Kriging - An illustration with the Meuse dataset from the gstat package

This illustration uses the Meuse dataset included in gstat which comprises of four heavy metals measured in the top soil in a flood plain along the river Meuse.Apparently polluted sediment is carried by the river and mostly deposited close to the river bank.

Packages needed for this inllustration includes gstat and sp. Note that there are other packages which can handle kriging and other geotatistical tools, such as geoR, geoRglm, and fields.

Note: To use gstat, data need to be projected (i.e., not in lat-long format)

```
library(sp)
library(gstat)
data(meuse)
class(meuse)
```

```
## [1] "data.frame"
```

Let's explore the Meuse dataset a bit more.

```
names(meuse)
```

```
##  [1] "x"       "y"       "cadmium" "copper"  "lead"    "zinc"    "elev"
##  [8] "dist"    "om"      "ffreq"   "soil"    "lime"    "landuse" "dist.m"
```

```
str(meuse)
```

```
## 'data.frame':    155 obs. of  14 variables:
##  $ x      : num  181072 181025 181165 181298 181307 ...
##  $ y      : num  333611 333558 333537 333484 333330 ...
##  $ cadmium: num  11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
##  $ copper : num  85 81 68 81 48 61 31 29 37 24 ...
##  $ lead   : num  299 277 199 116 117 137 132 150 133 80 ...
##  $ zinc   : num  1022 1141 640 257 269 ...
##  $ elev   : num  7.91 6.98 7.8 7.66 7.48 ...
##  $ dist   : num  0.00136 0.01222 0.10303 0.19009 0.27709 ...
##  $ om     : num  13.6 14 13 8 8.7 7.8 9.2 9.5 10.6 6.3 ...
##  $ ffreq  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ soil   : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 1 1 2 ...
##  $ lime   : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
##  $ landuse: Factor w/ 15 levels "Aa","Ab","Ag",..: 4 4 4 11 4 11 4 2 2 15 ...
##  $ dist.m : num  50 30 150 270 380 470 240 120 240 420 ...
```

```
summary(meuse)
```

```
##        x                y             cadmium           copper
##  Min.   :178605   Min.   :329714   Min.   : 0.200   Min.   : 14.00
##  1st Qu.:179371   1st Qu.:330762   1st Qu.: 0.800   1st Qu.: 23.00
##  Median :179991   Median :331633   Median : 2.100   Median : 31.00
##  Mean   :180005   Mean   :331635   Mean   : 3.246   Mean   : 40.32
##  3rd Qu.:180630   3rd Qu.:332463   3rd Qu.: 3.850   3rd Qu.: 49.50
##  Max.   :181390   Max.   :333611   Max.   :18.100   Max.   :128.00
##
##       lead             zinc             elev             dist
##  Min.   : 37.0   Min.   : 113.0   Min.   : 5.180   Min.   :0.00000
##  1st Qu.: 72.5   1st Qu.: 198.0   1st Qu.: 7.546   1st Qu.:0.07569
##  Median :123.0   Median : 326.0   Median : 8.180   Median :0.21184
##  Mean   :153.4   Mean   : 469.7   Mean   : 8.165   Mean   :0.24002
##  3rd Qu.:207.0   3rd Qu.: 674.5   3rd Qu.: 8.955   3rd Qu.:0.36407
##  Max.   :654.0   Max.   :1839.0   Max.   :10.520   Max.   :0.88039
##
##        om         ffreq  soil   lime       landuse       dist.m
##  Min.   : 1.000   1:84   1:97   0:111   W      :50   Min.   :  10.0
##  1st Qu.: 5.300   2:48   2:46   1: 44   Ah     :39   1st Qu.:  80.0
##  Median : 6.900   3:23   3:12           Am     :22   Median : 270.0
##  Mean   : 7.478                         Fw     :10   Mean   : 290.3
##  3rd Qu.: 9.000                         Ab     : 8   3rd Qu.: 450.0
##  Max.   :17.000                         (Other):25   Max.   :1000.0
##  NA's   :2                              NA's   : 1
```

Function coordinates: when the function coordinates is assigned, it will promotes the data.frame meuse into a SpatialPointsDataFrame which knows about its spatial coordinates. See the codes below. Compare the results of the function summary above and below.

```
coordinates(meuse) = ~x+y
class(meuse)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
summary(meuse)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##      min    max
## x 178605 181390
## y 329714 333611
## Is projected: NA
## proj4string : [NA]
## Number of points: 155
## Data attributes:
##     cadmium           copper            lead            zinc
##  Min.   : 0.200   Min.   : 14.00   Min.   : 37.0   Min.   : 113.0
##  1st Qu.: 0.800   1st Qu.: 23.00   1st Qu.: 72.5   1st Qu.: 198.0
##  Median : 2.100   Median : 31.00   Median :123.0   Median : 326.0
##  Mean   : 3.246   Mean   : 40.32   Mean   :153.4   Mean   : 469.7
##  3rd Qu.: 3.850   3rd Qu.: 49.50   3rd Qu.:207.0   3rd Qu.: 674.5
##  Max.   :18.100   Max.   :128.00   Max.   :654.0   Max.   :1839.0
##
##       elev             dist               om         ffreq  soil   lime
##  Min.   : 5.180   Min.   :0.00000   Min.   : 1.000   1:84   1:97   0:111
##  1st Qu.: 7.546   1st Qu.:0.07569   1st Qu.: 5.300   2:48   2:46   1: 44
##  Median : 8.180   Median :0.21184   Median : 6.900   3:23   3:12
##  Mean   : 8.165   Mean   :0.24002   Mean   : 7.478
##  3rd Qu.: 8.955   3rd Qu.:0.36407   3rd Qu.: 9.000
##  Max.   :10.520   Max.   :0.88039   Max.   :17.000
##                                     NA's   :2
##     landuse       dist.m
##  W      :50   Min.   :  10.0
##  Ah     :39   1st Qu.:  80.0
##  Am     :22   Median : 270.0
##  Fw     :10   Mean   : 290.3
##  Ab     : 8   3rd Qu.: 450.0
##  (Other):25   Max.   :1000.0
##  NA's   : 1
```

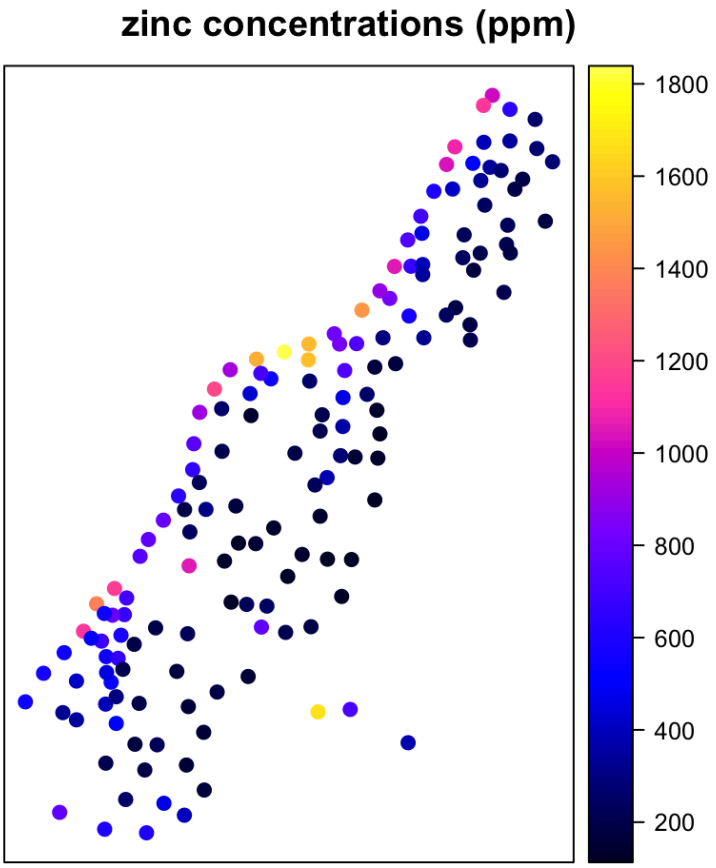
And now you can use the function coordinates to retrieve the spatial coordinates from a SpatialPointsDataFrame meuse.

```
coordinates(meuse)[5:15,]
```

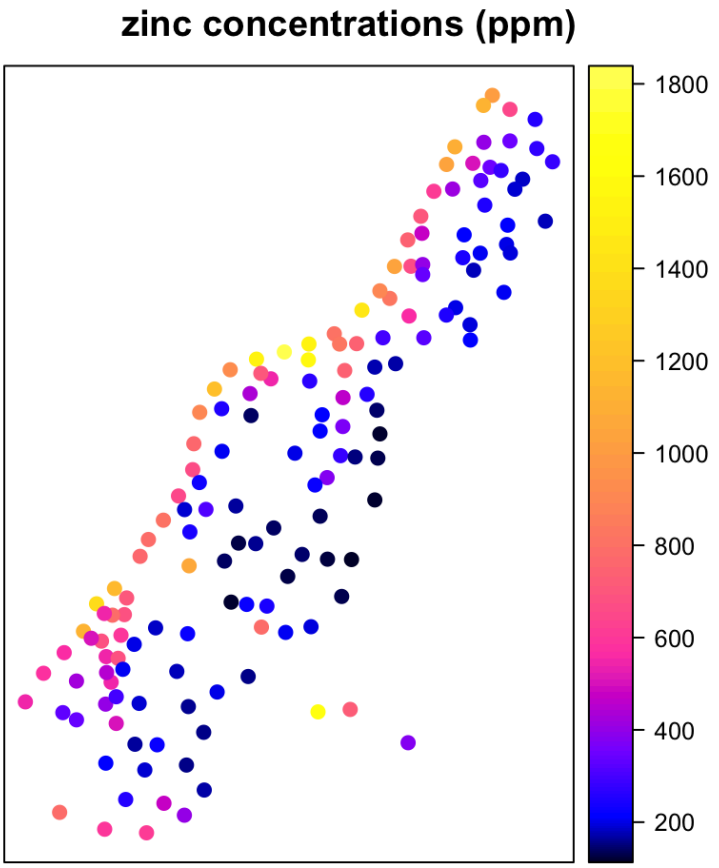
```
##         x      y
## 5  181307 333330
## 6  181390 333260
## 7  181165 333370
## 8  181027 333363
## 9  181060 333231
## 10 181232 333168
## 11 181191 333115
## 12 181032 333031
## 13 180874 333339
## 14 180969 333252
## 15 181011 333161
```

Plotting data: you can use two plotting functions spplot and bubble as illustrated below (note: the x- and y-axis are the spatial coordinates)

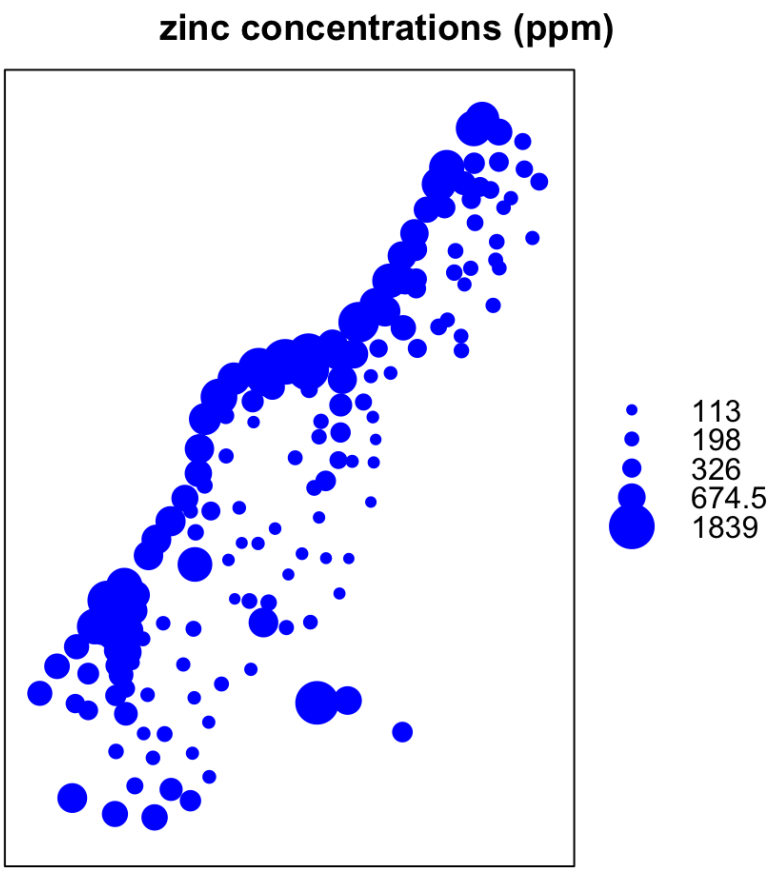
```
spplot(meuse, "zinc",  colorkey = TRUE, main = "zinc concentrations (ppm)")
```



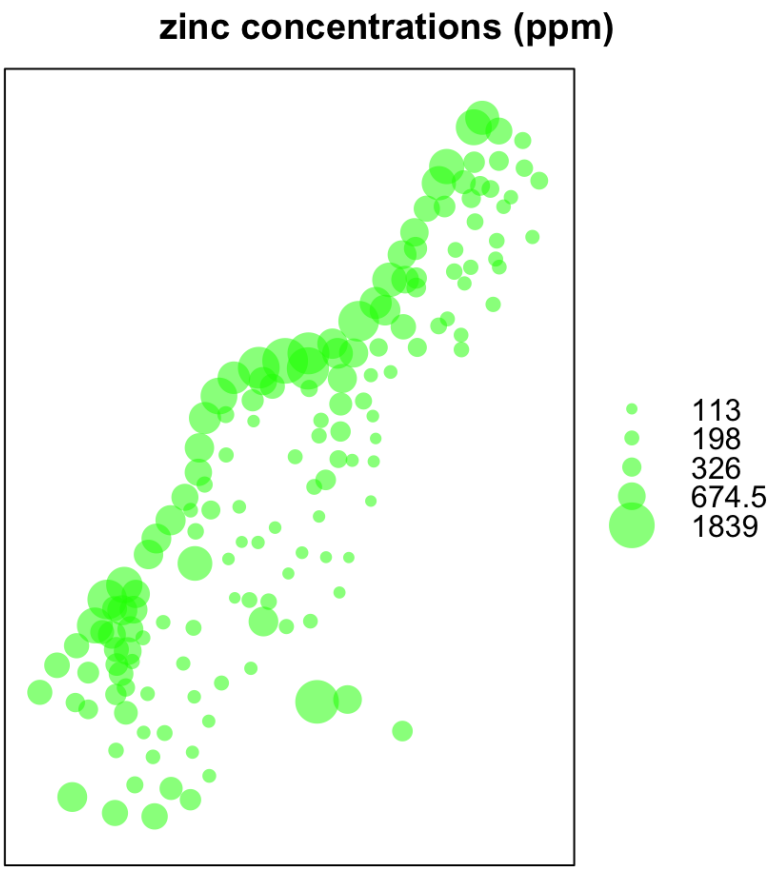

```
spplot(meuse, "zinc", do.log = T, colorkey = TRUE, main = "zinc concentrations (ppm)")
```




```
bubble(meuse, "zinc", col="blue", main = "zinc concentrations (ppm)")
```



```
bubble(meuse, "zinc", col=c("#00ff0088"), main = "zinc concentrations (ppm)")
```



Now we explore the meuse.grid dataset

```
data(meuse.grid)
summary(meuse.grid)
```

```
##        x                y              part.a           part.b
##  Min.   :178460   Min.   :329620   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:179420   1st Qu.:330460   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :179980   Median :331220   Median :0.0000   Median :1.0000
##  Mean   :179985   Mean   :331348   Mean   :0.3986   Mean   :0.6014
##  3rd Qu.:180580   3rd Qu.:332140   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :181540   Max.   :333740   Max.   :1.0000   Max.   :1.0000
##       dist        soil     ffreq
##  Min.   :0.0000   1:1665   1: 779
##  1st Qu.:0.1193   2:1084   2:1335
##  Median :0.2715   3: 354   3: 989
##  Mean   :0.2971
##  3rd Qu.:0.4402
##  Max.   :0.9926
```

```
str(meuse.grid)
```

```
## 'data.frame':    3103 obs. of  7 variables:
##  $ x     : num  181180 181140 181180 181220 181100 ...
##  $ y     : num  333740 333700 333700 333700 333660 ...
##  $ part.a: num  1 1 1 1 1 1 1 1 1 ...
##  $ part.b: num  0 0 0 0 0 0 0 0 0 ...
##  $ dist  : num  0 0 0.0122 0.0435 0 ...
##  $ soil  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 ...
##  $ ffreq : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 ...
```

```
class(meuse.grid)
```

```
## [1] "data.frame"
```

```
coordinates(meuse.grid) = ~x+y
class(meuse.grid)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
# sSee how the class of meuse.grid changes when "gridded" is used
gridded(meuse.grid) = TRUE
class(meuse.grid)
```

```
## [1] "SpatialPixelsDataFrame"
## attr(,"package")
## [1] "sp"
```

Do some visualization

```
image(meuse.grid["dist"])
title("distance to river (red = 0)")
```

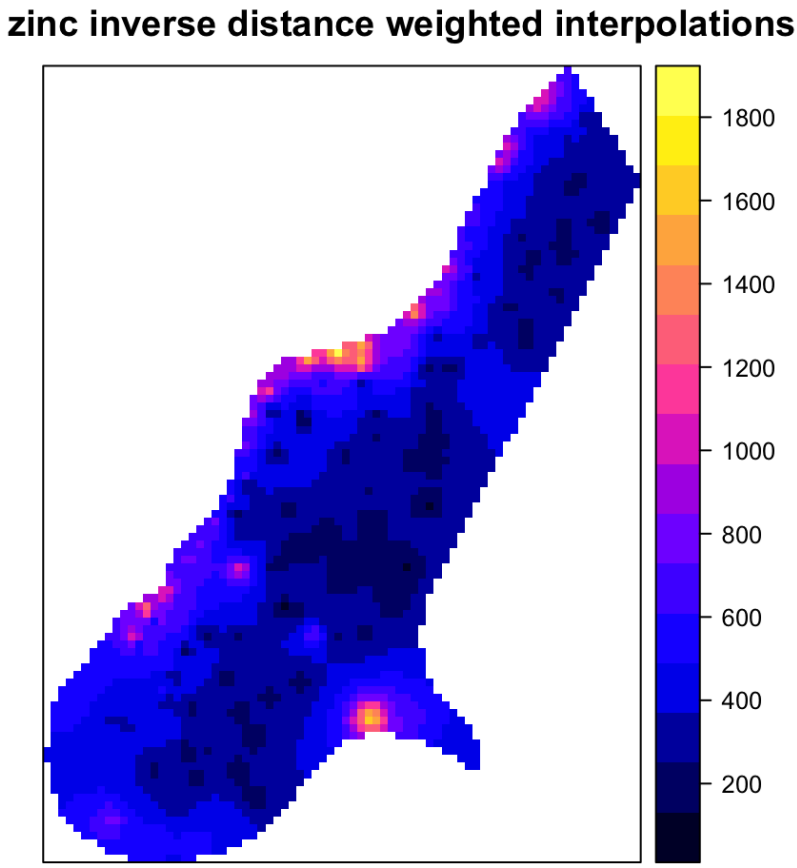

```
# Now we need package gstat
library(gstat)
# Apply the inverse distance weighted interpolation
zinc.idw = idw(zinc~1, meuse, meuse.grid)
```

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

```
# class of zinc.idw
class(zinc.idw)
```

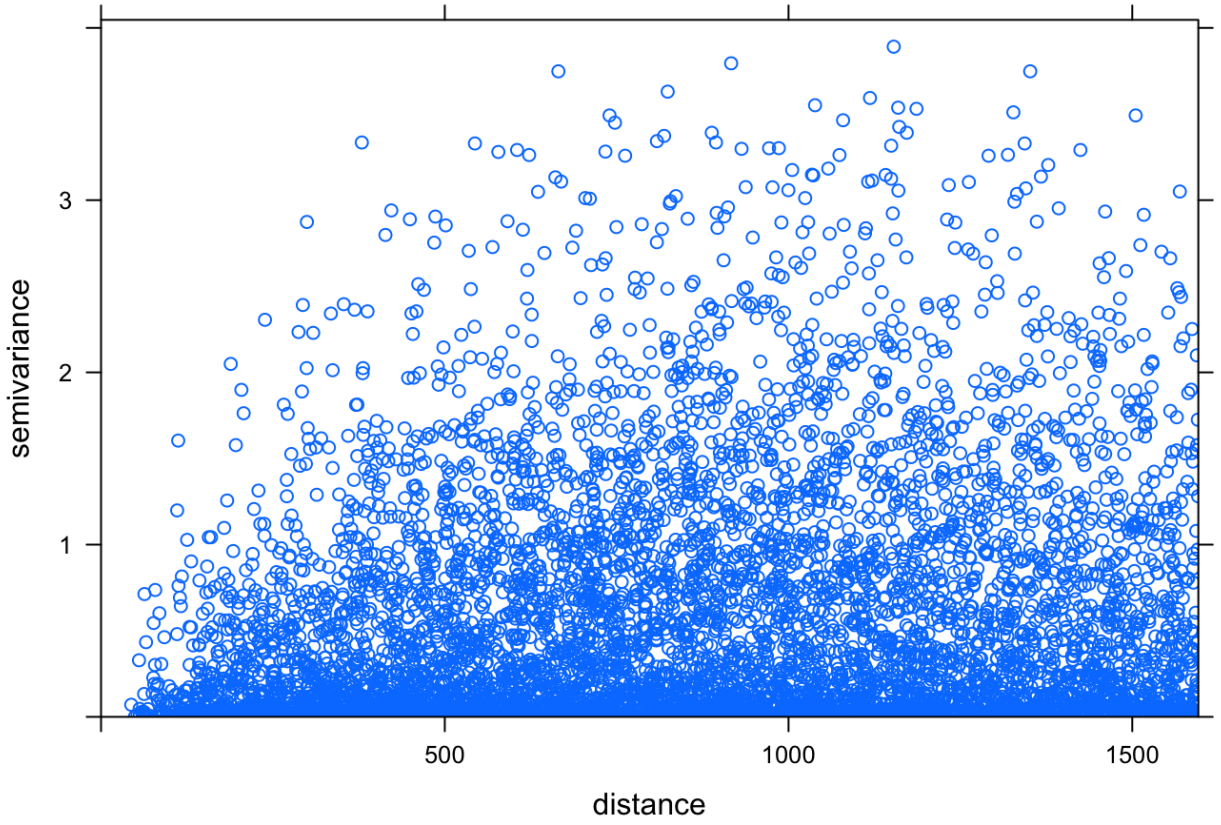
```
## [1] "SpatialPixelsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
# plot the
spplot(zinc.idw["var1.pred"], main = "zinc inverse distance weighted interpolations")
```



Variogram. We assume that there is a constant trend for the variable log(zinc)

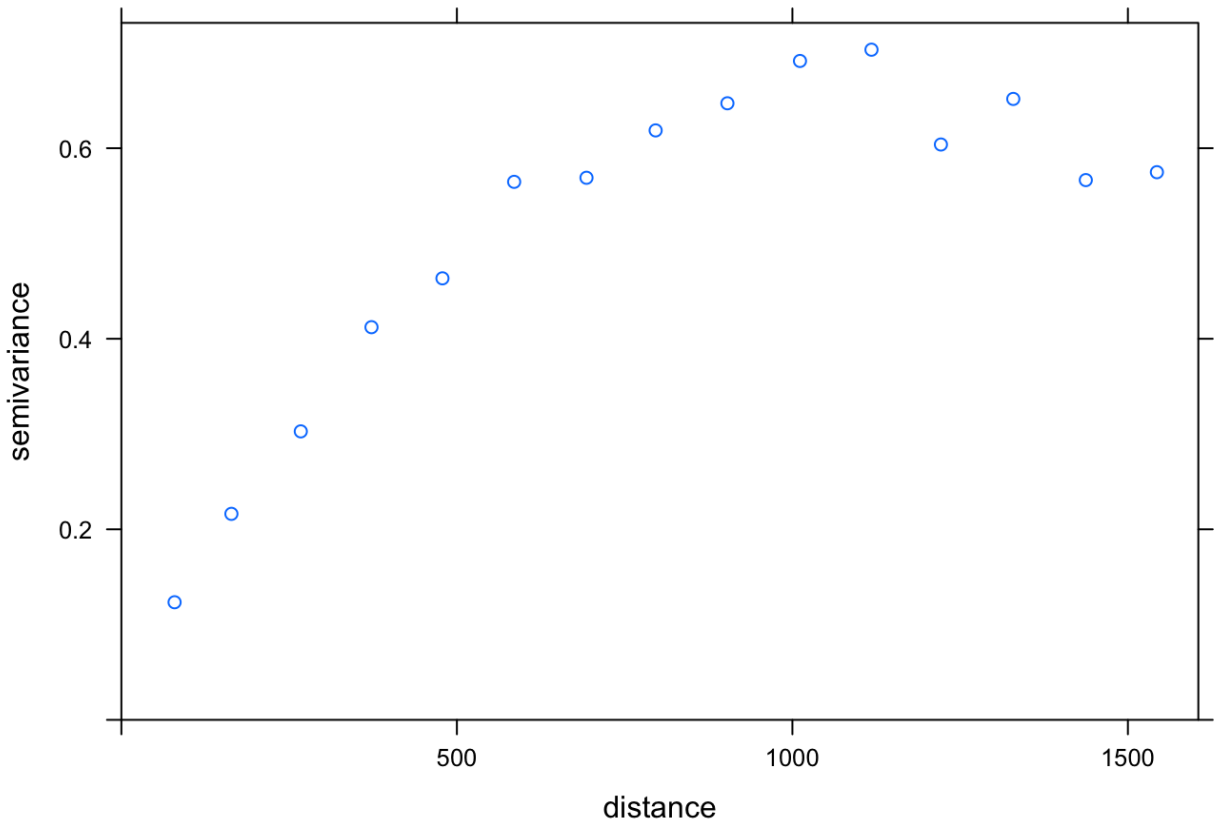
```
vgm1 = variogram(log(zinc)~1, meuse)
# plot variogram cloud
plot(variogram(log(zinc)~1, meuse, cloud=TRUE))
```




```
vgm1
```

```
##     np       dist     gamma dir.hor dir.ver   id
## 1   57   79.29244 0.1234479       0       0 var1
## 2  299  163.97367 0.2162185       0       0 var1
## 3  419  267.36483 0.3027859       0       0 var1
## 4  457  372.73542 0.4121448       0       0 var1
## 5  547  478.47670 0.4634128       0       0 var1
## 6  533  585.34058 0.5646933       0       0 var1
## 7  574  693.14526 0.5689683       0       0 var1
## 8  564  796.18365 0.6186769       0       0 var1
## 9  589  903.14650 0.6471479       0       0 var1
## 10 543 1011.29177 0.6915705       0       0 var1
## 11 500 1117.86235 0.7033984       0       0 var1
## 12 477 1221.32810 0.6038770       0       0 var1
## 13 452 1329.16407 0.6517158       0       0 var1
## 14 457 1437.25620 0.5665318       0       0 var1
## 15 415 1543.20248 0.5748227       0       0 var1
```

```
# plot vgm1
plot(vgm1)
```

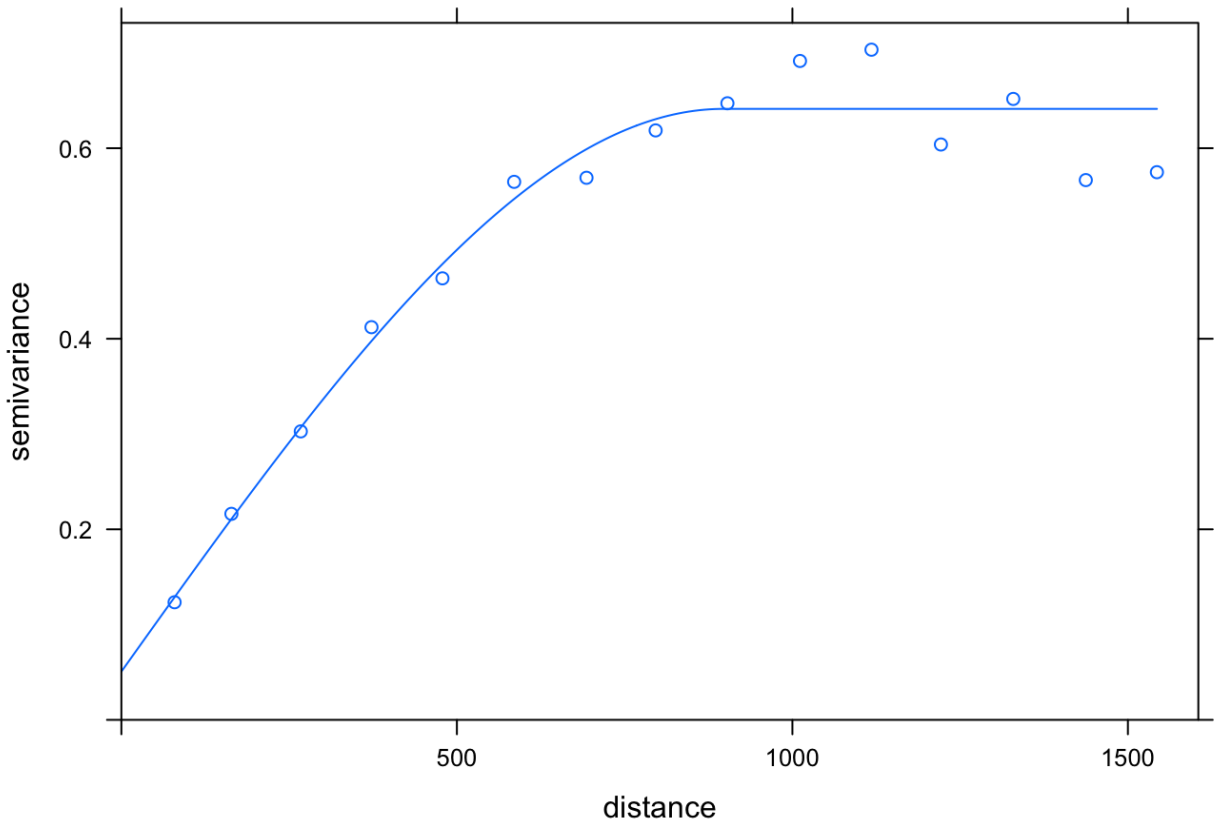


Variogram. Fit a theoretical variogram.

```
vgm1.fit = fit.variogram(vgm1, model = vgm(1, "Sph", 900, 1))
vgm1.fit
```

```
##   model      psill    range
## 1   Nug 0.05066243   0.0000
## 2   Sph 0.59060780 897.0209
```

```
# plot the fitted variogram and the observed variogram on the same graph
plot(vgm1, vgm1.fit)
```

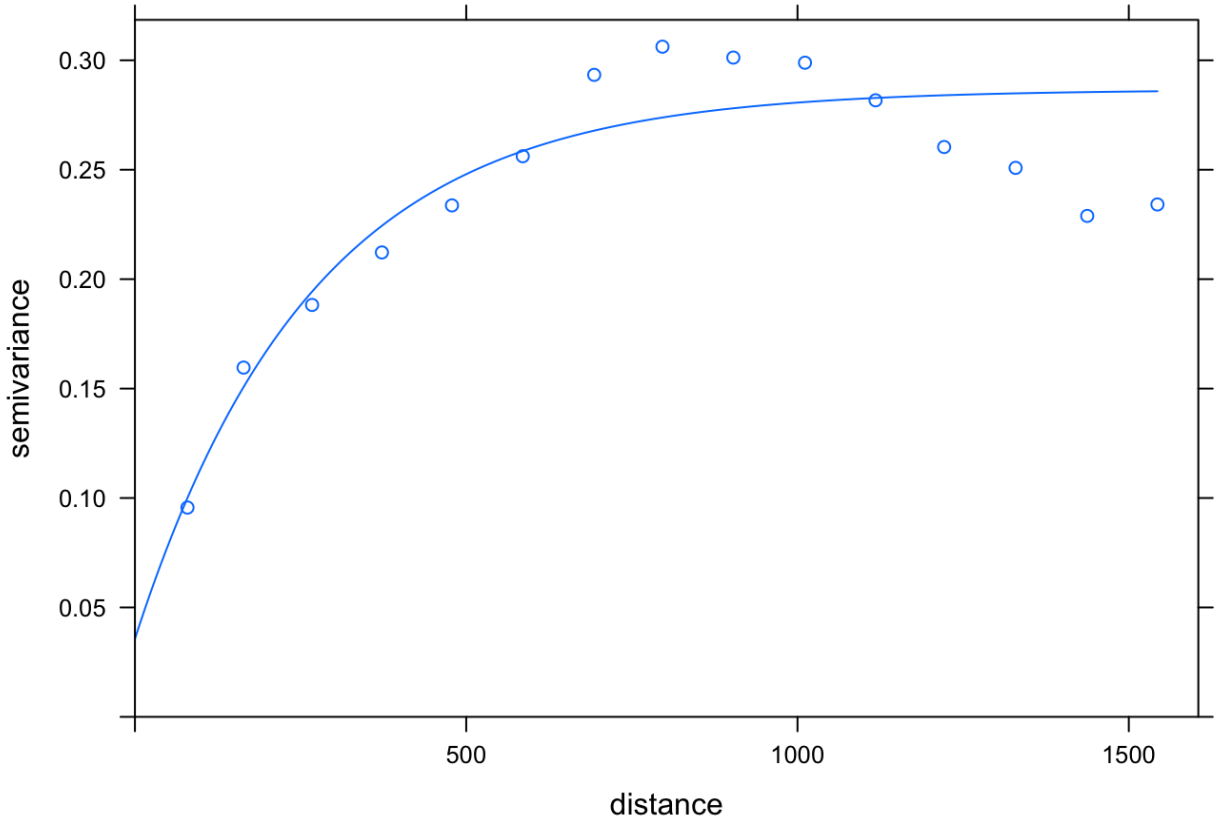


Variogram. Now we can specify a mean function for the variogram, e.g. using ~ dist as a predictor variable

```
vgm2 = variogram(log(zinc)~dist, meuse)
vgm2.fit = fit.variogram(vgm2, model = vgm(1, "Exp", 300, 1))
vgm2.fit
```

```
##   model      psill    range
## 1   Nug 0.03563295   0.0000
## 2   Exp 0.25099411 267.2971
```

```
plot(vgm2, vgm2.fit)
```

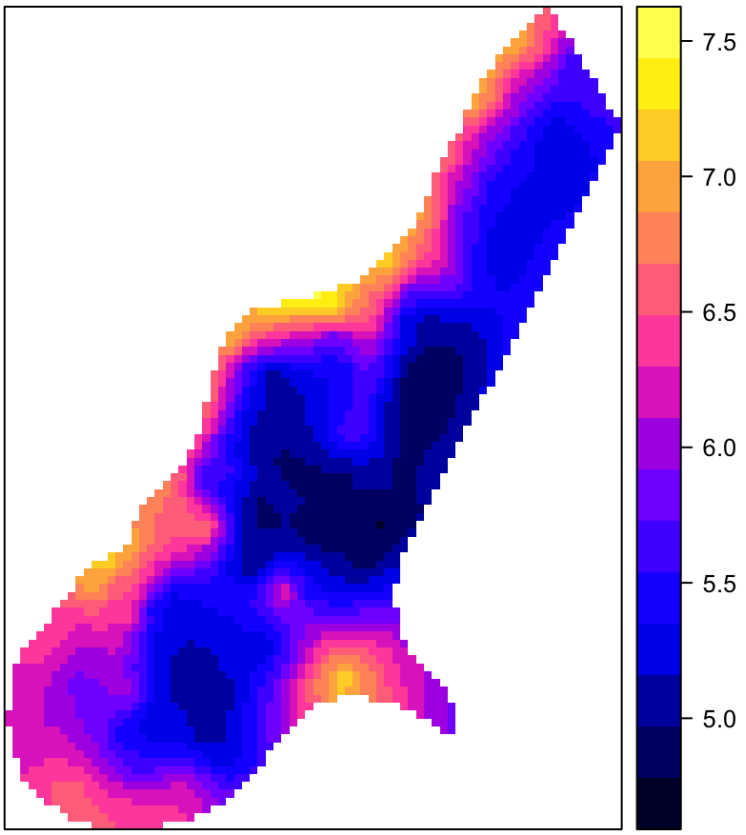


Kriging. We do an ordinary kriging.

```
vgm1.kriged = krige(log(zinc)~1, meuse, meuse.grid, model = vgm1.fit)
```

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

```
spplot(vgm1.kriged["var1.pred"])
```

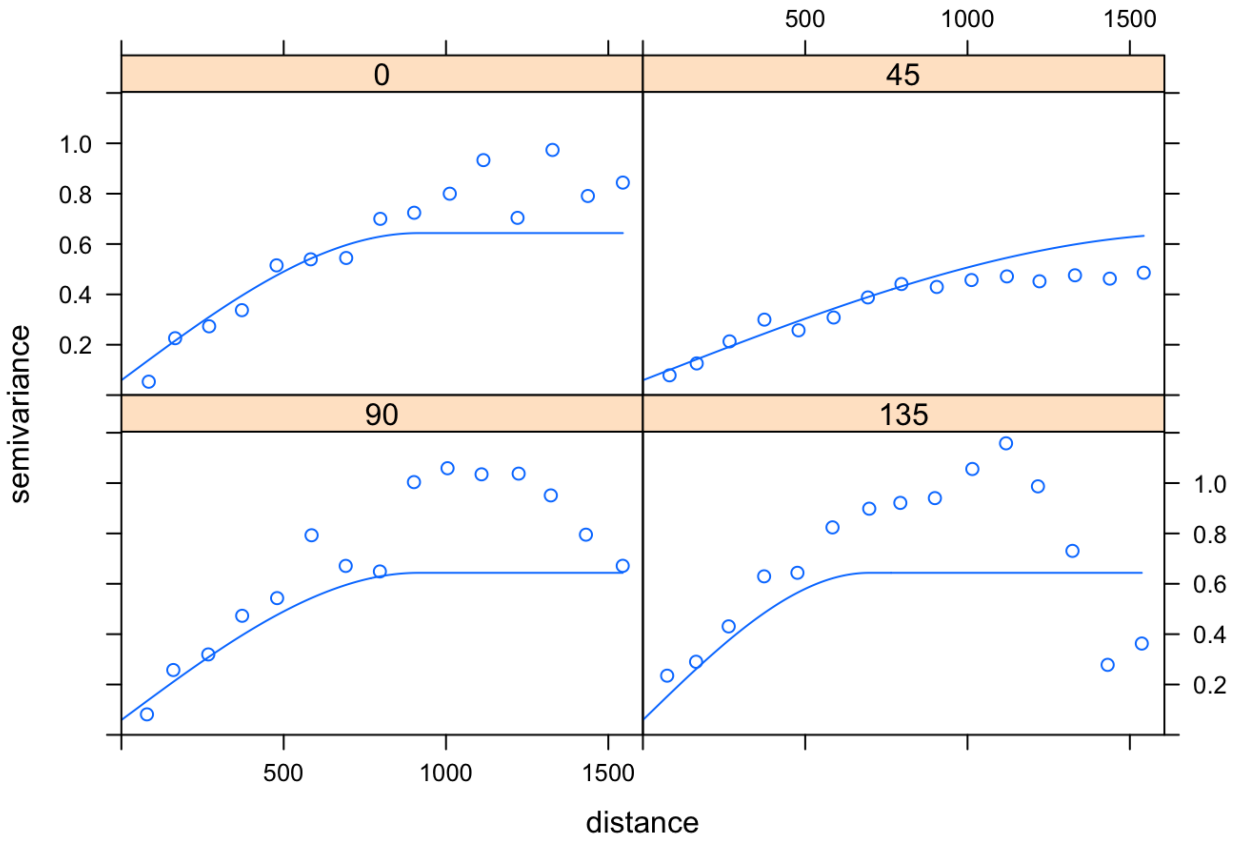


Directional variogram

```
vgm3 = variogram(log(zinc)~1, meuse, alpha = c(0, 45, 90, 135))
vgm3.fit = fit.variogram(vgm3, model = vgm(.59, "Sph", 1200, .05, anis = c(45, .4)))
vgm3.fit
```

```
##   model      psill    range ang1 anis1
## 1   Nug 0.05892218    0.000    0   1.0
## 2   Sph 0.58465000 1742.803   45   0.4
```

```
plot(vgm3, vgm3.fit, as.table = TRUE)
```

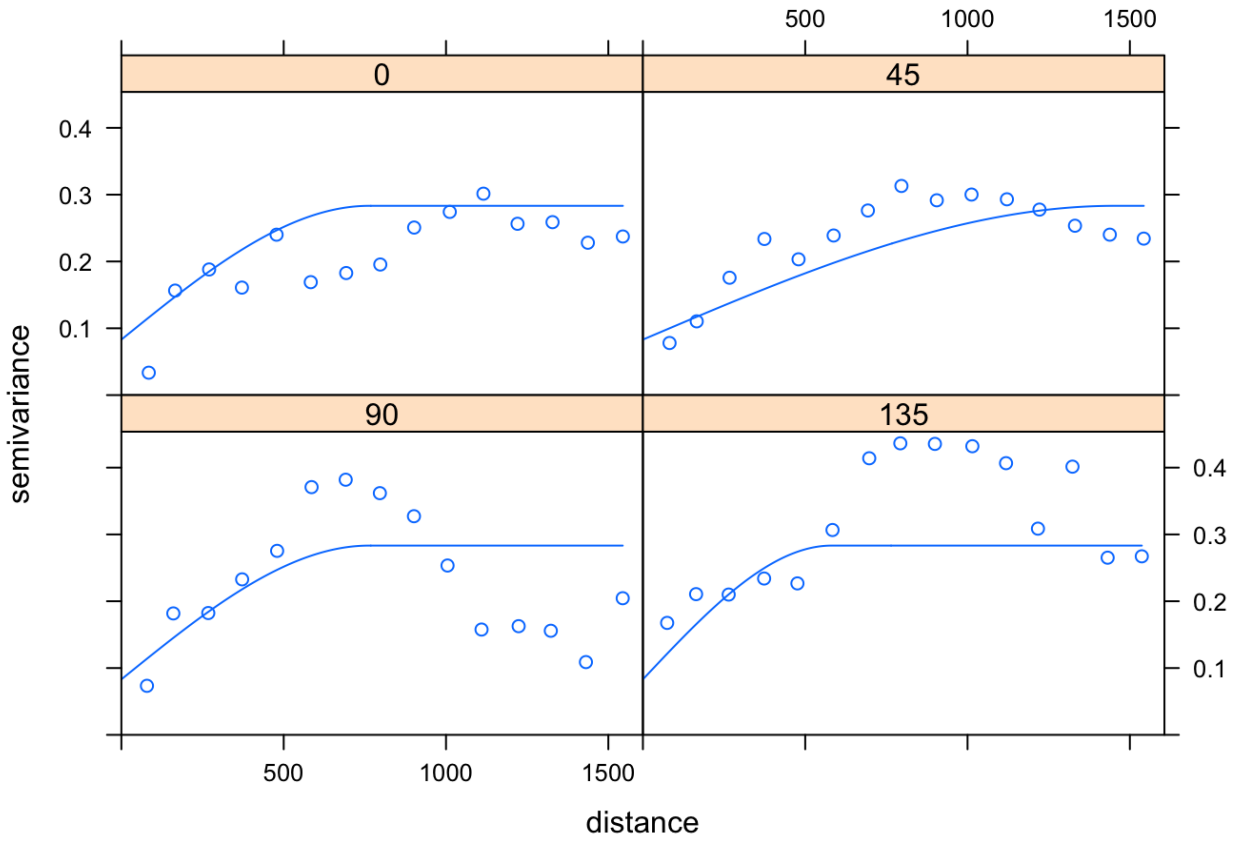


Another directional variogram

```
vgm4 = variogram(log(zinc)~dist, meuse, alpha = c(0, 45, 90, 135))
vgm4.fit = fit.variogram(vgm4, model = vgm(.59, "Sph", 1200, .05, anis = c(45, .4)))
vgm4.fit
```

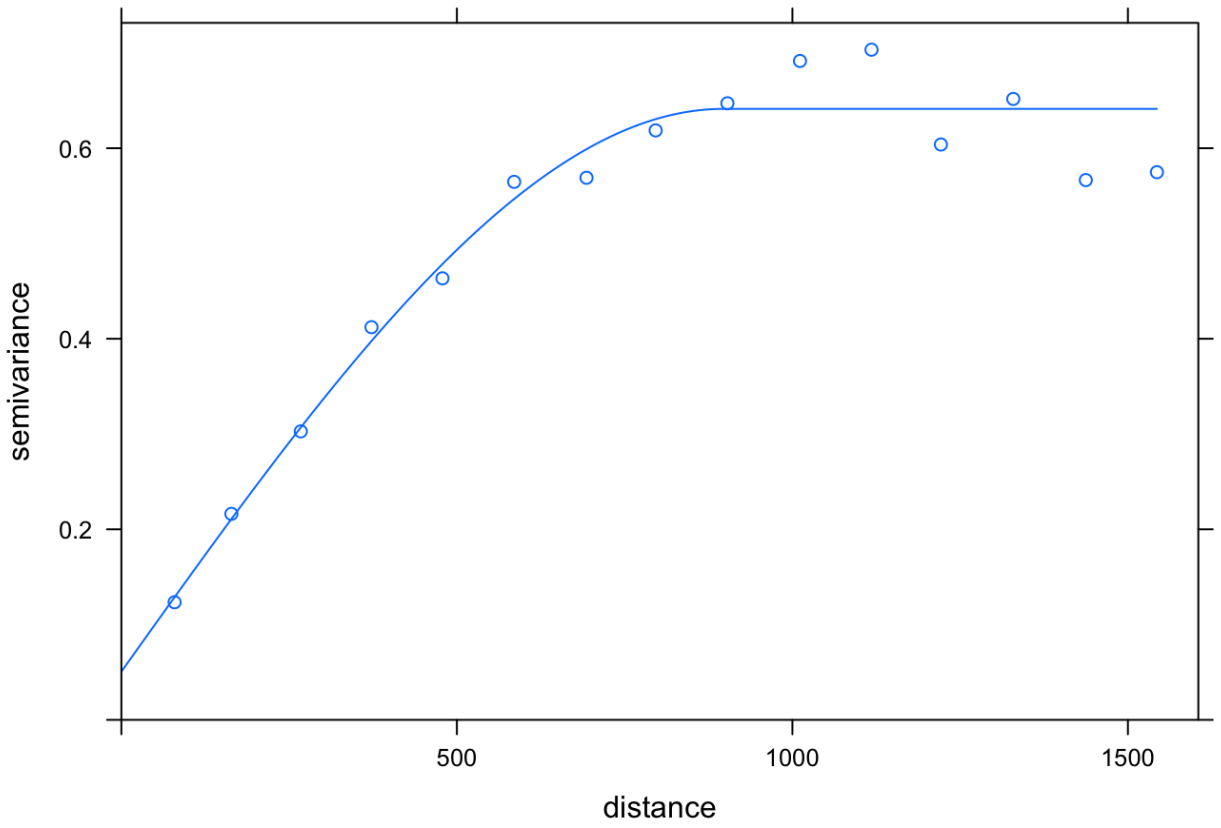
```
##   model      psill    range ang1 anis1
## 1   Nug 0.08305825    0.000    0   1.0
## 2   Sph 0.20023979 1452.826   45   0.4
```

```
plot(vgm4, vgm4.fit, as.table = TRUE)
```



Blocking Kriging

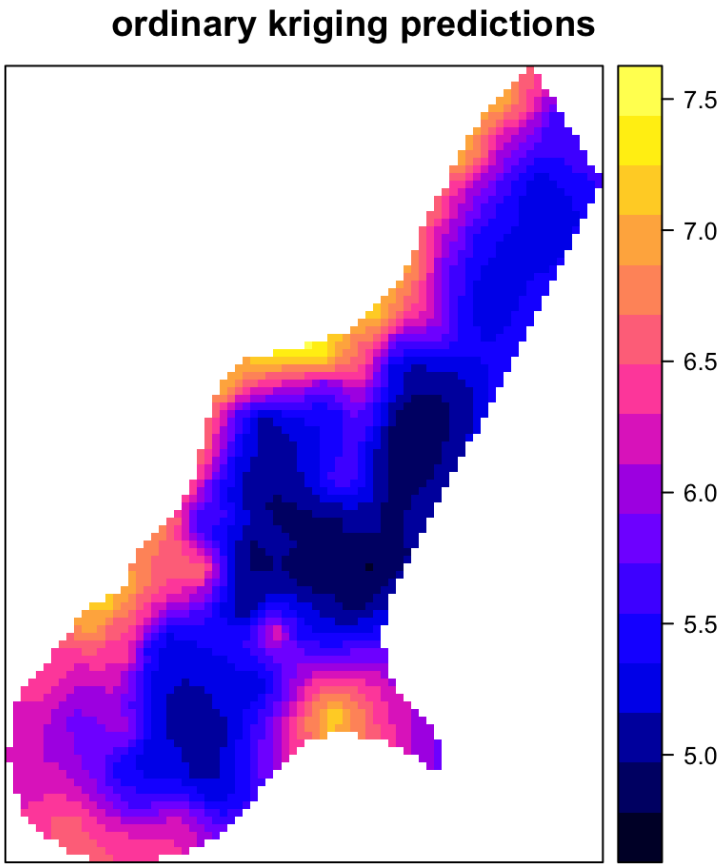
```
vgm5<- variogram(log(zinc)~1, meuse)
vgm5.fit <- fit.variogram(vgm5, model=vgm(psill=1, model="Sph", range=900, nugget=1))

plot(vgm5, vgm5.fit)
```



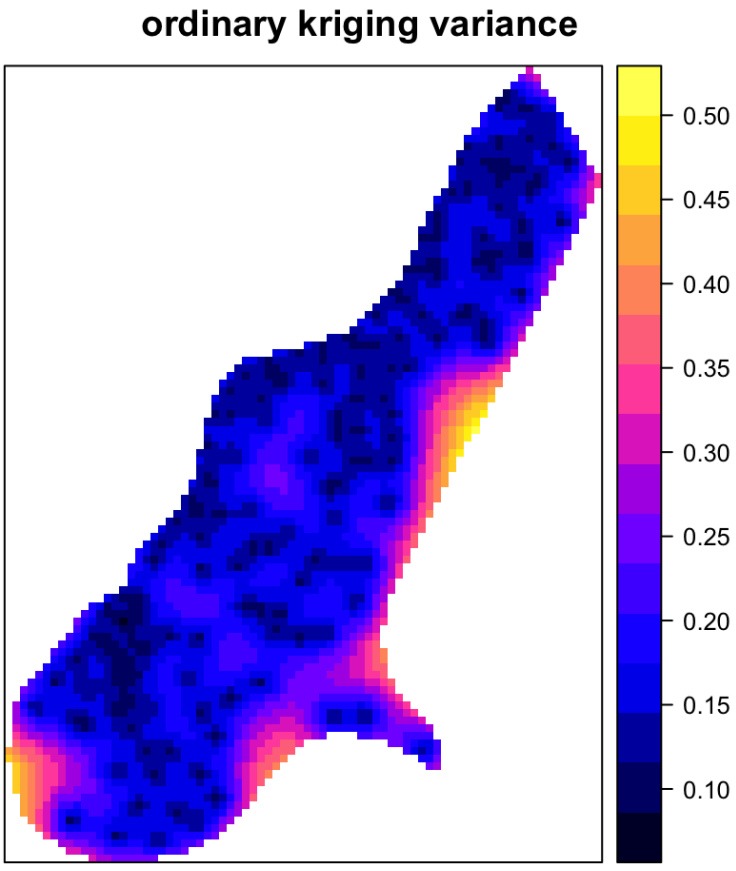

```
lzn.okriging <- krige(log(zinc)~1, meuse, meuse.grid, model = vgm5.fit )
```

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

```
spplot(lzn.okriging["var1.pred"], main = "ordinary kriging predictions")
```



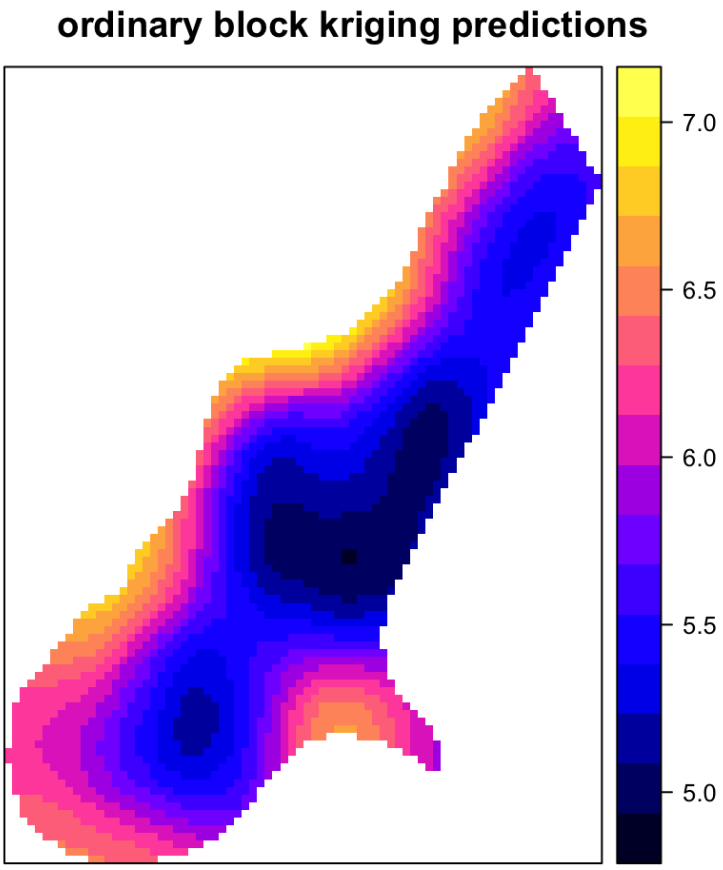

```
spplot(lzn.okriging["var1.var"],  main = "ordinary kriging variance")
```



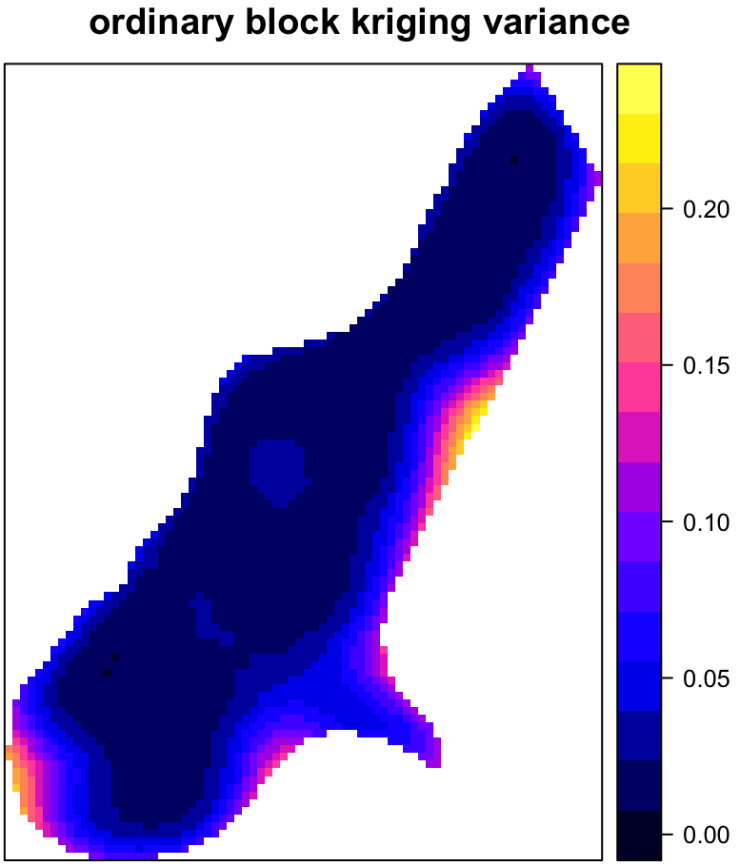
```
#Define a block size for block kriging
lzn.bokriging <- krige(log(zinc)~1, meuse, meuse.grid, model = vgm5.fit, block=c(500,500))
```

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

```
spplot(lzn.bokriging["var1.pred"], main = "ordinary block kriging predictions")
```

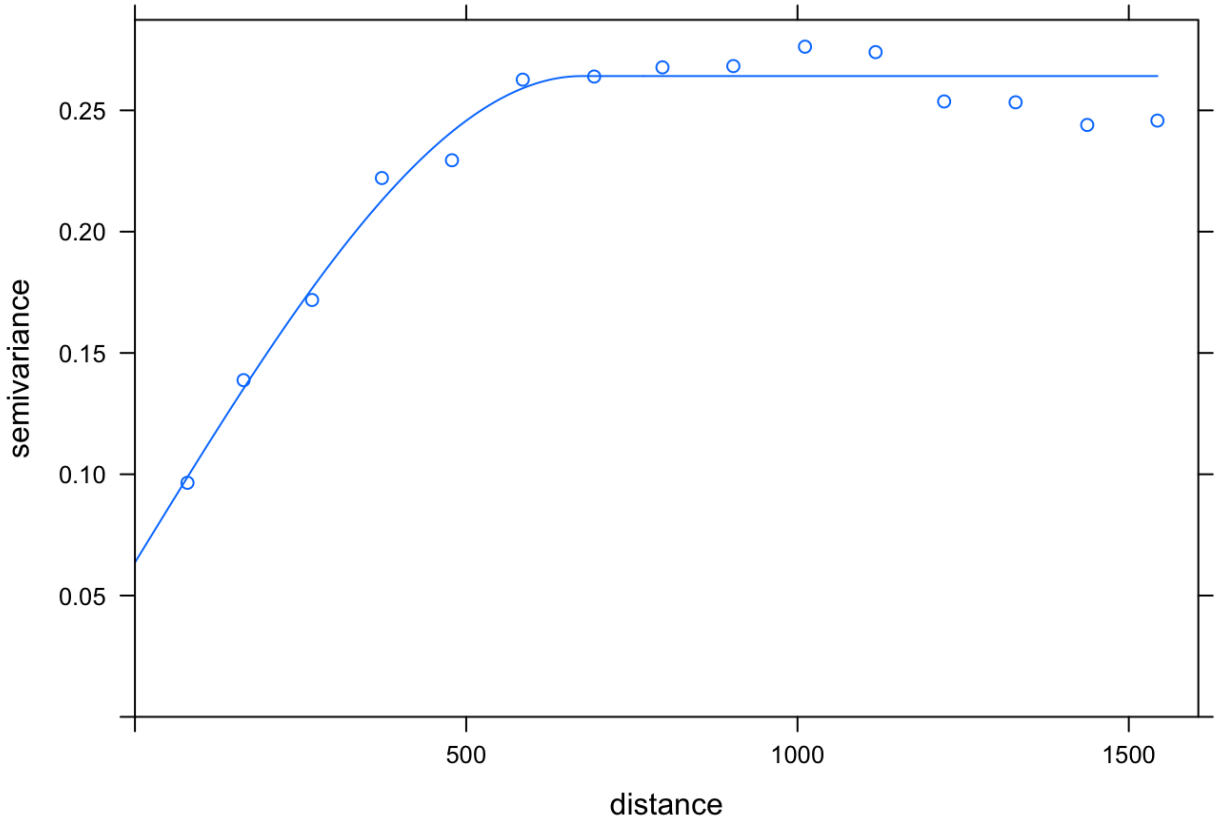



```
spplot(lzn.bokriging["var1.var"],  main = "ordinary block kriging variance")
```



Indicator Kriging

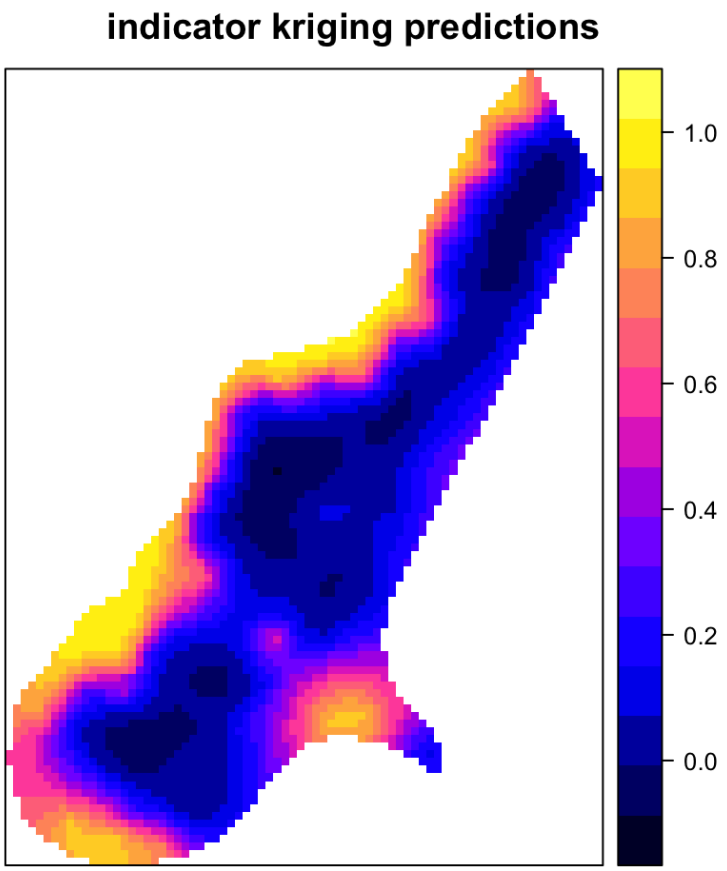
```
#Develop a variogram for indicator kriging
vgm6 <- variogram(I(log(zinc)>6)~1, meuse)
vgm6.fit <- fit.variogram(vgm6, model=vgm(psill=0.25, model="Sph", range=900, nugget=0.1))
plot(vgm6, vgm6.fit)
```



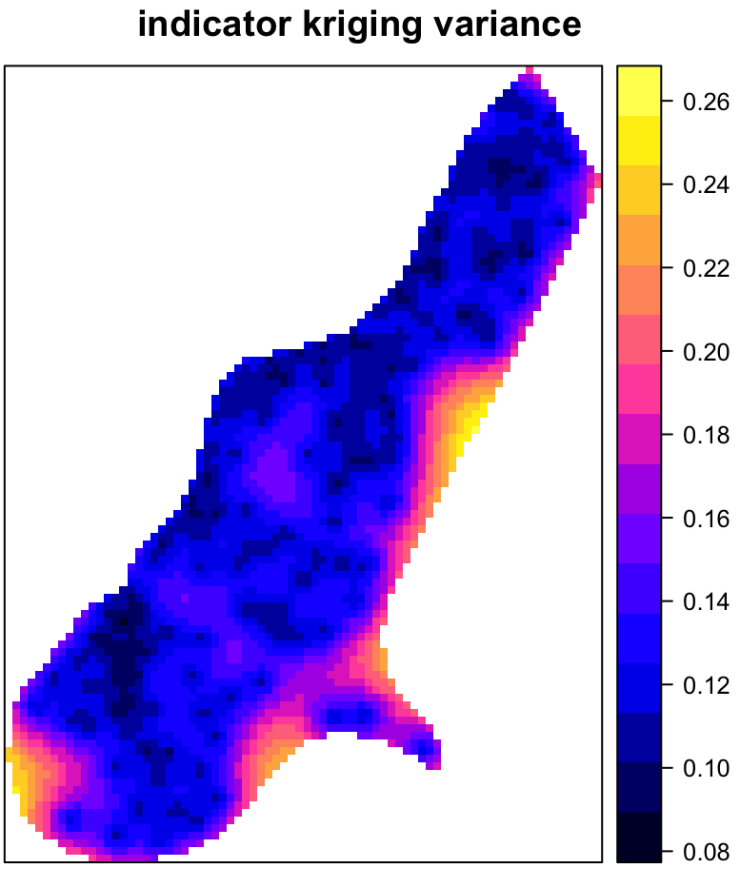

```
logzinc.ikriging <- krige(I(log(zinc)>6.21)~1, meuse, meuse.grid, vgm6.fit)
```

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

```
spplot(logzinc.ikriging["var1.pred"], main = "indicator kriging predictions")
```




```
spplot(logzinc.ikriging["var1.var"],  main = "indicator kriging variance")
```

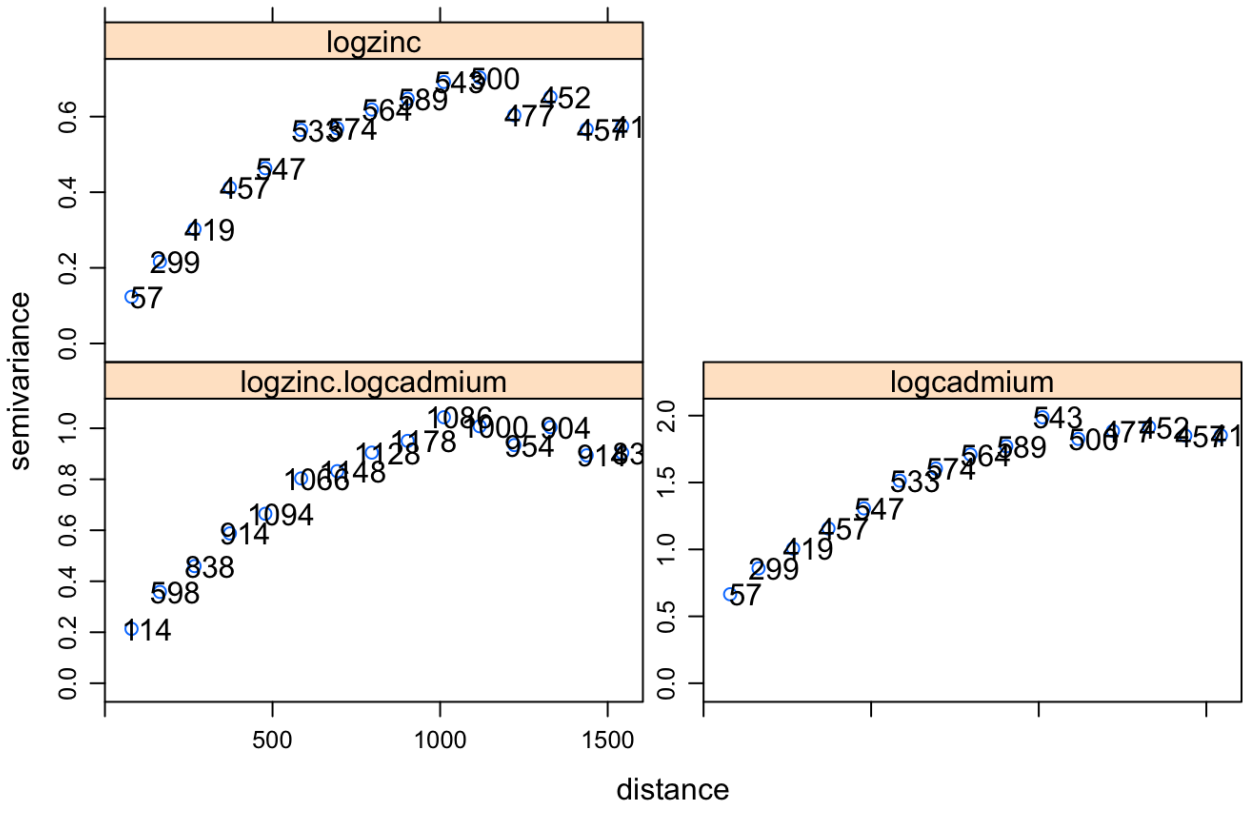


Co-Kriging

```
# Build a gstat structure containing the two sample sets
zinccadmium <- gstat(NULL, id = "logzinc", form = log(zinc) ~ 1, data=meuse)
zinccadmium <- gstat(zinccadmium, id = "logcadmium", form = log(cadmium) ~ 1, data=meuse)

# Compute and display the two direct variograms and one cross-variogram
v.cross <- variogram(zinccadmium)
str(v.cross)
```

```
## Classes 'gstatVariogram' and 'data.frame':   45 obs. of  6 variables:
##  $ np     : num  114 598 838 914 1094 ...
##  $ dist   : num  79.3 164 267.4 372.7 478.5 ...
##  $ gamma  : num  0.213 0.358 0.46 0.588 0.665 ...
##  $ dir.hor: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ dir.ver: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ id     : Factor w/ 3 levels "logzinc.logcadmium",..: 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "direct")='data.frame':   3 obs. of  2 variables:
##   ..$ id       : Factor w/ 3 levels "logcadmium","logzinc",..: 3 1 2
##   ..$ is.direct: logi  FALSE TRUE TRUE
##  - attr(*, "boundaries")= num  0 106 213 319 426 ...
##  - attr(*, "pseudo")= num 0
##  - attr(*, "what")= chr "semivariance"
```

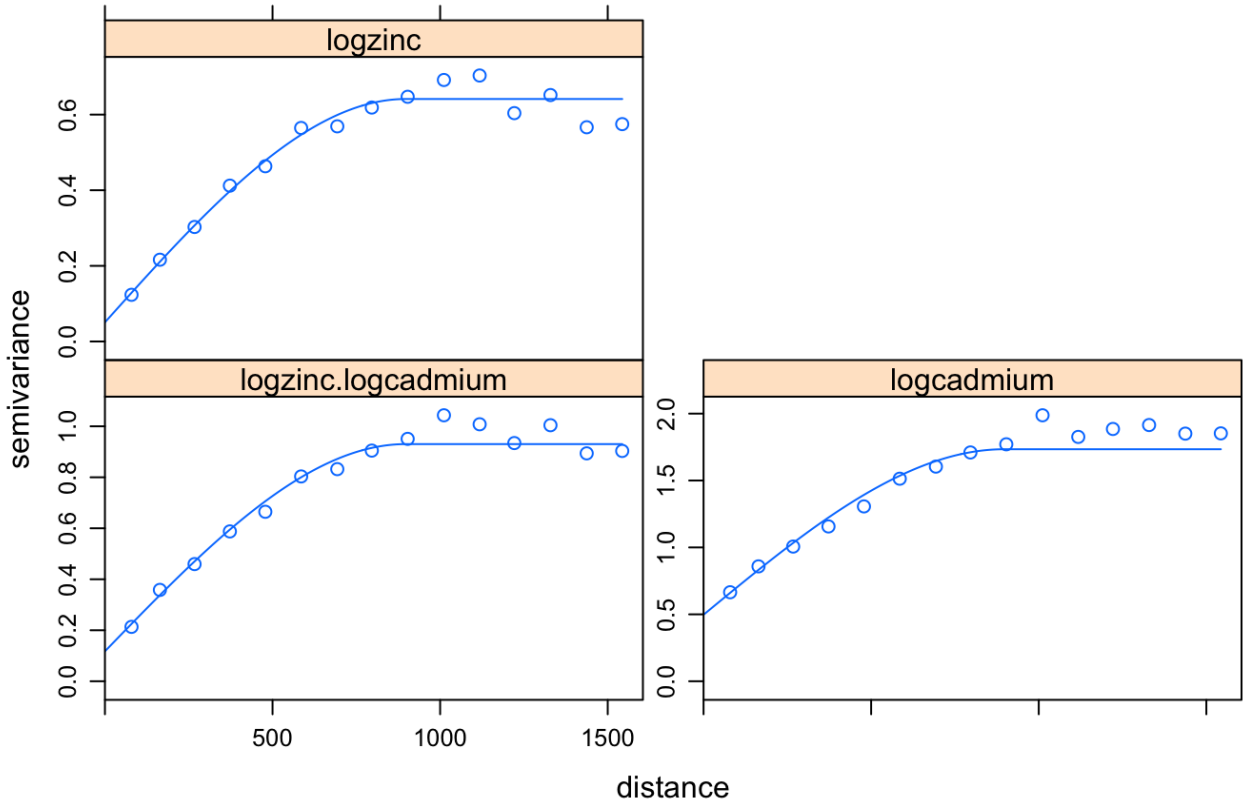
```
plot(v.cross, pl=T)
```



```
# Add variogram models to the gstat object and fit a them using the linear model of co-regionalisation
zinccadmium <- gstat(zinccadmium, id = "logzinc", model = vgm5.fit, fill.all=T)
zinccadmium
```

```
## data:
## logzinc : formula = log(zinc)`~`1 ; data dim = 155 x 12
## logcadmium : formula = log(cadmium)`~`1 ; data dim = 155 x 12
## variograms:
##                       model      psill    range
## logzinc[1]              Nug 0.05066243   0.0000
## logzinc[2]              Sph 0.59060780 897.0209
## logcadmium[1]           Nug 0.05066243   0.0000
## logcadmium[2]           Sph 0.59060780 897.0209
## logzinc.logcadmium[1]   Nug 0.05066243   0.0000
## logzinc.logcadmium[2]   Sph 0.59060780 897.0209
```

```
# use the fit.lmc method to fit all three variograms together
zinccadmium <- fit.lmc(v.cross, zinccadmium)
plot(variogram(zinccadmium), model=zinccadmium$model)
```




```
# use the predict.gstat method for co-kriging
cokrig.model <- predict.gstat(zinccadmium, meuse.grid)
```

```
## Linear Model of Coregionalization found. Good.
## [using ordinary cokriging]
```

```
str(cokrig.model)
```

```
## Formal class 'SpatialPixelsDataFrame' [package "sp"] with 7 slots
##   ..@ data       :'data.frame':  3103 obs. of  5 variables:
##   .. ..$ logzinc.pred          : num [1:3103] 6.47 6.6 6.48 6.35 6.74 ...
##   .. ..$ logzinc.var           : num [1:3103] 0.318 0.25 0.271 0.294 0.175 ...
##   .. ..$ logcadmium.pred       : num [1:3103] 1.62 1.81 1.66 1.5 2.01 ...
##   .. ..$ logcadmium.var        : num [1:3103] 1.103 0.967 1.008 1.053 0.819 ...
##   .. ..$ cov.logzinc.logcadmium: num [1:3103] 0.491 0.398 0.427 0.458 0.296 ...
##   ..@ coords.nrs : int [1:2] 1 2
##   ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
##   .. .. ..@ cellcentre.offset: Named num [1:2] 178460 329620
##   .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
##   .. .. ..@ cellsize         : Named num [1:2] 40 40
##   .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
##   .. .. ..@ cells.dim        : Named int [1:2] 78 104
##   .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
##   ..@ grid.index : int [1:3103] 69 146 147 148 223 224 225 226 300 301 ...
##   ..@ coords     : num [1:3103, 1:2] 181180 181140 181180 181220 181100 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:3103] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "x" "y"
##   ..@ bbox       : num [1:2, 1:2] 178460 329620 181540 333740
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "x" "y"
##   .. .. ..$ : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##   .. .. ..@ projargs: chr NA
```

```
# summarize predictions and their errors
summary(cokrig.model)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min    max
## x 178460 181540
## y 329620 333740
## Is projected: NA
## proj4string : [NA]
## Number of points: 3103
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## x            178460       40        78
## y            329620       40       104
## Data attributes:
##   logzinc.pred    logzinc.var      logcadmium.pred   logcadmium.var
##  Min.   :4.752   Min.   :0.08347   Min.   :-1.0824   Min.   :0.6055
##  1st Qu.:5.238   1st Qu.:0.13556   1st Qu.:-0.4276   1st Qu.:0.7203
##  Median :5.564   Median :0.16058   Median : 0.2234   Median :0.7744
##  Mean   :5.713   Mean   :0.18348   Mean   : 0.3034   Mean   :0.8232
##  3rd Qu.:6.185   3rd Qu.:0.20984   3rd Qu.: 0.9662   3rd Qu.:0.8791
##  Max.   :7.428   Max.   :0.49871   Max.   : 2.5737   Max.   :1.4647
##  cov.logzinc.logcadmium
##  Min.   :0.1674
##  1st Qu.:0.2398
##  Median :0.2742
##  Mean   :0.3061
##  3rd Qu.:0.3425
##  Max.   :0.7376
```

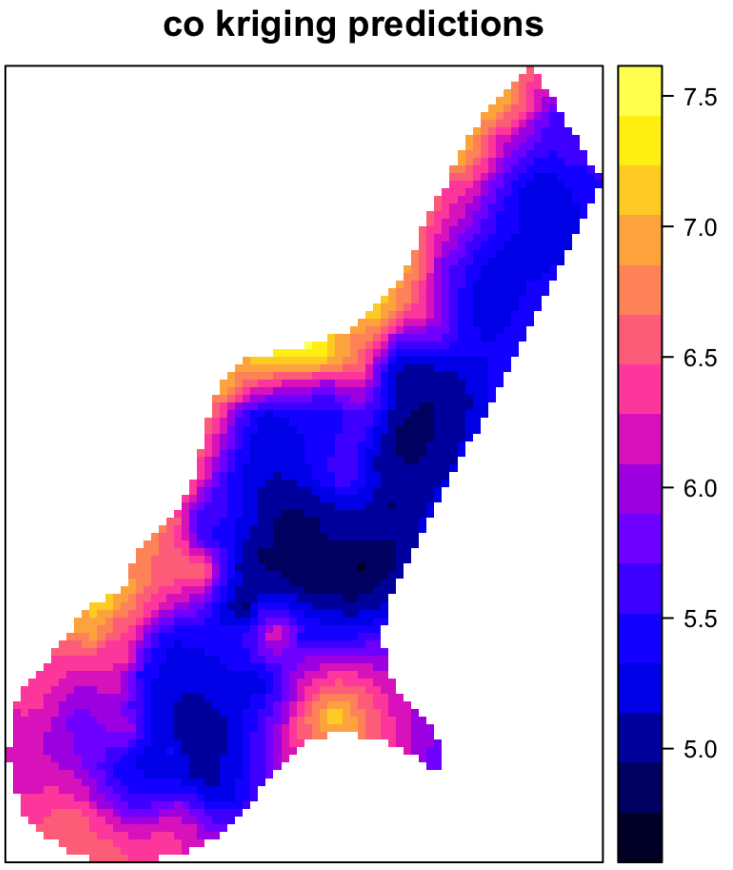
```
summary(cokrig.model$logcadmium.pred)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.0820 -0.4276  0.2234  0.3034  0.9662  2.5740
```

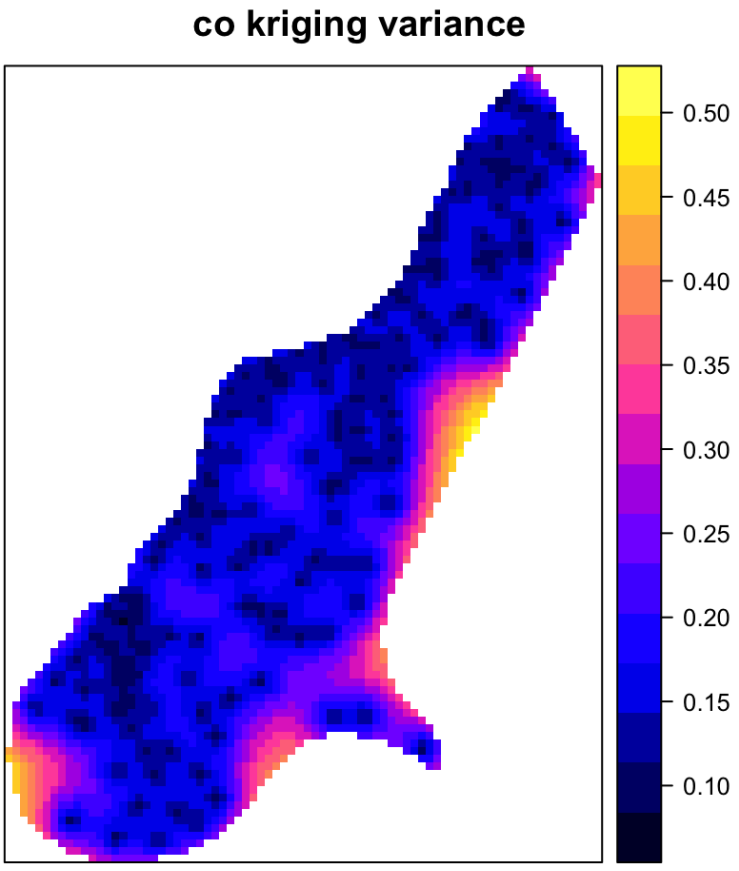
```
summary(cokrig.model$logcadmium.var)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6055  0.7203  0.7744  0.8232  0.8791  1.4650
```

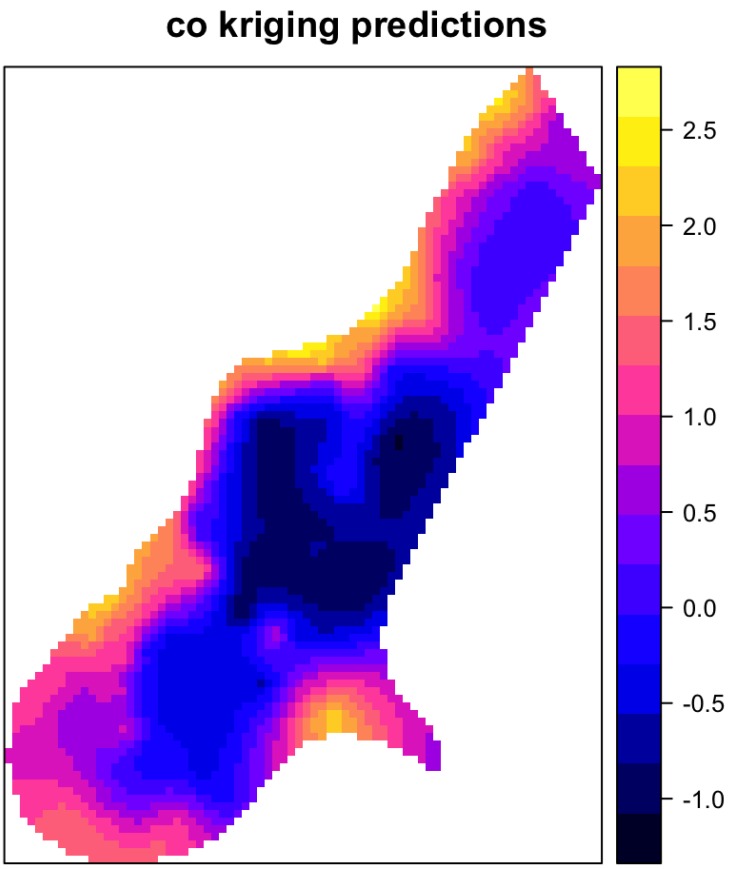
```
# Display the predictions and their errors for log(zinc) and log(cadmium)
spplot(cokrig.model["logzinc.pred"], main = "co kriging predictions")
```



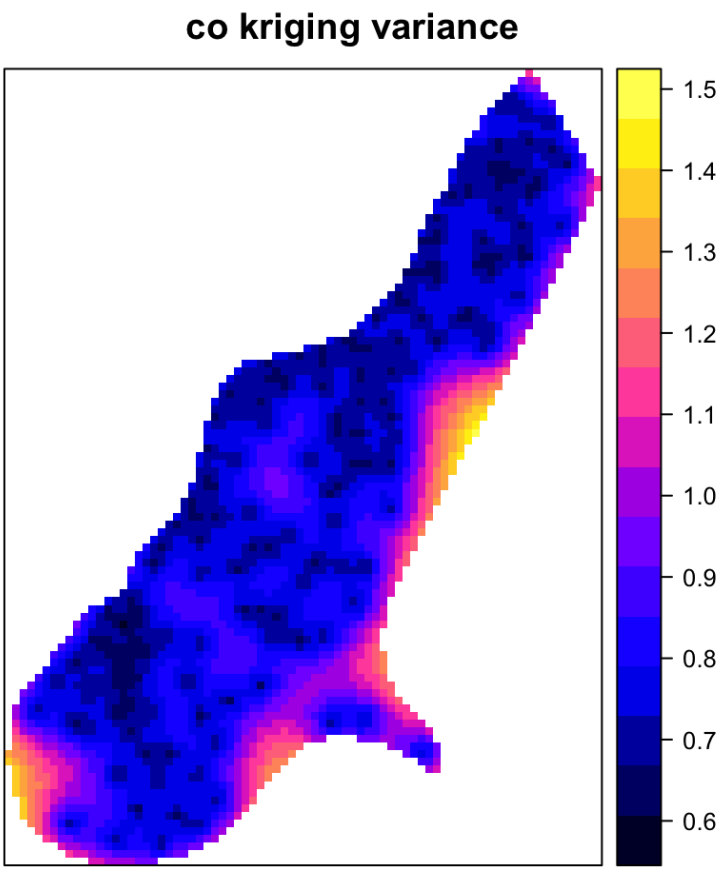

```
spplot(cokrig.model["logzinc.var"],  main = "co kriging variance")
```



```
spplot(cokrig.model["logcadmium.pred"], main = "co kriging predictions")
```




```
spplot(cokrig.model["logcadmium.var"],  main = "co kriging variance")
```



Indicator Co-Kriging

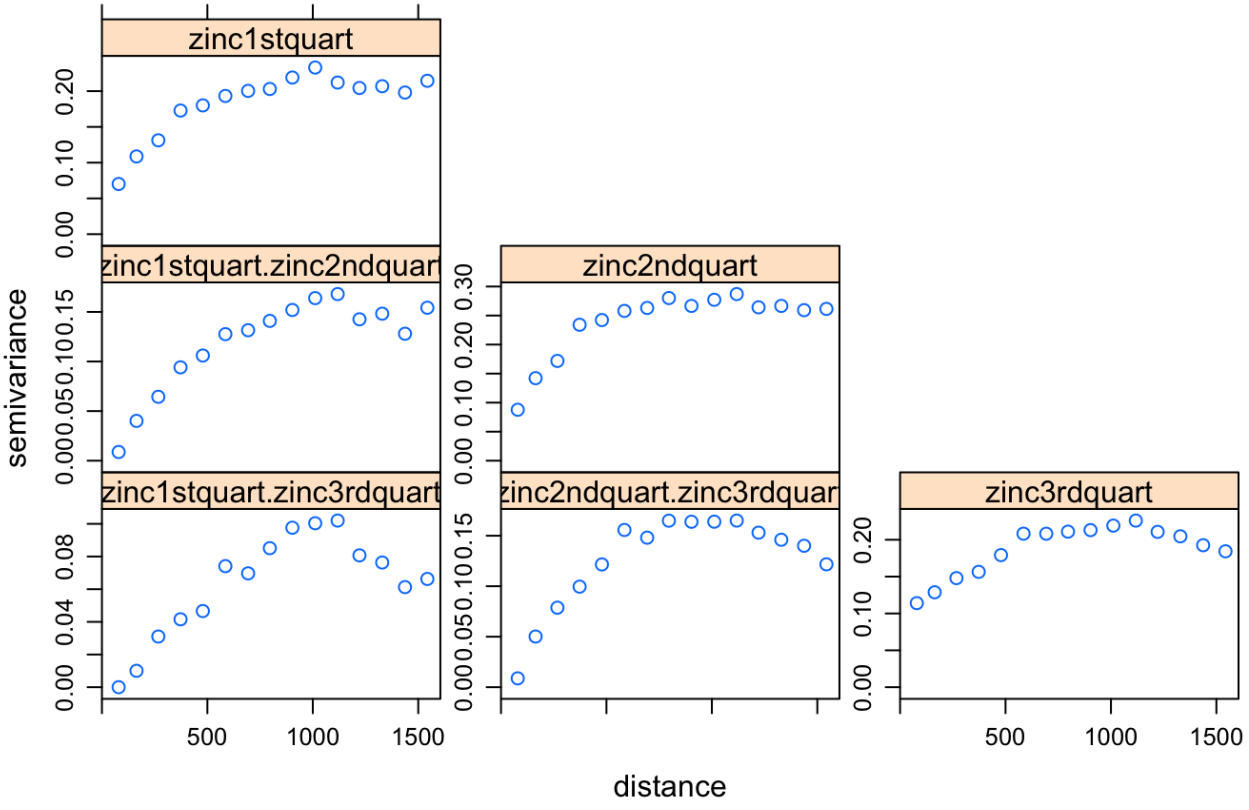
```
# # Create a categorical variable of zinc (grouped by quartile)
quartzinc <- quantile(meuse$zinc)
quartzinc
```

```
##     0%    25%    50%    75%   100%
##  113.0  198.0  326.0  674.5 1839.0
```

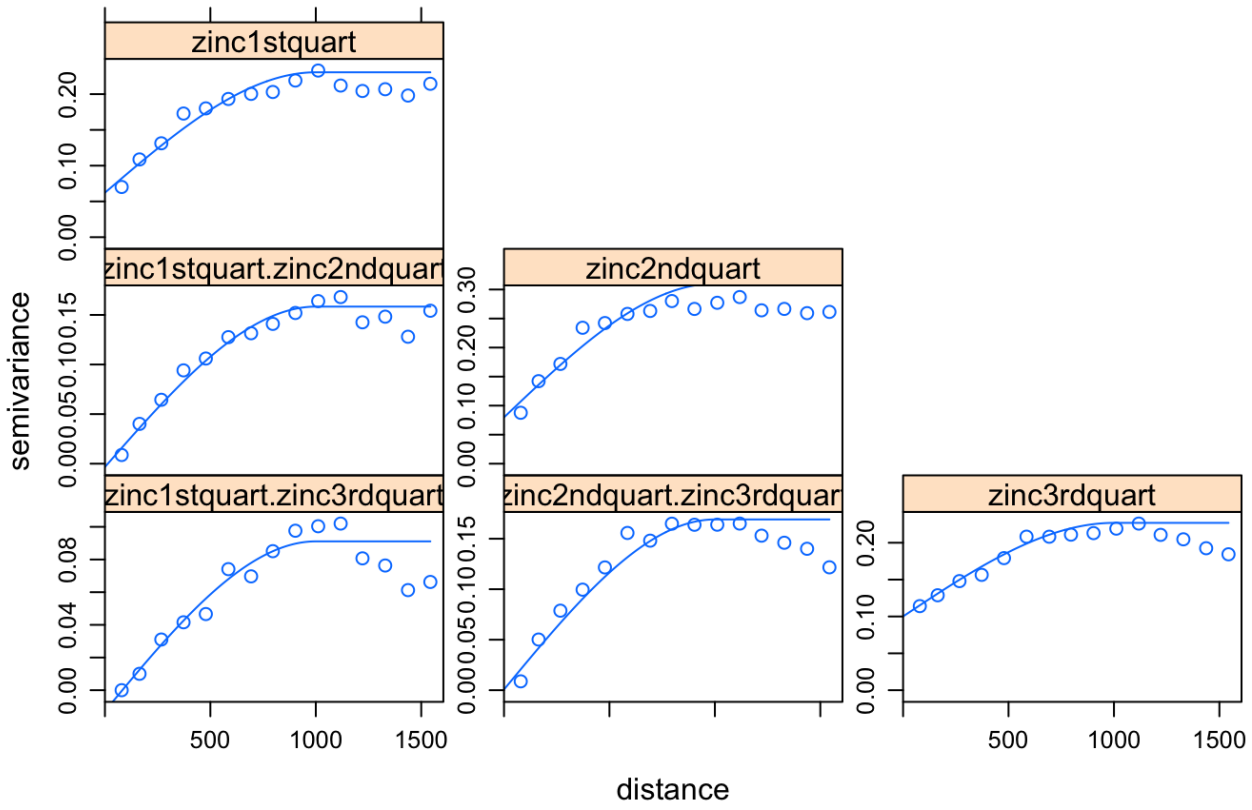
```
quartzinc[2]
```

```
## 25%
## 198
```

```
# create 3 separate datasets, one for each quartile. Example, one dataset will be 1-0 whether a zinc va
lue is below the 25th percentile. Another dataset will be 1-0 whether a zinc value is below the median
 mark.
zinc.quart <- gstat(NULL, id = "zinc1stquart", formula = I(zinc < quartzinc[2]) ~ 1, data = meuse,
                    nmax = 10, beta = 0.25, set = list(order = 4, zero = 1e-05))
zinc.quart <- gstat(zinc.quart, "zinc2ndquart", formula = I(zinc < quartzinc[3]) ~ 1, data = meuse,
                    nmax = 10, beta = 0.50)
zinc.quart <- gstat(zinc.quart, "zinc3rdquart", formula = I(zinc < quartzinc[4]) ~ 1, data = meuse,
                    nmax = 10, beta = 0.75)

# Create an initial semivariogram model with range equal 1200
zinc.quart <- gstat(zinc.quart, model = vgm(1, "Sph", 1000, 1), fill.all = T)

# Estimate the empiricalvariogram of each indicator
zincqvag <- variogram(zinc.quart)
plot(zincqvag)
```




```
# fit a single semivariogram model to all these empirical varioagrams using a linear model of coregiona
lization (the fit.lmc() function).
zinc.quartfit = fit.lmc(zincqvag, zinc.quart)
plot(zincqvag, model = zinc.quartfit)
```




```
# now do the co-kriging
cokrig.zincquart <- predict.gstat(zinc.quartfit, meuse.grid)
```

```
## Linear Model of Coregionalization found. Good.
## [using simple cokriging]
```

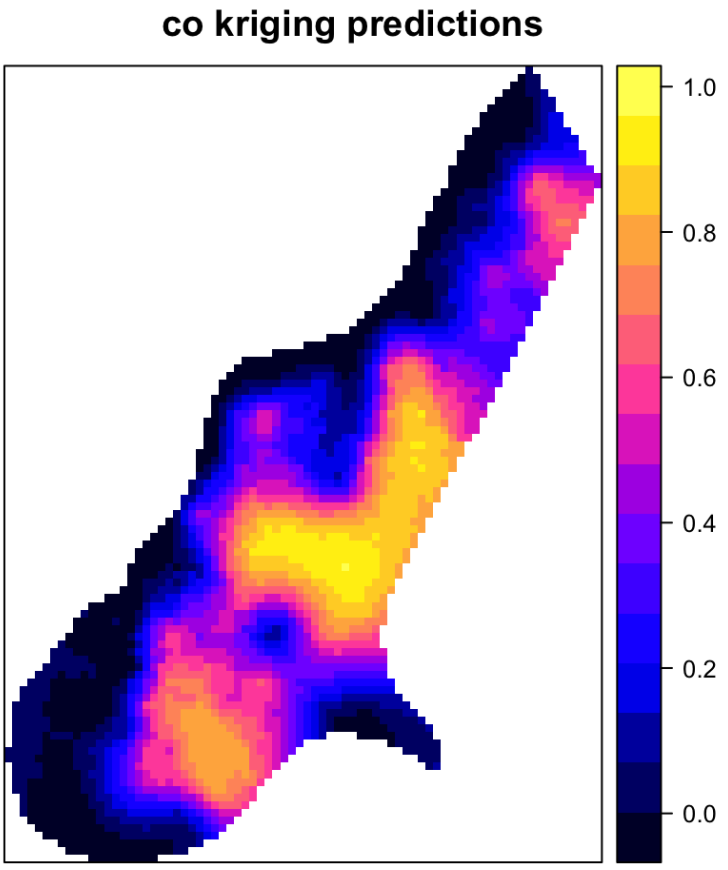
```
str(cokrig.zincquart)
```

```
## Formal class 'SpatialPixelsDataFrame' [package "sp"] with 7 slots
##   ..@ data       :'data.frame':  3103 obs. of  9 variables:
##   .. ..$ zinc1stquart.pred           : num [1:3103] 0.043046 0.000314 0.029345 0.042939 0 ...
##   .. ..$ zinc1stquart.var            : num [1:3103] 0.14 0.123 0.128 0.134 0.106 ...
##   .. ..$ zinc2ndquart.pred           : num [1:3103] 0.2503 0.1739 0.2451 0.291 0.0876 ...
##   .. ..$ zinc2ndquart.var            : num [1:3103] 0.182 0.158 0.165 0.173 0.133 ...
##   .. ..$ zinc3rdquart.pred           : num [1:3103] 0.576 0.517 0.574 0.613 0.451 ...
##   .. ..$ zinc3rdquart.var            : num [1:3103] 0.158 0.146 0.15 0.154 0.132 ...
##   .. ..$ cov.zinc1stquart.zinc2ndquart: num [1:3103] 0.064 0.0469 0.0521 0.0579 0.0285 ...
##   .. ..$ cov.zinc1stquart.zinc3rdquart: num [1:3103] 0.02845 0.01716 0.02063 0.0245 0.00495 ...
##   .. ..$ cov.zinc2ndquart.zinc3rdquart: num [1:3103] 0.0763 0.0593 0.0643 0.07 0.041 ...
##   ..@ coords.nrs : int [1:2] 1 2
##   ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
##   .. .. ..@ cellcentre.offset: Named num [1:2] 178460 329620
##   .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
##   .. .. ..@ cellsize         : Named num [1:2] 40 40
##   .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
##   .. .. ..@ cells.dim        : Named int [1:2] 78 104
##   .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
##   ..@ grid.index : int [1:3103] 69 146 147 148 223 224 225 226 300 301 ...
##   ..@ coords     : num [1:3103, 1:2] 181180 181140 181180 181220 181100 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:3103] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "x" "y"
##   ..@ bbox       : num [1:2, 1:2] 178460 329620 181540 333740
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "x" "y"
##   .. .. ..$ : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##   .. .. ..@ projargs: chr NA
```

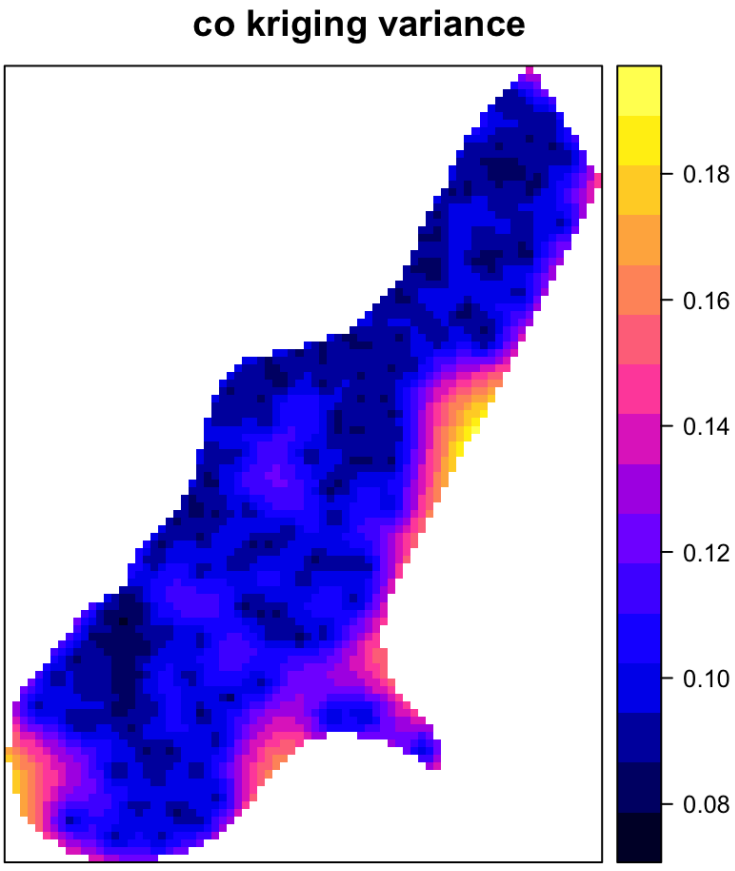
```
summary(cokrig.zincquart)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min    max
## x 178460 181540
## y 329620 333740
## Is projected: NA
## proj4string : [NA]
## Number of points: 3103
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## x            178460       40        78
## y            329620       40       104
## Data attributes:
##  zinc1stquart.pred zinc1stquart.var  zinc2ndquart.pred zinc2ndquart.var
##  Min.   :0.00000   Min.   :0.07852   Min.   :0.0000    Min.   :0.0978
##  1st Qu.:0.03981   1st Qu.:0.09278   1st Qu.:0.2856    1st Qu.:0.1166
##  Median :0.31988   Median :0.09936   Median :0.7084    Median :0.1257
##  Mean   :0.34905   Mean   :0.10584   Mean   :0.6101    Mean   :0.1344
##  3rd Qu.:0.59063   3rd Qu.:0.11312   3rd Qu.:0.9272    3rd Qu.:0.1442
##  Max.   :0.96129   Max.   :0.18936   Max.   :1.0000    Max.   :0.2516
##  zinc3rdquart.pred zinc3rdquart.var cov.zinc1stquart.zinc2ndquart
##  Min.   :0.2343    Min.   :0.1121   Min.   :0.005372
##  1st Qu.:0.6127    1st Qu.:0.1228   1st Qu.:0.017969
##  Median :0.9144    Median :0.1277   Median :0.024285
##  Mean   :0.8023    Mean   :0.1326   Mean   :0.030428
##  3rd Qu.:1.0000    3rd Qu.:0.1381   3rd Qu.:0.036929
##  Max.   :1.0000    Max.   :0.1958   Max.   :0.115197
##  cov.zinc1stquart.zinc3rdquart cov.zinc2ndquart.zinc3rdquart
##  Min.   :-0.009118             Min.   :0.01487
##  1st Qu.:-0.001283             1st Qu.:0.02875
##  Median : 0.002700             Median :0.03540
##  Mean   : 0.006664             Mean   :0.04184
##  3rd Qu.: 0.010691             3rd Qu.:0.04890
##  Max.   : 0.062379             Max.   :0.12679
```

```
# plot results
spplot(cokrig.zincquart["zinc1stquart.pred"], main = "co kriging predictions")
```




```
spplot(cokrig.zincquart["zinc1stquart.var"],  main = "co kriging variance")
```



```
# more fancy plotting
library(gridExtra)
```

```
## Loading required package: grid
```

```
grid.arrange(spplot(cokrig.zincquart["zinc1stquart.pred"], main = "co kriging predictions"),
             spplot(cokrig.zincquart["zinc2ndquart.pred"], main = "co kriging predictions"),
             ncol=2)
```