RGeostats tutorial

N. Desassis
D. Renard

MINES-ParisTech - Géosciences

ORGANIZATION

- 4 main tutorials
  - Variography
  - Estimation (kriging)
  - Multivariate Geostatistics
  - Simulations (part I-II-III)

Organization

- 4 main tutorials
  - Variography
  - Estimation (kriging)
  - Multivariate Geostatistics
  - Simulations (part I-II-III)
- For each tutorial
  - An ascii (text) file is available
  - Preamble
  - Several independent parts
  - Copy-Paste each line
  - You should run the preamble before each part!
  - Some parts are indicated as “see also”
  - You can also play by yourself

Scotland January temperatures

- The data set contains temperatures of January measured on 151 stations and averaged over twenty years (1961-1980)[1]
- The data set will be used for the first three parts

[1] **Hudson, G. and Wackernagel, H. (1994) Mapping temperature using kriging with external drift: theory and example from Scotland. International Journal of Climatology 14, 77-91.**

1. Preamble: load and prepare the data
2. Exploratory data analysis
3. Compute and plot an omnidirectional experimental variogram
4. Fit a model
5. Several directions
6. Variogram map
7. Variogram for data on a grid

```
# Load the data
# Change the name (to have a shorter name)
# Make the variable 4 (Elevation) inactive

data(Exdemo_Scotland_Temperatures)
dat=Exdemo_Scotland_Temperatures
dat=db.locate(dat,4)
```

```
# Load the data
# Change the name (to have a shorter name)
# Make the variable 4 (Elevation) inactive

data(Exdemo_Scotland_Temperatures)
dat=Exdemo_Scotland_Temperatures
dat=db.locate(dat,4)

#Last line equivalent to
dat=db.locate(dat,4,"NA")
dat=db.locate(dat,"Elevation","NA")
#...
```

```
?db.plot
dat
dat[]
hist(dat[,5])
db.stat(dat,fun="maxi",name=5)
plot(dat)
plot(dat,scalefactor=1)
plot(dat,scale=1)
plot(dat,scale=1,name.post=5)
```

```
#For documentation on vario-class
class?vario
```

```
#For documentation on vario-class
class?vario

vario=vario.calc(dat,lag=10,nlag=40)
```

```
#For documentation on vario-class
class?vario

vario=vario.calc(dat,lag=10,nlag=40)

?vario.plot
```

```
#For documentation on vario-class
class?vario

vario=vario.calc(dat,lag=10,nlag=40)

?vario.plot

plot(vario)
plot(vario,npairdw=TRUE)
plot(vario,npairdw=TRUE,npairpt=TRUE)
```

```
#For documentation on model-class
class?model
```

```
#For documentation on model-class
class?model

model=model.auto(vario)
```

```
#For documentation on model-class
class?model

model=model.auto(vario)

melem.name()
melem.name(c(1,2,3))
```

```
#For documentation on model-class
class?model

model=model.auto(vario)

melem.name()
melem.name(c(1,2,3))

model.auto(vario,struct=melem.name(c(1,2,3)))
model.auto(vario,struct=melem.name(c(1,3,2)))
```

```
#For documentation on model-class
class?model

model=model.auto(vario)

melem.name()
melem.name(c(1,2,3))

model.auto(vario,struct=melem.name(c(1,2,3)))
model.auto(vario,struct=melem.name(c(1,3,2)))

model.auto(vario,struct=melem.name(c(1,3,2)),
           maxiter=0)
```

```
model.auto(vario,struct=melem.name(c(1,2,3)),
           flag.noreduce=T)
```

```
model.auto(vario,struct=melem.name(c(1,2,3)),
           flag.noreduce=T)

model.auto(vario,struct=melem.name(c(1,2,3)),
           flag.noreduce=T,verbose=T)
```

```
model.auto(vario,struct=melem.name(c(1,2,3)),
           flag.noreduce=T)

model.auto(vario,struct=melem.name(c(1,2,3)),
           flag.noreduce=T,verbose=T)


model.auto(vario,struct=melem.name(c(1,2,3)),
           flag.noreduce=T,
           lower=c(0.02,NA,NA,NA,NA),
           upper=c(0.02,NA,NA,NA,400))
```

```
vario_4dir=vario.calc(dat,lag=15,nlag=15,
                      dir=c(0,45,90,135))
vario_4dir

m4dir=model.auto(vario_4dir,
                 struct=melem.name(c(1,3,2)))
```

```
#Display results

plot(vario_4dir,npairdw=TRUE)
plot(m4dir,add=T,vario=vario_4dir)
```

```
vm=vmap.calc(dat,nx=10,dx=15)
plot(vm)
```

```
vm=vmap.calc(dat,nx=10,dx=15)
plot(vm)

model=vmap.auto(vm,struct=melem.name(c(1,3,2)),
                flag.noreduce=T,
                lower=c(0.02,NA,NA,NA,NA),
                upper=c(0.02,NA,NA,NA,NA))
```

```
vario_4dir=vario.calc(dat,lag=15,nlag=15,
                      dir=c(0,45,90,135))

plot(vario_4dir)
plot(model,vario=vario_4dir,add=T)
```

```
data(Exdemo_Scotland_Elevations)
grid=Exdemo_Scotland_Elevations
```

```
data(Exdemo_Scotland_Elevations)
grid=Exdemo_Scotland_Elevations

plot(grid,scale=1,col=topo.colors(100),
     pos.legend=5)
```

```
data(Exdemo_Scotland_Elevations)
grid=Exdemo_Scotland_Elevations

plot(grid,scale=1,col=topo.colors(100),
     pos.legend=5)

vario=vario.grid(grid,nlag=30)
plot(vario)
```

```
model.auto(vario,struct=melem.name(c(1,3,2)))
```

```
model.auto(vario,struct=melem.name(c(1,3,2)))

model.auto(vario,struct=melem.name(c(1,3,2)),
           auth.aniso=F)
```

1. Preamble
   - Load and prepare the data
   - Create the variogram model
   - Create a neighborhood
2. Perform kriging
3. Cross-validation
4. Moving neighborhood

```
#1) Prepare the data

data(Exdemo_Scotland_Temperatures)
data(Exdemo_Scotland_Elevations)
dat=Exdemo_Scotland_Temperatures
grid=Exdemo_Scotland_Elevations
dat=db.locate(dat,4)
```

```
#2) Define the model

vario=vario.calc(dat,lag=10,nlag=40)
model=model.auto(vario,
                 struct=melem.name(c(1,3,2)))

# We could have used model.create (script) or
#                    model.input (interface)
# to create a model.
```

```
#3) Define the neighborhood

unique.neigh=neigh.init(type=0,ndim=2)

#Equivalent to

unique.neigh=neigh.input(ndim=2)
#answer 0
```

```
result0=kriging(dat,grid,model=model,
                neigh=unique.neigh)
```

```
result0=kriging(dat,grid,model=model,
                neigh=unique.neigh)
plot(result0,scale=1,
     col=topo.colors(100),pos.legend=5)
plot(dat,add=T)
```

```
result0=kriging(dat,grid,model=model,
                neigh=unique.neigh)
plot(result0,scale=1,
     col=topo.colors(100),pos.legend=5)
plot(dat,add=T)
plot(result0,name=7,scale=1,
     col=topo.colors(100),pos.legend=5)
plot(dat,add=T)
```

```
res.xv = xvalid(dat,model,neigh=unique.neigh)
hist(res.xv[,6],breaks=20)
hist(res.xv[,7],breaks=20)
mean(res.xv[,7]^2,na.rm=T)
plot(res.xv,scale=1)
```

```
res.xv = xvalid(dat,model,neigh=unique.neigh)
hist(res.xv[,6],breaks=20)
hist(res.xv[,7],breaks=20)
mean(res.xv[,7]^2,na.rm=T)
plot(res.xv,scale=1)

plot(result0,scale=1,
     col=topo.colors(100),pos.legend=5)
plot(res.xv,scale=1,add=T,
     col=1+as.numeric(res.xv[,6]>0))
```

```
moving.neigh=neigh.input()
#answers 2, 5, 200, n, n, 50
```

```
moving.neigh=neigh.input()
#answers 2, 5, 200, n, n, 50

result1=kriging(dat,grid,
                model=model,neigh=moving.neigh)
```

```
moving.neigh=neigh.input()
#answers 2, 5, 200, n, n, 50

result1=kriging(dat,grid,
                model=model,neigh=moving.neigh)
plot(result1,scale=1,
     col=topo.colors(100),pos.legend=5)
```

```
moving.neigh=neigh.input()
#answers 2, 5, 200, n, n, 50

result1=kriging(dat,grid,
                model=model,neigh=moving.neigh)
plot(result1,scale=1,
     col=topo.colors(100),pos.legend=5)
plot(result0[,6],result1[,6],cex=.2)
abline(0,1,col=2)
```

3) Multivariate Geostatistics

Introduction

- The temperature stations are not located at high elevations
- The temperature depends on the altitude
- We know the elevation of the stations (and some other elevations)
- We want to use this auxiliary information to improve the prediction
- In a second step, we will use the full Digital Terrain Model (DTM)

1. Preamble
2. Exploratory data analysis (bivariate)
3. Cross-variogram model
4. Co-kriging
5. Collocated co-kriging
6. External drift

The results of ordinary kriging are stocked in `result0`.

The results of ordinary kriging are stocked in `result0`.

```
data(Exdemo_Scotland_Temperatures)
data(Exdemo_Scotland_Elevations)
dat=Exdemo_Scotland_Temperatures
grid=Exdemo_Scotland_Elevations
```

The results of ordinary kriging are stocked in `result0`.

```
data(Exdemo_Scotland_Temperatures)
data(Exdemo_Scotland_Elevations)
dat=Exdemo_Scotland_Temperatures
grid=Exdemo_Scotland_Elevations
```

Creation of the variable indicating if the temperature is available or if there is only the elevation.

```
dat=db.add(dat,
           "Available_Temp"=(!is.na(dat[,5])),
           loctype="NA")
```

The results of ordinary kriging are stocked in `result0`.

```
data(Exdemo_Scotland_Temperatures)
data(Exdemo_Scotland_Elevations)
dat=Exdemo_Scotland_Temperatures
grid=Exdemo_Scotland_Elevations
```

Creation of the variable indicating if the temperature is available or if there is only the elevation.

```
dat=db.add(dat,
           "Available_Temp"=(!is.na(dat[,5])),
           loctype="NA")
```

```
unique.neigh=neigh.init(type=0,ndim=2)
```

```
dat=db.locate(dat,6,"sel")
plot(dat)
db.stat(dat,fun="maxi",names=4)
```

```
dat=db.locate(dat,6,"sel")
plot(dat)
db.stat(dat,fun="maxi",names=4)
dat[,6]=!dat[,6]
plot(dat,add=T,col=3)
db.stat(dat,fun="maxi",names=4)
```

```
dat=db.locate(dat,6,"sel")
plot(dat)
db.stat(dat,fun="maxi",names=4)
dat[,6]=!dat[,6]
plot(dat,add=T,col=3)
db.stat(dat,fun="maxi",names=4)
dat[,6]=!dat[,6]
correlation(dat,icol1=4,icol2=5,col=c(2,3),
            pos.legend=1)
```

```
dat=db.locate(dat,6,"sel")
plot(dat)
db.stat(dat,fun="maxi",names=4)

dat[,6]=!dat[,6]
plot(dat,add=T,col=3)
db.stat(dat,fun="maxi",names=4)

dat[,6]=!dat[,6]
correlation(dat,icol1=4,icol2=5,col=c(2,3),
            pos.legend=1)

regression(dat,icol1=4,icol2=5,flag.draw=T,
           save.coeff= T)
```

```
cross_vario=vario.calc(dat,lag=10,nlag=40)
m_model=model.auto(cross_vario,
                   struct=melem.name(c(1,3,2)),
                   flag.noreduce=T)
```

Isotopic

```
dat=db.locate(dat,6,"sel")
```

Isotopic

```
dat=db.locate(dat,6,"sel")

resultm1=kriging(dat,grid,model=m_model,
                 neigh=unique.neigh)
plot(resultm1,name="K.January_temp.estim",
     scale=1,pos.legend=5,col=topo.colors(100))
```

Heterotopic

```
dat=db.locate(dat,6)
```

Heterotopic

```
dat=db.locate(dat,6)

resultm2=kriging(dat,grid,model=m_model,
                 neigh=unique.neigh)
plot(resultm2,name="K.January_temp.estim",
     scale=1,pos.legend=5,col=topo.colors(100))
```

```
moving.neigh=neigh.input(ndim=2)
#Answers 2, 5, 20, n, n, 50
```

```
moving.neigh=neigh.input(ndim=2)
#Answers 2, 5, 20, n, n, 50

resultm3=kriging(dat,grid,model=m_model,
                 neigh=moving.neigh,flag.colk=T)
```

```
moving.neigh=neigh.input(ndim=2)
#Answers 2, 5, 20, n, n, 50

resultm3=kriging(dat,grid,model=m_model,
                 neigh=moving.neigh,flag.colk=T)

plot(resultm3,name="K.January_temp.estim",
     scale=1,pos.legend=5,col=topo.colors(100))
```

```
dat=db.locate(dat,4,"f")
grid=db.locate(grid,4,"f")
```

```
dat=db.locate(dat,4,"f")
grid=db.locate(grid,4,"f")

dat_r=regression(dat,flag.bivar=FALSE)
vario=vario.calc(dat_r,nlag=40,lag=10)
mres=model.auto(vario,struct=melem.name(c(5,3)),
                upper=c(NA,NA,NA,300))
```

```
resultm4=kriging(dat,grid,model=mres,
                 uc=c("1","f1"),
                 neigh=unique.neigh)
```

```
resultm4=kriging(dat,grid,model=mres,
                 uc=c("1","f1"),
                 neigh=unique.neigh)

plot(resultm4,name="K.January_temp.estim",
     scale=1,pos.legend=5,
     col=topo.colors(100))
```

```
grid=db.create(nx=c(100,100))
model=model.create(vartype= "Gaussian",range=30)
```

```
grid=db.create(nx=c(100,100))
model=model.create(vartype= "Gaussian",range=30)

grid = simtub(,grid,model=model,nbtuba=1,seed=0)
plot(grid,scale=1,col=topo.colors(100))
```

Try several times the last two commands.

```
grid=db.create(nx=c(100,100))
model=model.create(vartype= "Gaussian",range=30)

grid = simtub(,grid,model=model,nbtuba=1,seed=0)
plot(grid,scale=1,col=topo.colors(100))
```

Try several times the last two commands.
Try with

```
nbtuba = 2
```

and

```
nbtuba = 10
```

and

```
nbtuba = 100
```

and

```
nbtuba = 1000
```

A stationary Gaussian random function $Y = (Y(x), x \in \mathbb{R}^d)$ with

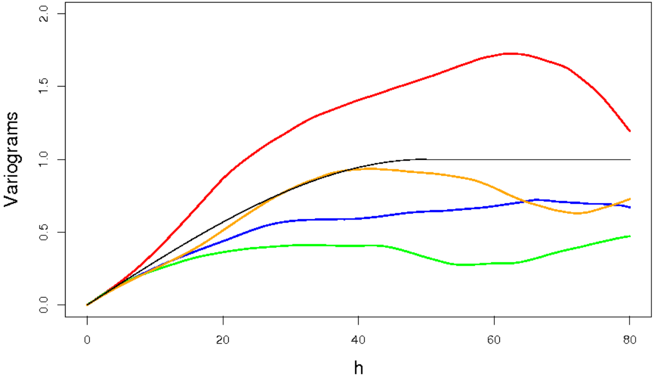
- mean m
- variance σ^2
- correlation function $\rho(h)$
- variogram $\gamma(h) = \sigma^2(1 - \rho(h))$

is simulated on a domain V.

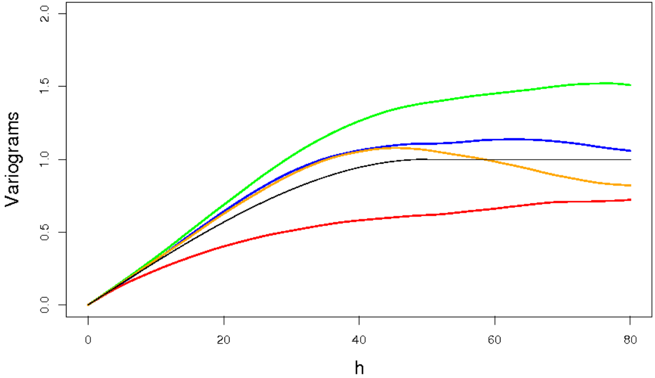
$(y(x), x \in V)$ is the realization

From the simulation $(y(x), x \in V)$ some experimental quantities can be computed:

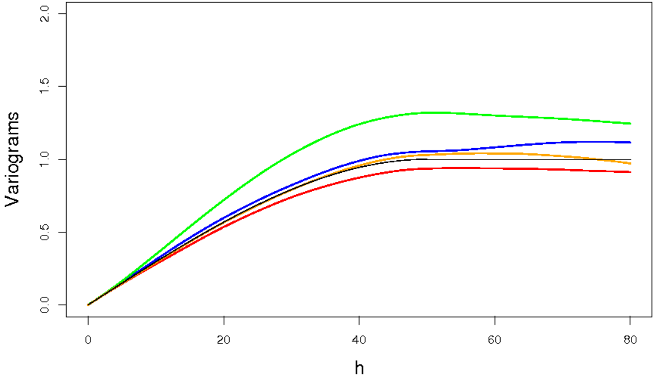
- the mean $y_{_V} = \dfrac{1}{|V|}\displaystyle\int_V y(x)dx$
- the dispersion variance $s^2(0\,|V) = \dfrac{1}{|V|}\displaystyle\int_V (y(x) - y_{_V})^2 dx$
- the regional variogram $\gamma_{_V}(h) = \dfrac{1}{2|V \cap V_{-h}|}\displaystyle\int_{V\cap V_{-h}} (y(x) - y(x+h))^2 dx$
- some proportions $p(a) = \dfrac{1}{|V|}\displaystyle\int_V 1_{y(x)<a} dx$



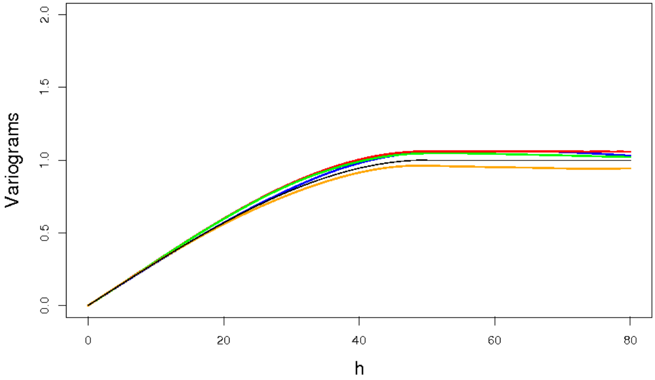
$$V = 100 \times 100 \qquad \gamma = Sph(1, 50)$$



$$V = 200 \times 200 \qquad \gamma = Sph(1, 50)$$



$$V = 400 \times 400 \qquad \gamma = Sph(1, 50)$$



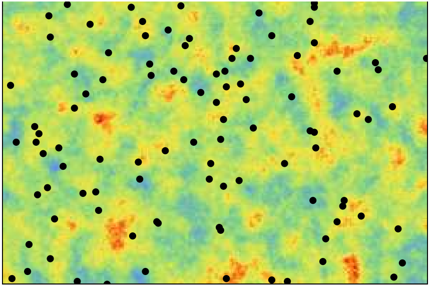
$$V = 800 \times 800 \qquad \gamma = Sph(1, 50)$$

The experimental quantities are not equal to their corresponding quantities in the model:

- $y_V \neq m$
- $s^2(0\,|V) \neq \sigma^2$
- $\gamma_V(h) \neq \gamma(h)$
- $p(a) \neq G\left(\dfrac{a-m}{\sigma}\right)$ (G is the Gaussian cumulative distribution function)

- they differ from one simulation to another (statistical fluctuations)
- the experimental statistics tends to their corresponding quantities in the model when the size of V increases (ergodicity)

How to simulate a realization $\{y(x), x \in \mathbb{R}^d\}$ of a second order Gaussian random function $\{Y(x), x \in \mathbb{R}^d\}$ with mean m and covariance function C with respect to the data $\{Y(x_i) = y_i, i = 1, \dots, n\}$?



Let

$$Y(x) = Y^{SK}(x) + Y(x) - Y^{SK}(x)$$

where

$Y^{SK}(x) = m + \sum_{j=1}^{n} \lambda_j(x)[Y(x_j) - m]$ simple kriging

$Y(x) - Y^{SK}(x)$ kriging residuals

Y^{SK} et $Y - Y^{SK}$ are to independant Gaussian random functions *(i) make an unconditional simulation* $\{s(x), x \in \mathbb{R}^d\}$ *and set* $s_j = s(x_j)$*.*

(ii) for all $x \in \mathbb{R}^d$*, compute the kriging weights* $(\lambda_j(x), j = 1, \ldots, n)$

(iii) set

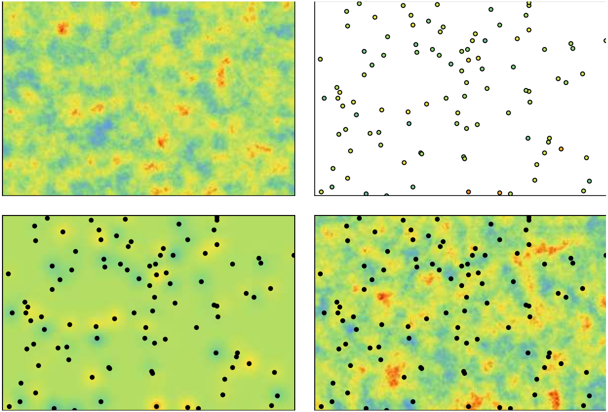
$$\begin{aligned} y^{CS}(x) &= y^{SK}(x) + s(x) - s^{SK}(x) \\ &= s(x) + \sum_{j=1}^{n} \lambda_j(x)(y_j - s_j) \end{aligned}$$

- If $x = x_i$, then $y^{CS}(x_i) = y_i + s(x_i) - s(x_i) = y_i$

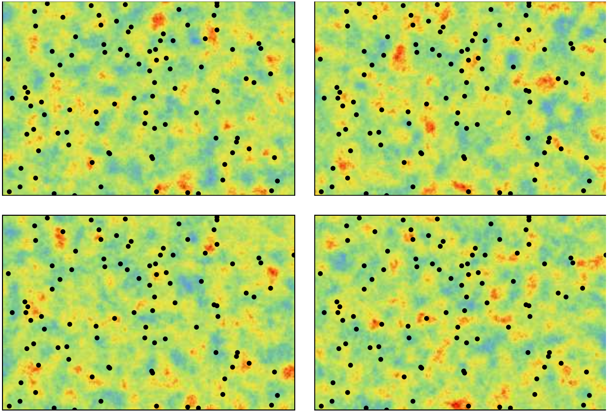
  The conditioning data are honored

- If $C(x - x_\alpha) \simeq 0$ for all $i = 1, \ldots, n$, then $y^{CS}(x) \simeq m + s(x) - m = s(x)$

  For points distant from the data, the simulation is unconditional



Simulation (TL), Synthetic data (TR), Simple kriging (BL), a conditional simulation (BR)



- Limitation of the Gaussian model:
  - Symetrical distribution
  - Inadapted to model data with heavy tail histogram
  - Conditional variance does not depend on the data (homoscedasticity)
- Anamorphosis: bijective function Φ defined on $\mathbb{R}$
- Model: data are a realization of a second order stationary Gaussian random function transformed by Φ
- Example : The log-normal model

$$\Phi(Y) = \exp(\mu + \sigma Y)$$

Model

$(z(x_1), \ldots, z(x_n))$ is a realization of $(Z(x), x \in \mathbb{R}^d)$, a transformed Gaussian random function:

- $Z(x) = \Phi(Y(x))$ for all $x \in \mathbb{R}^d$
- $(Y(x), x \in \mathbb{R}^d)$ a second order stationary Gaussian random function with correlation function C

How to simulate a realization $(z(x), x \in \mathbb{R}^d)$ from $(Z(x), x \in \mathbb{R}^d)$ such as $Z(x_i) = z(x_i)$ for $i = 1, \ldots, n$?

(i) transform the data in the Gaussian scale

$$y_i = \Phi^{-1}(z(x_i))$$

(ii) Perform a conditional simulation $\{y(x), x \in \mathbb{R}^d\}$ *from the Gaussian random function* $(Y(x), x \in \mathbb{R}^d)$ *which honors the data in the Gaussian case*

$$Y(x_i) = y_i$$

(iii) transform back the simulation for all $x \in \mathbb{R}^d$

$$z(x) = \Phi(y(x))$$

```
data(Exdemo_Scotland_Temperatures)
data(Exdemo_Scotland_Elevations)
dat=Exdemo_Scotland_Temperatures
grid=Exdemo_Scotland_Elevations
dat=db.locate(dat,4)
vario=vario.calc(dat,lag=10,nlag=40)
model=model.auto(vario, flag.noreduce=T,
                 struct=melem.name(c(1,2,3)),
                 lower=c(0.02,NA,NA,NA,NA),
                 upper=c(0.02,NA,NA,NA,NA))
unique.neigh=neigh.init(type=0,ndim=2)
```

```
m=mean(dat[,5],na.rm=T)
simu = simtub(dat,grid,model=model,mean=m,
              neigh=unique.neigh,uc="",
              nbtuba=1000,nbsimu=100)
plot(simu,scale=1,col=topo.colors(100),
     name=6,zlim=c(-2,5),pos.legend=5)
```

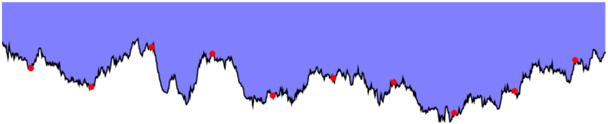
```
m=mean(dat[,5],na.rm=T)
simu = simtub(dat,grid,model=model,mean=m,
              neigh=unique.neigh,uc="",
              nbtuba=1000,nbsimu=100)
plot(simu,scale=1,col=topo.colors(100),
     name=6,zlim=c(-2,5),pos.legend=5)

result=db.compare(simu,fun="mean",name=6:105)
```

```
result0=kriging(dat,grid,uc="",mean=m,
                model=model,neigh=unique.neigh)
```

```
result0=kriging(dat,grid,uc="",mean=m,
                model=model,neigh=unique.neigh)

plot(result[,106],result0[,6],cex=.2)
abline(0,1,col=2)
```

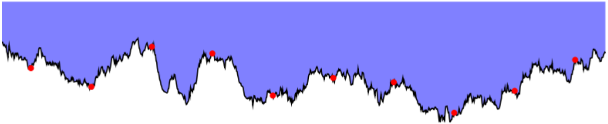
```
result0=kriging(dat,grid,uc="",mean=m,
                model=model,neigh=unique.neigh)

plot(result[,106],result0[,6],cex=.2)
abline(0,1,col=2)

result=db.compare(simu,fun="stdv",name=6:105)
plot(result[,107],result0[,7],cex=.2)
abline(0,1,col=2)
```

Example: a submarine cable has to be set on the seabed between Lisbonne and New-York.
How to predict its length from the bathymetry z sampled every 200m.
Uncertainty?



$$l = \int_a^b \sqrt{1 + [z(x)']^2} dx$$

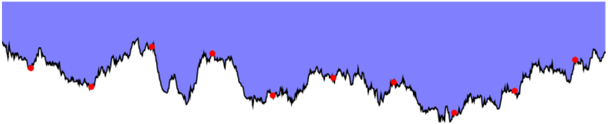
Natural idea



Measuring the length of the predicted (kriged) bathymetry $\hat{z}$

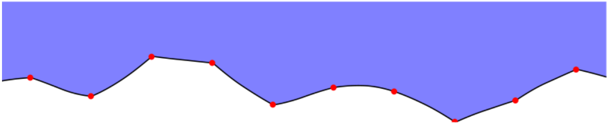
$$\hat{l} = \int_a^b \sqrt{1 + [\hat{z}(x)']^2} dx$$

Natural idea



Measuring the length of the predicted (kriged) bathymetry $\hat{z}$

$$\hat{l} = \int_a^b \sqrt{1 + [\hat{z}(x)']^2} dx$$



Systematic under-estimation: predicted trajectory is much smoother than the actual one

- $(X_0, X_1, \ldots, X_n)$ is a random vector with finite variance.
- The regression of X_0 on $(X_1, \ldots, X_n)$ is the function r such as the mean squared error

$$E[(X_0 - r(X_1, \ldots, X_n))^2]$$

is minimal

- $(X_0, X_1, \dots, X_n)$ is a random vector with finite variance.
- The regression of X_0 on $(X_1, \dots, X_n)$ is the function r such as the mean squared error

$$E[(X_0 - r(X_1, \dots, X_n))^2]$$

  is minimal
- It is given by the conditional expectation of X_0 knowing $X_1, \dots, X_n$:

$$r(x_1, \dots, x_n) = E[X_0 | X_1 = x_1, \dots, X_n = x_n].$$

- $(X_0, X_1, \ldots, X_n)$ is a random vector with finite variance.
- The regression of X_0 on $(X_1, \ldots, X_n)$ is the function r such as the mean squared error
$$E[(X_0 - r(X_1, \ldots, X_n))^2]$$
is minimal
- It is given by the conditional expectation of X_0 knowing $X_1, \ldots, X_n$:
$$r(x_1, \ldots, x_n) = E[X_0 | X_1 = x_1, \ldots, X_n = x_n].$$
- In the multi-Gaussian case, the regression is linear
In other words, simple kriging is optimal

- In other cases, we can compute $r(x_1, \ldots, x_n)$ by averaging conditional simulations of X_0 knowing $X_1 = x_1, \ldots, X_n = x_n$:

$$E[X_0 | X_1 = x_1, \ldots, X_n = x_n] \simeq \frac{1}{N} \sum_{i=1}^{N} X_0^{(i)}$$

where ${X_0}^{(i)}$ is the i^{th} conditional simulation of X_0 knowing $X_1 = x_1, \ldots, X_n = x_n$

- In other cases, we can compute $r(x_1, \dots, x_n)$ by averaging conditional simulations of X_0 knowing $X_1 = x_1, \dots, X_n = x_n$:

$$E[X_0 | X_1 = x_1, \dots, X_n = x_n] \simeq \frac{1}{N} \sum_{i=1}^{N} X_0^{(i)}$$

  where ${X_0}^{(i)}$ is the i^{th} conditional simulation of X_0 knowing $X_1 = x_1, \dots, X_n = x_n$

- For the submarine cable $X_0 \equiv \text{Length}$.

```
data(Exdemo_bathymetry_1D)
dat=Exdemo_bathymetry_1D

v=vario.calc(dat,lag=1,nlag=50)
model=model.auto(v,melem.name(c(5,2)),
                 flag.noreduce=T)

neigh=neigh.init(ndim=1,type=0)
```

Function to compute the length of the cable

```
calcul_length=function(db,index=1){
   x=db.extract(db,"x1")
   z=db.extract(db,paste("z",index,sep=""))
   sum(sqrt((x[-1]-x[-length(x)])^2
       +(z[-1]-z[-length(z)])^2))
   }
```

```
db_res=db.create(nx=10000,dx=1)
db_krig=kriging(dat,db_res,model=model,
                neigh=neigh)
resk=calcul_length(db_krig,1)
```

```
Nbsimus=100
db_simu=simtub(dat,db_res,model=model,
               mean=-1400,neigh=neigh,uc="",
               nbtuba=1000,nbsimu=Nbsimus)
```

```
Nbsimus=100
db_simu=simtub(dat,db_res,model=model,
               mean=-1400,neigh=neigh,uc="",
               nbtuba=1000,nbsimu=Nbsimus)

res=rep(NA,Nbsimus)
for(i in 1:Nbsimus)
     res[i]=calcul_length(db_simu,i)
```

```
hist(res,prob=T,xlim=range(c(resk,res)))
abline(v= calcul_length(db_krig),col=3)
abline(v= mean(res),col=4)
```

```
hist(res,prob=T,xlim=range(c(resk,res)))
abline(v= calcul_length(db_krig),col=3)
abline(v= mean(res),col=4)

plot(dat,xlim=c(0,1000),type="p",cex=.5,col=2)
plot(db_krig,add=T,lwd=2,col=3)
plot(db_simu,add=T,col=4,name=3,lty=2)
```

```
data(Exdemo_bathymetry_1D_full)
grid=Exdemo_bathymetry_1D_full
plot(grid,add=T,col=2)
```

```
data(Exdemo_bathymetry_1D_full)
grid=Exdemo_bathymetry_1D_full
plot(grid,add=T,col=2)

true_length=calcul_length(grid)
```

```
data(Exdemo_bathymetry_1D_full)
grid=Exdemo_bathymetry_1D_full
plot(grid,add=T,col=2)

true_length=calcul_length(grid)

hist(res,prob=T,xlim=range(c(resk,res)))
abline(v= calcul_length(db_krig),col=3)
abline(v= mean(res),col=4)
abline(v=true_length,col=2)
```