Introduction_to_spatial_statistics__#2

May 2, 2018

0.1 Lezione :: Analisi statistica spaziale di alcune proprietà dei suoli

0.1.1 Laurea magistrale in scienze forestali ed ambientali.Corso di Geografia e Valutazione del suolo

Giuliano Langella glangella@unina.it

Tobler's Low of Geography (1970): "Everything is related to everything else, but near things are more related than distant things".

0.1.2 LINKS:

https://cran.r-project.org/doc/contrib/intro-spatial-rl.pdf http://www.rspatial.org http://pakillo.github.io/R-GIS-tutorial/#iovec http://www.nickeubank.com/wp-content/uploads/2015/10/ http://neondataskills.org/R/extract-raster-data-R/#method-3-extract-values-using-a-shapefile

0.2 Step #2: Esercizio in R sulla geostatistica

Nota:Il kernel utilizzato per Jupyter è R per cui il codice è in tale linguaggio. Ai fini dello studio degli argomenti annotare i comandi necessari per eseguire le analisi (geo)statistiche proposte.
Argomenti trattati:

EXCEL

```
  <li>Illustrare la procedura in Excel, e consegnarla agli studenti (assegnare esercizio a casa
```

GSTAT: esempio built-in

```
  <li>Prendiamo confidenza  mediante l'uso di GSTAT  con (i) l'analisi variografica (variogramma
  <li>https://cran.r-project.org/web/packages/gstat/vignettes/gstat.pdf</li>
```

GSTAT: esempio applicativo in Valle Telesina

```
<ul>
  <li>Entriamo in R, installiamo il pacchetto gstat e creiamo lo schema di campionamento</li>
  <li>Entriamo in GIS (SAGA, QGIS, ...), importiamo il DEM, importiamo le coordinate dei punti d
  <li>Equipariamo mentalmente il processo di "estrazione" dei dati di quota dal DEM come all'int
```

```
  <li>Aprire RStudio</li>
  <ul>
    <li>Importare i dati di quota estratti (ovvero di C.O. misurati) nei punti di campionamento<
    <li>Costruire un variogramma sperimentale</li>
    <li>Eseguire il fitting mediante un variogramma teorico autorizzato (sferico ed esponenziale
    <li>Utilizzare il modello di variogramma per l'interpolazione mediante kriging</li>
    <li>Confrontare le mappe di kriging utilizzando i diversi modelli di variogramma</li>
    <li>Confrontare una mappa di kriging con il DEM originale</li>
    <li>(nel caso del campionamento pedologico, è come se potessimo confrontare le nostre interp
  </ul>
</ul>
```

0.3 2. GSTAT: esempio built-in

```
In [26]: library(repr)   # to size plots within Jupyter
         library(raster) # to use the raster function
```

Dati meuse di tipo vettoriale (come se fosse uno shapefile):

```
In [2]: library(sp)  # carica il pacchetto R "sp"
        data(meuse)  # carica i dati del caso studio "meuse"
        ls()         # lista delle variabili (Environment)
        class(meuse) # stampa il tipo di dato di "meuse"
        names(meuse) # stampa l'intestazione della tabella (column header)
```

'meuse'
'data.frame'
1. 'x' 2. 'y' 3. 'cadmium' 4. 'copper' 5. 'lead' 6. 'zinc' 7. 'elev' 8. 'dist' 9. 'om' 10. 'ffreq' 11. 'soil' 12. 'lime' 13. 'landuse' 14. 'dist.m'

La funzione "coordinates", quando utilizzata (es. = ~x+y), promuove il "data.frame" meuse in un tipo "SpatialPointsDataFrame" (SPDF), che ha consapevolezza circa le proprie coordinate spaziali:

```
In [3]: coordinates(meuse) = ~x+y # assegna a meuse le coordinate
        class(meuse)              # stampa il tipo di dato di "meuse"
```

'SpatialPointsDataFrame'

```
In [4]: summary(meuse) #ăstampa le stats di meuse
```

```
Object of class SpatialPointsDataFrame
Coordinates:
     min    max
x 178605 181390
y 329714 333611
Is projected: NA
```

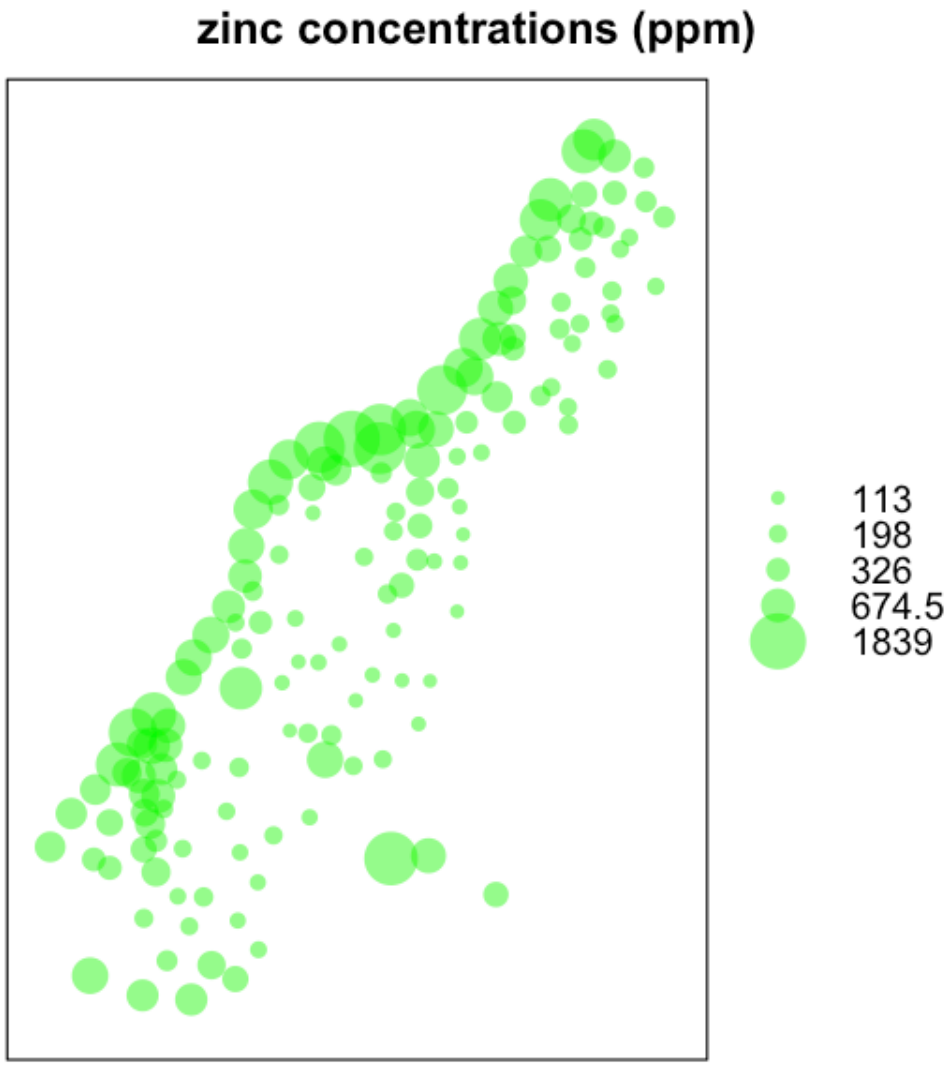
```
proj4string : [NA]
Number of points: 155
Data attributes:
    cadmium           copper            lead            zinc
 Min.   : 0.200   Min.   : 14.00   Min.   : 37.0   Min.   : 113.0
 1st Qu.: 0.800   1st Qu.: 23.00   1st Qu.: 72.5   1st Qu.: 198.0
 Median : 2.100   Median : 31.00   Median :123.0   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
```

In [5]:
```
# stampa le coordinate dei primi 5 punti geospaziali
coordinates(meuse)[1:5,]
```

	x	y
1	181072	333611
2	181025	333558
3	181165	333537
4	181298	333484
5	181307	333330

In [6]:
```
# plot geospaziale dei punti di campionamento
options(repr.plot.width=5,repr.plot.height=5)
bubble(meuse, "zinc", col=c("#00ff0088", "#00ff0088"),
       main = "zinc concentrations (ppm)")
```



Dati meuse di tipo grid o raster:

```
In [7]: data(meuse.grid)     # carica il dato "meuse.grid"
        summary(meuse.grid) # stampa le stats di "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
```

```
In [8]: class(meuse.grid) # stampa il tipo di dato di "meuse.grid"
```

'data.frame'

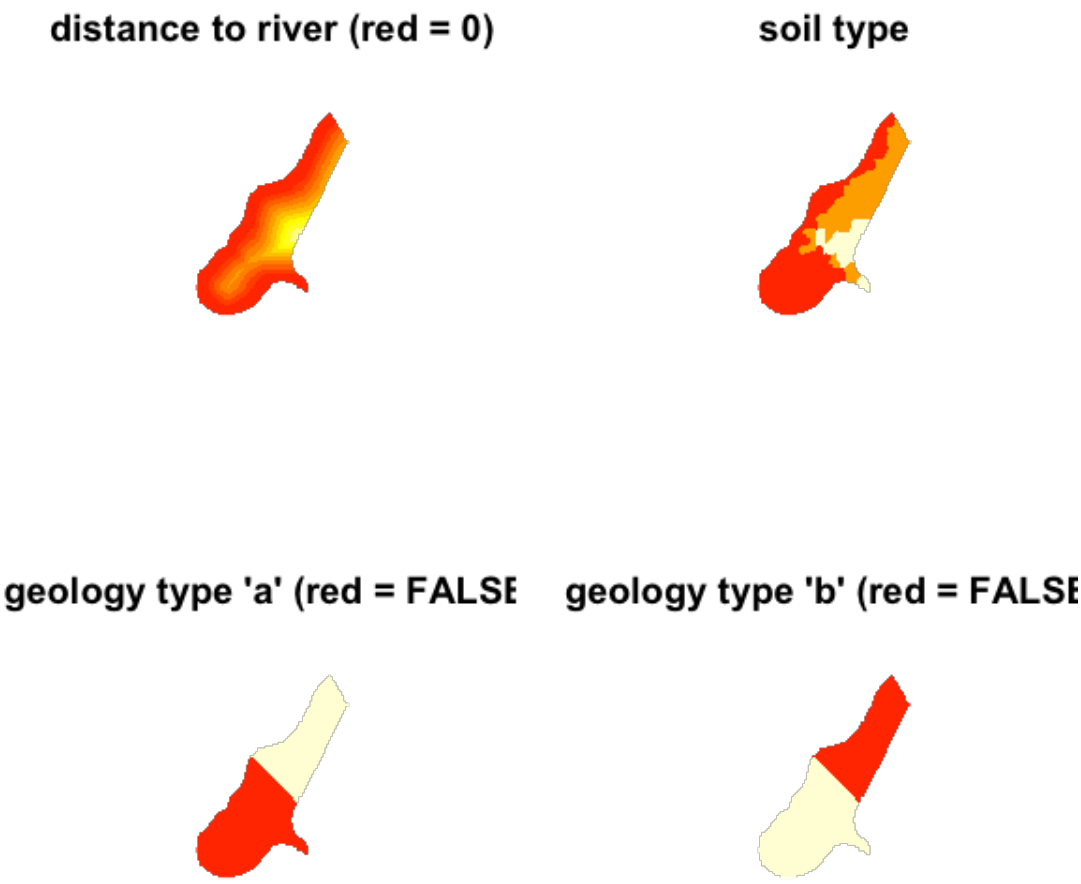
```
In [9]: coordinates(meuse.grid) = ~x+y # assegna a meuse.grid le coordinate
        class(meuse.grid)             # stampa il tipo di dato di "meuse.grid"
```

'SpatialPointsDataFrame'

```
In [10]: gridded(meuse.grid) = TRUE     # assegna a meuse.grid il tipo "grid"
         class(meuse.grid)              # stampa il tipo di dato di "meuse.grid"
```

'SpatialPixelsDataFrame'

```
In [11]: par( mfrow = c( 2, 2 ) ) # divide lo spazio dei grafici in 2 righe e 2 colonne
         image(meuse.grid["dist"]);title("distance to river (red = 0)")       # row 1 col 1
         image(meuse.grid["soil"]);title("soil type")                         # row 1 col 2
         image(meuse.grid["part.a"]);title("geology type 'a' (red = FALSE)")  # row 2 col 1
         image(meuse.grid["part.b"]);title("geology type 'b' (red = FALSE)")  # row 2 col 2
```



0.3.1 Inverso della distanza

```
In [12]: library(gstat) # carica il pacchetto GSTAT
```

$$\mathbf{1} \quad z_u(p) = \frac{\sum_{i=1}^{n}\left(\frac{z_i}{d_i^p}\right)}{\sum_{i=1}^{n}\left(\frac{1}{d_i^p}\right)}$$

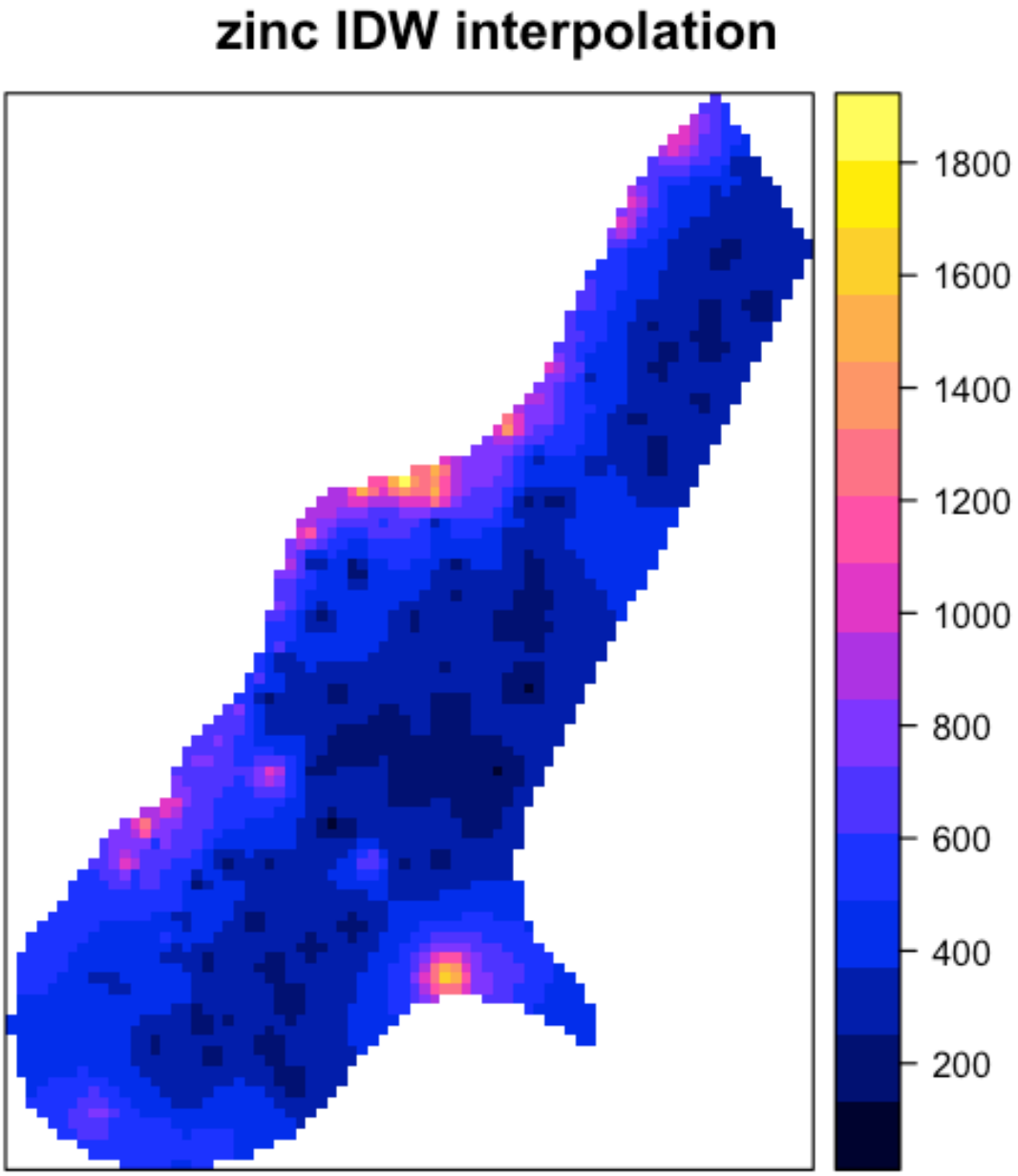
```
In [13]: # interpolazione inverso della distanza
         zinc.idw = idw(zinc~1,      # come interpolare Zn? (~1 : senza info ausiliarie)
                        meuse,       # dov'è Zn? (in meuse)
```

```
                    meuse.grid) # dove interpolare? (nei pixel meuse.grid)
        class(zinc.idw) # stampa il tipo di dato restituito da GSTAT
```

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

'SpatialPixelsDataFrame'

```
In [14]: options(repr.plot.width=4)
         spplot(zinc.idw["var1.pred"],          # mappa di Zn (pred)
                main = "zinc IDW interpolation")# titolo della mappa
```

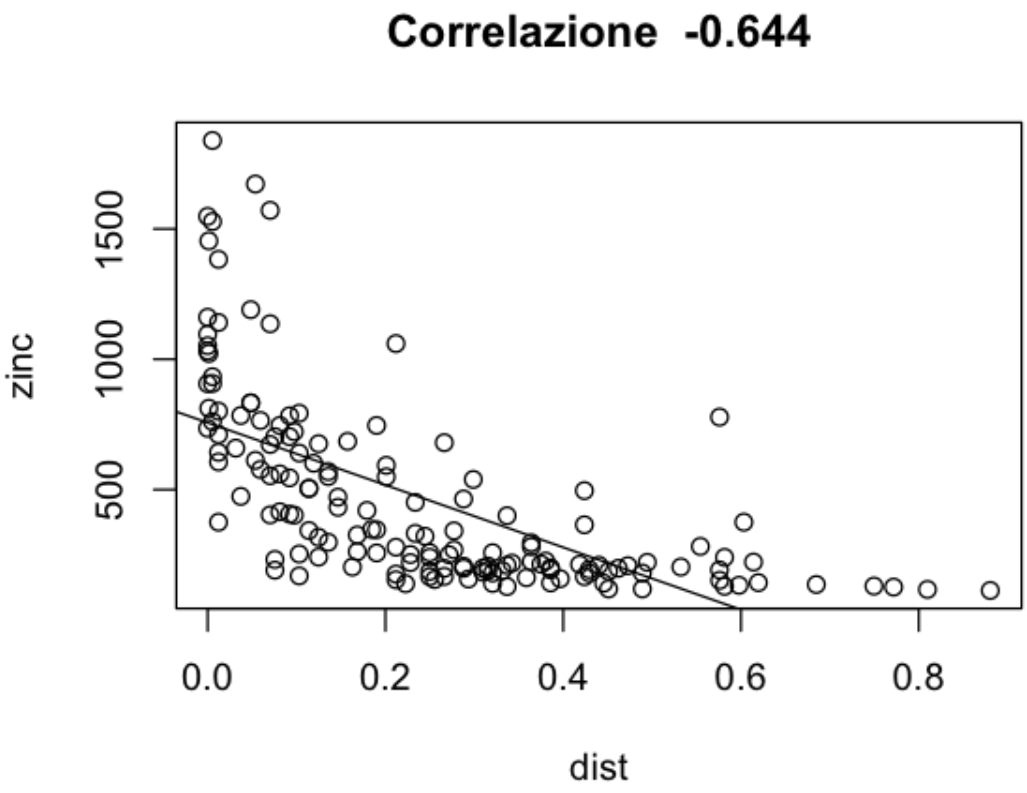


1.0.1 Variografia (ossia analisi geostatistica dei dati per mezzo del variogramma)

Comparando il grafico a bolle delle misure di zinco con la mappa della distanza dal fiume, diventa evidente che le concentrazioni maggiori del metallo sono misurate in prossimità del fiume.

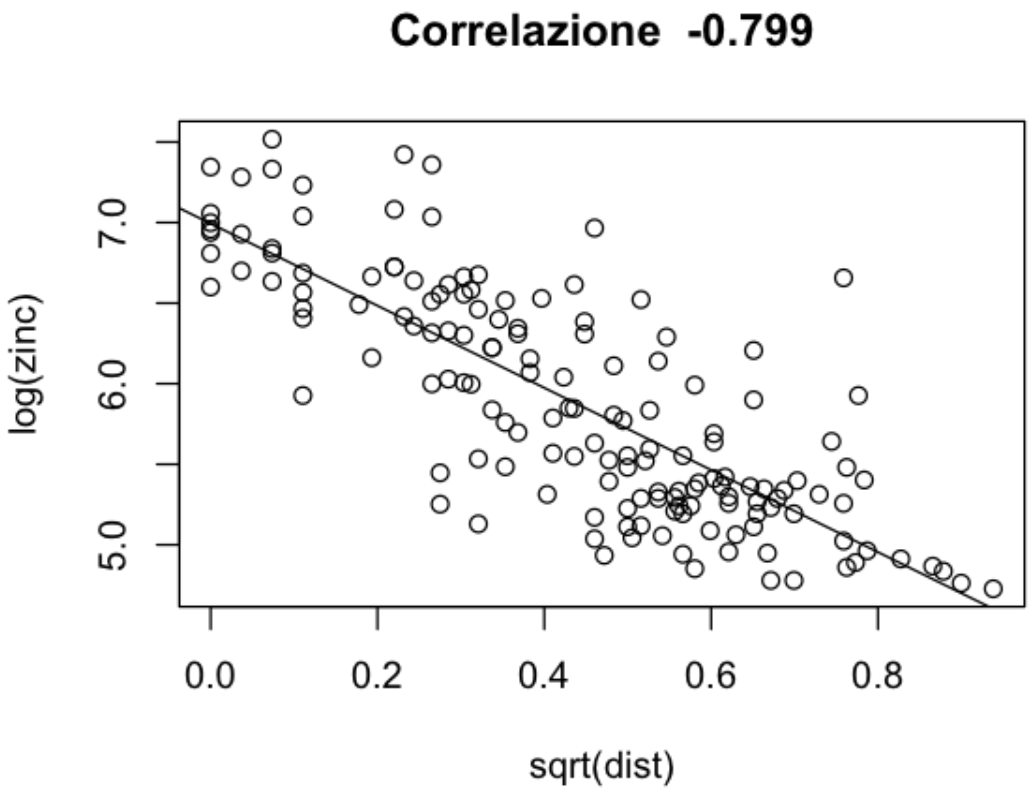
Seguono la correlazione e lo scatterplot con linea di tendenza sui dati grezzi:

```
In [15]: options(repr.plot.width=5,repr.plot.height=4)
         cor_raw = cor(cbind(meuse$zinc,meuse$dist))
         plot(zinc~dist, meuse)            # scatter dei punti Zn vs DIST
         abline(lm(zinc ~ dist, meuse))    # linea di tendenza
         title( paste("Correlazione ",round(cor_raw[2],3)) )
```



Il grafico sopra evidenzia un andamento non lineare. Questa relazione può essere linearizzata mediante una trasformazione logaritmica delle concentrazioni di zinco, e prendendo la radice quadrata della distanza dal fiume:

```
In [16]: # coefficiente di correlazione tra i valori di zinco e la distanza dal fiume:
         cor_transf = cor(cbind(log(meuse$zinc),sqrt(meuse$dist)))
         plot(log(zinc)~sqrt(dist), meuse)           # scatter dei punti Zn vs DIST
         abline(lm(log(zinc)~sqrt(dist), meuse))     # linea di tendenza
         title( paste("Correlazione ",round(cor_transf[2],3)) )
```



Analisi della presenza di "struttura spaziale" nei dati. Ossia costruzione del variogramma sperimentale ed osservazione (grafica o numerica) dell'andamento della varianza (*gamma*, γ) all'aumentare della distanza nelle coppie di punti.

```
In [17]: lzn.vgm = variogram(log(zinc)~1, meuse) # variogramma sperimentale, log()
         lzn.vgm # stampa i valori di distanza / varianza medi delle coppie di punti
```

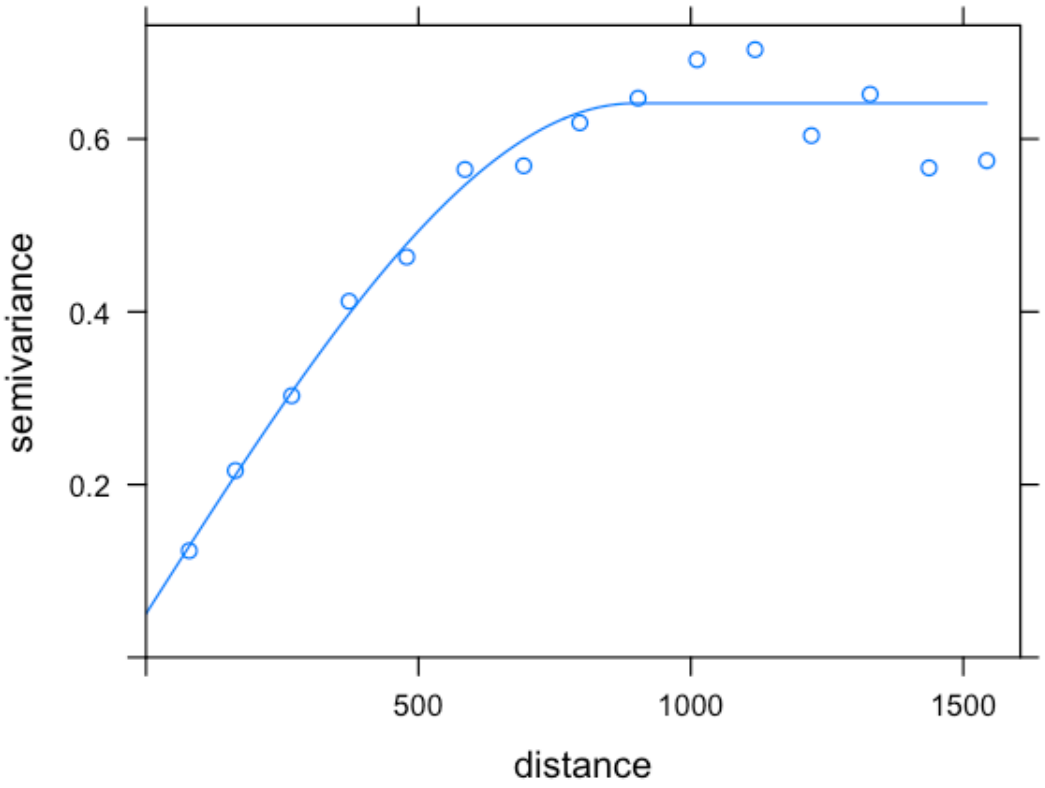
np	dist	gamma	dir.hor	dir.ver	id
57	79.29244	0.1234479	0	0	var1
299	163.97367	0.2162185	0	0	var1
419	267.36483	0.3027859	0	0	var1
457	372.73542	0.4121448	0	0	var1
547	478.47670	0.4634128	0	0	var1
533	585.34058	0.5646933	0	0	var1
574	693.14526	0.5689683	0	0	var1
564	796.18365	0.6186769	0	0	var1
589	903.14650	0.6471479	0	0	var1
543	1011.29177	0.6915705	0	0	var1
500	1117.86235	0.7033984	0	0	var1
477	1221.32810	0.6038770	0	0	var1
452	1329.16407	0.6517158	0	0	var1
457	1437.25620	0.5665318	0	0	var1
415	1543.20248	0.5748227	0	0	var1

Si esegue il fitting del variogramma sperimentale (lzn.vgm) con un modello teorico ammissibile. Abbiamo visto nella precedente lezione diversi modelli ('Sph', 'Exp', ...). I parametri {sill,nugget,range} del variogramma sperimentale devono essere passati alla funzione fit.variogram() che crea il variogramma di fitting. In R+GSTAT, il passaggio dei parametri è mediato dalla funzione vgm(sill,model,range,nugget). In definitiva, la funzione di fitting del variogramma è così fatta: - fit.variogram(exp_vgm, mod_vgm, ...) - ... indica altri parametri, come appresso indicato - fit.sills [DEFAULT = TRUE] - fit.ranges [DEFAULT = TRUE] - fit.method [DEFAULT = 7]

```
In [18]: lzn.fit = fit.variogram(lzn.vgm,                    # variogramma sperimentale di cui
                                 model = vgm(1, "Sph", 900, 1))# modello di variogramma usato pe
         lzn.fit # stampa i parametri del modello
```

model	psill	range	kappa	ang1	ang2	ang3	anis1	anis2
Nug	0.05066243	0.0000	0.0	0	0	0	1	1
Sph	0.59060780	897.0209	0.5	0	0	0	1	1

```
In [19]: options(repr.plot.width=5,repr.plot.height=4)
         plot(lzn.vgm, lzn.fit)
```

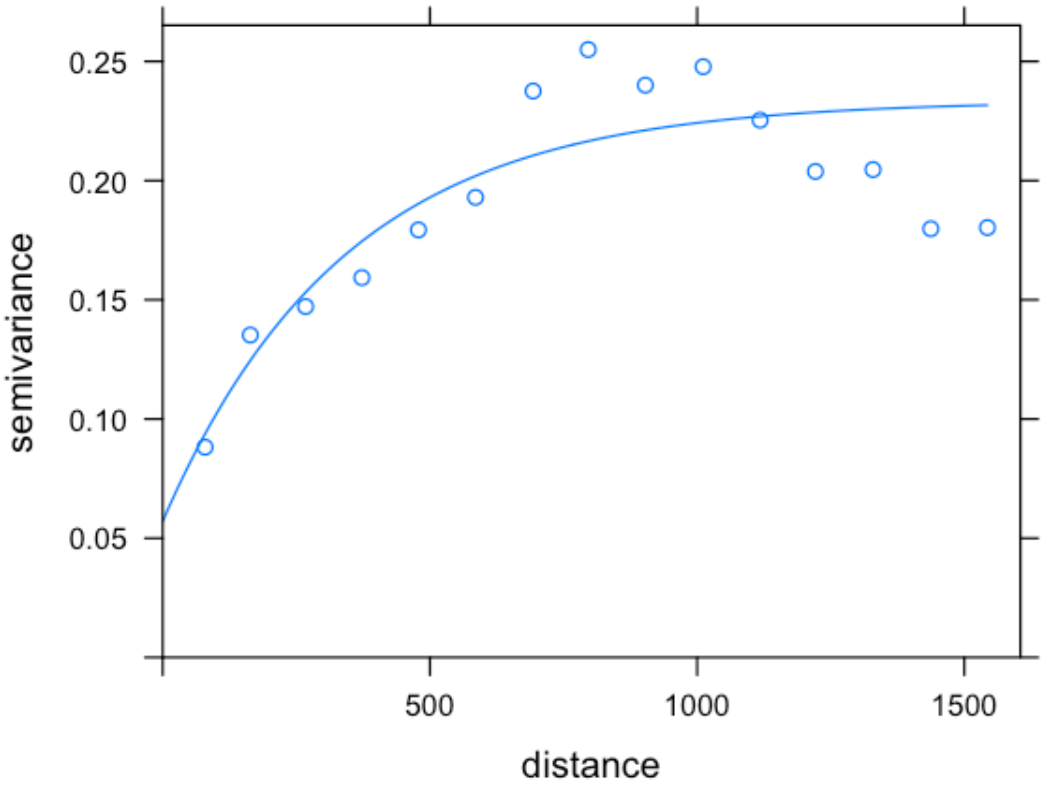


Invece della media costante - denotata da ~1, possiamo specificare una funzione media, ad esempio utilizzando ~sqrt(dist) come predittore:

```
In [20]: lznr.vgm = variogram(log(zinc)~sqrt(dist), meuse)
         lznr.fit = fit.variogram(lznr.vgm, model = vgm(1, "Exp", 300, 1))
         lznr.fit
```

model	psill	range	kappa	ang1	ang2	ang3	anis1	anis2
Nug	0.05712231	0.0000	0.0	0	0	0	1	1
Exp	0.17641559	340.3201	0.5	0	0	0	1	1

```
In [21]: plot(lznr.vgm, lznr.fit)
```



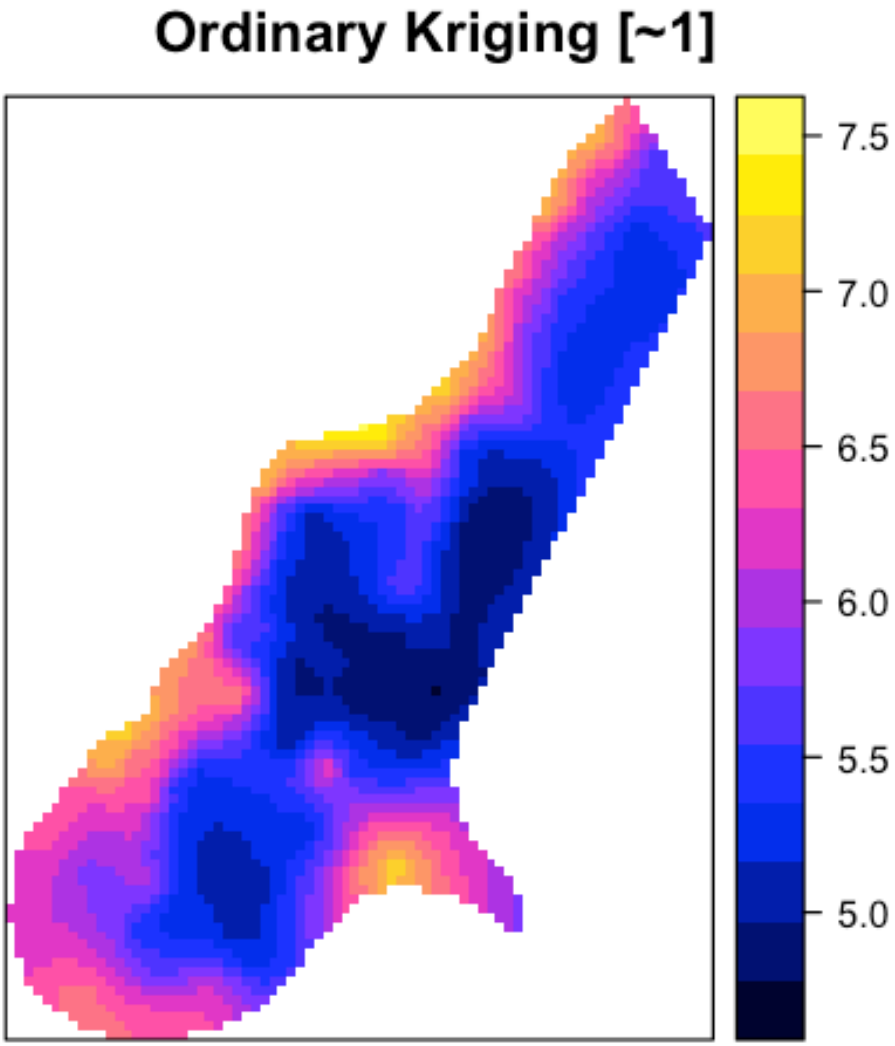
1.0.2 Interpolazione spaziale mediante kriging:

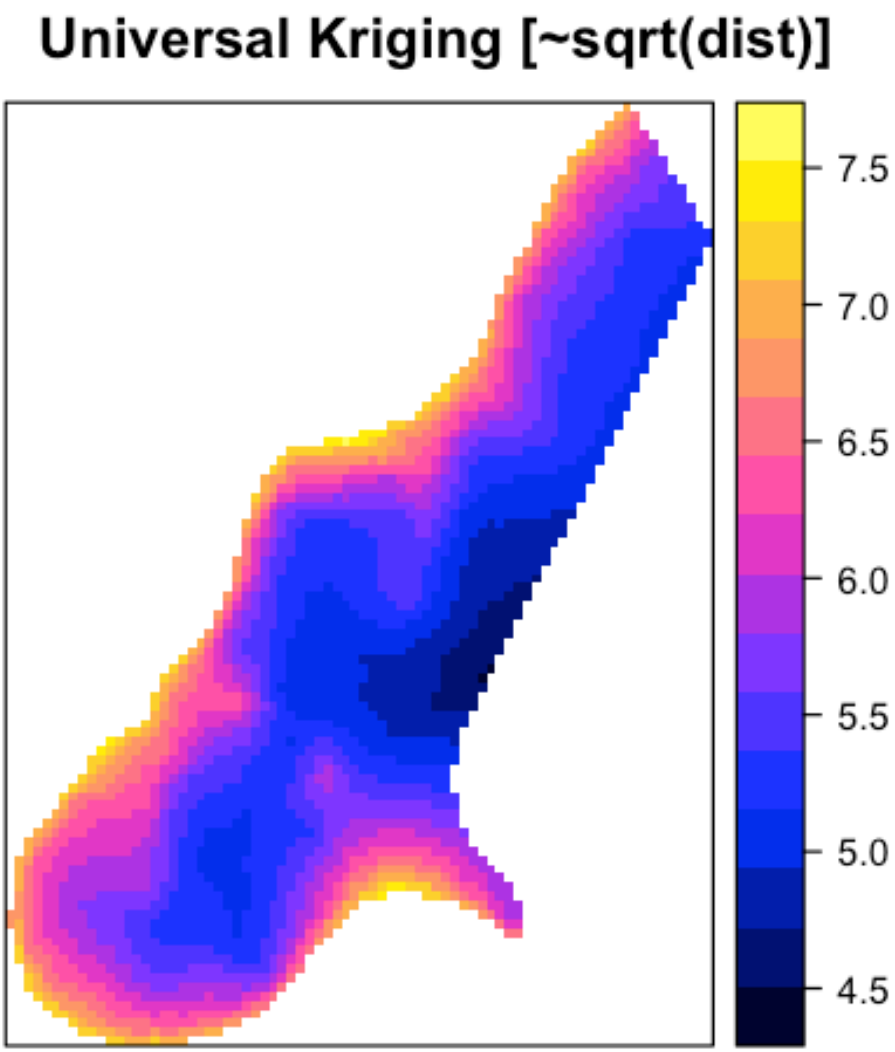
- ordinary kriging [~1]
- universal kriging [~sqrt(dist)]

```
In [22]: lzn.kriged = krige(log(zinc)~1, meuse, meuse.grid, model = lzn.fit)
         lznr.kriged = krige(log(zinc)~sqrt(dist), meuse, meuse.grid, model = lznr.fit)
```

```
[using ordinary kriging]
[using universal kriging]
```

```
In [23]: options(repr.plot.width=4)
         #par( mfrow = c( 2, 2 ) ) # divide lo spazio dei grafici in 2 righe e 2 colonne
         spplot(lzn.kriged["var1.pred"],main="Ordinary Kriging [~1]")
         spplot(lznr.kriged["var1.pred"],main="Universal Kriging [~sqrt(dist)]")
```


Ordinary Kriging [~1]
7.5
7.0
6.5
6.0
5.5
5.0



1.1 3. GSTAT: esempio applicativo in Valle Telesina

- importare un raster (DEM) in R
- creare un griglia regolare di punti di campionamento, metodo artigianale
- creazione di un raster, metodo rapido
- creare la lista di punti da esportare per uso in GIS (saltiamo)
- estrazione della ELEV nei punti di campionamento (in R, con/senza buffer)
- variogramma e fitting della ELEV
- interpolazione spaziale (kriging) della ELEV
- valutazione critica dei risultati dell'analisi

Caricare i pacchetti R richiesti dalla procedura seguente:

```
In [24]: #library(repr)   # to size plots within Jupyter
         #    options(repr.plot.width=4)
         require(gstat)   # to run geostatistical analysis
```

Dai metadati del DEM in ambiente GIS ricaviamo le coordinate del Bounding Box Il DEM è della Valle Telesina. Nel corso degli esperimenti statistici affronteremo lo studio di un dataset pedologico ottenuto in Valle Telesina. Per questa ragione eseguiremo le nostre analisi in questo areale.

```
In [27]: DEM <- raster("data/dem5m.tif")
         #plot(DEM)
```

```
In [28]: DEM
```

```
class       : RasterLayer
dimensions  : 3264, 4800, 15667200  (nrow, ncol, ncell)
resolution  : 5, 5  (x, y)
extent      : 453000, 477000, 4556000, 4572320  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=utm +zone=33 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
data source : /Users/giuliano/git/Didattica/jupyter/data/dem5m.tif
names       : dem5m
```

```
In [29]: extent(DEM)
```

```
class       : Extent
xmin        : 453000
xmax        : 477000
ymin        : 4556000
ymax        : 4572320
```

```
In [30]: bbox(DEM)
```

	min	max
s1	453000	477000
s2	4556000	4572320

Incolliamo le coordinate del Bounding Box della nostra area di studio, avendo cura di riconoscere opportunamente le dimensioni X (Easting) ed Y (Northing). Possiamo ottenere le info del bbox sia in R con extent(*GRID*) sia in GIS.

```
In [31]: xmin = 453000  #ămetri
         ymin = 4556000 #ămetri
         xmax = 477000  #ămetri
         ymax = 4572320 #ămetri
```

Calcoliamo l'estensione dell'area studio lungo X ed Y:

```
In [32]: xdelta = xmax-xmin
         ydelta = ymax-ymin
         # stampa i valori in metri:
         print( cbind(xdelta,ydelta) )

     xdelta ydelta
[1,]  24000  16320
```

Creare una griglia regolare di campionamento Si divide il dominio XY (=2D, ossia in termini matematici $\mathfrak{D}^2 \subset \Re^2$) dell'area studio in una griglia regolare di $N_x \times N_y$ punti di campionamento. Significa dividere per $N_x - 1$ l'estensione lungo X e per $N_y - 1$ l'estensione lungo Y.

```
In [33]: Nlen = 20
         Nlen = 20
         dx   = (xmax-xmin) / Nlen
         dy   = (ymax-ymin) / Nlen
```

Scriviamo le coordinate dei punti lungo i due transetti X ed Y. La funzione seq crea una sequenza regolare di valori, partendo da xmin fino ad xmax con uno step pari a dx. (abbiamo che il vettore xtransect è costituito da 101 elementi, come ytransect)

```
In [34]: xtransect = seq(xmin,xmax,dx)
         ytransect = seq(ymin,ymax,dy)
         # Numero di punti lungo X e lungo Y:
         Nx = length(xtransect)
         Ny = length(ytransect)
         # stampa il numero di elementi del vettore xtransect (ytransect ha le stesse dimensioni
         print( cbind(Nx,Ny) )

     Nx Ny
[1,] 21 21
```

Replichiamo il vettore transect a formare una matrice per ogni dimensione del dominio spaziale (quindi X ed Y):

```
In [35]: Xgrid = rep(xtransect,Ny) #ăXgrid è ancora un vettore, ma di Nx*Ny elementi
         Ygrid = rep(ytransect,Nx) #ăYgrid è ancora un vettore, ma di Nx*Ny elementi
         # Xgrid (e Ygrid) è un vettore con numero di elementi pari a Nx*Ny:
         class(Xgrid)  # ad un oggetto R "numeric" possiamo chiedere "length" ma non "dim"
         length(Xgrid) # infatti "length" restituisce il numero di elementi (Nx*Ny)
         dim(Xgrid)    # "dim" non funziona in quanto il tipo dati Xgrid è "numeric" e non "matr
```

'numeric'
441

```
NULL
```

Manipolazione dei vettori Xgrid ed Ygrid per formare le matrici 2D di N_x righe e N_y colonne:

```
In [36]: Xgrid = matrix( Xgrid, Nx, Ny )
         Xgrid = t(Xgrid)
         Ygrid = matrix( Ygrid, Nx, Ny )

         # Osserviamo le differenza rispetto alle stampe effettuate in precedenza su Xgrid:
         class(Xgrid)  # ad un oggetto R "matrix" possiamo chiedere sia "length" che "dim"
         length(Xgrid) # infatti "length" restituisce il numero di elementi (Nx*Ny)
         dim(Xgrid)    # "dim" restituisce il numero di elementi distinti per dimensione X ed Y
```

'matrix'
441
1. 21 2. 21

Il punto P di coordinate matriciali [1,1] è definito come P_1_1 ed ha le seguenti coordinate x(1,1) ed y(1,1):

```
In [37]: pi = paste("P_",1,"_",1,sep="")
         xi = Xgrid[1,1]
         yi = Ygrid[1,1]
         print( cbind(pi,xi,yi) )
```

```
     pi       xi       yi
[1,] "P_1_1" "453000" "4556000"
```

Il punto di campionamento P_1_2 è ubicato uno step dx a destra lungo l'asse X ed è fermo lungo l'asse Y:

```
In [38]: pi = paste("P_",1,"_",2,sep="")
         xi = Xgrid[1,2]
         yi = Ygrid[1,2]
         print( cbind(pi,xi,yi,dx,dy) )
```

```
     pi       xi       yi        dx     dy
[1,] "P_1_2" "454200" "4556000" "1200" "816"
```

per cui ha la stessa coordinata Y di P_1_1 ma la coordinata X è quella di P_1_1 + dx

Il punto di campionamento P_2_1 è ubicato uno step sotto lungo l'asse Y ed è fermo lungo l'asse X:

```
In [39]: pi = paste("P_",2,"_",1,sep="")
         xi = Xgrid[2,1]
         yi = Ygrid[2,1]
         print( cbind(pi,xi,yi,dx,dy) )
```

```
     pi       xi       yi        dx     dy
[1,] "P_2_1" "453000" "4556816" "1200" "816"
```

per cui ha la stessa coordinata X di P_1_1 ma la coordinata Y è quella di P_1_1 + dy

Metodo rapido per la creazione di un raster Leggere il CRS con codice EPSG:32632 in formato PROJ4 dal portale spatialreference.org: http://spatialreference.org/ref/epsg/32633/proj4/

In alternativa è possibile chiedere il CRS al DEM precedentemente importato in R:

```
In [40]: proj4string(DEM)
```

'+proj=utm +zone=33 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0'

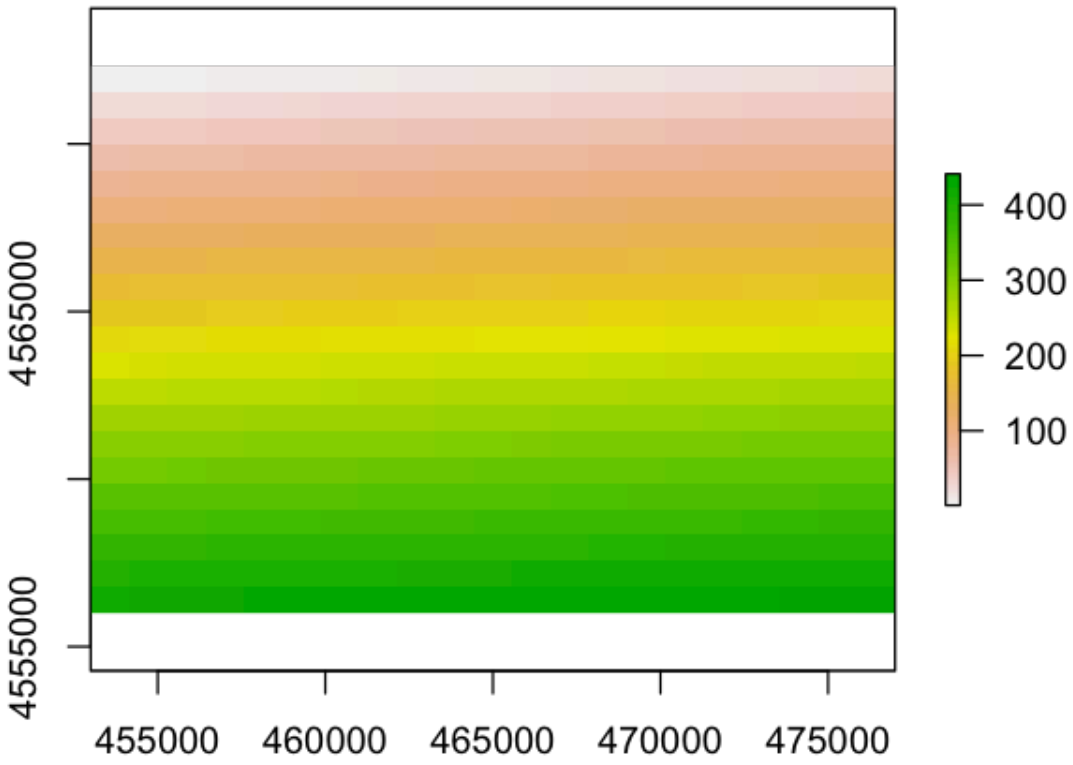
```
In [41]: GRID <- raster(ncol = Nx, nrow = Ny, xmn = xmin, xmx = xmax, ymn = ymin, ymx = ymax,
                        crs="+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs")
         GRID
```

```
class       : RasterLayer
dimensions  : 21, 21, 441  (nrow, ncol, ncell)
resolution  : 1142.857, 777.1429  (x, y)
extent      : 453000, 477000, 4556000, 4572320  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs +towgs84=0,0,0
```

```
In [42]: hasValues(GRID) # esiste il grigliato ma non è valorizzato!
```

FALSE

```
In [43]: # Note Nx*Ny is the total number of cells in the grid
         values(GRID) <- 1:(Nx * Ny)
         options(repr.plot.width=5,repr.plot.height=4)
         plot(GRID)
```



Creazione della lista di punti di coordinate X ed Y in corrispondenza dei quali eseguire il campionamento Nel nostro caso specifico, l'estrazione dei valori di quota dal DEM in ambiente GIS mediante l'uso di Points.txt

```
In [44]: # Attenzione: inserendo la [] dopo la funzione forza la richiesta degli elementi indica
         Nrows = dim(Xgrid)[1] # si chiede il primo elemento di output della funzione dim
         Ncols = dim(Xgrid)[2] # si chiede il secondo elemento di output della funzione dim
         i=0
         Pi <- vector(mode = "character", length = Nrows*Ncols)
         Xi <- vector(mode = "numeric",   length = Nrows*Ncols)
         Yi <- vector(mode = "numeric",   length = Nrows*Ncols)
         # questo è un ciclo for, serve a ripetere una stessa operazione N-volte scrivendola 1-v
         for (row in 1:Nrows){
             for (col in 1:Ncols){
                 i=i+1
                 # Stampa dell'etichetta del punto di campionamento, per le successive elaborazi
                 # Il risultato è del tipo P_riga_colonna
                 # Ad es. P_17_100 indica il punto di campionamento alla riga 17 e colonna 100
                 # Costruzione del campo etichetta dei punti Pi di campionamento, es. P1, P2, ec
```

```
            Pi[i] = paste("P_",row,"_",col, sep="")
            # Stampa del valore di coordinata X ed Y
            Xi[i] = Xgrid[row,col]
            Yi[i] = Ygrid[row,col]
            #print(Pi[i])
            #print(Xi[i])
            #print(Yi[i])
        }
    }
```

```
In [45]: Points = cbind(Pi,Xi,Yi)
         #creare la matrice XY e salvare su disco per poi importarla in GIS (QGIS/SAGA)
         write.table(Points, file = "data/Points.txt", append = FALSE, quote = FALSE, sep = ",",
                     eol = "\n", na = "NA", dec = ".", row.names = FALSE,
                     col.names = TRUE, qmethod = c("escape", "double"),
                     fileEncoding = "")
```

```
In [46]: class(Points)
```

'matrix'

1.2 Estrarre i dati di quota dal DEM (plugin :: Point Sampling Tool)

1.2.1 In QGIS:

Importare in GIS la tabella Points.txt appena creata e campionare il DEM (i suoli) nei punti della griglia di campionamento. - http://www.qgistutorials.com/it/docs/sampling_raster_data.html - https://pvanb.wordpress.com/2010/02/15/sampling-raster-values-at-point-locations-in-qgis/

1.2.2 In R:

- http://neondataskills.org/R/extract-raster-data-R/#method-3-extract-values-using-a-shapefile

Si crea un oggetto R del tipo Spatial Points, assegnando opportuno CRS:

```
In [47]: XY = cbind(Xi,Yi) # concatena Xi ed Yi a formare 2 colonne, a differenza di c()
         dimnames(XY)[[1]] = Pi # aggiunge un row-header a XY
         head(XY) #ăstampa le prime righe di XY per analizzare la tabella creata
```

	Xi	Yi
P_1_1	453000	4556000
P_1_2	454200	4556000
P_1_3	455400	4556000
P_1_4	456600	4556000
P_1_5	457800	4556000
P_1_6	459000	4556000

```
In [48]: XY_sp = SpatialPoints(XY)
         summary(XY_sp)
```

```
Object of class SpatialPoints
Coordinates:
       min     max
Xi  453000  477000
Yi 4556000 4572320
Is projected: NA
proj4string : [NA]
Number of points: 441
```

```
In [49]: # CRS : Coordinate Reference System
         proj4string(XY_sp) <- '+proj=utm +zone=33 +datum=WGS84
                                +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0'
         summary(XY_sp)
```

```
Object of class SpatialPoints
Coordinates:
       min     max
Xi  453000  477000
Yi 4556000 4572320
Is projected: TRUE
proj4string :
[+proj=utm +zone=33 +datum=WGS84 +units=m +no_defs +ellps=WGS84
+towgs84=0,0,0]
Number of points: 441
```

Estrazione del valore della variabile in corrispondenza della griglia di campionamento:

```
In [50]: ELEV <- extract(DEM,      # raster layer
                         XY_sp,    # SPDF with centroids
                         'simple')#ămethod = {'simple' or 'bilinear'}

         ELEV_buf <- extract(DEM, # raster layer
                 XY_sp,               # SPDF with centroids for buffer
                 buffer = 400,        # buffer size, units depend on CRS, in our case meters
                 fun=max,             # what value to extract (min, mean, max, myfun(),...)
                 df=FALSE)            # return a dataframe?
```

```
In [51]: summary(ELEV)
         summary(ELEV_buf)
```

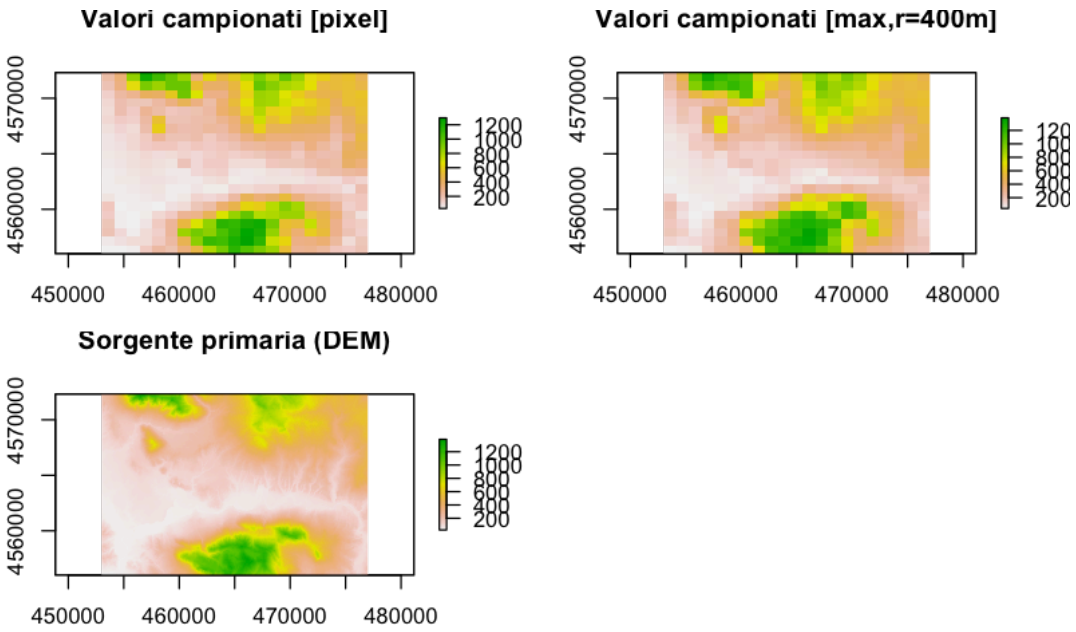
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   31.2   114.5   233.7   348.2   508.4  1297.5
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   40.3   156.4   315.2   426.2   620.0  1386.8
```

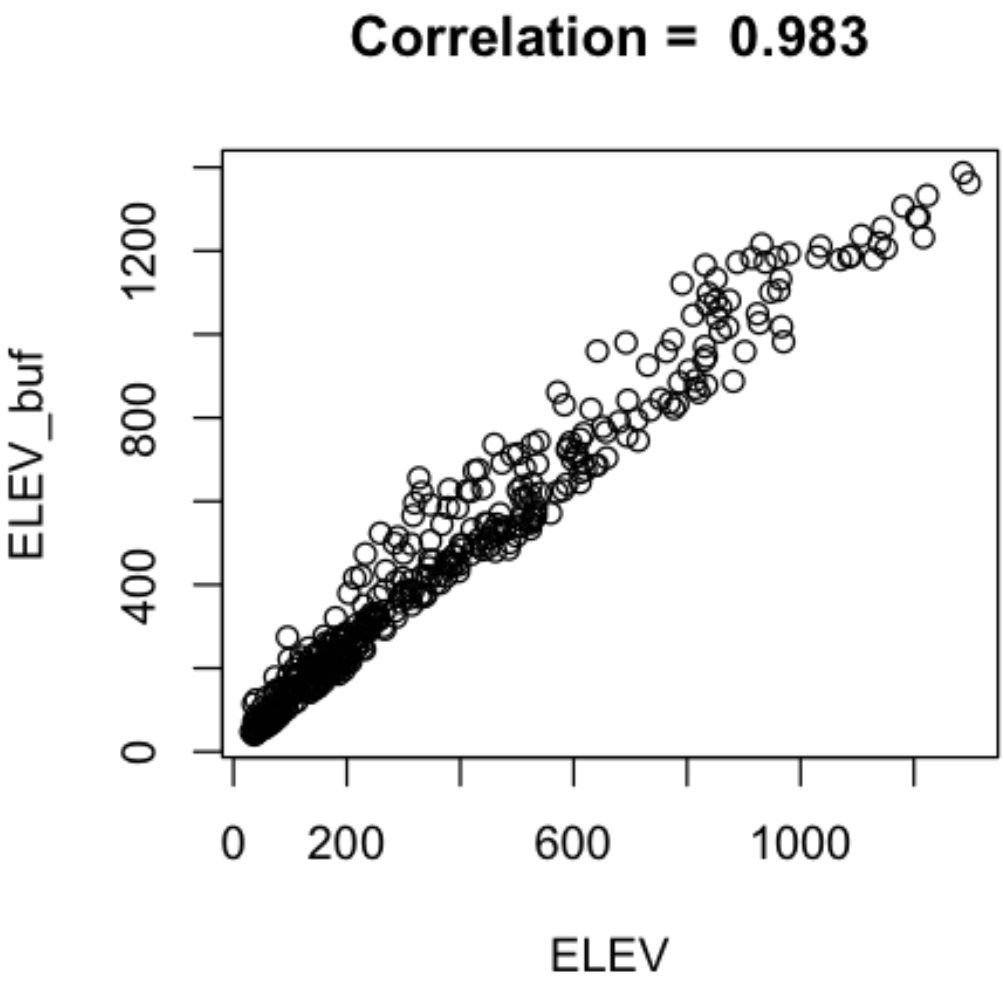
Confrontare i valori campionati con la sorgente primaria (DEM):

In [52]:
```
# crea matrice da vettore
mELEV     = matrix(ELEV,Nx,Ny)
mELEV_buf = matrix(ELEV_buf,Nx,Ny)
# traspone matrice
mELEV     = t(mELEV)
mELEV_buf = t(mELEV_buf)
# ribalta la matrice up/down
mELEV     = apply(mELEV, 2, rev)
mELEV_buf = apply(mELEV_buf, 2, rev)
# scrivi i valori di quota sul grigliato mappa
GRID_buf = GRID
values(GRID) <- mELEV
values(GRID_buf) <- mELEV_buf
```

In [53]:
```
options(repr.plot.width=7,repr.plot.height=4)
par( mfrow = c( 2, 2 ) )
plot(GRID,     main="Valori campionati [pixel]")
plot(GRID_buf,main="Valori campionati [max,r=400m]")
plot(DEM,      main="Sorgente primaria (DEM)")
```



In [54]:
```
options(repr.plot.width=4,repr.plot.height=4)
ceeb = cor(cbind(ELEV,ELEV_buf))
plot(ELEV,ELEV_buf,main=paste("Correlation = ",round(ceeb[2],3)) )
```



1.2.3 Costruire il variogramma sperimentale (su ELEV)

Abbiamo le coordindate in XY ed i valori della variabile in ELEV.

```
In [55]: print("XY")
         summary(XY)
         print("ELEV")
         summary(ELEV)
```

```
[1] "XY"
```

```
       Xi                Yi
```

```
 Min.   :453000   Min.   :4556000
 1st Qu.:459000   1st Qu.:4560080
 Median :465000   Median :4564160
 Mean   :465000   Mean   :4564160
 3rd Qu.:471000   3rd Qu.:4568240
 Max.   :477000   Max.   :4572320
```

```
[1] "ELEV"
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   31.2   114.5   233.7   348.2   508.4  1297.5
```

```
In [56]: Points = data.frame(XY,ELEV)
         class(Points)    # stampa il tipo di dato di Points
         head(Points)     # stampa le prime N righe di Points
         summary(Points) #ăstampa le statistiche di Points
```

'data.frame'

	Xi	Yi	ELEV
P_1_1	453000	4556000	31.2
P_1_2	454200	4556000	37.7
P_1_3	455400	4556000	48.3
P_1_4	456600	4556000	149.7
P_1_5	457800	4556000	231.2
P_1_6	459000	4556000	191.3

```
       Xi               Yi              ELEV
 Min.   :453000   Min.   :4556000   Min.   :  31.2
 1st Qu.:459000   1st Qu.:4560080   1st Qu.: 114.5
 Median :465000   Median :4564160   Median : 233.7
 Mean   :465000   Mean   :4564160   Mean   : 348.2
 3rd Qu.:471000   3rd Qu.:4568240   3rd Qu.: 508.4
 Max.   :477000   Max.   :4572320   Max.   :1297.5
```

La funzione "coordinates", quando utilizzata (es. sul lato sinistro di un segno = oppure <-), promuove il data.frame in un a SpatialPointsDataFrame, che ha consapevolezza circa le sue coordinates geospaziali:

```
In [57]: coordinates(Points) <- ~Xi+Yi
         class(Points)
```

'SpatialPointsDataFrame'

```
In [58]: summary(Points)
```

```
Object of class SpatialPointsDataFrame
Coordinates:
       min     max
Xi  453000  477000
Yi 4556000 4572320
Is projected: NA
proj4string : [NA]
Number of points: 441
Data attributes:
      ELEV
 Min.   :  31.2
 1st Qu.: 114.5
 Median : 233.7
 Mean   : 348.2
 3rd Qu.: 508.4
 Max.   :1297.5
```

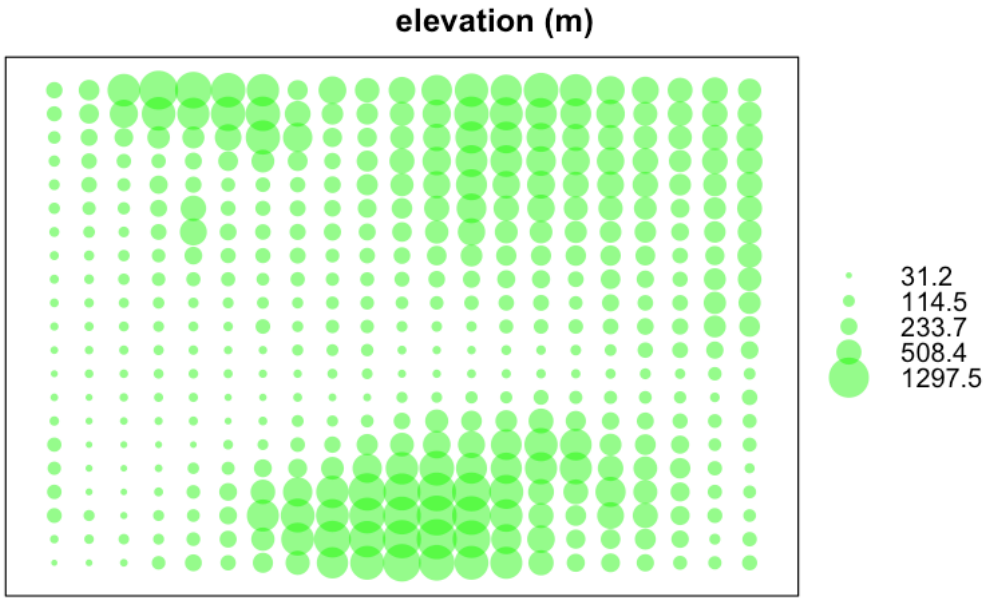
Possiamo assegnare ai punti il CRS corretto (con EPSG:32633):

```
In [59]: proj4string(Points) <- "+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"
         summary(Points)
```

```
Object of class SpatialPointsDataFrame
Coordinates:
       min     max
Xi  453000  477000
Yi 4556000 4572320
Is projected: TRUE
proj4string :
[+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
+towgs84=0,0,0]
Number of points: 441
Data attributes:
      ELEV
 Min.   :  31.2
 1st Qu.: 114.5
 Median : 233.7
 Mean   : 348.2
 3rd Qu.: 508.4
 Max.   :1297.5
```

Evidenziare la presenza di un gradiente lineare e smussato The plotting function used, bubble, assumes that the x- and y-axis are the spatial coordinates:

```
In [60]: options(repr.plot.width=7,repr.plot.height=4)
         bubble(Points, "ELEV",
                col=c("#00ff0088", "#00ff0088"), main = "elevation (m)")
```



1.2.4 NOTA: creare un GRID 2 o 4 volte meno denso del DEM (se troppo pesante)

Costruzione del GRID di interpolazione, con lo stesso grigliato del DEM (differente da ELEV)

```
In [61]: DEM
```

```
class       : RasterLayer
dimensions  : 3264, 4800, 15667200  (nrow, ncol, ncell)
resolution  : 5, 5  (x, y)
extent      : 453000, 477000, 4556000, 4572320  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=utm +zone=33 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
data source : /Users/giuliano/git/Didattica/jupyter/data/dem5m.tif
names       : dem5m
```

```
In [62]: xy=as.data.frame(coordinates(DEM))
         class(xy)
```

'data.frame'

```
In [63]: coordinates(xy)=~x+y
         class(xy)
```

'SpatialPoints'

```
In [64]: gridded(xy)=TRUE
         class(xy)
```

'SpatialPixels'

```
In [65]: proj4string(xy) <- "+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"
         summary(xy)
```

```
Object of class SpatialPixels
Coordinates:
      min     max
x  453000  477000
y 4556000 4572320
Is projected: TRUE
proj4string :
[+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
+towgs84=0,0,0]
Number of points: 15667200
Grid attributes:
  cellcentre.offset cellsize cells.dim
x          453002.5        5      4800
y         4556002.5        5      3264
```

```
In [66]: summary(Points)
```

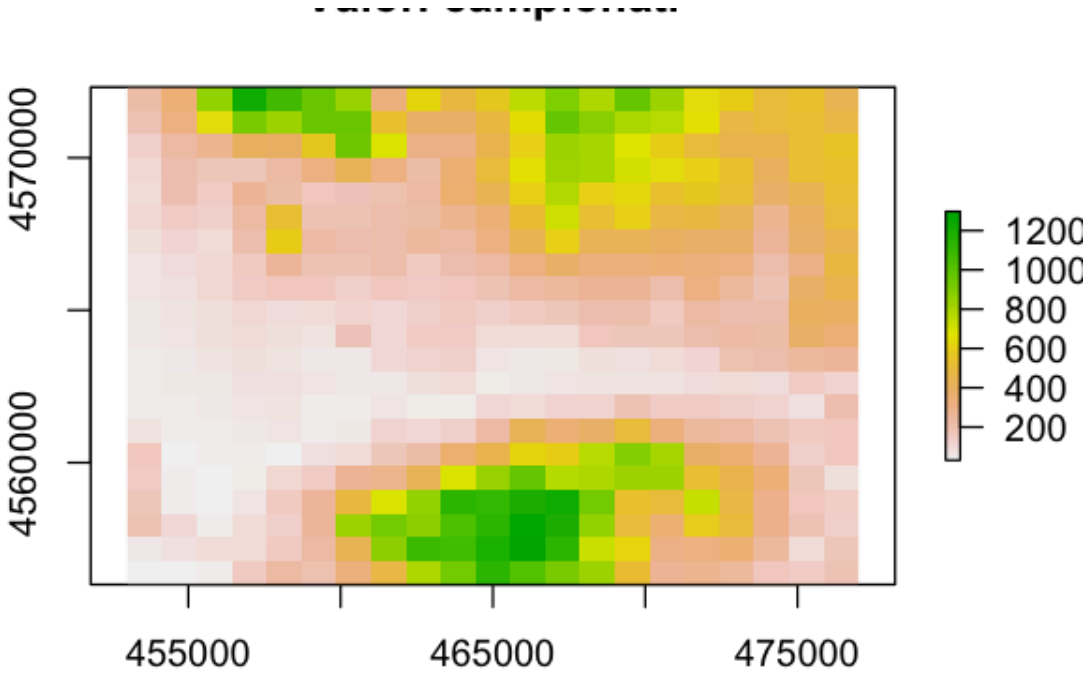
```
Object of class SpatialPointsDataFrame
Coordinates:
       min     max
Xi  453000  477000
Yi 4556000 4572320
Is projected: TRUE
proj4string :
[+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
+towgs84=0,0,0]
Number of points: 441
Data attributes:
      ELEV
 Min.   :  31.2
 1st Qu.: 114.5
 Median : 233.7
 Mean   : 348.2
 3rd Qu.: 508.4
 Max.   :1297.5
```

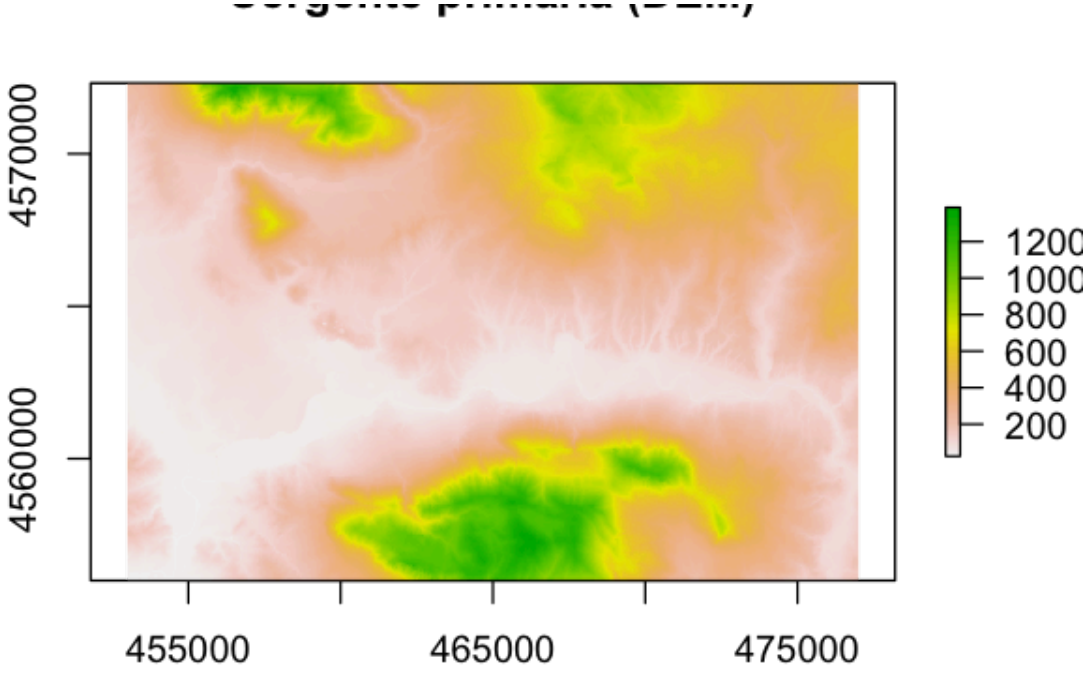
```
In [67]: elev.idw = idw(ELEV~1, Points, newdata=xy)
```

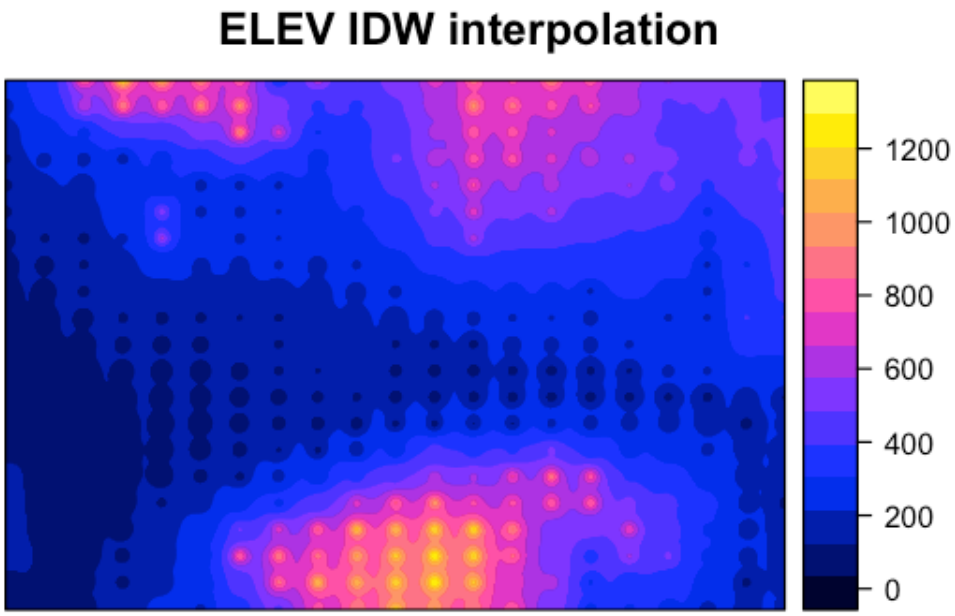
```
[inverse distance weighted interpolation]
```

```
In [68]: #par( mfrow = c( 2, 2 ) )
         options(repr.plot.width=5,repr.plot.height=3)
```

```
plot(GRID,main="Valori campionati")
plot(DEM, main="Sorgente primaria (DEM)")
spplot(elev.idw["var1.pred"], main = "ELEV IDW interpolation")
```



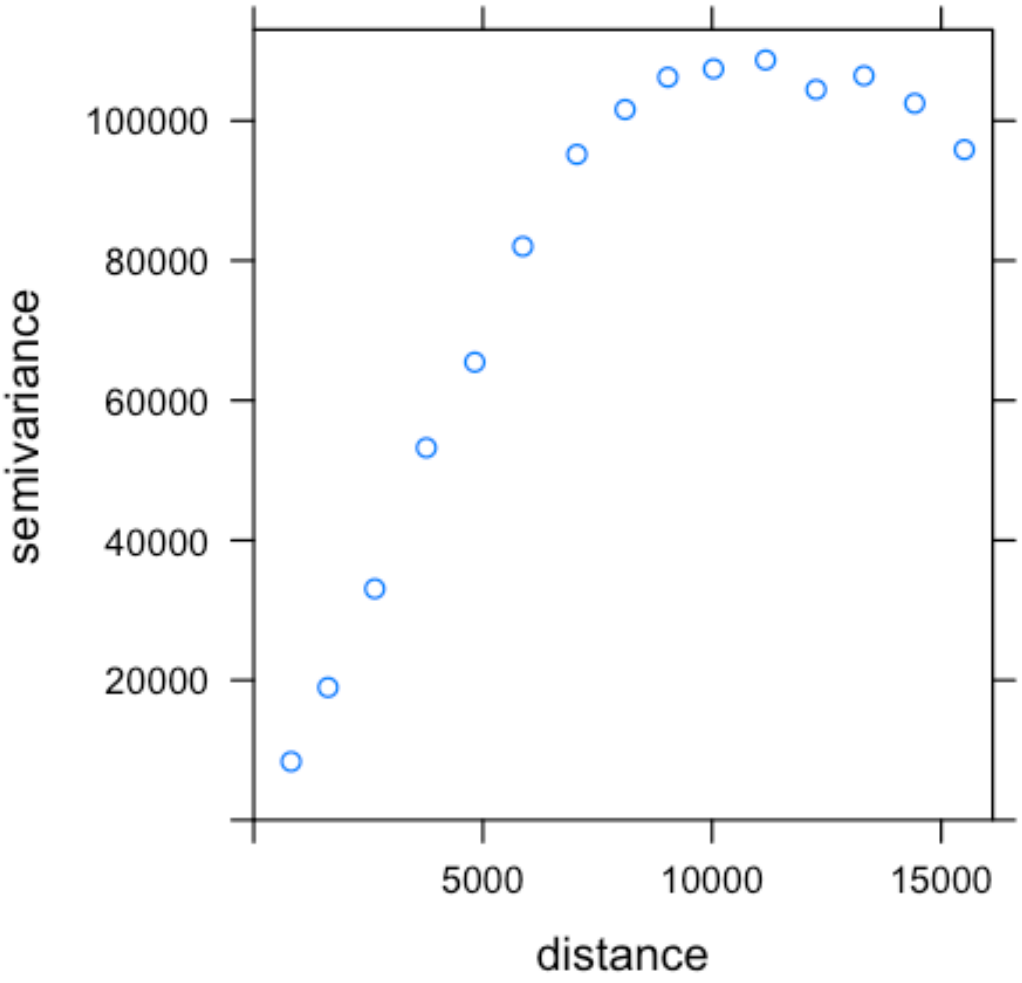


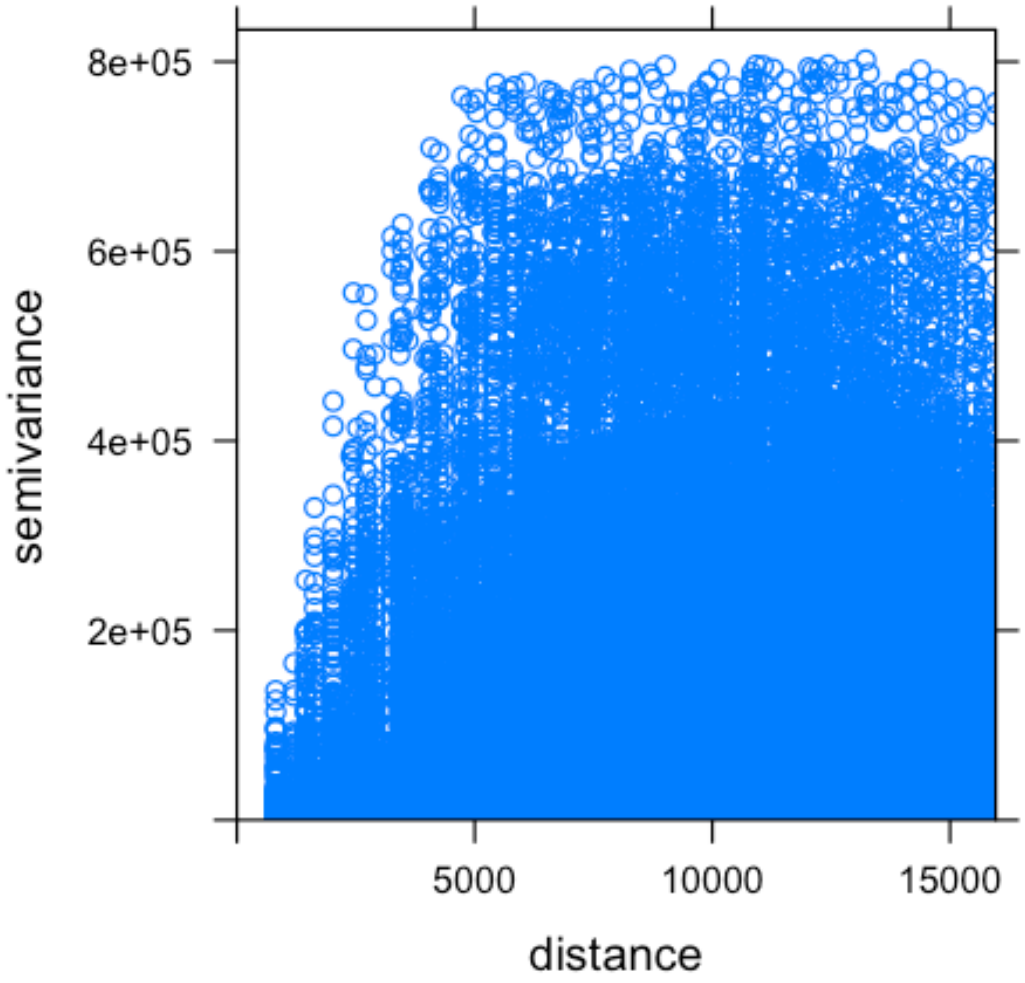


In [69]:
```
elev.vgm  = variogram(ELEV~1, Points, cutoff=16000)
elev.vgmc = variogram(ELEV~1, Points, cutoff=16000, cloud=TRUE)
elev.vgm
```

np	dist	gamma	dir.hor	dir.ver	id
420	816.000	8357.42	0	0	var1
2379	1620.689	18956.68	0	0	var1
2979	2641.119	33065.65	0	0	var1
5125	3767.210	53226.32	0	0	var1
4466	4822.958	65469.04	0	0	var1
6390	5871.565	82022.20	0	0	var1
7094	7050.894	95182.81	0	0	var1
5731	8103.642	101599.59	0	0	var1
6071	9035.502	106212.90	0	0	var1
6701	10032.662	107410.44	0	0	var1
7569	11167.861	108677.88	0	0	var1
5923	12270.723	104455.90	0	0	var1
6201	13315.492	106429.99	0	0	var1
5710	14422.137	102484.69	0	0	var1
4954	15504.799	95864.11	0	0	var1

In [70]:
```
options(repr.plot.width=4,repr.plot.height=4)
plot(elev.vgm)
plot(elev.vgmc)
```


semivariance
100000
80000
60000
40000
20000
5000
10000
15000
distance




```
In [71]: elev.fit1 = fit.variogram(elev.vgm, model = vgm(10000, "Sph", 8000, 1000) )

         elev.fit2 = fit.variogram(elev.vgm, model = vgm(10000, "Sph", 5000,   00),
                                   fit.sills = TRUE, fit.ranges = TRUE, fit.method=7 )

         elev.fit3 = fit.variogram(elev.vgm, model = vgm(10000, "Sph", 5000,   00),
                                   fit.sills = TRUE, fit.ranges = TRUE, fit.method=1 )


         elev.fit4 = fit.variogram(elev.vgm, model = vgm(95000,"Sph", 10000, 5000),
                                  fit.sills = FALSE, fit.ranges = FALSE )

In [72]: elev.fit1
         elev.fit2
```

```
elev.fit3
elev.fit4
```

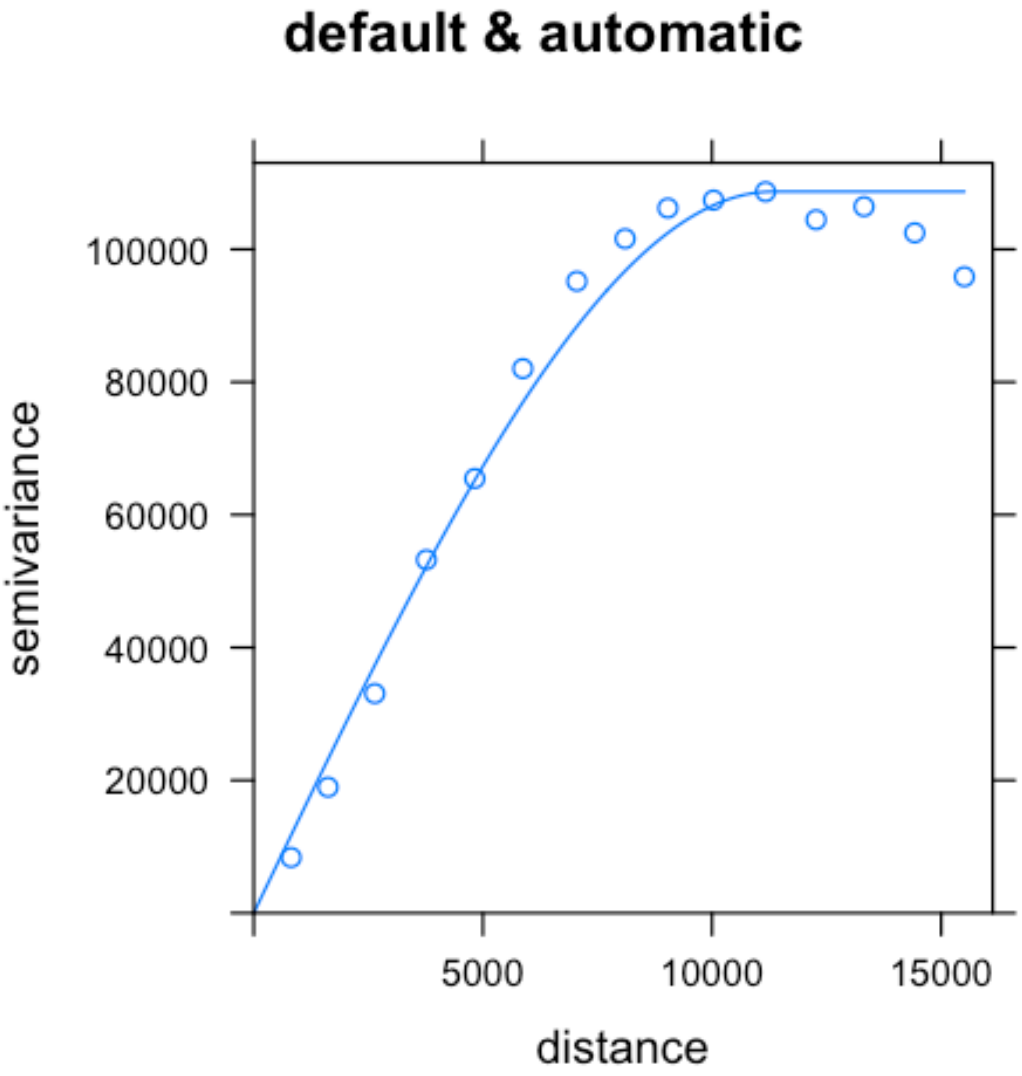
model	psill	range	kappa	ang1	ang2	ang3	anis1	anis2
Nug	0.0	0.00	0.0	0	0	0	1	1
Sph	108686.6	11325.86	0.5	0	0	0	1	1

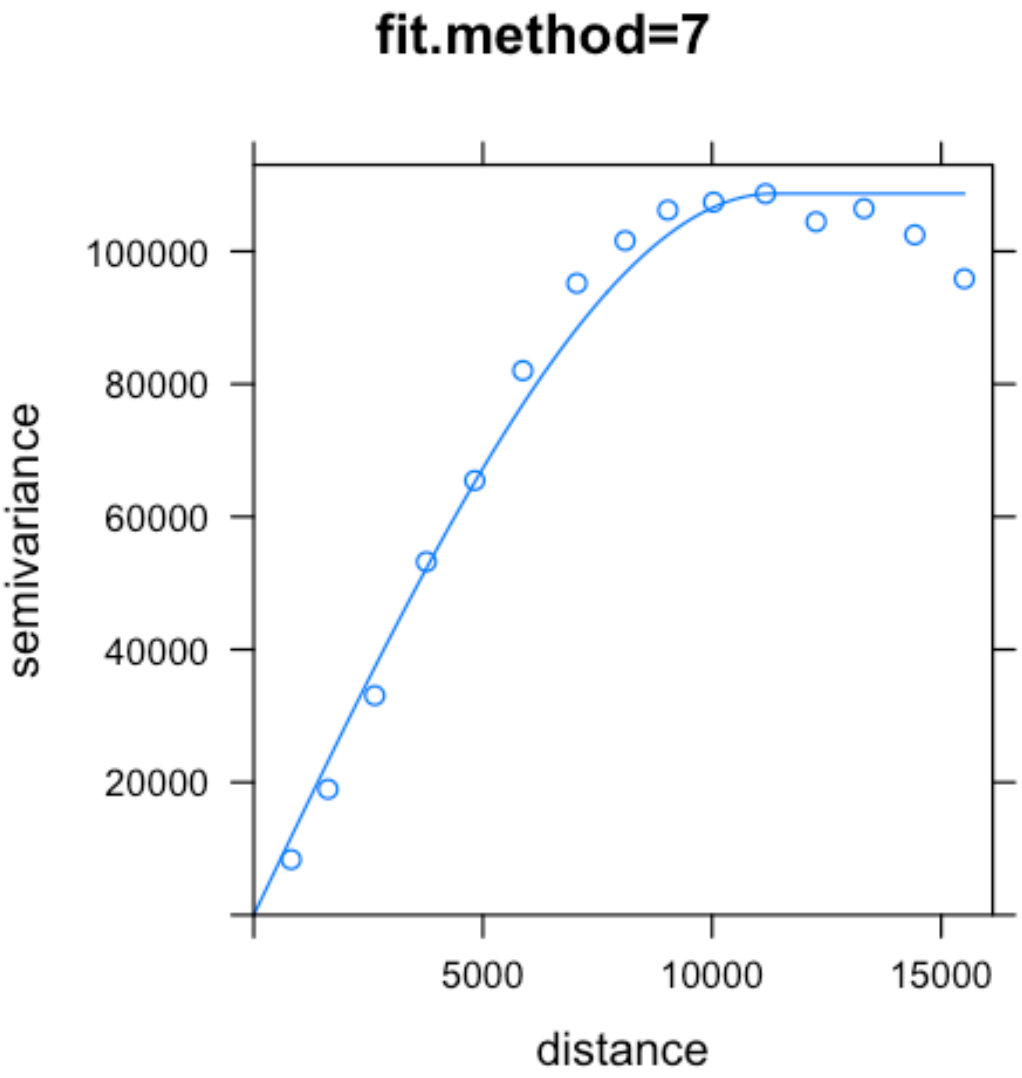
model	psill	range	kappa	ang1	ang2	ang3	anis1	anis2
Nug	0.0	0.00	0.0	0	0	0	1	1
Sph	108687.3	11325.98	0.5	0	0	0	1	1

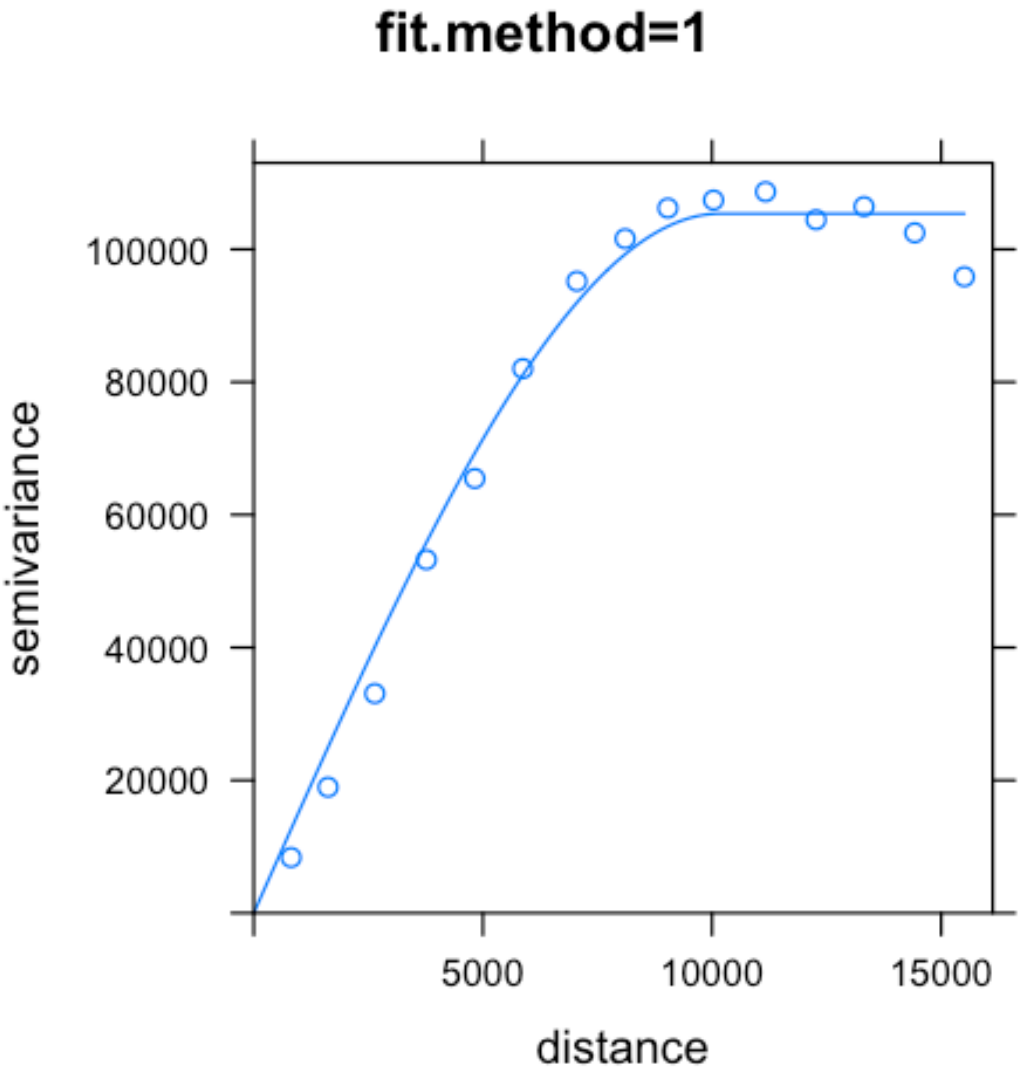
model	psill	range	kappa	ang1	ang2	ang3	anis1	anis2
Nug	0.0	0.00	0.0	0	0	0	1	1
Sph	105376.2	10178.26	0.5	0	0	0	1	1

model	psill	range	kappa	ang1	ang2	ang3	anis1	anis2
Nug	5000	0	0.0	0	0	0	1	1
Sph	95000	10000	0.5	0	0	0	1	1

In [73]:
```
par( mfrow = c( 2, 2 ) )
plot(elev.vgm,elev.fit1,main="default & automatic")
plot(elev.vgm,elev.fit2,main="fit.method=7")
plot(elev.vgm,elev.fit3,main="fit.method=1")
plot(elev.vgm,elev.fit4,main="manual setting")
```


default & automatic
semivariance
100000
80000
60000
40000
20000
5000
10000
15000
distance


fit.method=7
semivariance
100000
80000
60000
40000
20000
5000
10000
15000
distance


fit.method=1
semivariance
100000
80000
60000
40000
20000
5000
10000
15000
distance