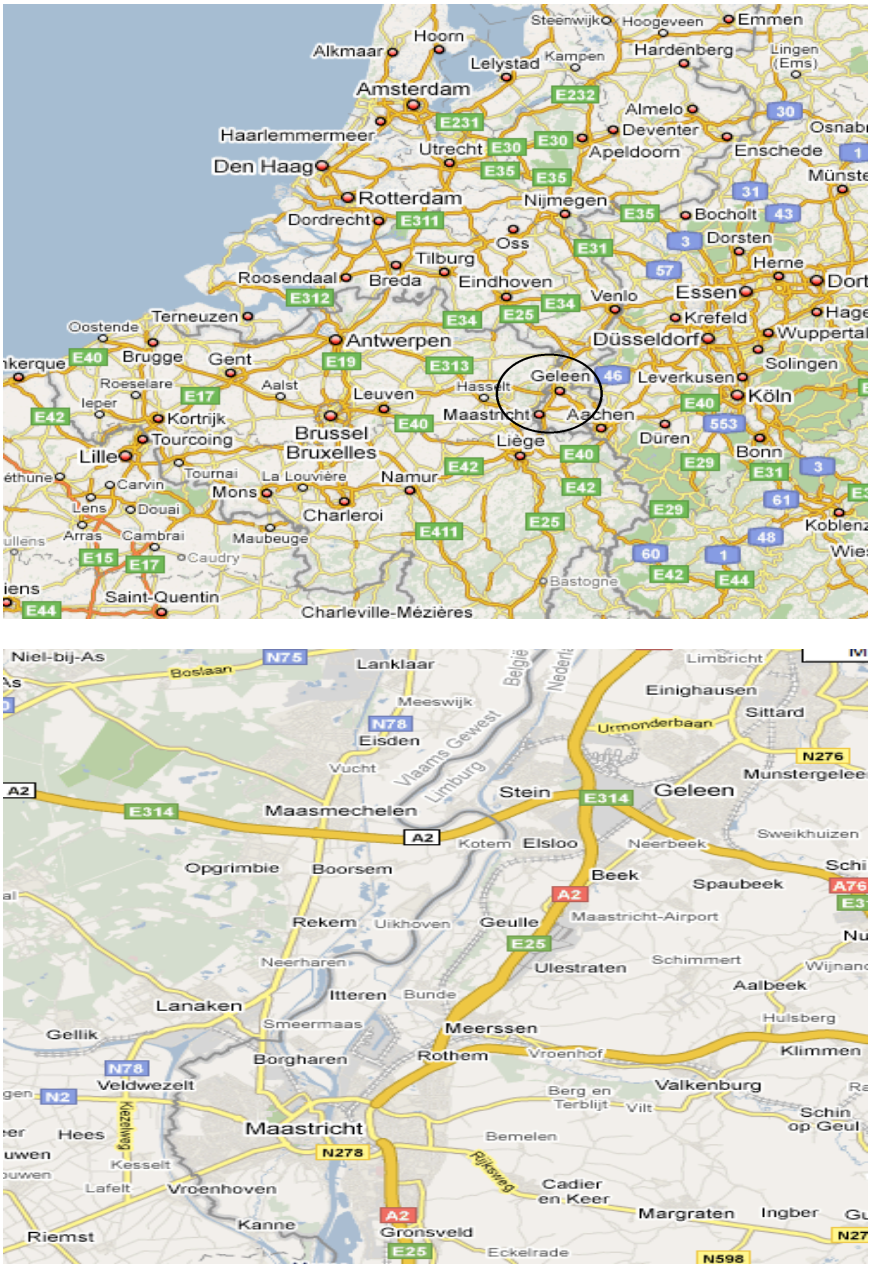
University of California, Los Angeles
Department of Statistics

Statistics C173/C273 **Instructor: Nicolas Christou**

Working with `gstat` - An example

We will use the data from the *Meuse* (Dutch *Maas*) river. 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 the river pollution. Here is the area of study:



You can access the data at

```
> a <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/soil.txt",
  header=T)
```

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)
```

Note: If you have already converted the data frame `a` into a SpatialPointsDataFrame using:

```
> coordinates(a) <- ~x+y
```

you cannot redefine the coordinates in the `gstat` object. Instead you can use:

```
> g <- gstat(id="log_lead", formula = log(lead)~1, data = a)
```

Type `variogram(g)` to see:

```
> variogram(g)
    np       dist     gamma dir.hor dir.ver       id
1   57   79.29244 0.1046520       0       0 log_lead
2  299  163.97367 0.1965929       0       0 log_lead
3  419  267.36483 0.2507668       0       0 log_lead
4  457  372.73542 0.3330690       0       0 log_lead
5  547  478.47670 0.3875716       0       0 log_lead
6  533  585.34058 0.4817750       0       0 log_lead
7  574  693.14526 0.5031432       0       0 log_lead
8  564  796.18365 0.5545787       0       0 log_lead
9  589  903.14650 0.5693882       0       0 log_lead
10 543 1011.29177 0.6098806       0       0 log_lead
11 500 1117.86235 0.6253271       0       0 log_lead
12 477 1221.32810 0.5126165       0       0 log_lead
13 452 1329.16407 0.5755737       0       0 log_lead
14 457 1437.25620 0.4676728       0       0 log_lead
15 415 1543.20248 0.4804887       0       0 log_lead
```

Note: The maximum distance up to which the variogram is computed is equal to $\frac{1}{3}$ of the diagonal of the bounding box defined by the coordinates of the data. It can be changed using the argument `cutoff` as follows:

```
> variogram(g, cutoff=1000)
```

To plot the variogram:

```
> plot(variogram(g, cutoff=1000))
```

Or you can do this:

```
> q <- variogram(g, cutoff=1000)
> plot(q)
```

For the interval width, `gstat` uses a default of the cutoff value divided by 15. This can be changed using the argument `width`:

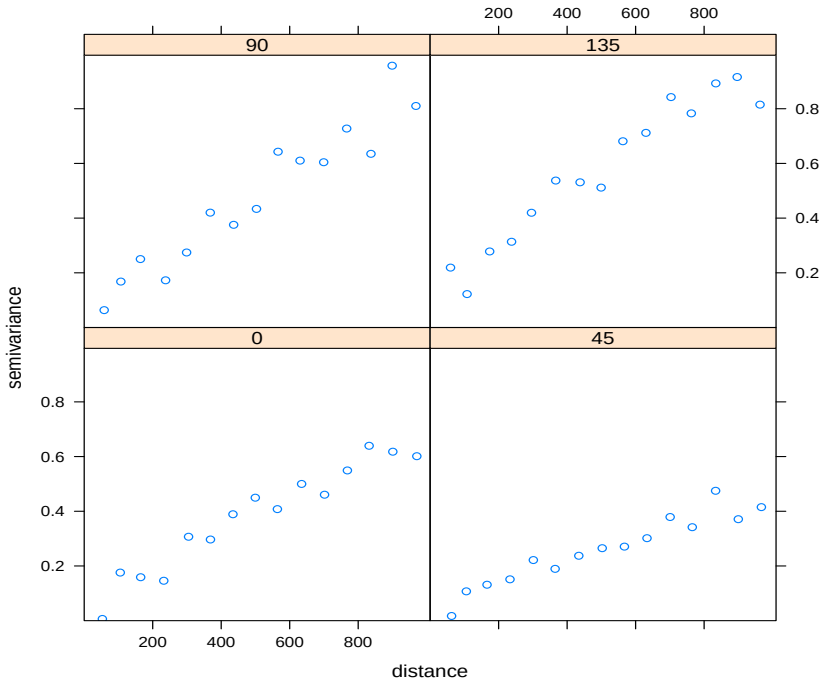
```
> variogram(g, cutoff=1000, width=50)
```

Direction and angular tolerance can be chosen respectively using the `alpha` and `tol.hor` arguments respectively. They are entered in degrees.

Construct the variogram on the 4 main directions $(0^o, 45^o, 90^o, 135^o)$:

```
> v <- variogram(g, alpha=c(0,45,90,135), cutoff=1000)
> plot(v)
```

For example, for the 45^o direction, points will be paired if they fall between 22.5^o and 67.5^o. We get the following graph:



In case there is a trend in the data:

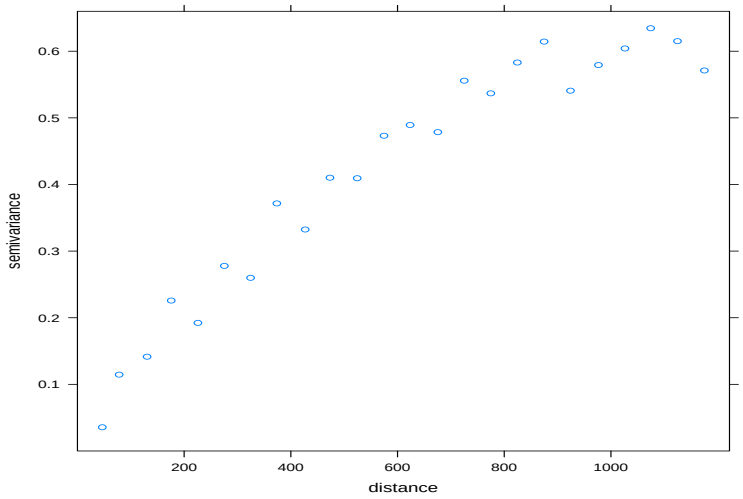
```
> g_trend <- gstat(id="log_lead", formula = log(lead)~x+y,
  locations = ~x+y, data = a)
```

And you can plot the new variogram:

```
> plot(variogram(g_trend))
```

Modeling the variogram
Consider the following sample varogram:

```
> q <- variogram(g, cutoff=1200, width=50)
> plot(q)
```



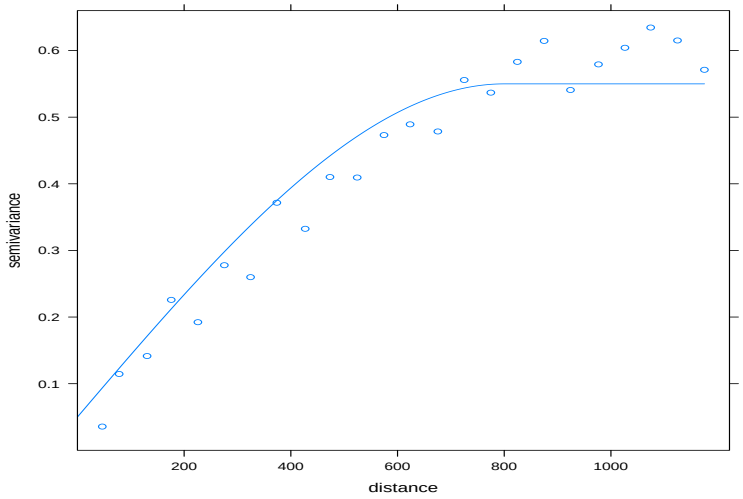
One simple way to fit a model variogram to this sample variogram is to estimate the parameters "by eye" and continue as follows:

```
> fit_var <- vgm(0.5,"Sph",800,0.05)
> plot(q, fit_var)
```

Or you can do this:

```
> plot(q,vgm(0.5,"Sph",800,0.05))
```

Here is the fitted model:



Variogram modeling using ordinary or weighted least squares:
The function used is `fit.variogram`.

```
> v.fit <- fit.variogram(q, vgm(0.5,"Sph",800,0.05))
```

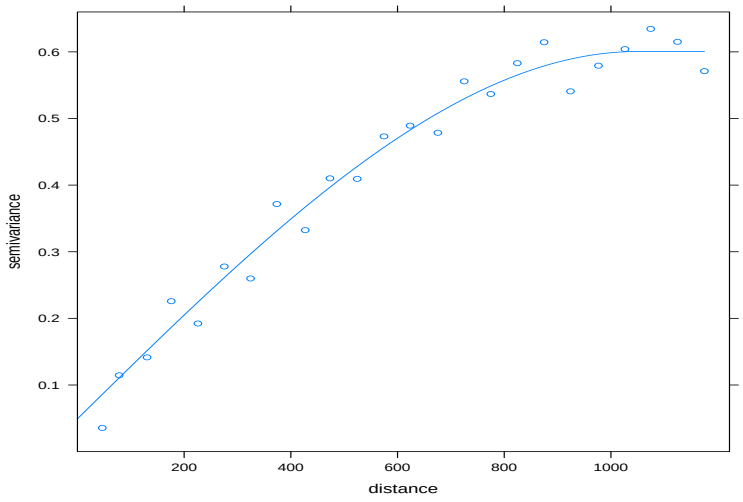
Note: The values above were the initial values for the partial sill, range, and nugget. Then the function `fit.variogram` uses a minimization procedure to fit a model variogram to the sample variogram. Type `v.fit` to get the estimates of the model parameters.

```
> v.fit
  model      psill    range
1   Nug 0.04927708    0.000
2   Sph 0.55124599 1049.791
```

We can plot now the fitted model variogram:

```
> plot(q,v.fit)
```

Here is the plot:



Note:
There are different weights you can use in the minimization procedure. The default (the ones used above) are $\frac{N_h}{h^2}$ where N_h is the number of pairs and h is the separation distance. You can choose 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	$\frac{N_h}{\gamma^2(h;\theta)}$ (Cressie's weights)
6	OLS (no weights)
7	$\frac{N_h}{h^2}$ (default)

Another example:

```
> a <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
  wolfcamp.txt", header=T)

> g <- gstat(id="level", formula = level~1, locations = ~x+y, data = a)
> plot(variogram(g))
> q <- variogram(g)
> plot(q)

> v <- variogram(g, alpha=c(0,45,90,135))
> plot(v)

#Remove the trend:
> g1 <- gstat(id="level", formula = level~x+y, locations = ~x+y, data = a)
> q1 <- variogram(g1)
> plot(q1)

#Variogram modeling:
> v.fit1 <- fit.variogram(variogram(g1), vgm(30000,"Sph",60,10000), fit.method=1)
> v.fit2 <- fit.variogram(variogram(g1), vgm(30000,"Sph",60,10000), fit.method=2)
> v.fit6 <- fit.variogram(variogram(g1), vgm(30000,"Sph",60,10000), fit.method=6)
> v.fit7 <- fit.variogram(variogram(g1), vgm(30000,"Sph",60,10000), fit.method=7)

#Or we can simply do this:
> v.fit1 <- fit.variogram(q1, vgm(30000,"Sph",60,10000), fit.method=1)
> v.fit2 <- fit.variogram(q1, vgm(30000,"Sph",60,10000), fit.method=2)
> v.fit6 <- fit.variogram(q1, vgm(30000,"Sph",60,10000), fit.method=6)
> v.fit7 <- fit.variogram(q1, vgm(30000,"Sph",60,10000), fit.method=7)

> v.fit1
> v.fit2
> v.fit6
> v.fit7

> plot(q1, v.fit1)
```

Data sets:

```
a1 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     wolfcamp.txt", header=TRUE)

a2 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     walker_lake_v.txt", header=TRUE)

a3 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     walker_lake_u.txt", header=TRUE)

a4 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     soil.txt", header=TRUE)

a5 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     o3.txt", header=TRUE)

a6 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     soil_complete.txt", header=TRUE)

a7 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     coal_ash.txt", header=TRUE)

a8 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     broom_barn_data.txt", header=TRUE)

a9 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     nc_sids.txt", header=TRUE)

a10 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/
     unemp_data.txt", header=TRUE)
```