Spatial Estimation and Interpolation

Bénédicte Fontez

June 2019

Random Process

A random process, RP Z_s,is defined as a set of usually dependent random variables $Z(s)$, one for each site s in the study area Γ.

$$RP\ Z_s = \{Z(s), \forall s \in \Gamma\}$$

To any set of N sites correspond a vector of N random variates, whose probability (or multivariate cdf) is:

$$F(Z(s_{(1)}), \ldots, Z(s_{(N)})) = Prob(Z(s_{(1)}) \leq z_1, \ldots, Z(s_{(N)}) \leq z_N)$$

The set of all such N-multivariate cdf for any positive integer N constitutes the spatial law of RP Z_s.

In practice, the analysis will be limited to no more than two sites at a time:

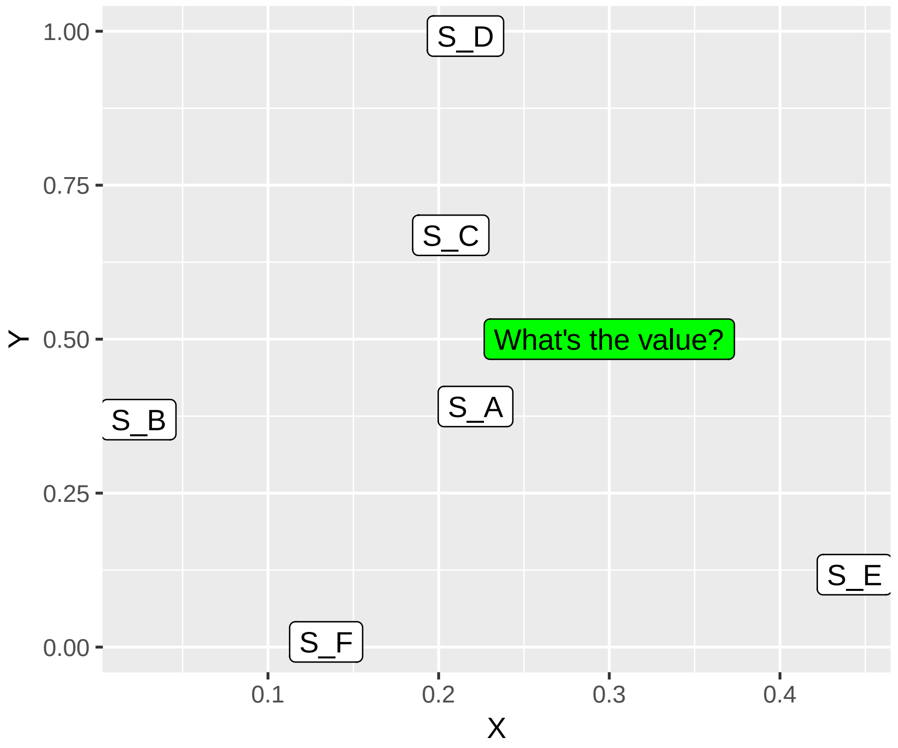
Covariance: $C(s_i, s_j) = E[Z(s_i)Z(s_j)] - E[Z(s_i)]E[Z(s_j)]$

Variogram: $2\gamma(s_i, s_j) = Var[Z(s_i) - Z(s_j)]$

IDW or Inverse Distance Weight

Local Estimation

The variable of interest $Z(s)$ was measured on six different sites $\{s_A, s_B, s_C, s_D, s_E, s_F\}$.



What is the value at a new site?

Principle

The IDW or Inverse Distance Weighted interpolation gives an estimation of $Z(s_{new})$ build as a weighted linear combination of the neighborhood values. The weight is:

- high for sites nearby s_{new}
- low for neighbours far away from s_{new}

Neighborhoods Definition

Several definitions exist, among them:

- $Neighborhood(s_{new})$ = k nearest neighbours/sites (localisation)
- $Neighborhood(s_{new})$ = neighbours/sites inside a circle centered on s_{new} and with a given radius R.

N_{new} will denote the number of sites in the $Neighborhood(S_{new})$ in the following.

Equation of the IDW

Each site s_i of the neighborhood has a weight inversely proportional to the distance between (s_i) and the site to predict/estimate (s_{new}):

$$\hat{Z}(s_{new}) = \frac{\sum_{i=1}^{N_{new}} \frac{Z(s_i)}{dist(s_i, s_{new})}}{\sum_{i=1}^{N_{new}} \frac{1}{dist(s_i, s_{new})}}$$

A broader definition uses a power function of the distance between s_i and s_{new}:

$$\hat{Z}(s_{new}) = \frac{\sum_{i=1}^{N_{new}} \frac{Z(s_i)}{dist(s_i, s_{new})^P}}{\sum_{i=1}^{N_{new}} \frac{1}{dist(s_i, s_{new})^P}}$$

Algorithm

1. Define the neighborhood
2. Create a grid of interpolation
3. Calculate the IDW interpolation at each node of the grid.
4. Plot the interpolation surface on a graph
5. Analyse the sensitivity of the interpolation to the definition/size of the neighborhood

Properties, Limits of the IDW Approach

-The IDW interpolation is an **exact estimation**. It gives the observed values as estimated for the sampled/observed sites.

- The **interpolation surface is continuous**/smooth

- The interpolation does not depend on the site configuration but on the distances between sites.

Example: SIC1997

- Data description

Dataset from the Spatial Interpolation Comparison exercise 1997 (SIC97) *Reference: Journal of Geographic Information and Decision Analysis, vol.2, no.2, pp. 1-11 (1997)*

- 467 daily rainfall measurements made in Switzerland on the 8th of May 1986 were used in SIC97 (*sic_full*).

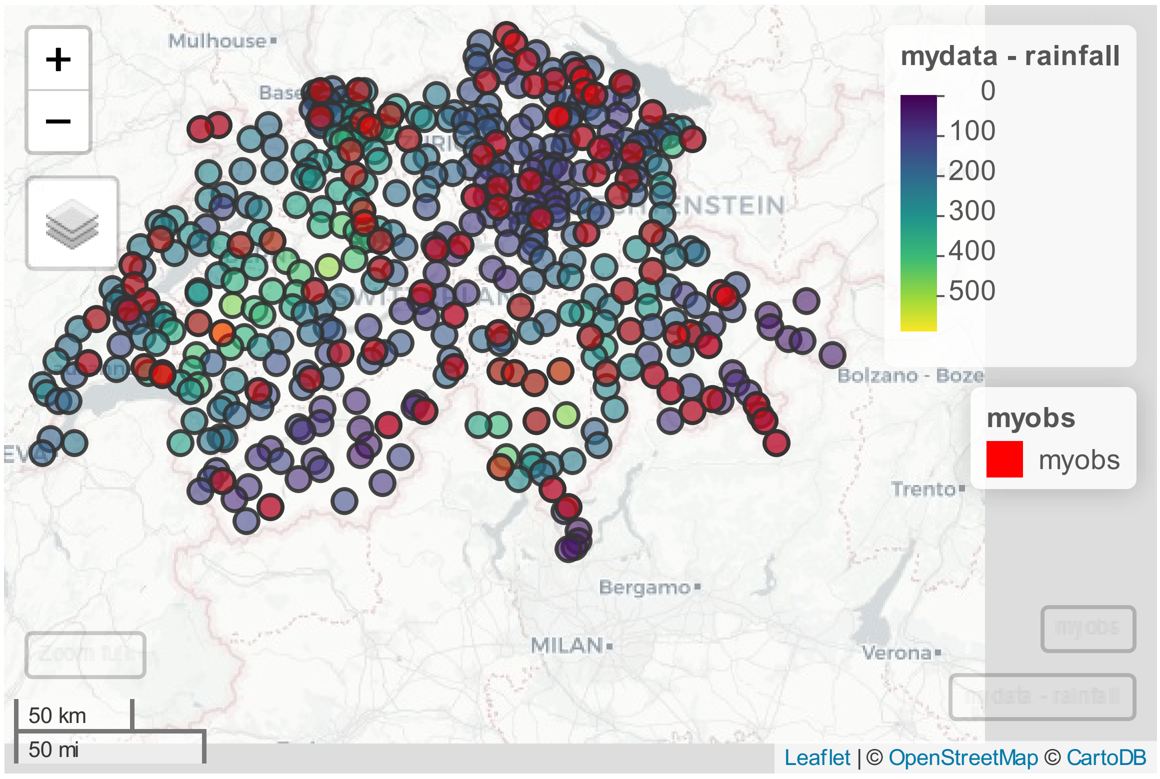
- From them, 100 observed data (*sic_obs*) were used to estimate the rainfall at the remaining 367 locations.

```
data(sic97)

mydata<-st_as_sf(sic_full)
mydataobs<-st_as_sf(sic_obs)
mydata$obs<- mydata$ID %in% mydataobs$ID

crs.suisse<-"+proj=somerc +lat_0=46.95240555555556 +lon_0=7.439583333333333 +k_0=1  +ellps=bessel +towg
st_crs(mydata)<-crs.suisse
myobs<-mydata[mydata$obs,]

mapview(mydata,zcol="rainfall")+mapview(myobs, col.regions="red", alpha.regions=0.5)
```

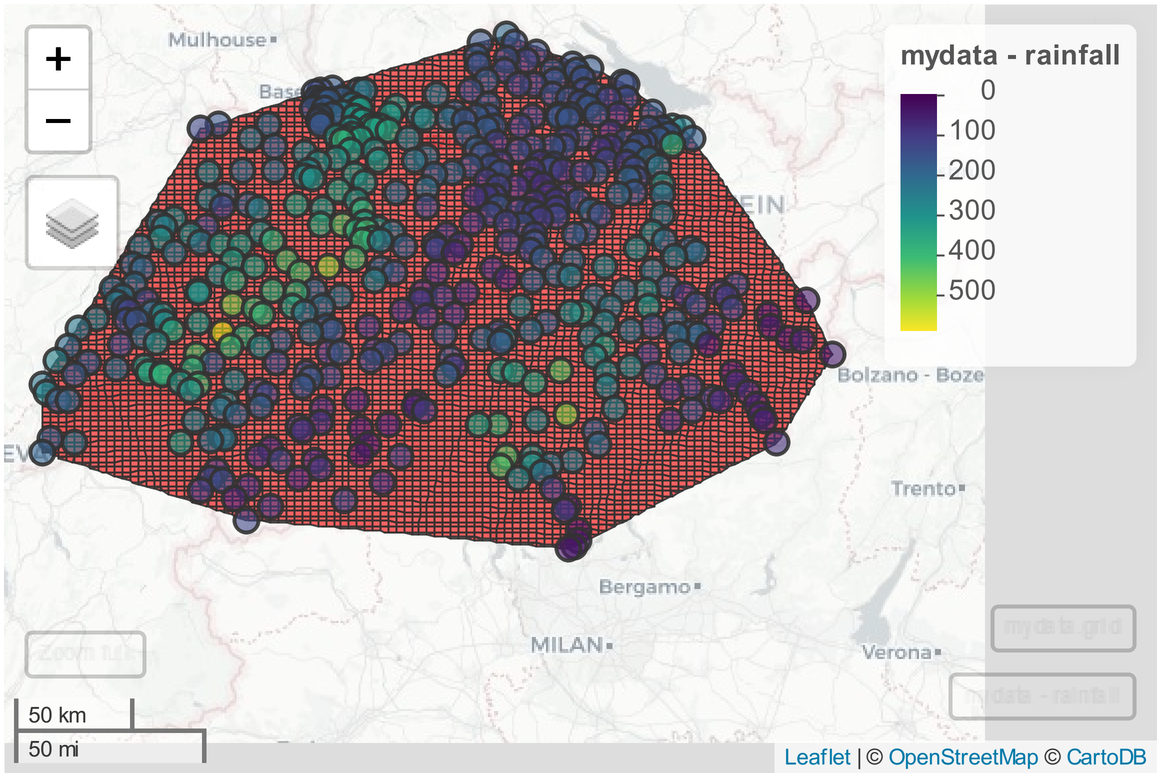


IDW Map

- Grid

```
chull<-mydata %>% st_union() %>% st_convex_hull()
mydata.points<-mydata %>% st_make_grid(n=100) %>% st_intersection(chull) %>% st_centroid()
mydata.grid<-mydata %>% st_make_grid(n=100)  %>% st_intersection(chull)

mapview(mydata,zcol="rainfall")+mapview(mydata.grid, col.regions="red")
```



- Map

```
# IDW interpolation (needs package stars)
obs_idw <- idw(rainfall~1,locations=myobs,newdata=st_sf(mydata.points))
```

```
## [inverse distance weighted interpolation]
```

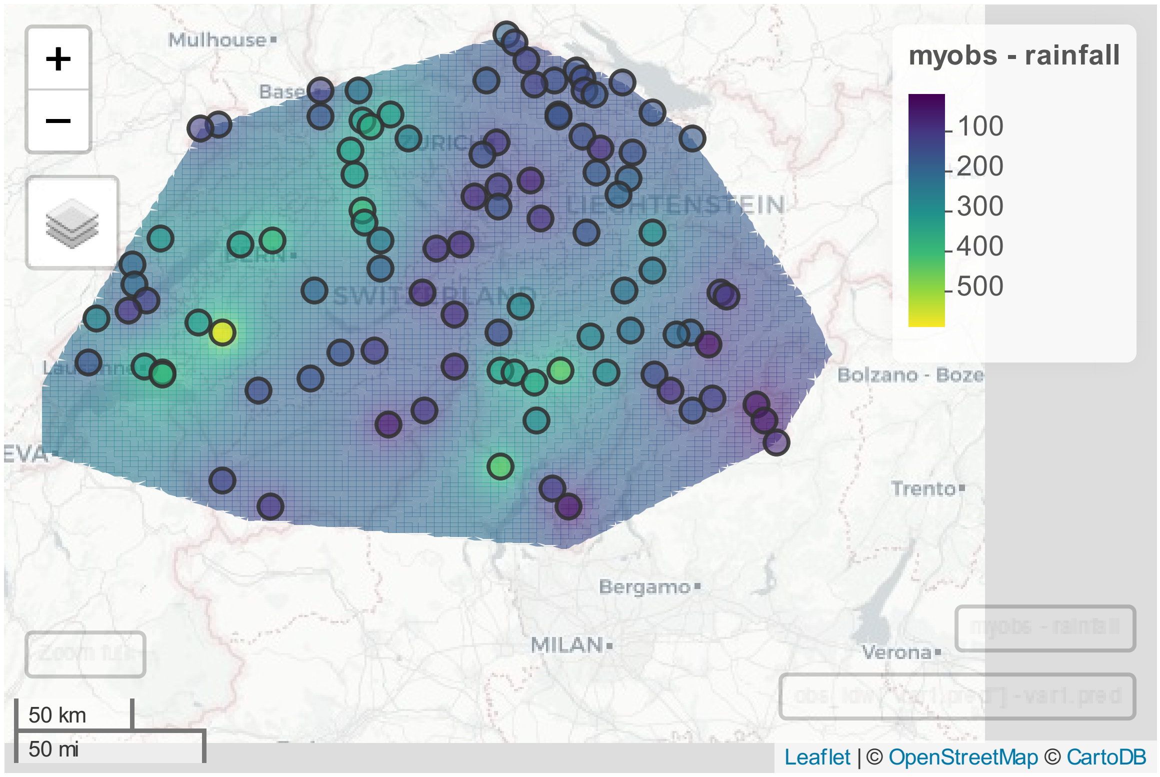
```
st_geometry(obs_idw)<-st_geometry(mydata.grid) # to assign polygons
#obs_idw <- idw(rainfall~1,locations=myobs,mydata)
summary(obs_idw)
```

```
##    var1.pred         var1.var              geometry
##  Min.   : 13.97   Min.   : NA    POLYGON      :7169
##  1st Qu.:136.90   1st Qu.: NA    epsg:NA      :   0
##  Median :173.92   Median : NA    +proj=some...:   0
##  Mean   :186.25   Mean   :NaN
##  3rd Qu.:227.96   3rd Qu.: NA
##  Max.   :566.62   Max.   : NA
##                   NA's   :7169
```

```
full_idw <- idw(rainfall~1,locations=mydata,newdata=st_sf(mydata.points))
```

```
## [inverse distance weighted interpolation]
```

```
#obs_idw["var1.pred"] %>% as_Spatial() %>% spplot()
mapview(obs_idw["var1.pred"], zcol="var1.pred",lwd=0,legend=FALSE)+mapview(myobs,zcol="rainfall")
```



```r
spplot(as_Spatial(obs_idw["var1.pred"]))
spplot(as_Spatial(full_idw["var1.pred"]))
```

Kriging

Principle:

- Characterize the spatial structure of the variable studied by a variogram
- Construct a linear combination that best predicts/estimates the value at a point by taking into account the correlations between points in the neighborhood
- Quantify uncertainty related to prediction (variance of the prediction/estimation)

Characterizing the Spatial Structure: Variogram

The variogram is a method which measures the average 'pattern dissimilarity' between two samples according to their distance.

The experimental variogram is computed from the data and adjusted with a mathematical model defined by three parameters.

- The nugget: the variance between two points at distances smaller than the shortest sampling interval. This variance is due to the measurement error or to the spatial discontinuity.
- The range: distance above which sites are non correlated.
- The sill: The variance between 2 non correlated sites.

The variogram shape is also determined by i) the slope or speed at which destructuring occurs according to distance, and ii) the direction in which the variogram was computed/constructed (depends on the spatial anisotropy).

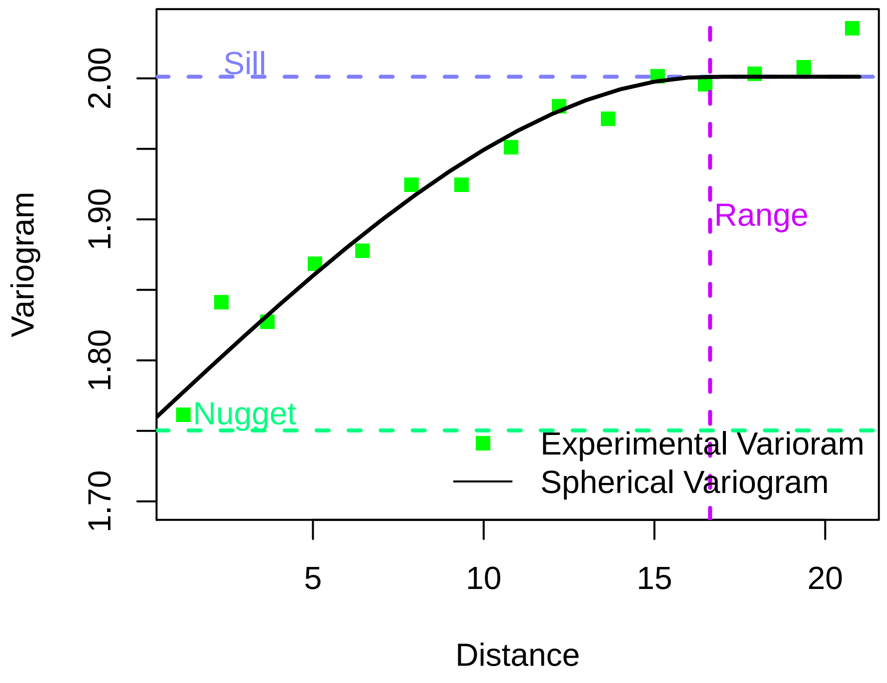


Figure 1: Variogram model parameters

Experimental Variogram

Each point of the experimental semivariogram represents half the variance between two sites separated by a specified distance. As the distance increases the variability between sites is assumed to increase.

- Computation

$$2\hat{\gamma}(h) = \frac{1}{N_h} \sum_{i,j \in Neigh(h)} (Z_{s_i} - Z_{s_j})^2$$

where $Neigh(h) = \{(i,j), \mid si - sj \mid = \mid h \mid\}$ is the set of pairs of points separated by a distance h and N_h is the number of elements in $Neigh(h)$.

Practical aspect: Cut the interval of observed distances ([h_{min},h_{max}]) into bins of the same length.

1. For all pairs (s_i, s_j) compute $V_{i,j} = (Z(s_i) - Z(s_j))^2$
2. Plot $V_{i,j}$ according to the distance $\mid s_i - s_j \mid$
3. Compute the mean value of all the points in each bin to get the experimental variogram point

The experimental variogram is sensitive to the choice of breaks and to h_{max}. Often h_{max} is not the maximum distance but half its value to prevent border effects in the experimental variogram.

Eventually, the experimental variogram can be computed along one or several direction(s), when the processus Z_s is spatially anisotropic.

- Automatic computation with package *gstat*

```r
# package gstat
v <- variogram(rainfall ~ 1, data= myobs)
v
```

```
##     np       dist     gamma dir.hor dir.ver   id
## 1   15   5078.697   554.700       0       0 var1
## 2   68  11926.084  3190.882       0       0 var1
## 3  111  19714.898  3683.126       0       0 var1
## 4  132  27743.181  8626.913       0       0 var1
## 5  142  35528.553  8879.391       0       0 var1
## 6  191  42984.622 11295.016       0       0 var1
## 7  172  50941.385 13502.174       0       0 var1
## 8  211  58613.468 15434.417       0       0 var1
## 9  229  66349.844 14101.290       0       0 var1
## 10 229  74535.224 16060.395       0       0 var1
## 11 225  82127.807 16137.349       0       0 var1
## 12 249  90317.707 14494.484       0       0 var1
## 13 240  97924.235 17336.248       0       0 var1
## 14 281 105896.406 13148.614       0       0 var1
## 15 256 113440.560 10941.543       0       0 var1
```

```r
plot(v)
```

Calibration of the Model Variogram

To choose the best model between the list of possible models, a visual inspection is often enough but some statistical criteria like AIC or the weighted Sum of Squares (WSS) are also used.

to go further, **WSS mathematical definition**:

$$WSS = \sum_{k=1}^{K} w(h_k) \left[\hat{\gamma}(h_k) - \gamma(h_k)\right]^2$$

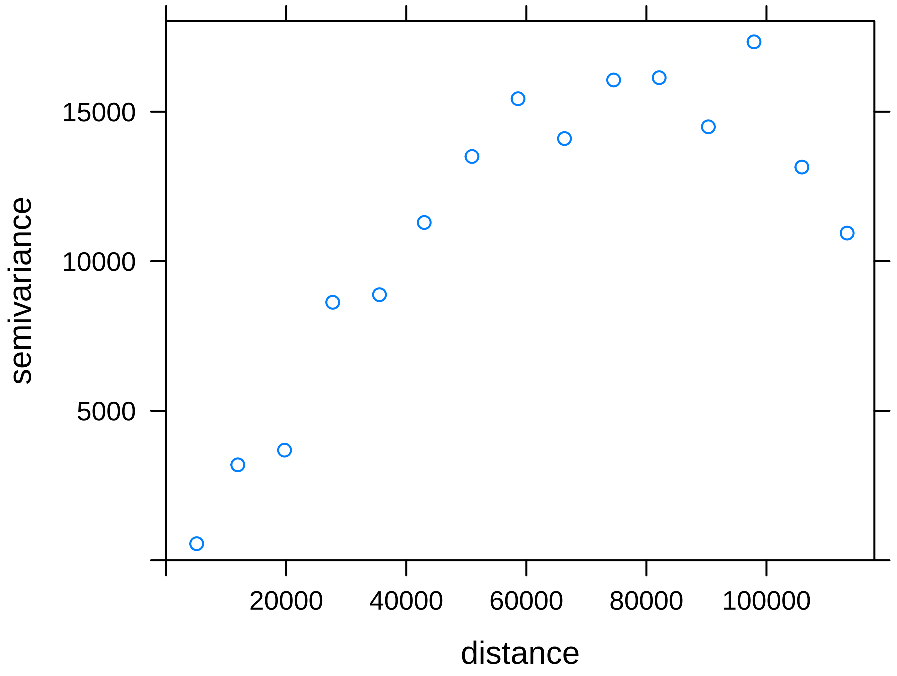


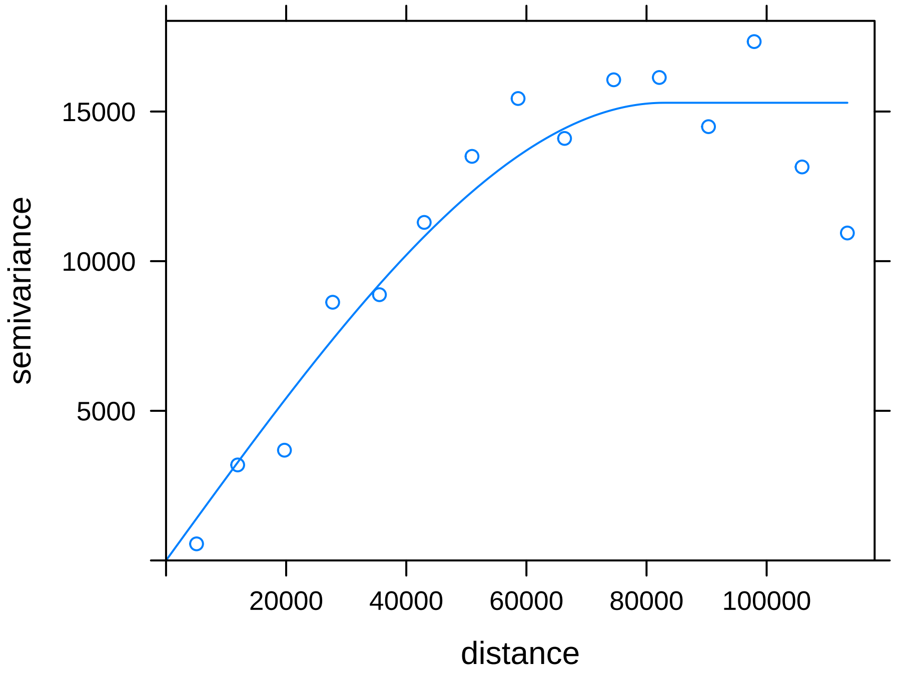
Figure 2: Semivariogram obtained with myobs dataset

where $2\hat{\gamma}(h_k)$ and $2\gamma(h_k)$ are respectively the experimental and the model variogram values for sites separated by a lag/distance h_k. The weight, $w(h_k)$, is usually proportional to the number of site pairs at lag h_k.

```
m.fit <- fit.variogram(v, vgm("Sph"))
m.fit
```

```
##   model    psill    range
## 1   Nug     0.00     0.00
## 2   Sph 15292.38 82946.36
```

```
plot(v,m.fit)
```



Comments on the variogram: For adjacent sites, the half variance (or semivariance) is around 0 and not null. This value is called the nugget effect. For a distance above 8.294636×10^4 the semivariance is stabilized and has attained a sill of 1.529238×10^4.

Stationarity Assumptions

The RP Z_s is assumed stationary to use variogram or kriging statistical tools.

A sufficient condition for the existence of the variogram is the intrinsic stationarity. Often, the RP is assumed to be stationary of order 2 (which implies the intrinsic stationarity).

- Stationarity of order 2:
  - Expected value exists and is constant ($E[Z(s_i)] = m, \forall s_i \in \Gamma$).
  - The covariance function $C(h)$ exists and depends only on the distance h.
- Intrinsic stationarity : Increments $Z(s_i) - Z(s_j)$ are stationary of order 2.

To go further, **mathematical model**: The RP Z_s is usually decomposed into a residual component (random process R_s) and a trend component (deterministic function $m(s)$):

$$Z(s_i) = R(s_i) + m(s_i)$$

The RP R_s is assumed stationary of order 2 with zero mean value.

Three kriging variants can be distinguished according to the function $m(s)$ used for the trend.

1. Simple Kriging (constant known mean value): $m(s_i) = m$, $\forall s_i \in \Gamma$
2. Ordinary Kriging (locally constant unknown mean): $\forall s_j \in Neighborhood(s_{new})$, $m(s_j) = m(s_{new})$
3. Universal kriging (non constant unknown mean)

Ordinary Kriging

Estimator (definition and properties)

The value at a new site $\hat{Z}(s_{new})$ is estimated from the samples in its neighborhood $Z(s_1), Z(s_2), \ldots, Z(s_{N_{new}})$ by a linear combination:

$$\hat{Z}(s_{new}) = \lambda_1(s_{new})\, Z(s_1) + \lambda_2(s_{new})\, Z(s_2) + \ldots + \lambda_{N_{new}}(s_{new})\, Z(s_{N_{new}})$$

where $\lambda_i(s_{new})$, $1 \leq i \leq N_{new}$, are determined by solving the system of equations corresponding to these two assumptions:

1) The expected value of the estimator is not biased.
2) The variance of the estimator is minimal.

The first assumption implies the following constraint: $\sum_{i=1}^{N_{new}} \lambda_i(s_{new}) = 1$

Each predictor $\hat{Z}(s_{new})$ is a random variable with a variance, often called the kriging variance.This variance depends only on the model variogram and on the spatial pattern of the sites (those observed and those to predict). Therefore this variance does not depend on the observed values $Z(s_1), Z(s_2), \ldots, Z(s_{N_{new}})$. The kriging variance is optimal for a Gaussian RP Z_s.

To go further **the ordinary kriging system**:

1. Remember that $\forall s_j \in Neighborhood(s_{new})$, $m(s_j) = m(s_{new})$, so:

$$\begin{aligned}\hat{Z}(s_{new}) &= \sum_{i=1}^{N_{new}} \lambda_i(s_{new})\,(Z(s_i) - m(s_{new})) + m(s_{new}) \\ &= \sum_{i=1}^{N_{new}} \lambda_i(s_{new})\, Z(s_i) + [1 - \sum_{i=1}^{N_{new}} \lambda_i(s_{new})]m(s_{new})\end{aligned}$$

   The unknown local mean value is filtered from the linear estimator when the kriging weights sum to 1.

2. The variance of the estimator is:

$$\begin{aligned}var[\hat{Z}(s_{new}) - Z(s_{new})] &= \sum_{i=1}^{N_{new}} \sum_{j=1}^{N_{new}} \lambda_i(s_{new})\, \lambda_j(s_{new})\, C(s_i, s_j) + C(0) \\ &- 2 \sum_{i=1}^{N_{new}} \lambda_i(s_{new})\, C(s_i, s_{new})\end{aligned}$$

   The minimization of this error variance under the non bias condition leads to the following system of linear equations:

$$\begin{array}{ll}\sum_{i=1}^{N_{new}} & \lambda_i(s_{new})\, \gamma(s_i, s_j) - m(s_{new}) = \gamma(s_i, s_{new}),\ i = 1, \ldots, N_{new} \\ \sum_{i=1}^{N_{new}} & \lambda_i(s_{new}) = 1\end{array}$$

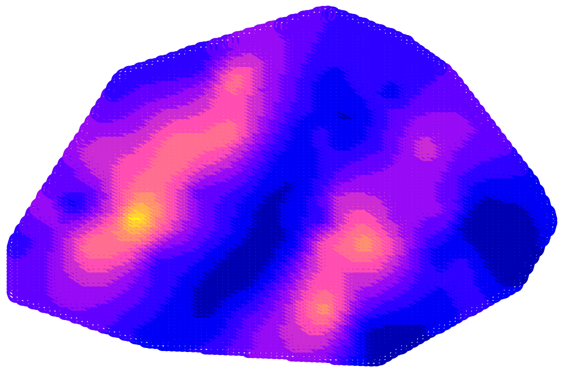
Kriging Map

```
NF.kriged = krige(rainfall ~ 1, locations=myobs, newdata=mydata.points, model = m.fit)
```

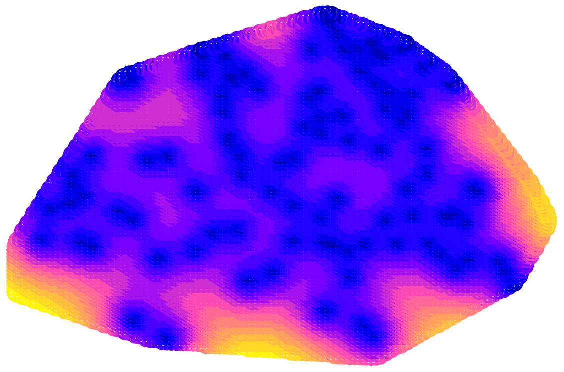
```
## [using ordinary kriging]
```

```
plot(NF.kriged)
```

var1.pred



var1.var



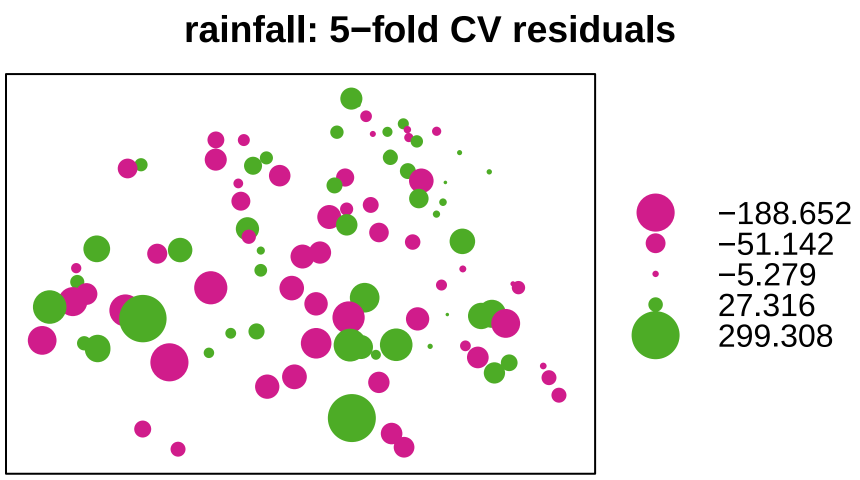
Cross Validation

krige.cv

Cross validation functions for simple, ordinary or universal point (co)kriging, kriging in a local neighbourhood.

Leave-one-out cross validation (LOOCV) visits a data point, and predicts the value at that location by leaving out the observed value, and proceeds with the next data point. (The observed value is left out because kriging would otherwise predict the value itself.) N-fold cross validation makes a partitions the data set in N parts. For all observation in a part, predictions are made based on the remaining N-1 parts; this is repeated for each of the N parts. N-fold cross validation may be faster than LOOCV.

```
NF.kriged.cv = krige.cv(rainfall ~ 1, locations=myobs, model = m.fit, nmax=20, nfold=5)
bubble(as_Spatial(NF.kriged.cv), "residual", main = "rainfall: 5-fold CV residuals")
```



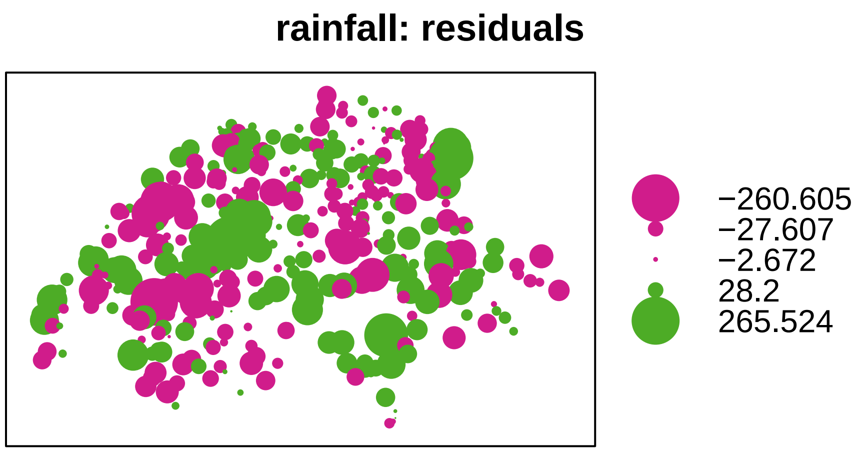
Prediction for Rainfall Data (SIC97)

- Residuals

```
#sic_nobs <- sic_full[-(1:100),]
NF.kriged.nobs = krige(rainfall ~ 1, myobs, model = m.fit, newdata=mydata[!mydata$obs, ],nmax=20)
```

```
## [using ordinary kriging]
```

```
NF.kriged.nobs$residuals <- mydata[!mydata$obs,]$rainfall-NF.kriged.nobs$var1.pred
bubble(as_Spatial(NF.kriged.nobs), "residuals", main = "rainfall: residuals")
```

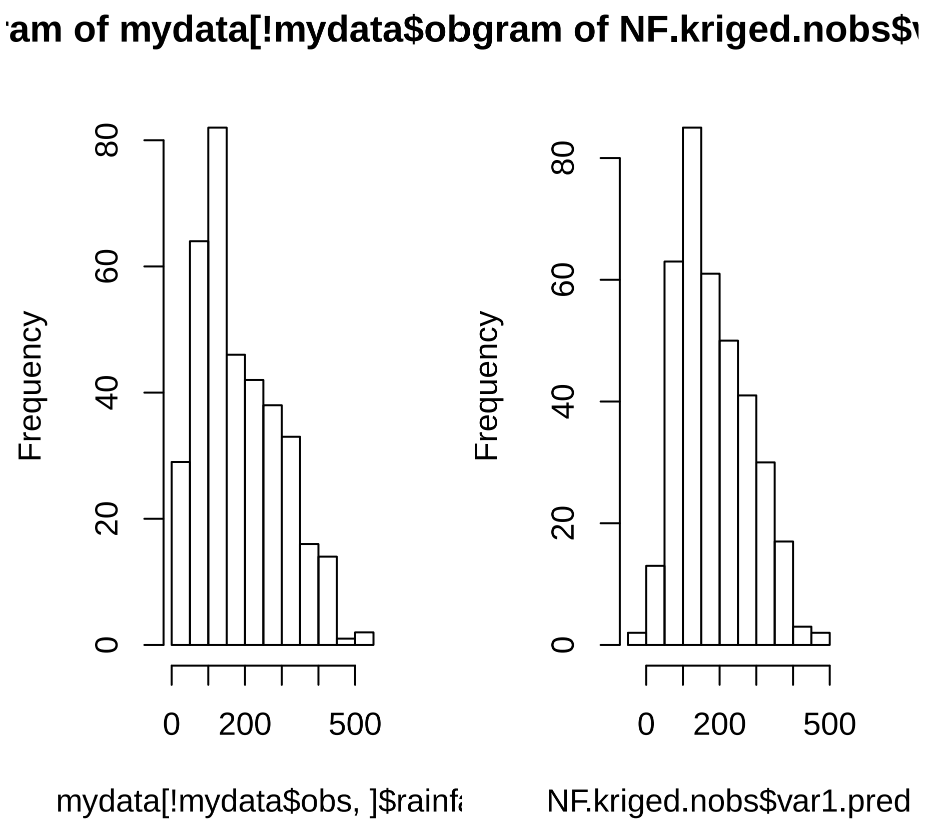


- Histogram of true and predicted values

```
summary(data.frame(obs=mydata[!mydata$obs,]$rainfall,pred=NF.kriged.nobs$var1.pred))
```

```
##       obs              pred
##  Min.   :  0.0   Min.   : -1.695
##  1st Qu.:100.0   1st Qu.:111.374
##  Median :162.0   Median :165.489
##  Mean   :185.4   Mean   :182.518
##  3rd Qu.:263.5   3rd Qu.:251.753
##  Max.   :517.0   Max.   :487.654
```

```
par(mfrow=c(1,2))
hist(mydata[!mydata$obs,]$rainfall)
hist(NF.kriged.nobs$var1.pred)
```



Kriging gives a smooth prediction with less dispersion of the rainfall values.

- Correlation between true and predicted values

```
#st_data<-function(SF){st_geometry(SF)<-NULL; return(SF)}
NF.kriged.nobs$rainfall<-mydata[!mydata$obs,]$rainfall

cor(NF.kriged.nobs$rainfall,NF.kriged.nobs$var1.pred)
```

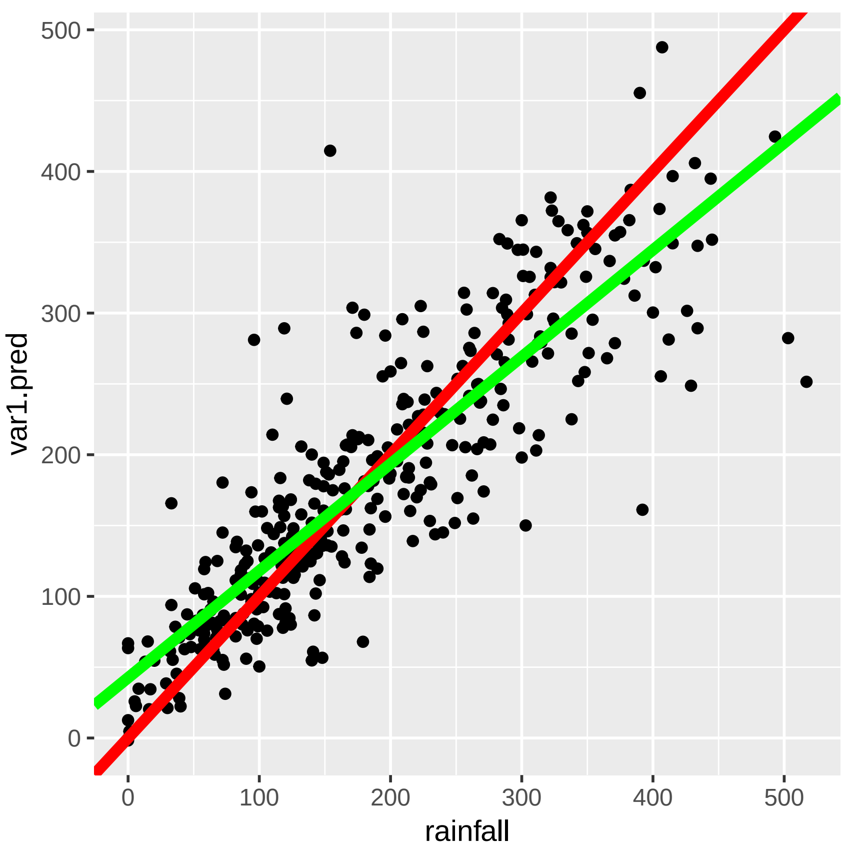
```
## [1] 0.8657555
```

```
var(NF.kriged.nobs$residuals)
```

```
## [1] 3095.841
```

```
coef <- coef(lm(var1.pred~rainfall,data=NF.kriged.nobs))

ggplot(NF.kriged.nobs,aes(x=rainfall,y=var1.pred)) +
  geom_point() +
  geom_abline(slope =1, intercept = 0, col="red",size=2) +
  geom_abline(slope =coef[2], intercept = coef[1], col="green",size=2)
```



Comparison with IDW Approach

The neighborhood can be defined in the same way for kriging and IDW. But:

- **IDW**: each site has a weight inversely proportional to the distance to the site to predict (s_{new}).
- **Kriging**: The weighting is built through the variogram model and the spatial pattern of the sites.

For both interpolators, the value on a new site is estimated by a weighted linear combination of the neighbors sites.

Universal Kriging or Kriging with a Trend

Universal Kriging assumes a linear or quadratic trend (where spatial coordinates could be used as explanatory variables).

$$Z(s_i) = m(s_i) + R(s_i)$$

with

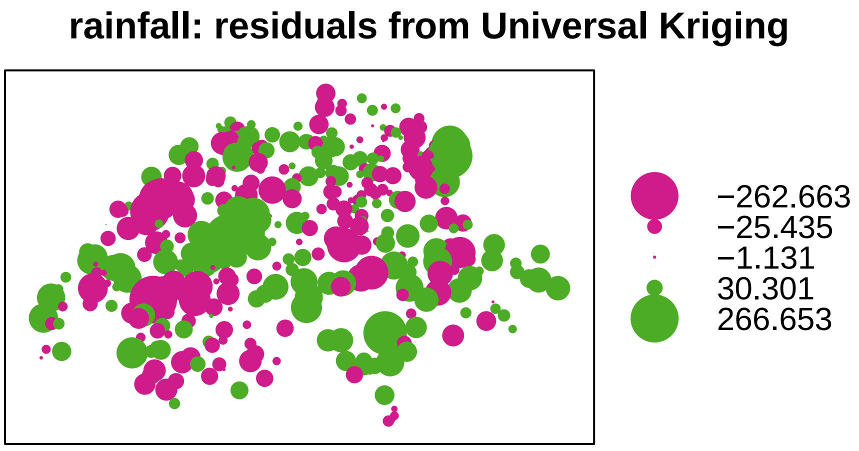
1. Linear trend $m(s_i) = \beta_0 + \beta_1 x_i + \beta_2 y_i$
2. Quadratic trend $m(s_i) = \beta_0 + \beta_1 x_i + \beta_2 y_i + \beta_3 x_i^2 + \beta_4 y_i^2 + \beta_5 x_i\, y_i$

In the rainfall example, the universal kriging brings no improvement.

```
st_add_xy<-function(SF){xy<-SF %>% st_centroid() %>% st_coordinates(); SF$x<-xy[,1]; SF$y<-xy[,2]; retu
NF.kriged.UK <- krige(rainfall ~ x+y, locations=st_add_xy(myobs), model = m.fit, newdata=st_add_xy(myda
```

```
## [using universal kriging]
```

```
residuals <- mydata[!mydata$obs,]$rainfall-NF.kriged.UK$var1.pred
NF.kriged.UK$residuals<-residuals
bubble(as_Spatial(NF.kriged.UK), "residuals", main = "rainfall: residuals from Universal Kriging")
```



Practical recommendation:

Do a Universal kriging or Estimating the trend and computing simple kriging predictions of the residuals is equivalent to Universal Kriging when a linear trend is assumed (Cressie, 1993, section 3.4.5).So:

1. Estimate the trend with a regression.
2. Compute the residuals
3. Carry out the variogram estimation and kriging on the residuals but use the **Simple Kriging**!
4. Add the trend to the kriging estimates

Sequential Gaussian Simulation

Goal: Estimate a characteristic or parameter of the RP Z_s, for example a probability map.

Principle

kriging gives an estimate of both the mean value and standard deviation of the normal (Gaussian) variable at each grid node.

Sequential Gaussian Simulation replaces the kriging mean value by a random draw from this normal distribution.

More details can be found in the book from Goaverts

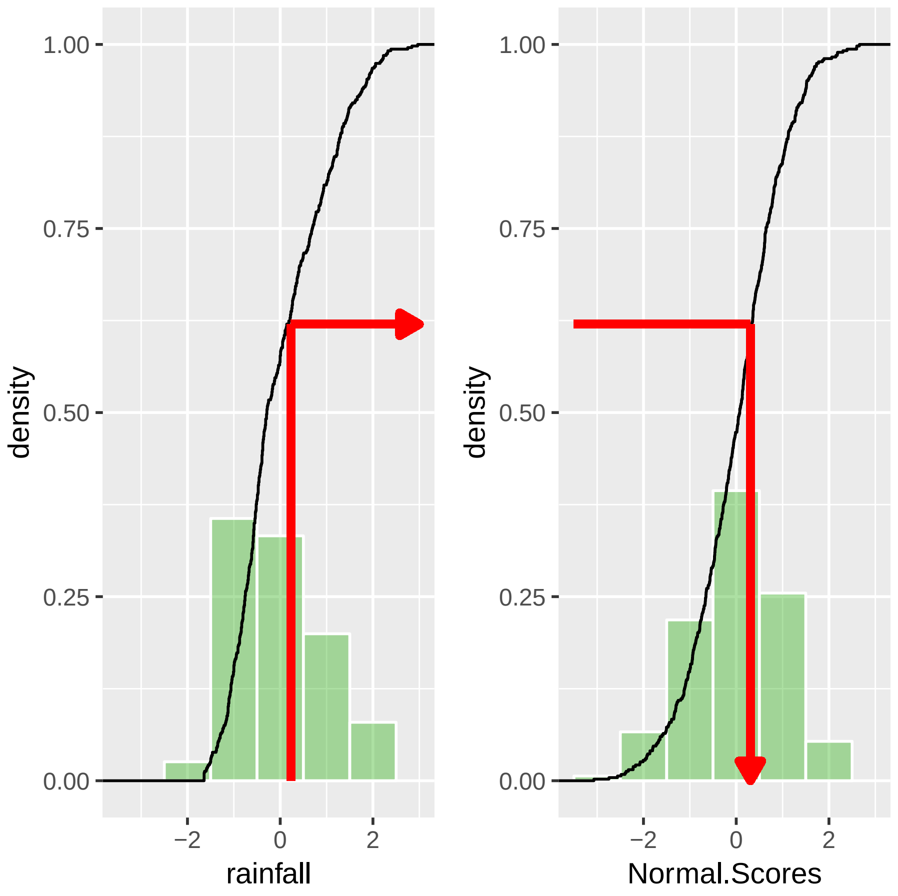


Figure 3: Illustration of the Normal Score Transformation for the Rainfall dataset

Normal Score Transformation for the Rainfall Example

When data are not Normally distributed, the data can be transformed into normal scores before doing the sequential Gaussian simulation on the normal scores. First, the rainfall dataset is normalized (centered and divided by its standard deviation).

We can see from figure 3 that a rainfall of 210 millimeters corresponds to a scaled value of $(210-184)/112 = 0.23$ and a probability of 0.62. The normal quantile (or Normal Scaled Score) for this probability is 0.30. This empirical quantile (210) to normal quantile (0.30) transformation preserves the rank of an observation and therefore the probability level.

Algorithm of the SGS

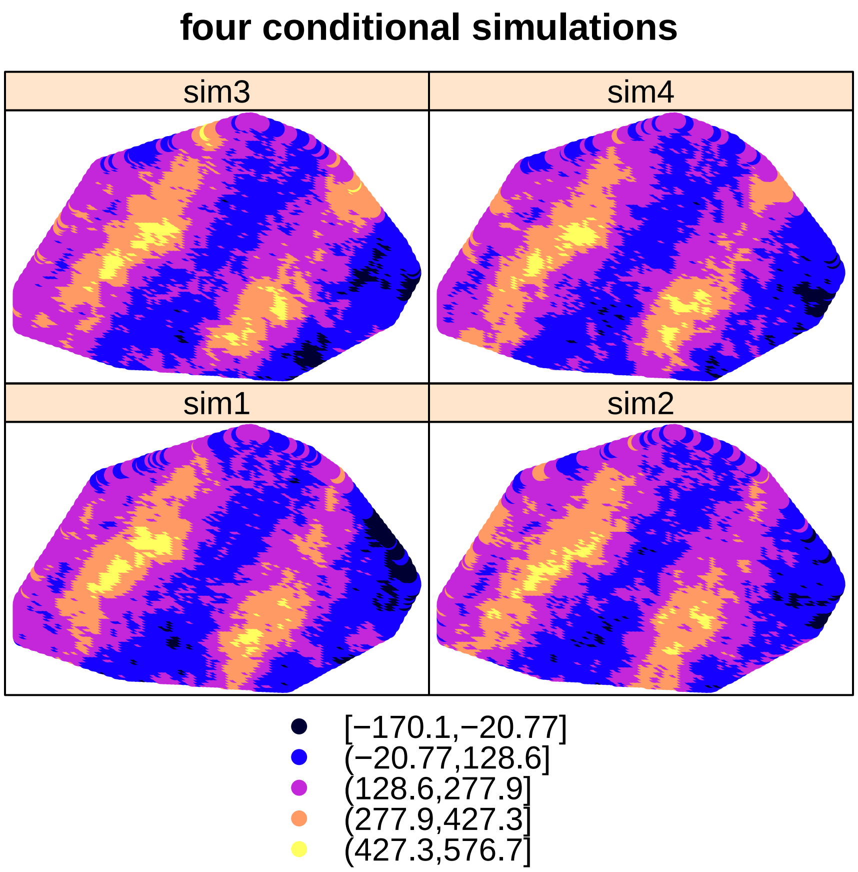
1. Transform data to normal scores
2. Perform a variogram analysis on the normal scores
3. Create a grid and generate a random path through the grid nodes
4. Use kriging to estimate a mean value and standard deviation at the first node
5. Set the variable value at that node from the random draw
6. Repeat for next nodes, including previously simulated nodes as data values in the kriging process

The previously simulated grid nodes are included as *data* in order to preserve the proper covariance structure between the simulated values.

```
NF.kriged.sim = krige(rainfall ~ 1, locations=mydata, newdata= mydata.points, model = m.fit, nmax=20, n
```

```
## drawing 4 GLS realisations of beta...
## [using conditional Gaussian simulation]
```

```
spplot(as_Spatial(NF.kriged.sim), main = "four conditional simulations")
```



To Go Further: Co-Kriging

When non exhaustive secondary information is available, it can be incorporated into the estimator using cokriging approach. This approach takes into account the secondary variables (from the RP $Z_{j,s}$ N_j, $j = 2, ..., p$) and their spatial cross correlation with the primary variable (from the RP $Z_{1,s}$). The secondary data can be possibly at different sites.

$$\begin{aligned}\hat{Z}_1(s_{new}) - m_1(s_{new}) = & \sum_{i=1}^{N_{1,new}} \lambda_{1,i}(s_{new})\,(Z_1(s_{1,i}) - m_1(s_{1,i})) \\ & + \sum_{j=1}^{p}\sum_{i=1}^{N_{j,new}} \lambda_{j,i}(s_{new})\,(Z_j(s_{j,i}) - m_j(s_{j,i}))\end{aligned}$$

All cokriging estimators required to be unbiased $E[\hat{Z}_1(s_{new}) - Z_1(s_{new})] = 0$ and to minimize the error variance $Var[\hat{Z}_1(s_{new}) - Z_1(s_{new})]$.

Each RP $Z_{j,s}$ is decomposed into a residual and a trend components:

$$Z_{j,s} = R_{j,s} + m_j(s),\ j = 1, \ldots, p$$

The residual component $R_{j,s}$ is modeled as a stationary RP with zero mean value and:

1. Covariance function: $Cov[R_j(s), R_j(s+h)] = C_j(h)$
2. Cross covariance function: $Cov[R_j(s), R_k(s+h)] = C_{jk}(h)$

Aknowledgment

Spatial data management was done with package `sf` with great help from Manuel Campagnolo (see its lecture notes for more details).

References

- M. Arnaud X. Emery (2000). Estimation et interpolation spatiale: méthodes déterministes et méthodes géostatistiques. Hermès.
- Cressie N. (1992). Statistics for spatial data. Wiley.
- Govaerts P. (1997). Geostatistics for natural resources evaluation. Oxford University Press.
- Webster R. et Oliver M.A. (1990). Statistical Methods in Soil and Land Resource Survey. Oxford University Press.

Practical Work: Co-Kriging with Rainfall Data

Georeferencing of the Rainfall Data

1. Download the elevation tiles (SRTM data) with locations in longitude/latitude
2. Map the tiles
3. Extract elevation values at rainfall locations

Co-Kriging

1. Building a dataset for experimental variogram(s)
2. Plot and fit variogram(s)
3. Predict values
4. Do an interpolation map

Exercise - Example Applied to Environmental Data

The data used in this tutorial comes from the article Kriging in estuaries: as the crow flies, or as the fish swims?, Laurie S. Little, Don Edwards, Dwayne E. Porter, Journal of Experimental Marine Biology and Ecology,213 (1997) 1-11.

Their study evaluated the relative accuracy of eight kriging methods for predicting contaminant and water quality variables measured in an urbanized estuary in South Carolina. The variable of interest was Fluoranthene concentration in oyster tissue was normalized by dividing by the proportion of lipid in tissue, as determined by the proportion of methylene chloride extracted solids.

Launch R

Load package *gstat* Load package *sp*

Data Importation

Download the file *data.txt* from Ticea

VARIABLE OF INTEREST : don$NormFluo or don[,11]

1. Spatialize your data with function *coordinates()*
2. Draw the data with function *bubble()*

Grid

1. Create a grid and plot it
2. Add points of observation on the plot

IDW Mapping

1. Apply function *idw()*
2. Draw the interpolation map with function *ssplot()*

Experimental Variogram

1. Compute the experimental variogram with function *variogram()*

ISOTROPY:

2. Draw the experimental variogram and comment

ANISOTROPY:

3. Specify the angles in which the experimental variogram will be computed (parameter alpha). Save this (these) variogram(s) under the name *obs.vgma* in *R*
4. Draw the experimental variogram(s) and comment

Model Variogram

Remark: to get the list of all possible models, write: *vgm()*

1. Try this command: *anis.vgm = vgm(psill=10000,model="Sph", range=6000,nugget=10, anis = c(60, 0.2))*

2. Draw the experimental and this spherical variograms on the same graph, comment

3. Apply the automatic fitting: *mod.vgma <- fit.variogram(obs.vgma,model=anis.vgm)*

4. Draw experimental and estimated spherical variograms on the same graph, comment

Kriging Map

1. Estimate the map by kriging on the grid (oridnary kriging) with function *krige()*
2. Map the prediction
3. Map the variance of the error of prediction
4. Comment differences between idw and kriging/ed maps (with and without nugget effects)