FHG2A – Repast Simphony

User Manual

Table of Contents

Section 1: Introduction

Section 1.1: Agent-based simulation

Agent-based simulation is a tool used to model complex systems. Rather than describing systems only at a broad level, and being limited by available data, agent-based simulation allows social scientists, biologists and others to define low-level rules and observe emergent behaviour and trends.

For example, instead of using mathematical constructs for predator and prey populations, we can define behavioural rules for individuals in each population. These individuals are called agents. We place them in an environment, and let the simulation run. The rules are applied to the current state, and the environment is updated. The agents may move in the environment (for example, prey move toward grassy patches, predators move towards prey). They may also interact with each other (for example, if a predator is on the same patch as prey, it consumes up to 5 of them). Then we update the environment again, and again, applying the same rules each time. (Each application of the rules is called a tick or epoch. The unit of time it represents depends on the model.) We leave the simulation running, and observe if the results are as expected (for example, if the predator/prey populations reach a stable ratio).

This approach allows us to quickly and easily define, run and reproduce experiments. It also allows us to refine the model to better match observed data, or test the model's predictive power under different conditions.

Section 1.2: FHG2A experiments

One such model is the ideal free distribution model (IFD). It predicts how populations migrate and consume based on available resources, and has been used to explain the adoption of agriculture by non-farming cultures. Although it is evident that agriculture was eventually adopted, there is only fragmentary archaeological data, which makes it difficult to support the IFD as opposed to any other model.

The simulation you are about to run applies the IFD logic to the known environmental conditions of hunter-gatherer cultures. It does this by populating a grid with food resources (prey and wild cereal) and tribes (populations of 20-40 individuals). Each epoch (a year) each tribe migrates to a new patch of land in search of food, grows in population, and consumes available food, all according to IFD logic. The simulation tests if under these conditions, tribes do in fact adopt agriculture as a food resource.

Four experiments are presented:

- *Experiment I: Gatherers.* These tribes consume only wild cereal.
- *Experiment II: Hunter-Gatherers.* These tribes consume prey and wild cereal.
- *Experiment IIIa: Ideal Free Distribution.* These tribes consume prey and wild cereal, and farm cereal if it is energy efficient to do so.
- *Experiment IIIb: Realistic Free Distribution.* These tribes follow the same rules as IFD tribes. Unlike all the other tribes, they can only migrate to immediately adjacent patches.

For more background on these experiments, see *Agents Adopting Agriculture: Modeling the Agricultural Transition* (van der Vaart et al, 2006).

Section 1.3: User manual

This manual will guide you in installing the FHG2A application, running the simulation and interpreting the results. It assumes a basic familiarity with the Ubuntu desktop, but will provide a step-by-step guide to installing and using the software.

It will also describe the interface so that you can interpret the results of the experiments, and use the raw data and graphs that are generated.

The FHG2A application was created using a simulation framework called Repast Simphony. This manual directs you to resources for adapting this simulation or creating your own.

Section 2: Install and run the program

Section 2.1: Installation instructions

These instructions have been tested on Ubuntu 12.04 and Ubuntu 13.04 using OpenJDK 7.

1. Get the installation file (fhg2a.jar).
2. Make sure you have execution permissions for the file.
3. Run it as you normally run a jar file. Either right-click it and choose Open with... OpenJDK Java 7 Runtime, or from the command line run `java -jar fhg2a.jar` (make sure you are using Java 7).
4. A language selection window will appear. Select your language. (This application has only been tested in the default English option.)
5. Click OK.
6. A welcome screen will appear (step 1 of 8). Click Next.
7. The information screen will appear (step 2 of 8). Click Next.
8. The license screen will appear (step 3 of 8). Select the radio button next to "I accept the terms of the license agreement." Click Next.
9. The installation path screen will appear (step 4 of 8). Select your preferred installation path or leave it at the default value. Click Next.
10. If you are installing to a folder that doesn't exist yet, a popup will appear confirming that the folder will be created. Click OK.
11. The pack selection screen will appear (step 5 of 8). Keep all packs selected. Click Next.
12. The installation process will begin (step 6 of 8). Wait for the progress bar to complete. Click Next.
13. The shortcut screen will appear (step 7 of 8). These shortcuts do not work on recent Ubuntu desktops. Untick "Create shortcuts in the Start Menu". Click Next.
14. The final screen will appear (step 8 of 8). Do not click "Generate an automatic installation script". Click Done.

Section 2.2: Starting the program

Now that you have installed the program, you are ready to start it. Simply open the folder you installed it to (usually `~/FHG2A`), and double-click the `start_model.command` icon. If you are prompted to run in terminal, display, cancel or run, select "Run".

The application will take a little while to load, and then an "FHG2A – Repast Simphony" window will appear. You are now ready to use the simulation.

Section 3: Controlling the simulation

Section 3.1: Interface

Once the program has started a window containing blank panes will appear. Click the Setup button to load the simulation:

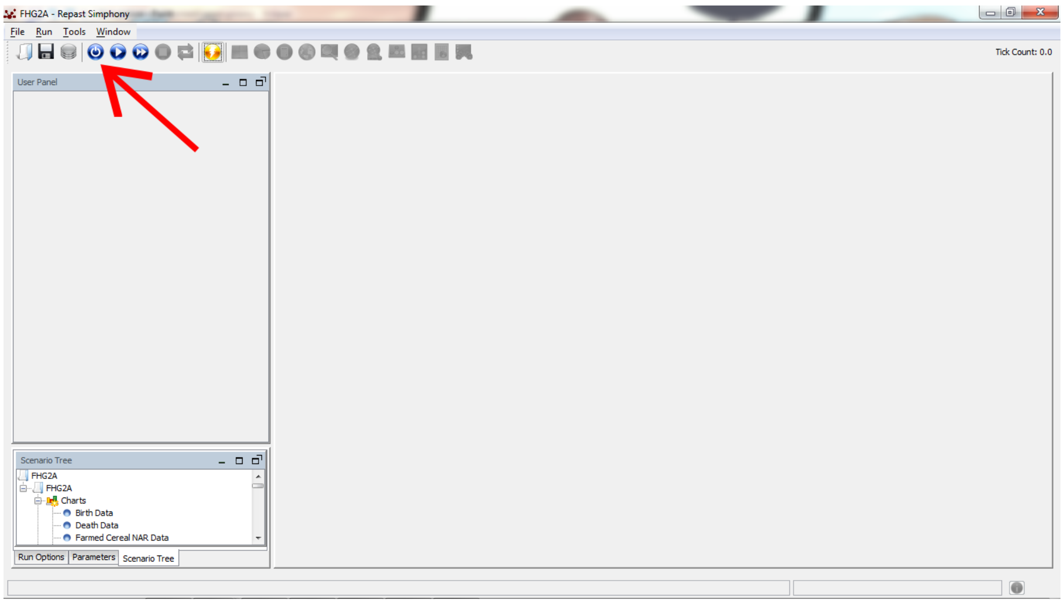
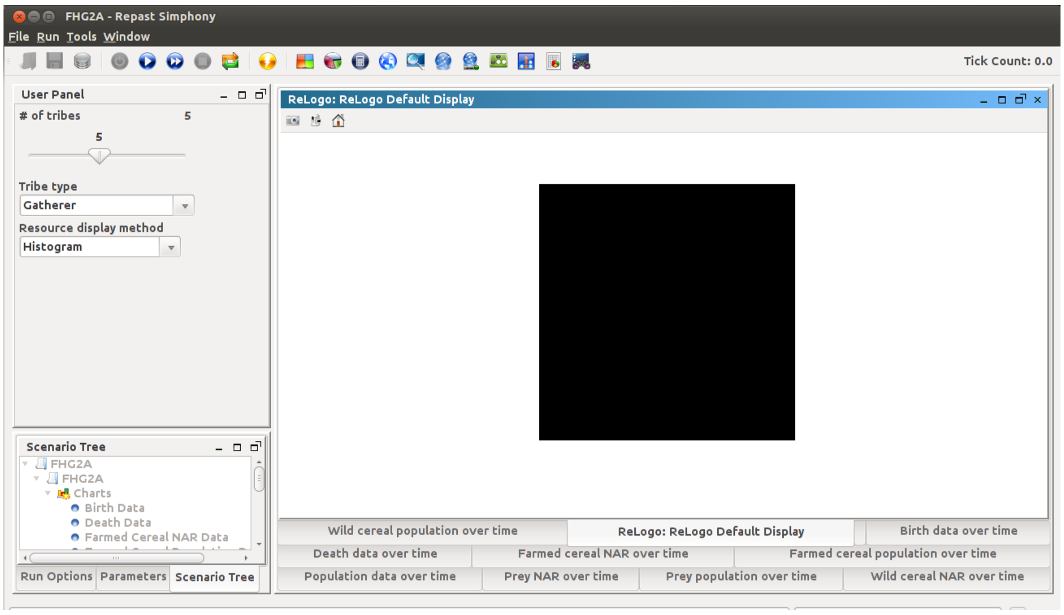


Figure 1: Blank panes before setup. Setup button highlighted.

After a moment, the interface will update with the simulation controls (these are summarised on the following page):



# of tribes 5

Tribe type: Gatherer

Resource display method: Histogram

Figure 2: Panes populated after setup.

Simulation controls

1. Setup button. This was used to load the simulation.
2. Play/pause button. This is used to start, pause and unpause the simulation.
3. Stop button. This is used to end the simulation.
4. Other toolbar buttons. These buttons provide advanced functionality that is beyond the scope of this manual.
5. Experiment. This dropdown allows you to choose which experiment to run (see *Introduction* for a brief description of each experiment).
6. Number of tribes. This slider allows you to determine the number of tribes on the grid at the start of the simulation. The default value is 5.
7. Speed. This slider allows you to determine how fast the simulation runs. Running the simulation slower allows you to observe the details of interaction more closely, running it faster allows you to observe the end state (or a cycle) more easily.
8. Scenario tree. This pane provides advanced functionality that is beyond the scope of this manual.
9. Environment pane. The black square in this pane is the environment that the simulation will appear on. To understand this display see *Interpreting the results*.
10. Tabs. These tabs give you access to other visualisation options, namely real-time graphs that update while the experiment is running.

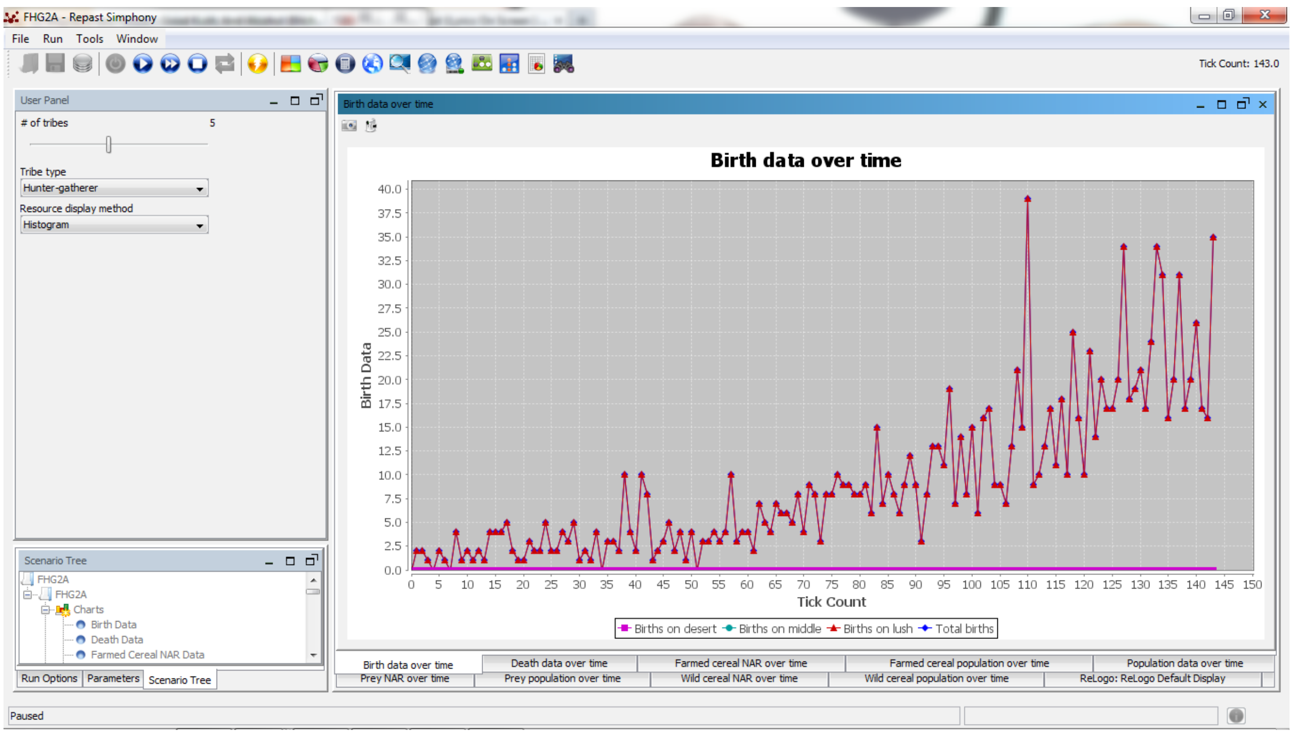
Section 3.2: Logging and graphing

The visual output is a useful tool in understanding the model, but deeper investigation requires more detailed. Because this is a simulation, we are not only able to visual the behaviour of the agents, but also record the environment state every single epoch.

These details are automatically logged every time you run the experiment. They appear as text files in the FHG2A folder with the following names (where * is replaced by the date and time that you start the simulation):

- *AgentLog*.txt* Records agent-related data.
- *FoodLog*.txt* Records resource-related data.
- *MovementLog*.txt* Records movement- and migration-related data.

Additionally, while the simulation is running you can choose to watch experimental variables (real-time graphs) rather than the visualisation. You can either select a tab before starting the simulation (preferred), or switch tabs while the simulation is running.



# of tribes 5

Tribe type: Hunter-gatherer

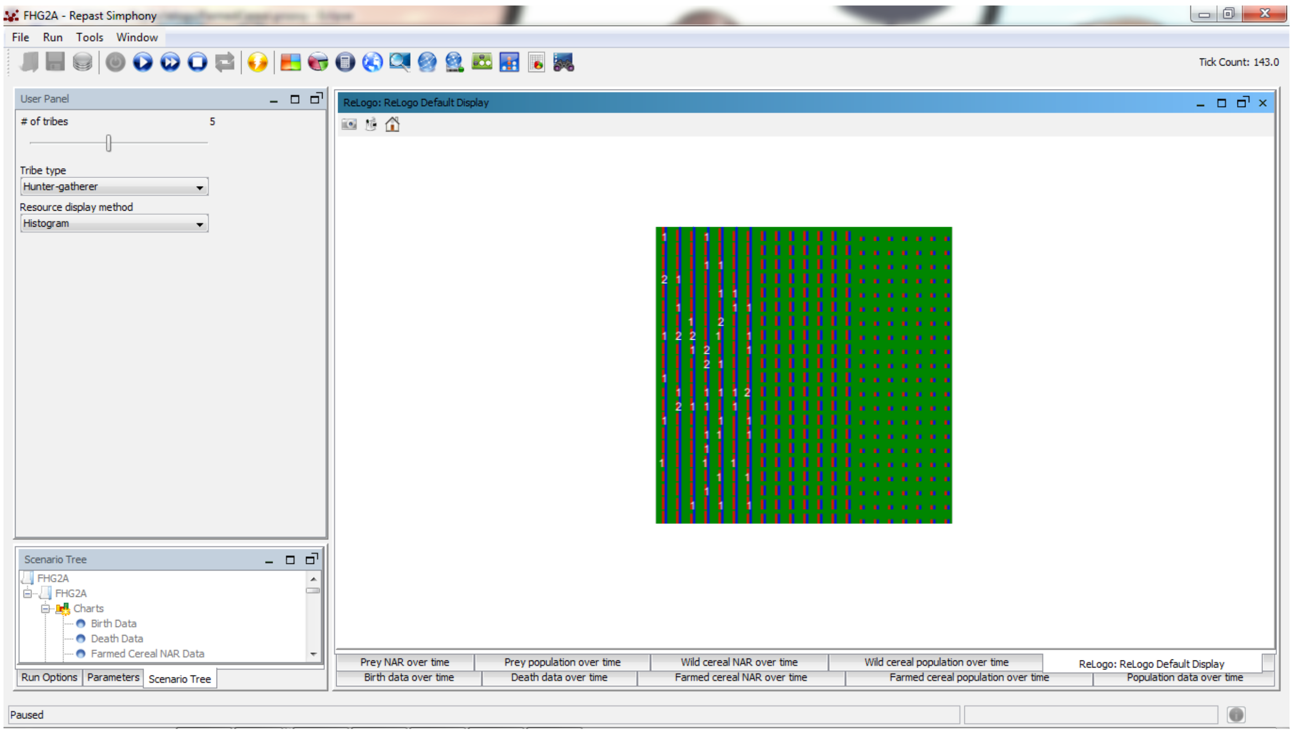
Resource display method: Histogram

Figure 3: Example real-time graph

Section 4: Interpreting the results

Section 4.1: Visualisation

To start the simulation, select the experiment conditions (or leave them at the default) and then click the Play button. After a moment, the environment will turn from black to green and numbers and histograms will appear in the environment:


FHG2A - Repast Simphony
File
Run
Tools
Window
User Panel
ReLogo: ReLogo Default Display
Scenario Tree
FHG2A
FHG2A
Charts
Birth Data
Death Data
Farmed Cereal NAR Data
Run Options
Parameters
Scenario Tree
Prey NAR over time
Prey population over time
Wild cereal NAR over time
Wild cereal population over time
ReLogo: ReLogo Default Display
Birth data over time
Death data over time
Farmed cereal NAR over time
Farmed cereal population over time
Population data over time

Tick Count: 143.0

# of tribes 5

Tribe type: Hunter-gatherer

Resource display method: Histogram

Paused

Figure 4: Simulation under way

Simulation features

1. Population labels. These numbers represent the number of individuals on a patch in units of 10. For example, 3 represents 30-39 individuals. There are two special cases: 1 represents 1-19 individuals, and X represents 100 or more individuals. These individuals may all belong to the same tribe, or to multiple tribes.
2. Resource histogram. These histograms represent the number of food resources available on a patch, where 100% is the maximum amount available in the best conditions. Red bars represent prey, blue bars represent wild cereal, and yellow bars represent farmed cereal.
3. Lush area. This region (on the left) has the best conditions for food resources to grow. They will often reach the maximum possible amount.
4. Medium area. This region (in the center) has middling conditions. Food resource will not reach the maximum amount here.
5. Arid area. This region (on the right) has poor conditions. Food resources will always be low here.

For example:

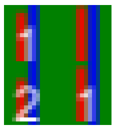


Figure 5: Example patches

In the figure above, the top right patch has no agents present. It has maximum prey and maximum wild cereal present, so we must be in the lush region.

The top left and bottom right patches have 1-19 individuals on each of them. The population on the top left have consumed some of the available prey.

The bottom left patch is currently populated by 20-29 individuals. They have consumed the most prey, but still haven't consumed as much as half of what is available.

Nobody in this diagram has started farming.

Section 4.2: Expected results

The following descriptions are direct summaries of original results by van der Vaart et al. The observed results (visually and in the logs) should match these closely:

Experiment I: Gatherers

Gatherers have knowledge of the entire world, but consume only wild cereal and cannot farm. They move to the best patch and consume as much available cereal as they need.

- Epoch 1. Tribes are drawn to the lush area. Tribe populations begin to grow.
- Epoch 267 (±5). Tribes start migrating into the medium area.
- Epoch 310 (±5). Tribes start migrating into the desert area. Tribe populations continue to grow, reaching a maximum of 0.39 (±0) agents per square kilometer.
- Epoch 326 (±8). Carrying capacity of food resources exhausted. Tribe populations crash. Survivors return to the lush area. Process restarts with cycles of approximately 267 epochs (±31).

Experiment II: Hunter-Gatherers

Hunter-gatherers have knowledge of the entire world, consume prey and wild cereal, but cannot farm. They move to the best patch and consume as much prey and wild cereal as they need, in proportion to their respective energy efficiencies.

- Epoch 1. Tribes are drawn to the lush area. Tribe populations begin to grow.
- Epoch 162 (±5). Tribes start migrating into the medium area.
- Epoch 200 (±6). Tribes start migrating into the desert area. Lush area starts to become crowded as tribes turn to wild cereal only. Crowding expands through other areas.
- Epoch 310 (±40). Carrying capacity of food resources exhausted. Tribe populations crash.

Experiment IIIa: Ideal Free Distribution

IFD tribes have knowledge of the entire world, consume prey and wild cereal, and will farm if it is more energy efficient than harvesting wild cereal. They move to the best patch and consume as much available food as they need, in proportion to energy efficiency.

- Epoch 1. Tribes are drawn to the lush area. Tribe populations begin to grow.
- Epoch 143. Tribes start migrating into the medium area.
- Epoch 181. Tribes start migrating into the desert area.
- Epoch 215. Food production begins in the lush area. Hectares of farmed land begins to grow.
- Epoch 459. Every single patch of arable land in the lush area is in use. Food production begins in the medium area.
- Epoch 492. Food production begins in the desert area.
- Epoch 499. Wild cereal becomes extinct.
- Epoch 507. All arable land is in use. Tribe populations reach an average density of 20 agents per square kilometer.
- Epoch 508. Prey becomes extinct.

Experiment IIIb: Realistic Free Distribution

RFD tribes are aware only of immediately adjacent patches. They consume prey and wild cereal, and will farm if it is more energy efficient than harvesting wild cereal. They move to the best patch in their vicinity and consume as much available food as they need, in proportion to energy efficiency.

- Epoch 1. Tribes are drawn to the lush area. Tribe populations begin to grow. Similar pattern to Experiment IIIa, but timing is slower.
- Epoch 2685. Desert area is filled with farming agents. (All arable land is in use?)

Section 5: Extending the experiment

Section 5.1: Introduction to Repast Simphony

The FHG2A simulation is written using the Repast Simphony framework, a set of tools designed for domain experts like yourself to create simulations in their field of expertise. These simulations are written in an easy-to-read, easy-to-write scripting language called ReLogo.

If you are familiar with ReLogo, this section will help orient you so that you can extend and adapt the simulation. If you are not familiar with ReLogo, we recommend reading the *ReLogo Getting Started Guide*, which provides a thorough introduction to the framework, and takes you through the steps of implementing a zombie infestation model.

Section 5.2: Loading the project

The Repast Simphony project forms a part of this submission. To open it, you will need to download and install Repast Simphony, and load the project following the instructions available on the Repast website. If you are interested in inspecting the project-specific code (as opposed to the framework code), see the folder `FHG2A/src/fhg2a/relogo`

Section 5.2: Notes for experienced programmers

Experienced programmers have probably worked in general-purpose object-oriented languages such as C++, Java, .NET or Python. ReLogo differs from these languages in that it is a *domain-specific language*, adapted to tightly fit the needs of agent-based simulation. It is in fact a specialised dialect of Groovy, an object-oriented scripting language derived from Java. As such, it is best understood as offering a familiar object-oriented paradigm within a strict framework. Thus, even if you are an experienced programmer, it is worth reading the *ReLogo Getting Started Guide* to learn about that framework.

ReLogo makes frequent use of closures, which for some may be an unfamiliar structure. For example, to execute an instruction on every turtle we call:

```
// retrieve all turtles
ask(turtles()) {

      // enter context of a single turtle

      // execute turtle.an_instruction() on current turtle
      an_instruction()

} // go to next turtle, repeat until done
```

Following the *ReLogo Getting Started Guide* will orientate you to their use if you have not encountered them before.

Finally, note that it is possible to program Repast Simphony simulations in languages other than ReLogo, or adapt ReLogo or the Repast Simphony framework themselves, but those topics are beyond the scope of this manual.