Spatial autocorrelation and neighbourhood structure

Bayesian modelling for spatial and spatio-temporal data

MSc in Epidemiology

Week 6

Recap on areal data

- The data are referenced at an aggregate level and are often defined by administrative boundaries (states, counties, districts etc.).
- Areal units are generally irregular geographic areas and in spatial epidemiology we have a collection of areal units.
- We care about how areal units connect to each other and we use neighbour information to define spatial relationships.
- In what follows, we will consider a study region to be divided into N disjunctive areas indexed by i, for $i = 1, \ldots, N$.

Spatial Autocorrelation

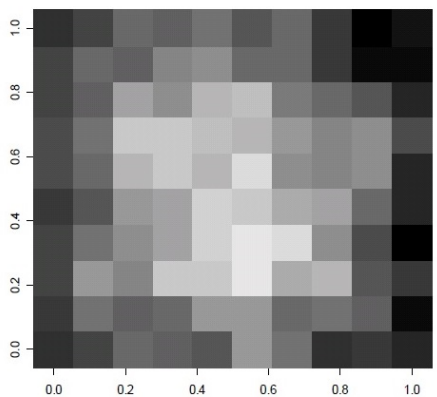
- A key insight is that observations in space cannot in general be assumed to be mutually independent, and that observations that are close to each other are likely to be similar (remember Waldo Tober's first law of geography).
- This spatial patterning, or spatial autocorrelation, may be treated as useful information about unobserved influences.
- Formally, spatial autocorrelation measures the correlation of a variable with *itself* through space. If Z_i is the attribute Z observed at location i, then the term spatial autocorrelation refers to the correlation between Z_i and Z_j.
- In other words, it quantifies the degree of which observations, at spatial locations, are similar to nearby observations.

Detecting spatial autocorrelation

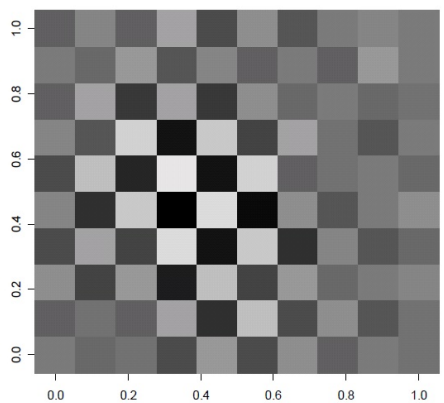
- The detection of spatial autocorrelation is useful in spatial analysis for identifying underlying data structures, the degree of spatial randomness, or clustering in the data.

- Tests of spatial autocorrelation examine whether the observed value of a variable at one location is independent of values of that variable at neighbouring locations.

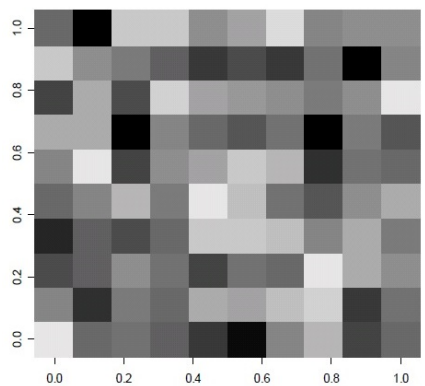
- Positive spatial autocorrelation indicates that similar values appear close to each other, or cluster, in space (figure a)
- Negative spatial autocorrelation indicates that neighboring values are dissimilar or, equivalently, that similar values are dispersed (figure b)
- Null spatial autocorrelation indicates that the spatial pattern is random (figure c)



(a)



(b)



(c)

Spatial neighbours [1]

- Key to analyse lattice (or regional) structures is the concept of *spatial connectivity* or *spatial proximity*.
- Let i and j index two members of the lattice (i.e. two locations such as two countries, or two districts etc.).
- With each pair of sites, we associate a weight w_{ij}, so that the spatial weights express the neighbour structure between the observations.

Spatial neighbours [2]

- Now, let N be the total number of areal units.
- The spatial relationship between the areas is represented as an adjacency matrix $\mathbf{W}$ with dimensions $N \times N$, where the entries w_{ij} of the matrix are the spatial weights:

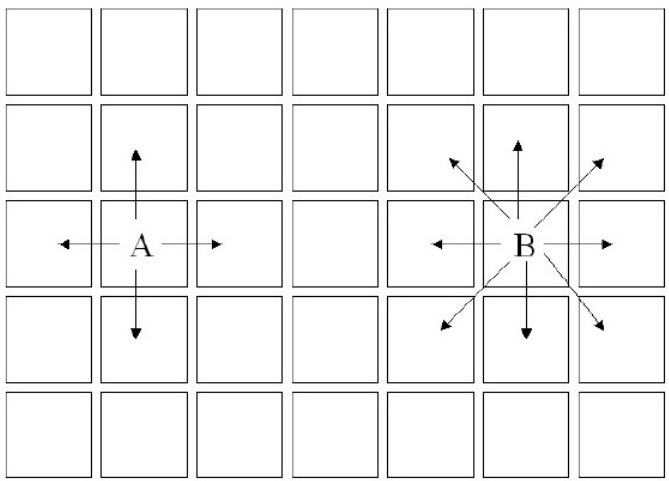
$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}$$

  - In its simple form, $w_{ij} = 1$ if areas i and j are adjacent, 0 otherwise[1].
  - The diagonal elements of $\mathbf{W}$ are zero, that is $w_{ii} = 0$.

[1] The binary (0/1) matrix $\mathbf{W}$ based on contiguity is sometime referred to as the first order contiguity weights because it define the immediate neighbours.

Weights based on contiguity

Operationally, we can distinguish between a rook (A) and a queen (B) criterion of contiguity between areas (in analogy to the moves allowed for the such-named pieces on a chess board):



The rook criterion defines neighbours by the existence of a common edge between two spatial units.
The queen criterion defines neighbours as spatial units sharing a common edge or a common vertex.

Weights based on geographical distance [1]

- Other ways of measuring the *closeness* between areas are based on the distance d_{ij} (Euclidean, Manhattan or other metrics) between units i and j (where $i \neq j$).
- Distance can be taken between geometric centroid of each polygon:
  - $w_{ij} = dij^{-1}$, inverse distance between the centroids of areas i and j
  - $w_{ij} = d_{ij}^{-\gamma}$, inverse distance raised to the power $\gamma > 0$
  - $w_{ij} = \exp(-\gamma d_{ij})$, negative exponential, with $\gamma > 0$
- The diagonal elements on **W** are equal to zero. The off-diagonal elements of **W**, will be non-zero, unless we impose a thresholds.

Weights based on geographical distance [2]

Commonly used thresholds are:

- absolute distance threshold: if $d_{ij} \geq x$ (i.e. the centroids of areas i and j are within a certain distance of each other), otherwise $w_{ij} = 0$;

- based on k nearest neighbours: only the k nearest neighbours of unit i are non-zero (in this case the weights will not be symmetric[2]).

[2] The matrix $\mathbf{W}$ is symmetric when $w_{ij} = w_{ji}$

Spatial neighbours in R

In R, we need to define:

- Neighbour connectivity (who is neighbour?)
- Neighbor weights (how much does the neighbour matter?)

To do so, we can work with the package `spdep` and we use:

- the function `poly2nb` to define neighbour connectivity according to rook or queen criterion (contiguity neighbours)
- the functions `knn2nb` and `dnearneigh` to define neighbour connectivity according to distance-based neighbours
- the function `nb2listw` to define spatial weights for neighbours lists (while `nb2mat` defines spatial weights matrices for neighbours lists)

For further details, see Roger Bivand's contribution `https://r-spatial.github.io/spdep/articles/nb.html`

Example of neighbours computation in R

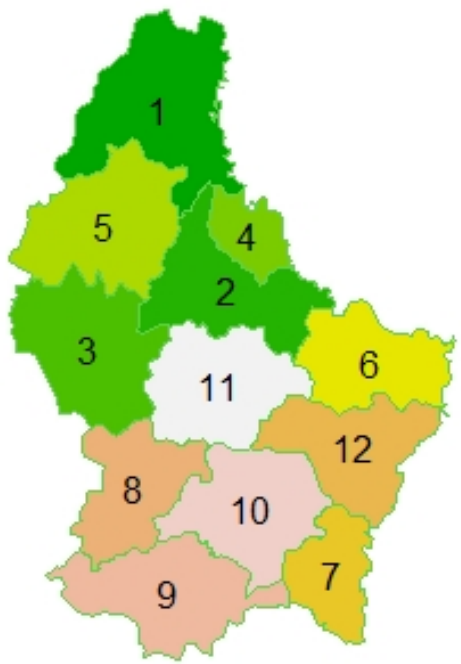
- We compute a neighbourhood structures for Luxembourg, one of the smallest country in Europe.
- We use the shapefile for Luxembourg available in the R package `raster`.
- The shapefile is the most commonly used file format for vector data. It is a collection of files with the same stem and different extensions, where the suffix for the main file is .shp:
  - *.shp contains the geometry of the object to represent
  - *.shx contains the spatial index
  - *.dbf contains the attribute data (dBASE table)
  - *.prj contains information on the CRS and the projection used to represent the geometry

```
# install.packages("raster")
# install.packages("spdep")
# remotes::install_github("r-spatial/mapview")

library(raster); library(spdep); library(mapview)

# Read in Luxembourg shapefile from package raster
lux <- shapefile(system.file("external/lux.shp", package="raster"))
summary(lux)
```

```
> summary(lux)
Object of class SpatialPolygonsDataFrame
Coordinates:
       min       max
x  5.74414  6.528252
y 49.44781 50.181622
Is projected: FALSE
proj4string : [+proj=longlat +datum=WGS84 +no_defs]
Data attributes:
      ID_1          NAME_1               ID_2          NAME_2
 Min.   :1.000   Length:12          Min.   : 1.00   Length:12
 1st Qu.:1.000   Class :character   1st Qu.: 3.75   Class :character
 Median :2.000   Mode  :character   Median : 6.50   Mode  :character
 Mean   :1.917                      Mean   : 6.50
 3rd Qu.:3.000                      3rd Qu.: 9.25
 Max.   :3.000                      Max.   :12.00
      AREA
 Min.   : 76.0
 1st Qu.:187.2
 Median :225.5
 Mean   :213.4
 3rd Qu.:253.0
 Max.   :312.0
```

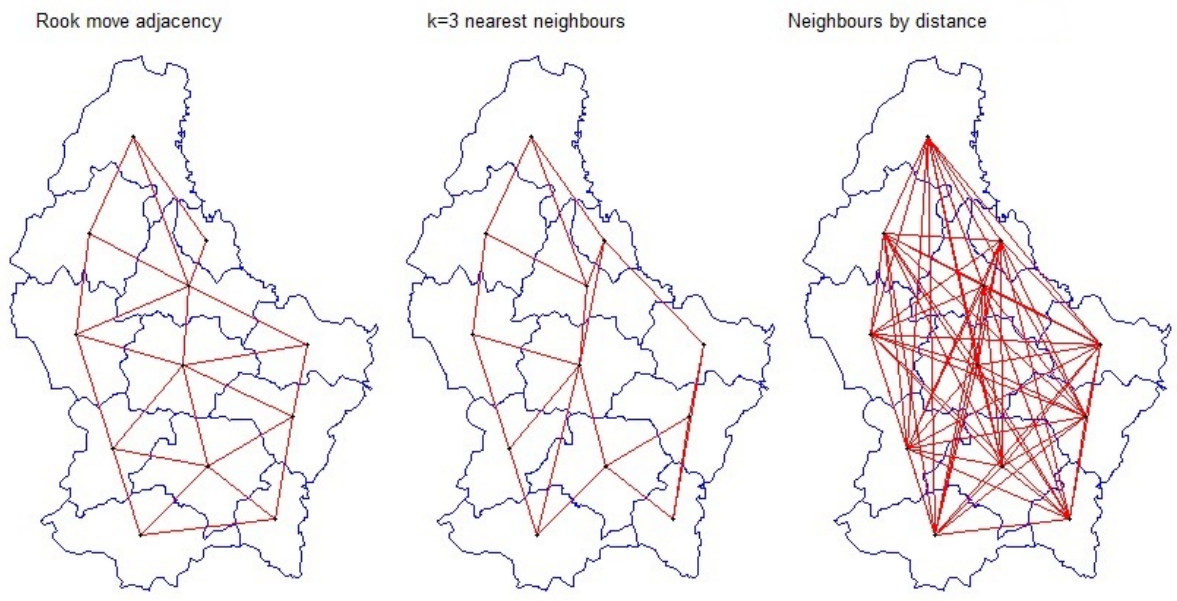
```
# Plot of Luxembourg divided into 12 cantons
# and display their name
par(mar=c(0,0,0,0)) # par sets or adjusts plotting parameters
                    # the parameter mar stands for margin size
plot(lux,border=3, col=terrain.colors(length(lux)), axes=F)
text(lux,"NAME_2",cex=0.5)

# note: we can also generate a vector of n contiguous colors
# using the functions rainbow(n), heat.colors(n),
# terrain.colors(n), topo.colors(n), and cm.colors(n).

# An interactive map of Luxembourg can be obtained using mapview package
mapView(lux)
```

```
# Contiguity-based neighbours

# Derive the neighbourhood structure from the shapefile using poly2nb function.
# Here rook's move contiguity is used (at least one edge is shared between polygons)
w.rook <- poly2nb(lux, row.names=lux$ID_2, queen=FALSE)

# The nb object w.rook lists for each polygon the neighboring polygons.
# For example, to see the neighbors for the first polygon in the object, type:
w.rook[[1]]

> w.rook[[1]]
[1] 2 4 5

# Check it by plotting Luxembourg with the numerical identifier of the cantons
par(mar=c(0,0,0,0))
plot(lux,border=3, col=terrain.colors(length(lux)), axes=F)
text(lux,"ID_2",cex=1)
```



```
# Now, assign the weights to each neighboring polygon.
# The argument style can take on a number of character values:
# W=row standardized, B=binary, C=globally standardized
# with zero.policy = TRUE we insert zero into the weights matrix where there is no connection

w.rook.l <- nb2listw(w.rook, style="B", zero.policy=TRUE) # binary (0/1) weights

# Check the weight of the first polygon's three neighbors
w.rook.l$weights[1]

> w.rook.l$weights[1]
[[1]]
[1] 1 1 1

# Inspect also the weight matrix
w.rook.m <- nb2mat(w.rook, style="B", zero.policy = TRUE) #  spatial weights matrix
w.rook.m

# Compute coordinates at centroids of each canton
coords <- coordinates(lux)

# Plot
par(mai=c(0,0,0,0))
plot(lux, col='white', border='blue')
plot(w.rook, coords, col='red', lwd=2, add=TRUE)
```

```
# Distance-based neighbours

IDs <- row.names(as(lux, "data.frame"))

w.d_k3 <- knn2nb(knearneigh(coords, k=3), row.names = IDs)# 3 nearest neighbours for spatial weigh

# Plot
par(mai=c(0,0,0,0))
plot(lux, col='white', border='blue')
plot(w.d_k3, coords, col='red', lwd=1, add=TRUE)

w.d10 <- dnearneigh(coords, 0, 10) # neighbourhood contiguity by distance

# Plot
par(mai=c(0,0,0,0))
plot(lux, col='white', border='blue')
plot(w.d10, coords, col='red', lwd=1, add=TRUE)
```



References

- Haining R., Guangquan L. (2020), Modelling Spatial and Spatial-Temporal Data. A Bayesian Approach, CRC Press, Sections 4.2, 4.3
- Bivand R., Creating Neighbours, available at: `https://r-spatial.github.io/spdep/articles/nb.html`