Analyzing Repast Simphony models in R

Antonio Bucchiarone

Fondazione Bruno Kessler, Trento – Italy
bucchiarone@fbk.eu

20 November 2019

Model Calibrations

- **Model calibration** is the task of estimate the set of values for input parameters of some simulation model.
  - Best fitting to any empirical data set available for the system under study.
  - To understand the *relative impact* of input parameters on the overall model outcomes.
- User friendly tools for experiment design and analysis would help modelers to improve the formal quality of their models.

R and Repast

FONDAZIONE BRUNO KESSLER

- Repast Simphony has support for running GNU R code from inside the framework user interface.
- Was not feasible running Repast models from R environment for controlling model in orde to implement
  - experimental designs,
  - parameter calibration, and
  - sensitivity analysis.
- **RRepast** open source project developed for invoking Repast Simphony models from inside GNU R environment.

https://cran.r-project.org/web/packages/rrepast/

RRepast Package

- The package contains R and Java code for *linking* the calls to the Repast subsystem.

- The package has two main groups of functions:

  - The **first** directly related to the *integration of Repast with R*, allowing the i*nstantiation*, *execution* and *control* of a model execution, as well as, gathering model output generated by any aggregated dataset defined into Repast model.

  - The **second** relies on the first group for running model but exposing a complete set of methods for parameter calibration,

RRepast in Action

- How to use the RRepast package for running Repast models.

1. Build an installer and install the Repast model.
2. Add the rrepast-integration.jar file, included in Rrepast distribution, to the lib directory of the installed Repast model.
3. Add the integration configuration to scenario file in the .rs directory of the installed model. The integration consists in the following code:

scenario.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Scenario>
<repast.simphony.action.file_sink context="Predator Prey" file="repast.simphony.action.file_sink_0.xml" />
<repast.simphony.action.data_set context="Predator Prey" file="repast.simphony.action.data_set_1.xml" />
<repast.simphony.dataLoader.engine.ClassNameDataLoaderAction context="Predator Prey" file="repast.simphony.dataLoader.engine.
ClassNameDataLoaderAction_2.xml" />
<repast.simphony.action.display context="Predator Prey" file="repast.simphony.action.display_3.xml" />
<repast.simphony.action.display context="Predator Prey" file="repast.simphony.action.display_4.xml" />
<repast.simphony.action.time_series_chart context="Predator Prey" file="repast.simphony.action.time_series_chart_5.xml" />
<model.initializer class="org.haldane.rrepast.ModelInitializerBroker" />
</Scenario>
```

Running the model in RRepast

- Minimal code to execute a Repast model from R

```
1  library(rrepast)
2
3  # The directory where The repast model has been installed
4  install.dir<- "c:/models/themodel"
5
6  # Instantiate and load the model
7  obj<- Model(modeldir=install.dir, dataset="dataset", TRUE)
8
9  # Run the model
10 Run(obj)
11
12 # The model's output
13 output<- GetResults()
```

Basic RRepast API Functions

Model(d, t, o, l)	This function creates an object instance for linking the Repast model to an R object. The required parameters are the directory where the model has been installed *(d)*, the duration of simulation in Repast ticks *(t)*, the name of any aggregated dataset of model for draining data generated by the model simulation*(o)* and a Boolean flag which tells the function to call the Load method. The default value is FALSE.
Load(m)	This function loads the Repast scenario from model's directory. The only required parameter *(m)* is an instance of Repast Model created with previous function.
Run(m, r, s)	The purpose of this function is to execute a single round of simulation using just one parameter set. The parameters for this function are a model instance *(m)*, the number of repetitions *(r)* and a collection the random seeds *(s)* to be used for each one of the repetitions. The only required parameter is the model instance, created with the **Model()** function. The default value for *r* is one.

Basic RRepast API Functions

RunExperiment(m, r, d, F)	Execute a complete experimental setup for different set of parameters. The parameters required are a model instance *(m)*, the number of replications *(r)*, the experimental design *(d)* and finally a user provided calibration function *(F)*. The experimental design parameter is an **R** data frame containing a complete set of model's parameter per row. The function returns a list with three data frame elements: the *paramset*, the *output* and *dataset* which holds respectively all simulated input parameters, the result of user provide calibration function and the complete dataset produced during the experiment execution.
GetSimulationParameters(e)	Returns the complete list of parameters declared by the model.
SetSimulationParameters(e, p)	Modify several parameters at once.
SaveSimulationData(t, e)	Exports the results of Run or RunExperiment to a csv or excel files.

RRepast API

- **Execution and control of Repast Simphony code.**
  - Instantiating and running a Repast Simphony model
  - Retrieving the declared model parameters, getting their default values, setting parameter values as well as running basic experimental design.
  - Saving simulation data.

- **Basic functions for experimental design.**
  - Adding the input factors and the relevant input range which the modeler wants to evaluate.
  - Functions for generating the experiments input using different sampling approaches.

- **High-level functions for a complete experimental in one call**.
  - «Easy» API functions
  - Complete method implementation which is accessible using just one R function call.

<<Easy>> APIs

- **Easy.Stability**
  - A Method for finding the *number of replications of simulation experiments* required for obtaining consistent outputs.

- **Easy.Calibration**
  - It estimates the best set of input parameters or factors, performing a set of model executions in order to sample the calibration function.
  - The objective is to minimize the output of calibration function provided by the user.

- **Easy.Sobol**
  - Sensitivity analysis is the task of evaluating the sensitivity of a model output Y to input variables (X1,…,Xp).

Second Assignment

- **Deadline:** Friday 20, December - 6pm

Ion Stoica, Robert Tappan Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan: **Chord: a scalable peer-to-peer lookup protocol for internet applications.** IEEE/ACM Trans. Netw. 11(1): 17-32 (2003)

Results from theoretical analysis and simulations show that ***Chord is scalable***: *Communication cost and the state maintained by each node scale logarithmically with the number of Chord nodes.*