# Package 'sensobol'

November 25, 2022

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

**Version** 1.1.3

**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), randtoolbox (>= 1.17.1), deSolve (>= 1.27.1), Rdpack (>= 2.1.2), Rfast (>= 2.0.1), rlang (>= 0.3.1), scales (>= 1.0.0), stats, stringr (>= 1.4.0), utils, Rcpp

**RdMacros** Rdpack

**Depends** R (>= 3.5.0)

**RoxygenNote** 7.2.2

**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** 2022-11-25 13:10:02 UTC

# R topics documented:



---

`sensobol-package` *sensobol: Computation of Variance-Based Sensitivity Indices*

---

## 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.

## Details

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

## Author(s)

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

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

## References

Puy A, Becker W, Lo Piano S, Saltelli A (2021). "A Comprehensive Comparison of Total-Order Estimators for Global Sensitivity Analysis." *International Journal for Uncertainty Quantification*. doi:10.1615/Int.J.UncertaintyQuantification.2021038133.

---

`bratley1988_Fun` *Bratley and Fox (1988) function*

---

## Description

It implements the Bratley and Fox (1988) function.

## Usage

```
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

```
# 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)
```

`bratley1992_Fun` *Bratley, Fox and Niederreiter (1992) function.*

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

## Value

A numeric vector with the model output.

## References

Bratley P, Fox BL, Niederreiter H (1992). "Implementation and tests of low-discrepancy sequences." *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, **2**(3), 195–213.

## 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

$$y = \sin(x_1) + a \sin(x_2)^2 + b x_3^4 \sin(x_1)\,,$$

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 $U(0, 1)$ to $U(-\pi, +\pi)$) is conducted internally.

## Value

A numeric vector with the model output.

## References

Ishigami T, Homma T (1990). "An importance quantification technique in uncertainty analysis for computer models." *Proceedings. First International Symposium on Uncertainty Modeling and Analysis*, **12**, 398–403.

## 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)
```

## load_packages *Load (and install) R packages.*

### Description

The function loads R packages. If the packages are not already in the local system, the function also downloads, installs and loads them.

### Usage

```
load_packages(x)
```

### Arguments

| | |
|---|---|
| x | A character vector with the name of the packages. |

### Examples

```
# Load packages:
## Not run: load_packages(c("sensobol", "data.table"))
```

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

### Description

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

### Usage

```
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).
- $f(x) = \frac{10-1}{1.1}^{-1}(x + 0.1)^{-1}$ (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 $\Psi = 0.3\mathcal{N}(0, 5) + 0.7\mathcal{N}(0, 0.5)$. 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

Becker W (2020). "Metafunctions for benchmarking in sensitivity analysis." *Reliability Engineering and System Safety*, **204**, 107189. doi:10.1016/j.ress.2020.107189.

Puy A, Becker W, Piano SL, Saltelli A (2020). "The battle of total-order sensitivity estimators." *arXiv*. 2009.01147, https://arxiv.org/abs/2009.01147.

## Examples

```
# 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` *Oakley & O'Hagan (2004) function*

---

## Description

It implements the Oakley and O'Hagan (2004) function.

## Usage

```
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

$$y = \mathbf{a}_1^T \boldsymbol{x} + \mathbf{a}_2^T \sin(\mathbf{x}) + \mathbf{a}_3^T \cos(\mathbf{x}) + \mathbf{x}^T \mathbf{M} \mathbf{x}\,,$$

where $\mathbf{x} = x_1, x_2, ..., x_k$, $k = 15$, and values for $\mathbf{a}_i^T, i = 1, 2, 3$ and $\mathbf{M}$ 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

Oakley JE, O'Hagan A (2004). "Probabilistic sensitivity analysis of complex models: a Bayesian approach." *Journal of the Royal Statistical Society B*, **66**(3), 751–769. doi:10.1111/j.14679868.2004.05304.x.

## 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)
```

`plot.sensobol` *Visualization of first, total, second, third and fourth-order Sobol' indices.*

## Description

It plots first, total, second, third and fourth-order Sobol' indices.

## Usage

```
## 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. If order = "fourth", 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

```
# 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 the model output value.*

## Description

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

## Usage

```
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

```
# 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 scatterplot 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

```
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

```
## 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

```
## 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

```
# 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

```
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. The numeric vector should not contain any NA or NaN values. |
| N | Positive integer, the initial sample size of the base sample matrix created with sobol_matrices. |
| 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

Khorashadi Zadeh F, Nossent J, Sarrazin F, Pianosi F, van Griensven A, Wagener T, Bauwens W (2017). "Comparison of variance-based and moment-independent global sensitivity analysis approaches by application to the SWAT model." *Environmental Modelling and Software*, **91**, 210–222. doi:10.1016/j.envsoft.2017.02.001.

## 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

It implements the Sobol' (1998) G function.

## 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 $k = 8$, $x_i \sim \mathcal{U}(0, 1)$ and $a = (0, 1, 4.5, 9, 99, 99, 99, 99)$.

### Value

A numeric vector with the model output.

### References

Sobol' IM (1998). "On quasi-Monte Carlo integrations." *Mathematics and Computers in Simulation*, **47**(2-5), 103–112. doi:10.1016/S03784754(98)000962.

### 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 fourth-order using state-of-the-art estimators.

### Usage

```
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. The numeric vector should not contain any NA or NaN values. |
| 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:<br>• `first = "saltelli"` (Saltelli et al. 2010).<br>• `first = "jansen"` (Jansen 1999).<br>• `first = "sobol"` (Sobol' 1993).<br>• `first = "azzini"` (Azzini et al. 2020). |
| total | Estimator to compute total-order indices. Options are:<br>• `total = "jansen"` (Jansen 1999).<br>• `total = "sobol"` (Sobol' 2001).<br>• `total = "homma"` (Homma and Saltelli 1996).<br>• `total = "janon"` (Janon et al. 2014).<br>• `total = "glen"` (Glen and Isaacs 2012).<br>• `total = "azzini"` (Azzini et al. 2020).<br>• `total = "saltelli"` (Saltelli et al. 2008). |
| order | Whether to compute "first", "second", "third" or fourth-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. |
| 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 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

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

where $N$ is the row dimension of the base sample matrix, and the unconditional sample variance as

$$\hat{V}(y) = \frac{1}{2N-1} \sum v = 1^N ((f(\mathbf{A})_v - \hat{f})^2 + (f(\mathbf{B})_v - \hat{f})^2) \,,$$

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

For the Azzini estimator,

$$\hat{V}(y) = \sum_{v=1}^{N} (f(\mathbf{A})_v - f(\mathbf{B})_v)^2 + (f(\mathbf{B}_A^{(i)})_v - f(\mathbf{A}_B^{(i)})_v)^2$$

and for the Janon estimator,

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

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

## Value

A `sensobol` object.

## References

Azzini I, Mara T, Rosati R (2020). "Monte Carlo estimators of first-and total-orders Sobol' indices." *arXiv*. 2006.08232, https://arxiv.org/abs/2006.08232.

Glen G, Isaacs K (2012). "Estimating Sobol sensitivity indices using correlations." *Environmental Modelling and Software*, **37**, 157–166. doi:10.1016/j.envsoft.2012.03.014.

Homma T, Saltelli A (1996). "Importance measures in global sensitivity analysis of nonlinear models." *Reliability Engineering and System Safety*, **52**, 1–17. doi:10.1016/09518320(96)000026.

Janon A, Klein T, Lagnoux A, Nodet M, Prieur C (2014). "Asymptotic normality and efficiency

of two Sobol index estimators." *ESAIM: Probability and Statistics*, **18**(3), 342–364. doi:10.1051/ps/2013040.

Jansen M (1999). "Analysis of variance designs for model output." *Computer Physics Communications*, **117**(1), 35–43. doi:10.1016/S00104655(98)001544.

Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S (2010). "Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index." *Computer Physics Communications*, **181**(2), 259–270. doi:10.1016/j.cpc.2009.09.018.

Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, Saisana M, Tarantola S (2008). *Global Sensitivity Analysis. The Primer*. John Wiley and Sons, Ltd, Chichester, UK. doi:10.1002/9780470725184.

Sobol' IM (1993). "Sensitivity analysis for nonlinear mathematical models." *Mathematical Modeling and Computational Experiment*, **1**(4), 407–414.

Sobol' IM (2001). "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates." *Mathematics and Computers in Simulation*, **55**(1-3), 271–280. doi:10.1016/S03784754(00)002706.

## 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, third and fourth-order indices.

## Usage

```
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", "third" or "fourth" to create a matrix to compute first, second, third or up to fourth-order Sobol' indices. The default is order = "first". |
| type | Approach to construct the sample matrix. Options are:<br>• type = "QRN" (default): It uses Sobol' (1967) Quasi-Random Numbers. through a call to the function sobol of the randtoolbox package.<br>• type = "LHS": It uses a Latin Hypercube Sampling Design (McKay et al. 1979) through a call to the function randomLHS of the lhs package.<br>• type = "R": It uses random numbers. |
| ... | Further arguments in sobol. |

## 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:

- $\mathbf{A}$, $\mathbf{B}$, $\mathbf{A}_B^{(i)}$.
- $\mathbf{A}$, $\mathbf{B}$, $\mathbf{B}_A^{(i)}$.
- $\mathbf{A}$, $\mathbf{B}$, $\mathbf{A}_B^{(i)}$, $\mathbf{B}_A^{(i)}$.

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 $\mathbf{A}$ and the $\mathbf{B}$ matrix. Depending on the sampling design, it also creates $k$ $\mathbf{A}_B^{(i)}$ ($\mathbf{B}_A^{(i)}$) matrices, where all columns come from $\mathbf{A}$ ($\mathbf{B}$) except the $i$-th, which comes from $\mathbf{B}$ ($\mathbf{A}$). All matrices are returned row-binded.

If order = "second", $\frac{k!}{2!(k-2)!}$ extra $(N, k)$ $\mathbf{A}_B^{(ij)}$ ($\mathbf{B}_A^{(ij)}$) matrices are created, where all columns come from $\mathbf{A}$ ($\mathbf{B}$) except the $i$-th and $j$-th, which come from $\mathbf{B}$ ($\mathbf{A}$). 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 $(N, k)$ $\mathbf{A}_B^{(ijl)}$ ($\mathbf{B}_A^{(ijl)}$) matrices are bound below those created for the computation of second-order effects. In these matrices, all columns come from $\mathbf{A}$ ($\mathbf{B}$) except the $i$-th, the $j$-th and the $l$-th, which come from $\mathbf{B}$ ($\mathbf{A}$). These matrices are needed to compute third-order effects, and are row-bound below those created for second-order effects.

The same process applies to create the matrices to compute fourth-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

McKay MD, Beckman RJ, Conover WJ (1979). "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code." *Technometrics*, **21**(2), 239–245. doi:10.1080/00401706.1979.10489755.

Sobol' IM (1967). "On the distribution of points in a cube and the approximate evaluation of integrals." *USSR Computational Mathematics and Mathematical Physics*, **7**(4), 86–112. doi:10.1016/00415553(67)901449.

## 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
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:<br>• `type = "QRN"`: It uses Sobol' (1967) Quasi-Random Numbers through a call to the function `sobol` of the `randtoolbox` package.<br>• `type = "R"`: It uses random numbers. |
| `...` | Further arguments in `sobol`. |

## Details

The user randomly selects $N_{star}$ 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 $\mathbf{s}_v = s_{v_1}, ..., s_{v_i}, ..., s_{v_k}$, where $v = 1, 2, ..., N_{star}$. 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 $\mathbf{s}_{v_{\sim i}}$ 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 $N_t = N_{star}(k(\frac{1}{\Delta h} - 1) + 1)$.

## Value

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

## References

Razavi S, Gupta HV (2016). "A new framework for comprehensive, robust, and efficient global sensitivity analysis: 2. Application." *Water Resources Research*, **52**(1), 440–455. doi:10.1002/2015WR017558, 2014WR016527.

Sobol' IM (1967). "On the distribution of points in a cube and the approximate evaluation of integrals." *USSR Computational Mathematics and Mathematical Physics*, **7**(4), 86–112. doi:10.1016/00415553(67)901449.

## Examples

```
# 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 *Computation of VARS Total order index (VARS-TO)*

---

## Description

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

## Usage

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

## 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 $\Delta h, 2\Delta h, 3\Delta h, ...$ 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 $\gamma(.)$ and co-variogram $C(.)$ functions by the following equation:

$$T_i = \frac{\gamma(h_i) + E\left[C_{\mathbf{x}_{\sim i}}(h_i)\right]}{\hat{V}(y)}$$

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

Razavi S, Gupta HV (2016). "A new framework for comprehensive, robust, and efficient global sensitivity analysis: 2. Application." *Water Resources Research*, **52**(1), 440–455. doi:10.1002/2015WR017558, 2014WR016527.

## Examples

```
# 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