Package ‘pse’

June 11, 2017

Type Package

Title Parameter Space Exploration with Latin Hypercubes

Version 0.4.7

Date 2017-06-11

Author Andre Chalom, Paulo Inacio Knegt Lopez de Prado

Maintainer Andre Chalom <andrechalom@gmail.com>

Depends R (>= 3.0.1), Hmisc

Imports utils, graphics, stats, boot, parallel

Suggests sensitivity, mcmc

Description Functions for creating Latin Hypercubes with
prescribed correlations and performing parameter space exploration.
Also implements the PLUE method.
Based on the package sensitivity, by Gilles Pujol,
Bertrand Iooss & Alexandre Janon.

License GPL-3

URL https://github.com/andrechalom/pse

BugReports https://github.com/andrechalom/pse

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-06-11 21:50:30 UTC

R topics documented:

`cv` *Coefficient of Variation*

Description

Returns the coefficient of variation of a sample.

Usage

```
cv(x, ...)
```

Arguments

`x`	Any numeric vector (or other data type for which `sd` and `mean` methods are defined)
`...`	Additional parameters for the `sd` and `mean` functions (such as `na.rm=TRUE`)

`get.results` *LHS Accessor Functions.*

Description

Instead of using the $ operator, using these accessor functions is the preferred method for accessing the data and result data frames from an `LHS` or `PLUE` object, as the internal structure of the object may vary between versions of the package.

Usage

```
get.results(obj, get.mean = TRUE)

get.data(obj)

get.N(obj)

get.ninputs(obj)

get.noutputs(obj)

get.repetitions(obj)
```

Arguments

`obj`	The LHS or PLUE object
`get.mean`	In case of stochastic models, when several model runs are required for the same data point, the `data` slot of the LHS object contains all the model outputs. Use `get.mean=TRUE` to get the average values for each point, or `get.mean=FALSE` to get all the results.

Details

`get.data` returns a data.frame consisting on the input data.

`get.results` returns an array with the model results. See the vignette on multiple runs for details on the `get.mean` argument.

`get.N`, `get.ninputs`, `get.noutputs` return a single number each, with the number of points in the hypercube (or sampling), number of input factors and number of response variables.

`get.repetitions` returns the number of model repetitions for each data point, created by `LHS(model, factors, N, repeti` or by `telling` several result sets to the same LHS object (or PLUE object).

`LHS` *Latin Hypercube Sampling for Uncertainty and Sensitivity Analyses*

Description

Generates the Latin Hypercube sampling for uncertainty and sensitivity analyses.

Usage

```
LHS(model = NULL, factors, N, q = NULL, q.arg = NULL, res.names = NULL,
  method = c("HL", "random"), opts = list(), nboot = 0, repetitions = 1,
  cl = NULL)

## S3 method for class 'LHS'
print(x, ...)

tell(x, y = NULL, ...)

## S3 method for class 'LHS'
tell(x, y, res.names = NULL, nboot = 0, ...)
```

Arguments

`model`	The function to be run, representing the model or simulation. If `NULL`, no function is run and the object generated is incomplete, see also the `tell` method.
`factors`	The names of the input variables (used for naming the 'data' data.frame and in plotting) Either a vector of strings or a single number representing the number of factors

`N`	The size of the hypercube, i.e., how many samples are generated. Must be at least the number of factors plus 2.
`q`	The quantile functions to be used. If only one is provided, it will be used for all parameters. Defaults to "qunif".
`q.arg`	A list containing the arguments for the 'q' functions. Each parameter must be specified by a named list, containing all of the arguments for the quantile distribution. If unsupplied, default values for the parameters are used.
`res.names`	Optional: what are the names of the model results? (Used mainly for plotting)
`method`	Currently, two methods are supported. "random" generates a simple LH, with no modifications. "HL" (the default) generates a random LH, and subsequently corrects the correlation matrix using the Huntington & Lyrintzis method.
`opts`	Further options for the method used. The method HL supports the following options: 'COR' The desired correlation matrix between the model variables. If none is provided, the function will generate a zero-correlation Latin Hypercube. 'eps' The tolerance between the prescribed correlation and the actual correlation present in the generated Latin Hypercube. 'maxIt' The maximum number of iterations to be run for each factor. The default is set by a heuristic, but it might need some adjustments.
`nboot`	Number of bootstrap replicates for calculating the PRCC.
`repetitions`	The number of model repetitions to be run for a single data point. See the vignette on stochastic models for details
`cl`	Cluster generated with the "parallel" library. May be of any type supported. If a cluster is provided, the model will be run in parallel or distributed across the cluster via clusterApply. No load balancing is provided, so the model results are reproducible. NOTE: You should manually export ALL objects required for the model to run, including the model function itself. See the help on `clusterExport` on package `parallel` for details.
`x`	An LHS/PLUE object. For "tell", an incomplete LHS object (created with model=NULL)
`...`	Currently ignored
`y`	A data.frame containing the model responses

Details

A Latin Hypercube of size `N` is generated from the desired quantile distribution functions

The following methods are currently supported for generating the LHS: random LHS and Huntington & Lyrintzis method for correcting the correlation matrix to be similar to the prescribed by the option `COR` (see the arguments for description).

The specified `model` is run with the data from the LHS. If `repetitions` is set to more than one, the model will be run several times for each data point.

Partial rank correlation coefficients are estimated using code based on the `prcc` function from the "sensitivity" package.

When the LHS function is called with no model (i.e., with argument `model=NULL`), it generates an incomplete object storing the Latin Hypercube samples, and allowing the user to run the simulation model independently. The method `tell` allows to pass the simulation results to the incomplete object.

`tell` and `ask` are S3 generic methods for decoupling simulations and sensitivity measures estimations in the package 'sensitivity'. In general, they are not used by the end-user for a simple R model, but rather for an external computational code. The LHS object implements only the `tell` method. For help on the other methods, see the help pages on the 'sensitivity' package.

Warning

NOTE: the tell method from sensitivity objects (like 'fast99') modifies the object passed as argument as a side effect. This is NOT the case with the LHS tell method.

Author(s)

Andre Chalom

Source

Uses internal code originally published on package sensitivity, by Gilles Pujol, Bertrand Iooss, Alexandre Janon

References

McKay, M.D. and Beckman, R.J. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics* 21: 239-244

Chalom, A. and Prado, P.I.K.L. 2012. Parameter space exploration of ecological models *arXiv*:1210.6278 [q-bio.QM]

Examples

```
completeLHS <- LHS(model=function(x) x[,1]+x[,2]*x[,3], factors=3, N=20)
incompleteLHS <- LHS(factors=5, N=30)
incompleteLHS <- tell(incompleteLHS, seq(1,30))

## Not run:
new.cluster <- parallel::makePSOCKcluster(c("localhost", "localhost"))
clusterLHS <- LHS(model=function(x) x[,1]/x[,2], factors=2, N=100, cl = new.cluster)
stopCluster(new.cluster)

## End(Not run)
```

LHScorcorr *Correlation Matrix Correction*

Description

Corrects the correlation matrix of a given Latin Hypercube Sample.

Usage

```
LHScorcorr(vars, COR = 0, method = c("Pearson", "Spearman"), eps = 0.005,
  echo = FALSE, maxIt = 0)
```

Arguments

`vars`	The data.frame or matrix containing the parameters from the "raw" Latin Hypercube Sample. Each column corresponds to one variable, and each line to one observation.
`COR`	The desired correlation matrix. The default is to have 0 correlation. You can supply a numeric square matrix with M rows, where M is the number of input factors. The *lower* triangular part of the matrix will be used as the desired correlation matrix.
`method`	A character string, which may be "Spearman" or "Pearson", indicating the correlation method to be used.
`eps`	The tolerance for the deviation between the prescribed correlation matrix and the result.
`echo`	Set to true to display information messages.
`maxIt`	Maximum number of iterations before giving up. Set to 0 to use a heuristic based on the size of the hypercube. Set to a negative number to never give up. *CAUTION*, this might result in an infinite loop.

Details

This function changes the order in which data is organized in order to force the correlation matrix to a prescribed value. This implementation uses the Hungtington-Lyrintzis algorithm.

This is mainly intended for use inside of the `LHS` function.

If you intend to use non-zero correlation terms, read Chalom & Prado (2012) for some important theoretical restrictions.

The correlation matrix may be specified by a Pearson or Spearman method. In order to generate the Spearman correlation, the function "rank transforms" the data using the `order` function, and thus works only if there are no ties in the data.

Value

A data.frame containing the same variables, but with the correlation matrix corrected.

References

Huntington, D.E. and Lyrintzis, C.S. 1998 Improvements to and limitations of Latin hypercube sampling. *Prob. Engng. Mech.* 13(4): 245-253.

Chalom, A. and Prado, P.I.K.L. 2012. Parameter space exploration of ecological models *arXiv*:1210.6278 [q-bio.QM]

`machinefile` *Convenience for Clustering*

Description

Provides a convenience interface for using MPD-style hostfiles to generate cluster objetcs. The hostfile should be written as a text file using the MPD style: one line for each host, which can be followed by a colon and a number indicating the number of processes to be started on that host. An example hostfile for starting three processes on two hosts named avalon and glastonbury would be:

Usage

```
machinefile(name)
```

Arguments

`name`	Filename of the hostfile.

Details

avalon glastonbury:2

Examples

```
## Not run:
library(parallel)
cl = makePSOCKcluster(machinefile("mpd.hosts"))
stopCluster(cl)

## End(Not run)
```

pic — *Partial Inclination Coefficient Estimates the partial inclination coefficient of a model response in relation with all model input variables.*

Description

Partial Inclination Coefficient Estimates the partial inclination coefficient of a model response in relation with all model input variables.

Usage

```
pic(X, y, nboot, conf, ...)

## S3 method for class 'LHS'
pic(X, y = NULL, nboot = 0, conf = 0.95, ...)

## Default S3 method:
pic(X, y, nboot = 0, conf = 0.95, ...)

## S3 method for class 'pic'
print(x, ...)
```

Arguments

`X`	A data frame (or object coercible by `as.data.frame`) containing the design of experiments (model input variables).
`y`	A vector containing the responses corresponding to the design of experiments (model output variables).
`nboot`	Number of bootstrap replicates
`conf`	Confidence of the bootstrap interval
`...`	Currently ignored.
`x`	The object returned by `pic`.

Author(s)

Andre Chalom, based on code by Gilles Pujol, Bertrand Iooss, Alexandre Janon

Source

based on code of pcc function on sensitivity package by Gilles Pujol, Bertrand Iooss, Alexandre Janon.

plotcv *Uncertainty and Sensitivity Plots.*

Description

The functions listed here are used in uncertainty and sensitivity estimation.

Usage

```
plotcv(obj, stack = FALSE, index.res = 1:get.noutputs(obj),
  col = index.res, quant = 0.99, ...)

plotecdf(obj, stack = FALSE, index.res = 1:get.noutputs(obj),
  col = index.res, xlab = NULL, ...)

plotprcc(obj, index.res = 1:dim(obj$res)[2], col = "orange", ylab = NULL,
  ...)

plotscatter(obj, res = NULL, index.data = NULL, index.res = NULL,
  add.lm = TRUE, ylab = NULL, ...)
```

Arguments

`obj`	The LHS or PLUE object containing the simulation results to be plotted. NOTICE: plotecdf and plotcv only accept LHS objects! For plotting the likelihood profile from a PLUE object, simply use `plot(obj)`
`stack`	If the results is a data.frame with several variables, `stack=FALSE` generates a series of plots, and `stack=TRUE` generates a single plot with the ECDF from all variables identified by different colors.
`index.res`	An optional vector indicating which columns from the results are to be plotted.
`col`	An optional vector indicating the colors to be used.
`quant`	Maximum quantile to be plotted on the ecdf (used to cut off extreme values in the labels)
`...`	Additional parameters to be passed to the lower level plotting function.
`xlab, ylab`	Labels for the x axis (ecdf) or y axis(prcc). The functions use the name provided in the res.names argument from the LHS function if left blank.
`res`	A data.frame consisting of the model results to be plotted on the y axis, if 'obj' is passed as a data.frame. If 'obj' is an LHS/PLUE object, this parameter is ignored.
`index.data`	An optional vector with the indices of the data columns to be plotted.
`add.lm`	Boolean. Whether to include a simple linear model on the plots. Defaults to TRUE.

Details

The function `plotscatter` produces a series of scatterplots from data.

The function `plotecdf` plots the empirical cumulative density function from an LHS object or PLUE object.

The function `plotprcc` plots the partial rank correlation coefficient from an LHS object or PLUE object.

Finally, the `plotcv` function plots the empirical cummulative density function (ecdf) of the coefficient of variation of the LHS resulting from a stochastic simulation, along with a dotted line representing the coefficient of variation of the whole result set. See the 'multiple' vignette for examples and interpretation.

The function plotscatter accepts an alternative invocation of `plotscatter(obj, res)` in which obj is a data.frame consisting on the data to be plotted on the x axis, and res is a data.frame consisting on the model results to be plotted on the y axis.

Examples

```
myLHS <- LHS(model=function(x) x[,1]+x[,2]*x[,3], factors=3, N=20, res.names="My Output")
plotecdf(myLHS, main="ECDF plot")
plotprcc(myLHS, main="PRCC plot")
plotscatter(myLHS)
```

`PLUE` *Profiled Likelihood Uncertainty Estimation*

Description

Performs a likelihood-based uncertainty estimation on a model. This analysis consists on a Metropolis Monte Carlo exploration of the parameter space and subsequent profiling of model results based on the likelihood of the input parameters.

Usage

```
PLUE(model = NULL, factors, N, LL, start, res.names = NULL,
  method = c("internal", "mcmc"), opts = list(), nboot = 0,
  repetitions = 1, cl = NULL)

## S3 method for class 'PLUE'
print(x, ...)

## S3 method for class 'PLUE'
plot(x, ...)
```

Arguments

`model`	The function to be run, representing the model or simulation.
`factors`	The names of the input variables (used for naming the 'data' data.frame and in plotting) Either a vector of strings or a single number representing the number of factors
`N`	The number of samples to be generated by the Metropolis algorithm.
`LL`	The POSITIVE Likelihood function to be used by the Metropolis algorithm. It must accept an array with length equal to the number of factors.
`start`	The initial point to be evaluated. Must have the same length as the number of factors.
`res.names`	Optional: what are the names of the model results? (Used mainly for plotting)
`method`	May be either "internal", which runs a naive and inneficient algorithm provided for test and didatic purposes, or "mcmc", which will run the `metrop` function from the `mcmc` package.
`opts`	Further options to be passed to the Metropolis function. See the help on `metrop`.
`nboot`	Number of bootstrap replicates for calculating the PRCC.
`repetitions`	The number of model repetitions to be run for a single data point. See the vignette on stochastic models for details
`cl`	Cluster generated with the "parallel" library. May be of any type supported. If a cluster is provided, the model will be run in parallel or distributed across the cluster via clusterApply. No load balancing is provided, so the model results are reproducible. NOTE: You should manually export ALL objects required for the model to run, including the model function itself. See the help on `clusterExport` on package `parallel` for details.
`x`	An LHS/PLUE object. For "tell", an incomplete LHS object (created with model=NULL)
`...`	Currently ignored

Details

A detailed description can be found on Chalom & Prado (2015).

References

Chalom, A. and Prado, P.I.K.L. 2015. Uncertainty analysis and composite hypothesis under the likelihood paradigm. *arXiv*:1508.03354 [q-bio.QM]

`print.pcc` *External functions*

Description

This function is derived from package "sensitivity", please proceed to its help files.

Usage

```
## S3 method for class 'pcc'
print(x, ...)
```

Arguments

`x`	A PCC or PRCC object
`...`	Currently ignored

`sbma` *Calculates the Symetrized Blest Measure of Agreement between two samples*

Description

The Symetrized Blest Measure of Agreement is an alternative measure of rank correlation (similar to Kendall's Tau and Spearman's Rho). This correlation measure is more sensitive to changes in the order of the first elements of a vector (see examples).

Usage

```
sbma(sample1, sample2, absolute = TRUE, ...)

## Default S3 method:
sbma(sample1, sample2, absolute = TRUE, ...)

## S3 method for class 'LHS'
sbma(sample1, sample2, absolute, ...)
```

Arguments

`sample1`	The first vector or LHS object to be compared.
`sample2`	The second vector or LHS object to be compared.
`absolute`	Logical. Should the absolute values of sample1 and sample2 be used in the calculation?
`...`	Additional arguments.

Details

This function calculates the SBMA between two samples or two LHS objects. In the second case, what is compared is the values of the "prcc" component of each Hypercube.

References

Maturi, T.A. and Elsayigh, A. 2010. A comparison of correlation coefficients via a three-step bootstrap approach. *Journal of Mathematics Research* 2(2): 3-10.

Examples

```
# SBMA is only affected by the rank of the values inside each vector
sbma(c(1,2,3,4), c(2,3,4,5))
# Changes in the first positions: high impact on the SBMA
sbma(c(1,2,3,4), c(2,1,3,4))
cor(c(1,2,3,4), c(2,1,3,4), method="spearman")
# Changes in the last positions: low impact on the SBMA
sbma(c(1,2,3,4), c(1,2,4,3))
cor(c(1,2,3,4), c(1,2,4,3), method="spearman")
```

`target.sbma` *Adaptative generation of Latin Hypercubes*

Description

Generates a series of Latin Hypercube Samples for a model until a pair of LHS present a measure of agreement equal to or greater than a specified target.

Usage

```
target.sbma(target, model, factors, q = NULL, q.arg = NULL,
  res.names = NULL, method = c("HL", "random"), opts = list(),
  init = length(factors) + 2, inc = 100, FUN = min)
```

Arguments

`target`	The desired SBMA.
`model`	The function to be run, representing the model or simulation. If `NULL`, no function is run and the object generated is incomplete, see also the `tell` method.
`factors`	The names of the input variables (used for naming the 'data' data.frame and in plotting) Either a vector of strings or a single number representing the number of factors
`q`	The quantile functions to be used. If only one is provided, it will be used for all parameters. Defaults to "qunif".
`q.arg`	A list containing the arguments for the 'q' functions. Each parameter must be specified by a named list, containing all of the arguments for the quantile distribution. If unsupplied, default values for the parameters are used.

`res.names`	Optional: what are the names of the model results? (Used mainly for plotting)
`method`	Currently, two methods are supported. "random" generates a simple LH, with no modifications. "HL" (the default) generates a random LH, and subsequently corrects the correlation matrix using the Huntington & Lyrintzis method.
`opts`	Further options for the method used. The method HL supports the following options: 'COR' The desired correlation matrix between the model variables. If none is provided, the function will generate a zero-correlation Latin Hypercube. 'eps' The tolerance between the prescribed correlation and the actual correlation present in the generated Latin Hypercube. 'maxIt' The maximum number of iterations to be run for each factor. The default is set by a heuristic, but it might need some adjustments.
`init`	The size of the initial LHS generated.
`inc`	The increment between successive runs. For example, if init = 5 and inc = 20, the first LHS will be generated with size 5, the second with size 25.
`FUN`	When the model returns more than one response, SBMA values are calculated for each variable. The FUN argument specifies how to combine these SBMA values. The recommended default is to chose the minimum value.

Value

Returns the largest LHS generated.

Index