JASSS

Oliver Mannion, Roy Lay-Yee, Wendy Wrapson, Peter Davis and Janet Pearson (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>

Received: 15-Mar-2011 Accepted: 13-Oct-2011 Published: 31-Jan-2012

Abstract

JAMSIM (JAva MicroSIMulation) is an innovative synthesis of open source packages that provides an environment and set of features for the creation of dynamic discrete-time microsimulation models that are to be executed, manipulated and interrogated by non-technical, policy-oriented users. Combining the leading open source statistical package R and one of the foremost agent-based modelling (ABM) graphical tools Ascape, JAMSIM is available as an open source tool, for public reuse and modification. Here we describe microsimulation, our functional requirements, a review of tools used by other micro-simulators and an evaluation of existing software, followed by the architecture, features and use of JAMSIM.

Keywords:
Microsimulation, Software Frameworks, Policy Tool, Java, R

Introduction

1.1 Microsimulation is an empirically based data modelling methodology traditionally used in the areas of taxation, pensions, and other types of economic activity, but is increasingly being applied to the social sciences (Brown and Harding 2002). Microsimulation can be distinguished from other types of computer modelling in that it simulates the state and behaviour of individual units such as people or households and relies on empirical data for initial conditions and rules (Gilbert and Troitzsch 2005). Investigation of point estimates and distributions of outcomes can then be performed for a range of heterogeneous subgroups of interest. In empirical terms, each unit is represented by a data record containing a unique identifier and a set of associated attributes, such as age, gender, marital status and education. A set of rules (for example transition probabilities) intended to represent individual preferences and tendencies, is then applied to each unit, leading to simulated changes in state and behaviour. The end result is an individual life history and its projection into a future time period.

1.2 Microsimulation can be a useful tool for policymakers because it allows an evaluation to be conducted of the effects of proposed government policy interventions before they are implemented in the real world. We have undertaken two policy-oriented microsimulation projects that have required us to model characteristics of certain sub-populations (children in one and elderly people in another), forecast outcomes, and develop and run scenarios of interest to policymakers. The ultimate goal of our research is to assist policymakers to provide sound advice and thus, as with many other microsimulation models, these models are designed for use by government agencies.

1.3 Microsimulation models used by policymakers ideally need to be encapsulated in self-contained software applications that can be run on a desktop computer and easily operated by non-technical users. While there is a large pool of software toolkits for agent-based modelling (ABM), there is a dearth of tools for microsimulation. We firstly describe the functional requirements of policy-oriented microsimulation models. We then evaluate existing software tools for this purpose, before describing the microsimulation tool JAMSIM (JAva MicroSIMulation) that we have developed.

Requirements

2.1 The functional requirements of the microsimulation toolkit were developed from the nature of the particular modelling problems we were addressing. The first was modelling primary care in an ageing society (PCASO) (Davis et al. 2010; Pearson et al. 2011) and was originally programmed for a technical audience in SAS. We wanted to convert this into a tool usable by policymakers. The second was a life course approach modelling health outcomes for children using a longitudinal data set. The requirements we developed for an end-user tool implementing these models are listed in Table 1.

Table 1: Functional requirements of our microsimulation models

Base file input	Create individual micro-level population units, or agents, from multiple data sets, which may require some data manipulation and merging.
Parameter input	Load multiple tables of parameters from CSV/Excel files, which can be modified in the user interface to test different scenarios.
Scheduling mechanism	Simple discrete-time. State changes occur at every time period.
Simulation techniques	Stochastic equations involving Monte Carlo draws from discretely specified probability distributions. Requires random number generation. Prediction from statistical models. Transition matrices. Multiple runs (with same parameters but different random seeds) to obtain a robust mean estimate and confidence interval. Reweighted inputs or outputs to adjust population distributions.
Scenario testing	The ability to directly change and/or reweight any continuous or categorical variable used in the model. The model can then be rerun with the changed variables and the outputs compared with a baseline run.
Output	The ability to export simulated results for analysis in external software as well as GUI features for the display of: i) Aggregates: charts and tables of aggregate information per year, e.g. frequency distribution, mean, standard deviation and other descriptive statistics; a facility to weight and reweight the sample to obtain representativeness and for scenario testing. ii) Individuals: the complete data set (i.e. state of each individual) after each year; ability to track an individual history (state

	changes over time).
User interface	To encourage a wide user-base, the user interface should be intuitive. It should be "easy" for an end-user to operate functions including simulation control, scenario testing, and output display.
Performance	External end-users should be able to perform multiple runs of a multi-year simulation on a desktop computer in a reasonable amount of time (maximum of 5 minutes).

Software evaluation—Microsimulation models

3.1 In determining an appropriate software platform for implementing our models, we began by researching software implementation details and application features of existing microsimulation models. In addition we consulted articles providing general software selection and development advice for microsimulation (Percival 2007; Scott 2003). We also communicated with existing microsimulation model users and developers on the SIMSOC mailing list, and our own contacts, to seek their feedback and advice. A summary of our microsimulation model survey is shown in Table 2. Rather than a comprehensive survey of all available microsimulation models, this table shows models that were either well known and widely used, similar enough in domain content and/or features to the type of models we sought to build, or distinct and unique in implementation and approach from other models in the field.

Table 2: Relevant Microsimulation Models and Software Application Details

Name / Yr	Type, Domain	Language	GUI Features	Availability
APPSIM 2005-	Dynamic life-course	C# (Percival 2007)	Simulation control, alignment, scenario testing, sensitivity analysis	Partner organisations only
CARESIM 1998-?	Dynamic, distributional effects of care charging regimes	SAS (Wittenberg et al. 2007)	No GUI	Unknown
EUROMOD 1996-	Static, tax-benefit system	C++ (Immervoll et al. 1999; Sutherland 2007; Sutherland et al. 2008)	Simulation control, scenario testing	On request for not-for-profit use
LABORsim 2006-?	Labour supply modelling	Java and JAS (Leombruni and Richiardi 2006)	Simulation control, parameter changes, output tables and graphs	JAS framework downloadable
LifePaths 1994-	Dynamic life-course	C++ and Modgen (Rowe and Gribble 2007)	Simulation control, output tables, BioBrowser	Downloadable
MicMac 2005-2009	Dynamic life-course	Java, R and JAMES II (Gampe et al. 2009; Zinn et al. 2009)	Simulation control, (non-integrated pre- and post-simulation processing in R)	Downloadable
MIDAS	Dynamic life-course	C++ and LIAM	Provides GUI but features unknown	On request
PENSIM 1997-	Pension plans	C++ with custom model specification language (Holmer et al. 2011)	Simulation control	Downloadable
SAGE 1999-2005	Dynamic life-course	C++ with custom model specification language (Evandrou et al. 2007; Scott 2003)	Simulation control	Unavailable
SVERIGE-3 ?	Dynamic, spatial life-course	C# with custom model and equation specification (Holm et al. 2007)	Provides GUI but features unknown	Unavailable due to data restrictions

3.2 Several of these models (LABORsim, LifePaths, MicMac, MIDAS, PENSIM, SAGE, SVERIGE-3) were built within a generalised framework that caters to more than a single specific model. However, because of the complexity of different specialist needs, there is currently no single predominant or generic gold standard microsimulation framework. Using a generalised framework allows the model developer to enjoy the benefits of modularisation and reuse. This includes reducing the amount of code and documentation that needs to be written from the ground up, which shortens development time and decreases costs. In addition, reusing software components that have already been extensively tested reduces error potential. Some level of reuse, if possible, is therefore desirable, and we agree with Ropella et al. (2002: 10) that in only very rare situations would it be advantageous to develop a system without utilising any existing software packages.

3.3 The generalised frameworks used by the above models are of two types. The first is a model specification framework (MSF) that consists of: i) a bespoke declarative language for model features such as agents, events/state transitions, parameters and output tables, and ii) an application environment that provides an event queuing/simulation scheduling mechanism, a user interface and input and output formats. Typically these frameworks do not require a model developer to be a computer programmer as well, but they do require learning the specifics of the model specification language and operation environment provided. Model specification frameworks are used by LifePaths, PENSIM, SAGE and SVERIGE-3.

3.4 Of the MSFs employed by these models only the Modgen framework (Statistics Canada 2011a) used by LifePaths is readily available and well supported. Modgen is a dynamic microsimulation-specific toolkit, as opposed to an ABM one, with a complete set of features including base file loading (from a custom .DAT file format), simulation, aggregate outputs (in tables) and individual agent life history output (via the included BioBrowser tool). Its custom model specification language introduces a learning curve for the model developer, but once learnt it can be used to implement a wide range of dynamic models. It supports both discrete and continuous time event frameworks (for details on the distinction, see Scott 2003:4). Although time-based processing is possible, in which all individuals are simulated at time t before moving to time *t+1*, Modgen primarily uses case-based processing of individuals. In other words, each case is simulated from the beginning to the end before the simulation of the next case begins (Statistics Canada 2011b). This limits the modelling of interactions to only between individuals in each case rather than the whole population. The advantage of case-based processing is that cases can be simulated in parallel. Modgen also allows open populations, or ones in which new individuals can be generated in response to certain events such as partnership. Overall, Modgen came very close to meeting all our requirements. However, although the support options were significant, we felt uncomfortable relying on the continued availability of that support within a closed source model. The closed source model also limited the ability of Modgen to be extended and extra functionality added if and when we required it.

3.5 Two other specification-style frameworks deserve mention. While SAS is a general purpose data analysis tool rather than a model specification framework per se, it is similar in that it provides a custom language and application environment. In part because of prior experience with the SAS language in other domains, SAS has been adopted by some microsimulation developers. CARESIM is an example of a SAS microsimulation model, as is our earlier work with the PCASO model (Pearson et al. 2011). However, because of SAS's lack of a user friendly GUI, it was ruled unsuitable for a tool that was aimed at policy end-users. Secondly, UMDBS (Sauerbier 2002) is a dynamic time-based MSF written in Smalltalk that often receives mention in software reviews. We also considered it for our modelling purposes but decided against it because of its limited user interface, its closed source and non-extensible nature, and its lack of support and recent releases.

3.6 The second type of generalised framework is a set of libraries, or packages, that together provide functionality to implement a microsimulation model. In comparison to an MSF like Modgen these frameworks are typically more modular and extensible, especially so if they are open source, but less complete in functionality. The functionality that is typically provided includes a simulation event queue, a user interface for simulation control, and varying levels of output functionality. Implementing a specific model in a library framework requires additional programming in a general purpose programming language like C++ or Java, rather than a model-specific language.

3.7 Of the microsimulation models sampled above, LABORsim, MicMac and MIDAS use a library framework. MIDAS uses the comprehensive and flexible C++ Life-cycle Income Analysis Model (LIAM) framework (O'Donoghue et al. 2009). LIAM offers single and multi-cohort dynamic life-course microsimulation with a graphical user interface and has been used to model the redistributive impacts of tax-benefit and pension systems. LIAM is used for problems that have closed populations, i.e. in which no new individuals are generated, and which use discrete-time state changes. It supports transition matrices, regression and arithmetic transformations, and includes a marriage market module and functionality to handle migration. Behavioural feedback loops can be incorporated and alignment with external data sources is possible. However, after consideration, the availability of this framework on a request only basis and a lack of documentation and structured support limited its appeal to us.

3.8 LABORsim uses the open source Java Agent-based Simulation (JAS) library (Sonnessa 2003). It is primarily an ABM framework with an event list that can fire events at specified discrete time intervals. It has not been explicitly designed with microsimulation in mind, although microsimulation can be accommodated within its framework. The advantage of JAS over other ABM toolkits is that it provides support for reading from CSV and Excel files. However its last release (v1.2.1) was in 2006 (Sonnessa 2011) and so without recent activity or a viable support base it was ruled out.

3.9 MicMac is a Java based dynamic microsimulation model that uses the JAMES II (Himmelspach and Uhrmacher 2007) library framework, which had a v0.8.3 alpha release in 2009 (Uhrmacher et al. 2011) and has planned future releases. JAMES II is readily available, open source, and based on an extensible plug-in framework with a large repository of existing plug-ins. Its focus is on biological simulation problems, although its plug-in design means it need not be restricted to any one particular simulation paradigm. JAMES II supports both discrete and continuous time simulations of open and closed populations. There has been recent activity on the JAMES II project, a range of publications that cite the Himmelspach and Uhrmacher (2007) conference paper, good documentation, and several projects that branch off JAMES II. However, its status as an "alpha" release led to our perception that it had not yet reached the maturity of other offerings.

Software evaluation—ABM toolkits

4.1 Outside of microsimulation, library frameworks have been very popular, and they are well developed in the ABM world. The main difference between ABM and microsimulation is that microsimulation models rely on empirical micro data sets where agent-based models do not. ABM toolkits typically lack the ability to easily read data sets of multiple variables and convert them into agents for simulation. Microsimulation models make use of statistical modelling techniques, and probabilities derived from empirical data, to transform inputs into outputs. ABM toolkits do not generally cater for these sorts of transitions but instead provide rule-based transformations which are more useful for the theoretically-derived models of the ABM world. Although ABM and microsimulation differ in intent, purpose and technique, there are enough similarities in software functionality that we decided to include general purpose ABM library toolkits in our selection process.

4.2 In August 2009 we searched relevant computer simulation journals and conducted internet searches to determine what was available. In particular we found and consulted a number of ABM software reviews (Gilbert and Bankes 2002; Lawson 2008; Nikolai and Madey 2009; Railsback et al. 2006; Tobias and Hofmann 2004). In total we identified 30 potential microsimulation/ABM toolkits or environments, of which most were discounted because their focus was on a different problem domain, or they lacked functionality, were not extensible, or were outdated and no longer offered support.

4.3 Of those we considered in depth, Anylogic Professional Edition v6.5 (XJ Technologies 2011) came closest to meeting our requirements. It provided the most comprehensive and visually appealing user interface, including graphs and advanced visualisations, of any of the tools we examined. It is, however, closed source, and it requires a license fee for each run-time instance of a deployed model. We felt that the inability to modify the source code meant we would not be able to fully tailor it to our needs, particularly in the area of statistical analysis. The remaining contenders were Repast Simphony v1.2 (North 2011a), Repast v3 (North 2011b), and Ascape v5.6 (Parker 2011). Like Anylogic they are all well developed, with substantial histories and ongoing development, and existing user and support bases. Unlike Anylogic they are all open source and so provide complete extensibility. However, they are ABM-specific and so lack data input, output, and analysis functionality.

4.4 The advantage of an open source library framework like Repast or Ascape is that a developer can select and reuse only those library functionalities that they require. Because no one tool provided all the functionality we required, we finally decided to combine elements of multiple open source libraries in order to meet our requirements. Our solution was to use Ascape as the user interface and for simulation control, and combine this with R (R Development Core Team 2011) for statistical analysis and output. This left us to write our own microsimulation transformation code in Java. This hybrid solution, "JAMSIM", was a compromise between relying completely on a single software package (none of which offered total flexibility or met all our requirements) and writing our own from scratch (a time-consuming process).

4.5 The existing packages we sought to utilise were either written in Java or provided interfaces to Java. We realise however, that, as the above survey shows, C++ has clearly been the language of choice of micro-simulators. This is probably largely influenced by their age, as at the time of model inception, alternative languages such as C# or Java were unavailable or lacked popularity. In addition C++, as a compiled language, offers the greatest performance potential. However, given the low computational requirements of our particular microsimulation models, we decided the potential performance gains from a machine-compiled language would not be necessary. Overall, we reasoned that the availability of higher level programming components in an interpreted language such as Java outweighed the advantages of speed provided by a direct machine-compiled language like C++. This was the same conclusion reached by Percival (2007) in selecting C# over C++. Finally, Java matched the skill set available in our organisation.

JAMSIM overview

5.1 JAMSIM is a unique synthesis of open source packages that provides an environment and set of features for the creation of microsimulation models that are to be executed, manipulated and interrogated by non-technical, policy-oriented users. JAMSIM 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. Table 3 lists the packages JAMSIM combines and the functionalities they provide. These functionalities include a user interface, file input, in-memory data set storage and transformation, access to statistical functions and graphing and table output capabilities. In particular, JAMSIM combines the leading open source statistical package R and the Java-based Ascape, one of the foremost agent-based modelling graphical tools. JAMSIM's loose coupling and open source heritage and development make it an accessible, extensible toolkit for developers who want complete control over the code base. The source code and binaries are freely available at http://code.google.com/p/jamsim/. This paper discusses features currently available in JAMSIM v1.3.4.

Table 3: Third party Java packages used in JAMSIM

Package	Description
Ascape	The Ascape agent-based modelling toolkit is used by JAMSIM as the GUI, for the main discrete-time based simulation loop, and for simulation control (stop, start, pause, etc.). It has been adapted to provide access to tabular and graphical outputs in a fashion modelled loosely on Modgen.
R	JAMSIM embeds the R statistical programming language which makes available the extensive statistical and graphing capabilities of R. Simulation objects, such as the agents, parameter files, and simulation outputs are available in R for analysis such as producing descriptive statistics. An R console is also embedded in the Ascape GUI for direct user interaction, and the Ascape GUI has been extended to display R graphics.
Casper datasets	A microsimulation model requires the manipulation of micro-data sets and additional parameter files. These data sets need to be modifiable and to have input and output interfaces. Data sets in JAMSIM are processed using Casper datasets, a generic, in-memory data set manipulation library.
Read My Tables	For the loading of tabular data stored in CSV or Excel files, such as the base file and parameter files.
Colt	JAMSIM provides conversion of microdata sets to matrices for manipulation in Colt. Colt is a high performance scientific and technical computing library for Java that provides multidimensional matrices, linear algebra, histogram and basic statistical capabilities.

JAMSIM Models

6.1 This section describes the types of microsimulation models JAMSIM can be and has been used for. In particular, JAMSIM is currently being used to implement policy-oriented health dynamic microsimulation models. Table 4 details the essential components of a JAMSIM model (JSM), describing what they do and in which location (language) they may be implemented.

Table 4: Components making up a JAMSIM model (JSM)

Component	Location	Description
RootScape	Java	Root component that defines, loads and sets up all other components. Responsible for the loading and initialisation processes.
ScapeData	Java	Defines all inputs used by the JSM, including the base file, data sets, data dictionary, parameter sets (including weighting calculators) and analysis menu commands.
MicroSimCell	Java	Defines the population unit, or agent, that makes up the base file and is the micro unit of analysis, e.g. patient, child, household, etc.
Iteration steps	Java	Steps that transform the base file. These implement statistical techniques and models that move the base file from time step *t* to time step *t+1*.
Outputs	Java / R	The output tables and graphs that display simulation results.

6.2 JAMSIM supports both static—providing a cross-sectional snapshot—and dynamic—incorporating change over time—models. To project a static model forward in time, final simulated results can be reweighted to produce a distribution that matches that of a future population. In a more realistic way, JAMSIM can dynamically model a series of state changes, or outcomes, for a set of individuals. For example, if we were modelling the early life course of children these outcomes may be a change in parental characteristic (e.g. Mother starts or stops smoking), a change in family status (e.g. parents break up or partner) or a health status outcome (e.g. a visit to the doctor). JAMSIM is dynamic in that these changes are generated as part of a time sequence in which the current state at time t is dependent on the previous state at time *t-1*.

6.3 JAMSIM was designed for modelling problems that involve discrete time state changes, rather than continuous time or case based state changes. JAMSIM's core simulation process is a sequential loop in which state changes can be generated at fixed intervals (iterations) simultaneously for all individuals. State changes may occur at each iteration for each individual according to a set of transition probabilities and the individual's unique demographic attributes. Transition probabilities for individuals may be specified via external tables.

6.4 After generating a unique transition probability for each individual, a Monte Carlo simulation is typically used to determine if a state change will actually occur for an individual or not. This is done by drawing a random number on the interval [0,1] from a specified distribution. The drawn random number is compared to the transition probability, and if it is less than the random number then the transition occurs. While JAMSIM is aimed at simulation models that involve probabilistic state changes, its Ascape heritage allows deterministic rule models to be used in much the same way as they are with ABMs.

6.5 The Ascape component of JAMSIM provides the ability to create not only closed but also open models, in which new population members can be generated during the simulation. As in Ascape, JAMSIM models can be composed of a hierarchy of `scapes', which is a collection of agents that can represent basic units such as a child, parent, household, etc. (see Parker 2001). This allows for models that have multiple levels of simulation and analysis, such as individual, family, household or subpopulation. Such models may also accommodate specific linkages between individuals and/or different units, such as links between parents and children, or individuals that are partnered.

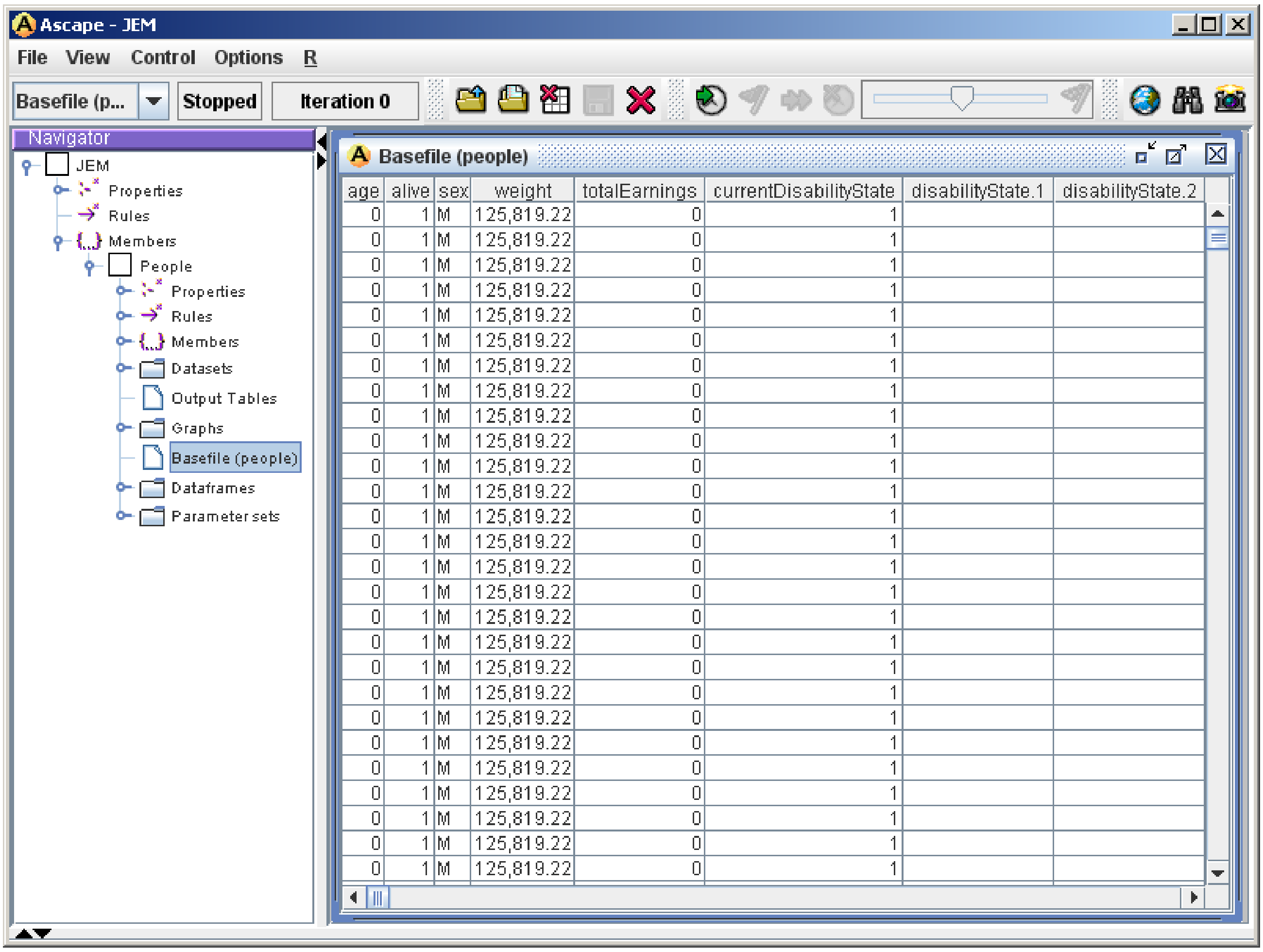
6.6 JAMSIM has been used for the development of static health microsimulation models. In particular it has been used to model pathways to primary care under scenarios of population ageing (Davis et al. 2010; Pearson et al. 2011). The implementation involved a data set of 13,500 patients. From sets of discretely specified probability distributions, the simulation models recent illness conditions for each patient, the number of times they might visit a general practitioner (GP), the primary diagnosis given by the GP, and the associated GP activity (e.g. investigation, prescription, followup, referral).

JAMSIM Model Simulation Process

7.1 The JAMSIM model simulation process consists of setup, simulation and output phases. This section describes these phases using the JAMSIM Example Model (JEM). JEM is an implementation of the Simpex model from Modgen (Statistics Canada 2011a). It is a fictitious and simplistic model that serves to illustrate key features of JAMSIM. JEM simulates the disability state of a population of males and females. A disability state is either no disability, or mild, moderate or severe disability. The disability state of each individual influences their earning capacity, which is the key output of the model.

Setup

7.2 A population in JAMSIM is represented by a collection of Java MicroSimCell objects. The initial state of a population can be generated from parameters, or each individual unit may be pre-specified in a base file. When starting from a base file, each item in the base file represents a single unit (e.g. child, patient, household) with a corresponding set of variables (e.g. gender, age, income, visits to the GP). When a JSM starts, it loads a user-specified CSV or Excel file that represents this initial base file. The variable data types can be determined automatically by inspection, or specified by a data set definition file. A variable data type may be specified as "optional", which will allow missing values. For each row in the base file, a corresponding MicroSimCell is created with the variables specified in the base file. After loading, the entire set of MicroSimCells is stored in R as a dataframe.

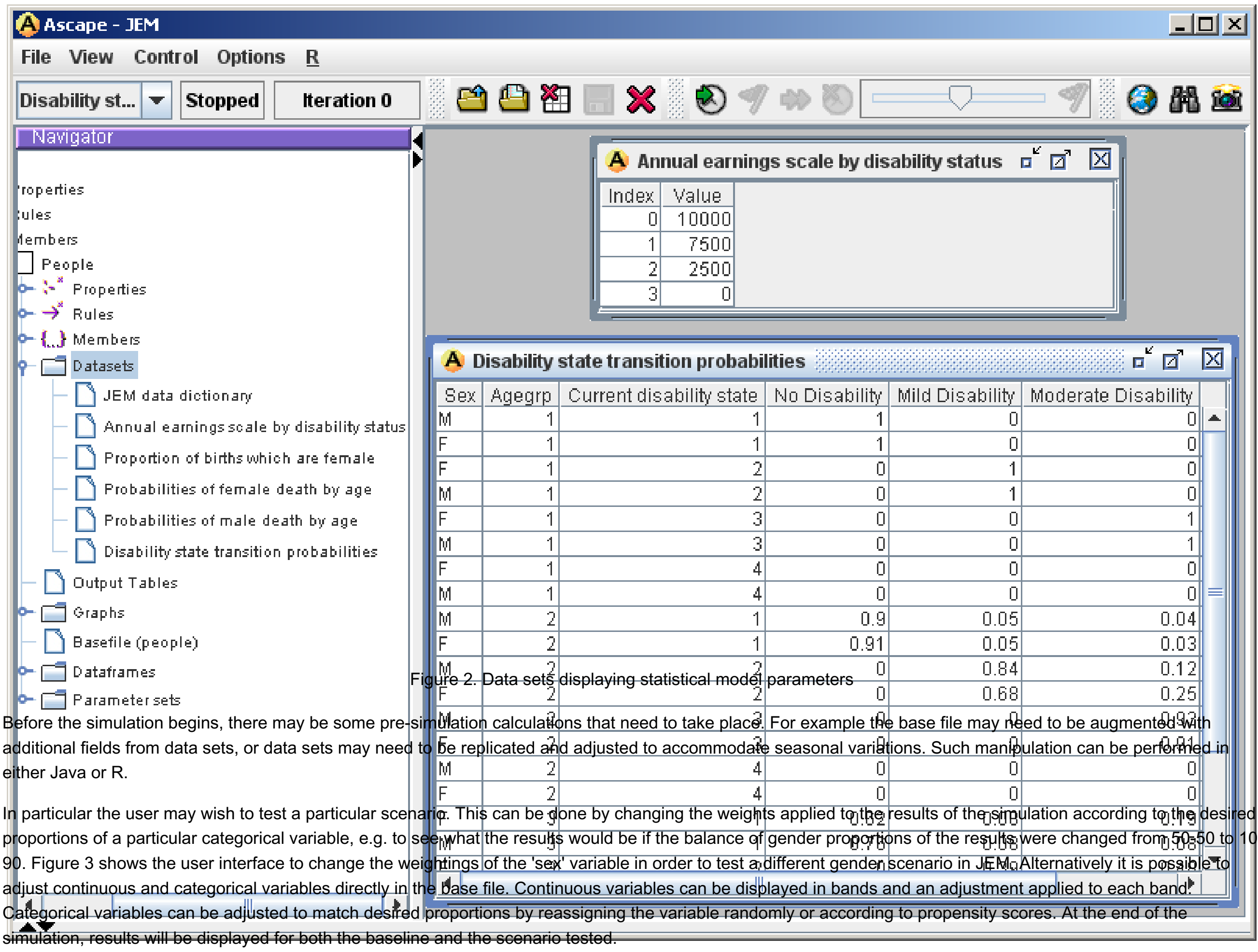


age	alive	sex	weight	totalEarnings	currentDisabilityState	disabilityState.1	disabilityState.2
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		
0	1	M	125,819.22	0	1		

Figure 1. Base file display

Figure 1 exhibits the base file from JEM. The fabricated base file, which is not based on any real world data, contains the variables age, alive, sex, weight, etc. Those variables that are currently empty (e.g. disabilityState.1) will be generated during simulation.

7.3 A JSM may rely on a range of external data, e.g. transition tables, statistical coefficients and intercepts, incidence rates, adjustment factors, and event probabilities. These can all be supplied as CSV or Excel files, and loaded in and represented as Casper data sets, matrices, and/or R dataframes. Once loaded they are available globally for use in both simulation and analysis. Figure 2 shows a screenshot of the data sets loaded from external data sources by JEM to be used in the simulation. The disability state transition probabilities shown below are used to model the change of disability state from year to year, and the earnings scale is used to calculate the amount earned per year, by disability state.



Ascape - JEM
File View Control Options R
Disability st... Stopped Iteration 0
Navigator
roperties
ules
lembers
People
Properties
Rules
Members
Datasets
JEM data dictionary
Annual earnings scale by disability status
Proportion of births which are female
Probabilities of female death by age
Probabilities of male death by age
Disability state transition probabilities
Output Tables
Graphs
Basefile (people)
Dataframes
Parameter sets

Annual earnings scale by disability status

Index	Value
0	10000
1	7500
2	2500
3	0

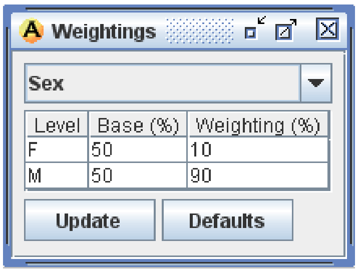
Disability state transition probabilities

Sex	Agegrp	Current disability state	No Disability	Mild Disability	Moderate Disability
M	1	1	1	0	0
F	1	1	1	0	0
F	1	2	0	1	0
M	1	2	0	1	0
F	1	3	0	0	1
M	1	3	0	0	1
F	1	4	0	0	0
M	1	4	0	0	0
M	2	1	0.9	0.05	0.04
F	2	1	0.91	0.05	0.03
M	2	2	0	0.84	0.12
F	2	2	0	0.68	0.25
M	2	3	0	0	0.93
F	2	3	0	0	0.91
M	2	4	0	0	0
F	2	4	0	0	0
F	3	1	0.82	[illegible]	[illegible]
M	3	1	0.78	0.08	0.08

Figure 2. Data sets displaying statistical model parameters

Before the simulation begins, there may be some pre-simulation calculations that need to take place. For example the base file may need to be augmented with additional fields from data sets, or data sets may need to be replicated and adjusted to accommodate seasonal variations. Such manipulation can be performed in either Java or R.

7.4 In particular the user may wish to test a particular scenario. This can be done by changing the weights applied to the results of the simulation according to the desired proportions of a particular categorical variable, e.g. to see what the results would be if the balance of gender proportions of the results were changed from 50-50 to 10-90. Figure 3 shows the user interface to change the weightings of the 'sex' variable in order to test a different gender scenario in JEM. Alternatively it is possible to adjust continuous and categorical variables directly in the base file. Continuous variables can be displayed in bands and an adjustment applied to each band. Categorical variables can be adjusted to match desired proportions by reassigning the variable randomly or according to propensity scores. At the end of the simulation, results will be displayed for both the baseline and the scenario tested.



Sex

Level	Base (%)	Weighting (%)
F	50	10
M	50	90

Figure 3. Reweighting results by the variable Sex

7.5 Finally, the user may wish to perform analyses on the base file before it is transformed by the simulation. This may involve inspecting graphs specified by the JSM, or by user-specified R commands entered on the console. Figure 4 illustrates a graphical analysis that shows how the distribution of gender has been changed to test the 10-90 female-male scenario in JEM.

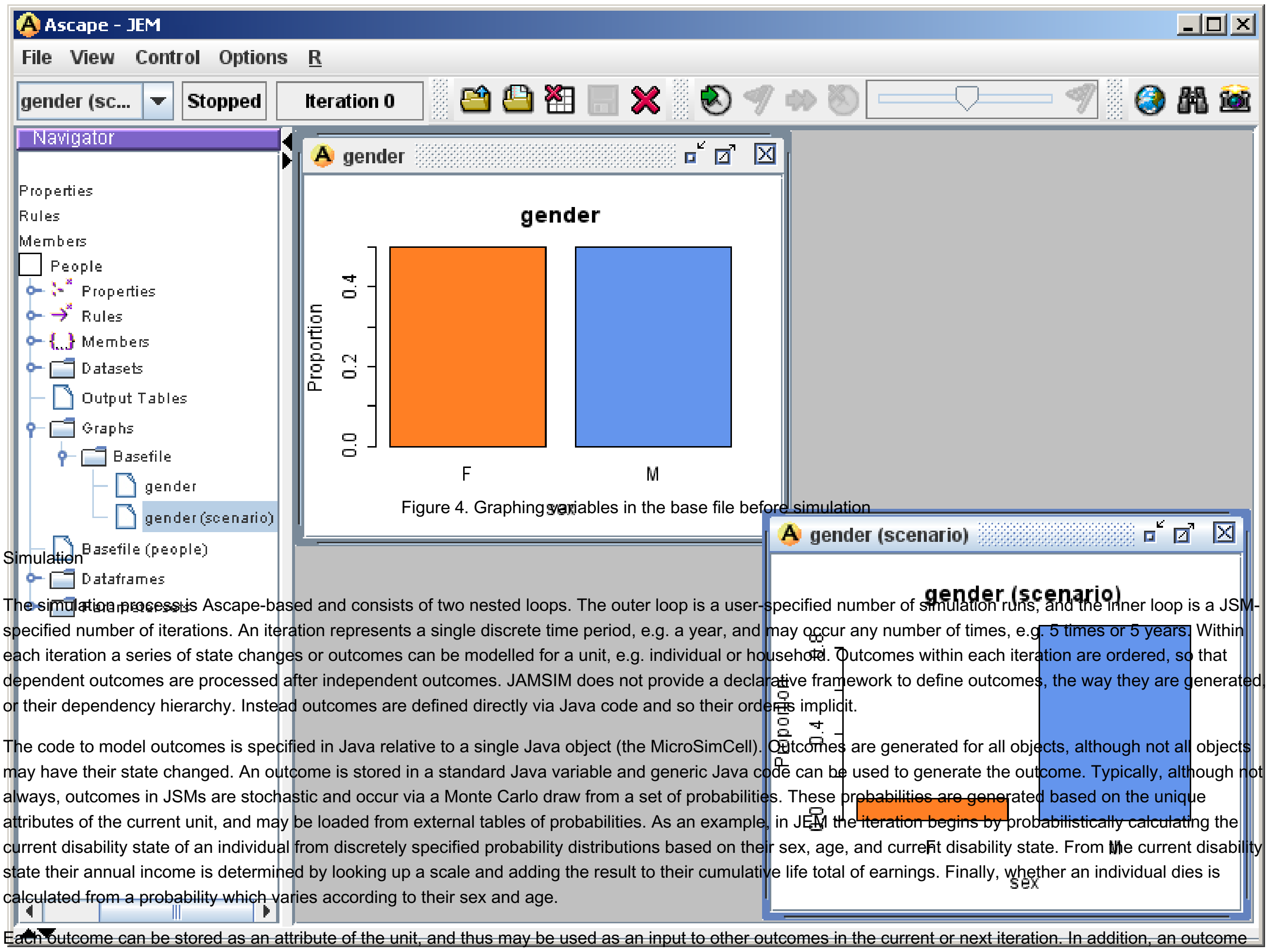


Figure 4. Graphing variables in the base file before simulation

Simulation

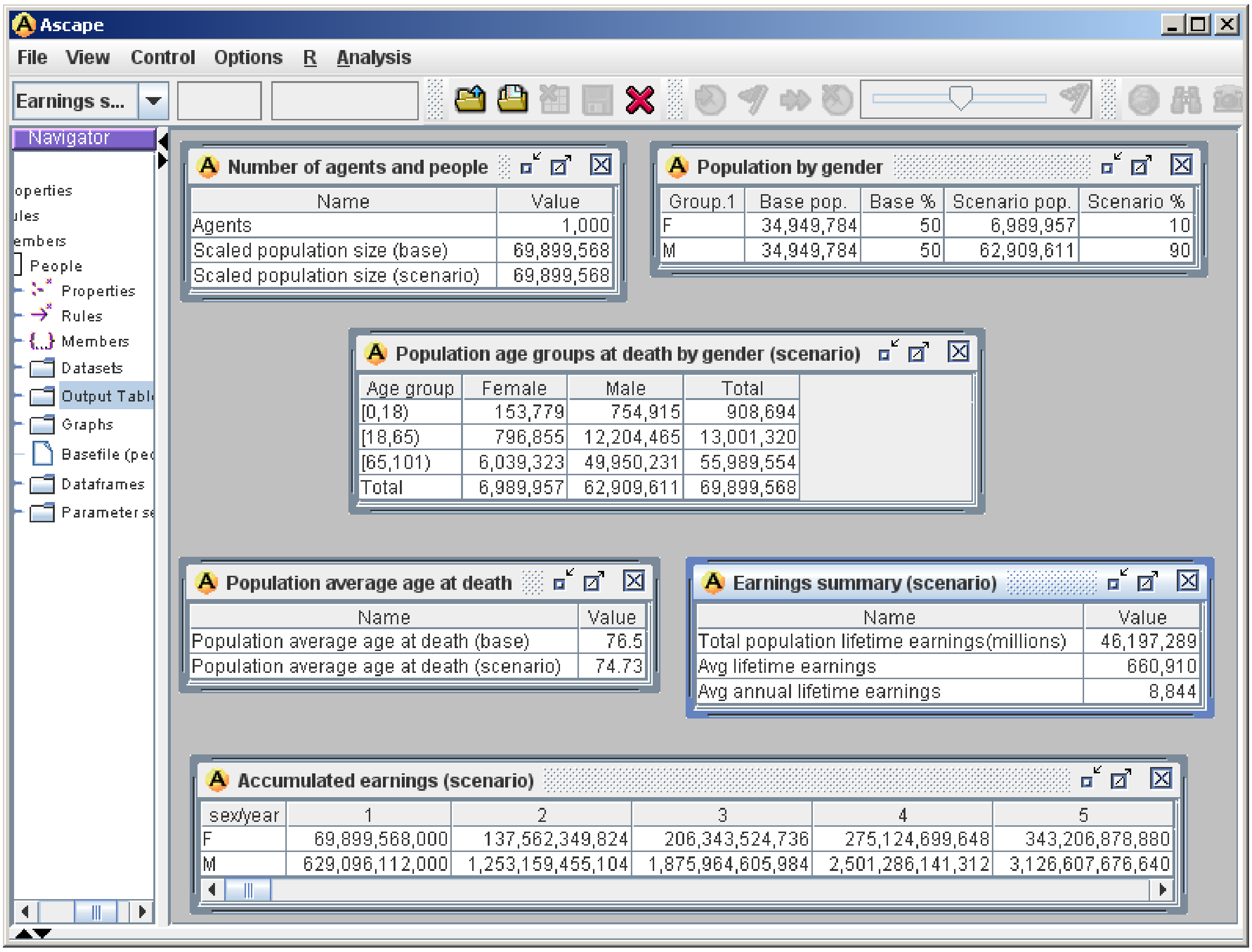
7.6 The simulation process is Ascape-based and consists of two nested loops. The outer loop is a user-specified number of simulation runs, and the inner loop is a JSM-specified number of iterations. An iteration represents a single discrete time period, e.g. a year, and may occur any number of times, e.g. 5 times or 5 years. Within each iteration a series of state changes or outcomes can be modelled for a unit, e.g. individual or household. Outcomes within each iteration are ordered, so that dependent outcomes are processed after independent outcomes. JAMSIM does not provide a declarative framework to define outcomes, the way they are generated, or their dependency hierarchy. Instead outcomes are defined directly via Java code and so their order is implicit.

7.7 The code to model outcomes is specified in Java relative to a single Java object (the MicroSimCell). Outcomes are generated for all objects, although not all objects may have their state changed. An outcome is stored in a standard Java variable and generic Java code can be used to generate the outcome. Typically, although not always, outcomes in JSMs are stochastic and occur via a Monte Carlo draw from a set of probabilities. These probabilities are generated based on the unique attributes of the current unit, and may be loaded from external tables of probabilities. As an example, in JEM the iteration begins by probabilistically calculating the current disability state of an individual from discretely specified probability distributions based on their sex, age, and current disability state. From the current disability state their annual income is determined by looking up a scale and adding the result to their cumulative life total of earnings. Finally, whether an individual dies is calculated from a probability which varies according to their sex and age.

7.8 Each outcome can be stored as an attribute of the unit, and thus may be used as an input to other outcomes in the current or next iteration. In addition, an outcome may be stored in an outcome array as a series across all iterations. This preserves the outcome's value in all previous iterations, rather than overwriting it on each iteration, so that it can be used to produce per-iteration results. After each iteration, the entire set of MicroSimCells is output to R as a dataframe. Output tables, showing frequencies or means, and graphs, may be generated between iterations (in either Java or R) and displayed to provide in-progress results.

Outputs

7.9 A simulation run is a single run through all iterations. After a run, the set of MicroSimCells will be in their final state and will include any outcome arrays containing results from all iterations. A set of run results may then be output and stored in R before the MicroSimCells are reset and the run is repeated. At the end of all runs, the individual run results can be collated in R or Java and the mean of results across all runs can be displayed in tables or graphs.



Number of agents and people

Name	Value
Agents	1,000
Scaled population size (base)	69,899,568
Scaled population size (scenario)	69,899,568

Population by gender

Group.1	Base pop.	Base %	Scenario pop.	Scenario %
F	34,949,784	50	6,989,957	10
M	34,949,784	50	62,909,611	90

Population age groups at death by gender (scenario)

Age group	Female	Male	Total
[0,18)	153,779	754,915	908,694
[18,65)	796,855	12,204,465	13,001,320
[65,101)	6,039,323	49,950,231	55,989,554
Total	6,989,957	62,909,611	69,899,568

Population average age at death

Name	Value
Population average age at death (base)	76.5
Population average age at death (scenario)	74.73

Earnings summary (scenario)

Name	Value
Total population lifetime earnings(millions)	46,197,289
Avg lifetime earnings	660,910
Avg annual lifetime earnings	8,844

Accumulated earnings (scenario)

sex/year	1	2	3	4	5
F	69,899,568,000	137,562,349,824	206,343,524,736	275,124,699,648	343,206,878,880
M	629,096,112,000	1,253,159,455,104	1,875,964,605,984	2,501,286,141,312	3,126,607,676,640

Figure 5. End of simulation output tables

7.10 Figure 5 shows the following single run table outputs from JEM:

- Number of agents and people
    - A summary table which shows the total number of agents, or cases, simulated—in this case 1,000. These are scaled up to a population size of 69,899,568 for both the base simulation and the scenario.
- Population by gender
    - A breakdown of the population by gender, for both the base simulation and a scenario in which 10% of the population are female and 90% are male.
- Population age groups at death by gender (scenario)
    - A cross tabulation showing the age group at death by gender.
- Population average age at death
    - The average age at death of the base and scenario populations.
- Earnings summary (scenario)
    - A summary of total and average earnings for the scenario population.
- Accumulated earnings (scenario)
    - The accumulated earnings per year by gender.

7.11 Likewise, output data can also be graphed using the functionality of R graphics. Figure 6 displays an age-sex pyramid, a bar graph of earnings by age group and gender for the scenario population, and line graphs which can be used to compare baseline and scenario accumulated earnings over the life course.

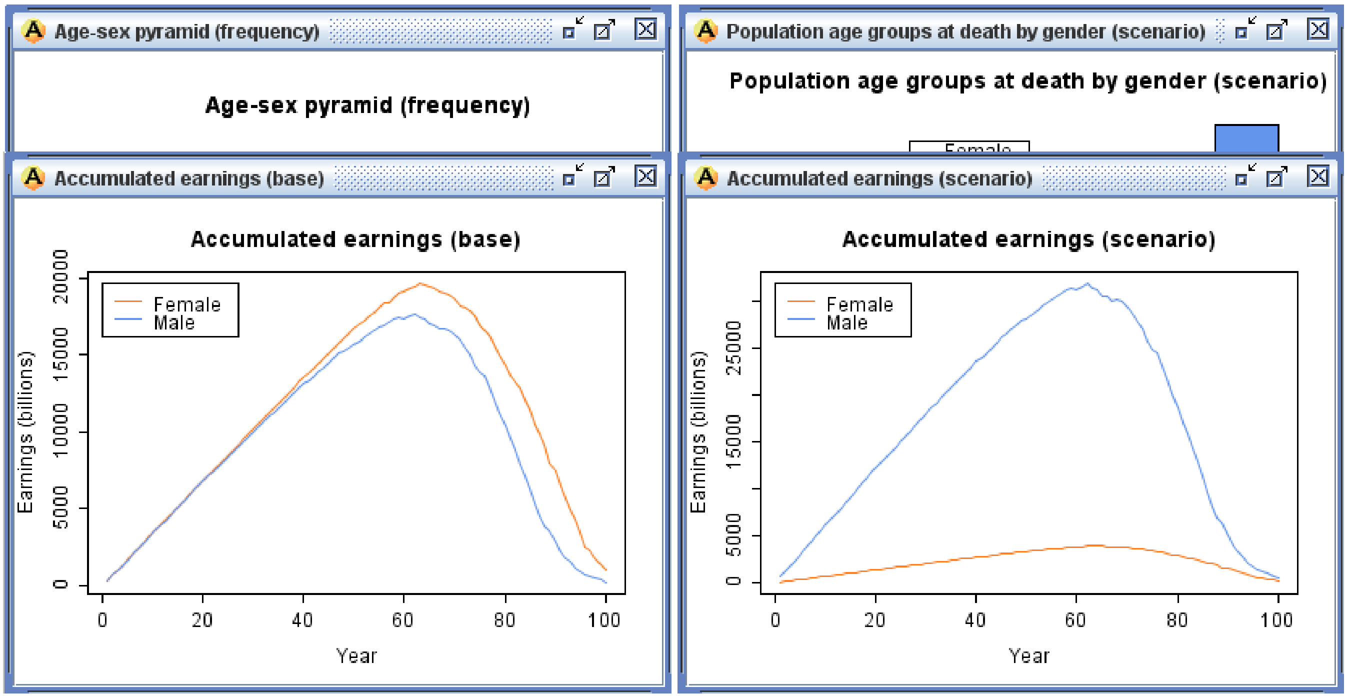

Age-sex pyramid (frequency)
Age-sex pyramid (frequency)
Population age groups at death by gender (scenario)
Population age groups at death by gender (scenario)
Female
Accumulated earnings (base)
Accumulated earnings (base)
Female
Male
Earnings (billions)
0
5000
10000
15000
20000
0
20
40
60
80
100
Year
Accumulated earnings (scenario)
Accumulated earnings (scenario)
Female
Male
Earnings (billions)
0
5000
15000
25000
0
20
40
60
80
100
Year

Figure 6. Output graphs including baseline comparison vs. scenario (weighted)

Limitations and future enhancements

8.1 JAMSIM has been built for models that run within the capacity and time constraints of a typical modern desktop machine. There is no hard upper limit to population sizes or the numbers of variables that may be used—instead, the main restriction is the amount of memory available. JAMSIM has been comfortably used on an Intel Core 2 Duo machine running Windows XP, with 4GB of RAM. As an example, in one simulation with a population of 80,000 units and 35 variables, the JAMSIM process running on this machine consumed 250MB of memory. The simulation consisted of 4 iterations in which 10 transitions were calculated every iteration. A single run took 9 seconds.

8.2 JAMSIM uses the same discrete-time simulation loop as Ascape. Because of this, it does not support some of the more comprehensive discrete-event scheduling mechanisms. For example, JAMSIM does not have an event queue and cannot generate events that occur on state changes. More generally, JAMSIM does not support continuous-time changes. Continuous-time allows state change events to occur at any time, rather than at a fixed time interval, and enables them to be triggered by other changes. For example, a fertility event can be re-computed whenever a partnering event occurs, using the new status immediately rather than waiting until the next cycle. This allows for more flexibility in the dependency of processes, and can better approximate the real world (for more on this see Scott 2003:15). While continuous-time changes and the associated use of survival functions has been prominent in some well-known microsimulation models and toolkits (e.g. Modgen), it has not been a feature of the health-related models we have developed.

8.3 JAMSIM does not provide a comprehensive set of domain level modules, e.g. marriage/partnering, migration, or labour market modules. Again, this is largely because the types of health situations we have modelled have not required these modules. These types of modules can be implemented in JAMSIM but the lack of any pre-existing code may make JAMSIM less appealing to those performing more general life-cycle dynamic microsimulation. Nor does JAMSIM provide other functionality typical of these types of simulations, in particular alignment functionality or a facility for behavioural feedback loops.

8.4 One of the key features of JAMSIM is the ability to test different scenarios. However, this is somewhat underdeveloped. Currently it is possible to reweight results, or to change continuous and categorical variables on individuals in the base file to produce a single scenario. Future development will include the ability to generate, save and load multiple scenarios, and to compare them with the baseline and with one another.

8.5 JAMSIM lacks a declarative framework for model variables and instead they are specified directly as Java variables and R objects. The problem this poses is that as the model grows it becomes difficult to keep track of all the instances where a variable is used, which increases the time it takes to make model changes. The disadvantage of a declarative framework is that it can add an extra level of runtime overhead. However, computing time is generally cheaper than programmer time and so future work on JAMSIM will involve development of a declarative framework which will allow model variables to be parameterised and specified in external parameter files rather than hard coded.

8.6 JAMSIM has been developed to be used by policy end-users, but data agreements may not allow such users access to the original data set. JAMSIM does not provide base file encryption functionality, which would in any case only offer a moderate level of protection as at some point the base file would need to be decrypted to be used. Instead the approach we plan to take is to use a synthetic base file that does not represent any real individuals but has distributions of relevant characteristics that are similar to a desired population. A safe alternative would be to only offer access to the application remotely in a secured environment. This would require establishing and maintaining the appropriate server infrastructure.

8.7 While JAMSIM combines R and Java, model outcomes to date have only been implemented in Java. R has only been used for results generation and graphical output. This underutilises the vast array of existing R code and packages that are useful for modelling outcomes, for example, the ability to generate outcomes from logistic and other types of regression models. Many of these R packages have underdeveloped corresponding open source packages in Java, or none at all, and replicating them would require significant work. In addition, modelling outcomes in R makes the transition to microsimulation easier for statisticians who may not be familiar with Java. For these reasons, development is currently underway to move the simulation loop and the modelling of outcomes into R. The Java components will be retained and used specifically for the user interface, as existing R user interfaces are relatively undeveloped and less attractive.

8.8 Other future plans include the migration of JAMSIM from the Ascape Swing GUI to the alternative Eclipse Rich Client Platform (RCP). The Eclipse RCP is a desktop application environment that provides a high-quality native looking GUI (McAffer and Lemieux 2006). The transition to the Eclipse RCP is planned because the current Ascape user interface has some limitations in terms of usability. For example, the Navigator tree used to access the components of a model is unordered and the hierarchy is unintuitive. Furthermore, some of the Ascape agent-based modelling features are present but are not used by JAMSIM and ought to be removed for a cleaner user interface. In contrast, an Eclipse-based user interface will allow more direct control over all user interface components. In addition, Eclipse incorporates the robust OSGI component model which supports modular and extensible plug-ins. This increases reusability of software components, contributing to reduced error rates and lower development costs. Overall, the Eclipse RCP environment will make the development of additional output features, such as a single scrollable window containing multiple tables and graphs, much easier.

8.9 Prototypes of these enhancements are currently under development and are being tested in a dynamic microsimulation model that simulates health and education outcomes for children. The implementation uses a base file of 1,100 children derived from a longitudinal study. For each individual it simulates child, parental and family factors and from these a set of final health and education outcomes. Intermediate factors and final outcomes are generated from probabilities derived from binomial, negative binomial and Poisson regression models. In addition, any of the variables used in the model can be altered to test a particular scenario and its influence on outcomes.

Conclusion

9.1 JAMSIM combines relevant components from open source packages to provide an environment and features for the development of dynamic discrete-time microsimulation models and their use by non-technical, policy-oriented users. It has been designed to be as flexible as possible, and a major strength is its open source nature, which gives it the potential for further enhancement by others in the modelling community.

Acknowledgements

The completed 'Primary Care in an Ageing Society' project was funded by the Health Research Council of New Zealand. The in-progress 'Modelling the Early Life Course' project is being funded by the Ministry of Science and Innovation. We thank all project team members. We would also like to acknowledge the constant, ongoing and ever responsive support of Miles Parker during the development of JAMSIM, and the very helpful comments of the anonymous reviewers of this paper. Finally, we are indebted to Martin von Randow for his meticulous proof reading of the final draft of this paper.

References

BROWN, Laurie and Harding, Ann (2002), 'Social Modelling and Public Policy: Application of Microsimulation Modelling in Australia', *Journal of Artificial Societies and Social Simulation,* 5 (4), 6 http://jasss.soc.surrey.ac.uk/5/4/6.html.

DAVIS, Peter, Lay-Yee, Roy, and Pearson, Janet (2010), 'Using micro-simulation to create a synthesised data set and test policy options: the case of health service effects under demographic ageing', *Health Policy,* 97 (2), 267-74. [doi:10.1016/j.healthpol.2010.05.014]

EVANDROU, Maria, et al. (2007), 'The SAGE Model : A Dynamic Microsimulation Population Model for Britain', in Anil Gupta and Ann Harding (eds.), *Modelling Our Future: Population Ageing, Health and Aged Care* (Amsterdam, The Netherlands: Elsevier).

GAMPE, Jutta, et al. (2009), 'The Microsimulation Tool of the MicMac-Project', *2nd General Conference of the International Microsimulation Association*(Ottawa, Canada).

GILBERT, Nigel and Bankes, Steven (2002), 'Platforms and methods for agent-based modeling', *Proceedings of the National Academy of Sciences of the United States of America,* 99 (3), 7197-8. [doi:10.1073/pnas.072079499]

GILBERT, Nigel and Troitzsch, Klaus (2005), *Simulation for the social scientist* (2nd edn.; Maidenhead: Open University Press).

HIMMELSPACH, Jan and Uhrmacher, Adelinde M. (2007), 'Plug'n simulate', *40th Annual Simulation Symposium*(Norfolk, Virginia, USA: IEEE), 137-43.

HOLM, Einar, et al. (2007), 'SVERIGE', in Anil Gupta and Ann Harding (eds.), *Modelling Our Future: Population Ageing, Health and Aged Care* (Amsterdam, The Netherlands: Elsevier).

HOLMER, Martin, Janney, Asa, and Cohen, Bob (2011), 'PENSIM Overview', (Office of Policy and Research, Employee Benefits Security Administration, U.S. Department of Labor).

IMMERVOLL, Herwig, O'Donoghue, Cathal, and Sutherland, Holly (1999), 'An Introduction to EUROMOD', *EUROMOD Working Papers*.

LAWSON, Tony (2008), 'Methods and Tools for the Microsimulation and Forecasting of Household Expenditure - A Review',*Technology and Social Change Working Papers*.

LEOMBRUNI, Roberto and Richiardi, Matteo (2006), 'LABORsim: An Agent-Based Microsimulation of Labour Supply—An Application to Italy',*Computational Economics,* 27 (1), 63-88. [doi:10.1007/s10614-005-9016-0]

MCAFFER, Jeff and Lemieux, Jean-Michel (2006), *Eclipse Rich Client Platform: Designing, Coding, and Packaging Javaª Applications*(Upper Saddle River, NJ: Addison-Wesley).

NIKOLAI, Cynthia and Madey, Gregory (2009), 'Tools of the trade: A survey of various agent based modeling platforms',*Journal of Artificial Societies and Social Simulation,* 12 (2), http://jasss.soc.surrey.ac.uk/12/2/2.html.

NORTH, Michael (2011a), 'Repast Simphony', http://repast.sourceforge.net/repast_simphony.html, accessed 7 March.

NORTH, Michael (2011b), 'Repast 3', http://repast.sourceforge.net/repast_3/, accessed 7 March.

O'DONOGHUE, Cathal, Lennon, John, and Hynes, Stephen (2009), 'The Life-cycle Income Analysis Model (LIAM): a study of a flexible dynamic microsimulation modelling computing framework', *International Journal of Microsimulation,* 2 (1), 16-31.

PARKER, Miles (2001), 'What is Ascape and Why Should You Care?', *Journal of Artificial Societies and Social Simulation,* 4 (1), 5 http://jasss.soc.surrey.ac.uk/4/1/5.html.

PARKER, Miles (2011), 'Ascape', http://ascape.sourceforge.net/, accessed 7 March.

PEARSON, Janet, et al. (2011), 'Primary Care in an Aging Society: Building and Testing a Microsimulation Model for Policy Purposes', *Social Science Computer Review,* 29 (1), 21-36. [doi:10.1177/0894439310370087]

PERCIVAL, Richard (2007), 'APPSIM-Software Selection and Data Structures',*NATSEM Working Papers* (Canberra).

R Development Core Team (2011), 'R: A Language and Environment for Statistical Computing', http://www.r-project.org/, accessed 7 March.

RAILSBACK, Steven, Lytinen, Steven, and Jackson, Stephen (2006), 'Agent-based simulation platforms: Review and development recommendations',*Simulation,* 82 (9), 609-09. [doi:10.1177/0037549706073695]

ROPELLA, Glen, Railsback, Steven, and Jackson, Stephen (2002), 'Software Engineering Considerations For Individual-Based Models',*Natural Resource Modeling,* 15 (1), 5-22. [doi:10.1111/j.1939-7445.2002.tb00077.x]

ROWE, Geoff and Gribble, Steve (2007), 'LifePaths Model', in Anil Gupta and Ann Harding (eds.), *Modelling Our Future: Population Ageing, Health and Aged Care* (Amsterdam, The Netherlands: Elsevier).

SAUERBIER, Thomas (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.

SCOTT, Anne (2003), 'A computing strategy for SAGE: 2. Programming considerations', (London: Citeseer).

SONNESSA, Michele (2003), 'JAS: Java Agent-based Simulation library. An open framework for algorithm-intensive simulations',*Workshop on Industrial and Labor Dynamics - The Agent-Based Computational Aproach* (Gandolfi 1999 edn.; Torino, Italy: World Scientific Publishing Co. Pte. Ltd.).

SONNESSA, Michele (2011), 'JAS: Java Agent-based Simulation Library', http://jaslibrary.sourceforge.net/, accessed 7 March.

STATISTICS CANADA (2011a), 'Modgen (Model generator)', http://www.statcan.gc.ca/microsimulation/modgen/modgen-eng.htm, accessed 7 March.

STATISTICS CANADA (2011b), 'Microsimulation approaches', http://www.statcan.gc.ca/microsimulation/modgen/new-nouveau/chap2/chap2-eng.htm, accessed 7 March.

SUTHERLAND, Holly (2007), 'EUROMOD - The Tax-Benefit Microsimulation Model for the European Union', in Anil Gupta and Ann Harding (eds.), *Modelling Our Future: Population Ageing, Health and Aged Care* (Amsterdam, The Netherlands: Elsevier).

SUTHERLAND, Holly, et al. (2008), 'Improving the Capacity and Usability of EUROMOD—Final Report', *EUROMOD Working Papers*.

TOBIAS, Robert and Hofmann, Carole (2004), 'Evaluation of free Java-libraries for social-scientific agent based simulation', *Journal of Artificial Societies and Social Simulation,* 7 (1), 6 http://jasss.soc.surrey.ac.uk/7/1/6.html.

UHRMACHER, Adelinde M., et al. (2011), 'JAMES II', http://wwwmosi.informatik.uni-rostock.de/mosi/projects/cosa/james-ii/, accessed 7 March.

WITTENBERG, Raphael, et al. (2007), 'PSSRU Long-Term Care Finance Model and CARESIM: Two Linked UK Models of Long-Term Care for Older People', in Anil Gupta and Ann Harding (eds.), *Modelling our future: population ageing health and aged care* (Amsterdam, The Netherlands: Elsevier).

XJ TECHNOLOGIES (2011), 'Anylogic Professional Edition', http://www.xjtek.com/anylogic/, accessed 7 March.

ZINN, Sabine, et al. (2009), 'MIC-CORE: A Tool for Microsimulation', *Winter Simulation Conference* (Austin, Texas, USA).