sensobol: an R package to compute variance-based sensitivity indices

Arnald Puy
Ecology and Evolutionary Biology,
Princeton University

Samuele Lo Piano
School of the Built Environment
University of Reading

Andrea Saltelli
Open Evidence Research
Universitat Oberta de Catalunya

Simon A. Levin
Ecology and Evolutionary Biology,
Princeton University

Abstract

The R package **sensobol** provides several functions to conduct variance-based uncertainty and sensitivity analysis, from the estimation of sensitivity indices to the visual representation of the results. It implements several state-of-the-art first and total-order estimators and allows the computation of up to third-order effects, as well as of the approximation error, in a swift and user-friendly way. Its flexibility makes it also appropriate for models with either a scalar or a multivariate output. We illustrate its functionality by conducting a variance-based sensitivity analysis of three classic models: the Sobol' (1998) G function, the logistic population growth model of Verhulst (1845), and the spruce budworm and forest model of Ludwig, Jones, and Holling (1976).

Keywords: R, Uncertainty, Sensitivity Analysis, Modeling.

1. Introduction

It has been argued that any form of knowledge based on mathematical modeling is conditional on a set, perhaps a hierarchy, of either stated or unspoken assumptions (Kay 2012; Saltelli, Bammer, Bruno, Charters, Di Fiore, Didier, Nelson Espeland, Kay, Lo Piano, Mayo, Pielke Jr, Portaluri, Porter, Puy, Rafols, Ravetz, Reinert, Sarewitz, Stark, Stirling, van der Sluijs, and Vineis 2020). Such assumptions range from the choice of the data and of the methods to the framing of the problem, including normative elements that identify the nature and the relevance of the problem itself. This conditional uncertainty is a property of the model and not of the reality that the model has the ambition to depict. Yet it affects the model output and hence any model-based inference aiming at guiding policies in the "real world". Identifying and understanding this conditional uncertainty is especially paramount when the model output serves to inform a political decision, and boils down to answering two classes of questions:

- How uncertain is the the inference? Is this uncertainty compatible with the taking of a decision based on the model otcomes? Given the uncertainty, are the policy options

distinguishable in their outcome?

- Which factor is dominating this uncertainty? Is this uncertainty reducible, e.g., with more data or deeper research? Are there a few dominating factors or is the uncertainty originating from several factors? Do the factors act singularly or in combination with one another?

The second class of questions is the realm of global sensitivity analysis, which aims to offer a diagnosis as to the composition of the uncertainty affecting the model output, and hence the model based inference (Saltelli, Andres, and Homma 1993; Homma and Saltelli 1996a; Saltelli, Ratto, Andres, Campolongo, Cariboni, Gatelli, Saisana, and Tarantola 2008). In helping to appreciate the extent and the nature of the problems linked to the use of a given model in a practical setting, global sensitivity analysis can be considered as a tool for the hermeneutics of mathematical modeling.

Global sensitivity analysis is well represented in international guidelines for impact assessment (Azzini, Listorti, Mara, and Rosati 2020a; Gilbertson 2018), as well as in many disciplinary journals (Jakeman, Letcher, and Norton 2006; Puy, Lo Piano, and Saltelli 2020c). However, the uptake of state-of-the-art global sensitivity analysis tools is still in its infancy. Most studies continue to prioritize local sensitivity or *one-at-a-time* analyses, which explore how the model output changes when one factor is varied and the rest is kept fixed at their nominal values (Saltelli, Aleksankina, Becker, Fennell, Ferretti, Holst, Li, and Wu 2019). This approach underexplores the input space and can not appraise interactions between factors, which are ubiquitous in many models. Some reasons behind the scarce use of global sensitivity analysis methods are lack of technical skills or resources available, unawareness of global sensitivity methods or simply reluctance due to their "destructive honesty": if applied properly, the uncertainty uncovered by a global sensitivity analysis might be so wide as to render the model largely impractical for policy-making (Leamer 2010; Saltelli *et al.* 2019).

This notwithstanding, there seems to be a progressive increase in the use of global sensitivity methods from 2005 onwards (Ferretti, Saltelli, and Tarantola 2016), as well as a higher acknowledgment of them being the ultimate acid test for the quality of any mathematical model. Recently, global sensitivity analysis has been identified as one of the most well-equipped scientific toolkits to tackle "deep uncertainty" (Steinmann, Wang, van Voorn, and Kwakkel 2020), and a multidisciplinary team of scholars lists it as one of the five cornerstones of responsible mathematical modeling (Saltelli *et al.* 2020).

1.1. Sensitivity analysis packages in R and beyond

The sparse uptake of global sensitivity methods contrasts with the many packages available in different languages. In Python there is the **SALib** package (Herman and Usher 2017), which includes the Sobol', Morris and the Fourier Amplitude Sensitivity Test (FAST) methods. In MATLAB, the **UQLab** offers the Morris method, the Borgonovo (2007) indices, Sobol' indices (with the Sobol' and Janon estimators) and the Kucherenko indices (Marelli and Sudret 2014). The **SAFE** package (Pianosi, Sarrazin, and Wagener 2015), developed originally for MATLAB / Octave but with scripts available for R and Python, includes variance-based analysis, elementary effects and the PAWN method (Pianosi and Wagener 2015).

To our knowledge, there are three packages on CRAN that implement global sensitivity analysis in R (R Core Team 2020): the **multisensi** package (Bidot, Lamboni, and Monod 2018),

specifically designed for models with a multivariate output; the **fast** package (Reusser D. 2015), which implements FAST (removed from CRAN on 29-08-2020); and the **sensitivity** package (Iooss, Janon, Pujol, with contributions from Baptiste Broto, Boumhaout, Veiga, Delage, Amri, Fruth, Gilquin, Guillaume, Le Gratiet, Lemaitre, Marrel, Meynaoui, Nelson, Monari, Oomen, Rakovec, Ramos, Roustant, Song, Staum, Sueur, Touati, and Weber 2020), the most comprehensive collection of functions in R for screening, global sensitivity analysis and robustness analysis.

sensobol differs from these R packages by the following characteristics:

1. *It offers a state-of-the-art compilation of variance-based sensitivity estimators.* Variance-based sensitivity analysis is regarded as the gold standard of global sensitivity methods. In its current version, **sensobol** comprises four first-order and eight total-order variance-based estimators, from the classic formulae of Sobol' (1993) or Jansen (1999) to the more recent contributions by Glen and Isaacs (2012), Razavi and Gupta (2016b,a) (VARS-TO) or Azzini, Mara, and Rosati (2020b).
2. *It is very flexible and user-friendly.* There is only one function to compute Sobol'-based sensitivity indices, `sobol_indices()`. Any first and total-order estimator can be *simultaneously* feed into the function provided that the user correctly specifies the sampling design (see Section 2.1). This contrasts with the **sensitivity** package (Iooss *et al.* 2020), which keeps estimators compartmentalized in different functions and hence prevents the user from combining them the way it better suits their needs. Furthermore, the compatibility of `sobol_indices()` with the **data.table** syntax makes the calculation of sensitivity indices for scalar outputs as easy as for multivariate outputs (see Section 3.3) (Dowle and Srinivasan 2019).
3. *It permits the computation of up to third-order effects.* Appraising high-order effects is paramount when models are non-additive (see Section 2). Although the total-order index already informs on whether a parameter is involved in interactions, sometimes a more precise account of the nature of this interaction is needed. **sensobol** opens the possibility to probe into these interactions through the computation of second and third-order effects regardless of the selected estimator.
4. *It offers publication-ready figures of the model output and sensitivity-related analysis.* **sensobol** relies on **ggplot2** (Wickham 2016) and the grammar of graphics to yield high-quality plots which can be easily modified by the user.
5. *It is more efficient than current implementations of variance-based estimators in R.* Our benchmark of **sensobol** and **sensitivity** functions suggest that the former may be approximately two times faster than the latter (See Annex, section 6.1).

The paper is organized as follows: in Section 2 we briefly describe variance-based sensitivity analysis. In Section 3 we walk through three examples of models with different characteristics and increasing complexity to show all the functionalities of **sensobol**. Finally, we summarize the main contributions of the package in Section 4.

2. Variance-based sensitivity analysis

Variance-based sensitivity indices use the variance to describe the model output uncertainty. Given a model of the form $y = f(\boldsymbol{x})$, $\boldsymbol{x} = x_1, x_2, \ldots, x_i, \ldots, x_k \in \mathbb{R}^k$, where y is a scalar

output and $x_1, ..., x_k$ are k uncertain parameters described by probability distributions, the analyst might be interested in assessing how sensitive y is to changes in x_i. One way of tackling this question is to check how much the variance in y decreases after fixing x_i to its "true" value x_i^*, i.e., $V(y|x_i = x_i^*)$. But the true value of x_i is unknown, so instead of fixing it to an arbitrary number, we take the mean of the variance of y after fixing x_i to all its possible values over its uncertainty range, while all other parameters are left to vary. This can be expressed as $E_{x_i}\left[V_{\boldsymbol{x}_{\sim i}}(y|x_i)\right]$, where $\boldsymbol{x}_{\sim i}$ denotes *all parameters-but-*x_i and $E(.)$ and $V(.)$ are the mean and the variance operator respectively. $E_{x_i}\left[V_{\boldsymbol{x}_{\sim i}}(y|x_i)\right] \leq V(y)$, and in fact,

$$V(y) = V_{x_i}\left[E_{\boldsymbol{x}_{\sim i}}(y|x_i)\right] + E_{x_i}\left[V_{\boldsymbol{x}_{\sim i}}(y|x_i)\right] , \tag{1}$$

where $V_{x_i}\left[E_{\boldsymbol{x}_{\sim i}}(y|x_i)\right]$ is known as the first-order effect of x_i and $E_{x_i}\left[V_{\boldsymbol{x}_{\sim i}}(y|x_i)\right]$ is the residual. When a parameter is important in conditioning $V(y)$, $V_{x_i}\left[E_{\boldsymbol{x}_{\sim i}}(y|x_i)\right]$ is high.

To illustrate this property, let's imagine we run a three-dimensional model, plot the model output y against the range of values in x_i, divide the latter in n bins and compute the mean y in each bin. This is represented in Figure 1, with the red dots showing the mean in each bin. The parameter whose mean y values vary the most has the highest direct influence in the model output; in this case, this is clearly x_1. This procedure applied over very small bins is actually $V_{x_i}\left[E_{\boldsymbol{x}_{\sim i}}(y|x_i)\right]$ and is the conditional variance of x_i on $V(y)$, V_i (Saltelli *et al.* 2008). When $x_1, x_2, \ldots, x_k$ are independent parameters, $V(y)$ can be decomposed as the sum of all partial variances up to the k-th order, as

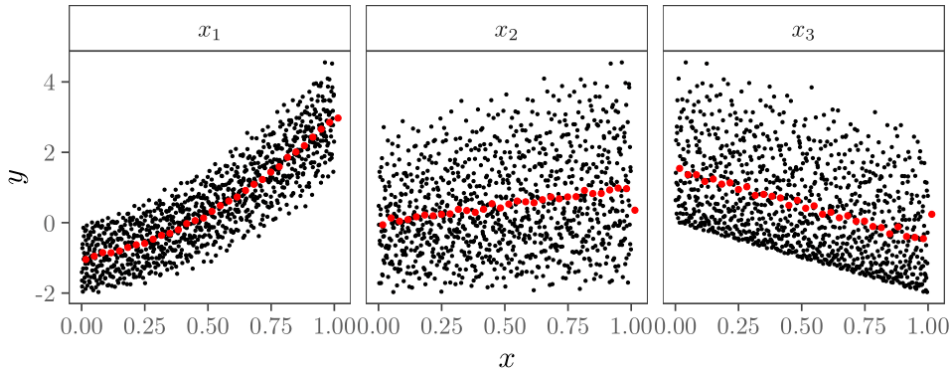

Figure 1: Scatterplot of y against x_i, $i = 1, 2, 3$. The red dots show the mean y value in each bin (we have set the number of bins arbitrarely at 30), and $N = 2^{10}$. The model is the polynomial function shown in Becker and Saltelli (2015), where $y = 3x_1^2 + 2x_1x_2 - 2x_3$, $x_i \sim \mathcal{U}(0, 1)$.

$$V(y) = \sum_{i=1} V_i + \sum_{i} \sum_{i<j} V_{ij} + ... + V_{1,2,...,k} , \tag{2}$$

where

$$\begin{aligned} V_i = V_{x_i}\left[E_{\boldsymbol{x}_{\sim i}}(y|x_i)\right] \quad V_{ij} = V_{x_i,x_j}\left[E_{\boldsymbol{x}_{\sim i,j}}(y|x_i,x_j)\right] \quad \ldots . \\ - V_{x_i}\left[E_{\boldsymbol{x}_{\sim i}}(y|x_i)\right] \\ - V_{x_j}\left[E_{\boldsymbol{x}_{\sim j}}(y|x_j)\right] \end{aligned} \tag{3}$$

Note that Equation 2 is akin to Sobol' (1993)'s functional decomposition scheme:

$$f(\boldsymbol{x}) = f_0 + \sum_i f_i(x_i) + \sum_i \sum_{i<j} f_{ij}(x_i, x_j) + \ldots + f_{1,2,\ldots,k}(x_1, x_2, \ldots, x_k)\,, \tag{4}$$

where

$$f_0 = E(y) \qquad f_i = E_{\boldsymbol{x}_{\sim i}} - f_0 \qquad f_{ij} = E_{\boldsymbol{x}_{\sim ij}} - f_i - f_j - f_0 \qquad \ldots\,, \tag{5}$$

and therefore

$$V_i = V\left[f_i(x_i)\right] \qquad V_{ij} = V\left[f_{ij}(x_i, x_j)\right] \qquad \ldots\,. \tag{6}$$

Sobol' (1993) indices are then calculated as

$$S_i = \frac{V_i}{V(y)} \qquad S_{ij} = \frac{V_{ij}}{V(y)} \qquad \ldots\,, \tag{7}$$

where S_i is the first-order effect of x_i, S_{ij} is the second-order effect of (x_i, x_j) (formed by the first order effect of x_i, x_j and their interaction), etc. S_i (S_{ij}) can thus be expressed as the fractional reduction in the variance of y which will be obtained if x_i (x_i, x_j) could be fixed. In variance-based sensitivity analysis, S_i is used to rank parameters given their contribution to the model output uncertainty, a setting known as "factor prioritization'' (Saltelli *et al.* 2008).

If we divide all terms in Equation 2 by $V(y)$, we get

$$\sum_{i=1}^{k} S_i + \sum_i \sum_{i<j} S_{ij} + \ldots + S_{1,2,\ldots,k} = 1. \tag{8}$$

When $\sum_{i=1}^{k} S_i = 1$, the model is *additive*, i.e., the variance of y can be fully decomposed as the sum of first-order effects, meaning that there are no interaction between parameters. However, this is rarely the case in real-life models, and first-order indices are usually not enough to account for all the model output variance.

This is better demonstrated with the example displayed in Figure 2: x_2 and x_3 do not have a first-order effect on y as $V_{x_i}\left[E_{\boldsymbol{x}_{\sim i}}(y|x_i)\right] \approx 0$. However, and unlike x_2, x_3 does influence y given the shape of the scatterplot, so it can not be an inconsequential parameter. Indeed, x_3 influences y through high-order effects, i.e., by interacting with some other parameter(s). In this specific case, it is clear that x_3 must interact with x_1 given that x_2 is non-influential. This notwithstanding, such appraisal of interactions can rarely be made through the visual inspection of scatterplots alone, and often requires computing higher-order terms in Equation 8.

Since there are $2^k - 1$ terms in Equation 8, a model with 10 parameters will have 1023 terms, making a full variance decomposition very arduous: just the computation of second-order terms for this model would require estimating 45 indices.

To circumvent this issue, Homma and Saltelli (1996b) proposed to compute the total-order index T_i, which measures the first-order effect of x_i jointly with its interactions with all the

other parameters. In other words, T_i includes all terms in Equation 2 with the index i, and is computed as follows:

$$T_i = 1 - \frac{V_{\boldsymbol{x}_{\sim i}}\left[E_{x_i}(y|\boldsymbol{x}_{\sim i})\right]}{V(y)} = \frac{E_{\boldsymbol{x}_{\sim i}}\left[V_{x_i}(y|\boldsymbol{x}_{\sim i})\right]}{V(y)}\,, \tag{9}$$

For a three-dimensional model, the total-order index of x_1 will thus be computed as $T_1 = S_1 + S_{1,2} + S_{1,3} + S_{1,2,3}$. Since $T_i = 0$ indicates that x_i does not convey any uncertainty to the model output, the total-order index has been used to screen influential from non-influential parameters, a setting known as "factor fixing'' (Saltelli *et al.* 2008).

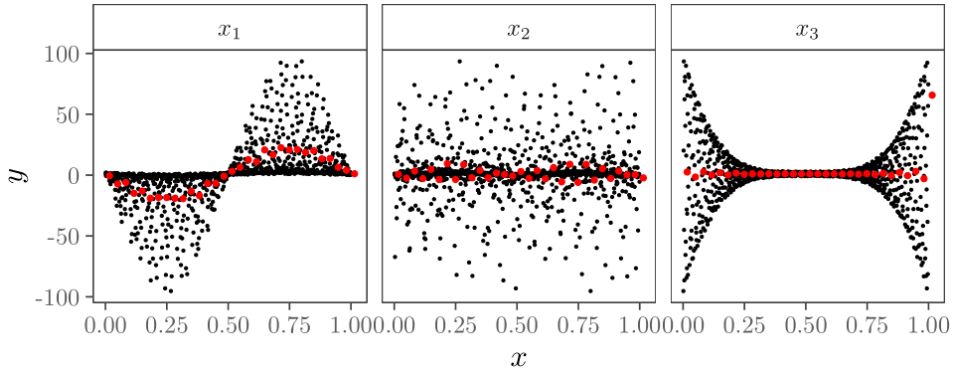

Figure 2: Scatterplot of y against x_i, $i = 1, 2, 3$. The red dots show the mean y value in each bin (we have set the number of bins arbitrarely at 30), and $N = 2^{10}$. The model is the Ishigami and Homma (1990) function.

2.1. Sampling design and sensitivity estimators

The computation of variance-based sensitivity indices requires two elements: 1) a sampling design, i.e., a strategy to arrange the sample points into the multidimensional space of the input factors, and 2) an estimator, i.e., a formula to compute the sensitivity measures (Lo Piano, Ferretti, Puy, Albrecht, and Saltelli 2021). Both elements are inextricably intertwined: the reliance on a given sampling design determines which estimators can be used, and the other way around.

The package **sensobol** currently offers support for five first-order and eight total-order sensitivity estimators, which rely on specific combinations of $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{A}_B^{(i)}$ or $\boldsymbol{B}_A^{(i)}$ matrices (Tables 1–2). Estimator 9 in Table 2 is known as VARS-TO and requires a different sampling design based on star-centers and cross-sections (Razavi and Gupta 2016a,b). We provide further information about VARS-TO in the Annex, section 6.2. All these estimators are sample-based and hence **sensobol** does not include emulators or surrogate models.

How are these matrices formed, and why are they required? Let $\boldsymbol{Q}$ be a $(N, 2k)$ matrix constructed using either random or quasi-random number generators, such as the Sobol' (1967, 1976) sequence or a Latin Hypercube Sampling design (McKay, Beckman, and Conover 1979)). The $\boldsymbol{A}$ and the $\boldsymbol{B}$ matrices include respectively the leftmost and rightmost k columns of the $\boldsymbol{Q}$ matrix. As for the $\boldsymbol{A}_B^{(i)}$ ($\boldsymbol{B}_A^{(i)}$) matrices, they are formed by all columns from the $\boldsymbol{A}$ ($\boldsymbol{B}$) matrix except the i-th, which comes from $\boldsymbol{B}$ ($\boldsymbol{A}$) (Equation 10, Figure 3).

Nº	Estimator	`first`	Author
1	$\frac{\frac{1}{N}\sum_{v=1}^{N} f(\boldsymbol{A})_v f(\boldsymbol{B}_A^{(i)})_v - f_0^2}{V(y)}$	`"sobol"`	Sobol' (1993)
2	$\frac{\frac{1}{N}\sum_{v=1}^{N} f(\boldsymbol{B})_v \left[f(\boldsymbol{A}_B^{(i)})_v - f(\boldsymbol{A})_v\right]}{V(y)}$	`"saltelli"`	Saltelli, Annoni, Azzini, Campolongo, Ratto, and Tarantola (2010)
3	$\frac{V(y) - \frac{1}{2N}\sum_{v=1}^{N}\left[f(\boldsymbol{B})_v - f(\boldsymbol{A}_B^{(i)})_v\right]^2}{V(y)}$	`"jansen"`	Jansen (1999)
4	$\frac{2\sum_{v=1}^{N}(f(\boldsymbol{B}_A^{(i)})_v - f(\boldsymbol{B})_v)(f(\boldsymbol{A})_v - f(\boldsymbol{A}_B^{(i)})_v)}{\sum_{v=1}^{N}\left[(f(\boldsymbol{A})_v - f(\boldsymbol{B})_v)^2 + (f(\boldsymbol{B}_A^{(i)})_v - f(\boldsymbol{A}_B^{(i)})_v)^2\right]}$	`"azzini"`	Azzini *et al.* (2020b)

Table 1: First-order estimators included in **sensobol** (v1.0.1).

In these matrices each column is a model input and each row a sampling point. Any sampling point in either $\boldsymbol{A}$ or $\boldsymbol{B}$ can be indicated as x_{vi}, where v and i respectively index the row (from 1 to N) and the column (from 1 to k).

First and total-order effects are then calculated by averaging several elementary effects computed rowwise: for S_i, we need pairs of points where all factors but x_i have different values (i.e., $\boldsymbol{A}_v$, $(\boldsymbol{B}_A^{(i)})_v$; or $\boldsymbol{B}_v$, $(\boldsymbol{A}_B^{(i)})_v$), and for T_i pairs of points where all factors except x_i have the same values (i.e., $\boldsymbol{A}_v$, $(\boldsymbol{A}_B^{(i)})_v$; or $\boldsymbol{B}_v$, $(\boldsymbol{B}_A^{(i)})_v$). The elementary effect for S_i thus requires moving from $\boldsymbol{A}_v$ to $(\boldsymbol{B}_A^{(i)})_v$ (or from $\boldsymbol{B}_v$ to $(\boldsymbol{A}_B^{(i)})_v$), therefore taking a step along $\boldsymbol{x}_{\sim i}$, whereby the elementary effect for T_i involves moving from $\boldsymbol{A}_v$ to $(\boldsymbol{A}_B^{(i)})_v$ (or from $\boldsymbol{B}_v$ to $(\boldsymbol{B}_A^{(i)})_v$), hence moving along x_i (Saltelli *et al.* 2010).

The function `sobol_matrices()` allows to create these sampling designs using either Sobol' (1967, 1976) Quasi-Random Numbers (`type = "QRN"`), Latin Hypercube Sampling (`type = "LHS"`) or Random numbers (`type = "R"`). In Figure 3 we show how these sampling methods differ in two dimensions. Comparatively, quasi-random numbers fill the input space quicker and more evenly, leaving smaller unexplored volumes. However, random numbers might provide more accurate sensitivity indices when the model under examination has important high-order terms (Kucherenko, Feil, Shah, and Mauntz 2011). In addition, Latin Hypercube Sampling might outperform quasi-random numbers for some specific function typologies (Kucherenko, Delpuech, Iooss, and Tarantola 2015). In general, quasi-random numbers are assumed to be the safest bet when selecting a sampling algorithm for a function of unknown behavior, and are the default setting in **sensobol**.

Nº	Estimator	`total`	Author
1	$\frac{\frac{1}{2N}\sum_{v=1}^{N}\left[f(\boldsymbol{A})_v - f(\boldsymbol{A}_B^{(i)})_v\right]^2}{V(y)}$	`"jansen"`	Jansen (1999)
2	$\frac{\frac{1}{N}\sum_{v=1}^{N} f(\boldsymbol{A})_v\left[f(\boldsymbol{A})_v - f(\boldsymbol{A}_B^{(i)})_v\right]}{V(y)}$	`"sobol"`	Sobol' (2001)
3	$\frac{V(y) - \frac{1}{N}\sum_{v=1}^{N} f(\boldsymbol{A}_v) f(\boldsymbol{A}_B^{(i)})_v + f_0^2}{V(y)}$	`"homma"`	Homma and Saltelli (1996b)
4	$1 - \frac{\frac{1}{N}\sum_{v=1}^{N} f(\boldsymbol{B})_v f(\boldsymbol{B}_A^{(i)})_v - f_0^2}{\frac{1}{N}\sum_{v=1}^{N} f(\boldsymbol{A})_v^2 - f_0^2}$	`saltelli`	Saltelli *et al.* (2008)
5	$1 - \frac{\frac{1}{N}\sum_{v=1}^{N} f(\boldsymbol{A})_v f(\boldsymbol{A}_B^{(i)})_v - f_0^2}{\frac{1}{N}\sum_{v=1}^{N} \frac{f(\boldsymbol{A}_v)^2 + f(\boldsymbol{A}_B^{(i)})_v^2}{2} - f_0^2}$	`"janon"`	Janon, Klein, Lagnoux, Nodet, and Prieur (2014) Monod, Naud, and Makowski (2006)
6	$1 - \left[\frac{1}{N-1}\sum_{v=1}^{N} \frac{[f(\boldsymbol{A})_v - \langle f(\boldsymbol{A})_v\rangle]\left[f(\boldsymbol{A}_B^{(i)})_v - \left\langle f(\boldsymbol{A}_B^{(i)})_v\right\rangle\right]}{\sqrt{V[f(\boldsymbol{A})_v]V\left[f(\boldsymbol{A}_B^{(i)})_v\right]}}\right]$	`"glen"`	Glen and Isaacs (2012)
7	$\frac{\sum_{v=1}^{N}[f(\boldsymbol{B})_v - f(\boldsymbol{B}_A^{(i)})_v]^2 + [f(\boldsymbol{A})_v - f(\boldsymbol{A}_B^{(i)})_v]^2}{\sum_{v=1}^{N}[f(\boldsymbol{A})_v - f(\boldsymbol{B})_v]^2 + [f(\boldsymbol{B}_A^{(i)})_v - f(\boldsymbol{A}_B^{(i)})_v]^2}$	`"azzini"`	Azzini *et al.* (2020b)
8	$\frac{E_{x^*_{\sim i}}\left[\gamma_{x^*\sim i}(h_i)\right] + E_{x^*\sim i}\left[C_{x^*\sim i}(h_i)\right]}{V(y)}$	See Annex, section 6.2	Razavi and Gupta (2016b,a).

Table 2: Total-order estimators included in **sensobol** (v1.0.1).

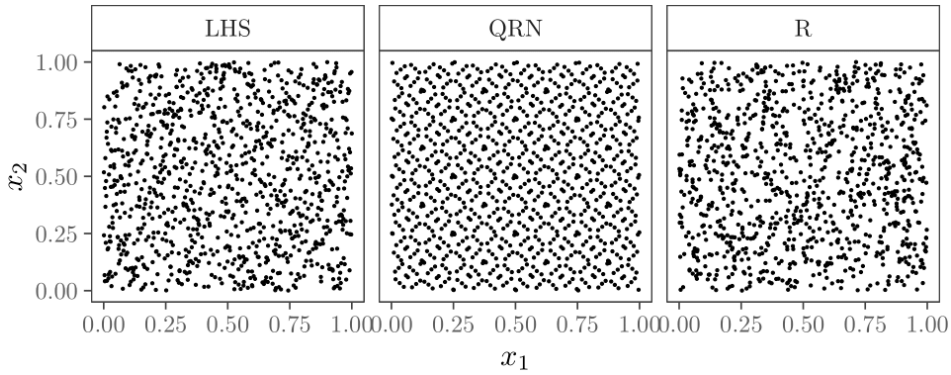

Figure 4: Sampling methods. Each dot is a sampling point. $N = 2^{10}$.

Once the sampling design is set, the computation of Sobol' indices is done with the function `sobol_indices()`. The arguments `first`, `total` and `matrices` are set by default at `first = "saltelli"`, `total = "jansen"` and `matrices = c("A", "B", "AB")` following best practices in sensitivity analysis (Saltelli *et al.* 2010; Puy *et al.* 2020a). However, any combination

$$
\begin{aligned}
\boldsymbol{Q} = & \overbrace{\qquad\qquad\qquad\qquad}^{\boldsymbol{A}} \overbrace{\qquad\qquad\qquad\qquad}^{\boldsymbol{B}} \\
& \begin{pmatrix}
0.50 & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 \\
0.75 & 0.25 & 0.75 & 0.25 & 0.75 & 0.25 & 0.75 & 0.25 \\
0.25 & 0.75 & 0.25 & 0.75 & 0.25 & 0.75 & 0.25 & 0.75 \\
0.38 & 0.38 & 0.62 & 0.12 & 0.88 & 0.88 & 0.12 & 0.62 \\
0.88 & 0.88 & 0.12 & 0.62 & 0.38 & 0.38 & 0.62 & 0.12
\end{pmatrix} \\
\boldsymbol{A}_B^{(1)} = & \begin{pmatrix}
0.50 & 0.50 & 0.50 & 0.50 \\
0.75 & 0.25 & 0.75 & 0.25 \\
0.25 & 0.75 & 0.25 & 0.75 \\
0.88 & 0.38 & 0.62 & 0.12 \\
0.38 & 0.88 & 0.12 & 0.62
\end{pmatrix} \\
\boldsymbol{A}_B^{(2)} = & \begin{pmatrix}
0.50 & 0.50 & 0.50 & 0.50 \\
0.75 & 0.25 & 0.75 & 0.25 \\
0.25 & 0.75 & 0.25 & 0.75 \\
0.38 & 0.88 & 0.62 & 0.12 \\
0.88 & 0.38 & 0.12 & 0.62
\end{pmatrix} \\
& \vdots \\
\boldsymbol{B}_A^{(1)} = & \begin{pmatrix}
0.50 & 0.50 & 0.50 & 0.50 \\
0.75 & 0.25 & 0.75 & 0.25 \\
0.25 & 0.75 & 0.25 & 0.75 \\
0.38 & 0.88 & 0.12 & 0.62 \\
0.88 & 0.38 & 0.62 & 0.12
\end{pmatrix} \\
\boldsymbol{B}_A^{(2)} = & \begin{pmatrix}
0.50 & 0.50 & 0.50 & 0.50 \\
0.75 & 0.25 & 0.75 & 0.25 \\
0.25 & 0.75 & 0.25 & 0.75 \\
0.88 & 0.38 & 0.12 & 0.62 \\
0.38 & 0.88 & 0.62 & 0.12
\end{pmatrix} \\
& \vdots
\end{aligned}
\tag{10}
$$

Figure 3: Example of the creation of an $\boldsymbol{A}$ and a $\boldsymbol{B}$ matrix, as well as $\boldsymbol{A}_B^{(i)}$ and $\boldsymbol{B}_A^{(i)}$ matrices. The $\boldsymbol{Q}$ matrix has been created with Sobol' (1967, 1976) Quasi-Random Numbers, $k = 4$ and $N = 5$. The figure is based on Puy *et al.* (2020a).

between all first and total-order estimators listed in Tables 1–2 is possible with the appro-

priate sampling design (Table 3). If the analyst selects estimators whose combination do not match the specific designs listed in Table 3, `sobol_indices()` will generate an error and urge to revise the specifications. This would be the case, for instance, if the analyst sets `first = "sobol"`, `total = "glen"` and `matrices = "c("A", "AB", "BA")`.

first	total	matrices	Nº model runs
"saltelli" "jansen"	"jansen" "sobol" "homma" "janon" "glen"	c("A", "B", "AB")	$N(k+2)$
"sobol"	"saltelli"	c("A", "B", "BA")	$N(k+2)$
"azzini"	"jansen" "sobol" "homma" "janon" "glen" "azzini" "saltelli"	c("A", "B", "AB", "BA")	$2N(k+1)$
"saltelli" "jansen" "sobol" "azzini"	"azzini"	c("A", "B", "AB", "BA")	$2N(k+1)$

Table 3: Available combinations of first and total-order estimators in **sensobol** (v1.0.1).

3. Usage

In this section we illustrate the functionality of **sensobol** through three different examples of increasing complexity. Let us first load the required packages:

```
R> library("sensobol")
R> library("data.table")
R> library("ggplot2")
```

3.1. Example 1: The Sobol' G function

In sensitivity analysis, the accuracy of sensitivity estimators is usually checked against test functions for which the variance and the sensitivity indices can be expressed analytically. **sensobol** includes five of these test functions: Ishigami and Homma (1990)'s, Sobol' (1998)'s (known as G function), Bratley, Fox, and Niederreiter (1992)'s, Bratley and Fox (1988)'s and Oakley and O'Hagan (2004)'s, as well as Becker (2020)'s metafunction (Table 4).

In this first example we illustrate the functionality of **sensobol** with the Sobol' G function, one of the most used benchmark functions in sensitivity analysis (Lo Piano *et al.* 2021; Puy, Lo Piano, and Saltelli 2020b; Saltelli *et al.* 2010). In its current implementation, the Sobol' G is an eight-dimension function with $S_1 > S_2 > S_3 > S_4$ and $(S_5, \ldots, S_8) \approx 0$ (Table 4, Nº 2). With this parametrization the Sobol' G function is a Type A function according to Kucherenko *et al.* (2011)'s taxonomy (a function with few important factors and minor interactions), with Type B and Type C functions designing those with equally important parameters but with few and large interactions respectively.

We first define the settings of the uncertainty and sensitivity analysis: we set the sample size N of the base sample matrix and the number of uncertain parameters k, and create a vector with the parameters' name. Since we will bootstrap the indices to get confidence intervals, we also set the number of bootstrap replicas R and define the bootstrap confidence interval method, which in this case will be the normal method. Finally, we set the confidence intervals at 0.95:

Nº	Test function	Author
1	$y = \sin(x_1) + a\sin(x_2)^2 + bx_3^4\sin(x_1)$, where $a = 2, b = 1$ and $(x_1, x_2, x_3) \sim \mathcal{U}(-\pi, +\pi)$	Ishigami and Homma (1990)
2	$y = \prod_{i=1}^{k} \frac{\lvert 4x_i - 2\rvert + a_i}{1 + a_i}$, where $k = 8$, $x_i \sim \mathcal{U}(0,1)$ and $a = (0, 1, 4.5, 9, 99, 99, 99, 99)$	Sobol' (1998)
3	$y = \sum_{i=1}^{k}(-1)^i \prod_{j=1}^{i} x_j$, where $x_i \sim \mathcal{U}(0,1)$	Bratley *et al.* (1992)
4	$y = \prod_{i=1}^{k} \lvert 4x_i - 2\rvert$, where $x_i \sim \mathcal{U}(0,1)$	Bratley and Fox (1988)
5	$y = \boldsymbol{a}_1^T\boldsymbol{x} + \boldsymbol{a}_2^T sin(\boldsymbol{x}) + \boldsymbol{a}_3^T cos(\boldsymbol{x}) + \boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x}$, where $\boldsymbol{x} = x_1, x_2, \ldots, x_k$, $k = 15$, and values for $\boldsymbol{a}_i^T, i = 1, 2, 3$ and $\boldsymbol{M}$ are defined by the authors	Oakley and O'Hagan (2004)
6	$y = \sum_{i=1}^{k} \alpha_i f^{u_i}(x_i) + \sum_{i=1}^{k_2} \beta_i f^{u_{V_{i,1}}}(x_{V_{i,1}}) f^{u_{V_{i,2}}}(x_{V_{i,2}}) + \sum_{i=1}^{k_3} \gamma_i f^{u_{W_{i,1}}}(x_{W_{i,1}}) f^{u_{W_{i,2}}}(x_{W_{i,2}}) f^{u_{W_{i,3}}}(x_{W_{i,3}})$	See Becker (2020) and Puy *et al.* (2020a) for details.

Table 4: Test functions included in **sensobol** (v1.0.0).

```
R> N <- 2 ^ 10
R> k <- 8
R> params <- paste("$x_", 1:k, "$", sep = "")
R> R <- 10^3
R> type <- "norm"
R> conf <- 0.95
```

The next step is to create the sample matrix. In this specific case we will use an $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{A}_B^{(i)}$ design and Sobol' quasi-random numbers to compute first and total-order indices. These are default settings in `sobol_matrices()`. In our call to the function we only need to define the sample size and the parameters:

```
R> mat <- sobol_matrices(N = N, params = params)
```

Once the sample matrix is defined we can run our model. Note that in `mat` each column is a model input and each row a sample point, so the model has to be coded as to run rowwise. This is already the case of the Sobol' G function included in **sensobol**:

```
R> y <- sobol_Fun(mat)
```

The package also allows the user to swiftly visualize the model output uncertainty by plotting an histogram of the model output obtained from the $\boldsymbol{A}$ matrix:

```
R> plot_uncertainty(Y = y, N = N) + labs(y = "Counts", x = "$y$")
```

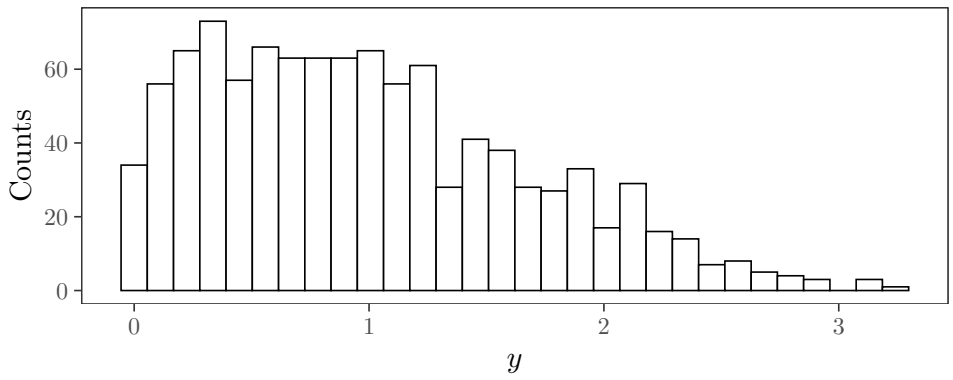

Figure 5: Empirical distribution of the Sobol' G model output.

Before computing Sobol' indices, it is recommended to explore how the model output maps onto the model input space. **sensobol** includes two functions to that aim, `plot_scatter()` and `plot_multiscatter()`. The first displays the model output y against x_i while showing the mean y value (i.e., as in Figures 1–2), and allows the user to identify patterns denoting sensitivity (Pianosi, Beven, Freer, Hall, Rougier, Stephenson, and Wagener 2016).

```
R> plot_scatter(data = mat, N = N, Y = y, params = params)
```

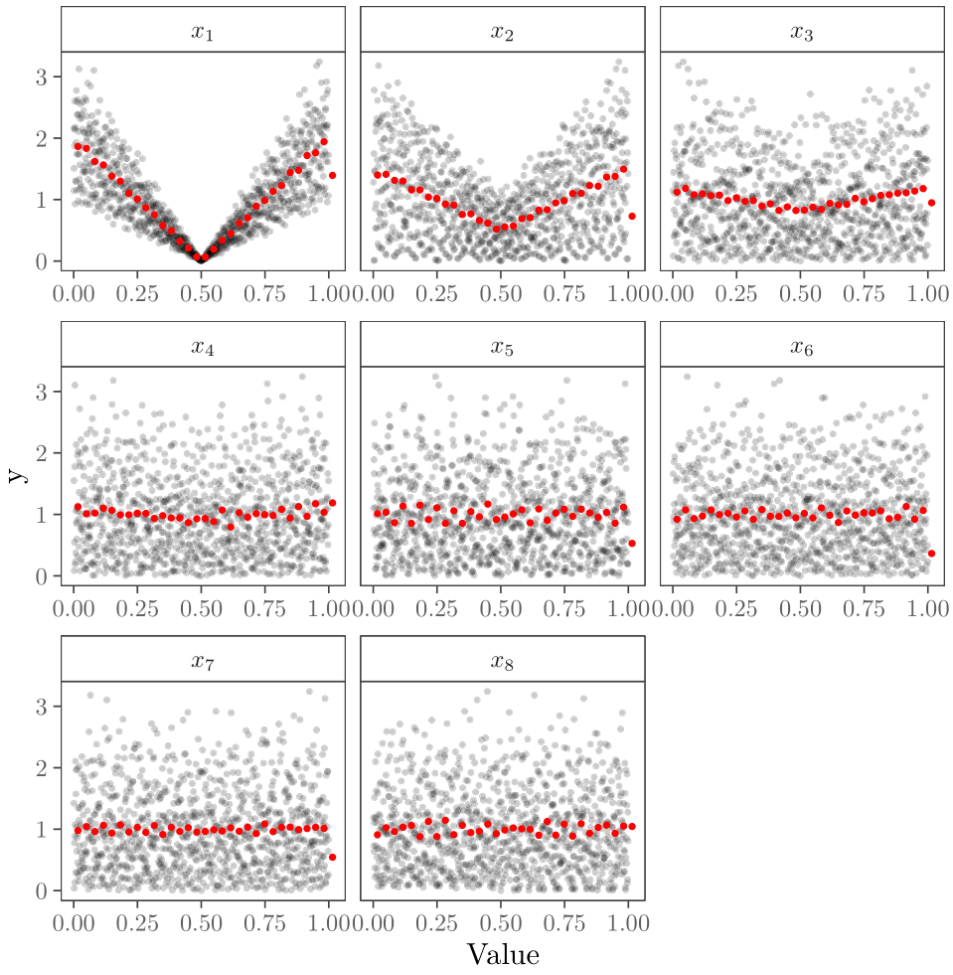

Figure 6: Scatter plots of model inputs against the model output for the Sobol' G function.

The scatter plots in Figure 6 evidence that x_1, x_2 and x_3 have more "shape" than the rest and thus have a higher influence on y than $(x_4, \ldots, x_8)$. However, scatter plots do not always permit to detect which parameters have a joint effect on the model output. To gain a first insight on these interactions, the function `plot_multiscatter()` plots x_i against x_j and maps the resulting coordinate to its respective model output value. Interactions are then visible by the emergence of colored patterns.

By default, `plot_multiscatter()` plots all possible combinations of x_i and x_j, which equal $\frac{k!}{2!(k-2)!} = 6$ possible combinations in this specific case. In high-dimensional models with several inputs this might lead to overplotting. To avoid this drawback, the user can subset the parameters they wish to focus on following the results obtained with `plot_scatter()`: if x_i does not show "shape" in the scatterplots of x_i against y, then it may be excluded from `plot_multiscatter()`.

Below we plot all possible combinations of pairs of inputs between $x_1 - x_4$, which are influential according to Figure 6:

```
R> plot_multiscatter(data = mat, N = N, Y = y, params = paste("$x_", 1:4, "$",
+   sep = ""))
```

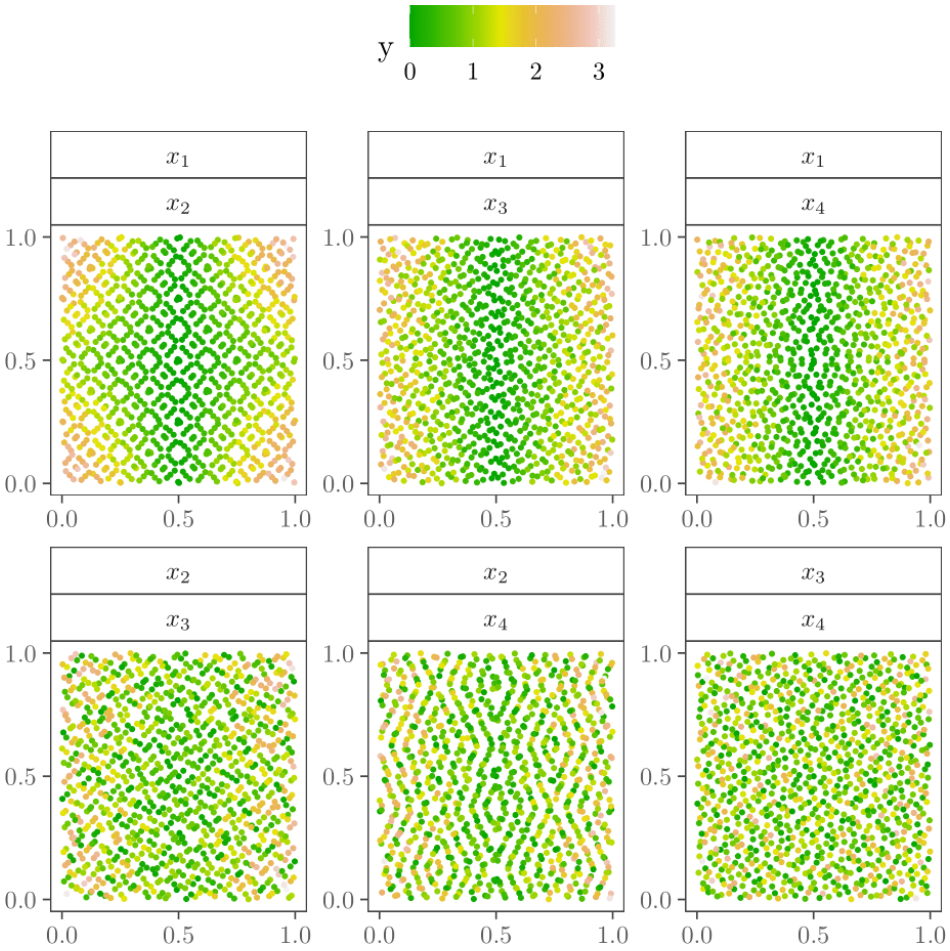

Figure 7: Scatterplot matrix of pairs of model inputs for the Sobol' G function. The topmost and bottommost label facets refer to the x and the y axis respectively.

The results suggest that x_1 might interact with x_2 given the colored pattern of the (x_1, x_2) facet: the highest values of the model output are concentrated in the corners of the (x_1, x_2) input space and thus result from combinations of high/low x_1 values with high/low x_2 values. In case the analyst is interested in assessing the exact weight of this high-order interaction, the computation of second-order indices would be required.

The last step is the computation of Sobol' indices. In this specific case, we set `boot = TRUE` to bootstrap the Sobol' indices and get confidence intervals:

```
R> ind <- sobol_indices(Y = y, N = N, params = params, boot = TRUE, R = R,
+   type = type, conf = conf)
```

The output of `sobol_indices()` is an S3 object of class `sensobol`, with the results stored in the component `results`. To improve the visualization of the object, we set the number of digits in each numerical column to 3:

```
R> cols <- colnames(ind$results)[1:5]
```

```
R> ind$results[, (cols):= round(.SD, 3), .SDcols = (cols)]
R> ind


First-order estimator: saltelli | Total-order estimator: jansen

Total number of model runs: 10240

Sum of first order indices: 0.9419303
    original   bias std.error low.ci high.ci sensitivity parameters
 1:    0.724  0.005     0.069  0.584   0.854          Si      $x_1$
 2:    0.184 -0.002     0.039  0.110   0.261          Si      $x_2$
 3:    0.025  0.000     0.015 -0.005   0.054          Si      $x_3$
 4:    0.010  0.000     0.008 -0.007   0.025          Si      $x_4$
 5:    0.000  0.000     0.001 -0.001   0.002          Si      $x_5$
 6:    0.000  0.000     0.001 -0.002   0.002          Si      $x_6$
 7:    0.000  0.000     0.001 -0.001   0.002          Si      $x_7$
 8:    0.000  0.000     0.001 -0.002   0.002          Si      $x_8$
 9:    0.799  0.001     0.036  0.728   0.868          Ti      $x_1$
10:    0.243  0.000     0.013  0.217   0.269          Ti      $x_2$
11:    0.035  0.000     0.002  0.030   0.039          Ti      $x_3$
12:    0.011  0.000     0.001  0.009   0.012          Ti      $x_4$
13:    0.000  0.000     0.000  0.000   0.000          Ti      $x_5$
14:    0.000  0.000     0.000  0.000   0.000          Ti      $x_6$
15:    0.000  0.000     0.000  0.000   0.000          Ti      $x_7$
16:    0.000  0.000     0.000  0.000   0.000          Ti      $x_8$
```

The output informs of the first and total-order estimators used in the calculation, the total number of model runs and the sum of the first-order indices: if $(\sum_{i=1}^{k} S_i) < 1$, the model is non-additive.

When `boot = TRUE`, the output of `sobol_indices()` displays the bootstrap statistics in the five leftmost columns (the observed statistic, the bias, the standard error and the low and high confidence intervals), and two extra columns linking each statistic to a sensitivity index (`sensitivity`) and a parameter (`parameters`). If `boot = FALSE`, `sobol_indices()` computes a point estimate of the indices and the output includes only the columns `original`, `sensitivity` and `parameters`.

The results indicate that x_1 is responsible for 72% of the uncertainty in y, followed by x_2 (18%). x_3 and x_4 have a very minor first-order effect, while the rest are non-influential. Note that the observed statistics suggest the presence of non-additivities: T_1 and T_2 (0.79 and 0.24) are respectively higher than S_1 and S_2 (0.72 and 0.18). As we have seen in Figure 7, x_1 and x_2 have a non-additive effect in y.

We can also compute the Sobol' indices of a dummy parameter, i.e., a parameter that has no influence on the model output, to estimate the numerical approximation error. This will be used later on to identify parameters whose contribution to the output variance is less than the approximation error, and therefore can not be considered influential. Like `sobol_indices()`, the function `sobol_dummy()` allows to obtain point estimates (the default) or bootstrap estimates. In this example we use the latter option:

```
R> ind.dummy <- sobol_dummy(Y = y, N = N, params = params, boot = TRUE, R = R)
```

The last stage is to plot the Sobol' indices and their confidence intervals, as well as the Sobol' indices of a dummy parameter, with the function `plot_sobol()`:

```
R> plot(ind, dummy = ind.dummy)
```

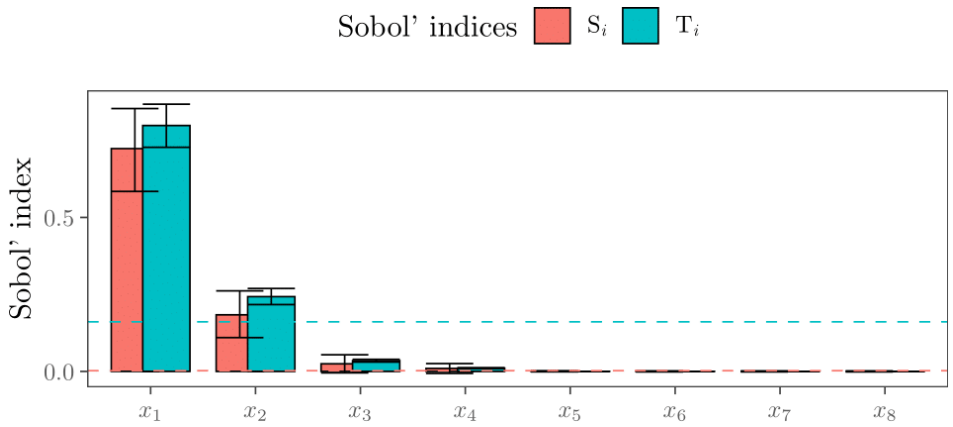

Figure 8: Sobol' indices of the Sobol' G function.

The error bars of S_1 and S_2 overlap with those of T_1 and T_2 respectively. In the case of the Sobol' G function we know that $T_1 > S_1$ and $T_2 > S_2$ because the analytic variance is known, but for models where this is not the case such overlap might hamper the identification of non-additivities. Under these circumstances, narrower confidence intervals can be obtained by increasing the sample size N and re-running the analysis from the creation of the sample matrix onwards.

The horizontal, blue / red dashed lines respectively mark the upper limit of the T_i and S_i indices of the dummy parameter. This helps in identifying which parameters condition the model output given the sample size constraints of the analysis: only parameters whose lower confidence intervals are not below the S_i and T_i indices of the dummy parameter can be considered truly influential, in this case x_1 and x_2. Note that although $T_3 \neq 0$, the T_i index of the dummy parameter is higher than T_3 and therefore T_3 can not be distinguished from the approximation error.

3.2. Example 2: A logistic population growth model

In this section we show how **sensobol** can be implemented to conduct a global uncertainty and sensitivity analysis of a dynamic model. We will use the following logistic population growth model:

$$\frac{dN}{dt} = rN(1 - \frac{N}{K}) \tag{11}$$

Malthusian models of population growth (i.e., exponential growth) can not forever describe the growth of a population because resources, including space, are limited and competitive

pressures ultimately impose limits on growth. Most ways to incorporate that limit to growth in models share similar dynamics, and the most intuitive and widely used is the form proposed by Verhulst in Equation 11, which was popularized in Ecology by Pearl and Reed (1920). In this model, the population N at time t is dependent on the growth rate r, the number of individuals N and the carrying capacity K, defined as the maximum number of individuals that a given environment can sustain. When N approaches K, the population growth slows down until the number of individuals converges to a constant (Figure 9).

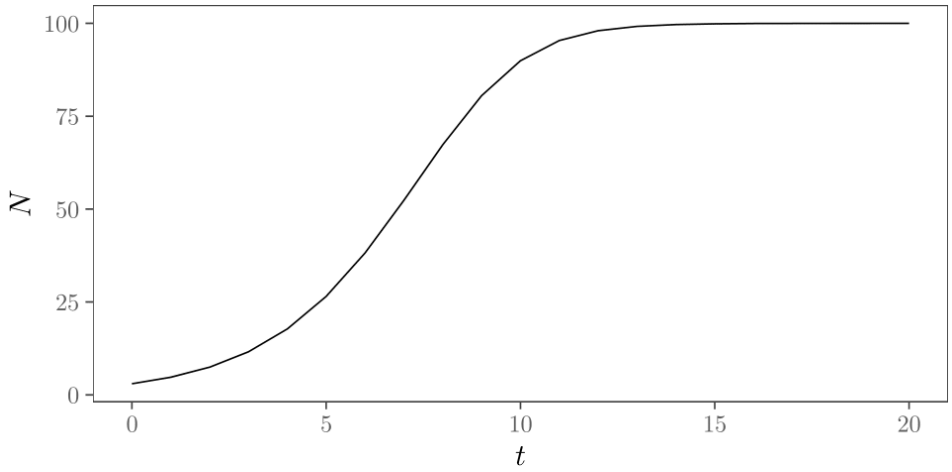

Figure 9: Dynamics of the logistic population growth model for $N_0 = 3$, $r = 0.6$ and $K = 100$.

Before moving forward to conduct a global sensitivity analysis of Equation 11, we set the sample size N of the base sample matrix at 2^{13} and create a vector with the name of the parameters. For this specific example we will use the Azzini *et al.* (2020b) estimators, which require a sampling design based on $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{A}_B^{(i)}$, $\boldsymbol{B}_A^{(i)}$ matrices. We will compute up to second-order effects, bootstrap the indices 10^3 times and compute the 95% confidence intervals using the percentile method.

```
R> N <- 2 ^ 13
R> params <- c("$r$", "$K$", "$N_0$")
R> matrices <- c("A", "B", "AB", "BA")
R> first <- total <- "azzini"
R> order <- "second"
R> R <- 10 ^ 3
R> type <- "percent"
R> conf <- 0.95
```

In the next two code snippets we code Equation 11 and wrap it up in a `mapply()` call to make it run rowwise:

```
R> population_growth <- function (r, K, X0) {
+   X <- X0
+   for (i in 0:20) {
+     X <- X + r * X * (1 - X / K)
```

```
+     }
+   return (X)
+ }
```

```
R> population_growth_run <- function (dt) {
+   return(mapply(population_growth, dt[, 1], dt[, 2], dt[, 3]))
+ }
```

We now construct the sample matrix. This time we set `type = "LHS"` to use a Latin Hypercube Sampling Design:

```
R> mat <- sobol_matrices(matrices = matrices, N = N, params = params,
+   order = order, type = "LHS")
```

Let's assume that, after surveying the literature and conducting fieldwork, we have agreed that the uncertainty in the model inputs can be fairly approximated with the distributions presented in Table 5. We thus transform each model input in `mat` to its specific probability distribution using the appropriate inverse quantile transformation:

Table 5: Summary of the parameters and their distributions (Chalom and Inacio Knegt Lopez 2017).

Parameter	Description	Distribution
r	Population growth rate	$\mathcal{N}(1.7, 0.3)$
K	Maximum carrying capacity	$\mathcal{N}(40, 1)$
N_0	Initial population size	$\mathcal{U}(10, 50)$

```
R> mat[, "$r$"] <- qnorm(mat[, "$r$"], 1.7, 0.3)
R> mat[, "$K$"] <- qnorm(mat[, "$K$"], 40, 1)
R> mat[, "$N_0$"] <- qunif(mat[, "$N_0$"], 10, 50)
```

The sample matrix in `mat` is now ready, and we can run the model:

```
R> y <- population_growth_run(mat)
```

And display the model output uncertainty:

```
R> plot_uncertainty(Y = y, N = N) + labs(y = "Counts", x = "$y$")
```

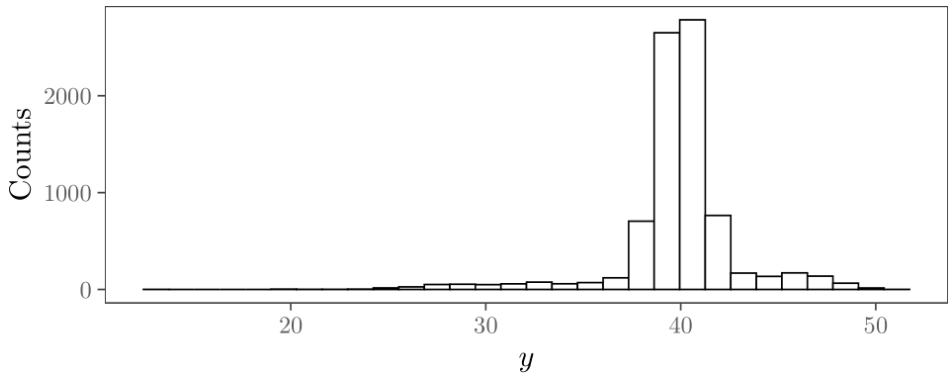

Figure 10: Empirical distribution of the logistic population growth model output.

The results suggest that after 20 time steps the number of individuals will most likely concentrate around 40. Note however the right and left tails of the distribution, indicating that a few simulations also yielded significantly lower and higher population values. These tails result from some specific combinations of parameter values and are indicative of interaction effects, which will be explored later on.

We can also compute some statistics to get a better grasp of the output distribution, such as quantiles:

```
R> quantile(y, probs = c(0.01, 0.025, 0.5, 0.975, 0.99, 1))
```

```
      1%     2.5%      50%    97.5%      99%     100%
27.80714 30.66101 40.00111 46.64511 47.91589 53.41604
```

With `plot_scatter()` we can map the model inputs onto the model output. Instead of plotting one dot per simulation, in this example we use hexagon bins by setting `method = "bin"` and internally calling `ggplot2::geom_hex()`. With this specification we divide the plane into regular hexagons, count the number of hexagons and map the number of simulations to the hexagon bin. `method = "bin"` is therefore a useful resource to avoid overplotting with `plot_scatter()` when the sample size of the base sample matrix (N) is high, as in this case ($N = 2^{13}$):

```
R> plot_scatter(data = mat, N = N, Y = y, params = params, method = "bin")
```

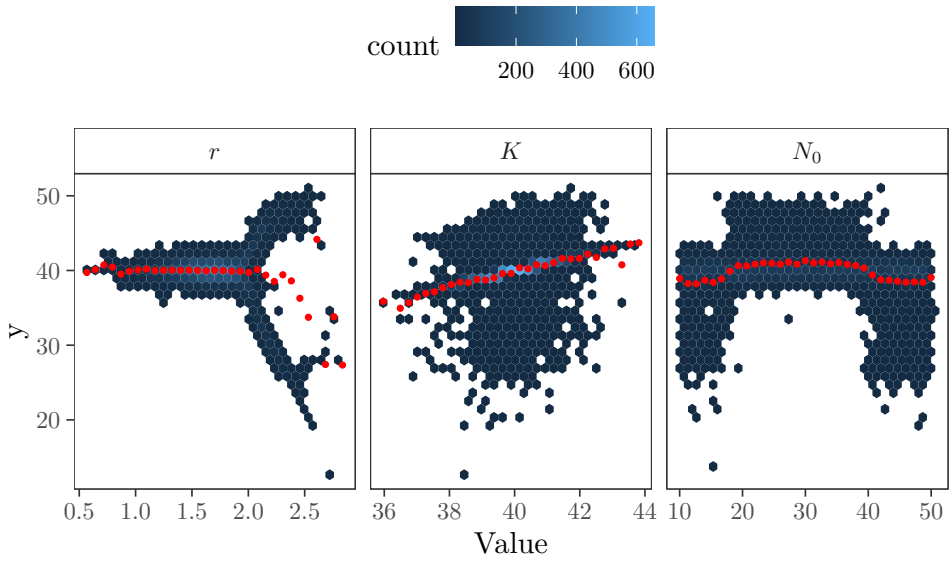

Figure 11: Scatterplot of model inputs against the model output for the population growth model.

Zones with a higher density of dots are highlighted by lighter blue colors. Note also the bifurcation of the model output at $r \approx 2$. This behavior is the result of the discretization of the logistic (May and Oster 1976). At $r > 2$, a cycle of length 2 emerges, followed as r is increased further by an infinite sequence of period-doubling bifurcations approaching a critical value after which chaotic behavior and strange attractors result.

We can also avoid overplotting in `plot_multiscatter()` by randomly sampling and displaying only n simulations. This is controlled by the argument `smpl`, which is `NULL` by default. Here we set `smpl` at 2^{11} and plot only 1/4th of the simulations.

```
R> plot_multiscatter(data = mat, N = N, Y = y, params = params, smpl = 2^11)
```

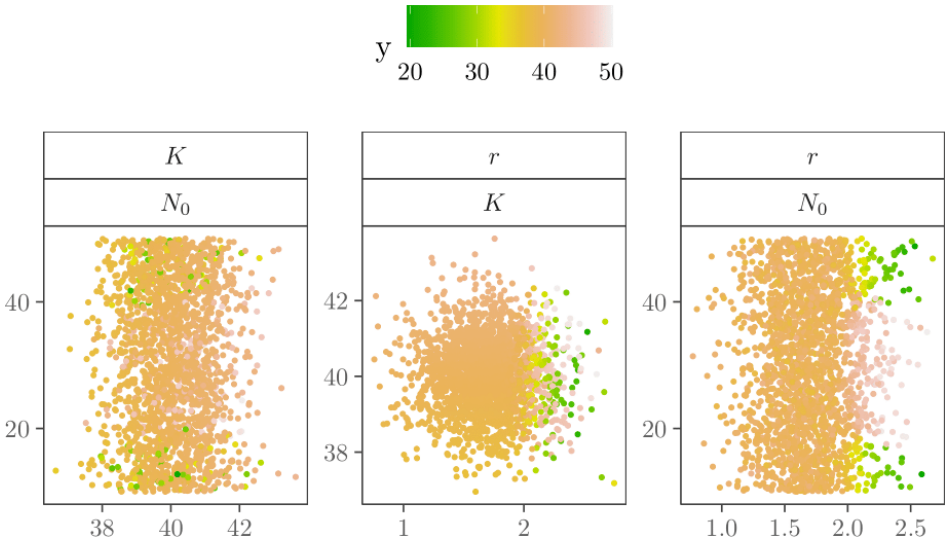

Figure 12: Scatterplot matrix of pairs of model inputs for the population growth model.

The results suggest that there may be interactions between (r, K) and (r, N_0): note the yellow-green colours concentrated on the right side of the (r, K) plot as well as on the top right and bottom right sides of the (r, N_0) plot. In the case of the pair (K, N_0), no clear pattern emerges.

These interactions, as well as the first-order effects of N_0, r, K, can be quantified with a call to `sobol_indices()`:

```
R> ind <- sobol_indices(matrices = matrices, Y = y, N = N, params = params,
+   first = first, total = total, order = order, boot = TRUE, R = R,
+   parallel = "no", type = type, conf = conf)
```

We round the number of digits of the numeric columns to 3 to better inspect the results, and print `ind`:

```
R> cols <- colnames(ind$results)[1:5]
R> ind$results[, (cols):= round(.SD, 3), .SDcols = (cols)]
R> ind
```

```
First-order estimator: azzini | Total-order estimator: azzini

Total number of model runs: 114688

Sum of first order indices: 0.2569059
   original   bias std.error low.ci high.ci sensitivity parameters
1:    0.028 -0.001     0.019 -0.010   0.067          Si         $r$
2:    0.114  0.000     0.004  0.107   0.122          Si         $K$
3:    0.115  0.000     0.011  0.093   0.136          Si       $N_0$
```

```
4:    0.760  0.000     0.011  0.737   0.781          Ti        $r$
5:    0.185  0.000     0.010  0.164   0.206          Ti        $K$
6:    0.861  0.001     0.020  0.825   0.900          Ti      $N_0$
7:   -0.004  0.000     0.010 -0.024   0.015         Sij    $r$.$K$
8:    0.673  0.001     0.022  0.627   0.718         Sij  $r$.$N_0$
9:    0.012  0.000     0.007 -0.002   0.025         Sij  $K$.$N_0$
```

Since in this example we have set `order = "second"`, the output also displays the second-order effects (S_{ij}) of the pairs (r, K), (r, N_0) and (K, N_0). Note that S_{r,N_0} conveys $\sim 67\%$ of the uncertainty in y, and that $S_r + S_K + S_{N_0} = 2.8 + 11.4 + 11.5 \approx 25\%$, meaning that first-order effects are responsible for only *circa* 1/4 th of the model output variance. The model behaviour is largely driven by a coupled effect which would have passed unnoticed should we had relied on a local sensitivity analysis approach, i.e., if we had moved one parameter at-a-time.

In any case, K and N_0 have the higher first-order effect in the model output. If the aim is to reduce the uncertainty in the prediction (i.e., to better assess the potential impact of a species on a territory), these results suggest to focus the efforts on better quantifying the initial population N_0, and/or on conducting further research on what is the maximum carrying capacity K of the environment for this particular species. Priority should be given to N_0 given its strong interaction with r.

In order to get an estimation of the approximation error we compute the Sobol' indices also for the dummy parameter:

```
R> ind.dummy <- sobol_dummy(Y = y, N = N, params = params, boot = TRUE, R = R)
```

And plot the output:

```
R> plot(ind, dummy = ind.dummy)
```

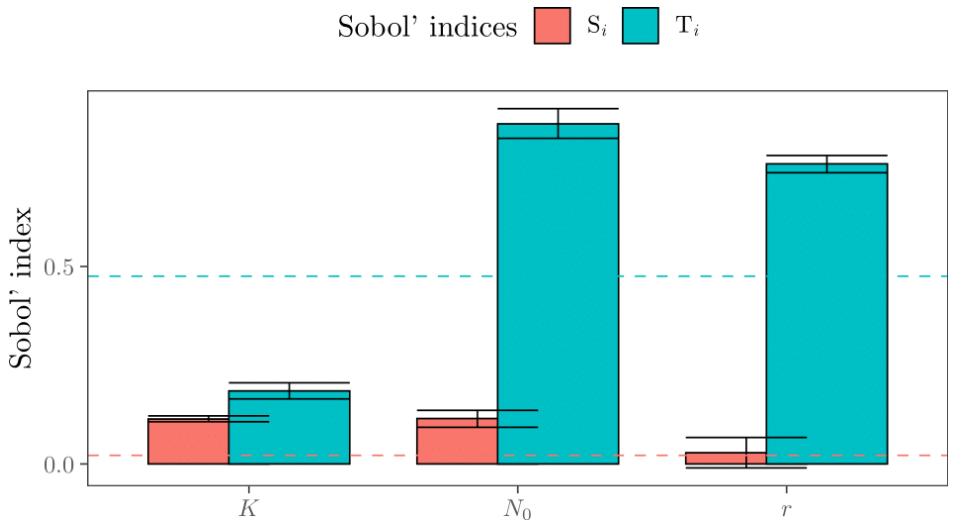

Figure 13: First and total-order Sobol' indices of the population growth model.

Note the importance of interactions, reflected in $T_{N_0} \gg S_{N_0}$ and $T_r \gg S_r$. It is also important to highlight that T_K is below the T_i index of the dummy parameter (the dashed, horizontal blue line at c. 0.5). This makes T_K indistinguishable from the approximation error. The same applies to S_r, whose 95% confidence interval overlaps with the S_i of the dummy parameter (the dashed, red line).

Finally, we can also plot second-order indices by setting `order = "second"` in a `plot()` call:

```
R> plot(ind, order = "second")
```

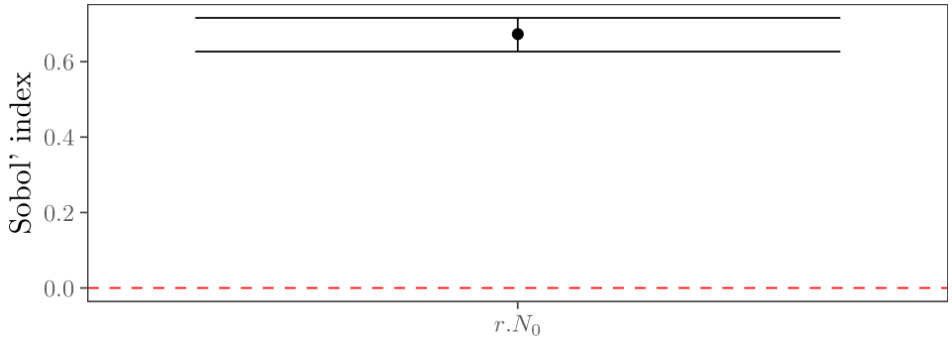

Figure 14: Second-order Sobol' indices.

Only S_{r,N_0} is displayed because `plot()` only returns second-order indices for which the lower confidence interval does not overlap with zero.

3.3. Example 3: The spruce budworm and forest model

The last example illustrates the flexibility of **sensobol** against systems of differential equations. Under these circumstances, the analyst might be interested in exploring the sensitivity of each model output or state variable to the uncertain inputs at different points in time, i.e., at the transient phase and/or at equilibrium. **sensobol** integrates with the **deSolve** and the **data.table** packages to achieve this goal in an easy and user-friendly manner (Soetaert, Petzoldt, and Setzer 2010; Dowle and Srinivasan 2019).

We rely here on the spruce budworm and forest model of Ludwig *et al.* (1976). Spruce budworm is a devastating pest of Canadian and high-latitude US balsam fir and spruce forests. A half-century ago, research teams led by Crawford Holling developed detailed models of the interaction between the budworm and their target species, models capable of reproducing the boom-and-bust dynamics and spatial patterns exhibited in real forests. Donald Ludwig pointed out that these models were overparameterized and that much simpler models could capture the essential dynamics in a more robust manner.

The basic idea of the model is that the dynamics of the system play out on multiple time scales. In particular, budworm population dynamics respond to forest quality on a fast time scale, leading to changes in forest quality on a slower time scale. In turn, the slow dynamics change the topological profile of the fast-time scale dynamics, introducing hysteretic oscillations reminiscent of relaxation oscillations. The simplest version of the budworm model in (Ludwig *et al.* 1976, Equations 3, 10, 11) can be non-dimensionalized easily, making transparent the reduction in dimension on the fast-time scale. In place of those, however, we consider

the more complicated version given by Equations 20-22 in that paper, yielding the explicit form

$$\begin{aligned}
\frac{dB}{dt} &= r_B B\left(1 - \frac{B}{KS}\frac{T_E^2 + E^2}{E^2}\right) - \beta\frac{B^2}{(\alpha S)^2 + B^2} \\
\frac{dS}{dt} &= r_S S\left(1 - \frac{SK_E}{EK_S}\right) \\
\frac{dE}{dt} &= r_E E\left(1 - \frac{E}{K_E}\right) - P\frac{B}{S}\frac{E^2}{T_E^2 + E^2}
\end{aligned}, \tag{12}$$

where B, S and E are the budworm density, the average size of the trees and the energy reserve of the trees respectively (Figure 15). Equation 12 allows a full characterization of the parameter space with empirical data (Table 6).

Table 6: Summary of the parameters and their distribution. *DU* stands for discrete uniform (Ludwig *et al.* 1976, Table 1).

Parameter	Description	Distribution
r_B	Intrinsic budworm growth rate	$\mathcal{U}(1.52, 1.6)$
K	Maximum budworm density	$\mathcal{U}(100, 355)$
β	Maximum budworm predated	$\mathcal{U}(20000, 43200)$
α	$\frac{1}{2}$ maximum density for predation	$\mathcal{U}(1, 2)$
r_S	Intrinsic branch growth rate	$\mathcal{U}(0.095, 0.15)$
K_S	Maximum branch density	$\mathcal{U}(24000, 25440)$
K_E	Maximum E level	$\mathcal{U}(1, 1.2)$
r_E	Intrinsic E growth rate	$\mathcal{U}(0.92, 1)$
P	Consumption rate of E	$\mathcal{U}(0.0015, 0.00195)$
T_E	E proportion	$\mathcal{U}(0.7, 0.9)$

Like in the previous examples, we first define the sample size of the base sample matrix N, a vector with the name of the parameters, the order of the sensitivity indices investigated, the number of bootstrap replicas R and the type of confidence intervals. Furthermore, we create a time sequence `timeOutput` specifying the times at which we will extract the model output to conduct the sensitivity analysis (25, 50, 75, 100, 125, 150). These time slices have been selected to get an insight into all the stages of the model (i.e., growth, equilibirum) (Figure 15).

```
R> N <- 2 ^ 9
R> params <- c("r_b", "K", "beta", "alpha", "r_s", "K_s", "K_e", "r_e", "P",
+   "T_e")
R> order <- "first"
R> R <- 10 ^ 3
R> type <- "norm"
R> conf <- 0.95
R> timeOutput <- seq(25, 150, 25)
```

Since the model in Equation 12 is a system of differential equations, we can easily code it following the guidelines set by the **deSolve** package (Soetaert *et al.* 2010):

```
R> budworm_fun <- function(t, state, parameters) {
+   with(as.list(c(state, parameters)), {
+     dB <- r_b * B * (1 - B / (K * S) * (T_e^2 + E^2) / E^2) - beta *
+       B^2/((alpha ^ S)^2 + B^2)
+     dS <- r_s * S * (1 - (S * K_e) / (E * K_s))
+     dE <- r_e * E * (1 - E / K_e) - P * (B / S) * E^2/(T_e^2 + E^2)
+     list(c(dB, dS, dE))
+   })
+ }
```

We can then create the sample matrix as in the previous examples:

```
R> mat <- sobol_matrices(N = N, params = params, order = order)
```

And transform each column to the probability distributions specified in Table 6:

```
R> mat[, "r_b"] <- qunif(mat[, "r_b"], 1.52, 1.6)
R> mat[, "K"] <- qunif(mat[, "K"], 100, 355)
R> mat[, "beta"] <- qunif(mat[, "beta"], 20000, 43200)
R> mat[, "alpha"] <- qunif(mat[, "alpha"], 1, 2)
R> mat[, "r_s"] <- qunif(mat[, "r_s"], 0.095, 0.15)
R> mat[, "K_s"] <- qunif(mat[, "K_s"], 24000, 25440)
R> mat[, "K_e"] <- qunif(mat[, "K_e"], 1, 1.2)
R> mat[, "r_e"] <- qunif(mat[, "r_e"], 0.92, 1)
R> mat[, "P"] <- qunif(mat[, "P"], 0.0015, 0.00195)
R> mat[, "T_e"] <- qunif(mat[, "T_e"], 0.7, 0.9)
```

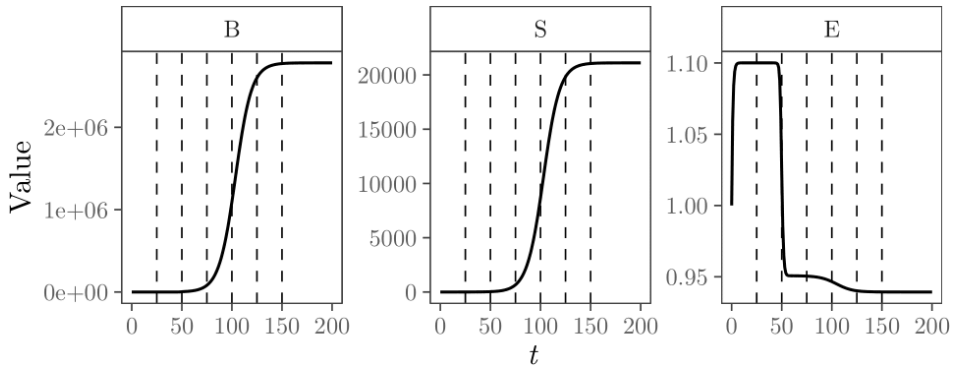

Figure 15: Dynamics of the spruce budworm and forest model. The vertical, dashed lines mark the times at which we will conduct the sensitivity analysis. Initial state values: $B = 1, S = 0.07, E = 1$. The parameter values are the mean values of the distributions shown in Table 6.

Since we wish to appraise how the uncertainties in the model inputs affect the three state variables through time, we need to run `budworm_fun()` rowwise over `mat` up to each of the times specified in `timeOutput`, and retrieve the correspondent model output. In order to speed up the computation, it might be convenient to arrange a parallel setting. To that aim, we load the packages `foreach` (Microsoft and Weston 2020b), `parallel` (R Core Team 2020), and `doParallel` (Microsoft and Weston 2020a).

```
R> library("foreach")
R> library("parallel")
R> library("doParallel")
```

In the next code snippet we design a nested loop to conduct the computations. Note that the function `budworm_fun()` is called through **sensobol**'s `sobol_ode()`, a wrapper around **deSolve**'s `ode` function which allows to retrieve only the model output at time t. In the nested loop, `sobol_ode()` runs `budworm_fun()` up to the j-th element of `timeOutput` with the value of the parameters as defined in the i-th row of `mat`. Once it has ran over all the rows of `mat` it starts over again, and computes the model output up to the $j + 1$ time with the parameters as defined in the i-th row. Since $j = 25, 50, 75, 100, 125, 150$, the model will run a total of six times over all the rows in `mat`.

Before executing the nested loop, we order the computer to use 75% of the cores available in order to take advantage of parallel computing.

```
R> n.cores <- makeCluster(floor(detectCores() * 0.75))
R> registerDoParallel(n.cores)
R> y <- foreach(j = timeOutput, .combine = "rbind") %:%
+   foreach(i = 1:nrow(mat), .combine = "rbind",
+           .packages = "sensobol") %dopar%
+   {
+     sobol_ode(d = mat[i, ], times = seq(0, j, 1),
+   state = c(B = 0.1, S = 007, E = 1), func = budworm_fun)
+     }
R> stopCluster(n.cores)
```

Now we can rearrange the data for the sensitivity analysis. We first create a column to link the model output with each time in `timeOutput`:

```
R> full.dt <- data.table(cbind(y, times = rep(timeOutput, each = nrow(mat))))
R> print(full.dt)
```

```
              B           S         E times
   1:   16164.60   131.65941 0.9505648    25
   2:   16502.68   182.76395 1.0475317    25
   3:   14454.00    94.79514 0.8514826    25
   4:   20518.10   215.36948 0.9395692    25
   5:   19044.85   111.68735 0.8902224    25
  ---
```

```
36860: 2124030.97 21470.17732 0.8937556   150
36861: 3439376.26 21032.03802 0.9330312   150
36862: 1475659.57 22284.33814 1.0787991   150
36863: 2656465.49 21815.62825 0.9380807   150
36864: 2275279.86 21485.64371 0.9975085   150
```

And transform the resulting `data.table` from wide to long format:

```
R> indices.dt <- melt(full.dt, measure.vars = c("B", "S", "E"))
R> print(indices.dt)
```

```
        times variable        value
     1:    25        B 1.616460e+04
     2:    25        B 1.650268e+04
     3:    25        B 1.445400e+04
     4:    25        B 2.051810e+04
     5:    25        B 1.904485e+04
    ---
110588:   150        E 8.937556e-01
110589:   150        E 9.330312e-01
110590:   150        E 1.078799e+00
110591:   150        E 9.380807e-01
110592:   150        E 9.975085e-01
```

With this transformation and the compatibility of **sensobol** with the **data.table** package (Dowle and Srinivasan 2019), we can easily compute variance-based sensitivity indices at each selected time step for each state variable. We first activate 75% of the CPUs to bootstrap the Sobol' indices in parallel, and then compute the Sobol' indices grouping by `times` and `variable`:

```
R> ncpus <- floor(detectCores() * 0.75)
R>
R> indices <- indices.dt[, sobol_indices(Y = value, N = N, params = params,
+                                        order = order, boot = TRUE,
+                                        first = "jansen", R = R,
+                                        parallel = "multicore",
+                                        ncpus = ncpus)$results,
+                        .(variable, times)]
```

We can then compute the Sobol' indices of the dummy parameter:

```
R> indices.dummy <- indices.dt[, sobol_dummy(Y = value, N = N, params = params),
+   .(variable, times)]
```

Finally, with some lines of code we can get to visualize the evolution of S_i and T_i indices through time for each state variable and uncertain model input:

```
R> ggplot(indices, aes(times, original, fill = sensitivity,
+                       color = sensitivity,
+                       group = sensitivity)) +
+   geom_line() +
+   geom_ribbon(aes(ymin = indices[sensitivity %in% c("Si", "Ti")]$low.ci,
+                   ymax = indices[sensitivity %in% c("Si", "Ti")]$high.ci,
+                   color = sensitivity),
+               alpha = 0.1, linetype = 0) +
+   geom_hline(data = indices.dummy[, parameters:= NULL][sensitivity == "Ti"],
+              aes(yintercept = original, color = sensitivity, group = times),
+                       lty = 2, size = 0.1) +
+   guides(linetype = FALSE, color = FALSE) +
+   facet_grid(parameters ~ variable) +
+   scale_y_continuous(breaks = scales::pretty_breaks(n = 3)) +
+   labs(x = expression(italic(t)),
+        y = "Sobol' indices") +
+   theme_AP() +
+   theme(legend.position = "top")
```

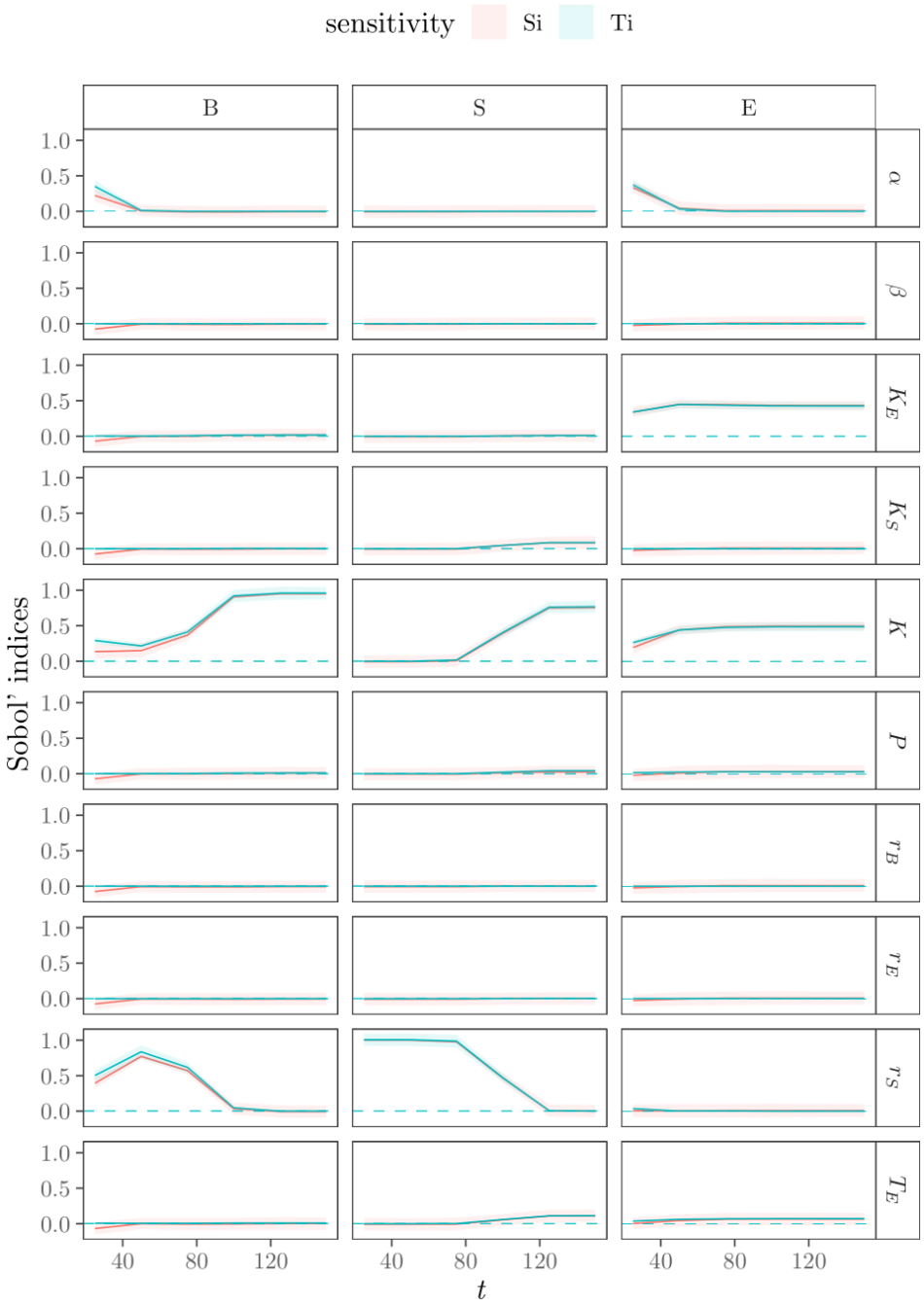

Figure 16: Evolution of Sobol' indices through time in the spruce budworm and forest model. The dashed, horizontal blue line shows the T_i of the dummy parameter.

The main results in Figure 16 can be summarized as follows:

1. The spruce budworm and forest model is largely additive up to $t \approx 80$ as interactions are very small and only affect the behavior of B ($S_\alpha < T_\alpha$, $S_K < T_K$, $S_{r_s} < T_{r_s}$). From $t > 80$, the model seems to be fully additive on all three state variables ($S_i \approx T_i$).

2. Given the uncertainty ranges defined in Table 6, only four parameters out of 10 are influential in conveying uncertainty to B, S and E: α, K, K_E and r_S:

   (a) The uncertainty in B is determined by α, K and r_S at $0 < t < 100$ and by K at $t > 100$.

   (b) The uncertainty in S is fully driven by r_S up to $t \approx 80$ and by K at $t > 120$.

   (c) The uncertainty in E is influenced by α, K_E and K at $0 < t < 40$ and by K_E and K at $t > 40$.

   The rest are non-influential and can be fixed at any value without modifying the model output. We should stress here that what is considered "influential" in a variance decomposition framework does not necesarily need to concur with the definition of "influential" from a systems dynamics perspective. The influence of a parameter in a variance decomposition framework is determined both by the functional form of the model and the probability distribution selected to describe the uncertainty of the parameters in the model input space.

4. Conclusions

Mathematical models are widely used to gain insights into complex processes, to predict the outcome of a variable of interest or the explore "what if" scenarios. In order to increase their transparency and ensure the quality of model-based inferences, it is paramount to scrutinize these model with a global sensitivity analysis. **sensobol** aims at furthering the uptake of global sensitivity analysis methods by the modeling community with a convenient, easy-to-use set of functions to compute variance-based analysis, the yardstick of global sensitivity methods. **sensobol** allows the user to combine several first and total-order estimators, to estimate up to third-order effects and to visualize the results in publication-ready plots. Due to its integration with **data.table** and **deSolve**, **sensobol** can easily compute variance-based indices for models with a scalar or multivariate model output, as well as for systems of differential equations.

sensobol will keep on developing as the search for more efficient variance-based estimators is an active field of research. We encourage the users to provide feedback and suggestions on how can the package be improved. The most recent updates can be followed on `https://github.com/arnaldpuy/sensobol`.

5. Acknowledgements

This work has been funded by the European Commission (Marie Skłodowska-Curie Global Fellowship, grant number 792178 to AP).

6. Annex

6.1. Benchmark of sensobol and sensitivity functions

We compare the execution time of **sensobol** and **sensitivity**, from the design of the sample matrix to the computation of the model output and the Sobol' indices. To ensure that the results are not critically conditioned by a particular benchmark design, we draw on Becker (2020) and Puy *et al.* (2020a) and compare the efficiency of **sensobol** and **sensitivity** on several randomly defined sensitivity settings. These settings are created by treating the base sample size N, the model dimensionality k and the functional test of the model as random parameters: N and k are described with the probability distributions shown in Table 7, and the test function is Becker (2020)'s metafunction (Table 4, Nº 6), which randomly combines p univariate functions in a multivariate function of dimension k. Becker's metafunction can be called in **sensobol** with `metafunction()`, and its current implementation includes cubic, discontinuous, exponential, inverse, linear, no-effect, non-monotonic, periodic, quadratic and trigonometric functions. We direct the reader to Becker (2020) and Puy *et al.* (2020a) for further information.

Table 7: Summary of the benchmark parameters N and k. $\mathcal{DU}$ stands for discrete uniform

Parameter	Description	Distribution
N	Base sample size of the sample matrix	$\mathcal{DU}(10, 100)$
k	Model dimensionality	$\mathcal{DU}(3, 100)$

We benchmark **sensobol** and **sensitivity** as follows:

- We create a $(2^{11}, 2)$ sample matrix using random numbers, where the first column is labeled N and the second column k.
- We describe N and k with the probability distributions in Table 7.
- For $v = 1, 2, \ldots, 2^{11}$ rows, we conducte two parallel sensitivity analysis using the functions and guidelines of **sensitivity** and **sensobol** respectively, with the base sample matrix as defined by N_v and k_v. The metafunction runs separately in both sensitivity analyses, and we bootstrap the sensitivity indices 100 times.
- We time the computation for both **sensobol** and **sensitivity** in each row.

The results suggest that `sensobol` may be a median of two times faster than `sensitivity`. We provide the code below:

Load required packages:

```
R> library("microbenchmark")
```

Define the settings of the analysis:

```
R> N <- 2 ^ 11
R> parameters <- c("N", "k")
R> R <- 10 ^ 2
```

Create the sample matrix:

```
R> dt <- sensobol::sobol_matrices(matrices = "A", N = N, params = parameters)
R> dt[, 1] <- floor(qunif(dt[, 1], 10, 10 ^ 2 + 1))
R> dt[, 2] <- floor(qunif(dt[, 2], 3, 100))
```

Run benchmark in parallel:

```
R> n.cores <- makeCluster(floor(detectCores() * 0.75))
R> registerDoParallel(n.cores)
R> ########################################
R> y <- foreach(i = 1:nrow(dt),
+               .packages = c("sensobol", "sensitivity")) %dopar%
+   {
+     params <- paste("x", 1:dt[i, "k"], sep = "")
+     N <- dt[i, "N"]
+     out <- microbenchmark::microbenchmark(
+       "sensobol" = {
+         params <- paste("X", 1:length(params), sep = "")
+         mat <- sensobol::sobol_matrices(N = N, params = params, type = "R")
+         y <- sensobol::metafunction(mat)
+         ind <- sensobol::sobol_indices(Y = y, N = N, params = params,
+                                        first = "jansen", total = "jansen",
+                                        boot = TRUE, R = R)$results},
+       "sensitivity" = {
+         X1 <- data.frame(matrix(runif(length(params) * N), nrow = N))
+         X2 <- data.frame(matrix(runif(length(params) * N), nrow = N))
+         x <- sensitivity::soboljansen(model = sensobol::metafunction,
+                                       X1, X2, nboot = R)
+       },
+       times = 1)
+   }
R> ########################################
R> stopCluster(n.cores)
```

Arrange the data and transform from nanoseconds to milliseconds:

```
R> out <- rbindlist(y)[, time := time / 1e+06]
```

Plot the results:

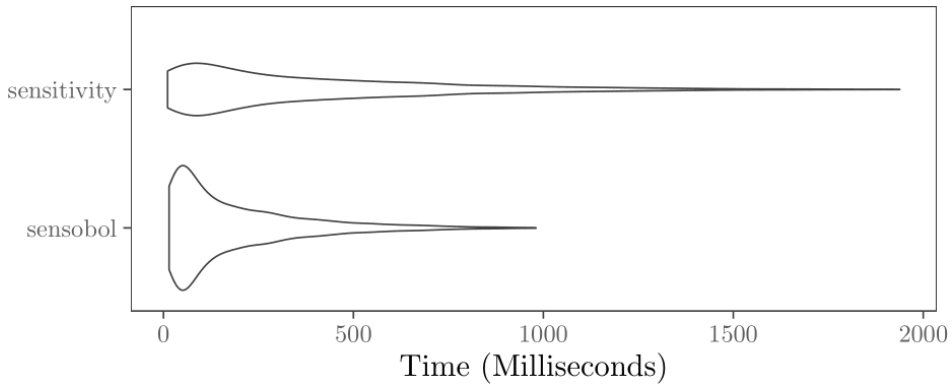

Figure 17: Benchmark of the sensitivity and sensobol packages. The comparison has been done with the Jansen estimators.

And compute the median:

```
R> out[, median(time), expr]
```

```
          expr       V1
1:    sensobol 119.9879
2: sensitivity 254.8925
```

6.2. Variogram Analysis of Response Surfaces (VARS-TO)

Given its reliance on variance and co-variance matrices, **sensobol** also offers support to compute the Variogram Analysis of Response Surfaces Total-Order index (VARS-TO) (Razavi and Gupta 2016b,a). VARS uses variograms and co-variograms to characterize the spatial structure and variability of a given model output across the input space, and allows to differentiate sensitivities as a function of scale h: if $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ are two points separated by a distance $\boldsymbol{h}$, and $y(\boldsymbol{x}_A)$ and $y(\boldsymbol{x}_B)$ is the corresponding model output y, the variogram $\gamma(.)$ is calculated as

$$\gamma(\boldsymbol{x}_A - \boldsymbol{x}_B) = \frac{1}{2} V\left[y(\boldsymbol{x}_A) - y(\boldsymbol{x}_B)\right] , \tag{13}$$

and the covariogram $C(.)$ as

$$C(\boldsymbol{x}_A - \boldsymbol{x}_B) = COV\left[y(\boldsymbol{x}_A), y(\boldsymbol{x}_B)\right] . \tag{14}$$

Since

$$V\left[y(\boldsymbol{x}_A) - y(\boldsymbol{x}_B)\right] = V\left[y(\boldsymbol{x}_A)\right] + V\left[y(\boldsymbol{x}_B)\right] - 2COV\left[y(\boldsymbol{x}_A), y(\boldsymbol{x}_B)\right] , \tag{15}$$

and $V\left[y(\boldsymbol{x}_A)\right] = V\left[y(\boldsymbol{x}_B)\right]$, then

$$\gamma(\boldsymbol{x}_A - \boldsymbol{x}_B) = V\left[y(\boldsymbol{x})\right] - C(\boldsymbol{x}_A, \boldsymbol{x}_B) . \tag{16}$$

If we want to compute the variogram for factor x_i, then

$$\gamma(h_i) = \frac{1}{2} E(y(x_1, ..., x_{i+1} + h_i, ..., x_n) - y(x_1, ..., x_i, ..., x_n))^2 . \tag{17}$$

Note that the difference in parentheses in Equation 17 involves taking a step along the x_i direction, and is analogous to computing the total-order index T_i (see Section 2.1). The equivalent of Equation 16 for the model input x_i would therefore be

$$\gamma_{\boldsymbol{x}^*_{\sim i}}(h_i) = V(y|\boldsymbol{x}^*_{\sim i}) - C_{\boldsymbol{x}^*_{\sim i}}(h_i) , \tag{18}$$

where $\boldsymbol{x}^*_{\sim i}$ is a fixed point in the space of non-x_i. To compute T_i in the framework of VARS (labelled as VARS-TO by Razavi and Gupta (2016b)), the mean value across the factors' space should be taken on both sides of Equation 18, e.g.,

$$E_{\boldsymbol{x}^*_{\sim i}}\left[\gamma^*_{\boldsymbol{x}_{\sim i}}(h_i)\right] = E_{\boldsymbol{x}^*_{\sim i}}\left[V(y|\boldsymbol{x}^*_{\sim i})\right] - E_{\boldsymbol{x}^*_{\sim i}}\left[C^*_{\boldsymbol{x}_{\sim i}}(h_i)\right] , \tag{19}$$

which can also be written as

$$E_{\boldsymbol{x}^*_{\sim i}}\left[\gamma^*_{\boldsymbol{x}_{\sim i}}(h_i)\right] = V(y)T_i - E_{\boldsymbol{x}^*_{\sim i}}\left[C^*_{\boldsymbol{x}_{\sim i}}(h_i)\right] , \tag{20}$$

and therefore

$$T_i = \text{VARS-TO} = \frac{E^*_{\boldsymbol{x}_{\sim i}}\left[\gamma^*_{\boldsymbol{x}_{\sim i}}(h_i)\right] + E^*_{\boldsymbol{x}_{\sim i}}\left[C^*_{\boldsymbol{x}_{\sim i}}(h_i)\right]}{V(y)} . \tag{21}$$

The computation of VARS does not require $\boldsymbol{A}$,$\boldsymbol{B}$, $\boldsymbol{A}_B^{(i)}$ ... matrices, but a sampling design based on stars. Such stars are created as follows: firstly, N_{star} points across the factor space need to be selected by the analyst using random or quasi-random numbers. These are the *star centres* and their location can be denoted as $\boldsymbol{s}_v = s_{v_1}, ..., s_{v_i}, ..., s_{v_k}$, where $v = 1, 2, ..., N_{star}$. Then, for each star centre, a cross section of equally spaced points Δh apart needs to be generated for each of the k factors, including and passing through the star centre. The cross section is produced by fixing $\boldsymbol{s}_{v_{\sim i}}$ 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 should be extracted. The total computational cost of this design is $N_t = N_{star}\left[k(\frac{1}{\Delta h} - 1) + 1\right]$.

In order to use VARS in **sensobol**, the analyst should follow the same steps as in the previous examples. Firstly, she should define the setting of the analysis, i.e., the number of star centers and distance h, and create a vector with the name of the parameters:

```
R> star.centers <- 100
R> h <- 0.1
R> params <- paste("X", 1:8, sep = "")
```

Then, the function `vars_matrices()` creates the sample matrix needed to compute VARS-TO:

```
R> mat <- vars_matrices(star.centers = star.centers, h = h, params = params)
```

We can then run the model rowwise, in this case the Sobol' (1998) G function (Table 4):

```
R> y <- sobol_Fun(mat)
```

And compute VARS-TO with the `vars_to()` function:

```
R> ind <- vars_to(Y = y, star.centers = star.centers, params = params, h = h)
R> ind


Number of star centers: 100 | h: 0.1

Total number of model runs: 7300
             Ti parameters
1: 0.8213028904         X1
```

```
2: 0.2526291054         X2
3: 0.0346579957         X3
4: 0.0104013502         X4
5: 0.0001079858         X5
6: 0.0001034112         X6
7: 0.0001076416         X7
8: 0.0001055614         X8
```

The current implementation of `vars_to()` does not allow to bootstrap the indices. This is planned for future **sensobol** releases.

References

Azzini I, Listorti G, Mara TA, Rosati R (2020a). *Uncertainty and Sensitivity Analysis for Policy Decision Making. An Introductory Guide*. Joint Research Centre, European Commission, Luxembourg. ISBN 9789276247524. doi:10.2760/922129.

Azzini I, Mara T, Rosati R (2020b). "Monte Carlo Estimators of First-and Total-Orders Sobol' Indices." 2006.08232, URL http://arxiv.org/abs/2006.08232.

Becker W (2020). "Metafunctions for Benchmarking in Sensitivity Analysis." *Reliability Engineering and System Safety*, **204**, 107189. ISSN 09518320. doi:10.1016/j.ress.2020.107189.

Becker W, Saltelli A (2015). "Design for sensitivity analysis." In A Dean, M Morris, J Stufken, D Bingham (eds.), *Handbook of Design and Analysis of Experiments*, pp. 627–674. CRC Press, Taylor & Francis, Boca Ratón. ISBN 9780429096341. doi:10.1201/b18619.

Bidot C, Lamboni M, Monod H (2018). ***multisensi**: Multivariate Sensitivity Analysis*. R package version 2.1.1. URL https://CRAN.R-project.org/package=multisensi.

Borgonovo E (2007). "A New Uncertainty Importance Measure." *Reliability Engineering and System Safety*, **92**(6), 771–784. ISSN 09518320. doi:10.1016/j.ress.2006.04.015.

Bratley P, Fox BL (1988). "ALGORITHM 659: Implementing Sobol's Quasirandom Sequence generator." *ACM Transactions on Mathematical Software (TOMS)*, **14**(1), 88–100.

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.

Chalom A, Inacio Knegt Lopez P (2017). ***pse**: Parameter Space Exploration with Latin Hypercubes*. R package version 0.4.7. URL https://CRAN.R-project.org/package=pse.

Dowle M, Srinivasan A (2019). ***data.table**: Extension of "data.frame"*. R package version 1.13.2. URL https://cran.r-project.org/package=data.table.

Ferretti F, Saltelli A, Tarantola S (2016). "Trends in Sensitivity Analysis Practice in the Last Decade." *Science of the Total Environment*, **568**, 666–670. ISSN 18791026. doi:10.1016/j.scitotenv.2016.02.133.

Gilbertson A (2018). "An Approach for Using Probabilistic Risk Assessment in Risk-Informed Decisions on Plant-Specific Changes to the Licensing Basis. Regulatory Guide 1.174, revision 3." *Technical report*, US Nuclear Regulatory Commission.

Glen G, Isaacs K (2012). "Estimating Sobol' Sensitivity Indices Using Correlations." *Environmental Modelling and Software*, **37**, 157–166. ISSN 13648152. doi:10.1016/j.envsoft.2012.03.014.

Herman J, Usher W (2017). "**SALib**: An Open-Source Python Library for Sensitivity Analysis." *The Journal of Open Source Software*, **2**(9), 97. ISSN 2475-9066. doi:10.21105/joss.00097.

Homma T, Saltelli A (1996a). "Importance Measures in Global Sensitivity Analysis of Nonlinear Models." *Reliability Engineering & System Safety*, **52**(1), 1–17. ISSN 09518320. doi:10.1016/0951-8320(96)00002-6.

Homma T, Saltelli A (1996b). "Importance Measures in Global Sensitivity Analysis of Nonlinear Models." *Reliability Engineering & System Safety*, **52**, 1–17. ISSN 09518320. doi:10.1016/0951-8320(96)00002-6.

Iooss B, Janon A, Pujol G, with contributions from Baptiste Broto, Boumhaout K, Veiga SD, Delage T, Amri RE, Fruth J, Gilquin L, Guillaume J, Le Gratiet L, Lemaitre P, Marrel A, Meynaoui A, Nelson BL, Monari F, Oomen R, Rakovec O, Ramos B, Roustant O, Song E, Staum J, Sueur R, Touati T, Weber F (2020). ***sensitivity***: *Global Sensitivity Analysis of Model Outputs*. R package version 1.22.1. URL https://cran.r-project.org/package=sensitivity.

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.

Jakeman A, Letcher R, Norton J (2006). "Ten Iterative Steps in Development and Evaluation of Environmental Models." *Environmental Modelling & Software*, **21**(5), 602–614. ISSN 13648152. doi:10.1016/j.envsoft.2006.01.004.

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. ISSN 1292-8100. doi:10.1051/ps/2013040.

Jansen M (1999). "Analysis of Variance Designs for Model Output." *Computer Physics Communications*, **117**(1-2), 35–43. ISSN 00104655. doi:10.1016/S0010-4655(98)00154-4.

Kay JA (2012). "Knowing When We Don't Know." *Technical report*, Institute for Fiscal Studies. URL https://www.ifs.org.uk/publications/6016.

Kucherenko S, Delpuech B, Iooss B, Tarantola S (2015). "Application of the Control Variate Technique to Estimation of Total Sensitivity Indices." *Reliability Engineering and System Safety*, **134**(July), 251–259. ISSN 09518320. doi:10.1016/j.ress.2014.07.008.

Kucherenko S, Feil B, Shah N, Mauntz W (2011). "The Identification of Model Effective Dimensions Using Global Sensitivity Analysis." *Reliability Engineering & System Safety*, **96**(4), 440–449. ISSN 09518320. doi:10.1016/j.ress.2010.11.003.

Leamer EE (2010). "Tantalus on the Road to Asymptopia." *Journal of Economic Perspectives*, **24**(2), 31–46. ISSN 08953309. doi:10.1257/jep.24.2.31.

Lo Piano S, Ferretti F, Puy A, Albrecht D, Saltelli A (2021). "Variance-Based Sensitivity Analysis: The Quest for Better Estimators and Designs Between Explorativity and Economy." *Reliability Engineering & System Safety*, **206**(October 2020), 107300. doi:10.1016/j.ress.2020.107300.

Ludwig D, Jones DD, Holling CS (1976). "Qualitative Analysis of Insect Outbreak Systems: the Spruce Budworm and Forest." *The Journal of Animal Ecology*, **47**(1), 315. ISSN 00218790. doi:10.2307/3939.

Marelli S, Sudret B (2014). "**UQLab**: A Framework for Uncertainty Quantification in MATLAB." In *The 2nd International Conference on Vulnerability and Risk Analysis and Management (ICVRAM 2014)*, Bourinet 2009, pp. 2554–2563. University of Liverpool, United Kingdomm July 13-16, Liverpool.

May RM, Oster GF (1976). "Bifurcations and Dynamic Complexity in Simple Ecological Models." *The American Naturalist*, **110**(974), 573–599.

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. ISSN 15372723. doi:10.1080/00401706.1979.10489755.

Microsoft, Weston S (2020a). ***doParallel**: Foreach Parallel Adaptor for the **parallel** Package*. R package version 1.0.16. URL https://CRAN.R-project.org/package=doParallel.

Microsoft, Weston S (2020b). ***foreach**: Provides Foreach Looping Construct*. R package version 1.5.1. URL https://CRAN.R-project.org/package=foreach.

Monod H, Naud C, Makowski D (2006). *Uncertainty and Sensitivity Analysis for Crop Models*. Elsevier. ISBN 0-444-52135-6. URL http://reseau-mexico.fr/sites/reseau-mexico.fr/files/06{_}zebook{_}CH-03.pdf.

Oakley J, O'Hagan A (2004). "Probabilistic Sensitivity Analysis of Complex Models: a Bayesian Approach." *Journal of the Royal Statistical Society B*, **66**(3), 751–769. ISSN 13697412. doi:10.1111/j.1467-9868.2004.05304.x.

Pearl R, Reed LJ (1920). "On the Rate of Growth of the Population of the United States Since 1790 and Its Mathematical Representation." *Proceedings of the National Academy of Sciences*, **6**(6), 275–288. ISSN 0027-8424. doi:10.1073/pnas.6.6.275.

Pianosi F, Beven K, Freer J, Hall JW, Rougier J, Stephenson DB, Wagener T (2016). "Sensitivity Analysis of Environmental Models: A Systematic Review with Practical Workflow." *Environmental Modelling and Software*, **79**, 214–232. ISSN 13648152. doi:10.1016/j.envsoft.2016.02.008. j.envsoft.2016.02.008.

Pianosi F, Sarrazin F, Wagener T (2015). "A MATLAB Toolbox for Global Sensitivity Analysis." *Environmental Modelling and Software*, **70**, 80–85. ISSN 13648152. doi:10.1016/j.envsoft.2015.04.009.

Pianosi F, Wagener T (2015). "A Simple and Efficient Method for Global Sensitivity Analysis Based on Cumulative Distribution Functions." *Environmental Modelling and Software*, **67**, 1–11. ISSN 13648152. doi:10.1016/j.envsoft.2015.01.004.

Puy A, Becker W, Piano SL, Saltelli A (2020a). "The Battle of Total-Order Sensitivity Estimators." *International Journal for Uncertainty Quantification*. 2009.01147, URL http://arxiv.org/abs/2009.01147.

Puy A, Lo Piano S, Saltelli A (2020b). "A Sensitivity Analysis of the PAWN Sensitivity Index." *Environmental Modelling and Software*, **127**, 104679. ISSN 13648152. doi:10.1016/j.envsoft.2020.104679. 1904.04488.

Puy A, Lo Piano S, Saltelli A (2020c). "Current Models Underestimate Future Irrigated Areas." *Geophysical Research Letters*, **47**(8). ISSN 0094-8276. doi:10.1029/2020GL087360.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. URL https://www.R-project.org/.

Razavi S, Gupta HV (2016a). "A New Framework for Comprehensive, Robust, and Efficient Global Sensitivity Analysis: 1. Theory." *Water Resources Research*, **52**(1), 423–439. ISSN 0043-1397. doi:10.1002/2015WR017559.

Razavi S, Gupta HV (2016b). "A New Framework for Comprehensive, Robust, and Efficient Global Sensitivity Analysis: 2. Application." *Water Resources Research*, **52**(1), 440–455. ISSN 0043-1397. doi:10.1002/2015WR017558. 2014WR016527.

Reusser D (2015). "**Fast**: Implementation of the Fourier Amplitude Sensitivity Test (fast)." URL https://github.com/cran/fast.

Saltelli A, Aleksankina K, Becker W, Fennell P, Ferretti F, Holst N, Li S, Wu Q (2019). "Why So Many Published Sensitivity Analyses Are False: A Systematic Review of Sensitivity Analysis Practices." *Environmental Modelling and Software*, **114**(January), 29–39. ISSN 13648152. doi:10.1016/j.envsoft.2019.01.012. 1711.11359.

Saltelli A, Andres TH, Homma T (1993). "Sensitivity Analysis of Model Output. An Investigation of New Techniques." *Computational Statistics and Data Analysis*, **15**(2), 211–238. ISSN 01679473. doi:10.1016/0167-9473(93)90193-W.

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. ISSN 00104655. doi:10.1016/j.cpc.2009.09.018.

Saltelli A, Bammer G, Bruno I, Charters E, Di Fiore M, Didier E, Nelson Espeland W, Kay J, Lo Piano S, Mayo D, Pielke Jr R, Portaluri T, Porter TM, Puy A, Rafols I, Ravetz JR, Reinert E, Sarewitz D, Stark PB, Stirling A, van der Sluijs J, Vineis P (2020). "Five Ways to Ensure that Models Serve Society: A Manifesto." *Nature*, **582**(7813), 482–484. ISSN 0028-0836. doi:10.1038/d41586-020-01812-9.

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

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. ISSN 00415553. doi:10.1016/0041-5553(67)90144-9.

Sobol' IM (1976). "Uniformly Distributed Sequences With an Additional Uniform Property." *USSR Computational Mathematics and Mathematical Physics*, **16**(5), 236–242. ISSN 00415553. doi:10.1016/0041-5553(76)90154-3.

Sobol' IM (1993). "Sensitivity Analysis For Nonlinear Mathematical Models." *Mathematical Modeling and Computational Experiment*, **1**(3), 407–414. ISSN 0234-0879.

Sobol' IM (1998). "On Quasi-Monte Carlo Integrations." *Mathematics and Computers in Simulation*, **47**(2-5), 103–112. ISSN 03784754. doi:10.1016/S0378-4754(98)00096-2.

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. ISSN 03784754. doi:10.1016/S0378-4754(00)00270-6.

Soetaert K, Petzoldt T, Setzer RW (2010). "Solving Differential Equations in R: Package **deSolve**." *Journal of Statistical Software*, **33**(9), 1–25. ISSN 15487660. doi:10.18637/jss.v033.i09.

Steinmann P, Wang JR, van Voorn GAK, Kwakkel JH (2020). "Don't Try to Predict COVID-19. If You Must, Use Deep Uncertainty Methods." *Review of Artificial Societies and Social Simulation*, (April 17). URL https://rofasss.org/2020/04/17/deep-uncertainty/.

Verhulst PF (1845). "Recherches Mathématiques Sur la Loi d'Accroissement de la Population." *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles*, **18**(8).

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag, New York. URL https://ggplot2.tidyverse.org.

Affiliation:

Arnald Puy
Ecology and Evolutionary Biology,
Princeton University
M31 Guyot Hall
New Jersey 08544, USA
E-mail: apuy@princeton.edu
URL: http://www.arnaldpuy.com

Samuele Lo Piano
School of the Built Environment
University of Reading
JJ Thompson Building, Whiteknights Campus
Reading RG6 6AF, United Kingdom

Andrea Saltelli
Open Evidence Research
Universitat Oberta de Catalunya
Carrer Almogàvers 165
Barcelona 08018, Spain

Simon A. Levin
Ecology and Evolutionary Biology,
Princeton University
203 Eno Hall
New Jersey 08544, USA