Modelos de simulação ABM e a plataforma LSD

Aula 1 – 2ª Parte

Outline

1. Properties of agent-based models
2. Definition of agent-based simulation models
3. Introduction to LSD features
4. Example of a LSD model

ABM: a definition

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

ABM: a definition

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

- Low-level (component) entities have weak constraints and a degree of heterogeneity.

ABM: a definition

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

ABM: a definition

Agent-based models are based on a **hierarchical** data structure subject to **real 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.
- Time dynamics independently operating on each variable.

ABM: a definition

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

ABM: a definition

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

Actually, ABM's are subject to a different type of constraints, such **temporal consistency**, that can be severely limiting.

Results are produced in terms of time series representing **extremely detailed records** of virtual histories. The strength of the results needs to be assessed depending on the theoretical claim of the model.

ABM: a definition

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

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** on variables’ values, not explicitly indicating an action. Solutions of the models are made by a set of values satisfying all the constraints at once, but they cannot provide any indication on the system for values outside the solution(s), including the possible patterns towards the solution(s).

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** on variables’ values, not explicitly indicating an action. Solutions of the models are made by a set of values satisfying all the constraints at once, but they cannot provide any indication on the system for values outside the solution(s), including the possible patterns towards 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, independently from the possible convergence, replicability, etc. It is up to the researcher to extract and justify knowledge from the dataset.

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, slowing their diffusion and questioning their usefulness.

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. Difficul to use;

ABM in Economics

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

1. Difficul 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. Difficul 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. Difficul 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.

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.

Methodological perspective

1. Research models **do not** aim at replicating reality.
   Similarity between empirical evidence and theoretical results can be a highly relevant step in a research project. When relevant, it requires a number of preconditions to be satisfied, such as identifying the relevant piece of reality and an adequate measurement system.

Methodological perspective

1. Research models **do not** aim at replicating reality.
   Similarity between empirical evidence and theoretical results can be a highly relevant step in a research project. When relevant, it requires a number of preconditions to be satisfied, such as identifying the relevant piece of reality and an adequate measurement system.
2. Research models aim at **explaining** reality.
   Explanations are logical connections between two states of the system by means of an explanatory mechanisms, $A \stackrel{\mu}{\Rightarrow} B$. Explanations are the elementary foundations of scientific knowledge.

Methodological assumptions

The scientific use of simulation models consist in four steps:

1. Build a **simplified** representation of a reality.

Methodological assumptions

The scientific use of simulation models consist in four steps:

1. Build a **simplified** representation of a reality.
2. Ensure that the model generates simulated events **compatible** with those observed in the real world.

Methodological assumptions

The scientific use of simulation models consist in four steps:

1. Build a **simplified** representation of a reality.
2. Ensure that the model generates simulated events **compatible** with those observed in the real world.
3. Find **interesting explanations** of simulated events, as if analysing the historical record of a virtual system.

Methodological assumptions

The scientific use of simulation models consist in four steps:

1. Build a **simplified** representation of a reality.
2. Ensure that the model generates simulated events **compatible** with those observed in the real world.
3. Find **interesting explanations** of simulated events, as if analysing the historical record of a virtual system.
4. Evaluate whether the **same explanations** may apply to real world cases.

Topics of the course

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

- **Design**: decide the content of the model required to fulfill a specific research project.

Topics of the course

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

- **Design**: decide the content of the model required to fulfill a specific research project.
- **Implementation**: turning an abstract model design 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 the content of the model required to fulfill a specific research project.
- **Implementation**: turning an abstract model design into a working simulation program.
- **Interpretation**: extracting knowledge (e.g. interesting explanations) from simulation runs.

Topics of the course

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

- **Design**: decide the content of the model required to fulfill a specific research project.
- **Implementation**: turning an abstract model design into a working simulation program.
- **Interpretation**: extracting knowledge (e.g. interesting explanations) from simulation runs.
- **Revision**: correcting or extending existing code, particularly relevant when using a gradual model building approach.

Topics of the course

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

- **Design**: decide the content of the model required to fulfill a specific research project.
- **Implementation**: turning an abstract model design into a working simulation program.
- **Interpretation**: extracting knowledge (e.g. interesting explanations) from simulation runs.
- **Revision**: correcting or extending existing code, particularly relevant when using a gradual model building approach.
- **Documentation**: simulation results must be properly formatted to report (and support) scientific claims.

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

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 language 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 language 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 a set of time-indexed variables associated to the algorithms used to compute their values:

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

Definition of simulation models

A simulation model is defined independently from the language 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 a set of time-indexed variables associated to the algorithms used to compute their values:

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

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

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

1. **Univocal.** The definition must be unambiguous, to avoid that the same definition admits implementations producing different 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, to avoid that the same definition admits implementations producing different 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, to avoid that the same definition admits implementations producing different 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 elementary components of AB models are the following:

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

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 computationally legal expression.

Functions

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

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

Functions provide values used only internally because they cannot be saved as time series. A given function at a given time step may produce several different values, or none.

Parameters

Parameters are labels associated to numerical values.

$$\alpha$$

Parameters are essentially variables with degenerate equations

$$\alpha_t = \alpha_{t-1}$$

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

In “hierarchical” models, vector-based representations are cumbersome to use and error-prone. Programming languages have developed a more powerful concept, including vectors as special cases, but far more general and flexible: **objects**.

Objects are containers, storing together different types of elements 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 a real-world entity.

Definition of simulation models

Object-based representations are *orthogonal* to vectors.

		Object-based			
		$ObjectA^1$	$ObjectA^2$	...	$ObjectA^N$
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
	$\overrightarrow{ObjectB}$	$ObjectB^1$	$ObjectB^2$	...	$ObjectB^N$

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 $t-1, t-2, ...$

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 ($t = 1$) the values for variables at time $t < 1$ are not available, and therefore must be supplied by users.

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 ($t=1$) the values for variables at time $t<1$ are not available, and therefore must be supplied by users.

In general, the same model structure will 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 given model structure.

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 $X_1 = Y_0 + \alpha$.

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 t=1 is the first time step of the simulation run. 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

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 t=1 is the first time step of the simulation run. Consequently, the modeller that must assign to Y a **lagged** (or past) value for Y as part of the initialization of the model.

In case equations use values with more than 1 lag (e.g. $X_t = Y_{t-3} + \alpha$) the initialization requires more lagged values, to be used at $t = 1$, $t = 2$ and $t = 3$.

Definition of simulation models

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

- **Equations**: pieces of code computing values for each variable and function in the model.

Definition of simulation models

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

- **Equations**: pieces of code computing 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. necessarily producing the same results) once we describe the following elements:

- **Equations**: pieces of code computing 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. necessarily producing the same results) once we describe the following elements:

- **Equations**: pieces of code computing 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 vs. generic programming languages

A simulation model can, in principle, be implemented in any programming language. However, a research simulation project is radically different in respect of a standard software project.

Programming vs. Simulating

	Software engineering	Simulations models
Programmers vs. users	Distinct people/roles/skills	Same people, researchers

Programming vs. Simulating

	Software engineering	Simulations models
Programmers vs. users	Distinct people/roles/skills	Same people, researchers
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very crucial

Programming vs. Simulating

	Software engineering	Simulations models
Programmers vs. users	Distinct people/roles/skills	Same people, researchers
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very crucial
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define elementary components and then piece them together

Programming vs. Simulating

	Software engineering	Simulations models
Programmers vs. users	Distinct people/roles/skills	Same people, researchers
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very crucial
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define elementary components and then piece them together
Implementation details	Hidden to final users	Extensively documented for assessment and re-use

Programming vs. Simulating

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

Programming vs. Simulating

	Software engineering	Simulations models
Programmers vs. users	Distinct people/roles/skills	Same people, researchers
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very crucial
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define elementary components and then piece them together
Implementation details	Hidden to final users	Extensively documented 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	Simulations models
Programmers vs. users	Distinct people/roles/skills	Same people, researchers
Objective	Pre-determined output, means irrelevant	Indeterminate output, the means are very crucial
Development strategy	Top-down: design the structure and then fill in the details	Bottom-up: define elementary components and then piece them together
Implementation details	Hidden to final users	Extensively documented 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 worlds replicating the puzzles of reality *and* providing tools to solve them.

LSD simulation models

The code for a computer program can be decomposed in two sections:

1. **Core functionalities**, the computational structures performing the feature elaborations of the program.

LSD simulation models

The code for a computer program can be decomposed in two sections:

1. **Core functionalities**, the computational structures performing the feature elaborations of the program.
2. **Interfaces**: ancillary code controlling the data flowing from the external environment (users and peripherals) to the program core functionalities, and viceversa.

LSD simulation models

The code for a computer program can be decomposed in two sections:

1. **Core functionalities**, the computational structures performing the feature elaborations of the program.
2. **Interfaces**: ancillary code controlling the data flowing from the external environment (users and peripherals) to the program core functionalities, and viceversa.

In general designing and implementing interfaces is far more complex than implementing the core functionalities.

Having poor interfaces may severely hinder the power of the program because its core functionalities cannot receive necessary data and/or its results cannot be appropriately communicated.

LSD principle

LSD allows users to generate a simulation program defining **only** the elements of a simulation model according to an intuitive and generic format.

LSD principle

LSD allows users to generate a simulation program defining **only** the elements of a simulation model according to an intuitive and generic format.

At each stage of a model design, development and use, LSD provides **automatically** all the necessary interfaces (model browser, debugger, results etc.) providing users with full access to any aspect of the model.

Design

The key features of the LSD design are the following:

1. Extremely limited number of building blocks: **variables** (parameters and functions) and **objects**.

Design

The key features of the LSD design are the following:

1. Extremely limited number of building blocks: **variables** (parameters and functions) and **objects**.
2. Computational content defined by a set of discrete-time difference equations, expressed as independent programming routines.

Design

The key features of the LSD design are the following:

1. Extremely limited number of building blocks: **variables** (parameters and functions) and **objects**.
2. Computational content defined by a set of discrete-time difference equations, expressed as independent programming routines.
3. Simulation **program** automatically assembled from the simulation **model** supplied by the user.

Design

The key features of the LSD design are the following:

1. Extremely limited number of building blocks: **variables** (parameters and functions) and **objects**.
2. Computational content defined by a set of discrete-time difference equations, expressed as independent programming routines.
3. Simulation **program** automatically assembled from the simulation **model** supplied by the user.
4. Automatic, context- and content-dependent interfaces to manipulate any aspect of the model.

Design

The key features of the LSD design are the following:

1. Extremely limited number of building blocks: **variables** (parameters and functions) and **objects**.
2. Computational content defined by a set of discrete-time difference equations, expressed as independent programming routines.
3. Simulation **program** automatically assembled from the simulation **model** supplied by the user.
4. Automatic, context- and content-dependent interfaces to manipulate any aspect of the model.
5. Efficient, powerful, scalable and multi-platform code (based on GNU C++).

LSD Components

A user needs to specify the following elements specifying the content of a **model**:

LSD Components

A user needs to specify the following elements specifying the content of a **model**:

- **Variables' equations**: separate chunks of code expressing how the generic instance of the variable updates its value at the generic time step.

LSD Components

A user needs to specify the following elements specifying the content of a **model**:

- **Variables' equations**: separate chunks of code expressing how the generic instance of the variable updates its value at the generic time step.
- **A model structure**: lists of objects containing variables, parameters, or other objects;

LSD Components

A user needs to specify the following elements specifying the content of a **model**:

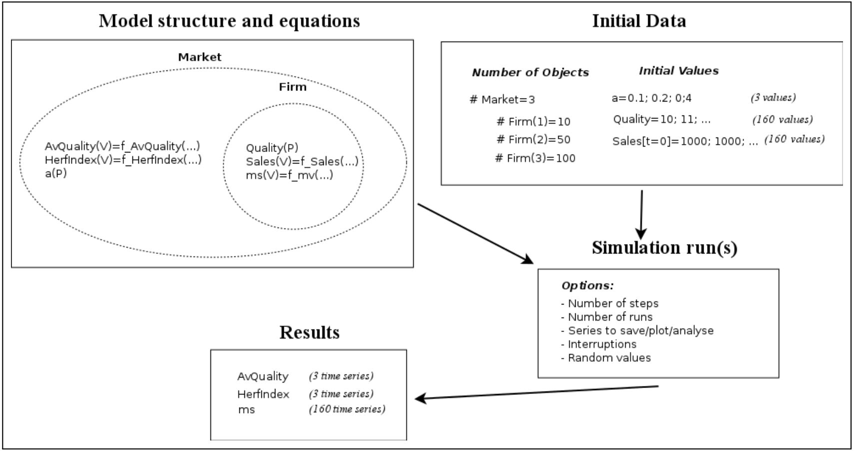
- **Variables' equations**: separate chunks of code expressing how the generic instance of the variable updates its value at the generic time step.
- **A model structure**: lists of objects containing variables, parameters, or other objects;
- **Initial data**: numerical values to initialize the model, such as the number of copies for each object and the values for parameters and variables at time *t=0*.

LSD Components

A user needs to specify the following elements specifying the content of a **model**:

- **Variables' equations**: separate chunks of code expressing how the generic instance of the variable updates its value at the generic time step.
- **A model structure**: lists of objects containing variables, parameters, or other objects;
- **Initial data**: numerical values to initialize the model, 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 Runs

1. The execution of simulation runs is completely automatic. The system assembles the available information and arranges automatically the required operations.

LSD Simulation Runs

1. The execution of simulation runs is completely automatic. The system assembles the available information and arranges automatically the required operations.
2. In case of errors (e.g. division by zero, missing elements, infinite loops) the system issues detailed reports.

LSD Simulation Runs

1. The execution of simulation runs is completely automatic. The system assembles the available information and arranges automatically the required operations.
2. In case of errors (e.g. division by zero, missing elements, infinite loops) the system issues detailed reports.
3. Users can interrupt the simulation to inspect the state of the model and analyse the time series produced.

LSD Simulation Output

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

LSD Simulation Output

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

LSD Simulation Output

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

LSD Simulation Output

1. Dataset containing the complete time series generated in a simulation run.
2. Management of multiple datasets, obtained by multiple replications of runs, e.g. 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.

LSD Simulation Output

1. Dataset containing the complete time series generated in a simulation run.
2. Management of multiple datasets, obtained by multiple replications of runs, e.g. 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.

LSD Simulation Output

1. Dataset containing the complete time series generated in a simulation run.
2. Management of multiple datasets, obtained by multiple replications of runs, e.g. 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).

LSD Simulation Output

1. Dataset containing the complete time series generated in a simulation run.
2. Management of multiple datasets, obtained by multiple replications of runs, e.g. 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**:

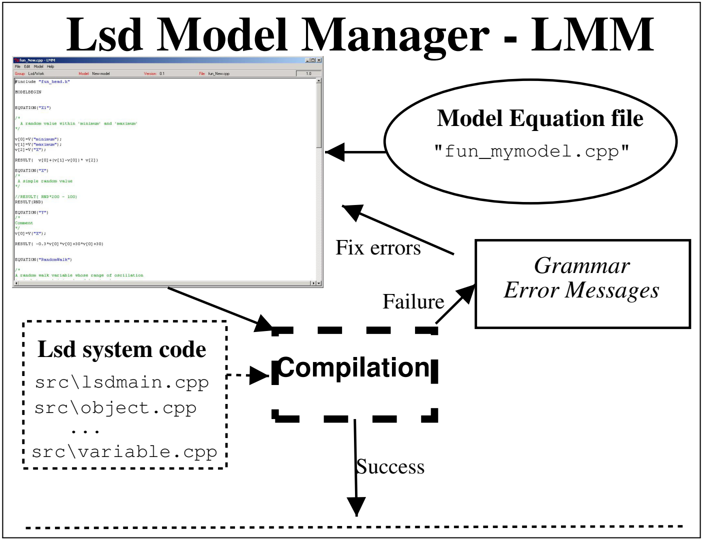
- **Equations**: implemented in a power programming language (C++) using a stylized format (script). Each equation is a chunk of lines expressing the content of the equation.
- **Configurations**: names of the model elements and initializations. Stored into text files, configurations are loaded, edited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

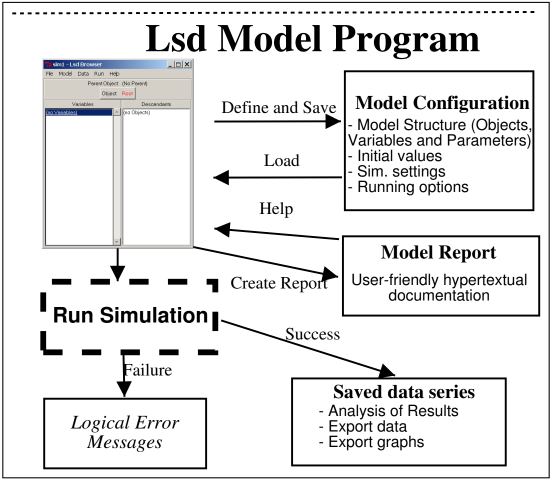


LSD technical components

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

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

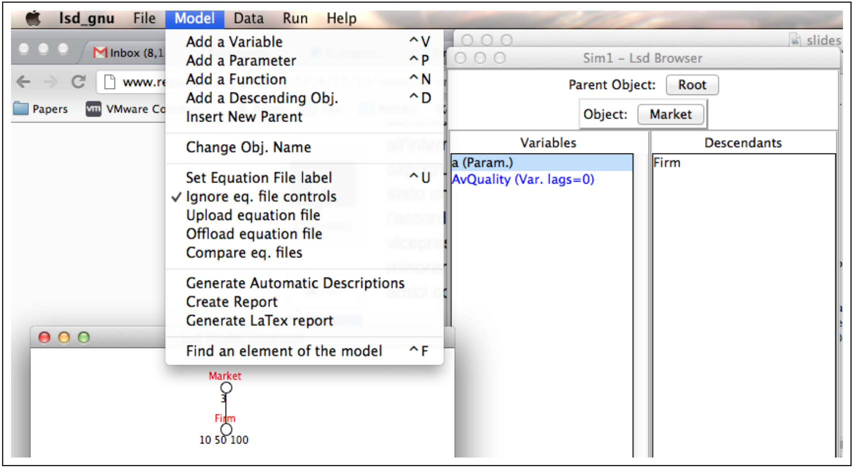
```
EQUATION("Sales")
/*
Sales expressed as
discrete-time repl. dynamics
*/
v[0]=V("Quality");
v[1]=VL("Sales",1);
v[2]=V("a");
v[3]=V("AvQuality");

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

Equation

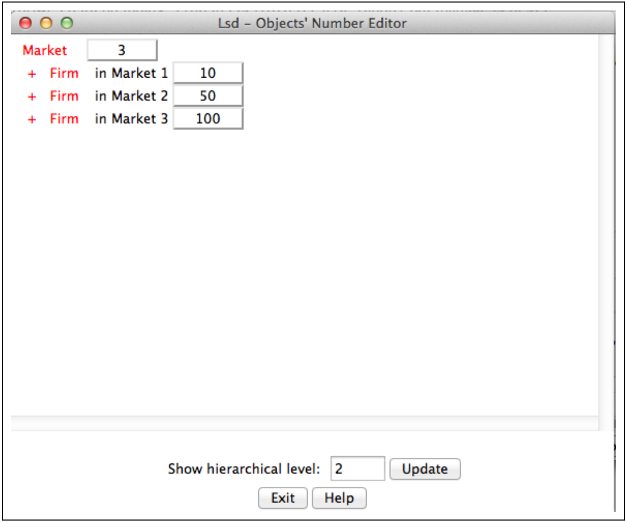
$$AvQuality[t] = \frac{\sum_{i=1}^{N} Sales[t]_i \times Quality_i}{\sum_{i=1}^{N} Sales[t]_i}$$

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

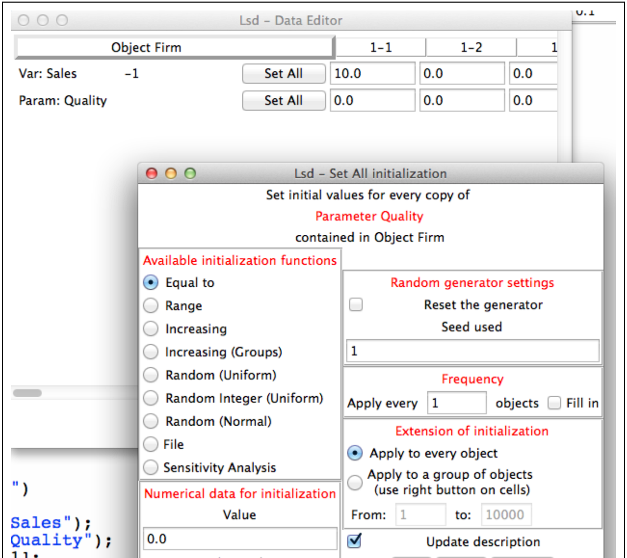
Define elements



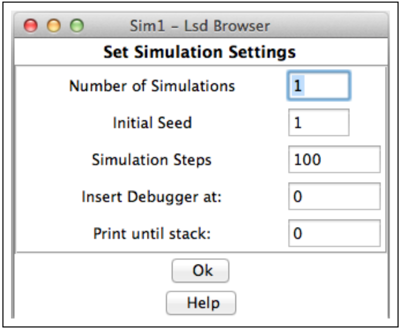
Number of Objects



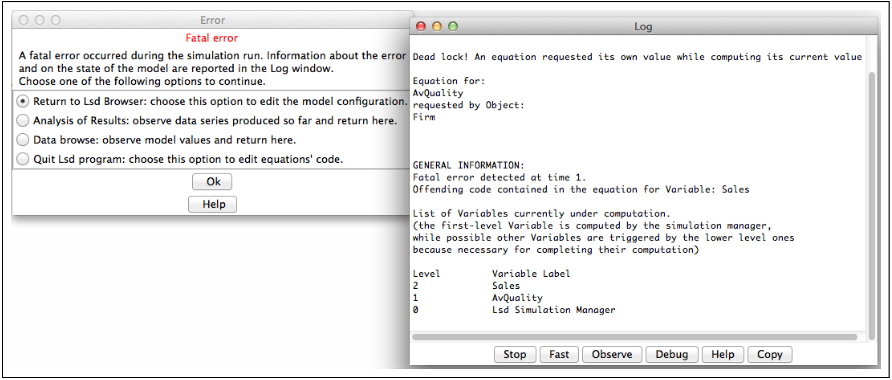
Initial Values



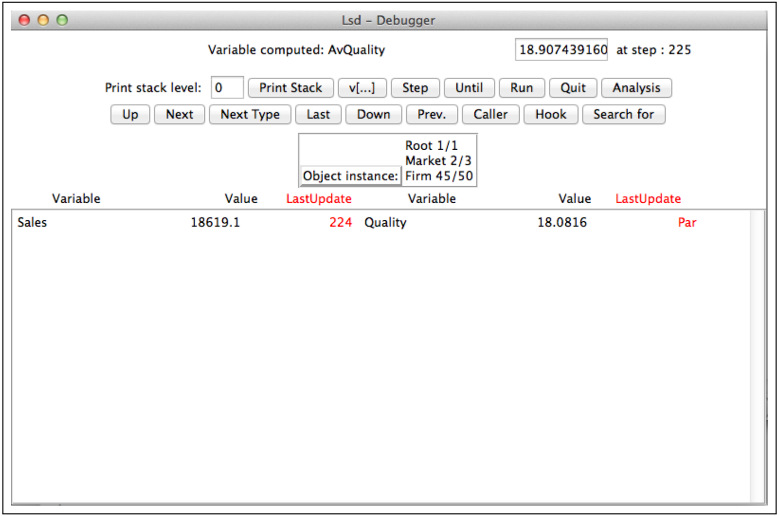
Simulation options



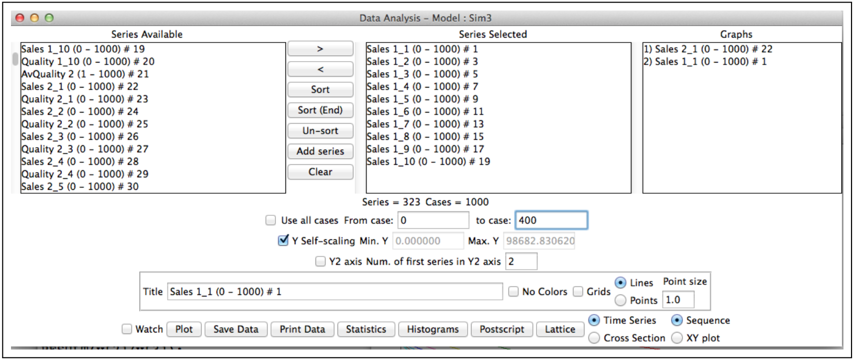
Error catching



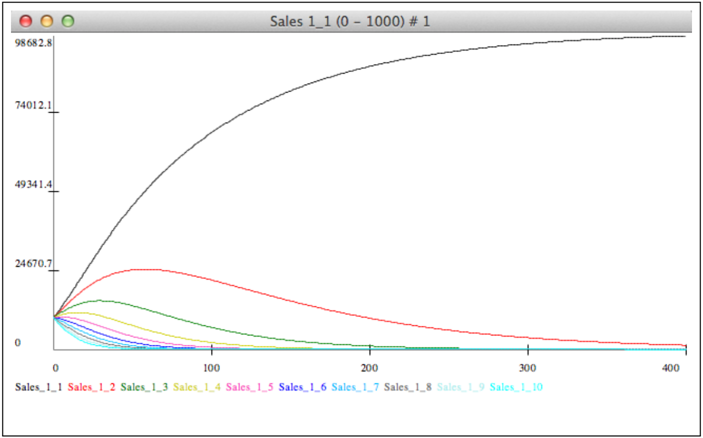
Inspect


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
Root 1/1
Market 2/3
Object instance:
Firm 45/50
Variable
Value
LastUpdate
Variable
Value
LastUpdate
Sales
18619.1
224
Quality
18.0816
Par

Results

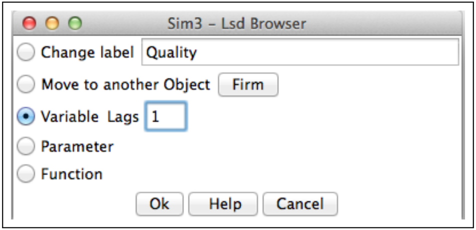

Data Analysis – Model : Sim3
Series Available
Sales 1_10 (0 – 1000) # 19
Quality 1_10 (0 – 1000) # 20
AvQuality 2 (1 – 1000) # 21
Sales 2_1 (0 – 1000) # 22
Quality 2_1 (0 – 1000) # 23
Sales 2_2 (0 – 1000) # 24
Quality 2_2 (0 – 1000) # 25
Sales 2_3 (0 – 1000) # 26
Quality 2_3 (0 – 1000) # 27
Sales 2_4 (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
Cross Section
Sequence
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

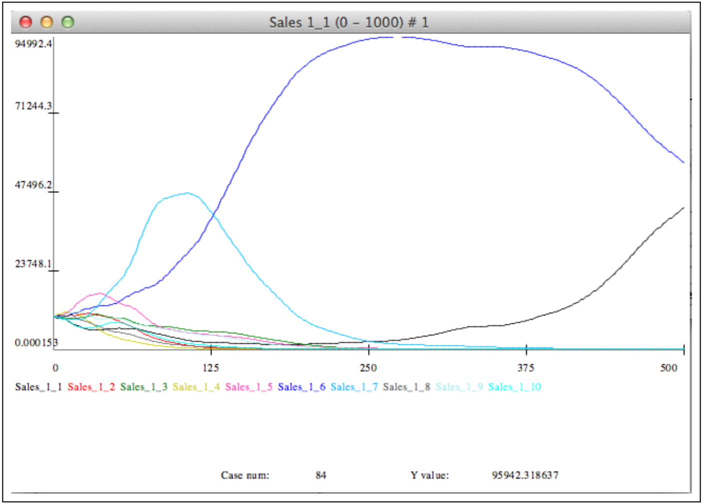
$$Quality[t] = Quality[t-1] + Random(Min, Max)$$



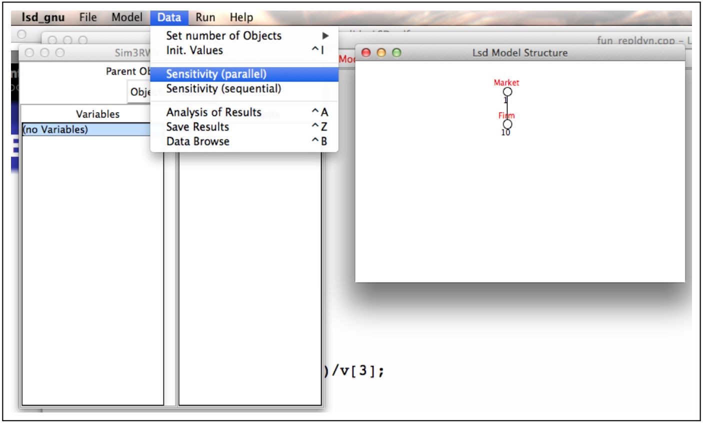
Endogenize

```
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.

Iniciando o LSD no Windows

Instalar o LSD no disco C:

Baixe o arquivo de instalação utilizando seu navegador:

https://github.com/SantAnnaKS/Lsd/archive/7.2-stable-2.zip

Abra o arquivo (clique duplo) **C:\LSD-7.2-stable-2.zip** com o Windows Explorer

Arraste ou copie a pasta **LSD-7.2-stable-2** para a pasta raiz **C:**

Abra a pasta **C:\LSD-7.2-stable-2** no Windows Explorer

Opcionalmente, renomeie a pasta para **C:\LSD**

Execute o arquivo (clique duplo) **add-shortcut.bat**

Abrir o LSD

Procure o ícone **LMM** do LSD no desktop e abra com um clique duplo

Se não encontrar o ícone, execute diretamente **C:\Lsd\run.bat**

Bibliografia

TESFATSION, L. Agent-Based Computational Economics: Modelling Economies as Complex Adaptive Systems. *Information Sciences*, 2002.

VALENTE, M.; ANDERSEN, E. S. A Hands-on Approach to Evolutionary Simulation: Nelson and Winter Models in the Laboratory for Simulation Development. *Electronic Journal of Evolutionary Modeling and Economic Dynamics,* 1003, 2002.

VALENTE, M. Agent-based Modelling with LSD. http://www.labsimdev.org, 2020.