Package 'spartan'

January 27, 2015

Type Package

Title Spartan (Simulation Parameter Analysis R Toolkit ApplicatioN)

Version 2.1

Date 2014-08-19

Author
Kieran Alden, Mark Read, Paul Andrews, Jon Timmis, Henrique Veiga-Fernandes, Mark Coles

Maintainer Kieran Alden <kieran.alden@gmail.com>

Description Computer simulations are becoming a popular technique to use in attempts to further our understanding of complex systems. SPARTAN, described in our 2013 publication in PLoS Computational Biology, 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. Version 1.3 adds support for Netlogo simulations, aiding simulation developers who use Netlogo to build their simulations perform the same analyses. We have also added user support through the group spartan-group[AT]york[DOT]ac[DOT]uk. Version 2.0 added the ability to read all simulations in from a single CSV file in addition to the prescribed folder structure in previous versions.

Suggests lhs, gplots, XML

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-19 18:41:52

R topics documented:

`Technique 1: Aleatory Analysis`

Technique 1: Aleatory Analysis

Description

Aleatory uncertainty is caused by inherent stochasticity within a simulation. For stochastic simulations, a number of replicate runs need to be performed for each parameter set, thus achieving a more representative result. This technique indicates the number of simulation runs necessary to reduce this uncertainty. This follows the method described by Read et al in the reference below. To use this, you should have chosen the number of runs you want to compare (for example, 1,5,50,100,300,500,and 800). For each sample size, you should choose a number of subsets (for example, 20). Then, you should create this number of subsets of runs for each sample size. So, choosing the values from the examples, we have 20 sets where the simulation was run once, 20 sets where each set contains the results of 5 runs, right through to 20 sets where each contains the results of 800 runs. This method then looks at each sample size used, and (a) generates the median distribution for all output measures for each of the subsets (b) goes through each of the subsets (20 in the example case), comparing the median of each measure with the respective median in the first subset, using the Vargha-Delaney A-Test (reference below) to give an indication of how different the results are, (c) for each sample size, creates graphes showing how different the results of each subset are (i.e. the A-Test result). A summary of the A-Test results is also output as a Comma Separated Value file, an example of which can be found in the data folder of this package (AA_Example_ATestMaxAndMedians.csv). A full tutorial on using this technique, along with example simulation output to use, can be found on the project website.

Note 1: From Spartan 2.0, you can specify your simulation data in two ways:
A - Set folder structure (as in previous versions of Spartan): This is shown in figure AA_Folder_Struc.png within the extdata folder of this package, and described in detail in the tutorial. Using this structure, the parameter FILEPATH should point to a directory that contains a folder that in turn contains the results for the sample sizes being analysed. For example, if the sample sizes being analysed were 1, 5, 50, 100, 300, 500, and 800, the folder specified by FILEPATH would contain seven folders, one for each of these sample sizes. The folder for each sample size then contains one folder for each

of the result subsets. In the example case, this would be 20 folders, numbered 1-20. Each of these folders will contain the number of simulation runs performed for the sample size being analysed. For example, if the uncertainty of 5 simulation runs is being examined, each of the 20 folders will contain the results from 5 simulation runs. If 100 runs were being examined, each of the 20 folders would contain the results from 100 runs, and so on. The folders containing the results from each run should be numbered from 1 to the number of runs performed.
B - Single CSV file Input. From Spartan 2.0, you can specify all your results in a single CSV file. An example of this file can be found in the extdata folder of the package, named AA_SimResponses.csv. Each row of this file should correspond to one of the sample sizes and subsets to be analysed. The first two columns should therefore be the sample size being analysed and the number of the subset for this sample size. Remaining columns then list the simulation responses for that set. For example, consider a sample set of 5. The first set of 5 runs, set 1, will exist in this CSV file as 5 rows, with the simulation result from each of the 5 runs in this set. See either the input structure detail on the YCIL website, or the example file for more detail. This technique will then process this file rather than the folder structure as Spartan did previously.

Note 2: From Spartan 2.0, performing this analysis at multiple timepoints is now performed using the same method calls below. There are no additional method calls for timepoint analysis.

This technique consists of four methods:
aa_summariseReplicateRuns: Only to be applied in cases where simulation responses are supplied in the folder structure (as in all previous versions of Spartan), useful for cases where the simulation is agent-based. Iterates through simulation runs for each sample size creating a CSV file containing results for all sample sizes and all subsets (in the same format as the new CSV file format discussed above). Where a simulation response is comprised of a number of records (for example a number of cells), the median value will be recorded as the response for this subset of the sample size being analysed. This file is output to a CSV file, named as stated by the parameter MEDIANS_SUMMARY_FILE_NAME. If doing this analysis over multiple timepoints, the timepoint will be appended to the filename given in MEDIANS_SUMMARY_FILE_NAME.
aa_getATestResults: Examines the CSV file produced either by the method above or provided by the user, analysing each sample size independently, to determine how 'different' the results of each of the subsets are. For each sampple size, the distribution of responses for each subset are compared with the first subset using the Vargha-Delaney A-Test. These scores are stored in a CSV file, with filename as stated in parameter ATESTRESULTSFILENAME. The A-Test results for a sample size are then graphed, showing how different each of the subsets are. An example graph can be seen in the extdata folder of this package (AA_5Samples.pdf). If doing this analysis over multiple timepoints, the timepoint will be appended to the filename given in ATESTRESULTSFILENAME and appended to the name of the graph.
aa_sampleSizeSummary: This takes each sample size to be examined in turn, and iterates through all the subsets, determining the median and maximum A-Test score observed for each sample size. A CSV file is created summarising the median and maximum A-Test scores for all sample sizes, named as stated in parameter SUMMARYFILENAME. If doing this analysis over multiple timepoints, the timepoint will be appended to the filename given in SUMMARYFILENAME.
aa_graphSampleSizeSummary: Produces a full graph of the data generated by the above method (by full, we mean the y-axis (the A-Test score) goes from 0-1, and the x axis contains all sample sizes examined), making it easy to see how uncertainty reduces with an increase in sample size. An example can be seen in the extdata folder of this package (AA_Results.pdf). This graph is named as stated in the parameter GRAPHOUTPUTFILE, with the timepoint appended if the analysis is for

multiple timepoints.

Usage

```
aa_summariseReplicateRuns(FILEPATH,SAMPLESIZES,MEASURES,RESULTFILENAME,
ALTFILENAME,OUTPUTFILECOLSTART,OUTPUTFILECOLEND,
AA_SIM_RESULTS,TIMEPOINTS=NULL,TIMEPOINTSCALE=NULL)

aa_getATestResults(FILEPATH,SAMPLESIZES,NUMSUBSETSPERSAMPLESIZE,
MEASURES,AA_SIM_RESULTS,ATESTRESULTSFILENAME,
LARGEDIFFINDICATOR,TIMEPOINTS=NULL,TIMEPOINTSCALE=NULL,GRAPHNAME=NULL)

aa_sampleSizeSummary(FILEPATH,SAMPLESIZES,MEASURES,ATESTRESULTSFILENAME,
SUMMARYFILENAME,TIMEPOINTS=NULL,TIMEPOINTSCALE=NULL)

aa_graphSampleSizeSummary(FILEPATH,MEASURES,MAXSAMPLESIZE,SMALL,MEDIUM,
LARGE,SUMMARYFILENAME,GRAPHOUTPUTFILE,TIMEPOINTS=NULL,
TIMEPOINTSCALE=NULL,GRAPHLABEL=NULL)
```

Arguments

FILEPATH	Directory where the results of the simulation runs, in folders or in single CSV file format, can be found
SAMPLESIZES	The sample sizes chosen (i.e. in our case, this would be an array containing 1,5,50,100,300,800)
NUMSUBSETSPERSAMPLESIZE	The number of subsets for each sample size (i.e in the tutorial case, 20)
RESULTFILENAME	Name of the simulation results file (e.g. "trackedCells_Close.csv"). In the current version, XML and CSV files can be processed. Only required if running the first method (to process results directly). If performing this analysis over multiple timepoints, it is assumed that the timepoint follows the file name, e.g. trackedCells_Close_12.csv.
ALTFILENAME	In some cases, it may be relevant to read from a further results file if the initial file contains no results. This filename is set here. In the current version, XML and CSV files can be processed. Only required if running the first method (to process results directly)
OUTPUTFILECOLSTART	Column number in the simulation results file where output begins - saves (a) reading in unnecessary data, and (b) errors where the first column is a label, and therefore could contain duplicates. Only required if running the first method (to process results directly)
OUTPUTFILECOLEND	Column number in the simulation results file where the last output measure is. Only required if running the first method.

`MEASURES` An array containing the names of the simulation output measures to be analysed. For example, in the tutorial simulation, we tracked a cells Velocity and Displacement. Our array would contain these two strings

`AA_SIM_RESULTS` Either - A: The name of the summary CSV file to be created by the first method (aa_summariseReplicateRuns) or B: The name of the provided CSV file that summarises the results of all runs for this analysis.

`ATESTRESULTSFILENAME`
Name of the file that will contain the A-Test scores for each sample size (created by aa_getATestResults).

`LARGEDIFFINDICATOR`
The A-Test determines there is a large difference between two sets if the result is greater than 0.2 either side of the 0.5 line. Should this not be suitable, this can be changed here

`SUMMARYFILENAME`
Name of the file generated by aa_sampleSizeSummary, listing the maximum and median A-Test results for each sample size.

`MAXSAMPLESIZE` The highest number of samples used. In our example case, this would be set to 300

`SMALL` The figure (>0.5) which is deemed a "small difference" between two sets being compared. Vargha-Delaney set this value to 0.56 - but this can be altered here

`MEDIUM` The figure (>0.5) which is deemed a "medium difference" between two sets being compared. Vargha-Delaney set this value to 0.66 - but this can be altered here

`LARGE` The figure (>0.5) which is deemed a "large difference" between two sets being compared. Vargha-Delaney set this value to 0.73 - but this can be altered here

`GRAPHOUTPUTFILE`
Filename that should be given to the generated summary graph. This must have a PDF file extension

`TIMEPOINTS` Implemented so this method can be used when analysing multiple simulation timepoints. If only analysing one timepoint, this should be set to NULL. If not, this should be an array of timepoints, e.g. c(12,36,48,60)

`TIMEPOINTSCALE` Implemented so this method can be used when analysing multiple simulation timepoints. Sets the scale of the timepoints being analysed, e.g. "Hours"

`GRAPHNAME` Used internally by the getATestResults method when producing graphs for multiple timepoints. Should not be set in function call.

`GRAPHLABEL` Used internally by the getATestResults method when producing graphs for multiple timepoints. Should not be set in function call.

References

This technique is described by Read et al (2012) in their paper: Techniques for Grounding Agent-Based Simulations in the Real Domain: a case study in Experimental Autoimmune Encephalomyelitis." The A-Test is described by by Vargha & Delaney (2000): "A critique and improvement of the CL Common Language Effect Size Statistics of McGraw and Wong"

Examples

```
# THE CODE IN THIS EXAMPLE IS THE SAME AS THAT USED IN THE TUTORIAL, AND
# THUS YOU NEED TO DOWNLOAD THE TUTORIAL DATA SET AND SET FILEPATH
# CORRECTLY TO RUN THIS

##---- Firstly, declare the parameters required for the four functions ----
library(XML)
library(spartan)

# The directory where you have extracted the example simulation results.
FILEPATH <- "/home/user/Downloads/AA_ABM/"
# The sample sizes that are to be analysed, contained within an array
SAMPLESIZES <- c(1,5,50,100,300)
# The simulation output measures to be analysed, again contained within an array
MEASURES<-c("Velocity","Displacement")
# The number of subsets used. By default use 20, as performed by Read et al in
# their published technique
NUMSUBSETSPERSAMPLESIZE<-20
# The output file containing the simulation results from that simulation run. Note
# there should be no file extension
RESULTFILENAME<-"trackedCells_Close.csv"
# Not used in this case, but this is useful in cases where two result files may
# exist (for example if tracking cells close to an area, 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
# Use this if simulation results are in CSV format.
# The column within the csv results file where the results start. This is useful
# as it restricts what is read in to R, getting round potential errors where the
# first column contains an agent label (as R does not read in CSV files where the
# first column contains duplicates)
OUTPUTFILECOLSTART<-10
# 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 to contain 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
AA_SIM_RESULTS<-"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"
# 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. Note no file extension
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:
#TIMEPOINTS<-c(12,36,48,60); TIMEPOINTSCALE<-"Hours"

## Not run:
# DONTRUN IS SET SO THIS IS NOT EXECUTED WHEN PACKAGE IS COMPILED - BUT THIS
# HAS BEEN TESTED WITH THE TUTORIAL DATA

##--- NOW RUN THE FOUR METHODS IN THIS ORDER ----

# A: RUN WHEN PROCESSING FOLDER STRUCTURE RESULTS FOR STOCHASTIC SIMULATIONS
aa_summariseReplicateRuns(FILEPATH,SAMPLESIZES,MEASURES,RESULTFILENAME,
ALTFILENAME,OUTPUTFILECOLSTART,OUTPUTFILECOLEND,AA_SIM_RESULTS,
TIMEPOINTS=TIMEPOINTS,TIMEPOINTSCALE=TIMEPOINTSCALE)

# B: GET A-TEST SCORES FOR ALL SAMPLE SIZES. PRODUCES A PLOT FOR ALL SAMPLE SIZES
aa_getATestResults(FILEPATH,SAMPLESIZES,NUMSUBSETSPERSAMPLESIZE,MEASURES,
AA_SIM_RESULTS,ATESTRESULTSFILENAME,LARGEDIFFINDICATOR,
TIMEPOINTS=TIMEPOINTS,TIMEPOINTSCALE=TIMEPOINTSCALE)

# C: SUMMARISE THESE RESULTS, OBTAINING MAX AND MIN FOR ALL SAMPLE SIZES
aa_sampleSizeSummary(FILEPATH,SAMPLESIZES,MEASURES,ATESTRESULTSFILENAME,
SUMMARYFILENAME,TIMEPOINTS=TIMEPOINTS,TIMEPOINTSCALE=TIMEPOINTSCALE)

# D: GRAPH THE SUMMARY OF ALL SAMPLE SIZES
aa_graphSampleSizeSummary(FILEPATH,MEASURES,300,SMALL,MEDIUM,LARGE,
SUMMARYFILENAME,GRAPHOUTPUTFILE,TIMEPOINTS=TIMEPOINTS,
TIMEPOINTSCALE=TIMEPOINTSCALE)

## End(Not run)
```

Technique 2: One-At-A-Time - Generate Parameter Value Sets

Technique 2: One-At-A-Time - Generate Parameter Value Sets

Description

The robustness of a simulation to parameter alteration can be determined through the use of this approach. Following the method described by Read et al in the reference below, the value of each parameter is adjusted independently, with the remaining parameters staying unchanged from their calibrated value. This method within the toolkit creates a set of simulation parameter sets

to enable such an analysis to be performed. One CSV file is created for each parameter being examined (with the filename being [Parameter Name]_Values.csv). Each CSV file will contain the parameters for runs that need to be performed. For each set of parameters, the simulation should be run for the number of times determined by Aleatory Analysis (Technique 1). Once this has been completed, the results can be analysed using the next method included within Technique 2 of this package. Two examples of the files produced exist in the data folder of this package (OAT_Sampling_chemoThreshold_Values.csv and OAT_Sampling_maxVCAMProabilityCutoff.csv)

Usage

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

Arguments

FILEPATH	Directory where the parameter samples should be output to
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
BASELINE	Array containing the values assigned to each of these parameters in the calibrated baseline
PMIN	Array containing the minimum value that should be used for each parameter. Sets a lower bound on sampling space
PMAX	Array containing the maximum value that should be used for each parameter. Sets an upper bound on sampling space
PINC	Array containing the increment value that should be applied for each parameter. For example, a parameter could have a minimum value of 10, and maximum value of 100, and be incremented by 10
PARAMVALS	Array containing a list of strings for each parameter, each string containing comma separated values that should be assigned to that parameter. Thus sampling can be performed for specific values for each parameter, rather than a uniform incremented value. This replaces the PMIN, PMAX, and PINC where this method is used.

References

This technique is described by Read et al (2012) in their paper: Techniques for Grounding Agent-Based Simulations in the Real Domain: a case study in Experimental Autoimmune Encephalomyelitis"

Examples

```
##- Firstly, declare the parameters required for the function -
# A: THE ROOT FILE PATH.
# WHERE THE PARAMETER SAMPLES SHOULD BE STORED
FILEPATH<-"/home/kieran/Downloads/OAT/"
# B: THE SIMULATION PARAMETERS BEING EXPLORED
PARAMETERS <- c("thresholdBindProbability","chemoThreshold",
"chemoUpperLinearAdjust","chemoLowerLinearAdjust",
```

```
"maxVCAMeffectProbabilityCutoff","vcamSlope")
# C: BASELINE VALUES FOR ALL PARAMETERS (CALIBRATION VALUES).
# THIS SETS THE VALUES OF THE OTHER PARAMETERS WHEN ANOTHER
# IS THAT OF FOCUS IN THE ANALYSIS
BASELINE<-c(50,0.3,0.2,0.04,0.60,1.0)
# E: PARAMETER VALUE INFORMATION
# YOU CAN SPECIFY THIS IN TWO WAYS: (i) THE MINIMUM AND
# MAXIMUM OF EACH PARAMETER, AND THE INCREMENT OVER WHICH THE
# SAMPLING WAS INCREASED (ii) A STRING LIST OF VALUES THAT
# PARAMETER WAS ASSIGNED IN SIMULATION
# EXAMPLE OF (i):
#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)
# EXAMPLE OF (ii)
#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")


##--NOW RUN THE METHOD, CHOOSING EITHER PMIN PMAX PINC OR PARAMVALS
## Not run:
oat_parameter_sampling(FILEPATH,PARAMETERS,BASELINE,PMIN=NULL,
PMAX=NULL,PINC=NULL,PARAMVALS=PARAMVALS)
oat_parameter_sampling(FILEPATH,PARAMETERS,BASELINE,PMIN=PMIN,
PMAX=PMAX,PINC=PINC,PARAMVALS=NULL)

## End(Not run)
```

Technique 2: One-At-A-Time - Perform Analysis of Results

One-At-A-Time - Perform Analysis of Results

Description

The robustness of a simulation to parameter alteration can be determined through the use of this approach. Following the method described by Read et al in the reference below, the value of each parameter is adjusted independently, with the remaining parameters staying unchanged from their calibrated value. Distributions of simulation responses under the perturbed parameter condition are compared with those at simulation baseline/calibrated values using the Vargha-Delaney A-Test. This test gives an indication of how different the two sets of results are. The set of A-Test results for each parameter is output to a CSV file for reference. Finally, a graph for each parameter is produced, showing the A-Test result for each parameter value and each simulation output measure. In addition, from Version 2.0, an additional method has been introduced that can be applied to stochastic simulations, capable of producing a count of the number of responses for a particular parameter that are equal to a certain criteria (such as true or false). This may provide additional information concerning how a simulation is behaving.

Note 1: From Spartan 2.0, you can specify your simulation data in two ways:
A - Set folder structure (as in previous versions of Spartan): This is shown in figure OAT_Folder_Struc.png

within the extdata folder of this package, and described in detail in the tutorial.
B - Single CSV file Input. From Spartan 2.0, you can specify all your results in a single CSV file. An example of this file can be found in the extdata folder of the package, named OAT_Medians.csv. Each row of this CSV file should contain the parameters upon which the simulation was run and the simulation response under those conditions. This may be a median value of those responses. There may be duplicate results for a parameter set, where the simulation has been run a number of times under the same condition (required for stochastic simulations).
Note 2: From Spartan 2.0, performing this analysis at multiple timepoints is now performed using the same method calls below. There are no additional method calls for timepoint analysis - the timepoints are specified in each method call.
Note 3: From Spartan 2.0, this method can also process parameter values that are specified as a list, rather than an increment between a minimum and maximum value. This may be useful for analysing specific values over a large range.

This technique consists of five methods:
oat_processParamSubsets: This method should only be used for stochastic simulations where the data is provided in the set folder structure (as in previous versions of Spartan). Each parameter, and all values that it has been assigned, are examined in turn. For each replicate run under those parameter conditions, the median of the simulation response is calculated. These medians for each simulation replicate, of each parameter set, are stored in a CSV file, creating the same single CSV file format that can also be provided as Spartan input. This file is named as stated in parameter CSV_FILE_NAME. This method can be performed for a number of simulation timepoints, producing these statistics for each timepoint taken.
oat_csv_result_file_analysis: This method takes either the CSV file created in the previous method or that provided by the user and analyses the impact that a change in a single parameter value has had on simulation response. This is performed by comparing the distribution of responses for a perturbed parameter condition with the distribution under baseline/calibrated conditions. This produces a CSV file, in the directory stated in FILEPATH, named as stated by parameter ATESTRESULTSFILENAME, containing the A-Test scores for all parameter conditions under which the simulation was run. An example of this file can be seen in the extdata folder of this package (OAT_ATestScores.csv). This method can be performed for a number of simulation timepoints, producing these statistics for each timepoint taken. **oat_graphATestsForSampleSize**: This takes each parameter in turn and creates a plot showing A-Test score against parameter value. This makes it easy to determine how the effect that changing the parameter has had on simulation results. Two examples can be found in the extdata folder of this package (OAT_chemoLowerLinearAdjust_Displacement.pdf and OAT_chemoUpperLinearAdjust.pdf).
oat_plotResultDistribution: Only applicable for stochastic simulations where the results are provided in the folder strutcure: this takes each parameter in turn, and creates a boxplot for each output measure, showing the result distribution for each value of that parameter. An example can be found in the extdata folder of this package (chemoLowerLinearAdjust_DisplacementBP.pdf).
oat_countResponsesOfDesiredValue: Counts the number of simulation responses where a output response equals a desired result, for a specified parameter. Outputs this information as a CSV file.

Usage

```
oat_processParamSubsets(FILEPATH,PARAMETERS,NUMRUNSPERSAMPLE,
MEASURES,RESULTFILENAME,ALTERNATIVEFILENAME,
OUTPUTCOLSTART,OUTPUTCOLEND,CSV_FILE_NAME,BASELINE,
PMIN=NULL,PMAX=NULL,PINC=NULL,PARAMVALS=NULL,
```

```
TIMEPOINTS=NULL,TIMEPOINTSCALE=NULL)

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

oat_graphATestsForSampleSize(FILEPATH,PARAMETERS,MEASURES,
ATESTSIGLEVEL,ATESTRESULTFILENAME,BASELINE,
PMIN=NULL,PMAX=NULL,PINC=NULL,PARAMVALS=NULL,
TIMEPOINTS=NULL,TIMEPOINTSCALE=NULL)

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

oat_countResponsesOfDesiredValue(FILEPATH,PARAMETERS,
RESULTFILENAME,OUTPUTCOLSTART,OUTPUTCOLEND,
PARAMETER,NUMRUNSPERSAMPLE,MEASURE,DESIREDRESULT,
OUTPUTFILENAME,BASELINE,PMIN=NULL,PMAX=NULL,
PINC=NULL,PARAMVALS=NULL,TIMEPOINTS=NULL,
TIMEPOINTSCALE=NULL)
```

Arguments

FILEPATH	Directory where either the simulation runs or single CSV file result can be found
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
PMIN	Array containing the minimum value that should be used for each parameter. Sets a lower bound on sampling space
PMAX	Array containing the maximum value that should be used for each parameter. Sets an upper bound on sampling space
PINC	Array containing the increment value that should be applied for each parameter. For example, a parameter could have a minimum value of 10, and maximum value of 100, and be incremented by 10
PARAMVALS	Array containing a list of strings for each parameter, each string containing comma separated values that should be assigned to that parameter. Thus sampling can be performed for specific values for each parameter, rather than a uniform incremented value. This replaces the PMIN, PMAX, and PINC where this method is used
NUMRUNSPERSAMPLE	The number of runs performed for each parameter subset. This figure is generated through Aleatory Analysis
MEASURES	Array containing the names of the output measures which are used to analyse the simulation

RESULTFILENAME Name of the simulation results file (e.g. "trackedCells_Close.csv"). In the current version, XML and CSV files can be processed. Only required if running the first method (to process results directly). If performing this analysis over multiple timepoints, it is assumed that the timepoint follows the file name, e.g. trackedCells_Close_12.csv.

ALTERNATIVEFILENAME
In some cases, it may be relevant to read from a further results file if he initial file contains no results. This filename is set here. Only required if running the first method (to process results directly).

OUTPUTCOLSTART Column number in the simulation results file where output begins - saves (a) reading in unnecessary data, and (b) errors where the first column is a label, and therefore could contain duplicates. Only required if running the first method (to process results directly)

OUTPUTCOLEND Column number in the simulation results file where the last output measure is. Only required if running the first method.

CSV_FILE_NAME If oat_processParamSubsets is used, this analyses the results of replicate simulation runs and creates a file containing the median value of each measure for every run. This specifies what that file should be called (e.g. Medians.csv). If the CSV file is provided, this should contain the name of the provided file.

BASELINE Array containing the values assigned to each of these parameters in the calibrated baseline

ATESTRESULTFILENAME
File name of the ATests result summary file created by oat_analyseAllParams. For one timepoint, this could be ATests.csv

ATESTSIGLEVEL The A-Test determines if there is a large difference between two sets if the result is greater than 0.21 either side of the 0.5 line. Should this not be suitable, this can be changed here

MEASURE_SCALE An array containing the measure used for each of the output measures (i.e. microns, microns/min). Used to label graphs

TIMEPOINTS Implemented so this method can be used when analysing multiple simulation timepoints. If only analysing one timepoint, this should be set to NULL. If not, this should be an array of timepoints, e.g. c(12,36,48,60)

TIMEPOINTSCALE Implemented so this method can be used when analysing multiple simulation timepoints. Sets the scale of the timepoints being analysed, e.g. "Hours"

PARAMETER Parameter of interest when counting simulation responses that meet a specific requirement

MEASURE The measure of interest when counting simulation responses that meet a specific requirement

DESIREDRESULT The specific requirement to match when counting simulation responses

OUTPUTFILENAME CSV file name to contain the counts where simulation responses meet a specific requirement

References

This technique is described by Read et al (2011) in their paper: Techniques for Grounding Agent-Based Simulations in the Real Domain: a case study in Experimental Autoimmune Encephalomyelitis"

Examples

```
# THE CODE IN THIS EXAMPLE IS THE SAME AS THAT USED IN THE TUTORIAL, AND
# THUS YOU NEED TO DOWNLOAD THE TUTORIAL DATA SET AND SET FILEPATH
# CORRECTLY TO RUN THIS

##---- Firstly, declare the parameters required for the four functions ----
### FIRST DECLARE THE PARAMETERS REQUIRED FOR THIS ANALYSIS:
# A: THE ROOT FILE PATH. EITHER WHERE THE SIMULATION RESPONSES ARE, OR
# WHERE A CSV FILE SUMMARISING THESE RESPONSES IS LOCATED
FILEPATH<-"/home/kieran/Downloads/OAT/RANGE/"
# B: EITHER (i) THE NAME OF THE FILE CONTAINING ALL THE SIMULATION OUTPUT
# OR (ii) THE NAME OF THE FILE THAT WILL BE CREATED TO SUMMARISE SIMULATION
# RESPONSES
CSV_FILE_NAME<-"OAT_Medians.csv"
# C: THE SIMULATION PARAMETERS BEING EXPLORED
PARAMETERS<-c("chemoLowerLinearAdjust","chemoUpperLinearAdjust")
# E: PARAMETER VALUE INFORMATION
# YOU CAN SPECIFY THIS IN TWO WAYS: (i) THE MINIMUM AND MAXIMUM OF EACH
# PARAMETER, AND THE INCREMENT OVER WHICH THE SAMPLING WAS INCREASED
# (ii) A STRING LIST OF VALUES THAT PARAMETER WAS ASSIGNED IN SIMULATION
# EXAMPLE OF (i):
PMIN<-c(0.015,0.10)
PMAX<-c(0.08,0.50)
PINC<-c(0.005,0.05)
PARAMVALS<-NULL
# EXAMPLE OF (ii)
#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
# F: BASELINE VALUES FOR ALL PARAMETERS (CALIBRATION VALUES)
BASELINE<-c(0.04,0.2)
# G: SIMULATION OUTPUT MEASURES
MEASURES<-c("Velocity","Displacement")
# H: NAME TO GIVE THE CSV FILE CONTAINING THE ATEST RESULTS FOR
# ROBUSTNESS ANALYSIS
ATESTRESULTFILENAME<-"OAT_ATestScores.csv"
# I: A-TEST RESULT VALUE EITHER SIDE OF 0.5 AT WHICH THE DIFFERENCE BETWEEN
#TWO SETS OF RESULTS IS SIGNIFICANT
ATESTSIGLEVEL<-0.21
# J: IF USING SPARTAN TO PROCESS THE RESULTS FROM REPLICATE RUNS, IN THE SET
# FOLDER STRUCTURE DESCRIBED IN THE TUTORIAL PAPER, ENTER THE NUMBER OF
# REPLICATE RUNS PERFORMED FOR EACH PARAMETER CONDITION
NUMRUNSPERSAMPLE<-300
# K: AGAIN IF PROCESSING INDIVIDUAL RUN RESULTS, NOT A FILE SUMMARISING SIM
# BEHAVIOUR, ENTER THE SIMULATION RESULT FILE NAME
RESULTFILENAME<-"trackedCells_Close.csv"
# L: USEFUL IN CASES WHERE TWO RESULT FILES MAY EXIST, AND WHERE A SECOND IS
# PROCESSED
# IF THERE ARE NO RESPONSES IN THE FIRST
ALTERNATIVEFILENAME<-NULL
# M: USE THIS IF SIM RESULTS ARE IN CSV FORMAT (ALSO ACCEPTS XML)
# THE COLUMN WITHIN THE CSV RESULTS FILE WHERE OUTPUT RESPONSES START. USEFUL
```

```
# AS IT RESTRICTS WHAT IS READ INTO R, GETTING AROUND POTENTIAL ERRORS WHERE
# THE FIRST COLUMN DUPLICATES
OUTPUTCOLSTART<-10
# N: COLUMN WHERE OUTPUT RESPONSES END
OUTPUTCOLEND<-11
# O: USED WHERE A SIMULATION IS BEING ANALYSED AT MULTIPLE TIMEPOINTS. THIS IS
# AN ADDENDUM TO OUR R JOURNAL ARTICLE, AND INSTRUCTIONS TO DO THIS CAN BE
# FOUND ON OUR WEBSITE
TIMEPOINTS<-NULL
TIMEPOINTSCALE<-NULL
# EXAMPLE OF TIMEPOINT STRUCTURE
#TIMEPOINTS<-c(12,36,48,60)
#TIMEPOINTSCALE<-"Hours"
# NOW RUN THE METHODS

## Not run:
# DONTRUN IS SET SO THIS IS NOT EXECUTED WHEN PACKAGE IS COMPILED - BUT THIS
# HAS BEEN TESTED WITH THE TUTORIAL DATA

# A - FOR STOCHASTIC SIMULATIONS IN THE SET FOLDER STRUCTURE, GENERATE THE
# MEDIAN SET FOR EACH SET OF RUNS FOR THE PARAMETER VALUE
oat_processParamSubsets(FILEPATH,PARAMETERS,NUMRUNSPERSAMPLE,MEASURES,
RESULTFILENAME,ALTERNATIVEFILENAME,OUTPUTCOLSTART,OUTPUTCOLEND,
CSV_FILE_NAME,BASELINE,PMIN,PMAX,PINC,PARAMVALS,TIMEPOINTS,TIMEPOINTSCALE)


# B - RUN THE ATEST FOR EACH PARAMETER VALUE, AND EACH PARAMETER
# USES EITHER THE CSV FILE GENERATED IN THE METHOD ABOVE OR ONE THAT IS
# SUPPLIED
oat_csv_result_file_analysis(FILEPATH,CSV_FILE_NAME,PARAMETERS,BASELINE,
MEASURES,ATESTRESULTFILENAME,PMIN,PMAX,PINC,PARAMVALS,
TIMEPOINTS,TIMEPOINTSCALE)

# C - GRAPH THE RESULTS FOR ALL MEASURES FOR EACH PARAMETER
oat_graphATestsForSampleSize(FILEPATH,PARAMETERS,MEASURES,ATESTSIGLEVEL,
ATESTRESULTFILENAME,BASELINE,PMIN,PMAX,PINC,PARAMVALS,TIMEPOINTS,
TIMEPOINTSCALE)

# D - GRAPH THE DISTRIBUTION OF THE RESULTS FOR THIS MEASURE, FOR
# THIS PARAMETER
oat_plotResultDistribution(FILEPATH,PARAMETERS,MEASURES,MEASURE_SCALE,
CSV_FILE_NAME,BASELINE,PMIN,PMAX,PINC,PARAMVALS,TIMEPOINTS,
TIMEPOINTSCALE)

# E - COUNT THE NUMBER OF TIMES A PARAMETER (SUCH AS vcamSlope
# PRODUCES AN OUTPUT RESPONSE OF 0 FOR AREA
oat_countResponsesOfDesiredValue(FILEPATH,PARAMETERS,RESULTFILENAME,OUTPUTCOLSTART,
OUTPUTCOLEND,"vcamSlope",NUMRUNSPERSAMPLE,"Area",0,OUTPUTFILENAME,
BASELINE,PMIN,PMAX,PINC,PARAMVALS,TIMEPOINTS,TIMEPOINTSCALE)

## End(Not run)
```

`Technique 3: Latin-Hypercube: Generate Parameter Value Sets`
LHC

Description

Though Technique 2 does elucidate the effects of perturbations of one parameter, it cannot show any non-linear effects which occur when two or more are adjusted simultaneously. A Global Sensitivity Analysis technique is needed to identify such effects, and to give an indication of the parameters which have the greatest influence on the simulation output. This technique uses the method described by Read et al in their paper reference below, which uses a latin-hypercube design as described by Saltelli and others to sample the parameter space. Ranges are set for each parameter, and all parameter values perturbed concurrently. This method creates the parameter value sets with which simulations should be run. This is output as a CSV file, an example of which can be seen in the data directory of this package (LHC_Parameters_for_Runs.csv). For each set of parameters, the simulation should be run for the number of times identified in Aleatory Analysis (Technique 1). Once this has been completed, the set of remaining methods within Technique 3 of this package can be used to analyse the results. Note: To run this, you will require the lhs library.

Usage

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

Arguments

`FILEPATH`	Directory where the parameter samples should be output to
`PARAMETERS`	Array containing the names of the parameters of which parameter samples will be generated
`NUMSAMPLES`	The number of parameter subsets to generate
`PMIN`	Array containing the minimum value that should be used for each parameter. Sets a lower bound on sampling space
`PMAX`	Array containing the maximum value that should be used for each parameter. Sets an upper bound on sampling space
`ALGORITHM`	Choice of algorithm to use to generate the hypercube. Can be set to either 'normal' or 'optimum'. Beware optimum can take a long time to generate an optimised parameter set (more than 24 hours in some circumstances)

References

This technique is described by Read et al (2011) in their paper: Techniques for Grounding Agent-Based Simulations in the Real Domain: a case study in Experimental Autoimmune Encephalomyelitis", and also in the Saltelli et al book: "Senstivity Analysis"

Examples

```
##-- Firstly, declare the parameters required for the function --
# A - THE FILEPATH WHERE RESULTS SHOULD BE OUTPUT TO
FILEPATH<-"/media/FreeAgent/package_Test_Data/LHC/Sampling/"
# B - THE NAMES OF THE PARAMETERS BEING SAMPLED
PARAMETERS <- c("thresholdBindProbability","chemoThreshold",
"chemoUpperLinearAdjust","chemoLowerLinearAdjust",
"maxVCAMeffectProbabilityCutoff","vcamSlope")
# C - Number of of samples to generate from the hypercube
NUMSAMPLES <- 65
# D - THE MINIMUM VALUE OF EACH PARAMETER
PMIN<-c(0,0.10,0.10,0.015,0.1,0.25)
# E - THE MAXIMUM VALUE OF EACH PARAMETER
PMAX<-c(100,0.9,0.50,0.08,1.0,5.0)
# F - THE ALGORITHM TO USE TO GENERATE THE HYPERCUBE
ALGORITHM <- "normal"

## Not run:
##---- NOW RUN THE METHOD ----
# Needs the lhs library
library(lhs)
lhc_generate_lhc_sample(FILEPATH,PARAMETERS,NUMSAMPLES,PMIN,PMAX,ALGORITHM)

## End(Not run)
```

Technique 3: Latin-Hypercube: Perform Analysis of Results

LHC: Perform Analysis of Results

Description

Though Technique 2 does elucidate the effects of perturbations of one parameter, it cannot show any non-linear effects which occur when two or more are adjusted simultaneously. Thus we have included this method described by Read et al, Saltelli et al, and others (Reference's below). The technique above provides the means to sample the parameter space using a latin hypercube approach, and this technique allows for the analysis of simulation results generated using these parameter sets. Where a simulation is stochastic (such as agent-based simulations), the simulation should be run n number of times (n can be established using Technique 1 - Aleatory Analysis), after which the first method of this technique will generate median values for each output measure over the n runs. A summary file is then created, containing all parameter value sets that were created in sampling alongside these calculated median results. An example of such a file can be seen in the data directory of this package (LHCSummary.csv). With this summary complete, each parameter being analysed is processed in turn, to determine if there are any correlations between the value of this parameter and simulation output result, although all other parameter values are being perturbed. Partial Rank Correlation Coefficients are generated for each output measure, for each parameter. These give a statistical indication of any correlations that have now become apparent. An example of this can again be seen in the data directory (corCoeffs.csv). To ease identification of such effects,

graphs are produced for each parameter, showing the parameter value against the simulation result (output measure).

Note 1: From Spartan 2.0, you can specify your simulation data in two ways:
A - Set folder structure (as in previous versions of Spartan): This is shown in figure LHC_Folder_Struc.png within the extdata folder of this package, and described in detail in the tutorial. Using this structure, the parameter FILEPATH should point to a directory that contains a number of folders, one for each of the parameter samples generated by the hypercube. These will in turn contain folders for all simulations run under those parameter conditions.
B - **B** - Single CSV file Input. From Spartan 2.0, you can specify all your results in a single CSV file. An example of this file can be found in the extdata folder of the package, named LHC_AllResults.csv. Each row of this CSV file should consist of the parameters that were generated by the hypercube and the simulation responses these generate. There may be more than one row of responses per set of simulation responses, if performing replicate runs.
Note 2: From Spartan 2.0, performing this analysis at multiple timepoints is now performed using the same method calls below. There are no additional method calls for timepoint analysis.

This technique consists of four methods:
lhc_process_sample_run_subsets: Only to be applied in cases where simulation responses are supplied in the folder structure (as in all previous versions of Spartan), useful for cases where the simulation is agent-based. Takes each parameter value set generated by the hypercube in turn, and analyses the replicate simulation results for that set. Produces a CSV file containing the parameters of the run and the median of each simulation response for each run. In cases where, for example, 300 runs have been performed for a parameter set, this file will contain 300 rows for that set, each accompanied by the median of each simulation response for that run. This file will be named as specified by parameter LHC_ALL_SIM_RESULTS_FILE. This method can be performed for a number of simulation timepoints, producing CSV files for each timepoint taken.
lhc_generateLHCSummary: Processes either the CSV file generated by the method above or one that has been supplied, going through each method and generating a file that summarises simulation responses under each parameter set. This CSV file, named as specified by parameter LHCSUMMARYFILENAME, will contain one row for each parameter set, accompanied by the median of all the responses contained in the LHC_ALL_SIM_RESULTS_FILE. An example of this file can be found in the extdata folder of this package, named LHC_Summary.csv. This method can also be performed for a number of simulation timepoints.
lhc_generatePRCoEffs: For each parameter, and each simulation output measure, calculates the Partial Rank Correlation Coefficient between the parameter value and the simulation results, giving a statistical measurement of any effect that is present. This is output to a CSV file, an example of which can be seen in the data folder of this package (EgSet_LHC_corCoeffs.csv). Can again be performed for a number of timepoints if required.
lhc_graphMeasuresForParameterChange: Produces a graph for each parameter, and each output measure, showing the simulation output achieved when that parameter was assigned that value. Eases identification of any non-linear effects. Two examples can be seen in the extdata folder (LHC_maxVCAMeffectProbabilityCutoff_Velocity.pdf and LHC_chemoThreshold_Velocity.pdf). Can again be performed for a number of timepoints if required.

Usage

```
lhc_process_sample_run_subsets(FILEPATH,
SPARTAN_PARAMETER_FILE,PARAMETERS,NUMSAMPLES,
```

```
NUMRUNSPERSAMPLE,MEASURES,RESULTFILENAME,
ALTERNATIVEFILENAME,OUTPUTCOLSTART,OUTPUTCOLEND,
LHC_ALL_SIM_RESULTS_FILE,TIMEPOINTS=NULL,
TIMEPOINTSCALE=NULL)

lhc_generateLHCSummary(FILEPATH,PARAMETERS,MEASURES,
LHC_ALL_SIM_RESULTS_FILE,LHCSUMMARYFILENAME,
SPARTAN_PARAMETER_FILE,TIMEPOINTS=NULL,
TIMEPOINTSCALE=NULL)

lhc_generatePRCoEffs(FILEPATH,PARAMETERS,MEASURES,
LHCSUMMARYFILENAME,CORCOEFFSOUTPUTFILE,
TIMEPOINTS=NULL,TIMEPOINTSCALE=NULL)

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

Arguments

FILEPATH	Directory where the simulation runs of single CSV file can be found
NUMSAMPLES	The number of parameter subsets that were generated in the LHC design. Only required if analysing results provided within Folder structure setup.
NUMRUNSPERSAMPLE	The number of runs performed for each parameter subset. This figure is generated through Aleatory Analysis. Only required if analysing results provided within Folder structure setup.
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
RESULTFILENAME	Name of the simulation results file (e.g. "trackedCells_Close.csv"). In the current version, XML and CSV files can be processed. Only required if running the first method (to process results directly). If performing this analysis over multiple timepoints, it is assumed that the timepoint follows the file name, e.g. trackedCells_Close_12.csv.
ALTERNATIVEFILENAME	In some cases, it may be relevant to read from a further results file if the initial file contains no results. This filename is set here. In the current version, XML and CSV files can be processed. Only required if running the first method (to process results directly)
OUTPUTCOLSTART	Column number in the simulation results file where output begins - saves (a) reading in unnecessary data, and (b) errors where the first column is a label, and therefore could contain duplicates. Only required if running the first method (to process results directly)
OUTPUTCOLEND	Column number in the simulation results file where the last output measure is. Only required if running the first method.

`SPARTAN_PARAMETER_FILE`
: Location of the file output by the latin-hypercube sampling method. Note if providing a single CSV file with parameter/response pairings, you do not need to provide this file, and can thus enter this parameter as NULL.

`LHC_ALL_SIM_RESULTS_FILE`
: If lhc_process_sample_run_subsets is used (i.e. results processed by folder structure), this will contain the output of that method. If specifying responses using a single CSV file, this will contain the name of that file (which should be in the FILEPATH folder).

`PARAMETERS`
: Array containing the names of the parameters of which parameter samples will be generated

`LHCSUMMARYFILENAME`
: Name of the LHC Summary file to be generated by lhc_generateLHCSummary. Contains each parameter set alongside the result gained when the simulation was run under that criteria. Example - LHC_Summary

`CORCOEFFSOUTPUTFILE`
: Name of the file to be generated by lhc_generatePRCoEffs. Contains the Partial Rank Correlation Coefficients for each parameter. Example - CorCoEffs

`MEASURE_SCALE`
: An array containing the measure used for each of the output measures (i.e. microns, microns/min). Used to label graphs

`TIMEPOINTS`
: Implemented so this method can be used when analysing multiple simulation timepoints. If only analysing one timepoint, this should be set to NULL. If not, this should be an array of timepoints, e.g. c(12,36,48,60)

`TIMEPOINTSCALE`
: Implemented so this method can be used when analysing multiple simulation timepoints. Sets the scale of the timepoints being analysed, e.g. "Hours"

References

This technique is described by Read et al (2011) in their paper: Techniques for Grounding Agent-Based Simulations in the Real Domain: a case study in Experimental Autoimmune Encephalomyelitis", and also in the Saltelli et al book: "Senstivity Analysis". Code to perform Partial Rank Correlation Coeffient has been downloaded from http://www.yilab.gatech.edu/pcor.R

Examples

```
# THE CODE IN THIS EXAMPLE IS THE SAME AS THAT USED IN THE TUTORIAL, AND
# THUS YOU NEED TO DOWNLOAD THE TUTORIAL DATA SET AND SET FILEPATH
# CORRECTLY TO RUN THIS

##--Firstly, declare the parameters required for the 4 functions--
# Folder containing the simulation results or single CSV result file
FILEPATH<-"/home/kieran/Downloads/LHC_Spartan2/"
# Array of the parameters to be analysed
PARAMETERS<-c("thresholdBindProbability","chemoThreshold",
"chemoUpperLinearAdjust","chemoLowerLinearAdjust",
"maxVCAMeffectProbabilityCutoff","vcamSlope")
# The simulation output measures being examined
MEASURES<-c("Velocity","Displacement")
# What each measure represents. Used in graphing results
```

```
MEASURE_SCALE<-c("microns/min","microns")
# The number of parameter value sets created in latin-hypercube
# sampling
NUMSAMPLES <- 500
# Number of runs performed for each parameter value set
NUMRUNSPERSAMPLE<-500
# The output file containing the simulation results from that
# simulation run
RESULTFILENAME<-"trackedCells_Close.csv"
# Not used in this case, but this is useful in cases where
# two result files may exist (for example if tracking cells close to
# an area, 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. Note no file extension if used.
ALTERNATIVEFILENAME<-NULL
# Use this if simulation results are in CSV format.
# The column within the csv results file where the results start.
# This is useful as it restricts what is read in to R, getting round
# potential errors where the first column contains an agent label
# (as R does not read in CSV files where the first column contains
# duplicates)
OUTPUTCOLSTART<-10
# Use this if simulation results are in CSV format.
# Last column of the output measure results
OUTPUTCOLEND<-11
# 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.
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
# method of this tutorial.) 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"
# 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.
LHCSUMMARYFILENAME<-"LHC_Summary.csv"
# File name to give to the file showing the Partial Rank Correlation
# Coefficients for each parameter.
CORCOEFFSOUTPUTFILE<-"EgSet_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:
#TIMEPOINTS<-c(12,36,48,60); TIMEPOINTSCALE<-"Hours"

## Not run:
# DONTRUN IS SET SO THIS IS NOT EXECUTED WHEN PACKAGE IS COMPILED - BUT THIS
# HAS BEEN TESTED WITH THE TUTORIAL DATA

##--- NOW RUN THE FOUR METHODS IN THIS ORDER ----
```

```
# A - FOR STOCHASTIC SIMS IN SET FOLDER STRUCTURE, GENERATE
# THE MEDIANS FOR EACH SET OF PARAMETER VALUES
# GENERATED BY THE HYPERCUBE

lhc_process_sample_run_subsets(FILEPATH,
LHC_PARAM_CSV_LOCATION,PARAMETERSNUMSAMPLES,
NUMRUNSPERSAMPLE,MEASURES,RESULTFILENAME,
ALTERNATIVEFILENAME,OUTPUTCOLSTART,OUTPUTCOLEND,
LHC_ALL_SIM_RESULTS_FILE,TIMEPOINTS=NULL,
TIMEPOINTSCALE=NULL)

# B - GENERATE THE SUMMARY FILE SHOWING THE PARAMETERS USED AND
# MEDIAN RESULTS FOR THE MEASURES OVER THE n RUNS
lhc_generateLHCSummary(FILEPATH,PARAMETERS,MEASURES,
LHC_ALL_SIM_RESULTS_FILE,LHCSUMMARYFILENAME,
LHC_PARAM_CSV_LOCATION=NULL,TIMEPOINTS=NULL,
TIMEPOINTSCALE=NULL)

# C- CALCULATE THE PARTIAL RANK CORRELATION COEFFICIENTS
lhc_generatePRCoEffs(FILEPATH,PARAMETERS,MEASURES,
LHCSUMMARYFILENAME,CORCOEFFSOUTPUTFILE,
TIMEPOINTS=NULL,TIMEPOINTSCALE=NULL)

# D - GRAPH THE RESULTS
lhc_graphMeasuresForParameterChange(FILEPATH,PARAMETERS,
MEASURES,MEASURE_SCALE,CORCOEFFSOUTPUTFILE,
LHCSUMMARYFILENAME,TIMEPOINTS=NULL,
TIMEPOINTSCALE=NULL)


## End(Not run)
```

`Technique 4: eFAST - Generate Parameter Value Sets`
eFAST

Description

This technique analyses simulation results generated through parametering using the eFAST approach (extended Fourier Amplitude Sampling Test, Saltelli et al, reference below). 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). This method performs sampling of parameter space using the eFAST approach. In the data folder of this package, there are seven example files, produced for 7 parameters over one resample curve. These filenames all begin Curve1_[ParameterName].

Usage

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

Arguments

FILEPATH	Directory where the parameter samples should be output to
NUMCURVES	The number of 'resamples' to perform (see eFAST documentation) - recommend using at least 3
NUMSAMPLES	The number of parameter subsets to generate - should be at least 65 for eFAST
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated. For eFAST, remember to add a parameter named 'Dummy'
PMIN	Array containing the minimum value that should be used for each parameter and the dummy. Sets a lower bound on sampling space
PMAX	Array containing the maximum value that should be used for each parameter and the dummy. Sets an upper bound on sampling space

References

For detailed information on how eFAST works, see either of the following: (a) Marino et al (2008): "A methodology for performing global uncertainty and sensitivity analysis in systems biology", (b) Saltelli et al (2000): "Sensitivity Analysis". MATLAB code is also available via an associated site stated in (a)

Examples

```
# THE CODE IN THIS EXAMPLE IS THE SAME AS THAT USED IN THE TUTORIAL, AND
# THUS YOU NEED TO DOWNLOAD THE TUTORIAL DATA SET AND SET FILEPATH
# CORRECTLY TO RUN THIS

##----Firstly, declare the parameters required for the function--
# A - FILEPATH OF WHERE THE PARAMETER DESIGN SHOULD BE OUTPUT TO
FILEPATH<-"/media/FreeAgent/package_Test_Data/eFAST/Sampling/"
# B - NUMBER OF RESAMPLES (SEARCH CURVES) TO EMPLOY
NUMCURVES <- 3
# C: NUMBER OF TIMES TO SAMPLE EACH CURVE
# (i.e. NUMBER OF PARAMETER SUBSETS)
```

```
NUMSAMPLES <- 65
# D: PARAMETERS, IN ORDER, TRY TO AVOID SPACES, AND REMEMBER DUMMY
PARAMETERS<-c("thresholdBindProbability","chemoThreshold",
"chemoUpperLinearAdjust","chemoLowerLinearAdjust",
"maxVCAMeffectProbabilityCutoff","vcamSlope","Dummy")
# E: ARRAY OF THE MINIMUM VALUES OF EACH PARAMETER (AND DUMMY)
PMIN<-c(0,0.1,0.10,0.015,0.1,0.25,1)
# F: ARRAY OF THE MAXIMUM VALUES OF EACH PARAMETER (AND DUMMY)
PMAX<-c(100,0.9,0.50,0.08,1.0,5.0,10)

## Not run:
# DONTRUN IS SET SO THIS IS NOT EXECUTED WHEN PACKAGE IS COMPILED - BUT THIS
# HAS BEEN TESTED WITH THE TUTORIAL DATA

##---- NOW RUN THE METHOD ----
efast_generate_sample(FILEPATH,NUMCURVES,NUMSAMPLES,PARAMETERS,
PMIN,PMAX)

## End(Not run)
```

`Technique 4: eFAST - Perform Analysis of Results`

eFAST: Perform Analysis of Results

Description

This technique analyses simulation results generated through parametering using the eFAST approach (extended Fourier Amplitude Sampling Test, Saltelli et al, reference below). 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 sample, and repeated for stochastic simulations (a number of repeats that has become apparent through analysis

of aleatory uncertainty, or use of Technique 1 within the spartan package). 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. The spartan package includes methods to both create parameter value samples using fourier frequency sampling, and to analyse the simulation results. This method does the latter.

Note 1: From Spartan 2.0, you can specify your simulation data in two ways:
A - Set folder structure (as in previous versions of Spartan): This is shown in figure eFAST_Folder_Struc.png within the extdata folder of this package, and described in detail in the tutorial. Using this structure, the parameter FILEPATH should point to a directory that contains a number of folders, one for each resample curve employed. Inside each of these folders will be a folder for each parameter, in turn holding a folder for each parameter sample set generated for that parameter. These in turn hold folders for each run of a simulation under those conditions.
B - **B** - CSV file Input. From Spartan 2.0, you can specify all your results in a CSV file. For eFAST, this is more complex than the other techniques: the user needs to provide one CSV file per curve/parameter pairing. Thus if you had 3 resample curves and 7 parameters, you will need to provide 21 files. Each of these files will contain the parameters under which the simulation was run, and the simulation measures generated by those parameters. Where a simulation produces multiples of each measure (for example a number of cells), this should be the median of those responses. This file may contain muliple simulation responses per parameter set, where the simulation has been run a number of times.
Note 2: From Spartan 2.0, performing this analysis at multiple timepoints is now performed using the same method calls below. There are no additional method calls for timepoint analysis.

There are three methods to this process:
efast_generate_medians_for_all_parameter_subsets: Only to be applied in cases where simulation responses are supplied in the folder structure (as in all previous versions of Spartan), useful for cases where the simulation is agent-based. Iterates through the folder structure analysing the result of each replicate run under the same parameter conditions, creating a CSV file for each curve/parameter pair. This will hold the parameters of the run and the median of each simulation response for that run. As stated earlier, more than one run result can exist in this file. Where a simulation is being analysed for multiple timepoints, this will iterate through the results at all timepoints, creating curve/parameter pair CSV files for all specified timepoints.
efast_get_overall_medians: This 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 parameter value sets generated. Here's an example. We examine resampling curve 1, and firstly examine parameter 1. For this parameter of interest, a number of different parameter value sets were generated from the frequency curves (lets say 65), thus we have 65 different sets of simulation results. The previous method produced a summary showing the median of each output measure for each run. Now, this method calculates the median of these medians, for each output measure, and stores these in the summary. 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 with the file name Curve[Number]_Results_Summary in the directory specified in FILEPATH. Again this can be done recursively for a number of timepoints if required.
efast_run_Analysis: Produces a file summarising the analysis; partitioning the variance between parameters and providing relevant statistics. These include, for each parameter of interest, first-

order sensitivity index (Si), total-order sensitivity index (STi), complementary parameters sensitivity index (SCi), and relevant p-values and error bar data calculated using a two-sample t-test and standard error respectively. For a more detailed examination of this analysis, see the Marino paper or Saltelli book references, or the tutorial on the package website. An example of the output file generated can be seen in the data folder of this package (eFAST_Analysis.csv) For ease of representation, the method also produces a graph showing this data for each simulation output measure. Two examples can be seen in the extdata folder of this package (eFAST_Displacement.pdf and eFAST_Velocity.pdf). Again, these graphs and summaries can be produced for multiple timepoints.

Usage

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

efast_get_overall_medians(FILEPATH,NUMCURVES,PARAMETERS,
NUMSAMPLES,MEASURES,TIMEPOINTS=NULL,
TIMEPOINTSCALE=NULL)

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

Arguments

FILEPATH	Directory where the simulation runs can be found, in folders or in CSV file format
NUMCURVES	The number of 'resamples' to perform (see eFAST documentation) - recommend using at least 3
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
NUMSAMPLES	The number of parameter subsets that were generated in the eFAST design
NUMRUNSPERSAMPLE	The number of runs performed for each parameter subset. This figure can be generated through Aleatory Analysis
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
OUTPUTCOLSTART	Column number in the simulation results file where output begins - saves (a) reading in unnecessary data, and (b) errors where the first column is a label, and therefore could contain duplicates. Only required if running the first method (to process results directly)
OUTPUTCOLEND	Column number in the simulation results file where the last output measure is. Only required if running the first method.

RESULTFILENAME Name of the simulation results file (e.g. "trackedCells_Close.csv"). In the current version, XML and CSV files can be processed. Only required if running the first method (to process results directly). If performing this analysis over multiple timepoints, it is assumed that the timepoint follows the file name, e.g. trackedCells_Close_12.csv.

ALTERNATIVEFILENAME
In some cases, it may be relevant to read from a further results file if the initial file contains no results. This filename is set here. In the current version, XML and CSV files can be processed. Only required if running the first method (to process results directly)

OUTPUTMEASURES_TO_TTEST
Which measures in the range should be tested to see if the result is statistically significant. To do all, and if there were 3 measures, this would be set to 1:3

EFASTRESULTFILENAME
File name under which the full eFAST analysis should be stored. This will contain the partitioning of variance for each parameter. Example: eFAST_Analysis

TTEST_CONF_INT The level of significance to use for the T-Test

GRAPH_FLAG Whether graphs should be produced summarising the output - should be TRUE or FALSE

TIMEPOINTS Implemented so this method can be used when analysing multiple simulation timepoints. If only analysing one timepoint, this should be set to NULL. If not, this should be an array of timepoints, e.g. c(12,36,48,60)

TIMEPOINTSCALE Implemented so this method can be used when analysing multiple simulation timepoints. Sets the scale of the timepoints being analysed, e.g. "Hours"

References

For detailed information on how eFAST works, see either of the following: (a) Marino et al (2008): "A methodology for performing global uncertainty and sensitivity analysis in systems biology", (b) Saltelli et al (2000): "Sensitivity Analysis". MATLAB code is also available via an associated site stated in (a)

Examples

```
# THE CODE IN THIS EXAMPLE IS THE SAME AS THAT USED IN THE TUTORIAL, AND
# THUS YOU NEED TO DOWNLOAD THE TUTORIAL DATA SET AND SET FILEPATH
# CORRECTLY TO RUN THIS

##---Firstly,declare the parameters required for the functions--
# Folder containing the simulation results, or CSV files
FILEPATH<-"/home/user/Downloads/eFAST/"
# Number of resample curves employed when the parameter space was
# sampled
NUMCURVES<-3
# Array of the parameters to be analysed
PARAMETERS <- c("BindProbability","ChemoThreshold",
"ChemoUpperLinearAdjust","ChemoLowerLinearAdjust",
"VCAMProbabilityThreshold","VCAMSlope","Dummy")
```

```
# The number of parameter value sets created in latin-hypercube
# sampling
NUMSAMPLES<-65
# Number of runs performed for each parameter value set
NUMRUNSPERSAMPLE<-300
# The simulation output measures being examined
MEASURES<-c("Velocity","Displacement")
# The output file containing the simulation results from that
# simulation run
RESULTFILENAME<-"trackedCells_Close.csv"
# Not used in this case, but this is useful in cases where two
# result files may exist (for example if tracking cells close
# to an area, 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.
ALTERNATIVEFILENAME<-NULL
# Used with CSV result file formats
# The column within the csv results file where the results start.
# This is useful as it restricts what is read in to R, getting round
# potential errors where the first column contains an agent label
# (as R does not read in CSV files where the first column contains
# duplicates)
OUTPUTCOLSTART<-10
# Used with CSV result file formats
# Last column of the output measure results
OUTPUTCOLEND<-11
# Name of the final result file for this analysis, showing the
# partitioning of the variance between input parameters
EFASTRESULTFILENAME<-"eFAST_Analysis.csv"
# Which of the output measures to T-Test for significance (if not all)
OUTPUTMEASURES_TO_TTEST<-1:2
# T-Test confidence level
TTEST_CONF_INT<-0.95
# Boolean to note whether summary graphs should be produced
GRAPH_FLAG<-TRUE
# 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:
TIMEPOINTS<-c(12,36,48,60); TIMEPOINTSCALE<-"Hours"

## Not run:
# DONTRUN IS SET SO THIS IS NOT EXECUTED WHEN PACKAGE IS COMPILED - BUT THIS
# HAS BEEN TESTED WITH THE TUTORIAL DATA

library(spartan)
# Import the graphing package
library(gplots)

##--- NOW RUN THE FOUR METHODS IN THIS ORDER ----
# FIRSTLY, WHERE MULTIPLE RUNS ARE PERFORMED,
# MEDIAN DISTRIBUTIONS NEED TO BE GAINED FOR EVERY RUN
```

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


# NOW NEED TO CREATE THE OUTPUT FILE THAT THE EFAST ANALYSIS SCRIPTS
# USE - A FILE SHOWING THE OVERALL MEDIAN RESULTS FOR EACH THE RUNS
# PERFORMED FOR EVERY PARAMETER OF INTEREST, FOR THAT CURVE.
# ONE FILE IS CREATED PER CURVE
efast_get_overall_medians(FILEPATH,NUMCURVES,PARAMETERS,NUMSAMPLES,
MEASURES,TIMEPOINTS,TIMEPOINTSCALE)

# NOW THESE ALLCURVE.CSV FILES HAVE BEEN GENERATED, FULL ANALYSIS
# CAN BEGIN
efast_run_Analysis(FILEPATH,MEASURES,PARAMETERS,NUMCURVES,
NUMSAMPLES,OUTPUTMEASURES_TO_TTEST,TTEST_CONF_INT,
GRAPH_FLAG,EFASTRESULTFILENAME,
TIMEPOINTS,TIMEPOINTSCALE)


## End(Not run)
```

Technique 5: SPARTAN and Netlogo

Technique 5: SPARTAN and Netlogo

Description

Versions 1.0-1.2 of SPARTAN focused on performing an analysis of simulations, but such demonstrations were never paired with tools that are used by researchers to develop such simulations. Version 1.3 offered the additional functionality of linking SPARTAN with Netlogo, allowing the user to automatically generate Netlogo experiment scripts using SPARTAN, then analysing the resultant simulation response using the techniques developed above. This makes use of Netlogo's BehaviourSpace feature, which allows the researcher to perform a sweep of potential parameter values, storing the results in a CSV table. The weakness with this feature is that the researcher is left to develop tools to analyse the simulation responses. Now, SPARTAN can process this table, generating a robust sensitivity analysis of the parameters of interest. On our laboratory website (www.ycil.org.uk), we have made available a slightly modified version of the Netlogo Virus model, and a tutorial demonstrating how SPARTAN can be used to develop parameter value samples for the techniques described above, and then used to analyse the resultant Netlogo simulations.

Note: Netlogo BehaviourSpace functionality can produce two forms of output: 'Spreadsheet' or 'Table'. This technique has been designed to work with the later. This is explained fully in the tutorial on our lab website.

The reader should make themselves familiar with the techniques described above, as this is not repeated here. The only difference is in the preparation of the Netlogo data: once this is done the

methods described in Techniques 2-4 are applied.

The relevant Netlogo Parameter Robustness Methods (Technique 2) are described below. See Technique 2 for a full description of each method:
oat_generate_netlogo_behaviour_space_XML: This generates a Netlogo XML Experiment file, to be used with the BehaviourSpace feature or Headless Netlogo, which perturbs each parameter over a set value space, as described in Technique 2.
oat_process_netlogo_result: Takes the Netlogo behaviour space file and extracts the required time-point information from it, storing this in a Spartan compatible CSV file. This CSV file is then processed using the methods described in Technique 2, with A-Test scores determined for each value assigned to each parameter. Once this method has been called, the researcher should use the oat_graphATestsForSampleSize and oat_plotResultDistribution methods of Technique 2 to graph the results.

The relevant Netlogo Latin-Hypercube Sampling and Analysis methods (Technique 3) are described below. See Technique 3 for a full description of each method:
lhc_generate_lhc_sample_netlogo: This generates a specified number of simulation parameters sets using latin-hypercube sampling. These are then processed into Netlogo XML experiment files, one for each set of parameters.

lhc_process_netlogo_result: Takes each parameter value set generated by the hypercube in turn, and analyses the Netlogo simulation results. For each parameter set, there will be n simulation results. This method goes through these results, producing a file containing the median of each output measure for each of the n runs. Thus, if a Netlogo simulation was replicated 10 times, the median file will contain 10 medians for each simulation output measure. Once this has been created, the user should run the lhc_generateLHCSummary, lhc_generatePRCoEffs. and lhc_graphMeasuresForParameterChange methods of Technique 3.

The relevant eFAST methods (Technique 4) are desribed below. See Technique 4 for a full description of each method:
efast_generate_sample_netlogo: Creates a set of parameter values, over the specified value space, as described in Technique 4. Then processes each of these into a Netlogo experiment XML file, from which a simulation can be run.
efast_process_netlogo_result: Takes each parameter value set generated by eFAST in turn, and analyses the Netlogo simulation results. For each parameter set, there will be n simulation results. This method goes through these results, producing a file containing the median of each output measure for each of the n runs. Thus, if a Netlogo simulation was replicated 10 times, the median file will contain 10 medians for each simulation output measure. The user should then run the efast_get_overall_medians and efast_run_Analysis methods of Technique 4.

Note that for all three methods, it is important to run the PARAMETERS and MEASURES parameters through the method table_header_check first (see examples below), as Netlogo converts any spaces or hyphens in these names to periods. This method will take care of this for you.

Usage

```
oat_generate_netlogo_behaviour_space_XML(FILEPATH,
```

```
NETLOGO_SETUPFILE_NAME,PARAMETERS,PARAMVALS,
NETLOGO_SETUP_FUNCTION,NETLOGO_RUN_FUNCTION,MEASURES,
EXPERIMENT_REPETITIONS,RUNMETRICS_EVERYSTEP)

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

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

lhc_process_netlogo_result(FILEPATH,LHCSAMPLE_RESULTFILENAME,
SPARTAN_PARAMETER_FILE,NUMSAMPLES,MEASURES,
LHC_ALL_SIM_RESULTS_FILE,TIMESTEP)

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

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

Arguments

FILEPATH	Directory where either the parameter samples are to be stored, or the simulation runs can be found
NETLOGO_SETUPFILE_NAME	Name to give, or given to, the Netlogo XML experiment file(s) created in sampling. For more than one, a sample number is appended
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
PARAMVALS	Array containing either the parameter value (if not of interest in the analysis), or range under which this is being explored (stated as as string e.g. "[5,50,5]" for a range of 5-50, increment of 5. This will differ for each of the three techniques, so the reader should read the relevant tutorial in detail.
NETLOGO_SETUP_FUNCTION	The name of the function in Netlogo that sets up the simulation. Commonly is named setup.
NETLOGO_RUN_FUNCTION	The name of the function in Netlogo that starts the simulation. Commonly named go.
MEASURES	Array containing the names of the Netlogo output measures which are used to analyse the simulation.

EXPERIMENT_REPETITIONS
: The number of times Netlogo should repeat the experiment for each set of parameter values.

RUNMETRICS_EVERYSTEP
: Boolean stating whether Netlogo should produce output for each timestep.

NETLOGO_BEHAVIOURSPACEFILE
: The name of the file produced by Netlogo for Parameter Robustness (Technique 2). This is the result file that is analysed.

BASELINE
: Array containing the baseline, or calibrated value, of each parameter.

PMIN
: Array containing the minimum value that should be used for each parameter. Sets a lower bound on sampling space.

PMAX
: Array containing the maximum value that should be used for each parameter. Sets an upper bound on sampling space.

PINC
: Array containing the increment value that should be applied for each parameter. For example, a parameter could have a minimum value of 10, and maximum value of 100, and be incremented by 10.

RESULTFILENAME
: In Technique 2, where SPARTAN is producing files showing the modified Netlogo CSV file, the name of the file that should be produced. For the other techniques, this is the name of the result file produced by Netlogo.

TIMESTEP
: For Technique 2, the timestep of the Netlogo simulation being analysed.

ATESTRESULTSFILENAME
: File name of the ATests result summary file created by oat_netlogo_analyseAllParams.

NUMSAMPLES
: The number of parameter subsets that were generated in the LHC or eFAST design.

ALGORITHM
: Choice of algorithm to use to generate the hypercube. Can be set to either 'normal' or 'optimum'. Beware optimum can take a long time to generate an optimised parameter set (more than 24 hours in some circumstances).

SPARTAN_PARAMETER_FILE
: Location of the file output by the latin-hypercube sampling method.

LHCSAMPLE_RESULTFILENAME
: Name of the result file generated by Netlogo, for a LHC parameter sample.

LHC_ALL_SIM_RESULTS_FILE
: Name of the LHC Summary file to be generated by lhc_process_netlogo_result. Contains each parameter set alongside the result gained when the simulation was run under that criteria.

NUMCURVES
: The number of 'resamples' to perform (see eFAST documentation) - recommend using at least 3.

EFASTSAMPLE_RESULTFILENAME
: Name of the result file generated by Netlogo, for a LHC parameter sample.

Examples

```
# THE CODE IN THIS EXAMPLE IS THE SAME AS THAT USED IN THE TUTORIAL, AND
# THUS YOU NEED TO DOWNLOAD THE TUTORIAL DATA SET AND SET FILEPATH
# CORRECTLY TO RUN THIS
```

```
## Not run:
## Example for Robustness Analysis
## INCLUDE THESE TWO LINES FOR NETLOGO - REMOVES ANY PERIODS PLACED IN THE
## PARAMETERS AND MEASURES IN THE SPREADSHEET
PARAMETERS<-table_header_check(PARAMETERS)
MEASURES<-table_header_check(MEASURES)

oat_process_netlogo_result(FILEPATH,NETLOGO_BEHAVIOURSPACEFILE,
PARAMETERS,BASELINE,PMIN,PMAX,PINC,MEASURES,CSV_FILE_NAME,
ATESTRESULTSFILENAME,TIMESTEP)


oat_graphATestsForSampleSize(FILEPATH,PARAMETERS,MEASURES,
ATESTSIGLEVEL,ATESTRESULTSFILENAME,BASELINE,
PMIN=PMIN,PMAX=PMAX,PINC=PINC,PARAMVALS=NULL)

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

## Example for Latin-Hypercube Analysis
lhc_process_netlogo_result(FILEPATH,LHCSAMPLE_RESULTFILENAME,
SPARTAN_PARAMETER_FILE,NUMSAMPLES,MEASURES,
LHC_ALL_SIM_RESULTS_FILE,TIMESTEP)

PARAMETERS<-table_header_check(PARAMETERS)
MEASURES<-table_header_check(MEASURES)

lhc_generateLHCSummary(FILEPATH,PARAMETERS,MEASURES,
LHC_ALL_SIM_RESULTS_FILE,LHCSUMMARYFILENAME,
SPARTAN_PARAMETER_FILE,TIMEPOINTS,TIMEPOINTSCALE)

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

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

## Example for eFAST Analysis
efast_process_netlogo_result(FILEPATH,
EFASTSAMPLE_RESULTFILENAME,PARAMETERS,NUMCURVES,
NUMSAMPLES,MEASURES,RESULTFILENAME,TIMESTEP)

PARAMETERS<-table_header_check(PARAMETERS)
MEASURES<-table_header_check(MEASURES)

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

efast_run_Analysis(FILEPATH,MEASURES,PARAMETERS,NUMCURVES,
NUMSAMPLES,OUTPUTMEASURES_TO_TTEST,TTEST_CONF_INT,
```

```
GRAPH_FLAG,EFASTRESULTFILENAME,
TIMEPOINTS,TIMEPOINTSCALE)

## End(Not run)
```

`Techniques 1-4: Internal Functions`
Internal Functions

Description

A number of internal functions (14) are used which are not directly called by the user employing this package. These are: (a) aa_getATestResults (b) aa_graphATestsForSampleSize (c) atest (d) normaliseATest (e) pcor.mat (f) pcor.rec (g) pcor.test (h) lhc_constructCoEffDataSet (i) efast_cvmethod (j) efast_graph_Results (k) efast_parameterdist (l) efast_sd (m) efast_setfreq (n) efast_ttest (o) table_header_check (p) num_decimals

---

`Utility: Generate Median Distribution(s)`
Generate Medians Subset (getMediansSubset)

Description

This utility function takes a set of simulation runs, all performed with the same parameter set, and generates the set of median output measure results. This set will contain the median for each output measure, for each run performed. This aids reduction in aleatory uncertainty and production of a representative result. The set of medians is returned to the function that calls this - such that it can be processed appropriately

Usage

```
getMediansSubset(FILEPATH,NUMRUNSPERSAMPLE,MEASURES,
RESULTFILENAME,ALTFILENAME,OUTPUTFILECOLSTART,
OUTPUTFILECOLEND)
```

Arguments

FILEPATH	Directory where the parameter sample medians should be output to
NUMRUNSPERSAMPLE	The number of runs performed for each parameter subset. This figure is generated through Aleatory Analysis
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
RESULTFILENAME	Name of the simulation results file (e.g. "trackedCells_Close.csv")

ALTFILENAME In some cases, it may be relevant to read from a further results file if the initial file contains no results. This filename is set here

OUTPUTFILECOLSTART
Column number in the results file where output begins - saves (a) reading in unnecessary data, and (b) errors where the first column is a label, and therefore could contain duplicates

OUTPUTFILECOLEND
Column number in the results file where the last output measure is

---

Utility: Sample Data *Utility: Sample Data*

Description

Sample Data sets to view for furthering understanding of output generated by the available methods. These are described in the detailed description of each technique above

Usage

```
data(AA_Example_ATestMaxAndMedians)
data(Curve1_chemoLowerLinearAdjust)
data(Curve1_chemoUpperLinearAdjust)
data(Curve1_chemoThreshold)
data(Curve1_Dummy)
data(Curve1_maxVCAMeffectProbabilityCutoff)
data(Curve1_Results_Summary)
data(Curve1_thresholdBindProbability)
data(Curve1_vcamSlope)
data(eFAST_Analysis)
data(EgSet_LHC_corCoeffs)
data(LHC_Parameters_for_Runs)
data(LHC_Summary)
data(OAT_Example_ATests_chemoLowerLinearAdjust)
data(OAT_Medians)
data(OAT_Sampling_chemoThreshold_Values)
data(OAT_Sampling_maxVCAMeffectProbabilityCutoff_Values)
```

Index