.000025	0.000015
28	0	30 e1	PY_urban		0.000002	0.000332	0.000002	0.000012	0.000040	0.000025	0.000017	0.000010	0.000000

Fig. 3.3 Screenshot of the origin–destination file used to create the parameter file dom_mig (opened with Excel)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	sex	agegr	edu	region	AN_urban	AD_urban	AR_urban	AS_urban	BR_urban	CH_urban	CT_urban	DN_urban	DD_urban
2	0	30	e1	AN_urban	0.896198	0.90324	0.903281	0.903566	0.903668	0.903933	0.903953	0.903973	0.903973
3	0	30	e1	AD_urban	0.000034	0.974058	0.97406	0.974085	0.974103	0.974134	0.974297	0.974306	0.97431
4	0	30	e1	AR_urban	0.000014	0.000167	0.944992	0.947421	0.947622	0.947698	0.947733	0.947733	0.947733
5	0	30	e1	AS_urban	0.000006	0.000225	0.00117	0.974101	0.974381	0.974559	0.974635	0.97464	0.974645
6	0	30	e1	BR_urban	0.000011	0.000504	0.000563	0.000936	0.95733	0.95747	0.958053	0.958117	0.958132
7	0	30	e1	CH_urban	0.000017	0.000256	0.00026	0.000327	0.000464	0.941639	0.941735	0.94175	0.941772
8	0	30	e1	CT_urban	0.000001	0.000184	0.000187	0.000202	0.000276	0.000328	0.972663	0.972667	0.972669
9	0	30	e1	DN_urban	0	0.000211	0.000211	0.000317	0.000423	0.000476	0.000582	0.962176	0.964185
10	0	30	e1	DD_urban	0.000116	0.000348	0.000387	0.000387	0.000812	0.000851	0.000928	0.002319	0.954664
11	0	30	e1	DL_urban	0.000019	0.00047	0.000477	0.000527	0.000641	0.001132	0.001205	0.001213	0.001219
12	0	30	e1	GA_urban	0.000251	0.001512	0.001518	0.001535	0.001561	0.001598	0.001621	0.001627	0.001794
13	0	30	e1	GJ_urban	0.000003	0.000204	0.000205	0.000222	0.00026	0.000298	0.000335	0.000396	0.000462
14	0	30	e1	HR_urban	0.000005	0.000186	0.000193	0.000268	0.000352	0.001693	0.00183	0.001839	0.001844
15	0	30	e1	HP_urban	0.000015	0.000343	0.000408	0.000606	0.000754	0.006291	0.006427	0.006452	0.006464
16	0	30	e1	JK_urban	0.000007	0.00029	0.000307	0.000406	0.000552	0.001061	0.001142	0.001149	0.00115
17	0	30	e1	JH_urban	0.000006	0.000155	0.00016	0.000198	0.002202	0.002262	0.003029	0.00304	0.003043
18	0	30	e1	KA_urban	0.000007	0.000943	0.000945	0.000957	0.000969	0.00101	0.001029	0.001039	0.001046
19	0	30	e1	KL_urban	0.000079	0.001097	0.001121	0.001164	0.001186	0.001231	0.001315	0.001342	0.001365
20	0	30	e1	LD_urban	0.000094	0.000471	0.000471	0.000471	0.00066	0.00066	0.000754	0.000848	0.000942
21	0	30	e1	MP_urban	0.000001	0.000205	0.000207	0.000224	0.000255	0.000295	0.002055	0.002066	0.00207
22	0	30	e1	MH_urban	0.000014	0.000535	0.000536	0.00055	0.000573	0.000612	0.000763	0.000826	0.000858
23	0	30	e1	MN_urban	0	0.00012	0.000291	0.000876	0.000936	0.001623	0.001652	0.001655	0.001655
24	0	30	e1	ML_urban	0.000028	0.000248	0.00056	0.003412	0.003513	0.003715	0.003761	0.003766	0.003766
25	0	30	e1	MZ_urban	0.000007	0.000036	0.000119	0.000356	0.000406	0.000464	0.000468	0.000468	0.000468
26	0	30	e1	NL_urban	0	0.000096	0.000287	0.002194	0.003303	0.003403	0.003451	0.003451	0.003451
27	0	30	e1	OR_urban	0.000014	0.001201	0.001209	0.001265	0.001405	0.001441	0.002413	0.002438	0.002453
28	0	30	e1	PV_urban	0.00002	0.000352	0.000354	0.000366	0.000406	0.000431	0.000448	0.000458	0.000458

Fig. 3.4 Screenshot of the parameter file dom_mig.csv (opened with Excel)

For the code, in a new population file (pop_dm), we merge the parameter file (dom_mig) to the last population file (pop_edu). Each individual will therefore have a (cumulative) probability of staying in the origin region and a (cumulative) probability of moving to any other region.

```
/*Domestic migration module*/
proc sort data= work.pop_edu; by region agegr edu sex year;run;
data work.pop_dm;
merge work.pop_edu (in=in1) param.dom_mig;
by region agegr edu sex;
if in1;
(...)
```

We store the origin region in a new variable “oldregion”, which will be used in subsequent steps. We generate a random number that we store in a temporary variable “a”. The random number needs to be tracked because the same number needs to be used in different places in modeling the event. We then create an array called *prob* that states the regions ({ * } is used to specify that the number of elements in the array is the number of arguments that follows).

```
(...)
oldregion=region;
a=rand('uniform');
array prob(*) AN_urban AD_urban AR_urban AS_urban BR_urban CH_urban CT_urban
DN_urban DD_urban DL_urban GA_urban GJ_urban HR_urban HP_urban JK_urban
JH_urban KA_urban KL_urban LD_urban MP_urban MH_urban MN_urban ML_urban
MZ_urban NL_urban OR_urban PY_urban PB_urban RJ_urban SK_urban TN_urban
TR_urban UT_urban UP_urban WB_urban AN_rural AD_rural AR_rural AS_rural
BR_rural CH_rural CT_rural DN_rural DD_rural DL_rural GA_rural GJ_rural
HR_rural HP_rural JK_rural JH_rural KA_rural KL_rural LD_rural MP_rural
MH_rural MN_rural ML_rural MZ_rural NL_rural OR_rural PY_rural PB_rural
RJ_rural SK_rural TN_rural TR_rural UT_rural UP_rural WB_rural;
(...)
```

Using the *do* statement on all elements stated in the array *prob*, we compare the random number “a” to the cumulative probability for every destination region *i*. When “a” is lower than the cumulative probability of region *i*, but higher than the one for the previous region (*i*−1), the label of the corresponding element in the array (we select it with the *vname* statement) is then assigned to the temporary variable “newregion”. For the first region in the array (*i* = 1, AN_urban in our example), we just compare the cumulative probability to the random number a (*i*−1 doesn’t exist). Thus, for any random number generated, only one option is possible.

```
(...)
do i=1 to dim(prob);
    if i=1 and a<prob(i) then newregion=vname(prob[i]);
    if 1<i and prob(i-1)<=a<prob(i) then newregion=vname(prob[i]);
end;
(...)
```

Once done, the variable “region” corresponds to the region of residence at time *t*, while the variable newregion corresponds to the region at time *t* + 5. As we did for the education module, we need to create a special variable in order to have the proper exposed population in the fertility module. We can assume that half of the migrants are exposed to the fertility rate of the destination, while the other half are exposed to the origin’s fertility rates. Therefore, we create accordingly a variable “region_fert” that will be used for the region of exposure in the fertility module.

```
(...)
region_fert=region;
if rand('uniform')<0.5 then region_fert=newregion;
(...)
```

When variables region and newregion are the same, the individual did not move. In order to count the number of migrants, we create a variable “dom_mig” that takes the value of 1 when “newregion” and “region” are different. We then replace the value of the variable “region” with the new region. The number of emigrants by region can therefore be obtained by doing a crosstable of dom_mig and oldregion, while the number of immigrants can be obtained by doing a crosstable of “dom_mig” and “region”. Note that in our example, only the population that survives until *t* + 5 (death = 0) can migrate, because data to estimate the mobility come from the question on the previous residence in censuses, which by definition is asked to the surviving population only.

```
(...)
if death=0 and newregion ne region then do; dom_mig=1; region=newregion; end;
(...)
```


Finally, we drop the temporary variables (i a newregion) and variables for parameters (AN_urban--WB_rural), because they will not be used anymore.

```
(...)
drop i a AN_urban--WB_rural newregion;
run;
```

In our example, rates are constant throughout the projection. To make them evolve, we can simply add a column “year” to the dom_mig file, filling in the rates for the future, and adding “year” as in merging variables.

3.4 Fertility Module

The fertility module is a bit more complex since when the event happens, a new individual (having its own characteristics) must be generated and added to the population file. Also, it needs to take into account that this new individual might not survive between birth and time $t + a$.

Over a 5-year-period, a cohort age X at time t will also be exposed to the risk of fertility of age group X and $X + 5$. The Lexis diagram below (Fig. 3.5) shows an example of exposure between t and $t + 5$ for the cohort age 15–19 at time t (the red lozenge). The cohort has half of the risk in the age group 15–19 (in orange), half of the risk in 20–24 (in light blue). The rates for the age group 20–24 (green square) is thus applied to two cohorts.

Fig. 3.5 Lexis diagram

To expose the population to appropriate fertility rates, in a new temporary population file (pop_birth) created from the last population file (pop_dm), we create a new variable “agegr_fert” that corresponds to the variable used for the age at fertility. We assume half of the population will change age group before the middle of the period. Therefore, agegr_fert is the same as the age group for half of the population, while for the other half, it corresponds the age group + 5. The split is done randomly by comparing a random number distributed uniformly between 0 and 1 generated with rand (‘uniform’) to 0.5.

```
/*Fertility module*/
/*Adjusting age for exposure*/
data work.pop_birth;
set work.pop_dm;
agegr_fert=agegr;
if rand('uniform')<0.5 then agegr_fert=agegr+5;
run;
(...)
```

Similar exposure-adjusted variables have been created for the education and the region of residence in their respective modules. After sorting the population file properly, in a new population (pop_birth1), we then implement the fertility rates from the parameter file param.fertility. Note that the link variables used are specific to the fertility module: agegr_fert, region_fert and edu_fert. Then, in a similar way, in a new population file pop_birth2, we add the parameters file for the sex ratio at birth by region and year (param.srb).

```
/*Implementing parameters*/
proc sort data=work.pop_birth; by region_fert agegr_fert edu_fert sex year;run;
data work.pop_birth1; merge work.pop_birth(in=in1) param.fertility; by region_fert
agegr_fert edu_fert sex year; if in1; run;
proc sort data=work.pop_birth1; by region_fert year;run;
data work.pop_birth2;
merge work.pop_birth1(in=in1) param.srb ;
by region_fert year;
if in1;
(...)
```

We can then proceed to the fertility event. First, we reset the variable identifying the mother of a young kid, since children born during the previous period will change age group. The events are applied to surviving individuals (death = 0) and to half of the deaths that we select randomly, as we assume that they die on average in the middle of the period. Then, using the Monte Carlo method, we simulate the fertility event: when a random uniform variable between 0 and 1 is lower than the fertility rate, then the individual gives birth during the period (identified with the variable young_kid).

```
/*Fertility event*/
young_kid=0;
if death=0 or (death=1 and rand('uniform')<0.5) then do;
  if rand('uniform')<5*asfr then young_kid=1; *Has a kid age 0-4;
end;
(...)
```

Once a woman gives birth, we generate a new individual to add to the population: her baby. For this purpose, we use the output statement,¹ which duplicates the line of the mother. We then have to determine the characteristics of the baby. A flag variable birth is first created (birth = 1), in order to compile the number of births in a further step. We also create a variable stating the age of the mother that can further be used in outputs. We assign the age group “-5” to the baby (as a reminder, -5 stands for those who are born between t and $t + 5$, while 0 is for those aged 0–4 at time t). We also define the year of birth in the variable “cohort” (2010 would thus stand for babies born between 2010 and 2014).

Then, with another random experiment, we determine if the baby is a boy or a girl using the sex ratio at birth (SRB) parameter (the variable srb—sex ratio at birth, which comes from the file srb.csv). The srb is expressed in this case as the number of baby girls for 1000 baby boys. Although in most countries, this is usually always about the same (about 953 girls for 1000 boys), it’s much lower in India because of sex-selective abortions that are very common in some regions (Retherford and Roy 2003). Thus, the SRB has region- and year-specific values, allowing us to build assumptions for its future evolution.

```
(...)
/*Create newborn and set their characteristics*/
output;
if young_kid=1 then do;
  birth=1;
  age_moth=agegr_fert;
  agegr=-5; young_kid=0; death=0; cohort=year; region_birth=region_fert;
  dom_mig=0; if region_birth ne region then dom_mig=1;
  sex=0; if rand('uniform') < (srb/(srb+1000)) then sex=1;
  eduM=edu_fert;
  edu='e1';
  if 5*asfr>1 then weight=weight*5*asfr;
output;
end;
run;
```

Since the line for the baby is a duplication of the line of the mother, the sample weight of the mother is automatically transferred to the baby. The region is also transferred, but, as a reminder, since the domestic mobility module occurs before, it now corresponds to the region at the end of the period and not the region at birth. The region at birth is the variable used for the exposure to fertility, region_fert, that we store in another variable, “region_birth”, which will be used when generating the outputs of the components of growth. When this region of birth differs from the region of residence at the end of the period, the baby is flagged as a domestic migrant (dom_mig = 1).

We also keep track of the education of the mother in the variable eduM (which is used for the mortality until the age of 15), and set the education of the baby to e1 (no education).

The projection uses 5-year steps, so it is possible to have a fertility rate higher than 1 for a specific group. In such cases, the rate will always be higher than the

¹ See the SAS documentation for more details about the output statement: <https://documentation.sas.com/?docsetId=lestmtsref&docsetTarget=n11ltvbis7ye1an1eryo4leh2mck.htm&docsetVersion=9.4&locale=en>.

random number generated for the fertility event, and some births will be missing. On average, more than one birth occurs per woman, so a simple way to get the correct number of births is to adjust the population weight of the baby by multiplying it by the fertility rate (such as what would be done in a deterministic microsimulation; see Chap. 6). In the case of a woman with a weight of 1000 and a fertility rate of 1.2, the weight of the newborn would thus be 1200.

The last step of the fertility module is to simulate the survival of the baby between the birth and the end of the period ($t + 5$). For this purpose, we will again need to use the parameters file containing survival ratios for kids (survival_k). We merge it with our last population file (pop_birth2), by region, age, sex, education of the mother and year. We then simulate the mortality event for babies born between t and $t + 5$ (identified by agegr = -5), the same way we did in the mortality module for the living population at time t . Because the mortality event occurs before migration, the region at death is the one where the baby is born (region = region_birth). Finally, since we don't need them anymore, we drop temporary variables and parameters.

```
/*Simulate the survival of babies*/
proc sort data=work.pop_birth2; by region agegr eduM sex year;run;
data work.pop_birth3;
merge work.pop_birth2(in=in1) param.survival_k;
by region agegr eduM sex year; if in1;

if agegr=-5 then do;
    if rand('uniform')>sx_k then do; death=1; region=region_birth; end;
end;
drop sx_k asfr srb agegr_fert edu_fert region_fert;
run;
```

3.5 Reclassification of Rural to Urban Areas

With the rapid urbanisation of India, cities are spreading and absorbing some rural areas. Thus, some people's environments can switch from rural to urban without migration. The multistate model provides annual rates of the reclassification of rural areas by region, age, sex and education. Those rates are included in the parameters file "urbanisation", in the column "ur", as illustrated in Fig. 3.6. Rates are region-specific and constant throughout the projection. The number 0.040471 for the region AD_rural mean that about 4% of the population living in a rural area of Andhra Pradesh are reclassified as living in an urban area every 5 years. We can note that some rates are very high, such as the one for Dadra & Nagar Havel (DN_rural) which is almost 67%. Regions with such high rates have small areas and rapid urbanisation. Consequently, the rural areas in such regions are expected to disappear within the next few decades.

The reclassification event occurs after all other demographic events. After sorting the last population file (pop_birth3) by region, we merge it with the parameters file that includes the reclassification rates (param.ur) in a new temporary population file

Fig. 3.6 Screenshot of the parameter file ur.csv (opened with Excel)

	A	B
1	region	ur
2	AD_rural	0.040471
3	AN_rural	0.033443
4	AR_rural	0.009291
5	AS_rural	0.019746
6	BR_rural	0.004528
7	CH_rural	0.508552
8	CT_rural	0.003557
9	DD_rural	0.214488
10	DL_rural	0.569817
11	DN_rural	0.667174
12	GA_rural	0.235545
13	GJ_rural	0.021988
14	HP_rural	0.028253
15	HR_rural	0.003228
16	JH_rural	0.022537
17	JK_rural	0.033185
18	KA_rural	0.017052
19	KL_rural	0.289824
20	LD_rural	0.585807
21	MH_rural	0.006803
22	ML_rural	0.02518
23	MN_rural	0.064766
24	MP_rural	0.020649
25	MZ_rural	0
26	NL_rural	0.034304
27	OR_rural	0.016038
28	PB_rural	0.070555
29	PY_rural	0.024741
30	RJ_rural	0.01492
31	SK_rural	0.021413
32	TN_rural	0.047827
33	TR_rural	0.084389
34	UP_rural	0.030316
35	UT_rural	0.010827
36	WB_rural	0.066503

called `pop_reclass`. Each individual living in a rural area now has a probability of being reclassified as living in an urban area.

For the coding of the event, we first store the old region in another variable (`oldregion2`). This variable will be used later when generating outputs of the projection. The event only applies to those who are still alive at this point of the projection (`death = 0`) and those living in a rural area, which we can select using the `substr` function. Options in the brackets of this function mean that we look to see if the 5 characters starting from the 4th character of the variable “region” contain the string “rural”.

```
/*Reclassification of rural to urban areas*/
proc sort data=work.pop_birth3; by region;run;
data work.pop_reclass;
merge work.pop_birth3 (in=in1) param.ur;
by region;
if in1;

oldregion2=region;
if death=0 and substr(region,4,5)='rural' then do;
  if rand('uniform')<ur then do;
    region=tranwrd(region,'rural','urban');
    reclass=1; end;
end;

drop ur;
run;
```

Once the appropriate population is selected, the event is modelled with the Monte Carlo method, comparing a random number to the reclassification rate (`ur`). When the event happens, the function `tranwrd` is used to change the value of the region variable from rural to urban. The first term in the brackets of this function selects the variable to modify (`region`); the second term identifies the string of characters to replace (`'rural'`); and the third term is the string of characters of substitution (`'urban'`). We keep track of reclassified individuals in a new variable labelled “`reclass`”, which will be used when generating outputs. Finally, we drop the reclassification rates (`ur`) from the dataset, as they won’t be used anymore.

3.6 Preparing the Population File for the Next Step

We have simulated changes in the population characteristics for the first 5 years of the projection. All demographic events are completed. We now have to prepare the population file for the next step. First, we make our population ages and we increment the year in a small “time module”. Starting from our last population file (`pop_reclass`), we create a new one (`pop2`) in which the year is increased by 5 as well as the age of survival (`death = 0`). For those who died during t and $t + 5$ (except for babies born during the period), we assume that only half died before their anniversary. This last statement will only be useful for outputs related to the age distribution of diseases.

```

/*Time module - Updating characteristics*/
data work.pop2;
set work.pop_reclass;
if death=0 or agegr=-5 then agegr=agegr+5;
if death=1 and agegr ne -5 and rand('uniform')<0.5 then agegr=agegr+5;
year=year+5;
run;

```

The population file pop2 represents the population at the end of the period, but still includes the deaths, which are flagged with the variable death = 1. For the next period of the projection, those individuals need to be removed. From pop2, we create a new population file “pop_2015” that is the surviving population in 2015. This file is stored in the library pop because it is the final projected population for 2015. We delete the deaths from the dataset and we drop variables that were used as flags for births, migrants, and individuals who were reclassified from rural to urban areas.

```

/*Cleaning the population file for next period*/
data pop.pop_2015;
set work.pop2;

if death=1 then delete;
drop birth dom_mig reclass oldregion oldregion2 region_birth;
run;

```

3.7 Generating Outputs

The population file pop_2015 is the dataset corresponding to the surviving population in 2015. The file pop_reclass is the last file containing a record of demographic events (with variables birth, death, dom_mig and reclass). From these two files, we will generate projection outputs. Variables included in this step of the model may vary according to the needs of the user. The code for generating outputs has no impact on the projection.

First, from the population file pop_2015, we create a flat file outputpop that aggregates the population by sex, region, age group and education. The option noprint is used, because the output is stored in a separate dataset (out=) and we don’t need to display the table in SAS. The option list is used to create one single table in which each column corresponds to a variable and each row is a specific combination of categories. The options norow, nocol, nopercents and nocum are stated in order to keep only frequencies in the table and to remove percentages and sums. In the output file, the column “count” is renamed “pop”, as it represents the population size.

```

/*Generating outputs*/
/*Population per age sex region and education*/
proc freq data=pop.pop_2015 noprint;
table year*agegr*sex*edu*region /list norow nocol nopercents nocum
out=work.outputpop (rename=(count=pop) drop=percent);
weight weight;
run;

```

An excerpt of the resulting file is shown in Fig. 3.7. Remember that since

	YEAR	agegr	SEX	edu	region	Frequency Count
1	2015	0	0	e1	AD_rural	2065319.0371
2	2015	0	0	e1	AD_urban	1316830.3415
3	2015	0	0	e1	AN_rural	9347.477045
4	2015	0	0	e1	AN_urban	4516.1015791
5	2015	0	0	e1	AR_rural	46320.596953
6	2015	0	0	e1	AR_urban	15613.463698
7	2015	0	0	e1	AS_rural	1449413.2988
8	2015	0	0	e1	AS_urban	242234.76644
9	2015	0	0	e1	BR_rural	5845179.0957
10	2015	0	0	e1	BR_urban	618466.87054
11	2015	0	0	e1	CH_rural	103.71906272
12	2015	0	0	e1	CH_urban	44317.066877
13	2015	0	0	e1	CT_rural	1205542.2086
14	2015	0	0	e1	CT_urban	267268.05909
15	2015	0	0	e1	DD_rural	4677.4048404
16	2015	0	0	e1	DD_urban	8288.0561332
17	2015	0	0	e1	DL_rural	11171.145541
18	2015	0	0	e1	DL_urban	679071.59736

Fig. 3.7 Screenshot of the results file outputpop (opened with SAS)

microsimulation is based on a stochastic process using random experiments, results may differ slightly on each run.

In our example, we also want to generate outputs for components of the growth. We need to use the temporary file pop2 for that purpose, since it is the last one that keeps track of demographic events, including births, deaths, domestic migration (both inflows and outflows), and individuals reclassified from rural to urban areas. We use the FREQ procedure to create frequency tables for each component.

For the birth event, we select births ($\text{birth} = 1$) with the *where* statement. In the *table* statement, we split births by age of the mother, education of the mother, sex, and region of birth, and in options, we specify that we want to keep only frequencies. We also rename the variable education of the mother ($\text{eduM} = \text{edu}$), the age of the mother ($\text{age_moth} = \text{age}$) and the region of birth ($\text{region_birth} = \text{region}$), because we will later merge this file with the population count. The resulting table is stored in the dataset *births* in the temporary library *work*. An excerpt of the table is shown in Fig. 3.8.

```

/*Components of population growth*/
proc freq data=work.pop_reclass noprint;
  table year*age_moth*eduM*region_birth/list norow nocol nopercnt nocom
out=work.birth(rename=(count=births age_moth=agegr eduM=edu region_birth=region)
drop=percent);
  weight weight;
  where birth=1;
run;

```

	year	agegr	edu	region	Frequency Count	sex
1	2015	15	e1	AD_rural	256329.93481	1
2	2015	15	e1	AD_urban	60235.141728	1
3	2015	15	e1	AN_rural	335.49081605	1
4	2015	15	e1	AN_urban	196.16959913	1
5	2015	15	e1	AR_rural	7472.0108875	1
6	2015	15	e1	AR_urban	3643.6293627	1
7	2015	15	e1	AS_rural	417782.12225	1
8	2015	15	e1	AS_urban	29036.251312	1
9	2015	15	e1	BR_rural	804778.52632	1
10	2015	15	e1	BR_urban	72230.703591	1
11	2015	15	e1	CH_rural	147.14945392	1
12	2015	15	e1	CH_urban	978.53899438	1
13	2015	15	e1	CT_rural	144660.53704	1

Fig. 3.8 Screenshot of the results file births (opened with SAS)

Using flag variables identifying deaths, migrants and reclassified individuals, we generate similar frequency tables for deaths, inflows, outflows, and changes from rural to urban areas, though we replace the education and age of the mother with the education and age of the individual (and remove the rename statement accordingly). For outflows and individuals reclassified from rural to urban areas, we need to state the former region of residence (oldregion and oldregion2, respectively) rather than the current one, and rename the variable in the dataset in order to allow us to merge it with another dataset later. For reclassified individuals, we also need to exclude domestic migrants (dom_mig ne 1) from the tables in order to avoid double counts. For both rural to urban reclassification and domestic migration, because they are implemented after the mortality event, we also add death = 0 in order to avoid counting a baby that died between the birth and the end of the year (the rest of the deceased population was already excluded in the modeling of the event).

```

proc freq data=work.pop2 noprint;
  table year*agegr*sex*edu*region/list norow nocol nopercnt nocum
out=work.death(rename=(count=deaths) drop=percent);
  weight weight;
  where death=1;
run;

proc freq data=work.pop2 noprint;
  table year*agegr*sex*edu*region/list norow nocol nopercnt nocum
out=work.inflow(rename=(count=inflow) drop=percent);
  weight weight;
  where dom_mig=1 and death=0;
run;

proc freq data=work.pop2 noprint;
  table year*agegr*sex*edu*oldregion/list norow nocol nopercnt nocum
out=work.outflow(rename=(count=outflow oldregion=region) drop=percent);
  weight weight;
  where dom_mig=1 and death=0;
run;

proc freq data=work.pop2 noprint;
  table year*agegr*sex*edu*region/list norow nocol nopercnt nocum
out=work.gain_urban(rename=(count=gain_urban) drop=percent);
  weight weight;
  where reclass=1 and dom_mig ne 1 and death=0;
run;

proc freq data=work.pop2 noprint;
  table year*agegr*sex*edu*oldregion2/list norow nocol nopercnt nocum
out=work.loss_rural(rename=(count=loss_rural oldregion2=region) drop=percent);
  weight weight;
  where reclass=1 and dom_mig ne 1 and death=0;
run;

```

We now have 7 datasets of outputs, one for the population count and six for components of growth. We will now merge them into a single dataset. We first need to add a column “sex”, taking the value of 1 for everyone in the file birth2015, because we want births to match the female population only.

```

/*Merging the population count and components of growth*/
data work.birth2015;
  set work.birth2015;
  sex=1;
run;

```

We can now merge all those files in a new dataset called “output2015” stored in the “results” library. Using an array called “change” that included numeric variables (selected with `_numeric_`), we switch missing values to 0 and remove decimals using the *round* statement.

```

data results.output2015;
  merge work.outputpop work.birth work.death work.inflow work.outflow work.gain_urban
work.loss_rural;
  by year agegr sex edu region;

  array change _numeric_;
  do over change;
    if change=. then change=0;
    change=round(change);
  end;
run;

```

As shown in Fig. 3.9, the resulting file output2015 now includes the population size by age, education, sex and region, as well as the components of growth for every specific sub-group.

	year	agegr	sex	edu	region	pop	births	deaths	inflow	outflow	gain_urban	loss_rural
1	2015	0	0	e1	AD_rural	2286805	0	104380	31277	36721	0	111923
2	2015	0	0	e1	AD_urban	1283619	0	18110	32821	32317	111923	0
3	2015	0	0	e1	AM_rural	8767	0	0	176	122	0	533
4	2015	0	0	e1	AM_urban	7106	0	0	2018	314	533	0
5	2015	0	0	e1	AP_rural	43621	0	217	484	173	0	65
6	2015	0	0	e1	AP_urban	15826	0	0	2180	736	65	0
7	2015	0	0	e1	AS_rural	1614077	0	82165	4264	12050	0	34701
8	2015	0	0	e1	AS_urban	197020	0	6107	10291	4051	34701	0
9	2015	0	0	e1	BR_rural	5815113	0	218505	32260	75622	0	23028
10	2015	0	0	e1	BR_urban	649526	0	14909	34233	23513	23028	0
11	2015	0	0	e1	CH_rural	548	0	6	0	205	0	4632
12	2015	0	0	e1	CH_urban	47528	0	0	8498	1178	4632	0
13	2015	0	0	e1	CT_rural	1158028	0	59435	13047	23423	0	2028
14	2015	0	0	e1	CT_urban	302290	0	8137	20510	19348	2028	0
15	2015	0	0	e1	DD_rural	2406	0	72	29	55	0	3204
16	2015	0	0	e1	DD_urban	10237	0	15	59	184	3204	0
17	2015	0	0	e1	DL_rural	17178	0	11	6116	638	0	11653
18	2015	0	0	e1	DL_urban	793251	0	6042	77001	16056	11653	0

Fig. 3.9 Screenshot of the file output2015 (opened with SAS)

3.8 Cleaning the Workspace

During the simulation of events, we created many temporary datasets. Before passing to the next period, we need to remove them from the memory. The code below is used to clean all datasets stored in the *work* library.

```
/*Clean temporary dataset*/
proc datasets lib=work kill nolist memtype=data;
quit;
```

3.9 Simulating for Next Periods

Up to now, we simulated the population for one step, from time t to $t + 5$ (2010 to 2015 in our example). We started from a base population file (pop_2010), and we ended up with another population file having the same structure for 2015 (pop_2015). This file is now the starting point for the next step, from 2015 to 2020. We could just use the same code, but call it pop_2015 rather than pop_2010 in the first module (mortality), changing the name of the final population file in the module preparing the population file for the next step (pop_2020 rather than pop_2015) and changing the years in the name of the outcome files.

Doing this, though easy, would however require many lines of code. For this kind of situation where codes are repeated with only small changes, SAS allows the user to create a macro that stores the repetitive code in a function and in which elements to change are identified as parameters. To run the code, we can further call the macro function and specify the appropriate parameters.

To create a macro, we need to insert the repeating code between statements *%macro name* and *%mend name*. Parameters are declared in brackets after stating the name of the macro, and would refer to the code identified by the same label with the prefix & in the codes.

In our example, we will create a macro called *microsim* with parameters *styr* (for starting year) and *endyr* (for ending year). The code to repeat starts by sorting the initial population file. We thus declare the macro just before this code with *%macro microsim(styr,endyr)*. Then, throughout the codes, we change labels that refer to the starting year (2010) by *&styr* and those that refer to the ending year (2015) by *&endyr*. We highlight them in yellow in the code below.

```
%macro microsim(styr,endyr);

proc sort data=pop.pop_&styr;by region agegr edu sex year;run;

/*Mortality module*/
/*For children age<15, we use the education of the mother. Therefore,
we pick parameters in a different dataset*/
data work.pop_survival1;
merge pop.pop_&styr(in=in1) param.survival_a;
by region agegr edu sex year;
if in1;
run;

(...)

/*Preparing the population file for next period*/
data pop.pop_&endyr;
set work.pop_pop2;

if death=1 then delete;
drop birth dom_mig reclass oldregion oldregion2 region_birth;
run;

/*Generating outputs*/
/*Population per age sex region and education*/
proc freq data=pop.pop_&endyr noprint;
table year*agegr*sex*edu*region /list norow nocol nopercnt nocum
out=work.outputpop(rename=(count=pop) drop=percent);
weight weight;
run;

(...)

/*Merging the population count and components of growth*/
(...)
data results.output&endyr;
merge work.outputpop work.birth work.death work.inflow work.outflow
work.gain_urban; work.loss_rural
by year agegr sex edu region;

array change _numeric_;
do over change;
if change=. then change=0;
change=round(change);
end;
run;
```

We close the macro function with the statement *%mend microsim* right after the last line of the code that needs to be repeated, which is the procedure for cleaning temporary work files.

```
/*Clean temporary dataset*/
proc datasets lib=work kill nolist memtype=data;
quit;

%mend microsim;
```

To run the microsimulation up to 2060, we could call the macro function `microsim` for every step from 2010 to 2060, for example:

```
%microsim(2010,2015);
%microsim(2015,2020);
(...)
%microsim(2055,2060);
```

Alternatively, since we repeat the macro by increasing parameters by 5 every step, we can imbed a loop function in a second macro (called *loop*) that automatizes the process. Parameters of this macro loop are the initial year of the first and the last step of the projection. When calling the macro using 2010 and 2055 as parameters, the projection will run until 2060 (2055 being the initial year of the last step that thus ends in 2060).

```
%macro loop(first,last);
  %local year;
  %do year = &first %to &last %by 5;
    %microsim (&year, %eval(&year+5))
  %end;
%mend loop;

%loop(2010, 2055);
```

The output of each year is then stored in a specific SAS format dataset (`.sas7bdat`) in the library *results*, as shown in Fig. 3.10.

To facilitate the analysis of projection outcomes, we can concatenate them together and export the resulting file in CSV. Because we want the initial population of 2010 to be included in the final dataset of results, we first need to create an output for the starting year (2010) having the same format as other outputs for projected years. We thus use the same code, but using “2010” instead of “&styr”.

Fig. 3.10 Screenshot of files included in the results library (opened with SAS)

```

/*Population by age sex region education for 2010*/
proc freq data=pop.pop_2010 noprint;
table year*agegr*sex*edu*region /list norow nocol nopercnt nocum
out=results.output2010(rename=(count=pop) drop=percent);
weight weight;
run;

data results.output2010;
set work.output2010;

array change _numeric_;
do over change;
    if change=. then change=0;
    change=round(change);
end;
run;

```

In a data step, we then concatenate all files starting with “outputpop” located in the library *results* by using the symbol ‘.’ instead of the year in the dataset name. The resulting file “outputTotal” thus contains the population outcome for all years. Values for components of growth are for the period $t-5$ to t , so they are empty for the year 2010, which is the starting year of the population.

```

/*Concatenating and exporting results*/
data results.outputTotal;
set results.outputpop;;
run;

```

Using the *export* procedure, we can then export the dataset into a CSV file located in the output folder. Personalised tables can then easily be generated with pivot tables from Excel.

```

proc export data=results.outputTotal
outfile=
'C:\Users\Guillaume\Desktop\Microsimulation\&scenario_name\outputs\outputTotal.csv'
dbms=csv
replace;
run;

```

3.10 Validation of Results

The validation of a new model is a necessary step that must be taken before using its outcomes for purposes of analysis. As with any population projection model, the aim of the microsimulation model we built is not to predict what will happen, but rather to forecast the population under conditional assumptions. In our case, those assumptions are taken from another projection model. Therefore, the microsimulation model should reproduce what occurs with that multistate model. In other words, our validation needs to confirm the demographic events are modelled correctly and that there is no error in the code. For this purpose, an external validation by error analysis is sufficient, as it is the most common procedure used to validate population projections (Grummer-Strawn and Espenshade 1991; Smith et al. 2002). If we had built our own assumptions, more sophisticated validation and sensitivity analysis

Table 3.1 Error between the multistate and the microsimulation model

Population size of the subgroup	Mean error	Mean absolute error	Mean relative error (%)
[0, 10,000[	−14	942	4
[10,000, 50,000[	−373	5,308	−2
[50,000, 100,000[	−916	9,821	−1
[100,000, ∞	−15,131	34,770	−2
All	−3,173	9,122	1

might have been required. For specific examples, see National Research Council (1991) or Caswell and Sánchez Gassen (2015).

We compared our outcomes with those we want to reproduce (the multistate model of KC et al. (2018)). In order to see if there was a systematic bias in the microsimulation model compared to the multistate model, we calculated the mean error and the mean absolute error in the projected population size in 2060 (Table 3.1). We disaggregated the population by the smallest possible comparable subgroups, these being the size of each age-, sex-, education- and region-specific group. We also split results according to the population size of the subgroup. Remember that since the microsimulation is stochastic, results may differ slightly between each run, and consequently, the difference with the multistate model results may also change slightly. We can see that the accuracy of the projection is relatively good, with a mean relative error of 1%. Because of the stochastic nature of the microsimulation, the smaller the population is, the higher the error. Thus, although the mean absolute error when the population size of the subgroup is between 0 and 10,000 is much higher and reaches 4%, it corresponds to a gap of only −14 individuals on average (mean error).

In Fig. 3.11 we compare the projected population by level of education between 2010 and 2060 from our microsimulation model to the outcome from the multistate. The microsimulation leads to very similar results, with a sharp increase in the population between 2010 to 2060 from 1.2 G to almost 1.8 G. Most of the increase is projected to be in the population with an upper secondary or postsecondary level of education, while the population with no education is projected to decline sharply.

In Fig. 3.12, we compare the projected age pyramid by education level in 2060 from each model. In Fig. 3.13, we show the population size by region in 2060. Again, we can observe that the microsimulation produces results similar to those of the multistate projection, though the discrepancy becomes larger for regions with smaller populations, given the Monte Carlo error.

Overall, the microsimulation replicates the multistate model outcomes quite well for broad aggregations, such as the population by age, sex, and education and for subgroups with relatively large populations. However, for more specific subgroups with very small populations, for instance, the men aged 95–99 with postsecondary education in an urban area of Dadra & Nagar Haveli, results may differ dramatically (183 in the multistate vs. 0 in our run of microsimulation). If we are interested in analysing outputs for those small subgroups, we could improve the accuracy of

Fig. 3.11 Comparison of projected population size of India by educational attainment from multistate model and microsimulation, 2010–2060

Fig. 3.12 Comparison of the projected age pyramid in 2060 by education in India from multistate model and microsimulation

results by increasing the sample size in the base population. Alternatively, we can also implement some events, such as mortality, using a deterministic approach rather than a stochastic one (see Chap. 6 for details).

Fig. 3.13 Comparison of the projected population size in 2060 by region, India, from multistate model and microsimulation

References

- Caswell, H., & Sánchez Gassen, N. (2015). The sensitivity analysis of population projections. *Demographic Research*, 33, 801–840.
- Grummer-Strawn, L., & Espenshade, T. J. (1991). Evaluating the accuracy of U.S. population projection models. In: *Improving information for social policy decisions—the uses of microsimulation modeling*. The National Academies Press, Washington D.C.
- Kc, S., Wurzer, M., Springer, 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>
- National Research Council. (1991). *Improving information for social policy decisions—the uses of microsimulation modeling: (Vol. I)*. The National Academies Press, Washington, DC.
- Retherford, R. D., & Roy, T. K. (2003). *Factors affecting sex-selective abortion in India and 17 major states*. International Institute for Population Sciences and Honolulu: East-West Center, Mumbai, India
- Smith, S. K., Tayman, J., & Swanson, D. A. (2002). *State and local populations projections: Methodology and analysis*. Springer

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.