Package 'ATmet'

May 6, 2020

Title Advanced Tools for Metrology

Version 1.2.1

Author Severine Demeyer and Alexandre Allard, with contributions from Bertrand Iooss

Maintainer Alexandre Allard <alexandre.allard@lne.fr>

Depends R (>= 2.7.0), DiceDesign, lhs, metRology, msm, sensitivity

Description A collection of functions for smart sampling and sensitivity analysis for metrology applications, including computationally expensive problems.

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2020-05-06 16:00:07 UTC

R topics documented:

`ATmet-package` *Advanced Tools for Metrology*

Description

Several functions for smart sampling and sensitivity analysis for metrology applications, including computationally expensive problems.

Details

The **ATmet** package implements sensitivity analysis functions for metrology applications

The function for smart sampling implements the Latin Hypercube Sampling (LHS) method using the 'lhs' package. The functions for sensitivity analysis implement the Standardized Rank Regression Coefficient (SRRC) and the Sobol' sensitivity indices using the 'sensitivity' package. These methods can be used for computationally expensive problems.

Note

This work is part of a joint research project within the European Metrology Research Programme (EMRP) called "Novel Mathematical and Statistical Approaches to Uncertainty Evaluation". The EMRP is jointly funded by the EMRP participating countries within EURAMET and the European Union.

Author(s)

Severine Demeyer and Alexandre Allard, with contributions from Bertrand Iooss

Maintainer: Alexandre Allard <alexandre.allard@lne.fr>

References

I.M. Sobol, S. Tarantola, D. Gatelli, S.S. Kucherenko and W. Mauntz, 2007, *Estimating the approximation errors when fixing unessential factors in global sensitivity analysis*, Reliability Engineering and System Safety, 92, 957-960.

A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto and S. Tarantola, 2010, *Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index*, Computer Physics Communications 181, 259-270.

R. Stocki, 2005, *A method to improve design reliability using optimal Latin hypercube sampling*, Computer Assisted Mechanics and Engineering Sciences 12, 87-105.

See Also

lhs

sensitivity

Examples

```
# **********************
# Smart sampling method
# **********************


N<- 100
  k<- 4
  x<- list("X1","X2","X3","X4")
  distrib<- list("norm","norm","unif","t.scaled")
  distrib.pars<- list(list(0,2),list(0,1),list(20,150),list(2,0,1))
```

```
  LHSdesign(N,k,distrib,distrib.pars,x)


# **********************
# Sensitivity analysis
# **********************

##Simulate the input sample
M=10000
Xmass <- data.frame(X1 = rnorm(M, 100, 5e-5),
                    X2 = rnorm(M, 0.001234, 2e-5),
                    X3 = runif(M, 1100, 1300),
                    X4 = runif(M,7000000,9000000),
                    X5 = runif(M,7950000,8050000))#Data-frame

#Define the measurement model (GUM-S1, 9.3)
calibMass <-function(x){
  return(((x[,1]+x[,2])*(1+(x[,3]-1200)*(1/x[,4]-1/x[,5]))-100)*1e3)
}

#####   Use SRRC with a model function    #####
#Apply sensitivityMet function to evaluate the associated SRRC indices
S_SRRC=sensitivityMet(model=calibMass,x=Xmass, nboot=100, method="SRRC", conf=0.95)
##Print the results
#First order indices
S_SRRC$S1

#####   Use Sobol with a computational code   #####
#Creation of the design for the computation of Sobol sensitivity indices
S_Sobol=sensitivityMet(model=NULL,x=Xmass,y=NULL, nboot=100, method="Sobol", conf=0.95)

#Obtain the design of experiment to submit to the code
XDesign=S_Sobol$SI$X

#Run the computational code with XDesign as a sample of the input quantities
#We use calibMass function (see GUM-S1) as an example
YDesign=calibMass(XDesign)

#Run the Sobol indices calculations with the outputs of the code
S_Sobol$SI=tell(x=S_Sobol$SI,y=YDesign)

##Print the results
#First order indices
S_Sobol$SI$S
#Total order indices
S_Sobol$SI$T
```

LHSdesign *Latin hypercube sampling for metrology applications*

Description

Creates latin hypercube sampling designs for metrology applications

Usage

```
LHSdesign(N,k,distrib,distrib.pars,x)
```

Arguments

N	The number of design points.
k	The number of the input variables of the numerical code.
distrib	A named list of length k of names of distribution functions associated with the input variables of the code. See Details for defaults.
distrib.pars	A named list of lists of parameters describing the distributions associated with distrib. If distrib is present but distrib.pars is not the function uses the standardized versions of the distributions, see Details.
x	A named list containing the names of the input variables of the numerical code. See Details for defaults.

Details

This function contains a wrapper for the optimumLHS function in package lhs.

If distrib or members of it are missing, an error message is displayed. Distributions have to be chosen among uniform(unif), triangular(triang), normal(norm), truncated normal(tnorm), student(t), location-scale student(t.scaled).

If distrib.pars is missing or misspecified, the standardized parameters of the associated distributions in distrib are used for all the variables in x:

unif : min=0, max=1

triang : min=0, max=1, mode=0.5

norm : mean=0, sd=1

tnorm : mean=0, sd=1, lower=0, upper=+Inf

t : nu=100

t.scaled : nu=100, mean=0, sd=1

If x or members of it are missing, arbitrary names of the form 'Xn' are applied to the columns of the output table. Names are automatically abbreviated to 15 characters.

Value

design.unif A table containing the LHS design in the uniform space.

design.phys A table containing the LHS design with margins in distrib.

Author(s)

Severine Demeyer <severine.demeyer@lne.fr>

Examples

```
N<- 100
k<- 4
x<- list("X1","X2","X3","X4")
distrib<- list("norm","norm","unif","t.scaled")
distrib.pars<- list(list(0,2),list(0,1),list(20,150),list(2,0,1))
LHSdesign(N,k,distrib,distrib.pars,x)
```

`MCdesign` *Monte Carlo sampling for metrology applications*

Description

Creates Monte Carlo sampling designs for metrology applications

Usage

```
MCdesign(N,k,distrib,distrib.pars,x)
```

Arguments

`N`	The number of design points.
`k`	The number of the input variables of the numerical code.
`distrib`	A named list of length `k` of names of distribution functions associated with the input variables of the code. See Details for defaults.
`distrib.pars`	A named list of lists of parameters describing the distributions associated with `distrib`. If `distrib` is present but `distrib.pars` is not the function uses the standardized versions of the distributions, see Details.
`x`	A named list containing the names of the input variables of the numerical code. See Details for defaults.

Details

This function creates a sampling design based on a Monte Carlo simulation.

If `distrib` or members of it are missing, an error message is displayed. Distributions have to be chosen among uniform(`unif`), triangular(`triang`), normal(`norm`), truncated normal(`tnorm`), student(`t`), location-scale student(`t.scaled`).

If `distrib.pars` is missing or misspecified, the standardized parameters of the associated distributions in `distrib` are used for all the variables in `x`:

unif : `min=0`, `max=1`

triang : `min=0`, `max=1`, `mode=0.5`

norm : `mean=0`, `sd=1`

tnorm : mean=0, sd=1, lower=0, upper=+Inf

t : nu=100

t.scaled : nu=100, mean=0, sd=1

If x or members of it are missing, arbitrary names of the form 'Xn' are applied to the columns of the output table. Names are automatically abbreviated to 15 characters.

Value

A table containing the MC design with margins in distrib.

Author(s)

Severine Demeyer <severine.demeyer@lne.fr>

Examples

```
N<- 100
k<- 4
x<- list("X1","X2","X3","X4")
distrib<- list("norm","norm","unif","t.scaled")
distrib.pars<- list(list(0,2),list(0,1),list(20,150),list(2,0,1))
MCdesign(N,k,distrib,distrib.pars,x)
```

---

sensitivityMet *Sensitivity analysis for metrology applications*

Description

Performs a sensitivity analysis for metrology applications

Usage

```
sensitivityMet(model, x, y, nboot, method,conf)
```

Arguments

model	a function representing a measurement model with an explicit mathemtical expression
x	a data frame that contains the input sample.
y	a vector of model responses.
nboot	an integer that denotes the number of bootstrap replicates.
method	a method for the evaluation of the sensitivity. Two methods are currently supported by sensitivityMet : "SRRC" for the standardized Rank Regression Coefficient and "Sobol" for the Sobol indices.
conf	the confidence level of the bootstrap confidence intervals

Details

If 'method = "SRRC"', the function uses 'src' function from the package 'sensitivity', with the option 'rank=TRUE' to compute SRRC sensitivity indices. This method needs the specification of both the input sample 'X' and either the vector of model responses 'y' either the measurement model as a R function in 'model'.

If 'method = "Sobol"', the function uses 'sobol2007' function from the package 'sensitivity'. The input sample 'x' is divided into two samples of equal dimensions. 'Xdesign' is returned, containing the design for the computation of Sobol indices. The user should evaluate 'Xdesign' with the computational code and provide the corresponding output values using 'tell' function. Details on the computation of Sobol indices are given in Sobol et al. (2007).

Both methods are applicable whether the measurement model is an explicit function (defined in 'model' as a R function) or an external code.

The argument 'nboot' is required in order to evaluate a confidence interval with a specified confidence level for the sensitivity indices. Default is 0 : in this case no bootstrap replicates and no confidence intervals are computed.

The argument 'conf' defines the confidence level for the bootstrap confidence intervals. Default is 0.95.

Value

'sensitivityMet' returns a list with the following components :

model	a function representing a measurement model with an explicit mathemtical expression
method	The method used for the evaluation of the sensitivity indices

If 'method = "SRRC"' :

x	a data frame that contains the random sample of the input quantities.
y	a vector of model responses.
S1	a data frame wihch summarizes the first order sensitivity indices obtained with the SRRC.

If 'method = "Sobol"' :

SI	an object of class 'sobol2007' for the computation of the Sobol sensitivity indices with the following list of components :
model	a function representing a measurement model with an explicit mathemtical expression
X1	the first random sample
X2	the second random sample
nboot	the number of bottstrap replicates
conf	the confidence level for bootstrap confidence intervals
X	a 'data.frame' containing the design of experiment
call	the matched call of the function 'sobol2007'

y	the response used
V	the estimations of Variances of the Conditional Expectations (VCE) with respect to each factor and also with respect to the complementary set of each factor ("all but Xi")
S	the estimations of the Sobol first order indices
T	the estimations of the Sobol total sensitivity indices

Author(s)

Alexandre Allard <alexandre.allard@lne.fr>

References

I.M. Sobol, S. Tarantola, D. Gatelli, S.S. Kucherenko and W. Mauntz, 2007, Estimating the approximation errors when fixing unessential factors in global sensitivity analysis, Reliability Engineering and System Safety, 92, 957-960.

A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto and S. Tarantola, 2010, Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index, Computer Physics Communications 181, 259-270.

Examples

```
rm(list=ls())

##Simulate the input sample
M=10000
Xmass <- data.frame(X1 = rnorm(M, 100, 5e-5),
                    X2 = rnorm(M, 0.001234, 2e-5),
                    X3 = runif(M, 1100, 1300),
                    X4 = runif(M,7000000,9000000),
                    X5 = runif(M,7950000,8050000))#Data-frame

#Define the measurement model (GUM-S1, 9.3)
calibMass <-function(x){
  return(((x[,1]+x[,2])*(1+(x[,3]-1200)*(1/x[,4]-1/x[,5]))-100)*1e3)
}

#####   Use SRRC with a model function    #####
#Apply sensitivityMet function to evaluate the associated SRRC indices
S_SRRC=sensitivityMet(model=calibMass,x=Xmass, nboot=100, method="SRRC", conf=0.95)
##Print the results
#First order indices
S_SRRC$S1

#####   Use Sobol with a computational code   #####
#Creation of the design for the computation of Sobol sensitivity indices
S_Sobol=sensitivityMet(model=NULL,x=Xmass,y=NULL, nboot=100, method="Sobol", conf=0.95)

#Obtain the design of experiment to submit to the code
XDesign=S_Sobol$SI$X
```

```
#Run the computational code with XDesign as a sample of the input quantities
#We use calibMass function (see GUM-S1) as an example
YDesign=calibMass(XDesign)

#Run the Sobol indices calculations with the outputs of the code
S_Sobol$SI=tell(x=S_Sobol$SI,y=YDesign)

##Print the results
#First order indices
S_Sobol$SI$S
#Total order indices
S_Sobol$SI$T
```

Index