Spatial data analysis

Lecture 6

Joaquin Cavieres

1. Introduction

Scale, aggregation, and distance are key concepts in spatial data analysis that can be difficult understand at the beginning.

For example, we have some polygons with soils data for a country. These polygons cover, on average, an area of 100 * 100 = 10,000 km ^{2}. You could transfer the soil properties associated with each polygon, e.g. pH, to a raster with 1 km ^{2} spatial resolution; and now might (incorrectly) say that you have a 1 km2 spatial resolution soils map. So we need to distinguish the resolution of the representation (data) and the resolution of the measurements or estimates.

The scale matter could affect to our estimates of length and size. For example if you wanted to know the length of the coastline of Britain, you could use the length of spatial dataset representing that coastline. You could get rather different numbers depending on the data set used. The higher the resolution of the spatial data, the longer the coastline would appear to be. This is not just a problem of the representation (the data), also at a theoretical level, one can argue that the length of the coastline is not defined, as it becomes infinite if your resolution approaches zero.

Resolution also affects our understanding of relationships between variables of interest. In terms of data collection this means that we want data to be at the highest spatial (and temporal) resolution possible (affordable).

1.1. Distance

Distance is a numerical description of how far apart things are. It is the most fundamental concept in geography. Considering the Waldo Tobler's first law of the geography which says 'everything is related to everything else, but near things are more related than distant things', we need to quantify this relationship in some way and it is not always as easy a question as it seems.

1.1.1. Distance matrix

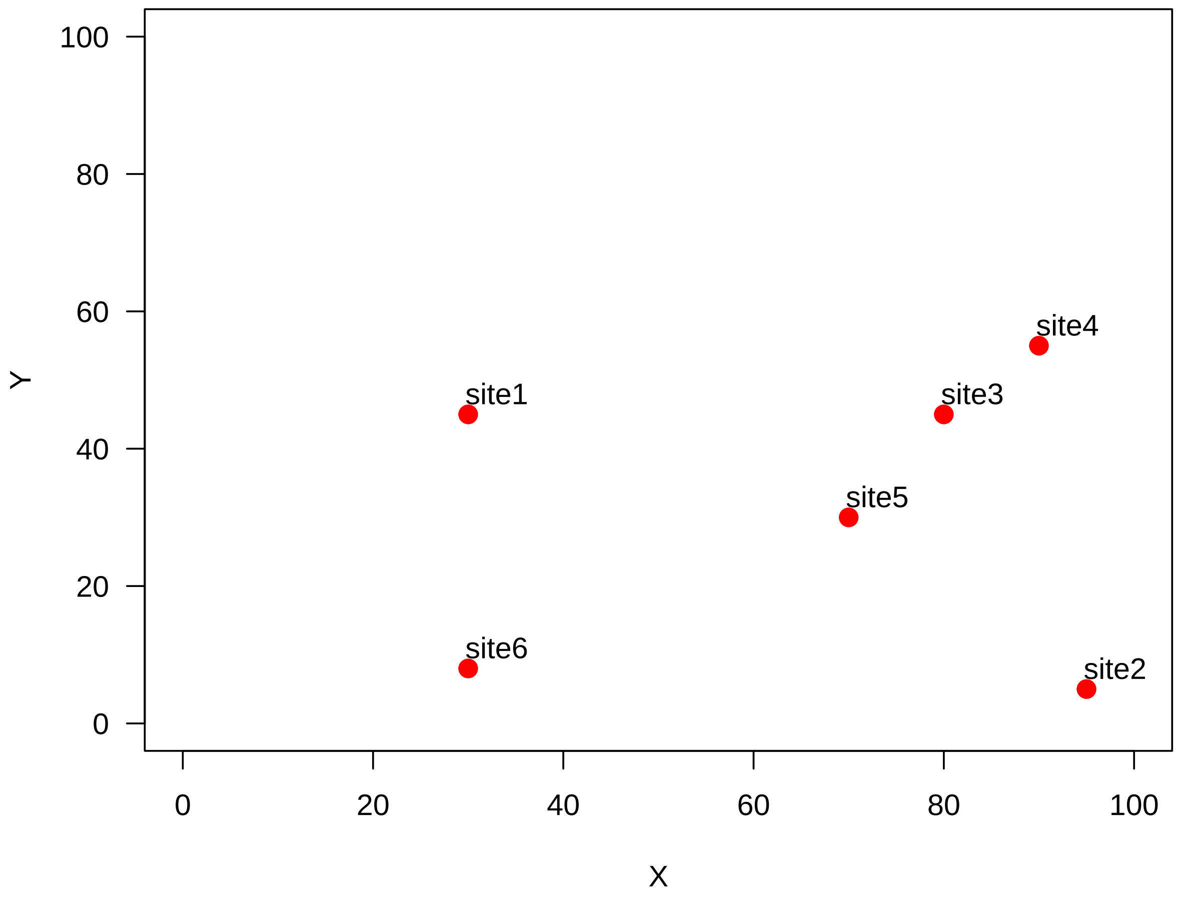
The computation of "distances" are commonly represented by the distance matrix. In this object (the distance matrix) we have all the values calculated for the distances between the objects of interest. For example,

```
site1 <- c(30, 45)
site2 <- c(95, 5)
site3 <- c(80, 45)
site4 <- c(90, 55)
site5 <- c(70, 30)
site6 <- c(30, 8)
pts <- rbind(site1, site2, site3, site4, site5, site6)
pts
```

```
##       [,1] [,2]
## site1   30   45
## site2   95    5
## site3   80   45
## site4   90   55
## site5   70   30
## site6   30    8
```

where `[, 1]` and `[, 2]` are the coordinates for the sites. Right now we can plot the points created:

```
plot(pts, xlim=c(0,100), ylim=c(0,100), pch=20, cex=2, col='red',
     xlab='X', ylab='Y', las=1)
text(pts+3, c("site1", "site2", "site3", "site4", "site5", "site6"))
```


100
80
60
40
20
0
Y
site4
site1
site3
site5
site6
site2
0
20
40
60
80
100
X

To calculate the distance matrix between points we will use the `dist()` function. This distance can be calculated using different methods, but here we will use the classic Euclidean distance:

$$\text{dist\_mat} = \sqrt{\sum_i (x_i - y_i)^2}$$

and in R this matrix can be computed as following:

```
dist_mat <- dist(pts)
dist_mat
```

```
##          site1    site2    site3    site4    site5
## site2 76.32169
## site3 50.00000 42.72002
## site4 60.82763 50.24938 14.14214
## site5 42.72002 35.35534 18.02776 32.01562
## site6 37.00000 65.06919 62.20129 76.21680 45.65085
```

We also can convert the distance matrix to a normal matrix as:

```
normal_dist_mat <- as.matrix(dist_mat)
round(normal_dist_mat)
```

```
##       site1 site2 site3 site4 site5 site6
## site1     0    76    50    61    43    37
## site2    76     0    43    50    35    65
## site3    50    43     0    14    18    62
## site4    61    50    14     0    32    76
## site5    43    35    18    32     0    46
## site6    37    65    62    76    46     0
```

1.1.2. Distance for longitude/latitude coordinates

For this example we have to convert our `pts` created before to a coordinates degrees, that is, longitude and latitude. Doing this we can calculate the Cartesian distance using the function `pointDistance()` from the “raster” package or because using the `dist()` function would give us an incorrect calculation.

```
library(raster)
geo_dist <- pointDistance(pts, lonlat=TRUE)
geo_dist <- as.dist(geo_dist)
geo_dist
```

```
##         1       2       3       4       5
## 2 7665634
## 3 3877671 4666524
## 4 4299435 5562527 1320437
## 5 3858744 3814445 1881598 3201492
## 6 4100293 7190687 6306165 7397475 4826042
```

1.2. Spatial influence

So, what is the influence between geographic objects? It is an important question that we have to answer before we do spatial statistics and modelling. This influence can be represented ad a function of adjacency or distance, and is often expressed as a spatial weights matrix.

1.2.1. Two nearest neighbours

How we can compute the "two nearest neighbours" (or three, or four) adjacency-matrix? Well, for each row, we first get the column numbers in order of the values in that row (that is, the numbers indicate how the values are ordered).

```
cols <- apply(normal_dist_mat, 1, order)
# we need to transpose the result
cols <- t(cols)
```

and then get columns 2 to 3,

```
cols <- cols[, 2:3]
cols
```

```
##       [,1] [,2]
## site1    6    5
## site2    5    3
## site3    4    5
## site4    3    5
## site5    3    4
## site6    1    5
```

As we now have the column numbers, we can make the row-column pairs that we want (using the `rowcols()` function).

```
rowcols <- cbind(rep(1:6, each=2), as.vector(t(cols)))
head(rowcols)
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    1    5
## [3,]    2    5
## [4,]    2    3
## [5,]    3    4
## [6,]    3    5
```

We use these pairs as indices to change the values in matrix object `adj` as:

```
a <-  normal_dist_mat < 50
a
```

```
##       site1 site2 site3 site4 site5 site6
## site1  TRUE FALSE FALSE FALSE  TRUE  TRUE
## site2 FALSE  TRUE  TRUE FALSE  TRUE FALSE
## site3 FALSE  TRUE  TRUE  TRUE  TRUE FALSE
## site4 FALSE FALSE  TRUE  TRUE  TRUE FALSE
## site5  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## site6  TRUE FALSE FALSE FALSE  TRUE  TRUE
```

```
diag(a) <- NA
adj50 <- a * 1
adj50
```

```
##       site1 site2 site3 site4 site5 site6
## site1    NA     0     0     0     1     1
## site2     0    NA     1     0     1     0
## site3     0     1    NA     1     1     0
## site4     0     0     1    NA     1     0
## site5     1     1     1     1    NA     1
## site6     1     0     0     0     1    NA
```

```
adj <- adj50 * 0
adj[rowcols] <- 1
adj
```

```
##       site1 site2 site3 site4 site5 site6
## site1    NA     0     0     0     1     1
## site2     0    NA     1     0     1     0
## site3     0     0    NA     1     1     0
## site4     0     0     1    NA     1     0
## site5     0     0     1     1    NA     0
## site6     1     0     0     0     1    NA
```

1.2.2. Weights matrix

Rather than expressing spatial influence as a binary value (adjacent or not), it is often expressed as a continuous value. The simplest approach is to use inverse distance.

```
w_mat <- 1 / normal_dist_mat
round(w_mat, 4)
```

```
##        site1  site2  site3  site4  site5  site6
## site1    Inf 0.0131 0.0200 0.0164 0.0234 0.0270
## site2 0.0131    Inf 0.0234 0.0199 0.0283 0.0154
## site3 0.0200 0.0234    Inf 0.0707 0.0555 0.0161
## site4 0.0164 0.0199 0.0707    Inf 0.0312 0.0131
## site5 0.0234 0.0283 0.0555 0.0312    Inf 0.0219
## site6 0.0270 0.0154 0.0161 0.0131 0.0219    Inf
```

The “spatial weights” in the weights matrix are often “row-normalized”, hence the sum of the weights for each row in the matrix is the same. Thus, we will change the `Inf` values by `NA` as follows:

```
w_mat[!is.finite(w_mat)] <- NA
```

to so compute the row sums:

```
row_sum <- rowSums(w_mat, na.rm=TRUE)
row_sum
```

```
##      site1      site2      site3      site4      site5      site6
## 0.09997759 0.10006394 0.18566577 0.15140654 0.16030267 0.09349799
```

and divide the rows by their totals (the sum should be 1).

```
w_mat <- w_mat / row_sum
rowSums(w_mat, na.rm=TRUE)
```

```
## site1 site2 site3 site4 site5 site6
##     1     1     1     1     1     1
```

1.2.3. Spatial influence for polygons

In the previous point we saw the adjacency for a set of points, but here we will look at it for polygons.

```
library(terra)
pol <- vect(system.file("ex/lux.shp", package="terra"))
```

We will create a “rook’s case” neighbors matrix

```
w_pol <- adjacent(pol, "rook", pairs=FALSE)
dim(w_pol)
```

```
## [1] 12 12
```

```
w_pol[1:6,1:11]
```

```
##   1 2 3 4 5 6 7 8 9 10 11
## 1 0 1 0 1 1 0 0 0 0  0  0
## 2 1 0 1 1 1 1 0 0 0  0  0
## 3 0 1 0 0 1 0 0 0 1  0  0
## 4 1 1 0 0 0 0 0 0 0  0  0
## 5 1 1 1 0 0 0 0 0 0  0  0
## 6 0 1 0 0 0 0 0 1 0  0  0
```

compute the number of neighbors for each area and express it in percentage

```
i <- rowSums(w_pol)
i
```

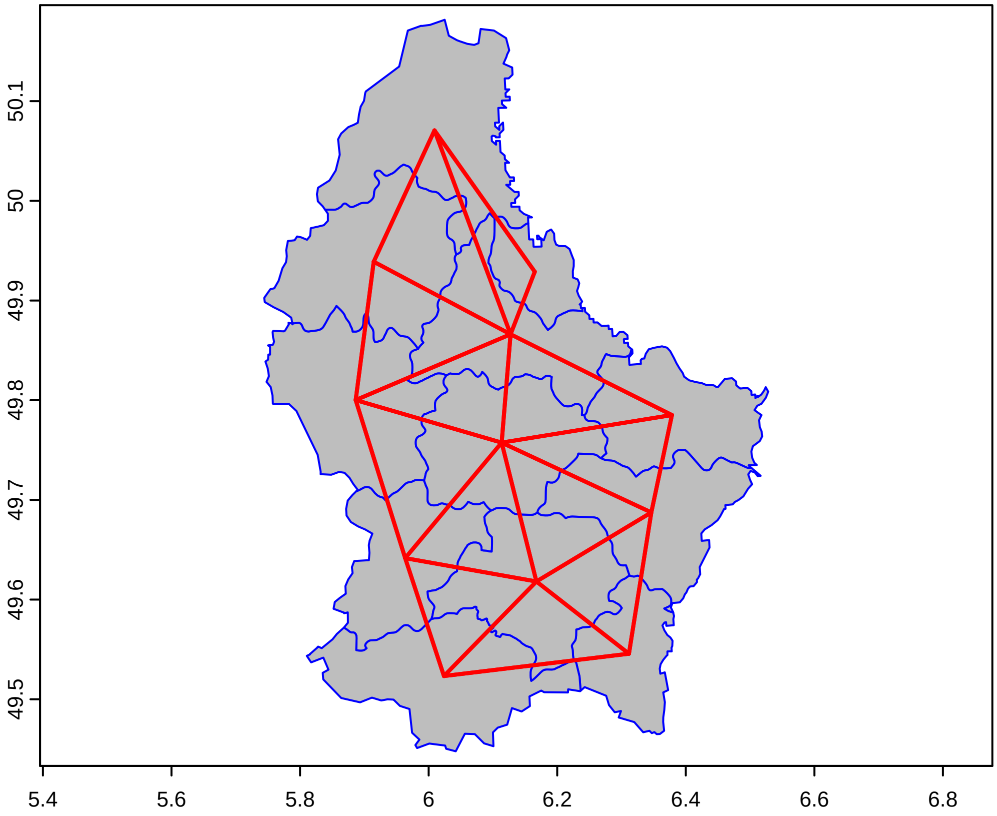
```
##  1  2  3  4  5  6  7  8  9 10 11 12
##  3  6  4  2  3  3  3  4  4  3  5  6
```

```
round(100 * table(i) / length(i), 1)
```

```
## i
##    2    3    4    5    6
##  8.3 41.7 25.0  8.3 16.7
```

to finally link the polygons in a plot:

```
plot(pol, col="gray", border="blue")
nb <- adjacent(pol, "rook")
v <- centroids(pol)
p1 <- v[nb[,1], ]
p2 <- v[nb[,2], ]
lines(p1, p2, col="red", lwd=2)
```



There are other alternatives to compute the "spatial influence", for example:

- Distance based
- Nearest neighbors
- Raster based distance metrics

2. Spatial autocorrelation

Spatial autocorrelation is one of the most important concepts in spatial statistics. It is a both a nuisance, as it complicates statistical tests, and a feature, as it allows for spatial interpolation. Its computation and properties are often misunderstood.

The correlation (in spatial statistics is called "autocorrelation") is a measure of similarity between nearby observations. To see this we have to review the concept but viewed from a temporal dimension.

2.1. Temporal autocorrelation

If we consider a same object across time, let's say a person, and we are measuring its weight and wealth in a specific interval of time. So, if in a couple of years the weight went from 60 to 80 kg, the probability that this increment of kg was done by day is low, thus probably went up gradually, with the occasional tapering off, or even reverse in direction. This same could happen with your bank account, but probably marked by a monthly increase. To measure the degree of association between those two variables (weight and wealth) over time, we can compute the correlation of each observation with the next observation.

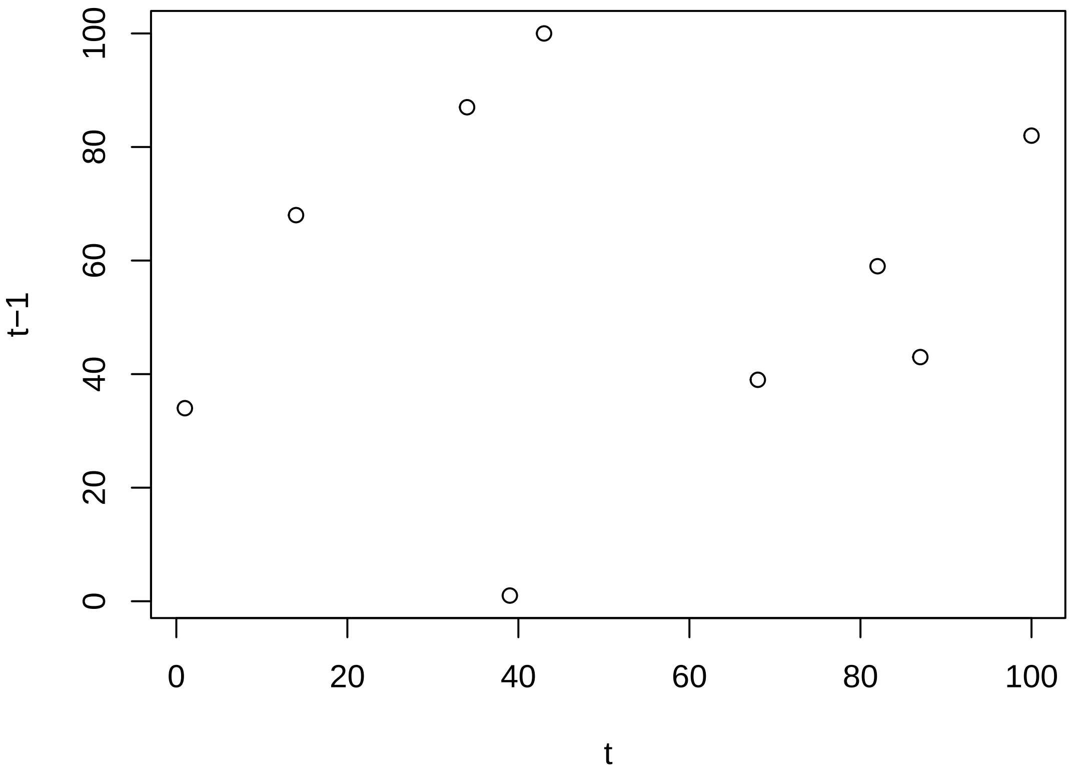
We will create the next vector of daily "temporal" observations:

```r
set.seed(0)
t <- sample(100, 10)
t
```

```
##  [1]  14  68  39   1  34  87  43 100  82  59
```

and compute the autocorrelation as

```r
a <- t[-length(t)]
b <- t[-1]
plot(a, b, xlab='t', ylab='t-1')
```



```
# Correlation
cor(a, b)
```

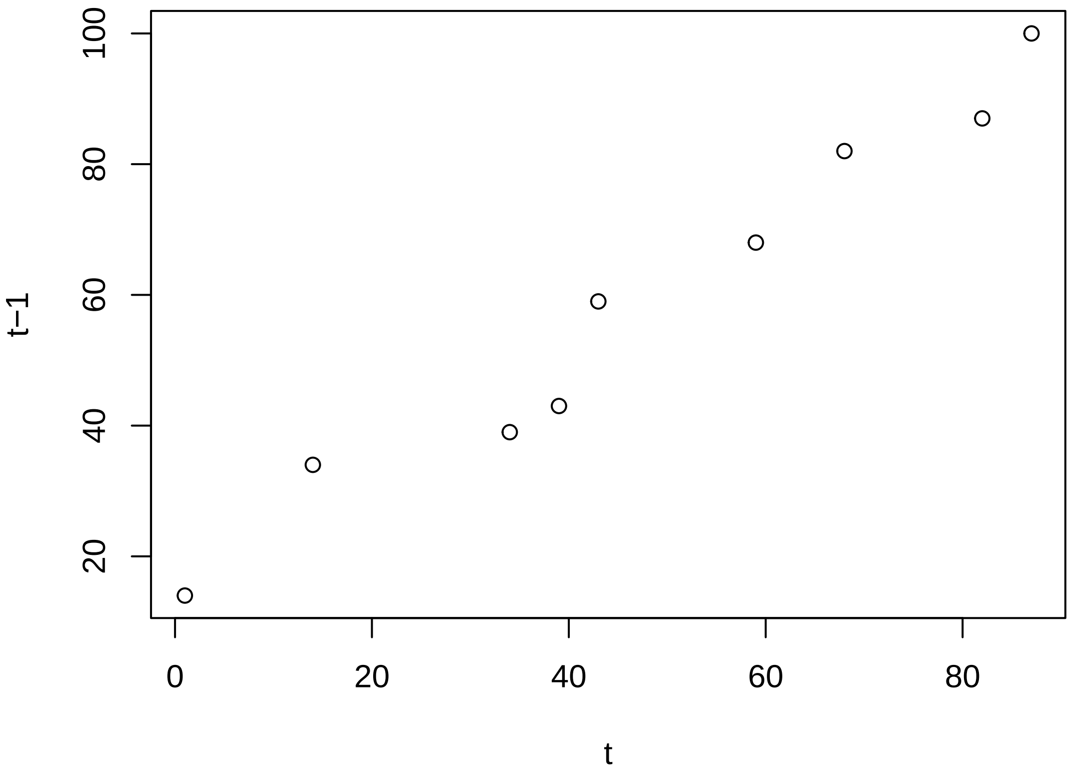
```
## [1] 0.1227634
```

The correlation computed is small (0.2), and in this case we computed the "one-lag" autocorrelation, that is, we compare each value to its immediate neighbour, and not to other nearby values. But, after sorting the numbers in the vector `t`, the autocorrelation becomes strong.

```
t <- sort(t) # sort is the function to order the numbers in the vector
t
```

```
##  [1]   1  14  34  39  43  59  68  82  87 100
```

```
a <- t[-length(t)]
b <- t[-1]
plot(a, b, xlab='t', ylab='t-1')
```



```
# New correlation
cor(a, b)
```

```
## [1] 0.9819258
```

2.2. Spatial autocorrelation

Spatial autocorrelation is an extension of the temporal autocorrelation. However, it is a little more complicated to understand given that we are working in a two dimensional space and it has complex structures, hence could be not obvious how to determine what is “near”.

Measures of spatial autocorrelation describe the degree two which observations (values) at spatial locations (whether they are points, areas, or raster cells), are similar to each other. So we need two things: observations and locations.

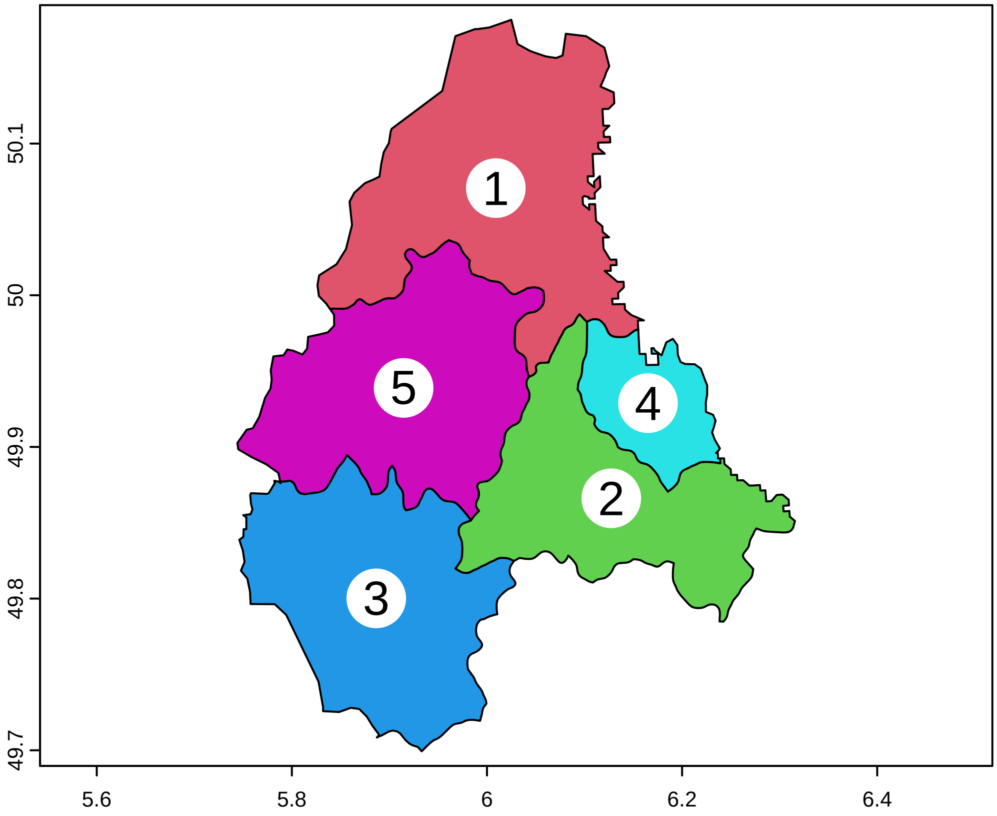
Spatial autocorrelation in a variable can be exogenous (it is caused by another spatially autocorrelated variable, e.g. rainfall) or endogenous (it is caused by the process at play, e.g. the spread of a disease). For example,

```r
p <- vect(system.file("ex/lux.shp", package="terra"))
p <- p[p$NAME_1=="Diekirch", ]
p$value <- c(10, 6, 4, 11, 6)
as.data.frame(p)
```

```
##   ID_1   NAME_1 ID_2   NAME_2 AREA   POP value
## 1    1 Diekirch    1 Clervaux  312 18081    10
## 2    1 Diekirch    2 Diekirch  218 32543     6
## 3    1 Diekirch    3  Redange  259 18664     4
## 4    1 Diekirch    4  Vianden   76  5163    11
## 5    1 Diekirch    5    Wiltz  263 16735     6
```

In this case we are interested in calculating the spatial autocorrelation in the variable "AREA". If the value of the spatial autocorrelation is high, then the regions with a similar size would be spatially clustered.

```r
plot(p, col=2:7)
xy <- centroids(p) # A function to get the centroids of the polygons
points(xy, cex=6, pch=20, col='white')
text(p, 'ID_2', cex=1.5)
```



2.2.1. Adjacent polygons

Now the important part, try to determine which polygons are "near" and what is the way to quantify that. For this we will use the adjacency as criterion. To find adjacent polygons, we will use the package "spdep".

```
library(spdep)
w_mat <- adjacent(p, symmetrical=TRUE)
class(w_mat)
```

```
## [1] "matrix" "array"
```

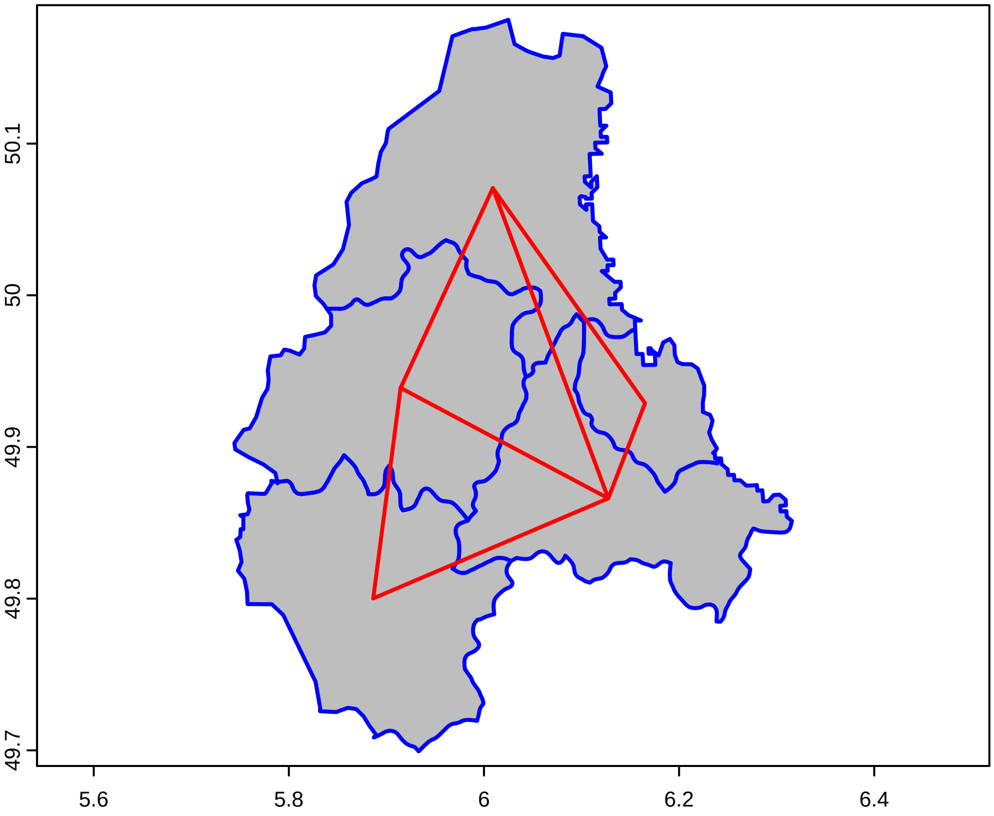
```
head(w_mat)
```

```
##      from to
## [1,]    1  2
## [2,]    1  4
## [3,]    1  5
## [4,]    2  3
## [5,]    2  4
## [6,]    2  5
```

What is the adjacency matrix?

We will plot the links between polygons,

```
plot(p, col='gray', border='blue', lwd=2)
p1 <- xy[w_mat[,1], ]
p2 <- xy[w_mat[,2], ]
lines(p1, p2, col='red', lwd=2)
```



and the spatial weights matrix, reflecting the intensity of the geographic relationship between observations.

```
w_mat <- adjacent(p, pairs=FALSE)
w_mat
```

```
##   1 2 3 4 5
## 1 0 1 0 1 1
## 2 1 0 1 1 1
## 3 0 1 0 0 1
## 4 1 1 0 0 0
## 5 1 1 1 0 0
```

2.2.2. Moran's index

In statistics, Moran's I is a measure of spatial autocorrelation and it is characterized by a correlation in a signal among nearby locations in space. It can be computed using the following equation:

$$I = \frac{n}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i - \bar{y})(y_j - \bar{y})}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}}$$

It could be complicated to calculate, but it is not much more than an expanded version of the formula to compute the correlation coefficient. We will compute the Moran's index in R:

```
# Number of obseervations
n <- length(p)
```

Calculating the value of y and $\bar{y}$:

```
y <- p$value
ybar <- mean(y)
```

Calculating the following expression: $(y_i - \bar{y})(y_j - \bar{y})$

```
diff_y <- y - ybar
g <- expand.grid(diff_y, diff_y)
yiyj <- g[,1] * g[,2]
```

Create a matrix of the multiplied pairs

```
pairs_mat <- matrix(yiyj, ncol=n)
```

and multiply this matrix with the weights to set to zero the value for the pairs that are not adjacent.

```
pairs_mat_w <- pairs_mat * w_mat
pairs_mat_w
```

```
##       1     2    3     4     5
## 1  0.00 -3.64 0.00  9.36 -3.64
## 2 -3.64  0.00 4.76 -5.04  1.96
## 3  0.00  4.76 0.00  0.00  4.76
## 4  9.36 -5.04 0.00  0.00  0.00
## 5 -3.64  1.96 4.76  0.00  0.00
```

Now we will sum the values, to get this bit of Moran's I using the following equation:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i - \bar{y})(y_j - \bar{y})$$

and in R that is:

```
sum_pairs_mat_w <- sum(pairs_mat_w)
sum_pairs_mat_w
```

```
## [1] 17.04
```

The next step is to divide this value by the sum of weights:

```
sum_mat_w <- sum(w_mat)
sum_w  <- sum_pairs_mat_w / sum_mat_w
```

and compute the inverse variance of y

```
vr <- n / sum(diff_y^2)
```

to finally compute the Moran's index

```
MI <- vr * sum_w
MI
```

```
## [1] 0.1728896
```

Now we will compute the Moran's index using the package "spdep". Here we will create a spatial weights matrix, and then use the `autocor()` function:

```
w_2 <-  adjacent(p, "queen", pairs=FALSE)
w_2
```

```
##   1 2 3 4 5
## 1 0 1 0 1 1
## 2 1 0 1 1 1
## 3 0 1 0 0 1
## 4 1 1 0 0 0
## 5 1 1 1 0 0
```

```
ac <- autocor(p$value, w_2, "moran")
ac
```

```
## [1] 0.1728896
```

References

Bivand, R. S., Pebesma, E. J., Gomez-Rubio, V., & Pebesma, E. J. (2008). Applied spatial data analysis with R (Vol. 747248717, pp. 237-268). New York: Springer.

Spatial Data Science with R and “terra”. https://rspatial.org/index.html.