Package ‘pse’
February 20, 2015

Type Package
Title Parameter space exploration with Latin Hypercubes
Version 0.4.0
Date 2014-08-02
Author Andre Chalom, Paulo Inacio Knegt Lopez de Prado
Maintainer Andre Chalom <andrechalom@gmail.com>
Depends R (>= 3.0.1), Hmisc, 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
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-12-20 15:46:40

R topics documented:

  cv ............................................................... 2
decoupling ..................................................... 2
get.data .......................................................... 3
LHS ................................................................. 4
LHScorcorr ....................................................... 6
machinefile ...................................................... 7
pic ................................................................. 8
plotcv ............................................................ 8
plotecdf ......................................................... 9
plotscatter ...................................................... 10
PLUE ............................................................. 11
sbma .............................................................. 12
target.sbma ...................................................... 13
Index

| cv  | Coefficient of variation |

Description

Returns the coefficient of variation of a sample.

Usage

\[
\text{cv}(x, \ldots)
\]

Arguments

\begin{itemize}
\item \textit{x} \hspace{1cm} \text{Any numeric vector (or other data type for which \texttt{sd} and \texttt{mean} methods are defined)}
\item \ldots \hspace{1cm} \text{Additional parameters for the \texttt{sd} and \texttt{mean} functions (such as \texttt{na.rm=TRUE})}
\end{itemize}

decoupling

Decoupling Simulations and Estimations

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 \texttt{tell} method. For help on the other methods, see the help pages on the ‘sensitivity’ package.

Usage

\[
tell(x, y = \text{NULL}, \ldots)
\]

Arguments

\begin{itemize}
\item \textit{x} \hspace{1cm} \text{the incomplete LHS object.}
\item \textit{y} \hspace{1cm} \text{a data.frame of model responses.}
\item \ldots \hspace{1cm} \text{currently not supported.}
\end{itemize}

Value

\texttt{tell} returns the LHS object with the results data.frame. \textbf{Note: the methods in ‘sensitivity’ alter the original object as a side effect. This is NOT the case with the pse methods.}

Author(s)

Original code by Gilles Pujol, LHS method implemented by Andre Chalom
get.data

Access the internal data and results data frames

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 object, as the internal structure of the object may vary between versions of the package.

Support for these accessors is being developed for PLUE objects as well, but some bugs may occur.

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.

Value

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, 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, repetitions) or by telling several result sets to the same LHS object.
Latin Hypercube Sampling for uncertainty and sensitivity analyses

Description

Generates the Latin Hypercube sampling for uncertainty and sensitivity analyses. 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.

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.

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'
tell(x, y, res.names = NULL, nboot = 0, ...)

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

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)
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.

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.

Number of bootstrap replicates for calculating the PRCC.

The number of model repetitions to be run for a single data point. See the vignette on stochastic models for details

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.

An incomplete LH object (created with model=NULL)

A data.frame containing the model responses

Currently ignored

An object of the class LHS.

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.

Andre Chalom

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

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

Corrects the correlation matrix of a given Latin Hypercube Sample

Description

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.
Intended for use inside of the LHS function.

Usage

LHScorcorr(vars, COR=0, eps = 0.005, echo=FALSE, maxIt=0)

Arguments

vars  The data.frame 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.
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.

Value

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


Description

Provides a convenience interface for using MPD-style hostfiles to generate cluster objects.

Usage

machinefile(name)

Arguments

name Filename of the hostfile.

Details

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:

avalon glastonbury:2

Examples

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

## End(Not run)
```
### pic

**Partial Inclination Coefficient**

**Description**

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

**Usage**

```r
pic(X, y, nboot, conf, ...)  
## 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
- `x` the object returned by `pcc`
- `...` Currently ignored

**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

**Coefficient of variation plot**

**Description**

The `plotcv` function plots the empirical cumulative 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.
**plotecdf**

**Usage**

```
plotcv(x, stack = FALSE, index.res = 1, col = index.res, ...)
```

**Arguments**

- `x`: An LHS model with more than one repetition.
- `stack`: Warning: not implemented in this version!
- `index.res`: Warning: not implemented in this version!
- `col`: The color of the plot.
- `...`: Other parameters to be passed to the lower level plotting function.

---

**plotecdf**

*Uncertainty and sensitivity plots*

---

**Description**

Plots the empirical cumulative density function or the partial rank correlation coefficient from an LHS object or PLUE object.

**Usage**

```
plotecdf(obj, stack = FALSE, index.res, col, xlab = NULL, ...)
plotprcc(obj, index.res, col, ylab = NULL, ...)
```

**Arguments**

- `obj`: The LHS or PLUE object containing the simulation results to be plotted.
  NOTICE: plotecdf only accepts 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.
- `xlab`, `ylab`: Label for the x axis (ecdf) or y axis(prcc). Uses the name provided in the res.names argument from the LHS function if left blank.
- `...`: Additional parameters to be passed to the lower level plotting function.

**Examples**

```r
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**  
*Produces a series of scatterplots from data*

**Description**

Produces a series of scatterplots from data.

**Usage**

```r
plotscatter(vars, res = NULL, index.data, index.res, add.lm, ylab = NULL, ...)
```

**Arguments**

- `vars`: Either a data.frame consisting of the data to plot on the x axis, or an LHS object, or a PLUE object containing the data to be plot.
- `res`: A data.frame consisting of the model results to be plotted on the y axis, if vars is passed as a data.frame. If vars is an LHS object, this parameter is ignored.
- `index.data`: The indices of the data columns to be plotted. Useful when plotting from LHS objects.
- `index.res`: The indices of the result columns to be plotted. Useful when plotting from LHS objects.
- `add.lm`: Boolean. Whether to include a simple linear model on the plots. Defaults to TRUE.
- `ylab`: Label for the x axis. Uses the name provided in the res.names argument from the LHS function if left blank.
- `...`: Additional parameters to pass to the lower level plotting functions.

**Note**

This function was called `corPlot` in older releases, renamed to `plotscatter` for padronization with other graphical functions. The function `corPlot` was removed in pse 0.4.0.

**Examples**

```r
myLHS <- LHS(model=function(x) x[,1]+x[,2]*x[,3], factors=3, N=20)
plotscatter(myLHS)
```
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.

A detailed description is being prepared for publication.

Usage

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

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 inefficient algorithm provided for test and didactic 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 `mcmc::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 A PLUE object
... Further options

Examples

```r
my.LL <- function(x) {
  dbinom(10, 100, x[1], log=TRUE)
  + dbinom(30, 100, x[2], log=TRUE)
  + dbinom(90, 100, x[3], log=TRUE)
}
plue <- PLUE(model=function(x) x[,1]*x[,2]*x[,3], factors=3, N=10000, LL=my.LL, start=rep(0.5, 3))
print(plue)
plot(plue)
```

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 function calculates the SBMA between two samples

Usage

```r
sbma(sample1, sample2, ...)
```

Arguments

- `sample1` The first vector or LHS object to be compared.
- `sample2` The second vector or LHS object to be compared.
- `...` Additional arguments.

References

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, inc, 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)
- **q**: The quantile functions to be used. If only one is provided, it will be used for all parameters.
- **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.
- **res.names**: Optional: what are the names of the model results? (Used mainly for plotting)
- **method**: The desired method for the generation of the hypercubes. Please refer to the LHS help page.
- **opts**: Additional options for the method chosen. Please refer to the LHS help page.
- **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

cv, 2

decoupling, 2

get.data, 3
get.N(get.data), 3
get.ninputs (get.data), 3
get.noutputs (get.data), 3
get.repetitions (get.data), 3
get.results (get.data), 3

LHS, 4
LHSScorr, 6

machinefile, 7

pic, 8
plot.PLUE (PLUE), 11
plotcv, 8
plotcdf, 9
plotprcc (plotcdf), 9
plotsscatter, 10
PLUE, 11
print.LHS (LHS), 4
print.pic (pic), 8
print.PLUE (PLUE), 11

sbma, 12

target.sbma, 13
tell (decoupling), 2
tell.LHS (LHS), 4