Gaëtan de Menten, Gijs Dekkers, Geert Bryon, Philippe Liégeois and Cathal O'Donoghue (2014)

LIAM2: a New Open Source Development Tool for Discrete-Time Dynamic Microsimulation Models

Journal of Artificial Societies and Social Simulation 17 (3) 9
<http://jasss.soc.surrey.ac.uk/17/3/9.html>

Received: 23-Jan-2014 Accepted: 27-Apr-2014 Published: 30-Jun-2014

Abstract

Most existing microsimulation models have been developed by separate (teams of) researchers. The drawback of each team working on its own is that they have to put a lot of time and effort in the customary development of fairly general simulation tools. Hence, economies of scale cannot be exploited, which makes microsimulation models even more expensive than strictly necessary. The objective of this paper is to present LIAM2, a free and open source modelling framework designed for the development of discrete-time dynamic models. It is meant to make microsimulation models much easier to develop. This paper makes a comparison with other simulation frameworks, presents a minimal LIAM2 model and discusses its performance in terms of data capacity and simulation speed.

Keywords:
Microsimulation, Software, LIAM2

Introduction

1.1 Microsimulation is a modelling technique that operates at the level of individual units such as persons, households, vehicles or firms. Each unit has a set of associated attributes – e.g. a person has an associated age, sex, and employment status. In the case of discrete-time dynamic models, a set of rules (intended to represent individual preferences) are applied to these units at each time step, leading to simulated changes in state and possibly behaviour. The aim of such simulations is to give insight about both the overall aggregate change of some characteristics and, most importantly, the way these changes are distributed in the population that is being modelled. The methodology is often used to design and evaluate public policies that are affected by earlier events and choices as is the case for pensions or education.

1.2 The technology to run analyses like this in an operational capacity exist since the 1970s (see Orcutt et al. 1976) but progress in the field has been relatively slow. Reasons for this include the lack of knowledge transfer and shared model frameworks (Li & O'Donoghue 2013).

1.3 With the static model EUROMOD (Lelkes & Sutherland 2009), the dynamic model MIDAS (Dekkers et al. 2010; Dekkers et al. 2013) and the spatial model SMILE (O'Donoghue et al. 2012) as the most notable exceptions, most existing models have been developed by separate (teams of) researchers. In many cases, those technical parts that are necessary for any model but that are not part of a model as such (for example the routines to read and write data, to handle alignment, etc.) are (re-)developed in an ad hoc manner by each team separately. These are all general tools, but as any team that wishes to develop a model needs to have them, they have to put a lot of time and effort in the development of these tools. Hence, economies of scale cannot be exploited, which makes microsimulation models even more expensive than strictly necessary.

1.4 Furthermore, in many cases, the simulation tools are also created by the modellers themselves. Since these modellers are not necessarily skilled with programming, the result is not always very efficient in terms of both speed and memory usage.

1.5 A solution to these problems lies in the use of simulation modelling packages which include those generic functionalities, such as UMDBS (Sauerbier 2002), Modgen[1] (Spielauer 2006), LIAM (O'Donoghue et al. 2009), GENESIS (Strassburg & Tracey 2011)

and JAMSIM (Mannion et al. 2012).

1.6 The objective of this paper is to present LIAM2, a new open source microsimulation modelling package designed for the development of discrete-time dynamic models, while also allowing for static modelling.

1.7 LIAM2 can be downloaded from http://liam2.plan.be. The bundles contain either a 32 or 64-bits version of the executable, a text editor pre-configured to work with LIAM2, extensive documentation (Bryon et al. 2014), as well as a number of demonstration models using a synthetic dataset. The source code and issue tracker (to report bugs or propose enhancements) are hosted on GitHub at https://github.com/liam2/liam2. Discussion groups for users and developers can be found respectively at: http://groups.google.com/group/liam2-users and http://groups.google.com/group/liam2-dev.

1.8 In this paper we shall start by providing some context for the development of LIAM2 in relation to other existing modelling frameworks. The features of the tool and the structure of a model will be presented next. Thereafter the performance of LIAM2 will be discussed and conclusions will be drawn.

Context and comparison to other tools

2.1 In this section we provide some context for the development of the LIAM2 framework and provide a justification for the directions taken. LIAM2 is the result of an ongoing collaboration between individual researchers from various institutions. The software package is being developed and tested at the Federal Planning Bureau (FPB). Additional testing is done by the Luxembourg Team (CEPS/INSTEAD and IGSS). Finally, Cathal O'Donoghue shared the source code of the first version of LIAM, which the FPB used in the AIM project (Dekkers et al. 2010).

2.2 The basic idea behind LIAM2 is to separate 'modellers' from 'programmers', where the former are responsible for the model and the latter take up the development of the framework. Modellers should not have to care about technical issues, such as data reading and writing, parsing expressions, evaluating them, or using multiple processors. Nothing prevents motivated modellers to dive into the low level code to study, modify, or add a particular feature, but the most important point is that they usually do not have to.

2.3 The framework builds upon lessons learned from earlier frameworks such as LIAM and the operational implementations in MIDAS and LIAM-Ireland (Baroni & O'Donoghue 2009). The goal we set ourselves was to provide a development environment that:

- has a low barrier to entry, in terms of both price (it should be free) and knowledge (it should be accessible to modellers with little programming background).
- is generic enough to be used to develop a wide range of microsimulation models. It should allow for the simulation of whatever kind of objects the modeller chooses. To limit the scope of the project, we concentrated ourselves on discrete-time dynamic[2] models with a cross-sectional time step[3] .
- is efficient in terms of modeller time (enables quick model development). To achieve that we strive for a syntax that is simple and highly expressive yet very readable.
- includes out of the box a large enough body of microsimulation-specific functions so that modellers do not have to implement these themselves (including several alignment algorithms, matching, etc.).
- is efficient to simulate. It should use advanced methods for data-handling and numerical computation, thereby allowing models to run on large datasets at high speed.
- is open source, so that developers worldwide can add to the toolbox and share current and future methodological developments. LIAM2 source code is licensed under the GNU Public License version 3 (Free Software Foundation 2007).

Our hope is also to stimulate collaboration and the exchange of tacit information between development teams through the use of a common development approach.

Comparative analysis

2.4 There is unfortunately not much literature on comparisons between microsimulation modelling tools, with the papers by O'Donoghue (2001) and Li and O'Donoghue (2013) the notable exceptions. This section compares LIAM2 with LIAM, Modgen, JAMSIM and GENESIS. UMDBS is not included in this discussion, because it is to our knowledge no longer in use.

2.5 LIAM2 has been developed based on experience gained through the use of LIAM in the development of MIDAS (Dekkers et al. 2013). Thus LIAM2 is the "spiritual successor" of LIAM. The two packages share that they are freely available. But LIAM2 is more general, more convenient, considerably faster, and allows for much larger datasets. Furthermore, where every part of a model (definition of entities, fields, their interactions and all processing rules) required separate ascii-files in LIAM, this can now been grouped in one file if desired. Finally, LIAM2 is broader than LIAM in terms of the tools and possibilities it offers the model developer.

2.6 Modgen is a software tool developed and maintained by Statistics Canada that is freely available but not open source. It is essentially a C++ library that allows modellers to include common microsimulation procedures and methodologies. So model

development in Modgen requires programming in C++ (and buying and using Visual Studio). LIAM2 is, by contrast, a development environment that has a syntax that is meant to be simple and readable yet very expressive. Even though LIAM2 itself is being developed mainly in Python (Python Programming Language), this has little consequences for the end user[4]. Another difference is that Modgen provides a graphical user interface, whereas LIAM2 does not. For model developers, we believe it does not make much of a difference in practice, but it allows third parties to use the model. Finally, Modgen is more general than LIAM2, as it appears to support, for example, both discrete time and continuous-time models whereas LIAM2 focuses on discrete-time models with cross-sectional ageing. For this particular class of models though, and especially when alignment is involved, we hope LIAM2 provides a better experience to the modeller.

2.7 JAMSIM is conceptually equivalent to Modgen, but it is developed in Java and R and it is open source. It is "less a framework and more a loose coupling of a set of open source packages to provide a base set of functionalities for microsimulation" (Mannion et al. 2012, 5.1). Except for being open source, it shares some of the most important advantages and drawbacks of Modgen: it is fairly general and has a nice graphical user interface but the model needs to be developed in a low level language (Java, even though reporting is done with R).

2.8 GENESIS is a generic dynamic simulation model developed and used within the UK Department of Work and Pensions. It is composed of a generic Model Engine programmed in SAS and parameter spreadsheets held in Excel and has been used to develop a number of models within this department including Pensim2 and benefit forecast models. Compared to LIAM2, GENESIS is not available to external users and it requires SAS, which is not free. Furthermore, it is more specific than LIAM2, in that it is essentially a generic model relying on existing SAS utilities. LIAM2, by contrast, is a development environment, and not a model; it is thus independent of the data available and does not require staying within the boundaries set by one (albeit generic) model or SAS utilities. Finally, preliminary tests have shown that LIAM2 seems faster in simulation than GENESIS.

Characteristics and properties of LIAM2

3.1 Motivated by the need for improved performance, greater flexibility and the wish for more collaboration between teams of researchers, we now describe some of the technical choices and properties of LIAM2.

Technical choices

3.2 To foster collaboration, we wanted LIAM2 and all its underlying tools to be open source. This, most importantly, includes the development language and the on-disk file format handling routines.

3.3 LIAM2 is developed primarily in Python, a powerful general purpose programming language that is freely available under an open source licence. It has a large body of standard libraries and allows for C or C++ extension modules. Python (and by consequence LIAM2) runs on all major operating systems and has 32-bit and 64-bit versions available. Even though LIAM2 itself is written in Python, modelling in LIAM2 does not require any knowledge or use of Python[5].

3.4 The starting point of any microsimulation model is a dataset containing a large number of individual objects. In many cases, it is a survey or administrative dataset representing actual objects at a certain point in time, including the variety of relations and links between these objects. Being able to efficiently handle potentially large and complex datasets was a first requirement for our modelling framework. LIAM2 internally reads and stores data in Hierarchical Data Format 5 (HDF5; see HDF Group). This is a set of libraries designed to store and organize large amounts of numerical data[6]. It is freely available under an open-source license. Users will usually provide their data in the form of separate CSV files, and LIAM2 can convert them to a single HDF5 file, which can thereafter be used as a starting dataset for several runs of a simulation or even different model variants. Simulation results are also stored in HDF5 format for later processing and optionally as CSV (cf. infra).

Features

3.5 Dynamic microsimulation models require a number of components including

- Dataset and parameter input
- Database handling tools
- Generic stochastic simulations
- Calibration or Alignment Tools
- Non-stochastic calculations
- Life-cycle routines: creation or removal of objects
- Output and reporting routines
- Other specific routines such as matching routines.

3.6 LIAM2 provides many different generic functions out of the box. These include: mathematical functions, conditional functions, aggregate functions (including the mean, median, percentiles and Gini), temporal functions (lags, durations), functions for stochastic changes (logits, simple univariate Monte-Carlo simulations), life-cycle functions (birth, death, clone), and a matching function (a.k.a. 'marriage market').

3.7 LIAM2 provides extensive functionalities to calibrate to exogenous information of any number of dimensions, most notably alignment by sorting (Li & O'Donoghue 2014) and the pageant alignment method (Chénard 2000). A particularly interesting feature pertaining to alignment is that the argument used to rank individuals for deciding who experiences an event can be any expression. In the case it consists of one or more logits, the result reflects the a priori individual probability of the event occurring.

3.8 By manipulating the ranking expression, the modeller can impose an a priori high or low probability of the event happening for specific individuals, and can therefore affect the overall characteristics of the group of individuals that experience the event[7]. This a priori order might not under all circumstances result in the simulated event to happen or not, since a) the modeller can choose to add a stochastic element to the score, and b) some individuals with a low (high) risk may still (not) be selected, if the number of individuals with the high a priori risk is lower (higher) than the number of transitions required by the alignment process.

3.9 By contrast to these score/probability manipulations where the alignment targets take precedence over individual probabilities, the alignment procedure itself also introduces the concept of (optional) take and leave filters. These force inclusion or exclusion of individuals with specified characteristics in the selection of the event, even if that means deviating from the alignment target. The individuals with variables specified in the "take" command will a priori be selected for the event. Suppose that for a certain event, there are 20 individuals in a particular category, that the alignment data specifies that 50% individuals should experience the event, and that there are 3 individuals out of those 20 who meet the conditions specified in the take. We thus need to select 10 individuals. The 3 individuals satisfying the take filter will be selected a priori and the alignment process will select the remaining 7 from the rest of the sample. The "leave" command works the other way around: those who match the conditions are a priori excluded from the event happening.

3.10 Another important functionality of LIAM2 is that it respects the existing values of a variable in a certain period. Thus, if a value for an endogenous variable is available for one or more periods of time, LIAM2 will not overwrite this value but use it in the remainder of the model instead. This feature has been introduced to allow for retrospective modelling. For example, in the Belgian version of the MIDAS model, a microsimulation model for social security (henceforth MIDAS_BE), a retrospective simulation is used to compute pension claims[8]. Like many Western-European countries, Belgium has a Bismarckian-style public pension system where past labour market performance and earnings determine the pension benefit at retirement. Thus several retrospective variables are needed to simulate pension benefits. Pension claims are thus computed for every retrospective year, based on the labour market status of active individuals for these years. At the end of the retrospective simulation, the retrospective information is summarized into a few variables. Those variables are then merged into the starting dataset for the prospective model.

3.11 Another application of this feature is that the output of one prospective run of the model can be used as input for another prospective run. For example, one could make one prospective run of the demographic module of a model, and use it as input for a separate run without the demographic module. This saves some running time but, more importantly, it ensures that the population is identical for various simulation runs, which allows for a more straightforward identification of individual winners and losers of a specific policy measure. Furthermore, some models can be used for various applications and these can potentially need very different output. Since producing output can be time-consuming, it may make sense to create several specific "output models" which can be simulated independently from each other and from the actual model to only produce what one actually needs.

3.12 In many models, the starting dataset needs to be modified to be consistent with the model. In LIAM2, there is an initialization section which can contain any kind of process to run prior to the actual time-dependent simulation. This allows not only to use LIAM2 for static simulation, but also for a straightforward exchange of modules between the static and dynamic model.

3.13 Before ending this section, note that LIAM2 does not provide any functionality for estimation nor solving optimization problems (for example when individuals decision is influenced by simultaneous choices made by other individuals)[9].

The building blocks of a LIAM2 model

3.14 The purpose of any microsimulation model is to change the characteristics of objects through processes of various types, possibly taking into account the relations between these objects.

3.15 In order to do so, a typical model in LIAM2 uses the following concepts: entities, fields, processes, links, globals and macro's.

- An *entity* represents a class of objects in a model. One could for example have a model with a "person" entity and a "household" entity. LIAM2 is generic in that regard, as it can handle any kind of object.
- Each entity has a set of attributes called *fields*. There are three types of fields: bool (booleans), int (integers) and float (floating-point numbers). Note that a field may, but need not, be included in the starting dataset.
- The content of a field can be changed by one or several *processes*. These are the main part of the model and describe how each individual evolves over time.
- A *link* is a relation between entities. For example, mothers are linked to their children, spouses are linked together, and persons are linked to households, and vice versa.
- A *global* is a parameter that is not related to a specific entity. It can be used to access any numerical information, such as time series, tables of probabilities for events, ... For example, the different amounts and thresholds that describe a social security system can be specified through globals.
- Finally, a *macro* is a construct which allows to give a name to an arbitrary expression (a value, or a piece of code), so

that it can be reused easily. Macros can make models considerably more readable and easier to develop and maintain.

3.16 The text box presents a very rudimentary setup of a working model (other than data importing functionality).

```
globals:
    periodic:
        - RETAGE: int

entities:
    person:
        fields:
            - age: int
            - years_in_ret: {type: int, initialdata: false}

        processes:
            age: age + 1
            years_in_ret: if(age >= RETAGE, age - RETAGE, -1)

simulation:
    processes:
        - person: [age, years_in_ret]

    input:
        file: base.h5
    output:
        file: simulation.h5
    start_period: 2013
    periods: 10
```

3.17 A typical model file consists of three main blocks: *globals*, *entities* and *simulation*. The first block lists the globals or parameters to be used in the model. Our example model contains only one parameter: RETAGE (the mandatory retirement age). The values of this parameter for all simulation years need to be included in the starting dataset.

3.18 The second block is the entities block. It describes for each entity used in the model (persons, households, firms, etc.) its fields, links, macros and processes. In this example, we have only a single "person" entity. The "fields" block describes its fields: 'age' and 'years_in_ret' (the number of years that somebody has been in retirement). Note that only 'age' is present in the input dataset, and 'years_in_ret' is not. This variable is thus missing until it receives a value from a process. The next sub-block is called "processes" and it lists the processes that pertain to the entity. It is usually the largest block for each entity. In this case, there are only two processes, of which the name coincides with the dependent variable. The first process increases age by 1 for every individual (for each period). The second process sets the field 'years_in_ret' as the difference between the age of the individual and the mandatory retirement age. This process also illustrates conditional statements, because 'years_in_ret' has a value of -1 if the individual is younger than the mandatory retirement age. Note, finally, that the order in which processes are defined in this entities block is not relevant.

3.19 The third block is the *simulation* block. The main part of this block defines which processes to run for the simulation and in what order. This block also contains the paths to the input- and output-datasets, the starting period and the number of periods (years in this case) to simulate. This model starts in 2013 and simulates 10 years (up to 2022). This model (and its starting dataset) contains only a person entity. The input dataset, named "base.h5" should contain the identification number (id) of each person, his or her age and the global RETAGE. Output is written to "simulation.h5" and it will contain the 'id', 'age' and 'years_in_ret' of each person for each of the 10 periods simulated.

3.20 The above example is of course a gross simplification designed to illustrate the setup of a model in LIAM2. It contains just one entity, no links, no procedures, no macros, no initialization section and just two deterministic processes. More complex and illustrative examples can be found on the LIAM2 website.

Output and interaction

3.21 LIAM2 writes the simulation results to a HDF5 output file. It can be analysed with a HDF5 browser or it can be transformed to a format readable by more traditional data analysis packages. However, output from a dynamic microsimulation model often is sizable (for example, a typical output dataset of the MIDAS_BE model has a size of roughly 120GB). As a result, it may be

inconvenient to transfer the data to other packages. Instead, one may choose to do part of the result analysis within LIAM2 itself.

3.22 There are a number of features within the framework that can help with that, including an interactive mode and the capacity to output to CSV.

3.23 In interactive mode, one can analyze the input and simulation data using any function available for model development. This, combined with the possibility to interrupt the simulation by placing a breakpoint anywhere in the model, opens extensive possibilities for debugging and data analysis. When reaching such a breakpoint or when reaching the end of the model, LIAM2 will go in interactive mode, thereby allowing the user to analyse the data by inputting any LIAM2 command.

3.24 LIAM2 can write the output of any command to CSV files. This allows models to write, for each period, any selection of variables describing certain individuals or aggregates of those variables. Results can also be presented and saved as graphs of various types, including (stacked) bar charts, line charts and pie charts. All in all, this allows for tailor-made outputs that summarize the simulation data, and that can be used immediately to assess the simulation results or create tables and figures. It is therefore possible to do a large part of the results analysis within the model itself.

Performance

4.1 An assessment of the performance of any modelling platform needs to cover two different aspects: the size of the dataset the toolbox can handle and the speed of simulation. But for frameworks like LIAM2, it is a tricky exercise because the performance entirely depends on what is done with it. A natural reaction is to try to divide the performance metrics by the size of the model. But there is no standard metric for the latter. The number of lines is a flawed metric because some functions run orders of magnitude faster than others and the expressiveness of the language offered to modellers can vary considerably between tools. The only reliable method to compare LIAM2 performance to other similar tools would be to have a few different "representative" models run with the different tools (comparing using one model would not be enough since one model can run fastest using a particular tool and another model using another tool).

4.2 Given the extensive investment that would be required to create even one non trivial model and translate it to the different tools (assuming it can be translated at all), we have not done that exercise and can only discuss the performance we got for the MIDAS_BE model with the current version of the LIAM2 framework.

4.3 The version of MIDAS_BE whose results are presented in the table below includes 143 parameters, 122 (permanent) variables, 21 aligned processes (using 37 alignment files) and 12 CSV output files. It is being developed and tested using a dataset of 300,000 persons. The actual simulation runs are based on an expanded version of this dataset, being 2.2 million persons or one-fifth of the Belgian population.

Table 1: Performance of MIDAS_BE using LIAM2 - simulation run from 2002 to 2060.

Starting dataset of 300K persons	Starting dataset of 2,200K persons
Total runtime:	Total runtime:
28 minutes 18 seconds	2 hours 54 minutes 11 seconds
18582 individuals/s/period on average	19317 individuals/s/period on average
top 5 processes:	top 5 processes:
1. output_process: 13m 26s (46%)	1. output_process: 1hr 22m 14s (47%)
2. family_allowances: 2m 2s (7%)	2. family_allowances: 13m 4s (7%)
3. matching_process: 1m 31s (5%)	3. matching_process: 12m 17s (7%)
4. welfare: 1m 28s (5%)	4. welfare: 9m 39s (6%)
5. inwork_process_working_t-1: 50s (3%)	5. inwork_process_working_t-1: 4m 41s (3%)
Peak memory use: 832MB	Peak memory use: 4.38GB

4.4 Table 1 presents the performance of MIDAS_BE using the most recent version of LIAM2 as of this article (version 0.8.1 of March 19th, 2014, 64-bits executable) on a computer with a core i5 CPU. It lists total run times, "individuals per second per period", peak memory usage and run times of the 5 processes in the model taking up most of the time, both in time units and proportionally to the total run time.

4.5 Reading this table, we can make a few observations. The most obvious one is that a simulation of a non trivial model for 59 periods on a 2.2 million individuals' dataset can be completed in a time frame that is still practical (roughly 3 hours) on a desktop computer. Based on the "individuals per second per period" indicator (which corrects for the number of periods, the sample size and its change over time), we see that run times scale almost linearly with the sample size. We also see that it scales better than linearly regarding the memory usage. Also, changing the size of the starting dataset does not induce major changes to the proportional simulation times.

4.6 The "output process" computes many ad-hoc aggregates to assess the simulation results and writes them to CSV files. Interestingly, it is by far the most time-consuming process; in both cases more than 45% of overall run time. It is thus a good example of a module which would benefit from being simulated independently, and potentially split into smaller parts, as discussed previously.

4.7 Concerning the data capacity, LIAM2 has no inherent limit on the number of individuals it can handle but it is currently limited by the amount of memory (RAM) in the computer running the simulation. The latest version (0.8.1) has a peak total memory usage of (very) approximately 12 bytes per variable-individual[10] for large datasets. This means that assuming, for example, a computer with 8 GB of memory and a model using 100 variables, LIAM2 (64-bits) can simulate up to approximately 7 million individuals. This should also be improved in future versions of LIAM2.

4.8 Finally, one should note that even though a model developed using LIAM2 runs reasonably fast and scales well with the size of the dataset, it is currently several times slower than what could be theoretically achieved by a good programmer "hard-coding" the model in the simulation software. But a model developed that way would take much longer to create and would be much harder to change. Also, one should bear in mind that LIAM2 is still being actively developed and there is still much room for improvements (both regarding memory usage and speed of simulation) in future versions.

Conclusions

5.1 Microsimulation models are currently more difficult, more costly and slower to develop than strictly necessary in part because of the lack of easy to use frameworks to develop such models and the little collaboration between development teams. LIAM2 was developed with the ambition to address these issues. It attempts to free model developers from the low-level details that are inherent to most microsimulation techniques, while offering the possibility to develop models in easily readable code.

5.2 This paper compares LIAM2 to other microsimulation tools and concludes that it is the spiritual successor of LIAM and seems complementary to ModGen and JAMSIM. It then discusses the various features of LIAM2, including its extensive possibilities for alignment by sorting, which make it possible to have microsimulation models replicate external information. The paper also presents the simulation speed and capacity of LIAM2 to handle large datasets.

5.3 Since LIAM2 is open source, we hope that it will benefit from the methodological and applied work of others in the field of microsimulation, and that it will be a vehicle for further cooperation between teams.

Acknowledgements

LIAM2 is the result of the MiDaL project (December 2009 – November 2011), "Towards the development of a dynamic Microsimulation toolbox "LIAM-II "and the complementary implementation of administrative Data needed for dynamic microsimulation of pensions in Luxembourg", in which the Federal Planning Bureau collaborated with CEPS/INSTEAD. The authors gratefully acknowledge the support of the European Community Programme for Employment and Social Solidarity - PROGRESS (2007–2013), under the Grant VS/2009/0569, as well as the General Inspectorate for Social Security (IGSS) of Luxembourg. The authors also wish to thank Raphaël Desmet and Raymond Wagener for their contributions to the project and comments on this paper.

Notes

[1] http://www.statcan.gc.ca/microsimulation/modgen/modgen-eng.htm

[2] Static models are possible as well

[3] All individuals are simulated at the same time for one period, then for the next period, etc.

[4] The syntax is in fact very close to Python but we diverged from it for the few cases where we thought it was possible to make it easier or more practical for modellers

[5] Although programming in Python is obviously needed if one wants to change the source code of LIAM2 in order to add to or change the functionalities that it offers to modellers.

[6] To illustrate how large this can be, NASA uses HDF as the prescribed format for storing data from the Earth Observing System (EOS, see Schmidt 2000)

[7] For example, suppose that the event is "becoming unemployed given that one currently works" and suppose that the modeller gives women a higher a priori probability than men. Then proportionally more women than men will lose their job and women will over time become the larger group among the unemployed.

[8] MIDAS_BE (an acronym for 'MIcrosimulation for the Development of Adequacy and Sustainability') includes the detailed simulation of demographic transitions and schooling, earnings, labour market transitions, social security benefits, means-tested minimum benefits, and a gross-net trajectory.

[9] Although, in some simple cases, this can be approximated by a multiple step process: first simulate group A, then group B depending on A, then possibly group A again, ...

[10] For this purpose the number of variables is the number of fields across all entities plus the number of local variables in the procedure with the most local variables.

References

BARONI, E. & O'Donoghue, C. (2009). Poverty impact of state welfare pension reform on the elderly: An analysis of the reform proposals in the 2007 Irish Green Paper. http://www.combatpoverty.ie/publications/workingpapers/2009-09_WP_PovertyImpactOfStatePensionReformOnTheElderly.pdf Archived at: http://www.webcitation.org/6P1giXskU

BRYON, G., Dekkers, G. & de Menten, G. (2014), *LIAM2 User Guide, Release 0.8.1* http://liam2.plan.be/download/LIAM2UserGuide-0.8.1.pdf Archived at: http://www.webcitation.org/6P4Oern1h The latest version can be found in various formats at: http://liam2.plan.be/pages/documentation.html

CHÉNARD, D. (2000). Individual Alignment and Group Processing: An Application To Migration Processes In DYNACAN. In L. Mitton, H. Sutherland, & M. Weeks (Eds.), *Microsimulation Modelling for Policy Analysis: Challenges and Innovations* (pp. 238–247). Cambridge: Cambridge University Press.

DEKKERS, G., Buslei, H., Cozzolino, M., Desmet, R., Geyer, J., Hofmann, D., Raitano, M., Steiner, V., Tanda, P., Tedeschi, S. & Verschueren, F. (2010). The flip side of the coin: The consequences of the European budgetary projections on the adequacy of social security pensions. *European Journal of Social Security*, *12*, 94-120.

DEKKERS, G., Desmet, R., Fasquelle, N. & Weemaes, S. (2013). *The social and budgetary impacts of recent social security reform in Belgium.* Paper presented at the IMPALLA-ESPANET International Conference "Building blocks for an inclusive society: empirical evidence from social policy research", Luxembourg.

FREE SOFTWARE FOUNDATION (2007). *GNU General Public License, version 3*. https://www.gnu.org/licenses/gpl.html Archived at: http://www.webcitation.org/6P4OCXu8

HDF GROUP. *HDF5 Technologies*. http://www.hdfgroup.org/about/hdf_technologies.html Archived at: http://www.webcitation.org/6P1fqLBwh

LELKES, O. & Sutherland, H. (Eds.) (2009). *Tax and benefit policies in the enlarged Europe: Assessing the impact with microsimulation models*. Farnham: Ashgate.

LI, J. & O'Donoghue, C. (2013). A survey of dynamic microsimulation models: uses, model structure and methodology. *International Journal of Microsimulation*, *6*, 3–55.

LI, J. & O'Donoghue, C. (2014). Evaluating Binary Alignment Methods in Microsimulation Models, *Journal of Artificial Societies and Social Simulation, 17 (1) 15*. http://jasss.soc.surrey.ac.uk/17/1/15.html

MANNION, O., Lay-Yee, R., Wrapson, W., Davis, P. & Pearson, J. (2012). JAMSIM: A microsimulation modelling policy tool. *Journal of Artificial Societies and Social Simulation*, 15 (1) 8. http://jasss.soc.surrey.ac.uk/15/1/8.html

O'DONOGHUE, C. (2001). Dynamic microsimulation: A methodological survey. *Brazilian Electronic Journal of Economics*, 4(2).

O'DONOGHUE, C., Lennon, J. & Hynes, S. (2009). The Life-Cycle Income Analysis Model (LIAM): A study of a flexible dynamic microsimulation modelling computing framework. *International Journal of Microsimulation*, *2*, 16–31.

O'DONOGHUE, C., Ballas, D., Clarke, G., Hynes, S. & Morrissey, K. (2012). *Spatial microsimulation for rural policy analysis*. Heidelberg: Springer.

ORCUTT, G., Caldwell, S. & Wertheimer, R. (1976). *Policy exploration through microanalytic simulation*. Washington: Urban Institute.

PYTHON SOFTWARE FOUNDATION, *Python Programming Language*. http://python.org/ Archived at: http://www.webcitation.org/6P4P7v76u

SAUERBIER, T. (2002). UMDBS-a new tool for dynamic microsimulation. *Journal of Artificial Societies and Social Simulation* 5 (2) 5. http://jasss.soc.surrey.ac.uk/5/2/5.html

SCHMIDT, Laurie J. (2000). *The Universal Language of HDF-EOS*. http://earthobservatory.nasa.gov/Features/HDFEOS/. NASA Distributed Active Archive Centers, National Aeronautics and Space Administration. Archived at: http://www.webcitation.org/6P4NIA6ID

SPIELAUER, M (2006). *The "LifeCourse" Model, a competing risk cohort microsimulation model: source code and basic concepts of the generic microsimulation programming language Modgen* (MPIDR Working Paper No WP 2006-046). Retrieved from the Max Planck Institute for Demographic Research website: http://www.demogr.mpg.de/en/projects_publications/publications_1904/mpidr_working_papers/the_lifecourse_model_a_competin

STRASSBURG, V. & Tracey, K. (2011). *Genesis User Guide (Genesis 76).* Department of Work and Pensions, London, U.K.