Point patterns

- Consider the point process $\{Z(s) : \mathbf{s} \in D \subset R^2\}$. A realization of this process consists of a pattern (arrangement) of points in D. (D is a random set.) These points are called the events of the point process. If we only partially observed these events then we have a sampled point pattern. When all events are observed we have a mapped point pattern.
- A point pattern is called completely random pattern if the following criteria hold:
  - The average number of events (the intensity, $\lambda(s)$) is homogeneous throughout D.
  - The number of events in two non-overlapping subregions A_1 and A_2 are independent.
  - The number of events in any subregion follows the Poisson distribution.
- With this in mind... observed point patterns are tested initially for a complete spatial random pattern. If the hypothesis is rejected then further investigation is needed to explain the nature of the spatial point pattern.
- Most processes don't follow a complete spatial random pattern. Events may be independent in non-overlapping subregions, but the intensity $\lambda(s)$ is not homogeneous throughout D. For example, more events will be present in regions where the intensity is high and less will be present where the intensity is low. The intensity may be constant, but the presence of an event can attract or drive away other events nearby.
- `R` packages for the analysis of spatial point patterns:

`spatial`
`splancs`
`spatstat`
`maptools`

See Chapter 7 of Bivand, R.S., Pebesma, E.J., Gómez-Rubio, V. (2008). "Applied Spatial Data Analysis with R," Use R!, Springer.

- Preliminary analysis of a point pattern:
  It is focused on the spatial distribution of the observed events to make inference on the process that generated them. We are interested in (a) the distribution of the events in space and (b) existence of possible interaction between them.

- Poisson process:
  There are many types of Poisson processes: Homogeneous Poisson process (HPP), inhomogeneous Poisson process (IPP), Poisson cluster process, and compound Poisson process. A process is called homogeneous Poisson process if the two following properties hold:

  – If $N(A)$ denotes the number of events in subregion $A \subset D$, then $N(A) \sim \text{Poisson}(\lambda v(A))$, where $0 < \lambda < \infty$ is the constant intensity of the process.
  – If A_1 and A_2 are two disjoint subregions of D, then $N(A_1)$ and $N(A_2)$ are independent.

  If the intensity function $\lambda(s)$ varies spatially then the first condition does not hold, but the second condition may still hold. In this case the process is called inhomogeneous Poisson process. (The homogeneous Poisson process is a special case of the inhomogeneous Poisson process.) The homogeneous Poisson process is also called the stationary Poisson process, while

the inhomogeneous Poisson process is called the non-stationary Poisson process.

- Testing for complete spatial randomness:
  We want to test if the observed point pattern is a realization of a homogeneous Poisson process. The statistical tests are based on counts of events in regions (quadrats) or based on distances. When the sampling distributions are difficult one can rely on simulations methods. There are two methods of simulations: The Monte Carlo test and simulation envelopes.

- Test based on quadrats:
  With quadrat sampling we count the number of events in subsets of the study region A. Usually the quadrats are rectangular, but other shapes are also possible (e.g. circular).
  Example
  The data set `longleaf`: 584 long-leaf pine trees from the Wade Tract, a forest in Thomas County Georgia. The data consists of the location of each tree (x, y) coordinates and its diameter at breast height (`dbh`) in centimeters. Area covered 200m $\times$ 200m. One hundred non-overlapping quadrats (each one with radius 6 meters) were randomly chosen in the area of study and the number of trees were counted in each quadrat. (See Cressie, "Statistics for Spatial Data," Wiley, 1993, pp. 581-585.)

```
library(spatstat)
data(longleaf)
```

Trees per quadrat	Observed frequency	Estimated frequency
0	34	23.93
1	33	34.22
2	17	24.47
3	7	11.66
4	3	4.17
5	1	1.19
6	1	0.28
7	2	0.06
8	1	0.01
9	0	0.00
10	1	0.00

Estimate Poisson parameter $\hat{\lambda} = \frac{34\times0+33\times1+...+1\times10}{100} = 1.43$. The estimated frequencies are computed using the Poisson probability mass function. For example, the expected number of quadrats with zero trees will be $100 \times P(Y = 0) = 100\frac{1.43^0 exp(-1.43)}{0!} = 23.93$. To test the hypothesis of a complete spatial randomness (which is synonymous with the homogeneous Poisson process) one can use the χ^2 goodnes-of-fit test.

$$X^2 = \sum_{i=1}^{6} \frac{(O_i - E_i)^2}{E_i} = \frac{(34 - 23.93)^2}{23.93} + \ldots + \frac{(6 - 1.54)^2}{1.54} = 21.67.$$

Since $21.67 > \chi^2_{0.95;4} = 9.49$ the null hypothesis of homogeneous Poisson process is rejected.

- Other uses of quadrats:
  See paper by Greig-Smith (1952). "The Use of Random and Contiguous Quadrats in the Study of the Structure of Plant Communities."

The grouping of events into contiguous quadrats creates a lattice. Lattice data is another type of spatial data where the goal is to identify regions with high values in close proximity suggesting a cluster. The Moran's I and Geary's c statistics can be applied to test for clustering.

- Once complete spatial randomness is rejected, the next step is to see if regularity (tendency for regular spacing) or clustering is present.

- Test based on distances:
  Distance methods use the exact location of the events. They do not dependent on the arbitrary choice of quadrat size or shape. See Cressie (1993), p. 604 for various test statistics based on distances along with their asymptotic distributions.

- Test based on simulations:
  - Monte Carlo tests
    These can be used for many statistical analysis spatial or non-spatial. The general idea: Compute a test statistic from the observed data, call it q_0. Then simulate the random process say, g times. For each realization compute the test statistics $q1, \ldots, q_g$. Then rank the simulated test statistics and place the observed test statistic q_0 in the ordered array and compute the p-value. For example the average nearest neighbor distance can be used. It can be compute using simulations by generating points independently and uniformly in the area of interest.

– Simulation envelopes:
Find the nearest neighbor distance for event $i = 1, \ldots, n$. Let $d_1, d_2, \ldots, d_n$ be the nearest neighbor distances.

Compute the estimate of the distribution function $\hat{G}(d)$ of nearest-neighbor event distances as follows:
Let $I(d_i \leq r)$ be the indicator function that takes the value 1 if $d_i \leq r$. Compute $\hat{G}(r) = \frac{1}{n} \Sigma_{i=1}^{n} I(d_i \leq r)$ for various distance r.

For the same distances r compute the theoretical G function under complete spatial randomness which equal to $G(r) = 1 - exp(-\lambda \pi r^2)$.

Plot $\hat{G}(r)$ against $G(r)$. Under complete spatial randomness the plot should be roughly linear.

To measure the departure from linearity we should find the sampling distribution of $\hat{G}(r)$ under complete spatial randomness, which not easy, because of the dependence between the distances (for example, if the nearest neighbor for point 1 is point 2 then the nearest neighbor for point 2 will be point 1, and so on). We therefore assess linearity using simulations. We compute $\hat{G}(r)$ for many simulations. Each simulation consists of n independent uniformly distributed points in the area of interest. For each simulation we compute the minimum and maximum value of $\hat{G}(r)$ to construct the simulation envelope.

Overview of geostatistics

- Let $Z(s)$ and $Z(s+h)$ two random variables at locations s and $s+h$. Intrinsic stationarity is defined as follows:

$$E(Z(s+h) - Z(s)) = 0$$

and

$$Var(Z(s+h) - Z(s)) = 2\gamma(h)$$

The quantity $2\gamma(h)$ is known as the variogram and is very crucial in geostatistics. The variogram says that differences of variables lagged h-apart vary in a way that depends only on h through the length of h. This is called *isotropic* variogram as opposed to *anisotropic* variogram which depends not only on the length h but also the direction. Because of the assumption of constant mean (no trend) we have $E(Z(s)) = \mu$ and we can write

$$\text{Var}(Z(s+h) - Z(s)) = E(Z(s+h) - Z(s))^2$$

Therefore we can use the method of moments estimator for the variogram (also called the classical estimator):

$$2\hat{\gamma}(h) = \frac{1}{N(h)} \sum_{N(h)} (Z(s_i) - Z(s_j))^2,$$

where the sum is over $N(h)$ such that $s_i - s_j = h$.

Robust estimator:
Cressie and Hawkins (1980) proposed the following estimator for the variogram which is robust to outliers compared to the classical estimator:

$$2\bar{\gamma}(h) = \frac{\left\{ \frac{1}{N(h)} \sum_{N(h)} |Z(s_i) - Z(s_j)|^{\frac{1}{2}} \right\}^4}{0.457 + \frac{0.494}{N(h)}}$$

where the sum is over $N(h)$ such that $s_i - s_j = h$.

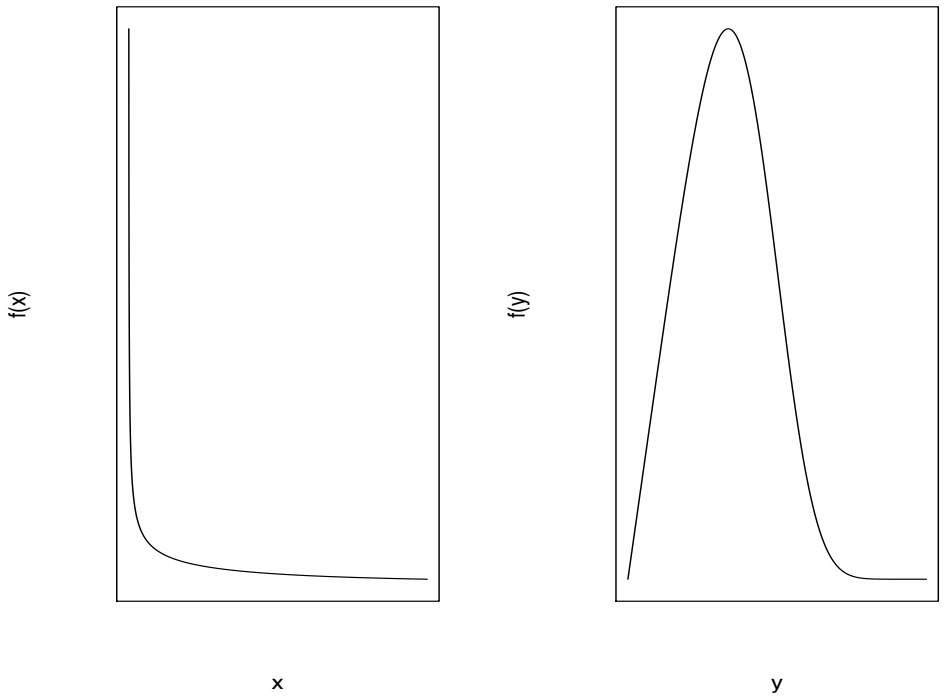
The idea behind the robust estimator is described below: If the process $Z(s)$ follows the normal distribution then

$$Z(s+h) - Z(s) \sim N\left(0, \sqrt{2\gamma(h)}\right)$$

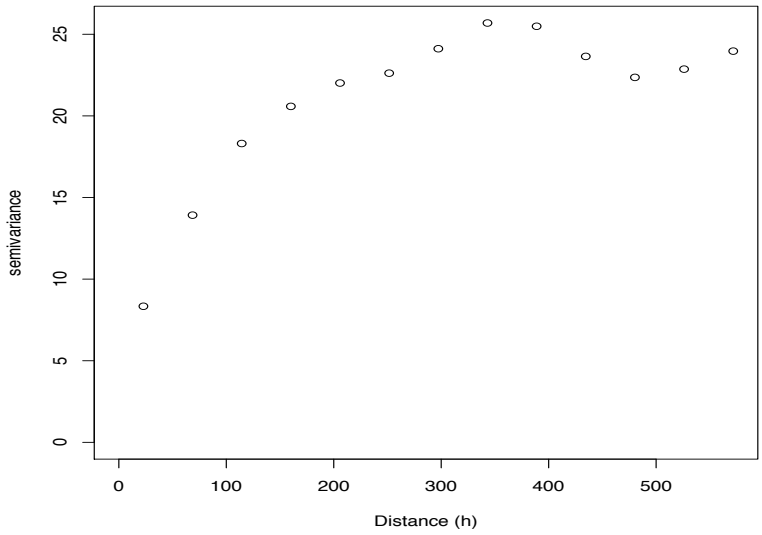
Therefore:

$$\left(\frac{Z(s+h) - Z(s)}{\sqrt{2\gamma(h)}}\right)^2 \sim \chi_1^2 \quad (\chi^2 \text{ distribution with 1 degree of freedom}).$$

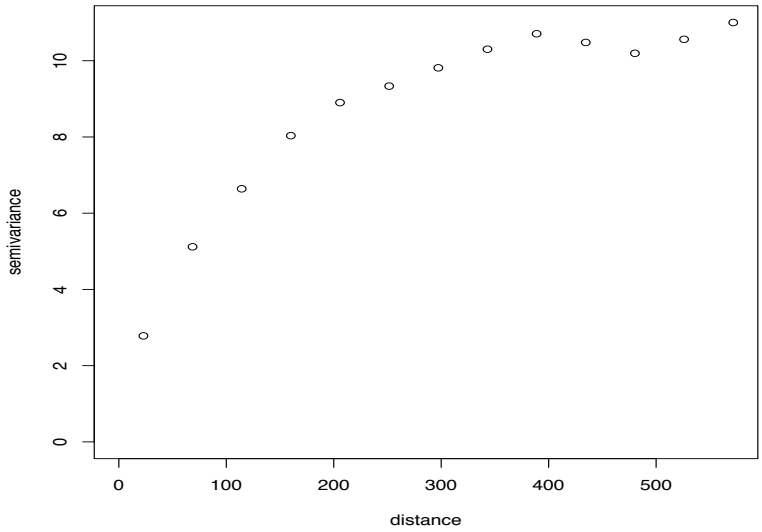
This is a highly skewed distribution. However if $X \sim \chi_1^2$ then $Y = X^{\frac{1}{4}}$ has an approximately symmetric distribution (see figure below). We would expect that the quantity $(Z(s+h) - Z(s))^{\frac{1}{2}}$ will behave much better than $(Z(s+h) - Z(s))^2$.



A typical semivariogram plot (classical estimator):



The semivariogram plot (robust estimator):



- **Modeling the sample variogram**
  Once the sample variogram is computed, a function is fit to it. In other words, we try to come up with what the variogram graph would look like if we had the entire population of all possible pairs. Popular variogram models that are used are the linear, spherical, and exponential (also see graphs on next pages).

  Linear model:
  This is the simplest model for the semivariogram graph. It depends on one parameter, the slope b.

$$\gamma(h;\theta) = \begin{cases} 0, & h = 0 \\ c_0 + bh, & h \neq 0 \end{cases}$$

$$\theta = (c_0, b)', \text{ where } c_0 \geq 0 \text{ and } b \geq 0.$$

  Spherical model:
  This is the model that proposed by Matheron. It has two parameters: The range of influence and the sill (or plateau) which the graph reaches at distances h larger then the range. Generally, the range of influence is the distance beyond which pairs are unrelated.

$$\gamma(h;\theta) = \begin{cases} 0, & h = 0 \\ c_0 + c_1(\frac{3}{2}(\frac{h}{\alpha}) - \frac{1}{2}(\frac{h}{\alpha})^3), & 0 < h \leq \alpha \\ c_0 + c_1, & h \geq \alpha \end{cases}$$

$$\theta = (c_0, c_1, \alpha)', \text{ where } c_0 \geq 0,\ c_1 \geq 0, \text{ and } \alpha \geq 0.$$

  Exponential model:
  This model represents an exponential decay of influence between two sample (larger distance between two samples means larger decay). It depends on two parameters, the range and the sill (plateau).

$$\gamma(h;\theta) = \begin{cases} 0, & h = 0 \\ c_0 + c_1(1 - exp(-\frac{h}{\alpha})), & h \neq 0 \end{cases}$$

$$\theta = (c_0, c_1, \alpha)', \text{ where } c_0 \geq 0,\ c_1 \geq 0, \text{ and } \alpha \geq 0.$$

  Other models:
  There are other models, such as, the Gaussian model, the hole effect model, the Paddington mix model, the circular model, cubic model, Matérn function, etc.

Variogram calculations and fitting

```
#Example using stat:
library(gstat)
#Access the data:
a <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
wolfcamp.txt", header=T)

#Create a gstat object:
g <- gstat(id="level", formula = level~1, locations = ~x+y, data = a)

q <- variogram(g)
plot(q)

#Or
plot(variogram(g))

#There is a trend.
g_trend <- gstat(id="level", formula = level~x+y,
  locations = ~x+y, data = a)

#Plot new variogram:
q <- variogram(g_trend)
plot(q)

#Fit a noel variogram by eye:
fit_var <- vgm(30000,"Sph",70,10000)

plot(q, fit_var)

#Variogram fitting using OLS, GLS, etc.
v.fit <- fit.variogram(q, vgm(30000,"Sph",60,10000))

v.fit1 <- fit.variogram(variogram(g_trend), vgm(30000,"Sph",60,10000), fit.method=1)
v.fit2 <- fit.variogram(variogram(g_trend), vgm(30000,"Sph",60,10000), fit.method=2)
v.fit6 <- fit.variogram(variogram(g_trend), vgm(30000,"Sph",60,10000), fit.method=6)
v.fit7 <- fit.variogram(variogram(g_trend), vgm(30000,"Sph",60,10000), fit.method=7)

plot(q, v.fit1)
plot(q, v.fit2)
plot(q, v.fit6)
plot(q, v.fit7)

See also
http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/geoR_soil_data_fitting.txt and
http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/geoR_soil_data_fitting.txt .
```

Ordinary kriging
Kriging (Matheron 1963) owes its name to D. G. Krige a South African mining engineer and it was first applied in mining data. Kriging assumes a random field expressed through a variogram or covariance function. It is a very popular method to solve the spatial prediction problem. Let $\mathbf{Z} = (Z(s_1), Z(s_2), ..., Z(s_n))'$ be the vector of the observed data at known spatial locations $s_1, s_2, ..., s_n$. The objective is to estimate the unobserved value $Z(s_0)$ at location s_0.

The model:
The model assumption is:

$$Z(s) = \mu + \delta(s)$$

where $\delta(s)$ is a zero mean stochastic term with variogram $2\gamma(\cdot)$. The variogram was discussed in previous handouts in detail.

The Kriging System
The predictor assumption is

$$\hat{Z}(s_0) = \sum_{i=1}^{n} w_i Z(s_i)$$

i.e. it is a weighted average of the sample values, and $\sum_{i=1}^{n} w_i = 1$ to ensure unbiasedness. The w_i's are the weights that will be estimated.

Kriging minimizes the mean squared error of prediction

$$min \;\; \sigma_e^2 = E[Z(s_0) - \hat{Z}(s_0)]^2$$

or

$$min \;\; \sigma_e^2 = E\left[Z(s_0) - \sum_{i=1}^{n} w_i Z(s_i)\right]^2$$

For intrinsically stationary process the last equation can be written as:

$$\sigma_e^2 = 2\sum_{i=1}^{n} w_i \gamma(s_0 - s_i) \; - \; \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \gamma(s_i - s_j) \tag{1}$$

See next page for the proof:

Let's examine $(Z(s_0) - \sum_{i=1}^n w_i Z(s_i))^2$:

$$\left(z(s_0) - \sum_{i=1}^n w_i z(s_i)\right)^2 =$$

$$z^2(s_0) - 2z(s_0)\sum_{i=1}^n w_i z(s_i) + \sum_{i=1}^n \sum_{j=1}^n w_i w_j z(s_i) z(s_j) =$$

$$\sum_{i=1}^n w_i z^2(s_0) - 2\sum_{i=1}^n w_i z(s_0) z(s_i) + \sum_{i=1}^n \sum_{j=1}^n w_i w_j z(s_i) z(s_j)$$

$$-\frac{1}{2}\sum_{i=1}^n w_i z^2(s_i) - \frac{1}{2}\sum_{j=1}^n w_j z^2(s_j) + \sum_{i=1}^n w_i z^2(s_i) =$$

$$-\frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n w_i w_j \left[z(s_i) - z(s_j)\right]^2 + \sum_{i=1}^n w_i \left[z(s_0) - z(s_i)\right]^2$$

If we take expectations on the last expression we have

$$-\frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n w_i w_j E\left[z(s_i) - z(s_j)\right]^2 + \sum_{i=1}^n w_i E\left[z(s_0) - z(s_i)\right]^2 =$$

$$-\frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n w_i w_j \text{var}\left[z(s_i) - z(s_j)\right] + \sum_{i=1}^n w_i \text{var}\left[z(s_0) - z(s_i)\right]$$

But $\text{var}\left[z(s_i) - z(s_j)\right] = 2\gamma(\cdot)$ is the definition of the variogram, and therefore the previous expression is written as: $2\sum_{i=1}^n w_i \gamma(s_0 - s_i) \; - \; \sum_{i=1}^n \sum_{j=1}^n w_i w_j \gamma(s_i - s_j)$
Therefore kriging minimizes

$$\sigma_e^2 = E[(Z(s_0) - \sum_{i=1}^n w_i Z(s_i)]^2 =$$

$$2\sum_{i=1}^n w_i \gamma(s_0 - s_i) \; - \; \sum_{i=1}^n \sum_{j=1}^n w_i w_j \gamma(s_i - s_j)$$

subject to

$$\sum_{i=1}^n w_i = 1$$

The minimization is carried out over $(w_1, w_2, ..., w_n)$, subject to the constraint $\sum_{i=1}^n w_i = 1$. Therefore the minimization problem can be written as:

$$\min \; 2\sum_{i=1}^n w_i \gamma(s_0 - s_i) \; - \; \sum_{i=1}^n \sum_{j=1}^n w_i w_j \gamma(s_i - s_j) \; - \; 2\lambda(\sum_{i=1}^n w_i - 1) \tag{2}$$

where λ is the Lagrange multiplier. After differentiating (2) with respect to $w_1, w_2, ..., w_n$, and λ and set the derivatives equal to zero we find that

$$-\sum_{j=1}^n w_j \gamma(s_i - s_j) + \gamma(s_0 - s_i) - \lambda = 0, \quad i = 1, ..., n$$

and

$$\sum_{i=1}^n w_i = 1$$

Using matrix notation the previous system of equations can be written as

$$\mathbf{\Gamma W} = \boldsymbol{\gamma}$$

Therefore the weights $w_1, w_2, ..., w_n$ and the Lagrange multiplier λ can be obtained by

$$\mathbf{W} = \mathbf{\Gamma^{-1}}\boldsymbol{\gamma}$$

where

$$\mathbf{W} = (w_1, w_2, ..., w_n, \lambda)$$

$$\boldsymbol{\gamma} = (\gamma(s_0 - s_1), \gamma(s_0 - s_2), ..., \gamma(s_0 - s_n), 1)'$$

$$\mathbf{\Gamma} = \begin{cases} \gamma(s_i - s_j), & i = 1, 2, ..., n, \quad j = 1, 2, ..., n, \\ 1, & i = n+1, \quad j = 1, ..., n, \\ 1, & j = n+1, \quad i = 1, ..., n, \\ 0, & i = n+1, \quad j = n+1. \end{cases}$$

The variance of the estimator:
So far, we found the weights and therefore we can compute the estimator: $\hat{Z}(s_0) = \sum_{i=1}^{n} w_i Z(s_i)$. How about the variance of the estimator, namely σ_e^2?

We multiply

$$-\sum_{j=1}^{n} w_j \gamma(s_i - s_j) + \gamma(s_0 - s_i) - \lambda = 0$$

by w_i and we sum over all $i = 1, \cdots, n$ to get:

$$-\sum_{i=1}^{n} w_i \sum_{j=1}^{n} w_j \gamma(s_i - s_j) + \sum_{i=1}^{n} w_i \gamma(s_0 - s_i) - \sum_{i=1}^{n} w_i \lambda = 0$$

Or

$$-\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \gamma(s_i - s_j) + \sum_{i=1}^{n} w_i \gamma(s_0 - s_i) - \sum_{i=1}^{n} w_i \lambda = 0$$

Therefore,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \gamma(s_i - s_j) = \sum_{i=1}^{n} w_i \gamma(s_0 - s_i) - \sum_{i=1}^{n} w_i \lambda$$

If we substitute this result into equation (1) we finally get:

$$\sigma_e^2 = \sum_{i=1}^{n} w_i \gamma(s_i - s_0) + \lambda \tag{3}$$

The Kriging System

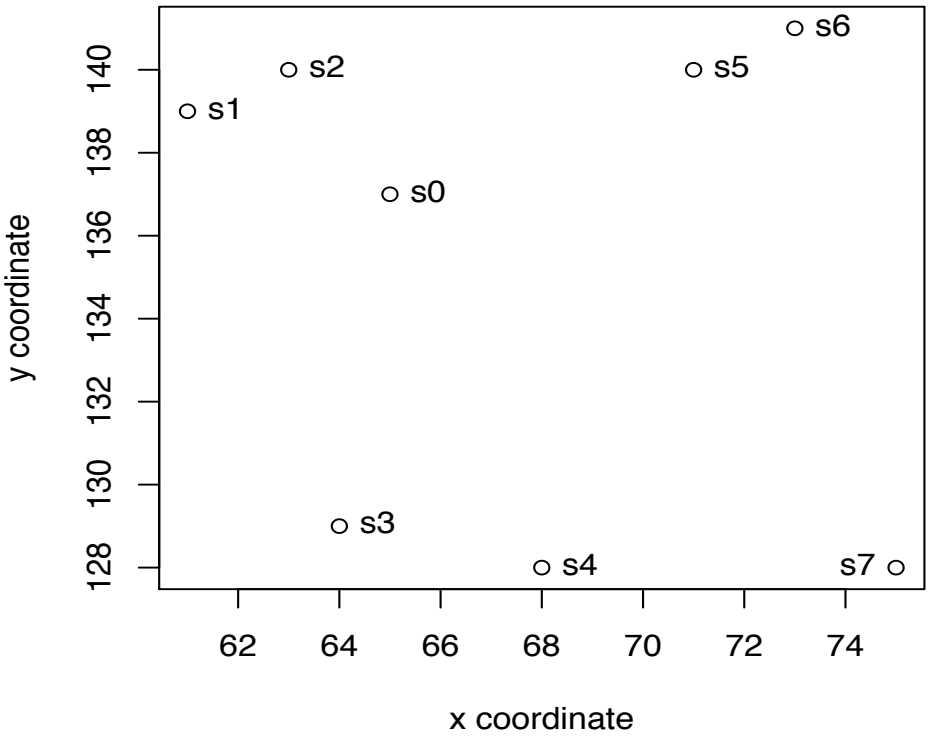
$$\begin{pmatrix} \gamma(s_1 - s_1) & \gamma(s_1 - s_2) & \gamma(s_1 - s_3) & \cdots & \gamma(s_1 - s_n) & 1 \\ \gamma(s_2 - s_1) & \gamma(s_2 - s_2) & \gamma(s_2 - s_3) & \cdots & \gamma(s_2 - s_n) & 1 \\ \cdots & \cdots & \ddots & \cdots & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \cdots & 1 \\ \gamma(s_n - s_1) & \gamma(s_n - s_2) & \gamma(s_n - s_3) & \cdots & \gamma(s_n - s_n) & 1 \\ 1 & 1 & \cdots & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_n \\ \lambda \end{pmatrix} = \begin{pmatrix} \gamma(s_0 - s_1) \\ \gamma(s_0 - s_2) \\ \vdots \\ \vdots \\ \gamma(s_0 - s_n) \\ 1 \end{pmatrix}$$

We understand now why the sample variogram cannot be used directly in the kriging system, and instead a theoretical variogram is used. First, the $\boldsymbol{\gamma}$ vector may call for variogram values for distances that are not available from sample data. There are situations where the distance from the point being estimated to a particular sample is smaller than the distance between pairs of available samples. Since the sample data set cannot provide any pairs for these small distances, we must rely on a function that provides variogram values for all distances. Second, the use of the sample variogram does not guarantee the existence and uniqueness of the solution to the kriging system. In other words the sample variogram version of $\boldsymbol{\Gamma}$ might not be positive definite. To be guaranteed of having one and only one solution, we must ensure that our system has the property of positive definiteness. This is ensured by the choice of one of the model variograms mentioned earlier.

A simple example:
Consider the following data

s_i	x	y	$z(s_i)$
s_1	61	139	477
s_2	63	140	696
s_3	64	129	227
s_4	68	128	646
s_5	71	140	606
s_6	73	141	791
s_7	75	128	783
$\mathbf{s_0}$	**65**	**137**	**???**

Our goal is to estimate the unknown value at location s_0. Here is the $x - y$ plot:



For these data, let's assume that we use the exponential semivariogram model with parameters $c_0 = 0, c_1 = 10, \alpha = 3.33$.

$$\gamma(h) = 10(1 - e^{-\frac{h}{3.33}}).$$

We need to construct the matrix $\mathbf{\Gamma}$ and the vector $\boldsymbol{\gamma}$. First we calculate the distance matrix as shown below:

$$\text{Distance matrix} = \begin{pmatrix} & s_0 & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & s_7 \\ s_0 & 0.00 & 4.47 & 3.61 & 8.06 & 9.49 & 6.71 & 8.94 & 13.45 \\ s_1 & 4.47 & 0.00 & 2.24 & 10.44 & 13.04 & 10.05 & 12.17 & 17.80 \\ s_2 & 3.61 & 2.24 & 0.00 & 11.05 & 13.00 & 8.00 & 10.05 & 16.97 \\ s_3 & 8.06 & 10.44 & 11.05 & 0.00 & 4.12 & 13.04 & 15.00 & 11.05 \\ s_4 & 9.49 & 13.04 & 13.00 & 4.12 & 0.00 & 12.37 & 13.93 & 7.00 \\ s_5 & 6.71 & 10.05 & 8.00 & 13.04 & 12.37 & 0.00 & 2.24 & 12.65 \\ s_6 & 8.94 & 12.17 & 10.05 & 15.00 & 13.93 & 2.24 & 0.00 & 13.15 \\ s_7 & 13.45 & 17.80 & 16.90 & 11.05 & 7.00 & 2.65 & 13.15 & 0.00 \end{pmatrix}$$

The ij_{th} entry in the matrix above was computed as follows:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Now, we can find the entries of the matrix $\mathbf{\Gamma}$ and the vector $\boldsymbol{\gamma}$. Each entry will be computed using the exponential semivariogram $\gamma(h) = 10(1 - e^{-\frac{h}{3.33}})$. Here they are:

$$\mathbf{\Gamma} = \begin{pmatrix} 0 & 4.893 & 9.564 & 9.800 & 9.510 & 9.740 & 9.952 & 1 \\ 4.893 & 0 & 9.637 & 9.798 & 9.093 & 9.510 & 9.938 & 1 \\ 9.564 & 9.637 & 0 & 7.095 & 9.800 & 9.889 & 9.637 & 1 \\ 9.800 & 9.798 & 7.095 & 0 & 9.755 & 9.847 & 8.775 & 1 \\ 9.510 & 9.093 & 9.800 & 9.755 & 0 & 4.893 & 9.775 & 1 \\ 9.740 & 9.510 & 9.889 & 9.847 & 4.893 & 0 & 9.806 & 1 \\ 9.952 & 9.938 & 9.637 & 8.775 & 9.775 & 9.806 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\boldsymbol{\gamma} = \begin{pmatrix} 7.384 \\ 6.614 \\ 9.109 \\ 9.420 \\ 8.664 \\ 9.316 \\ 9.823 \\ 1 \end{pmatrix}$$

The weights and the Lagrnange multiplier can be obtained as follows:

$$\boldsymbol{W} = \boldsymbol{\Gamma}^{-1}\boldsymbol{\gamma} = \begin{pmatrix} 0 & 4.893 & 9.564 & 9.800 & 9.510 & 9.740 & 9.952 & 1 \\ 4.893 & 0 & 9.637 & 9.798 & 9.093 & 9.510 & 9.938 & 1 \\ 9.564 & 9.637 & 0 & 7.095 & 9.800 & 9.889 & 9.637 & 1 \\ 9.800 & 9.798 & 7.095 & 0 & 9.755 & 9.847 & 8.775 & 1 \\ 9.510 & 9.093 & 9.800 & 9.755 & 0 & 4.893 & 9.775 & 1 \\ 9.740 & 9.510 & 9.889 & 9.847 & 4.893 & 0 & 9.806 & 1 \\ 9.952 & 9.938 & 9.637 & 8.775 & 9.775 & 9.806 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} 7.384 \\ 6.614 \\ 9.109 \\ 9.420 \\ 8.664 \\ 9.316 \\ 9.823 \\ 1 \end{pmatrix}.$$

The answer is:

$$\boldsymbol{W} = \begin{pmatrix} 0.174 \\ 0.317 \\ 0.129 \\ 0.086 \\ 0.151 \\ 0.057 \\ 0.086 \\ 0.906 \end{pmatrix}.$$

The last element of the **W** vector is the Lagrange multiplier, $\lambda = 0.906$. We can verify that the sum of the elements 1 through 7 is equal to 1, as it should be.

The predicted value at location s_0 is equal to:

$$\hat{z}(s_0) = \sum_{i=1}^{n} w_i z(s_i) = 0.174(477) + \cdots + 0.086(783) = 592.59.$$

And the variance:

$$\sigma_e^2 = \sum_{i=1}^{n} w_i \gamma(s_i - s_0) + \lambda = 0.174(7.384) + \cdots + 0.086(9.823) + 0.906 = 8.96.$$

Under the assumption that $Z(s)$ is Gaussian a 95% confidence interval can be computed as follows:

$$592.59 \pm 1.96\sqrt{8.96}$$

Or

$$577.09 \leq Z(s_0) \leq 588.83$$

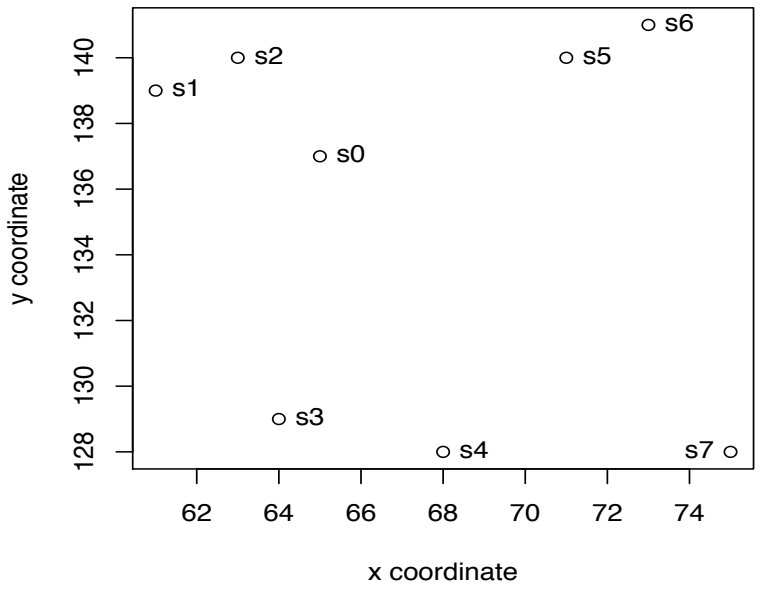
Ordinary kriging using `geoR` and `gstat`

We discuss next kriging using the `R` packages `geoR` and `gstat`. We will use the numerical example from last lecture. Here it is:

A simple example:
Consider the following data

s_i	x	y	$z(s_i)$
s_1	61	139	477
s_2	63	140	696
s_3	64	129	227
s_4	68	128	646
s_5	71	140	606
s_6	73	141	791
s_7	75	128	783
$\mathbf{s_0}$	**65**	**137**	**???**

Our goal is to predict the unknown value at location s_0. Here is the $x - y$ plot:



For these data, let's assume that we use the exponential semivariogram model with parameters $c_0 = 0, c_1 = 10, \alpha = 3.33$.

$$\gamma(h) = c_0 + c_1(1 - e^{-\frac{h}{\alpha}}) = 10(1 - e^{-\frac{h}{3.33}}).$$

which is equivalent to the covariance function

$$C(h) = \begin{cases} c_0 + c_1, & h = 0 \\ c_1 e^{-\frac{h}{\alpha}}, & h > 0 \end{cases} \Rightarrow C(h) = \begin{cases} 10, & h = 0 \\ 10e^{-\frac{h}{3.33}}, & h > 0 \end{cases}$$

The predicted value at location s_0 is equal to:

$$\hat{z}(s_0) = \sum_{i=1}^{n} w_i z(s_i) = 0.174(477) + \cdots + 0.086(783) = 592.59.$$

And the variance:

$$\sigma_e^2 = \sum_{i=1}^{n} w_i \gamma(s_i - s_0) + \lambda = 0.174(7.384) + \cdots + 0.086(9.823) + 0.906 = 8.96.$$

Kriging using `geoR`:
We will use now the `geoR` package to find the same result. First we read our data as a geodata object:

```
> a <- read.table("http://www.stat.ucla.edu/~nchristo/statistics403/
       kriging_11.txt", header=TRUE)
> b <- as.geodata(a)
```

To predict the unknown value at locaton $(x = 65, y = 137)$ we use the following:

```
> prediction <- ksline(b, cov.model="exp", cov.pars=c(10,3.33), nugget=0,
  locations=c(65,137))
```

where,

`b`	The geodata
`cov.model`	The model we are using
`cov.pars`	The parameters of the model (partial sill and range)
`nugget`	The value of the nugget effect
`locations`	The coordinates (x, y) of the points to be predicted

The object "prediction" contains among other things the predicted value at location $x = 65, y = 137$ and its variance. We can obtain them as follows:

```
> prediction$predict
[1] 592.7587
> prediction$krige.var
[1] 8.960294
```

Suppose now we want to predict the value at many locations. The following commands will produce a grid whose points will be predicted using kriging:

```
> x.range <- as.integer(range(a[,1]))
> y.range <- as.integer(range(a[,2]))
> grd <- expand.grid(x=seq(from=x.range[1], to=x.range[2], by=1),
  y=seq(from=y.range[1], to=y.range[2], by=1))
]
> q <- ksline(b, cov.model="exp",cov.pars=c(10,3.33), nugget=0,
  locations=grd)
```

In case you have a `variofit` output you can use it as an input of the argument `krige` as follows (this is only an example):

```
var1 <- variog(b, max.dist=1000)
fit1 <- variofit(var1, cov.model="exp", ini.cov.pars=c(1000, 100),
        fix.nugget=FALSE, nugget=250)

q <- krige.conv(b, locations=grd, krige=krige.control(obj.model=fit1))
```

We can access the predicted values and their variances using `q$predict` and `q$krige.var`. Here are the first 5 predicted values with their variances:

```
> cbind(q$predict[1:5], q$krige.var[1:5])
         [,1]     [,2]
[1,] 458.4491 9.245493
[2,] 413.2103 7.850838
[3,] 362.4674 5.927999
[4,] 338.9828 4.516906
[5,] 393.3933 5.280417
```

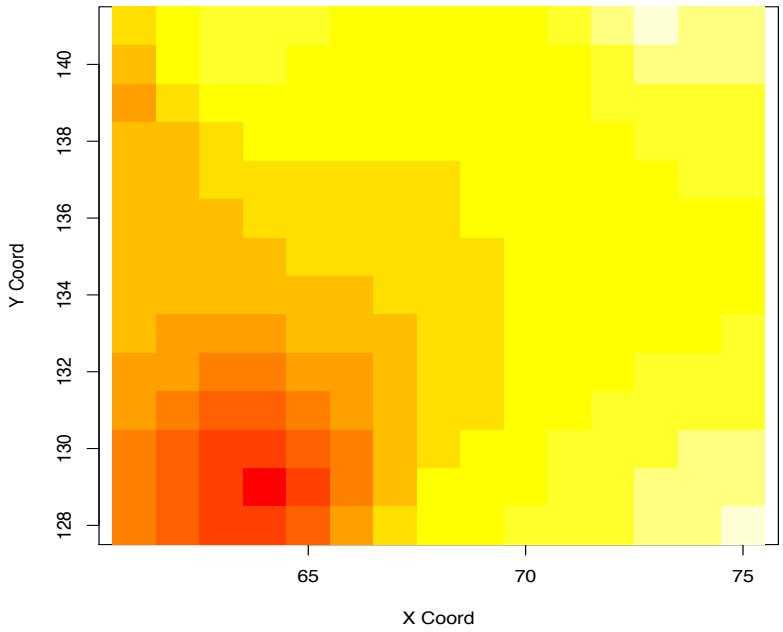
To construct the raster map we type:

```
image(q, val=q$predict)
```

Or simply:

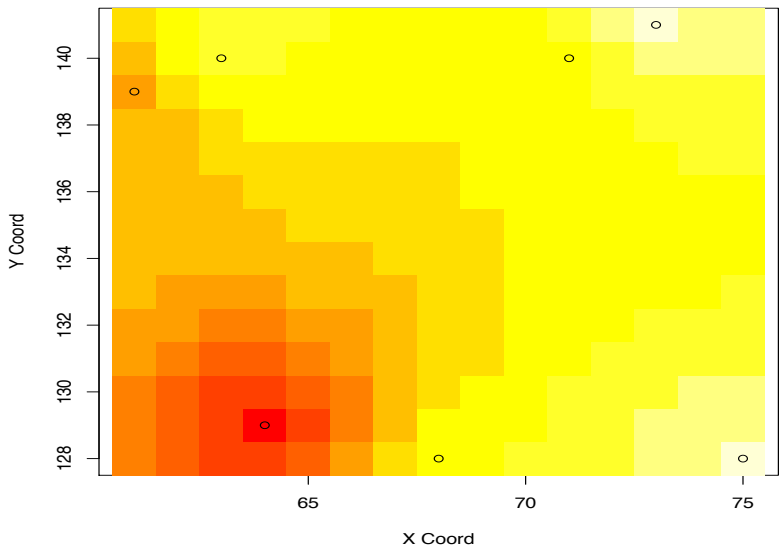
```
image(q)
```

Here is the plot:



And here is the plot with the data points:

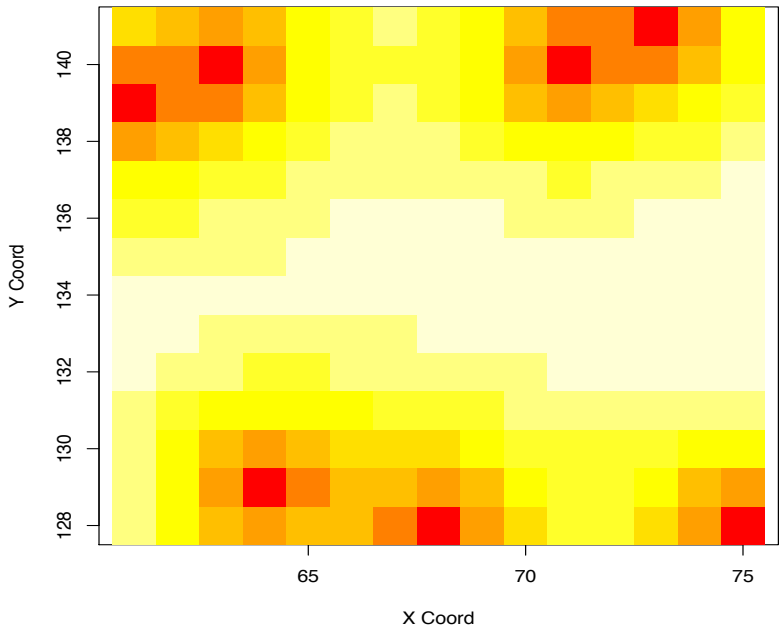
```
points(a)
```



We can construct a raster map of the variances:

```
> image(q, val=q$krige.var)
```

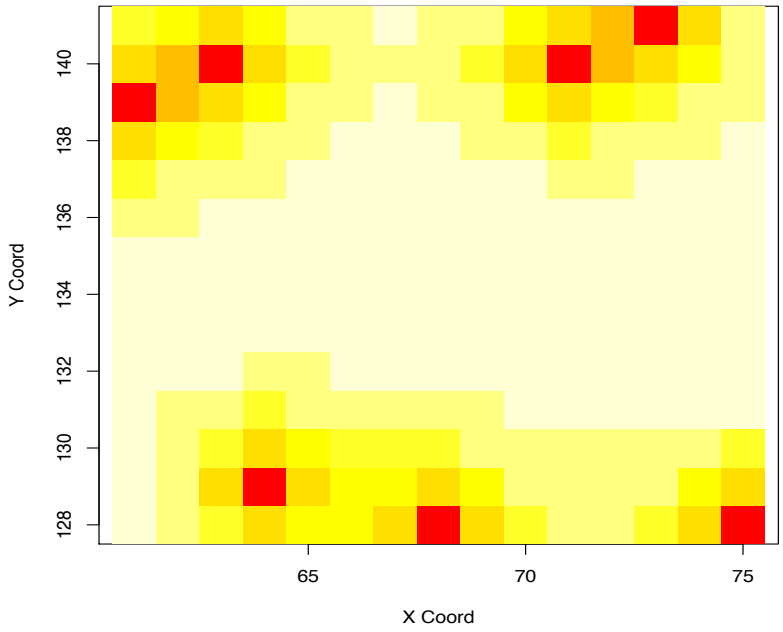
Here is the plot:



And also we can construct a raster map of the standard errors:

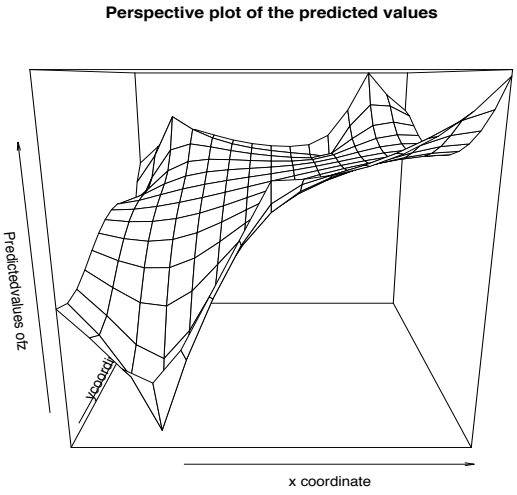
```
> image(q, val=sqrt(q$krige.var))
```

Here is the plot:



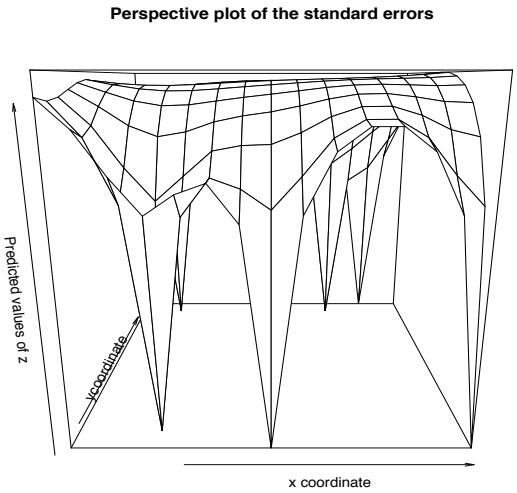
The following command will construct a perspective plot of the predicted values:

```
> persp(x,y,matrix(q$predict,15,14), xlab="x coordinate",
  ylab="y coordinate", zlab="Predicted values of z",
  main="Perspective plot of the predicted values")
```



And here is the perspective plot of the standard errors:

```
> persp(x,y,matrix(sqrt(q$krige.var),15,14), xlab="x coordinate",
  ylab="ycoordinate", zlab="Predicted values of z",
  main="Perspective plot of the standard errors")
```



Kriging using `gstat`:
We will use now the `gstat` package to find the same result. First we read our data and create the grid for prediction as follows:

```
> a <- read.table("http://www.stat.ucla.edu/~nchristo/statistics403/
       kriging_11.txt", header=TRUE)
> x.range <- as.integer(range(a[,1]))
> y.range <- as.integer(range(a[,2]))
> grd <- expand.grid(x=seq(from=x.range[1], to=x.range[2], by=1),
  y=seq(from=y.range[1], to=y.range[2], by=1))
```

We now define the model. Normally the model must be estimated from the sample variogram, but for this simple example we assume that it is given as below:

```
> library(gstat)
> m <- vgm(10, "Exp", 3.33, 0)
```

There are two ways to perform ordinary kriging with `gstat`. The data and the grid are used as data frames, with the extra argument `locations` as shown below:

```
> q1 <- krige(id="z", formula=z~1, data=a, newdata=grd, model = m,
  locations=~x+y)
```

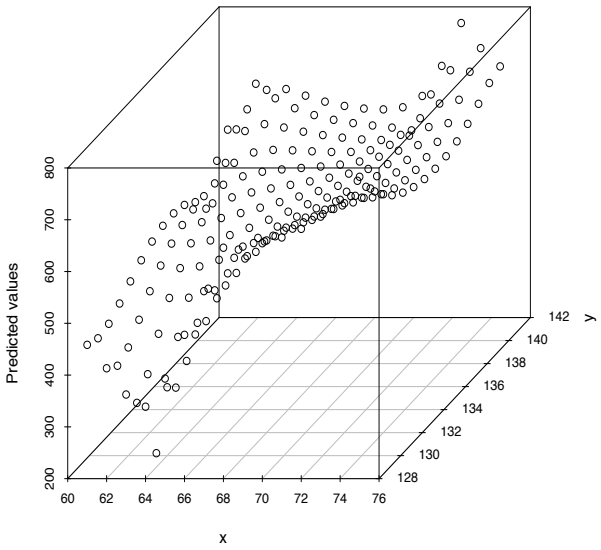
The other way is to convert the data and the grid as spatial data points data frame:

```
> coordinates(a) <- ~x+y
> coordinates(grd) <- ~x+y
> q2 <- krige(id="z", formula=z~1, a, newdata=grd, model = m)
```

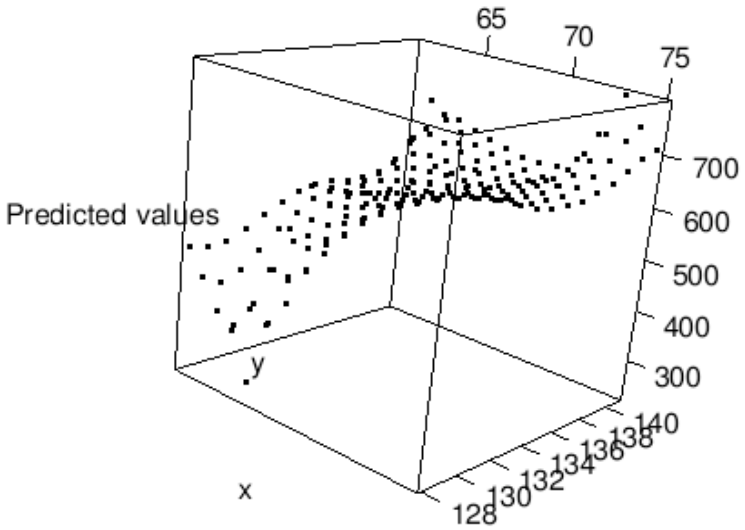
Important note: If we use the second way the argument `data=` is not allowed. We simply use the name of the data, here just `a`. Also, `q1` is a data frame, while `q2` is spatial data points data frame. Using `q1` we can create a 3D plot with the libraries `scatterploted` and `rgl` as follows:

```
> library(scatterplot3d)
> library(rgl)
> scatterplot3d(q1$x, q1$y, q1$z.pred, xlab="x", ylab="y",
  zlab="Predicted values")
> plot3d(q1$x, q1$y, q1$z.pred, size=3)
```

Here are the plots:
(a). Using the `scatterplot3d` command.



(b). Using the `plot3d` command.



A complete example on kriging using `gstat`:
We will use again the soil data from the *Maas* river. Here is some background.
The actual data set contains many variables but here we will use the x, y coordinates and the concentration of lead and zinc in *ppm* at each data point. The motivation for this study is to predict the concentration of heavy metals around the banks of the Maas river in the area west of the town Stein in the Netherlands. These heavy metals were accumulated over the years because of river pollution. Here is the area of study:

You can access the data at

```
> a <- read.table("http://www.stat.ucla.edu/~nchristo/statistics403/
       soil.txt", header=TRUE)
# Save the original image function:
> image.orig <- image
```

To load the gstat package type

```
> library(gstat)
```

First, we will compute the descriptive statistics of the data set, construct the stem-and-leaf plots, histograms, and boxplots:

```
> stem(a$lead)
> boxplot(a$lead)
> hist(a$lead)
> stem(a$zinc)
> boxplot(a$zinc)
> hist(a$zinc)
> summary(a)
```

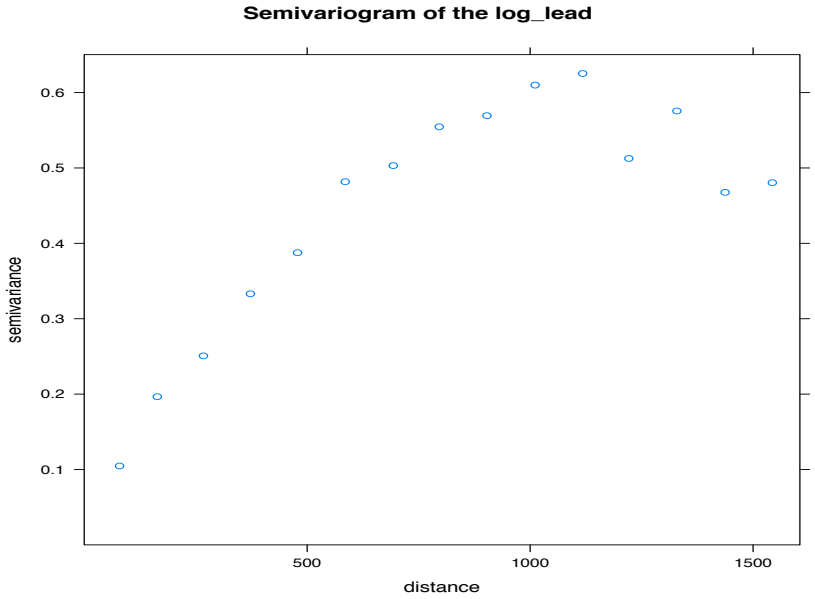
Transform the data (logarithm transformation):

```
> log_lead <- log10(a$lead)
> log_zinc <- log10(a$zinc)
> stem(log_lead)
> boxplot(log_lead)
> hist(log_lead)
> stem(log_zinc)
> boxplot(log_zinc)
> hist(log_zinc)

#Create a gstat object;
> g <- gstat(id="log_lead", formula = log(lead)~1, locations = ~x+y,
  data = a)

#Plot the variogram:
> plot(variogram(g), main="Semivariogram of the log_lead")

#Fit a model variogram to the sample variogram:
> v.fit <- fit.variogram(variogram(g), vgm(0.5,"Sph",1000,0.1))
> plot(variogram(g),v.fit)

#Note:  The values above were the initial values for the partial sill,
#range, and nugget.  Then the function fit.variogram uses a minimization
#procedure to fit a model variogram to the sample variogram.  Type v.fit
#to get the estimates of the model parameters.

> v.fit
```

```
  model      psill    range
1   Nug 0.05156252   0.0000
2   Sph 0.51530678 965.1506

#There are different weights you can use in the minimization procedure.  The
#default (the one used above) is $N_h/h^2$ where $N_h$ is the number of pairs
#and $h$ the separation distance.  You can chose the type of weights by using
#the argument fit.method=integer, where integer is a number from the table
#below:

fit.method               weights
1                        N_h
2                        N_h/gamma(h;theta)^2 (Cressie's weights)
6                        OLS (no weights)
7                        N_h/h^2 (default)


#Use kriging to estimate the value of log(lead) at the grid values.
#First we create the grid.
> x.range <- as.integer(range(a[,1]))
> x.range
> y.range <- as.integer(range(a[,2]))
> y.range
> grd <- expand.grid(x=seq(from=x.range[1], to=x.range[2], by=50),
y=seq(from=y.range[1], to=y.range[2], by=50))

#We want now to use kriging to predict log(lead) at each point on the grid:
> pr_ok <- krige(id="log_lead",log(lead)~1, locations=~x+y,
model=v.fit,
data=a, newdata=grd)

#To find what the object pr_ok contains type:
> names(pr_ok)
[1] "x"             "y"             "log_lead.pred" "log_lead.var"

#To see the predicted values you type:
> pr_ok$log_lead.pred

#And the kriging variances:
> pr_ok$log_lead.var
```

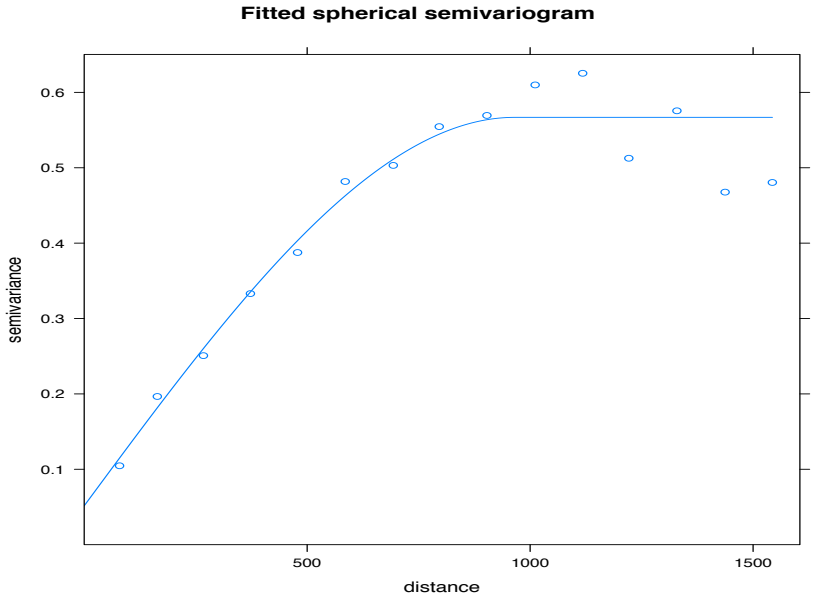
The plot of the sample variogram:

```
> plot(variogram(g), main="Semivariogram of the log_lead")
```



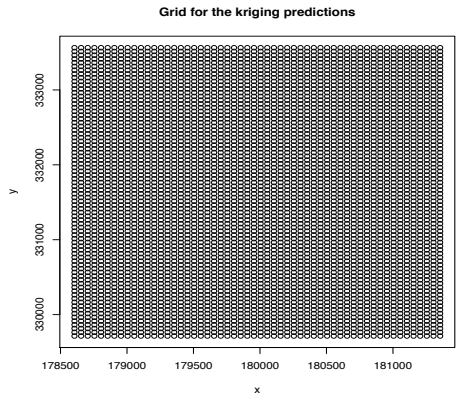
The fitted spherical variogram to the sample variogram:

```
> plot(variogram(g),v.fit)
```



Here is the grid for the kriging predictions:

```
> plot(grd)
```



The vector of the predicted values must be collapsed into a matrix with the `matrix` function:

```
#Collapse the predicted values into a matrix:
qqq <- matrix(pr_ok$log_lead.pred,
length(seq(from=x.range[1], to=x.range[2], by=50)),
length(seq(from=y.range[1], to=y.range[2], by=50)))
```

And we can use the image function to create the raster map of the predicted values:

```
> image(seq(from=x.range[1], to=x.range[2], by=50),
  seq(from=y.range[1], to=y.range[2], by=50), qqq,
  xlab="West to East", ylab="South to North", main="Predicted values")

> points(a)  #The data points can be plotted on the raster map.
```