# Applying spartan to Understand Parameter Uncertainty in Simulations

*by Kieran Alden, Mark Read, Paul S Andrews, Jon Timmis and Mark Coles*

**Abstract** In attempts to further understand the dynamics of complex systems, the application of computer simulation is becoming increasingly prevalent. Whereas a great deal of focus has been placed in the development of software tools that aid researchers develop simulations, similar focus has not been applied in the creation of tools that perform a rigorous statistical analysis of results generated through simulation: vital in understanding how these results offer an insight into the captured system. This encouraged us to develop **spartan**, a package of statistical techniques designed to assist researchers in understanding the relationship between their simulation and the real system. Previously we have described each technique within **spartan** in detail, with an accompanying immunology case study examining the development of lymphoid tissue. Here we provide a practical introduction to the package, demonstrating how each technique is run in R, to assist researchers in integrating this package alongside their chosen simulation platform.

## Introduction

Utilising mathematical or agent-based computational models to further understand the dynamics of complex systems is becoming increasingly popular. With the benefits of performing simulated experiments where the conditions are fully controlled, or where ethical and financial constraints are avoided, researchers can generate a predictive tool that has the capacity to inform future studies of the captured system. To aid the adoption of this approach, much focus has been placed on providing researchers with software tools through which they can generate a simulation (Ivanyi et al., 2007; Wilensky, 1999; Luke, 2005; Meier-Schellersheim et al., 2006). Although these simulation development platforms provide easy to use functionality for generating a large set of simulation results, these platforms do not equip the researcher with the means of statistically analysing these results. However, undertaking a full statistical analysis to gain a full appreciation of the way the simulated biological system behaves is vital.

Our recent work utilised agent-based modelling to further understand the development of secondary lymphoid organs in the gastrointestinal tract. Thorough statistical analyses of simulation behaviour demonstrated that the simulator produced emergent cell behaviour that is statistically similar to that observed in the laboratory (Patel et al., 2012; Alden et al., 2012). As the analyses suggested this is the case, we were able to utilise our tool to perform *in silico* experimentation to inform future laboratory based studies. In order to do this, fully understanding the relationship between the simulation and the real-world system that has been captured is vital: can a result attributed to the real-world system, or is it an artefact of simulation implementation or uncertainty in parameter values? Such understanding is key where the aim is to transfer an hypothesis generated through simulation to the real-world. This led us to develop **spartan** (Simulation Parameter Analysis R Toolkit Application) (Alden et al., 2013b), a package of statistical techniques that aid the researcher in understanding this relationship such that a simulation can provide novel insight into the system being studied. We do note that in addition to **spartan**, several other packages for sensitivity analysis are also available (Lamboni et al., 2013; Dancik, 2013; Pujol et al., 2014). However **spartan** has been compiled to equip developers of biological simulations with the statistical techniques necessary to maximise the potential of the simulation platform they generate. This package can be applied to agent-based simulations such as ours, and to traditional ordinary or partial differential equation models. To encourage adoption of these techniques by the community, **spartan** is supported by detailed tutorials and a growing number of case studies: the first of which is demonstrated in this paper. Although the development of **spartan** has resulted from a number of such biological modelling studies, we do not rule out the potential application of the package outside of the biological sciences domain.

**spartan** has been implemented in R, released under a GPLv2 license, and is available from CRAN. The package is a compilation of four previously described statistical techniques (Read et al., 2012; Marino et al., 2008; Saltelli and Bollardo, 1998) that each provide an alternative means of analysing simulation data. The first technique helps a researcher ensure that a response generated by a non-determinstic simulation is a true representation of the parameter set upon which it is being run. The second is a local sensitivity analysis technique that suggests how robust simulation behaviour is to parameter perturbation, with techniques 3 and 4 being global sensitivity analysis techniques that help understand the key pathways affecting simulation behaviour. Our complementary **spartan** paper (Alden et al., 2013b) describes each technique in detail, and utilises our case study of lymphoid tissue development to demonstrate to the researcher the impact that such statistical techniques could

have in understanding their simulation behaviour. Here we take this description a stage further, and demonstrate the *application* of each technique in R. Alongside full examples of the R code and commands required to run each technique, we have provided the reader with example simulation data, available from our laboratory website (www.ycil.org.uk). As such, our description below is fully reproducible, and encourages simulation developers to adopt **spartan** alongside their chosen simulation development tool. For space reasons, we focus on the application of rather than the full implementation detail of each technique, and as such this paper should be seen as an accompaniment to our previous **spartan** publication.

## Prerequisites

The following are required to run **spartan** as we describe in this paper:

- The R statistical environment, version 2.13.1 or later.
- The **spartan** (Alden et al., 2013a) R package, downloaded from CRAN
- The **lhs** (Carnell, 2012), **gplots** (Warnes et al., 2013), and **XML** (Lang, 2013) R packages, also available for download from CRAN.
- The example simulation results, available from our laboratory website (www.ycil.org.uk)

## The case study

Our demonstration in this paper utilises data from our previously described agent-based lymphoid tissue development simulator (Patel et al., 2012; Alden et al., 2012). In this system two types of cells migrate into the developing gut and form patches of tissue at varying locations along the gut tract 72 hours later. These mature to form lymphoid organs that trigger an adaptive immune response to gut-based pathogens. Lab experimentation has revealed that cells behave in a different manner (in terms of their velocity and displacement over a 1 hour period) around an initial forming patch than elsewhere in the gut, for reasons not currently understood. To aid our understanding of tissue development, we adopted an agent-based simulation approach. In the example analyses demonstrated here, we are interested in two emergent cell behaviour responses: velocity and displacement, and how these are affected by the values assigned to simulation parameters that capture certain aspects of the biological system. These are captured at simulated hour twelve of tissue development: a time-point where we have experimental data which can be compared to simulation response. The simulator has six parameters for which there is uncertainty in parameter value, each previously described in depth (Alden et al., 2012). Here we demonstrate techniques that perturb the values of each of these parameters to reveal the role each has on the emergent cell behaviour responses.

## Technique 1: Consistency analysis

Aleatory uncertainty is caused by inherent stochasticity within a non-deterministic simulation, and it is critical that the impact this has on simulation response is understood (Helton, 2008). This is especially the case for agent-based simulations such as our case study, where pseudo-random number generation can lead to the simulation producing different responses for identical parameter value sets. To mitigate this, a number of replicate simulation runs should be performed, achieving a representative result for a given set of parameters. **spartan** Technique 1, developed from a method described by Read et al. (2012) can be applied to stochastic simulation systems, to provide an indication of the number of simulation runs necessary to reduce this uncertainty, while allowing researchers to balance the computational resource required against accuracy of results. This technique is not applicable to deterministic simulations where identical responses are generated each time the simulation is run with a specified set of parameters.

For the full detail of the algorithm we direct the reader to our previously published work (Read et al., 2012; Alden et al., 2013b). As an overview, the Consistency Analysis Technique in **spartan** operates by comparing a number of distributions of simulation responses run under identical parameter values. Through altering the number of replicate simulation responses within each distribution, the number of runs required to ensure statistical consistency of response can be determined. In the example given in Read et al. (2012), 20 such distributions are used. Each contains a number of simulation responses. Distributions 2-20 are contrasted with distribution 1 using the Vargha-Delaney A-Test (Vargha and Delaney, 2000), a non-parametric effect magnitude test which provides a statistical measure of the difference between two distributions, suggesting that the results are consistent or the distribution requires a larger number of simulation runs. As would be assumed, increasing the number

of replicates decreases the variance between the distributions. **spartan** can be used to determine this number, while ensuring that an excess number of runs is avoided.

We can provide more detail with the aid of an exemplar application, utilising the example data available from our laboratory website (www.ycil.org.uk). The first step is to define the objects required for this analysis, as below. The comment above each object notes the reason for the declaration:

```
# Firstly, import the package
library(spartan)
# Directory where the example simulation results for this technique were extracted
FILEPATH <- "/home/user/AA"
# Sample sizes (number of simulation replicates in each distribution) to be analysed
SAMPLESIZES <- c(1, 5, 50, 100, 300)
# The simulation output measures to be analysed
MEASURES <- c("Velocity", "Displacement")
# Number of distributions being compared. Default: 20, as performed by Read et al
NUMSUBSETSPERSAMPLESIZE <- 20
# Output file name containing the simulation responses.
RESULTFILENAME <- "trackedCells_Close.csv"
# Not used in this case. Useful where two result files exist (e.g.\ if tracking cells
# close and those further away, two output files could be used). Here, results in a
# second file are processed if the first is blank or does not exist.
ALTFILENAME <- NULL
# Notes the column in the CSV results  file where the results start.
# Useful as it restricts what is read in to R, getting round potential errors where
# the first column contains a label
OUTPUTFILECOLSTART <- 10
# Last column of the output measure results
OUTPUTFILECOLEND <- 11
# Use this if simulation results are in CSV format.
# Last column of the output measure results
OUTPUTFILECOLEND <- 11
# File either A: created by method 1 of this technique, containing the median of each
# output measure of each simulation run in that subset, or B: The name of the provided
# single CSV file containing the simulation responses. So if you are using the CSV
# structured tutorial data, this fill be the name of that CSV file.
MEDIANS_SUMMARY_FILE_NAME <- "AA_SimResponses.csv"
# The results of the A-Test comparisons of the twenty subsets for each sample size
# are stored within an output file. This parameter sets the name of this file.
# Note no file extension. Current versions of spartan output to CSV files
ATESTRESULTSFILENAME <- "AA_ATest_Scores.csv"
# A summary file is created containing the maximum and median
# A-Test values for each sample size. This parameter sets the name of this file.
SUMMARYFILENAME <- "AA_ATestMaxAndMedians.csv"
# The A-Test value either side of 0.5 which should be considered a 'large difference'
# between two sets of results. Use of 0.23 was taken from the Vargha-Delaney
# publication but can be adjusted here as necessary.
LARGEDIFFINDICATOR <- 0.23
# A-Test values above 0.5 (no difference) which should be considered as small,
# medium, and large differences between two result sets. Used in the graph
# summarising all sample sizes.
SMALL <- 0.56
MEDIUM <- 0.66
LARGE <- 0.73
# Name of the graph which summarises the analysis results for all sample sizes.
# Current versions of spartan output to pdf.
GRAPHOUTPUTFILE <- "AA_ATestMaxes.pdf"
# Timepoints being analysed. Must be NULL if no timepoints being analysed, or else
# be an array of timepoints. Scale sets the measure of these timepoints
TIMEPOINTS <- NULL; TIMEPOINTSCALE <- NULL
# Example Timepoints, if being used:
#TIMEPOINTS <- c(12, 36, 48, 60); TIMEPOINTSCALE <- "Hours"
```

In this example, we are determining whether 1,5,50,100 and 300 replicates (stated in `SAMPLESIZES`)

of a simulation run are sufficient to produce a result that is representative of the parameter set upon which the simulation was run. Similar to the original description of this technique in Read et al. (2012), we utilise 20 distributions for each sample size (i.e. 20 distributions each containing 1 run, 20 more distributions each containing 5 runs, and so on up to 300). **spartan** (from Version 2.0) can process simulation output provided in two formats, both of which we describe next.

The first format consists of the folder structure seen in Figure 1(A), where the result of each simulation run is provided in an organised structure. This structure is comprised of three levels: the sample size, a folder for each distribution for that size, and a folder for the results for each run in that distribution. Such a structure is suitable in cases where the simulation results have been generated on a computer cluster, meaning the results need no post-processing once stored in the correct location. The second format consists of a sole CSV file that summarises the results of each run for each distribution, for each sample size. Each row of the CSV file should contain the sample size being analysed when this simulation was run, the distribution the run belongs to, and the median of each simulation response for that run. There are examples of both formats within the exemplar data that is available from our website (www.ycil.org.uk).

In our case study, each simulation result contains behavioural responses (velocity and displacement) for a number of cells. To compare the distributions using the A-Test, we need to calculate the median of each response, for each run, to produce a result set that summarises the runs in each distribution. To do this, we use the `aa_summariseReplicateRuns` function, which we describe in the next paragraph. Note however that this function should only be run when results are being provided in the first input format: the organised folder structure. The function iterates through these folders to summarise the simulation runs in a single CSV file, matching the format that can be provided by the second input format (the single CSV file).

For example, consider distribution 1 for a sample size of 50 runs. Each run contains the results for a set of cells that were tracked over the course of one simulated hour. For each run, **spartan** calculates the median of each simulation response (cell velocity and displacement in our case), storing these as a row in a CSV file that summarises cell behaviour across all the runs, named as stated in `MEDIANS_SUMMARY_FILE_NAME`. The appropriate set of medians is then generated for distributions 2-20, with these calculates repeated for all sample sizes being analysed.

```
aa_summariseReplicateRuns(FILEPATH, SAMPLESIZES, MEASURES, RESULTFILENAME,
        ALTFILENAME, OUTPUTFILECOLSTART, OUTPUTFILECOLEND, MEDIANS_SUMMARY_FILE_NAME,
        TIMEPOINTS, TIMEPOINTSCALE)
```

Whether providing results in the organised folder structure or as a single CSV file, the researcher will now have their results in a format that can be processed by the methods contained in **spartan**. Where providing a single CSV file as input of simulation results, the name of this file should be stated in parameter `MEDIANS_SUMMARY_FILE_NAME`.

Statistical difference between the 20 distributions of each sample size is calculated by contrasting the responses of distributions 2-20 in turn with distribution 1, using the Vargha-Delaney A-Test, using the function below. To ease understanding of the results, a graph is automatically produced for each sample size (example in Figure 1(B)), plotting the A-Test result for the nineteen comparisons, and automatically saved in the directory specified by `FILEPATH`.

```
aa_getATestResults(FILEPATH, SAMPLESIZES, NUMSUBSETSPERSAMPLESIZE, MEASURES,
        MEDIANS_SUMMARY_FILE_NAME, ATESTRESULTSFILENAME, LARGEDIFFINDICATOR,
        TIMEPOINTS, TIMEPOINTSCALE)
```

The A-Test returns the probability that a randomly selected sample from one distribution is larger than a randomly selected sample from another. Hence, a result of 0.5 means their medians are the same (even if their variance is not). Only the magnitude of the result away from 0.5 is important, the direction depends simply on which distribution has higher values. Larger differences between score and 0.5 indicate bigger differences between two distributions.

However, what is of real interest is the difference between distributions against number of replicate runs that is contained in each. These results are summarised using the following two functions:

```
aa_sampleSizeSummary(FILEPATH, SAMPLESIZES, MEASURES, ATESTRESULTSFILENAME,
        SUMMARYFILENAME, TIMEPOINTS, TIMEPOINTSCALE)

aa_graphSampleSizeSummary(FILEPATH, MEASURES, 300, SMALL, MEDIUM, LARGE,
        SUMMARYFILENAME, GRAPHOUTPUTFILE, TIMEPOINTS, TIMEPOINTSCALE)
```

The first creates a CSV file, in the folder specified in the `FILEPATH` variable, stating the maximum A-Test score across the distributions for each simulation output response, for each number of simulation

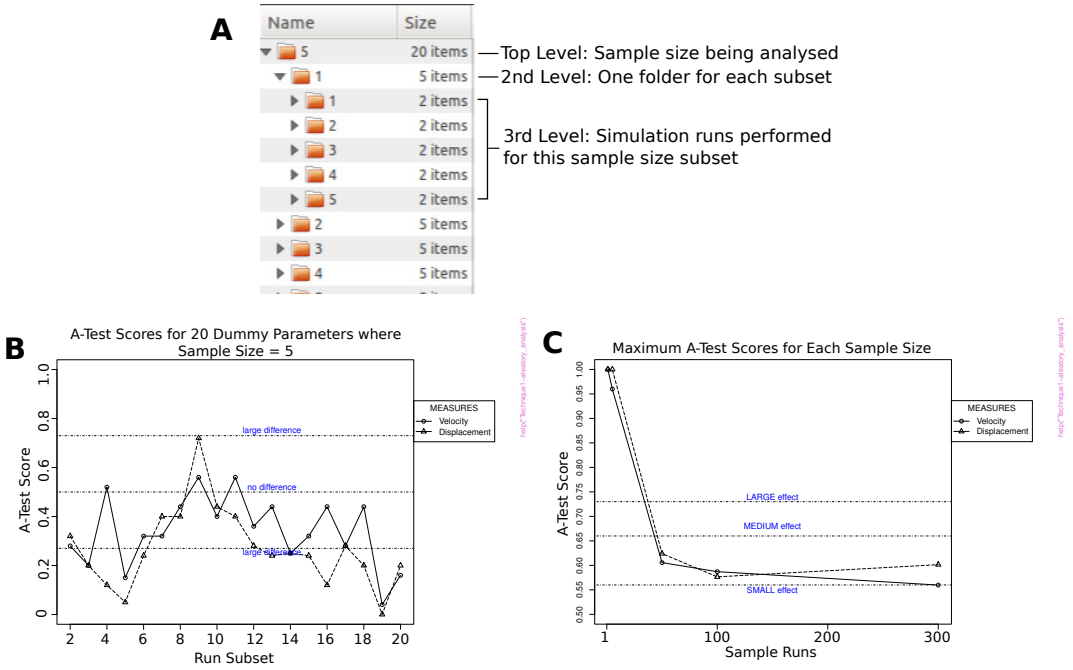


**Figure 1:** Use of **spartan** to mitigate aleatory uncertainty. A: The folder structure that should be utilised if providing all simulation result files rather than a CSV file summarising all results. B: A-Test scores for 20 distributions of simulation runs, with each distribution containing 5 runs. Note the substantial differences between distributions, although the same parameter values have generated these results. This suggests that 5 runs is insufficient to achieve a representative result. C: The maximum A-Test score across 20 distributions of simulation runs for sample sizes of 1,5,50,100,and 300. As direction is of no concern, the A-Test scores are normalised such that scores below 0.5 are assigned corresponding values above 0.5. B and C are reproduced from Alden et al. (2013b)

replicates examined. The second produces a plot of this result (Figure 1(C)), saved in the same folder. The objective is to find a sample size which minimises the A-Test value for all measures. As argued previously (Alden et al., 2013b), based on these results we did not deem 300 runs to be sufficiently close enough to the 'no difference' line for both Velocity and Displacement measures, and as such ran all our analyses with 500 runs (not included in data set due to download size). With this result taken into account, we are confident that the hypotheses we generate from simulation-based experimentation are derived from parameter values, and not inherent stochasticity caused by simulation implementation. This is an important consideration for agent-based models such as ours, and understanding the origin of changes in simulation behaviour is vital when interpreting results.

You will have noticed that each function call contains the two parameters TIMEPOINTS and TIMEPOINTSCALE. Each function is capable of performing this analysis for multiple timepoints of a simulation, and will iterate through each as required. In addition to the steps above, such an analysis requires that:

- An array of timepoints being examined is provided in the `TIMEPOINTS` object, and the scale in which these are measured provided in `TIMEPOINTSCALE` (see the object declarations above for an example).
- Simulation results (if provided in the folder structure input method), or the CSV files summarising simulation results, should have the timepoint at which these were produced appended to the filename. For example, in the exemplar data, there are CSV files named AA_SimResponses_12.csv, AA_SimReponses_36.csv, etc, containing cell behaviour characteristics at hours 12 and 36 respectively. The same exists in the simulation result folders, where the simulation results are named trackedCells_Close_12.csv, trackedCells_Close_36.csv, etc. It is important that the result files are named in this manner to aid spartan finding these result files.

## Technique 2: Parameter robustness

We noted previously that for our case study, there are six parameters for which there is uncertainty in parameter value. In our case, this uncertainty applies as no biological experimental technique has

or can be used to determine a value. In other cases, such uncertainty may be derived from a lack of available data, or the generation of a hypothetical model. The Parameter Robustness technique included in **spartan** can be utilised to examine the implication this uncertainty or parameter estimation has on simulation response. If altering the value of a particular parameter from a baseline or calibrated value has a significant effect on simulator output, the simulation is highly sensitive to that parameter, and caution should be applied when both interpreting the result and establishing a value for that parameter.

Robustness analysis is performed by perturbing each parameter individually, using a 'one at a time' approach (Read et al., 2012). The value of the parameter of interest is perturbed, with all other parameters remaining at their baseline value. Simulation responses under these perturbed conditions are contrasted with responses under baseline conditions, using the Vargha-Delaney A-Test (Vargha and Delaney, 2000) described above, to determine if a scientifically significant behavioural alteration has occurred. This provides the researcher with an indication of how robust the simulation response is to parameter alteration and indicates parameter values at which simulation behaviour changes. In cases where expert knowledge is available, such parameter value windows can be contrasted to an accepted range of values, to determine if the simulation is behaving as expected. **spartan** includes methods to produce simulation parameter value sets for this statistical technique and to analyse the resultant simulation response under those conditions.

### Parameter sampling

The package contains a method which can produce a set of simulation parameters for each parameter of interest. Simulations should then be run on each of the generated parameter sets. These are generated as follows:

```
# Import the package
library(spartan)
# Set a folder where the parameter value samples should be output to
FILEPATH <- "/home/user/OAT/Sampling"
# Set the names of the parameters for which values are being generated for
PARAMETERS <- c("thresholdBindProbability", "chemoThreshold",
"chemoUpperLinearAdjust", "chemoLowerLinearAdjust",
"maxVCAMeffectProbabilityCutoff", "vcamSlope")
# The calibrated values, or baseline values, of each stated parameter
BASELINE <- c(50, 0.3, 0.2, 0.04, 0.60, 1.0)
# Parameter Value Information
# You can specify this in two ways:
# 1. The minimum and maximum of each parameter, and increment over which
# sampling should be increased.
# 2. A string list of values that parameter should be assigned in sampling
# Example of 1:
PMIN <- c(0, 0.10, 0.10, 0.015, 0.1, 0.25)
PMAX <- c(100, 0.9, 0.50, 0.08, 1.0, 5.0)
PINC <- c(10, 0.1, 0.05, 0.005, 0.05, 0.25)
PARAMVALS <- NULL
# Example of 2:
#PARAMVALS <- c("0, 50, 90","0.10, 0.3, 0.8","0.10, 0.25, 0.4",
        "0.015, 0.04, 0.08", "0.1, 0.5, 0.9","0.25, 1.25, 2.0, 3.0, 5.0")
# If using method 1, PARAMVALS must be set to NULL. If using method 2, PMIN,
# PMAX, and PINC must be set to NULL

oat_parameter_sampling(FILEPATH, PARAMETERS, BASELINE, PMIN, PMAX,
        PINC, PARAMVALS)
```

This will produce a CSV for for each parameter, each with a file name matching the parameter which is being perturbed. In our case, 6 files are generated, and stored in the folder specified in `FILEPATH`. Simulations should then be run on each parameter value set in each file, and analysed using the technique described in the next section.

### Analysing the results

In this section we will demonstrate how **spartan** can produce statistical information that reveals how robust the simulation behaviour is to a change in parameter value. Again we provide more detail

with the aid of an exemplar application, utilising the example data available from our laboratory website (www.ycil.org.uk). Note that to reduce the size of the download data, results for only two of the six parameters have been included. A full analysis of the six parameters has been described previously (Alden et al., 2012). The first step in this demonstration is to define the objects required for this analysis, as below. For space reasons, we have not duplicated the comments for objects that were declared in Technique 1 or in parameter sampling above: the reader should refer to these if required.

```
library(spartan)
# Folder containing the example simulation results. Make sure the folder is unzipped
FILEPATH <- "/home/user/OAT/Results"
# Array of the parameters to be analysed.
# Note only two of the six here for download size reasons
PARAMETERS <- c("chemoLowerLinearAdjust", "chemoUpperLinearAdjust")
# Similar to the sampling function discussed above, there are two ways to specify
# parameter value information in the analysis. Ensure you are using the appropriate
# method, setting these to NULL if using the alternative (see comments in sampling
# function description).
# Method 1:
PMIN <- c(0.015, 0.10)
PMAX <- c(0.08, 0.50)
PINC <- c(0.005, 0.05)
PARAMVALS<-NULL
# Method 2:
#PARAMVALS <- c("0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05, 0.055, 0.06,
#        0.065, 0.07,0.075, 0.08", "0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5")
#PMIN <- NULL; PMAX <- NULL; PINC <- NULL
BASELINE <- c(0.04, 0.2)
MEASURES <- c("Velocity", "Displacement")
# What each measure represents. Used in graphing results
MEASURE_SCALE <- c("microns/min", "microns")
RESULTFILENAME <- "trackedCells_Close.csv"
OUTPUTCOLSTART <- 10
OUTPUTCOLEND <- 11
ALTERNATIVEFILENAME <- NULL
# Either 1: The name of the CSV file containing all simulation output (see description
# that follows in this section) or name to give the summary file that spartan generates
CSV_FILE_NAME <- "OAT_Medians.csv"
# Number of replicate runs performed for each parameter value set
NUMRUNSPERSAMPLE <- 300
# 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.
# Current versions of spartan output this to a CSV file
ATESTRESULTSFILENAME <- "EgSet_ATests.csv"
# A-Test result value either side of 0.5 at which the difference between two sets of
# results is significant
ATESTSIGLEVEL <- 0.23
# Timepoints being analysed. Must be NULL if no timepoints being analysed, or else
# be an array of timepoints. Scale sets the measure of these timepoints
TIMEPOINTS <- NULL; TIMEPOINTSCALE <- NULL
# Example Timepoints, if being used:
#TIMEPOINTS <- c(12, 36, 48, 60); TIMEPOINTSCALE <- "Hours"
```

In this example, we are analysing the robustness of two parameters, perturbed between 0.015-0.08 and 0.10-0.50 accordingly. Similar to Technique 1, the researcher can provide their simulation results in two formats: the organised folder structure detailed in Figure 2(A) or a single CSV file. The folder structure is comprised of three levels: the parameter being analysed, the value assigned to that parameter, and results for each run under those parameter conditions. If providing a summary of results in a single CSV file, each row should contain the value of each parameter being analysed (matching the parameters stated in `PARAMETERS`) and the median of each simulation response for simulation runs under those conditions. For stochastic simulations where a number of runs are required to produce a consistent result (see Technique 1), a number of results for each parameter set should be provided in the CSV file. For example, in our case study we have repeated each simulation 300 times, thus 300 lines exist in the CSV file for each parameter value set. Examples of both formats of input have been provided in the exemplar data available from our website (www.ycil.org.uk).

The first function that comprises this technique should only be run where the researcher is providing results in the organised folder structure in Figure 2(A). Similarly to Technique 1, this method iterates through the simulation results in each folder, producing the CSV file that summarises all simulation results. For our case study, this function calculates the median of each cell behaviour response for each of the 300 runs, storing this in the CSV file. Once each parameter-value pair has been processed, this CSV file is saved within the folder specified in `FILEPATH`, with the filename as stated by `CSV_FILE_NAME`. PMIN, PMAX, PINC, and PARAMVALS should be set appropriately as described in the comments of the object specifications above.

```
oat_processParamSubsets(FILEPATH, PARAMETERS, NUMRUNSPERSAMPLE, MEASURES,
        RESULTFILENAME, ALTERNATIVEFILENAME, OUTPUTCOLSTART, OUTPUTCOLEND,
        CSV_FILE_NAME, BASELINE, PMIN, PMAX, PINC, PARAMVALS,
        TIMEPOINTS, TIMEPOINTSCALE)
```

Whether providing results in the organised folder structure or as a single CSV file, the researcher will now have their results in a format that can be processed by the methods contained in **spartan**. Where providing a single CSV file as input of simulation results, the name of this file should be stated in parameter `CSV_FILE_NAME`.

To gain statistical information concerning the impact that a change in a single parameter has on simulation response, we call the following function.

The results of each parameter/response pair in the CSV file are processed, comparing the distribution of responses for simulations under one parameter value set conditions with responses at the calibrated/baseline value. The Vargha-Delaney A-Test described previously is utilised to perform this comparison of distributions. As an example, consider one of the parameters we are analysing in the example set, chemoLowerLinearAdjust. This has a baseline value of 0.04. The simulation has two output measures, velocity and displacement. The method will read all results from the CSV file where the parameter value is its calibrated value, and contrast these with results where this parameter value has been perturbed, determining the impact this value change has had on simulation response. With all parameter values examined, **spartan** produces a CSV file stating the A-Test values for each value assigned to that parameter (named as stated in object `ATESTRESULTSFILENAME`.

```
oat_csv_result_file_analysis(FILEPATH, CSV_FILE_NAME, PARAMETERS, BASELINE,
        MEASURES, ATESTRESULTFILENAME, PMIN, PMAX, PINC,
        PARAMVALS, TIMEPOINTS, TIMEPOINTSCALE)
```

To ease assessment of these results, **spartan** can then produce a plot for each parameter, showing the difference in simulation behaviour across the parameter value space. An example of such a plot can be seen in Figure 2(B).

```
oat_graphATestsForSampleSize(FILEPATH, PARAMETERS, MEASURES, ATESTSIGLEVEL,
        ATESTRESULTSFILENAME, BASELINE, PMIN, PMAX, PINC, PARAMVALS, TIMEPOINTS,
        TIMEPOINTSCALE)
```

Finally, it may be interesting, especially for stochastic simulations, to see the distribution of simulation responses for each parameter value. A boxplot of these results can be produced for each parameter using the method below. Note that as this is produced directly from simulation responses, this plot can currently only be produced for simulation input provided in the first method, the organised folder structure.

```
oat_plotResultDistribution(FILEPATH, PARAMETERS, MEASURES, MEASURE_SCALE,
        CSV_FILE_NAME, BASELINE, PMIN, PMAX, PINC, PARAMVALS, TIMEPOINTS,
        TIMEPOINTSCALE)
```

Similarly to Technique 1, it is possible to perform this technique for multiple simulation timepoints. To do this, the researcher needs to follow the same requirements detailed at the end of the description of Technique 1.

Running this analysis in R with the example data should produce the two A-Test graphs seen in Figures 2(B) and 2(C). Both are interesting for different reasons. In Figure 2(B), responses for both simulation measures greatly vary when the value of that parameter is perturbed, suggesting that assigning an appropriate value is critical. This change in behaviour can be compared with existing data, if applicable, or expert opinion, to determine if the parameter is having the desired effect. Yet in Figure 2(C), it appears that the simulation response does not significantly change for all parameter values that were explored, thus the simulation is robust to a change in parameter value. There is a large uncertainty in the value that should be assigned to this parameter. Reading of this result is dependent

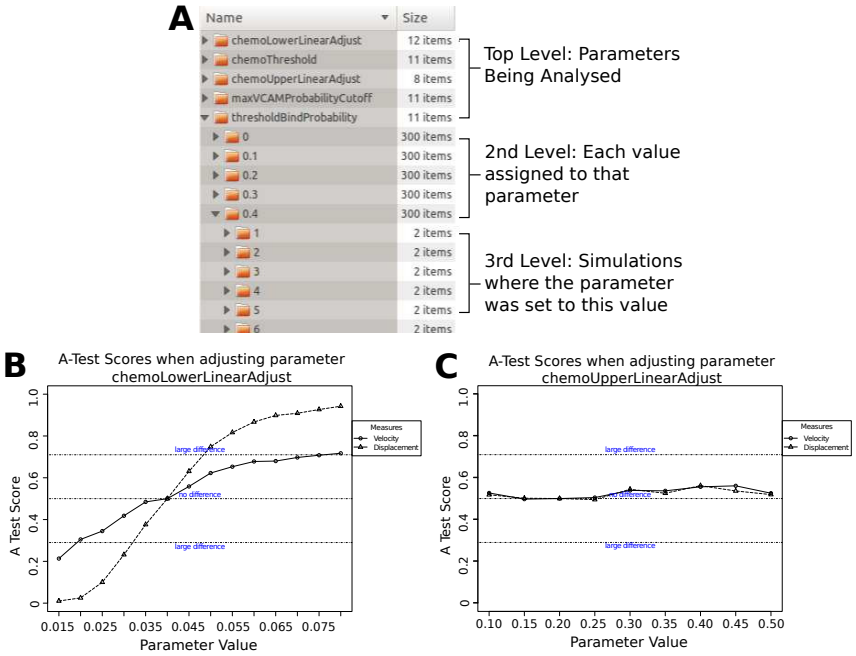


**Figure 2:** Use of **spartan** to determine how robust a simulation is to parameter perturbation. A: The folder structure that should be utilised if providing all simulation result files rather than a CSV file summarising all results. B: A-Test scores generated when the parameter controlling initial expression of a chemoattractant is perturbed. C: A-Test scores generated when the parameter controlling saturation of a chemoattractant is perturbed. B and C are reproduced from Alden et al. (2013b)

on the context of the work being performed, as this may suggest that assigning the parameter a correct value is not critical for simulation response, yet could also suggest that efforts should be made to obtain a value for this parameter using other means (i.e. lab experimentation).

This technique provides researchers with the ability to understand if the parameter has captured the required behaviour correctly, and how that factor alone impacts simulation response. As noted above the generated results can be interpreted a number of ways, depending on the context. If expert opinion, or biological data in our case, concerning the range of possible parameter values is available, it can be contrasted with analysis results to infer how well the simulation captures the biology. If this is not available, the analysis results can reveal the implications of not knowing the data: there is little concern if the parameter has little effect, otherwise efforts should be focused on careful calibration, or it can point to the need to further investigate the real world system.

## Technique 3: Latin-hypercube analysis

Although parameter robustness is useful for establishing robustness to alteration in the value of a single parameter, the technique can not reveal any second-order effects that occur when two or more parameters are perturbed simultaneously. The effect one parameter has on simulation response may be highly dependent on the value of another. A global sensitivity analysis is required to examine this. In compiling **spartan** we have provided two previously described global sensitivity analysis techniques. This section describes the first of these, generating simulation parameter value sets using latin-hypercube sampling (Read et al., 2012; Marino et al., 2008; Saltelli et al., 2000). Through simultaneously perturbing all parameters, those that highly influence simulation behaviour can be revealed, in turn suggesting the pathways or mechanisms that have greatest impact on simulation response. This has the potential to provide a unique insight into the system being studied.

For each parameter of interest, a parameter value space is assigned. Simulation values for each parameter are then selected using a latin-hypercube, a way of sampling a large volume with minimal number of parameter value samples while also reducing any possible correlations across each generated parameter value set. Simulation runs are performed for all the generated points in the parameter space, and correlations between parameter values and corresponding simulation responses are calculated. Large correlations indicate influential parameters. Although each parameter is investigated in turn, the analysis is global as all parameter values are changing: the effect of influential parameters can transcend any parameter specific noise **spartan** includes methods to produce simulation parameter value sets for this statistical technique and to analyse the resultant simulation response under those

conditions.

### Parameter sampling

**spartan** utilises the **lhs** R package to assist in the generation of simulation parameters using latin-hypercube sampling. This package can produce a number of sets of values, for a given number of parameters, either using a 'normal' algorithm that is fairly fast, or an 'optimal' sample that ensures the parameter space is fully explored. Although the latter offers that potential benefit, it can take a while to generate the parameter value sets. The researcher should choose a number of simulation parameter sets that should be generated from the hypercube. In our case study, we adopt the approach described in Read et al. (2012), where the parameter space is sampled 500 times to ensure adequate coverage across the range of values for each parameter. Each set of the 500 samples contains a value for the six parameters of interest. The complete set of hypercube samples is output as a CSV file. Simulations should now be performed under the conditions specified by each parameter set, each repeated for an appropriate number of runs (determined using Technique 1) where the simulation is not deterministic. In our case study, each of the 500 samples was run 300 times, to ensure the impact that inherent aleatory uncertainty has on simulation response is mitigated.

```
# Import the packages
library(spartan)
library(lhs)
# The folder where the parameter samples should be output to
FILEPATH <- "/home/user/LHC/Sampling"
# Names of the parameters to generate values for.
PARAMETERS <- c("thresholdBindProbability", "chemoThreshold", "chemoUpperLinearAdjust",
        "chemoLowerLinearAdjust", "maxVCAMeffectProbabilityCutoff", "vcamSlope")
# The number of parameter sample sets to create using the hypercube
NUMSAMPLES <- 500
# The minimum value in the range for each parameter
PMIN <- c(0, 0.10, 0.10, 0.015, 0.1, 0.25)
# The maximum value in the range for each parameter
PMAX <- c(100, 0.9, 0.50, 0.08, 1.0, 5.0)
# Algorithm to use to generate the hypercube. Can be normal (quick) or optimal,
# which can take a long time (especially for high number of parameters)
ALGORITHM <- "normal"

lhc_generate_lhc_sample(FILEPATH, PARAMETERS, NUMSAMPLES, PMIN, PMAX, ALGORITHM)
```

### Analysing the results

In this section we will demonstrate how **spartan** can produce statistical information that identifies any non-linear effects between parameters. Again we utilise our exemplar application and the example data available from our laboratory website (www.ycil.org.uk). Note that to reduce the size of the download data, the original simulation responses have not been included in the data: rather the summaries of the number of runs performed under each condition are provided. However, we do describe how these summaries are constructed below. The first step in this demonstration is to define the objects required for this analysis, as below. Again to save space, we have not duplicated the comments for objects that were declared in previous examples: the reader should refer to these if required.

```
library(spartan)
# Folder containing the example simulation results. Make sure the folder is unzipped
FILEPATH <- "/home/user/LHC/LHC_Results"
PARAMETERS <- c("thresholdBindProbability", "chemoThreshold", "chemoUpperLinearAdjust",
        "chemoLowerLinearAdjust", "maxVCAMeffectProbabilityCutoff", "vcamSlope")
MEASURES <- c("Velocity", "Displacement")
MEASURE_SCALE <- c("microns/min", "microns")
# Number of parameter value sets created in latin-hypercube sampling
NUMSAMPLES <- 500
# Number of simulation runs performed for each parameter value set
NUMRUNSPERSAMPLE <- 300
RESULTSFILENAME <- "trackedCells_Close.csv"
ALTERNATIVEFILENAME <- NULL
```

```
OUTPUTCOLSTART <- 10
OUTPUTCOLEND <- 11
# This is either 1: The name of the single CSV file that summarises all simulation runs
# of all parameter sets generated by the hypercube (using input format 2), or the name
# to assign this file when spartan produces it from simulation results (using input
# format 1).
LHC_ALL_SIM_RESULTS_FILE <- "LHC_AllResults.csv"
# Location of a file containing the parameter value sets generated by the hypercube
# sampling (i.e. the file generated in the previous function of this paper). However
# if providing a CSV file with all results, you do not need to provide this
LHC_PARAM_CSV_LOCATION <- "Tutorial_Parameters_for_Runs.csv"
# spartan produces a summary file showing the parameter value sets alongside the
# median results for each simulation output measure. This names this file.
# Note no file extension
LHCSUMMARYFILENAME <- "LHC_Summary.csv"
# File name to give to the file showing the Partial Rank Correlation Coefficients
# for each parameter.
CORCOEFFSOUTPUTFILE <- "LHC_corCoeffs.csv"
# Timepoints being analysed. Must be NULL if no timepoints being analysed, or else
# be an array of timepoints. Scale sets the measure of these timepoints
TIMEPOINTS<-NULL; TIMEPOINTSCALE<-NULL
# Example Timepoints, if being used:
#TIMEPOINTS <- c(12, 36, 48, 60); TIMEPOINTSCALE <- "Hours"
```

In this example, we are analysing the result of a global sensitivity analysis for six parameters. The latin-hypercube has been used to produce 500 parameter value sets (`NUMSAMPLES`). Similarly to Techniques 1 and 2 above, this technique can process simulation responses in two formats. The first consists of an organised folder structure that contains CSV files containing the results of each simulation run (Figure 3(A)). The structure is comprised of two levels: the number of the latin-hypercube sample being analysed, containing simulation runs under those parameter conditions. However, if a researcher prefers, they can provide a CSV file summarising the simulation responses for all parameter sets generated by the hypercube. Each row of the CSV file should contain the parameter values selected from the hypercube and the median of each simulation response produced in that simulation run. For stochastic simulations where a number of runs are required to produce a consistent result (see Technique 1), a number of results for each parameter set should be provided in the CSV file. For example, in our case study we have repeated each simulation 300 times, thus 300 lines exist in the CSV file for each parameter value set generated by the hypercube.

It is this CSV file that the exemplar data from our website contains: the original simulation data has not been provided due to the large file size. However, this CSV file would be constructed using the method below. Similarly to Technique 2 above, the function processes each hypercube parameter value set in turn, producing a summary of the responses for each simulation run under those parameter value conditions. In the exemplar simulation data, it can be noted that multiple results have been included for each parameter set, to mitigate the impact of any aleatory uncertainty (Technique 1). With all parameter value sets processed, the CSV file is output with the filename stated in `LHC_ALL_SIM_RESULTS_FILE`.

```
lhc_process_sample_run_subsets(FILEPATH, LHC_PARAM_CSV_LOCATION, PARAMETERS, NUMSAMPLES,
        NUMRUNSPERSAMPLE, MEASURES, RESULTFILENAME, ALTERNATIVEFILENAME, OUTPUTCOLSTART,
        OUTPUTCOLEND, LHC_ALL_SIM_RESULTS_FILE)
```

With the CSV file in place (either constructed using the method above or provided), the next **spartan** function (below) summarises this further, generating the median of each simulation response for all runs under each hypercube parameter condition. The summary file is saved in the directory specified in `FILEPATH`, named as stated in `LHCSUMMARYFILENAME`.

```
lhc_generateLHCSummary(FILEPATH, PARAMETERS, MEASURES, LHC_ALL_SIM_RESULTS_FILE,
        LHCSUMMARYFILENAME, LHC_PARAM_CSV_LOCATION, TIMEPOINTS, TIMEPOINTSCALE)
```

With the parameter values and median of each simulation response together, **spartan** can now process this table. Each parameter is taken in turn, and the simulation responses ordered by the value assigned to that parameter. The objective is to identify any correlation between the value assigned to the parameter and the simulation response that is discernible despite the noise created by all other parameters varying randomly. Strong correlations correspond to influential parameters.

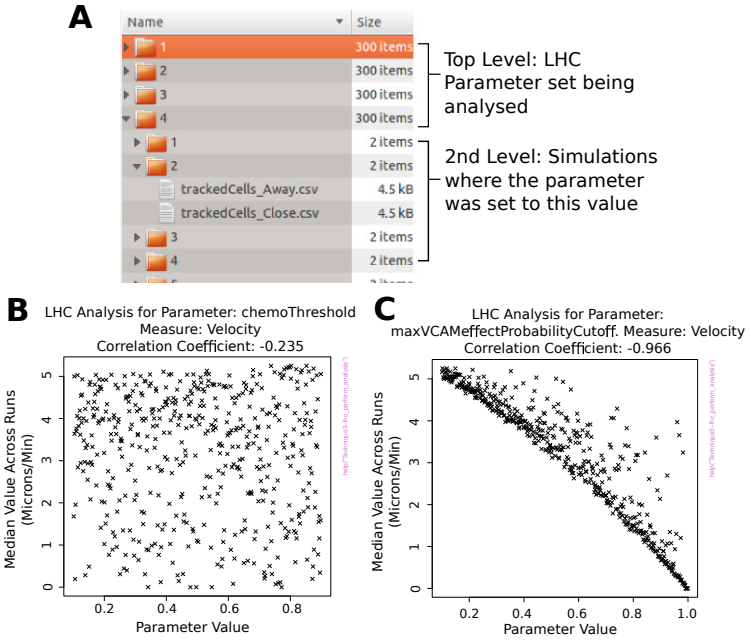


**Figure 3:** Use of **spartan** to identify compound effects between parameters using latin-hypercube sampling. A: The folder structure that should be utilised if providing all simulation result files rather than a CSV file summarising all results. B: Simulation responses sorted by the parameter that captures chemoattractant expression level required to influence cell motility, showing no trend between parameter value and result. C: Responses sorted by the parameter capturing the level of cell adhesion required to restrict cell motility, where a clear trend is apparent. B and C are reproduced from Alden et al. (2013b)

**spartan** provides two methods for helping to identify any correlations. The first is the calculation of Partial Rank Correlation Coefficients (PRCC) for each parameter. This statistical measure accounts for non-linear relationships between parameter and response, correcting for any effects by other parameters on the response. When the first of the below methods is run, these coefficients are calculated for each parameter, with the coefficients and p-values stored in a CSV file named as specified in `CORCOEFFSOUTPUTFILE`. The second method aids identification of any non-linear effects by producing a plot for each parameter, showing the simulation response observed for each value assigned to that parameter. Now any strong correlations, or any interesting areas where correlations appear at certain values, can be easily identified. The PRCC value is stated in the graph header, providing a statistical measure alongside the visual of simulation responses. Once the methods are run, these graphs will be found in the folder specified by `FILEPATH`.

```
lhc_generatePRCoEffs(FILEPATH, PARAMETERS, MEASURES, LHCSUMMARYFILENAME,
        CORCOEFFSOUTPUTFILE, TIMEPOINTS, TIMEPOINTSCALE)

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

Two of the plots produced for the example data can be seen in Figure 3(B) and 3(C). In Figure 3(B), no clear trend emerges for this parameter. In contrast, there is a strong correlation between the value of the parameter in Figure 3(C) and the simulation response (cell velocity). This suggests that, although a number of other parameters are also being perturbed, the value of this parameter has a significant impact on simulation response. Through this analysis, we identified that cell adhesion may be a key pathway at this point in lymphoid organ development, and thus is a pathway to be explored further in the laboratory. This demonstrates the impact that running a global sensitivity analysis could have on simulation response. With **spartan**, the researcher has the tools to perform both the parameter sampling and analysis of simulation data, to help draw such conclusions.

Similarly to the techniques above, it is possible to perform this technique for multiple simulation timepoints. To do this, the researcher needs to follow the same requirements detailed at the end of the description of Technique 1.

## Technique 4: Extended fourier amplitude sampling test

The above global sensitivity analysis technique is termed a sampling-based technique. As an alternative, **spartan** includes a variance-based technique, the Extended Fourier Amplitude Sampling Test (eFAST) (Marino et al., 2008; Saltelli and Bollardo, 1998). Whereas Technique 3 looks for correlations between simulation response and value of a specific parameter, eFAST partitions simulation output variance that is caused by parameter perturbation between the input parameters, providing a statistical measure of the proportion of variance accounted for by each parameter of interest. This measure gives a strong indication of the influence of each parameter, and thus the influential pathways in the simulation.

Of the four techniques this is by far the most complex, and the reader is directed to our **spartan** publication (Alden et al., 2013b) and the available literature that describes eFAST in detail (Marino et al., 2008; Saltelli and Bollardo, 1998) for a comprehensive description. As an overview, a parameter space is set for each parameter and an additional parameter, the 'dummy'. The reasoning for the introduction of a dummy parameter that has no impact on simulation response is noted later. Taking each in turn as that of interest, values are chosen for all parameters through the use of sinusoidal functions of a particular frequency through the parameter space, with the frequency assigned to the parameter of interest significantly different from that assigned to the others being examined. From each curve, a number of parameter values are selected. As sinusoidal curves have symmetrical properties, there is the potential that the same value sets could be chosen more than once. This is avoided by the introduction of resample curves, that introduce a phase shift into each frequency. With this introduced, the sampling is performed again, and repeated for a set number of resample curves (or phase shifts). It is vital the reader appreciates the importance of setting the number of samples from each curve and the number of resample curves correctly, making themselves familiar with equations in Marino et al. (2008) that aid this decision. From this process, the researcher creates a number of value sets for each parameter, for each curve. Where there are a large number of parameters in the analysis, this can lead to a large number of parameter value sets under which conditions simulations should be run. Thus this technique can be computationally expensive (Tarantola et al., 2006), especially for stochastic simulations where these runs need to be repeated.

The analysis of the simulation responses takes the frequency used to generate the parameter set into account. Utilising Fourier analysis, variation in the simulation response can be partitioned between the parameters under study. For each parameter, two statistical measures are calculated: the fraction of output variance that can be explained by the value the parameter has been assigned (Si), and the fraction of variance in the response due to higher-order non-linear affects between other simulation parameters and the parameter of interest. A parameter is deemed to have a statistically significant impact on simulation response when the sensitivity measures Si and STi are contrasted to those calculated for the dummy parameter, using a two-sample t-test.

### Parameter sampling

Generation of parameter samples using the eFAST technique noted above has been greatly aided by Marino et al. (2008) making their MATLAB code available, easing translation of this technique into R. The researcher should ensure they have read the available literature for this technique to ensure the number of curves and value samples taken from these curves is chosen appropriately (Alden et al., 2013b; Marino et al., 2008; Saltelli and Bollardo, 1998). As described in brief above, the analysis focuses on each parameter at a time, for each curve. Thus, when the below function is run, **spartan** produces a CSV file for each parameter, for each resample curve. In the example below, we have 7 parameters (6 plus the 'dummy') and 3 resample curves: leading **spartan** to produce 21 CSV files. Simulation runs should be performed for each parameter value set in these files: 1365 sets in the case of this example. It is easy to note how this technique becomes computationally expensive as parameter numbers increase, especially for stochastic simulations.

```
# Import the package
library(spartan)
# The folder where the parameter samples should be output to
FILEPATH <- "/home/user/eFAST/"
# Number of resample curves (phase shifts) to use in the sampling procedure
NUMCURVES <- 3
# Names of the parameters to generate parameter value samples for.
PARAMETERS <- c("BindProbability", "ChemoThreshold", "ChemoUpperLinearAdjust",
        "ChemoLowerLinearAdjust", "VCAMProbabilityThreshold", "VCAMSlope", "Dummy")
# The number of parameter sample sets to create for each curve
NUMSAMPLES <- 65
```

```
# The minimum value in the range for each parameter
PMIN <- c(0, 0.10, 0.10, 0.015, 0.1, 0.25, 1)
# The maximum value in the range for each parameter
PMAX <- c(100, 0.9, 0.50, 0.08, 1.0, 5.0, 10)

efast_generate_sample(FILEPATH, NUMCURVES, NUMSAMPLES, PARAMETERS, PMIN, PMAX)
```

### Analysing the results

Here we utilise our example simulation data to demonstrate the application of the eFAST technique available in **spartan**. The first step in this demonstration is to define the objects required for this analysis, as below. Again to save space, we have not duplicated the comments for objects that were declared in previous examples: the reader should refer to these above if necessary. To aid graphing of the results generated by this analysis, we utilise the **gplots** R package.

```
library(spartan)
library(gplots)
# Folder containing the eFAST simulation results. Make sure example data is unzipped
FILEPATH <- "/home/user/eFAST/Results"
PARAMETERS <- c("BindProbability", "ChemoThreshold", "ChemoUpperLinearAdjust",
        "ChemoLowerLinearAdjust", "VCAMProbabilityThreshold", "VCAMSlope", "Dummy")
MEASURES <- c("Velocity", "Displacement")
RESULTFILENAME <- "trackedCells_Close.csv"
ALTERNATIVEFILENAME <- NULL
OUTPUTCOLSTART <- 10
OUTPUTCOLEND <- 11
# Number of resample curves employed when the parameter space was sampled
NUMCURVES <- 3
# The number of parameter sample sets taken from each curve
NUMSAMPLES <- 65
# Number of simulation runs performed for each parameter value set
NUMRUNSPERSAMPLE <- 300
# Which of the output measures to T-Test for significance
OUTPUTMEASURES_TO_TTEST <- 1:2
# T-Test confidence level
TTEST_CONF_INT <- 0.95
# Name of the final result file for this analysis, showing the partitioning of
# the variance between input parameters
EFASTRESULTFILENAME <- "EgSet_eFAST_Analysis.csv"
# Boolean to note whether summary graphs should be produced
GRAPH_FLAG <- TRUE
TIMEPOINTS <- NULL; TIMEPOINTSCALE <- NULL
```

In this example, we are utilising Fourier frequency analysis to examine the results of perturbing six parameters simultaneously (7 including the dummy). Similarly to all the previous techniques described in this paper, the researcher can specify their simulation result input in two formats. The first consists of the organised folder structure described in Figure 4(A). The structure is comprised of four levels: the sample curve number, then parameter of interest for that curve, containing the number of the set of simulator values generated for that parameter, that in turn holds the responses for simulations run under those conditions. As noted previously, such a structure is suitable in cases where a large number of simulation runs have been performed, usually on a computer cluster. This technique is one of these cases, as the large number of parameter value sets generated requires a large number of simulation results. However, the researcher can also provide their input in a series of CSV files, one per curve-parameter pairing. Each row of this file should contain the parameter set generated from the sinusoidal curve and the median simulation response when a run was performed under those conditions. Multiple rows can exist per parameter set, representing multiple runs performed to mitigate aleatory uncertainty. If the researcher chooses this method, they should follow the convention of naming the file *Curve[Curve Number]_Parameter[Parameter Number]_Results.csv*, for example Curve1_Parameter1_Results.csv.

The exemplar simulation data contains the latter: the summary CSV files for each curve-parameter pairing. With this analysis requiring a large number of simulation runs, it has not been possible to provide all the simulation runs in an appropriate size download. Similarly to the previously described

techniques, **spartan** processes the runs in these folders to create a CSV file summarising the simulation runs: one per curve-parameter pair, formatted as described above. In our case study, where we have multiple values for each response (i.e. cells), the median of these responses is calculated and stored in the summary file, for each run. Where the responses for all simulation runs are available, the following function can be used to produce these files:

```
efast_generate_medians_for_all_parameter_subsets(FILEPATH, NUMCURVES, PARAMETERS,
        NUMSAMPLES, NUMRUNSPERSAMPLE, MEASURES, RESULTFILENAME, ALTERNATIVEFILENAME,
        OUTPUTCOLSTART, OUTPUTCOLEND, TIMEPOINTS, TIMEPOINTSCALE)
```

With the CSV file summaries in place (either provided by the researcher or generated using the function above), the variance in simulation response can be partitioned between the parameters. To do this, the algorithm needs to generate a further summary of all simulation responses: one for each parameter resampling curve employed. This is generated by the following function. A CSV summary file is produced that shows, for each parameter of interest, the median of each simulation output response under the generated parameter value set conditions. As an example, consider we are examining curve 1, parameter 1. For this parameter of interest, 65 different parameter value sets were generated from the frequency curves, thus we have 65 different sets of simulation results. The previous method produced a summary of the responses for each simulation run under those conditions: this method now takes the median of each response and stores this in the summary alongside the parameter values that generated these medians. Thus, for each parameter of interest, median responses for 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. This is stored in the directory specified by the `FILEPATH` object, with the name Curve[Curve Number]_Summary.csv.

```
efast_get_overall_medians(FILEPATH, NUMCURVES, PARAMETERS, NUMSAMPLES, MEASURES,
        TIMEPOINTS, TIMEPOINTSCALE)
```

It is this summary that can then be processed by the eFAST algorithm, producing statistical measures that partition the variance in simulation response between the parameters of interest:

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

In the folder specified in `FILEPATH`, an overall summary of the analysis will have been produced, named as stated in `EFASTRESULTFILENAME`. The summary file contains a row showing the name of each parameter being examined, followed by a number of statistical measures generated for each simulation response: the fraction of output variance explained by a variation of that parameter (Si); the fraction of variance accounted for by that parameter and any higher-order or non-linear effects between the parameter and others (STi); the variance caused by all parameters excluding that of interest (SCi); p-values for Si and STi when contrasted with those of the dummy parameter using the two-sample t-test; Standard Error values, which can be calculated as there are a number of resample curves. For ease of representation, a graph is produced for each simulation output response showing the Si and STi values: as demonstrated in Figure 4(B). A parameter is deemed to have a significant impact on simulation response when the Si and STi statistical measures are contrasted with those for the dummy parameter, known to have no effect on simulation response. In terms of the example data, it is clearly apparent from the graph that one parameter has a highly significant effect on the cell velocity simulation response. Two-sample t-test comparisons however suggest there are four parameters that are statistically significant in contrast with the dummy parameter, with no significant role for the pathways represented by the remaining two parameters.

Again it is possible to perform this technique for multiple simulation timepoints, if the requirements detailed at the end of the description of Technique 1 are followed.

Although this is a complex statistical technique, gaining a statistical measure of the impact of each parameter on simulation variance is useful in fully understanding simulator behaviour, especially if this conclusion is supported by other techniques above. Unless the reader has significant computational resources, this technique is more powerful for analyses containing just a few parameters.

## Conclusion

Here we have presented an exemplar application of the statistical techniques available in **spartan**. The four techniques that we have compiled within the package have proved very fruitful in suggesting

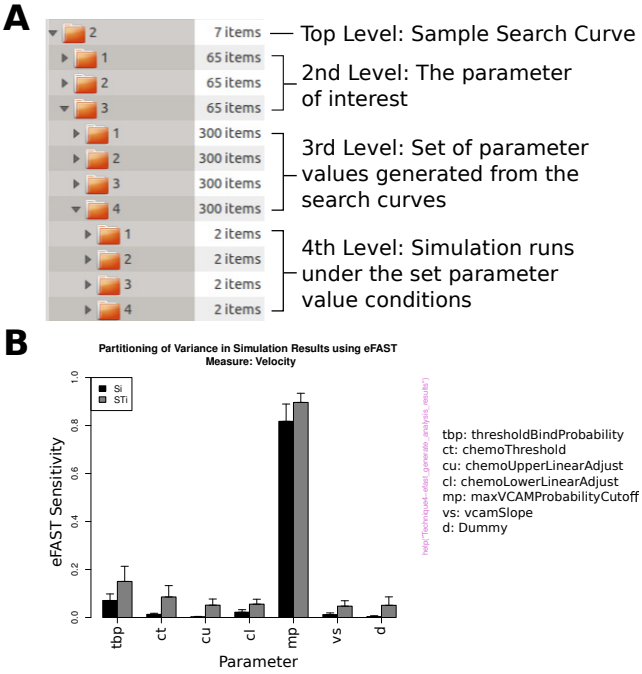


**Figure 4:** Use of eFAST in partitioning variance in response between parameters. A: The folder structure that should be utilised if providing all simulation result files rather than a CSV file summarising all results. B: Sensitivity measures for the cell velocity response. Black bars: Si - variance explained by the value assigned to that parameter. Grey bars: STi: Variance caused by this parameter and non-linear effects between this parameter and others. B is reproduced from Alden et al. (2013b).

potential biological pathways that should be explored further in attempts to understand the development of secondary lymphoid tissue. Through providing this demonstration, example simulation data, and the accompanying **spartan** publication, we encourage researchers across disciplines to utilise the toolkit to increase confidence in their generated simulator. **spartan** can produce statistical analyses to accompany simulation results, revealing that the reseacher has considered the effect that any inherent stochasticity has on their results, that they understand how robust their simulator is to parameter perturbation, and that they understand the key pathways or mechanisms identified through global sensitivity analysis.

The scope for applying the techniques available is much wider than agent-based models of biological systems: capacity exists to utilise **spartan** alongside traditional and partial differential equation models as well as those where an agent-based approach has been adopted. In these scenarios the researcher would not need to perform the first function of all the techniques described above, where the results from a number of runs under the same parameter condition are summarised to take stochastic variation into account. Instead, single simulation results are placed into the same file structure described in each method, and the **spartan** analysis run as described above.

## Acknowledgements

This work was part-funded by the Wellcome Trust [ref:097829] through the Centre for Chronic Diseases and Disorders (C2D2) at the University of York. Jon Timmis is part funded by the Royal Society and The Royal Academy of Engineering. Mark Read was funded by FP7: ICT grant GA 270382 “Collective Cognitive Robotics,” Paul Andrews was funded by EPSRC grant EP/I005943/1 “Resilient Futures.”

## Bibliography

K. Alden, J. Timmis, P. S. Andrews, H. Veiga-Fernandes, and M. C. Coles. Pairing experimentation and computational modelling to understand the role of tissue inducer cells in the development of lymphoid organs. *Frontiers in Immunology*, 3:1–20, 2012. [p63, 64, 69]

K. Alden, M. Read, P. Andrews, J. Timmis, H. Veiga-Fernandes, and M. Coles. *spartan: Spartan (Simulation Parameter Analysis R Toolkit ApplicatioN)*, 2013a. URL http://CRAN.R-project.org/package=spartan. R package version 2.1. [p64]

K. Alden, M. Read, J. Timmis, P. S. Andrews, H. Veiga-Fernandes, and M. C. Coles. Spartan: A comprehensive tool for understanding uncertainty in simulations of biological systems. *PLoS Computational Biology*, 9(2), 2013b. [p63, 64, 67, 71, 74, 75, 78]

R. Carnell. *lhs: Latin hypercube samples*, 2012. URL http://CRAN.R-project.org/package=lhs. R package version 0.10. [p64]

G. M. Dancik. *mlegp: Maximum likelihood estimates of gaussian processes*, 2013. URL http://CRAN.R-project.org/package=mlegp. R package version 3.1.4. [p63]

J. C. Helton. Uncertainty and sensitivity analysis for models of complex systems. In T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick, editors, *Computational Methods in Transport: Verification and Validation*, pages 207–228. Springer, 2008. [p64]

M. Ivanyi, R. Bocsi, L. Gulyas, V. Kozma, and R. Legendi. The multi-agent simulation suite. In *AAAI Fall Symposium Series*, pages 56–63, 2007. [p63]

M. Lamboni, H. Monod, and C. Bidot. *multisensi: Multivariate sensitivity analysis*, 2013. URL http://CRAN.R-project.org/package=multisensi. R package version 1.0.7. [p63]

D. T. Lang. *XML: Tools for parsing and generating XML within R and S-Plus.*, 2013. URL http://CRAN.R-project.org/package=XML. R package version 3.98-1.1. [p64]

S. Luke. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005. [p63]

S. Marino, I. B. Hogue, C. J. Ray, and D. E. Kirschner. A methodology for performing global uncertainty and sensitivity analysis in systems biology. *Journal of Theoretical Biology*, 254(1):178–96, 2008. [p63, 71, 75]

M. Meier-Schellersheim, X. Xu, B. Angermann, E. J. Kunkel, T. Jin, and R. N. Germain. Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method. *PLoS Computational Biology*, 2(7):710–724, 2006. [p63]

A. Patel, N. Harker, L. Moreira-Santos, M. Ferreira, K. Alden, J. Timmis, K. E. Foster, A. Garefalaki, P. Pachnis, P. S. Andrews, H. Enomoto, J. Milbrandt, V. Pachnis, M. C. Coles, D. Kioussis, and H. Veiga-Fernandes. Differential RET responses orchestrate lymphoid and nervous enteric system development. *Science Signalling*, 5(235), 2012. [p63, 64]

G. Pujol, B. Iooss, A. Janon, P. Lemaitre, L. Gilquin, L. L. Gratiet, T. Touati, B. Ramos, J. Fruth, and S. D. Veiga. *sensitivity: Sensitivity Analysis*, 2014. URL http://CRAN.R-project.org/package=sensitivity. R package version 1.9. [p63]

M. Read, P. S. Andrews, J. Timmis, and V. Kumar. 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, 2012. [p63, 64, 66, 68, 71, 72]

A. Saltelli and R. Bollardo. An alternative way to compute Fourier amplitude sensitivity test (FAST). *Comput. Stat. Data Anal.*, 26(4):445–460, 1998. [p63, 75]

A. Saltelli, K. Chan, and E. M. Scott. *Sensitivity analysis*. Wiley series in probability and statistics Wiley, 2000. [p71]

S. Tarantola, D. Gatelli, and T. Mara. Random balance designs for the estimation of first order global sensitivity indices. *Reliability Engineering & System Safety*, 91(6):717–727, 2006. [p75]

A. Vargha and H. D. Delaney. A critique and improvement of the cl common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioural Statistics*, 25:101–132, 2000. [p64, 68]

G. R. Warnes, B. Bolker, L. Bonebakker, R. Gentleman, W. H. A. Liaw, T. Lumley, M. Maechler, A. Magnusson, S. Moeller, M. Schwartz, and B. Venables. *gplots: Various R programming tools for plotting data*, 2013. URL http://CRAN.R-project.org/package=gplots. R package version 2.12.1. [p64]

U. Wilensky. *NetLogo itself*. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL., 1999. URL http://ccl.northwestern.edu/netlogo/. [p63]

*Kieran Alden*
*York Computational Immunology Laboratory, Centre for Immunology & Infection, and Dept of Electronics*
*University of York, Heslington, York. YO10 5DD*
*UK* kieran.alden@york.ac.uk

*Mark Read*
*York Computational Immunology Laboratory and Dept of Electronics*
*University of York, Heslington, York. YO10 5DD*
*UK* mark.read@sydney.edu.au

*Paul Andrews*
*York Computational Immunology Laboratory and Dept of Computer Science*
*University of York, Heslington, York. YO10 5DD*
*UK* paul.andrews@york.ac.uk

*Jon Timmis*
*York Computational Immunology Laboratory and Dept of Electronics*
*University of York, Heslington, York. YO10 5DD*
*UK* jon.timmis@york.ac.uk

*Mark Coles*
*York Computational Immunology Laboratory and Centre for Immunology & Infection*
*University of York, Heslington, York. YO10 5DD*
*UK* mark.coles@york.ac.uk