Package 'spartan'

September 6, 2017

Type Package

Title Simulation Parameter Analysis R Toolkit ApplicatioN: Spartan

Version 3.0.0

Description Computer simulations are becoming a popular technique to use in attempts to further our understanding of complex systems. SPARTAN, first described in our 2013 publication in PLoS Computational Biology, provided 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 added support for Netlogo simulations, aiding simulation developers who use Netlogo to build their simulations perform the same analyses. 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. Version 3.0 offers significant additional functionality that permits the creation of emulations of simulation results, derived using the same sampling techniques in the global sensitivity analysis techniques, and the generation of combinations of these machine learning algorithms to one create one predictive tool, more commonly known as an ensemble model. Version 3.0 also improved the standard of the graphs produced in the original sensitivity analysis techniques, and introduced a polar plot to examine parameter sensitivity.

License GPL-2

Encoding UTF-8

LazyData true

Imports lhs, gplots, XML, plotrix, mlegp, ggplot2, neuralnet, psych, mco

Depends randomForest, e1071, R (>= 2.10.0)

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL http://www.york.ac.uk/ycil/software/spartan

BugReports http://github.com/kalden/spartan/issues

NeedsCompilation no

Author Kieran Alden [cre, aut],
Mark Read [aut],
Paul Andrews [aut],
Jason Cosgrove [aut],
Mark Coles [aut],
Jon Timmis [aut]

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

Repository CRAN

Date/Publication 2017-09-06 08:34:32 UTC

R topics documented:

`aa_getATestResults` *Calculates the A-Test scores observed for all sets, for each sample size*

Description

Examines the summary CSV file produced either by the method `aa_summariseReplicateRuns` 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. 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.

Usage

```
aa_getATestResults(FILEPATH, SAMPLESIZES, NUMSUBSETSPERSAMPLESIZE, MEASURES,
  ATESTRESULTSFILENAME, LARGEDIFFINDICATOR, AA_SIM_RESULTS_FILE = NULL,
  AA_SIM_RESULTS_OBJECT = NULL, TIMEPOINTS = NULL, TIMEPOINTSCALE = NULL,
  GRAPHNAME = 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)
`MEASURES`	An array containing the names of the simulation output measures to be analysed.
`ATESTRESULTSFILENAME`	Name of the file that will contain the A-Test scores for each sample size
`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

AA_SIM_RESULTS_FILE
: The name of the CSV file containing the simulation responses, if reading from a CSV file

AA_SIM_RESULTS_OBJECT
: The name of the R object containing the simulation responses, if not reading from a CSV file

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.

---

aa_graphATestsForSampleSize
: *Produce a plot for each sample size, showing the A-Test scores for each set of that size*

Description

Produce a plot for each sample size, showing the A-Test scores for each set of that size

Usage

```
aa_graphATestsForSampleSize(FILEPATH, ATESTS, MEASURES, LARGEDIFFINDICATOR,
  GRAPHOUTPUTNAME, TIMEPOINT, TIMEPOINTSCALE)
```

Arguments

FILEPATH
: Directory where the results of the simulation runs, in folders or in single CSV file format, can be found

ATESTS
: Name of the file where the calculated A-Test scores can be found

MEASURES
: An array containing the names of the simulation output measures to be analysed.

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

GRAPHOUTPUTNAME
: Name of the graph to be output for each sample size. Should be in PDF format

TIMEPOINT
: Timepoint for which this plot is being created

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

aa_graphSampleSizeSummary *Plots a comparison of the maximum A-Test score for each sample size*

Description

Produces a full graph of the data generated by aa_sampleSize_Summary (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. This graph is named as stated in the parameter GRAPHOUTPUTFILE, with the timepoint appended if the analysis is for multiple timepoints.

Usage

```
aa_graphSampleSizeSummary(FILEPATH, MEASURES, MAXSAMPLESIZE, SMALL, MEDIUM,
  LARGE, GRAPHOUTPUTFILE, SAMPLESUMMARY_OBJECT = NULL,
  SAMPLESUMMARY_FILE = NULL, 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
MEASURES	An array containing the names of the simulation output measures to be analysed.
MAXSAMPLESIZE	The highest number of samples used.
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
SAMPLESUMMARY_OBJECT	The name of an R object in the environment containing the summary A-Test scores for this sample size
SAMPLESUMMARY_FILE	The name of the CSV containing the summary A-Test scores for this sample size
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"

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

---

aa_sampleSizeSummary *Determines the median and maximum A-Test score observed for each sample size*

Description

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.

Usage

```
aa_sampleSizeSummary(FILEPATH, SAMPLESIZES, MEASURES, SUMMARYFILENAME,
  ATESTRESULTS_FILE = NULL, ATESTRESULTS_OBJECT = NULL, TIMEPOINTS = NULL,
  TIMEPOINTSCALE = 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

MEASURES An array containing the names of the simulation output measures to be analysed.

SUMMARYFILENAME
Name of the file generated that lists the maximum and median A-Test results for each sample size.

ATESTRESULTS_FILE
The name of a CSV file containing the A-Test results calculated by aa_getATestResults, if reading from a CSV file.

ATESTRESULTS_OBJECT
The name of an R object containing the A-Test results calculated by aa_getATestResults, if not reading from a CSV file

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"

`aa_summariseReplicateRuns` *Summarise results in set folder structure into one single CSV file*

Description

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 non-deterministic. 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 SUMMARYFILENAME. If doing this analysis over multiple timepoints, the timepoint will be appended to the filename given in SUMMARYFILENAME

Usage

```
aa_summariseReplicateRuns(FILEPATH, SAMPLESIZES, MEASURES, RESULTFILENAME,
  ALTFILENAME, OUTPUTFILECOLSTART, OUTPUTFILECOLEND, SUMMARYFILENAME,
  TIMEPOINTS = NULL, TIMEPOINTSCALE = 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
`MEASURES`	An array containing the names of the simulation output measures to be analysed.
`RESULTFILENAME`	Name of the simulation results file. In the current version, XML and CSV files can be processed. 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.
`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.
`OUTPUTFILECOLEND`	Column number in the simulation results file where the last output measure is.
`SUMMARYFILENAME`	Name of the file generated that lists the maximum and median A-Test results for each sample size.

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"

`analysenetwork_structures` *Analyse each network structure provided as a potential NN structure*

Description

Analyse each network structure provided as a potential NN structure

Usage

```
analysenetwork_structures(fold_size, dataset, model_formula, parameters,
  measures, algorithm_settings)
```

Arguments

fold_size	Number of rows in which the dataset is divided into folds
dataset	Dataset used to assess each strutcure
model_formula	Parameters and measures formula used to specify the input and output layers of the hidden network
parameters	Names of the parameters that form the input nodes of the neural network
measures	Names of the simulation responses that form the output node of the neural network
algorithm_settings	Object output from the function emulation_algorithm_settings, containing the settings of the machine learning algorithms to use in emulation creation. In this case, the settings parameter we are interested in is number of generations

Value

mean squared errors for the assessed structures

atest — *Calculates the A-test score for two distributions*

Description

Calculates the A-test score for two distributions

Usage

```
atest(x, y)
```

Arguments

x, y	distributions of simulation results

Value

A-Test score

calculate_fold_MSE — *Calculate the mean squared error for this fold in k-fold cross validation*

Description

Calculate the mean squared error for this fold in k-fold cross validation

Usage

```
calculate_fold_MSE(nn_predictions, test_fold, nn, measures)
```

Arguments

nn_predictions	Predictions made by neural net
test_fold	The test data on which performance is being assessed
nn	The generated neural network
measures	The simulation output responses that are being predicted

Value

Mean Squared error for this fold.

`calculate_weights_for_ensemble_model` *Internal function to calculate the weights for all emulators in the ensemble*

Description

Internal function to calculate the weights for all emulators in the ensemble

Usage

```
calculate_weights_for_ensemble_model(all_model_predictions, emulator_test_data,
  measures, emulator_types, num_of_generations = 8e+05)
```

Arguments

`all_model_predictions`	Set of test set predictions obtained for all emulators in the ensemble
`emulator_test_data`	Data on which the ensemble performance will be assessed
`measures`	Simulation responses the model should predict
`emulator_types`	Machine learning techniques being employed
`num_of_generations`	Number of generations for which the neural network that is generating the weights should attempt to converge within

Value

weights to use for each emulator in the ensemble

`createAndEvaluateFolds` *Create and evaluate folds within k-fold cross validation*

Description

Create and evaluate folds within k-fold cross validation

Usage

```
createAndEvaluateFolds(fold_size, test_fold_start, test_fold_end, dataset,
  network_struc, formula, parameters, measures, algorithm_settings)
```

Arguments

fold_size	Number of rows of the dataset that form each fold
test_fold_start	Starting position of the training or test fold in the dataset
test_fold_end	End position of the training or test fold in the dataset
dataset	Dataset for which the training and test folds are being developed
network_struc	Network structure for which the folds are being assessed
formula	Parameters and measures formula to use in creating the network
parameters	Names of the parameters that form the input nodes of the neural network
measures	Names of the simulation responses that form the output node of the neural network
algorithm_settings	Object output from the function emulation_algorithm_settings, containing the settings of the machine learning algorithms to use in emulation creation. In this case, the settings parameter we are interested in is number of generations

Value

MSE errors for all network structures

---

createtest_fold *Create test data fold for k-fold cross validation*

Description

Create test data fold for k-fold cross validation

Usage

```
createtest_fold(fold, number_of_folds, test_fold_start, test_fold_end, dataset)
```

Arguments

fold	Number of the fold being examined
number_of_folds	Total number of folds
test_fold_start	Position at which the test fold starts in the dataset
test_fold_end	Position at which the test fold ends in the dataset
dataset	Testing data

Value

Testing data for this fold of k-fold cross validation

createTrainingFold *Create training data fold for k-fold cross validation*

Description

Create training data fold for k-fold cross validation

Usage

```
createTrainingFold(fold, number_of_folds, training_fold_start,
  training_fold_end, dataset)
```

Arguments

fold	Number of the fold being examined
number_of_folds	Total number of folds
training_fold_start	Position at which the training fold starts in the dataset
training_fold_end	Position at which the training fold ends in the dataset
dataset	Training data

Value

Training data for this fold of k-fold cross validation

create_abc_settings_object *Creates ensemble-specific parameters for ABC analysis*

Description

The EasyABC model wrapper can only take one parameter input: the parameter values. This is problematic as to generate a prediction for those values, we must provide the names of the simulation parameters and measures, the built ensemble, and whether or not the parameter set and responses have been normalised. To get around that problem, this method creates an object in the working directory that contains these values, and the ensemble abc wrapper provided in spartan can then read these in. Thus, this method MUST be run before using the EasyABC package with the ensemble

Usage

```
create_abc_settings_object(parameters, measures, built_ensemble,
  normalise_values = FALSE, normalise_result = FALSE)
```

Arguments

parameters	Array containing the names of the parameters for which posterior distributions will be generated
measures	Names of the simulation output responses which the ensemble predicts
built_ensemble	An ensemble object created by spartan
normalise_values	Whether the data provided by the EasyABC algorithm should be normalised (as the ensemble must take data between 0 and 1). More than likley this is TRUE, to ensure the posterior distributions are presented in their correct scale
normalise_result	Whether the results produced in running abc generated parameter sets using the ensemble should be rescaled.

---

create_ensemble *Internal function to create the ensemble*

Description

Internal function to create the ensemble

Usage

```
create_ensemble(ensemble_emulations, all_emulator_predictions,
  emulator_test_data, measures, emulator_types, pre_normed_mins,
  pre_normed_maxes, algorithm_settings = NULL, normalise = FALSE,
  timepoint = NULL)
```

Arguments

ensemble_emulations	All emulations to build into the ensemble
all_emulator_predictions	Test set predictions from all emulators, on which the ensemble will be trained / emulators weighted
emulator_test_data	Data on which the ensemble performance will be assessed
measures	Simulation responses the model should predict
emulator_types	Machine learning techniques being employed
pre_normed_mins	The minimum values of each parameter prior to data normalisation. Used to rescale the results
pre_normed_maxes	The maximum values of each parameter prior to data normalisation. Used to rescale the results

algorithm_settings
: Object output from the function emulation_algorithm_settings, containing the settings of the machine learning algorithms to use in emulation creation. Used here to obtain settings relevant to ensemble creation - namely number of generations and whether the ensemble should be saved to file, as well as whether plots should be produced showing ensemble performance.

normalise
: Whether the predictions generated when testing the ensemble should be normalised for presenting test results

timepoint
: Simulation timepoint for which an ensemble is being created

Value

Generated ensemble object

---

create_neural_network *Create neural network emulator, using neuralnet package*

Description

Create neural network emulator, using neuralnet package

Usage

```
create_neural_network(formula, training_set, fold_number, normalised_data,
  network_structure, number_of_generations)
```

Arguments

formula
: Parameters and measures formula to use in creating the network

training_set
: Training data on which to train the network

fold_number
: Number of the fold being created in k-fold cross validation

normalised_data
: Normalised version of the training data

network_structure
: Hidden layers structure to use to create the network

number_of_generations
: Number of generations to train the network within in the hope of convergence, else training terminates

Value

Neural network created

`determine_optimal_neural_network_structure`
Determine the optimal hidden layer structure from those provided

Description

Determine the optimal hidden layer structure from those provided

Usage

```
determine_optimal_neural_network_structure(dataset, parameters, measures,
  algorithm_settings)
```

Arguments

`dataset`	Data on which the neural network is being trained
`parameters`	Names of the parameters that form the input nodes of the neural network
`measures`	Names of the simulation responses that form the output node of the neural network
`algorithm_settings`	Object output from the function emulation_algorithm_settings, containing the settings of the machine learning algorithms to use in emulation creation. In this case, the settings parameter we are interested in is number of generations

Value

Optimal network structure from those provided for assessment

`efast_generate_medians_for_all_parameter_subsets`
Generates summary file for stochastic simulations stored in multiple files

Description

Only to be applied in cases where simulation responses are supplied in the folder structure shown in the R Journal paper, 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.

Usage

```
efast_generate_medians_for_all_parameter_subsets(FILEPATH, NUMCURVES,
  PARAMETERS, NUMSAMPLES, NUMRUNSPERSAMPLE, MEASURES, RESULTFILENAME,
  ALTERNATIVEFILENAME, OUTPUTCOLSTART, OUTPUTCOLEND, 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
RESULTFILENAME	Name of the simulation results file. In the current version, XML and CSV files can be processed. 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.
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.
OUTPUTCOLEND	Column number in the simulation results file where the last output measure is. Only required if running the first method.
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	Sets the scale of the timepoints being analysed, e.g. "Hours"

---

efast_generate_sample *Generates parameter sets for variance-based eFAST Sensitivity Analysis*

Description

This technique analyses simulation results generated through sampling using the eFAST approach (extended Fourier Amplitude Sampling Test). 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. Samples are output to CSV file, one per parameter/curve pairing

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

---

efast_generate_sample_netlogo

Prepares Netlogo experiment files for a variance-based sensitivity analysis, using eFAST

Description

Creates a set of parameter values, over the specified value space, using the sampling method described in the eFAST technique. Then processes each of these into a Netlogo experiment XML file, from which a simulation can be run.

Usage

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

Arguments

FILEPATH	Directory where the parameter samples and XML models are to be stored
NUMCURVES	The number of 'resamples' to perform (see eFAST documentation) - recommend using at least 3.
NUMSAMPLES	The number of parameter subsets to be generated for each curve in the eFAST design.
MEASURES	Array containing the names of the Netlogo output measures which are used to analyse the simulation.
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). The reader should read the relevant tutorial in detail.
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_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.

---

efast_get_overall_medians

Calculates the summary stats for each parameter set (median of any replicates)

Description

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 method efast_generate_medians_for_all_parameter_subsets 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.

Usage

```
efast_get_overall_medians(FILEPATH, NUMCURVES, PARAMETERS, NUMSAMPLES, MEASURES,
  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
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
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	Sets the scale of the timepoints being analysed, e.g. "Hours"

---

efast_graph_Results *Plot the parition of variance in a simulation response for each measure*

Description

Plot the parition of variance in a simulation response for each measure

Usage

```
efast_graph_Results(RESULTS_FILE_PATH, PARAMETERS, si, sti, errors_si,
  errors_sti, MEASURES, TIMEPOINT, TIMEPOINTSCALE)
```

Arguments

RESULTS_FILE_PATH	Where the eFAST results were saved to
PARAMETERS	Simulation parameters being explored
si	Vector of Si values calculated in eFAST for all parameters
sti	Vector of STi values calculated in eFAST for all parameters
errors_si	Vector of confidence intervals for Si values for all parameters
errors_sti	Vector of confidence intervals for STi values for all parameters
MEASURES	Simulation output measures
TIMEPOINT	Timepoint being analysed
TIMEPOINTSCALE	Scale in which the timepoints are measures

---

efast_netlogo_get_overall_medians

Deprecated: Use `efast_netlogo_get_overall_medians`

Description

Deprecated: Use `efast_netlogo_get_overall_medians`

Usage

```
efast_netlogo_get_overall_medians(FILEPATH, NUMCURVES, PARAMETERS, NUMSAMPLES,
  MEASURES, 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
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
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	Sets the scale of the timepoints being analysed, e.g. "Hours"

`efast_netlogo_run_Analysis` *Deprecated: Use* `efast_run_Analysis`

Description

Deprecated: Use `efast_run_Analysis`

Usage

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

Arguments

`FILEPATH`	Directory where the simulation runs can be found, in folders or in CSV file format
`MEASURES`	Array containing the names of the output measures which are used to analyse the simulation
`PARAMETERS`	Array containing the names of the parameters of which parameter samples have been generated
`NUMCURVES`	The number of 'resamples' to perform (see eFAST documentation) - recommend using at least 3
`NUMSAMPLES`	The number of parameter subsets that were generated in the eFAST design
`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
`TTEST_CONF_INT`	The level of significance to use for the T-Test (e.g. 0.95)
`GRAPH_FLAG`	Whether graphs should be produced summarising the output - should be TRUE or FALSE
`EFASTRESULTFILENAME`	File name under which the full eFAST analysis should be stored. This will contain the partitioning of variance for each parameter.
`TIMEPOINTS`	mplemented 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`	Sets the scale of the timepoints being analysed, e.g. "Hours"

efast_process_netlogo_result *Analyses Netlogo simulation data for parameter sets generated for eFAST*

Description

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 `efast_get_overall_medians` and `efast_run_Analysis` to analyse the results

Usage

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

Arguments

FILEPATH	Directory where the simulation runs can be found
EFASTSAMPLE_RESULTFILENAME	Name of the result file generated by Netlogo, for an eFAST parameter sample.
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
NUMCURVES	The number of 'resamples' to perform (see eFAST documentation) - recommend using at least 3.
NUMSAMPLES	The number of parameter subsets that were generated from each curve in the eFAST design
MEASURES	Array containing the names of the Netlogo output measures which are used to analyse the simulation.
RESULTFILENAME	Name of the result file that will be produced which summarises all results
TIMESTEP	Timestep of the Netlogo simulation being analysed.

efast_run_Analysis *Runs the eFAST Analysis for the pre-generated summary file*

Description

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 references in the R Journal paper. For ease of representation, the method also produces a graph showing this data for each simulation output measure. These graphs and summaries can be produced for multiple timepoints.

Usage

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

Arguments

FILEPATH	Directory where the simulation runs can be found, in folders or in CSV file format
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
PARAMETERS	Array containing the names of the parameters of which parameter samples have been generated
NUMCURVES	The number of 'resamples' to perform (see eFAST documentation) - recommend using at least 3
NUMSAMPLES	The number of parameter subsets that were generated in the eFAST design
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
TTEST_CONF_INT	The level of significance to use for the T-Test (e.g. 0.95)
GRAPH_FLAG	Whether graphs should be produced summarising the output - should be TRUE or FALSE
EFASTRESULTFILENAME	File name under which the full eFAST analysis should be stored. This will contain the partitioning of variance for each parameter.
TIMEPOINTS	mplemented 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	Sets the scale of the timepoints being analysed, e.g. "Hours"
GRAPHTIME	Value never needs stating, used internally to produce graphs when processing multiple timepoints

emulated_lhc_values *Latin-hypercube value set use to demonstrate emulated sensitivity analysis*

Description

A dataset containing values for the six parameters that control the simulation detailed in the case study section of the vignette. These parameters are defined the accompanying publications referred to in the vignette

Usage

```
data(emulated_lhc_values)
```

Format

A list with 500 rows (one per parameter set) and six columns

Details

- stableBindProbability. Parameter values between 0 and 100
- chemokineExpressionThreshold. Parameter values between 0 and 1
- initialChemokineExpressionValue. Parameter values between 0.1 and 0.5
- maxChemokineExpressionValue. Parameter values between 0.015 and 0.08
- maxProbabilityOfAdhesion. Parameter values between 0 and 1
- adhesionFactorExpressionSlope. Parameter values between 0.25 and 5

emulate_efast_sampled_parameters *Emulate simulations for a set of eFAST generated parameter values*

Description

This method runs an ensemble for all parameter value sets specified in a CSV file generated from spartan eFAST sampling techniques. The output is a set of CSV files that can then be analysed using the spartan analysis methods detailed in Technique 4.

Usage

```
emulate_efast_sampled_parameters(filepath, surrogateModel, parameters, measures,
  num_curves, ensemble_set = TRUE, normalise = FALSE, timepoint = NULL)
```

Arguments

filepath	Path to the folder containing the set of CSV files for eFAST analysis
surrogateModel	Ensemble or emulator to use to emulate the simulation responses for those sets
parameters	Simulation parameter names
measures	Simulation output response names
num_curves	Number of resample curves used in eFAST sampling
ensemble_set	Boolean stating whether this analysis is being run using an ensemble of machine learning methods (TRUE), or a single emulator (FALSE)
normalise	Whether the parameters in the sampling CSV files need to be normalised before input to the emulator (which must take values between 0 and 1)
timepoint	Simulation timepoint being analysed, if more than one. See the vignette for analysing more than one timepoint

---

emulate_lhc_sampled_parameters

Emulate simulations for a set of latin-hypercube generated parameter values

---

Description

This method runs an ensemble for all parameter value sets specified in a CSV file generated from spartan latin-hypercube sampling techniques. The output is a CSV file that can then be analysed using the spartan analysis methods detailed in Technique 3.

Usage

```
emulate_lhc_sampled_parameters(filepath, surrogateModel, parameters, measures,
  measure_scale, param_file = NULL, dataset = NULL, ensemble_set = TRUE,
  normalise = FALSE, timepoint = NULL, timepointscale = NULL)
```

Arguments

filepath	Path to where the analysis output should be stored
surrogateModel	Ensemble or emulator to use to emulate the simulation responses for those sets
parameters	Simulation parameter names
measures	Simulation output response names
measure_scale	Scale of each of the simulation responses
param_file	Name of the CSV file generated by spartan (or of) parameter values, separated in columns - if reading these in from a file
dataset	Name of the R dataset in the environment that contains the parameter sets (the tutorial one is emulated_lhc_values)
ensemble_set	Boolean stating whether this analysis is being run using an ensemble of machine learning methods (TRUE), or a single emulator (FALSE)

normalise	Whether the parameters in the sampling CSV file need to be normalised before input to the emulator (which must take values between 0 and 1)
timepoint	Simulation timepoint being analysed, if more than one. See the vignette for analysing more than one timepoint
timepointscale	Scale of the timepoints, if being used

`emulation_algorithm_settings` *Initialise machine-learning algorithms settings for emulation creation*

Description

Some of the machine learning algorithms incorporated have their own specific settings (neural network and random forest, for example). To keep the methods generic to multiple algorithms and to save passing multiple objects around, these are declared in this function and returned as an algorithmSettings object. Where the user wishes to use the default values, there is no need to run this function: this will be created during emulator generation. However, should the user wish to change any of the settings, such as the number of generations in the neural network or the number of trees in random forest, they should run this method with the new values. In addition, this object will also contain two other settings: whether graphs should be plotted of the accuracy of the emulator against the training and test sets, and whether the emulator object that is created should be stored in the working directory. Parameters this object stores are detailed in the arguments section. However, for neural network emulation, the user is required to initialise this object with a list of neural network hidden layer structures to evaluate. Should this not be done, an error message will be produced and the call will terminate.

Usage

```
emulation_algorithm_settings(num_trees = 500, num_of_generations = 8e+05,
  num_of_folds = 10, network_structures = NULL, save_emulators = TRUE,
  save_ensemble = TRUE, plot_test_accuracy = TRUE)
```

Arguments

num_trees	Number of trees used to generate a random forest model. If a random forest is not selected as the emulation technique, this argument does not need to be provided. Defaults to 500
num_of_generations	Used in neural network generation, as the maximum number of steps used in training the network. If this is reached, then training is stopped. If a neural network is not selected as the emulation technique, this argument does not need to be provided. Defaults to 800,000
num_of_folds	Number of folds (subsets of training data) used in k-fold cross validation when developing a neural network emulator. If a neural network is not selected as the emulation technique, this argument does not need to be provided. Defaults to 10.

`network_structures`
: List of neural network structures to examine using k-fold cross validation when developing a neural network emulator. Should be a list in the form of number of nodes in each hidden layer. For example, c(c(4),c(4,3)) would consider two structures, one for a single hidden layer of 4 nodes, and another with two hidden layers of 4 and 3 nodes. If a neural network is not selected as the emulation technique, this argument does not need to be provided.

`save_emulators`
: Boolean indicating whether the generated emulator for each simulation output should be saved to file (as an Rda object). This is stored in the current working directory. Defaults to TRUE

`save_ensemble`
: Used in Technique 7, to state whether an ensemble generated from multiple emulators should be saved to file. Defaults to TRUE

`plot_test_accuracy`
: Boolean indicating whether a plot showing a comparison against the emulator predicted values to those observed in the test set should be created. This ggplot is stored as a PDF in the current working directory. Defaults to FALSE

Value

List of all the elements in the parameters above

`emulator_parameter_evolution`
Evolve parameter sets that meet a desired ensemble outcome

Description

This method takes a user specified fitness function and runs the nsga2 algorithm on an ensemble using the nsga2 implementation provided in the mco package, in an attempt to locate parameters that achieve a desired response (determined by the fitness function). The method outputs a list describing the values for each simulation output measure, (or objective, res), an evolved set of parameter inputs (par), and a boolean stating whether the candidate is pareto optimal (pareto.optimal)

Usage

```
emulator_parameter_evolution(function_to_evaluate, nsga2_user_set_parameters,
  nsga2_settings)
```

Arguments

`function_to_evaluate`
: A user-defined function that NSGA2 seeks to minimise

`nsga2_user_set_parameters`
: An object containing the emulator input and output names, the input parameters for function to evaluate, minimum and maximum values for emulator inputs. These should be set using the function that creates that object prior to running this method

nsga2_settings An object containing the population size, number of generations, crossover probability and mutation probability to be assessed. Again see the function nsga2_settings to set these values before running this function

Value

List containing evolved parameter sets, the output for the ensemble using those sets, and whether these sets are pareto optimal

---

emulator_predictions *Used to generate predictions from an emulator, normalising data if required*

Description

With an emulator object produced, this can be used to generate predictions on unseen data. This method is called with the emulation object, parameters, meaasures, and unseen data. A flag should also be set as to whether the unseen data, and thus the generated prediction, need to be normalised and rescaled accordingly. Unseen data being input into the emulator must be scaled between 0 and 1, with predictions rescaled after generation.

Usage

```
emulator_predictions(emulation, parameters, measures, data_to_predict,
  normalise = FALSE, normalise_result = FALSE)
```

Arguments

emulation	The emulation object to use to make the predictions
parameters	Parameters on which the model will take as input
measures	Simulation responses the model should predict
data_to_predict	Unseen values for the parameters for which the measures should be predicted
normalise	Whether the data_to_predict should be normalised
normalise_result	Whether the resultant predictions should be normalised

Value

Predictions generated for this unseen data

`ensemble_abc_wrapper` *Wrapper to allow EasyABC functions to run using Ensemble*

Description

Provides a means of running the ensemble within the EasyABC methods. This method should be stated as the "model" argument of EasyABC methods such as ABC_sequential. Note that as arguments cannot be passed to the model function, these must be created before calling any EasyABC method (see code description for `create_abc_settings_object`. The created object most contain the simulation parameters, output measures, an ensemble object to be run, declared as built_ensemble, and whether or not the prior (and generated predictions) should be normalised (normalise). Should this object not exist an error message will be produced.

Usage

```
ensemble_abc_wrapper(x)
```

Arguments

x	Set of parameter values generated by an EasyABC method

Value

Ensemble prediction using parameter set in x

`generate_emulators_and_ensemble`
Generate a set of emulators and combine into an ensemble

Description

This method generates all requested emulators then combines these into one ensemble. This takes as input a list of the emulation objects to create (could be random forest, support vector machine, neural network, general linear model, and gaussian process model), the simulation parameters and output response labels, an object created by the partitioned_dataset method (training, testing, and validation datasets), and an object created by method emulation_algorithm_settings. The latter sets key arguments used in emulation creation, as detailed in the description accompanying that method.

Usage

```
generate_emulators_and_ensemble(model_list, parameters, measures,
  partitioned_data, algorithm_settings = NULL, timepoint = NULL,
  normalised = FALSE)
```

Arguments

model_list	Vector of the types of emulation model to create. Accepted abbreviations are: SVM (Support-Vector Machine), GP (Gaussian Process Model), NNET (Neural Network), RF (Random Forest), GLM (General Linear Model)
parameters	Vector containing the names of the simulation parameters in the dataset on which the emulator is being trained
measures	Vector containing the simulation outputs that the emulators should be able to predict
partitioned_data	Object output from the function partition_dataset, an object containing training, testing, and validation data
algorithm_settings	Object output from the function emulation_algorithm_settings, containing the settings of the machine learning algorithms to use in emulation creation. If no setting changes are required, and a neural network is not being generated, this can be left out, and will be generated by generate_requested_emulations (so this defaults to NULL). If you are making any changes to the settings or generating a neural network, you must create this object before calling generate_requested_emulations.
timepoint	If using multiple timepoints, the timepoint for which emulators are being created
normalised	Whether the emulator data has been normalised or not. Affects how training and test output predictions are displayed

Value

A list containing the ensemble, the time taken to generate it, and the sampling mins and maxes used in its creation such that unseen data used by and predictions generated by the ensemble can be scaled and rescaled correctly

---

generate_ensemble_from_existing_emulations

Generate an ensemble from previously created spartan emulation objects

---

Description

Where emulations have already been created, this method combines these to form one ensemble. This takes as input a list of the emulator objects, the simulation parameters and output response labels, and a set of test data from which the performance weights will be evolved. We would recommend providing the testing set of the output from the partition_dataset method. An option is given, by setting these within emulation_algorithm_settings, to save the ensemble object to file, as well as produce plots showing the accuracy of the generated ensemble for the test data set

Usage

```
generate_ensemble_from_existing_emulations(existing_emulations, parameters,
  measures, observed_data, algorithm_settings = NULL, normalise = FALSE,
  timepoint = NULL)
```

Arguments

existing_emulations	Vector of emulator objects created by method generate_requested_emulations
parameters	Array containing the names of the parameters for which values are input into each emulation
measures	Array containing the names of the output measures predicted by each emulation
observed_data	Dataset to train the new ensemble on. We recommend using the test data in the set generated by partition_data method, and not the training set, as the emulators themselves have been trained on that data and the ensemble could thus overfit.
algorithm_settings	Object output from the function emulation_algorithm_settings, containing the settings of the machine learning algorithms used in emulation creation. Here this is needed to decide whether any accuracy plots should be produced during ensemble creation, whether or not the ensemble should be saved to file, and to specify the number of generations for which the neural network that is generating the algorithm weightings should run.
normalise	Whether the predictions generated when testing the ensemble should be normalised for presenting test results
timepoint	If using multiple timepoints, the timepoint for which this ensemble is being created

Value

A list containing the ensemble, the time taken to generate it, and the sampling mins and maxes used in its creation such that unseen data used by and predictions generated by the ensemble can be scaled and rescaled correctly

---

generate_ensemble_training_set

Internal function used to combine test set predictions from emulators to form the ensemble training set

---

Description

Internal function used to combine test set predictions from emulators to form the ensemble training set

Usage

```
generate_ensemble_training_set(emulator, parameters, measures, observed_data,
  all_model_predictions)
```

Arguments

emulator	An emulator object from which the test set data is being predicted
parameters	Vector containing the names of the simulation parameters in the dataset on which the emulator is being trained
measures	Vector containing the simulation outputs that the emulators should be able to predict
observed_data	Data obtained from experimentation on the simulator itself, and now used to train the ensemble
all_model_predictions	The set of predictions from numerous emulators to which this set of predictions is being added

Value

updated all_model_predictions containing predictions for this emulator. This updated list becomes the training set for the ensemble.

---

generate_requested_emulations

Generate emulators for specified machine learning techniques with provided data

---

Description

This method generates an emulator model from a training set for a specified technique, and generates performance statistics from the test set. The currently implemented techniques are a neural network (using the neuralnet package), a random forest (from the randomforest package), a support vector machine (from package e1071), a gaussian process model (from package mlegp), and a general linear model. Where a neural network is desired, the hyper-parameters are determined using k-fold cross validation from a set of specified network structures. Where a simulation has multiple outputs, an emulator model is created for each output response. This method provides capacity to save the generated emulator models to file, in Rda format, and plot a comparison of the predicted responses to a set of those of the training and test sets, giving correlation of determination (R-squared) and mean squared error values. The method returns a list of emulators of a specified technique, one for each simulation output, and the performance statistics for each measure, including the time taken to generate these emulators. If the training data has been normalised, minimum and maximum sampling values for each parameter are also returned such that any predictions generated using this emulation can be rescaled correctly. If plots are desired (by setting a flag in emulation_algorithm settings), plots produced are stored as PDF's in the working directory. The same applies to saving the generated emulator, set by the saveEmulation flag in emulation_algorithm_settings. Note that it must be specified as to whether the data being provided in partitioned_data has been normalised or

not: this affects the output of the plots (as the data is rescaled back to its original scale if the data was normalised). Similarly to the rest of spartan, this method can create emulations for multiple timepoints.

Usage

```
generate_requested_emulations(model_list, partitioned_data, parameters,
  measures, algorithm_settings = NULL, timepoint = NULL,
  normalised = FALSE)
```

Arguments

model_list	Vector of the types of emulation model to create. Accepted abbreviations are: SVM (Support-Vector Machine), GP (Gaussian Process Model), NNET (Neural Network), RF (Random Forest), GLM (General Linear Model)
partitioned_data	Object output from the function partition_dataset, an object containing training, testing, and validation data
parameters	Vector containing the names of the simulation parameters in the dataset on which the emulator is being trained
measures	Vector containing the simulation outputs that the emulators should be able to predict
algorithm_settings	Object output from the function emulation_algorithm_settings, containing the settings of the machine learning algorithms to use in emulation creation. If no setting changes are required, and a neural network is not being generated, this can be left out, and will be generated by generate_requested_emulations (so this defaults to NULL). If you are making any changes to the settings or generating a neural network, you must create this object before calling generate_requested_emulations.
timepoint	If using multiple timepoints, the timepoint for which emulators are being created
normalised	Whether the emulator data has been normalised or not. Affects how training and test output predictions are displayed

Value

Emulation objects, bundled into a list, with the required sampling information to rescale the data these emulations produce if required

---

graph_Posteriors_All_Parameters

Graph posterior distributions generated for all parameters, to PDF file

---

Description

We provide a means of plotting the produced posterior distribution for all parameters for which the value is being explored. Output to PDF in the working directory

Usage

```
graph_Posteriors_All_Parameters(abc_resultset, parameters, sampleMins,
  sampleMaxes)
```

Arguments

abc_resultset	Result object obtained from the EasyABC package
parameters	Array containing the names of the parameters for which posterior distributions will be generated
sampleMins	Minimum value of the range over which each parameter was explored using ABC
sampleMaxes	Maximum value of the range over which each parameter was explored using ABC

---

kfoldCrossValidation *Perform k-fold cross validation for assessing neural network structure performance*

Description

Perform k-fold cross validation for assessing neural network structure performance

Usage

```
kfoldCrossValidation(dataset, parameters, measures, algorithm_settings)
```

Arguments

dataset	Data on which the neural network is being trained and assessed
parameters	Names of the parameters that form the input nodes of the neural network
measures	Names of the simulation responses that form the output node of the neural network
algorithm_settings	Object output from the function emulation_algorithm_settings, containing the settings of the machine learning algorithms to use in emulation creation. In this case, the settings parameter we are interested in is number of generations

Value

Mean Squared errors for all potential structures

lhc_calculatePRCCForMultipleTimepoints

Calculates the PRCC for each parameter at each timepoint, storeing PRCC and P-Value in two different files to make the plot function easier

Description

Calculates the PRCC for each parameter at each timepoint, storeing PRCC and P-Value in two different files to make the plot function easier

Usage

```
lhc_calculatePRCCForMultipleTimepoints(FILEPATH, CORCOEFFSOUTPUTFILE,
  TIMEPOINTS, MEASURES)
```

Arguments

FILEPATH	Directory containing all PRCC containing result files
CORCOEFFSOUTPUTFILE	Name of the file containing the PRCCS for one timepoint. Assume that the timepoint is appended on to the end (i.e. results_12.csv for hour 12 - the timepoint is appended by spartan so only specify results.csv
TIMEPOINTS	Simulation timepoints to analyse
MEASURES	Simulation output measures being analysed, in a vector

lhc_countSignificantParametersOverTime

Count number of significant (p<0.01) parameters over a timecourse

Description

Count number of significant (p<0.01) parameters over a timecourse

Usage

```
lhc_countSignificantParametersOverTime(FILEPATH, MEASURES, TIMEPOINTS)
```

Arguments

FILEPATH	Where pre-processed P-Value dataset is found
MEASURES	Simulation output responses vector
TIMEPOINTS	Timepoints to analyse

lhc_generateLHCSummary *Summarises simulation behaviour for each parameter set, by median of distribution of replicate runs*

Description

Processes either the CSV file generated by lhc_process_sample_run_subsets or one that has been supplied, going through each line of that file 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. This method can also be performed for a number of simulation timepoints

Usage

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

Arguments

FILEPATH	Directory where the simulation runs of single CSV file can be found
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
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).
LHCSUMMARYFILENAME	Name of the LHC Summary file to be generated. Contains each parameter set alongside the result gained when the simulation was run under that criteria.
SPARTAN_PARAMETER_FILE	Location of the file output by the latin-hypercube sampling method (list of parameters). 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.
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"

lhc_generatePRCoEffs *Generate Partial Rank Correlation Coefficients for parameter/response pairs*

Description

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. Can be performed for a number of timepoints if required.

Usage

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

Arguments

FILEPATH	Directory where the simulation runs of single CSV file can be found
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
LHCSUMMARYFILENAME	Name of the LHC Summary file to be generated. Contains each parameter set alongside the result gained when the simulation was run under that criteria.
CORCOEFFSOUTPUTFILE	Name of the generated CSV file generated
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"

lhc_generateTimepointFiles *Generates spartan-compatible timepoint files if simulation results over time are in one file*

Description

Generates spartan-compatible timepoint files if simulation results over time are in one file

Usage

```
lhc_generateTimepointFiles(FILEPATH, SPARTAN_PARAMETER_FILE,
  RUN_SUMMARY_FILE_NAME, NUMSAMPLES, NUMRUNSPERSAMPLE, TIMEPOINTS)
```

Arguments

FILEPATH	Filepath to all LHC results being analysed
SPARTAN_PARAMETER_FILE	CSV file generated by spartan, containing all parameter sets under which the simulation was run
RUN_SUMMARY_FILE_NAME	Simulation output file, containing multiple timepoints
NUMSAMPLES	Number of parameter samples generated using the hypercube
NUMRUNSPERSAMPLE	Number of replicate runs performed for each sample
TIMEPOINTS	Simulation timepoints to extract from the result file

---

lhc_generate_lhc_sample

Generates sets of simulation parameters using latin-hypercube sampling

---

Description

Though robustness analysis 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 in the tutorial, which uses a latin-hypercube design 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. For each set of parameters, the simulation should be run for the number of times identified in Aleatory Analysis. Once this has been completed, the set of remaining methods within spartan 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)

`lhc_generate_lhc_sample_netlogo`

Prepares Netlogo experiment files for a sampling-based sensitivity analysis, using latin-hypercube sampling

---

Description

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.

Usage

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

Arguments

FILEPATH	Directory where the parameter samples are to be stored
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.
NUMSAMPLES	The number of parameter subsets to be generated in the LHC 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).
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_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.

---

lhc_generate_netlogo_PRCoEffs

Deprecated. Use `lhc_generatePRCoEffs` *instead*

Description

Deprecated. Use `lhc_generatePRCoEffs` instead

Usage

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

Arguments

FILEPATH
: Directory where the simulation runs of single CSV file can be found

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

MEASURES
: Array containing the names of the output measures which are used to analyse the simulation

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

CORCOEFFSOUTPUTFILE
: Name of the generated CSV file generated

---

lhc_graphMeasuresForParameterChange

Generates parameter/measure plot for each pairing in the analysis

---

Description

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.

Usage

```
lhc_graphMeasuresForParameterChange(FILEPATH, PARAMETERS, MEASURES,
  MEASURE_SCALE, CORCOEFFSOUTPUTFILE, LHCSUMMARYFILENAME,
  OUTPUT_TYPE = c("PDF"), TIMEPOINTS = NULL, TIMEPOINTSCALE = NULL,
  GRAPHTIME = NULL)
```

Arguments

FILEPATH	Directory where the simulation runs of single CSV file can be found
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
MEASURE_SCALE	Scale in which each of the output responses is measured. Used to label plots
CORCOEFFSOUTPUTFILE	File produced by spartan containing the Partial Rank Correlation Coefficients for each parameter/measure pairing
LHCSUMMARYFILENAME	Name of the LHC Summary file to be generated. Contains each parameter set alongside the result gained when the simulation was run under that criteria.
OUTPUT_TYPE	Type of graph to plot. Can be PDF, PNG, TIFF, BMP, etc, all formats supported by ggplot2
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"
GRAPHTIME	The timepoint being processed, if any. NULL if not.

---

lhc_graphPRCCForMultipleTimepoints

Produce a plot of PRCC values obtained at multiple timepoints

---

Description

Calculates the PRCC for each parameter at each timepoint in the TIMEPOINTS vector. Unlike the other methods in Spartan, this stores PRCC and P-Value in 2 different files to make the plot function easier.

Usage

```
lhc_graphPRCCForMultipleTimepoints(FILEPATH, MEASURES, TIMEPOINTS)
```

Arguments

FILEPATH	Directory where the pre-processed PRCC values can be found
MEASURES	Simulation response measures
TIMEPOINTS	Timepoints being analysed

---

lhc_netlogo_graphMeasuresForParameterChange

Deprecated. Use lhc_graphMeasuresForParameterChange *instead*

Description

Deprecated. Use lhc_graphMeasuresForParameterChange instead

Usage

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

Arguments

FILEPATH	Directory where the simulation runs of single CSV file can be found
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
MEASURE_SCALE	Scale in which each of the output responses is measured. Used to label plots
CORCOEFFSOUTPUTFILE	File produced by spartan containing the Partial Rank Correlation Coefficients for each parameter/measure pairing
LHCSUMMARYFILENAME	Name of the LHC Summary file to be generated. Contains each parameter set alongside the result gained when the simulation was run under that criteria.
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"

lhc_plotCoEfficients *Plots the PRCC coefficients against each other for ease of comparison*

Description

Plots the Partial Rank Correlation Coefficients for either all measures or for one individual measure, for all simulation parameters.

Usage

```
lhc_plotCoEfficients(FILEPATH, CORCOEFFSOUTPUTFILE, MEASURES, PRINTOPT,
  TIMEPOINTS = NULL, TIMEPOINTSCALE = NULL)
```

Arguments

FILEPATH	Location of the LHC result set
CORCOEFFSOUTPUTFILE	Name of the CSV file in FILEPATH containing the Partial Rank Correlation Coefficients
MEASURES	Names of the simulation responses
PRINTOPT	Used in plotting Partial Rank Correlation Coefficients, should be either "ALL" or "INDIVIDUAL"
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"

lhc_polarplot *Creates a polar plot for each response, showing PRCC for each parameter*

Description

Added in Spartan 3.0. Provides a means of plotting the partial rank correlation coefficients as a polar plot, to ease comparison of these values.

Usage

```
lhc_polarplot(FILEPATH, PARAMETERS, MEASURES, CORCOEFFSOUTPUTFILE,
  TIMEPOINTS = NULL, TIMEPOINTSCALE = NULL)
```

Arguments

FILEPATH	Directory where the simulation runs of single CSV file can be found
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
CORCOEFFSOUTPUTFILE	File produced by spartan containing the Partial Rank Correlation Coefficients for each parameter/measure pairing
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"

---

lhc_process_netlogo_result

Analyses Netlogo simulations generated for a latin-hypercube based sensitivity analysis

---

Description

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 `lhc_generateLHCSummary`, `lhc_generatePRCoEffs`, and `lhc_graphMeasuresForParameterChange` as per analysing any data in spartan that was not generated by Netlogo

Usage

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

Arguments

FILEPATH	Directory where either the simulation runs can be found
LHCSAMPLE_RESULTFILENAME	Name of the result file generated by Netlogo, for a LHC parameter sample.
SPARTAN_PARAMETER_FILE	Location of the file output by the latin-hypercube sampling method, containing the parameters on which this experiment was performed
NUMSAMPLES	The number of parameter subsets that were generated in the LHC design

MEASURES Array containing the names of the Netlogo output measures which are used to analyse the simulation.

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

TIMESTEP The timestep of the Netlogo simulation being analysed

---

`lhc_process_sample_run_subsets`
Summarises results of runs for parameter sets generated by a latin-hypercube

---

Description

Only to be applied for simulations that are stochastic, and responses are supplied in the folder structure detailed in the R Journal paper, 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.

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)
```

Arguments

FILEPATH Directory where the simulation runs of single CSV file can be found

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.

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 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. In the current version, XML and CSV files can be processed. 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.
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.
OUTPUTCOLEND	Column number in the simulation results file where the last output measure is.
LHC_ALL_SIM_RESULTS_FILE	Name to be given to the CSV file that summarises all simulation runs for all parameter sets
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"

`normaliseATest` *Normalises the A-Test such that it is above 0.5*

Description

Normalises the A-Test such that it is above 0.5

Usage

```
normaliseATest(result)
```

Arguments

result	A-Test score to be normalised

Value

Normalised A-Test score

normalise_dataset *Normalise a dataset such that all values are between 0 and 1*

Description

Normalise a dataset such that all values are between 0 and 1

Usage

```
normalise_dataset(dataset, sample_mins, sample_maxes, parameters)
```

Arguments

dataset	LHC Summary file being used in emulator creation
sample_mins	The minimum value used for each parameter in generating the latin-hypercube sample
sample_maxes	The maximum value used for each parameter in generating the latin-hypercube sample
parameters	Simulation parameters the emulation will be fed as input

Value

List of the scaled data and the mimumum/maximum sample values for each, to aid rescale of the data and any predictions made using it.

nsga2_set_user_params *Initialise analysis specific parameters for NSGA-2*

Description

Creates an object of the analysis parameters that will be used to evolve parameter sets or screen parameters for NSGA-2. The user should ensure this is calle first, establishing this object such that it can be passed in to the relevant method

Usage

```
nsga2_set_user_params(built_ensemble, parameters, measures, desiredResponses,
  sampleMins, sampleMaxes)
```

Arguments

built_ensemble	Ensemble object that will be used in the NSGA-2 algorithm to generate predictions
parameters	Names of simulation parameters for which values are input to the ensemble
measures	Names of the simulation measures for which the ensemble predicts
desiredResponses	Vector of desired responses for the simulation measures, used by the fitness function to determine goodness of fit for evolved parameter sets
sampleMins	Minimum value of the range of each parameter to be used in evolving parameter sets
sampleMaxes	Maximum value of the range of each parameter to be used in evolving parameter sets

Value

List of the above objects for passing in as settings object to NSGA-2 related methods

---

num.decimals *Diagnostic function used to determine number of decimal places*

Description

Diagnostic function used to determine number of decimal places

Usage

```
num.decimals(x)
```

Arguments

x	Numeric value to examine

Value

Number of decimal places

`oat_countResponsesOfDesiredValue` *Counts the number of simulation responses where a output response equals a desired result, for a specified parameter.*

Description

For example, how many simulations produce a value of 0 as the output. Outputs this information as a CSV file.

Usage

```
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 for which local analyses are being conducted
RESULTFILENAME	Name of the simulation results file. In the current version, XML and CSV files can be processed. If performing this analysis over multiple timepoints, it is assumed that the timepoint follows the file name, e.g. trackedCells_Close_12.csv.
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.
OUTPUTCOLEND	Column number in the simulation results file where the last output measure is. Only required if running the first method.
PARAMETER	Current parameter being analysed in this loop
NUMRUNSPERSAMPLE	The number of runs performed for each parameter subset. This figure is generated through Aleatory Analysis
MEASURE	Current simulation output measure being analysed in this loop
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
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 minimum value that should be used for each parameter. Sets a lower bound on sampling space
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
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"

`oat_csv_result_file_analysis`

Performs a robustness analysis for supplied simulation data, comparing simulation behaviour at different parameter values

---

Description

This method takes either the CSV file created in `oat_processParamSubsets` or 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. This method can be performed for a number of simulation timepoints, producing these statistics for each timepoint taken.

Usage

```
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)
```

Arguments

FILEPATH	Directory where either the simulation runs or single CSV file result can be found
CSV_FILE_NAME	Name of the CSV file in which the results of all simulations exist (or have been summarised)
PARAMETERS	Array containing the names of the parameters for which local analyses are being conducted

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

MEASURES Array containing the names of the output measures which are used to analyse the simulation

ATESTRESULTFILENAME
File name of the ATests result summary file that will be created For one timepoint, this could be ATests.csv. For additional timepoints, the time is added to the file name

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 minimum value that should be used for each parameter. Sets a lower bound on sampling space

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

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"

`oat_generate_netlogo_behaviour_space_XML`
Creates a Netlogo compatible behaviour space experiment for robustness analysis

---

Description

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, altering one parameter at a time in this case

Usage

```
oat_generate_netlogo_behaviour_space_XML(FILEPATH, NETLOGO_SETUPFILE_NAME,
  PARAMETERS, PARAMVALS, NETLOGO_SETUP_FUNCTION, NETLOGO_RUN_FUNCTION, MEASURES,
  EXPERIMENT_REPETITIONS, RUNMETRICS_EVERYSTEP)
```

Arguments

FILEPATH	Directory where the parameter samples are to be stored
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). See tutorial for more 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.

---

oat_graphATestsForSampleSize

Takes each parameter in turn and creates a plot showing A-Test score against parameter value.

Description

This makes it easy to determine how the effect that changing the parameter has had on simulation results. Graph for each parameter is output as a PDF

Usage

```
oat_graphATestsForSampleSize(FILEPATH, PARAMETERS, MEASURES, ATESTSIGLEVEL,
  ATESTRESULTFILENAME, 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 for which local analyses are being conducted
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
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
ATESTRESULTFILENAME	File name of the ATests result summary file that will be created For one timepoint, this could be ATests.csv. For additional timepoints, the time is added to the file name
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 minimum value that should be used for each parameter. Sets a lower bound on sampling space
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
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"

---

oat_parameter_sampling

Create parameter samples for robustness (local) analysis

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, 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. Once this has been completed, the results can be analysed using the robustness analysis methods included within this package

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.

---

oat_plotResultDistribution

For stochastic simulations plots the distribution of results for each parameter value

---

Description

Only applicable for stochastic simulations where the results are provided in the folder structure: this takes each parameter in turn, and creates a boxplot for each output measure, showing the result distribution for each value of that parameter.

Usage

```
oat_plotResultDistribution(FILEPATH, PARAMETERS, MEASURES, MEASURE_SCALE,
  CSV_FILE_NAME, 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 for which local analyses are being conducted
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
MEASURE_SCALE	An array containing the measure used for each of the output measures (i.e. microns, microns/min). Used to label graphs
CSV_FILE_NAME	Name of the file created that summarises the median value of each measure for every run. This specifies what that file should be called (e.g. Medians.csv).
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 minimum value that should be used for each parameter. Sets a lower bound on sampling space
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
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"

---

oat_processParamSubsets

Summarises stochastic, repeated, simulations for all robustness parameter sets into a single file.

Description

This method should only be used for stochastic simulations where the data is provided in the set folder structure (see the spartan R Journal paper). 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.

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)
```

Arguments

FILEPATH	Directory where either the simulation runs or single CSV file result can be found
PARAMETERS	Array containing the names of the parameters for which local analyses are being conducted
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. In the current version, XML and CSV files can be processed. 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.
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.
OUTPUTCOLEND	Column number in the simulation results file where the last output measure is. Only required if running the first method.
CSV_FILE_NAME	Name of the file created that summarises the median value of each measure for every run. This specifies what that file should be called (e.g. Medians.csv).
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 minimum value that should be used for each parameter. Sets a lower bound on sampling space
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
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"

`oat_process_netlogo_result`

Takes a Netlogo behaviour space file and performs a robustness analysis from that simulation data

---

Description

From a Netlogo behaviour space file, extracts the required timepoint information from it, storing this in a Spartan compatible CSV file. This CSV file is then processed using the methods described in `oat_csv_result_file_analysis`, 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 to graph the results.

Usage

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

Arguments

FILEPATH	Location where the behaviour space results can be found
NETLOGO_BEHAVIOURSPACEFILE	The name of the file produced by Netlogo for Parameter Robustness (Technique 2). This is the result file that is analysed.
PARAMETERS	Array containing the names of the parameters for which parameter samples were be generated
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.
MEASURES	Array containing the names of the Netlogo output measures which are used to analyse the simulation.
RESULTFILENAME	Name of the results summary file that should be produced when analysing the Netlogo results
ATESTRESULTSFILENAME	File name of the ATests result summary file created by `oat_analyseAllParams`
TIMESTEP	The timestep of the Netlogo simulation being analysed.

partition_dataset *Partition latin-hypercube summary file to training, testing, and validation*

Description

Used in the development of emulations of a simulation using a latin-hypercube summary file

Usage

```
partition_dataset(dataset, parameters, percent_train = 75,
  percent_test = 15, percent_validation = 10, seed = NULL,
  normalise = FALSE, sample_mins = NULL, sample_maxes = NULL,
  timepoint = NULL)
```

Arguments

dataset	LHC summary file to partition
parameters	Simulation parameters the emulation will be fed as input
percent_train	Percent of the dataset to use as training
percent_test	Percent of the dataset to use as testing
percent_validation	Percent of the dataset to use as validation
seed	For specifying a particular seed when randomly splitting the set
normalise	Whether the data needs to be normalised (to be between 0 and 1). For emulation creation to be successful, all data must be normalised prior to use in training and testing
sample_mins	The minimum value used for each parameter in generating the latin-hypercube sample
sample_maxes	The maximum value used for each parameter in generating the latin-hypercube sample
timepoint	Simulation timepoint for which this summary file was created

Value

Partitioned dataset containing training, testing, and validation sets, in addition to the sample mins and maxes such that any predictions that are generated using this normalised data can be rescaled correctly

perform_aTest_for_all_sim_measures

Performs A-Test to compare all simulation measures

Description

for this set, the simulation baseline behaviour, and performs the A-Test to get a comparison for all simulation measures

Usage

```
perform_aTest_for_all_sim_measures(PARAMETER_SET, BASELINE_RESULT,
  PARAMETER_SET_RESULT, MEASURES)
```

Arguments

PARAMETER_SET	Set of simulation parameters for which a set of runs was performed
BASELINE_RESULT	Simulation behaviour under baseline conditions
PARAMETER_SET_RESULT	Simulation behaviour under conditions in this parameter set
MEASURES	An array containing the names of the simulation output measures to be analysed.
	@export

plotATestsFromTimepointFiles

Plots the A-Tests for all timepoints being examined

Description

When plotting a time-course analysis, it may be useful to compare results gained at multiple time-points, and determine the differences in performance over time. This function provides a means of plotting those results

Usage

```
plotATestsFromTimepointFiles(FILEPATH, PARAMETERS, ATESTRESULTFILENAME,
  ATESTSIGLEVEL, MEASURES, PMIN, PMAX, PINC, TIMEPOINTS)
```

Arguments

FILEPATH	Directory where the results of the simulation runs, in folders or in single CSV file format, can be found
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
ATESTRESULTFILENAME	Name of the CSV file containing the A-Test results to be plotted
ATESTSIGLEVEL	Value to plot for a large difference between distributions on this plot
MEASURES	An array containing the names of the simulation output measures to be analysed.
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
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)

---

ploteFASTSiFromTimepointFiles

Plot the Si value for all parameters for multiple simulation timepoints

---

Description

Permits easy comparison of when a parameter may become more influential than others throughout a simulation timecourse

Usage

```
ploteFASTSiFromTimepointFiles(FILEPATH, PARAMETERS, MEASURES,
  EFASTRESULTFILENAME, TIMEPOINTS, TIMEPOINTSCALE)
```

Arguments

FILEPATH	Where the eFAST results have been stored
PARAMETERS	Names of simulation parameters being explored
MEASURES	Names of simulation output responses
EFASTRESULTFILENAME	Name of the CSV file output by eFAST Analysis, containing all the Si and STi values
TIMEPOINTS	Timepoints to include in this analysis
TIMEPOINTSCALE	Scale in which the timepoints are measured

plotPRCCSFromTimepointFiles

Plots Graphs for Partial Rank Correlation Coefficients Over Time

Description

Produces plots to show how the impact of a parameter changes over time, measured by the change in PRCC

Usage

```
plotPRCCSFromTimepointFiles(FILEPATH, PARAMETERS, MEASURES, CORCOEFFSFILENAME,
  TIMEPOINTS, TIMEPOINTSCALE, DISPLAYPVALS = FALSE)
```

Arguments

FILEPATH	Directory where the simulation runs of single CSV file can be found
PARAMETERS	Array containing the names of the parameters of which parameter samples will be generated
MEASURES	Array containing the names of the output measures which are used to analyse the simulation
CORCOEFFSFILENAME	Name of the CSV file containining the correlation coefficients
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"
DISPLAYPVALS	Boolean stating whether PRCC p-values should be printed on the graph

plot_compare_sim_observed_to_model_prediction

Internal function used to create accuracy plots of the emulation against observed data

Description

Outputs plot to PDF in the current working directory

Usage

```
plot_compare_sim_observed_to_model_prediction(observed, predicted, technique,
  measure, mse, graph_file_name, timepoint = NULL)
```

Arguments

observed	Observed dataset (testing or validation)
predicted	Predicted dataset
technique	The machine learning technique used to develop the emulator
measure	The simulation output response being plotted
mse	Mean Squared Error between predicted and observed
graph_file_name	Name to give the produced PDF plot
timepoint	If using multiple timepoints, the timepoint for which the emulator has been created

---

produce_accuracy_plots_all_measures

Internal function used to create accuracy plots of the emulation against observed data, for all measures

---

Description

Outputs plot to PDF in the current working directory

Usage

```
produce_accuracy_plots_all_measures(technique, measures, model_predictions,
  observed_data, timepoint = NULL)
```

Arguments

technique	The machine learning technique used to develop the emulator
measures	All simulation output responses to plot
model_predictions	Predicted dataset
observed_data	Observed dataset (testing or validation)
timepoint	If using multiple timepoints, the timepoint for which the emulator has been created

produce_accuracy_plots_single_measure

Internal function used to create accuracy plots of the emulation against observed data

Description

Outputs plot to PDF in the current working directory

Usage

```
produce_accuracy_plots_single_measure(technique, measure, model_predictions,
  observed_data, timepoint = NULL)
```

Arguments

technique	The machine learning technique used to develop the emulator
measure	The simulation output response being plotted
model_predictions	Predicted dataset
observed_data	Observed dataset (testing or validation)
timepoint	If using multiple timepoints, the timepoint for which the emulator has been created

screen_nsga2_parameters

Screens NSGA-2 related parameters, guiding which to select for evolving parameter sets

Description

This method performs a sensitvity analysis of key settings for the nsga2 algorithm. Different values of generation number, crossover and mutation rate are assessed and the values for each objective, along with the variance of the parameter inputs are written out to file in .csv format so the user can assess which settings are best suited to the chosen application. Values for the crossover and mutation distribution indices, used in simulated binary crossover, are left at their default settings, but can be overwritten when running the emulator_parameter_evolution method.

Usage

```
screen_nsga2_parameters(function_to_evaluate, nsga2_user_set_parameters,
  nsga_sensitivity_parameters, nsga2_settings)
```

Arguments

`function_to_evaluate`
: A user-defined function that NSGA2 seeks to minimise

`nsga2_user_set_parameters`
: An object containing the emulator input and output names, the input parameters for function to evaluate, minimum and maximum values for emulator inputs. These should be set using the function that creates that object prior to running this method, nsga2_set_user_params

`nsga_sensitivity_parameters`
: an object containing the minimum and maximum values of generation number, crossover probability and mutation probability to be assessed

`nsga2_settings`
: An object containing the population size, number of generations, crossover probability and mutation probability to be assessed

Value

Output showing the results of this sensitivity analysis for the NSGA-2 parameters

---

selectSuitableStructure

Selects the most suitable neural network structure from the potentials made

---

Description

The user will provide a list of neural network objects to assess. These will be assessed on their performance by mean squared error. This method simply selects the most suitable structure (where MSE is lowest)

Usage

```
selectSuitableStructure(network_errors)
```

Arguments

`network_errors`
: MSE obtained for each measure for all neural network layer structures examined

Value

Lowest error network structure

set.nsga_sensitivity_params

Set parameters for NSGA-2 sensitivity analysis

Description

Establish the parameters for the NSGA-2 sensitivity analysis, creating an object that is used within the method that screens NSGA-2 parameters.

Usage

```
set.nsga_sensitivity_params(generation_min, crossover_min, mutation_min,
  generation_max, crossover_max, mutation_max, seed)
```

Arguments

generation_min	Minimum value for number of generations
crossover_min	Minimum value for crossover
mutation_min	Minimum value for mutation rate
generation_max	Maximum value for number of generations
crossover_max	Maximum value for crossover
mutation_max	Maximum value for mutation rate
seed	Random seed value to use in the algorithm

Value

List of the above, for passing in as a settings object to NSGA-2 related methods

sim_data_for_emulation

Set of parameter and response pairs for training an emulator of a simulation

Description

This dataset contains 500 sets of parameter values and the responses that were observed under those conditions when run through the simulator. This is used as a dataset to show how one could use a set of simulation results to train emulators, that in turn could be combined to form an ensemble.

Usage

```
data(sim_data_for_emulation)
```

Format

A list with 500 rows (one per parameter set) and nine columns

Details

- stableBindProbability. Parameter values between 0 and 100
- chemokineExpressionThreshold. Parameter values between 0 and 1
- initialChemokineExpressionValue. Parameter values between 0.1 and 0.5
- maxChemokineExpressionValue. Parameter values between 0.015 and 0.08
- maxProbabilityOfAdhesion. Parameter values between 0 and 1
- adhesionFactorExpressionSlope. Parameter values between 0.25 and 5
- Velocity. Simulation response measure for cell speed
- Displacement. Simulation response measure for cell displacement
- PatchArea. Simulation response measure for size of formed clusters

---

tutorial_consistency_set

Example dataset showing the structure for consistency analysis data

---

Description

This dataset contains exemplar results from a consistency analysis, that shown in the tutorial. This data can be read in and processed using the functions detailed in the vignette. The important thing to obtain from this object is the structure the data is expected to be in

Usage

```
data(tutorial_consistency_set)
```

Format

A list with 9060 rows and four columns

Details

- SampleSize. Number of simulation executions in each subset
- Set. The number of the subset this result belongs to
- Velocity. The median velocity observed in simulation executions in this set
- Displacement. The median displacement observed in simulation executions in this set

`updateErrorForStructure` *Add the MSE for a newly examined structure to the list of those already seen*

Description

Add the MSE for a newly examined structure to the list of those already seen

Usage

```
updateErrorForStructure(network_ms_errors, network_struc, average_errors,
  measures)
```

Arguments

`network_ms_errors`	Mean Squared errors for a newly evaluated structure
`network_struc`	The network structure evaluated
`average_errors`	The current list of MSEs for all structures being evaluated
`measures`	Names of the simulation responses that form the output node of the neural network

`use_ensemble_to_generate_predictions` *Predict simulation responses for a parameter set using an ensemble*

Description

This takes a set of unseen parameter values and uses an ensemble to make predictions of the responses that the simulator would generate

Usage

```
use_ensemble_to_generate_predictions(generated_ensemble, data_to_predict,
  parameters, measures, normalise_values = FALSE, normalise_result = FALSE)
```

Arguments

`generated_ensemble`	Ensemble to use to make predictions
`data_to_predict`	Parameter sets to make predictions from
`parameters`	Simulation parameter names

measures Simulation output response names

normalise_values
Whether the unseen parameter sets should be normalised to be between 0 and 1

normalise_result
Whether the generated predictions should be normalised to be between 0 and 1

Value

List of predictions made for specified responses for all parameter sets

---

visualise_data_distribution

Used to diagnose skew in a training dataset before use in emulation

---

Description

Useful for determining how useful a simulation dataset is for training the range of emulators available in this package. This is output as a PDF.

Usage

```
visualise_data_distribution(dataset, measure, graphname, num_bins = 30)
```

Arguments

dataset	Dataset being visualised
measure	Simulation response measure to visualise
graphname	Name of the graph to produce (as a PDF)
num_bins	Number of bins to use in the histogram

---

weight_emulator_predictions_by_ensemble

Internal function to weight emulator predictions by that calculated for the ensemble

---

Description

Internal function to weight emulator predictions by that calculated for the ensemble

Usage

```
weight_emulator_predictions_by_ensemble(model_weights, all_model_predictions,
  measures, num_generations = 8e+05)
```

Arguments

`model_weights`	Weights for all emulators in the ensemble, based on test set performance
`all_model_predictions`	Set of test set predictions obtained for all emulators in the ensemble
`measures`	Simulation responses the model should predict
`num_generations`	Number of generations for which the neural network that is generating the weights should attempt to converge within

Value

predictions generated by the ensemble

Index