# Package ‘sensobol’

April 26, 2022

**Title**  Computation of Variance-Based Sensitivity Indices

**Version**  1.1.1

**Maintainer**  Arnald Puy <arnald.puy@pm.me>

**Description**  It allows to rapidly compute, bootstrap and plot up to fourth-order Sobol’-based sensitivity indices using several state-of-the-art first and total-order estimators. Sobol’ indices can be computed either for models that yield a scalar as a model output or for systems of differential equations. The package also provides a suit of benchmark tests functions and several options to obtain publication-ready figures of the model output uncertainty and sensitivity-related analysis. An overview of the package can be found in Puy et al. (2022) <doi:10.18637/jss.v102.i05>.

**License**  GPL-3

**Encoding**  UTF-8

**Imports**  boot (>= 1.3.20), data.table (>= 1.12.0), ggplot2 (>= 3.1.0),
lhs (>= 1.0.2), magrittr (>= 1.5), matrixStats (>= 0.54.0),
randoop (>= 1.17.1), deSolve (>= 1.27.1), Rdpack (>= 2.1.2), Rfast (>= 2.0.1), rngtools (>= 0.3.1), scales (>= 1.0.0),
stats, stringr (>= 1.4.0), utils, Rcpp

**RdMacros**  Rdpack

**Depends**  R (>= 3.5.0)

**RoxygenNote**  7.1.1

**Suggests**  knitr, rmarkdown, testthat (>= 2.1.0), covr

**VignetteBuilder**  knitr

**URL**  https://github.com/arnaldpuy/sensobol

**BugReports**  https://github.com/arnaldpuy/sensobol/issues

**LinkingTo**  Rfast, Rcpp, RcppArmadillo,

**NeedsCompilation**  yes

**Author**  Arnald Puy [aut, cre] (<https://orcid.org/0000-0001-9469-2156>),
Bertrand Ioos [ctb] (Author of included ‘sensitivity’ fragments),
Gilles Pujol [ctb] (Author of included 'sensitivity' fragments),
RStudio [cph] (Copyright holder of included 'sensitivity' fragments)

**Repository**  CRAN

**Date/Publication**  2022-04-26 11:30:02 UTC
\section*{Description}

It allows to rapidly compute, bootstrap and plot up to third-order Sobol'-based sensitivity indices using several state-of-the-art first and total-order estimators. Sobol' indices can be computed either for models that yield a scalar as a model output or for systems of differential equations. The package also provides a suit of benchmark tests functions and several options to obtain publication-ready figures of the model output uncertainty and sensitivity-related analysis.

\section*{Details}

A comprehensive empirical study of several total-order estimators included in sensobol can be found in Puy et al. (2021).

\section*{Author(s)}

Arnald Puy (<arnald.puy@pm.me>)

\textbf{Maintainer:} Arnald Puy (<arnald.puy@pm.me>)
bratley1988_Fun

Description

It implements the Bratley and Fox (1988) function.

Usage

```r
bratley1988_Fun(X)
```

Arguments

- `X` A data frame or numeric matrix where each column is a model input and each row a sample point.

Details

The function requires $k$ model inputs and reads as follows:

$$y = \prod_{i=1}^{k} |4x_i - 2|,$$

where $x_i \sim \mathcal{U}(0, 1)$.

Value

A numeric vector with the model output.

Examples

```r
# Define settings (test with k = 10)
N <- 100; params <- paste("X", 1:10, sep = "")

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Bratley and Fox (1988) function
Y <- bratley1988_Fun(mat)
```

Description

It implements the Bratley et al. (1992) function.

Usage

bratley1992_Fun(X)

Arguments

X

A data frame or numeric matrix where each column is a model input and each row a sample point.

Details

The function requires \( k \) model inputs and reads as:

\[
y = \sum_{i=1}^{k} (-1)^i \prod_{j=1}^{i} x_j,
\]

where \( x_i \sim U(0, 1) \).

Value

A numeric vector with the model output.

References


Examples

# Define settings (test with \( k = 10 \))
N <- 100; params <- paste("X", 1:10, sep = "")

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Bratley et al. (1992) function
Y <- bratley1992_Fun(mat)
ishigami_Fun

Ishigami function

Description

It implements the Ishigami and Homma (1990) function.

Usage

ishigami_Fun(X)

Arguments

X
A data frame or numeric matrix where each column is a model input and each row a sample point.

Details

The function requires 3 model inputs and reads as

\[ y = \sin(x_1) + a \sin(x_2)^2 + bx_3^4 \sin(x_1), \]

where \( a = 2, b = 1 \) and \( (x_1, x_2, x_3) \sim \mathcal{U}(-\pi, +\pi) \). The transformation of the distribution of the model inputs from \( \mathcal{U}(0, 1) \) to \( \mathcal{U}(-\pi, +\pi) \) is conducted internally.

Value

A numeric vector with the model output.

References


Examples

# Define settings
N <- 100; params <- paste("X", 1:3, sep = "")

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Ishigami function
Y <- ishigami_Fun(mat)
Description

Random metafunction based on Becker (2020)'s metafunction.

Usage

```r
metafunction(data, k_2 = 0.5, k_3 = 0.2, epsilon = NULL)
```

Arguments

- **data**: A numeric matrix where each column is a model input and each row a sampling point.
- **k_2**: Numeric value indicating the fraction of active pairwise interactions (between 0 and 1). Default is `k_2 = 0.5`.
- **k_3**: Numeric value indicating the fraction of active three-wise interactions (between 0 and 1). Default is `k_2 = 0.2`.
- **epsilon**: Integer value. It fixes the seed for the random number generator. The default is `epsilon = NULL`.

Details

The metafunction randomly combines the following functions in a metafunction of dimension `k`:

- \( f(x) = x^3 \) (cubic).
- \( f(x) = 1 \) if \( x > 0.5 \), 0 otherwise (discontinuous).
- \( f(x) = \frac{e^x}{e-1} \) (exponential).
- \( f(x) = \frac{10^{-1}}{1+1} (x + 0.1)^{-1} \) (inverse).
- \( f(x) = x \) (linear).
- \( f(x) = 0 \) (no effect).
- \( f(x) = 4(x - 0.5)^2 \) (non-monotonic).
- \( f(x) = \frac{\sin(2\pi x)}{2} \) (periodic).
- \( f(x) = x^2 \) (quadratic).
- \( f(x) = \cos(x) \) (trigonometric).

It is constructed as follows:

\[
y = \sum_{i=1}^{k} \alpha_i f^{u_i}(x_i) + \sum_{i=1}^{k_2} \beta_i f^{u_{V_{i,1}}}(x_{V_{i,1}}) f^{u_{V_{i,2}}}(x_{V_{i,2}}) + \sum_{i=1}^{k_3} \gamma_i f^{u_{W_{i,1}}}(x_{W_{i,1}}) f^{u_{W_{i,2}}}(x_{W_{i,2}}) f^{u_{W_{i,3}}}(x_{W_{i,3}})
\]

where `k` is the model dimensionality, \( u \) is a \( k \)-length vector formed by randomly sampling with replacement the ten functions mentioned above, \( V \) and \( W \) are two matrices specifying the number...
of pairwise and three-wise interactions given the model dimensionality, and $\alpha, \beta, \gamma$ are three vectors of length $k$ generated by sampling from a mixture of two normal distributions $\Psi = 0.3N(0,5) + 0.7N(0,0.5)$. See Puy et al. (2020) and Becker (2020) for a full mathematical description of the metafunction approach.

Value

A numeric vector with the function output.

References


Examples

```r
# Define settings (number of model inputs = 86)
N <- 100; params <- paste("X", 1:86, sep = "")

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute metafunction
Y <- metafunction(mat)
```

---

oakley_Fun  

Description


Usage

```r
oakley_Fun(X)
```

Arguments

- **X**  
  A data frame or numeric matrix where each column is a model input and each row a sample point.
Details

The function requires 15 model inputs and reads as

\[ y = a_1^T x + a_2^T \sin(x) + a_3^T \cos(x) + x^T M x, \]

where \( x = x_1, x_2, ..., x_k, k = 15 \), and values for \( a_i^T, i = 1, 2, 3 \) and \( M \) are defined by Oakley and O'Hagan (2004). The transformation of the distribution of the model inputs from \( U(0, 1) \) to \( N(0, 1) \) is conducted internally.

Value

A numeric vector with the model output.

References


Examples

# Define settings
N <- 100; params <- paste("X", 1:15, sep = "")

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Oakley and O'Hagan (2004) function
Y <- oakley_Fun(mat)

## S3 method for class 'sensobol'
plot(x, order = "first", dummy = NULL, ...)

Arguments

- x: The output of `sobol_indices`.
- order: If `order = "first"`, it plots first and total-order effects. If `order = "second"`, it plots second-order effects. If `order = "third"`, it plots third-order effects. Default is `order = "first"`.
- dummy: The output of `sobol_dummy`. Default is NULL.
- ...: Other graphical parameters to plot.
Value

A ggplot object.

Examples

```r
# Define settings
N <- 1000; params <- paste("X", 1:3, sep = ";")  # R <- 10

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Ishigami function
Y <- ishigami_Fun(mat)

# Compute and bootstrap Sobol' indices
ind <- sobol_indices(Y = Y, N = N, params = params, boot = TRUE, R = R)

# Plot Sobol' indices
plot(ind)
```

---

**plot_multiscatter**

*Pairwise combinations of model inputs with the colour proportional to the model output value.*

Description

It plots all pairwise combinations of model inputs with the colour proportional to the model output value.

Usage

```r
plot_multiscatter(data, N, Y, params, smpl = NULL)
```

Arguments

- `data` The matrix created with `sobol_matrices`.
- `N` Positive integer, the initial sample size of the base sample matrix created with `sobol_matrices`.
- `Y` A numeric vector with the model output obtained from the matrix created with `sobol_matrices`.
- `params` Character vector with the name of the model inputs.
- `smpl` The number of simulations to plot. The default is NULL.

Value

A ggplot2 object.
Examples

```r
# Define settings
N <- 1000; params <- paste("X", 1:3, sep = ""); R <- 10

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Ishigami function
Y <- ishigami_Fun(mat)

# Plot scatter matrix
plot_multiscatter(data = mat, N = N, Y = Y, params = params)
```

---

**plot_scatter**  
*Scatter plots of the model output against the model inputs.*

Description

It creates scatter plots of the model output against the model inputs.

Usage

```r
plot_scatter(data, N, Y, params, method = "point", size = 0.7, alpha = 0.2)
```

Arguments

- **data**: The matrix created with `sobol_matrices`.
- **N**: Positive integer, the initial sample size of the base sample matrix created with `sobol_matrices`.
- **Y**: A numeric vector with the model output obtained from the matrix created with `sobol_matrices`.
- **params**: Character vector with the name of the model inputs.
- **method**: The type of plot. If `method = "point"` (the default), each simulation is a point. If `method = "bin"`, bins are used to aggregate simulations.
- **size**: Number between 0 and 1, argument of `geom_point()`. Default is 0.7.
- **alpha**: Number between 0 and 1, transparency scale of `geom_point()`. Default is 0.2.

Value

A `ggplot2` object.
Examples

# Define settings
N <- 1000; params <- paste("X", 1:3, sep = ""); R <- 10

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Ishigami function
Y <- ishigami_Fun(mat)

# Plot scatter
plot_scatter(data = mat, Y = Y, N = N, params = params)

plot_uncertainty

Visualization of the model output uncertainty

Description

It creates an histogram with the model output distribution.

Usage

plot_uncertainty(Y, N = NULL)

Arguments

Y  
A numeric vector with the model output obtained from the matrix created with
sobol_matrices.

N  
Positive integer, the initial sample size of the base sample matrix created with
sobol_matrices.

Value

A ggplot2 object.

Examples

# Define settings
N <- 1000; params <- paste("X", 1:3, sep = ""); R <- 10

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Ishigami function
Y <- ishigami_Fun(mat)

# Plot uncertainty
plot_uncertainty(Y = Y, N = N)
print.sensobol  
*Display the results obtained with the sobol_indices function.*

**Description**
Display the results obtained with the sobol_indices function.

**Usage**

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

**Arguments**

- `x`: A sensobol object produced by sobol_indices.
- `...`: Further arguments passed to or from other methods.

**Value**

The function `print.sensobol` informs on the first and total-order estimators used in the computations, the total number of model runs and the sum of first-order index. It also plots the estimated results.

---

print.vars  
*Display the results obtained with the vars_to function.*

**Description**
Display the results obtained with the vars_to function.

**Usage**

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

**Arguments**

- `x`: A vars object produced by vars_to.
- `...`: Further arguments passed to or from other methods.

**Value**

The function `print.vars` informs on the number of star centers, the value of h used and the total number of model runs. It also plots the VARS-TO indices.
sobol_convergence

check_convergence_of_Sobol_indices.

Description

It checks the convergence of Sobol’ indices on different sub-samples of the model output.

Usage

sobol_convergence(
  matrices,
  Y,
  N,
  sub.sample,
  params,
  first,
  total,
  order = order,
  seed = 666,
  plot.order,
  ...
)

Arguments

matrices Character vector with the required matrices. The default is matrices = c("A", "B", "AB"). See sobol_matrices.
Y Numeric vector with the model output obtained from the matrix created with sobol_matrices.
N Positive integer, the initial sample size of the base sample matrix created with sobol_matrices.
sub.sample Numeric vector with the sub-samples of the model output at which to check convergence.
params Character vector with the name of the model inputs.
first Estimator to compute first-order indices. Check options in sobol_indices.
total Estimator to compute total-order indices. Check options in sobol_indices.
order Whether to plot convergence for “second” or “third” order indices.
seed Whether to compute “first”, “second”, or “third” -order Sobol’ indices. Default is order = "first".
plot.order Whether to plot convergence for “second” or “third”-order indices.
...
Further arguments in sobol_indices.

Value

A list with the results and the plots
Examples

```r
# Define settings
matrices <- c("A", "B", "AB")
params <- paste("X", 1:3, sep = "")
N <- 2^10
first <- "saltelli"
total <- "jansen"
order <- "second"

# Create sample matrix
mat <- sobol_matrices(N = N, params = params, order = order)

# Compute Ishigami function
Y <- ishigami_Fun(mat)

# Check convergence at specific sample sizes
sub.sample <- seq(100, N, 500) # Define sub-samples
sobol_convergence(matrices = matrices, Y = Y, N = N, sub.sample = sub.sample,
params = params, first = first, total = total, order = order, plot.order = order)
```

---

sobol_dummy

Computation of Sobol' indices for a dummy parameter

Description

This function computes first and total-order Sobol' indices for a dummy parameter following the formulae shown in Khorashadi Zadeh et al. (2017).

Usage

```r
sobol_dummy(
  Y,  
  N, 
  params, 
  boot = FALSE, 
  R = NULL, 
  parallel = "no", 
  ncpus = 1, 
  conf = 0.95,  
  type = "norm"
)
```

Arguments

- **Y**: A numeric vector with the model output obtained from the matrix created with `sobol_matrices`
- **N**: Positive integer, the initial sample size of the base sample matrix created with `sobol_matrices`
sobol_Fun

params A character vector with the name of the model inputs.
boot Logical. If TRUE, the function bootstraps the Sobol' indices. If FALSE, it provides point estimates. Default is boot = FALSE.
R Positive integer, number of bootstrap replicas.
parallel The type of parallel operation to be used (if any). If missing, the default is taken from the option "boot.parallel" (and if that is not set, "no"). For more information, check the parallel option in the boot function of the boot package.
ncpus Positive integer: number of processes to be used in parallel operation: typically one would chose this to the number of available CPUs. Check the ncpus option in the boot function of the boot package.
conf Confidence intervals, number between 0 and 1. Default is conf = 0.95.
type Method to compute the confidence intervals. Default is type = "norm". Check the type option in the boot function of the boot package.

Value
A data.table object.

References

Examples
# Define settings
N <- 100; params <- paste("X", 1:3, sep = ""); R <- 10

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Ishigami function
Y <- ishigami_Fun(mat)

# Compute and bootstrap Sobol' indices for dummy parameter
ind.dummy <- sobol_dummy(Y = Y, N = N, params = params, boot = TRUE, R = R)

sobol_Fun  Sobol' G function

Description
Usage

sobol_Fun(X)

Arguments

X A data frame or numeric matrix.

Details

The function requires eight model inputs and reads as

\[ y = \prod_{i=1}^{k} \frac{|4x_i - 2| + a_i}{1 + a_i}, \]

where \( k = 8 \), \( x_i \sim U(0, 1) \) and \( a = (0, 1, 4.5, 9, 99, 99, 99, 99) \).

Value

A numeric vector with the model output.

References


Examples

# Define settings
N <- 100; params <- paste("X", 1:8, sep = "")

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Sobol' G
Y <- sobol_Fun(mat)


sobol_indices Computation of Sobol' indices

Description

It allows to compute Sobol’ indices up to the third order using state-of-the-art estimators.
Usage

```r
sobol_indices(
  matrices = c("A", "B", "AB"),
  Y,
  N,
  params,
  first = "saltelli",
  total = "jansen",
  order = "first",
  boot = FALSE,
  R = NULL,
  parallel = "no",
  ncpus = 1,
  conf = 0.95,
  type = "norm"
)
```

Arguments

- **matrices** Character vector with the required matrices. The default is `matrices = c("A", "B", "AB")`. See [sobol_matrices](#).
- **Y** numeric vector with the model output obtained from the matrix created with `sobol_matrices`.
- **N** Positive integer, the initial sample size of the base sample matrix created with `sobol_matrices`.
- **params** Character vector with the name of the model inputs.
- **first** Estimator to compute first-order indices. Options are:
  - `first = "saltelli"` (Saltelli et al. 2010).
  - `first = "jansen"` (Jansen 1999).
  - `first = "sobol"` (Sobol’ 1993).
  - `first = "azzini"` (Azzini et al. 2020).
- **total** Estimator to compute total-order indices. Options are:
  - `total = "jansen"` (Jansen 1999).
  - `total = "sobol"` (Sobol’ 2001).
  - `total = "homma"` (Homma and Saltelli 1996).
  - `total = "janon"` (Janon et al. 2014).
  - `total = "glen"` (Glen and Isaacs 2012).
  - `total = "azzini"` (Azzini et al. 2020).
  - `total = "saltelli"` (Saltelli et al. 2008).
- **order** Whether to compute "first", "second", or "third"-order Sobol’ indices. Default is `order = "first"`.
- **boot** Logical. If TRUE, the function bootstraps the Sobol’ indices. If FALSE, it provides point estimates. Default is `boot = FALSE`.
- **R** Positive integer, number of bootstrap replicas. Default is `NULL`. 

sobol_indices

parallel The type of parallel operation to be used (if any). If missing, the default is taken from the option "boot.parallel" (and if that is not set, "no"). For more information, check the parallel option in the boot function of the boot package.

cpus Positive integer: number of processes to be used in parallel operation: typically one would chose this to the number of available CPUs. Check the ncpus option in the boot function of the boot package.

conf Confidence interval if boot = TRUE. Number between 0 and 1. Default is conf = 0.95.

type Method to compute the confidence interval if boot = TRUE. Default is "norm". Check the type option in the boot function of the boot package.

Details
Any first and total-order estimator can be combined with the appropriate sampling design. Check Table 3 of the vignette for a summary of all possible combinations, and Tables 1 and 2 for a mathematical description of the estimators. If the analyst mismatches estimators and sampling designs, the function will generate an error and urge to redefine the sample matrices or the estimators.

For all estimators except Azzini et al. (2020)'s and Janon et al. (2014)'s, sobol_indices() calculates the sample mean as

\[ \hat{f}_0 = \frac{1}{2N} \sum_{v=1}^{N} (f(A)_v + f(B)_v), \]

where \( N \) is the row dimension of the base sample matrix, and the unconditional sample variance as

\[ \hat{V}(y) = \frac{1}{2N-1} \sum_v = 1^N ((f(A)_v - \hat{f})^2 + (f(B)_v - \hat{f})^2), \]

where \( f(A)_v, f(B)_v \) indicates the model output \( y \) obtained after running the model \( f \) in the \( v \)-th row of the \( A \) (\( B \)) matrix.

For the Azzini estimator,

\[ \hat{V}(y) = \sum_{v=1}^{N} (f(A)_v - f(B)_v)^2 + (f(B^{(i)}_v) - f(A^{(i)}_v))^2 \]

and for the Janon estimator,

\[ \hat{V}(y) = \frac{1}{N} \sum_{v=1}^{N} \frac{f(A)^2_v + f(A^{(i)}_v)^2}{2} - f_0^2 \]

where \( f(A^{(i)}_v), f(B^{(i)}_v) \) is the model output obtained after running the model \( f \) in the \( v \)-th row of an \( A^{(i)}_v \) (\( B^{(i)}_v \)) matrix, where all columns come from \( A \) (\( B \)) except the \( i \)-th, which comes from \( B \) (\( A \)).

Value
A sensobol object.
References


models.” Reliability Engineering & System Safety, 52, 1–17. doi: 10.1016/0951-8320(96)000026, 
https://doi.org/10.1016/0951-8320(96)00002-6.

ps/2013040.

S0010-4655(98)00154-4.

sensitivity analysis of model output. Design and estimator for the total sensitivity index.” Computer 

9780470725184.


S03784754(00)002706, https://doi.org/10.1016/S0378-4754(00)00270-6.

See Also

Check the function boot for further details on the bootstrapping with regards to the methods available for the computation of confidence intervals in the type argument.

Examples

# Define settings
N <- 1000; params <- paste("X", 1:3, sep = ""); R <- 10

# Create sample matrix
mat <- sobol_matrices(N = N, params = params)

# Compute Ishigami function
Y <- ishigami_Fun(mat)
# Compute and bootstrap Sobol' indices
ind <- sobol_indices(Y = Y, N = N, params = params, boot = TRUE, R = R)

---

sobol_matrices

**Creation of the sample matrices**

**Description**

It creates the sample matrices to compute Sobol' first and total-order indices. If needed, it also creates the sample matrices required to compute second and third-order indices.

**Usage**

```r
sobol_matrices(
  matrices = c("A", "B", "AB"),
  N,
  params,
  order = "first",
  type = "QRN",
  ...
)
```

**Arguments**

- **matrices** Character vector with the required matrices. The default is `matrices = c("A", "B", "AB")`.
- **N** Positive integer, initial sample size of the base sample matrix.
- **params** Character vector with the name of the model inputs.
- **order** One of "first", "second" or "third" to create a matrix to compute first, second or up to third-order Sobol indices. The default is `order = "first"`.
- **type** Approach to construct the sample matrix. Options are:
  - `type = "QRN"` (default): It uses Sobol' (1967) Quasi-Random Numbers through a call to the function `sobol` of the `randtoolbox` package.
  - `type = "LHS"`: It uses a Latin Hypercube Sampling Design (McKay et al. 1979) through a call to the function `randomLHS` of the `lhs` package.
  - `type = "R"`: It uses random numbers.
  ...

**Details**

Before calling `sobol_matrices`, the user must decide which estimators will be used to compute first and total-order indices, for this option conditions the design of the sample matrix and therefore the argument `matrices`. See Table 3 in the vignette for further details on the specific sampling designs required by the estimators.

The user can select one of the following sampling designs:
• $A, B, A_B^{(i)}$.
• $A, B, B_A^{(i)}$.
• $A, B, A_B^{(i)}, B_A^{(i)}$.

If order = "first", the function creates an $(N, 2k)$ matrix according to the approach defined by type, where the leftmost and the rightmost $k$ columns are respectively allocated to the $A$ and the $B$ matrix. Depending on the sampling design, it also creates $k$ $A_B^{(i)}$ ($B_A^{(i)}$) matrices, where all columns come from $A$ ($B$) except the $i$-th, which comes from $B$ ($A$). All matrices are returned row-binded.

If order = "second", $\frac{k!}{2!(k-2)!}$ extra $(N, k)$ $A_B^{(ij)}$ ($B_A^{(ij)}$) matrices are created, where all columns come from $A$ ($B$) except the $i$-th and $j$-th, which come from $B$ ($A$). These matrices allow the computation of second-order effects, and are row-bound to those created for first and total-order indices.

If order = "third", $\frac{k!}{3!(k-3)!}$ extra $(N, k)$ $A_B^{(ijl)}$ ($B_A^{(ijl)}$) matrices are bound below those created for the computation of second-order effects. In these matrices, all columns come from $A$ ($B$) except the $i$-th, the $j$-th and the $l$-th, which come from $B$ ($A$). These matrices are needed to compute third-order effects, and are row-bound below those created for second-order effects.

All columns are distributed in $(0,1)$. If the uncertainty in some parameter(s) is better described with another distribution, the user should apply the required quantile inverse transformation to the column of interest once the sample matrix is produced.

Value

A numeric matrix where each column is a model input distributed in $(0,1)$ and each row a sampling point.

References


Examples

# Define settings
N <- 100; params <- paste("X", 1:10, sep = " "); order <- "third"

# Create sample matrix using Sobol’ Quasi Random Numbers.
mat <- sobol_matrices(N = N, params = params, order = order)

# Let’s assume that the uncertainty in X3 is better described
# with a normal distribution with mean 0 and standard deviation 1:
mat[, 3] <- qnorm(mat[, 3], 0, 1)
sobol_ode

Wrapper around deSolve ode.

Description

It solves a system of ordinary differential equations and extracts the model output at the selected times.

Usage

sobol_ode(d, times, timeOutput, state, func, ...)

Arguments

d  Character vector with the name of the model inputs.
times  Time sequence as defined by ode.
timeOutput  Numeric vector determining the time steps at which the output is wanted.
state  Initial values of the state variables.
func  An R function as defined by ode.
...  Additional arguments passed to ode.

Value

A matrix with the output values.

Examples

# Define the model: the Lotka-Volterra system of equations
lotka_volterra_fun <- function(t, state, parameters) {
    with(as.list(c(state, parameters)), {
        dX <- r * X * (1 - X / K) - alpha * X * Y
        dY <- -m * Y + theta * X * Y
        list(c(dX, dY))
    })
}

# Define the settings of the sensitivity analysis
N <- 2 ^ 5 # Sample size of sample matrix
params <- c("r", "alpha", "m", "theta", "K", "X", "Y") # Parameters

# Define the times
times <- seq(5, 20, 1)

# Define the times at which the output is wanted
timeOutput <- c(10, 15)

# Construct the sample matrix
vars_matrices

mat <- sobol_matrices(N = N, params = params)

# Transform to appropriate distributions
mat[, "r"] <- qunif(mat[, "r"], 0.8, 1.8)
mat[, "alpha"] <- qunif(mat[, "alpha"], 0.2, 1)
mat[, "m"] <- qunif(mat[, "m"], 0.6, 1)
mat[, "theta"] <- qunif(mat[, "theta"], 0.05, 0.15)
mat[, "K"] <- qunif(mat[, "K"], 47, 53)
mat[, "X"] <- floor(mat[, "X"] * (15 - 8 + 1) + 8)
mat[, "Y"] <- floor(mat[, "Y"] * (2 - 6 + 1) + 6)

# Run the model
y <- list()
for (i in 1:nrow(mat)) {
    y[[i]] <- sobol_ode(d = mat[i, ],
        times = times,
        timeOutput = timeOutput,
        state = c(X = mat[i, "X"], Y = mat[i, "Y"]),
        func = lotka_volterra_fun)
}

vars_matrices

STAR-VARS sampling strategy

Description

It creates the STAR-VARS matrix needed to compute VARS-TO following Razavi and Gupta (2016).

Usage

vars_matrices(star.centers, params, h = 0.1, type = "QRN", ...)

Arguments

star.centers  Positive integer, number of star centers.
params        Character vector with the name of the model inputs.
h             Distance between pairs. The user should select between 0.001, 0.002, 0.005,
              0.01, 0.02, 0.05, 0.1, 0.2. Default is h = 0.1.
type          Approach to construct the STAR-VARS. Options are:
              • type = "QRN": It uses Sobol’ (1967) Quasi-Random Numbers through a
call to the function sobol of the randtoolbox package.
              • type = "R": It uses random numbers.

... Further arguments in sobol.
Details

The user randomly selects $N_{\text{star}}$ points across the factor space using either Sobol’ Quasi Random Numbers (type = "QRN") or random numbers (type = "R"). These are the star centres and their location can be denoted as $s_v = s_{v1}, ..., s_{vk}$, where $v = 1, 2, ..., N_{\text{star}}$. Then, for each star centre, the function generates a cross section of equally spaced points $\Delta h$ apart for each of the $k$ model inputs, including and passing through the star centre. The cross section is produced by fixing $s_{vi}$ and varying $s_i$. Finally, for each factor all pairs of points with $h$ values of $\Delta h, 2\Delta h, 3\Delta h$ and so on are extracted. The total computational cost of this design is $N_t = N_{\text{star}}(k(\frac{1}{\Delta h} - 1) + 1)$.

Value

A matrix where each column is a model input and each row a sampling point.

References


Examples

```r
# Define settings
star.centers <- 10; params <- paste("X", 1:5, sep = ""); h <- 0.1

# Create STAR-VARS
mat <- vars_matrices(star.centers = star.centers, params = params, h = h)
```

---

**VARS-TO**

**Description**

It computes VARS-TO following Razavi and Gupta (2016).

**Usage**

```r
vars_to(Y, star.centers, params, h, method = "all.step")
```
**vars_to**

**Arguments**

- **Y** A numeric vector with the model output obtained from the matrix created with `vars_matrices`.
- **star.centers** Positive integer, number of star centers.
- **params** Character vector with the name of the model inputs.
- **h** Distance between pairs.
- **method** Type of computation. If `method = "all.step"`, all pairs of points with values \( \Delta h, 2\Delta h, 3\Delta h, \ldots \) are used in each dimension. If `method = "one.step"`, only the pairs \( \Delta h \) away are used. The default is `method = "all.step"`.

**Details**

VARS is based on variogram analysis to characterize the spatial structure and variability of a given model output across the input space (Razavi and Gupta 2016). Variance-based total-order effects can be computed as by-products of the VARS framework. The total-order index is related to the variogram \( \gamma(\cdot) \) and co-variogram \( C(\cdot) \) functions by the following equation:

\[
T_i = \frac{\gamma(h_i) + E[C_{x\sim_i}(h_i)]}{\hat{V}(y)}
\]

where \( x_{\sim_i} \) is a vector of all \( k \) factors except \( x_i \).

**Value**

A `data.table` with the VARS-TO indices of each parameter.

**References**


**Examples**

```r
# Define settings
star.centers <- 10; params <- paste("X", 1:3, sep = ""); h <- 0.1

# Create STAR-VARS
mat <- vars_matrices(star.centers = star.centers, params = params, h = h)

# Run model
y <- sensobol::ishigami_Fun(mat)

# Compute VARS-TO
ind <- vars_to(Y = y, star.centers = star.centers, params = params, h = h)
ind
```
Index

* modeling
  sensobol-package, 2

* sensitivity
  sensobol-package, 2

* uncertainty
  sensobol-package, 2

boot, 15, 18, 19
bratley1988_Fun, 3
bratley1992_Fun, 4

ishigami_Fun, 5

metafunction, 6

oakley_Fun, 7
ode, 22

plot.sensobol, 8
plot_multiscatter, 9
plot_scatter, 10
plot_uncertainty, 11
print.sensobol, 12
print.vars, 12

randomLHS, 20

sensobol (sensobol-package), 2
sensobol-package, 2
sobol, 20, 23
sobol_convergence, 13
sobol_dummy, 8, 14
sobol_Fun, 15
sobol_indices, 8, 13, 16
sobol_matrices, 9–11, 13, 14, 17, 20
sobol_ode, 22

vars_matrices, 23, 25
vars_to, 24

26