NAME

simulation::montecarlo − Monte Carlo simulations

SYNOPSIS

package require **Tcl ?8.4?**

package require **simulation::montecarlo 0.1**

package require **simulation::random**

package require **math::statistics**

::simulation::montecarlo::getOption *keyword*

::simulation::montecarlo::hasOption *keyword*

::simulation::montecarlo::setOption *keyword value*

::simulation::montecarlo::setTrialResult *values*

::simulation::montecarlo::setExpResult *values*

::simulation::montecarlo::getTrialResults

::simulation::montecarlo::getExpResult

::simulation::montecarlo::transposeData *values*

::simulation::montecarlo::integral2D ...

::simulation::montecarlo::singleExperiment *args*

DESCRIPTION

The technique of *Monte Carlo simulations* is basically simple:

- generate random values for one or more parameters.
- evaluate the model of some system you are interested in and record the interesting results for each realisation of these parameters.
- after a suitable number of such trials, deduce an overall characteristic of the model.

You can think of a model of a network of computers, an ecosystem of some kind or in fact anything that can be quantitatively described and has some stochastic element in it.

The package *simulation::montecarlo* offers a basic framework for such a modelling technique:

```
#
# MC experiments:
# Determine the mean and median of a set of points and compare them
#
::simulation::montecarlo::singleExperiment -init {
   package require math::statistics

   set prng [::simulation::random::prng_Normal 0.0 1.0]
} -loop {
   set numbers {}
   for { set i 0 } { $i < [getOption samples] } { incr i } {
```

```
        lappend numbers [$prng]
      }
      set mean   [::math::statistics::mean $numbers]
      set median [::math::statistics::median $numbers] ;# ? Exists?
      setTrialResult [list $mean $median]
    } -final {
      set result [getTrialResults]
      set means   {}
      set medians {}
      foreach r $result {
        foreach {m M} $r break
        lappend means   $m
        lappend medians $M
      }
      puts [getOption reportfile] "Correlation: [::math::statistics::corr $means $medians]"

    } -trials 100 -samples 10 -verbose 1 -columns {Mean Median}
```

This example attemps to find out how well the median value and the mean value of a random set of numbers correlate. Sometimes a median value is a more robust characteristic than a mean value - especially if you have a statistical distribution with "fat" tails.

PROCEDURES

The package defines the following auxiliary procedures:

::simulation::montecarlo::getOption *keyword*

Get the value of an option given as part of the *singeExperiment* command.

string *keyword*

Given keyword (without leading minus)

::simulation::montecarlo::hasOption *keyword*

Returns 1 if the option is available, 0 if not.

string *keyword*

Given keyword (without leading minus)

::simulation::montecarlo::setOption *keyword value*

Set the value of the given option.

string *keyword*

Given keyword (without leading minus)

string *value*

(New) value for the option

::simulation::montecarlo::setTrialResult *values*

Store the results of the trial for later analysis

list *values*

List of values to be stored

::simulation::montecarlo::setExpResult *values*

Set the results of the entire experiment (typically used in the final phase).

list *values*

List of values to be stored

::simulation::montecarlo::getTrialResults
Get the results of all individual trials for analysis (typically used in the final phase or after completion of the command).

::simulation::montecarlo::getExpResult
Get the results of the entire experiment (typically used in the final phase or even after completion of the *singleExperiment* command).

::simulation::montecarlo::transposeData *values*
Interchange columns and rows of a list of lists and return the result.

list *values*
List of lists of values

There are two main procedures: *integral2D* and *singleExperiment*.

::simulation::montecarlo::integral2D ...
Integrate a function over a two-dimensional region using a Monte Carlo approach.

Arguments PM

::simulation::montecarlo::singleExperiment *args*
Iterate code over a number of trials and store the results. The iteration is gouverned by parameters given via a list of keyword-value pairs.

int *n* List of keyword-value pairs, all of which are available during the execution via the *getOption* command.

The *singleExperiment* command predefines the following options:

- -*init code*: code to be run at start up
- -*loop body*: body of code that defines the computation to be run time and again. The code should use *setTrialResult* to store the results of each trial (typically a list of numbers, but the interpretation is up to the implementation). Note: Required keyword.
- -*final code*: code to be run at the end
- -*trials n*: number of trials in the experiment (required)
- -*reportfile file*: opened file to send the output to (default: stdout)
- -*verbose*: write the intermediate results (1) or not (0) (default: 0)
- -*analysis proc*: either "none" (no automatic analysis), standard (basic statistics of the trial results and a correlation matrix) or the name of a procedure that will take care of the analysis.
- -*columns list*: list of column names, useful for verbose output and the analysis

Any other options can be used via the getOption procedure in the body.

TIPS

The procedure *singleExperiment* works by constructing a temporary procedure that does the actual work. It loops for the given number of trials.

As it constructs a temporary procedure, local variables defined at the start continue to exist in the loop.

KEYWORDS

math, montecarlo simulation, stochastic modelling

CATEGORY

Mathematics

COPYRIGHT