LIAM 2: a new open source development tool for the development of discrete-time dynamic microsimulation models

Gaëtan de Mentena, Gijs Dekkera, Raphaël Desmetb, Philippe Liégeois
c

a. Federal Planning Bureau (FPB), Belgium.
c. CEPS/INSTEAD, Luxembourg, and DULBEA (ULB), Belgium.

Version 6 : Very preliminary - Please do not quote or distribute

Keywords:

JEL-codes:

Acknowledgement:

This paper 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”. 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.
Summary:
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. Furthermore, as modellers often are not professional programmers, the result is not necessarily the most efficient in terms of simulation speed.

The objective of this paper is to introduce and present the simulation modelling package LIAM2. This is a free, open source, user-friendly modelling framework, designed for the development of discrete-time dynamic models with cross-sectional dynamic ageing and making profit from state-of-the-art methods for data-handling and simulation optimization. The paper presents a very rudimentary model and discusses the performance of LIAM2 in terms of data capacity and simulation speed. Though the latter could (and might) be better, the capacity of LIAM2 to handle large datasets is quite good, seeing that the current version of the Belgian MIDAS model simulates a dataset of no less than 2.2 million individuals.
1. Introduction

The objective of this paper is to introduce and present the simulation modelling package LIAM2. This is an open source modelling framework designed for the development of static models and discrete-time dynamic models with cross-sectional dynamic ageing\(^1\).

Microsimulation models are usually large and rather complicated models consisting of many lines of code. With the static model EUROMOD (Lelkes and Sutherland, 2009) and the dynamic model MIDAS (Dekkers et al., 2010) as the most notable exceptions, most existing models have been developed by separate (teams of) researchers. Also, in many cases, the technical instruments that are necessary for simulating any model but that are not part of the model as such, are developed by each team separately in an ad hoc manner. This includes the routines for data-reading and writing, search and alignment, syntactic analysis (parsing), and so forth. Any team that wishes to develop a model needs such tools and then has to put a lot of time and effort in the customary development of these on its own. Hence, economies of scale cannot be exploited, which makes microsimulation models even more expensive than strictly necessary. Furthermore, as modellers often are not professional programmers, the result is not necessarily the most efficient in terms of simulation speed.

A solution to these problems lies in the use of pre-existing simulation tools for modelling and development. Li and O’Donoghue (forthcoming) group packages that are used in the development of social policy oriented microsimulation models in three categories. The first two are general purpose programming languages (such as C/C++/C#, Python, Fortran, ...) and statistical packages (Stata, SAS, R). The third category includes “simulation modelling packages”, such as LIAM (O’Donoghue et al., 2009), ModGen\(^2\) (Spielauer, 2006), JAMSIM (Mannion et al., 2012), GENESIS (Strassburg and Tracey, 2011) and LIAM2\(^3\).

What this last development tool is can be explained through the analogy with a statistical package such as Stata. This allows researchers to use advanced statistical techniques and regression models without having to program the “nuts and bolts” themselves. So anyone can estimate a regression model without having to invert the data matrix or maximize a likelihood function.

LIAM2 wants to take up the same role in the field of microsimulation. The toolbox is made as generic as possible so that it can be used to develop a wide range of static or dynamic microsimulation models with cross-sectional ageing.

The basic idea behind LIAM2 is to:

- allow for comprehensible modelling of large discrete-time microsimulation models and the straightforward use of various simulation techniques
- separate ‘modellers’ from ‘programmers’, where the former are responsible for the model and the latter take up the development of the critical methodological issues, such as data reading and writing, parsing and so forth

\(^1\) All individuals are simulated at the same time for one period, then for the next period, etc.

\(^2\) http://www.statcan.gc.ca/microsimulation/modgen/modgen-fra.htm

\(^3\) UMDBS, developed by Sauerbier (2001; see Li & O’Donoghue, op.cit.) is not included in this list, because it is to our knowledge no longer in use.
use state-of-the-art methods for data-handling and simulation optimization, thereby allowing these models to cope with larger datasets at higher speed
- be as user-friendly as possible, allowing the modeller considerable freedom in applying through an easy end-user language the simulation procedures to its specific means
- stimulate collaboration and the exchange of tacit information between development teams through the use of a common development approach
- be open source, so that developers worldwide can add to the toolbox and share current and future methodological developments

LIAM 2 is the result of intense and ongoing collaboration between individual researchers from various institutions. It is primarily being developed at the Federal Planning Bureau (FPB). The Luxembourg teams (CEPS/INSTEAD and the General Inspectorate for Social Security, IGSS) are testing LIAM 2. Finally, Cathal O’Donoghue provides methodological information on microsimulation techniques and shared the source code of the first version of LIAM, which the FPB used in the AIM project (see Dekkers et al., 2010).

2. Situating LIAM2 in the family of simulation modelling packages

What are the differences between LIAM2 and other simulation modelling packages, like e.g. LIAM, ModGen, JAMSIM or GENESIS? There unfortunately is not much literature on such a comparison, with Li and O’Donoghue (forthcoming) the notable exception.

LIAM2 has been developed based on experience gained through the use of LIAM in the development of MIDAS (see Dekkers et al., 2010; Dekkers and Desmet, 2011). Thus LIAM2 is the ‘spiritual successor’ of LIAM, both packages being freely available. But LIAM2 is more general, considerably faster, and allows for much larger datasets to be simulated (cf. infra). Furthermore, where every part of a model (definition of entities, fields, their interactions and all processing rules) required separate ASCII-files in LIAM, this has now been grouped in one file. Finally, LIAM2 is broader than LIAM in terms of the tools and possibilities it offers the model developer.

ModGen is a freely available software tool developed and maintained by Statistics Canada. It is essentially a C++ library that allows modellers to include common microsimulation procedures and methodologies. So model development in ModGen still requires programming in C++ (and buying and using Visual Studio). JAMSIM is sharing the same base concept; 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).

LIAM2 is, by contrast, a development environment that has its own syntax. Even though LIAM2 itself has been developed in Python, this has no consequences for the end user. Another difference is that ModGen is more general than LIAM2. Although it is capable of handling discrete time models with cross-sectional ageing and alignment (see, for example, Paxtot et al., 2011), it is developed for continuous-time models, such as LIFEPATHS (Légaré and Décarie, 2011). JAMSIM focuses on the development of agent based models. LIAM2 is more specific than ModGen and maybe JAMSIM, focussing on the easy handling and an
efficient simulation of large static and discrete-time dynamic models with cross-sectional ageing.

GENESIS is a generic dynamic simulation model developed and used within the UK Ministry of Work and Pensions. It can be used by analysts to simulate many different scenarios without requiring changes to the model code. The model is made up of two distinct areas, the SAS code that makes up the generic Model Engine and the parameter spreadsheets held in Excel. The Processing Rules spreadsheets are set up in a standard structure. A run takes the processing rules and parameters tables from the parameter spreadsheets and then uses this to convert the generic model into a specific model. This specific model is then simulated against the data tables. Data in GENESIS can come from a variety of sources that are brought into a standard data model. With the generic model thus comes a dataset and it is not possible, or—at least- very cumbersome to take a different starting dataset for a specific application.

GENESIS is not freely available: first of all, to date and to our knowledge, it is only used within the Ministry of Work and Pensions. Secondly, it requires SAS, which is not free. Furthermore, it is more specific than LIAM2, in that it is essentially a generic model and dataset. It is therefore not possible to go outside the ‘boundaries’ of this generic model/dataset. 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 a generic model. Finally, LIAM2 is possibly faster in simulation than GENESIS.

Summarizing, LIAM2 is the spiritual successor of LIAM, but it is superior to its ‘ancester’ in terms of modelling scope, speed of simulation, data handling capacity. It seems that LIAM2 is more general than GENESIS and more specific than ModGen and JAMSIM, it does not require programming (unlike ModGen and JAMSIM), nor does it imply modifying an existing generic model and dataset (unlike GENESIS). It is specifically designed for the development of large discrete-time dynamic microsimulation models with cross sectional ageing and alignment, and it can be used for static models more or less as a side-effect. Finally, LIAM2 is available without any direct or indirect costs and is open source.

3. Looking into LIAM2

The objective of this section is to provide some basic knowledge about the contents of LIAM2 and the method for defining and simulating a model through such a package. Please refer to http://liam2.plan.be/ or Bryon et al. (2011) for more detailed information.

LIAM2 has been developed in Python (Python, 2011), a powerful general programming language that is freely available under an open source licence. Python offers large standard libraries, allows for C or C++ extension modules. It runs on all major operating systems.

LIAM2 reads and stores data in HDF5 format (HDF group, 2011). Hierarchical Data Format (HDF) is a set of libraries designed to store and organize large amounts of numerical data. It is freely available under an open-source license. Thus, data can be provided either in the form of separate CSV files, and LIAM2 can convert it into HDF5, which can thereafter be used as a

5 To illustrate how large this can be, NASA uses HDF as the official format for all products derived by the Earth Observing System (EOS, see National Center for Atmospheric Research, 2011)
starting dataset for simulations. Simulation results are also stored in HDF5 format for later processing. However, LIAM2 also has extensive reporting possibilities (cf. infra).

LIAM2 is a generic package, meaning that it allows for the simulation of whatever objects (called “entities” in LIAM2) the modeller chooses: individuals, firms, cells, etc. In this paper, however, we implicitly assume that basic objects are individuals.

Defining a typical model requires describing those objects and the variables that describe them, the way they interact (links) and how they behave over time (processes). Furthermore, the input-dataset and the simulation order have to be specified. All this is done in one file. LIAM uses the YAML-markup language (YAML, 2011), which stands for “YAML Ain't Markup Language”. It is a data serialization standard for all programming languages which meaning for the LIAM2 end-user is simply, e.g., that indentation is meaningful while writing the model in terms of structuring the several components.

Like JAMSIM, LIAM2 can be used for both static microsimulation and dynamic microsimulation with cross-sectional dynamic ageing. In many models, static ageing techniques are being used in a starting phase to create or impute variables in the starting dataset in a way that is consistent with the model. These procedures are grouped in the ‘init-section’ of a LIAM2-model, and thus add to the starting dataset prior to the actual time-dependent simulation process. However, one can use this feature to develop static microsimulation models.

LIAM2 allows for mathematical functions, conditional expressions, aggregate functions (including the mean, median, and Gini), temporal functions (lags, durations), random functions (uniform, normal), stochastic changes (logits, Monte-Carlo simulations), life-cycle functions (new([entity]), remove([entity]), clone), and a matching function (a.k.a. ‘marriage market’), that can be applied to all object levels.

Finally, LIAM2 is designed for extensive use of “alignment” by ranking on exogenous information of, up to, 3 dimensions (Li, 2011, chapter 4) and “selecting” entities for any kind of event on the basis of such a ranking. A first particularly interesting feature pertaining to this alignment is that the argument reflecting the risk (to be selected) can be one or more traditional logits pertaining to one alignment table, but it can also be a deterministic ranking number. By manipulating the score of individuals, the modeller can impose a priori to have a high of low probability of the event happening for a specific sub-class of objects. This a priori order will not under all circumstances conditionally include or exclude, since a stochastic element is added through drawing a random number from a logistic distribution. By contrast, the alignment procedure can also introduce optional “take and leave” commands. These force inclusion or exclusion of objects with specified characteristics in the selection of the event. The individuals with variables specified in the “take” command will a priori be selected for the event. Suppose that the alignment specifies that 10 individuals should experience a certain event, and that there are 3 individuals who meet the conditions specified in the take. Then these 3 individuals will be selected a priori and the alignment process will select the remaining 7 candidates from the rest of the sample. The leave command works the other way round: those who match the condition in that statement, are a priori excluded from the event.

Note that this conversion needs to be done once. After an HDF5 file is created, various models can use it for all sorts of simulations.
happening. The “take” and “leave” are absolute conditions, which mean that the individuals meeting these conditions will always experience the event.

Another useful functionality of LIAM2 is that it respects the existing values of a variable in a certain period. Thus, if a variable in a certain period of time is available, LIAM2 will not overwrite it but use it instead. This feature has been introduced to allow for retrospective modelling. So, for example, we start our simulations in 1947 and we have for each year the labour market status of the individual. The retrospective part of the model MIDAS then simulates the pension claim that the individual builds up for every retrospective year. At the end of the retrospective simulation, so in 2001, the model produces the 2001-dataset (which is also in its input) including the endogenous variables that summarize the retrospective information (number of years in each scheme, mean indexed income subject to ceiling and floor, application of minimum right in previous years, ...). In short, the retrospective model produces the starting dataset for the prospective model. Another application 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 the model MIDAS, and use it as input for a separate run without the demographic module. This not only introduces a considerable savings in runtime, but it also means that the result of the demographic module is identical for various simulation runs, which allows for a more straightforward identification of individual winners and losers of a specific policy measure.

3.1. A birds-eye view on a model

A typical model in LIAM2 is involving several kinds of elements: the “entities”, the “fields”, the “links”, the “processes”, the “globals”, the “macros” and a few environmental initializations (input, output, number of periods and the start period).

An “entity” is an object with a unique identifier. It can consist of other entities (e.g. households that are composed of individuals), and it is described by a set of attributes. Each of these attributes is called a “field”. There are three types of fields: booleans, integers and floats. Note that a field may, but need not, to be included in the starting dataset. A “link” is a relation between an entity and one or more other entities. For example, mothers are linked to their children, spouses are linked together, and individuals are linked to households, and vice versa. It can be mentioned that as soon as link have been defined, accessing a field of a linked object (possibly of the same entity) is very simple through a command of the type “link_name.field_name” (for example, “mother.age”, but also “mother.mother.age” and so forth). A “global” is a parameter that is not related to a specific entity. For example, the parameters that describe a social security system enter the model as globals. Finally, a “macro” is a container that operates on the level of the fields. Macros may include a value, or a piece of code, and is re-evaluated each time the macro is referenced. The advantage of using macros is that they make the model code considerably easier to develop, understand and maintain.

The Box 1 below presents a very rudimentary setup of a model (other than data importing functionality). It is typically composed of three main blocks : “globals”, “entities” and a “simulation” block. Within each of them, indentation helps in structuring the different components.
Box 1 : A Simple model in LIAM2

The model starts in 2002 ("start_period") and simulates 10 years ("periods"), so up to 2012. The unique type of entity is referring to individuals ("person") in the model. The input dataset, named “base.h5” contains the identification number of the individual and his or her age ("input"). Output is written to “simulation.h5” ("output").

The first block of the model lists the globals or parameters to be used in the model. In our example, there is one parameter only, RETAGE, which is the mandatory retirement age.

The second block contains, for every entity used in the model, the fields, the links with other entities (none in our example), the macros (none in our example) and processes. Our model contains only individuals as entities, for whom the fields “age” and “t_in_retirement” are defined. Note that only “age” is present in the dataset, and “t_in_retirement” is undefined (“void”) until it receives a value from a process. Now comes the main part of this block which is the list of processes that pertain to the entities. In this case, there are two processes, of which the name coincides with the dependent variable. The first process increases age by 1 for every individual. The second process sets the field “t_in_retirement” as the difference between the age of the individual and the mandatory retirement age. Or, it is the number of years in retirement, assuming that one cannot enter into retirement before the mandatory retirement age. This process also illustrates conditional statements: “t_in_retirement” has a value of -1 if the individual is younger than the mandatory retirement age RETAGE. Note, finally, that the order of the processes in this entities block is not relevant. Those are just defined here, in order to be used later on in the next block.

The third block is the simulation block. This contains the names and paths to the input- and output-datasets, the starting period and the number of simulation years. But the main part of this block is the processes-block, where the processes available in the model are listed in the
order of simulation. Our simple model consists of two processes only: simulating the age and the number of years that somebody has been in retirement.

The above example is of course made very simple, just designed to illustrate the setup of a model in LIAM2. It contains no “init-section” for static simulation, just one entity level, no links, no macros and just two deterministic processes.

3.2. Output, interaction and debugging

As said, LIAM2 writes the simulation results to a HDF5 output file. This can be analysed with the help of an HDF5 browser. But it can also be transformed to a format readable by the more traditional packages, such as Stata, SAS or R. However, this is often a very large dataset, and it is cumbersome to transform it. And even then can the simulation dataset be too big to handle.

To overcome these problems and to allow for a rapid debugging and analysis of the simulation results, LIAM2 is offering several tools. First of all, it can be run in interactive mode. All functions available can then be typed in the console to inspect and analyze the simulation data. This, combined with the possibility to put breakpoints anywhere in the model, opens important possibilities for debugging and data analysis. When reaching such a breakpoint or when reaching the end of the model, LIAM2 will shift to the interactive mode, thereby allowing the user to analyse the data by inputting any LIAM2 command (see Box 2 as an example).

Box 2 : Outputting information to the console, an example

```
show(groupby(age / 10, gender))
...
gender   | False | True  | total
(age / 10) |       |       |      
0        | 818   | 803   | 1621 
1        | 800   | 800   | 1600 
2        | 1199  | 1197  | 2396 
3        | 1598  | 1598  | 3196 
4        | 1697  | 1696  | 3393 
5        | 1496  | 1491  | 2987 
6        | 1191  | 1182  | 2373 
```

Secondly, LIAM2 includes an extensive class of output routines, which allows any model to write individual data as well as aggregates to CSV files. For example, one can create periodic CSV files containing a selection of variables describing certain individuals in any specific period. But one can also create pivot tables, and key output variables (such as means or medians, but also more specific indicators such as at risk of poverty rates, Gini’s) and have them append to a CSV file over all simulation years. All in all, this allows for a tailor-made range of output datasets that summarize the simulation data, and that can immediately be used to assess the simulation results, create tables and figures. It is therefore possible to develop all output variables and tables needed to assess the results of the model in that same model.
3.3. Performance

The performance needs to cover two different aspects. The first dimension of performance is the size of the dataset the toolbox can handle. The second is the speed of simulation. The two are obviously connected, since it does not do much good to know that a simulation model can in theory simulate a large number of entities if the simulation speed is so low that it takes a decade to simulate one year. Inversely is an enormous speed of simulation of limited interest if the toolbox can handle datasets too much restricted in terms of volume.

At this point, LIAM2 is being used on a regular basis in the development and use of the model MIDAS for Belgium. This is a model of 3500 lines of code, 142 parameters, 132 (permanent) variables, 18 aligned processes, 14 CSV output files. This model is being developed, tested and validated on a dataset of 300,000 individuals. The actual simulation runs are being done on an expanded version of this dataset, being 2.2 million individuals.

One simulation run from 2002 to 2060 on the testing dataset of 300 thousand individuals takes 2 hours and 16 minutes. A simulation run on the 2.2 million individuals dataset takes 14 hours and 41 minutes. The simulation times are fast, but not extremely fast. The most CPU-consuming processes are the alignments, the matching process between individuals, and the creation of output CSV files. A partial and first explanation of the simulation time is the large number of alignments that MIDAS uses. For example, one of the fastest models in the world, INAHSIM, does not use any alignment at all. A second reason is that, because of the large starting dataset, the output dataset has a size of roughly 200 GB. This is too big to be convenient, so that we have chosen to develop all output variables and tables needed to assess the results of the model directly from within LIAM2, thereby avoiding post-simulation manipulation of the output dataset. Finally, even though LIAM2 is fast, a loss in performance is also the price one pays for not “hard coding” the model in the simulation software. It should be added that some more technical parts of LIAM2 are still under development. We thus expect that these performance times might be improved in the future.

4. Conclusions

This paper presents and discusses LIAM2, a new open source development tool for the development of static and discrete-time dynamic microsimulation models. It is a simulation framework that allows for comprehensible modelling and the straightforward use of various simulation techniques. LIAM2 is the spiritual successor of LIAM and is complementary to, e.g., ModGen.

The basic idea behind LIAM2 is to provide, on the one side, a user-friendly development environment for the development of large discrete-time dynamic microsimulation models, making profit from state-of-the-art methods for data-handling and simulation optimization. LIAM2 is also aiming, on the other side, to stimulate collaboration and the exchange of tacit information between development teams through the use of a common development approach.

The paper presents a very rudimentary model and discusses the performance of LIAM2 in terms of data capacity and simulation speed. Though the latter could (and might) be better, the
capacity of LIAM2 to handle large datasets is quite good, seeing that the current version of MIDAS simulates a dataset of no less than 2.2 million individuals.

An important characteristic of LIAM2 is that it is freely available under a OSI-approved open source license (Open Source Initiative, 2011). On the LIAM2 website (http://liam2.plan.be/) one can download a bundle, containing the LIAM2 executable, a Notepad++ text editor, pre-configured to work with LIAM2 models, the documentation in html, pdf and chm (windows help) format, and, finally, a demonstration model with a synthetic data set of 20,200 persons grouped in 14,700 households. Furthermore, this website allows registering to the LIAM2-announce mailing list, and sending requests for help or information via email. Finally, the source code of LIAM2 will be disposal shortly.
References


