Hypermodern Screening

Tobias Stenzel

May 31, 2020

CONTENTS

The `hypermodern-screening` package provides tools for efficient global sensitivity analyses based on elementary effects. Its unique feature is the option to compute these effects for models with correlated input parameters. The underlying conceptual approach is developed by Stenzel (2020). The fundamental idea comes from Ge and Menendez (2017). It is the combination of inverse transform sampling with an intelligent juggling of parameter positions in the input vector to create different dependency hierarchies. The package does also include a variety of sampling methods.

Install `hypermodern-screening` from PyPI with

```
$ pip install hypermodern_screening
```

CHAPTER

ONE

DOCUMENTATION STRUCTURE

The documentation consists of two main parts. The first part is close to the implementation and the second part provides some background starting from basic definitions of uncertainty qunatification.

The first part describes the concepts that are implemented in this package. It comprises three sections. These are *sampling schemes* tailored to the computation of elementary effects (EEs), *EEs* for correlated parameters and the importance of *sigma normalization*. The first two sections are accompanied by tutorials written in jupyter notebooks that demonstrate the usage of the package:

- Sampling Schemes:

- Elementary Effects:

The second part embedds EEs within uncertainty quantification and, in particular, sensitivity analysis. Thereby, the motivation to compute EEs is clarified. It also contains a short discussion of methods for models with correlated input parameters.

1.1 Sampling Schemes

Click at the following *nbviewer* or *mybinder* badges to view the tutorial notebook that accompanies this section.

The two presented sampling schemes are the trajectory and the radial design. Although the schemes are tailored to the computation of EEs, positional differences between them cause differences in their post-processing.

According to several experiments with common test functions by [Campolongo.2011], the best design is the radial design ([Saltelli.2002]) and the most commonly used is the trajectory design ([Morris.1991]). Both designs are comprised by a $(k+1) \times k$-dimensional matrix. The elements are generated in $[0, 1]$. Afterwards, they can potentially be transformed to the distributions of choice. The columns represent the different input parameters and each row is a complete input parameter vector. To compute the aggregate qualitative measures, a set of multiple matrices, or sample subsets, of input parameters has to be generated.

A matrix in radial design is generated the following way: draw a vector of length $2k$ from a quasi-random sequence. The first row, or parameter vector, is the first half of the sequence. Then, copy the first row to the remaining k rows. For each row k' of the remaining $2, \ldots, k+1$ rows, replace the k'-th element by the k'-th element of the second half of the vector. This generates a matrix of the following form:

$$\underset{(k+1)\times k}{\boldsymbol{R}} = \begin{pmatrix} a_1 & a_2 & ... & a_k \\ \boldsymbol{b_1} & a_2 & ... & a_k \\ a_1 & \boldsymbol{b_2} & ... & a_k \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & ... & \boldsymbol{b_k} \end{pmatrix}.$$

Note here, that each column consists only of the respective first row element, except in one row. From this matrix, one EE can be obtained for each parameter X_i. This is achieved by using the $(i+1)$-th row as function argument for the minuend and the first row as the subtrahend in the EE formula in Equation (ref{eq:EE}). Then, $\Delta^{(i,j)} = b_i^{(j)} - a_i^{(j)}$. The asterisk is an index for all elements of a vector.

$$d_i = \frac{Y(\boldsymbol{a_{\sim i}}, b_i) - Y(\boldsymbol{a})}{b_i - a_i} = \frac{Y(\boldsymbol{R_{i+1,*}}) - Y(\boldsymbol{R_{1,*}})}{b_i - a_i}.$$

If the number of radial subsamples is high, the draws from the quasi-random sequence lead to a fast coverage of the input space (compared to a random sequence). However, a considerable share of steps will be large, the maximum is $1 - \epsilon$ in a sample space of $[0, 1]$. This amplifies the aforementioned problem of EE-based measures with non-linear functions. The quasi-random sequence considered here is the Sobol' sequence. It is comparably successful in the dense coverage of the unit hypercube, but also conceptually more involved. Therefore, its presentation is beyond the scope of this work. As the first elements of each Sobol' sequence, the direction numbers, are equal I draw the sequence at once for all sets of radial matrices.

Next, I present the trajectory design. As we will see, it can lead to a relatively representative coverage for a very small number of subsamples but also to repetitions of similar draws. I skip the equations that generate a trajectory and instead present the method verbally. There are different forms of trajectories. I focus on the common version presented in [Morris.1991] that generates equiprobable elements. The first step is to decide the number p of equidistant grid points in interval $[0, 1]$. Then, the first row of the trajectory is composed of the lower half value of these grid points. Now, fix $\Delta = p/[2(p-1)]$. This function implies, that adding Δ to the lowest point in the lowest half results in the lowest point of the upper half of the grid points, and so on. It also implies that 0.5 is the lower bound of Δ. The rest of the rows is constructed, first, by copying the row one above and, second, by adding Δ to the i-th element of the $i+1$-th row. The created matrix scheme is depicted below.

$$\underset{(k+1)\times k}{\boldsymbol{T}} = \begin{pmatrix} a_1 & a_2 & ... & a_k \\ \boldsymbol{b_1} & a_2 & ... & a_k \\ \boldsymbol{b_1} & \boldsymbol{b_2} & ... & a_k \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{b_1} & \boldsymbol{b_2} & ... & \boldsymbol{b_k} \end{pmatrix}$$

In contrary to the radial scheme, each b_i is copied to the subsequent row. Therefore, the EEs have to be determined by comparing each row with the row above instead of with the first row. Importantly, two random transformations are common. These are, first, randomly switching rows, and second, randomly interchanging the i-th column with the $(k-i)$-th column and then reversing the column. The first transformation is skipped as it does not add additional coverage and because we need the stairs-shaped design to facilitate later transformations which account for correlations between input parameters. The second transformation is adapted because it is important to also have negative steps and because it sustains the stairs shape. Yet, this implies that Δ is also parameter- and trajectory-specific. Let f and h be additional indices representing the input parameters. The derivative formula is adapted to the trajectory design as follows:

$$d_i = \frac{Y(\boldsymbol{b_{f \leq i}}, \boldsymbol{a_{h > i}}) - Y(\boldsymbol{b_{f < i}}, \boldsymbol{a_{h \geq i}})}{b_i - a_i} = \frac{Y(\boldsymbol{T_{i+1,*}}) - Y(\boldsymbol{T_{i,*}})}{b_i - a_i}.$$

The trajectory design involves first, a fixed grid, and second and more importantly, a fixed step Δ, s.t. $\{\Delta\} = \{\pm\Delta\}$. This implies less step variety and less space coverage vis-á-vis the radial design for a larger number of draws.

To improve the sample space coverage by the trajectory design, [Campolongo.2007] develop a post-selection approach based on distances. The approach creates enormous costs for more than a small number of trajectories. This problem is effectively mitigated by [Ge.2014]. The following describes the main ideas of both contributions.

The objective of both works is to select k matrices. [Campolongo.2007] assign a pair distance to each pair of trajectories in the start set. Thereafter, they identify each possible combination of k from N trajectories. Then, they compute an aggregate distance for each combination based on the single pair distances. Finally, the optimized trajectory set is the subset with the highest aggregate distance.

This is computationally very costly because each aggregate distance is a sum of a binomial number of pair distances. For example, $\binom{30}{15} = 155117520$. To decrease the computation time, [Ge.2014] propose two improvements. First, in each iteration i, they select only $N(i) - 1$ matrices from a set containing $N(i)$ trajectories until the set size has decreased to k. Second, they compute the pair distances in each iteration based on the aggregate distances and the pair distances from the first set. Due to numerical imprecisions, their improvement does not always result in the same set as obtained from [Campolongo.2007]. However, the sets are usually very similar in terms of the aggregate distance. This thesis only uses the first step in [Ge.2014] to post-select the trajectory set because the second step does not provide any gain. [This refers only to my implementation.]

So far, we have only considered draws in [0,1]. For uncorrelated input parameters from arbitrary distributions with well-defined cumulative distribution function (CDF), Φ^{-1}, one would simply evaluate each element (potentially including the addition of the step) by the inverse CDF, or quantile function, Φ, of the respective parameter. One intuition is, that Φ^{-1} maps the sample space to [0,1]. Hence Φ can be used to transform random draws in [0,1] to the sample space of the arbitrary distribution. This is a basic example of so-called inverse transform sampling ([Devroye.1986]) which we recall in the next section.

The following section describes the computation of Elementary Effects for correlated input parameters from samples in trajectory and radial design.

1.2 Elementary Effects

Click at the following *nbviewer* or *mybinder* badges to view the tutorial notebook that accompanies this section.

This section describes the approach to extend the EE-based measures to input parameters that are correlated. It largely follows [Ge.2017]. Their main achievement is to outline a transformation of samples in radial and trajectory design that incorporates the correlation between the input parameters. This implies, that the trajectory and radial samples cannot be written as before. The reason is that the correlations of parameter X_i, to which step Δ^i is added, imply that all other parameters $\boldsymbol{X_{\sim i}}$ in the same row with non-zero correlation with X_i are changed as well. Therefore, the rows cannot be denoted and compared as easily by a's and b's as in the last section. Transforming these matrices allows to re-define the EE-based measures accordingly, such that they sustain the main properties of the ordinary measures for uncorrelated parameters. The property is being a function of the mean derivative. The section covers the approach in a simplified form, focussing on normally distributed input parameters. Yet, [Ge.2017] do not fully develop these measures. I explain how their measures can lead to arbitrary rankings for correlated input parameters. I first cover the inverse transform sampling method that incorporates correlations between random draws from the parameter distributions. Second, I describe the Elementary Effects that I redesigned based on my analysis of [Ge.2017] and the drawbacks therein. Lastly, I comment on these drawbacks in more detail.

1.2.1 Inverse transform sampling

The section deals with developing a recipe for transforming draws $\boldsymbol{u} = \{u_1, u_2, ..., u_k\}$ from $[0, 1]$ for an input parameter vector to draws $\boldsymbol{x} = \{x_1, x_2, ..., x_k\}$ from an arbitrary joint normal distribution. We will do this in three steps.

For this purpose, let $\boldsymbol{\Sigma}$ be a non-singular variance-covariance matrix and let $\boldsymbol{\mu}$ be the mean vector. The k-variate normal distribution is denoted by $\mathcal{N}_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Creating potentially correlated draws $\boldsymbol{x}$ from $\mathcal{N}_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is simple. Following [Gentle.2006], this can be achieved the following way: draw a k-dimensional row vector of independent and identically distributed (i.i.d.) standard normal deviates from the univariate $N(0, 1)$ distribution, such that $\boldsymbol{z} = \{z_1, z_2, ..., z_k\}$, and compute the Cholesky decomposition of Σ, such that $\boldsymbol{\Sigma} = \boldsymbol{T^T T}$. The lower triangular matrix is denoted by $\boldsymbol{T^T}$. Then apply the operation in the equation below to obtain the correlated deviates $\boldsymbol{x}$ from $\mathcal{N}_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Then,

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{T^T z^T}.$$

The next step is to understand that we can split the operation in thje above equation into two subsequent operations. The separated first part allows to potentially map correlated standard normal deviates to other distributions than the normal one. For this, let $\boldsymbol{\sigma}$ be the vector of standard deviations and let $\boldsymbol{R_k}$ be the correlation matrix of $\boldsymbol{x}$.

The first operation is to transform the standard deviates $\boldsymbol{z}$ to correlated standard deviates $\boldsymbol{z_c}$ by using $\boldsymbol{z_c} = \boldsymbol{Q^T z^T}$. In this equation, $\boldsymbol{Q^T}$ is the lower matrix from the Cholesky decomposition $\boldsymbol{R_k} = \boldsymbol{Q^T Q}$. This is equivalent to the above approach in [Gentle.2006] for the specific case of the multivariate standard normal distribution $\mathcal{N}_k(0, R_k)$. This is true because for multivariate standard normal deviates, the correlation matrix is equal to the covariance matrix.

The second operation is to scale the correlated standard normal deviates: $\boldsymbol{z = z_c(i)\sigma(i) + \mu}$., where the i s indicate an element-wise multiplication. This equation is specific to normally distributed parameters.

The last step to construct the final approach is to recall the inverse transform sampling method. Therewith, we can transform the input parameter draws $\boldsymbol{u}$ to uncorrelated standard normal draws $\boldsymbol{z}$. Then we will continue with the two operations in the above paragraph. The transformation from $\boldsymbol{u}$ to $\boldsymbol{z}$ is denoted by $F^{-1}(\Phi^c)$, where the c in Φ^c stands for the introduced correlations. This transformation is summarized by the following three steps:

$$\begin{aligned}
\text{Step 1: } & \boldsymbol{z = \Phi(u)} \\
\text{Step 2: } & \boldsymbol{z_c = Q^T z^T} \\
\text{Step 3: } & \boldsymbol{x = \mu + z_c(i)\sigma(i)}
\end{aligned}$$

I denote these three steps by $F^{-1}(\Phi^c) = \mathcal{T}_2$.

To map u to different sample spaces, Step 3 can be substituted. For instance, this could be achieved by applying $\Phi^{-1,u}$ and the inverse CDF of the desired distribution to z_c. [The procedure described by the three steps above is equivalent to an inverse Rosenblatt transformation and a linear inverse Nataf transformation for parameters in normal sample space and connects to Gaussian copulas. For the first two transformations, see [Lemaire.2013], p. 78 - 113. These concepts can be used to transform deviates in [0,1] to the sample space of arbitrary distributions by using the properties sketched above under different conditions.]

The one most important point to understand is that the transformation comprised by the three steps is not unique for correlated input parameters. Rather, the transformation changes with the order of parameters in vector $\boldsymbol{u}$. This can be seen from the lower triangular matrix $\boldsymbol{Q^T}$. To prepare the next equation, let $\boldsymbol{R_k} = (\rho_{ij})_{ij=1}^{k}$ and sub-matrix $\boldsymbol{R_h} = (\rho_{ij})_{ij=1}^{h}$, $h \leq k$. Also let $\boldsymbol{\rho_i^{*,j}} = (\rho_{1,j}, \rho_{2,j}, ..., \rho_{i-1,j})$ for $j \geq i$ with the following abbreviation $\boldsymbol{\rho_i} := \boldsymbol{\rho_i^{*,i}}$.

Following [Madar.2015], the lower matrix can be written as

$$\boldsymbol{Q^T} = \begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ \rho_{1,2} & \sqrt{1-\rho_{1,2}^2} & 0 & ... & 0 \\ \rho_{1,3} & \frac{\rho_{2,3}-\rho_{1,2}\rho_{1,3}}{\sqrt{1-\rho_{1,2}^2}} & \sqrt{1-\boldsymbol{\rho_3 R_2^{-1} \rho_3^T}} & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{1,k} & \frac{\rho_{2,k}-\rho_{1,2}\rho_{1,k}}{\sqrt{1-\rho_{1,2}^2}} & \frac{\boldsymbol{\rho_{3,k}-\rho_3^{*,k} R_2^{-1} \rho_3^T}}{\sqrt{1-\boldsymbol{\rho_3 R_2^{-1} \rho_3^T}}} & ... & \sqrt{1-\boldsymbol{\rho_k R_2^{-1} \rho_k^T}} \end{pmatrix}.$$

This equation, together with Step 2, implies that the order of the uncorrelated standard normal deviates $\boldsymbol{z}$ constitutes a hierarchy amongst the correlated deviates $\boldsymbol{z_c}$ in the following manner: the first parameter is not subject to any correlations, the second parameter is subject to the correlation with the first parameter, the third parameter is subject to the correlations with the parameters before, etc. Therefore, if parameters are correlated, typically $\boldsymbol{Q^T z^T} \neq \boldsymbol{Q^T (z')^T}$ and $F^{-1}(\Phi)(\boldsymbol{u}) \neq F^{-1}(\Phi)(\boldsymbol{u'})$, where $\boldsymbol{z'}$ and $\boldsymbol{u'}$ denote $\boldsymbol{z}$ and $\boldsymbol{u}$ in different orders. The next section shows how the three sampling steps play in the design of the Elementary Effects for correlated parameters.

1.2.2 Elementary Effects

I redesign the measures in [Ge.2017] by scaling the Δ in the denominator according to the nominator. I refer to the redesigned measures as the correlated and the uncorrelated Elementary Effects, d_i^c and d_i^u. The first measure includes the respective parameters effect on the other parameters and the second measure excludes it. The idea is, that both parameters have to be *very small* for one parameter to be fixed as constant or non-random. It is still a open reasearch question what *very small* should be. The measures are given below for arbitrary input distributions and for samples in trajectory and radial design.

$$\begin{aligned} d_i^{c,T} &= \frac{f\big(\mathcal{T}(\boldsymbol{T_{i+1,*}}; i-1)\big) - f\big(\mathcal{T}(\boldsymbol{T_{i-1,*}}; i)\big)}{\mathcal{T}(\lfloor\rangle) - \mathcal{T}(\dashv\rangle)} \\ d_i^{u,T} &= \frac{f\big(\mathcal{T}(\boldsymbol{T_{i+1,*}}; i)\big) - f\big(\mathcal{T}(\boldsymbol{T_{i,*}}; i)\big)}{\mathcal{T}(\lfloor\rangle) - \mathcal{T}(\dashv\rangle)} \\ d_i^{c,R} &= \frac{f\big(\mathcal{T}(\boldsymbol{R_{i+1,*}}; i-1)\big) - f\big(\mathcal{T}(\boldsymbol{R_{1,*}}; i-1)\big)}{\mathcal{T}(\lfloor\rangle) - \mathcal{T}(\dashv\rangle)} \\ d_i^{u,R} &= \frac{f\big(\mathcal{T}(\boldsymbol{R_{i+1,*}}; i)\big) - f\big(\mathcal{T}(\boldsymbol{R_{1,*}}; i)\big)}{\mathcal{T}(\lfloor\rangle) - \mathcal{T}(\dashv\rangle)}. \end{aligned}$$

In the above equations, $\mathcal{T}(\cdot; i) := \mathcal{T}_3\Big(\mathcal{T}_2\big(\mathcal{T}_1(\cdot; i)\big); i\Big)$. $\mathcal{T}_1(\cdot; i)$ orders the parameters, or row elements, to establish the right correlation hierarchy. $\mathcal{T}_2$, or $F^{-1}(\Phi^c)$, correlates the draws in $[0, 1]$ and transforms them to the sample space. $\mathcal{T}_3(\cdot; i)$ reverses the element order back to the start, to be able to apply the subtraction in the numerator of the EEs row-by-row. Index i in $\mathcal{T}_1(\cdot; i)$ and $\mathcal{T}_3(\cdot; i)$ stands for the number of initial row elements that are cut and moved to the back of the row in the same order. Applying $\mathcal{T}(\boldsymbol{T_{i+1,*}}; i-1)$ and $\mathcal{T}(\boldsymbol{T_{i+1,*}}; i)$ to all rows i of trajectory $\boldsymbol{T}$ gives the

following two transformed trajectories:

$$\mathcal{T}_1(\boldsymbol{T_{i+1,*}}; i-1) = \begin{pmatrix} a_k & a_1 & ... & ... & a_{k-1} \\ \boldsymbol{b_1} & a_2 & ... & ... & a_k \\ \boldsymbol{b_2} & a_3 & ... & ... & \boldsymbol{b_1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{b_k} & \boldsymbol{b_1} & ... & ... & \boldsymbol{b_{k-1}} \end{pmatrix}$$

$$\mathcal{T}_1(\boldsymbol{T_{i,*}}; i-1) = \begin{pmatrix} a_1 & a_2 & ... & ... & a_k \\ a_2 & ... & ... & a_k & \boldsymbol{b_1} \\ a_3 & ... & ... & \boldsymbol{b_1} & \boldsymbol{b_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{b_1} & \boldsymbol{b_2} & ... & ... & \boldsymbol{b_k} \end{pmatrix}$$

Two points can be seen from Equation above equations. First, $\mathcal{T}_1(\boldsymbol{T_{i+1,*}}; i-1)$ and $\mathcal{T}_1(\boldsymbol{T_{i,*}}; i)$, i.e. the $(i+1)$-th row in the first equation and the (i)-th row in the second equation, only differ in the i-th element. The difference is $b_i - a_i$. Thus, these two rows can be used to compute the EEs like in the uncorrelated case in the qualitative GSA section. However, in this order, the parameters are in the wrong positions to be directly handed over to the function, as the i-th parameter is always in front. The second point is that in $\mathcal{T}_1(\boldsymbol{T_{i+1,*}}; i-1)$, b_i is in front of the i-th row. This order prepares the establishing of the right correlation hierarchy by $\mathcal{T}_2$, such that the Δ in $a_i + \Delta$ is included to transform all other elements representing $X_{\sim i}$. Importantly, to perform $\mathcal{T}_2$, mean vector $\boldsymbol{x}$ and covariance matrix $\boldsymbol{\Sigma}$ and its transformed representatives have always to be re-ordered according to each row. Then, $\mathcal{T}_3$ restores the original row order and d_i^{full} can comfortably be computed by comparing function evaluations of row $i+1$ in $\mathcal{T}(\boldsymbol{T_{i+1,*}}; i-1)$ with function evaluations of row i in $\mathcal{T}(\boldsymbol{T_{i,*}}; i-1)$. Now, the two transformed trajectories only differ in the i-th element in each row i.

Assuming samples in trajectory design, one can also write the denominator equivalently but more explicitly for all four EEs as $F^{-1}\big({Q^T}_{k,*k-1}(j) R^T_{i+1,*k-1}(j) + {Q^T}_{k,k}\Phi^u(b_i)\big) - F^{-1}\big({Q^T}_{k,*k-1}(j) R^T_{i,*k-1}(j) + {Q^T}_{k,k}\Phi^u(a_i)\big)$. Not accounting for Q_t like [Ge.2017] leads to arbitrarily decreased independent Elementary Effects for input parameters with higher correlations.

The transformation for the samples in radial design are equivalent except that the subtrahend samples do only consist of reordered first rows.

For $X_1, ..., X_k \sim \mathcal{N}_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the denominator of $d_i^{u,*}$ simplifies drastically to

$$\begin{aligned} &\big(\mu_i + \sigma_i\big({Q^T}_{k,*k-1}(j) T^T_{i+1,*k-1}(j) + {Q^T}_{k,k}\Phi^u(b_i)\big) \\ -&\big(\mu_i + \sigma_i\big({Q^T}_{k,*k-1}(j) T^T_{i+1,*k-1}(j) + {Q^T}_{k,k}\Phi^u(a_i)\big) \\ &= \sigma_i {Q^T}_{k,k}\big(\Phi^u(b_i) - \Phi^u(a_i)\big). \end{aligned}$$

In this package, the implementation restricts itself to uniform and normally distributed input parameters. It resembles the expression in the last equation.

1.2.3 Drawbacks in [Ge.2017]

For the following explanation, I refer to a normal sample space. The drawback in the EE definitions by [Ge.2017] is that Δ is transformed multiple times in the numerator but not in the denominator. Therefore, these measures are not Elementary Effects in the sense of a derivative. To understand the intuition, it is easier to view Δ as $b - a$. The transformation in the numerator is performed by applying $F^{-1}(\Phi^c)$ to $u_i^j = a_i^j + \Delta^{(i,j)}$. The implications of this approach is twofold. The first implication is that function f is evaluated at arguments that are non-linearily transformed by Step 1 and Step 3. Then, the difference in the numerator is divided by the difference between the changed input parameter and the base input parameter – in unit space. Therefore, numerator and denominator refer to different sample spaces. This makes the results hard to interpret. It also increases the influence of more extreme draws in $[0, 1]$ because Φ^{-1} is very sensitive to these. Therefore, it will take a larger sample for the aggregate EE measures

in [Ge.2017] to converge. Additionally, these problems are amplified if there are large differences between the inputs' standard deviation through the subsequent multiplicative scaling. The large sensitivity to more extreme draws implies also a higher sensitivity to larger differences in $\Delta = b - a$. Therefore, the results will differ in their level depending on the sampling scheme. The largest drawback, however, is that $b_i - a_i$ in the denominator of d_i^{ind} does not account for the transformation of the $b_i - a_i$ in the nominator by the establishing of correlations in Step 2. This transformation decreases $b_i - a_i$ proportional to the degree of correlations of the respective input parameter as can be seen by the last row of the lower Cholesky matrix. Hence, d_i^{ind} is inflated by the the input parameters' correlations even tough this measure is introduced as an independent effect. It actually is a dependent effect as well. Because the full EE has to be interpreted with regards to the independent EE, this problem spills over to d_i^{full}. For these reasons, I argue that these measures can not be used for screening.

The next section explains why it is important to sigma-normalize elementary effects.

1.3 Sigma Normalization

The aim of this section is to show how important it can be to sigma-normalize the Elementary Effects or the derived statistics thereof. A code example is given by:

```
# Compute sigma-normalized statistics of Elementary Effects.
measures_sigma_norm = hms.compute_measures(ees_list, sd_qoi, sd_inputs, sigma_
↪norm=True)
```

Let $g(X_1, ..., X_k) = \sum_{i=1}^{k} c_i X_i$ be an arbitrary linear function. Let $\rho_{i,j}$ be the linear correlation between X_i and X_j. Then, for all $i \in 1, ..., k$, I expect

$$d_i^{u,*} = c_i,$$
$$d_i^{c,*} = \sum_{j=1}^{k} \rho_{i,j} c_j.$$

These results correspond to the intuition provided by the example in [Saltelli.2008], p. 123. Both equations state that, conceptually, the result does not depend on the sampling scheme.

Let us consider the case without any correlations between the inputs. Additionally, let $c_i = \{3, 2, 1\}$ and $\sigma_{X_i}^2 = \{1, 4, 9\}$ for $i \in \{1, 2, 3\}$. The following results are derived from [Saltelli.2008]. Let us first compute the Sobol' indices. As g does not include any interactions, $S_i^T = S_i$. Additionally, we have $\text{Var}(Y) = \sum_{i=1}^{k} c_i^2 \sigma_{X_i}^2$ and $\text{Var}_{\boldsymbol{X_{\sim i}}}\left(E_{X_{\sim i}}[Y|\boldsymbol{X_{\sim i}}]\right) = c_i^2 \sigma_{X_i}^2$. Table 1 compares three different sensitvity measures. These are the total Sobol' indices, S_i^T (Measure I, the mean absolute EE, γ_i^* (Measure II), and the *squared* sigma-normalized mean absolute EE, $(\mu_i^* \frac{\sigma_{X_i}}{\sigma_Y})^2$ (Measure III).

Table 1: Table 1: Importance measures for parametric uncertainty

Parameter	Measure I	Measure II	Measure III
X_1	9	3	9
X_2	8	2	8
X_3	9	1	9

In context of screening, S_i^T is the objective measure that we would like to predict approximately. We observe that γ_i^* ranks the parameters incorrectly. The reason is that γ_i^* is only a measure of the influence of X_i on Y and not of the influence of the variation and the level of X_i on the variation of Y. We also see that $(\mu_i^* \frac{\sigma_{X_i}}{\sigma_Y})^2$ is an exact predictor for S_i^T as it does not only generate the correct ranking but also the right effect size. Importantly, this result is specific to a linear function without any interactions and correlations. However, it underlines the point that γ_i^* alone is not sufficient for screening. Following Ge.2017, one approach would be to additionally consider the EE variation, σ_i. However, analysing two measures at once is difficult for models with a large number of input parameters. Table 1 indicates

that $(\mu_i^* \frac{\sigma_{X_i}}{\sigma_Y})^2$ and also $\mu_i^* \frac{\sigma_{X_i}}{\sigma_Y}$ can be an appropriate alternative. The actual derivative version of this measure is also recommended by guidelines of the Intergovernmental Panel for Climate Change ([IPCC.1999]).

1.4 Uncertainty Quantification

According to [Smith.2014], Model-based forecasting includes two main steps: the first step is calibration. In this step, the input parameters of the model are estimated. The second step is the prediction. The prediction contains the model evaluation at the estimated parameters. This allows us to make statements about potential scenarios. These statements are made in a probabilistic way. Thereby, the uncertainty of these statements is emphasised.

There are four sources of uncertainty in modern forecasting that are based on complex computational models ([Smith.2014]). The first source, the model uncertainty, is the uncertainty about whether the mathematical model represents the reality sufficiently. The second source, the input uncertainty, is the uncertainty about the size of the input parameters of the model. The third one, the numerical uncertainty, comes from potential errors and uncertainties introduced by the translation of a mathematical to a computational model. The last source of uncertainty, the measurement uncertainty, is the accuracy of the experimental data that is used to approximate and calibrate the model.

We deal with the second source of uncertainty, the input uncertainty. In my view, this is the source for which UQ offers the most and also the strongest instruments. This might result from the fact that the estimation step produces standard errors as basic measures for the variation or uncertainty in the input parameter estimates. These can then be used to compute a variety of measures for the impact of the input uncertainty on the variation in the model output.

The following explains the basic notation. It is essential to define the quantity that one wants to predict with a model. This quantity is called output, or quantity of interest, and is denoted by Y. For instance, the QoI can be the impact of a 500 USD tuition subsidy for higher education on average schooling years. The uncertain model parameters $X_1, X_2, ..., X_k$ are denoted by vector $\boldsymbol{X}$. The function that computes QoI Y by evaluating a model and, if necessary, post-processing the model output is denoted by $f(X_1, X_2, ..., X_k)$. Thus,

$$Y = f(\boldsymbol{X}). \tag{1.1}$$

From this point forward, I also refer to f as the model. Large-scale UQ applications draw from various fields such as probability, statistics, analysis, and numerical mathematics. These disciplines are used in a combined effort for parameter estimation, surrogate model construction, parameter selection, uncertainty analysis, local sensitivity analysis (LSA), and GSA, amongst others. Drawing from [Smith.2014]) I briefly sketch the first four components. The last two components, local and especially global sensitivity analysis, are discussed more extensively after that.

Parameter estimation covers the calibration step. There is a large number of estimation techniques for various types of models.

If the run time of a model is too long to compute the desired UQ measures, surrogate models are constructed to substitute the original model f ([McBrider.2019]). These surrogate models are functions of the model input parameters which are faster to evaluate. The functions are also called interpolants because they are computed from a random sample of input vectors, drawn from the input distribution and evaluated by the model. Typically, a surrogate model is computed by minimising a distance measure between a predetermined type of function and the model evaluations at the sample points. Therefore, the surrogate model interpolates this sample. Some specifications, like orthogonal polynomials, have properties which can simplify the computation of UQ measures tremendously ([Xiu.2010]).

Another way to reduce the computation time, not directly of the model but of UQ measures, is to reduce the number of uncertain input parameters as part of a parameter selection. The decision which parameters to fix is made based on sensitivity measures. This is called **screening** or *factor fixing* ([Saltelli.2008]). This point will be taken up again in the next section.

Uncertainty analysis is the core of the prediction step. It comprises two parts. The first part is the construction of the QoI's probability distribution by propagating the input uncertainty through the model. For instance, this can be achieved by evaluating a sample of random input parameters (as also required for the construction of a surrogate model). The second part is the computation of descriptive statistics like the probabilities for a set of specific events

in the QoI range using this distribution. Both steps are conceptually simple. The construction of the probability distribution is also important for designing subsequent steps like a sensitivity analysis. For example, if the distribution is unimodal and symmetric, variance-based UQ measures are meaningful. If the distribution has a less tractable, then density-based measures are better suited ([Plischke.2013])).

The next section is a short and general presentation of sensitivity analysis.

1.5 Sensitivity Analysis

This section draws from [Saltelli.2004] and [Saltelli.2008]. They define sensitivity analysis as "the study of how uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input" ([Saltelli.2004], p. 42)). This apportioning implies a ranking of input parameters in terms of their importance for the model output. [Saltelli.2004] (2004, p. 52) define the most important parameter as "the one that [if fixed to its true, albeit unknown value] would lead to the greatest reduction in the variance of the output Y." Therefore, a factor is not important if it influences the level of output Y but rather its variance.

Sensitivity analysis includes different objectives. These have to be determined at first because the choice of methods depends on these objectives. Typically, the main and final goal is factor prioritisation. This is the aforementioned ranking of input parameters in terms of their importance. This ranking can be used to concentrate resources on data acquisition and estimation of a subset of parameters. The methods meeting the demands of factor prioritisation best are called quantitative. These typically require the highest computational effort.

There are multiple other objectives. The objective in this package is **screening** or factor fixing. It is basically the same as factor prioritisation except that it only aims to identify the input parameters that can be fixed at a given value without significantly reducing the output variance. Therefore, it focuses on the lowest parameters in the potential importance ranking. The reason why one would pursue factor fixing instead of factor prioritisation is computational costs. As factor fixing generates less information than factor prioritisation, less powerful methods can be applied. These methods require less computational resources and are called qualitative. Factor fixing can be used to prepare factor prioritisation for models that are more costly to evaluate. In this sense, it serves the same purpose as surrogate modelling.

Another important distinction is local versus global sensitivity analysis (GSA). It essentially refers to the applied methods. In fact, the definition by [Saltelli.2004] is already tailored to a global sensitivity analysis. In contrast to the given definition, "until quite recently, sensitivity analysis was [...] defined as a local measure of the effect of a given input on a given output" ([Saltelli.2004], p. 42)). This older definition differs from the definition used here in two aspects. First, it emphasises the level of output rather than its variance. Second, it describes the measure as a local one. The drawback of this approach becomes clear by considering an example of a local sensitivity measure. This measure is the so-called system derivate $D_i = \frac{\partial Y}{\partial X_i}$ ([Rabitz.1989]). The derivative is typically computed at the mean, $\overline{X_i}$, of the estimate for X_i. D_i is a so-called one-at-a-time measure because it changes only one factor. It has the following four drawbacks: first, it does not account for the interaction between multiple input parameters because it is one-at-a-time. Second, if the model derivation can not be derived analytically, the choice of the (marginal) change in X_i is arbitrary. Third, the local derivative at $\overline{X_i}$ is only representative for the whole sample space of a random input if the model is linear in X_i. Fourth, the measure does not relate to the output variance $Var(Y)$. For these reasons, the field, its definitions and its methods have evolved beyond the notion of local sensitivity analysis. Yet, until recently, the main part of applications in different disciplines, such as physics ([Saltelli.2004] or economics ([Harenberg.2019]), still uses local measures.

The next to sections present quantitative and qualitative GSA. **Screening** measures belong to the latter.

1.6 Quantitative GSA

Quantitative GSA aims to determine the precise effect size of each input parameter and its variation on the output variation. The most common measures in quantitative GSA are the Sobol' sensitivity indices. The next equation gives the general expression for the first order index. Let $\text{Var}_{X_i}(Y|X_i)$ denote the variance of the model output Y conditional on input parameter X_i. Then,

$$S_i = \frac{\text{Var}_{X_i}(Y|X_i)}{\text{Var}(Y)}.$$

The equation becomes clearer with the following equivalent expression in the next one. For this purpose, let $\sim i$ denote the set of indices except i. The expectation of Y for one specific value of X_i equals the average of the model evaluations from a sample, $\boldsymbol{\chi_{\sim i}}$, of $\boldsymbol{X_{\sim i}}$ and a given value $X_i = x_i^*$. Then, we use $E[f(X_i = x_i^*, \boldsymbol{\chi_{\sim i}})] = E_{\boldsymbol{X_{\sim i}}}[Y|X_i]$ to write the first-order Sobol' index as the variance of $E_{\boldsymbol{X_{\sim i}}}[Y|X_i]$ over all x_i^* as

$$S_i = \frac{\text{Var}_{X_i}\left(E_{\boldsymbol{X_{\sim i}}}[Y|X_i]\right)}{\text{Var}(Y)}.$$

The first-order index does not include the additional variance in Y that may occur from the interaction of $\boldsymbol{X_{\sim i}}$ with X_i. This additional variance is included in the total-order Sobol' index given by the next equation. It is the same expression as above except that the positions for X_i and $\boldsymbol{X_{\sim i}}$ are interchanged. Conditioning on $\boldsymbol{X_{\sim i}}$ accounts for the inclusion of the interaction effects of X_i.

$$S_i^T = \frac{\text{Var}_{\boldsymbol{X_{\sim i}}}\left(E_{X_{\sim i}}[Y|\boldsymbol{X_{\sim i}}]\right)}{\text{Var}(Y)}$$

Computing these measures requires many function evaluations, even if an estimator is used as a shortcut ([Saltelli.2004]). The more time-intense one function evaluation is, the more utility does factor fixing based on qualitative measures provide.

1.7 Qualitative GSA

Qualitative GSA deals with the computation of measures that can rank random input parameters in terms of their impact on the function output and the variability thereof. This is done to a degree of accuracy that allows distinguishing between influential and non-influential parameters. If the measures for some input parameters are negligibly small, these parameters can be fixed so that the number of random input parameters decreases for a subsequent quantitative GSA. This section explains the qualitative measures and the trade-off between computational costs and accuracy.

The most commonly used measures in qualitative GSA is the mean EE, μ, the mean absolute EEs, μ^*, and the standard deviation of the EEs, σ. The EE of X_i is given by one individual function derivative with respect to X_i. The "change in", or the "step of" the input parameter, denoted by Δ. The only restriction is that $X_i + \Delta$ is in the sample space of X_i. The Elementary Effect, or derivative, is denoted by

$$d_i^{(j)} = \frac{f(\boldsymbol{X_{\sim i}^{(j)}}, X_i^{(j)} + \Delta^{(i,j)}) - f(\boldsymbol{X})}{\Delta^{(i,j)}},$$

where j is an index for the number of r observations of X_i. Note, that the EE, $d_i^{(j)}$, is equal to the aforementioned local measure, the system derivate $S_i = \frac{\partial Y}{\partial X_i}$, except that the value Δ has not to be infinitesimally small. To offset the third drawback of d_i and S_i, that base vector X_i does not represent the whole input space, one computes the mean EE, μ_i, based on a random sample of X_i from the input space. The second drawback, that interaction effects are disregarded, is also offset because elements $X_{\sim i}$ are also resampled for each new X_i. The mean EE is given by

$$\mu_i = \frac{1}{r}\sum_{j=1}^{r} d_i^{(j)}.$$

Thus, μ_i is the global version of $d_i^{(j)}$. The standard deviation of the EEs writes $\sigma_i = \sqrt{\frac{1}{r}\sum_{j=1}^{r}(d_i^{(j)} - \mu_i)^2}$. The mean absolute EE, μ_i^*, is used to prevent observations of opposite sign to cancel each other out:

$$\mu_i^* = \frac{1}{r}\sum_{j=1}^{r}\left|d_i^{(j)}\right|.$$

Step $\Delta^{(i,j)}$ may or may not vary depending on the sample design that is used to draw the input parameters.

One last variant is provided in [Smith.2014]. That is, the scaling of $\mu_i^* by \frac{\sigma_{X_i}}{\sigma_Y}$. This measure is called the sigma-normalized mean absolute EE:

$$\mu_{i,\sigma}^* = \mu_i^* \frac{\sigma_{X_i}}{\sigma_Y}.$$

This improvement is necessary for a consistent ranking of X_i. Otherwise, the ranking would be distorted by differences in the level of the the input parameters. The reason is that the input space constrains Δ. If the input space is larger, the base value of X_i can be changed by a larger Δ.

From the aforementioned set of drawbacks of the local derivate $D_i = \frac{\partial Y}{\partial X_i}$, two drawbacks are remaining for the EE method. The first drawback is the missing direct link to the variation in $Var(Y)$. The second drawback is that the choice of Δ is somewhat arbitrary if the derivative is not analytic. To this date, the literature has not developed convincing solutions for these issues.

In an attempt to establish a closer link between EE-based measures and Sobol' indices, [Kucherenko.2009] made two conclusions: the first conclusion is that there is an upper bound for the total index, S_i^T, such that

$$S_i^T \leq \frac{\frac{1}{r}\sum_{j=1}^{r} d_i^{2(j)}}{\pi^2 \sigma_Y}.$$

This expression makes use of the squared EE. In light of this characteristic, the use of σ_i as a measure that aims to include the variation of d_i^j appears less relevant. Nevertheless, this rescaling makes the interpretation more difficult. The second conclusion is that the Elementary Effects method can lead to false selections for non-monotonic functions. This is also true if functions are non-linear. The reason is linked to the aforementioned second drawback, the arbitrary choice of step Δ. More precisely, depending on the sampling scheme, Δ might be random instead of arbitrary and constant. In both cases, Δ can be too large to approximate a derivative. If, for example, the function is highly non-linear of varying degree with respect to the input parameters $\boldsymbol{X}$, $\Delta > \epsilon$ can easily distort the results. Especially if the characteristic length of function variation is much smaller than Δ.

1.8 Correlated inputs

So far, all described measures assumed uncorrelated input parameters. Typically, this assumption is not met by practical applications as joint estimations of parameters tend to produce correlated estimates. This gives reason to expand sensitivity measures to a more general setting.

Today, several recent contributions deal with the extension of the Sobol' sensitivity measures to the setup with correlated inputs. For instance, estimators for two complementary sets of measures are developed by [Kucherenko.2012] and [Mara.2015]. On the other hand, the only contribution to the computation of EE-based screening measures for correlated parameters is made by [Ge.2017]. Some authors, e.g. [Saltelli.2004] even negate the necessity for expanded Elementary Effects due to overcomplexity. Obviously, this can lead to false results.

1.9 References

1.10 License

1.11 Modules

1.11.1 hypermodern_screening.sampling_schemes

1.11.2 hypermodern_python.select_sample_set

Decrease a set of sample sets in order to increase its representativeness.

These approaches are developed in the context of the trajectory design because it can not cover the space very densely. The methods are taken from the effective screening design in [1] and the efficient screening design in [2].

References

[1] Campolongo, F., J. Cariboni, and A. Saltelli (2007). An effective screening design for sensitivity analysis of large models. Environmental modelling & software 22 (10), 1509–1518. [2] Ge, Q. and M. Menendez (2014). An efficient sensitivity analysis approach for computationally expensive microscopic traffic simulation models. International Journal of Transportation 2 (2), 49–64.

hypermodern_screening.select_sample_set.**campolongo_2007**(*sample_traj_list*, *n_traj*)

Implement the post-selected sample set in [1].

Takes a list of Morris trajectories and selects the *n_traj* trajectories with the largest distance between them. Returns the selection as array with n_inputs at the verical and n_traj at the horizontal axis and as a list. It also returns the diagonal matrix that contains the pair distance between each trajectory pair.

Parameters

- **sample_traj_list** (*list of ndarrays*) – Set of samples.
- **n_traj** (*int*) – Number of samples to choose from *sample_traj_list*.

Return type Tuple[List, ndarray, List]

Returns

- **sample_traj_list** (*list of ndarrays*) – Set of trajectories.
- **select_dist_matrix** (*ndarray*) – Symmetric *distance_matrix* of selection.
- **select_indices** (*list*) – Indices of selected samples.

hypermodern_screening.select_sample_set.**combi_wrapper**(*iterable*, *r*)

Wrap *itertools.combinations*, written in C, see [1].

Parameters

- **iterable** (*iterable object*) – Hashable container like a list of distinct elements to combine.
- **r** (*int*) – Number to draw from *iterable* with putting back and regarding the order.

Returns All possible combinations in ascending order.

Return type list_list

Example

```
>>> combi_wrapper([0, 1, 2, 3], 2)
[[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]
```

References

[1] https://docs.python.org/2/library/itertools.html#itertools.combinations.

hypermodern_screening.select_sample_set.**compute_pair_distance**(*sample_0*, *sample_1*)

Compute the distance measure between a pair of samples.

The aggregate distance between sum of the root of the square distance between each parameter vector of one sample to each vector of the other sample.

Parameters

- **sample_0** (ndarray) – Sample with paramters in cols and draws as rows.
- **sample_1** (ndarray) – Sample with paramters in cols and draws as rows.

Returns Pair distance.

Return type distance

Raises

- **AssertionError** – If sample is not in trajectory or radial design shape.
- **AssertionError** – If the sample shapes differ.

Notes

The distance between two samples is sum of the root of the square distance between each parameter vector of one sample to each vector of the other sample.

hypermodern_screening.select_sample_set.**distance_matrix**(*sample_list*)

Compute symmetric matrix of pair distances for a list of samples.

Parameters **sample_list** (List[ndarray]) – Set of samples.

Returns Symmatric matrix of pair distances.

Return type distance_matrix

hypermodern_screening.select_sample_set.**final_ge_menendez_2014**(*sample_traj_list*, *n_traj*)

Implement both "improvements" in [2] vis-a-vis [1].

Parameters

- **sample_traj_list** (List[ndarray]) – Set of samples.
- **n_traj** (int) – Number of samples to choose from *sample_traj_list*.

Return type Tuple[List[ndarray], ndarray, List[int]]

Returns

- *sample_traj_list* – Set of trajectories.
- *select_dist_matrix* – Symmetric *distance_matrix* of selection.
- *select_indices* – Indices of selected samples.

See also:

next_combi_total_distance_gm14()

Notes

This function, is in fact much slower than *intermediate_ge_menendez_2014* because it uses more for loops to get the pair distances from the right combinations that must be subtracted from the total distances. This function selects n_traj trajectories from n_traj_sample trajectories by iteratively selecting n_traj_sample - i for i = 1,. . . ,n_traj_sample - n-traj. For this purpose, next_combi_total_distance_gm14 computes the total distance of each trajectory combination by using the total distance of each combination in the previous step and subtracting each pair distance with the dropped trajectory, that yielded the lowest total distance combinations in the previous step.

hypermodern_screening.select_sample_set.**intermediate_ge_menendez_2014**(*sample_traj_list*, *n_traj*)

Implement the essential of the two "improvements" in[2] vis-a-vis [1].

This is basically a wrapper around *select_trajectories_wrapper_iteration*.

Parameters

- **sample_traj_list** (*list of ndarrays*) – Set of samples.
- **n_traj** (*int*) – Number of samples to choose from *sample_traj_list*.

Return type Tuple[List, ndarray, List]

Returns

- **sample_traj_list** (*list of ndarrays*) – Set of trajectories.
- **select_dist_matrix** (*ndarray*) – Symmetric *distance_matrix* of selection.
- **select_indices** (*list*) – Indices of selected samples.

See also:

select_trajectories_wrapper_iteration()

Notes

Oftentimes this function leads to diffent combinations than *select_trajectories*. However, their total distance is very close to the optimal solution.

hypermodern_screening.select_sample_set.**next_combi_total_distance_gm14**(*combi_total_distance*, *pair_dist_matrix*, *lost_index*)

Select the set of samples minus one sample.

Based on the algorithmic computation of the *total_distance* proposed by [2]. I.e. by re-using and adjusting the first *combi_total_distance* matrix each iteration. Used for selecting iteratively rather than by brute force.

Parameters

- **combi_total_distance_next** – Matrix with n_traj + 1 rows. The first n_traj cols are filled with indices of samples and the last column is the *total_distance* of the combinations of samples marked by indices in the same row and the columns before.
- **pair_dist_matrix** (ndarray) – Distance matrix of all combinations and their total_distance.
- **lost_index** (int) – index of the sample that will be dropped from the samples in the above objects.

Return type Tuple[ndarray, ndarray, ndarray]

Returns

- *combi_total_distance_next – combi_total_distance* without the dropped sample.
- *pair_dist_matrix_next – pair_dist_matrix* without the dropped sample.
- *lost_index – lost_index* without the dropped sample one iteration before.

Notes

The function computes the total distance of each trajectory combination by using the total distance of each combination in the previous step and subtracting each pair distance with the dropped trajectory, that yielded the lowest total distance combinations in the previous step. This function, is in fact much slower than *select_trajectories_wrapper_iteration* because it uses more for loops to get the pair distances from the right combinations that must be subtracted from the total distances.

hypermodern_screening.select_sample_set.**select_sample_set_normal**(*samp_list*, *n_select*, *numeric_zero*)

Post-select set of samples based on [0,1] and transform it to stnormal space.

Parameters

- **samp_list** (List[ndarray]) – Sub-samples.
- **n_select** (int) – Number of sub-samples to select from *samp_list*.
- **numeric_zero** (float) – *if normal is True*: Prevents *scipy.normal.ppt* to return *-Inf* and *Inf* for 0 and 1.

Return type Tuple[List[ndarray], List[ndarray]]

Returns

- *samp_list*
- *steps_list*

Notes

Function for post-selection is *intermediate_ge_menendez_2014* because it is the fastest.

See also:

intermediate_ge_menendez_2014()

hypermodern_screening.select_sample_set.**select_trajectories**(*pair_dist_matrix*, *n_traj*)

Compute *total distance* for each *n_traj* combinations of a set of samples.

Parameters

- **pair_dist_matrix** (ndarray) – *distance_matrix* for a sample set.
- **n_traj** (int) – Number of sample combinations for which the *total_distance* is computed.

Return type Tuple[List, ndarray]

Returns

- **max_dist_indices** (*list of ints*) – Indices of samples in *pair_dist_matrix* that are part of the combination with the largest *total_distance*.
- **combi_total_distance** (*ndarray*) – Matrix with n_traj + 1 rows. The first n_traj cols are filled with indices of samples and the last column is the *total_distance* of the combinations of samples marked by indices in the same row and the columns before.

Raises

- **AssertionError** – If *pair_dist_matrix* is not symmetric.

- **AssertionError** – If the number of combinations does not correspong to the combinations indicated by the size of *pair_dist_matrix*.

Notes

This function can be very slow because it computes distances between np.binomial(len(pair_dist_matrix, n_traj) pairs of trajectories. Example: *np.biomial(30,15)* = 155117520. This selection function yields precise results because each total distance for each possible combination of trajectories is computed directly. The faster, iterative methods can yield different results that are, however, close in the total distance. The total distances tend to differentiate clearly. Therefore, the optimal combination is precisely determined.

hypermodern_screening.select_sample_set.**select_trajectories_wrapper_iteration**(*pair_dist_matrix*, *n_traj*)

Select the set of samples minus one sample.

Used for selecting iteratively rather than by brute force. Implements the main step of the essential of the two "improvements" from [2] to [1].

Parameters

- **pair_dist_matrix** (*ndarray*) – Distance matrix of all combinations and their total_distance.
- **n_traj** (*int*) – number of samples to choose from a set of samples based on their *total_distance*.

Return type Tuple[List, ndarray]

Returns

- **tracker_keep_indices** (*list*) – Indices of samples part of the selection.
- **combi_total_distance** (*ndarray*) – Matrix with n_traj + 1 rows. The first n_traj cols are filled with indices of samples and the last column is the *total_distance* of the combinations of samples marked by indices in the same row and the columns before.

See also:

select_trajectories()

Notes

Oftentimes this function leads to diffent combinations than *select_trajectories*. The reason seems to be that this function deviates from the optimal path due to numerical reasons as different combinations may be very close (see [2]). However, the total sum of the returned combinations are close. Therefore, the *total_distance* loss is negligible compared to the speed gain for large numbers of trajectory combinations. This implies that, *combi_total_distance* always differs from the one in *select_trajectories* because it only contains the combination indices from the last iteration if n_traj is smaller than the sample set minus 1. The trick using *tracker_keep_indices* is an elegant solution.

hypermodern_screening.select_sample_set.**total_distance**(*distance_matrix*)

Compute the total distance measure of all pairs of samples in a set.

The equation corresponds to Equation (10) in [2].

Parameters **distance_matrix** (*ndarray*) – diagonal matrix of distances for sample pairs.

Returns **total_distance** – total distance measure of all pairs of samples in a set.

Return type float

1.11.3 hypermodern_python.transform_distributions

Functions for the inverse Rosenblatt / inverse Nataf transformation (u to z_c).

hypermodern_screening.transform_distributions.**covariance_to_correlation**(*cov*)

Convert covariance matrix to correlation matrix.

Parameters **cov** (ndarray) – Covariance matrix.

Returns Correlation matrix.

Return type corr

hypermodern_screening.transform_distributions.**transform_stnormal_normal_corr**(*z_row*, *cov*, *mu*)

Transform u to z_c.

Transformation from standard normal to multivariate normal space with given correlations following [1], page 77-102.

Step 1) Compute correlation matrix. Step 2) Introduce dependencies to standard normal sample. Step 3) Destandardize sample to normal space.

Parameters

- **z_row** (ndarray) – Row of uncorrelated standard normal deviates.
- **cov** (ndarray) – Covariance matrix of correlated normal deviates.
- **mu** (ndarray) – Expectation values of correlated normal deviates

Return type Tuple[ndarray, float]

Returns

- *x_norm_row* – Row of correlated normal deviates.
- *correlate_step* – Lower right corner element of the lower Cholesky matrix.

Notes

Importantly, the step in the numerator of the uncorrelated Elementary Effect is multiplied by *correlate_step*. Therefore, this factor has to multiply the step in the denominator as well to not violate the definition of the function derivation. This method is equivalent to the one in [2], page 199 which uses the Cholesky decomposition of the covariance matrix directly. This saves the scaling by SD and expectation. This method is simpler and slightly more precise than the one in [3], page 33, for normally distributed paramters. [1] explains how Rosenblatt and Nataf transformation are equal for normally distributed deviates.

References

[1] Lemaire, M. (2013). Structural reliability. John Wiley & Sons. [2] Gentle, J. E. (2006). Random number generation and Monte Carlo methods. Springer Science & Business Media. [3] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivity analysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

hypermodern_screening.transform_distributions.**transform_uniform_stnormal_uncorr**(*uniform_deviat*, *numeric_zero=0.0*

Transorm u to z_u.

Converts sample from uniform distribution to standard normal space without regarding correlations.

Parameters

- **uniform_deviates** (ndarray) – Draws from Uniform[0,1].
- **numeric_zero** (float) – Used to substitute zeros and ones before applying *scipy.stats.norm* to not obtain *-Inf* and *Inf*.

Returns *uniform deviates* converted to standard normal space without correlations.

Return type stnormal_deviates

See also:

morris_trajectory()

Notes

This transformation is already applied as option in *morris_trajectory*. The reason is that *scipy.stats.norm* transforms the random draws from the unit cube non-linearily including the addition of the step. To obtain non-distorted screening measures, it is important to also account for this transformation of the step in the denominator to not violate the definition of the function derivation. The parameter *numeric_zero* can be highly influential. I prefer it to be relatively large to put more proportional, i.e. less weight on the extremes.

1.11.4 hypermodern_python.transform_reorder

Functions for reordering the sample rows following [1].

The intuition behind the reordering in general is the following: To compute the uncorrelated Elementary Effects, one moves the sampled elements that have been changed by *step* to the back of the row. For the correlated EE, one leaves the newly changed element in front, but moves the elements that were changed in rows above to the end. These compose the left parts of the numerator in the EE definition. One then subtracts the same row, except that the changed element is unchanged. The reason for these reorderings is that the correlation technique works hierarchically, like Dominoes. The element before is unaffected by the correlation of the elements thereafter. This implies that the first element is unchanged, as for the correlated EE. Therefore, the step is involved in correlating the other elements without becoming changed itself. The opposite is true for the uncorrelated EE. The above procedure is dervied from the ordering in trajectory samples. It also works for the radial design. Other functions order the expectations and covariance matrix accordingly. They are also used to initialize the correlating loops in the two functions in *transform_ee.py* in the right order.

References

[1] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivityanalysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

hypermodern_screening.transform_reorder.**ee_corr_reorder_sample**(*sample*)

For each row i (non-pythonic), move the first i-1 elements to the back.

Parameters **sample** (ndarray) – sample.

Returns Reordered sample.

Return type sample_reordered

Notes

There is no *row_plus_one=False* option because this is equivalent with *uncorr_reorder_sample(sample, row_plus_one=True)*.

hypermodern_screening.transform_reorder.**ee_uncorr_reorder_sample**(*sample*, *row_plus_one=True*)

For each row i (non-pythonic), move the first i elements to the back.

Parameters

- **sample** (ndarray) – sample.
- **row_plus_one** (bool) – Add 1 to row index, i.e. start with second row.

Returns Reordered sample.

Return type sample_reordered

hypermodern_screening.transform_reorder.**reorder_cov**(*cov*)

Arrange covariance matrix according to the expectation vector.

(When the first element is moved to the end.)

Parameters **cov** (ndarray) – Covariance matrix of row.

Returns Reordered covariance matrix of row.

Return type cov_reordered

hypermodern_screening.transform_reorder.**reorder_mu**(*mu*)

Move the first element of the expectation vector to the end.

Parameters **mu** (ndarray) – Expectation values of row.

Returns Reordered expectation values of row.

Return type mu_reordered

hypermodern_screening.transform_reorder.**reverse_ee_corr_reorder_sample**(*sample_reordered*)

Reverse of function *corr_reorder_sample*.

Parameters **sample_reordered** (ndarray) – Reordered sample.

Returns Trjectory in original order.

Return type sample

hypermodern_screening.transform_reorder.**reverse_ee_uncorr_reorder_sample**(*sample_reordered*, *row_plus_one=True*)

Reverse of function *uncorr_reorder_sample*.

Parameters **sample_reordered** (*ndarray*) – Reordered sample.

Returns **sample** – Trjectory in original order.

Return type ndarray

hypermodern_screening.transform_reorder.**reverse_reorder_cov**(*cov_reordered*)

Reverse of function *reorder_cov*.

Parameters **cov_reordered** (*ndarray*) – Reordered covariance matrix.

Returns **cov** – Covarince matrix in original order.

Return type ndarray

hypermodern_screening.transform_reorder.**reverse_reorder_mu**(*mu_reordered*)
Reverse of function *reorder_mu*.

Parameters **mu_reordered** (ndarray) – Reordered expectation values of row.

Returns Expectation values of row in original order.

Return type mu

1.11.5 hypermodern_python.transform_ee

Compute the component for the redesigned EE expressions.

Functions to compute the arguments for the function evaluations in the numerator of the individual uncorrelated and correlated Elementary Effects following [1], page 33 and 34, and coefficients that scale the step. These functions can handle samples in both, trajectory and radial, designs.

References

[1] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivity analysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

hypermodern_screening.transform_ee.**trans_ee_corr**(*sample_list*, *cov*, *mu*, *radial=False*)
Transform list of samples to two lists of transformed samples.

(For the computation of the correlated Elementary Effects.)

Parameters

- **sample_list** (List) – Set of untransformed samples.
- **cov** (ndarray) – Covariance matrix.
- **mu** (ndarray) – Expectation value.
- **radial** (bool) – Sample is in trajectory or radial design.

Returns samples containing the rows that are the arguments for the LHS function evaluation for the correlated Elementary Effect.

Return type trans_piplusone_iminusone

Raises **AssertionError** – If the dimension of *mu*, *cov* and the elements in *sample_list* do not fit together.

Notes

For the trajectory design, the transformation for the rows on the RHS of the correlated Elementary Effects is equal to the one on the LHS of the uncorrelated Elementary Effects. Therefore, if *radial is False*, this transformation is skipped and left to *trans_ee_uncorr_samples*. To compute the EEs from radial samples, the arguments of the subtracted function are the first row of the sample. Yet, they must be reordered and transformed according to their order, too.

See also:

trans_ee_uncorr_samples()

hypermodern_screening.transform_ee.**trans_ee_uncorr**(*sample_list*, *cov*, *mu*, *radial=False*)
Transform list of samples to two lists of transformed samples.

(For the computation of the uncorrelated Elementary Effects.)

Parameters

- **sample_list** (List[ndarray]) – Set of untransformed samples.
- **cov** (*np.ndarray*) – Covariance matrix.
- **mu** (ndarray) – Expectation value.
- **radial** (bool) – Sample is in trajectory or radial design.

Return type Tuple[List[ndarray], List[ndarray], List[ndarray]]

Returns

- *trans_piplusone_i* – samples containing the rows that are the arguments for the LHS function evaluation for the uncorrelated Elementary Effect.
- *trans_pi_i* – samples containing the rows that are the arguments for the RHS function evaluation for the uncorrelated Elementary Effect.
- *coeff_step* – Factors in the denumerator of the uncorrelated Elementary Effect. Accounts for the decorrelation of the Step.

Raises **AssertionError** – If the dimension of *mu*, *cov* and the elements in *sample_list* do not fit together.

Notes

The rows in the two different transformed samples equal to T(p_{i+1}, i) and T(p_{i}, i). Understanding the transformations may require to write up the first transformation from p_i and p_{i+1} to T_1(p_{i}, i) and T_1(p_{i+1}, i). T_1 shifts the first i elements to the end for each row p_{i}. This function creates list of transformations of whole samples. The rows in the samples for T(p_{i}, i) that are to be subtracted from T(p_{i+1}, i), are still positioned one below compared to the samples for T(p_{i}, i). Therefore, importantly, one needs to compare each row in a sample from *trans_pi_i* with the respective row one below in *trans_piplusone_i*. To compute the EEs from radial samples, the arguments of the subtracted function are the first row of the sample. Yet, they must be reordered and transformed according to their order, too.

1.11.6 hypermodern_python.screening_measures

Compute Elementary Effects from transformed samples and derived measures.

Computes the screening measures for correlated inputs that I improved upon [1] by adjusting the step in the denumeroter to the transformed step in the nominator in order to not violate the definition of the function derivative.

References

[1] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivityanalysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

hypermodern_screening.screening_measures.**compute_measures**(*ee_i*, *sd_x=array([1])*, *sd_y=array([1])*, *sigma_norm=False*, *ub=False*)

Compute aggregate measures based on (individual) Elementary Effects.

Parameters

- **ee_i** (ndarray) – (individual) Elementary Effects of input paramters (cols).
- **sd_x** (ndarray) – Parameters' SD.
- **sd_y** (ndarray) – QoI's SD.
- **sigma_norm** (bool) – Indicates wether to compute measures normalized by *sd_x / sd_y*.
- **ub** (bool) – Indicates wether to compute squared EEs and measures normalized by *var_x / var_y*.

Returns

contains:

ee_mean Mean Elementary Effect for each parameter.

ee_abs_mean Mean absolute correlated Elementary Effect for each parameter.

ee_sd SD of correlated Elementary Effects for each parameter.

Return type measures_list

Notes

ub follows http://www.andreasaltelli.eu/file/repository/DGSA_MATCOM_2009.pdf.

hypermodern_screening.screening_measures.**screening_measures**(*function*, *traj_list*, *step_list*, *cov*, *mu*, *radial=False*)

Compute screening measures for a set of paramters.

Parameters

- **function** (Callable) – Function or Model of which its parameters are subject to screening.
- **traj_list** (List[ndarray]) – List of transformed trajectories according to [1].
- **step_list** (List[ndarray]) – List of steps that each parameter takes in each trajectory.
- **cov** (ndarray) – Covariance matrix of the input parameters.
- **mu** (ndarray) – Expectation values of the input parameters.
- **radial** (bool) – Sample is in trajectory or radial design.

Return type Tuple[List[ndarray], List[ndarray]]

Returns

- *measures_list* –

  contains:

  ee_uncorr Mean uncorrelated Elementary Effect for each parameter.

  ee_corr Mean correlated Elementary Effect for each parameter.

  abs_ee_uncorr Mean absolute uncorrelated Elementary Effect for each parameter.

  abs_ee_corr Mean absolute correlated Elementary Effect for each parameter.

  sd_ee_uncorr SD of uncorrelated Elementary Effects for each parameter.

  sd_ee_corr SD of correlated Elementary Effects for each parameter.

- *obs_list* –

contains:

ee_uncorr_i Observations of uncorrelated Elementary Effects.

ee_corr_i Observations of correlated Elementary Effects.

Notes

The samples can be in trajectory or in radial design and the deviates can be from an arbitrary (correlated) normal distribution or an uncorrelated Uniform[0,1] distribution. Unorrelated uniform paramters require different interpretion of *mu* as a scaling summand rather than the expectation value. It might be necessary to multiply the SDs by *(n_trajs/(n_trajs - 1))* for the precise formula. However, this leads to problems for the case of only one trajectory - which is used in *test_screening_measures_uncorrelated_g_function*.

CHAPTER

TWO

REFERENCES

Stenzel, T. (2020): Uncertainty Quantification for an Eckstein-Keane-Wolpin model with correlated input parameters. *Master's thesis, University of Bonn*.

Ge, Q. and Menendez, M. (2017). Extending Morris method for qualitative global sensitivity analysis of models with dependent inputs. *Reliability Engineering & System Safety 100(162)*, 28-39.

BIBLIOGRAPHY

[Campolongo.2007] Campolongo, F., A. Saltelli, and J. Cariboni (2011). From screening to quantitative sensitivity analysis. a unified approach. Computer Physics Communications 182 (4), 978–988.

[Campolongo.2011] Campolongo, F., A. Saltelli, and J. Cariboni (2011). From screening to quantitative sensitivity analysis. a unified approach. Computer Physics Communications 182 (4), 978–988.

[Devroye.1986] Devroye, L. (1986). Sample-based non-uniform random variate generation. In Proceedings of the 18th conference on Winter simulation, 260–265.

[Ge.2014] Ge, Q. and M. Menendez (2014). An efficient sensitivity analysis approach for computationally expensive microscopic traffic simulation models. International Journal of Transportation 2 (2), 49–64.

[Ge.2017] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivity analysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

[Gentle.2006] Gentle, J. E. (2006). Random number generation and Monte Carlo methods. Springer Science & Business Media.

[IPCC.1999] IPCC (1999). Ipcc expert meetings on good practice guidance and uncertainty management in national greenhouse gas inventories. Background papers.

[Kucherenko.2009] Kucherenko, S. et al. (2009). Derivative based global sensitivity measures and their link with global sensitivity indices. Mathematics and Computers in Simulation 79 (10), 3009–3017.

[Kucherenko.2012] Kucherenko, S., S. Tarantola, and P. Annoni (2012). Estimation of global sensitivity indices for models with dependent variables. Computer physics communications 183 (4), 937–946.

[Lemaire.2013] Lemaire, M. (2013). Structural reliability. John Wiley & Sons.

[Madar.2015] Madar, V. (2015). Direct formulation to cholesky decomposition of a general nonsingular correlation matrix. Statistics & probability letters 103, 142–147.

[Mara.2015] Mara, T. A., S. Tarantola, and P. Annoni (2015). Non-parametric methods for global sensitivity analysis of model output with dependent inputs. Environmental modelling & software 72, 173–183.

[McBrider.2019] McBride, K. and K. Sundmacher (2019). Overview of surrogate modeling in chemical process engineering. Chemie Ingenieur Technik 91 (3), 228–239.

[Morris.1991] Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. Technometrics 33 (2), 161–174.

[Plischke.2013] Plischke, E., E. Borgonovo, and C. L. Smith (2013b). Global sensitivity measures from given data. European Journal of Operational Research 226 (3), 536–550.

[Rabitz.1989] Rabitz, H. (1989). Systems analysis at the molecular scale. Science 246 (4927), 221–226.

[Smith.2014] Smith, R. C. (2014). Uncertainty Quantification: Theory, Implementation, and Applications. Philadelphia: SIAM-Society for Industrial and Applied Mathematics.

[Saltelli.2002] Saltelli, A. (2002). Making best use of model evaluations to compute sensitivity indices. Computer physics communications 145 (2), 280–297.

[Saltelli.2004] Saltelli, A., S. Tarantola, F. Campolongo, and M. Ratto (2004). Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models. John Wiley & Sons.

[Saltelli.2008] Saltelli, A., M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola (2008). Global Sensitivity Analysis: The Primer. John Wiley & Sons.

[Xiu.2010] Xiu, D. (2010). Numerical methods for stochastic computations: a spectral method approach. Princeton university press.

PYTHON MODULE INDEX

h

INDEX