"Tecniche di impiego del software open-source per l'analisi geostatistica e l'econometria territoriale"

Iacopo Bernetti

ATTENZIONE: BOZZA NEANCHE RILETTA!

Introduzione

L'analisi spaziale dei geodati spesso si avvale di strumenti propri delle varie branche della statistica, arrivando a proporre metodologie di analisi dedicate, quali la geostatistica e l'analisi statistica spaziale. Contrariamente a quanto saremmo portati a pensare, la possibilità di attingere ad una massa sterminata di informazioni, propria dei geodati, rischia di impedire di fatto di utilizzarne anche solo una parte: non basta infatti avere solo l'accesso teorico ad una informazione, ma occorre che essa sia effettivamente fruibilie.

E` proprio questo il problema centrale della statistica: rendere davvero utilizzabili grandi quantità di informazioni, teoricamente disponibili, ma di fatto difficilmente gestibili, relative agli oggetti della propria indagine. Infatti tutte le informazioni - per contribuire effettivamente ad accrescere la conoscenza di un fenomeno - hanno bisogno di essere trattate da vari punti di vista: occorrono tecniche accurate di rilevazione, occorre procedere a accurate selezioni, occorre un lavoro di organizzazione e di sintesi. D'altra parte il lavoro statistico ha senso solo se si confronta con grandi quantità di informazioni. La statistica raccoglie e restituisce in forma organizzata grandi quantità di informazioni sulla base di una duplice esigenza: quella predittiva e quella descrittiva. Per questo motivo la scienza statistica è comunemente suddivisa in due branche principali: statistica descrittiva e statistica inferenziale.

La statistica descrittiva ha come scopo quello di sintetizzare i dati attraverso i suoi strumenti grafici (diagrammi a barre, a torta, istogrammi, boxplot) e indici (indicatori statistici , indicatori di posizione come la media, di variazione come la varianza e la concentrazione, di correlazione, ecc.) che descrivono gli aspetti salienti dei dati osservati, formando così il contenuto statistico.

La statistica inferenziale ha come obiettivo, invece, quello di fare affermazioni, con una possibilità di errore controllata, riguardo la natura teorica (la legge probabilistica) del fenomeno che si osserva. La conoscenza di questa natura permetterà poi di fare previsione (si pensi, ad esempio, che quando si dice che "l'inflazione il prossimo anno avrà una certa entità" deriva dal fatto che esiste un modello dell'andamento dell'inflazione derivato da tecniche inferenziali). La statistica inferenziale è fortemente legata alla teoria della probabilità. La statistica inferenziale si suddivide poi in altri capitoli, di cui i più importanti sono la stima puntuale, la stima intervallare e la verifica delle ipotesi.

Intorno al 1950, a questi due primi capitoli della statistica, se ne affiancò un terzo, la statistica esplorativa, ad opera di John Wilder Tukey. In questo approccio i dati risultati da un esperimento vengono indagati attraverso metodi di sintesi (grafica e numerica) al fine di formulare ipotesi riguardo la legge di probabilità sottesa al fenomeno studiato (questa è la differenziazione con la statistica inferenziale, in cui è sempre sottesa un'ipotesi riguardo la legge di probabilità di cui i dati sono la controparte osservabile).

I metodi di statistica descrittiva, inferenziale ed esplorativa costituiscono strumenti molto utili per l'analisi dei geodati, ma non sono in grado di considerare esplicitamente le caratteristiche

geometriche e spaziali proprie dell'informazione territoriale. Proprio per questo motivo sono stati proposte metodologie di indagine specifiche per le stime statistiche spaziali, quali la statistica spaziale e la la geostatistica.

La geostatistica offre delle tecniche di interpolazione spaziale che possono fornire delle stime sul valore assunto da una variabile in una posizione in cui la misurazione non è stata effettuata in base a dei dati rilevati su punti vicini. La geostatistica ambientale fornisce le basi per capire ed utilizzare le varie tecniche per la stima del valore di una variabile spaziale nelle aree dove tale variabile non è stata misurata. In altre parole è uno strumento per fare delle ipotesi “il più possibile” corrette sulla continuità spaziale di una data variabile. La geostatistica studia i fenomeni territoriali che si sviluppano su base spaziale a partire dalle informazioni derivanti da un loro campionamento. I metodi di interpretazione di superfici che vengono comunemente utilizzati possono essere di natura sia deterministica che stocastica

Il concetto di base della statistica spaziale deriva direttamente dalla “prima legge della geografia” di Tobler secondo cui: “everything is related to everything else, but near things are more related than distant things” (Tobler, 1979). Il concetto alla base della statistica spaziale e la co-dipendenza o co-variazione delle variabili nello spazio geografico: cioè i valori vicini possono essere fra loro correlati positivamente o negativamente. Ci sono almeno tre possibili spiegazioni per questo fenomeno. Una prima causa può la co-occorrenza di particolari fattori nel medesimo spazio geografico: per esempio elevati tassi di criminalità sono generalmente associati a fenomeni di povertà e di degrado sociale e urbanistico. Una seconda causa può essere determinata da fattori di causa-effetto, tipica delle variabili climatiche e ambientali, che tendono ad avere valori simili in dipendenza di analoghe variabili territoriali, come distanza dalla costa, quota, ecc.. Infine una terza causa può derivare circoli viziosi o virtuosi derivanti da interazioni socioeconomiche: la facilità di scambi di informazione e di ricchezza può portare a condizioni di sviluppo economico e di benessere sociale omogenee rispetto alla localizzazione spaziale.

R!

Introduzione

R è un ambiente costituito da librerie, macro e oggetti finalizzati all'elaborazione e all'analisi grafica dei dati, principalmente con metodi statistici. R è basato sul linguaggio S a cui è strettamente legato un altro `ambiente' commerciale probabilmente più conosciuto, S-Plus. R, a differenza di S-Plus, è un GNU-Sofware, ovvero disponibile gratuitamente sotto i vincoli della GPL (General Public Licence) presso il sito http://cran.r-project.org/ per diversi ambienti operativi. Nel presente capitolo saranno dati solamente gli elementi di R utili ai fini dell'analisi statsitica dei geodati, rimandando alla ampia letteratura disponibile in rete per approfondimenti[1].

[1] Claudio Agostinelli, Introduzione ad R, versione 0.3, ottobre 2000 (in formato PDF e PostScript) http://www.dst.unive.it/~claudio/R/index.html#manuale [consultato in data 15/01/04]
Roberto Baggiani, Introduzione ad R, versione 6.0, 24 ottobre 2004 http://digilander.libero.it/robicox/manuali/pdf/mainr.pdf (in formato PDF) [consultato in data 26/10/04]
A. Brazzale, M. Chiogna, C. Gaetan e N. Sartori, Laboratorio di R, Materiale didattico per i laboratori del corso di Modelli Statistici I, A.A. 2001/2002, ISIB-CNR, Padova (in formato PDF e PS) http://www.isib.cnr.it/~brazzale/ModStatI/ [consultato in data 20/05/04]
Stefano Iacus, Corso introduttivo all'uso dell'ambiente statistico R, Dipartimento di Economia Politica e Aziendale, Università di Milano marzo 2003 (in formato HTML) http://www.economia.unimi.it/iacus/corsoR/ [consultato in data

R è’ un ambiente interattivo a linea di comando, ossia i comandi producono una risposta immediata, e prevedono una programmazione orientata agli oggetti. Avviato il programma R, apparirà una nuova finestra con il simbolo “>” (figura 1). che indica che l’ambiente è pronto per ricevere delle istruzioni.

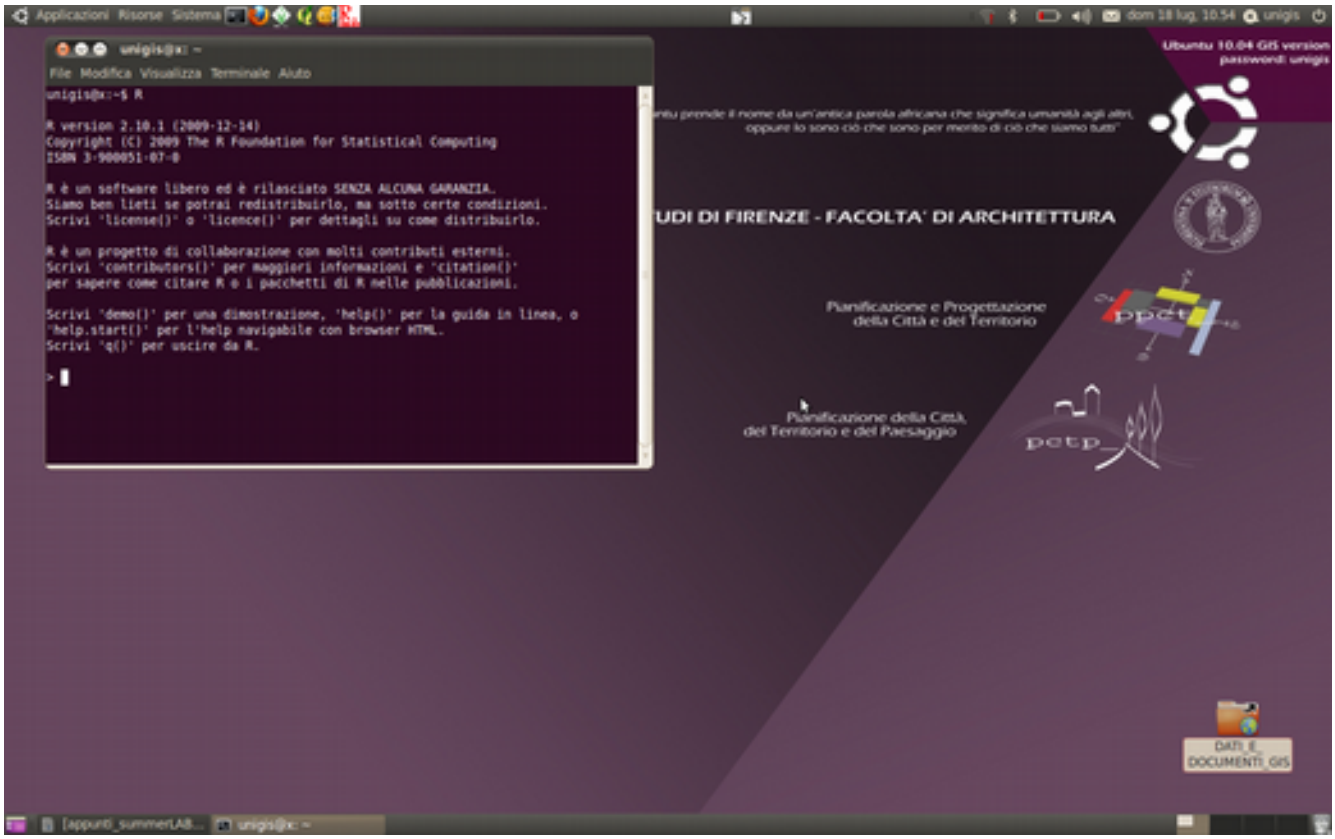


Figura 1. Shell comandi di R in Ubuntu unigis.

18/02/04]
Angelo M. Mineo, Una guida all'utilizzo dell'ambiente statistico R, 2003 (in formato PDF) http://cran.r-project.org/doc/contrib/Mineo-dispensaR.pdf [consultato in data 15/01/04]
Vito M. R. Muggeo, Il linguaggio R: concetti introduttivi ed esempi, versione 1.0, giugno 2002 (in formato PDF) http://cran.r-project.org/doc/contrib/nozioniR.pdf [consultato in data 15/01/04]
Francesca Parpinel, La statistica applicata attraverso il programma R, febbraio 2000 (in formato PDF) http://venus.unive.it/statcomp/r/man_Parpinel.pdf [consultato in data 15/01/04]
Lea Petrella, Silvia Polettini, Laboratorio di Statistica I, metodi matematici e statistici, Dipartimento di Matematica, Università Roma 3, 2002 (in formato PDF e PostScript) http://www.mat.uniroma3.it/didatticacds/corsi/didattica_interattiva/aa_01_02/st1/st1.html [consultato in data 15/01/04]
Alessio Pollice, Esercitazioni con R, Dipartimento di Scienze Statistiche, Università di Bari (in formato PDF) http://www.dip-statistica.uniba.it/html/docenti/pollice/materiale.htm [consultato in data 25/10/04]
Vito Ricci, Analisi delle serie storiche con R, novembre 2004 (in formato PDF) http://cran.r-project.org/doc/contrib/Ricci-ts-italian.pdf [consultato in data 30/11/04]
Luca Scrucca, Note sul linguaggio e ambiente statistico R, Dipartimento di Scienze Statistiche, Università degli Studi di Perugia, 18 ottobre 2004 (in formato PDF) http://www.stat.unipg.it/~luca/R-note.pdf [consultato in data 26/10/04]

Il programma presenta una serie di pacchetti aggiuntivi che ne aumentano notevolmente la potenzialità. Quando il programma `e installato oltre alla sua configurazione base sono automaticamente installati una serie di pacchetti aggiuntivi che sono molto utili nelle analisi statistiche. L'elenco di tali pacchetti si ottiene con il comando:

```
library()
```

Alcuni di questi pacchetti non sono immediatamente utilizzabili ma devono essere caricati in R con il comando:

```
library("nomepacchetto")
```

In rete possiamo trovare molti pacchetti che permettono di svolgere molteplici analisi statistiche dei dati. Per caricare tali pacchetti dobbiamo digitare da shell il seguente comando:

```
install.packages("nome pacchetto", dependencies=TRUE)
```

l'opzione `dependencies=TRUE` consente di installare tutti i pacchetti che sono necessari per eseguire quello che desideriamo istallare. Da oltre 10 anni R ha incrementato il numero di pacchetti finalizzati all'analisi spaziale dei geodati. Tutti pacchetti fanno riferimento ad un "capostipite" chiamato **sp** che contiene gli elementi di base necessari (le cosiddette classi, cioè le strutture di dati e i metodi o procedure che agiscono su tale struttura) per l'elaborazione dei geodati (punti, linee, poligoni e dati raster). Sulla base del pacchetto **sp** sono stati creati molti altri successivi pacchetti per l'analisi spaziale generalistica o specialistica (figura x).

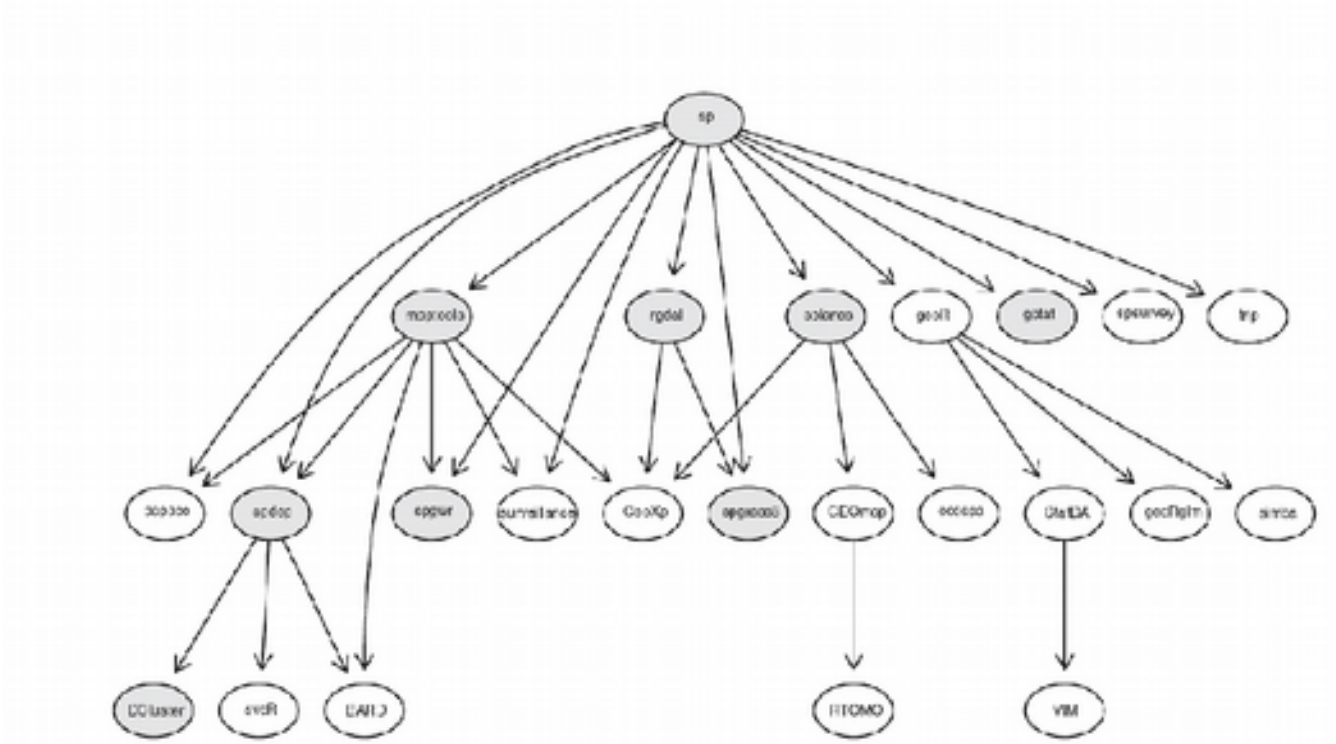


Figura 2. Struttura dei pacchetti di analisi spaziale disponibili in R all'aprile del 2008.

Data la complessità della struttura dei pacchetti presenti nell'archivio di R-cran, questi sono stati aggregati tematicamente in “task views” gestite a lotro volta dal pacchetto **ctv** (create task views). I pacchetti relativi all'analisi spaziale sono presenti nella task view spatial e possono essere installati digitando in successione i seguenti comandi.

```
install.packages("ctv")
library(ctv)
install.views("Spatial").
```

Classi e procedure in R

Sebbene lo scopo di questo capitolo non sia quello di fornire una introduzione all'uso del linguaggio R, può essere opportuno illustrare alcune delle sue peculiarità. L'impiego più semplice di R è quello di una comune calcolatrice scientifica, in grado di svolgere operazioni senza salvare alcun dato in memoria. Per esempio l'area di un cerchio di raggio 10 può essere calcolata come segue:

```
pi * 10^2
```

In realtà per effettuare questa operazione R utilizza due procedure (dette anche funzioni o operatori) la moltiplicazione `"*"` e l'elevamento a potenza `"^"` che agiscono su “argomenti”: la costante di sistema `"pi"`, `10` e `2`. Seguendo la “filosofia” più propria di R l'operazione potrebbe essere scritta come:

```
"*"(pi, "^"(10,2)).
```

Apparentemente con la pressione del tasto invio viene visualizzato il risultato. Propriamente, seguendo la filosofia della programmazione per oggetti di R, una procedura “print” viene applicata alla classe “risultato” usando gli argomenti di default. Sarebbe possibile infatti scrivere per avere il risultato con due cifre decimali:

```
print("*"(pi, "^"(10,2)), digit=2).
```

Se noi assegnamo il risultato alla variabile x usando la procedura assgnazione `<-` la procedura print può essere impiegata o implicitamente, digitando la variabile stessa o esplicitamente, con la specificazione dell'argomento:

```
x<-"*"(pi, "^"(10,2))
x
print(x, digit=2)
```

Possiamo ora vedere che la variabile x contiene un oggetto di una particolare classe (un numero) tramite la procedura class:

```
class(x)
```

Con la procedura getClass è possibile avere la descrizione completa (metadata) di un a classe:

```
getClass("numeric")
```

Otteniamo il seguente risultato,

```
Class “numeric” [package "methods"]
No Slots, prototype of class "numeric"
Extends: "vector"
Known Subclasses:
Class "integer", directly
Class "ordered", by class "factor", distance 3
```

nel quale viene specificato che: sulla classe agisce il pacchetto generico "method"; la classe non ha Slot, quindi non contiene definizioni provenienti da altre classi. Vengono poi elencate le classi che estndono quella in esame e le sottoclassi che derivano da questa con la relativa "distanza".

Una classe più complessa è l'archivio di dati data.frame, che costituisce un oggetto più complesso, formato da più slots: una matrice di dati alfanumerica(.data), da un indicatore sequenziale di numero di riga (row.names), da un vettore alfanumerico di identificatori di nomi di campo (names), dalla definizione della classe secondo lo standard S3 (.S3Class).

Per esaminare un esempio di data.frame è possibile utilizzare una qualsiasi base dati .dbf, im portata utilizzando il pacchetto **foreign**, che fornuisce i metodi per l'importazione di basi dati esterne:

```
library(foreign)

setwd('/home/unigis/DATI_E_DOCUMENTI_GIS/geostatistica/R_dati/')
```

Il comando precedentespecifica la cartella di lavoro

```
comuni.toscana<-read.dbf('istat/comuni_toscana.dbf')
```

Applicando le funzioni `class(), names(), str()` e `summary()` possiamo esaminare la struttura del data.frame.

```
class(comuni.toscana)
```

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

```
names(comuni.toscana)
```

```
 [1] "NOME"      "ISTAT"     "PIL"       "REDDITO"   "POPCENTR"  "PCENTR"
 [7] "SUP"       "POP"       "LAU"       "DIP"       "LAV"       "DISOCC"
[13] "PIND"      "DENS"      "PSTUD"     "PDIS"      "SIGLAPROV"
```

```
str(comuni.toscana)
```

```
'data.frame':   287 obs. of  17 variables:
 $ NOME     : Factor w/ 287 levels "ABBADIA SAN SALVATORE",..: 261 273 277 279 282 283 2 3 23 91 ...
 $ ISTAT    : int  9046030 9046031 9046032 9046033 9046034 9046035 9047001 9047002 9047003 9047004 ...
 $ PIL      : num  16327 9758 1933 757529 17335 ...
 $ REDDITO  : num  9639 9170 9940 12889 12267 ...
 $ POPCENTR : int  2720 1122 305 59655 1296 1098 379 13838 7062 1084 ...
 $ PCENTR   : int  84 102 84 97 74 83 55 95 89 65 ...
 $ SUP      : int  81 41 27 32 36 35 31 12 16 44 ...
 $ POP      : int  3243 1095 363 61267 1744 1319 688 14607 7944 1655 ...
```

```
 $ LAU      : int  96 21 16 5293 38 52 14 364 390 52 ...
 $ DIP      : int  520 168 72 16542 316 246 140 2592 1849 361 ...
 $ LAV      : int  1127 380 150 25722 698 511 329 6943 3594 689 ...
 $ DISOCC   : num  90 13 17 2533 50 ...
 $ PIND     : num  34.1 31.5 37 27.2 78.1 ...
 $ DENS     : num  40 26.7 13.4 1914.6 48.4 ...
 $ PSTUD    : num  19 17.3 24.2 35.6 20.3 ...
 $ PDIS     : num  7.99 3.42 11.33 9.85 7.16 ...
 $ SIGLAPROV: Factor w/ 10 levels "AR","FI","GR",..: 5 5 5 5 5 5 9 9 9 9 ...
 - attr(*, "data_types")= chr  "C" "N" "N" "N" ...
```

```
summary(comuni.toscana)
```

```
                  NOME         ISTAT              PIL
 ABBADIA SAN SALVATORE:  1   Min.   :9045001   Min.   :   1877
 ABETONE              :  1   1st Qu.:9047020   1st Qu.:  17335
 AGLIANA              :  1   Median :9050006   Median :  42046
 ALTOPASCIO           :  1   Mean   :9050526   Mean   : 163581
 ANGHIARI             :  1   3rd Qu.:9051038   3rd Qu.: 114590
 ARCIDOSSO            :  1   Max.   :9100007   Max.   :8255904
 (Other)              :281   NA's   :      2   NA's   :      2
    REDDITO         POPCENTR          PCENTR            SUP
 Min.   : 8087   Min.   :   260   Min.   : 36.00   Min.   :  6.00
 1st Qu.:10076   1st Qu.:  1561   1st Qu.: 70.00   1st Qu.: 34.00
 Median :10725   Median :  4207   Median : 80.00   Median : 62.00
 Mean   :10842   Mean   : 10829   Mean   : 78.15   Mean   : 79.37
 3rd Qu.:11471   3rd Qu.:  9182   3rd Qu.: 88.00   3rd Qu.: 99.00
 Max.   :15281   Max.   :351358   Max.   :108.00   Max.   :474.00
 NA's   :    2   NA's   :     2   NA's   :  2.00   NA's   :  2.00
      POP              LAU               DIP              LAV
 Min.   :   277   Min.   :   10.0   Min.   :   67   Min.   :   150
 1st Qu.:  2363   1st Qu.:   96.0   1st Qu.:  471   1st Qu.:   920
 Median :  5382   Median :  239.0   Median : 1149   Median :  2335
 Mean   : 12395   Mean   :  879.8   Mean   : 2954   Mean   :  5320
 3rd Qu.: 11480   3rd Qu.:  537.0   3rd Qu.: 2381   3rd Qu.:  4966
 Max.   :375984   Max.   :50572.0   Max.   :99805   Max.   :155732
 NA's   :     2   NA's   :    2.0   NA's   :    2   NA's   :     2
     DISOCC            PIND            DENS             PSTUD
 Min.   :   3.0   Min.   : 5.37   Min.   :   7.11   Min.   :14.84
 1st Qu.:  53.0   1st Qu.:27.93   1st Qu.:  35.19   1st Qu.:23.23
 Median : 127.0   Median :40.60   Median :  78.21   Median :25.92
 Mean   : 342.8   Mean   :43.16   Mean   : 205.86   Mean   :26.42
 3rd Qu.: 273.0   3rd Qu.:57.65   3rd Qu.: 240.64   3rd Qu.:28.89
 Max.   :8685.0   Max.   :91.67   Max.   :3686.12   Max.   :44.40
 NA's   :   2.0   NA's   : 2.00   NA's   :   2.00   NA's   : 2.00
      PDIS          SIGLAPROV
 Min.   : 1.69   FI     :43
 1st Qu.: 4.63   AR     :39
 Median : 5.73   PI     :39
 Mean   : 6.19   SI     :36
 3rd Qu.: 7.25   LU     :35
 Max.   :15.14   (Other):93
 NA's   : 2.00   NA's   : 2
```

Il data.frame importato contiene dati socioeconomici dei comuni della Toscana: `PIL`, reddito pro capite (`REDDITO`), la popolazione che vive in centrio e nuclei (`POPCENTR`), popolaione totale (`POP`), numero di laureati (`LAU`), lavoratori dipendenti (`DIP`), lavoratori totali (`LAV`), e disoccupati (`DIS`); nonché indicatori derivati: densità popolazione (`DENS`), percentuale popolazioone lautreata (`PLAV`), che vive in centri e nuclei (`PCENTR`), che lavoraa nell'indiustria manifatturiera (`PIND`), o disoccupata (`PDIS`). di

Come si può notare dai risultati ottenuti, i campi alfanumerici del nome del comune (`NOME`) e dlla sigla della provincia (`SIGLAPROV`) appartengono alla classe “factor”.

Oggetti spaziali

La classe fondamentale degli oggetti spaziali in R è “Spatial”, contenuta nel pacchetto **sp** e contiene solo 2 slot. Il primo slot è una matrice quadrata di coordinate geografiche minime e massime, chiamata `bbox`, con due nomi di colonna (`c('min','max')`), e due righe: le coordinate x e y. Il secondo slot invece è una classe speciale detta CRS (Coordinate Reference System) a sua volta definito da uno Slot costrituito da unjh astringa di caratteri che definiscono il sistema di ccordinate secondo uno standard internazionale detto PROJ.4.
La classe spatial è suddivisa in diverse subclasses definite dai diversi metodi di rappresentazione degli oggetti geografici nel formato vettoriale e raster:

```
library(sp)

getClass("Spatial")
```

```
Class “Spatial” [package "sp"]

Slots:
Name:         bbox proj4string
Class:      matrix         CRS

Known Subclasses:
Class "SpatialPoints", directly
Class "SpatialLines", directly
Class "SpatialPolygons", directly
Class "SpatialPointsDataFrame", by class "SpatialPoints", distance 2
Class "SpatialPixels", by class "SpatialPoints", distance 2
Class "SpatialLinesDataFrame", by class "SpatialLines", distance 2
Class "SpatialGrid", by class "SpatialPoints", distance 3
Class "SpatialPixelsDataFrame", by class "SpatialPoints", distance 3
Class "SpatialGridDataFrame", by class "SpatialPoints", distance 4
Class "SpatialPolygonsDataFrame", by class "SpatialPolygons", distance 2
```

Gli oggetti SpatialGrid e SpatialPixel

Gli oggetti geografici riferiti alla rappresentazione raster sono costituiti dagli Slots necessari a definire questa topologia.

```
getClass("SpatialGrid")
```

```
Class “SpatialGrid” [package "sp"]

Slots:
Name:          grid   grid.index       coords         bbox  proj4string
Class: GridTopology      integer       matrix       matrix          CRS

Extends:
```

```
Class "SpatialPixels", directly
Class "SpatialPoints", by class "SpatialPixels", distance 2
Class "Spatial", by class "SpatialPixels", distance 3

Known Subclasses: "SpatialGridDataFrame"
```

L'elemento caratterizzante è la classe `GridTopology,` che definisce le variabili necessarie a descrivere la cella della griglia raster. Una volta nota la dimensione della cella tramite la classe GridTopology, il sistema di coordinate tramite la classe `CRS` e la finestra cartografica tramite lo Slot `bbox` è possibile ricostruire la topologia della mappa ratster secondo quanto illustrato al capitolo 1.
Se alla griglia viene associato un data.frame contenente informazioni su almeno un attributo della cella otteniamo l'oggetto derivato `SpatialGridDataFrame,` che costituisce uno dei geodati fomdamentali per l'analisi statistiche tramite R. Di seguito è analizzata la struttura di un oggetto `SpatialGridDataFrame` contenente dati sulla piovosità totale annuale media in Toscana.

```
str(pioggia)
Formal class 'SpatialGridDataFrame' [package "sp"] with 6 slots
```

(di seguito vengono descritti gli Slot)
(`data`: data.frame relativo all'attributo; il nome dell'unico campo dati è `band1` ed è formato da 979524 valori di piovosità per ciascuna cella del dato raster)

```
  ..@ data       :'data.frame': 979524 obs. of  1 variable:
  .. ..$ band1: num [1:979524] 2266 2230 2250 2264 2279 ...
```

(`grid` è la griglia, a sua volta formata da tre slots: `cellcentre.offset` coordinate del centro della cella in basso a sinistra; `cellsize` dimensione della cella 250 metri; `cells.dim` dimensione del dato raster 966 colonne e 1014 righe)

```
  ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
  .. .. ..@ cellcentre.offset: Named num [1:2] 537625 4681375
  .. .. .. ..- attr(*, "names")= chr [1:2] "x" "y"
  .. .. ..@ cellsize         : num [1:2] 250 250
  .. .. ..@ cells.dim        : int [1:2] 966 1014
```

(`grid.index` è uno slot non utilizzato in questo specifico oggetto)

```
..@ grid.index : int(0)
```

(`coords` coordinate degli esstremi della `bbox`)

```
  ..@ coords     : num [1:2, 1:2] 537625 778875 4681375 4934625
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "x" "y"
```

(`bbox`)

```
  ..@ bbox       : num [1:2, 1:2] 537500 4681250 779000 4934750
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x" "y"
  .. .. ..$ : chr [1:2] "min" "max"
```

(stringa della classe `CRS` relativa alla proiezione UTM WGS84 zona 32)

```
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
       .. .. ..@ projargs: chr " +proj=utm +zone=32 +ellps=WGS84 +datum=WGS84
     +units=m +no_defs +towgs84=0,0,0"

summary(pioggia)

Object of class SpatialGridDataFrame
Coordinates:
      min     max
x  537500  779000
y 4681250 4934750
Is projected: TRUE
proj4string :
[+proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
+towgs84=0,0,0]
Number of points: 2
Grid attributes:
  cellcentre.offset cellsize cells.dim
x            537625      250       966
y           4681375      250      1014
Data attributes:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
     0.0    932.4   1082.0   1219.0   1400.0   4118.0 300475.0
```

L'oggetto `SpatialPixelObject` differisce da `SpatialGridObject` per il metodo di registrazione dei dati, dal momento che nel primo i valori mancanti, per esempio la piovosità fuori dalla regione, non sono registrati singolar,ente nel disco rigido. Questo comporta un minor uso della memoria di massa, ma una maggiore lentezza nel caricamento e nel salvataggio della base dati.

Gli oggetti SpatialPointDataFrame

La classe SpatialPointDataFrame è una delle classi dedicate ai geodati vettoriali. La struttura è simile alla precedenbte, come illustrato in questo esempio relativo punti di rilevamento pluviometrico in Toscana.

```
str(pluvio)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
```

(il data.frame è relatyivo a 362 punti di rilevamento pluviometrico con 2 variabili: l'identificativo della stazione (`cat`) e la piovosità annua (`z`))

```
  ..@ data       :'data.frame': 362 obs. of  2 variables:
  .. ..$ cat: num [1:362] 3 8 9 12 15 16 17 18 19 20 ...
  .. ..$ z  : num [1:362] 862 2623 1016 1114 1048 ...
  ..@ coords.nrs : num(0)
```

(la principale differenza è nello slot `coords` che contiene due vettori relativi alle coordinate dei punti)

```
  ..@ coords     : num [1:362, 1:2] 741854 655858 639769 636762 731322 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
  ..@ bbox       : num [1:2, 1:2] 563650 4692630 767746 4923285
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
```

```
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr " +proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m
+no_defs +towgs84=0,0,0"
```

Con un comando implicito di print è in questo caso possibile visualizzare i dati contenuti nell'oggetto.

```
pluvio
```

```
        coordinates cat         z
1   (741854, 4768110)   3  862.2000
2   (655858, 4875730)   8 2622.8000
....#
```

Gli oggetti SpatialPolygonDataFrame

E' senz'altro la categria di oggetti più macchinosa, in quanto lo slot polygons che definisce il mosaico di poligoni contiene ta sua volta tanti slot quanti sono i singoli poligoni, ciascuno definito da una linea chiusa, con le relative coordinate dei vertici, dalle coordinate del centroide, da eventuali fori del poligono, dall'identificativo e dall'area.

```
str(comuni.toscana)
```

```
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 287 obs. of  17 variables:
  .. ..$ NOME     : Factor w/ 287 levels "ABBADIA SAN SALVATORE",..: 261 273 277 279 282 283 2 3 23
91 ...
  .. ..$ ISTAT    : num [1:287] 9046030 9046031 9046032 9046033 9046034 ...
  .. ..$ PIL      : num [1:287] 16327 9758 1933 757529 17335 ...
  .. ..$ REDDITO  : num [1:287] 9639 9170 9940 12889 12267 ...
  .. ..$ POPCENTR : num [1:287] 2720 1122 305 59655 1296 ...
  .. ..$ PCENTR   : int [1:287] 84 102 84 97 74 83 55 95 89 65 ...
  .. ..$ SUP      : int [1:287] 81 41 27 32 36 35 31 12 16 44 ...
  .. ..$ POP      : int [1:287] 3243 1095 363 61267 1744 1319 688 14607 7944 1655 ...
  .. ..$ LAU      : int [1:287] 96 21 16 5293 38 52 14 364 390 52 ...
  .. ..$ DIP      : int [1:287] 520 168 72 16542 316 246 140 2592 1849 361 ...
  .. ..$ LAV      : int [1:287] 1127 380 150 25722 698 511 329 6943 3594 689 ...
  .. ..$ DISOCC   : num [1:287] 90 13 17 2533 50 ...
  .. ..$ PIND     : num [1:287] 34.1 31.5 37 27.2 78.1 ...
  .. ..$ DENS     : num [1:287] 40 26.7 13.4 1914.6 48.4 ...
  .. ..$ PSTUD    : num [1:287] 19 17.3 24.2 35.6 20.3 ...
  .. ..$ PDIS     : num [1:287] 7.99 3.42 11.33 9.85 7.16 ...
  .. ..$ SIGLAPROV: Factor w/ 10 levels "AR","FI","GR",..: 5 5 5 5 5 5 9 9 9 9 ...
```

(slot contenente tutti i poligoni del mosaico)

```
 ..@ polygons   :List of 287
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 604274 4874301
  .. .. .. .. .. .. ..@ area   : num 80110060
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:106, 1:2] 599969 600984 601716 602022
602520 ...
  .. .. .. ..@ plotOrder: int 1
```

```
  .. .. .. ..@ labpt    : num [1:2] 604274 4874301
  .. .. .. ..@ ID       : chr "0"
  .. .. .. ..@ area     : num 80110060
```

(Seguono altri 286 slot relativi ai comuni della Toscana)
...
(Slot che contiene l'ordine di disegno dei diversi poligoni)

```
..@ plotOrder  : int [1:287] 263 179 266 194 273 267 275 44 165 177 ...
```

(Slot tipici della classe Spatial)

```
  ..@ bbox       : num [1:2, 1:2] 554856 4678249 771686 4924947
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "r1" "r2"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr " +proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m
+no_defs +towgs84=0,0,0"
```

La gestione degli SpatialDataFrame

L'importazione dei dati

Per i formati ESRI sono previsti specifici pacchetti, uno dei più utilizzati è la Geospatial Data Abstraction Library, **rgdal,** (GDAL, pronunciato “Goodal”) derivantre da un progetto coordinato da Frank Warmerdan. La sintassi per l'importazione è la seguente:

```
library(rgdal)
```

(importazione shapefile: dns è la diurectory mentre layer è il nome del file senza estensione shp)

```
comuni<-readOGR(dsn='istat', layer='ISTAT')
```

(esportazione shapefile)

```
writeOGR(comuni, dsn='istat',layer='comuni_out' , driver="ESRI Shapefile")
```

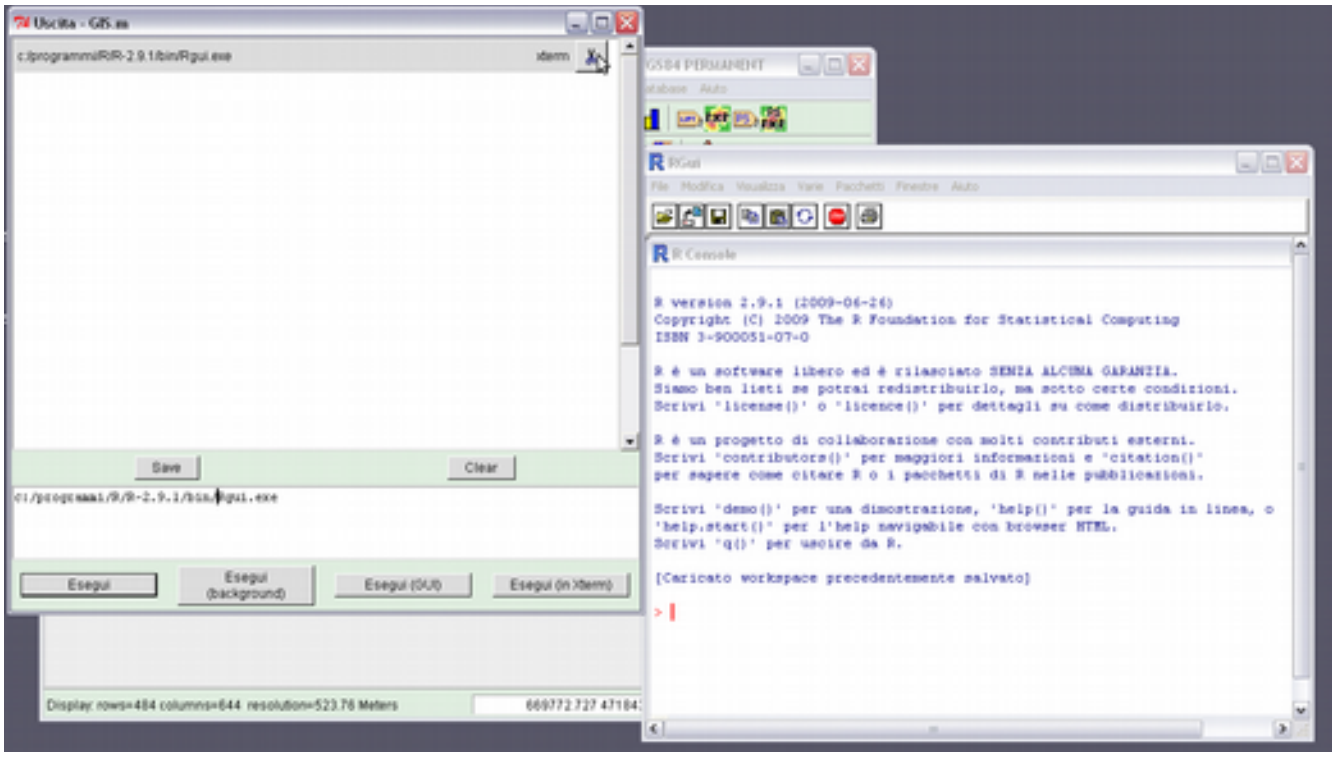
(importazione GEOTIF)

```
pioggia<-readGDAL('pioggia.tif')
```

(esportazione GEOTIF)

```
writeGDAL(pioggia, 'pioggia_out.tif', drivername='GTiff')
```

Nel caso di GRASS il pacchetto dedicato è **spgrass6** coordinato da Roger Bivand. In questo caso, per poter facilmente indirizzare le funzioni di importazione verso il mapset desiderato è opportuno lanciare R da shell di GRASS nel caso di linux o da finestra Xterm di GRASS Tlck nel caso di Windows, come illustrato in figura:



La sintassi è illustrata nei seguenti esempi.

```
library(spgrass6)
```

(importazione vettoriale)

```
acqua<-readVECT6('qualita_acqua')
```

(esportazione)

```
writeVECT6(acqua, 'newacqua')
```

(importazione raster)

```
dem<-readRAST6('dem_tosc')
```

(esportazione)

```
writeRAST6(dem, 'new_dem')
```

Nel caso di dati raster una particolare attenzione deve essere posta al nome che viene automaticamente assegnato all'attributo del layer importato. Se queste sono state importate tramite pacchetto **rgdal** i dati sono contenuti nel data.frame band1, mentre se provengono da GRASS il nome della variabile è uguale al nome originale della base dati raster del mapset. Tale nome può essere facilmente cambiato attarverso una assegnazione del tipo:

```
names(pioggia)<-'piovosita'
```

L'interrogazione dei dati spaziali

```
livorno<-comuni[comuni$SIGLAPROV=="LI",]
reddito.basso<-comuni[comuni$REDDITO<10000,]
plot(reddito.basso)
crisi<-comuni[(comuni$REDDITO<10000 & comuni$PDIS>9),]
plot(crisi)
sviluppo<-comuni[(comuni$REDDITO>10500 | comuni$PSTUD>40),]
plot(sviluppo)
```

Elaborazione degli attributi di dati spaziali

```
comuni$denslau<-comuni$LAU/comuni$SUP
comuni$logdenslau<-log(comuni$denslau)
spplot(comuni, "logdenslau")
```

Overlay di dati spaziali

Overlay raster

```
names(pioggia)<-"pioggiatot"
names(dem)<-"quota"
geog<-cbind(dem,pioggia)
```

Overlay raster/vector.

```
idx<-overlay(geog,comuni)
geog$prov<-comuni$SIGLAPROV[idx]
```

Dissolvere dati spaziali

Innazitutto creiamo la geometria dei poligoni uniti:

```
province<-unionSpatialPolygons(comuni, comuni$SIGLAPROV)
```

Il risultato è un oggetto di classe SpatialPolygon senza DataFrame. L'ID dei poligonio è la sigla della provincia. Per unire i dati creiamo innanzitutto un vettore di dati aggregati:

```
province.pop<-aggregate(comuni$POP, by=list(comuni$SIGLAPROV), sum)
```

Assegnamo alle righe del vettore il nome della provincia corrispondente:

```
> row.names(province.pop)<-province.pop$Group.1
```

Creiamo l'oggetto SpatialPolygonDataFrame come combinazione dell'oggetto SpatialPolygon e dell'oggetto DataFrame:

```
province2<-SpatialPolygonsDataFrame(province, province.pop)
```

Un esempio un po' più articolato. Estraiamo un subset di variabili fra di loro sommabili:

```
> comuni.sub<-data.frame(comuni$PIL,comuni$POP,comuni$LAU,comuni$DIP,comuni$LAV
```

aggreghiamo il dataset per province:

```
province.data<-aggregate(comuni.sub, by=list(comuni$SIGLAPROV), sum)#
```

costruiamo il dataframe:

```
row.names(province.data)<-province.data$Group.1
province3<-SpatialPolygonsDataFrame(province, province.data)
```

calcoliamo gli indici socioeconomici:

```
province3$perc.dip<-province3$comuni.DIP/province3$comuni.LAV
spplot(province3, "perc.dip")
province3$pil.pro.cap<-province3$comuni.PIL/province3$comuni.POP
comuni.agricoltura<-> spplot(province3, "pil.pro.cap")
province3$pil.lav<-province3$comuni.PIL/province3$comuni.LAV
spplot(province3, "pil.lav")#
```

Join di dati geografici

La procedura si articola su più fasi.
carichiamo il pacchetto **maptools**,

```
library(maptools)
```

Importiamo la base dati "agricoltura" in formato dbf (sarebbe necessario il pacchetto **foreign**, ma questo viene caricato da **maptools**), contenente dati sulle superfici agricole nei comuni toscani,

```
agricoltura<-read.dbf('agricoltura.dbf')
```

creiamo una corrispondenza fra gli identificativi di riga della base dati geografica "comuni" e la base dati alfanumerica "agricoltura"

```
comuni1<-spChFIDs(comuni, as.character(comuni$NOME))
o<-match(comuni1$NOME,agricoltura$NOME)
agricoltura1<-agricoltura[o,]
row.names(agricoltura1)<-comuni1$NOME
```

aggiungiamo le colonne al data.frame della classe SpatialPolygon.

```
comuni2<-spCbind(comuni1,agricoltura1)
```

Salvare

Per salvare nel formato binario di R la sintassi generica è la seguente:

```
save(x,y,..., file='filexy')
```

Successivamente i file x, y , ecc. saranno disponibili in qualsiasi sessione tramite il comando:

```
load('filexy')
```

Per il caso precedente:

```
save(province3, file='province')
```

La rappresentazione grafica degli attributi

R è dotato di numerose modalità di rappresentazione grafica dei dati, quelle di maggior interesse per i casi in esame sono la modalità tradizionale, i grafici Trellis e la modalità specifica di rappresentazione dedicata ai dati spaziali.
La modalità tradizionale si applica ai data.frame degli oggetti della classe Spatial senza considerare esplicitamente la loro posizione geografica ma solo i valori numerici. La funzione più semplice ' l'istogramma, che consente di rappresentare un istogramma delle frequenze, come quello della distribuzione del reddito medio pro capite dei comuni Toscani:

```
library(rgdal)
ibrary(sp)
comuni<-readOGR(dsn='istat', layer="ISTAT")
hist(comuni$REDDITO)
```

In R la modalità grafica è regolata sia dalle opzioni sia dal ripetersi incrementale di operazioni sulla stessa finestra di output come nell'esempio in cui diverse mofalità di rappresentazione di un istogramma sono riportate nella stessa finestra dividendola in quattro parti:

```
par(mfrow=c(2,2))
hist(comuni$REDDITO)
hist(comuni$REDDITO, breaks=20)
hist(comuni$REDDITO,
   breaks=c(8000,9000,10000,10100,10200,10500,11000,15000,20000) )
hist(comuni$REDDITO, col="grey")
par(mfrow=c(1,1))
```

Un'altra interessante modalità di rappresentazione dei dati è il box and wischer plot:

```
mpar(mfrow=c(1,2))
boxplot(comuni$PSTUD)
boxplot(comuni$PDIS)
mpar(mfrow=c(1,1))
```

Una delle astrazioni fondamentali impiegate in R è la “formula”, cioè la relazione fra due variabili, la prima considerata come variabile dipendente e la seconda indipendente. Separate da un carattere speciale, la tilde (~). Un esempio di classe formula si ha proprio nelle rappresentazioni grafiche:

```
plot(REDDITO~PSTUD, data=comuni)
```

La formula può anche comprendere variabili “factor”:

```
plot(REDDITO~SIGLAPROV, data=comuni)
```

La rappresentaziione grafica ei dati può essere effettuata anche su basi dati raster.

```
par(mfrow=c(2,1))
hist(pioggia$band1)
boxplot(pioggia$band1, horizontal=T)
```

E' possibile anche facilmente analizzare la diversa relazione fra variabili su dato raster:

```
boxplot(geog$pioggia~geog$prov)
plot(geog$pioggiatot~geog$prov, cex=0.1)
```

I grafici di Trellis

```
library(lattice)
histogram(~comuni$REDDITO | comuni$SIGLAPROV, layout=c(5,2))
xyplot(comuni$REDDITO~comuni$PSTUD | comuni$SIGLAPROV, layout=c(5,2))
densityplot(~comuni$REDDITO | comuni$SIGLAPROV, layout=c(2,5)
```

Analisi statistica dei dati circolari

```
library(circular)
library(rgdal)
esposizione<-readGDAL('esposizione.tif')
uso2<-readGDAL('uso2.tif')
uso3<-readGDAL('uso3.tif')
names(esposizione)<-'gradi'
names(uso2)<-'corine'
names(uso3)<-'corine3'
territorio<-cbind(esposizione,uso2,uso3)
#
# Il comando cbind crea un dataframe formato dalla combinazione dei dati delle tre
  griglie
#
rose.diag((circular(territorio$gradi[territorio$corine3==111],
  units="degree",template="geographics")), bins=12, prop=2, main="Urbanizzato
  continuo")
esp.vigneto<-circular(territorio$gradi[territorio$corine3==221], units="degree",
  template="geographics")
summary(esp.vigneto)

         n         Mean          Rho
5917.0000000  245.8716098    0.1196753

territorio$gradi<-circular(territorio$gradi, units="degree", template="geographic")
summary(territorio)

Object of class SpatialGridDataFrame
```

```
Coordinates:
      min     max
x  537500  779000
y 4681250 4934750
Is projected: TRUE
proj4string :
[+proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
+towgs84=0,0,0]
Number of points: 2
Grid attributes:
  cellcentre.offset cellsize cells.dim
x            537625      250       966
y           4681375      250      1014
Data attributes:
  gradi                corine             corine3
 n   : 9.795e+05   Min.   :    11.00   Min.   :  0.00
 Mean:-2.770e+00   1st Qu.:    21.00   1st Qu.:  0.00
 Rho : 6.078e-01   Median :    31.00   Median :  0.00
                   Mean   :    26.46   Mean   : 99.87
                   3rd Qu.:    31.00   3rd Qu.:231.00
                   Max.   :    52.00   Max.   :521.00
                   NA's   :611798.00

aggregate(territorio$gradi, by=list(territorio$corine), mean.circular, na.rm=TRUE)
aggregate(territorio$gradi, by=list(territorio$corine), rho.circular, na.rm=TRUE)
```

La geostatistica

La geostatistica ambientale fornisce le basi per capire ed utilizzare le varie tecniche per la stima del valore di una variabile spaziale nelle aree dove tale variabile non è stata misurata. Tipticamente, un fenemeno territoriale diffuso non conosciuto viene 'campionato' tramite una serie di punti distribuiti in modo più o meno regolare. Esempi classici sono:

- le stazioni metereologiche che forniscono misure puntuali (cioè riferite ad un singolo punto) di dati (temperatura, pioggia, vento, ecc.) la cui variazione è continua nello spazio;
- le stazioni di misura della qualità dell'aria;
- i rilievi quali/quantitativi delle acque sotterranee tramite campioni prelevati da pozzi;
- il rilievo delle caratteristiche dei suoli tramite profili pedologici;
- gli inventari forestali realizzati tramite aree di saggio;
- ecc.

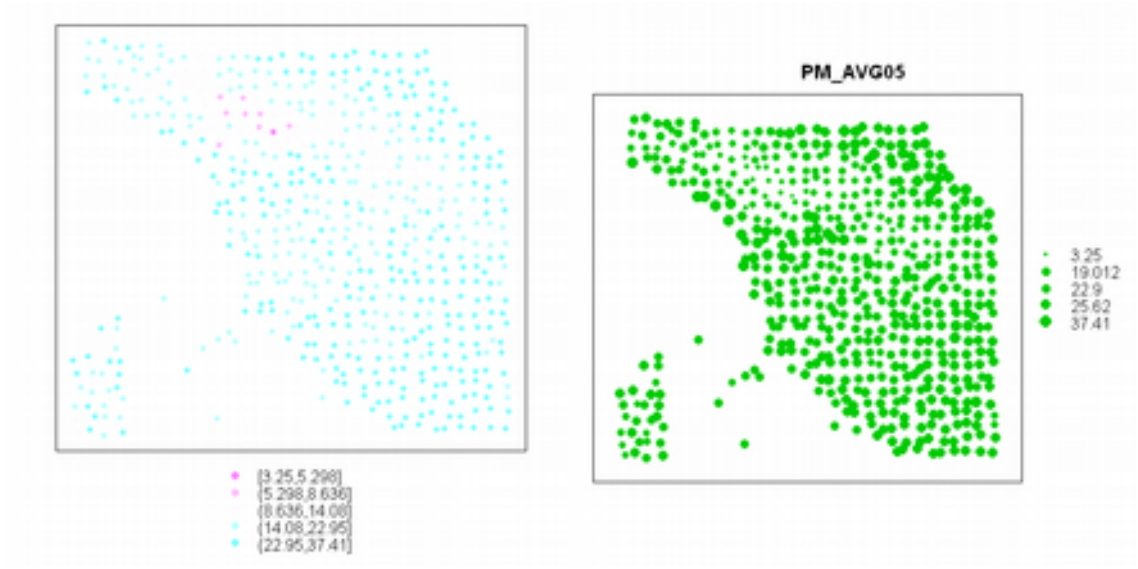
Compito delle tecniche di geostatistica è quello di ricostruire la miglio stima possibile della 'superficie spaziae tridimensionale' *Z* rappresentativa del fenomeno, dove *Z* può essere una carta climatica, una carta della qualità dell'aria, una carta della percentuale di argilla del terreno, una carta della qualità delle acque di falda, ecc.. I punti dovrebbero essere in numero, posizione e densità tali da formare un campionamento 'statisticamente rappresentativo'; tale principio però nella pratica è spesso disatteso per ragioni sia pratiche che di economicità del rilievo. Se necessario le tecniche di geostatistica nella stima della superficie *Z* possono aiutarsi con altre variabili spaziali che si suppone siano correlate al fenemeno oggetto di studio (p.e. Modelli digitali del terreno, mappe geologiche, carte della vegetazione, ecc.)

La maggior parte delle analisi più tipicamente geostatistiche saranno realizzate tramite la libreria **`gstat`** che copre il più ampio campo di tecniche geostatistiche. Molte funzioni sono però presenti anche in altre librerie e altri pacchetti necessari saranno di volta in volta specificati.

Statistica esplorativa

La prima fase di una analisi geostatistica è la cosiddetta esplorazione spaziale dei dati. Questa ha lo scopo di visualizzare 'geograficamente' la variazione del fenomeno e quindi di fare alcune prime ipotesi sulla sua distribuzione spaziale e sulle altre variabili coinvolte.
Negli esempi seguenti utilizzeremo i rilievi fatti dall'Agenzia Europea per l'Ambiente relativi alla qualità dell'aria in toscana, e precisamente il dato relativo alla media della concentrazione del PM10 nel 2005.

```
setwd('/home/unigis/DATI_E_DOCUMENTI_GIS/geostatistica/R_dati/')
library(spdep)
library(rgdal)
qual.aria<-readOGR(dsn='eea_aria', layer='EEA_aria')
spplot(qual.aria, "PM_AVG05", do.log=T)
bubble(qual.aria, "PM_AVG05", do.log=T, maxsize=1.5)
```



L'analisi esplorativa ci consente di fare una prima ipotesi: la concentrazione media del PM10 varia negativamente con l'aumentare della quota.

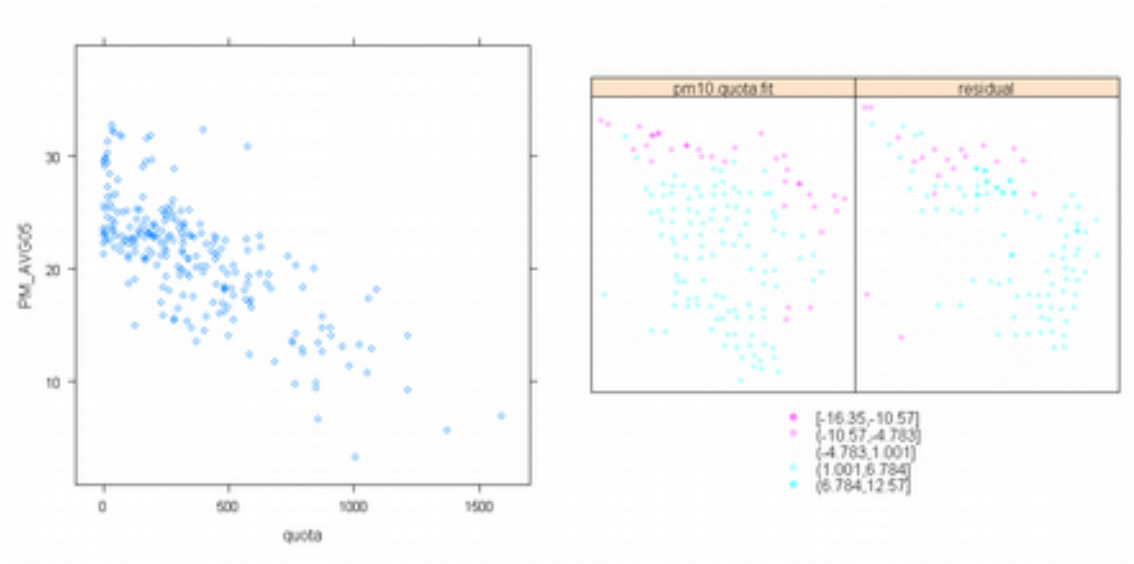
```
dem<-readGDAL('dem.tif')
names(dem)<-'quota'

pm10.quota<-overlay(dem,qual.aria)
qual.aria$quota<-pm10.quota$quota

library(lattice)
xyplot(PM_AVG05~quota, as.data.frame(qual.aria))
```

Il comando seguente è necessario in quanto i due dataset non si sovrappongono esattamente come estensione geografica: il dato sulla qualità dell'aria copre una superficie maggiore della regione mentre il DEM è limitato dal confine amministrativo. Dove il DEM non è disponibile l'overlay restituisce un valore NA che blocca la maggior parte delle elaborazioni.

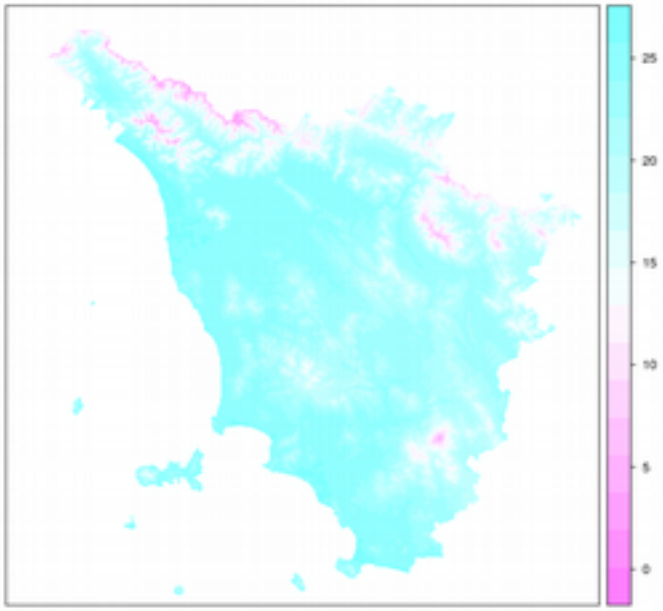
```
toscana.aria<-subset(qual.aria,qual.aria$quota!="NA")

pm10.quota<-lm(PM_AVG05~quota, toscana.aria)

summary(pm10.quota)

Call:
lm(formula = PM_AVG05 ~ quota, data = toscana.aria)

Residuals:
    Min      1Q  Median      3Q     Max
-9.2777 -2.0787 -0.1971  2.0559 12.5681

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 25.7718551  0.3480465   74.05   <2e-16 ***
quota       -0.0131652  0.0007609  -17.30   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 3.456 on 234 degrees of freedom
Multiple R-squared: 0.5613,     Adjusted R-squared: 0.5594
F-statistic: 299.4 on 1 and 234 DF,  p-value: < 2.2e-16
```

```
toscana.aria$pm10.quota.fit<-predict(pm10.quota,toscana.aria)-
mean(predict(pm10.quota,toscana.aria))
toscana.aria$residual<-residuals(pm10.quota)
spplot(toscana.aria, c("pm10.quota.fit","residual"))
```




```
dem$PM10<-dem$quota*-0.0131652+25.7718551
writeGDAL(dem['PM10'],'PM10_pred_quota.tif',drivername='GTiff')
spplot(dem,'PM10')
```



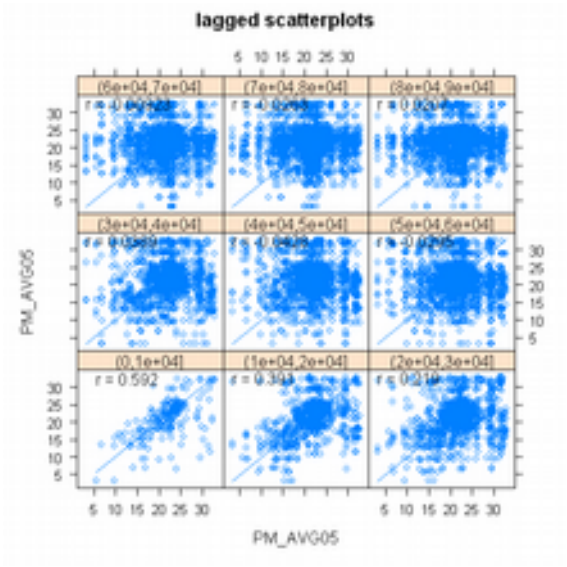
Il principale difetto di questa stima è che la distribuzione spoaziale el fenemeno è spiegata solamente dalla quota e quindi è una trasformata (in questo caso addirittura lineare) dei quest'ultima. La stima è, anche ad un esame sommario, ovviamente non soddisfacente. In teoria sarebbe necessario conoscere ed avere un dato spaziale di tutte le variabili territoriali coinvolte (p.e. direzione prevalente dei venti, luoghi di origine degli inquinanti, ostacoli naturali e artificiali alla loro diffusione, ecc.). Spesso in questa ed altre stime sono note solo poche di queste variabili. In questo caso è gioco-forza adottare l'ipotesi base della geostatistica, cioè che osservazioni fra loro vicine risentano dell'effetto di valori simili di queste variabili più che osservazioni fra loro lontane. Per verificare tale assunto lo strumento è da impiegare è il cosiddetto variogramma.

L'analisi esploratoria del variogramma

Un metodo semplice, empirico, ma efficace per valutare in modo visuale (in altre parole 'esplorativo') se punti fra loro vicini sono più simili rispetto a punti fra loro lontani - o come si dice in termini statistici sono fra loro 'correlati spazialmente' - è la analisi esploratoria del variogramma.

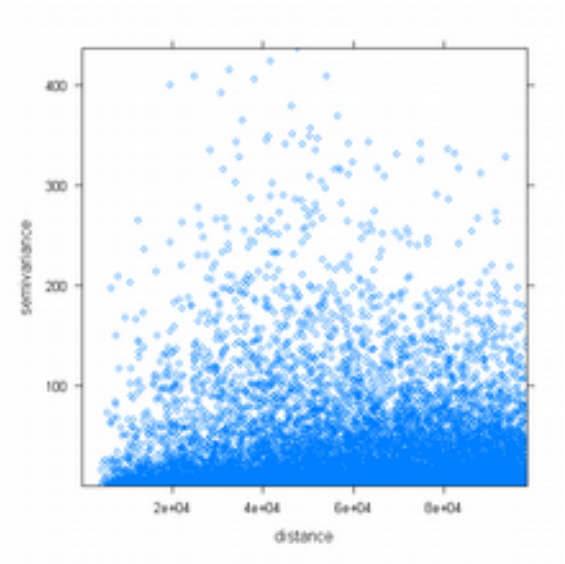
La procedura seguente consente di tracciare dei diagrammi a nebulosa di punti prendendo in considerazione tutte le coppie che si trovano, per ciascun grafico, entro un cero intervallo di distanza. In questo caso vengono prese in considerazione 9 classi di distanza che vanno dall'intervallo 0 – 10km (grafico in basso a sinistra) fino al range 80-90 km (grafico in alto a destra). L'analisi ci dice che la correlazione fra valori di PM10 di coppie di punti diminuisce al crescere della loro distanza, quindi esiste una correlazione spaziale.

```
library(gstat)
hscat(PM_AVG05~1, toscana.aria, (0:9)*10000)
```



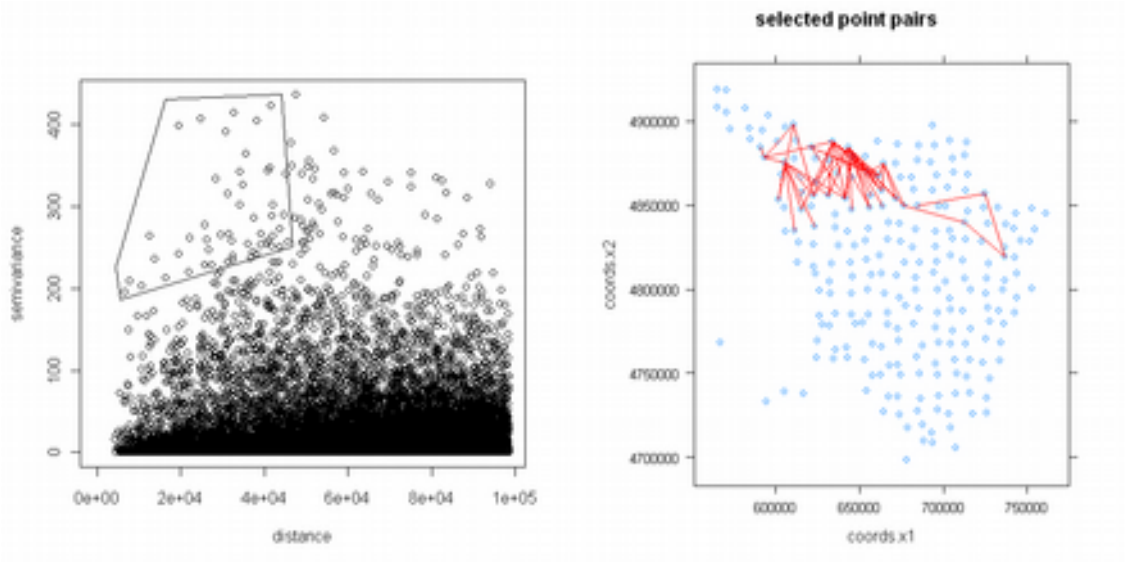
Una seconda tecnica di analisi visuale esplorativa consiste nel plottare per ogni possibile coppia di punti x e y il valore di scarto quadratico $(PM10_x - PM10_y)^2$ in funzione della distanza tra x e y.

```
plot(variogram(PM_AVG05~1, toscana.aria, cloud=TRUE))
```



La tecnica seguente consente di selezionare aree di interesse del diagramma per esplorare punti di variazione spaziale notevoli. Il caso seguente, selezionando nella figura a destra punti vicini con alta variazione, evidenzia nella mappa a sinistra le zone con elevati gradienti del fenomeno.

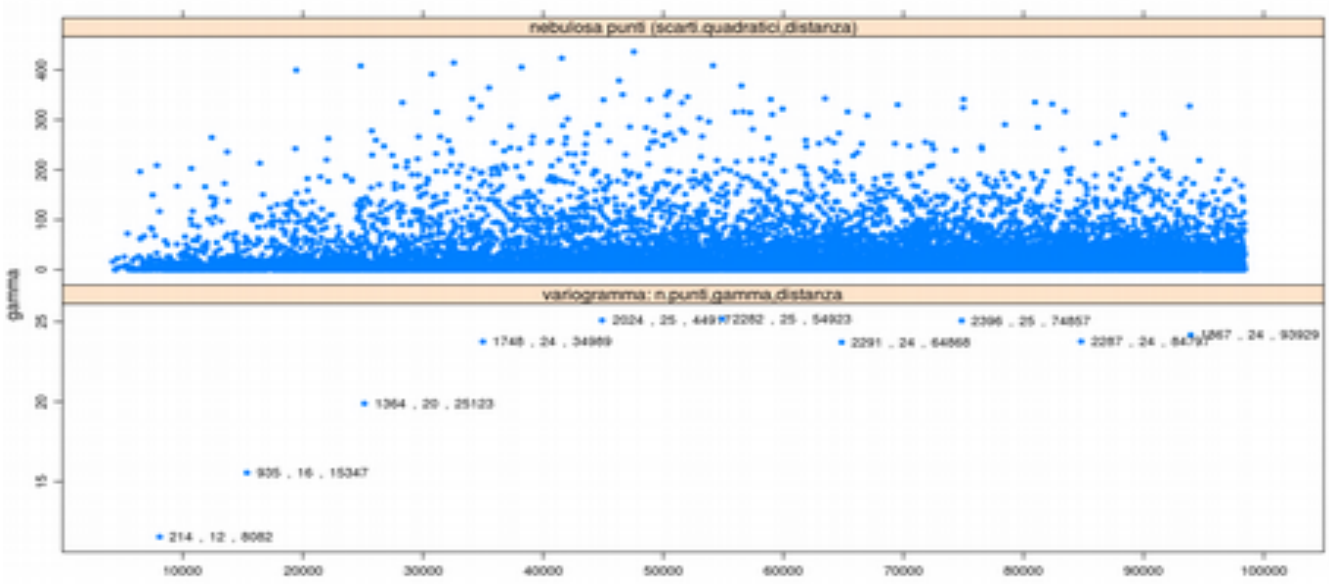
```
plot(variogram(PM_AVG05~1, toscana.aria, cloud=TRUE))
sel<-plot(variogram(PM_AVG05~1, toscana.aria, cloud=TRUE), digitize=TRUE)
# [1] "mouse-left digitizes, mouse-right closes polygon"
plot(sel, toscana.aria)
```



I risultati mostrano che i punti collegati non sono distribuiti casualmente ma collegano zone metropolitane con aree collinari e montuose del preappennino e dell'appennino. Dall'analisi effettuata sembra possibile quindi rigettare l'ipotesi di assenza di correlazione spaziale.

Il (semi)variogramma

Il variogramma (o semivariogramma) è lo strumento più utilizzato per misurare la continuità e variazione spaziale di variabili regionalizzate. Il variogramma delinea la dipendenza spaziale ossia la probabilità che osservazioni più vicine nello spazio siano più "somiglianti" di quelle poste a maggiore distanza. Si tratta di una funzione che mette in relazione la semivarianza (γ) con la distanza (h) che separa coppie di dati sperimentali. La figura seguente mostra la relazione fra (semi)varianza e nebulosa dei punti precedentemente calcolata[2].



Ciascun punto nel grafico inferiore rappresenta il campionamento di tutte le osservazioni entro intervalli di distanza pari a 10 km; accanto a ciascun punto è riportato il numero delle osservazioni entro ciascuna classe, il valore della (semi)varianza[3] - un indice correlato con la similitudine nel valore di PM10 - e la distanza media fra le osservazioni. Per esempio il punto in basso a destra deriva

[2] La figura è stata ottenuta tramite il seguente script (un po' complesso):

```
cld <- variogram(PM_AVG05 ~ 1, toscana.aria, cloud = TRUE)
svgm <- variogram(PM_AVG05 ~ 1, toscana.aria, width=10000)
d <- data.frame(gamma = c(cld$gamma, svgm$gamma),
dist = c(cld$dist, svgm$dist),
id = c(rep("nebulosa punti (scarti.quadratici,distanza)", nrow(cld)), rep("variogramma:
n.punti,gamma,distanza", nrow(svgm)))
)
xyplot(gamma ~ dist | id, d,
scales = list(y = list(relation = "free", ylim = list(NULL, c(-.005,0.7)))),
layout = c(1, 2), as.table = TRUE,
panel = function(x,y, ...) {
    if (panel.number() == 2)
          ltext(x+10, y, paste("  ",svgm$np,' , ',round(svgm$gamma,digits=0),' ,
',round(svgm$dist,digits=0)), adj = c(0,0.5),cex=0.75) #$
    panel.xyplot(x,y,...)
},
xlim = c(0, 105000),
cex = .6, pch = 19 )
```

[3] La formula esatta è la seguente:

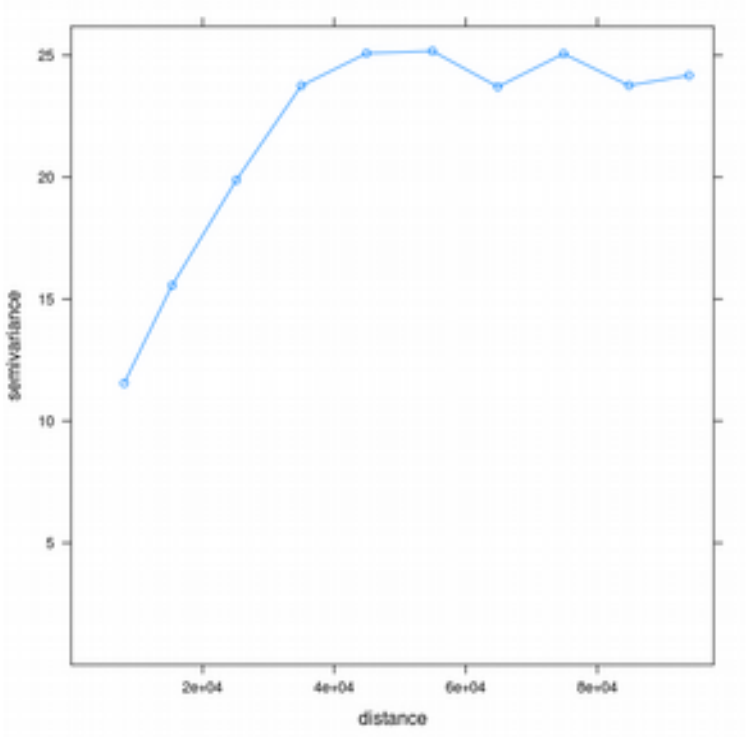
da 214 osservazioni entro i 10 km (grafico superiore), con un valore della semivarianza pari a 12 e una distanza media fra le osservazioni di circa 8 km. Si può osservare che la similarità dei valori diminuisce al crescere della distanza fra i punti.

La funzione che calcola i punti del semivariogramma è la seguente:

```
v<-variogram(PM_AVG05~1,toscana.aria,width=10000)
```

L'opzione width indica l'ampiezza della classe di distanza da utilizzare. Se non viene indicata il programma imposta 1/13 della distanza della coppia più lontana.

```
plot(v,type='b')
```



L'interpolazione del semivariogramma.

I punti (o anche la spezzata che li unisce) costituiscono il cosidetto “semivariogramma sperimentale” in quanto calcolato sui punti campione disponibili. Il passo successivo dell'analisi è quello di individuare un modello matematico che si adatti ai dati sperimentali e che consenta di interpretarli sulla base delle

$$\gamma_{dist} = \frac{1}{2N_{dist}} \sum_{1}^{N_{dist}} \left(PM_{dist,i} - PM_{dist,j} \right)^2$$

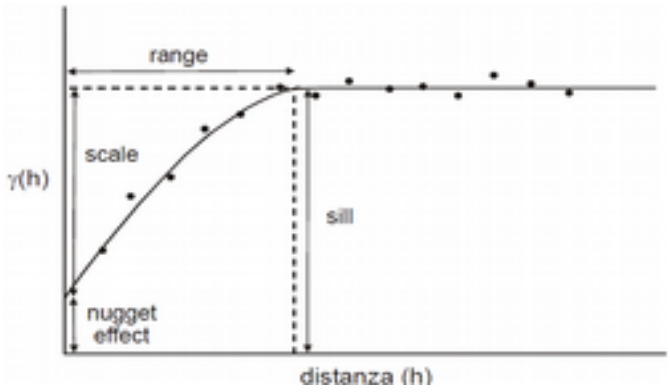
con N_{dist} numero punti nell'intervallo di distanza $Pm_{dist,i}$ e $Pm_{dist,j}$ valori della variabile PM10 per ogni coppia di punti $i \neq j$ all'interno della classe di distanza.

caratteristiche del fenomeno studiato.

La figura seguente mostra un modello matematico applicato ai punti di un variogramma sperimentale. Come evidenziato in figura le caratteristiche da evidenziare sono le seguenti:

- l'intercetta sull'asse delle ordinate – il cosiddetto nudget-effect (effetto “pepita”): tale intercetta, in teoria, significherebbe che due punti infinitamente vicini possono avere una variabilità nella variabile territoriale studiata (nel nostro caso inquinamento da PM10); tale fenomeno, poco verosimile, potrebbe avere diverse giustificazioni, quali errori di misura, modello non perfettamente adatto ai punti sperimentali, ecc.. Se il valore è eccessivamente alto rispetto al parametro successivo (scala), il modello non è rappresentativo del fenomeno studiato.
- *Range*: è il valore di distanza oltre il quale le osservazioni non sono più fra loro influenzate.
- *`Sill:`* E’ conosciuto anche come altezza del variogramma. Individua il valore di γ(h) in corrispondenza del quale la semivarianza non mostra più apprezzabili variazioni. L’esistenza ed il valore del sill sono elementi importanti per la caratterizzazione della continuità e variazione spaziale dei dati sperimentali.

I valori del sill, range, nugget effect, scale forniscono importanti indicazioni riguardo alla struttura dei dati, ovvero riguardo alla modalità con cui la variabile in esame varia con la distanza, e sono indispensabili per la modellizzazione del variogramma sperimentale.



I modelli a disposizione sono moltissimi. Per vedere quelli disponibili in R digitare:

```
show.vgms()
```

Uno dei più utilizzati è il c.d. modello sferico. La funzione `vgm` consente di generare diversi modelli di variogramma. La sintassi, per il modello sferico, è la seguente:

```
vgm(psill=25, model="Sph", range=40000, nugget=10)
```

Per generare il modello di variogramma però è necessario adattare (“fittare” in gergo statistico) il modello ai dati `sperimentali:`

```
v.fit<-fit.variogram(v, vgm(psill=25, model="Sph", range=40000, nugget=10))
```

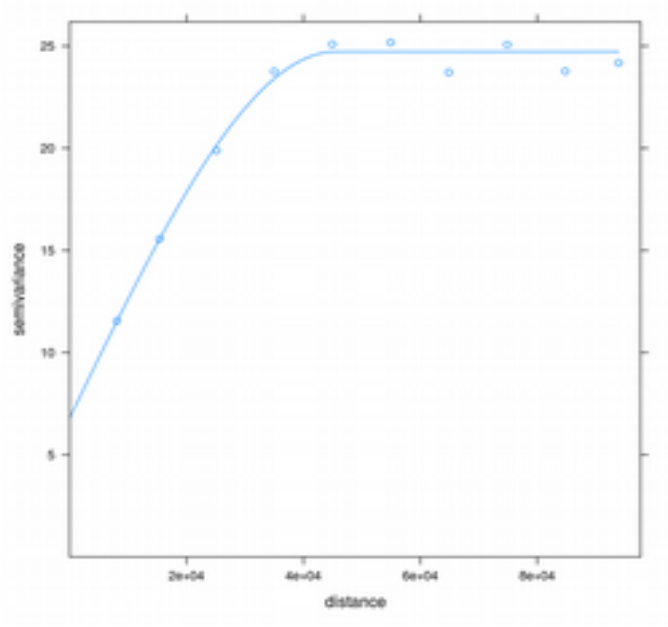
I dati di psill, range e nugget sono inseriti in modo approssimativo e rappresentano solo dei possibili

valori di partenza per agevolare il calcolo. I dati interpolati sono i seguenti:

```
  model     psill    range
1   Nug  6.806977     0.00
2   Sph 17.919232 45508.04
```

Plottando sia il variogramma sperimentale sia il mdello è possibile apprezzare il buon adattamento ai dati sperimentali.

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



La stima di variogrammi multivariati

Le stime territoriali tramite variogramma possono fornire molte più informazioni se si analizzano più variabili che si ritengono fra loro collegate, come - nel nostro esempio – diverse forme di inquinamento come PM10 e ossidi di azoto NOX e la densità di popolazione. Calcolando i la tavola di correlazione è possibile notare come queste siano fra loro collegate. Il "collegamento" potrebbe inoltre essere rafforzato (ma anche smentito) da effetti di tipo spaziale.

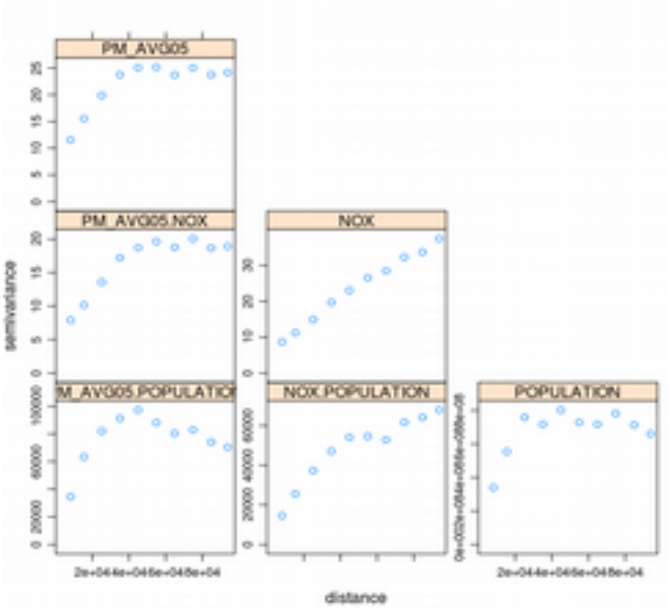
```
cor(as.data.frame(toscana.aria)[c("PM_AVG05","NOX_AVG05","POPULATION")])
```

```
           PM_AVG05 NOX_AVG05 POPULATION
PM_AVG05   1.0000000 0.5819956  0.5397856
NOX_AVG05  0.5819956 1.0000000  0.4109855
POPULATION 0.5397856 0.4109855  1.0000000
```

Per poter analizzare i c.d. Variogrammi multivriati è necessario utilizzare un oggetto specifico di R: l'oggetto `gstat`. La sequenza di comandi è la seguente:

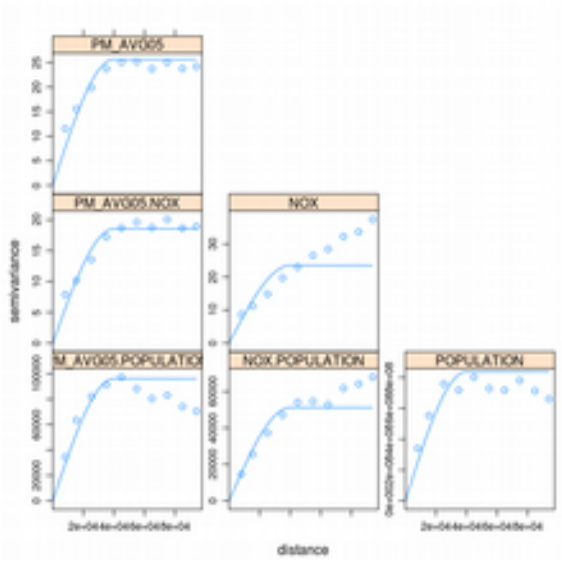
```
g<-gstat(NULL, "PM_AVG05", PM_AVG05~1, toscana.aria)
```

```
# il comando precedente genera l'oggetto gstat g con i dati relativi al PM10
g<-gstat(g, "NOX", NOX_AVG05~1, toscana.aria)
# questo comando aggiunge all'oggetto g già esistente i dati di NOX
g<-gstat(g, "POPULATION", POPULATION/1000~1, toscana.aria)
# aggiunge la popolazione
vm<-variogram(g, width=10000)
plot(vm)
```



I variogrammi sperimentali consentono di registrare una certa presenza di effetti spaziali di tipo incrociato, migliori nel caso della densità di popolazione, un po' meno marcati per i NOX che sembrano seguire un modello di distribuzione territoriale diverso rispetto a popolazione e PM10 che hanno valori di sill, range e nugget relativamente simili e modellizzabili con il modello sferico, sia nei variogrammi individuali sia in quelli incrociati.
Il modello di variogramma si costruisce come nel caso precedente:

```
vm.mfit<-fit.lmc(vm, g, vgm(model="Sph", range=40000))
plot(vm,vm.mfit)
```



I modelli fittati sembrano confermare le considerazioni fatte per i variogrammi mutivariati sperimentali.

Le stime predittive spaziali

Le stime predittive hanno lo scopo di ricostruire, tramite una stima statistica, la superficie Z di diffusione territoriale della grandezza in esame sulla base dei punti campione noti. La stima più efficiente dipende dalle informazioni disponibili contenute nei punti campione, in altre basi dati ancillari e in generale, in letteratura relativamente al fenomeno studiato. La figura seguente riporta un quadro dei metodi applicabili nelle varie situazioni.
L'interpolazione sulla base della distanza inversa pesata (IDW) è applicabile quando non è possibile calcolare, con i dati a disposizione, un variogramma soddisfacente. Se però è disponibile un dato ancillare che presenta una relazione statisticamente significativa e logicamente giustificabile con i pinti campione è possibile ricostruire la superficie Z tramite tale modello attraverso una stima predittiva locale. Se invece il variogramma mostra una buona correlazione spaziale è possibile impiegare la cosiddetta tecnica del kriging.

Metodi di interpolazione non geostatistici

La tecnica della distanza pesata ricostruisce per ogni localizzazione s_0 della superficie incognita $Z(s_0)$ tramite una media pesata sulla distanza tra questa e i punti campione:

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

con

$$w(s_i) = \|s_i - s_0\|^{-p}$$

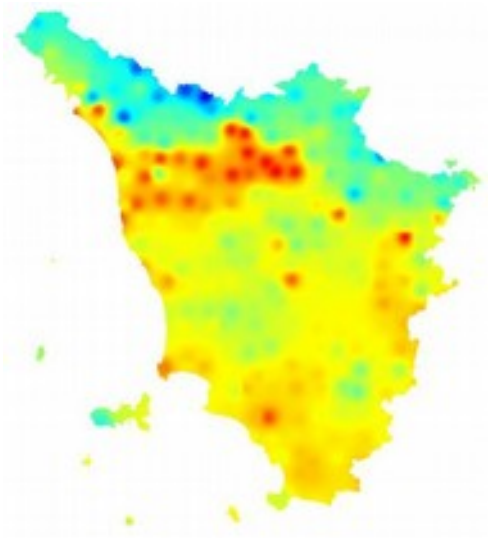
dove la coppia di linee verticali doppie indica la distanza euclidea fra s_i e s_0 e p il peso che indica quanto velocemente decade l'influenza di ciascun punto campione al crescere della sua distanza.

In R è disponibile la funzione `idw`:

```
idw.out<-idw(PM_AVG05~1, qual.aria, dem, idp=2.5)
```

Il primo parametro della formula rappresenta la funzione da applicare, il secondo parametro lo SpatialPointDataFrame dei punti campione, il terzo parametro la SpatialGridDataFrame i cui parametri geografici saranno impiegati per definire la superficie predetta *Z*, infine idp è il peso *p*. Il comando successivo consente di salvare un file GeoTiff visualizzabile con Qgis.

```
writeGDAL(idw.out, 'idw.tiff', drivername="GTiff", type='Byte' )
```



Il Kriging

Il metodo del kriging è simile al metodo delle funzioni inverse della distanza pesata (IDW), poiché anch'esso attribuisce un peso alle singole misure della variabile nell'intorno del punto da interpolare,

da cui dipende la stima del valore nel prediction point. Nel kriging i pesi da attribuire sono espressione sia della distanza che separa il punto misurato dal prediction point sia della correlazione spaziale d'insieme dei punti sperimentali utilizzati nell'interpolazione. La combinazione ottimale dei pesi da applicare nella stima del valore della variabile in ogni prediction point, è espressione del modello di dipendenza e relazione spaziale dei dati sperimentali coinvolti nell'interpolazione. A tale riguardo, il kriging si basa sull'assunzione che punti vicinali avranno un certo grado di correlazione mentre punti ampiamente distanti saranno statisticamente indipendenti. La correlazione spaziale (o autocorrelazione) dei punti sperimentali implicati nell'interpolazione è ovviamente quantificata nel *kriging* attraverso il variogramma.
Esistono diversi tipi di kriging, tra cui kriging ordinario (ordinary kriging) e kriging universale (universal kriging), per diverse tipologie di variabili. Ciò che li differenzia è il tipo di variabile usata: il kriging ordinario può lavorare solo con variabili stazionarie del secondo ordine (presentano cioè media costante e varianza dipendente solo dal lag muovendosi da punto a punto), il kriging universale può invece lavorare anche con variabili non stazionarie cioè che presentano cioè una anisotropia relativamente alla superficie.
In R i kriging è risolto dalla funzione krige:

Per il kriging ordinario è necessario solo indicare la variabile, lo SpatialPointDafaFrame, la SpatialGridDataFrame ed infine il variogramma.

```
# kriging ordinario
lz.ok<-krige(PM_AVG05~1,toscana.aria,dem,v.fit)
writeGDAL(lz.ok, 'OK.tiff', drivername="GTiff", type='Byte' )
```

Nell'esempio in esame è lecito supporre che, se il fenomeno dell'inquinamento da PM10 mostra una relazione con la quota, sia possibile che la superficie presenti una anisotropia descrivibile tramite questa variabile. Per applicare il modello del kriging universale è innanzitutto necessario creare una relazione funzionale fra PM10 e quota; Il variogramma anisotropo (o non stazionario) sarà calcolato sulla base di tale relazione.

```
# kriging universale
f<-PM_AVG05~quota
vt<-variogram(f,toscana.aria,width=10000)
plot(vt)
vt.fit<-fit.variogram(vt, vgm(psill=12, model="Sph", range=30000, nugget=4))
plot(vt,vt.fit)
vt.fit

  model    psill    range
1   Nug 2.454840     0.00
2   Sph 8.490954 22813.44
```

Sulla base di tale variogramma è npossibile calcolare la superfice tramite il metodo del kriging universale.

```
lz.uk<-krige(PM_AVG05~quota,toscana.aria,dem,vt.fit)
writeGDAL(lz.uk, 'UK.tiff', drivername="GTiff", type='Byte' )
```

Un ultimo esempio di kriging è il cosiddetto kriging a blocchi. Con questa forma di elkaborazione, di derivazione dalle scienze geologiche, si assume che la superficie sia anisotropa per superfici discrete. Nel nostro esempio supponiamo – irrealisticamante – una anisotropia per limiti comunali.

```
# Block polygon kriging
comuni<-readOGR(dsn='istat', layer='ISTAT')
lz.pols=krige(PM_AVG05~1, toscana.aria, comuni, v.fit)
writeGDAL(lz.pols, 'UK.tiff', drivername="GTiff", type='Byte' )
```

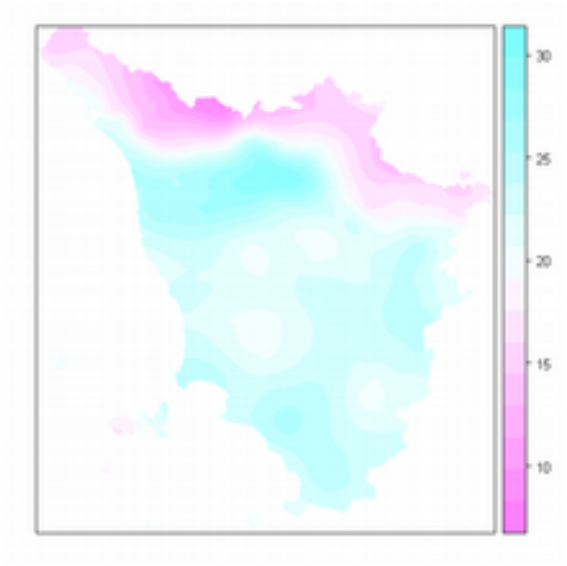


Illustrazione 1: Ordinary Kriging

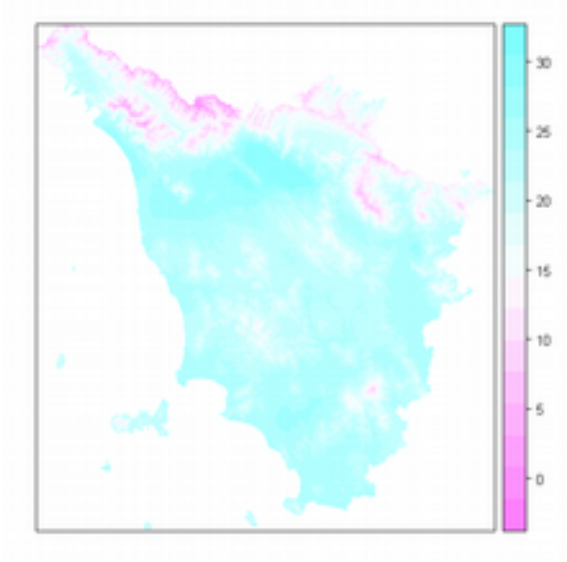


Illustrazione 2: Universal Kriging

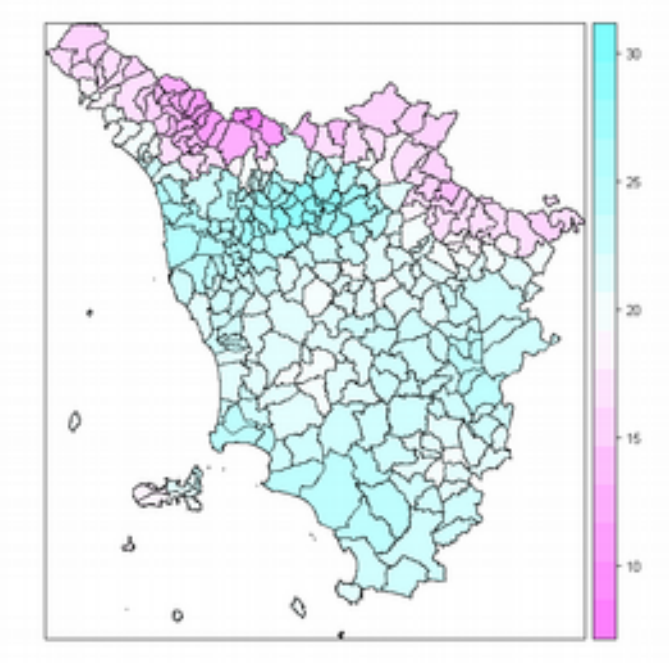
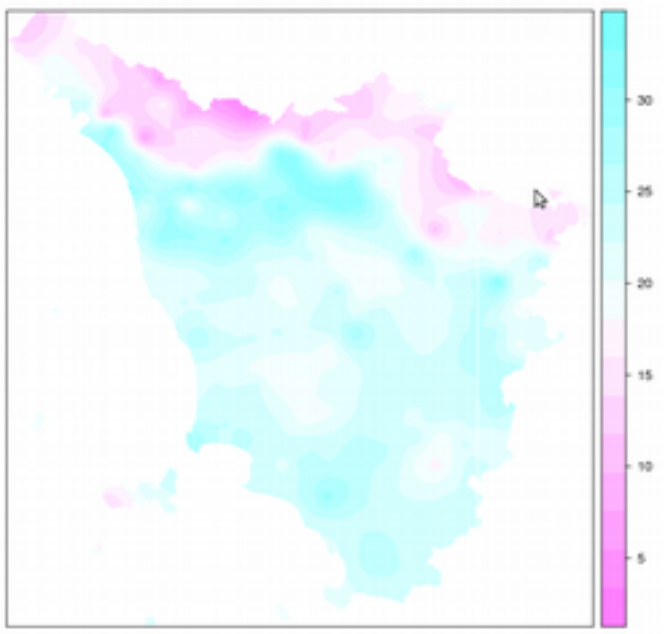


Illustrazione 3: Block Kriging

Stime predittive multivariate: il cokriging

Impiegando un variogramma multivariato è possibile calcolare tramite il cokriging le relative superfici:

```
cok.map<-predict(vm.mfit,dem)
writeGDAL(cok.map, 'COK.tiff', drivername="GTiff", type='Byte')
```



La diagnostica della stima

Finora abbiamo valutato i risultati dei diversi modelli solo dal punto di vista 'visuale'. E' possibile però cercare di validare i risultati, anche in relazione alle molte possibili scelte che è necessario fare, in modo più accurato. Una delle tecniche più semplici è la cosiddetta *cross validation*. Nella *cross validation* innazitutto si dividono i punti campione in due gruppi (non necessariamente uguali): il primo gruppo servirà a costruire il modello spaziale mentre il secondo gruppo a validare i risultati. Nel nostro caso, avendo a disposizione 362 osservazioni, ne utilizzeremo 160 per la costruzione del modello e le rimanenti per la validazione.

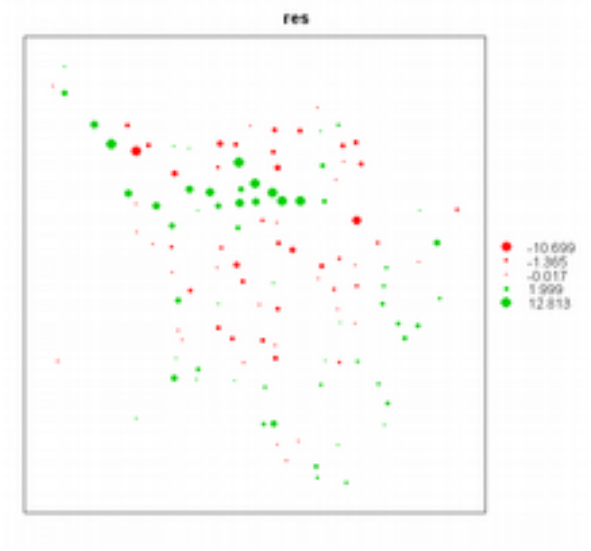
```
sel<-sample(1:236, 160)
m.model<-toscana.aria[sel, ]
m.valid<-toscana.aria[-sel, ]
#
v100.fit<-fit.variogram(variogram(PM_AVG05~1, m.model, width=10000), vgm(psill=25,
model="Sph", range=40000, nugget=10))
```

```
m.valid.pr<-krige(PM_AVG05~1, m.model, m.valid, v100.fit)
#
resid.kr<-m.valid$PM_AVG05-m.valid.pr$var1.pred
summary(resid.kr)
#
# vengono visualizzate le statistiche dei residui
#
resid.mean<-m.valid$PM_AVG05-mean(m.valid$PM_AVG05)
R2<-1-sum(resid.kr^2)/sum(resid.mean^2)
R2
#
# viene visualizzato l'R^2
#
```

L'R^2 indica che la predizione kriging è un miglior stimatore della semplice media con un indice di circa 0,4 da un minimo di 0 ad un massimo di 1. La mappa dei residui, plottata con il comando

```
m.valid.pr$res<-resid.kr
bubble(m.valid.pr, 'res')
```

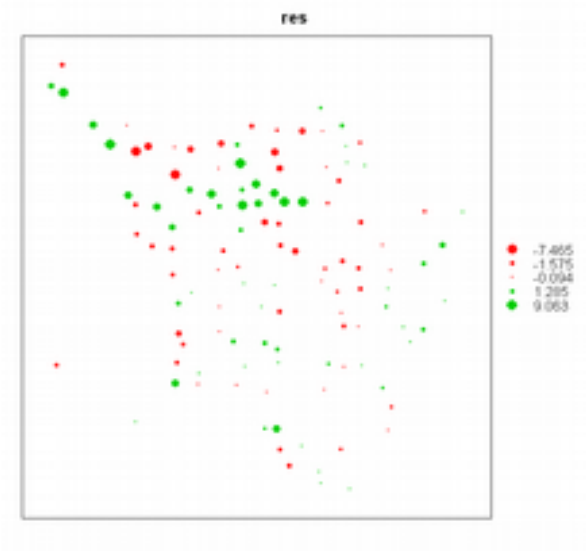
consente di visualizzare le zone dove maggiore è l'errore fra il dato campionato e quello stimato con il kriging.



Possiamo ripetere le elaborazioni effettuate con il modello di kriging universale, che visualmente sembrava migliore.

```
v100.fit<-fit.variogram(variogram(PM_AVG05~1, m.model, width=10000), vgm(psill=12,
model="Exp", range=30000, nugget=4))
m.valid.pr<-krige(PM_AVG05~quota, m.model, m.valid, v100.fit)
resid.kr<-m.valid$PM_AVG05-m.valid.pr$var1.pred
resid.mean<-m.valid$PM_AVG05-mean(m.valid$PM_AVG05)
R2<-1-sum(resid.kr^2)/sum(resid.mean^2)
R2
m.valid.pr$res<-resid.kr
```

```
bubble(m.valid.pr, 'res')
```



La stima sembra nettamente migliorata.

Statistica spaziale

La statistica spaziale/territoriale è quell'insieme di tecniche statistiche che trattano i dati tenendo conto esplicitamente della loro natura spaziale, ossia della posizione in cui questi si sono manifestati nello spazio. Proprio perché si tratta di un insieme di tecniche, non si tratta di una materia consolidata ma presenta ancora anime diverse, in base alla disciplina scientifica di appartenenza degli studiosi che hanno presentato tecniche ad hoc per specifici problemi (Zani 1993).
Alcune delle materie che più hanno contribuito allo sviluppo della statistica spaziale sono (in ordine alfabetico):

- biologia (ad es. studio delle popolazioni biologiche)
- economia e econometria (ad es. lavoro, marketing territoriale)
- geografia (problemi legati alla cartografia e telerilevamento)
- medicina (in particolare l'epidemiologia)
- matematica e geometria
- pianificazione territoriale (gestione del territorio)
- statistica (descrittiva, inferenza, applicata)

A differenza della geostatistica, caratterizzata da **fenomeni diffusivi (**che si manifestano in modo continuo sul territorio ma con differente intensità, ad es. la temperatura), la statistica spaziale studia **fenomeni dispersivi**, cioè che si manifestano in parti ben circoscritte del territorio, come ad es. le città, aziende agricole, fabbricati. Si tratta di una classificazione, spesso non univoca perché dipende dal punto di vista del ricercatore (es. onde radio), fatta in base al fenomeno e non rispetto al territorio. Altra differenza è che la statistica spaziale si basa su **partizioni irregolari** dello spazio (invece che su griglie regolari) cioè su mosaici di poligoni, p.e. confiini amministrativi, fogli catastali, particelle forestali, ecc..

Eterogeneità e autocorrelazione spaziale

La prima legge di Tobler afferma che: "Tutto è correlato con tutto, ma le cose vicine sono più correlate delle cose lontane". Su questa semplice constatazione si fonda uno dei concetti più importanti della statistica spaziale quello di **autocorrelazione spaziale**. Per autocorrelazione perciò si intende la correlazione di una variabile con sé stessa nello spazio: se aree vicine sono più simili di quelle lontane allora si ha una autocorrelazione positiva; se aree vicine sono più diverse di quelle lontane si ha autocorrelazione negativa. Valori di autocorrelazione positivi indicano fenomeni di clusterizzazione dei dati mentre quelli negativi indicano una distribuzione di valori senza un preciso pattern territoriale.
L'interdipendenza o autocorrelazione spaziale si manifesta quando esiste una relazione funzionale tra ciò che accade in una posizione specifica dello spazio e ciò che accade in altre posizioni.
In altre parole, le caratteristiche di un determinato fenomeno in una regione non sono spiegate unicamente da determinanti interne alla stessa ma anche da alcune proprie delle altre regioni, più o meno vicine.
L'opposto della autocorrelazione è la ETEROGENEITÀ SPAZIALE che si manifesta quando non si rintraccia, nello spazio, alcuna forma stabile di relazione funzionale tra i valori assunti dal fenomeno in analisi. Sono state sviluppate molte tecniche per lo studio dell'autocorrelazione. Il principio comune di queste tecniche si fonda sul principio di comparazione della distribuzione in analisi con una

distribuzione di riferimento definita sotto una ipotesi nulla (null hypotesis – no spatial autocorrelation).
L'ipotesi nulla implica che lo spazio non influenza in alcun modo la distribuzione dei valori in analisi ovvero, in altre parole, che una particolare localizzazione è del tutto ininfluente sul valore delle osservazioni.
I test di autocorrelazione per una singola variabile sono basati sul valore di indicatori che combinano un valore misurato in una data localizzazione con i valori dell'intorno (scarti).
Fondamentalmente questi test rappresentano delle misure che tengono conto della associazione dei valori (covarianza, correlazione, differenza) della associazione nello spazio (ad esempio contiguità, distanza) si ritiene che esista una autocorrelazione spaziale quando l'indicatore statistico assume un valore "estremo" in riferimento al valore ottenuto nella ipotesi nulla (assenza di autocorrelazione).

La contiguità spaziale
Per poter definire la "vicinanza" o la "lontananza" in una partizione irregolare dello spazio è fondamentale il concetto di contiguità. Si definisce come “struttura spaziale”, la partizione, irrregolare (o regolare), del territorio considerato. Ma per poter studiare i fenomeni che si manifestano su di essa questo non è sufficiente: è necessario definire anche l’interazione territoriale ossia il modo con cui le unità territoriali (Ut) s’influenzano reciprocamente. Per formulare l’ipotesi di interazione tra unità territoriali è necessario distinguere tra **contiguità** e **connessione** delle stesse. Il primo concetto è legato alla **contiguità fisica**: due Ut si definiscono contigue se hanno almeno un punto dei propri perimetri in comune (in tal senso si potranno formulare più casi). La **connessione** invece non è limitata solo all’aspetto fisico, ma può anche essere riferita ad un sistema di pesi astratto o di tipo economico. Ad ogni modo, più il peso è elevato e maggiore è relazione territoriale tra le aree territoriali.
Nasce il problema di individuare il vicinato di una generica i-esima Ut. Questo in via generica può essere visto come l’insieme delle aree che sono “prossime” all’unità territoriale considerata. Se indichiamo il vicinato con N(i), è possibile definire la nodalità v(i) la cardinalità dell’insieme N(i).
Per poter definire il vicinato è necessario definire prima, oltre la struttura territoriale, il tipo di interazione tra Ut ipotizzata.
In R, usando la libreria **spdep** le relazioni di contiguità spaziale sono gestite da una specifica classe di oggetti nb. La creazione di un oggetto nb è possibile utilizzando la funzione ploy2nb. I comandi seguenti mostrano il funzionamento di un oggetto appartenente a tale classe.

```
library(spdep)
library(rgdal)
setwd('...geostatistica/R_dati/')
comuni<-readOGR(layer='comuni_toscana' , dsn='istat')
IDs<-comuni$ISTAT
spplot(comuni,'REDDITO')
comuni.Qnb<-poly2nb(comuni,row.names=IDs)
# row.names serve a identificare gli oggetti nella matrice di contiguità
comuni.Qnb
summary(comuni.Qnb)
str(comuni.Qnb)
```

Come si può notare dall'output ottenuto, il summary consente di ottenere molti dati riassuntivi relativamente ai rapporti di contiguità fra gli oggetti (links), quali il numero medio di collegamenti la distribuzione di frequenza per numero di oggetti confinanti con ciascun poligono, ecc.
Una prima cosa da notare è che il sommario dei risultati segnala la presenza di 2 comuni con zero collegamenti. Sebbene, come vedremo in seguito, la cosa possa

```
essere "trattata", è preferibile ripulire il modello territoriale da oggetti che,
non essendo collegati, non sono significativi per l'analisi:

comuni[comuni$ISTAT==9049005,]$NOME;comuni[comuni$ISTAT==9053012,]$NOME
nonzero.link<-comuni$ISTAT!=9049005 & comuni$ISTAT!=9053012
# questo comando crea un vettore TRUE/FALSE con i codici a 0 link come falsi
comuni<-comuni[nonzero.link,]
# questo comando estrae dallo SpatialPolygonDataFrame gli oggetti TRUE
```

Ripetendo i passi precedenti:

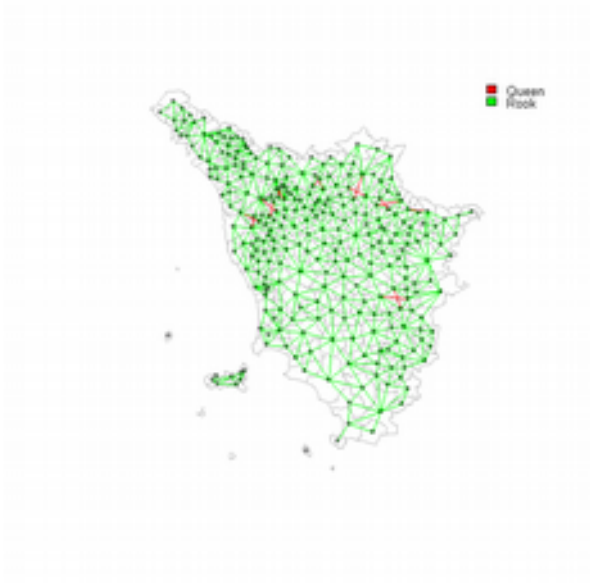
```
IDs<-comuni$ISTAT
comuni.Qnb<-poly2nb(comuni,row.names=IDs)
comuni.Qnb
```

Per default la condizione di contiguità sussiste se è condiviso anche un solo punto fra due oggetti. Tale relazione è detta contiguità della regina (queen), dal movimento dell'omonimo pezzo degli scacchi. Una condizione di contiguità più restrittiva è la contiguità della torre, che prevede che due oggetti condividano almeno un segmento della spezzata di confine. La contiguità della torre è specificabile tramite il parametro `queen=FALSE`:

```
comuni.Rnb<-poly2nb(comuni, queen=FALSE)
```

Tramite un appropriato plottaggio si può apprezzare la (poca nel caso in esame) differenza fra i due rapporti di contiguità spaziale.

```
plot(comuni, border="grey")
plot(comuni.Qnb, coordinates(comuni), col="red", add=TRUE, cex=0.6)
plot(comuni.Rnb, coordinates(comuni), col="green", add=TRUE, cex=0.6)
legend("topright", legend=c("Queen", "Rook"), fill=c("red", "green"), bty="n",
  cex=0.8, y.intersp=0.8)
```



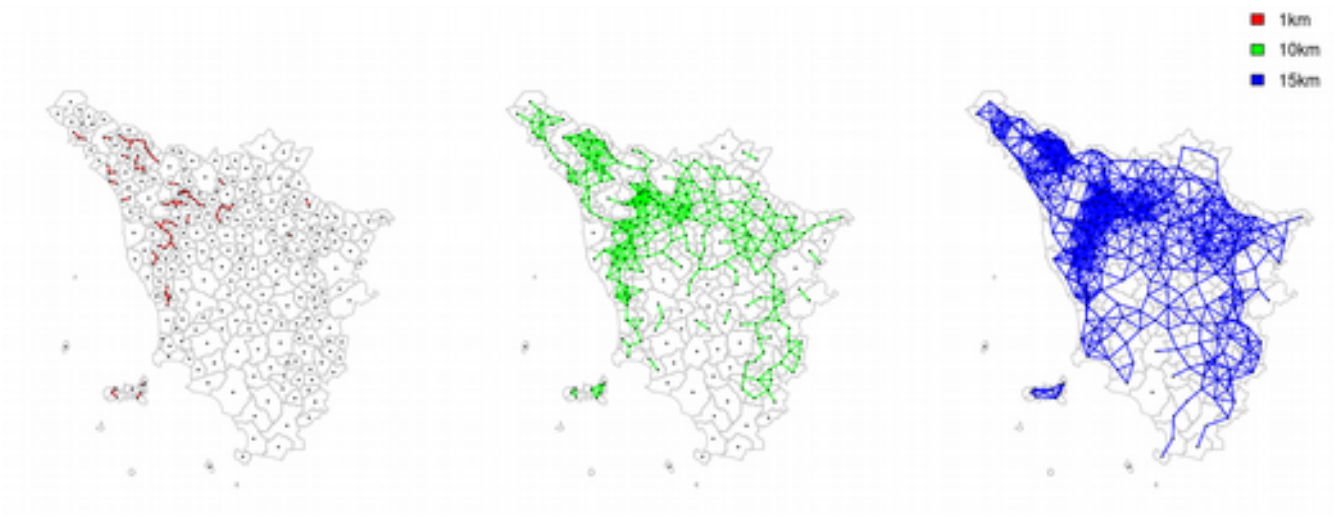
Il criterio di connessione

La contiguità geometrica non è l'unico né il più soddisfacente metodo per rappresentare rapporti di connessione fra due entità. Infatti due entità potrebbero confinare fisicamente ma non essere assolutamente collegate relativamente all'ogetto dello studio (p.e. Perché mancano strade, perché ci sono barriere come crinali, fiumi, ecc.). Esistono quindi altri metodi per definire il modello territoriale. Uno dei più semplici è quello basato sulla distanza fra punti rappresentativi del mosaico deegli oggetti territoriali. Per convenzione, se il mosaico è irregolare viene calcolata la distanza fra il baricentro del poligono; nulla vieta però di utilizzare localizzazioni più significative, come ad esempio la località abitata più importante, il baricentro di una certa forma di utilizzazione di uso del suolo o altro.
Una volta individuato il riferimento per il calcolo della distanza il rapporto di collegamento può essere definito tramite una soglia di distanza massima tollerata oppure individuando i *k* oggetti comunque più vecini a quello considerato.
In R esistono numerose funzioni che consentono di definire il modello territoriale sulla base della distanza o dei k elementi più prossimi (detto anche criterio *knn*, da *k-nearest neighbour objects*).

```
coords<-coordinates(comuni)
# estraee le coordinate dei centroidi dallo SpatialPolygonDataFrame comuni
dsts<-unlist(nbdists(comuni.Qnb,coords))
# calcola le distanze utilizzando l'oggetto nb e le coordinate relative al layer
dei comuni.
summary(dsts)
```

Con le righe suindicate è possibile avere una statistica delle distanze fra i centroidi, potendo così rilevare che la massima distanza in linea d'aria è pari a circa 25 chilometri. La funzione `dnearneigh` consente di calcolare l'oggetto `nb` relativo ai rapporti di connessione regolati da un intervallo minimo fra `d1` e `d2` di distanze.

```
IDs<-comuni$ISTAT
comuni.5km<-dnearneigh(coords,d1=0,d2=5000,row.names=IDs)
comuni.10km<-dnearneigh(coords,d1=0,d2=10000,row.names=IDs)
comuni.15km<-dnearneigh(coords,d1=0,d2=15000,row.names=IDs)
par(mfrow=c(1,3))
plot(comuni, border="grey")
plot(comuni.5km, coordinates(comuni), col="red", add=TRUE, cex=0.3)
plot(comuni, border="grey")
plot(comuni.10km, coordinates(comuni), col="green", add=TRUE, cex=0.3)
plot(comuni, border="grey")
plot(comuni.15km, coordinates(comuni), col="blue", add=TRUE, cex=0.3)
legend("topright", legend=c('1km', '10km', '15km'), fill=c('red', 'green', 'blue'),
   bty="n", cex=1.5, y.intersp=1.5)
```



Per quanto riguarda la procedura knn:

```
comuni.1k<-knn2nb(knearneigh(coords,k=1),row.names=IDs)
comuni.2k<-knn2nb(knearneigh(coords,k=2),row.names=IDs)
comuni.3k<-knn2nb(knearneigh(coords,k=3),row.names=IDs)
par(mfrow=c(1,3))
plot(comuni, border="grey")
plot(comuni.1k, coordinates(comuni), col="red", add=TRUE, cex=0.3)
plot(comuni, border="grey")
plot(comuni.2k, coordinates(comuni), col="green", add=TRUE, cex=0.3)
plot(comuni, border="grey")
plot(comuni.3k, coordinates(comuni), col="blue", add=TRUE, cex=0.3)
legend("topright", legend=c('1knn', '2knn', '3knn'), fill=c('red', 'green',
  'blue'), bty="n", cex=1.5, y.intersp=1.5)
```

I Pesi

Una volta individuato il criterio di connessione è necessario valutarne l'intensità *w*. Nella forma più semplice *w* si basa sul concetto di contiguità binaria secondo cui la struttura delle prossimità è espressa da valori 0-1. Se due unità spaziali hanno un confine in comune, di lunghezza maggiore di zero,

saranno considerati contigui e saranno contrassegnati dal valore 1. Viceversa se non sono contigui la loro accoppiata avrà valore 0. In R la funzione `nb2listw` ha lo scopo di trasformare una lista di relazioni di vicinato (oggetto di classe `nb`) in una lista i di intensità di connesstione (oggetto di classe `listw`).

```
comuni.w3k<-nb2listw(comuni.3k)
```

I pesi così ottenuti sono sottoforma di lista. Per poterli visualizzare sotto forma di matrice:

```
edit(listw2mat(comuni.w3k))
```

Si ottiene una rappresentazione in stile *spreadsheet* della cosiddetta matrice di pesi. La matrice è quadrata ed ha i diversi comuni sia nelle righe che nelle colonne. Gli elementi maggiori di zero esprimono l'intensità di connessione fra gli oggetti indicati in riga e colonna. Esplorando il tabellone è facile rendersi conto che i pesi sono tutti identici e uguali a 0.3333. Questo è facilmente spiegabil in quanto il modello di coclass(ntiguità impiegato è il knn con 3 vicini più prossimi ed il metodo di default della funzione assegna a ciascun link una intensità pari a 1 fratto la somma di tutti i link dell'elemento. Tale metodo però non è l'unico disponibile. La funzione ammette i seguenti metodi, selezionati dall'opzione `style=` :

1. `B` è il cosiddetto peso binario, in cui ogni connessione ha valore unitario; l'intensità di un legame è indipendente dall'esistenza di altri legami 'concorrenti'.
2. `W` è il valore di default e standardizza il peso con un coefficiente pari a 1 diviso il numero dei links dell'oggetto con il maggior numero di collegamenti; l'intensità di un legame diviene così inversamente proporzionale al numero degli altri diversi legami;
3. `C` è il metodo della standardizzazione globale, simile al precedente, ma divide ciascun collegamento per il numero dei link dell'oggetto a cui si riferisce anziché dell'oggetto con il maggior numero di links
4. `S` è il c.d. variance-stabilizing coding scheme proposto da Tiefelsdorf et al. 1999, p. 167-168, con lo scopo di avere un peso con valori intermedi tra `B, C` e `W.`

Infine si deve notare che per forzare la costruzione di una lista di pesi per oggetti `nb` con elementi senza connessioni è necessario impostare l'opzione `zero.policy=TRUE`. Di seguito alcuni esempi:

```
par(mfrow=c(2,2))
comuni.RnbwB<-nb2listw(comuni.Rnb, zero.policy=TRUE, style='B')
hist(sapply(comuni.RnbwB$weights,sum))
# questa istruzione consente di visualizzare la distribuzione di frequenza dei pesi
comuni.RnbwW<-nb2listw(comuni.Rnb,zero.policy=TRUE,style='W')
hist(sapply(comuni.RnbwW$weights,sum))
comuni.RnbwC<-nb2listw(comuni.Rnb,zero.policy=TRUE,style='C')
hist(sapply(comuni.RnbwC$weights,sum))
comuni.RnbwS<-nb2listw(comuni.Rnb,zero.policy=TRUE,style='S')
hist(sapply(comuni.RnbwS$weights,sum))
```

Oltre ai metodi pre-impostati, tramite la l'opzione `glist` è possibile costruire pesi con qualsiasi tipo di funzione. Una applicazione particolarmente interessante è data dalla possibilità di costruire set di pesi inversamente proporzionali alla distanza fra i baricentri. Applichiamo tale possibilità alla relazione di

connessione data dalla distanza massima di 25 chilometri:

```
coords<-coordinates(comuni)
comuni.25km<-dnearneigh(coords,d1=0,d2=25000,row.names=IDs)
dist25<-nbdists(comuni.25km, coordinates(comuni))
distw<-lapply(dist25, function(x) 1/(x/10000))
comuni.dist25W<-nb2listw(comuni.25km,glist=distw, style='B', zero.policy=TRUE)
hist(sapply(comuni.dist25W$weights,sum))
```

L'autocorrelazione spaziale

La misura più diffusa di autoccorrelazione spaziale di dati è il Moran's I (indice I di Moran) che misura il grado di autocorrelazione presente nell'intero data set sottoposto ad analisi.

Formalmente l'indice di Moran per un numero n di osservazioni nella variabile y ed in corrispondenza di i localizzazioni si scrive:

$$I = \frac{\sum_{i}^{n}(y_i - \bar{y})(\sum_{j}^{n} w_{ij}(y_j - \bar{y}))}{\sum_{i}^{n}(y_i - \bar{y})^2}$$

Nella formulazione di Moran yi rappresenta il valore della variabile nella localizzazione i-esima e y_j i valori della variabile nelle j-eseme localizzazioni che costituiscono l'intorno della posizione i-esima e w_{ij} rappresenta il peso descritto nella matrice dei pesi per la coppia di localizzazioni i - j.

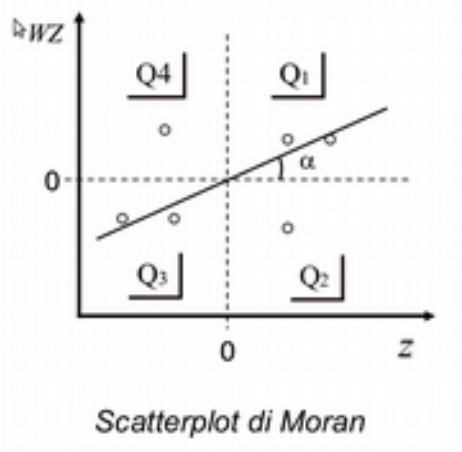
Il termine $\sum_j w_{ij} (y_j - y))$ viene indicato come scarto spaziale della localizzazione i-esima e quindi l'indice I rappresenta il coefficiente di correlazione degli scarti stessi e misura la pendenza della retta di regressione.

La statistica I è strutturalmente simile al coefficiente di correlazione e come questo varia tra -1.0 e +1.0; ne segue che alti valori di I indicano alta autocorrelazione spaziale, viceversa bassi valori di I.

Differentemente dal coefficiente di correlazione, però, l'indice I non assume un valore teorico nullo in corrispondenza della condizione di indipendenza, ma un valore negativo molto prossimo a zero . La statistica di Moran può essere calcolata con le seguenti funzioni:

```
mst<-moran.test(comuni$REDDITO, nb2listw(comuni.Qnb,style='B'))
mst

mst<-
moran.test(comuni$REDDITO,nb2listw(comuni.15km,style='B',zero.policy=TRUE),zero.pol
icy=TRUE)
mst
# N.B. Nel caso di oggetti con zero link è necessario specificare per ogni
funzione l'opzione zero.policy.
```

La statistica di Moran può utilmente essere rappresentata tramite un grafico che fornisce nformazioni visulamente molto utili. Il cosiddetto Moran scatterplot riporta nelle ascisse la variabile x (normalizzata o meno) e sulle ordinate il cosiddetto ritardo spaziale (Wx) che rappresenta la media dei valori degli elementi confinanti con ciascun oggetto moltiplicata per il relativo peso di connessione. In questa rappresentazione la I di moran rappresenta il coefficiente angolare di una retta passante per i valori medi degli assi (origine degli assi se le variabili sono standardizzate). Se quindi i punti sono dispersi fra i quattro quadranti si avrà una scarsa autocorrelazione spaziale, se invece esiste una chiara relazione lsa retta tenderà ad inclinarsi verso i 45 gradi e la collocazione delle osservazioni potrà permettere una chiara interpretazione delle relazioni spaziali esistenti: il primo ed il terzo quadrante rappresentano aree di valori con correlazioni positive (alto - alto, basso - basso) mentre il secondo ed il quarto quadrante rappresentano aree dei dati a correlazione negativa.



Scatterplot di Moran

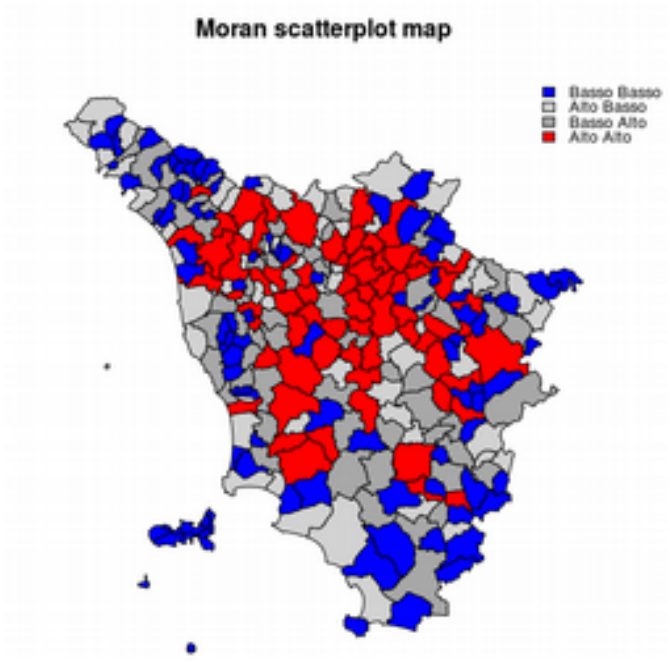
R dispone di una specifica funzione per la costruzione dei moran scatterplot, che può essere applicata a valori normalizzati o puri.

```
mp<-moran.plot(comuni$REDDITO, listw=nb2listw(comuni.Qnb, style="B"),
   labels=comuni$NOME)
msp<-moran.plot(scale(comuni$REDDITO)[,1], listw=nb2listw(comuni.Qnb, style="B"),
   labels=comuni$NOME, xlim=c(-4,5),ylim=c(-6,15), pch=10)
# Questa seconda formulazione applica il plot a variabili standardizzate tramite la
   funzione scale() e modifica, tramite i parametri xlim e ylim i limiti dellarea
   rappresentata.
```

Può essere interesante individuare i comuni appartenenti ai diversi quadranti geograficamente anziché sullo scatterplot, in modo da visualizzare i cluster che si vanno a formare, R non dispone di una funzione ad-hoc, quindi il procedimento è un po' complesso:

```
x<-comuni$REDDITO
lhx <- cut(x, breaks=c(min(x), mean(x), max(x)), labels=c('B', 'A'),
   include.lowest=TRUE)
```

```
# Questo comando individua la componente del quadrante della varabile dividendo in
  alto e basso
wx<-lag(nb2listw(comuni.Qnb, style='B'),comuni$REDDITO)
# la funzione lag() calcola i cosiddetti ritardi spaziali, cioè per ogni comune, il
  valore medio dei confinanti pesato per l'intensità di collegamento.
lhwx <- cut(wx, breaks=c(min(wx), mean(wx), max(wx)), labels=c('B', 'A'),
  include.lowest=TRUE)
# si classifica l'ordinata del grafico nei valori alto e basso
lhlh <- interaction(lhx, lhwx, drop=TRUE)
# interaction() combina le due classificazioni creando un vettore di etichette
cols <- rep(1, length(lhlh))
# con questa si crea un vettore numerico di lunghezza pari  al numero di comuni e
  quindi della stessa lunghezza di lhlh
cols[lhlh == 'B.B'] <- 1
cols[lhlh == 'A.B'] <- 2
cols[lhlh == 'B.A'] <- 3
cols[lhlh == 'A.A'] <- 4
# sulla base dei valori dell'oggetto lhlh si classifica sulla base dei 4 quadranti
  del Moran scatterplot
plot(comuni, col=c("blue", "lightgrey", "darkgrey", "red")[cols])
# si plotta il mosaico dei comuni colorando i poligoni sulla base del valore
  dell'oggetto cols
legend("topright", legend=c('Basso Basso', 'Alto Basso', 'Basso Alto', 'Alto
  Alto'), fill=c("blue", "lightgrey", "darkgrey", "red"), bty="n", cex=0.8,
  y.intersp=0.8)
title("Moran scatterplot map")
# si disegna la legenda ed il titolo
```



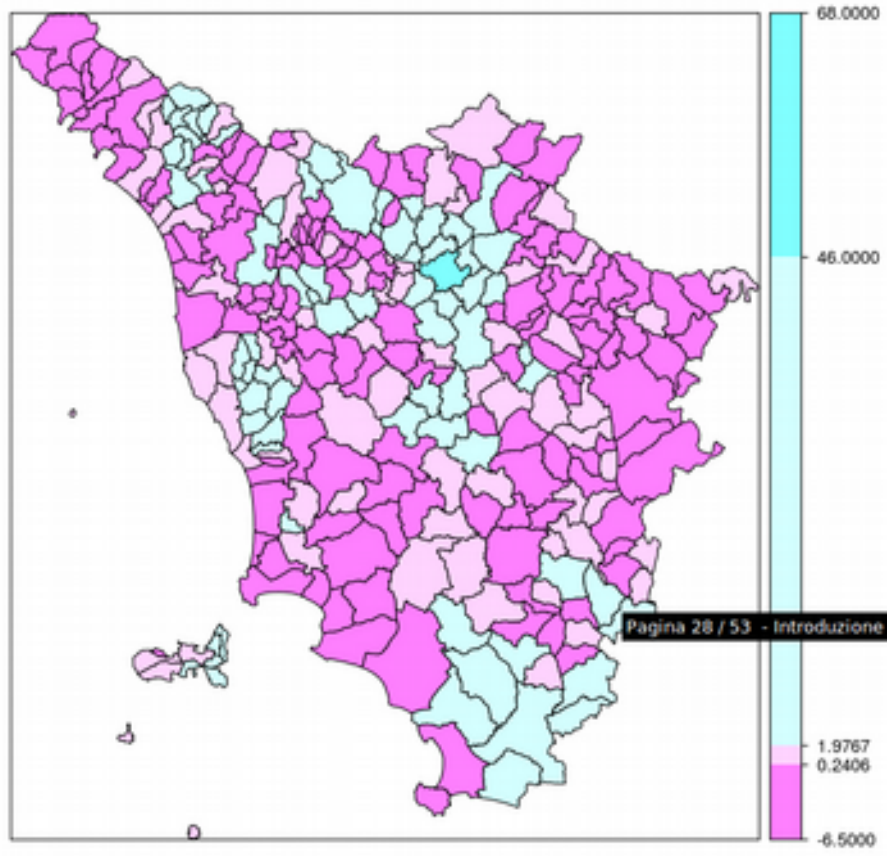
Il Moran Scatterplot ha il grande vantaggio di trasferire un indicatore medio valido per un insieme di regioni su un grafico che consente di esplorare meglio la sua natura distinguendo le regioni rispetto a diversi tipi di interdipendenza. In questa direzione è spesso utile associare a questo indicatore globale e alla sua rappresentazione grafica anche un indicatore di autocorrelazione locale, in grado cioè che di misurare l'interdipendenza per ognuna delle regioni in esame.

Il LISA (Local Indicator of Spatial Association) consente in modo efficace di associare ad ogni unità territoriale una misura del livello di associazione spaziale rispetto al suo intorno. Ovviamente si assume che la somma dei diversi LISA per ciascuna regione sia proporzionale alla corrispondente misura globale. L'indicatore solitamente utilizzato come LISA è lo stesso I di Moran calcolato a livello locale.

Anche in questo caso l'ipotesi nulla è l'assenza di autocorrelazione spaziale per cui se il test, che si distribuisce come una normale standardizzata, ha valori significativamente positivi, avremo un cluster di regioni con caratteristiche simili. Viceversa, valori significativamente negativi indicheranno un cluster di regioni diversificate. In altre parole, per ogni unità territoriale sarà possibile indicare il tipo di correlazione (negativa o positiva) e il suo livello di significatività. Anche in questo caso la rappresentazione su una mappa può essere utile per individuare ulteriori cluster di regioni con correlazioni positive significative contrapposte a cluster di regioni con correlazioni negative significative. Potranno invece essere escluse quelle correlazioni che non raggiungeranno il livello di significatività che venivano invece riportate nel mappa del Moran Scatterplot.

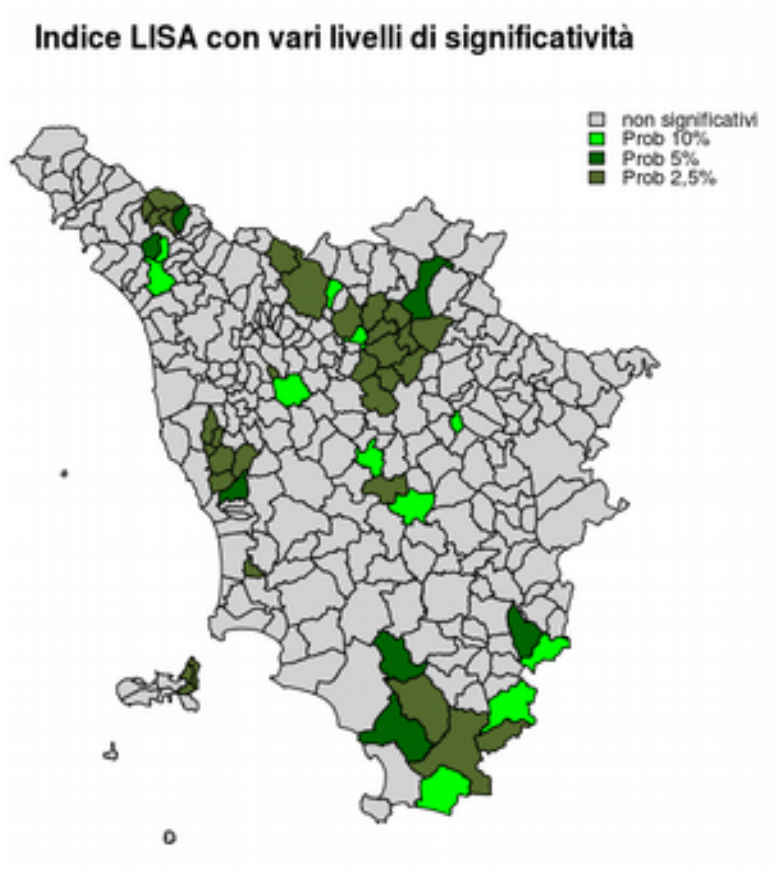
Per calcolare LISA la funzione è `localmoran`, i risultati della variabile I sono contenuti nella prima colonna dell'oggetto `lm` prodotto.

```
lm<-localmoran(comuni$REDDITO, listw=nb2listw(comuni.Qnb, style="B"))
summary(lm)
### STOP
# Contributi locali alla statistica I
comuni@data$Zlm<-lm[,1]
cuts<-c(-6.5,-0.3786,0.2406,1.9767,46,68)
# N.B.: I tagli sono posizionati sui quartili
spplot(comuni, 'Zlm', colorkey=list(labels=list(at=cuts)),at=cuts)
### STOP
# Livelli di significatività
plm<-rep(1, length(lm))
plm[lm[,5]<0.1]<-2
plm[lm[,5]<0.05]<-3
plm[lm[,5]<0.025]<-4
plot(comuni, col=c("grey", "green", "blue","red")[plm])
legend("topright", legend=c("non significativi", "Prob 10%", "Prob 5%", "Prob
  2,5%"), fill=c("grey", "green", "blue", "red"), bty="n", cex=0.8, y.intersp=0.8)
title('Indice LISA con vari livelli di significatività')
```

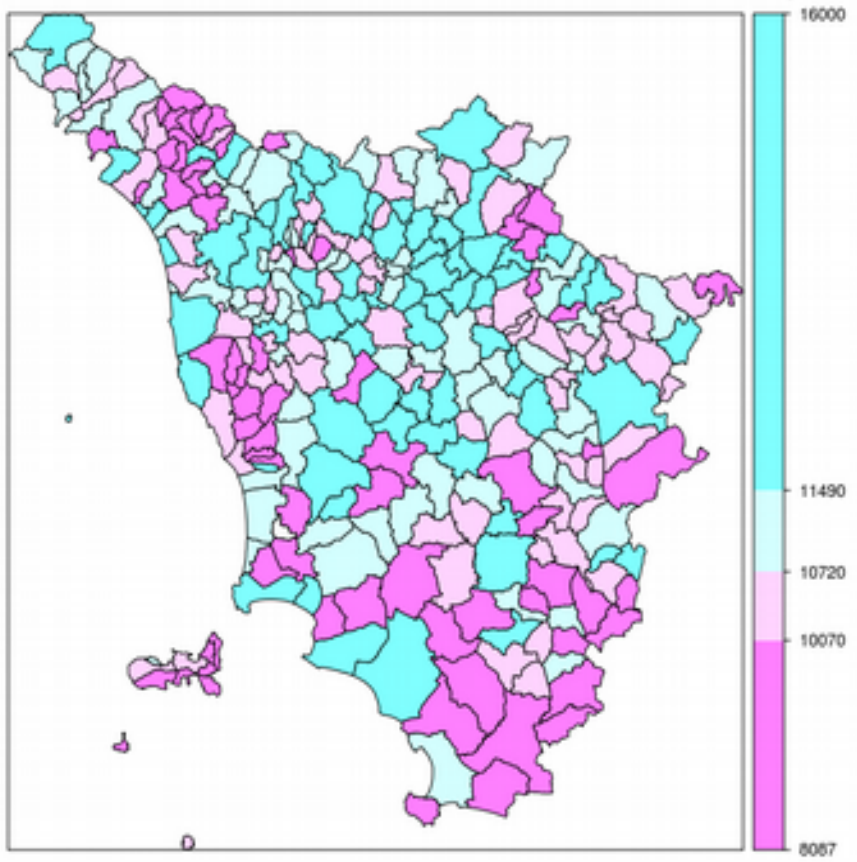


I livelli di significatività sono contenuti nella quinta colonna della matrice dati dell'oggetto `lm`:

```
# Livelli di significatività
comuni@data$PrI<-lm[,5]
plm<-rep(1, length(lm))
plm[lm[,5]<0.1]<-2
plm[lm[,5]<0.05]<-3
plm[lm[,5]<0.025]<-4
plot(comuni, col=c("lightgrey", "green", "darkgreen", "darkolivegreen")[plm])
legend("topright", legend=c("non significativi", "Prob 10%", "Prob 5%", "Prob
   2,5%"), fill=c("lightgrey", "green", "darkgreen", "darkolivegreen"), bty="n",
   cex=0.8, y.intersp=0.8)
title('Indice LISA con vari livelli di significatività')
```



Confrontando le due mappe precedentemente ottenute con la seguente, riportante i livelli di reddito, anch'essi tagliati sui quartili, è possibile individuare cluster di comuni omogenei ad alto e basso reddito. Per esempio casi di clusters significativi omogenei di comuni ad alto reddito si hanno intorno a Firenze e a Pistoia; mentre cluster significativi, omogenei, a basso reddito si hanno a sud di Grosseto, per i comuni pisani al confine con il nord della provincia di Livorno ed in Garfagnana.




```
# La mappa precedente è stata ovviamente otteunta come segue:
cuts<-c(8087,10070,  10720 ,  11490 ,  16000)
spplot(comuni, 'REDDITO', colorkey=list(labels=list(at=cuts)),at=cuts)
```

La regionalizzazione (o clustrering geografico): il metodo SKATER

Per regionalizzazione si intende la creazione di oggetti geografici formati da l'unione di elementi contigui simili per uno o più caratteristiche. La creazione di zone geografiche il più possibile omogenee è tipico di molte procedure di pianificazione in ambito territoriale, rurale e ambientale.
Sono state proposte molte procedure per tentare di risolvere il problema (spesso conflittuale) di avere zone omogenee per caratteristiche, più o meno correlate spazialmente, il più possibile fra loro contigue. Le tipologie principali sono le seguenti:

1) metodi di clustering statistico non geografico seguito da procedure di “accorpamento” geografico a posteriori (generalizzazione spaziale)
2) metodi di clustering statistico non geografico con l'introduzione di uno o più parametri di vicinanza fra i dati
3) metodi euristici recursivi (basati su processi *trial and error*)
4) metodi di clustering vincolati spazialemente.

L'ultima categoria è senz'altro più moderna ed ha perciò ricevuto la maggior attenzione da parte della recente letteratura; uno degli approcci che ha avuto maggiore applicazione è il metodo SKATER (Spatial 'K'luster Analysis by Tree Edge Removal).

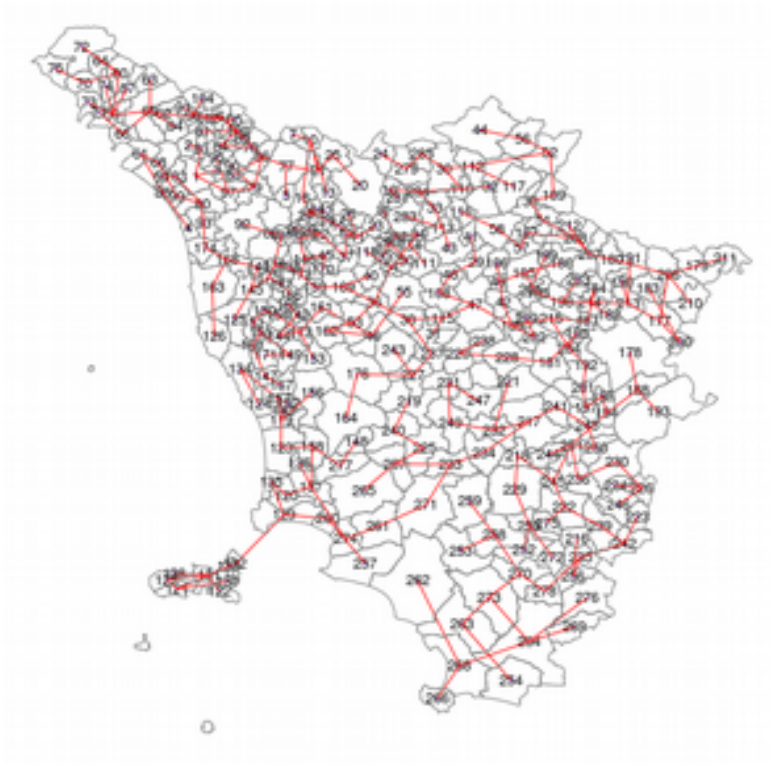
Anche il metodo SKATER impiega il concetto di grafo di connettività per identificare le condizioni di adiacenza spaziale fra oggetti, come illustrato nei paragrafi precedenti.
Per poter però considerare i rapporti di connettività basati non su di una sola caratteristica ma su più caratteri, SKATER associa ad ogni collegamento un indice di similarità/dissimilarità basato sul concetto di distanza multidimensionale pesata.
Successivamente, per ridurre la complessità del grafico di connettività, si eliminano progressivamente le connessioni fra oggetti a più elevata dissimilarità, fino ad ottenere un grafo che riesca a connettere tutti gli oggetti con percorsi complessivamente caratterizzati dal minor diversità possibile. Ina altre parole la diversità fra due oggetti connessi rappresenta il "costo che si deve pagare per muoversi da un oggetto all'altro": si definisce "minimum spanning tree" (MST) quel grafico che consente di collegare tutti gli oggetti pagando il minor costo possibile.
Infine tagliando progressivamente l'MST eliminando i rami residui a più alta dissimilarità, si ottengono infine oggetti separati caratterizzati dalla massima possibile omogeneità interna e dalla più alta possibile eterogeneità fra gli agglomerati ottenuti.

Nell'esempio che segue cercheremi di creare zone omogenee fra i comuni italiani sulla base delle seguenti caratteristiche socioeconomiche: PIL, reddito pro capite, percentuale popolazione residente nei centri urbani, percentuale dei laureati, percentuale disoccupati e densità della popolazione. In R è possibile creare un *minimum spanning tree* con la seguente procedura:

```
library(spdep)
### Normalizziamo i dati per poter creare un indice multidimensionale di
  similarità/dissimilarità
dpad <- data.frame(scale(comuni@data[,c(4,6,14,15,16)]))
summary(dpad)
### ricreiamo per sicurezza la lista delle relazioni di contiguità, teoricamente
  con un qualsiasi metodo che permetta di connettere TUTTI gli oggetti
IDs<-comuni$ISTAT
comuni.nb<-knn2nb(knearneigh(coords,k=8),row.names=IDs)
### la funzione che segue calcola il "costo di dissimilarità" con il metodo della
  distanza euclidea
lcosts <- nbcosts(comuni.nb, dpad, method='euclidean')
### i pesi sono calcolati sulla base di questo costo.
nb.w <- nb2listw(comuni.nb, lcosts, style="B")
### si calcola, con apposita funzione il minimum spanning tree
mst.bh <- mstree(nb.w,5)
class(mst.bh)
head(mst.bh)
```

Ovviamente mst.bh è un oggetto di classe `mst`. Di fatto questo è una matrice in cui ogni riga rappresenta una connessione, i tre dati associati sono i codici dei nodi connessi e il costo di connessione. L'oggetto mst si può ovviamente plottare.

```
### the mstree plot
plot(mst.bh, coordinates(comuni), col=2, cex.lab=.7, cex.circles=0.035, fg="blue")
plot(comuni, border=gray(.5), add=TRUE)
```



I cluster geografici sono ottenuti tagliando il grafo nei punti ad alta dissimilarità, in maniera simile a quanto avviene nella cluster analysis non spaziale. Il metodo SKATER consente di specificare il numero desiderato di cluster, la dimensione minima di ciascun cluster in termini di oggetti (numero di comuni nel nostro caso) o in termini di minimo valore della somma di un attributo (per esempio superfice agricola minima del cluster).

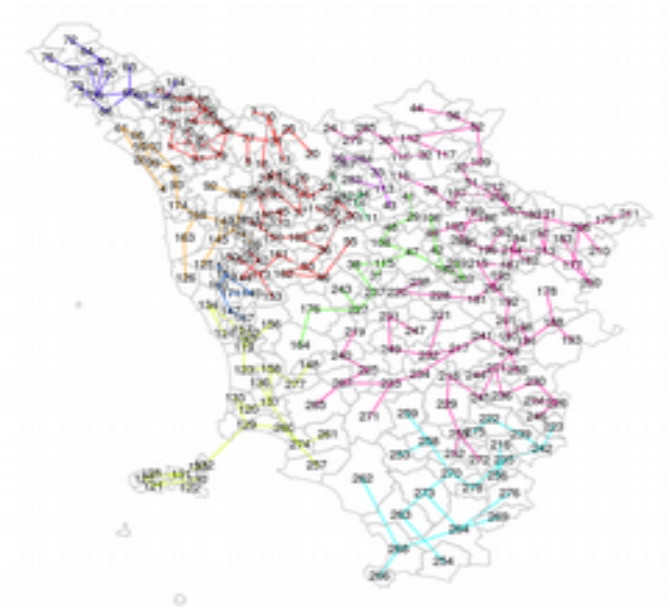
Di seguito un esempio con le seguenti caratteristiche:

1. dissimilarità misurata tramite la distanza euclidea fra gli attributi;
2. 10 cluster ciascuno composto da un minimo di 5 comuni;

```
### ncuts è il numero di agglomerati che si vogliono ottenere; method è il metodo
  di raggruppamento, ncuts numero minimo di comuni in un agglomerato
res1 <- skater(mst.bh[,1:2], dpad, method='euclidean', ncuts=9, crit=5)
```

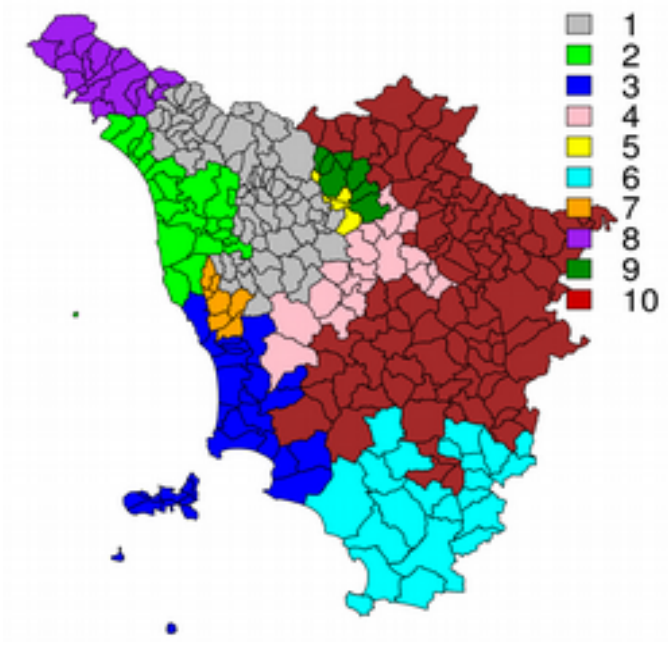
Anche res1 è un oggetto (tipo skater) piuttosto complesso, costituito da una lista di altri oggetti. Il più interessante è lo slot `res1$groups` che rappresenta un vettore, con lo stesso ordine del data.frame originario dei comuni contenente il codice del gruppo a cui appartiene l'oggetto. L'oggetto si può plottare, visualizzando i tagli che si ottengono nel minimum spanning tree:

```
### the skater plot
plot(res1, coordinates(comuni), cex.circles=0.035, cex.lab=.7)
plot(comuni, border='gray', add=TRUE)
```



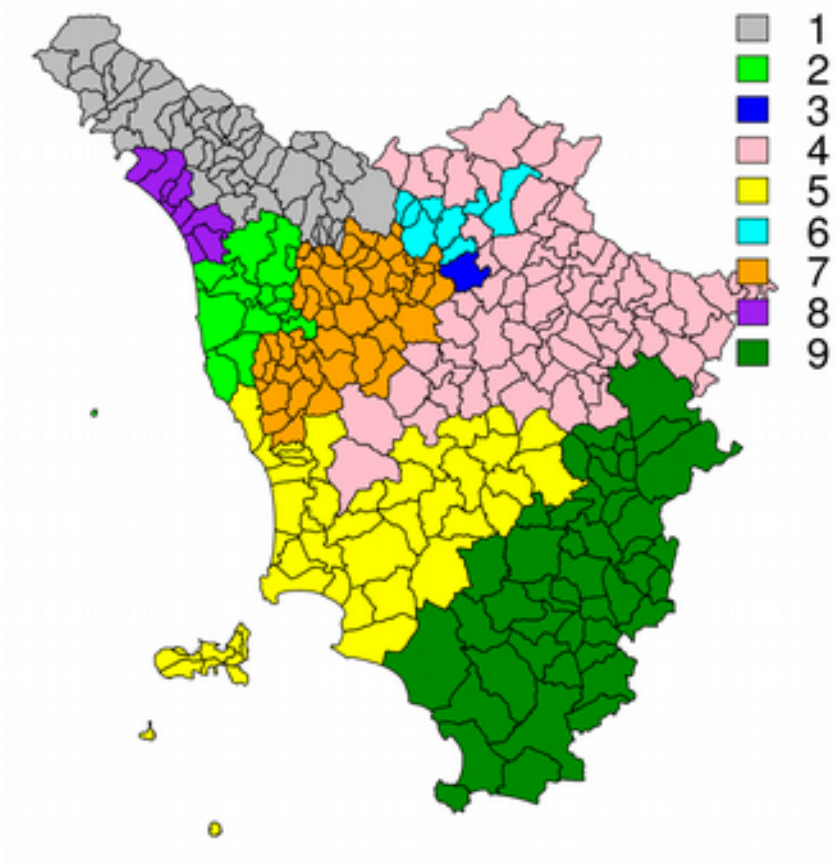
Per poter meglio visualizzare i gruppi ottenuti è possibile sia associare al dataframe comuni i gruppi ottenuti, sia plottarli in una mappa.

```
plot(comuni, col=c("grey", "green", "blue", "pink", "yellow", "cyan", "orange",
   "purple", "green4", "brown")[res1$groups])
legend("topright", legend=c('1', '2', '3', '4', '5', '6', '7', '8', '9', '10'),
   fill=c("grey", "green", "blue", "pink", "yellow", "cyan", "orange", "purple",
   "green4", "red3", "brown"), bty="n", cex=2, y.intersp=0.8)
title("SKATER")
```



I criteri di dimensione minima degli oggetti possono anche essere espressi sulla base di un parametro più complesso, come, per esempio, quello avere agglomerati che comprendano almeno il 7% della popolazione regionale.

```
res2 <- skater(mst.bh[,1:2], dpad, method='euclidean', ncuts=8, crit=7,
  vec.crit=comuni$POP/sum(comuni$POP)*100)
table(res2$groups)
comuni@data$gruppi<-as.factor(res2$groups)
colori<-c("grey", "green", "blue", "pink", "yellow", "cyan", "orange", "purple",
  "green4", "brown")
legenda<-c('1', '2', '3', '4', '5', '6', '7', '8', '9','10')
plot(comuni, col=colori[1:(max(res2$groups))][res2$groups])
legend("topright", legend=legenda[1:(max(res2$groups))], fill=c("grey", "green",
  "blue", "pink", "yellow", "cyan", "orange", "purple", "green4", "red3"),
  bty="n", cex=2, y.intersp=0.8)
title("gruppi con minimo 7% popolazione")
```



Un esempio di statistica spaziale con limiti non amministrativi è il caso della elaborazione di dati a griglia. Fra le griglie, un caso particolarmente interessante è rappresentato dalle griglie esagonali, in quanto ogni elemento ha distanza uguale dagli oggetti che geograficamente lo circondano.
Le griglie (in particolare quelle esagonali) hanno particolare interesse per studiare fenomeni diffusi, come quelli ambientali. L'esempio che segue è applicato allo studio delle relazioni spaziali per la valutazione della distribuzione geografica della multifunzionalità in agricoltura.
Le funzioni considerate sranno le seguenti:

- `idoneità territoriale (IT) per l'agricoltura di qualità`
- IT per l'agriturismo
- IT per gli interventi agroambientali da parte delle aziende

- funzione naturalistica del territorio
- funzione paesaggistica
- `importanza periurbana dell'agricoltura`

Le seguenti righe importano mappe raster di tali funzioni e creano uno SpatialGridDataFrame.

```
setwd('/home/unigis/DATI_E_DOCUMENTI_GIS/geostatistica/R_dati/hex')
library(rgdal)
agriqual<-readGDAL('f_agriqual')
names(agriqual)<-'agriqual'
agritur<-readGDAL('f_agritur')
names(agritur)<-'agritur'
agroamb<-readGDAL('f_agroamb')
names(agroamb)<-'agroamb'
amb<-readGDAL('f_amb')
names(amb)<-'amb'
pae<-readGDAL('f_pae_rur')
names(pae)<-'pae'
periurb<-readGDAL('f_periurb')
names(periurb)<-'periurb'
multidim<-cbind(agriqual,agritur,agroamb,amb,pae,periurb)
```

Le righe seguenti creano una 'tassellazione' eagonale sulla base dell'oggetto multidim

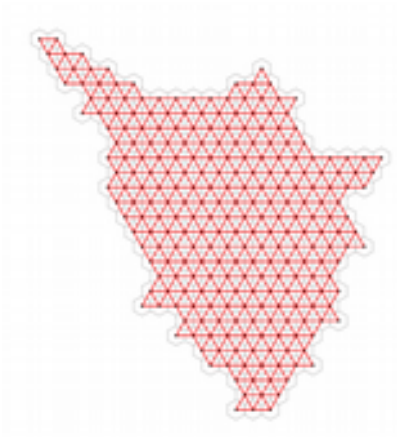
```
hex_p<-spsample(multidim, type='hexagonal', cellsize=10000)
hex_poly<-HexPoints2SpatialPolygons(hex_p)
plot(hex_poly)
```

Successivamente si associa a ciascun oggetto della griglia il database contenuto nell'oggetto `multidim`.

```
multidim$x<-coordinates(multidim)[,1]
multidim$y<-coordinates(multidim)[,2]
multidim.grid<-data.frame(multidim)
gridded(multidim.grid)=~x+y
dati<-as.data.frame(multidim.grid)[overlay(multidim.grid,hex_p),]
hex_dati<-SpatialPolygonsDataFrame(hex_poly, dati, match.ID=FALSE)
summary(hex_dati)
```

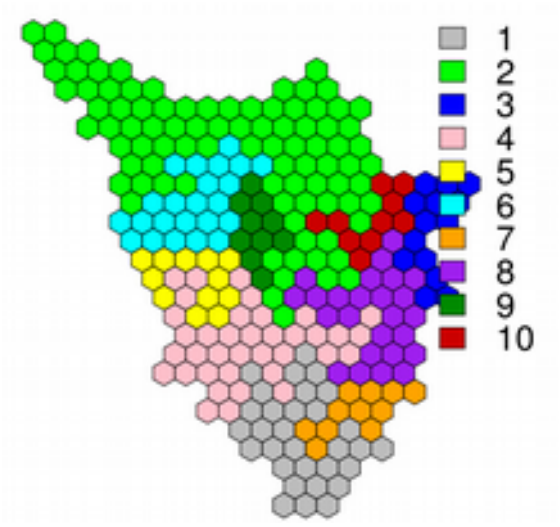
La costruzione del modello territoriale è semplice. In caso di elementi isolati è possibile connettere tutti i tasselli con una knn di grado 6, ricordando che ogni esagono a 6 confinanti.

```
library(spdep)
hex.nb<-poly2nb(hex_dati)
plot(hex_dati, border="grey")
plot(hex.nb, coordinates(hex_dati), col="red", add=TRUE, cex=0.3)
```



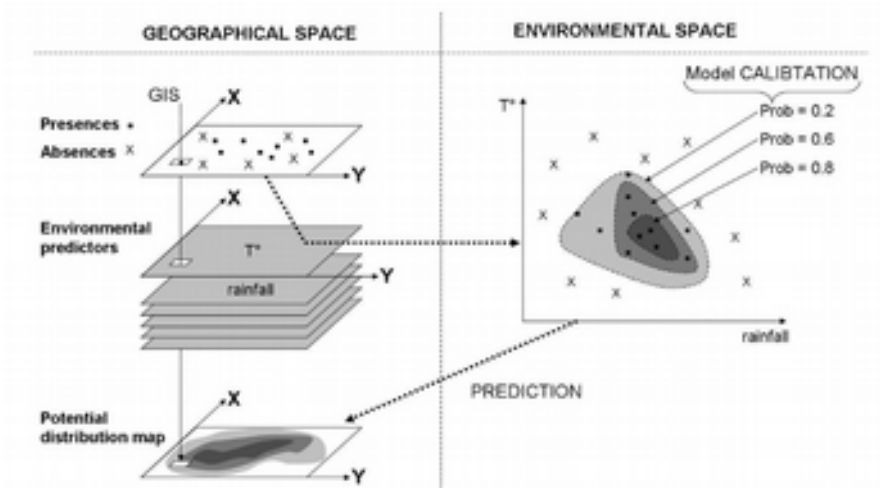
E' ora possibile esaminare tutte dipendenze spaziali, per esempio (il più complesso, tramite una regionalizzazione SKATER.

```
dpad <- data.frame(scale(hex_dati@data[,c(1:6)]))
lcosts <- nbcosts(hex.nb, dpad)
nb.w <- nb2listw(hex.nb, lcosts, style="B")
mst.hex <- mstree(nb.w,5)
res.hex <- skater(mst.hex[,1:2], dpad, method="euclidean", ncuts=9, crit=10)
table(res.hex$groups)
plot(res.hex, coordinates(bh), cex.circles=0.035, cex.lab=.7)
plot(hex_dati, col=c("grey", "green", "blue", "pink", "yellow", "cyan", "orange",
"purple", "green4", "red3", "brown")[res.hex$groups])
legend("topright", legend=c('1', '2', '3', '4', '5', '6', '7', '8', '9', '10'),
fill=c("grey", "green", "blue", "pink", "yellow", "cyan", "orange", "purple",
"green4", "red3", "brown"), bty="n", cex=2, y.intersp=0.8)
title("SKATER esagonale")
```



L'elaborazione statistica di dati spaziali: una applicazione ai modelli bioclimatici di distribuzione delle specie

I modelli di distribuzione delle specie (MDS), conosciuti anche come modelli di nicchia, modelli di habitat o envelope-model, si basano sull'ipotesi di poter predirre la distribuzione potrenziale spaziale di un fenomeno biologico mettendo in relazione la localizzazione dell'occorrenza di tale fenomeno (e la non-occorrenza) tramite variabili geografiche predittive che siu suppone siano correlate alle esigenze ecologiche della specie. Le più comuni applicazioni di tali modelli sono la predizione della distribuzione delle specie e la valutazione d'impatto dei cambiamenti di habitat, per esempio per effetto dei cambiamenti climatici.
Le fasi di costruzione di un modello di analisi spazaile sono le seguenti.
a) individuazione della localizzazione dell'occorrenza diella specie o del fenomeno
b) individuazione delle variabili ambientali e climatiche tramite un database spaziale
c) costruzione di un modello di predizione probabilistica di presenza della specie



I modelli di MDS sono implementati in R in molti modi. Nella presente applicazione ci riferiremo prevalentemente ai pacchetti **`dismo`** e **`raster`** :

```
install.packages(c('raster', 'dismo'))
library(dismo)
setwd('C:/SummerLab2013/R_dati')
```

Preparazione dei dati

Per gli esempi si useranno i seguenti databases: Inventario Forestale Toscano e il geodatabase worldclim. Worlclim è un set di layer cartografici contenenti dati climatici e scenari di cambiamento climatico alla risoluzione di circa 1km quadro. I layer contengono interpolazioni di dati mensili di precipitazione totale, temperatura minima, massima e media e 19 variabili bioclimatiche derivate.
Gli indici climatici sono i seguenti.

BIO1 = Annual Mean Temperature

BIO2 = Mean Diurnal Range (Mean of monthly (max temp - min temp))
BIO3 = Isothermality (BIO2/BIO7) (* 100)
BIO4 = Temperature Seasonality (standard deviation *100)
BIO5 = Max Temperature of Warmest Month
BIO6 = Min Temperature of Coldest Month
BIO7 = Temperature Annual Range (BIO5-BIO6)
BIO8 = Mean Temperature of Wettest Quarter
BIO9 = Mean Temperature of Driest Quarter
BIO10 = Mean Temperature of Warmest Quarter
BIO11 = Mean Temperature of Coldest Quarter
BIO12 = Annual Precipitation
BIO13 = Precipitation of Wettest Month
BIO14 = Precipitation of Driest Month
BIO15 = Precipitation Seasonality (Coefficient of Variation)
BIO16 = Precipitation of Wettest Quarter
BIO17 = Precipitation of Driest Quarter
BIO18 = Precipitation of Warmest Quarter
BIO19 = Precipitation of Coldest Quarter

```
# Supponendo di aver archiviato i files scaricati da worldclim in una cartella
  C:/clima
# files <- list.files(path='C:/clima',full.names=TRUE )
# tramite la funzione stack del pacchetto raster creiamo un oggetto raster
  multibanda dai files
# temp<-stack(files)
# tagliamo il geodatabase
# e<-extent(9.7,12.5,42.3,44.6)
# temp2<-crop(temp,e)
# convertiamo in spatial grid data frame
# bio<-as(temp2,'SpatialGridDataFrame')
rm(temp2)
rm(temp)
load('IFT.Rdata')
load('bio.Rdata')

summary(IFT)
summary(bio)
# struttura del dataset
# Il primo passaggio consiste nel selezionare la specie da esaminare, nell'esempio
  il pino nero:
Specie<-subset(IFT,IFT$COMPSPE1==76)
```

La maggior parte dei MDS richiede non solo dati di presenza delle specie ma anche localizzazioni i n cui la specie non è presente, i cosiddetti dati di assenza ricavati tramite una procedura di selezione random di punti inventariali non a pino nero.

```
notSpecie<-IFT[IFT@data$COMPSPE1!=76,]
notSpecie<-notSpecie[sample(nrow(notSpecie@data),nrow(Specie@data)),]
Specie$pres<-1
notSpecie$pres<-0
TotSpecie<-rbind(Specie,notSpecie)
```

Rientra nella preparazione del dato la suddivisione dell'insieme delle oserevazioni in due sottoinsiemi, uno dei quali sarà impiegato per la costruzione del modello, l'altro per la stia dell'efficienza della procedura, secondo la procedura della "cross-validazione". dei set di informazioni utilizzate per la stima del La cross-validazione è un metodo statistico per validare un modello predittivo. Preso un campione di dati, esso viene suddiviso in sottoinsieme alcuni dei quali vengono usati per la costruzione del modello (insiemi di allenamento, training sets) e gli altri da confrontare con le predizioni del modello (insiemi di validazione, *validation sets*). Mediando la qualità delle predizioni tra i vari insiemi di validazione da una misura dell'accuratezza delle predizioni.

```
# Creazione del testing e del training sets
#
Specie$folds<-0
Specie$folds<-kfold(Specie,5)
# con l'istruzione kfold si assegna casualemnte ad ogni osservazione un numero
  compreso fra 1 e il secondo parametro, in questo caso 5
notSpecie$folds<-0
notSpecie$folds<-kfold(notSpecie,5)
# l'istruzione seguente genera il training set, costituito mdall'80% delle
  osservazioni
TrainSpecie<-rbind(Specie[Specie@data$folds!=1,],notSpecie[notSpecie@data$folds!
  =1,])
# si costruisce il testing set, con il 20% delle osservazioni
TestSpecie<-
  rbind(Specie[Specie@data$folds==1,],notSpecie[notSpecie@data$folds==1,])
```

Sovrapponendo l'oggetto SpatialPointDataFrame della specie selezionata (in questo caso il pino nero) con lo SpatialGridDatFrame dei dati bioclimatici si ottiene uno SpatialPointDataFrame con i dati bioclimatici corrispondenti alla localizzazione del punto.

```
TrBioSpecie<-overlay(bio,TrainSpecie)
TrBioSpecie$pres<-TrainSpecie$pres
TeBioSpecie<-overlay(bio,TestSpecie)
TeBioSpecie$pres<-TestSpecie$pres
summary(TrBioSpecie)
summary(TeBioSpecie)
```

Per la stima dei MDS ci sono molti metodi. La caratteristica comune a quelli che analzzeremo nella dispensa è che il risultato ottenuto può essere interpretato come una mappa di probabilità di avere un habitat potenzialmente idoneo alla specie considerata.

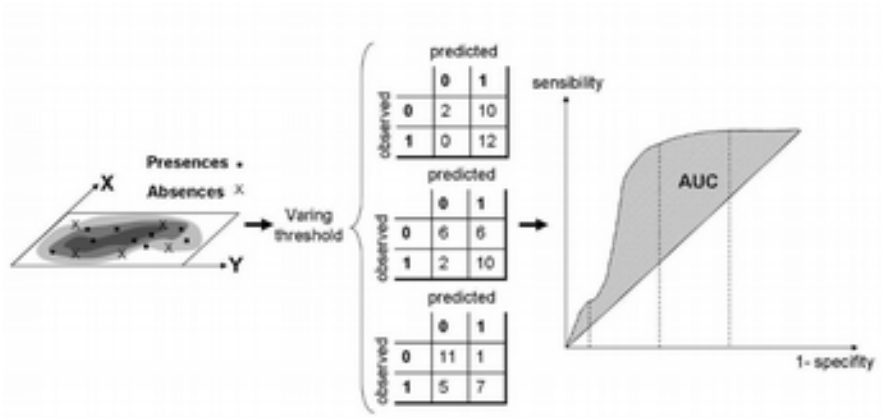
```
Il modello più "semplice" per ottenere tale risultato è la regressione logistica.
La regressione logistica è un caso particolare di modello lineare generalizzato
avente come funzione link la funzione logit. Si tratta di un modello di regressione
applicato nei casi in cui la variabile dipendente y sia di tipo dicotomico
riconducibile ai valori 0 e 1, come lo sono tutte le variabili che possono assumere
esclusivamente due valori: vero o falso.
```

Analisi statistica, la regressione logistica

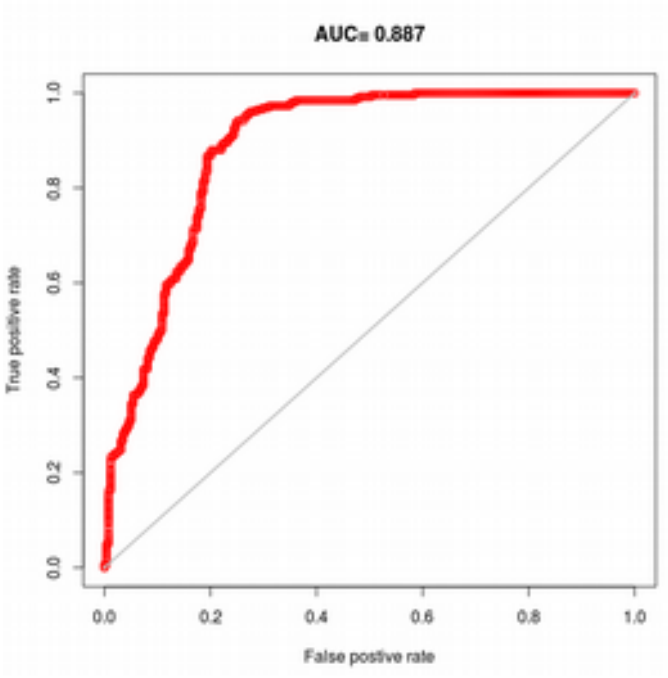
```
# Analisi statistica
```

```
# GLM logistica
#
m1<-glm(factor(pres)~., data=TrBioSpecie@data,family = 'binomial')
summary(m1)
```

Uno dei metodi di cross-validation più flessibile come campo di applicazione è la analisi ROC (Receiver Operating Characteristic). L'analisi ROC viene effettuata tramite lo studio della funzione che nel modello lega la probabilità di ottenere un risultato vero positivo nella classe delle presenze, alla probabilità di ottenere un risultato falso positivo nella classe delle assenze. Dal grafico ROC è possibile calcolare l' AUC (Area Under the Curve) o Area sottesa alla curva, la quale fornisce un'indicazione della performance del modello. Secondo la classificazione proposta da Swets (1988) il valore dell'area sottesa varia da 0.5 a 1, con un valore di 0.5 per modelli con nessuna capacità di discriminare tra presenze e assenze e con un valore di 1 per un modello di capacità discriminante perfetta.

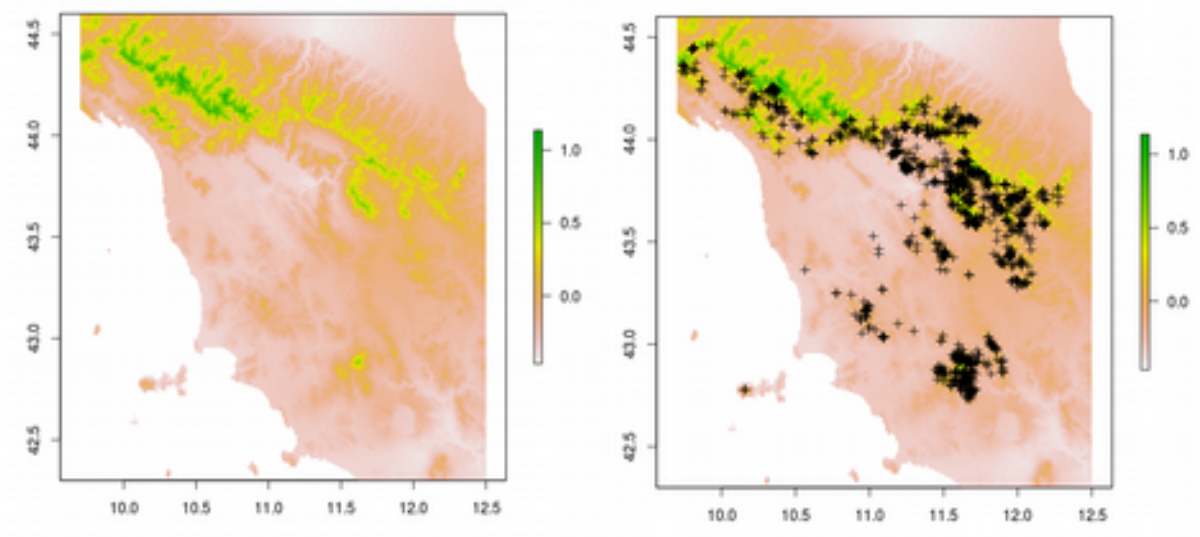



```
e=evaluate(TeBioSpecie[TeBioSpecie@data$pres==1,],TeBioSpecie[TeBioSpecie@data$pres
==0,],m1)

plot(e,'ROC')
```



Una volta costruito e validato il modello è possibile impiegarlo per costruira la mappa di probabilità di idoneità dell'habitat.

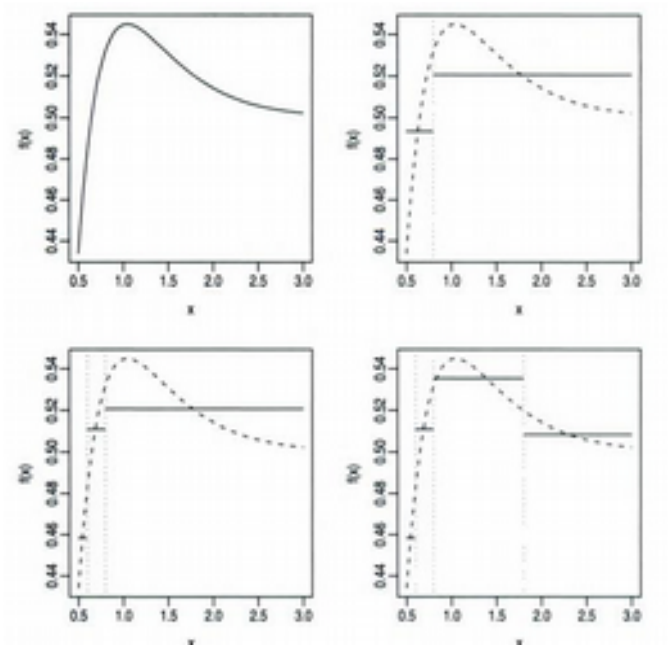
```
predictors<-stack(bio)
# il comando trasforma l'oggetto SpatialGridDataFrame in oggetto raster
pr.m1<-predict(predictors,m1,type='terms')
plot(pr.m1)
```



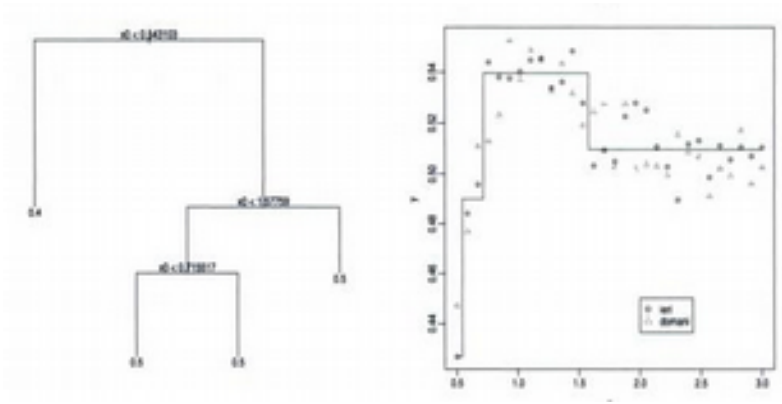
Gli alberi di regressione

Il metodo più semplice per approssimare una qualsiasi funzione $y=f(x)$ è quello di usare una funzione

approssimante a gradini, cioè una funzione costante a tratti con intervali. L'idea è illustrata nelle figure seguenti.

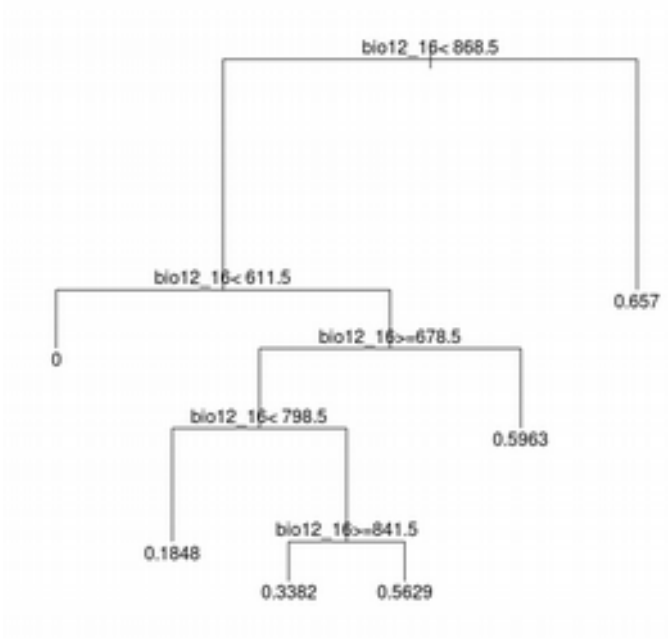


Tali caratteristiche della funzione approssimante consentono di poterla rappresentare tramiteun albero binario, costituito da una struttura le cui componenti sono affermazioni di tipo logico, dette nodi, come rappresentato nella figura seguente.



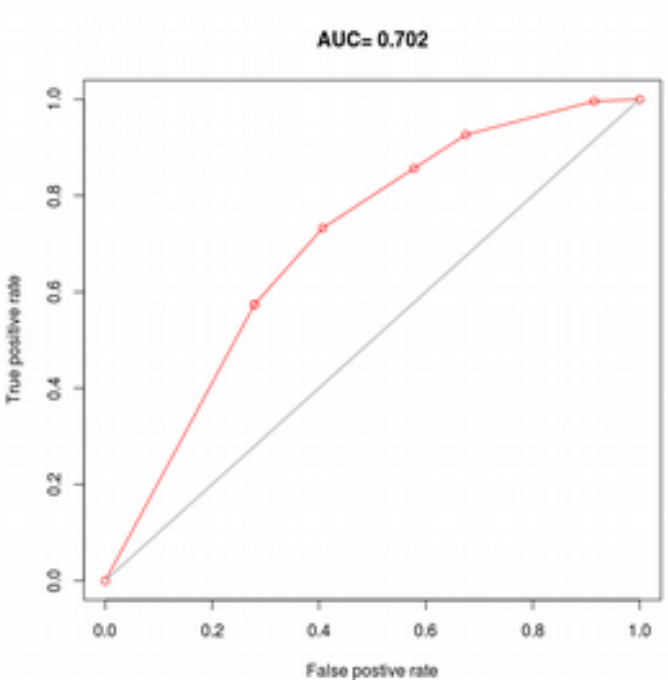
In R esistono molte librerie dedicate agli alberi di regressione, quella "storica" è `rpart`.

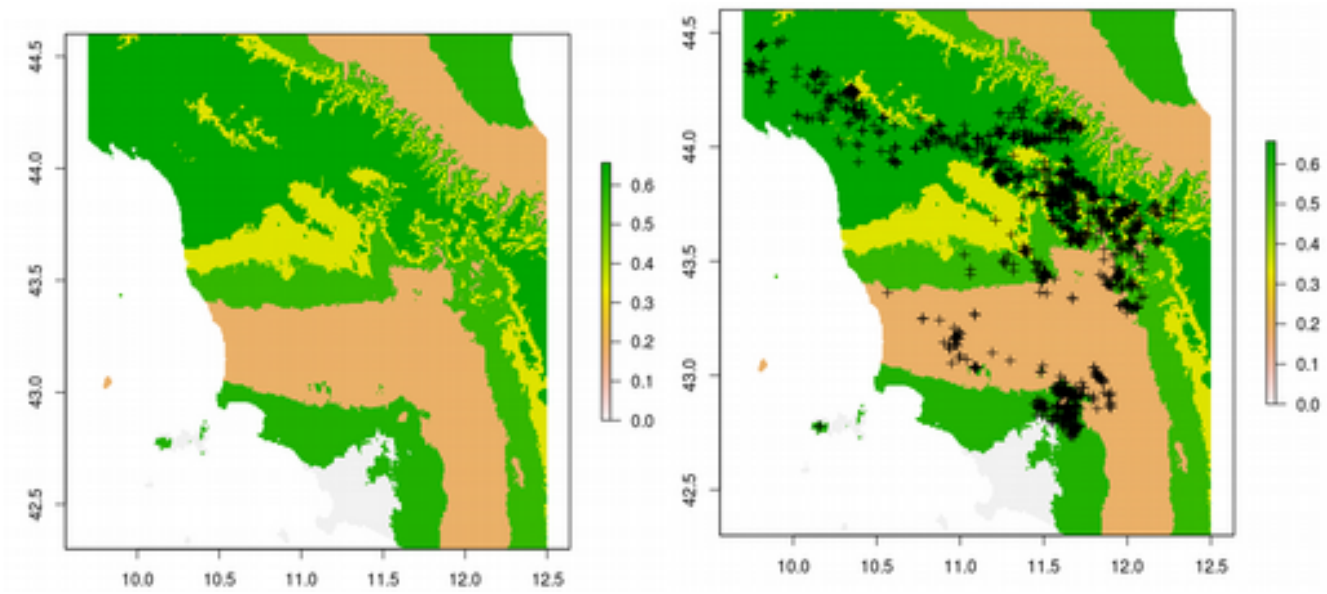
```
#
# Alberi di regressione
#
library(rpart)
t1<-rpart(pres~bio12_16,data=TrBioSpecie@data)
plot(t1)
text(t1)
```



Anche in questo caso è possibile calcolare la curva ROC e la mappa di probabilità.

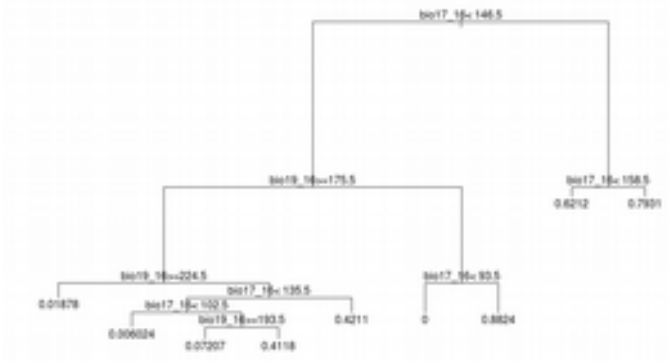
```
e=evaluate(TeBioSpecie[TeBioSpecie@data$pres==1,],TeBioSpecie[TeBioSpecie@data$pres
   ==0,],t1)
plot(e,'ROC')
predictors<-stack(bio)
pr.t1<-predict(predictors,t1)
plot(pr.t1)
```



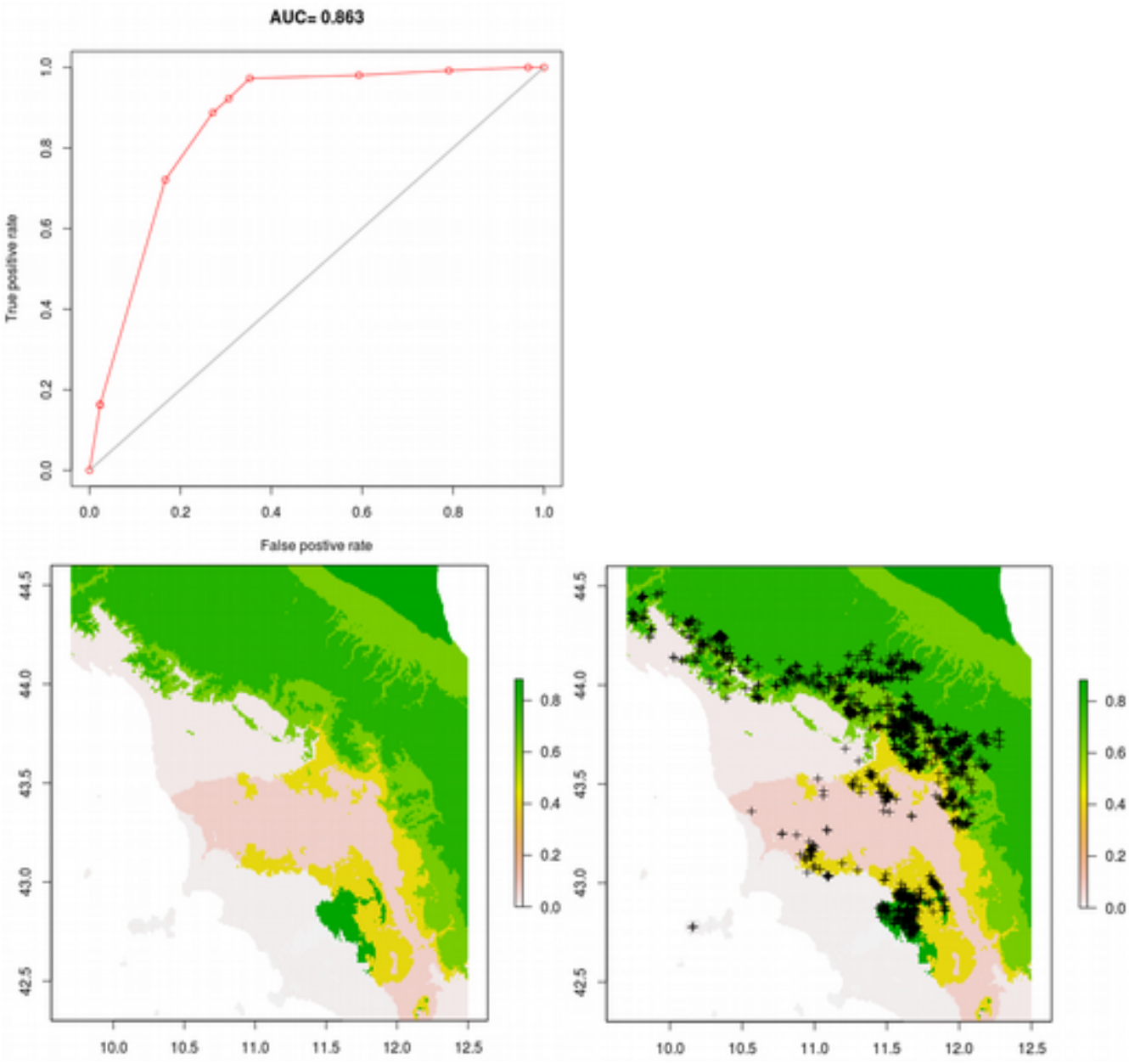


L'albero di regressione può essere anche multivariato:

```
t2<-rpart(pres~bio12_16+bio17_16+bio19_16,data=TrBioSpecie@data)
plot(t2)
text(t2)
```



```
e=evaluate(TeBioSpecie[TeBioSpecie@data$pres==1,],TeBioSpecie[TeBioSpecie@data$pres
   ==0,],t2)
plot(e,'ROC')
predictors<-stack(bio)
pr.t2<-predict(predictors,t2)
plot(pr.t2)
```

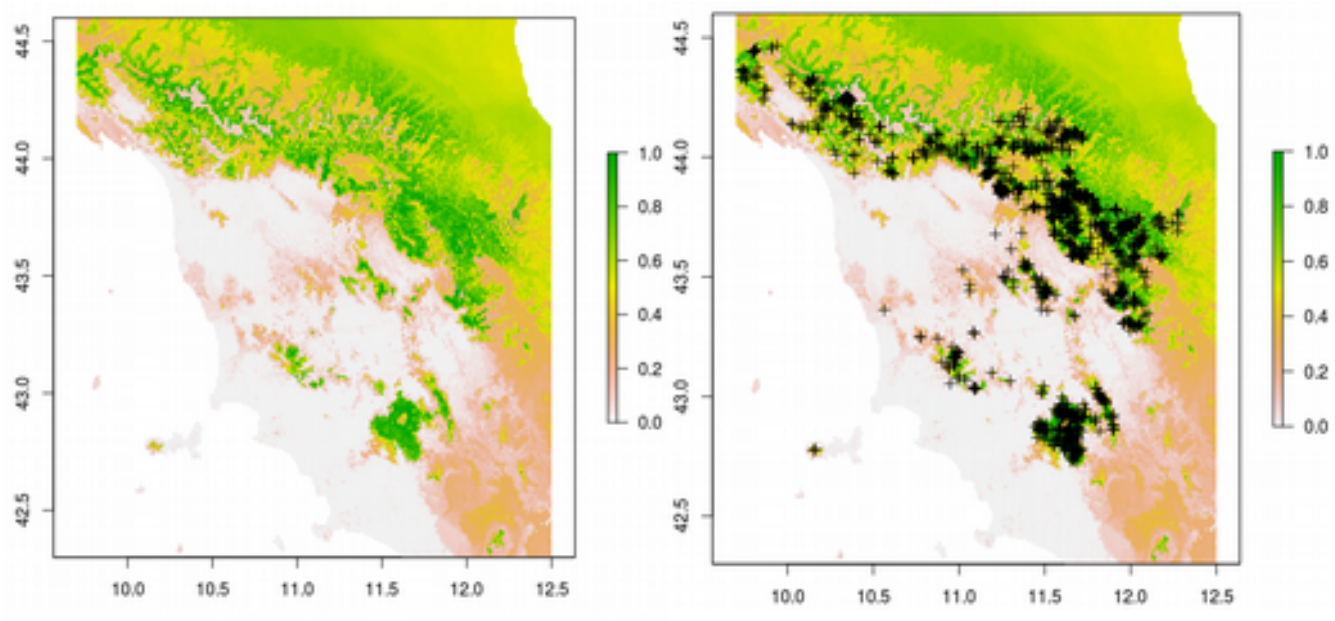


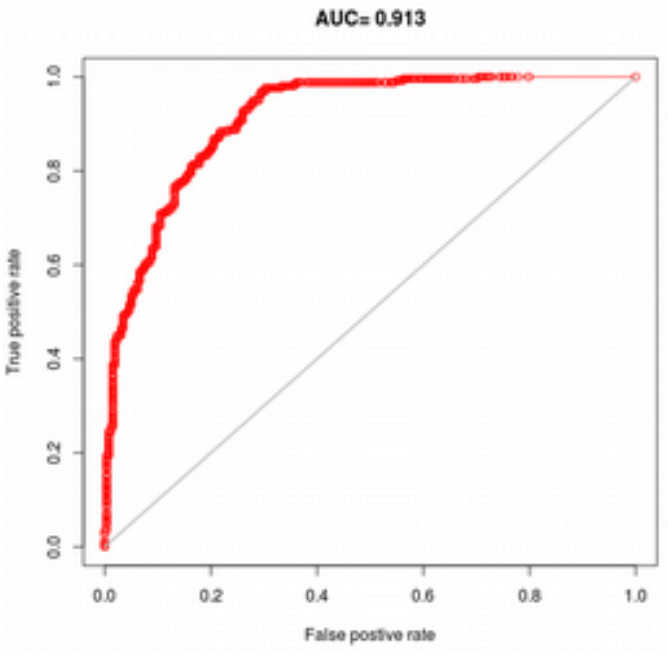
Le foreste casuali

Le Foreste Casuali (BREIMAN, 2001) sono un modello multi-albero costituito da una collezione o ensemble di alberi di classificazione. Il vantaggio nell'uso di un ensemble di alberi risiede nelle capacità predittive del modello, ben superiori a quelle di un unico albero. In una Foresta Casuale i singoli alberi vengono costruiti selezionando in maniera randomizzata campioni dalle osservazioni e dalle variabili predittrici. Nello specifico, ogni albero viene costruito utilizzando dei dataset replicati (con la tecnica del bootstrap) e un subset di variabili predittrici selezionate in maniera randomizzata. Gli alberi poi vengono combinati insieme per creare per ottenere le predizioni. Questa procedura risulta molto vantaggiosa in quanto viene annullato il rischio di overfitting ossia l'eccessivo adattamento del modello ai dati osservati, che comporterebbe la perdita di generalità e di capacità predittive del

modello.

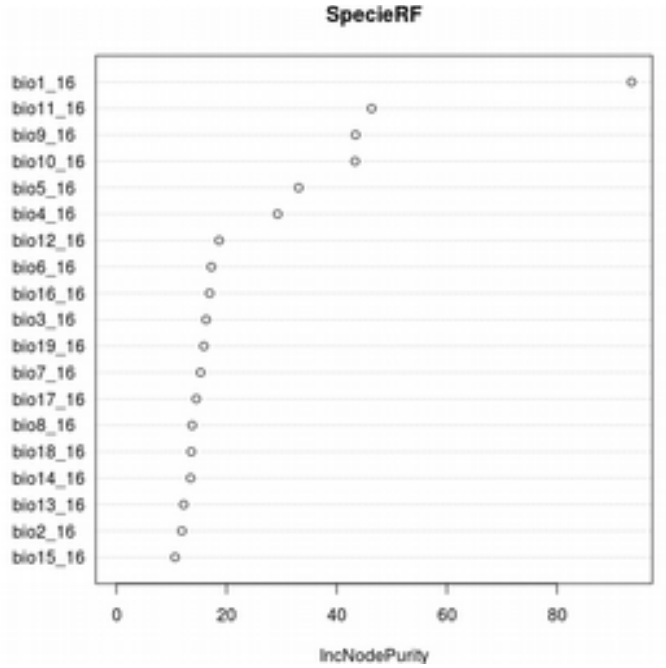
```
#
# modello Random forest
#
library(randomForest)
SpecieRF<-randomForest((pres)~.,data=TrBioSpecie@data,na.action=na.omit)
e=evaluate(TeBioSpecie[TeBioSpecie@data$pres==1,],TeBioSpecie[TeBioSpecie@data$pres
  ==0,],SpecieRF)
plot(e,'ROC')
pr.rf<-predict(predictors,SpecieRF)
plot(pr.rf)
```





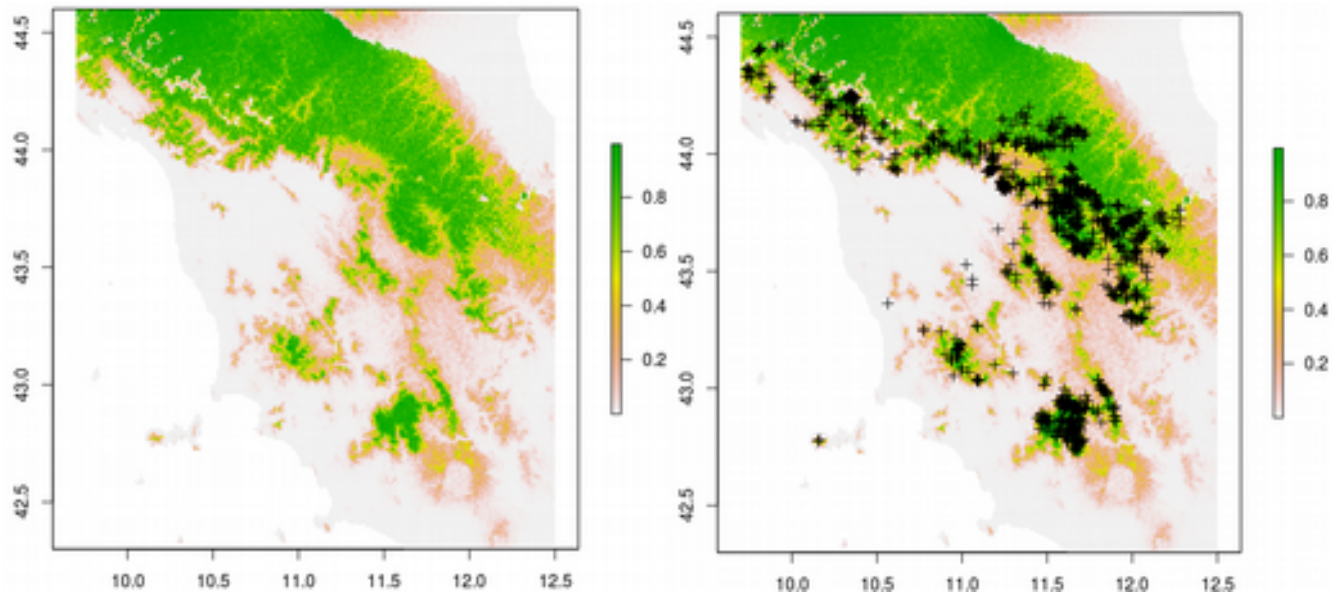
Nella costruzione del modello Foresta Casuale Breiman (2001) ha proposto vari metodi per valutare l'importanza di ogni variabile in un modello del tipo Foreste Casuali. Nel presente caso l'importanza è stata valutata tramite la riduzione del decremento medio dell'accuratezza di predizione (errore di predizione). Questo metodo si basa sull'errore di predizione di tutti i valori di ogni variabile per ogni albero. L'errore di predizione di una variabile è valutata per ogni albero attraverso la permutazione di tutti i valori di tale variabile. La differenza tra l'errore di predizione prima della permutazione e l'errore dopo la permutazione rappresenta l'importanza di una variabile.

```
varImpPlot(SpecieRF)
```



Le reti neurali

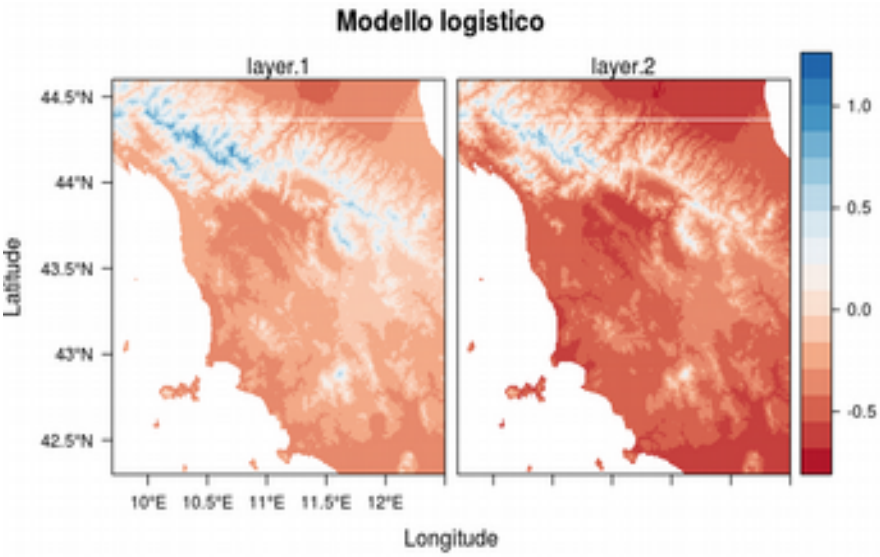
```
library(nnet)
nn1<-multinom(pres~.,TeBioSpecie@data)
e=evaluate(TeBioSpecie[TeBioSpecie@data$pres==1,],TeBioSpecie[TeBioSpecie@data$pres
==0,],nn1)
e
pr.nn1<-predict(predictors,nn1,type='probs')
plot(pr.nn1)
```



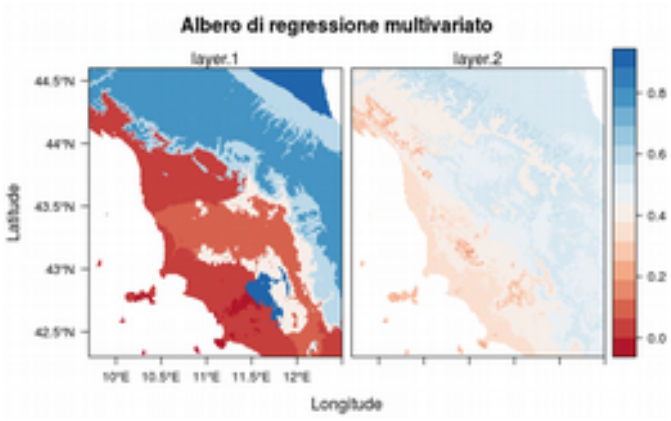
L'impiego dei MDS per valutare gli impatti dei cambiamenti climatici.

I modelli di distribuzione dell'habitat sono anche stati ampiamente utilizzati per valutare gli impatti antropici, per testare ipotesi biogeografiche e valutare gli effetti dei cambiamenti climatici (Guisan e Zimmerman, 2000; Thomas et al., 2004; Araujo et al., 2005a; 2005b; Araujo e Rahbek, 2006; Araujo e New, 2007; Lassalle et al., 2008; Algar, 2009; Anderson et al., 2009; Svenning et al., 2009). Le potenzialità predittive di questi modelli presentano sicuramente limitazioni che dipendono dai dati utilizzati e dal tipo di modello impiegato, ma rappresentano anche uno degli elementi che ha maggiormente determinato il loro successo nella comunità scientifica. Esiste infatti un ampio spettro di applicazioni proposte per valutare l'effetto di possibili cambiamenti dell'ecosistema sull'utilizzo degli habitat, che possono manifestarsi a diverse scale, dalla variazione del tenore delle precipitazioni al riscaldamento globale (Stoner et al., 2001). Molte di queste applicazioni sono incentrate sulla modellazione degli habitat di specie importanti per la conservazione (Pearce e Ferrier, 2001), e risulta quindi chiaro l'interesse che suscita la possibilità di valutare l'effetto di possibili strategie di gestione del territorio, di possibili impatti derivanti da interventi sull'ambiente e di quegli interventi progettati per il recupero di habitat disturbati o perduti (ad esempio in Burnside et al., 2002).
Nelle applicazioni sono impiegati i dati bioclimatici dello scenario IPCC4 secondo il modello HadCM3 (*Hadley Centre Coupled Model, version 3*) - SRES scenario A2A all'anno 2020.

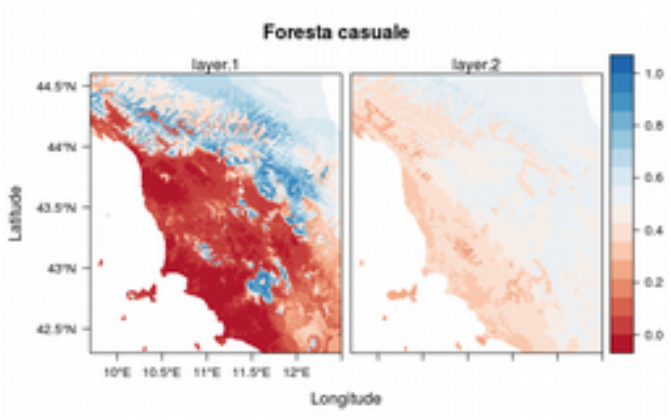
```
#`proiezione al 2020
load('bio2020.Rdata')
predictors2020<-stack(bio2020)
pr2020.m1<-predict(predictors2020,m1,type='terms')
library(rasterVis)
levelplot(stack(pr.m1,pr2020.m1),par.settings=RdBuTheme,main='Modello logistico')
```




```
pr2020.t2<-predict(predictors2020,SpecieRF)
levelplot(stack(pr.t2,pr2020.t2),par.settings=RdBuTheme,main='Albero di regressione
  multivariato')
```




```
pr2020.rf<-predict(predictors2020,SpecieRF)
levelplot(stack(pr.rf,pr2020.rf),par.settings=RdBuTheme,main='Foresta casuale')
```




```
pr2020.nn1<-predict(predictors2020,nn1,type='probs')
```

```
levelplot(stack(pr.nn1,pr2020.nn1),par.settings=RdBuTheme,main='Rete Neurale')
```

Bibliografia essenziale

Roger S. Bivand, Edzer J. Pebesma, and Virgilio Gómez-Rubio (2008) Applied Spatial Data in **R** Springer: New York, NY.

Raspa G. **Dispense di Geostatistica**, (1995), **addendum1** e **addendum2**. (Ctrl + click sul links rossi per aprire i collegamenti)