HW_Cucca_Armano

2024-11-18

```
setwd("C:/Users/inter/Desktop/statspaz")
```

E' stata settata la working directory

```
library(sp)
NO2 <- read.csv("NO2.csv")
no2<-NO2
class(no2)

## [1] "data.frame"

confini <- read.table("confini_piemonte.txt", header = T)
head(confini)

##          x       y
## 1 413904.4 5049225
## 2 413922.4 5049234
## 3 413969.2 5049256
## 4 414038.8 5049311
## 5 414081.7 5049349
## 6 414112.9 5049385
```

Abbiamo caricato il pacchetto "sp". Il pacchetto serve per lavorare su classi e metodi riguardanti dati spaziali.

E' stato caricato il dataset riguardante i dati di NO2 e anche quello sui confini della regione piemonte. Abbiamo rinominato il dataset NO2 in no2.

```
no2$UTM_X <- no2$UTM_X*1000
no2$UTM_Y <- no2$UTM_Y*1000


head(no2)

##    UTM_X   UTM_Y STAZIONE  NO2_OBS   NO2_CTM
## 1 394518 5001005  Borgaro 37.00000 51.623567
## 2 362750 5032242 Ceresole  5.00000  0.327836
## 3 408380 4983914 ChieriBe 24.00000 33.488997
## 4 386869 5003485  Druento 27.00000 19.767697
## 5 412269 5033687  IvreaLi 54.00000 20.367582
## 6 398765 5003348    Leini 45.41315 40.910211
```

Abbiamo ricodificato le due variabili UTM_X e UTM_Y (ci restituiscono le coordinate dei siti) per poter osservare i siti all'interno dei confini del piemonte ma anche nelle regioni confinanti.

```
library(ggplot2)
```

```
## Warning: il pacchetto 'ggplot2' è stato creato con R versione 4.3.3
```

```
library(sf)
```

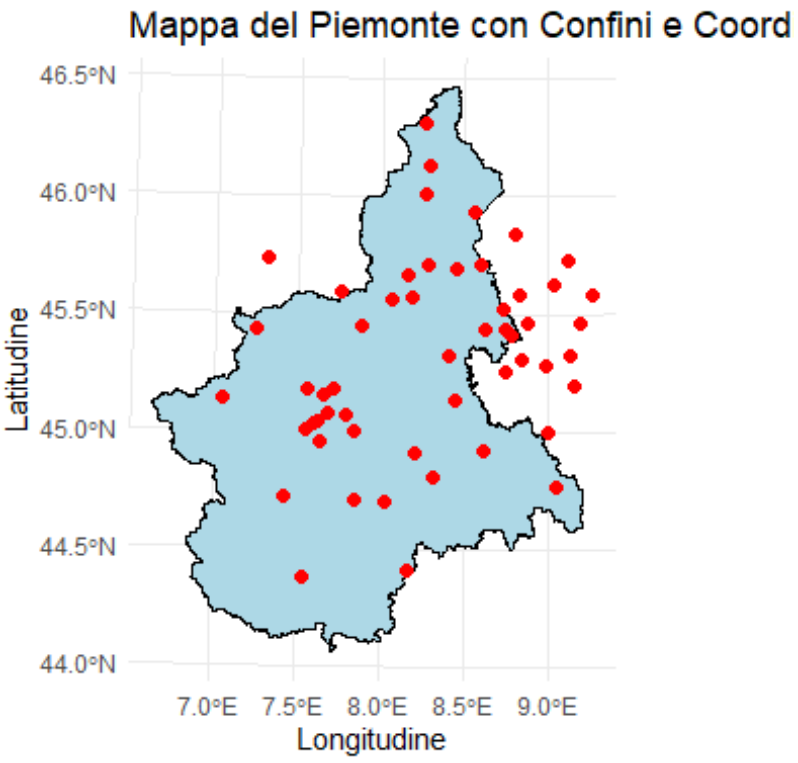
```
## Linking to GEOS 3.11.2, GDAL 3.7.2, PROJ 9.3.0; sf_use_s2() is TRUE
```

```
confini_sf <- st_as_sf(confini, coords = c("x", "y"), crs = 32632)
```

```
confini_sf <- st_cast(st_combine(confini_sf), "POLYGON")
```

```
no2_sf <- st_as_sf(no2, coords = c("UTM_X", "UTM_Y"), crs = 32632)
```

```
ggplot() +
  geom_sf(data = confini_sf, fill = "lightblue", color = "black", size = 0.5) +  # Confini del Piemonte
  geom_sf(data = no2_sf, color = "red", size = 2) +                               # Punti del dataset 'no2'
  theme_minimal() +
  labs(title = "Mappa del Piemonte con Confini e Coordinate",
       x = "Longitudine", y = "Latitudine")
```



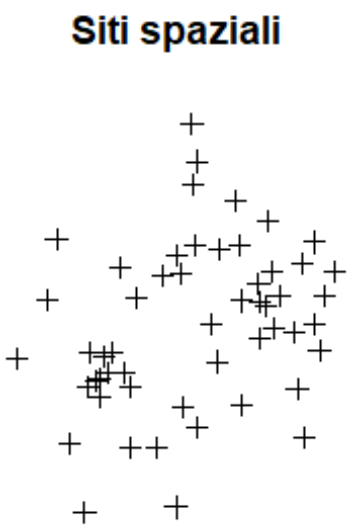
Abbiamo caricato i pacchetti “ggplot2” per ottenere una visualizzazione grafica e “sf” (simple features) che ci permette di lavorare con dati spaziali. I dataset “confini” e “no2” sono stati convertiti in un oggetto sf selezionando le colonne delle coordinate per entrambi i dataset, viene inserito lo stesso crs per ottenere la medesima proiezione grafica e sistema di coordinate. Con la funzione st_cast() richiamando “POLYGON” riusciamo a ottenere un poligono dalla geometria

combinata dalla funzione st_combine(), questo ci serve per ottenere un singolo poligono che rappresenterà tutta l'area. E' stato creato il grafico tramite la funzione ggplot() che utilizza geom_sf() per poter ottenere una visualizzazione grafica utilizzando dei dati spaziali; il grafico è stato personalizzato attraverso la scelta di colori e dimensioni.

```
coordinates(no2) <- c("UTM_X", "UTM_Y")

class(no2)

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

names(no2)

## [1] "STAZIONE" "NO2_OBS"  "NO2_CTM"
```

Specifichiamo quali colonne sono le coordinate e converte ad un oggetto SPDF. Con la funzione names() guardiamo i nomi delle colonne e possiamo osservare come UTM_X e UTM_Y non vengano più restituiti perchè ora sono le nostre coordinate.

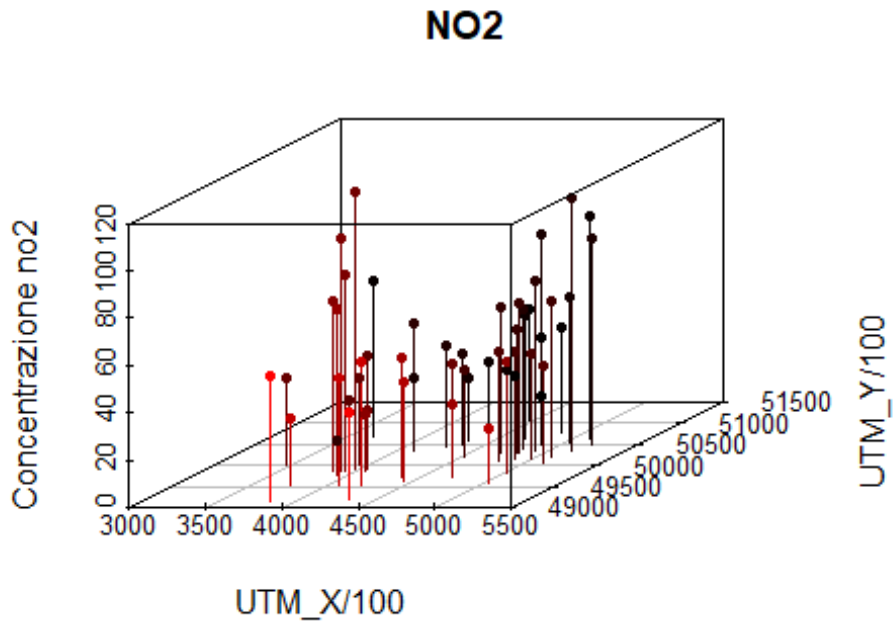
```
plot(no2, main="Siti spaziali")
```



Visualizziamo il plot che ci restituisce solamente i siti del nostro dataset, che vengono inseriti attraverso le coordinate create precedentemente.

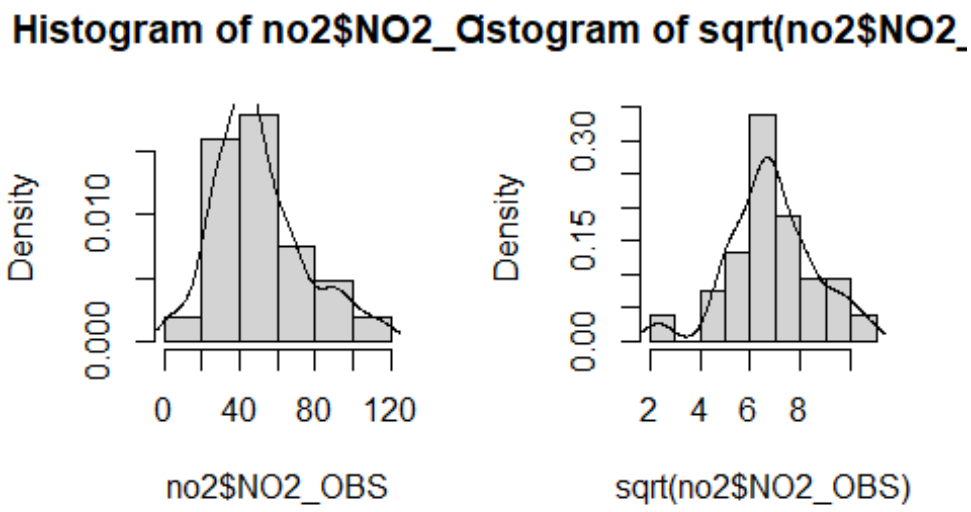
```
library(scatterplot3d)
scatterplot3d(no2$UTM_X/100,no2$UTM_Y/100,no2$NO2_OBS,type = "h", highlight.3d =
```

```
TRUE, pch = 20,main = "NO2",xlab='UTM_X/100',ylab='UTM_Y/100',zlab='Concentrazione
no2')
```



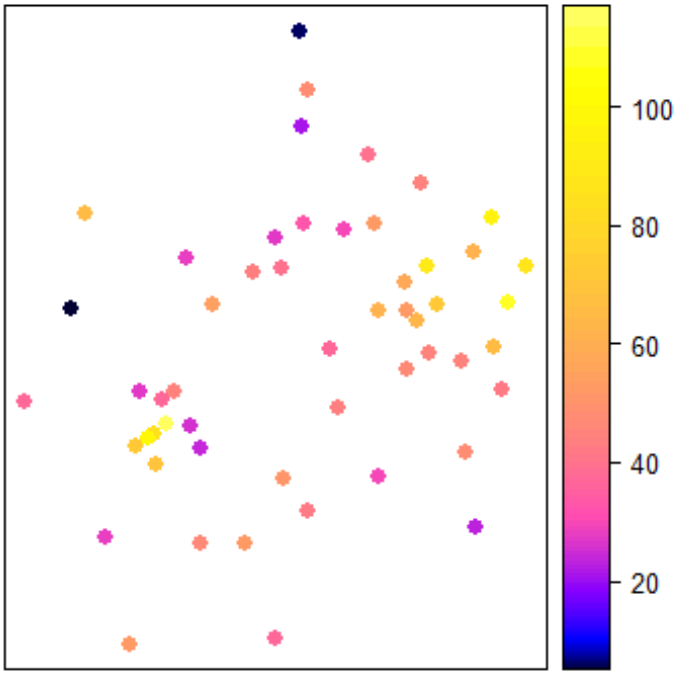
Carichiamo il pacchetto "scatterplot3d" per ottenere dei grafici in 3D. Abbiamo creato un grafico 3D scalando le nostre variabii UTM_X e UTM_Y per non avere valori troppo elevati, è possibile osservare il valore di NO2_OBS (concentrazione di NO2) in base alla posizione del sito , con "highlight.3d = T" otteniamo punti 3D in differenti colori in base alla relazione con Y migliorando la visibilità dei punti nello spazio.

```
par(mfrow=c(1,2),pty='s')
hist(no2$NO2_OBS, freq=FALSE)
lines(density(no2$NO2_OBS))
hist(sqrt(no2$NO2_OBS), freq=FALSE)
lines(density(sqrt(no2$NO2_OBS)))
```

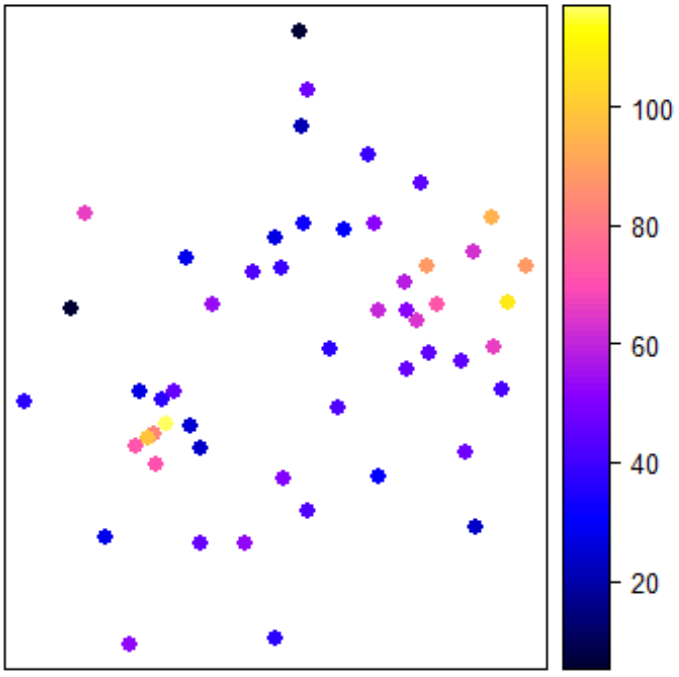


Abbiamo diviso l'area dei grafici in 2 per poter inserire i due plot insieme riguardanti NO2_obs. Sono stati messo a confronto i due istogrammi riguardanti i dati originali e trasformati con la radice quadrata, con freq = FALSE impostiamo l'asse della Y sulla densità; dal secondo grafico coi dati trasformati è possibile notare una migliore simmetria anche attraverso la stima della densità ottenuta grazie ala funzione lines().

```
spplot(no2, "NO2_OBS", do.log = T, colorkey = TRUE)
```

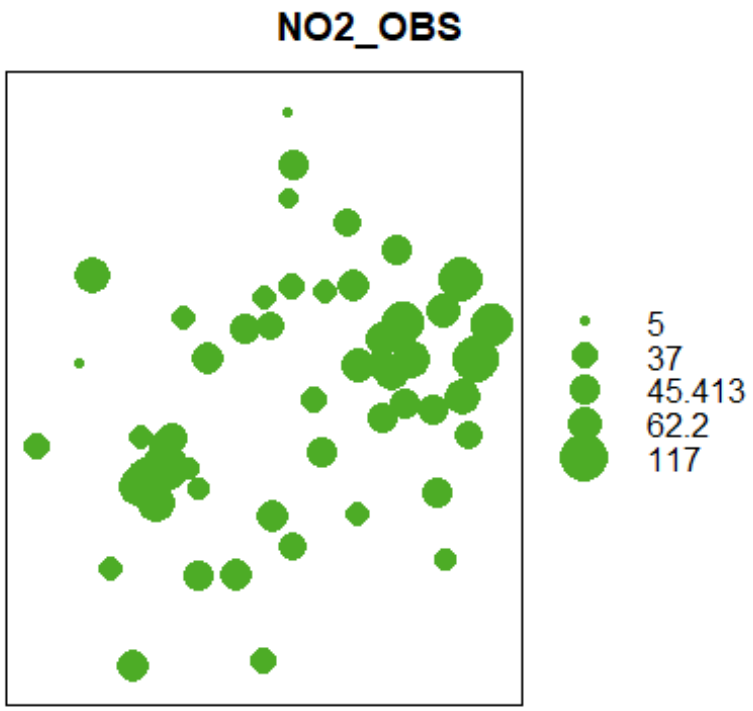



```
spplot(no2, "NO2_OBS", do.log = F, colorkey = TRUE)
```



Da questi grafici possiamo osservare la quantità di NO2_OBS per ogni sito, inserendo do.log = con T riuscimo a vedere meglio le differenze basandole con il log.

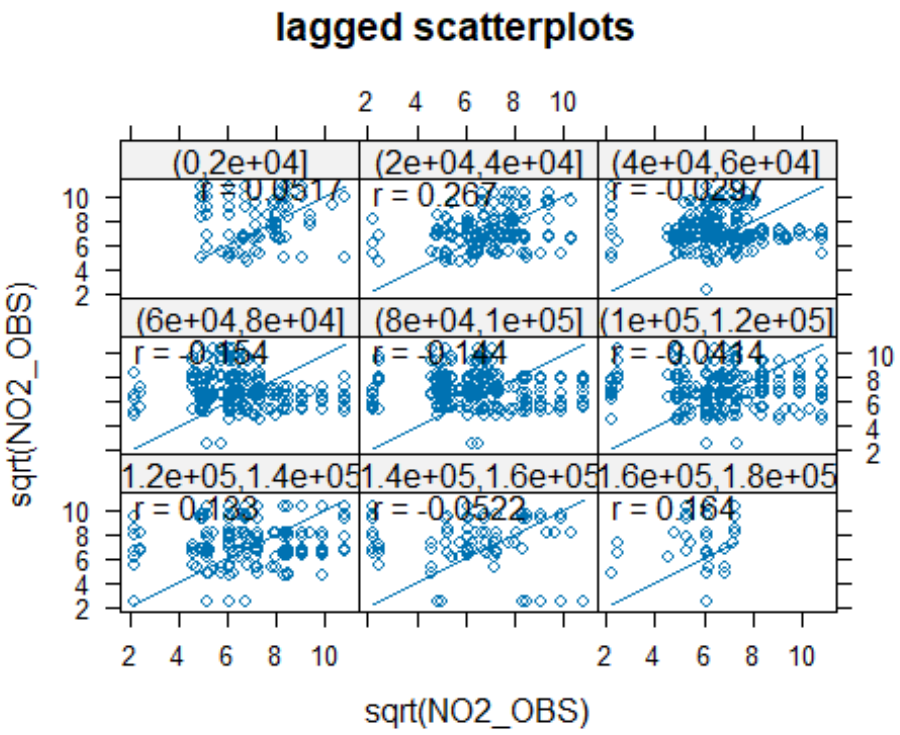
```
bubble(no2, "NO2_OBS", do.log = T)
```



Questo plot ha lo stesso scopo di quelli visualizzati precendemente, ma questa volta per capire la quantità di NO2_OBS per ogni sito non andremo a visualizzare il colore dei pallini ma la grandezza.

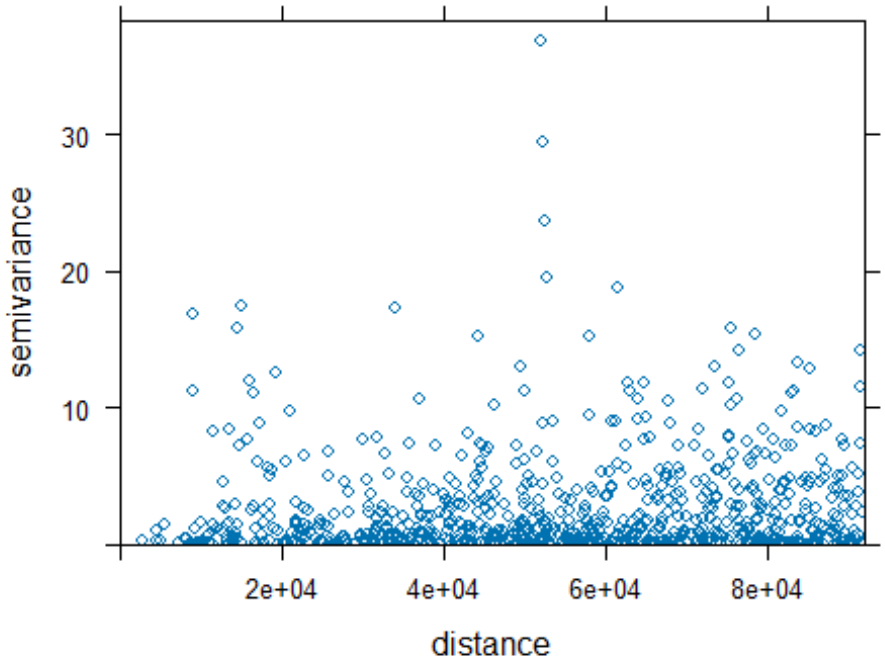
#VARIOGRAFIA

```
dist <- dist(coordinates(no2), method="euclidean")
summary(dist)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2603   50533   81117   82851  112981  222005

library(gstat)

## Warning: il pacchetto 'gstat' è stato creato con R versione 4.3.2

hscat(sqrt(NO2_OBS) ~ 1, no2, (0:9)*20000)
```

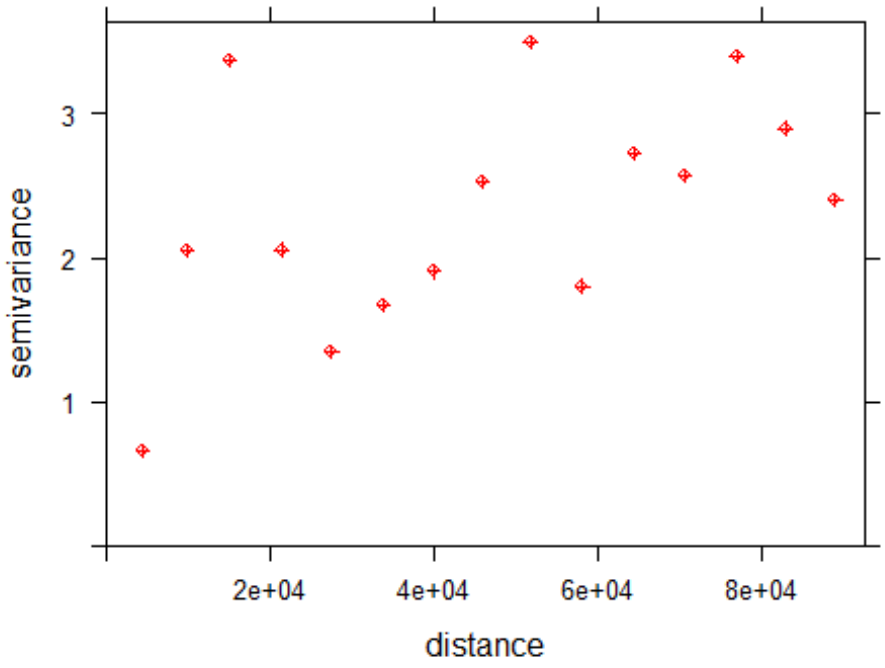


Procediamo con l'analisi del varianza grazie ai variogrammi e alla variografia. iniziamo richiamando un summary() dell'oggetto distanza per creare uno scatterplot che contenga range di distanze sensati. richiamo il pacchetto Gstat necessario per l'intera analisi sulla varianza spaziale dopodiche cominciamo a studiare la correlazione spaziale della variabile presa in considerazione, la radice del no2 osservato (per far si che la var sia piu simmetrica), con il lagged scatterplot grazie al comando hscat(), specificando gli intervalli delle distanza per i quali vogliamo le coppie di osservazioni (distanti h l'una dall'altra). Il lagged scatterplot non sembra darci molta informazione sula correlazione spaziale della variabile, pensiamo che questo sia dovuto ad una scarsa numerosita di osservazioni

```
cld <- variogram(sqrt(NO2_OBS) ~ 1, no2, cloud = TRUE)
svgm <- variogram(sqrt(NO2_OBS) ~ 1, no2)
#semplice plot
plot(cld)
```



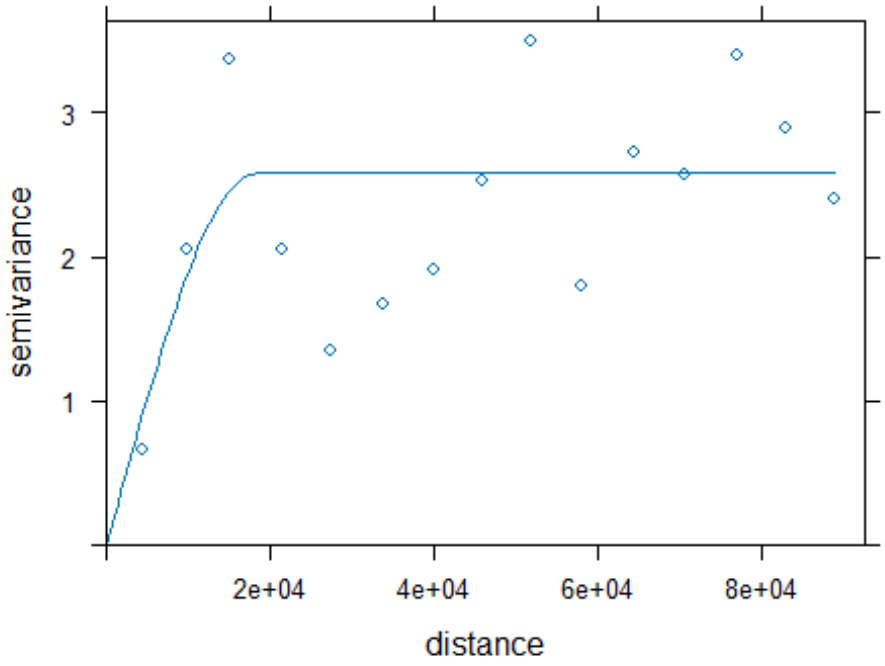

```
plot(svgm, col="red", pch=10)
```



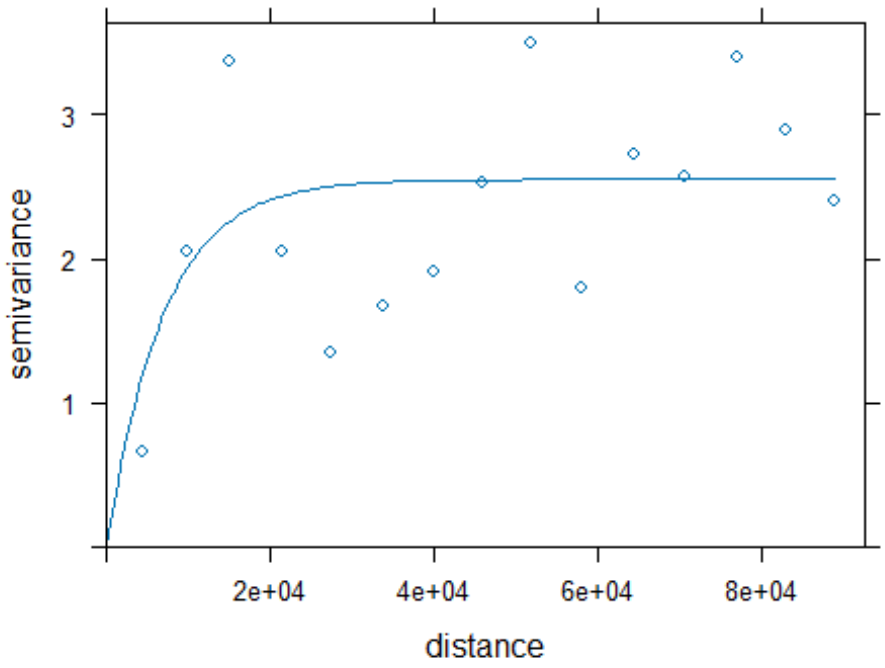
procediamo con la creazione della nuvola di variogramma e del variogramma campionario. Il comando per la creazione degli oggetti è il medesimo cioè variogram(), come si nota il comando richiede una funzione cosicche dopo riesca a fare i residui. Notiamo anche che per ottenere la nuvola di

variogramma abbiamo dovuto mttere l'argomento cloud=TRUE. Dopodiche passiamo al plot dei due oggetti con l'ononimo comando, mostrano la varianza di tutte le coppie di valori ad una data distanza, il cloud di tutte le coppie , il variogramma campionario stima quelle ottenute con la rispettiva formula

```
v.fit <- fit.variogram(svgm, vgm(2, "Sph" ,40000,0.5))

## Warning in fit.variogram(object, model, fit.sills = fit.sills, fit.ranges =
## fit.ranges, : No convergence after 200 iterations: try different initial
## values?

v.fit2 <- fit.variogram(svgm, vgm(2, "Exp" ,20000,0.5))

## Warning in fit.variogram(svgm, vgm(2, "Exp", 20000, 0.5)): No convergence after
## 200 iterations: try different initial values?

plot(svgm,v.fit)
```




```
plot(svgm,v.fit2)
```




```
v.fit

##   model    psill    range
## 1   Nug 0.000000     0.00
## 2   Sph 2.576338 18360.58

v.fit2

##   model    psill    range
## 1   Nug 0.000000    0.000
## 2   Exp 2.549844 6922.617
```

Dopo aver ottenuto il variogramma campionario procediamo con la variografia adattandoci modelli teorici di variogramma che andranno astimare i parametri partendo dai nostri. Con la funzione fit.variogram() procediamo prima con un modello sferico indicando rispettivamente i parametri di inizializzazione psill, il tipo di modello teorico, il range e il nugget. pur non sembrando facile adattare una variogramma teorico a quello campionario visto la sua struttura vediamo come il modello sferico e quello esponenziale si adattino similiramente, per capire effettivamente il migliore calcoliamo per tutti e due il SSerr con il comando attr()

```
attr(v.fit, "SSErr")

## [1] 3.756213e-07

attr(v.fit2, "SSErr")

## [1] 4.775764e-07
```

Oltre che confermare quanto ci sembrava, cioè un adattamento molto molto simile dei duei modelli di variogramma , possiamo dire che quello sferico sia leggermente migliore anche se la differenza rimane minima.