Agent-based Modeling using LSD

Marco VALENTE[1,2]

[1]LEM, S. Anna School of Advanced Studies, Pisa

[2]University of L'Aquila

ABM: a definition

Agent-based models are based on a **hierarchical** data structure subject to **time dynamics**.

- Low-level (component) entities have weak constraints and a degree of heterogeneity.
- High-level (aggregate) entities are at least partly influenced by the states of the low level entities.
- Asynchronous time dynamics independently operating on each variable.

ABM: a definition

Compared to mathematical models (system of equations), AB models are perceived as subject to fewer constraints and producing weaker results.

Actually, ABM's are subject to logical and temporal consistency, which impose strong constraints.

ABM: a definition

The difference between standard (mathematical) modeling and ABM concerns a subtle difference in the use of the term "equations":

1. In mathematical models equations are **a-temporal constraints** to be respected, not indicating explicitly an action. Solutions are a set of values satisfying all the constraints, but nothing can be said for values outside the solution(s).
2. In simulation models equations are **time-specific descriptions** of an agent's (or entity) action, with "solutions" being the time patterns generated. ABM's generate virtual histories, to be used to test hypotheses and synthetic conjectures.

ABM in Economics

In Economics ABM offer an alternative to maximization and equilibrium models, and hence are popular among heterodox scholars. ABM's have been used for a wide variety of topics: innovation, growth, organization theory, consumer theory, policy, etc.

ABM in Economics

In Economics ABM offer an alternative to maximization and equilibrium models, and hence are popular among heterodox scholars. ABM's have been used for a wide variety of topics: innovation, growth, organization theory, consumer theory, policy, etc.

ABM's are, however, strongly criticized, even by members of the community, for several reasons, limiting their diffusion and questioning their utility.

ABM in Economics

Some of these criticisms are deserved. The most relevant are listed below:

ABM in Economics

Some of these criticisms are deserved. The most relevant are listed below:

1. Difficulty to use;

ABM in Economics

Some of these criticisms are deserved. The most relevant are listed below:

1. Difficulty to use;
2. Lack of disclosure on the model contents;

ABM in Economics

Some of these criticisms are deserved. The most relevant are listed below:

1. Difficulty to use;
2. Lack of disclosure on the model contents;
3. Difficulty of replicating claimed results;

ABM in Economics

Some of these criticisms are deserved. The most relevant are listed below:

1. Difficulty to use;
2. Lack of disclosure on the model contents;
3. Difficulty of replicating claimed results;
4. Lack of accepted methodology to assess the results.

They will be addressed during the course proposing solutions for both the technical and methodological issues.

Course approach

Students will be required to develop and use simulation programs for increasingly complex models. This will entail learning to implement simulation models and to extract scientifically relevant knowledge from the exercise.

Each lesson will present a model entailing increasing technical difficulty. Students will implement the model on their own (and at their pace) following the instructions from online instructions.

The course is expected to benefit also students not planning to use ABM in their research, since the computational logic is one of the tools relevant for the discipline along math, statistics, law, history etc.

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Outline

- **Definitions**: a normal form of simulation models.

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Outline

- **Definitions**: a normal form of simulation models.
- **Introduction to Lsd**: equations, structures and configurations of models.

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Outline

- **Definitions**: a normal form of simulation models.
- **Introduction to Lsd**: equations, structures and configurations of models.
- **Tutorials**: implementation of example models: linear, random walk, replicator dynamics, ...

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Outline

- **Definitions**: a normal form of simulation models.
- **Introduction to Lsd**: equations, structures and configurations of models.
- **Tutorials**: implementation of example models: linear, random walk, replicator dynamics, ...
- **Lectures**: presentation of papers based on agent-based models.

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Outline

- **Definitions**: a normal form of simulation models.
- **Introduction to Lsd**: equations, structures and configurations of models.
- **Tutorials**: implementation of example models: linear, random walk, replicator dynamics, ...
- **Lectures**: presentation of papers based on agent-based models.
- **Methodology**: discuss methodological issues in economics.

Why using simulations?

The methodology for using simulations in social sciences is hotly debated. But is also highly relevant, since it is at the base of many technical and theoretical issues.

Why using simulations?

The methodology for using simulations in social sciences is hotly debated. But is also highly relevant, since it is at the base of many technical and theoretical issues.

Though not discussing formally the methodology we will suggest the approach underlying the LSD design.

Methodological assumptions

1. Research models must **not** replicate the reality, necessarily.

Methodological assumptions

1. Research models must **not** replicate the reality, necessarily.
2. Research models must **explain** reality.

Methodological assumptions

A few comments are required:

1. We talk about research simulation models. Other applications of simulation models (e.g. forecasting) may require a different methodology.

Methodological assumptions

A few comments are required:

1. We talk about research simulation models. Other applications of simulation models (e.g. forecasting) may require a different methodology.
2. The pivotal concept is what we mean by *reality*. Economic events are ill-defined, lacking direct measures, and never replicated identically through history.

Methodological assumptions

A few comments are required:

1. We talk about research simulation models. Other applications of simulation models (e.g. forecasting) may require a different methodology.
2. The pivotal concept is what we mean by *reality*. Economic events are ill-defined, lacking direct measures, and never replicated identically through history.
3. *Explaining* needs a formal definition, though we will skip this and rely on the intuitive meaning of the term.

Methodological assumptions

We sustain that the scientific use of simulation models consist in four steps:

1. Build a simplified representation of a reality.

Methodological assumptions

We sustain that the scientific use of simulation models consist in four steps:

1. Build a simplified representation of a reality.
2. Ensure that the model generates data compatible with (some of) the properties observed in the real world.

Methodological assumptions

We sustain that the scientific use of simulation models consist in four steps:

1. Build a simplified representation of a reality.
2. Ensure that the model generates data compatible with (some of) the properties observed in the real world.
3. Find interesting explanations of simulated events, as if analysing the record of a virtual history.

Methodological assumptions

We sustain that the scientific use of simulation models consist in four steps:

1. Build a simplified representation of a reality.
2. Ensure that the model generates data compatible with (some of) the properties observed in the real world.
3. Find interesting explanations of simulated events, as if analysing the record of a virtual history.
4. Evaluate whether the same explanations apply to the real world cases, too.

Methodological assumptions

Notice that the comparison between virtual and observed data (*validation*) is of relatively lesser importance, being only one step in the use of simulation modeling for research focused on finding interesting *explanations*.

The concept of explanation can be formally defined and, used as synonym of knowledge scientifically assessed.

Simulation models vs programs

Simulation models and simulation programs are distinct entities.

Simulation models vs programs

Simulation models and simulation programs are distinct entities.

Simulation **models** as abstract, logical constructs, much like a theorem or a mathematical system of equations. They are defined by logical/mathematical operations located in time.

Simulation models vs programs

Simulation models and simulation programs are distinct entities.

Simulation **models** as abstract, logical constructs, much like a theorem or a mathematical system of equations. They are defined by logical/mathematical operations located in time.

A simulation **program** is one of the ways to generate the implicit outcomes of the model, and requires a large amount of technical software.

Topics of the course

Using a standard programming language the most difficult task for modelers of ABM is *not* the coding of the model. Rather it is the coding of ancillary tools necessary to declare the model's elements, assign initial values, analyse the state of the model, interpret and export results, etc.

Topics of the course

Using a standard programming language the most difficult task for modelers of ABM is *not* the coding of the model. Rather it is the coding of ancillary tools necessary to declare the model's elements, assign initial values, analyse the state of the model, interpret and export results, etc.

Using LSD, contrary to most languages, the modeller supplies only the definitions of the elements in the model. The system automatically produces professional tools to control and access any aspect of the model required for its scientific use.

Topics of the course

The stages for using a model for research, discussed during the course, are the following:

- **Design**: decide what the model should be like to contribute to a research project.

Topics of the course

The stages for using a model for research, discussed during the course, are the following:

- **Design**: decide what the model should be like to contribute to a research project.
- **Implementation**: turning an abstract idea into a working simulation program.

Topics of the course

The stages for using a model for research, discussed during the course, are the following:

- **Design**: decide what the model should be like to contribute to a research project.
- **Implementation**: turning an abstract idea into a working simulation program.
- **Interpretation**: extracting knowledge from simulation models.

Topics of the course

The stages for using a model for research, discussed during the course, are the following:

- **Design**: decide what the model should be like to contribute to a research project.
- **Implementation**: turning an abstract idea into a working simulation program.
- **Interpretation**: extracting knowledge from simulation models.
- **Revision**: the implementation must always proceed gradually, revising and extending previous code.

Topics of the course

The stages for using a model for research, discussed during the course, are the following:

- **Design**: decide what the model should be like to contribute to a research project.
- **Implementation**: turning an abstract idea into a working simulation program.
- **Interpretation**: extracting knowledge from simulation models.
- **Revision**: the implementation must always proceed gradually, revising and extending previous code.
- **Documentation**: simulation results must be properly formatted to report (and support) scientific claims.

Topics of the course

In the rest of this introductory talk we will address the following issues:

1. Define a normal form for simulation models.
2. Describe the LSD overall structure and introduce its interfaces.

Definition of simulation models

A simulation model is defined independently from the medium implementing it. We need a definition of simulation model such that to perfectly identify the results produced by running the model.

Definition of simulation models

A simulation model is defined independently from the medium implementing it. We need a definition of simulation model such that to perfectly identify the results produced by running the model.

Simulation model: generic definition of how a set of time-indexed variables is computed:

$$X_t = f_X(X_{t-k}, Y_t, \alpha), Y_t = f_Y(...), Z_t = f_Z(...)$$

Definition of simulation models

A simulation model is defined independently from the medium implementing it. We need a definition of simulation model such that to perfectly identify the results produced by running the model.

Simulation model: generic definition of how a set of time-indexed variables is computed:

$$X_t = f_X(X_{t-k}, Y_t, \alpha), Y_t = f_Y(...), Z_t = f_Z(...)$$

Simulation results: sequence(s) of values across simulation time steps:

$$\{X_1, X_2, ..., X_t, ..., X_T\}, \{Y_1, Y_2, ..., Y_t, ..., Y_T\}, \{Z_1, Z_2, ..., Z_t, ..., Z_T\}$$

Definition of simulation models

Continuous time models are rarely relevant in ABM, and, in any case, are necessarily solved by means of (discrete) numerical solutions.

Notice that we choose to refer to *time driven* simulation models, as opposed to *event driven* models. The two styles of modelling are equivalent, since they can be turned into one another.

Definition of simulation models

The definition of simulation model we provide is meant to satisfy a few requirements:

1. **Univocal**. The definition must be unambiguous, making impossible to include different implementations of the same model generating different (or undetermined) results.

Definition of simulation models

The definition of simulation model we provide is meant to satisfy a few requirements:

1. **Univocal**. The definition must be unambiguous, making impossible to include different implementations of the same model generating different (or undetermined) results.
2. **User friendly**. It must be as close as possible to (one of) the way(s) people usually refer to models in natural language.

Definition of simulation models

The definition of simulation model we provide is meant to satisfy a few requirements:

1. **Univocal**. The definition must be unambiguous, making impossible to include different implementations of the same model generating different (or undetermined) results.
2. **User friendly**. It must be as close as possible to (one of) the way(s) people usually refer to models in natural language.
3. **Easy to edit**. Implementing a model is a continuous process of unplanned revisions of existing code, thus the implementation needs to allow changes without effort.

Definition of simulation models

The definition of simulation model we provide is meant to satisfy a few requirements:

1. **Univocal**. The definition must be unambiguous, making impossible to include different implementations of the same model generating different (or undetermined) results.
2. **User friendly**. It must be as close as possible to (one of) the way(s) people usually refer to models in natural language.
3. **Easy to edit**. Implementing a model is a continuous process of unplanned revisions of existing code, thus the implementation needs to allow changes without effort.
4. **Scalable**. ABM models frequently require large dimensions, hence the implementation should technically allow for large scale models.

Definition of simulation models

In the following we list the elements composing a simulation model, providing definitions such that no ambiguity is left about the simulation results produced with the model.

The elements proposed as elementary components are the following:

1. **Variables**
2. **Parameters**
3. **Functions**
4. **Objects**

We will show that no other information is necessary in order to define a model.

Variables

Variables are labels, or symbols, that at ***each time step*** are associated to one and only one numerical value.
The numerical value of a variable is computed executing an *equation*, defined as any computational elaboration of the values of some elements defined in the model.

$$X_t = f_X(...)$$

The equation $f_X(...)$ may contain any legal computational expression.

Parameters

Parameters are labels associated to numerical values.
Parameters do not change value of their own accord.

α

Functions

Functions are, like variables, numerical values computed as result of an equation. However, the values generated by functions are not associated to time steps, but are computed **on request** during the execution of other equations.

$$X = f(...)$$

Notice that functions provide values used only internally because they cannot be saved as results because the same function may produce several values, or none, at the same time step.

Objects

In almost all cases a model is designed to contain many copies, or instances, of variables, parameters and functions. They share the same label and properties (i.e. equations) but are distinguished from one another.

In mathematical format we normally use vectors to store multiple elements, using the same label with different indexes to refer to each member of a given set:

$$\vec{X} = \{X^1, X^2, ..., X^i, ..., X^n\}$$
$$\vec{X} = \{Y^1, Y^2, ..., Y^i, ..., Y^n\}$$
$$\vec{Z} = \{Z^1, Z^2, ..., Z^i, ..., Z^n\}$$

Objects

However, in “hierarchical” models, vector-based representations are extremely annoying. Here are two examples of practical difficulties.

Consider a variable referred to a firm (among many) operating in a market (among many). The model will then refer to this variable using two indexes for the firm and the market containing. Extending the model to countries would require adding a third index to each and every position in the code referring to the variables of firms.

Troubles emerge also when we deal with models of dynamic sets. Adding a new firm to a market requires to extend all the vectors referring to this entity. And adding a variable requires changes to each position adding or removing firms.

Objects

Programming languages have developed a more powerful concept, that includes vectors, but it is far more general: objects.

Objects are containers, storing together different types of elements that are, somehow, forming an identifiable unit. Programming using objects is far simpler than using vectors. Moreover, objects are particularly useful for simulations, since the unit representing an object can easily be associated to the real-world entity that the model refers to.

Definition of simulation models

Object-based representations are equivalent to a matrix-based representation.

		Object-based			
		$ObOne^1$	$ObOne^2$	...	$ObOne^N$
Vector-based	$\vec{X}$	X^1	X^2	...	X^N
	$\vec{Y}$	Y^1	Y^2	...	Y^N
	$\vec{\alpha}$	α^1	α^2	...	α^N
	$Ob\vec{Two}$	$ObTwo^1$	$ObTwo^2$	...	$ObTwo^N$

Object-based representations are far more flexible than vectors, easily expressing, for example, the equivalent of nested matrices and matrices with different number of rows in each column.

Model Structure

In summary, we can call the **structure** of a model the set of the following elements:

1. **Variables**. Symbols associated to a single value at each time step, computed according to a specified equation.
2. **Parameters**. Symbols associated to values not changing of their own accord.
3. **Functions**. Symbols providing values computed by an equation on request by other equations (independently from the time).
4. **Objects**. Units containing a set of other elements.

Model Configuration

The structure of a model is an abstract description of its elements, defining generically how the values of a generic time step t can be computed on the base of the values inherited from previous time steps $t-1, t-2, ...$

When we start the simulation the values of the model at time $t < 1$ are not available, and therefore must be provided by the user.

The same model structure will then produce different results depending on the numerical values assigned at $t = 0$. Let's see which numerical values for each type of element can affect the results. Call the set of relevant values the **initialization** of a model.

Model Configuration

A first part of the initialization is the **number of objects**, since it also determines the number of other elements.

Notice that the assignment of objects' numbers may be quite elaborated, with different number of entities for different "branches" of the model.

Model Configuration

Obviously, every parameter must be assigned an initial value. But also, possibly, some variables and functions may require one or more values.

Model Configuration

Obviously, every parameter must be assigned an initial value. But also, possibly, some variables and functions may require one or more values.

Consider the equation
$X_t = Y_{t-1} + \alpha$

At time $t = 1$, the very first step of the simulation, the equation becomes:
$X_1 = Y_0 + \alpha$
Y_0 cannot be produced by the model, since 1 is the first time step. Consequently, the modeller that must assign to Y a *lagged* (or past) value for Y as part of the initialization of the model.

Model Configuration

An equation may also require more than one lag. Consider, for example, the following equation:

$$X_t = Y_{t-3} + \alpha$$

For the first 3 time steps the model requires the values of Y_{-2}, Y_{-1} and Y_0, and therefore the user must assign three lagged values to Y in order to avoid ambiguities.

Model Configuration

An equation may also require more than one lag. Consider, for example, the following equation:

$$X_t = Y_{t-3} + \alpha$$

For the first 3 time steps the model requires the values of Y_{-2}, Y_{-1} and Y_0, and therefore the user must assign three lagged values to Y in order to avoid ambiguities.

Functions also may require “lagged” values, though they do not refer to previous time steps, but to previous calls, or executions, of the function’s equation.

Definition of simulation models

A simulation model is therefore univocally defined (i.e. producing the same results) once we describe the following elements:

- **Equations**: set of routines to compute values for each variable and function in the model

Definition of simulation models

A simulation model is therefore univocally defined (i.e. producing the same results) once we describe the following elements:

- **Equations**: set of routines to compute values for each variable and function in the model
- **Configuration:**
  - **Structure**: list of variables, parameters, functions each positioned within a set of hierarchically related objects.

Definition of simulation models

A simulation model is therefore univocally defined (i.e. producing the same results) once we describe the following elements:

- **Equations**: set of routines to compute values for each variable and function in the model
- **Configuration:**
  - **Structure**: list of variables, parameters, functions each positioned within a set of hierarchically related objects.
  - **Initialization**: number of objects, values for parameters, lagged values for variables and functions

Definition of simulation models

A simulation model is therefore univocally defined (i.e. producing the same results) once we describe the following elements:

- **Equations**: set of routines to compute values for each variable and function in the model
- **Configuration:**
  - **Structure**: list of variables, parameters, functions each positioned within a set of hierarchically related objects.
  - **Initialization**: number of objects, values for parameters, lagged values for variables and functions
  - **Sim. settings**: num. of time steps, num. of simulation runs, pseudo-random sequences, visualization and saving options.

LSD simulation models

Any programming language can, in principle, implement any model. However, most languages (for ABM or generic) require also a lot of complex technical code to interact with the model, or pose rigid limitations on running or extending an existing model.

LSD allows users to generate a simulation program defining **only** the elements of a simulation model according to the format proposed above.

LSD provides **automatically** simulation programs complete with interfaces, debugger, graphics etc. allowing the full access to any relevant aspect of the model.

LSD Motivation

LSD is a complete suite dedicated to **design**, **implement**, **revise**, **analyse**, **document** and **re-use** agent-based simulation models for research purposes.

Writing and using a computer program for research purposes is completely different from the same activity in standard software development.

Programming vs. Simulating

	Software engineering	Research simulations

Programming vs. Simulating

	Software engineering	Research simulations
Programmers vs. users	Distinct people/roles/skills	Same people

Programming vs. Simulating

	Software engineering	Research simulations
Programmers vs. users	Distinct people/roles/skills	Same people
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very relevant

Programming vs. Simulating

	Software engineering	Research simulations
Programmers vs. users	Distinct people/roles/skills	Same people
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very relevant
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define and aggregate elementary components

Programming vs. Simulating

	Software engineering	Research simulations
Programmers vs. users	Distinct people/roles/skills	Same people
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very relevant
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define and aggregate elementary components
Implementation details	Hidden to final users	Required for assessment and re-use

Programming vs. Simulating

	Software engineering	Research simulations
Programmers vs. users	Distinct people/roles/skills	Same people
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very relevant
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define and aggregate elementary components
Implementation details	Hidden to final users	Required for assessment and re-use
Unexpected result	Bug, to be removed	Potentially relevant discovery, to be investigated

Programming vs. Simulating

	Software engineering	Research simulations
Programmers vs. users	Distinct people/roles/skills	Same people
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very relevant
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define and aggregate elementary components
Implementation details	Hidden to final users	Required for assessment and re-use
Unexpected result	Bug, to be removed	Potentially relevant discovery, to be investigated
Extending beyond original scope	Impossible, complexity crisis	Desirable/necessary

Programming vs. Simulating

	Software engineering	Research simulations
Programmers vs. users	Distinct people/roles/skills	Same people
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very relevant
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define and aggregate elementary components
Implementation details	Hidden to final users	Required for assessment and re-use
Unexpected result	Bug, to be removed	Potentially relevant discovery, to be investigated
Extending beyond original scope	Impossible, complexity crisis	Desirable/necessary
Summary	Black-box providing a well-defined and predictable output	Virtual world replicating the puzzles of reality *and* allowing their solving.

LSD Goal

A **theoretical model** is implemented in a **computer program** requiring a lot of highly sophisticated technical code to define, observe and control the model.

The goal of LSD is to allow modelers/users to work exclusively on the content of the model producing **automatically** all the interfaces required to access the model and without posing limitations to the kind of model.

Design

The key points of the LSD design are the following:

1. Extremely limited number of building blocks: *variable*s and *object*s.

Design

The key points of the LSD design are the following:

1. Extremely limited number of building blocks: *variable*s and *object*s.
2. Model as a set of discrete difference equations.

Design

The key points of the LSD design are the following:

1. Extremely limited number of building blocks: *variable*s and *object*s.
2. Model as a set of discrete difference equations.
3. Modular, self-assembling computational structure.

Design

The key points of the LSD design are the following:

1. Extremely limited number of building blocks: *variable*s and *object*s.
2. Model as a set of discrete difference equations.
3. Modular, self-assembling computational structure.
4. Automatic, context- and content-dependent high-performance interfaces.

Design

The key points of the LSD design are the following:

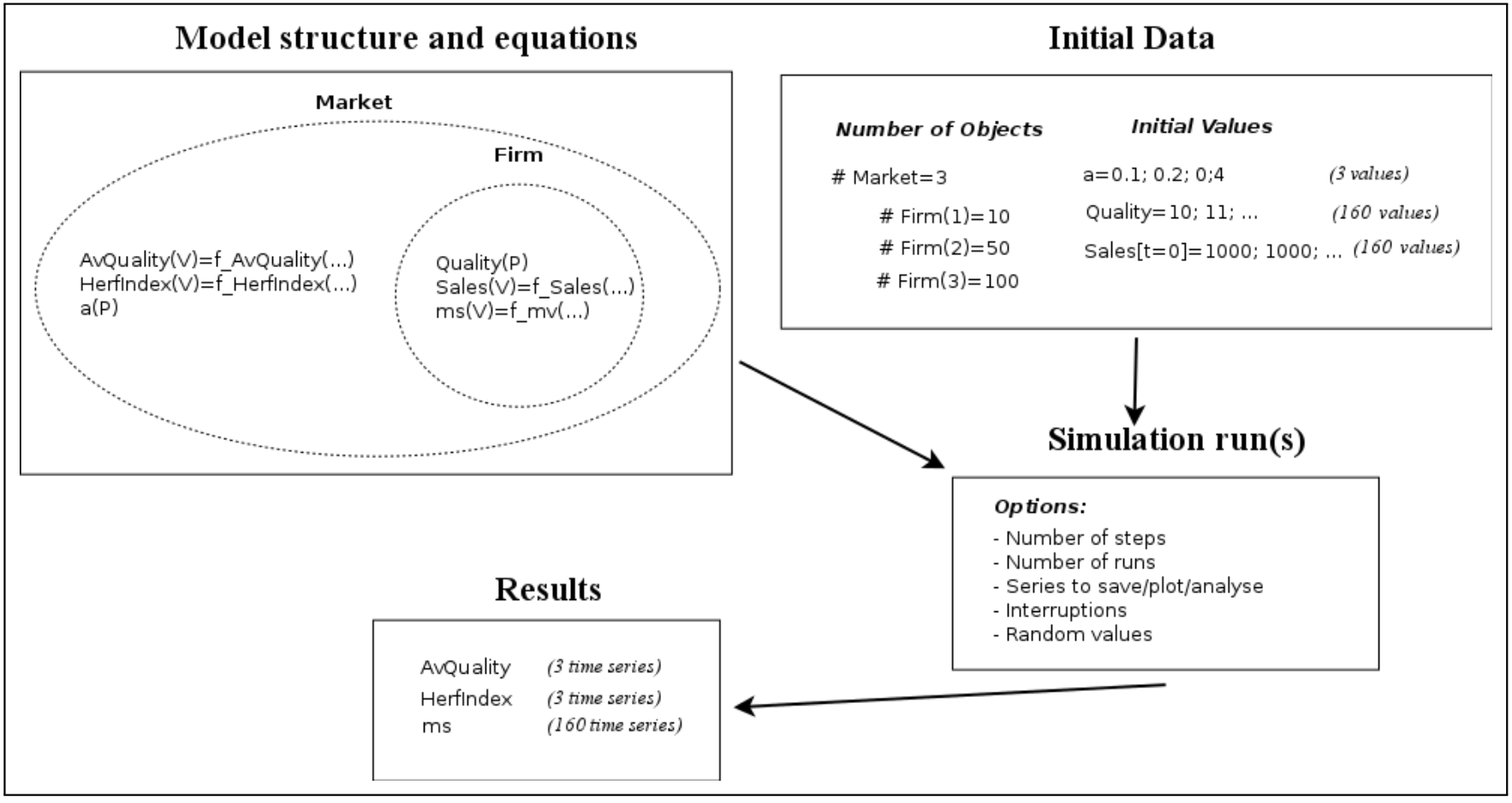
1. Extremely limited number of building blocks: *variable*s and *object*s.
2. Model as a set of discrete difference equations.
3. Modular, self-assembling computational structure.
4. Automatic, context- and content-dependent high-performance interfaces.
5. Efficient, powerful and multi-platform code (GNU C++).

LSD Components

A **model** is made of:

- **Variables' equations**: a chunk of code expressing how the generic instance of the variable updates its value at the generic time step.
- **A model structure**: sets of objects containing variables, parameters, or other objects;
- **Initial data**: numerical values to initialize the mode, such as the number of copies for each object and the values for parameters and variables at time *t=0*.
- **Sim.options**: number of steps, results to save, pseudo-random events, running modes, etc.

LSD components



LSD Simulation stages

1. Modelers are required to provide exclusively model-related information using graphical interfaces and intuitive commands (scripting language).

LSD Simulation stages

1. Modelers are required to provide exclusively model-related information using graphical interfaces and intuitive commands (scripting language).
2. The execution of simulation runs is completely automatic. The system assembles the available information and arranges automatically the required operations.

LSD Simulation stages

1. Modelers are required to provide exclusively model-related information using graphical interfaces and intuitive commands (scripting language).
2. The execution of simulation runs is completely automatic. The system assembles the available information and arranges automatically the required operations.
3. In case of errors (e.g. division by zero, missing elements, infinite loops) the system issues detailed reports.

LSD Simulation stages

1. Modelers are required to provide exclusively model-related information using graphical interfaces and intuitive commands (scripting language).
2. The execution of simulation runs is completely automatic. The system assembles the available information and arranges automatically the required operations.
3. In case of errors (e.g. division by zero, missing elements, infinite loops) the system issues detailed reports.
4. At any time users can interrupt the simulation to inspect the state of the model and analyse the time series produced.

LSD Simulation stages

1. Modelers are required to provide exclusively model-related information using graphical interfaces and intuitive commands (scripting language).
2. The execution of simulation runs is completely automatic. The system assembles the available information and arranges automatically the required operations.
3. In case of errors (e.g. division by zero, missing elements, infinite loops) the system issues detailed reports.
4. At any time users can interrupt the simulation to inspect the state of the model and analyse the time series produced.
5. An integrated module allows to manage even massive amounts of hierarchically-structured simulation results.

Output

1. Dataset containing the complete time series generated in a simulation run.

Output

1. Dataset containing the complete time series generated in a simulation run.
2. Multiple datasets, obtained by multiple replications of runs, automatically managed, for robustness and sensitivity tests.

Output

1. Dataset containing the complete time series generated in a simulation run.
2. Multiple datasets, obtained by multiple replications of runs, automatically managed, for robustness and sensitivity tests.
3. Graphical and statistical analysis, particularly suited for LSD data.

Output

1. Dataset containing the complete time series generated in a simulation run.
2. Multiple datasets, obtained by multiple replications of runs, automatically managed, for robustness and sensitivity tests.
3. Graphical and statistical analysis, particularly suited for LSD data.
4. Run-time analytical tools, including graphs, messages and custom controls.

Output

1. Dataset containing the complete time series generated in a simulation run.
2. Multiple datasets, obtained by multiple replications of runs, automatically managed, for robustness and sensitivity tests.
3. Graphical and statistical analysis, particularly suited for LSD data.
4. Run-time analytical tools, including graphs, messages and custom controls.
5. Intra-simulation dynamic analysis: advance step-by-step with full read-write access.

Output

1. Dataset containing the complete time series generated in a simulation run.
2. Multiple datasets, obtained by multiple replications of runs, automatically managed, for robustness and sensitivity tests.
3. Graphical and statistical analysis, particularly suited for LSD data.
4. Run-time analytical tools, including graphs, messages and custom controls.
5. Intra-simulation dynamic analysis: advance step-by-step with full read-write access.
6. User-defined output (compatible with C++ libraries).

Output

1. Dataset containing the complete time series generated in a simulation run.
2. Multiple datasets, obtained by multiple replications of runs, automatically managed, for robustness and sensitivity tests.
3. Graphical and statistical analysis, particularly suited for LSD data.
4. Run-time analytical tools, including graphs, messages and custom controls.
5. Intra-simulation dynamic analysis: advance step-by-step with full read-write access.
6. User-defined output (compatible with C++ libraries).
7. HTML automatic documentation: list of elements with hyperlinks to relevant information.

LSD major features

Major features of LSD are the following:

- **Universal**. LSD can implement any computational expression
- **User friendly**. Requires users to insert only model-relevant information expressed as discrete equations and using graphical interfaces.
- **Modular**. Users can revise any portion of the model and the system automatically updates the model program as required.
- **Powerful and scalable**. LSD is implemented as compiled C++ code, running on any system and fully exploiting available hardware.

LSD architecture

LSD implements with different tools the **equations** of the model and the rest, called generally **configurations**:

LSD architecture

LSD implements with different tools the **equations** of the model and the rest, called generally **configurations**:

- **Equations**: implemented in a power programming language (C++) using a stylized format (script). Each equation is a chunk of lines expressing the content of the equation.

LSD architecture

LSD implements with different tools the **equations** of the model and the rest, called generally **configurations**:

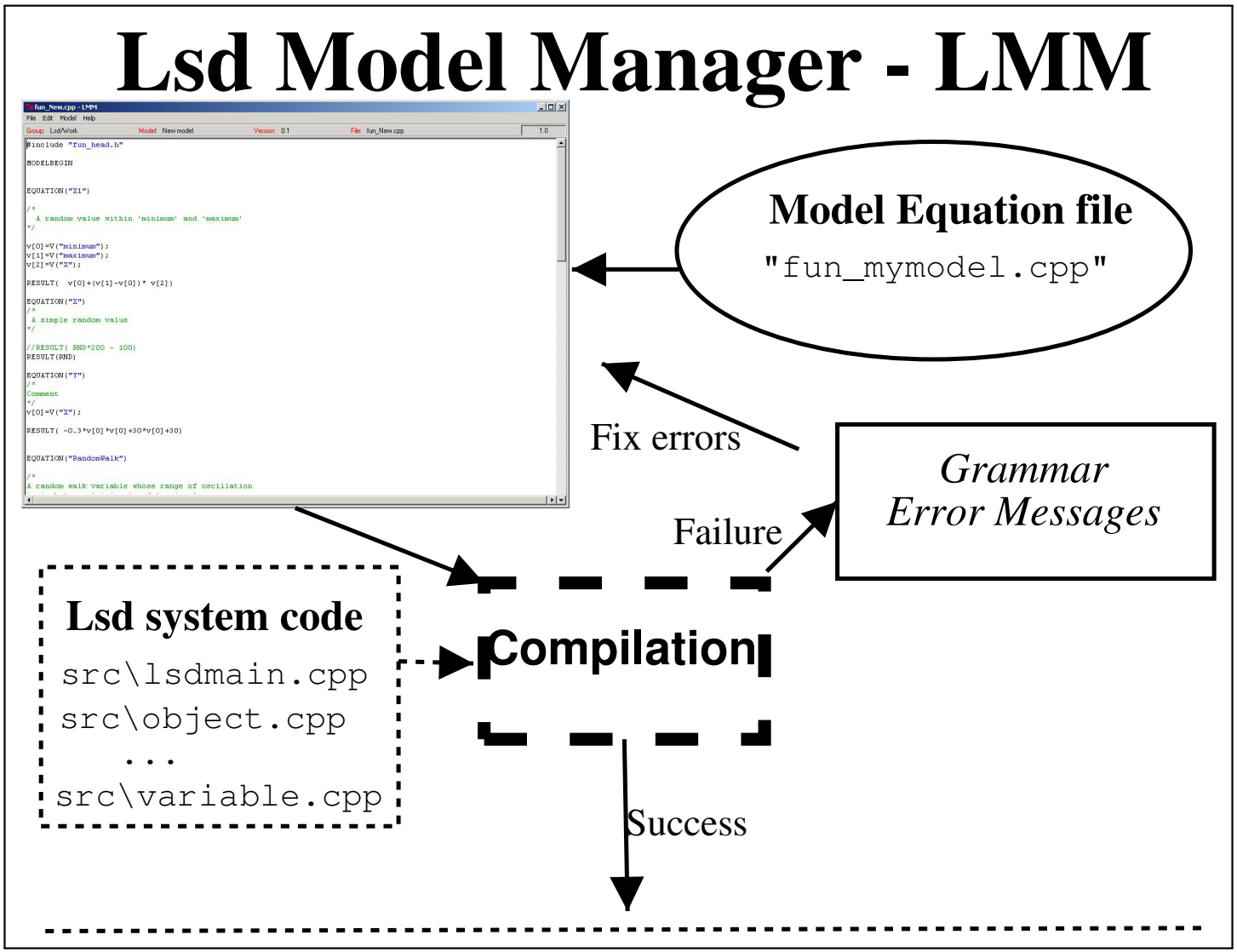
- **Equations**: implemented in a power programming language (C++) using a stylized format (script). Each equation is a chunk of lines expressing the content of the equation.
- **Configurations**: names of the model elements and initializations. Stored into text files, configurations are loaded, edited and saved by means of intuitive and flexible graphical interfaces.

LSD technical components

LSD is distributed with a program called *LSD Model Manager* (LMM) performing the following tasks:

1. Organize the projects and manage the required files.
2. Assist in the writing of the equations.
3. Manage the compilation process.
4. Provide indications on grammar errors in the equations' code.

LSD technical components

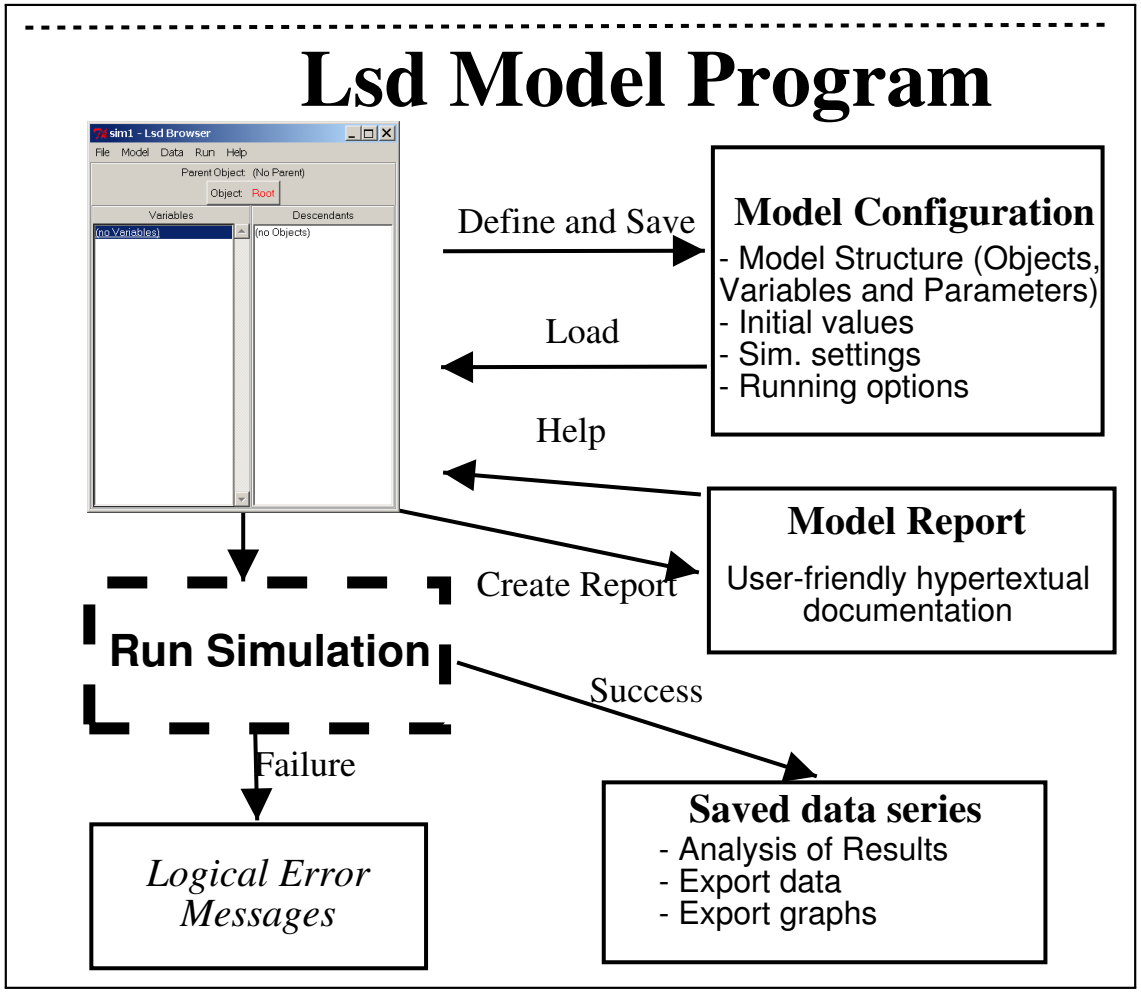


LSD technical components

On success LMM generates an executable called *LSD Model Program* embodying the equations of the model and allowing every remaining operation concerning the model:

1. Define, save and load model configurations.
2. Run single or multiple simulations.
3. Analyse the results, at run-time or at the end of the simulation, generating data-sets and graphs .
4. Investigate the model state before, during or after a run.
5. Catch and report on errors at run time, keeping data produced until the stop.
6. Document a model with its own interfaces, or exporting reports in HTML or LaTeX format

LSD technical components



Using LSD

Let's see an example of using LSD. We will perform the following operations to implement a discrete version of the Replicator Dynamics model.

The operations are the following:

- Write the code for the variables' equations.
- Assign number of objects.
- Assign initial values to parameters and variables at $t = 0$.
- Run simulations.
- Analyse the results.
- Document model and results.

Equation

$$Sales[t] = Sales[t-1] \times (1 + a \times \frac{Quality - AvQuality[t]}{AvQuality[t]})$$

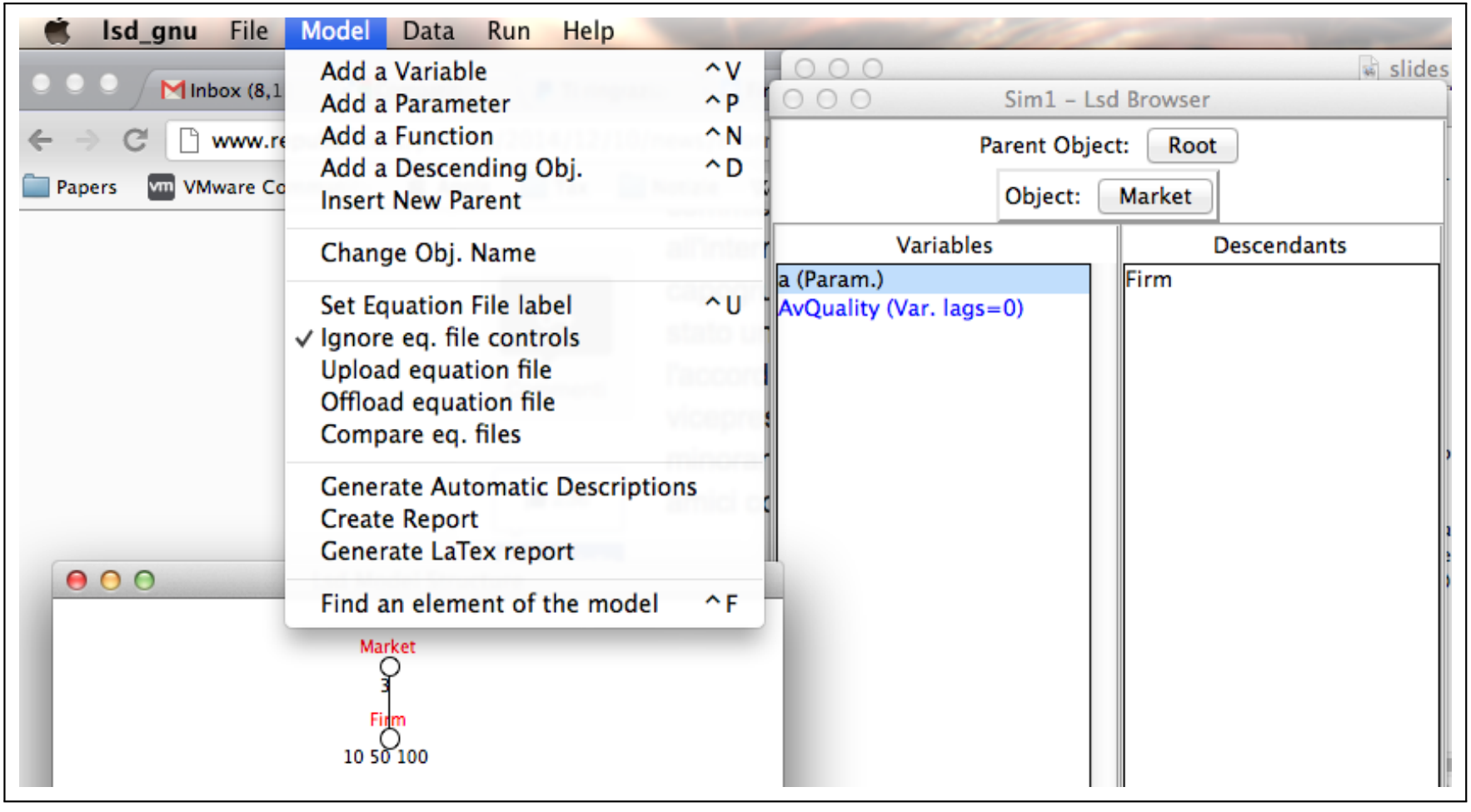
```
EQUATION("Sales")
/*
Sales expressed as
discrete-time repl. dynamics
*/
v[0]=V("Quality");
v[1]=VL("Sales",1);
v[2]=V("a");
v[3]=V("AvQuality");

v[4]=v[1]+v[1]*v[2]*(v[0]-v[3])/v[3];
RESULT(v[4])
```

Equation

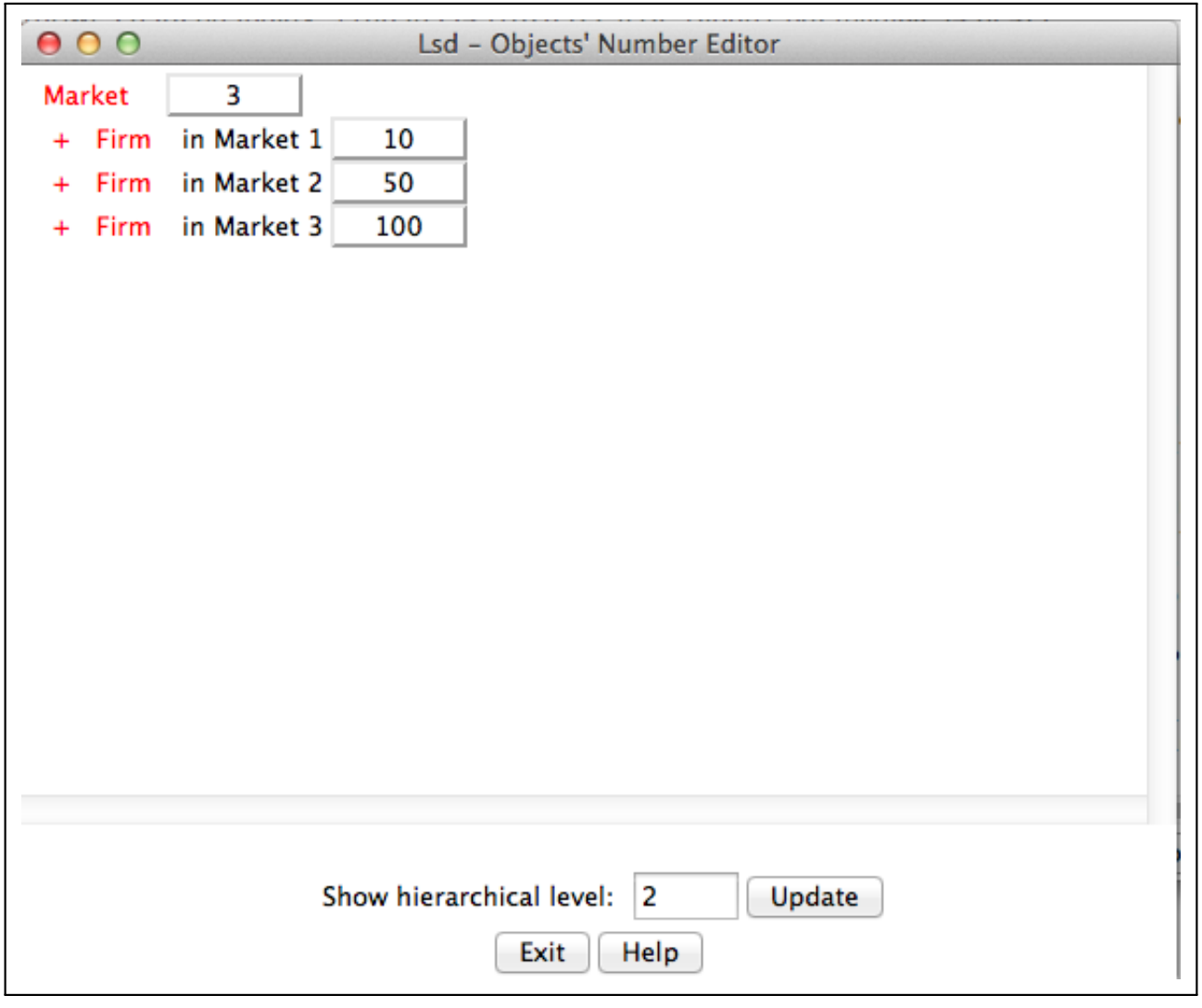
$$AvQuality[t] = \frac{\sum_{i=1}^{N} Sales[t]_i \times Quality_i}{\sum_{i=1}^{N} Sales[t]_i}$$

```
EQUATION("AvQuality")
/*
Average quality, computed as av. weighted by sales
*/
v[3]=0,v[2]=0;
CYCLE(cur,"Firm")
 {
  v[0]=VS(cur,"Sales");
  v[1]=VS(cur,"Quality");
  v[2]+=v[0]*v[1];
  v[3]+=v[0];
 }
RESULT(v[2]/v[3]);
```

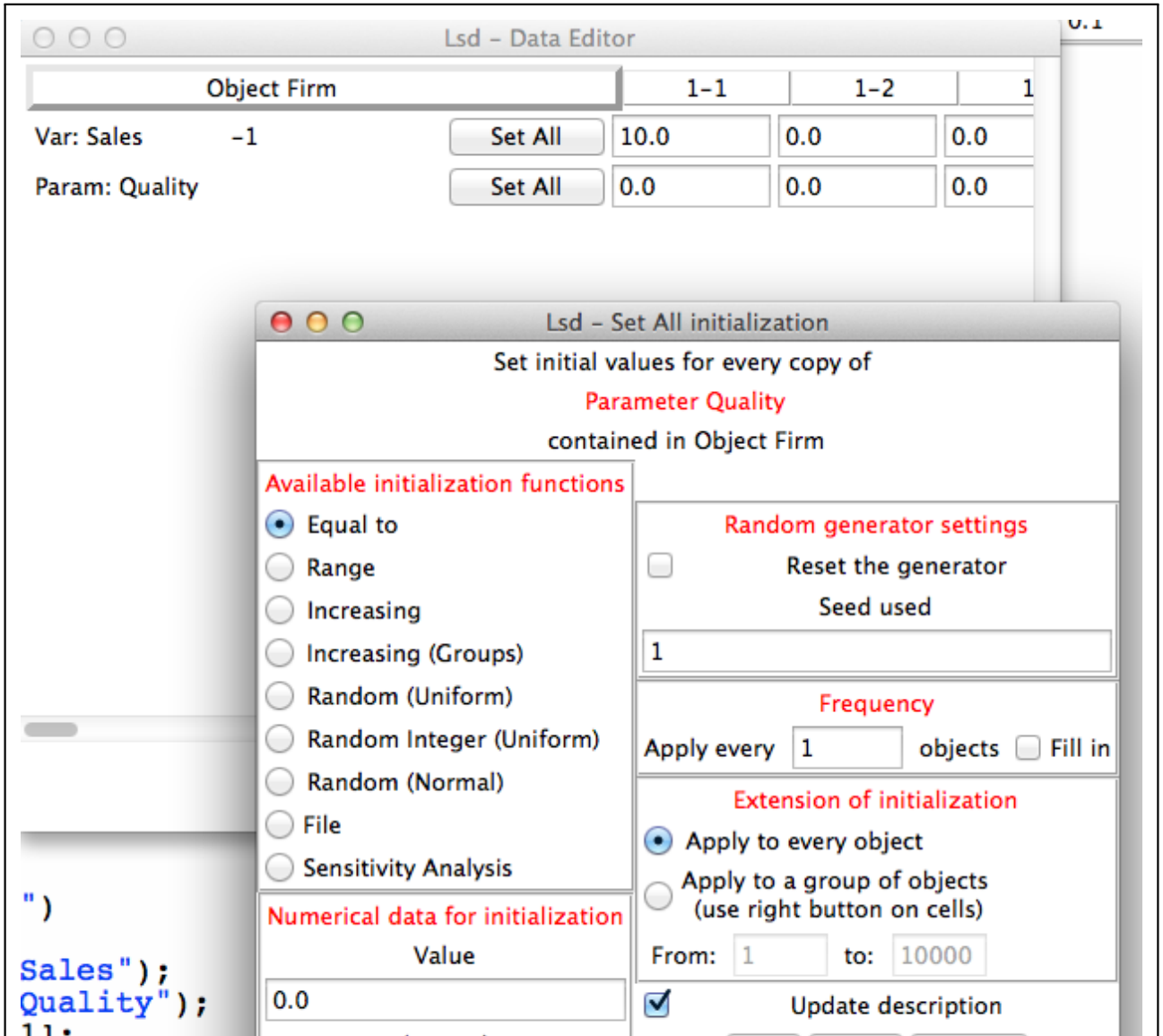
Define elements



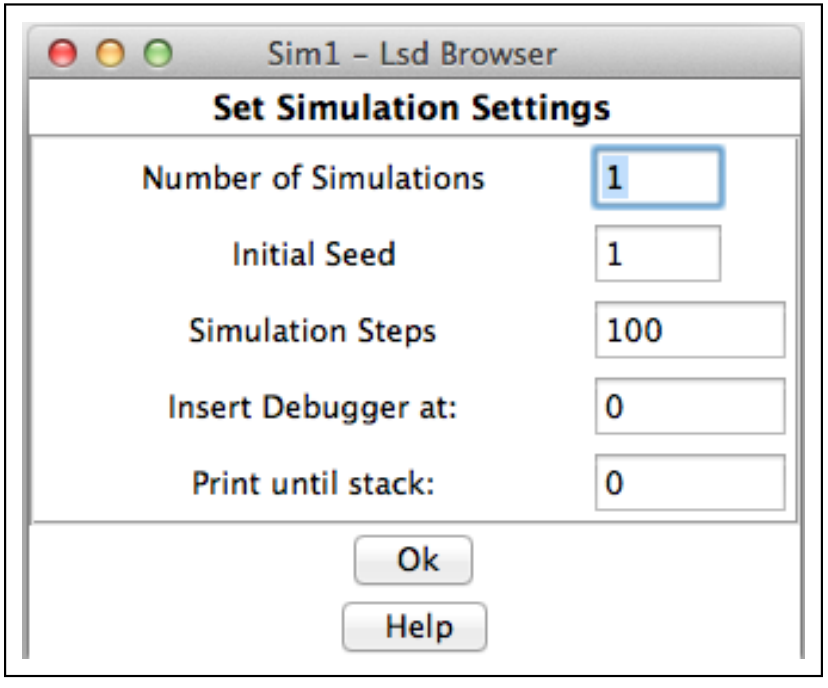
Number of Objects



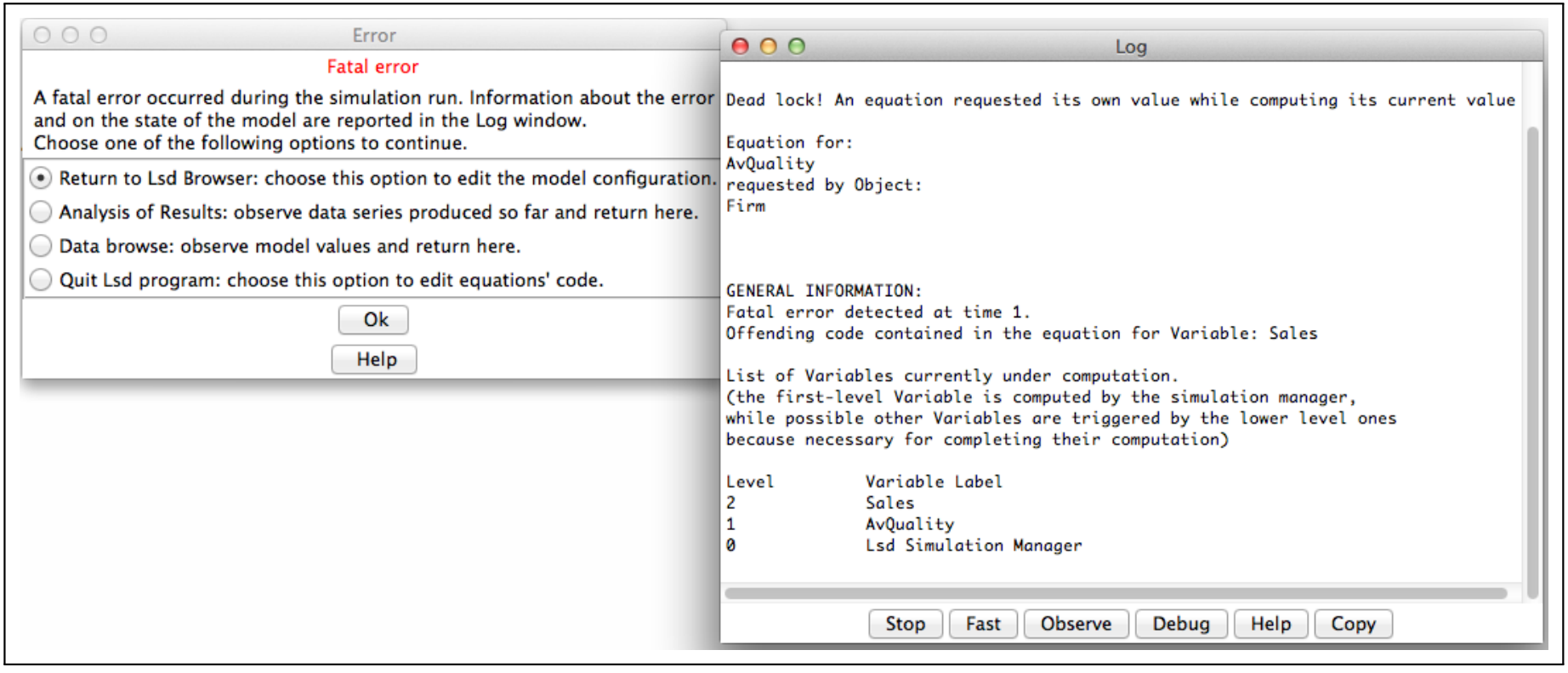
Initial Values



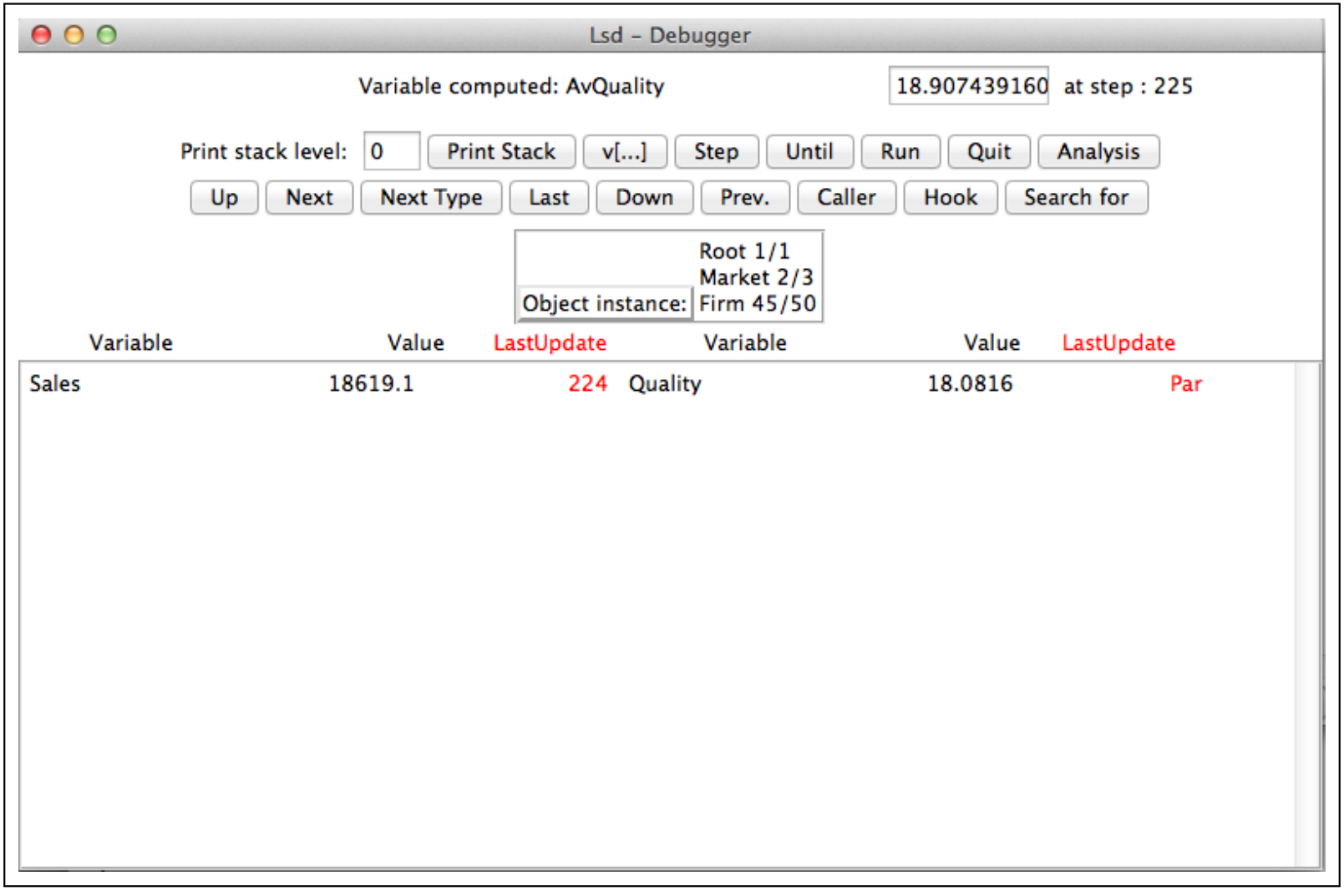
Simulation options



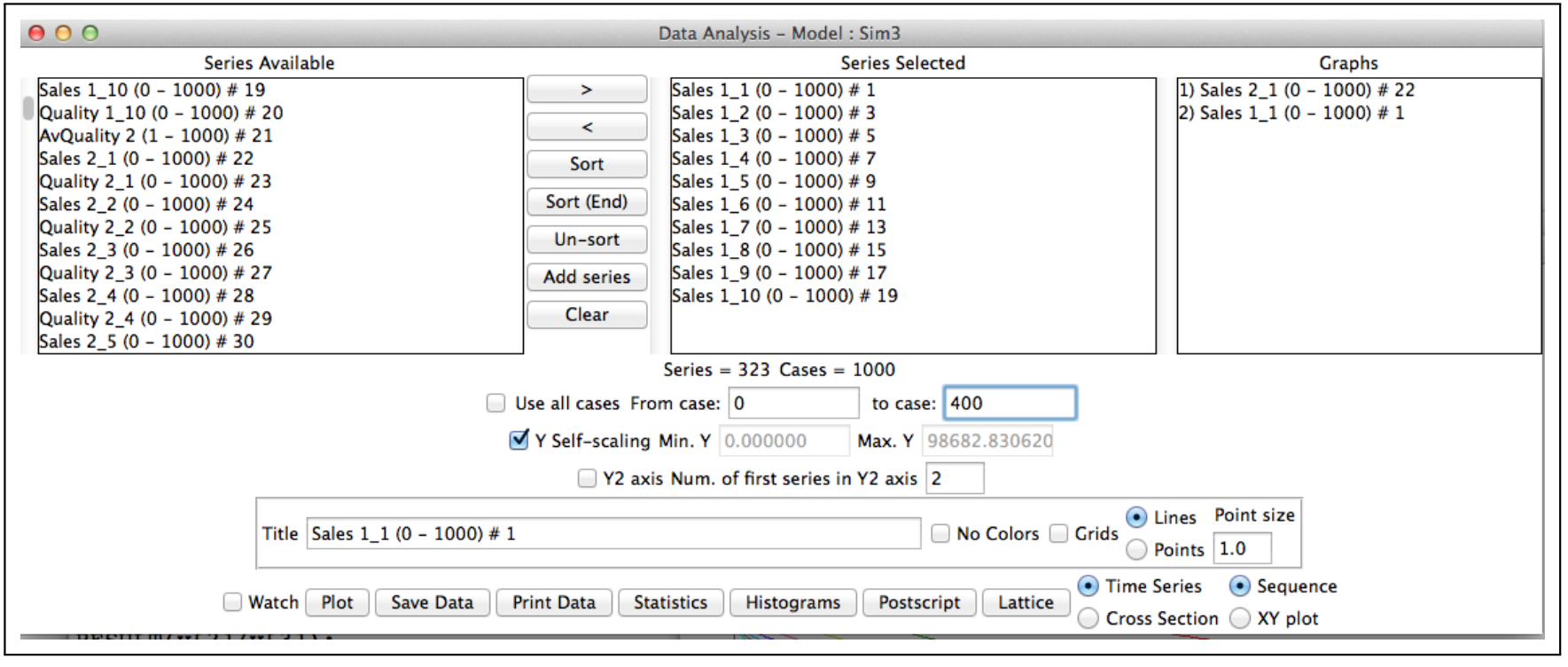
Error catching



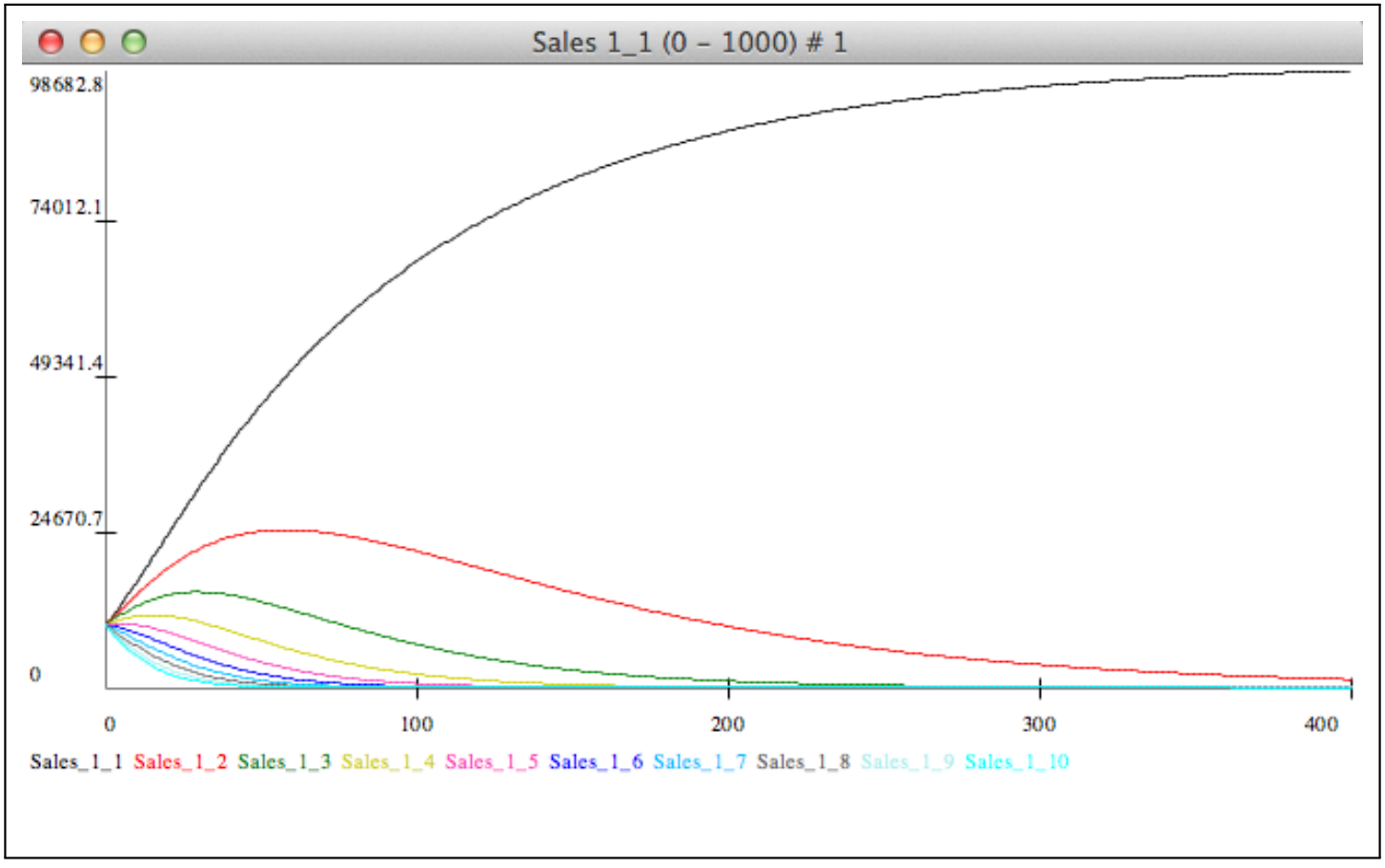
Inspect


Lsd – Debugger
Variable computed: AvQuality
18.907439160
at step : 225
Print stack level:
0
Print Stack
v[...]
Step
Until
Run
Quit
Analysis
Up
Next
Next Type
Last
Down
Prev.
Caller
Hook
Search for
Object instance:
Root 1/1
Market 2/3
Firm 45/50
Variable
Value
LastUpdate
Variable
Value
LastUpdate
Sales
18619.1
224
Quality
18.0816
Par

Results

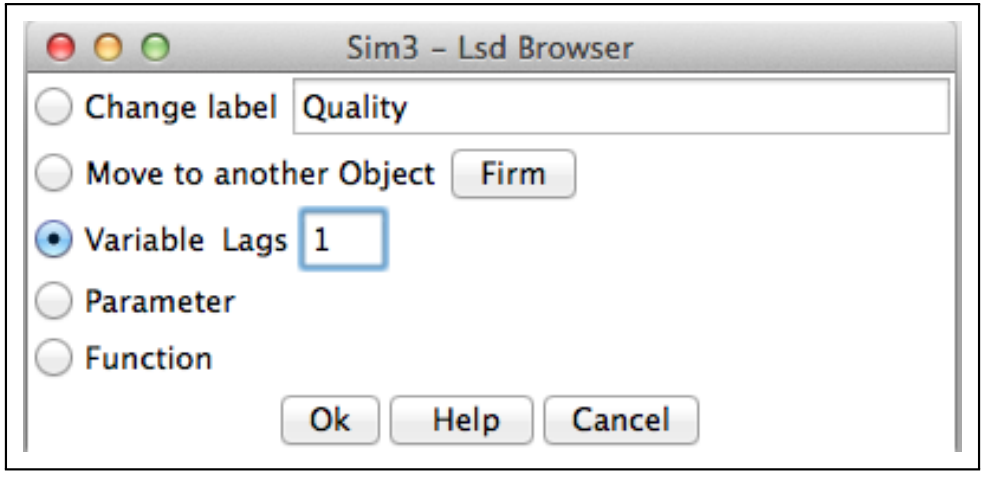

Data Analysis – Model : Sim3
Series Available
Sales 1_10 (0 – 1000) # 19
Quality 1_10 (0 – 1000) # 20
AvQuality 2 (1 – 1000) # 21
Sales 2_1 (0 – 1000) # 22
Quality 2_1 (0 – 1000) # 23
Sales 2_2 (0 – 1000) # 24
Quality 2_2 (0 – 1000) # 25
Sales 2_3 (0 – 1000) # 26
Quality 2_3 (0 – 1000) # 27
Sales 2_4 (0 – 1000) # 28
Quality 2_4 (0 – 1000) # 29
Sales 2_5 (0 – 1000) # 30
>
<
Sort
Sort (End)
Un-sort
Add series
Clear
Series Selected
Sales 1_1 (0 – 1000) # 1
Sales 1_2 (0 – 1000) # 3
Sales 1_3 (0 – 1000) # 5
Sales 1_4 (0 – 1000) # 7
Sales 1_5 (0 – 1000) # 9
Sales 1_6 (0 – 1000) # 11
Sales 1_7 (0 – 1000) # 13
Sales 1_8 (0 – 1000) # 15
Sales 1_9 (0 – 1000) # 17
Sales 1_10 (0 – 1000) # 19
Graphs
1) Sales 2_1 (0 – 1000) # 22
2) Sales 1_1 (0 – 1000) # 1
Series = 323 Cases = 1000
Use all cases From case: 0 to case: 400
Y Self-scaling Min. Y 0.000000 Max. Y 98682.830620
Y2 axis Num. of first series in Y2 axis 2
Title Sales 1_1 (0 – 1000) # 1
No Colors Grids
Lines Points
Point size 1.0
Watch Plot Save Data Print Data Statistics Histograms Postscript Lattice
Time Series Sequence
Cross Section XY plot

Results


Sales 1_1 (0 – 1000) # 1
98682.8
74012.1
49341.4
24670.7
0
0
100
200
300
400
Sales_1_1 Sales_1_2 Sales_1_3 Sales_1_4 Sales_1_5 Sales_1_6 Sales_1_7 Sales_1_8 Sales_1_9 Sales_1_10

Extend

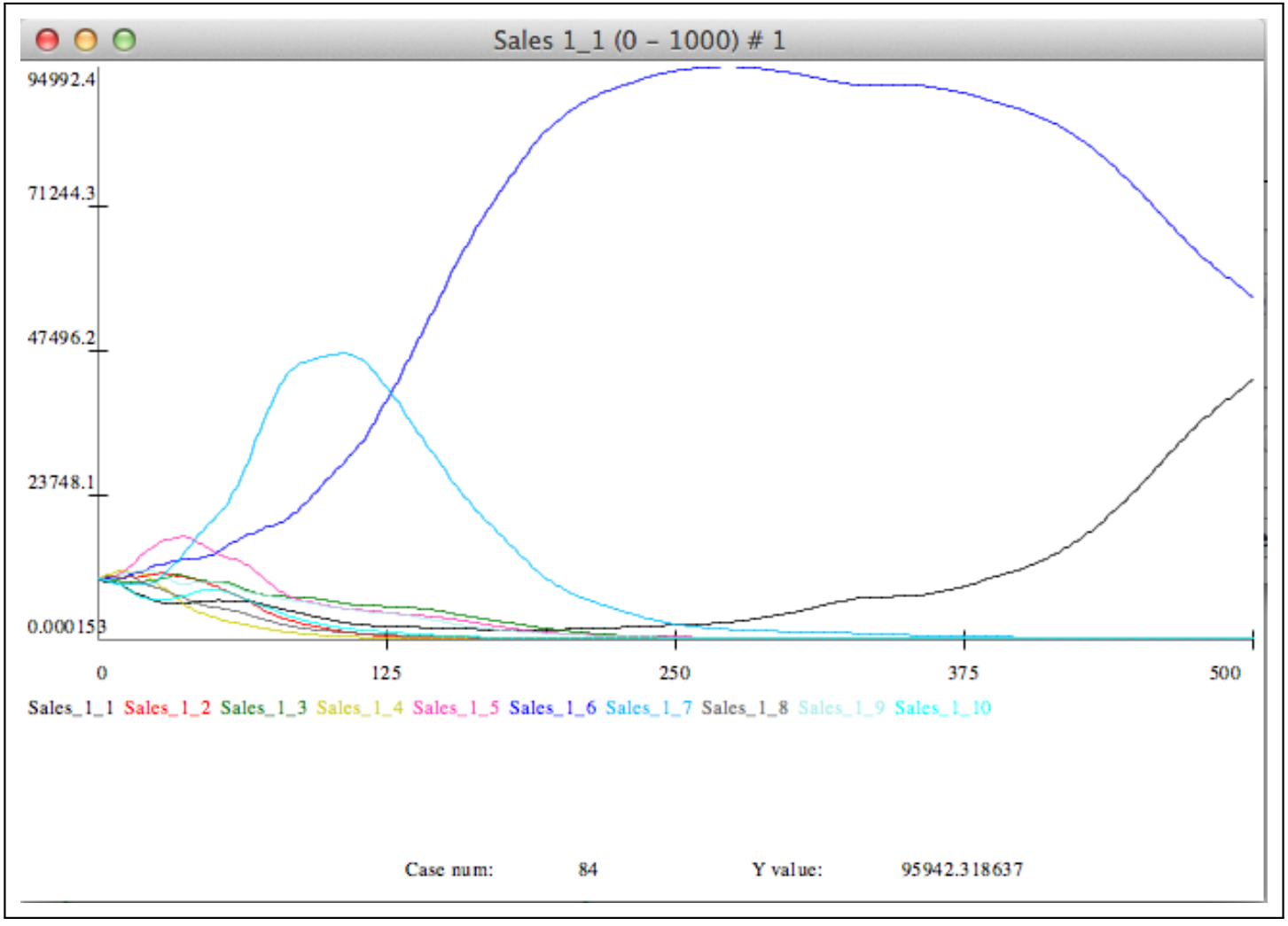
$$Quality[t] = Quality[t-1] + Random(Min, Max)$$



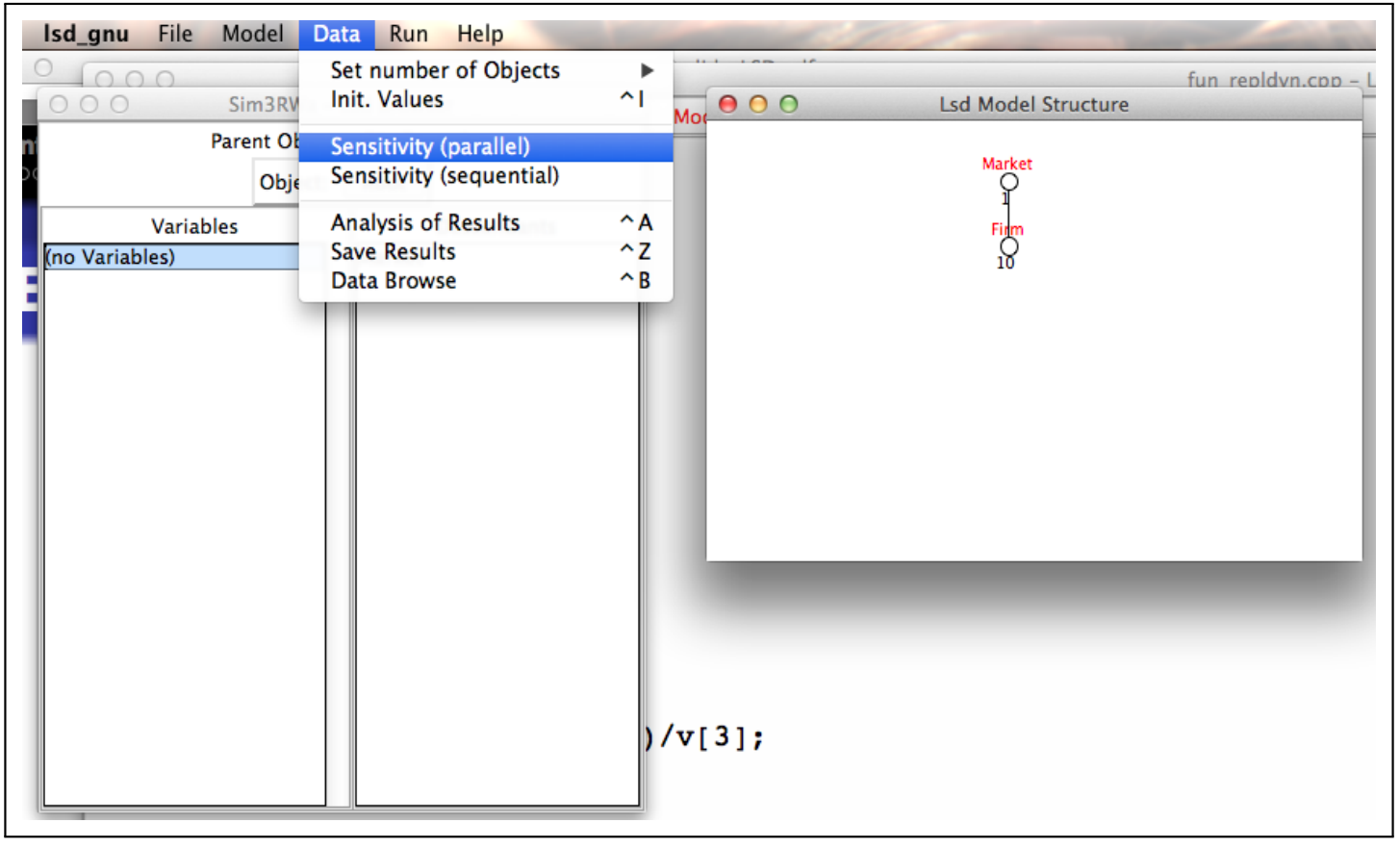
Endogenize

```
EQUATION("Quality")
/*
Quality expressed as a Random Walk process
*/

v[0]=VL("Quality",1);
v[1]=V("min");
v[2]=V("Max");

v[3]=UNIFORM(v[1],v[2]);
v[4]=v[0]+v[3];
RESULT(v[4] )
```

Extend


Sales 1_1 (0 – 1000) # 1
94992.4
71244.3
47496.2
23748.1
0.000153
0
125
250
375
500
Sales_1_1 Sales_1_2 Sales_1_3 Sales_1_4 Sales_1_5 Sales_1_6 Sales_1_7 Sales_1_8 Sales_1_9 Sales_1_10
Case num: 84
Y value: 95942.318637

Sensitivity/robustness



Entry/Exit

```
EQATION("NumberFirms")
/*
Control entry and exit
*/
v[1]=0;
v[3]=V("AvQuality");
CYCLE_SAFE(cur, "Firm")
 {
  v[0]=VS(cur,"Sales");
  if(v[0]<0.01)
   {//INTERACTS(cur, "Small", v[0]);
    DELETE(cur);
   }
  else
   v[1]++;
 }
v[2]=V("ProbEntry");
if(RND<v[2])
 {
  cur=ADDOBJ("Firm");
  v[4]=v[3]*(1+UNIFORM(-0.05, 0.05) );
  WRITELS(cur,"Quality",v[4], t);
  WRITELS(cur,"Sales",100, t);
  v[1]++;
 }
RESULT(v[1] )
```

References

LSD is available for all platform: Windows (no additional software needed), Mac OS and Linux.

To install LSD download the latest version from `github.com/marcov64/Lsd` and unzip the file in a suitable folder. See the `Readme.txt` file for installation instructions.

References

- `www.labsimdev.org`: Info, manuals, forum, etc.
- `github.com/marcov64/Lsd`: download, patches, contributions
- Documentation: manual and tutorial, available from the LMM help pages.
- Menus **Help**: context-dependent assistance available on all LSD interfaces.