Gaussian Process Regression and Emulation

STAT8810, Fall 2017

M.T. Pratola

September 22, 2017

Today

Experimental Design;
Sensitivity Analysis

Designing Your Experiment

- If you will run a simulator model, or otherwise collect data in a prescribed manner (i.e. someone has not simply handed you the data), then you should select the settings of the input variables, $\mathbf{x}_i, i = 1, \ldots, n$ in a "sensible" manner.

Designing Your Experiment

- If you will run a simulator model, or otherwise collect data in a prescribed manner (i.e. someone has not simply handed you the data), then you should select the settings of the input variables, $\mathbf{x}_i, i = 1, \ldots, n$ in a "sensible" manner.
- Usually this is defined by a criterion.

Designing Your Experiment

- If you will run a simulator model, or otherwise collect data in a prescribed manner (i.e. someone has not simply handed you the data), then you should select the settings of the input variables, $\mathbf{x}_i, i = 1, \ldots, n$ in a "sensible" manner.
- Usually this is defined by a criterion.
    - For example, minimize the prediction error of your statistical emulator.

Designing Your Experiment

- If you will run a simulator model, or otherwise collect data in a prescribed manner (i.e. someone has not simply handed you the data), then you should select the settings of the input variables, $\mathbf{x}_i, i = 1, \ldots, n$ in a "sensible" manner.
- Usually this is defined by a criterion.
    - For example, minimize the prediction error of your statistical emulator.
- This is a large and complex subject, so we will limit ourselves to designs which are more generally useful for predicting "black-box" simulators.

Optimal Design

- Assume a statistical emulator model for $f(\mathbf{x})$ - say $Z(\mathbf{x}) \sim GP(\boldsymbol{\mu}, \sigma^2\mathbf{R})$ with known mean μ, variance σ^2 and correlation function parameters $\boldsymbol{\rho}$.

Optimal Design

- Assume a statistical emulator model for $f(\mathbf{x})$ - say $Z(\mathbf{x}) \sim GP(\boldsymbol{\mu}, \sigma^2 \mathbf{R})$ with known mean μ, variance σ^2 and correlation function parameters $\boldsymbol{\rho}$.
- In optimal design we optimize some criterion with respect to the settings of $\mathbf{x}_i, i = 1, \ldots, n$.

Optimal Design

- Assume a statistical emulator model for $f(\mathbf{x})$ - say $Z(\mathbf{x}) \sim GP(\boldsymbol{\mu}, \sigma^2\mathbf{R})$ with known mean μ, variance σ^2 and correlation function parameters $\boldsymbol{\rho}$.
- In optimal design we optimize some criterion with respect to the settings of $\mathbf{x}_i, i = 1, \ldots, n$.
- The design is the collection of best settings at which to collect our data,

$$\mathbf{D}^* = (\mathbf{x}_1^*, \ldots, x_n^*) \text{ such that } \mathbf{x}_i^* \in \chi \subseteq \mathbb{R}^p.$$

Optimal Design

- Assume a statistical emulator model for $f(\mathbf{x})$ - say $Z(\mathbf{x}) \sim GP(\boldsymbol{\mu}, \sigma^2\mathbf{R})$ with known mean μ, variance σ^2 and correlation function parameters $\boldsymbol{\rho}$.
- In optimal design we optimize some criterion with respect to the settings of $\mathbf{x}_i, i = 1, \ldots, n$.
- The design is the collection of best settings at which to collect our data,

$$\mathbf{D}^* = (\mathbf{x}_1^*, \ldots, x_n^*) \text{ such that } \mathbf{x}_i^* \in \chi \subseteq \mathbb{R}^p.$$

- The general form of the problem is

$$\mathbf{D}^* = \arg\min_{\mathbf{D}} \mathcal{J}(\mathbf{D})$$

where $\mathbf{D}$ is searched over all possible n-run designs. Typically this optimization is done over a discretization of χ rather than the continuous version.

Optimal Design

- Note that this is itself a non-trivial problem.

Optimal Design

- Note that this is itself a non-trivial problem.
- If we discretize χ as an N-grid then we have $\binom{N}{n}$ possible designs.

Optimal Design

- Note that this is itself a non-trivial problem.
- If we discretize χ as an N-grid then we have $\binom{N}{n}$ possible designs.
- Typically we will plug-in estimates of $\boldsymbol{\mu}, \sigma^2, \boldsymbol{\rho}$ as taking into account their uncertainty makes the computational cost much worse.

Optimal Design

- Note that this is itself a non-trivial problem.
- If we discretize χ as an N-grid then we have $\binom{N}{n}$ possible designs.
- Typically we will plug-in estimates of $\boldsymbol{\mu}, \sigma^2, \boldsymbol{\rho}$ as taking into account their uncertainty makes the computational cost much worse.
- We are trying to optimize $n \times p$ parameters in this problem - a high-dimensional optimization problem.

Optimal Design

- For our purpose, we will most often be interested in prediction/emulation so an appropriate design criterion is the *Integrated Mean Squared Error* criterion†,

$\hat{Z}(x) = E[Z(x) \mid \underbrace{Z_1, \ldots, Z_n}]$

$Z(x_1^*)$
$\vdots$
$Z(x_n^*)$

$$\mathcal{J}(\mathbf{D}) = \frac{1}{\sigma^2} \int_{\chi} E\left[\left(Z(\mathbf{x}) - \hat{Z}(\mathbf{x})\right)^2\right] d\mathbf{x}$$

where in our usual assumed simple setup ($\boldsymbol{\mu} = 0$) we have

$D = \{x_i^*\}_{i=1}^n$

$$\widehat{Z}(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{R}^{-1} \mathbf{Z}$$

or in the general setup

$$\widehat{Z}(\mathbf{x}) = \mathbf{f}^T \beta + \mathbf{r}(\mathbf{x})^T \mathbf{R}^{-1} (\mathbf{Z} - \mathbf{F}\beta).$$

† A convenient closed-form expression is available in Sacks, Welch, Mitchell and Wynn: *Design and Analysis of Computer Experiments*, Statistical Science, vol.4, pp.409–423 (1989).

Space-Filling Designs

- In order to simplify the criterion, so-called “space-filling” designs were proposed.

Space-Filling Designs

- In order to simplify the criterion, so-called “space-filling” designs were proposed.
- These involve a distance metric $\delta(\mathbf{x}_i, \mathbf{x}_j)$ with the properties

$$\delta(\mathbf{x}_i, \mathbf{x}_j) = \delta(\mathbf{x}_j, \mathbf{x}_i)$$

$$\delta(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \text{ with equality iff } \mathbf{x}_i = \mathbf{x}_j$$

$$\delta(\mathbf{x}_i, \mathbf{x}_j) \leq \delta(\mathbf{x}_i) + \delta(\mathbf{x}_j).$$

Minimax Distance Designs

- Consider n-run designs $\mathbf{D}$ selected from a finite discretization $\mathcal{D}$ of χ.

Minimax Distance Designs

- Consider n-run designs $\mathbf{D}$ selected from a finite discretization $\mathcal{D}$ of χ.
- $\mathbf{D}^*$ is a minimax distance design if

$$\min_{\mathbf{D}} \max_{\mathbf{x}\in\mathcal{D}} \delta(\mathbf{x}, \mathbf{D}) = \max_{\mathbf{x}\in\mathcal{D}} \delta(\mathbf{x}, \mathbf{D}^*) \equiv \delta^*$$

where $\delta(\mathbf{x}, \mathbf{D}) = \min_{\mathbf{x}'\in\mathcal{D}} \delta(\mathbf{x}, \mathbf{x}')$.

Minimax Distance Designs

- Consider n-run designs $\mathbf{D}$ selected from a finite discretization $\mathcal{D}$ of χ.
- $\mathbf{D}^*$ is a minimax distance design if

$$\min_{\mathbf{D}} \max_{\mathbf{x} \in \mathcal{D}} \delta(\mathbf{x}, \mathbf{D}) = \max_{\mathbf{x} \in \mathcal{D}} \delta(\mathbf{x}, \mathbf{D}^*) \equiv \delta^*$$

  where $\delta(\mathbf{x}, \mathbf{D}) = \min_{\mathbf{x}' \in \mathcal{D}} \delta(\mathbf{x}, \mathbf{x}')$.
- Idea: cover the design space at n points with spheres of minimum radius – ensures design points are never too far away from points *not* in the design.

Maximin Distance Designs

- $\mathbf{D}^*$ is a maximin distance design if

$$\max_{\mathbf{D}} \min_{\mathbf{x},\mathbf{x}' \in \mathbf{D}} \delta(\mathbf{x}, \mathbf{x}') = \min_{\mathbf{x},\mathbf{x}' \in \mathbf{D}^*} \delta(\mathbf{x}, \mathbf{x}') \equiv \delta^*.$$

Maximin Distance Designs

- $\mathbf{D}^*$ is a maximin distance design if

$$\max_{\mathbf{D}} \min_{\mathbf{x},\mathbf{x}' \in \mathbf{D}} \delta(\mathbf{x}, \mathbf{x}') = \min_{\mathbf{x},\mathbf{x}' \in \mathbf{D}^*} \delta(\mathbf{x}, \mathbf{x}') \equiv \delta^*.$$

- Idea: cover the deisgn space at n points with spheres of maximum radius – ensures no two design points are too close to one another, so each one has a larger area of "coverage".

Maximin Distance Designs

- $\mathbf{D}^*$ is a maximin distance design if

$$\max_{\mathbf{D}} \min_{\mathbf{x},\mathbf{x}' \in \mathbf{D}} \delta(\mathbf{x}, \mathbf{x}') = \min_{\mathbf{x},\mathbf{x}' \in \mathbf{D}^*} \delta(\mathbf{x}, \mathbf{x}') \equiv \delta^*.$$

- Idea: cover the deisgn space at *n* points with spheres of maximum radius – ensures no two design points are too close to one another, so each one has a larger area of "coverage".
- Generally preferred from a computational perspective since it only involves distances amongst points in the design rather than distances between design and non-design points as in minimax.

Minimax/Maximin Distance Designs

- Johnson et al.† relate these distance-based criteria with model-based critera for GP models when the correlation goes to zero - i.e. the response behaves like it is independent at far-way input settings.

† Johnson, Moore and Ylvisaker: *Minimax and Maximin Distance Desgins*, Journal of Statistical Planning and Inference, vol. 26, pp. 131–148 (1990).

Minimax/Maximin Distance Designs

- Johnson et al.† relate these distance-based criteria with model-based critera for GP models when the correlation goes to zero - i.e. the response behaves like it is independent at far-way input settings.
- The connection is interesting but beyond our scope.

† Johnson, Moore and Ylvisaker: *Minimax and Maximin Distance Desgins*, Journal of Statistical Planning and Inference, vol. 26, pp. 131–148 (1990).

Minimax/Maximin Distance Designs

- Johnson et al.† relate these distance-based criteria with model-based critera for GP models when the correlation goes to zero - i.e. the response behaves like it is independent at far-way input settings.
- The connection is interesting but beyond our scope.
- The idea is that in initial phases of data collection, our relatively few input settings where we will collect data will be remote from one another and this construction mimics this behaviour and gives us a criterion to optimize in selecting such input settings.

† Johnson, Moore and Ylvisaker: *Minimax and Maximin Distance Designs*, Journal of Statistical Planning and Inference, vol. 26, pp. 131–148 (1990).

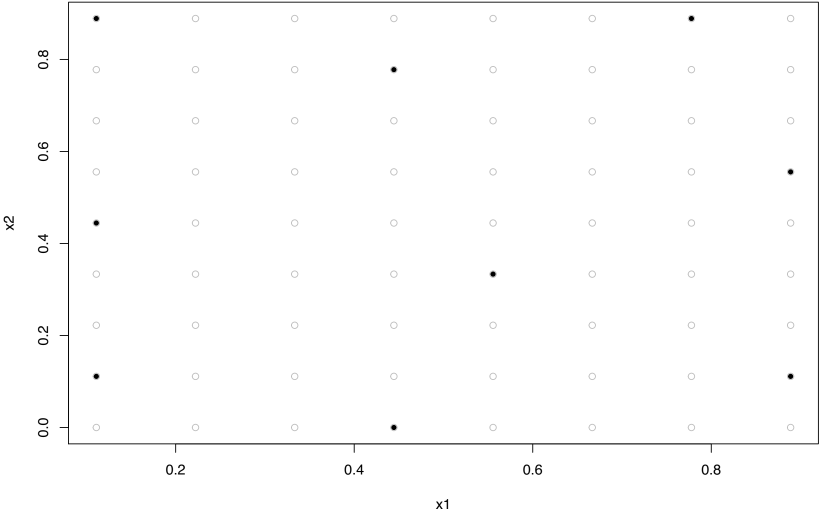
Example - Minimax Distance Design

```
library(fields)
cands=as.matrix(expand.grid(seq(0,1,length=10),seq(0,1,leng
nd=9
design=cover.design(cands,nd,nruns=10)$design
```

```
## Warning in cover.design(cands, nd, nruns = 10): Number c
## (nn) reduced to the actual number of candidates
```

```
plot(design,pch=20,xlab="x1",ylab="x2")
points(cands,col="grey")
```

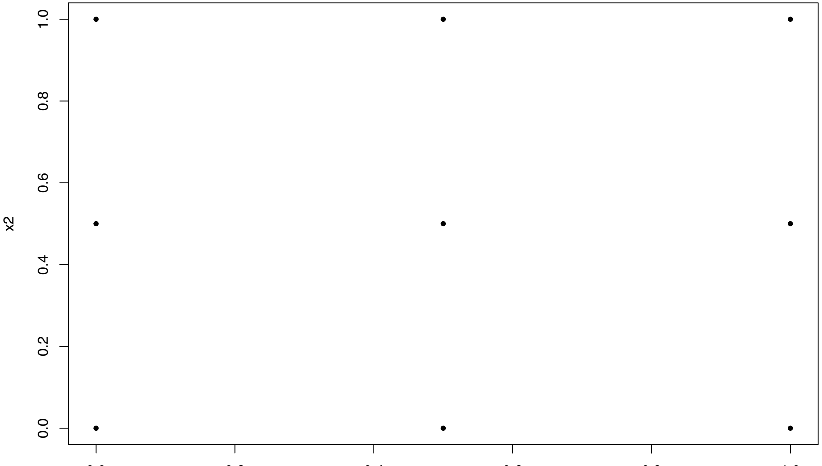
Example - Minimax Distance Design



Grids

Why not just a grid of points?

```
design=as.matrix(expand.grid(seq(0,1,length=3),seq(0,1,leng
plot(design,pch=20,xlab="x1",ylab="x2")
```



Grids

- Grids appear space-filling.

Grids

- Grids appear space-filling.
- But # of points grows exponentially with dimension:

$$p = 5, n = 5 \Rightarrow 5^5 = 625 \text{ points!}$$

Grids

- Grids appear space-filling.
- But # of points grows exponentially with dimension:

$$p = 5, n = 5 \Rightarrow 5^5 = 625 \text{ points!}$$

- Lower-dimensional projections are also poor.

Grids

- Grids appear space-filling.
- But # of points grows exponentially with dimension:

$$p = 5, n = 5 \Rightarrow 5^5 = 625 \text{ points!}$$

- Lower-dimensional projections are also poor.
- So we would like space-fillingness *and* non-collapsingness.

Latin Hypercube Designs (LHS)

- In a Latin Hypercube Design, the p input axes are stratified into n partitions:

$$[0, \frac{1}{n}), \ldots, [\frac{n-1}{n}, 1]$$

Latin Hypercube Designs (LHS)

- In a Latin Hypercube Design, the p input axes are stratified into n partitions:

$$[0, \frac{1}{n}), \ldots, [\frac{n-1}{n}, 1]$$

- The n design points are selected as

$$x_i^j = (\pi^j(i) - 0.5)/n$$

for $j = 1, \ldots, p$ and $i = 1, \ldots, n$ where $\pi^j(i)$ are independent random permutations of the integers $1, \ldots, n$.

Latin Hypercube Designs (LHS)

- In a Latin Hypercube Design, the p input axes are stratified into n partitions:

$$[0, \frac{1}{n}), \ldots, [\frac{n-1}{n}, 1]$$

- The n design points are selected as

$$x_i^j = (\pi^j(i) - 0.5)/n$$

  for $j = 1, \ldots, p$ and $i = 1, \ldots, n$ where $\pi^j(i)$ are independent random permutations of the integers $1, \ldots, n$.

- Idea is to place the integers $1, 2, \ldots, n$ into cells defined by the partitions so that each integer appears exactly once in each of the strata for the p dimensions.

Latin Hypercube Designs (LHS)

- In a Latin Hypercube Design, the p input axes are stratified into n partitions:

$$[0, \frac{1}{n}), \ldots, [\frac{n-1}{n}, 1]$$

- The n design points are selected as

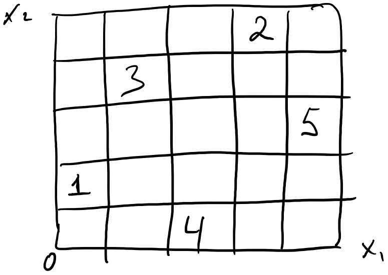
$$x_i^j = (\pi^j(i) - 0.5)/n$$

  for $j = 1, \ldots, p$ and $i = 1, \ldots, n$ where $\pi^j(i)$ are independent random permutations of the integers $1, \ldots, n$.
- Idea is to place the integers $1, 2, \ldots, n$ into cells defined by the partitions so that each integer appears exactly once in each of the strata for the p dimensions.
- So in 2D, LHS designs have the property that each row/column of the design has only 1 design point.

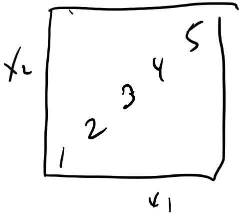
LHS Example

$n=5$

$$\pi^1(1) = 1 \qquad \pi^2(1) = 2$$
$$\pi^1(2) = 4 \qquad \pi^2(2) = 5$$
$$\pi^1(3) = 2 \qquad \pi^2(3) = 4$$
$$\pi^1(4) = 3 \qquad \pi^2(4) = 1$$
$$\pi^1(5) = 5 \qquad \pi^2(5) = 3$$



- Gives design settings for $p = 1$ as $X^1 = \left(\frac{1-0.5}{5}, \frac{4-0.5}{5}, \ldots\right) = (0.1, 0.7, 0.3, 0.5, 0.9)$ and for $p = 2$ as $X^2 = (0.3, 0.9, 0.7, 0.1, 0.5)$



LHS Example

$$\begin{array}{ll}
\pi^1(1) = 1 & \pi^2(1) = 2 \\
\pi^1(2) = 4 & \pi^2(2) = 5 \\
\pi^1(3) = 2 & \pi^2(3) = 4 \\
\pi^1(4) = 3 & \pi^2(4) = 1 \\
\pi^1(5) = 5 & \pi^2(5) = 3
\end{array}$$

- Gives design settings for $p = 1$ as $X^1 = \left(\frac{1-0.5}{5}, \frac{4-0.5}{5}, \ldots\right) = (0.1, 0.7, 0.3, 0.5, 0.9)$ and for $p = 2$ as $X^2 = (0.3, 0.9, 0.7, 0.1, 0.5)$
- Overall design given by $X = \left[{X^1}^T, {X^2}^T\right]$

Latin Hypercube Designs

- McKay et al. (1979)† showed for a function of the form

$$Y = h(X_1, \ldots, X_k)$$

monotonic in each X_j and a monotonic transformation of Y given by $g(Y)$ then for estimators of the form

$$T(Y) = \sum_{i=1}^{n} g(Y_i),$$

the variance of the estimator using LHS is reduced compared to simple random sampling and stratified sampling.

† McKay, Conover and Beckman: *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, vol.21, pp.239–245 (1979).

Latin Hypercube Designs

- Stein (1987)† showed that if a function $f(\mathbf{x})$ satisfies $\int f(\mathbf{x})^2 < \infty$ and has the form

$$f(\mathbf{x}) = f_0 + \sum_{j=1}^{p} f_j(\mathbf{x}) + e(\mathbf{x})$$

where $f_0 = \int f(\mathbf{x}) d\mathbf{x}$ and $f_j(\mathbf{x}) = \int (f(\mathbf{x}) - \mu) d\mathbf{x}_{-j}$ then

$$\begin{aligned} \mathrm{Var}_{LHS}\left(\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)\right) &= \frac{1}{n}\int e(\mathbf{x})^2 + o\left(\frac{1}{n}\right) \\ &< \frac{1}{n}\int e(\mathbf{x})^2 d\mathbf{x} + \frac{1}{n}\sum_{j=1}^{p}\int f_j(\mathbf{x})^2 d\mathbf{x} \\ &= \mathrm{Var}_{iid} \end{aligned}$$

† Stein: *Large sample properties of simulations using Latin hypercube sampling*, Technometrics, vol.29, pp.143–151 (1987)

Latin Hypercube Designs

- On the other hand, it's possible to get a bad LHS, such as the points along the diagonal.

Latin Hypercube Designs

- On the other hand, it's possible to get a bad LHS, such as the points along the diagonal.
- So typically LHS is combined with another criterion that enforces space-fillingness.

Latin Hypercube Designs

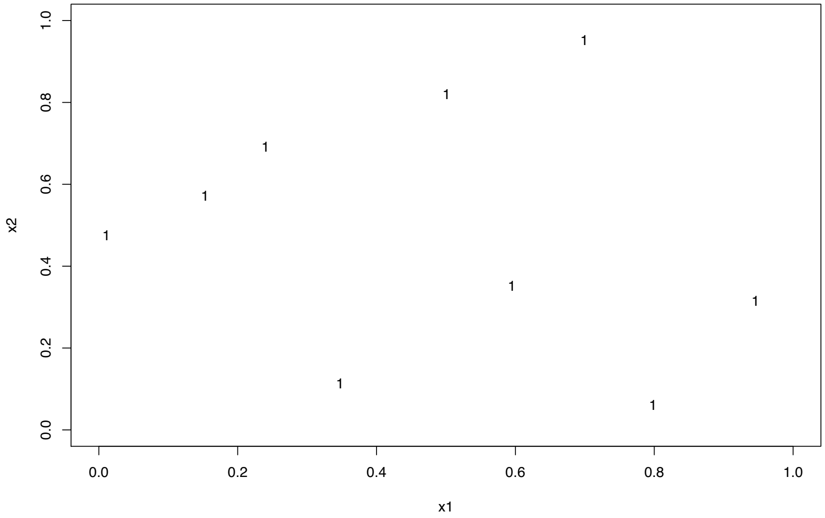
- On the other hand, it's possible to get a bad LHS, such as the points along the diagonal.
- So typically LHS is combined with another criterion that enforces space-fillingness.
  - e.g. among the LHS designs of size n, choose the LHS that is best from a minimax distance perspective.

LHS Example

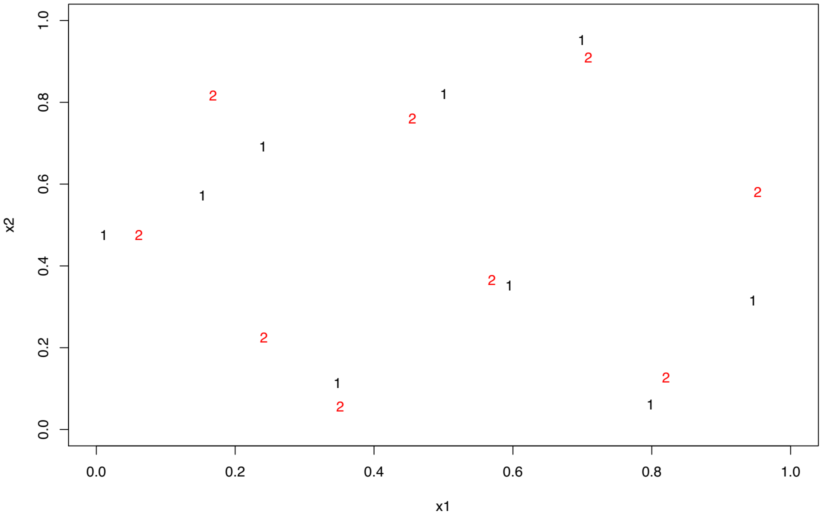
- R package lhs offers a few implementations.

```
library(lhs)
set.seed(66) # only to replicate this output
n=9
p=2
design1=randomLHS(n,p)  # default algorithm
set.seed(66)
design2=optimumLHS(n,p) # maximize mean distance
                        # between design points
set.seed(66)
design3=maximinLHS(n,p) # maximize the min distance
                        # between design points
```

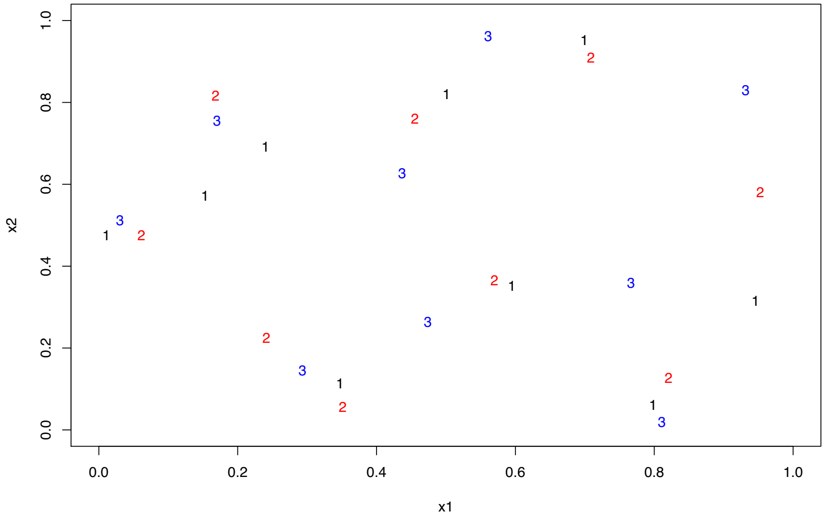
LHS Example



LHS Example



LHS Example



Sequential Designs

- Often we would like to perform a small initial design and then based on the data observed sequentially collect more data to refine our estimate of interest.

† Jones, Schonlau and Welch: *Efficient Global Optimization of Expensive Black Box Functions*, Journal of Global Optimization, vol. 13, pp.455–492 (1998).

Sequential Designs

- Often we would like to perform a small initial design and then based on the data observed sequentially collect more data to refine our estimate of interest.
- For example, in uncertainty quantification the goal is often to optimize a complicated response by use of our statistical GP emulator.

† Jones, Schonlau and Welch: *Efficient Global Optimization of Expensive Black Box Functions*, Journal of Global Optimization, vol. 13, pp.455–492 (1998).

Sequential Designs

- Often we would like to perform a small initial design and then based on the data observed sequentially collect more data to refine our estimate of interest.
- For example, in uncertainty quantification the goal is often to optimize a complicated response by use of our statistical GP emulator.
    - a natural sequential design setup in this case is to select points that increasingly refine our estimate of the optimum.

† Jones, Schonlau and Welch: *Efficient Global Optimization of Expensive Black Box Functions*, Journal of Global Optimization, vol. 13, pp.455–492 (1998).

Sequential Designs

- Often we would like to perform a small initial design and then based on the data observed sequentially collect more data to refine our estimate of interest.
- For example, in uncertainty quantification the goal is often to optimize a complicated response by use of our statistical GP emulator.
  - a natural sequential design setup in this case is to select points that increasingly refine our estimate of the optimum.
- A popular approach is the expected improvement method of Jones et al.†

† Jones, Schonlau and Welch: *Efficient Global Optimization of Expensive Black Box Functions*, Journal of Global Optimization, vol. 13, pp.455–492 (1998).

Expected Improvement

- Start with an initial n-run single-shot experiment (say, space-filling).

Expected Improvement

- Start with an initial n-run single-shot experiment (say, space-filling).
- Let $f_{min} = \min(y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n))$ and define the improvement function to be

$$I(\mathbf{x}) = \max(f_{min} - Y(\mathbf{x}), 0).$$

Expected Improvement

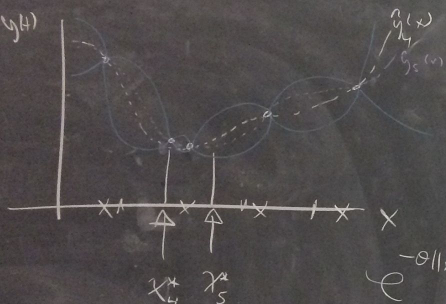
- Start with an initial n-run single-shot experiment (say, space-filling).
- Let $f_{min} = \min(y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n))$ and define the improvement function to be

$$I(\mathbf{x}) = \max(f_{min} - Y(\mathbf{x}), 0).$$

- Note that $I(\mathbf{x})$ is a random variable, so one might try to look at the expected improvement as the optimality criteria,

$$\begin{aligned} E[I(\mathbf{x})] &= E[max(f_{min} - Y(\mathbf{x}), 0)] \\ &= (f_{min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + \hat{s}(\mathbf{x})\phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) \end{aligned}$$

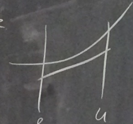
where Φ denotes the standard Normal c.d.f. and ϕ denotes the standard Normal p.d.f.



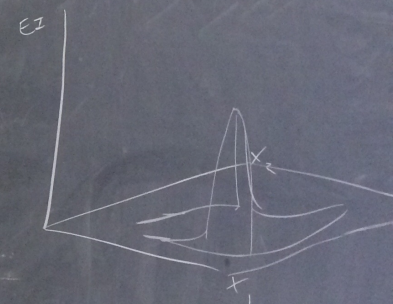
$$E[y(x) \mid y_1, \ldots, y_n] = \hat{y}_n(x) = r^T R^{-1} \underline{y}$$

$$\hat{s}_n^2(x) = \sigma^2 (1 - r^T R^{-1} r)$$

$$e^{-\theta \|x - x\|^2}$$



$$\sim N(\hat{y}_n(x), \hat{s}_n^2(x))$$



Expected Improvement

- Want to sequentially select a new design point, $\mathbf{x}^*$ that maximizes the expected improvement. So what does EI do?

Expected Improvement

- Want to sequentially select a new design point, $\mathbf{x}^*$ that maximizes the expected improvement. So what does EI do?
- It turns out:

$$\frac{\partial E[\cancel{(}I(\mathbf{x}))]}{\partial \hat{y}} = -\Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) < 0$$

$$\frac{\partial E[\cancel{(}I(\mathbf{x}))]}{\partial \hat{s}} = \phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) > 0$$

Expected Improvement

- Want to sequentially select a new design point, $\mathbf{x}^*$ that maximizes the expected improvement. So what does EI do?
- It turns out:

$$\frac{\partial E[I(\mathbf{x}))]}{\partial \hat{y}} = -\Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) < 0$$

$$\frac{\partial E[I(\mathbf{x}))]}{\partial \hat{s}} = \phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) > 0$$

- So we can interpret this as meaning the expected improvement increases as $\hat{y}$ decreases and it also increases as $\hat{s}$ increases.

Expected Improvement

- Want to sequentially select a new design point, $\mathbf{x}^*$ that maximizes the expected improvement. So what does EI do?
- It turns out:

$$\frac{\partial E[I(I(\mathbf{x}))]}{\partial \hat{y}} = -\Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) < 0$$

$$\frac{\partial E[I(I(\mathbf{x}))]}{\partial \hat{s}} = \phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) > 0$$

- So we can interpret this as meaning the expected improvement increases as $\hat{y}$ decreases and it also increases as $\hat{s}$ increases.
- EI trades-off between choosing a sequential design point that further reduces the minimum value f_{min} or reduces the uncertainty of the response surface.

Example: EI on Branin Test Function

We'll look at applying EI to the Branin test function – see
`https://www.sfu.ca/~ssurjano/branin.html`

```
library(DiceOptim)
library(rgl)

# get our initial starting design
#set.seed(7)
#design=optimumLHS(9,2)
design=as.matrix(expand.grid(seq(0,1,length=3),seq(0,1,leng
colnames(design)=c("x1","x2")
y.branin=apply(design,1,branin)
```

Example: EI on Branin Test Function

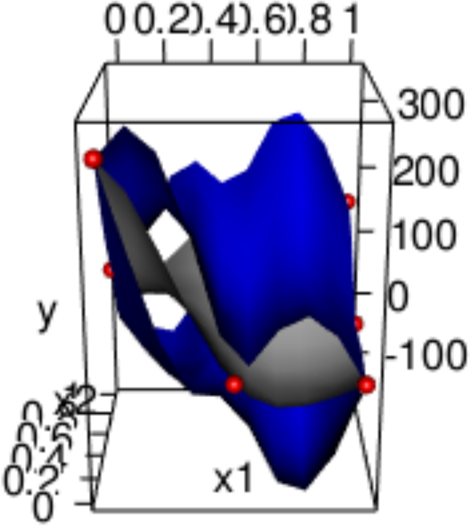
```
# Fit the GP model - built into the DiceOptim package
fit.gp=km(~1, design = data.frame(x = design),
          response = y.branin,covtype = "gauss")
```

```
##
## optimisation start
## -------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~1
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10
##   - parameters upper bounds :  2 2
##   - best initial criterion value(s) :  -53.33026
```

Example: EI on Branin Test Function

```
persp3d(seq(0,1,length=10),seq(0,1,length=10),
        matrix(yhat$mean,10,10),col="grey",
        xlab="x1",ylab="x2",zlab="y")
persp3d(seq(0,1,length=10),seq(0,1,length=10),
        matrix(yhat$lower95),col="blue",add=TRUE)
persp3d(seq(0,1,length=10),seq(0,1,length=10),
        matrix(yhat$upper95),col="blue",add=TRUE)
plot3d(design[,1],design[,2],y.branin,type="s",
        radius=7,col="red",add=TRUE)
```

Example: EI on Branin Test Function



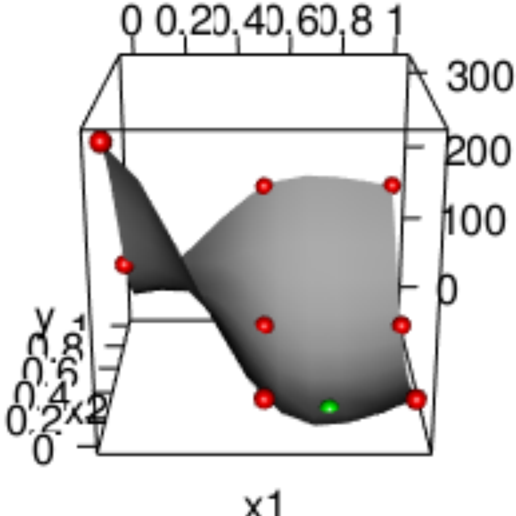
Example: EI on Branin Test Function

```r
# Calculate EI
ego=apply(as.matrix(X),1,EI,fit.gp,type="UK",
          minimization=TRUE)
persp3d(seq(0,1,length=10),seq(0,1,length=10),
        matrix(ego,10,10),col="grey",
        xlab="x1",ylab="x2",zlab="y")


# Get next x that maximizes the EI
x.new=max_EI(fit.gp,lower=rep(0,2),upper=rep(1,2),
             parinit = 0.5,minimization=TRUE)$par
yhat.xnew=predict(fit.gp,newdata=x.new,type="UK")$mean
```

Example: EI on Branin Test Function

```
## Warning in genoud(EI, nvars = d, max = TRUE, pop.size =
## Ignoring 'starting.values' because length(staring.values

##
##
## Mon Sep 25 13:23:43 2017
## Domains:
##  0.000000e+00   <=  X1   <=    1.000000e+00
##  0.000000e+00   <=  X2   <=    1.000000e+00
##
## Data Type: Floating Point
## Operators (code number, name, population)
##  (1) Cloning...........................  2
##  (2) Uniform Mutation..................  1
##  (3) Boundary Mutation.................  1
##  (4) Non-Uniform Mutation..............  1
##  (5) Polytope Crossover................  1
```

Example: EI on Branin Test Function

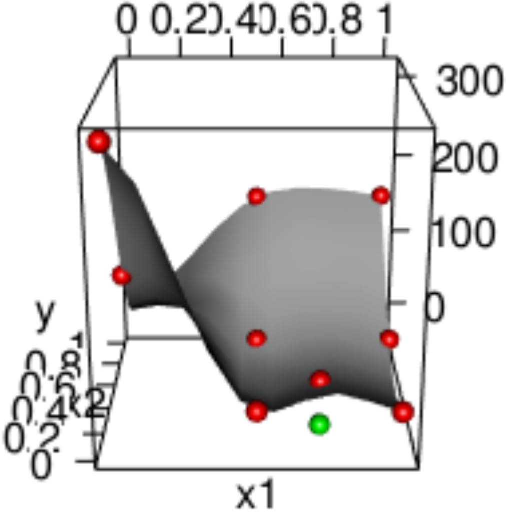


Example: EI on Branin Test Function

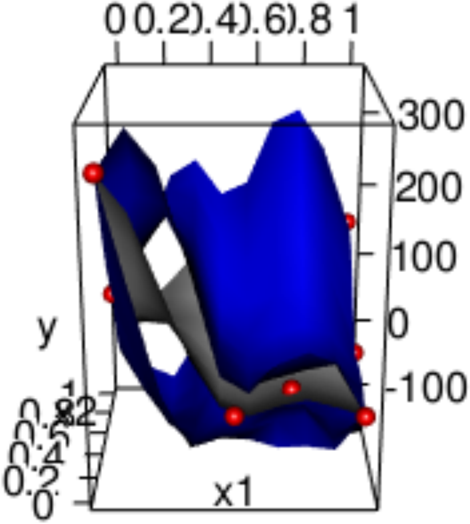
```
# Update by evaluating our expensive function
y.new=apply(x.new,1,branin)
y.branin=c(y.branin,y.new)
design=rbind(design,x.new)

# Refit the GP model
fit.gp=km(~1, design = data.frame(x = design),
          response = y.branin, covtype = "gauss")
```

```
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~1
## * covariance model :
```

Example: EI on Branin Test Function



Example: EI on Branin Test Function

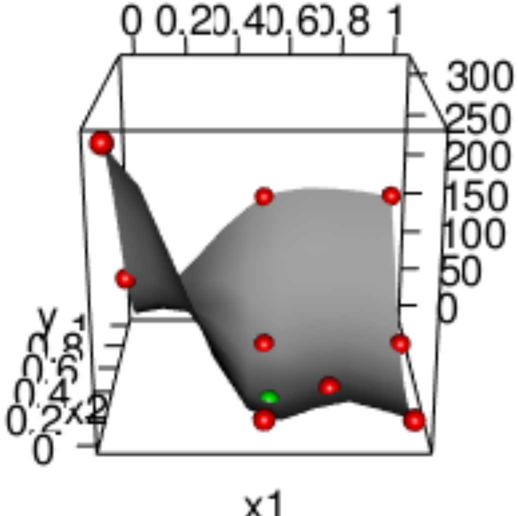


Example: EI on Branin Test Function

```
# Calculate EI
ego=apply(as.matrix(X),1,EI,fit.gp,type="UK",
          minimization=TRUE)
persp3d(seq(0,1,length=10),seq(0,1,length=10),
        matrix(ego,10,10),col="grey",
        xlab="x1",ylab="x2",zlab="y")

# Get next x that maximizes the EI
x.new=max_EI(fit.gp,lower=rep(0,2),upper=rep(1,2),
             parinit = 0.5,minimization=TRUE)$par
```

```
## Warning in genoud(EI, nvars = d, max = TRUE, pop.size =
## Ignoring 'starting.values' because length(staring.values

##
##
## Mon Sep 25 13:23:43 2017
```

Example: EI on Branin Test Function

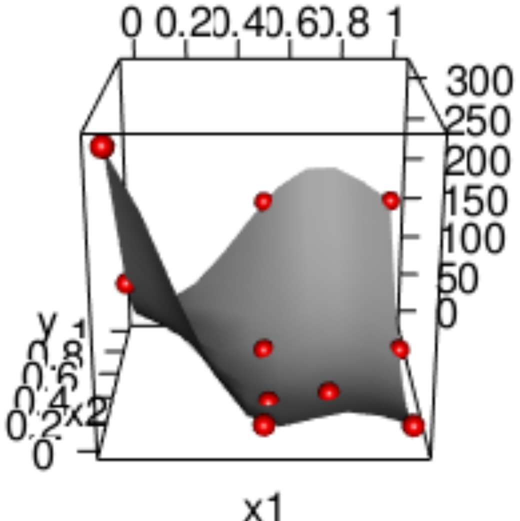


Example: EI on Branin Test Function

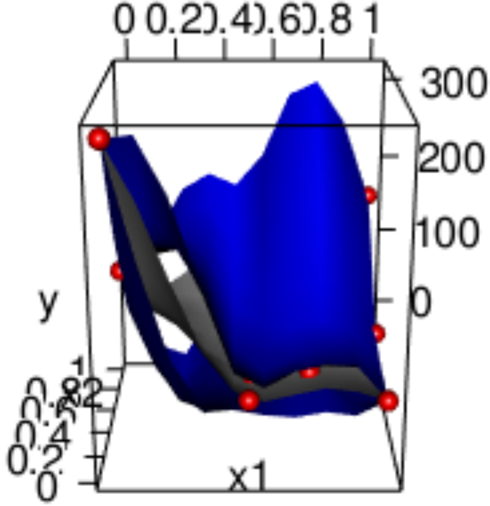
```
# Update by evaluating our expensive function
y.new=apply(x.new,1,branin)
y.branin=c(y.branin,y.new)
design=rbind(design,x.new)

# Refit the GP model
fit.gp=km(~1, design = data.frame(x = design),
          response = y.branin,covtype = "gauss")
```

```
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~1
## * covariance model :
```

Example: EI on Branin Test Function



Example: EI on Branin Test Function

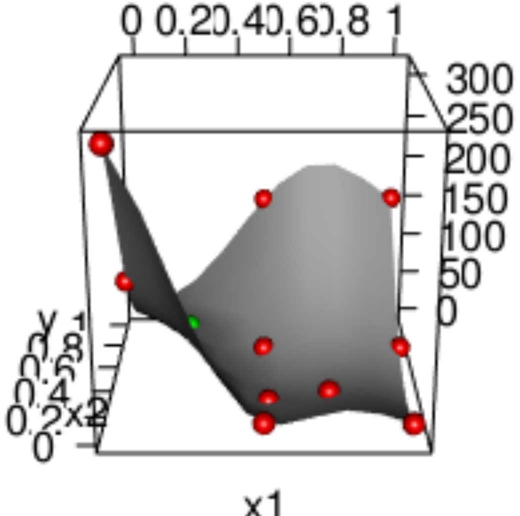


Example: EI on Branin Test Function

```
# Calculate EI
ego=apply(as.matrix(X),1,EI,fit.gp,type="UK",
          minimization=TRUE)
persp3d(seq(0,1,length=10),seq(0,1,length=10),
        matrix(ego,10,10),col="grey",
        xlab="x1",ylab="x2",zlab="y")

# Get next x that maximizes the EI
x.new=max_EI(fit.gp,lower=rep(0,2),upper=rep(1,2),
             parinit = 0.5,minimization=TRUE)$par
```

```
## Warning in genoud(EI, nvars = d, max = TRUE, pop.size =
## Ignoring 'starting.values' because length(staring.value

##
##
## Mon Sep 25 13:23:44 2017
```

Example: EI on Branin Test Function



Global Sensitivity Analysis (SA)

- A variance-based decomposition of your (unknown) function f.

Global Sensitivity Analysis (SA)

- A variance-based decomposition of your (unknown) function f.
- Total variation of model output is decomposed into terms of increasing dimensionality. Think “functional ANOVA”.

Global Sensitivity Analysis (SA)

- A variance-based decomposition of your (unknown) function f.
- Total variation of model output is decomposed into terms of increasing dimensionality. Think “functional ANOVA”.
- Three main scenarios:

Global Sensitivity Analysis (SA)

- A variance-based decomposition of your (unknown) function f.
- Total variation of model output is decomposed into terms of increasing dimensionality. Think "functional ANOVA".
- Three main scenarios:
  - Factor screening - identify the influential factors in a system with many factors.

Global Sensitivity Analysis (SA)

- A variance-based decomposition of your (unknown) function f.
- Total variation of model output is decomposed into terms of increasing dimensionality. Think "functional ANOVA".
- Three main scenarios:
  - Factor screening - identify the influential factors in a system with many factors.
  - Attribute output uncertainty to uncertainty in input factors.

Global Sensitivity Analysis (SA)

- A variance-based decomposition of your (unknown) function f.
- Total variation of model output is decomposed into terms of increasing dimensionality. Think "functional ANOVA".
- Three main scenarios:
  - Factor screening - identify the influential factors in a system with many factors.
  - Attribute output uncertainty to uncertainty in input factors.
- There is also a local SA where the emphasis is on local impact of factors on the response. Think derivatives.

Sensitivity Analysis

- Assume the input space, $\chi \in \mathbb{R}^k$ is a k-dimensional unit hypercube.

† Sobol': *On sensitivity estimation for nonlinear mathematical models*, Matematicheskoe Modelirovanie, 2.1, pp.112–118 (1990).

Sensitivity Analysis

- Assume the input space, $\chi \in \mathbb{R}^k$ is a k-dimensional unit hypercube.
- The Sobol'† decomposition of $f(\mathbf{x}), \quad \mathbf{x} \in \chi$ is

$$
\begin{aligned}
f(x_1, \ldots, x_k) &= f_0 + \sum_{i=1}^{k} f_i(x_i) + \sum_{1 \le i < j \le k} f_{ij}(x_i, x_j) + \ldots \quad (1) \\
&+ f_{1,2,\ldots,k}(x_1, \ldots, x_k)
\end{aligned}
$$

† Sobol': *On sensitivity estimation for nonlinear mathematical models*, Matematicheskoe Modelirovanie, 2.1, pp.112–118 (1990).

Sensitivity Analysis

- Assume the input space, $\chi \in \mathbb{R}^k$ is a k-dimensional unit hypercube.
- The Sobol'† decomposition of $f(\mathbf{x}), \quad \mathbf{x} \in \chi$ is

$$\begin{aligned} f(x_1, \ldots, x_k) &= f_0 + \sum_{i=1}^{k} f_i(x_i) + \sum_{1 \le i < j \le k} f_{ij}(x_i, x_j) + \ldots (1) \\ &+ f_{1,2,\ldots,k}(x_1, \ldots, x_k) \end{aligned}$$

- For this decomposition to hold, f_0 must be a constant and

$$\int_0^1 f_{i_1,\ldots,i_s}(x_{i_1}, \ldots, x_{i_s}) dx_{i_j} = 0 \text{ for } 1 \le j \le s. \qquad (2)$$

† Sobol': *On sensitivity estimation for nonlinear mathematical models*, Matematicheskoe Modelirovanie, 2.1, pp.112–118 (1990).

Sensitivity Analysis

- A consequence of the constraint (2) is that all summands in (1) are orthogonal, e.g.,

$$\int_{\chi} f_{i_1,\ldots,i_s} f_{j_1,\ldots,j_l} d\mathbf{x} = 0 \quad \text{if } (i_1,\ldots,i_s) \neq (j_1,\ldots,j_l).$$

Sensitivity Analysis

- A consequence of the constraint (2) is that all summands in (1) are orthogonal, e.g.,

$$\int_{\chi} f_{i_1,\ldots,i_s} f_{j_1,\ldots,j_l} d\mathbf{x} = 0 \quad \text{if } (i_1, \ldots, i_s) \neq (j_1, \ldots, j_l).$$

- This is because at least one of the indices in $(i_1, \ldots, i_s)$ and $(j_i, \ldots, j_l)$ will not be repeated in both sets of indices, and so the integral vanishes by (2).

Sensitivity Analysis

- Another consequence is that

$$f_0 = \int_{\chi} f(\mathbf{x}) d\mathbf{x}.$$

† Sobol: *Sensitivity estimates for nonlinear mathematical models*, Mathematical Modelling and Computational Experiments, 1.4, pp.407–414 (1993).

Sensitivity Analysis

- Another consequence is that

$$f_0 = \int_{\chi} f(\mathbf{x}) d\mathbf{x}.$$

- Sobol'† showed the decomposition (1) is unique and all the terms can be calculated as

$$f_i(x_i) = -f_0 + \int_0^1 \cdots \int_0^1 f(\mathbf{x}) d\mathbf{x}_{-i}$$

$$f_{ij}(x_i, x_j) = -f_0 - f_i(x_i) - f_j(x_j) + \int_0^1 \cdots \int_0^1 f(\mathbf{x}) d\mathbf{x}_{-(i,j)}$$

and so on.

† Sobol: *Sensitivity estimates for nonlinear mathematical models*, Mathematical Modelling and Computational Experiments, 1.4, pp.407–414 (1993).

Sensitivity Analysis

- Sobol' then defines the *total variance* of $f(\mathbf{x})$ to be

$$\begin{aligned} D &= \int_{\chi} f^2(\mathbf{x}) - f_0^2 \\ &= E\left[f(\mathbf{x})^2\right] - E\left[f(\mathbf{x})\right]^2 \\ &= \mathrm{Var}(f(\mathbf{x})) \end{aligned}$$

where $E[\cdot]$ is taken with respect to a density $\pi(\mathbf{x})$. Usually this is taken to be Uniform on χ.

Sensitivity Analysis

- Similarly, the *partial variances* are

$$D_{i_1,\ldots,i_s} = \int_0^1 \cdots \int_0^1 f^2_{i_1,\ldots,i_s}(x_{i_1},\ldots,x_{i_s})dx_{i_1}\cdots dx_{i_s}$$

where $1 \leq i_1 < \cdots < i_s \leq k$ and $s = 1,\ldots,k$.

Sensitivity Analysis

- Similarly, the *partial variances* are

$$D_{i_1,\ldots,i_s} = \int_0^1 \cdots \int_0^1 f_{i_1,\ldots,i_s}^2(x_{i_1},\ldots,x_{i_s})dx_{i_1}\cdots dx_{i_s}$$

where $1 \leq i_1 < \cdots < i_s \leq k$ and $s = 1,\ldots,k$.

- For example,

$$\begin{aligned}
D_1 &= \int_0^1 f_1^2(x_1)dx_1 \\
&= E\left[f_1^2(x_1)\right] \\
&= E\left[\left(\int \cdots \int f(\mathbf{x})d\mathbf{x}_{-1} - f_0\right)^2\right] \\
&= \mathrm{Var}_{X_1}\left(E\left[f(\mathbf{x})|X_1 = x_1\right]\right)
\end{aligned}$$

Sensitivity Analysis

- In all we have

$$D = \sum_{i=1}^{k} D_i + \sum_{1\le i<j\le k} D_{ij} + \ldots + D_{1,2,\ldots,k}$$

and

$$\begin{aligned}
\mathrm{Var}(f) &= \sum_i \mathrm{Var}_{X_i}\left(E\left[f(\mathbf{x})|X_i = x_i\right]\right) \\
&+ \sum_{1\le i<j\le k} \mathrm{Var}_{X_i,X_j}\left(E\left[f(\mathbf{x})|X_i = x_i, X_j = x_j\right]\right) \\
&+ \ldots + \mathrm{Var}_{X_1,\ldots,X_k}\left(E\left[f(x)|X_1 = x_1, \ldots, X_k = x_k\right]\right)
\end{aligned}$$

where the last term is zero.

Sensitivity Analysis

- The *sensitivity indices* are given by

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}$$

for $1 \leq i_1 < \cdots < i_s \leq k$.

Sensitivity Analysis

- The *sensitivity indices* are given by

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}$$

  for $1 \le i_1 < \cdots < i_s \le k$.
- S_i is called the *first-order sensitivity index* for factor X_i, which measures the main effect of X_i on the output.

Sensitivity Analysis

- The *sensitivity indices* are given by

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}$$

  for $1 \leq i_1 < \cdots < i_s \leq k$.
- S_i is called the *first-order sensitivity index* for factor X_i, which measures the main effect of X_i on the output.
  - i.e. the fractional contribution of X_i to the overall variance of $f(\mathbf{x})$.

Sensitivity Analysis

- The *sensitivity indices* are given by

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}$$

  for $1 \leq i_1 < \cdots < i_s \leq k$.
- S_i is called the *first-order sensitivity index* for factor X_i, which measures the main effect of X_i on the output.
  - i.e. the fractional contribution of X_i to the overall variance of $f(\mathbf{x})$.
- $S_{ij}, \quad i \neq j$ is the *second-order sensitivity index* which measures the interaction effect.

Sensitivity Analysis

- The *sensitivity indices* are given by

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}$$

  for $1 \le i_1 < \cdots < i_s \le k$.
- S_i is called the *first-order sensitivity index* for factor X_i, which measures the main effect of X_i on the output.
  - i.e. the fractional contribution of X_i to the overall variance of $f(\mathbf{x})$.
- $S_{ij}, \quad i \neq j$ is the *second-order sensitivity index* which measures the interaction effect.
  - i.e. the part of the variation in $f(\mathbf{x})$ due to X_i and X_j that cannot be explained by the sum of the individual first-order effects of X_i and X_j.

Sensitivity Analysis

- The *sensitivity indices* are given by

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}$$

  for $1 \leq i_1 < \cdots < i_s \leq k$.
- S_i is called the *first-order sensitivity index* for factor X_i, which measures the main effect of X_i on the output.
  - i.e. the fractional contribution of X_i to the overall variance of $f(\mathbf{x})$.
- $S_{ij}, \quad i \neq j$ is the *second-order sensitivity index* which measures the interaction effect.
  - i.e. the part of the variation in $f(\mathbf{x})$ due to X_i and X_j that cannot be explained by the sum of the individual first-order effects of X_i and X_j.
- etc.

Sensitivity Analysis

- The *sensitivity indices* are given by

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}$$

  for $1 \leq i_1 < \cdots < i_s \leq k$.
- S_i is called the *first-order sensitivity index* for factor X_i, which measures the main effect of X_i on the output.
  - i.e. the fractional contribution of X_i to the overall variance of $f(\mathbf{x})$.
- $S_{ij}, \quad i \neq j$ is the *second-order sensitivity index* which measures the interaction effect.
  - i.e. the part of the variation in $f(\mathbf{x})$ due to X_i and X_j that cannot be explained by the sum of the individual first-order effects of X_i and X_j.
- etc.
- Note that $\sum_{i=1}^{k} S_i + \sum_{1 \leq i < j \leq k} S_{ij} + \ldots + S_{1,2,\ldots,k} = 1$.

Sensitivity Analysis

- The sum of all sensitivity indices involving factor X_i is called the *total sensitivity index* for factor i, TS_i.

† Homma and Saltelli: *Importance measures in global sensitivity analysis of model output*, Reliability Engineering and Systems Safety, vol.52, pp.1–17 (1996).

Sensitivity Analysis

- The sum of all sensitivity indices involving factor X_i is called the *total sensitivity index* for factor i, TS_i.
    - e.g. For a model with 3 factors X_1, X_2 and X_3, then

$$TS_1 = S_1 + S_{12} + S_{13} + S_{123}.$$

† Homma and Saltelli: *Importance measures in global sensitivity analysis of model output*, Reliability Engineering and Systems Safety, vol.52, pp.1–17 (1996).

Sensitivity Analysis

- The sum of all sensitivity indices involving factor X_i is called the *total sensitivity index* for factor i, TS_i.
  - e.g. For a model with 3 factors X_1, X_2 and X_3, then

$$TS_1 = S_1 + S_{12} + S_{13} + S_{123}.$$

- Homma and Saltelli† show that by partitioning $\mathbf{X}$ into X_i and X_{-i}, the total sensitivity index TS_i can be computed as

$$TS_i = S_i + S_{i,(-i)} = 1 - S_{-i}$$

† Homma and Saltelli: *Importance measures in global sensitivity analysis of model output*, Reliability Engineering and Systems Safety, vol.52, pp.1–17 (1996).

Sensitivity Analysis

- The sum of all sensitivity indices involving factor X_i is called the *total sensitivity index* for factor i, TS_i.
    - e.g. For a model with 3 factors X_1, X_2 and X_3, then

$$TS_1 = S_1 + S_{12} + S_{13} + S_{123}.$$

- Homma and Saltelli† show that by partitioning $\mathbf{X}$ into X_i and X_{-i}, the total sensitivity index TS_i can be computed as

$$TS_i = S_i + S_{i,(-i)} = 1 - S_{-i}$$

    - e.g. $TS_1 = 1 - S_2 - S_3 - S_{23}$ in the above example.

† Homma and Saltelli: *Importance measures in global sensitivity analysis of model output*, Reliability Engineering and Systems Safety, vol.52, pp.1–17 (1996).

Sensitivity Analysis

- The sum of all sensitivity indices involving factor X_i is called the *total sensitivity index* for factor i, TS_i.
  - e.g. For a model with 3 factors X_1, X_2 and X_3, then

$$TS_1 = S_1 + S_{12} + S_{13} + S_{123}.$$

- Homma and Saltelli† show that by partitioning $\mathbf{X}$ into X_i and X_{-i}, the total sensitivity index TS_i can be computed as

$$TS_i = S_i + S_{i,(-i)} = 1 - S_{-i}$$

  - e.g. $TS_1 = 1 - S_2 - S_3 - S_{23}$ in the above example.
- This construction is computationally friendlier since it takes only one Monte Carlo integration (more on this in a moment).

† Homma and Saltelli: *Importance measures in global sensitivity analysis of model output*, Reliability Engineering and Systems Safety, vol.52, pp.1–17 (1996).

Sensitivity Analysis

- The sum of all sensitivity indices involving factor X_i is called the *total sensitivity index* for factor i, TS_i.
  - e.g. For a model with 3 factors X_1, X_2 and X_3, then

$$TS_1 = S_1 + S_{12} + S_{13} + S_{123}.$$

- Homma and Saltelli† show that by partitioning $\mathbf{X}$ into X_i and X_{-i}, the total sensitivity index TS_i can be computed as

$$TS_i = S_i + S_{i,(-i)} = 1 - S_{-i}$$

  - e.g. $TS_1 = 1 - S_2 - S_3 - S_{23}$ in the above example.
- This construction is computationally friendlier since it takes only one Monte Carlo integration (more on this in a moment).
- Here S_{-i} is the sum of all $S_{i_1,\ldots,i_s}$ terms that *do not* involve the index i.

† Homma and Saltelli: *Importance measures in global sensitivity analysis of model output*, Reliability Engineering and Systems Safety, vol.52, pp.1–17 (1996).

Sensitivity Analysis

- In other words,

$$TS_i = 1 - \frac{D_{-i}}{D} = \frac{\frac{E_{\mathbf{X}_{-i}}[\mathrm{Var}(f(\mathbf{x})|\mathbf{X}{-i})]}{\mathrm{Var}(f(\mathbf{x}))}}{D}$$

  where $\frac{D_{-i}}{D}$ is the total fractional variance *complement* to factor X_i.

Sensitivity Analysis

- In other words,

$$TS_i = 1 - \frac{D_{-i}}{D} = \frac{\frac{E_{\mathbf{X}_{-i}}[\mathrm{Var}(f(\mathbf{x})|\mathbf{X}-i)]}{\mathrm{Var}(f(\mathbf{x}))}}{D}$$

  where $\frac{D_{-i}}{D}$ is the total fractional variance *complement* to factor X_i.
- We think of TS_i as the total contribution of factor X_i to the total variation of $f(\mathbf{x})$.

Sensitivity Analysis

- In other words,

$$TS_i = 1 - \frac{D_{-i}}{D} = \frac{\frac{E_{\mathbf{X}-i}[\mathrm{Var}(f(\mathbf{x})|\mathbf{X}-i)]}{\mathrm{Var}(f(\mathbf{x}))}}{D}$$

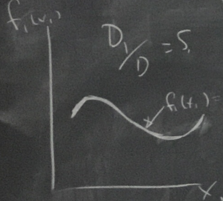
  where $\frac{D_{-i}}{D}$ is the total fractional variance *complement* to factor X_i.
- We think of TS_i as the total contribution of factor X_i to the total variation of $f(\mathbf{x})$.
- If S_i and TS_i are similar, it means that factor X_i primarily affects the variance of f through its main effect.

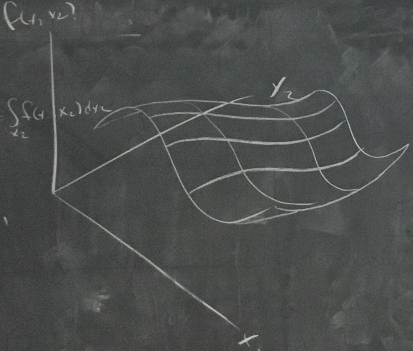
Sensitivity Analysis

- In other words,

$$TS_i = 1 - \frac{D_{-i}}{D} = \frac{\frac{E_{\mathbf{X}-i}[\mathrm{Var}(f(\mathbf{x})|\mathbf{X}-i)]}{\mathrm{Var}(f(\mathbf{x}))}}{D}$$

  where $\frac{D_{-i}}{D}$ is the total fractional variance *complement* to factor X_i.
- We think of TS_i as the total contribution of factor X_i to the total variation of $f(\mathbf{x})$.
- If S_i and TS_i are similar, it means that factor X_i primarily affects the variance of f through its main effect.
- If S_i and TS_i are different, then the higher-order effects and interactions involving X_i contribute to the variance of f.





$$f_0 + f_1(x_1) + f_2(x_2) + f_{12}(x_1, x_2)$$

$$+ \Big[\underbrace{\int_{x_2} f(x_1, x_2)\,dx_2}_{m_1(x_1)} - f_0\Big] + \Big[\underbrace{\int_{x_1} f(x_1, x_2)\,dx_1}_{m_2(x_2)} - f_0\Big] + \Big[f_{12}(x_1, x_2) - f_0 - \Big(\underbrace{\int_{x_2} f(x_1, x_2)\,dx_2}_{m_1(x_1)} - f_0\Big) - \Big(\underbrace{\int_{x_1} f(x_1, x_2)\,dx_1}_{m_2(x_2)} - f_0\Big)\Big]$$

$n = 25$

$N = 1{,}000$ $\hat{y}(x)$

Computing the Integrals

- Say we have some integral, $h = \int_{\chi} g(\mathbf{x}) d\mathbf{x}$ that we wish to compute.

Computing the Integrals

- Say we have some integral, $h = \int_{\chi} g(\mathbf{x}) d\mathbf{x}$ that we wish to compute.
- Approximate the integral using Monte Carlo integration (MC):

Computing the Integrals

- Say we have some integral, $h = \int_{\chi} g(\mathbf{x})d\mathbf{x}$ that we wish to compute.
- Approximate the integral using Monte Carlo integration (MC):
  - Draw $\mathbf{X}$ as N i.i.d. Uniform(0,1) random variates

Computing the Integrals

- Say we have some integral, $h = \int_{\chi} g(\mathbf{x}) d\mathbf{x}$ that we wish to compute.
- Approximate the integral using Monte Carlo integration (MC):
  - Draw $\mathbf{X}$ as N i.i.d. Uniform(0,1) random variates
  - Calculate $\hat{h} = \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the ith row of $\mathbf{X}$.

Computing the Integrals

- Say we have some integral, $h = \int_{\chi} g(\mathbf{x}) d\mathbf{x}$ that we wish to compute.
- Approximate the integral using Monte Carlo integration (MC):
  - Draw $\mathbf{X}$ as N i.i.d. Uniform(0,1) random variates
  - Calculate $\hat{h} = \frac{1}{n} \sum_{i=1}^{n} g(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the ith row of $\mathbf{X}$.
  - Then $\hat{h}$ is a Monte Carlo estimate of $\int_{\chi} g(\mathbf{x}) d\mathbf{x}$

Computing the Integrals

- Say we have some integral, $h = \int_{\chi} g(\mathbf{x})d\mathbf{x}$ that we wish to compute.
- Approximate the integral using Monte Carlo integration (MC):
  - Draw $\mathbf{X}$ as N i.i.d. Uniform(0,1) random variates
  - Calculate $\hat{h} = \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the ith row of $\mathbf{X}$.
  - Then $\hat{h}$ is a Monte Carlo estimate of $\int_{\chi} g(\mathbf{x})d\mathbf{x}$
- Even better: sample $\mathbf{X}$ using LHS, for instance. This is called Quasi Monte Carlo (QMC).

Computing Sensitivities via Monte Carlo

- Draw random samples $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ both of size N.

Computing Sensitivities via Monte Carlo

- Draw random samples $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ both of size N.
- Compute:

$$\widehat{f}_0 = \frac{1}{N} \sum_{m=1}^{N} f(\mathbf{x}_m)$$

$$\widehat{D} = \frac{1}{N} \sum_{m=1}^{N} f^2(\mathbf{x}_m) - \widehat{f}_0^2$$

$$\widehat{D}_i = \frac{1}{N} \sum_{m=1}^{N} f(\mathbf{x}_{-i,m}^{(1)}, x_{i,m}^{(1)}) f(\mathbf{x}_{-i,m}^{(2)}, x_{i,m}^{(1)}) - \widehat{f}_0^2$$

and

$$\widehat{D}_{-i} - \widehat{f}_0^2 = \frac{1}{N} \sum_{m=1}^{N} f(\mathbf{x}_{-i,m}^{(1)}, x_{i,m}^{(1)}) f(\mathbf{x}_{-i,m}^{(1)}, x_{i,m}^{(2)})$$

where $\mathbf{x}_{-i,m} = (\ldots, x_{i-1,m}, x_{i+1,m}, \ldots)$ and superscripts indicate using respective columns from two independent sampling matrices, and $\mathbf{x}$ (no superscript) uses either sample.

Computing Sensitivities via Monte Carlo

- Our sensitivities are then estimated as

$$\widehat{S}_i = \frac{\widehat{D}_i}{\widehat{D}} \text{ and } \widehat{TS}_i = 1 - \frac{\widehat{D}_{-i}}{\widehat{D}}$$

Computing Sensitivities for UQ

- How big to make N? The bigger the better.

Computing Sensitivities for UQ

- How big to make N? The bigger the better.
  - sometimes you may see $\widehat{S}_i < 0$ which is not possible, so this is due to numerical error.

Computing Sensitivities for UQ

- How big to make N? The bigger the better.
  - sometimes you may see $\widehat{S}_i < 0$ which is not possible, so this is due to numerical error.
- In practice applying this algorithm to $f(\mathbf{x})$ is infeasible, so the idea is to replace it with our statistical emulator, $\widehat{f}(\mathbf{x})$.

Computing Sensitivities for UQ

- How big to make N? The bigger the better.
  - sometimes you may see $\widehat{S}_i < 0$ which is not possible, so this is due to numerical error.
- In practice applying this algorithm to $f(\mathbf{x})$ is infeasible, so the idea is to replace it with our statistical emulator, $\widehat{f}(\mathbf{x})$.
  - this "plug-in" approach does not propagate uncertainties in $\widehat{f}$ through to $\widehat{S}_i$, $\widehat{TS}_i$. But there are ways of doing this - e.g. Bayesian approach (later).

Computing Sensitivities for UQ

- How big to make N? The bigger the better.
  - sometimes you may see $\widehat{S}_i < 0$ which is not possible, so this is due to numerical error.
- In practice applying this algorithm to $f(\mathbf{x})$ is infeasible, so the idea is to replace it with our statistical emulator, $\widehat{f}(\mathbf{x})$.
  - this "plug-in" approach does not propagate uncertainties in $\widehat{f}$ through to $\widehat{S}_i$, $\widehat{TS}_i$. But there are ways of doing this - e.g. Bayesian approach (later).
- So in UQ our sensitivities are subject to two sources of uncertainty: the MC sample size N in approximating the integrals, and the uncertainty of our emulator $\widehat{f}$ since we cannot freely evaluate our model f.

Sensitivity Analysis

- Effectively, SA is based on the following decomposition of the response variance:

$$\mathrm{Var}(f) = \mathrm{Var}_{X_i}(E_{\mathbf{X}_{-i}})(f|X_i)) + E_{X_i}(\mathrm{Var}_{\mathbf{X}_{-i}}(f|X_i))$$

where the first term is the main or first-order effect, and

$$\mathrm{Var}(f) = \mathrm{Var}_{\mathbf{X}_{-i}}(E_{X_i}(f|X_{-i})) + E_{\mathbf{X}_{-i}}(\mathrm{Var}_{X_i}(f|X_{-i}))$$

where the second term is the total-order effect of X_i.

Saltelli and Homma: *Sensitivity Analysis of model output: an investigation of new techniques*, Computational Statistics and Data Analysis, vol.15, pp.211–238 (1993).

Saltelli, Tarantola and Chan: *A quantitative, model-independent method for global sensitivity analysis of model output*, Technometrics, vol.41, pp.39–56 (1999).

Saltelli, Chan and Scott: *Sensitivity Analysis*, John Wiley & Sons, New York, NY. ISBN #0-471-99892-3 (2000).

Oakley and O'Hagan: *Probabilistic sensitivity analysis of complex models: a Bayesian*

Sensitivity Analysis

- Effectively, SA is based on the following decomposition of the response variance:

  $$\mathrm{Var}(f) = \mathrm{Var}_{X_i}(E_{\mathbf{X}_{-i}})(f|X_i)) + E_{X_i}(\mathrm{Var}_{\mathbf{X}_{-i}}(f|X_i))$$

  where the first term is the main or first-order effect, and

  $$\mathrm{Var}(f) = \mathrm{Var}_{\mathbf{X}_{-i}}(E_{X_i}(f|X_{-i})) + E_{\mathbf{X}_{-i}}(\mathrm{Var}_{X_i}(f|X_{-i}))$$

  where the second term is the total-order effect of X_i.
- For additive models, these diagonal terms are equal.

Saltelli and Homma: *Sensitivity Analysis of model output: an investigation of new techniques*, Computational Statistics and Data Analysis, vol.15, pp.211–238 (1993).

Saltelli, Tarantola and Chan: *A quantitative, model-independent method for global sensitivity analysis of model output*, Technometrics, vol.41, pp.39–56 (1999).

Saltelli, Chan and Scott: *Sensitivity Analysis*, John Wiley & Sons, New York, NY. ISBN #0-471-99892-3 (2000).

Oakley and O'Hagan: *Probabilistic sensitivity analysis of complex models: a Bayesian

Sensitivity Analysis

- Effectively, SA is based on the following decomposition of the response variance:

$$\mathrm{Var}(f) = \mathrm{Var}_{X_i}(E_{\mathbf{X}-i})(f|X_i)) + E_{X_i}(\mathrm{Var}_{\mathbf{X}-i}(f|X_i))$$

  where the first term is the main or first-order effect, and

$$\mathrm{Var}(f) = \mathrm{Var}_{\mathbf{X}-i}(E_{X_i}(f|X_{-i})) + E_{\mathbf{X}-i}(\mathrm{Var}_{X_i}(f|X_{-i}))$$

  where the second term is the total-order effect of X_i.
- For additive models, these diagonal terms are equal.
- If the model is linear, $\frac{\mathrm{Var}_{X_i}(E_{\mathbf{X}-i}(f|X_i))}{\mathrm{Var}(f)} = \beta^2_{X_i}$.

Saltelli and Homma: *Sensitivity Analysis of model output: an investigation of new techniques*, Computational Statistics and Data Analysis, vol.15, pp.211–238 (1993).

Saltelli, Tarantola and Chan: *A quantitative, model-independent method for global sensitivity analysis of model output*, Technometrics, vol.41, pp.39–56 (1999).

Saltelli, Chan and Scott: *Sensitivity Analysis*, John Wiley & Sons, New York, NY. ISBN #0-471-99892-3 (2000).

Oakley and O'Hagan: *Probabilistic sensitivity analysis of complex models: a Bayesian*

Example

- Consider the following function as our simulator which depends on 5 inputs that are scaled to $[0,1]^5$:

$$f(x) = 10sin(2\pi x_1 x_2) + (x_3 - 0.5)^2 + x_4 + x_5$$

Example

- Consider the following function as our simulator which depends on 5 inputs that are scaled to $[0, 1]^5$:

$$f(x) = 10sin(2\pi x_1 x_2) + (x_3 - 0.5)^2 + x_4 + x_5$$

- x_1, x_2 affect the response in a non-linear way through the $sin(\cdot)$ function

Example

- Consider the following function as our simulator which depends on 5 inputs that are scaled to $[0,1]^5$:

$$f(x) = 10sin(2\pi x_1 x_2) + (x_3 - 0.5)^2 + x_4 + x_5$$

- x_1, x_2 affect the response in a non-linear way through the $sin(\cdot)$ function
- x_3 is a quadratic effect

Example

- Consider the following function as our simulator which depends on 5 inputs that are scaled to $[0, 1]^5$:

$$f(x) = 10sin(2\pi x_1 x_2) + (x_3 - 0.5)^2 + x_4 + x_5$$

- x_1, x_2 affect the response in a non-linear way through the $sin(\cdot)$ function
- x_3 is a quadratic effect
- x_4, x_5 are linear effects

Example

- Because this function is known in closed-form and is rather amenable to hand calculations, we can derive the marginal 1-way effects.

Example

- Because this function is known in closed-form and is rather amenable to hand calculations, we can derive the marginal 1-way effects.
- For instance, recall that $f_i(x_i) = -f_0 + \int_{x_{-i}} f(x) dx_{-i}$

Example

- Because this function is known in closed-form and is rather amenable to hand calculations, we can derive the marginal 1-way effects.
- For instance, recall that $f_i(x_i) = -f_0 + \int_{x_{-i}} f(x) dx_{-i}$
- We calculate the 1-way marginal effects as:

$$f_1(x_1) = -\frac{10}{2\pi x_1} cos(2\pi x_1) + \frac{10}{2\pi x_1} + \frac{13}{12}$$

$$f_2(x_2) = -\frac{10}{2\pi x_2} cos(2\pi x_2) + \frac{10}{2\pi x_2} + \frac{13}{12}$$

$$f_3(x_3) = 3.87964 + (x_3 - 0.5)^2 + 1$$

$$f_4(x_4) = 3.87964 + \frac{7}{12} + x_4$$

$$f_5(x_5) = 3.87964 + \frac{7}{12} + x_5$$

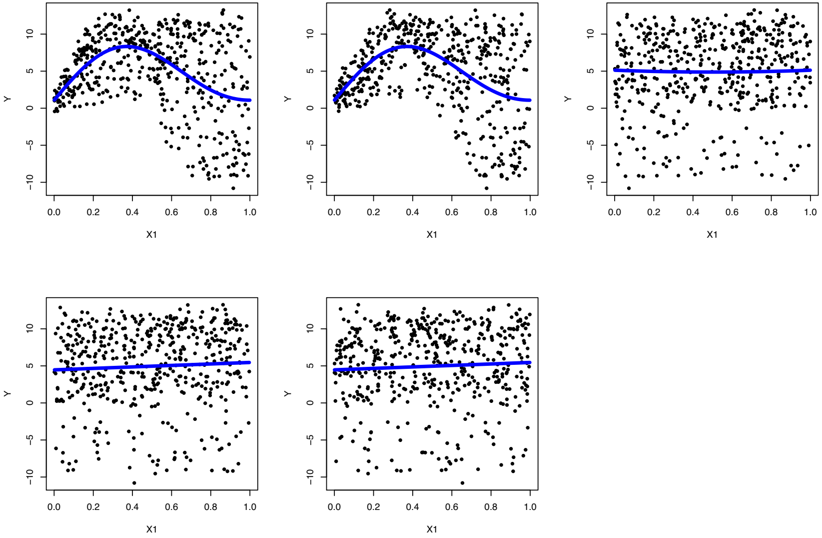
Example

```
# Generate data
set.seed(88) # just to replicate this example
n=500
X=matrix(runif(n*5),ncol=5)
f=10*sin(2*pi*X[,1]*X[,2])+(X[,3]-0.5)^2+X[,4]+X[,5]
y=f+rnorm(n,sd=1)

# true 1-way marginal effects
f1=-10/(2*pi*X[,1])*cos(2*pi*X[,1])+10/(2*pi*X[,1])+13/12
f2=-10/(2*pi*X[,2])*cos(2*pi*X[,2])+10/(2*pi*X[,2])+13/12
f3=3.87964+(X[,3]-0.5)^2+1
f4=3.87964+7/12+X[,4]
f5=3.87964+7/12+X[,5]
```

Example

```
# plot
par(mfrow=c(2,3))
plot(X[,1],y,xlab="X1",ylab="Y",pch=20,xlim=c(0,1))
ix=sort(X[,1],index.return=TRUE)$ix
lines(X[ix,1],f1[ix],lwd=4,col="blue")
plot(X[,2],y,xlab="X1",ylab="Y",pch=20,xlim=c(0,1))
ix=sort(X[,2],index.return=TRUE)$ix
lines(X[ix,2],f2[ix],lwd=4,col="blue")
plot(X[,3],y,xlab="X1",ylab="Y",pch=20,xlim=c(0,1))
ix=sort(X[,3],index.return=TRUE)$ix
lines(X[ix,3],f3[ix],lwd=4,col="blue")
plot(X[,4],y,xlab="X1",ylab="Y",pch=20,xlim=c(0,1))
ix=sort(X[,4],index.return=TRUE)$ix
lines(X[ix,4],f4[ix],lwd=4,col="blue")
plot(X[,5],y,xlab="X1",ylab="Y",pch=20,xlim=c(0,1))
ix=sort(X[,5],index.return=TRUE)$ix
lines(X[ix,5],f5[ix],lwd=4,col="blue")
```

Example



Example

```r
library(sensitivity)
N=10000
X1=data.frame(matrix(runif(N*5),ncol=5))
X2=data.frame(matrix(runif(N*5),ncol=5))
f.test <-function(X) {
    10*sin(2*pi*X[,1]*X[,2])+(X[,3]-0.5)^2+X[,4]+X[,5]
}
si.S=sobolEff(model=f.test,X1=X1,X2=X2,order=1,nboot=0)
si.TS=sobolEff(model=f.test,X1=X1,X2=X2,order=0,nboot=0)
```

Example

First-order sensitivity indices.

```
si.S$S
```

```
##      original std. error min. c.i. max. c.i.
## X1  0.212749   0.011655  0.189907  0.235591
## X2  0.219223   0.011626  0.196437  0.242009
## X3 -0.001839   0.009954 -0.021349  0.017671
## X4 -0.001164   0.009944 -0.020654  0.018326
## X5  0.001597   0.009931 -0.017867  0.021061
```

Example

Total sensitivity indices.

```
si.TS$S
```

```
##    original std. error min. c.i. max. c.i.
## X1 0.776517   0.011585  0.753811  0.799223
## X2 0.782737   0.011602  0.759997  0.805477
## X3 0.000191   0.000004  0.000184  0.000198
## X4 0.002856   0.000055  0.002749  0.002963
## X5 0.002858   0.000054  0.002752  0.002964
```