Introducing spatial microsimulation with R: a practical

Lovelace, Robin
r.lovelace@leeds.ac.uk

April 19, 2014

Contents

1 Foundations

1.1 Prerequisites for this practical

This practical is about spatial microsimulation in the software R. We suggest you install and take a look at this powerful program before getting started. We recommend using R within RStudio, which makes using R much easier. Instructions to install both R and RStudio can be found online: rstudio.com/ide/download/desktop.

The other prerequisite for the course is downloading the example data. These can be downloaded in a single zip file which can be found on the course's GitHub repository: github.com/Robinlovelace/smsim-course. Click on the "Download ZIP" button to the right of this page and extract the folder into your desktop or other suitable place on your computer.

Once the folder has been successfully extracted open it in your browser and take a look around. You will sub-folders entitled 'cakeMap', 'data', 'figures' and 'rcode'. The contents of each can be guessed from their titles and will become clearer throughout the course of this

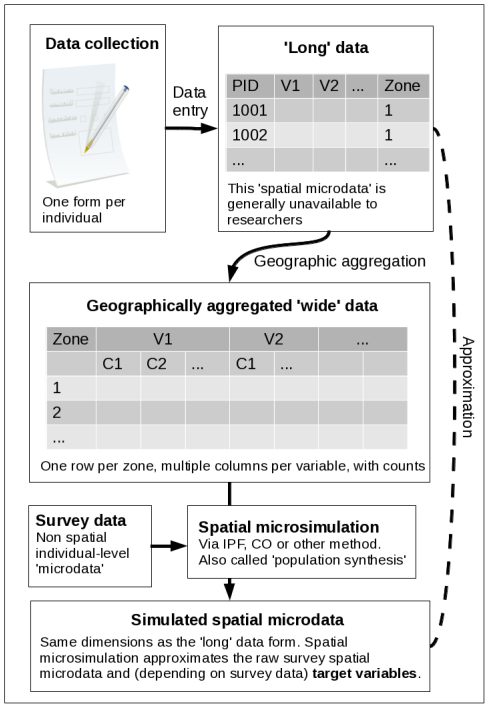

Figure 1: Schema of iterative proportional fitting (IPF) and combinatorial optimisation in the wider context of the availability of different data formats and spatial microsimulation.

tutorial. When you get to the sections that use R code, it is useful for R to operate from within the smsim-course-master folder. Probably the best way to set this up is to open the file 'smsim-course.Rproj' from within RStudio. Try this now and click on the 'Files' tab in the bottom right hand window of RStudio (add screenshot here!!!). Before using the power of R in RStudio it's worth understanding a bit about 'IPF', the algorithm we will use to generate the synthetic population or 'spatial microdata' (fig. 1).

1.2 Learning spatial microsimulation by doing it

As Kabacoff (2011, xxii) put it regarding R, "the best way to learn is to experiment" and the same applies to spatial microsimulation. We believe you will learn the technique/art best not by reading about it, but by doing it. Mistakes are inevitable in any challenging task and should not be discouraged. In fact it is by making blunders, identifying and then correcting them that many people learn best. Think of someone learning to skate: no one ever picks up a skateboard for the first time being able to 'surf the sidewalks'. It takes time, patience and plenty of falls before you master the art. The same applies to spatial microsimulation.

Spatial microsimulation works by taking *microdata* at the individual level and using aggregate-level constraints to allocate these individuals to zones. The two main methods are *deterministic reweighting* and *combinatorial optimisation*. This practical takes the former approach using a process called *iterative proportional fitting* (IPF). IPF is used to increase the weights of individuals who are representative of the target area and reduce the weights of individuals who are relatively rare (Lovelace and Ballas, 2013). The output is a *spatial microdataset*.

A schematic of the process is shown in fig. 1. Take a look at the image and think about the process. But we don't want to get bogged down in theory or applications in this course: we want to 'get our hands dirty'. Next we will do just that, by applying the process to some example data.

1.3 Some input data

Let's start with a very basic dataset with a handful of individuals who will be assigned to zones via IPF. To aid understanding, we will do this first by hand to understand the process before automating it on the computer. Table 1 shows 5 individuals, who are defined by two constraint variables: age and sex. Each has two categories. Table 2 contains data for zones for a hypothetical area. Table 3 illustrates this table in a different form, which shows our ignorance of interaction between age and sex.

Table 1: A hypothetical input microdata set (the original weights set to one). The bold value is used subsequently for illustrative purposes.

Individual	Sex	Age	Weight
1	Male	59	1
2	Male	54	1
3	Male	35	**1**
4	Female	73	1
5	Female	49	1

Table 2: Small area constraints (s).

Constraint ⇒	i		j	
Category ⇒	i_1	i_2	j_1	j_2
Area ⇓	Under-50	Over-50	Male	Female
1	8	4	6	6

Table 3: Small area constraints expressed as marginal totals, and the cell values to be estimated.

Marginal totals	j			
	Age/sex	Male	Female	T
i	Under-50	**?**	?	8
	Over-50	?	?	4
	T	6	6	12

Table 4 presents the hypothetical microdata in aggregated form, that can be compared directly to the aggregate data presented in Table 3.

1.4 The IPF equation

Using these tables we readjust the weights of the individuals so that their sum equals the total population of the area presented in Table 3. The weights are multiplied by the aggregate level values from Table 2 and divided by the respective marginal total of the microdata (see 4). This is done one constraint at a time, as described by eq. (1) for constraint i (age in this case):

$$w(n+1)_{ij} = \frac{w(n)_{ij} \times sT_i}{mT(n)_i} \tag{1}$$

where $w(n+1)_{ij}$ is the new weight for individuals with characteristics i (age, in this case), and j (sex), $w(n)_{ij}$ is the original weight for individuals with these characteristics, sT_i is element marginal total of the small area constraint, s (Table 2) and $mT(n)_i$ is the marginal total of

Table 4: The aggregated results of the weighted microdata set ($m(1)$). Note, these values depend on the weights allocated in Table 1 and therefore change after each iteration

Marginal totals		j		
	Age/sex	Male	Female	T
i	Under-50	**1**	1	2
	Over-50	2	1	3
	T	3	2	5

category j of the aggregated results of the weighted microdata, m (Table 4). n represents the iteration number.

Do not worry too much about understanding the above equation for now. More important is implementing it. Follow the emboldened values in the tables to see how the new weight of individual 3 is calculated for the sex constraint. Table 5 illustrates the weights that result. Notice that the sum of the weights is equal to the total population, from the constraint variables.

Table 5: Reweighting the hypothetical microdataset in order to fit Table 2.

Individual	Sex	age-group	Weight	New weight, w(2)
1	Male	Over-50	1	$1 \times 4/3 = \frac{4}{3}$
2	Male	Over-50	1	$1 \times 4/3 = \frac{4}{3}$
3	Male	Under-50	1	$\mathbf{1} \times \mathbf{8}/\mathbf{2} = 4$
4	Female	Over-50	1	$1 \times 4/3 = \frac{4}{3}$
5	Female	Under-50	1	$1 \times 8/2 = 4$

After the individual level data have been re-aggregated (table 6), the next stage is to repeat eq. (1) for the age constraint to generate a third set of weights, by replacing the i in sT_i and $mT(n)_i$ with j and incrementing the value of n:

$$w(3)_{ij} = \frac{w(2)_{ij} \times sT_j}{mT(2)_j} \tag{2}$$

1.5 Test your understanding

To test your understanding of IPF, apply eq. (2) to the information above and that presented in table 6. This should result in the following vector of new weights, for individuals 1 to 5. Calculate the correct values and pencil them in in place of the question marks. One 'sanity' check of your method here is whether the sum of these weights is still equal to twelve.

$$w(3) = (\frac{6}{5}, \frac{?}{?}, \frac{18}{5}, \frac{?}{?}, \frac{9}{2}) \tag{3}$$

Notice also that after each iteration the fit between the marginal totals of m and s improves. The total absolute error (TAE) improves between $m(1)$ to $m(2)$, falling from 14 to 6 in table 4 and table 6 above. TAE for $m(3)$ (not shown, but calculated by aggregating $w(3)$) improves even more, to 1.3. This number would eventually converge to 0 through subsequent iterations, a defining feature of IPF.

1.6 IPF in a spreadsheet

Computer code is absent from most people's daily lives so the R code we use to automate IPF will likely seem like a foreign language to many. Spreadsheet programs like Microsoft Excel and LibrOffice Calc are comparatively well known, though. To ensure smooth transition between the IPF process described in mathematics above and its implementation in R this section provides

Table 6: The aggregated results of the weighted microdata set after constraining for age ($m(2)$).

Marginal totals	i			
	Age/sex	Male	Female	T
j	Under-50	4	4	8
	Over-50	$\frac{8}{3}$	$\frac{4}{3}$	4
	T	$6\frac{2}{3}$	$5\frac{1}{3}$	12

an intermediary stage: the speed of computation and the visual support of a graphical user interface (GUI).

1.7 Implementing IPF in R

So far we have implemented IPF by hand and in widely known spreadsheet programs. This section explains how the IPF *algorithm* described above is implemented in R, using the same input data.[1]

Loading in the data

Usually the input data for spatial microsimulation are loaded from .csv files, one for each constraint and one for the input microdata. These are read-in with the command `read.csv`. For the purposes of understanding how the model works, the dataset is read line by line, following the example above. The following code creates example datasets, based on the same hypothetical survey of 5 individuals described above, and 5 small areas. The spatial microsimulation model will select individuals based on age and sex and mode of transport (mode of transport is also used on the larger online example described in footnote 1). For consistency with the (larger) model used for the paper, the individual level data will be referred to as USd (Understanding Society dataset) and the geographic data as all.msim (for all constraint variables). The code to read-in the individual level data are presented in code sample 1. When called, the data are then displayed as a table (see listing 2). The same procedure applies to the geographical data

Listing 1: Manual input of individual level data in R

```
# Read in the data in long form (normaly read.table() used)
c.names <- c("id", "age", "sex")
USd <- c(       1, 59, "m",
                2, 54, "m",
                3, 35, "m",
                4, 73, "f",
                5, 49, "f")
USd <- matrix(USd, nrow = 5, byrow = T) # Long data into matrix
USd <- data.frame(USd) # Convert this into a dataframe
names(USd) <- c.names # Add correct column names
USd$age <- as.numeric(levels(USd$age)[USd$age]) # Age is a numeric
```

(listing 3).

IPF relies on the assumption that all constraint variables will contain the same number of people. This is logical (how can there be more people classified by age than by sex?) but can cause problems for constraint variables that use only a subset of the total population, such as those who responded to questions on travel to work. To overcome this problem, it is possible to normalise the constraint variables, setting the total for each to the one that has the most

[1] A fuller tutorial is available from Rpubs, a site dedicated to publishing R analyses that are reproducible. It uses the RMarkdown mark-up language, which enables R code to be run and presented within documents. See http://rpubs.com/RobinLovelace/5089 .

Listing 2: Output of the USd data frame

```
USd # Show the data frame in R
##   id age sex
## 1  1  59   m
## 2  2  54   m
## 3  3  35   m
## 4  4  73   f
## 5  5  49   f
```

Listing 3: Geographic data input

```
 category.labels <- c("16-49", "50+" # Age constraint
             ,"m", "f" # Sex constraint
             # more constraints could go here
             )
all.msim <- c(  8, 4,    6, 6,   # Original aggregate data
                2, 8,    4, 6,   # Elderly
                7, 4,    3, 8,   # Female dominated
                5, 4,    7, 2,   # Male dominated
                7, 3,    6, 4    # Young
                )
all.msim <- matrix(all.msim, nrow = 5, byrow = T)
all.msim <- data.frame(all.msim) # Convert to dataframe
names(all.msim) <- category.labels # Add correct column names
```

reliable total population. This worked example simply checks whether or not they are (listing 4).

Listing 4: R code to check the constrain populations match

```
 # Check totals for each constraint match
rowSums(all.msim[,1:2]) # Age constraint
## [1] 12 10 11  9 10
rowSums(all.msim[,3:4]) # Sex constraint
## [1] 12 10 11  9 10

rowSums(all.msim[,1:2]) == rowSums(all.msim[,3:4])
## [1] TRUE TRUE TRUE TRUE TRUE
```

Reweighting the survey dataset

Iterative proportional fitting determines the weight allocated to each individual for each zone to best match the geographically aggregated data. A weight matrix is therefore created, with rows corresponding to individuals and columns to zones, as described in section 1.3.

There is great scope for taking the analysis further: some further tests and plots are presented on the on-line versions of this section. The simplest case is contained in Rpubs document 6193 and a more complex case (with three constraints) can be found in Rpubs document 5089. For now, however, we progress to a more complex example, CakeMap.

2 CakeMap: A worked example

2.1 Loading the input data

2.2 Preparing the input data for IPF

2.3 Performing IPF on the CakeMap data

3 Applying the technique in the real world

3.1 Model checking

To make an analogy with food safety standards, openness about mistakes is conducive to high standards (Powell et al., 2011). Transparency in model verification is desirable for similar reasons. The two main strategies are 1) comparing the model results with knowledge of how it *should* perform *a-priori* (model checking) and 2) comparison between the model results and empirical data (validation).

A proven method of checking that data analysis and processing is working is wide ranging and continual visual exploration of its output (Janert, 2010).

3.2 Model validation

Beyond 'typos' or simple conceptual errors in model code, more fundamental questions should be asked of spatial microsimulation models. The validity of the assumptions on which they are built, and the confidence one should have in the results are important.

3.3 Integerisation

3.4 Estimating home locations

4 Glossary

- **Algorithm**: a series of computer commands which are executed in a well defined order. Algorithms process input data and produce an output.
- **Iteration**: one instance of a process that is repeated many times until a predefined end point, often within an *algorithm*.
- **Iterative proportional fitting** (IPF): an iterative process implemented in mathematics and algorithms to find the maximum likelihood of cells that are constrained by multiple sets of marginal totals. To make this abstract definition even more confusing, there are multiple terms which refer to the process, including 'biproportional fitting' and 'matrix raking'. In plain English, IPF in the context of spatial microsimulation can be defined as *a statistical technique for allocating weights to individuals depending on how representative they are of different zones*. IPF is a type of deterministic reweighting, meaning that random numbers are not needed to generate the result and that the output weights are real (not integer) numbers.
- **Iteration**

5 References

Janert, P.K., 2010. Data analysis with open source tools. O'Reilly Media.

Kabacoff, R., 2011. R in Action. Manning Publications Co.

Lovelace, R., Ballas, D., 2013. Truncate, replicate, sample: A method for creating integer weights for spatial microsimulation. Computers, Environment and Urban Systems 41, 1–11.

Powell, D.a., Jacob, C.J., Chapman, B.J., 2011. Enhancing food safety culture to reduce rates of foodborne illness. Food Control 22, 817–822.