Tutorial on using the *spartan* package to analyse agent-based simulation results

For use with *spartan* package version 1.2 onwards

Technique 5: SPARTAN and Netlogo

1 Introduction

spartan, or (**S**imulation **P**arameter **A**nalysis **R** **T**oolkit **A**pplicatio**N**) is an R package which aids the understanding of the effect aleatory and epistemic uncertainty have on the output from a simulation. This set of tutorials makes use of available example simulation output to demonstrate how a variety of methods can be applied to further understand the results that have been generated. Following through each example should make it easier to apply the tookit to results generated by any agent-based computer simulation. This tutorial focuses on how SPARTAN can provide parameter samples for and analyse Netlogo Simulations.

2 The *spartan* Package

Computer simulations are becoming a popular technique to use in attempts to further our understanding of complex systems. This package provides code for four techniques described in available literature which aid the analysis of simulation results, at both single and multiple timepoints in the simulation run. The first technique addresses aleatory uncertainty in the system caused through inherent stochasticity, and determines the number of replicate runs necessary to generate a representative result. The second examines how robust a simulation is to parameter perturbation, through the use of a one-at-a-time parameter analysis technique. Thirdly, a latin hypercube based sensitivity analysis technique is included which can elucidate non-linear effects between parameters and indicate implications of epistemic uncertainty with reference to the system being modelled. Finally, a further sensitivity analysis technique, the extended Fourier Amplitude Sampling Test (eFAST) has been included to partition the variance in simulation results between input parameters, to determine the parameters which have a significant effect on simulation behaviour.

3 The Case Study

To ensure our demonstration in this tutorial can be replicated, we will utilise a model that was available in the Model Library, albeit with a small number of modifications.

The virus model (Wilensky, 1998) simulates how a virus is transmitted and perpetuated amongst a population, based on a number of factors. The model is fully described with the simulation, so we direct readers to study that detail prior to performing this tutorial. As an overview, there are four parameters: the number of people in the population (people), the ease at which the virus spreads (infectiousness), the probability a person recovers from the virus (chance-of-recover), and the duration (weeks) after which the person either recovers or dies (duration). This tutorial will aid researchers understand how a Netlogo model can be better understood using the parameter analysis techniques available within SPARTAN. We recommend that you have read either the tutorials for Techniques 1-4 or through or PLoS Computational Biology paper (Spartan: A Comprehensive Tool for Understanding Uncertainty in Simulations of Biological Systems, Alden et al 2013) before commencing this tutorial.

The version of the model in the library has no defined end point, so our first change is to stop the simulation after 100 years. Secondly, judgements concerning each parameter are constructed by varying the value the parameter is assigned and determining the effect on simulation response. The current model states the percentage of people at the current timepoint who are infected and immune. As we want to examine performance across the simulation, we have added four output measures: counts of the number of people who have died through not recovering from the infection (death-thru-sickness), the number who died but were immune (death-but-immune), the number who died through old age and never caught the infection (death-old-age), and the number of people who died while infected but during the time period allowed for recovery (death-old-and-sick). We then use SPARTAN to determine how three of the four parameters above (we exclude people for the reasons this is linked to the output responses) impact simulation behaviour. Thirdly, for reasons that will become clearer later, we introduce a new global parameter, dummy, which has no role in the code, and thus no impact on the simulation,

To ease the reproduction of results in this tutorial, we have made the modified version of the model available from the SPARTAN website (www.ycil.org.uk)

4 Scope

Do note that the idea of this tutorial is to demonstrate the application of the toolkit, and is not intended to act as a full introduction to using Sensitivity Analysis techniques in the analysis of simulation results. Where useful, links to further reading have been included.

5 Prerequisites

- The R statistical environment, version 2.13.1 or later.
- The spartan R package, downloaded from the Comprehensive R Archive Network (CRAN) or from the project website.
- The lhs, gplots, and XML R packages, available for download from CRAN.
- The Netlogo Virus model, available for download from our website if you want to run the model yourself, or the sample data from our runs.
- We used the Headless version of Netlogo to run the simulations for all the sets of parameters that were generated for all techniques in this tutorial. To aid the adoption of this approach, the scripts we used to do this are all available on our website (www.ycil.org.uk).

6 Parameter Robustness (Technique 2 in SPARTAN)

The robustness of a Netlogo simulation to parameter alteration can be determined through the use of this approach. Following the method described by Read et al (2012), a set of parameters of interest are identified, and a range of potential values each parameter could lie within is assigned. The technique examines the sensitivity to a change in one parameter. Thus, the value of each is perturbed independently, with all other parameters remaining at their calibrated value.

This technique works with the Netlogo BehaviourSpace feature. In that feature, you can specify a range to explore for each parameter, and Netlogo will construct an XML file from this information and run the experiments. The results are stored in a CSV table, which the researcher can then analyse. SPARTAN works by producing this XML file without the need to be in Netlogo, and then once the researcher has performed runs based on the information in that file, SPARTAN can analyse the resultant data. Note that this tutorial builds on the information in Technique 2 rather than replaces it, so it is recommended that you are aware of the detail in that tutorial.

6.1 Parameter Sampling

1. Open a text editor (gedit or similar). Now we are going to declare the variables required by the package to produce the Netlogo experiment file. Type or copy the declarations below. Firstly, the *spartan* and *XML* libraries are imported, the latter to aid production of the Netlogo file. The variables required for this analysis are then declared in capital letters. The line underneath, beginning with a #, is a description of that being declared. Make sure you set the FILEPATH variable correctly to match the folder where you would like the Netlogo experiment file to be output to.

```
library(spartan)
library(XML)

# Directory where the sample file should be stored
FILEPATH<-"/home/user/robustness/"

# Name to give the setup file. No need for file extension
NETLOGO_SETUPFILE_NAME<-"nl_robustness_setup"

# Parameters in simulation. List all the parameters that Netlogo is
# required to know,
PARAMETERS<-c("people","infectiousness","chance-recover","duration")

# Now values for each of the parameters above
# For parameters not being analysed, simply list the value
# For parameters being analysed, put the min, max and increment values
# of the  parameter in square brackets, in double quotes, e.g.
# "[0.1,0.5,0.1]"
# Encapsulate strings, i.e.: "\"/home/user/Experiment/\""
PARAMVALS<-c(150,"[10,90]","[10,90]","[5,40]")

# Name of the setup simulation function in Netlogo simulation
# Usually setup
NETLOGO_SETUP_FUNCTION<-"setup"

# Name of the function that starts the simulation. Usually go
NETLOGO_RUN_FUNCTION<-"go"

# Simulation output measures
MEASURES<-c("death-thru-sickness","death-but-immune",
"death-old-age","death-old-and-sick")

# Number of times Netlogo should repeat the experiment for each
# parameter set
EXPERIMENT_REPETITIONS<-1

# Whether Netlogo should collect metrics at each timestep
RUNMETRICS_EVERYSTEP<-"true"
```

2. To get the value sets for each parameter, the following method is used. Copy the text below into the R script file below the declarations you have made above:

```
oat_generate_netlogo_behaviour_space_XML(FILEPATH,
NETLOGO_SETUPFILE_NAME,PARAMETERS,PARAMVALS,NETLOGO_SETUP_FUNCTION,
```

```
NETLOGO_RUN_FUNCTION,MEASURES,EXPERIMENT_REPETITIONS,
RUNMETRICS_EVERYSTEP)
```

3. Save the file with a suitable filename and .R extension (for example OAT_Netlogo_Sampling.R)

4. If you are using Linux or Mac OS, open a command prompt, and navigate to the directory where this file was saved. Type the following:

   ```
   Rscript OAT_Netlogo_Sampling.R
   ```

   If you are using Windows, open R and type the following into the R Command Prompt (where [*path to directory* is the full path to where OAT_Netlogo_Sampling.R is saved]:

   ```
   source("C:\[\textit{path to directory}]\OAT_Netlogo_Sampling.R")
   ```

5. This will produce one XML file, in the directory you specified, containing a set-up for a Netlogo BehaviourSpace run. Run the experiment using Netlogo (we did this in the terminal in Linux using the headless version of Netlogo, code for this available on our website, but running this in Netlogo is fine too). This will in turn produce one CSV file, saved in the directory of your choosing when you run the experiment.

6.2 Analysing The Simulation Data

This tutorial is going to assume that you are using the example data from our lab website, but the steps are just as applicable if you ran the experiments yourself.

1. Download the Virus Robustness Experiment result set from the project website

2. With this data available, open a text editor (gedit, notepad, or similar). Now we are going to declare the variables required to run this analysis. Type or copy in the following text. Again, the *spartan* library is imported. The variables required for this analysis are then declared in capital letters. The line underneath, beginning with a #, is a description of that being declared. Make sure you set the FILEPATH variable correctly to state where the simulation results have been extracted to.

   ```
   # Import the package
   library(spartan)
   library(XML)

   # Folder containing the Netlogo Behaviour Space table, AND where the
   # processed results will be written to
   FILEPATH<-"/home/user/robustness/"
   # Name of the Netlogo Behaviour Space Table file
   NETLOGO_BEHAVIOURSPACEFILE<-"virus_oat.csv"
   # Array of the parameters to be analysed, but ONLY those perturbed
   PARAMETERS<-c("infectiousness","chance-recover","duration")
   # Value assigned to each parameter at calibration (the baseline value)
   BASELINE<-c(60,50,20)
   # The maximum value for each parameter
   PMAX<-c(90,90,40)
   # The minimum value explored for each parameter
   PMIN<-c(10,10,5)
   # Amount the parameter value was incremened during sampling
   PINC<-c(10,10,5)
   ```

```
# Timestep of interest. The behaviour space table is likely to contain
# all timesteps - this narrows the analysis
TIMESTEP<-5200
# The simulation output measures being examined. Should be specified
# as they are in the Netlogo file
MEASURES<-c("death-thru-sickness","death-but-immune","death-old-age",
"death-old-and-sick")
# For each parameter value being analysed, a file is created
# containing the median of each output measure, of each simulation run
# for that value. This sets the name of this file. No file extension
RESULTFILENAME<-"ParamValResponses"
# File format of median results file that will be generated in this
# tutorial. Should be either XML or CSV. This is used as, for some
# applications, a simulation results set for processing may have been
# generated using methods other than spartan, and may not be in CSV
# format.
RESULTFILEFORMAT<-"csv"
# The results of the A-Test comparisons of each parameter value
# against that of the parameters baseline value are output as a file.
# This sets the name of this file. Note no file extension.
# Current versions of spartan output to CSV files
ATESTRESULTSFILENAME<-"VirusOAT_ATests"
# A-Test result value either side of 0.5 at which the difference
# between two sets of results is significant
ATESTSIGLEVEL<-0.21
# What each measure represents. Used in graphing results
MEASURE_SCALE<-c("Number of People","Number of People",
"Number of People","Number of People")
# Not used in this case, but when a simulation is analysed at multiple
# timepoints (see tutorials 1-4)
TIMEPOINTS<-NULL; TIMEPOINTSCALE<-NULL
```

3. Now to run the first of the four methods (we are going to do each individually in the tutorial so the functionality becomes apparent but in reality you will run all three methods one after another in the same text file). Copy the below into the text file under the above declarations:

   ```
   oat_process_netlogo_result(FILEPATH,NETLOGO_BEHAVIOURSPACEFILE,
   PARAMETERS,BASELINE,PMIN,PMAX,PINC,MEASURES,RESULTFILEFORMAT,
   RESULTFILENAME,TIMESTEP)
   ```

   As stated in Technique 2, SPARTAN looks to process files in a specific folder structure: top level the parameter being studied, next level the value assigned to that parameter, next level the simulation runs themselves. Here, all the results, for all parameters and values, are in one file. So the above technique processes this file, recovering the results for each parameter and value pair and creating the folder structure specified/shown in Technique 2. Now, it is possible to use techniques already availble in SPARTAN to process this data, rather than write new techniques.

4. Save the file with a suitable filename and .R extension (for example OAT_Analyse_Result.R) and run the script (see the instructions in Parameter Sampling above). If you look in the folder that you specified in the FILEPATH, you will now see a folder structure as described, one for each parameter being analysed.

5. We are now going to run the second method, which gives an indication of the change in behaviour caused by this parameter value change. In the same file in the text editor, add the following method call, then save the file.

```
oat_netlogo_analyseAllParams(FILEPATH,PARAMETERS,BASELINE,PMIN,
PMAX,PINC,MEASURES,RESULTFILEFORMAT,RESULTFILENAME,
ATESTRESULTSFILENAME)
```

   Again, this method examines each parameter in turn. The aim is to compare the Netlogo results generated above with those obtained when the parameter was set to its calibrated, baseline, value. As an example, consider one of the parameters we are analysing in the example set, duration. This has a baseline value of 20. The simulation has four output measures. The method will read in the results generated by the first method where the parameter value is its baseline value. Then, it will, in turn, read in the median results for every other parameter value. So, for example, the minium value for duration is 5. The method will read in the responses for all simulation runs for this value. For each output measure, the responses are then compared using the Vargha-Delaney A-Test (2000), which indicates 'how different' the two sets of results are, or the effect the parameter value change has had on the simulation result. The next value for this parameter is then considered, and the same process followed. Once this is all complete, the A-Test scores for each parameter value comparison are output to a CSV file.

6. Return to the command prompt and run the R script using the same command as previously. This will then generate a results file for each parameter, within that parameters folder, summarising the A-Test results for all values it has been assigned. These files will take the name set in the R variable ATESTRESULTSFILENAME. Navigate to one of the parameter folders and make sure you are familiar with the structure of this file.

7. Go back to the text editor. We are now going to use the two methods within the package to represent the results generated previously in a graphical format. Add the following to the script text file:

```
oat_netlogo_graphATestsForSampleSize(FILEPATH,PARAMETERS,PMIN,
PMAX,PINC,MEASURES,ATESTSIGLEVEL,
ATESTRESULTSFILENAME,TIMEPOINTS,TIMEPOINTSCALE)

oat_netlogo_plotResultDistribution(FILEPATH,PARAMETERS,PMIN,
PMAX,PINC,MEASURES,MEASURE_SCALE,RESULTFILEFORMAT,
RESULTFILENAME,TIMEPOINTS,TIMEPOINTSCALE)
```

8. Return to the command prompt and run the R script using the same command as previously.

   The first of these methods produces a graph for each parameter, which shows the A-Test scores for each value it has been assigned, when compared to its calibrated value. This plot will show all simulation output measures on the same graph. Good examples can be seen in Tutorial 2. This technique helps identify any trends that become apparent when the value of a parameter is changed.

   The final method again produces a graph for each parameter, but this time also for each simulation output measure. This generates a box plot showing the distribution of simulation results for that measure, for each parameter value, again making it easy to identify any trends in simulation behaviour against parameter value.

7 Latin-Hypercube Sampling and Analysis (Technique 3 in SPARTAN)

Though Technique 2 of this toolkit elucidates the effects of perturbations of one parameter, it cannot show any non-linear effects which occur when two or more are adjusted simultaneously. This can be achieved using Technique 3, a Global Sensitivity Analysis technique. A number of parameter value sets are created through a latin-hypercube sampling approach, which selects values for each parameter from the parameter space, while aiming to reduce any possible correlations when the sample is produced. SPARTAN then constructs a Netlogo experiment file for each parameter value set. The researcher then runs these in Netlogo. With there being a number of experiment files constructed with this technique, we would recommend that the reader uses a scripting language to perform these experiments, using the headless version of Netlogo. The script we have used to do this can be found on our lab website as an example. Once the runs are complete, SPARTAN then analyses the resultant data, revealing any correlations between parameter and value, and thus indicating the paramters of greatest influence on the simulation.

7.1 Parameter Sampling

1. Open a text editor. Now we are going to declare the variables required by the package to produce the parameter value sets. Type or copy in the text below.

   The last variable, ALGORITHM, controls whether a fully optimised latin-hypercube algorithm is used, or parameter values chosen from each section of the hypercube randomly. Both these algorithms are taken from the lhs package. Note that although an optimised sample may be preferable, the generation of parameter values using an optimal algorithm may take a long time (in our experience, over 24 hours for just 7 parameters). The ALGORITHM variable can be set to either "normal" or "optimal"

```
library(spartan)
library(lhs)
library(XML)

# Directory where the samples should be stored
FILEPATH<-"/home/user/LHC/"

# Parameters in simulation. List all the parameters that Netlogo is
# required to know,
PARAMETERS<-c("people","infectiousness","chance-recover","duration")

# Now values for each of the parameters above
# For parameters not being analysed, simply list the value
# For parameters being analysed, put the min and max values of the
# parameter in square brackets, in double quotes, e.g. "[0.1,0.5]"
# Encapsulate strings, i.e.: "\"/home/user/Experiment/\""
PARAMVALS<-c(150,"[10,90]","[10,90]","[5,40]")

# Number of parameter samples to take from hypercube
NUMSAMPLES<-500

# Name of function that sets up Netlogo simulation. Usually setup
NETLOGO_SETUP_FUNCTION<-"setup"

# Name of function that starts Netlogo simulation. Usually go
NETLOGO_RUN_FUNCTION<-"go"
```

```
# Simulation output measures
MEASURES<-c("death-thru-sickness","death-but-immune","death-old-age",
"death-old-and-sick")

# Number of times the Netlogo experiment is repeated for each parameter
# set
EXPERIMENT_REPETITIONS<-1

# Whether Netlogo metrics should be collected at each timestep
RUNMETRICS_EVERYSTEP<-"true"

# Algorithm to use to generate the hypercube (normal or optimal)
ALGORITHM<-"normal"
```

2. To get the value sets for each parameter, and the Netlogo XML experiment files, the following method is used. Copy the text below into the R script file below the declarations you have made above.

```
lhc_generate_lhc_sample_netlogo(FILEPATH,PARAMETERS,PARAMVALS,
NUMSAMPLES,ALGORITHM,EXPERIMENT_REPETITIONS,RUNMETRICS_EVERYSTEP,
NETLOGO_SETUP_FUNCTION,NETLOGO_RUN_FUNCTION,MEASURES)
```

3. Save the script and run as described previously. In the file path you have specified in your script, you will now find 500 sets of Netlogo experiment files, and one CSV file summarising the values selected from the hypercube.

7.2 Analysing The Simulation Data

This tutorial is going to assume that you are using the example data from our lab website, but the steps are just as applicable if you ran the experiments yourself.

1. Download the Virus Latin-Hypercube Experiment result set from the project website. Note for space reasons that the Netlogo parameter files themselves are not included in the download.

2. With this data available, open a text editor. Now we are going to declare the variables required to run this analysis. Type or copy in the following text. Again, the *spartan* library is imported. The variables required for this analysis are then declared in capital letters. The line underneath, beginning with a #, is a description of that being declared. Make sure you set the FILEPATH variable correctly to state where the simulation results have been extracted to.

```
# Import the package
library(spartan)
library(XML)

# Folder containing the Netlogo Behaviour Space table,
# and where the processed results will be written to
FILEPATH<-"/home/user/LHC/"
# Name of the result file generated by Netlogo. Note the sample number
# and CSV is appended to this
LHCSAMPLE_RESULTFILENAME<-"lhcResult"
```

```
# Number of parameter samples generated from the hypercube
NUMSAMPLES<-500
# Timestep of interest. The behaviour space table is likely to contain
# all timesteps - this narrows the analysis
TIMESTEP<-5200
# Parameters of interest in this analysis
PARAMETERS<-c("infectiousness","chance-recover","duration")
# The simulation output measures being examined. Should be specified
# as they are in the Netlogo file
MEASURES<-c("death-thru-sickness","death-but-immune","death-old-age",
"death-old-and-sick")
# What each measure represents. Used in graphing results
MEASURE_SCALE<-c("Number of People","Number of People",
"Number of People","Number of People")
# For each parameter value being analysed, a file is created
# containing the median of each output measure, of each simulation
# run for that value. This sets the file name. No file extension
RESULTFILENAME<-"ParamValResponses"
# File format of median results file that will be generated in
# this tutorial. Should be either XML or CSV. This is used as,
# for some applications, a simulation results set for processing
# may have been generated using methods other than spartan, and
# may not be in CSV format.
RESULTFILEFORMAT<-"csv"
# Location of a file containing the parameter value sets
# generated by the hypercube sampling (i.e. the file generated
# in the previous method of this tutorial. Unlike above, an
# extension should be specified.
LHC_PARAM_CSV_LOCATION<-"/home/user/LHC_Parameters_for_Runs.csv"
# File format of median results file that will be generated in this
# tutorial. Should be either XML or CSV. This is used as, for some
# applications, a simulation results set for processing may have
# been generated using methods other than spartan, and may not be
# in CSV format.
MEDIANSFILEFORMAT<-"csv"
# For each parameter value set being analysed, a file is created
# containing the median of each output measure, of each simulation
# run for that value. This sets the name of this file. Note no
# file extension
MEDIANSFILENAME<-"ParamValResponses"
# File name to give to the summary file that is produced showing
# the parameter value sets alongside the median results for each
# simulation output measure. Note no file extension
LHCSUMMARYFILENAME<-"Virus_LHCSummary"
# File name to give to the file showing the Partial Rank Correlation
# Coefficients for each parameter. Again note no file extension
CORCOEFFSOUTPUTFILE<-"EgSet_corCoeffs"
# Not used in this case, but when a simulation is analysed at
# multiple timepoints (see Tutorials 1-4)
TIMEPOINTS<-NULL; TIMEPOINTSCALE<-NULL
```

3. Now to examine the first of the four methods (we are going to do each individually in the tutorial so the functionality becomes apparent but in reality you will run all three methods one after another in the same text file). Copy the below into the text file under

the above declarations:

```
lhc_process_netlogo_result(FILEPATH,LHCSAMPLE_RESULTFILENAME,
NUMSAMPLES,MEASURES,RESULTFILEFORMAT,RESULTFILENAME,TIMESTEP)
```

4. Save the script with an appropriate name and .R extension, and run as described previously. This method has transferred the Netlogo experiment result into a format that can be processed by SPARTAN, CSV files. Each parameter value set is summarised, with the results from the Netlogo run(s) stored as a CSV row. The methods used in Technique 3 can now be used to process this data.

5. Back in the text editor, add the following method declaration to the file, then run the script.

   ```
   lhc_generate_netlogo_LHCSummary(FILEPATH,LHC_PARAM_CSV_LOCATION,
   PARAMETERS,NUMSAMPLES,MEASURES,MEDIANSFILEFORMAT,
   MEDIANSFILENAME,LHCSUMMARYFILENAME)
   ```

   This again goes through each parameter value set in turn, but this time looks at the Netlogo responses generated above for each run. The median of these responses, for each simulation output measure, is calculated. The method then reads in the values that were assigned for each parameter in that run from the sampling result CSV file stated in LHC_PARAM_CSV_LOCATION (as was generated during sampling) and stores the median simulation results alongside the parameter values that were assigned. Once this is completed for all parameter value sets, this is output as a CSV file, filename specified in LHCSUMMARYFILENAME.

6. With this summary produced, it is now possible to elucidate any non-linear effects that are apparent when all parameter values are perturbed concurrently. To get a statistical measure of the presence of any effects, the Partial Rank Correlation Coefficient (PRCC) is generated for each parameter. This examines each parameter in turn and each simulation output value, considering the value it has been assigned against the simulation result. The result of the calculation for each parameter is output to a CSV file, with filename as specified by CORCOEFFSOUTPUTFILE. Add the following declaration and run the script:

   ```
   lhc_generate_netlogo_PRCoEffs(FILEPATH,PARAMETERS,MEASURES,
   LHCSUMMARYFILENAME,CORCOEFFSOUTPUTFILE)
   ```

7. Go back to the text editor. We are now going to represent the results graphically, making any effects present easier to notice. Add the following to the script text file:

   ```
   lhc_netlogo_graphMeasuresForParameterChange(FILEPATH,PARAMETERS,
   MEASURES,MEASURE_SCALE,CORCOEFFSOUTPUTFILE,
   LHCSUMMARYFILENAME,TIMEPOINTS,TIMEPOINTSCALE)
   ```

   This produces a graph for each parameter of interest, and each simulation output measure for that parameter, plotting the value that parameter was assigned against the simulation result. This makes trends easy to identify. For each graph, the respective Partial Rank Correlation Coeffient is included in the title to give a statistical measure to the results seen visually.

8 eFAST Sampling and Analysis (Technique 4 in SPARTAN)

This technique analyses simulation results generated through parametering using the eFAST approach (extended Fourier Amplitude Sampling Test, Saltelli et al, reference at end of tutorial). This perturbs the value of all parameters at the same time, with the aim of partitioning the variance in simulation output between input parameters. Values for each parameter are chosen using fourier frequency curves through a parameters potential range of values. A selected number of values are selected from points along the curve. Though all parameters are perturbed simultaneously, the method does focus on one parameter of interest in turn, by giving this a very different sampling frequency to that assigned to the other parameters. Thus for each parameter of interest in turn, a sampling frequency is assigned to each parameter and values chosen at points along the curve. So a set of simulation parameters then exists for each parameter of interest. As this is the case, this method can be computationally expensive, especially if a large number of samples is taken on the parameter search curve, or there are a large number of parameters. On top of this, to ensure adequate sampling each curve is also resampled with a small adjustment to the frequency, creating more parameter sets on which the simulation should be run. This attempts to limit any correlations and limit the effect of repeated parameter value sets being chosen. Thus, for a system where 8 parameters are being analysed, and 3 different sample curves used, 24 different sets of parameter value sets will be produced. Each of these 24 sets then contains the parameter values chosen from the frequency curves. This number of samples should be no lower than 65 (see the Marino paper for an explanation of how to select sample size).

Once the sampling has been performed, simulation runs should be performed for each set generated. The eFAST algorithm then examines the simulation results for each parameter value set and, taking into account the sampling frequency used to produce those parameter values, partitions the variance in output between the input parameters.

SPARTAN has the ability to sample the space using the technique, produce Netlogo experiment files for these samples, through which the simulations can be run, and then analyse the resultant simulations

8.1 Parameter Sampling

1. Open a text editor. Now we are going to declare the R variables required by the package to produce the parameter value sets. Type or copy in the text below. Firstly, the *spartan* library is imported. The variables required for this analysis are then declared in capital letters. The line underneath each, beginning with a #, is a description of that being declared. Make sure you set the FILEPATH parameter correctly to match the folder where you would like the parameter value sets to be output to. Note here the additional parameter: Dummy. Statistical inference in eFAST is generated though comparing the variance of each parameter to that of a parameter known to have no influence on the simulation. The dummy parameter sample fulfils this role.

```
library(spartan)
library(lhs)
library(XML)

# The directory where the samples should be stored
FILEPATH<-"/home/user/eFAST/"

# Parameters for this simulation. List ALL the parameters that
# Netlogo needs to know, even those not being perturbed. Make sure
# you include the dummy
PARAMETERS<-c("people","infectiousness","chance-recover","duration",
"dummy")
```

```
# Now values for each of the parameters above
# For parameters not being analysed, simply list the value
# For parameters being analysed, put the min and max values of the
# parameter in square brackets, in double quotes, e.g. "[0.1,0.5]"
# Encapsulate strings, i.e.: "\"/home/user/Experiment/\""
PARAMVALS<-c(150,"[10,90]","[10,90]","[5,40]","[1,10]")

# Number of resampling curves to use
NUMCURVES<-3

# Number of value samples to take from each parameter curve
NUMSAMPLES<-65

# The name of the function in Netlogo that sets up the simulation
# Usually setup
NETLOGO_SETUP_FUNCTION<-"setup"

# Name of the function in Netlogo that starts the simulation
# Usually go
NETLOGO_RUN_FUNCTION<-"go"

MEASURES<-c("death-thru-sickness","death-but-immune",
"death-old-age","death-old-and-sick")

# Number of times the Netlogo run should be repeated for each parameter
# set
EXPERIMENT_REPETITIONS<-1

# Whether Netlogo should collect metrics at all timesteps
RUNMETRICS_EVERYSTEP<-"true"
```

2. To get the value sets for each parameter, the following method is used. Copy the text below into the R script file below the declarations you have made above:

```
efast_generate_sample_netlogo(FILEPATH,NUMCURVES,NUMSAMPLES,MEASURES,
PARAMETERS,PARAMVALS,EXPERIMENT_REPETITIONS,RUNMETRICS_EVERYSTEP,
NETLOGO_SETUP_FUNCTION,NETLOGO_RUN_FUNCTION)
```

3. Save the file with a suitable filename and .R extension and run as described previously. In the file path that you specified, you will now find a file structure consisting of a folder for each curve, each containing a folder for each parameter, and then in turn a number of Netlogo experiment files: one for each parameter sample. Remember eFAST works by examining one parameter of interest at a time, going through the sample space using a significantly different frequency to that at which the others are sampled. Thus, for each resample curve, one set of parameter value sets is produced for each parameter of interest. Thus, in this case, we have 3 resample curves and 4 parameters (including the 'dummy'), so 12 different sets of 65 parameter samples are produced. This explains why the analysis can be computationally expensive, as this is 780 different parameter sets on which simulations will need to be performed. Then, for each set, for stochastic simulations, a number of runs are performed to produce a robust result which considers the effect of Aleatory Uncertainty (see Technique 1). In our case, we are ok with 1 run, but in cases

where more was required, such as the example in our SPARTAN publication, this can result in hundreds of thousands of runs.

8.2 Analysing The Simulation Data

This tutorial is going to assume that you are using the example eFAST data from our lab website, but the steps are just as applicable if you ran the experiments yourself.

1. Download the Virus eFAST Experiment result set from the project website. Note for space reasons that the Netlogo parameter files themselves are not included in the download.

2. With this data available, open a text editor. Now we are going to declare the variables required to run this analysis. Type or copy in the following text. Again, the *spartan* library is imported. The variables required for this analysis are then declared in capital letters. The line underneath, beginning with a #, is a description of that being declared. Make sure you set the FILEPATH variable correctly to state where the simulation results have been extracted to.

```
library(spartan)
library(XML)
library(gplots)

# The directory where the netlogo experiment file should be stored
FILEPATH<-"/home/kieran/Documents/Tutorial/eFAST/"
# The parameters being examined in this analysis. Include the dummy
PARAMETERS<-c("infectiousness","chance-recover","duration",
"dummy")
# Values of each of the parameters above. For a range, state the min
# and max in square brackets, in double quotes
PARAMVALS<-c(150,"[10,90]","[10,90]","[5,40]","[1,10]")
# Number of resampling curves to use
NUMCURVES<-3
# Number of value samples to take from each curve
NUMSAMPLES<-65
# The output measures by which you are analysing the results.
MEASURES<-c("death-thru-sickness","death-but-immune","death-old-age",
"death-old-and-sick")
# Name of the result file generated by Netlogo. The sample number and
# .csv are added to this
EFASTSAMPLE_RESULTFILENAME<-"efast_result_set"
# Timestep of interest. The behaviour space table is likely to contain
# all timesteps. This narrows the analysis
TIMESTEP<-5200
# File created containing the median of each output measure, of each
# simulation for this parameter set. Note no file extension
RESULTFILENAME<-"ParamValResponses"
# Format of the above file (xml or csv)
RESULTFILEFORMAT<-"csv"
# File summarising the results of numerour runs for a parameter set:
# a median of medians
MEDIANSFILENAME<-"ParamValResponses"
# File format of the above file (xml or csv)
MEDIANSFILEFORMAT<-"csv"
# Summary file for each resample curve. Used to produce error stats
```

```
CURVERESULTSFILENAME<-"Virus_AllResults"
# Output measures to t-test to gain statistical significance
OUTPUTMEASURES_TO_TTEST<-1:4
# T-Test confidence interval
TTEST_CONF_INT<-0.95
# Boolean noting whether graphs should be produced
GRAPH_FLAG<-TRUE
# Name of the final result file summarising the analysis, showing the
# partitioning of the variance between parameters. Note no file
# extension
EFASTRESULTFILENAME<-"Virus_eFAST_Analysis"
# Not used in this case, but when a simulation is analysed at
# multiple timepoints (see Tutorials 1-4)
TIMEPOINTS<-NULL; TIMEPOINTSCALE<-NULL
```

3. Now to examine the first of the four methods (again we look at each individually so the functionality becomes apparent but in reality you will run all three methods one after another in the same text file). Copy the below into the text file under the above declarations:

   ```
   efast_process_netlogo_result(FILEPATH,EFASTSAMPLE_RESULTFILENAME,
   PARAMETERS,NUMCURVES,NUMSAMPLES,MEASURES,RESULTFILEFORMAT,
   RESULTFILENAME,TIMESTEP)
   ```

4. Save the script with an appropriate name and .R extension, and run as described previously. This method has transferred the Netlogo experiment result into a format that can be processed by SPARTAN, CSV files. Each parameter value set is summarised, with the results from the Netlogo run(s) stored as a CSV row. The methods used in Technique 4 can now be used to process this data.

5. Back in the text editor, add the following method declaration to the file, then run the script.

   ```
   efast_netlogo_get_overall_medians(FILEPATH,NUMCURVES,PARAMETERS,
   NUMSAMPLES,MEASURES,MEDIANSFILEFORMAT,MEDIANSFILENAME,
   CURVERESULTSFILENAME)
   ```

   This second method produces a summary of the results for a particular resampling curve. This shows, for each parameter of interest, the median of each simulation output measure for each of the 65 (or more if you altered this) parameter value sets generated.

   Here's an example. We examine resampling curve 1, and firstly examine parameter 1 (infectiousness in this case). For this parameter of interest, 65 different parameter value sets were generated from the frequency curve, thus we have 65 different sets of simulation results. The previous method produced a SPARTAN compatible output from the Netlogo result, showing the response of each output measure for each run. Now, this method calculates the median of these responses, for each output measure, and stores these in a summary file. Thus, for each parameter of interest, the medians of each of the 65 sets of results are stored. The next parameter is then examined, until all have been analysed. This produces a snapshot showing the median simulation output for all parameter value sets generated for the first resample curve. These are stored within the second level of the folder structure, with filename as specified by the R variable CURVERESULTSFILENAME, and the process repeated for the remaining curves. Once you have run the method, have a look at this file and make sure you understand the output.

6. Now that these summary files have been produced, we can use the eFAST analysis algorithm in our attempts to partition the output variance between input parameters. In the same file in the text editor, add the following method call, save the file and run the script:

```
efast_netlogo_run_Analysis(FILEPATH,MEASURES,PARAMETERS,
NUMCURVES,NUMSAMPLES,OUTPUTMEASURES_TO_TTEST,TTEST_CONF_INT,
GRAPH_FLAG,CURVERESULTSFILENAME,EFASTRESULTFILENAME,
TIMEPOINTS,TIMEPOINTSCALE)
```

Navigate to the top level of the folder structure. You will see that a file Virus_eFAST_Analysis.csv has been produced. Open this file in a spreadsheet so its structure can be examined. You will notice that the first column contains the parameters that have been explored. For each, there are then a number of statistics, for each output measure. The important values are:

(i) The Si measure. Si is the first-order sensitivity index (where i is the parameter). Calculated as the simulation variance for that parameter over the unique sampling frequency it was assigned, it represents the fraction of model output variance that is explained by an input variation of a given parameter.

(ii) The STi measure. STi is the total-order sensitivity index (again i is the parameter). This statistic again indicates the fraction of model output variance, but this time includes any higher-order, non-linear effects between the parameter of interest and its complementary parameters.

(iii) The SCi measure. SCi is the variance caused by the parameter of interests complementary set (or, all other parameters bar that of interest).

(iv) Error values. As the analysis has been done for a number of sample curves, it is possible to calculate other statistical measures such as the standard error. This is useful for plotting the results graphically.

For ease of representation, the method also produces a graph showing this data for each measure. These will have been saved in the top level directory.

Have a look at the graphs. You will see that the parameters of interest, and the dummy parameter, are on the x axis. A parameters Si and STi value are deemed significant when a comparison is drawn with the Si and STi of the dummy parameter. In other statistical tests a comparison is made with zero, but that is not possible for eFAST (see Marino paper). So instead the dummy is used. This has no influence on simulation results, but will still be assigned an Si and STi value by the algorithm. Any parameters of interest which have an STi that is less than or equal to the dummy parameter is considered not significantly different from zero. Comparisons between each parameter of interest and that of the dummy parameter are made using a two-sample t-test. This leads to the calculation of a p-value for the sensitivity indexes, which can also be seen in the summary spreadsheet.

9 Further Reading

The following two references may be useful in understanding this technique in more detail:

- Alden, K., Read, M., Andrews, P.S., Timmis, J., Veiga-Fernandes, H., Coles, M. (2013) Spartan: A Comprehensive Tool for Understanding Uncertainty in Simulations of Biological Systems. PLoS Computational Biology, 9 (2).
- Read, M., Andrews, P.S., Timmis, J. & Kumar, V. (2012) Techniques for Grounding Agent-Based Simulations in the Real Domain : a case study in Experimental Autoimmune Encephalomyelitis. Mathematical and Computer Modelling of Dynamical Systems, 18(1):67-86.

- Marino, S., Hogue, I.B., Ray, C.J. & Kirschner, D.E. (2008) A methodology for performing global uncertainty and sensitivity analysis in systems biology. Journal of theoretical biology, 254 (1), p.pp.178-96.
- Saltelli, A., Chan, K. & Scott, E.M. (2000) Sensitivity Analysis, Wiley series in probability and statistics Wiley.
- The *spartan* Manual, spartan-Manual.pdf, within the spartan package describes in more detail each method within the package
- Vargha, A. & Delaney, H.D. (2000) A critique and improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. Journal of Educational and Behavioural Statistics, 25, p.pp.101-132.
- The spartan Manual, spartan-Manual.pdf, within the spartan package describes in more detail each method within the package