ENBIS pre-conference workshop

Introduction to Kriging using R and JMP

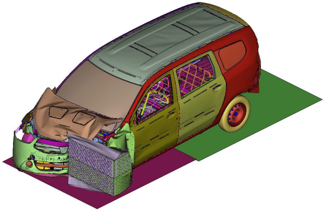
Nicolas Durrande – Mines Saint-Étienne

durrande@emse.fr

Context

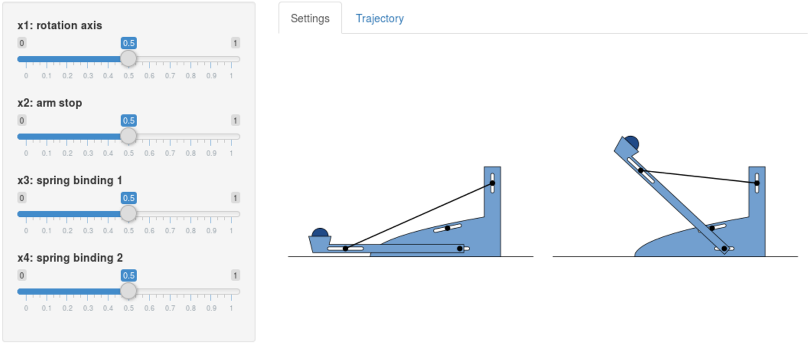
There is a wide variety of situations where getting data is extremely expensive.

- real world experiments
- destructive tests
- prototyping
- numerical experiments

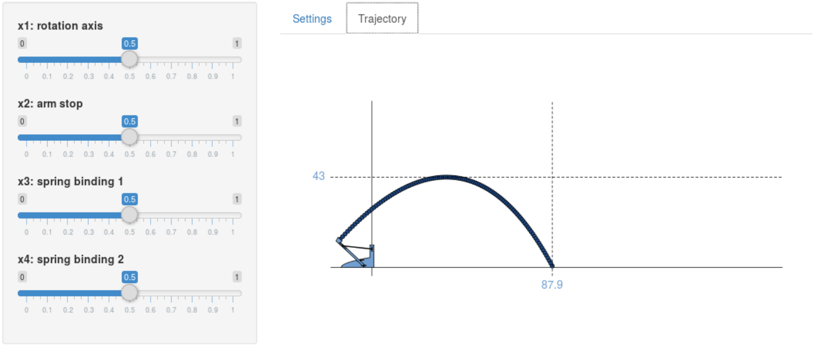


The number of experiments is thus limited.

During the lab session, we will consider a numerical simulator of a catapult depending on 4 inputs $x_1, \dots, x_4 \in [0, 1]$.


x1: rotation axis 0.5
x2: arm stop 0.5
x3: spring binding 1 0.5
x4: spring binding 2 0.5

This numerical simulator returns the length and height of the throw



x1: rotation axis 0.5

x2: arm stop 0.5

x3: spring binding 1 0.5

x4: spring binding 2 0.5

Lab session R (5 min)

1. Download and open the file "morning_lab.R" from Nicolas Durrande's homepage

   `https://sites.google.com/site/nicolasdurrandehomepage/seminars.`

2. Execute the first two uncommented lines.

Lab session JMP (5 min)

1. do as above...

or

1. go to: `https://durrande.shinyapps.io/catapult`

The experiment output can be seen as a function of the input parameters

$$y = f(x).$$

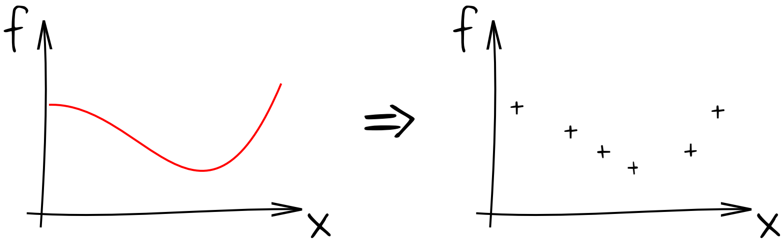
where f is a **costly to evaluate function**.

In the following, we will assume that

- $x \in \mathbb{R}^d$: There are many input parameters
- $y \in \mathbb{R}$: The output is a scalar.

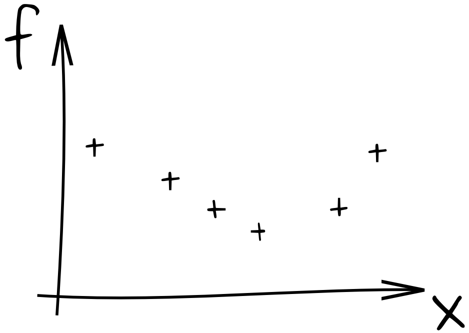
The fact that f is **costly to evaluate** changes a lot of things...

1. Ploting the function is not possible...



The fact that f is **costly to evaluate** changes a lot of things...

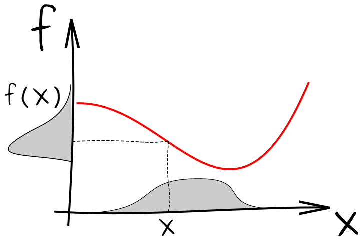
2. Computing integrals is not possible...

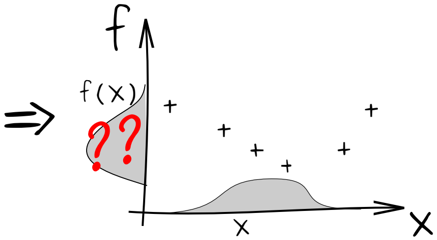


What is the mean value of f?

The fact that f is **costly to evaluate** changes a lot of things...

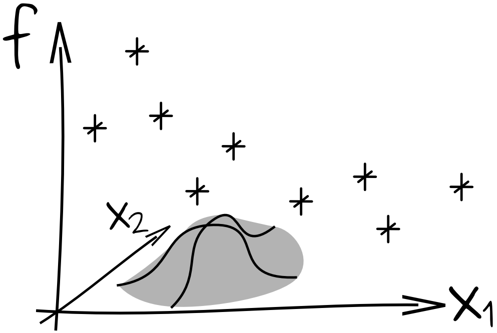
3. Uncertainty propagation is not possible...





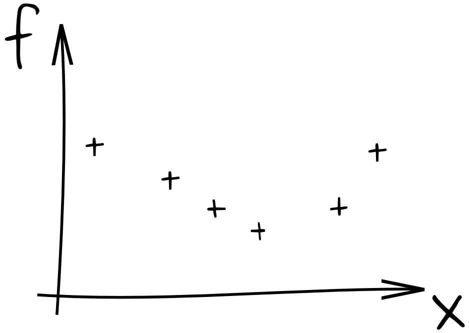
The fact that f is **costly to evaluate** changes a lot of things...

4. Sensitivity analysis is not possible...



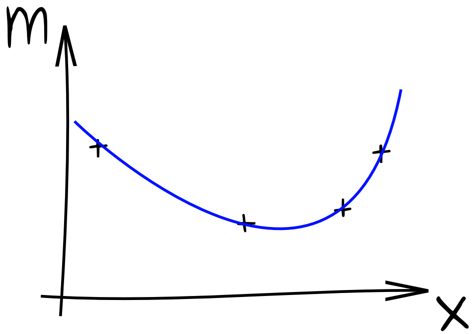
The fact that f is **costly to evaluate** changes a lot of things...

5. Optimisation is also tricky...



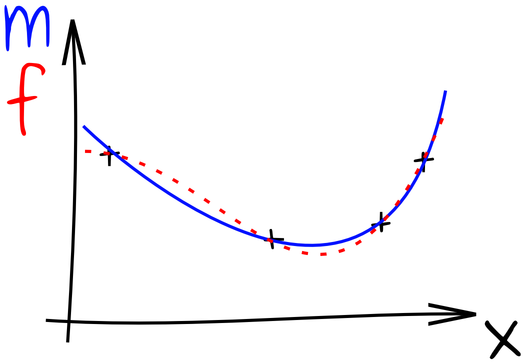
Surrogate models

The principle of statistical modelling is to use a few observations to build a mathematical approximation of the function.

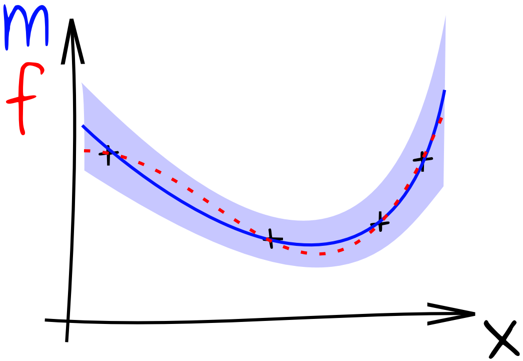


The model can then be used to answer all previous questions

Of course, there is a difference between f and m...



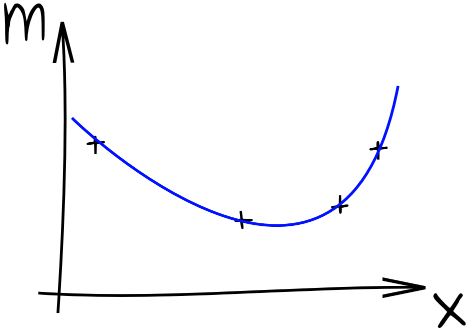
What about **statistical models**?
We want to be able to quantify the model error:



The confidence intervals can be used to obtain a **measure of uncertainty on the value of interest**.

There are many kinds of surrogate models:

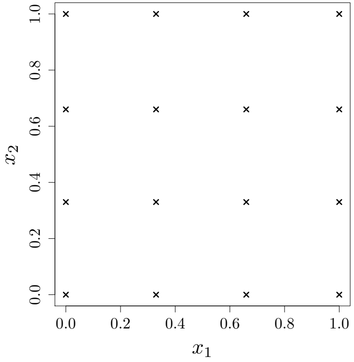
- linear regression
- polynomial chaos
- neural networks
- Kriging (or Gaussian process regression)



Example: linear regression for the catapult simulator

We consider the simulator as a function of only x_1 and x_2 and 16 observation points where the simulator will be run:

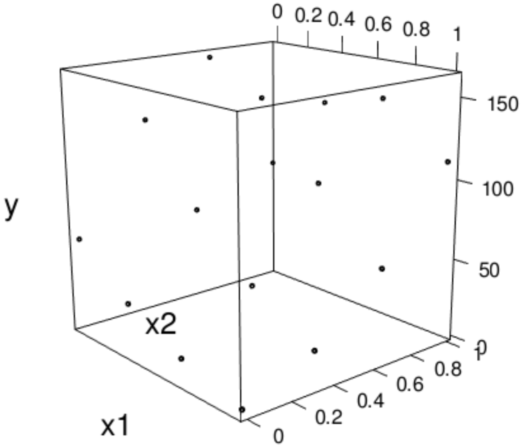
$$X = \begin{pmatrix} 0.00 & 0.00 \\ 0.33 & 0.00 \\ 0.66 & 0.00 \\ 1.00 & 0.00 \\ 0.00 & 0.33 \\ \vdots & \vdots \\ 1.00 & 1.00 \end{pmatrix}$$



Example: linear regression for the catapult simulator

We run the simulator (variables x_3 and x_4 are set to 0.5). We obtain

$$Y = \begin{pmatrix} 57.8 \\ 30.1 \\ 12.4 \\ 1.2 \\ 126.2 \\ \vdots \\ 111.0 \end{pmatrix}$$



Example: linear regression for the catapult simulator

We assume the following linear regression model

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \varepsilon_i$$

where the β_k are unknown and the ε_i are $\mathcal{N}(0, \sigma^2)$ and independents.

The β_k can be estimated by minimising the residual sum of squares:

```
> XY <- data.frame(x1=X[,1],x2=X[,2],y=Y)
> mr <- lm(y ~ 1 + x1 + x2, data = XY)
```

Example: linear regression for the catapult simulator

We obtain $\beta = (62.76,\ -59.71,\ 95.30)$:

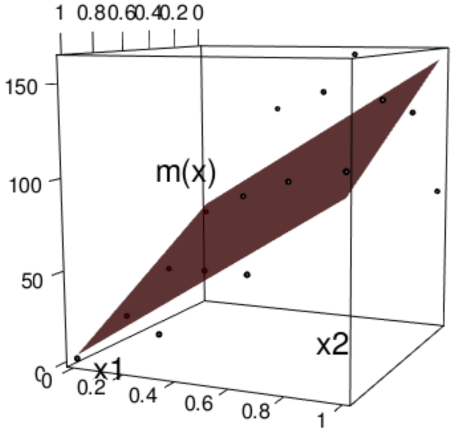
```
> summary(mr)

Residuals:
    Min      1Q  Median      3Q     Max
-78.011 -11.058  -3.427  15.187  35.165

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    62.76      15.66   4.008  0.00149 **
x1            -59.71      19.67  -3.035  0.00957 **
x2             95.30      19.67   4.845  0.00032 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 29.3 on 13 degrees of freedom
Multiple R-squared:  0.7155,Adjusted R-squared:  0.6717
F-statistic: 16.34 on 2 and 13 DF,  p-value: 0.0002832
```

Example: linear regression for the catapult simulator



Example: linear regression for the catapult simulator

The model can be improved by changing the basis functions:

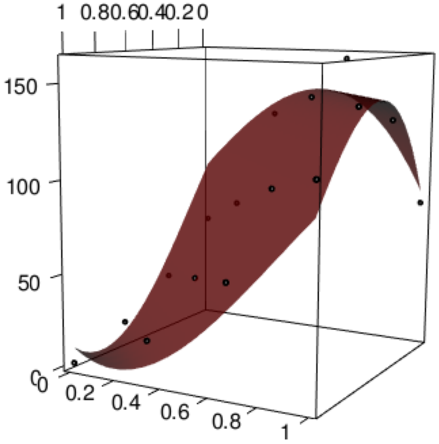
$$\phi_0 = 1,\ \phi_1 = x_1 - x_2,\ \phi_2 = (x_1 - x_2)^2,\ \text{and}\ \phi_3 = (x_1 - x_2)^3.$$

```
Residuals:
    Min      1Q  Median      3Q     Max
-35.042  -8.153   2.001   9.579  20.957

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      92.836      5.386  17.236 7.86e-10 ***
I(x1 - x2)     -147.913     16.510  -8.959 1.16e-06 ***
I((x1 - x2)^2)  -44.608     12.747  -3.500 0.004386 **
I((x1 - x2)^3)  109.325     22.661   4.824 0.000416 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.26 on 12 degrees of freedom
Multiple R-squared:  0.9191,Adjusted R-squared:  0.8989
F-statistic: 45.44 on 3 and 12 DF,  p-value: 7.933e-07
```

Example: linear regression for the catapult simulator



One can distinguish 4 important steps for building a surrogate model :

1. Choosing the evaluation points (Design of Experiments)
2. Evaluating the costly function
3. Choosing the type of model and estimating its parameters
4. Validating the model
5. Computing the quantity of interest

We will now detail these, with a focus on Kriging models.

Design of experiments

Design specificities for computer experiments

Two main constraints and their consequences.

1. For deterministic simulators, running twice the simulator at the same location gives the same result

   $\rightarrow$ Avoid replications

2. A simulator generally models a complex phenomenon with strongly non-linear behaviour

   $\rightarrow$ Fill the space, in order to avoid missing an area

Additional constraint:

3. The aforementioned principles should apply in lower subspaces, since the *true* dimensionality can be lower.

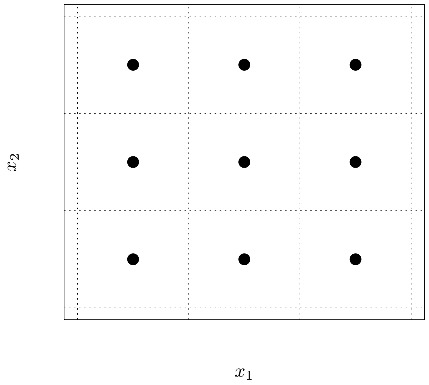
Example: $f_{\text{sim}}(x_1, x_2) = g(x_1)$

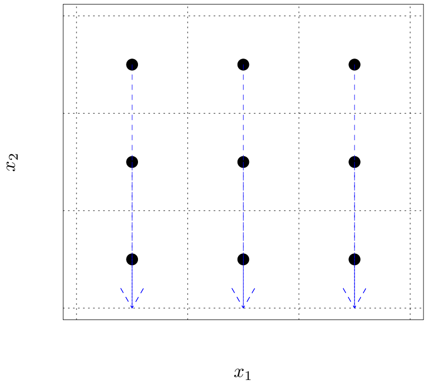
$\rightarrow$ It is useless to run the simulator with same x_1 locations!

$\rightarrow$ The x_1 locations should fill the space

A potentially bad design: How to waste 2/3 of runs!

Only 3 points among 9 are useful if f_{sim} depends only on x_1 (or x_2).





Various design types can be found in the literature:

- Latin Hypercubes
- Low discrepancy sequences
- Voronoi Tessellations
- ...

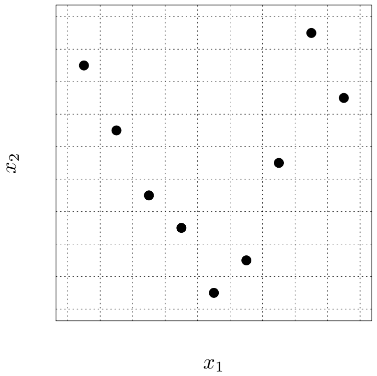
Various criteria can be used to asses a DoE quality

- Maximin
- Minimax
- Discrepancy
- IMSE
- ...

Some of them are implemented in the `DiceDesign package`

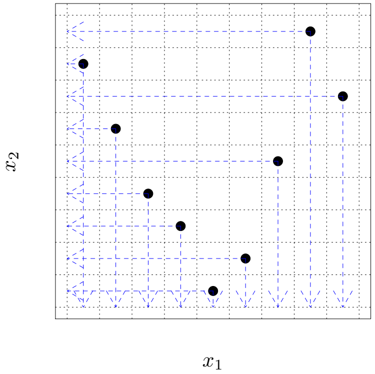
Better designs: Latin hypercubes

```
library(DiceDesign)
X <- lhsDesign(n = 9, dimension = 2, seed = 3,
                    randomized = FALSE)$design
```



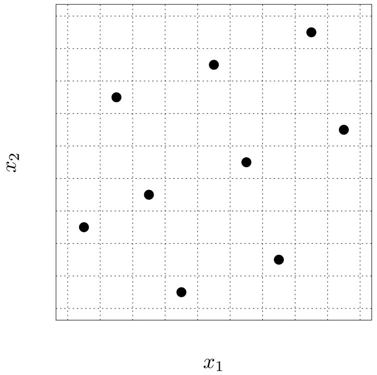
Better designs: Latin hypercubes

```
library(DiceDesign)
X <- lhsDesign(n = 9, dimension = 2, seed = 3,
                    randomized = FALSE)$design
```



Good(?) designs: space-filling Latin hypercubes

```
X0 <- lhsDesign(n = 9, dimension = 2)$design
X <- maximinESE_LHS(X0, it=2)$design
```



Lab session R (25 min)

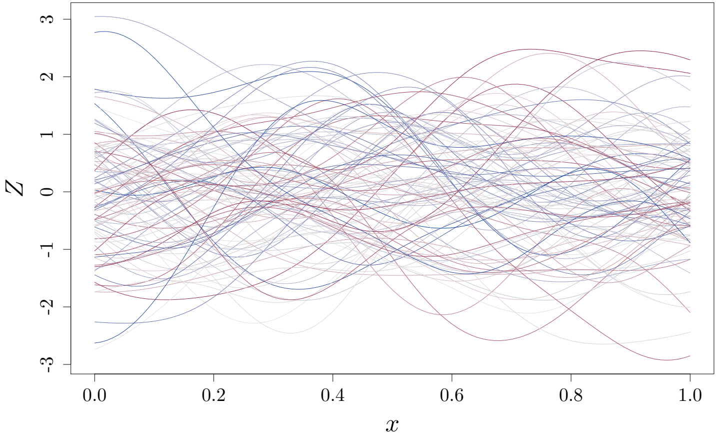
1. Load the DiceDesign package and build a DoE of 16 points in $[0, 1]^4$. Store it in a 16×4 matrix `X`.
2. Plot the columns of `X` against each other (`pairs(X)`).
3. Use the numerical simulator to compute the output values. Store them in a vector `Y`.
4. plot `Y` against the various input variables.
5. What optimal input values can you suggest?

Lab session JMP (25 min)

1. Open the file *Intro_to_Kriging_JMP.jrn*
2. Repeat the previous trials with a linear regression model.
3. Create a Design of experiments.
4. Use the numerical simulator to compute the output values. Store them in a vector `Y`.
5. plot `Y` against the various input variables.
6. What optimal input values can you suggest?

Kriging or Gaussian process regression

A random process is similar to a random variable but each draw is a function:



Gaussian processes are particular random processes: Z is a GP if any linear combination of $Z(x_i)$ is normally distributed.

The distribution of a

- **Gaussian random variable** is fully described by its *mean (scalar)* and *variance (positive scalar)*.
- **Gaussian vector** (i.e. multivariate normal) is fully described by its *mean vector* and *covariance matrix* (symmetric positive semi-definite matrix).
- **Gaussian process** is fully described by its *mean function* and *covariance function* (or kernel):

$$k(x, y) = \mathrm{cov}[Z(x), Z(y)]$$

The mean of a GP can be any function, but the kernel must satisfy two properties:

- It is symmetric: $k(x, y) = k(y, x)$
- It is positive semi-definite (psd):

$$\forall n \in \mathbb{N}, \forall x_i \in D, \forall \alpha \in \mathbb{R}^n, \ \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0$$

This can be written in a matrix fashion:

$$\forall n \in \mathbb{N}, \forall X \in D^n, \forall \alpha \in \mathbb{R}^n, \ \alpha^t k(X, X) \alpha \geq 0$$

Furthermore any symmetric psd function can be seen as the covariance of a Gaussian process. This equivalence is known as the Loeve theorem.

Exercise

Prove that the following functions are valid covariance functions

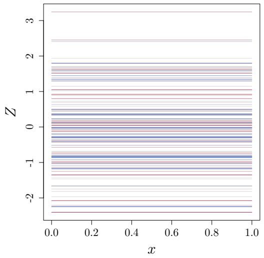
1. $k(x, y) = 1$
2. $k(x, y) = \delta_{x=y}$
3. $k(x, y) = \exp(-(x-y)^2)$

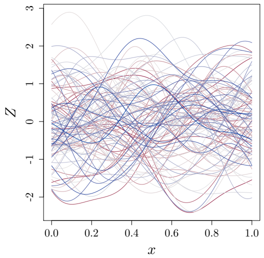
Exercise solutions

1. It can be proved directly from the definition. This is the covariance of the following GP:

2. As above, it can be proved using the definition. This is the covariance function of the white noise process.

3. This cannot be proved from the definition, as for most kernels. However, this is a valid covariance function.





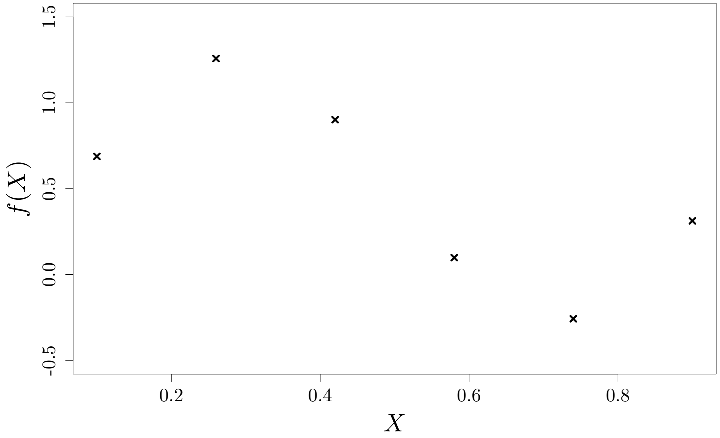
In order to plot sample paths from a GP $Z \sim \mathcal{N}(m(.), k(., .))$, we will consider samples of the GP discretised on a fine grid.

Exercise: Simulating sample paths

Let X be a set 100 regularly spaced points over the input space of Z.

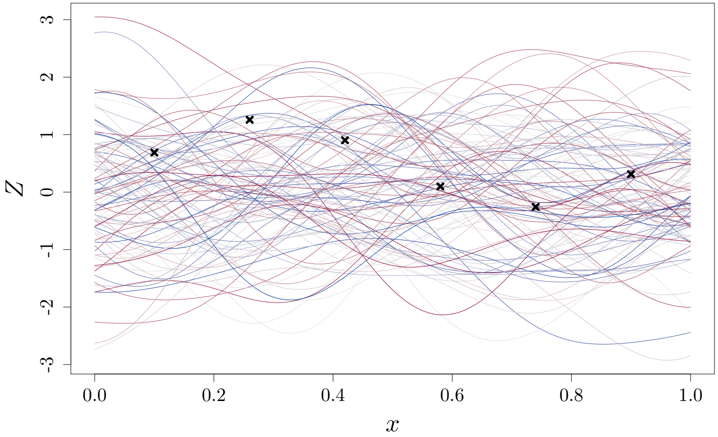
- What is the distribution of $Z(X)$?
- How to simulate samples from $Z(X)$?

We have observed the function f for a set of points $X = (X_1, \ldots, X_n)$:



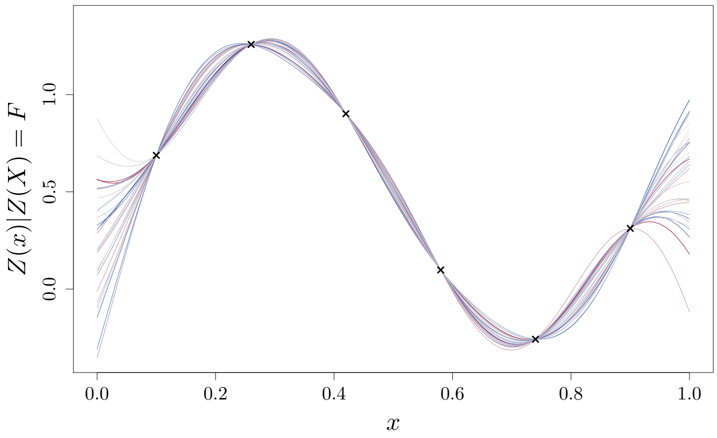
The vector of observations is $F = f(X)$ (ie $F_i = f(X_i)$).

We assume f is a sample from a given GP and we know it interpolates...

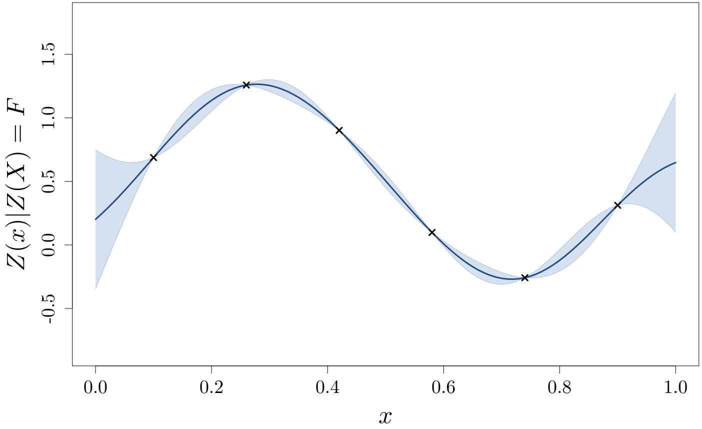


What can we say if we combine these two informations?

We obtain the following **conditional samples**:



Which can be summarized by a mean function and confidence intervals:



The conditional distribution $Z(x)|Z(X){=}F$ is still Gaussian and it can be computed analytically:

mean value

$$\begin{aligned} m(x) &= \mathrm{E}[Z(x)|Z(X){=}F] \\ &= k(x,X)k(X,X)^{-1}F \end{aligned}$$

predicted variance

$$\begin{aligned} v(x) &= \mathrm{var}[Z(x)|Z(X){=}F] \\ &= k(x,x) - k(x,X)k(X,X)^{-1}k(X,x) \end{aligned}$$

predicted covariance

$$\begin{aligned} c(x,y) &= \mathrm{cov}[Z(x),Z(y)|Z(X){=}F] \\ &= k(x,y) - k(x,X)k(X,X)^{-1}k(X,y) \end{aligned}$$

where $k(x,y) = \mathrm{cov}[Z(x),Z(y)]$.

A few remarkable properties of GPR models

- They can interpolate the data-points
- The prediction variance does not depend on the observations
- The mean predictor does not depend on the variance
- The mean predictor (usually) come back to zero when for predictions far away from the observations.

Lab session R (25 min)

1. Load the `DiceKriging` package look at the help of the function `km`.
2. Create a Kriging model on the catapult data with default kernel and parameters.
3. Use the functions `sectionview` and `sectionview3d` from the package `DiceView`.
4. Do some variables seem more influential than others?

Lab session JMP (25 min)

1. Create a Kriging model on the catapult data with default kernel and parameters.
2. Explore the model using the diagnostics.
3. Do some variables seem more influential than others?

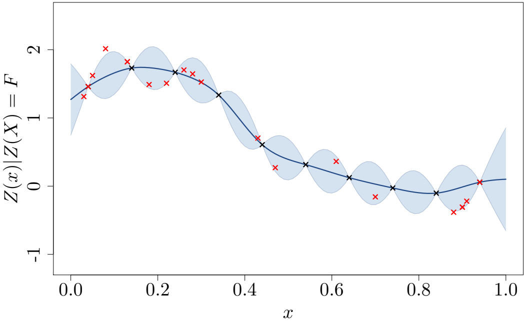
Model validation

We have seen that given some observations $F = f(X)$, it is very easy to build lots of models, either by changing the kernel parameters or the kernel itself.

The interesting question now is to know how to get a good model. To do so, we will need to answer the following questions:

- What is a good model?
- How to measure it?

The idea is to introduce new data and to compare the model prediction with reality



Since GPR models provide a mean and a covariance structure for the error they both have to be assessed.

Let X_t be the test set and $F_t = f(X_t)$ be the associated observations.

The accuracy of the mean can be measured by computing:

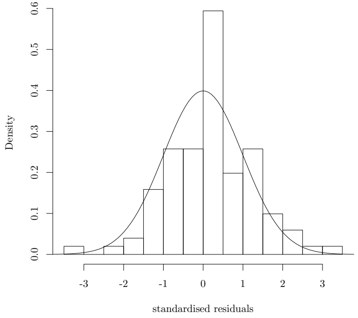
Mean Square Error	$MSE = \mathrm{mean}((F_t - m(X_t))^2)$
A "normalised" criterion	$Q_2 = 1 - \dfrac{\sum (F_t - m(X_t))^2}{\sum (F_t - \mathrm{mean}(F_t))^2}$

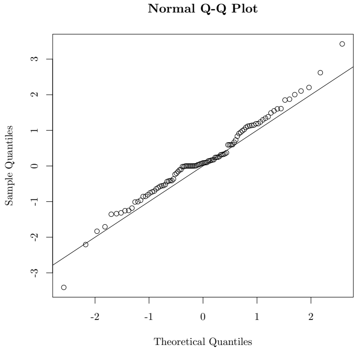
On the above example we get $MSE = 0.038$ and $Q_2 = 0.95$.

The predicted distribution can be tested by normalising the residuals.

According to the model, $F_t \sim \mathcal{N}(m(X_t), c(X_t, X_t))$.

$c(X_t, X_t)^{-1/2}(F_t - m(X_t))$ should thus be independents $\mathcal{N}(0, 1)$:





When no test set is available, another option is to consider cross validation methods such as leave-one-out.

The steps are:

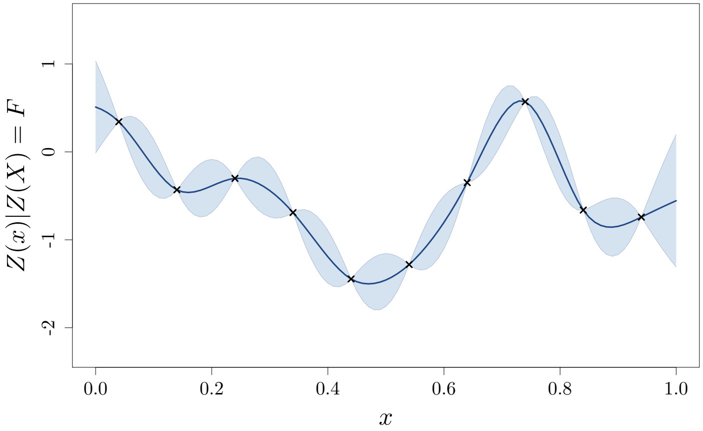
1. build a model based on all observations except one
2. compute the model error at this point

This procedure can be repeated for all the design points in order to get a vector of error.

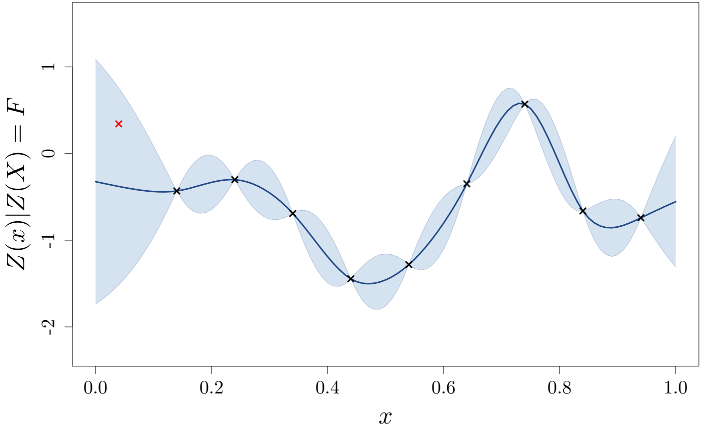
On the previous example we obtain $MSE = 0.24$ and $Q_2 = 0.34$.

Why doesn't the model perform as good previously?

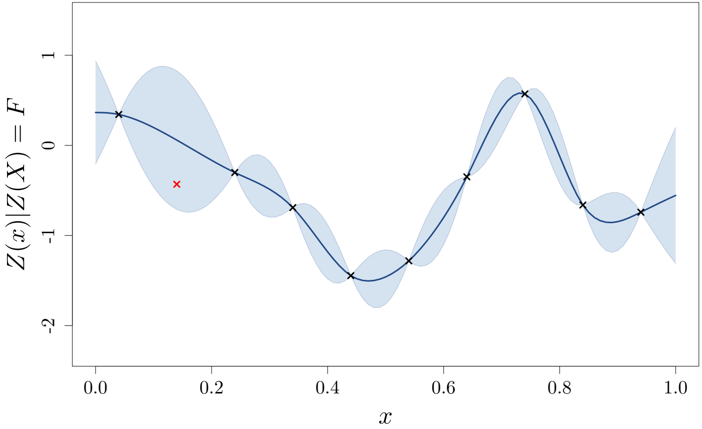
Model to be tested:



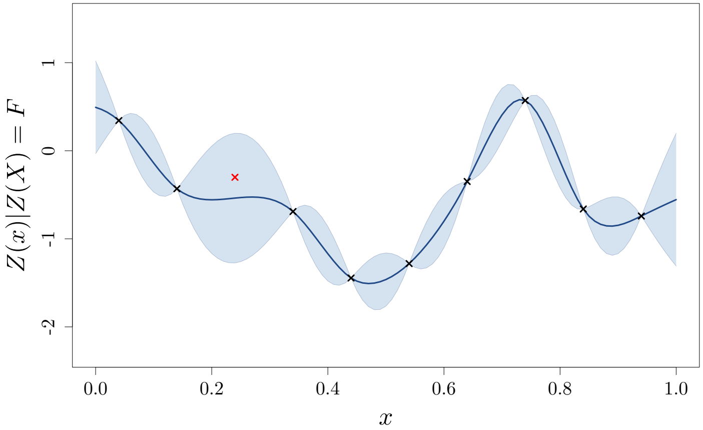
Step 1:



Step 2:

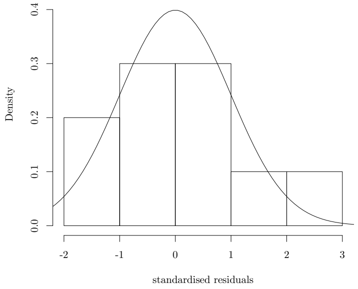


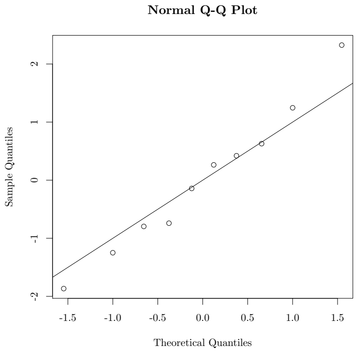
Step 3:



It turns out that the error is always computed at the 'worst' location!

We can also look at the residual distribution. For leave-one-out, there is no joint distribution for the residuals so they have to be standardised independently. We obtain:


Density
standardised residuals


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles