Spatial data

Contents

Introduction to spatial analysis

As with any analysis, spatial analyses begin with data exploration. The presence of a spatial pattern in the data often depends on our sampling, but it can also result from circumstances we couldn't control. Whether one or the other, there are ways to check if spatial patterns are present in our models and if our measures are subject to spatial autocorrelation. Therefore, we will use indices to verify the presence of these patterns:

Moran index

Moran I index, or Moran Global allows testing if a significant correlation is present between neighboring regions. The Moran's I index is a spatial autocorrelation coefficient of the z, weighted by the weights w_{ij}. Therefore, it takes values between -1 and 1

$$I = \frac{N}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij}(z_i - \bar{z})(z_j - \bar{z})}{\sum_i (z_i - \bar{z})^2}$$

In this equation, we recognize the expression of a correlation, which is the product of the deviations from the mean of two variables z_i and z_j, divided by the product of their standard deviations (which is the same, so we get the variance). The contribution of each pair (i, j) is multiplied by its weight w_{ij}, and the term on the left (the number of regions N divided by the sum of the weights) ensures that the result is bounded between -1 and 1. Positive spatial autocorrelation indicates that elements or values are clustered in space,

and neighboring locations tend to have similar values or characteristics. Negative spatial autocorrelation, on the other hand, indicates that elements or values are dispersed in space, and neighboring locations tend to have different values or characteristics.

Since the distribution of I is known in the absence of spatial autocorrelation, this statistic allows testing the null hypothesis that there is no spatial correlation between neighboring regions.

Displaying data on a map is very useful for planning analysis. There are several ways to create maps in R. Among the packages that allow you to display maps, you will find rnaturalearth. To practice, we will use a dataset where "budburst," a phenological phase that determines the start of the growing season in trees, has been observed. Each point indicates the coordinates of a stand where the budburst date was observed. The date is indicated in Julian days (DOY - Day of the year).

To start, we import the data along with their coordinates. At this stage, we will produce a shapefile using the st_as_sf function from the sf package. At this stage, it is crucial to specify the Coordinate Reference System (CRS) used to position the points in space. The CRS not only provides information about the unit of measurement (angles, distances) and the projection used but also encompasses essential details such as the reference spheroid (the approximate shape of the Earth: conical, cylindrical, or planar) and a datum (providing information about the orientation of the spheroid relative to the Earth). These details are crucial to ensure the accuracy of calculations performed on spatial data. For example, when calculating distances between points, the unit of distance measurement depends on the CRS used. Furthermore, understanding the parameters of the CRS allows correctly projecting the data into other coordinate systems if necessary, thus ensuring the consistency of geospatial analyses. You can explore and verify information about the CRS on specialized websites such as epsg.io. The most commonly used CRS is usually WGS84, which is also used in GPS satellite positioning systems (epsg.io/4326).

```
require(rnaturalearth)
```

```
## Le chargement a nécessité le package : rnaturalearth
```

```
## Warning: le package 'rnaturalearth' a été compilé avec la version R 4.3.2
```

```
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##      (status 2 uses the sf package in place of rgdal)
```

```
## Support for Spatial objects (`sp`) will be deprecated in {rnaturalearth} and will be removed in a fu
```

```
require(sp)
```

```
## Le chargement a nécessité le package : sp
```

```
require(sf)
```

```
## Le chargement a nécessité le package : sf
```

```
## Linking to GEOS 3.11.2, GDAL 3.6.2, PROJ 9.2.0; sf_use_s2() is TRUE
```

```
P1phasespatialtraining1 <-  P1phasespatialtraining1 <-
read.csv("C:/Users/buttoval/Documents/ECL8202/donnees/P1phasespatialtraining1.csv",
    sep = ";")


P1phasespatialtraining1_shapefile <- st_as_sf(P1phasespatialtraining1,
    coords = c("x_long", "y_lat"), crs = "WGS84")
```

Once the shapefile is produced, we can use the ne_countries function to download a map of the country from which we extracted the data. The data is associated with the catalog of public maps: Natural Earth Data (https://www.naturalearthdata.com/). Maps can also be downloaded from the website and are offered at different scales: large (1:10 m), medium (1:50 m), and small (1:110 m), depending on the desired level of detail.

We can download the maps directly from R using the ne_download function. Here's a first world map, to which we can overlay layers representing lakes and oceans. A complete list of available layers is presented in the tables of the rnaturalearth vignette (https://cran.r-project.org/web/packages/rnaturalearth/vignettes/rnaturalearth.html).

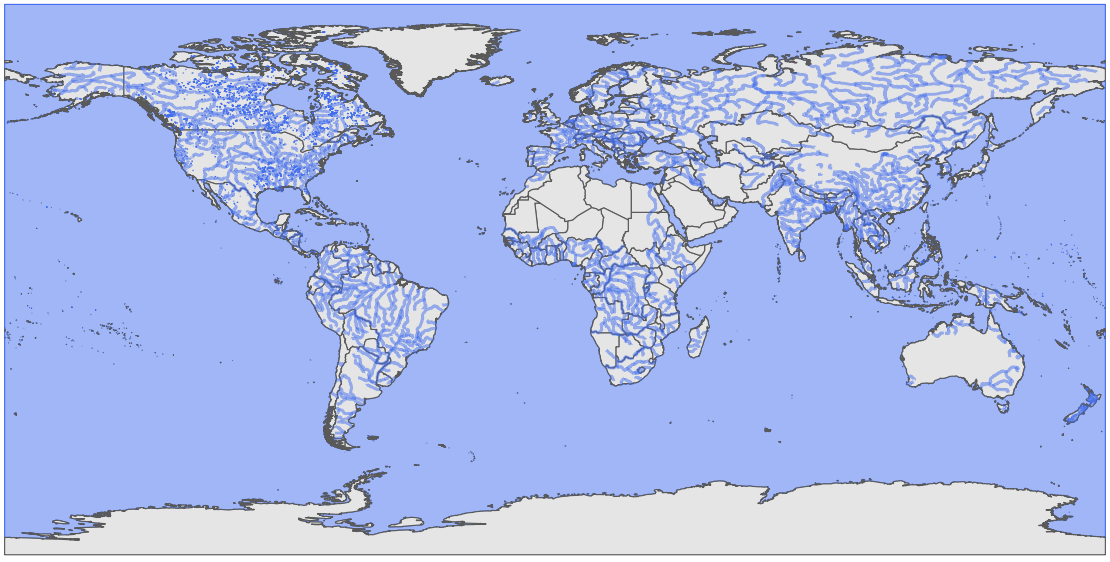
```
sfdf_world <- ne_download(scale = 10, type = "countries", returnclass = "sf")

sfdf_lakes <- ne_download(scale = 10, type = "lakes_north_america",
    category = "physical", returnclass = "sf")

sfdf_oceans <- ne_download(scale = 10, type = "ocean", category = "physical",
    returnclass = "sf")

sfdf_riverslake <- ne_download(scale = 10, type = "rivers_lake_centerlines",
    category = "physical", returnclass = "sf")


ggplot(data = sfdf_world) + geom_sf(data = sfdf_oceans, col = "royalblue2",
    fill = "royalblue2", alpha = 0.5) + geom_sf() + geom_sf(data = sfdf_lakes,
    col = "royalblue2", alpha = 0.5) + geom_sf(data = sfdf_riverslake,
    col = "royalblue2", alpha = 0.5)
```

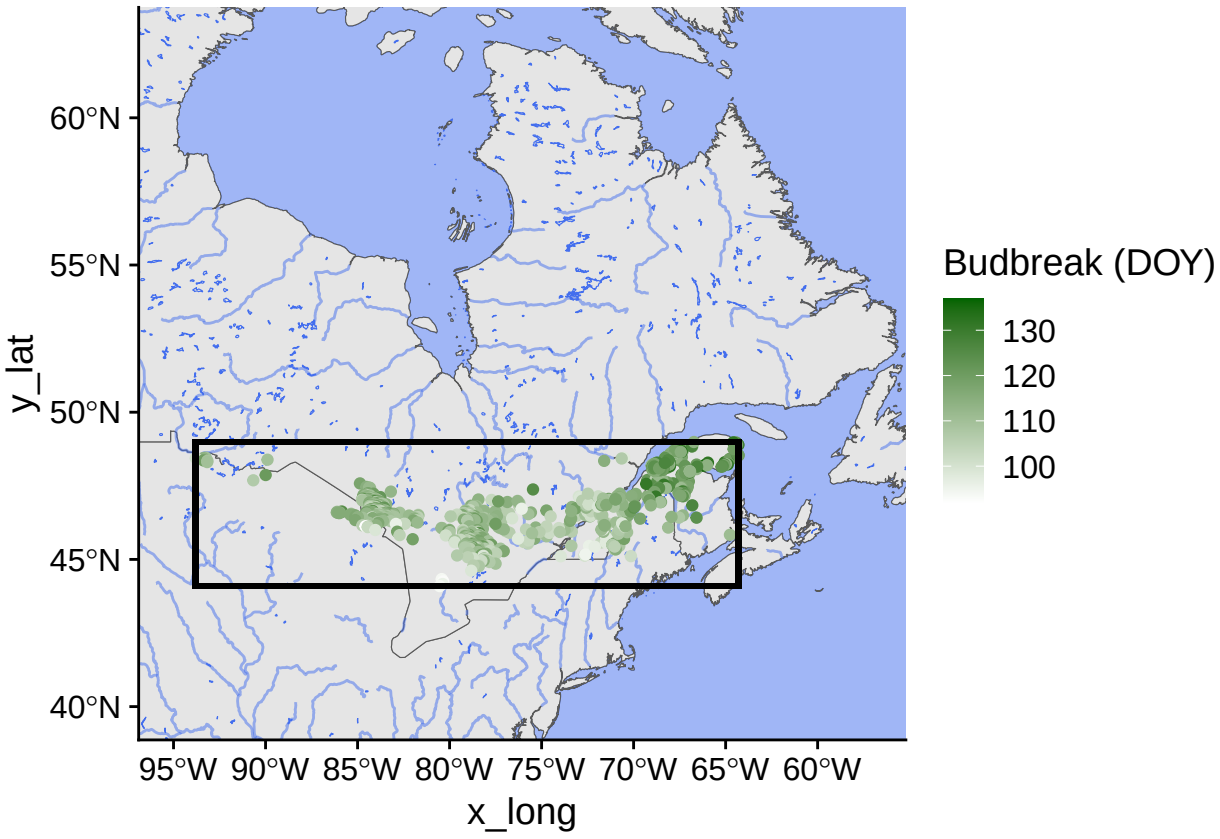


To better visualize our points, it's now useful to restrict the map to our area of interest, which is the northern part of America. We can use geom_rect to display only this area. If necessary, we can also create a bounding box around our coordinates using geom_rect. The coordinates of the square can be obtained using the st_bbox function. The default CRS of objects obtained by rnaturalearth is WGS84, so we don't need to transform it, but we could have used the code "shapefile %>% st_transform(our CRS)" for reprojection.

```
coordbox <- as.data.frame(st_bbox(P1phasespatialtraining1_shapefile))

ggplot(data = sfdf_world) + geom_sf(data = sfdf_oceans, col = "royalblue2",
    fill = "royalblue2", alpha = 0.5) + geom_sf() + geom_sf(data = sfdf_lakes,
    col = "royalblue2", alpha = 0.5) + geom_sf(data = sfdf_riverslake,
    col = "royalblue2", alpha = 0.5) + geom_point(P1phasespatialtraining1,
    mapping = aes(x_long, y_lat, color = DOY)) + scale_color_gradient(name = "Budbreak (DOY)",
    low = "white", high = "darkgreen") + coord_sf(xlim = c(-95,
    -57.1), ylim = c(40, 62.62)) + geom_rect(xmin = coordbox$x[1],
    ymin = coordbox$x[2], xmax = coordbox$x[3], ymax = coordbox$x[4],
    fill = NA, colour = "black", size = 1)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



We already observe a tendency towards earlier budbreak towards the southwest and later towards the northeast. But how can we validate these patterns?

We can start by calculating the Moran's I index. We will do this in several steps:

We begin by creating the coordinate matrix: We start by creating a matrix containing the latitudinal and longitudinal coordinates of each observation point. This will allow us to spatially represent our data.

```
require(spdep)
```

```
## Le chargement a nécessité le package : spdep
```

```
## Warning: le package 'spdep' a été compilé avec la version R 4.3.2
```

```
## Le chargement a nécessité le package : spData
```

```
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```
latlon = cbind(P1phasespatialtraining1$x_long, P1phasespatialtraining1$y_lat)
```

Conversion en objet spatial : Ensuite, nous convertissons cette matrice en un objet spatial afin de pouvoir effectuer des analyses spatiales.

```
latlon<-coordinates(latlon)
```

Conversion to spatial object: Next, we convert this matrix into a spatial object so that we can perform spatial analyses.

```
IDs <- row.names(as.data.frame(x = P1phasespatialtraining1$x_long,
    y = P1phasespatialtraining1$y_lat))
```

Creation of the neighbor list: We use the k-nearest neighbors algorithm to create a list of neighbors for each point. Here, we use k=1 to consider only direct links between the closest points. This line creates a neighbor list based on the nearest neighbor (k=1) using the latitudinal and longitudinal coordinates. This means that each point is linked to its nearest neighbor. Since k=1, every observation has a neighbor. We aim to assess spatial autocorrelation by considering only direct links between the closest points. This may be appropriate in some cases, especially when we assume that spatial influence decreases rapidly with distance and that values are strongly influenced by their immediate neighborhood. ”

```
Neigh_nb <- knn2nb(knearneigh(latlon, k = 1, longlat = TRUE),
    row.names = IDs)
```

Standardization of weights: We standardize the weights of the neighbor list so that the sum of weights for each spatial unit equals 1. This allows for appropriate comparison of weights between different spatial units. Here, style = “W” indicates that the weights for each spatial unit are standardized such that their sum equals 1 (this is called row standardization).

```
Neigh_nb_moran<-nb2listw(Neigh_nb,style="W")
```

Moran’s Test: Finally, we apply Moran’s test to our data. This allows us to assess the spatial autocorrelation of our variable of interest (in this case, the day of the year of plant budburst) using the standardized neighbor list we created earlier.

```
Moran_I <- moran.test(P1phasespatialtraining1$DOY, listw = Neigh_nb_moran)

Moran_I
```

```
##
##  Moran I test under randomisation
##
## data:  P1phasespatialtraining1$DOY
## weights: Neigh_nb_moran
##
## Moran I statistic standard deviate = 12.302, p-value < 2.2e-16
```

```
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.557497347      -0.001297017       0.002063322
```

In the table, we find the values of the standard deviation of the Moran I statistic (Moran I statistic standard deviate). This indicates the standardized deviation of the Moran I statistic from its mean under the null hypothesis of no spatial autocorrelation. When the standardized deviation of the Moran I statistic deviates from zero towards high positive values, it suggests strong positive spatial autocorrelation, meaning similar values tend to cluster in space. In our case, we obtained a value of 12.302, which is quite high.

The Moran I statistic is 0.6, indicating positive spatial autocorrelation. This means there is a tendency for similar values to cluster in space.

The p-value associated with the Moran I statistic is less than 0.05, suggesting rejection of the null hypothesis. In this case, the null hypothesis would be the absence of spatial autocorrelation.

Overall, these results suggest strong spatial autocorrelation (high Moran I statistic) with a very low probability that this is due to randomness (very low p-value), and the alternative hypothesis is that spatial autocorrelation is greater than expected by chance.

"Expectation" refers to the expected value of the Moran I statistic under the null hypothesis of no spatial autocorrelation. Specifically, it is the theoretical average of the Moran I statistic calculated over many random repetitions of the sample in which the spatial distribution of values is random. If the observed Moran I statistic significantly differs from this expected value, it suggests significant spatial autocorrelation in your data.

Indice de Moran Local:

Si l'autocorrelation spatiale est significatif, il est possible de calculer le Moran Local qui évalue l'autocorrélation spatiale pour chaque unité spatiale individuelle (points, zones etc..). Cet indicateur identifie les zones spécifiques où l'autocorrélation spatiale est significativement différente de celle attendue par hasard.

Pour calculer l'indice de Moran sur R il est possible

$$I_i = \frac{(z_i - \bar{z})}{S^2} \sum_j w_{ij}(z_j - \bar{z})$$

Here, z_i represents the value of the variable for spatial unit i, $\bar{z}$ is the mean of the variable values for all spatial units. z_{ij}, w_{ij} represent the weight between spatial units i and j, z_j is the value of the variable for spatial unit j; S^2 is the variance of the variable across all spatial units. To calculate the Local Moran's Index, we start by calculating the distances between the nearest neighbors:

```
dstsP1 <- unlist(nbdists(Neigh_nb, latlon))
```

After calculating the distances between the nearest neighbors, we then determine the maximum distance between the nearest neighbors and create different neighborhood structures based on this distance:

```
max_1nnP1 <- max(dstsP1)

Neigh_kd2P1 <- dnearneigh(latlon, d1 = 0, d2 = 2 * max_1nnP1,
    row.names = IDs)

# neighbors within 2X maximum distance
```

Creation of weights for the neighborhood structure and calculation of the Local Moran's Index. We can choose Style B (binary) for creating weights of the neighborhood structure. We opt to use Binary weighting

Style (B) because each pair of spatial units is considered either connected (1) or not connected (0). This facilitates the identification of spatial clusters, which is ultimately the main objective of the local Moran analysis.

```
weightsP1 <- nb2listw(Neigh_kd2P1, style = "B")

# row standardized binary weights, using minimum distance
# for one neighbor

weightsP1
```

```
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 772
## Number of nonzero links: 182720
## Percentage nonzero weights: 30.65854
## Average number of links: 236.6839
## 2 disjoint connected subgraphs
##
## Weights style: B
## Weights constants summary:
##     n     nn     S0     S1        S2
## B 772 595984 182720 365440 187478256
```

```
localP1 <- localmoran(x = P1phasespatialtraining1$DOY, listw = nb2listw(Neigh_kd2P1,
    style = "B"))

head(localP1)
```

```
##           Ii         E.Ii     Var.Ii       Z.Ii Pr(z != E(Ii))
## 1 -62.134967 -0.067335093  30.035740 -11.325203   9.845025e-30
## 2  49.310064 -0.111485688  46.483906   7.248782   4.205357e-13
## 3 148.952876 -0.760734852 465.290603   6.940634   3.903441e-12
## 4   2.430195 -0.004170537   2.115739   1.673612   9.420693e-02
## 5 -47.977739 -0.737517583 359.903025  -2.490114   1.277022e-02
## 6  95.099477 -0.230898178 139.861426   8.060878   7.574863e-16
```

Dans l'objet crée par la fonction localmoran, nous pouvons trouver: -L'indice de Moran local pour l'unité spatiale i he Local Moran's Index for spatial unit i: a measure of local spatial autocorrelation for this spatial unit taking into account the values of its local neighbors.

-Expected Local Moran's Index (E.Ii): the average value of the Local Moran's Index in a set of random samples under the null hypothesis of no spatial autocorrelation

-Variance of the Local Moran's Index(Var.Ii): he Variance of the Local Moran's Index (Var.I: the dispersion of values of the Local Moran's Index around its expected value. A high variance suggests large variability in the values of the Local Moran's Index.

- The Z score of Moran index (Z.Ii): the number of standard deviations from the expected value in the observed Local Moran's Index.
- Pr(z != E(Ii)): he probability that the Z score of the Local Moran's Index differs from the expected value. If this probability is less than a specified alpha threshold (usually 0.05), we reject the null hypothesis of no local spatial autocorrelation in favor of the alternative hypothesis of significant local spatial autocorrelation.

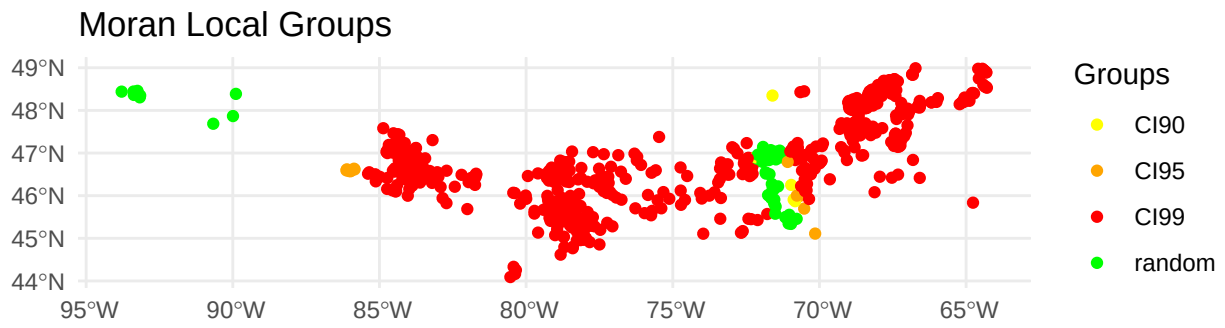
With the z scores, it's possible to classify observations into groups based on the Local Moran's Index and identify areas where spatial aggregation is significant. These categories are based on the significance thresholds

of the z scores for the Local Moran's Index. They are used to identify significant spatial clusters at different levels of statistical confidence (e.g., 90%, 95%, and 99%). The "4-random" category is used for spatial units where spatial autocorrelation is not significant.

```
moran.mapP1 <- cbind(P1phasespatialtraining1_shapefile,
    localP1)


moran.mapP1$groups_class <- with(moran.mapP1, ifelse(Z.Ii >=
    -24 & Z.Ii < -2.58, "CI99", ifelse(Z.Ii >= -2.58 &
    Z.Ii < -1.96, "CI95", ifelse(Z.Ii >= -1.96 & Z.Ii <
    -1.65, "CI90", ifelse(Z.Ii >= -1.65 & Z.Ii < 1.65,
    "random", ifelse(Z.Ii >= 1.65 & Z.Ii < 1.96, "CI90",
        ifelse(Z.Ii >= 1.96 & Z.Ii < 2.58, "CI95",
            "CI99")))))))
```

```
ggplot() + geom_sf(data = moran.mapP1, aes(color = groups_class)) +
    scale_color_manual(values = c(CI99 = "#ff0000", CI95 = "#ffa500",
        CI90 = "#ffff00", random = "#00ff00"), name = "Groups") +
    labs(title = "Moran Local Groups") + theme_minimal()
```



Most of the values fall within the 99% confidence interval (CI99). The values of the studied variable, the day of budburst (DOY), are strongly clustered in space, forming significant spatial clusters.

If we're interested in investigating significant spatial clusters, it would be possible to conduct a hotspot-coldspot analysis to determine if very high values and those that are smaller tend to cluster in different parts of the map. To do this, we need to calculate another index, the Getis-Ord G_i* which indicates spatial concentration and evaluates whether the values of a variable are randomly distributed in space or if they

exhibit a tendency towards spatial concentration, i.e., clusters of high or low values G_i*. calculates z-statistics for each spatial unit and can be used to identify hotspots (high values surrounded by high values) and coldspots (low values surrounded by low values).

Once again, the localG_perm function from the spdep package helps us calculate from the spatial weighting matrix we used calculate G_i* from the spatial weighting matrix we used before.

```
local_gpermP1 <- localG_perm(P1phasespatialtraining1_shapefile$DOY,
    weightsP1)

P1phasesta_shape <- cbind(P1phasespatialtraining1_shapefile,
    gstat = as.matrix(local_gpermP1))
```

Absolutely, it's also possible to categorize groups based on Z values to identify hotspots (positive Z) and coldspots (negative Z). Hotspots are associated with positive Z values, indicating that values in these areas are higher than expected compared to the overall spatial distribution of the data. Coldspots are associated with negative Z values, indicating that values in these areas are lower than expected compared to the overall spatial distribution of the data.

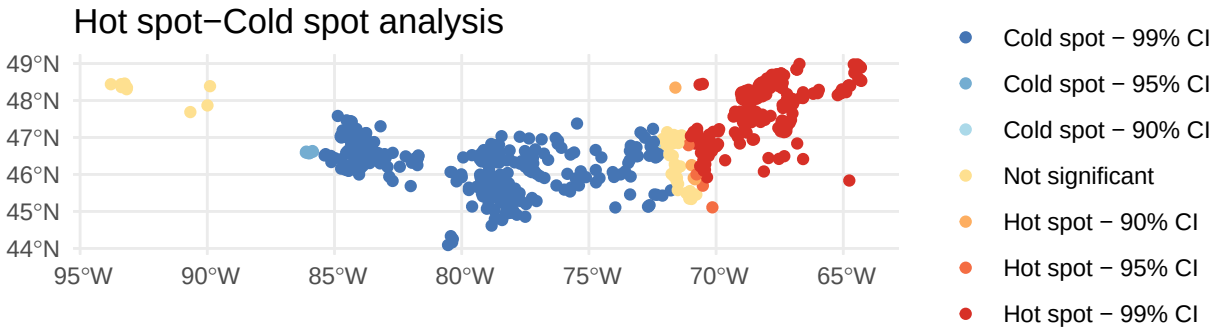
```
P1phasesta_shape$groups_class <- with(P1phasesta_shape,
    ifelse(gstat >= min(P1phasesta_shape$gstat) & gstat <
        -2.58, "1-CI99", ifelse(gstat >= -2.58 & gstat <
        -1.96, "2-CI95", ifelse(gstat >= -1.96 & gstat <
        -1.65, "3-CI90", ifelse(gstat >= -1.65 & gstat <
        1.65, "4-random", ifelse(gstat >= 1.65 & gstat <
        1.96, "5-CI90", ifelse(gstat >= 1.96 & gstat <
        2.58, "6-CI95", "7-CI99")))))))

P1phasesta_shape$groups_classlabelled <- as.factor(P1phasesta_shape$groups_class)

levels(P1phasesta_shape$groups_classlabelled) <- list(`Cold spot - 99% CI` = "1-CI99",
    `Cold spot - 95% CI` = "2-CI95", `Cold spot - 90% CI` = "3-CI90",
    `Not significant` = "4-random", `Hot spot - 90% CI` = "5-CI90",
    `Hot spot - 95% CI` = "6-CI95", `Hot spot - 99% CI` = "7-CI99")




ggplot() + geom_sf(data = P1phasesta_shape, aes(color = groups_classlabelled)) +
    scale_color_manual("", values = c("#4575b4", "#74add1",
        "#abd9e9", "#fee090", "#fdae61", "#f46d43",
        "#d73027")) + labs(title = "Hot spot-Cold spot analysis") +
    theme_minimal()
```



#Variograms

En effet, en géostatistique, le variogramme est un outil fondamental pour étudier la structure de l'autocorrélation spatiale d'une variable géographique. Le variogramme mesure la variance des différences entre les valeurs de cette variable à différents emplacements spatiaux, en fonction de la distance qui les sépare. Il fournit ainsi des informations sur la manière dont la similarité entre les valeurs de la variable diminue à mesure que la distance spatiale entre les emplacements augmente.

The Variogram γ_z of the variable z is equal to half of the mean squared difference between the values of z for two points (x_i, y_i) and (x_j, y_j) separeted by a distance h.

$$\gamma_z(h) = \frac{1}{2}\mathrm{E}\left[(z(x_i, y_i) - z(x_j, y_j))^2\right]_{d_{ij}=h}$$

In this equation, the function E with the term $d_{ij} = h$ denotes the statistical expectation (or mean) of the squared difference between the values of z or points separated by a distance h.

If we prefer to express the autocorrelation $\rho_z(h)$ between measurements of z separated by a distance h, it is related to the variogram by the equation:

$$\gamma_z = \sigma_z^2(1 - \rho_z)$$

,

où σ_z^2 est la variance globale de z.

Theoretical Models of the Variogram

Several parametric models have been proposed to represent spatial correlation as a function of the distance between sampling points. Let's first consider a correlation that decreases exponentially.

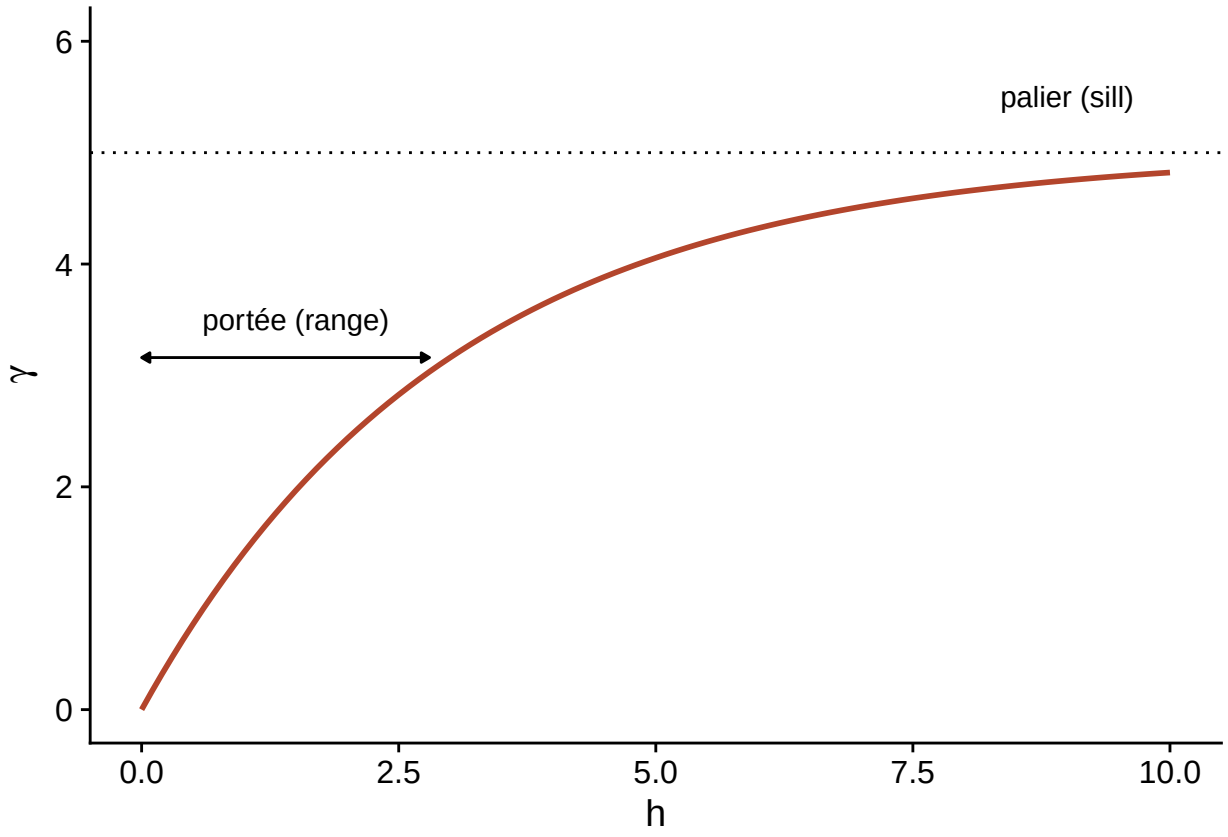
$$\rho_z(h) = e^{-h/r}$$

here, $\rho_z = 1$ for $h = 0$and the correlation multiplied by $1/e \approx 0.37$ or each increase of r in distance. In this context, r is the (*range*) of the correlation

From the above mentioned regression we can compute the correspoding variagram:

$$\gamma_z(h) = \sigma_z^2(1 - e^{-h/r})$$

Here a graphical representation of this variogram:

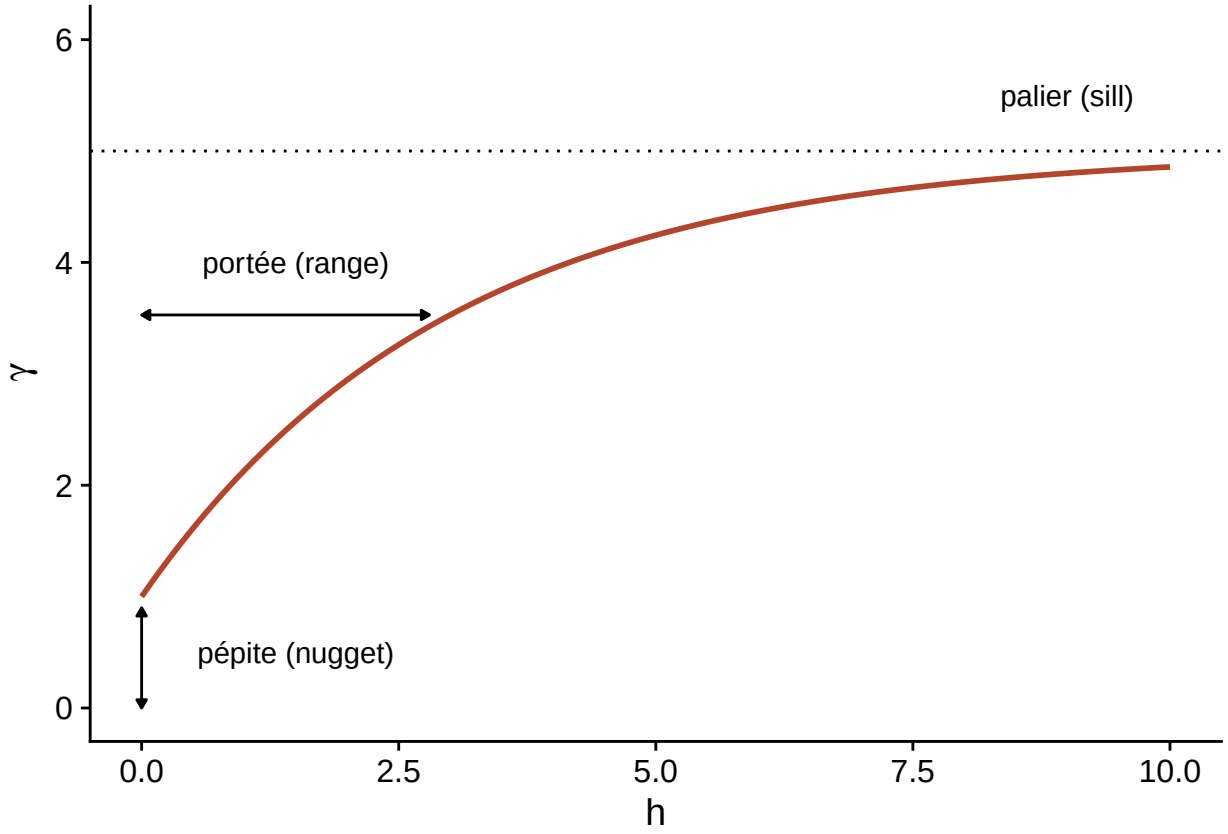


Due to the exponential function, the value of γ approaches the global variance σ_z^2 at large distances without exactly reaching it. This asymptote is called the **sill** in geostatistical context and is represented by the symbol s.

Finally, it is sometimes unrealistic to assume perfect correlation when the distance is zero, due to possible variation. We can add a *nugget* effect, noted n, so that $\gamma = n$ if $h = 0$. The term "nugget" comes from the mining origin of these techniques, where a nugget of ore could be the source of a sudden variation in concentration at a small scale.

By adding the nugget effect, the rest of the variogram is "compressed" to maintain the same sill, resulting in the following equation.

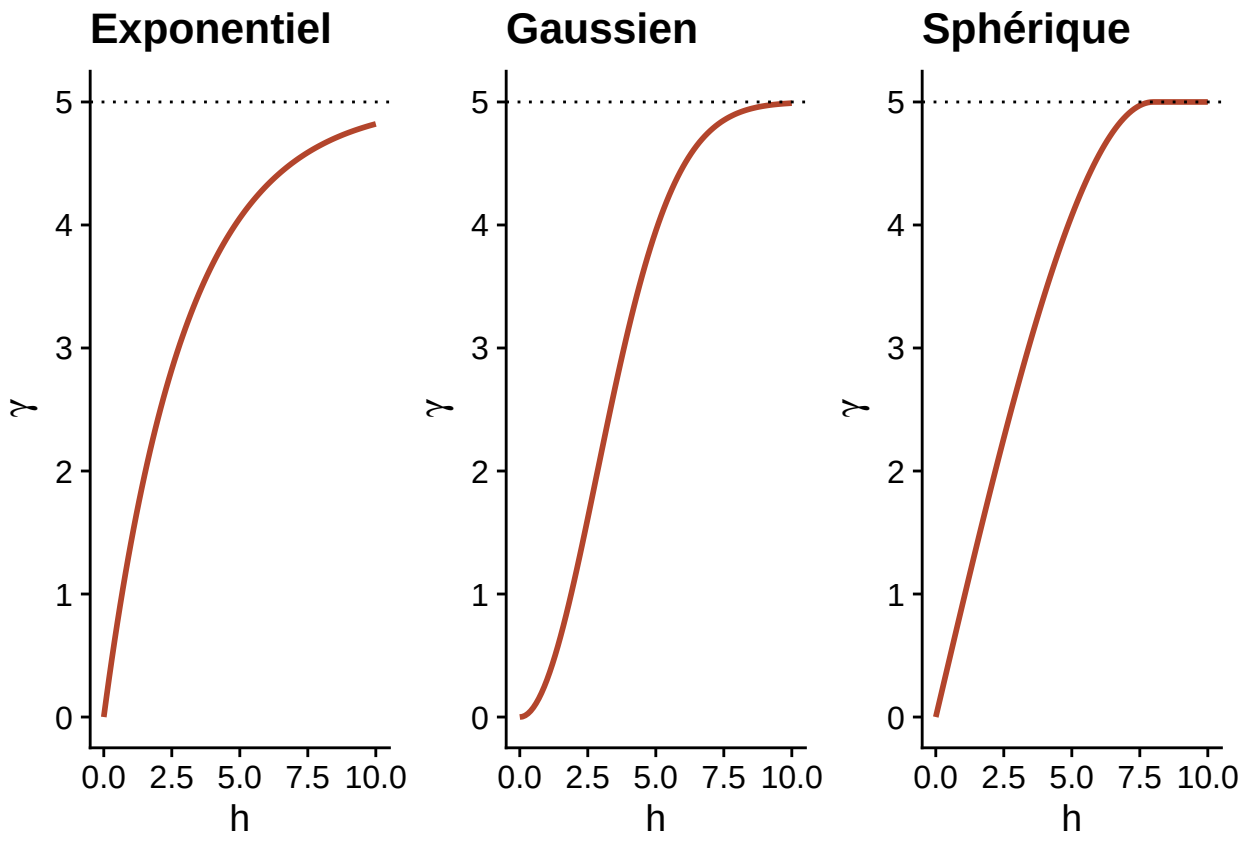
$$\gamma_z(h) = n + (s - n)(1 - e^{-h/r})$$



In addition to the exponential model, two other common theoretical models for the variogram are the Gaussian model (where the correlation follows a half-normal curve), and the spherical model (where the variogram increases linearly initially, then bends and reaches the sill at a distance equal to its range r). The spherical model allows the correlation to be exactly 0 at large distances, rather than gradually approaching zero as in the case of the other models.

Modèle	$\rho(h)$	$\gamma(h)$
Exponentiel	$\exp\left(-\frac{h}{r}\right)$	$s\left(1 - \exp\left(-\frac{h}{r}\right)\right)$
Gaussien	$\exp\left(-\frac{h^2}{r^2}\right)$	$s\left(1 - \exp\left(-\frac{h^2}{r^2}\right)\right)$
Sphérique ($h < r$) *	$1 - \frac{3}{2}\frac{h}{r} + \frac{1}{2}\frac{h^3}{r^3}$	$s\left(\frac{3}{2}\frac{h}{r} - \frac{1}{2}\frac{h^3}{r^3}\right)$

* Pour le modèle sphérique, $\rho = 0$ et $\gamma = s$ si $h \geq r$.



Empirical Variogram

To estimate $\gamma_z(h)$ from empirical data, we need to define distance classes, grouping different distances within a margin $\pm\delta$ around a distance h, and then calculate the mean squared difference for pairs of points in this distance class.

$$\hat{\gamma_z}(h) = \frac{1}{2N_{\text{paires}}} \sum \left[(z(x_i, y_i) - z(x_j, y_j))^2 \right]_{d_{ij}=h\pm\delta}$$

Nous verrons dans la partie suivante comment estimer un variogramme dans R.

Variogram and Temporal Data

A variogram can also be estimated based on differences in time, which is in this case considered as a one-dimensional space. This allows modeling temporal dependence for a series of measurements taken at irregular intervals, when autoregressive models seen in the last class do not apply.

Regression Model with Spatial Correlation

The following equation represents a multiple linear regression including residual spatial correlation:

$$v = \beta_0 + \sum_i \beta_i u_i + z + \epsilon$$

Here, v denote the response variable and u the predictor, to avoid confusion with spatial coordinates x and y

In addition to the independent residual ϵ between observations, the model includes a term z representing the spatially correlated portion of the residual variance

Here are suggested steps to follow to apply this type of model:

-Fit the regression model without spatial correlation. -Check for the presence of spatial correlation using the empirical variogram of the residuals. -Fit one or more regression models with spatial correlation. Comparison using AIC may be necessary to choose the form of correlation.

We will see in the last part of the course how to include spatial correlation terms in complex models, such as mixed or Bayesian models.

Geostatistical Models in R

Geostatistical models aim to represent the spatial distribution of continuous variables measured at certain sampling points. They assume that measurements of these variables at different points are correlated based on the distance between these points. Among the applications of geostatistical models are spatial data smoothing (e.g., producing a map of a variable over an entire region based on point measurements) and prediction of these variables for unsampled points.

The **gstat** package contains functions related to geostatistics. For this example, we will use the dataset "oxford" from this package, which contains measurements of physical and chemical properties for 126 soil samples from a site, along with their coordinates "XCOORD" and "YCOORD".

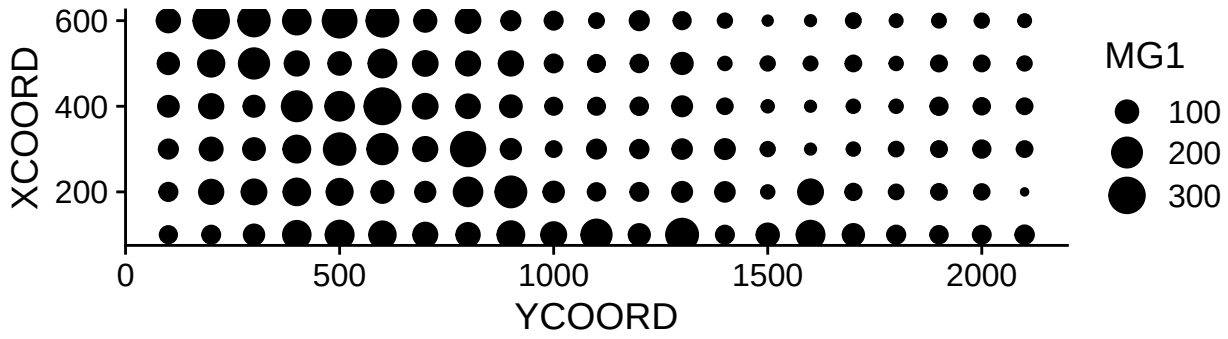
```
library(gstat)

data(oxford)
str(oxford)
```

```
## 'data.frame':    126 obs. of  22 variables:
##  $ PROFILE  : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ XCOORD   : num  100 100 100 100 100 100 100 100 100 100 ...
##  $ YCOORD   : num  2100 2000 1900 1800 1700 1600 1500 1400 1300 1200 ...
##  $ ELEV     : num  598 597 610 615 610 595 580 590 598 588 ...
##  $ PROFCLASS: Factor w/ 3 levels "Cr","Ct","Ia": 2 2 2 3 3 2 3 2 3 3 ...
##  $ MAPCLASS : Factor w/ 3 levels "Cr","Ct","Ia": 2 3 3 3 3 2 2 3 3 3 ...
##  $ VAL1     : num  3 3 4 4 3 3 4 4 4 3 ...
##  $ CHR1     : num  3 3 3 3 3 2 2 3 3 3 ...
##  $ LIME1    : num  4 4 4 4 4 0 2 1 0 4 ...
##  $ VAL2     : num  4 4 5 8 8 4 8 4 8 8 ...
##  $ CHR2     : num  4 4 4 2 2 4 2 4 2 2 ...
##  $ LIME2    : num  4 4 4 5 5 4 5 4 5 5 ...
##  $ DEPTHCM  : num  61 91 46 20 20 91 30 61 38 25 ...
##  $ DEP2LIME : num  20 20 20 20 20 20 20 20 40 20 ...
##  $ PCLAY1   : num  15 25 20 20 18 25 25 35 35 12 ...
##  $ PCLAY2   : num  10 10 20 10 10 20 10 20 10 10 ...
##  $ MG1      : num  63 58 55 60 88 168 99 59 233 87 ...
##  $ OM1      : num  5.7 5.6 5.8 6.2 8.4 6.4 7.1 3.8 5 9.2 ...
##  $ CEC1     : num  20 22 17 23 27 27 21 14 27 20 ...
##  $ PH1      : num  7.7 7.7 7.5 7.6 7.6 7 7.5 7.6 6.6 7.5 ...
##  $ PHOS1    : num  13 9.2 10.5 8.8 13 9.3 10 9 15 12.6 ...
##  $ POT1     : num  196 157 115 172 238 164 312 184 123 282 ...
```

Supposons que nous souhaitons modéliser la concentration de magnésium (`MG1`), représentée en fonction de la position spatiale dans le graphique suivant.

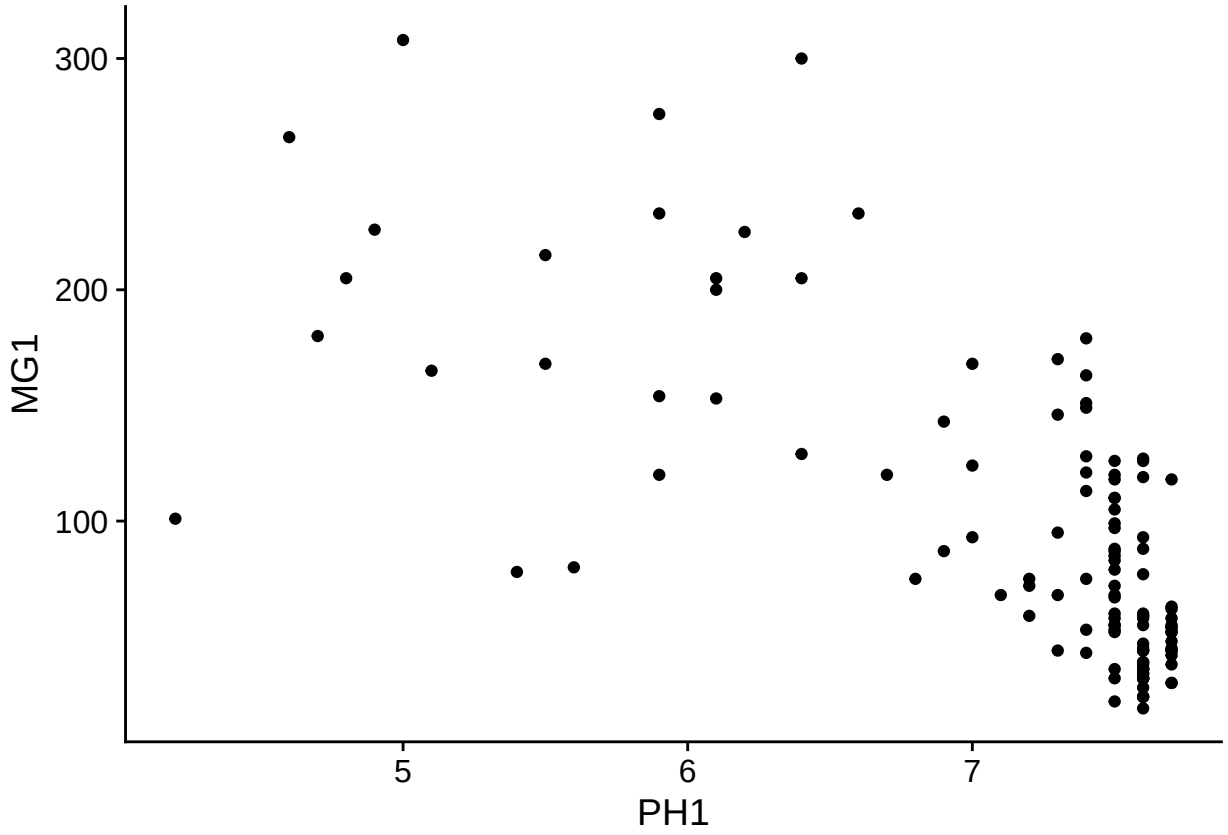
```
library(ggplot2)
ggplot(oxford, aes(x = YCOORD, y = XCOORD, size = MG1)) +
    geom_point() +
    coord_fixed()
```



Notez que les axes x et y ont été inversés par souci d'espace. La fonction `coord_fixed()` de *ggplot2* assure que l'échelle soit la même sur les deux axes, ce qui est utile pour représenter des données spatiales.

Nous voyons tout de suite que ces mesures ont été prises sur une grille de 100 m de côté. Il semble que la concentration de magnésium soit spatialement corrélée, bien qu'il puisse s'agir d'une corrélation induite par une autre variable. Nous savons notamment que la disponibilité de l'ion magnésium est reliée (négativement) au pH du sol (`PH1`).

```
ggplot(oxford, aes(x = PH1, y = MG1)) +
    geom_point()
```



La fonction `variogram` de *gstat* sert à estimer un variogramme à partir de données empiriques. Voici le résultat obtenu pour la variable `MG1`.

```
require(gstat)

var_mg <- variogram(MG1 ~ 1, locations = ~ XCOORD + YCOORD, data = oxford)
var_mg
```

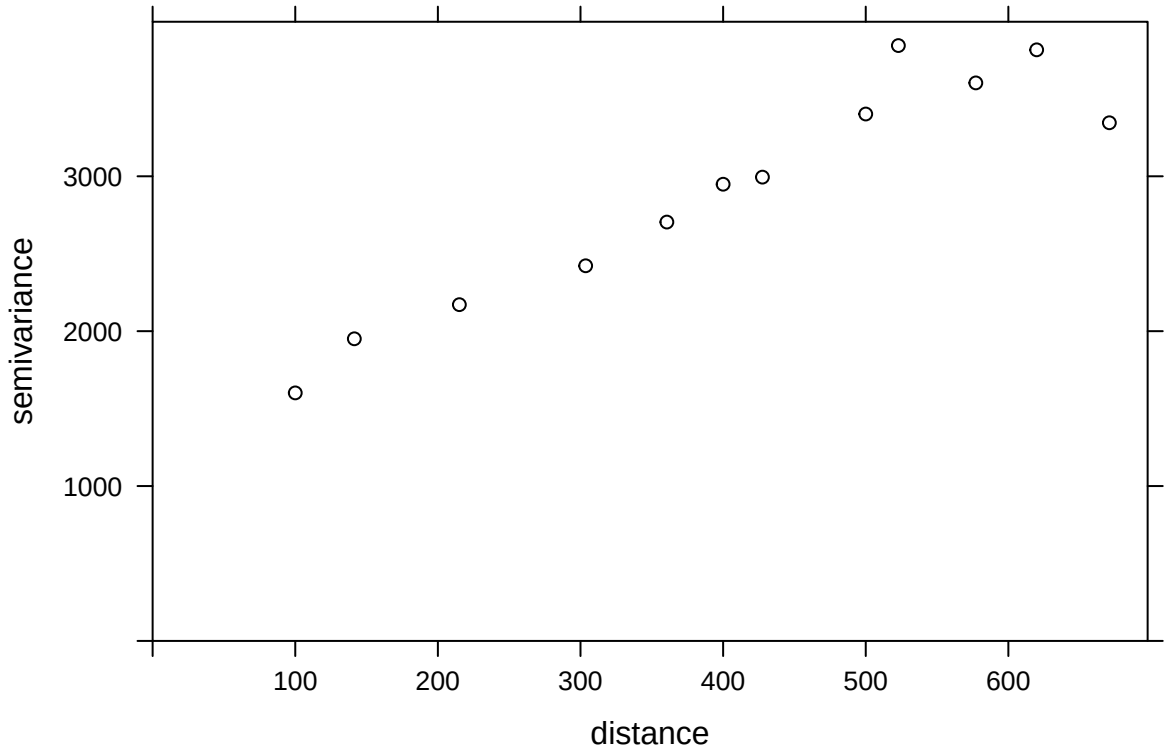
```
##     np     dist    gamma dir.hor dir.ver   id
## 1  225 100.0000 1601.404       0       0 var1
## 2  200 141.4214 1950.805       0       0 var1
## 3  548 215.0773 2171.231       0       0 var1
## 4  623 303.6283 2422.245       0       0 var1
## 5  258 360.5551 2704.366       0       0 var1
## 6  144 400.0000 2948.774       0       0 var1
## 7  570 427.5569 2994.621       0       0 var1
## 8  291 500.0000 3402.058       0       0 var1
## 9  366 522.8801 3844.165       0       0 var1
## 10 200 577.1759 3603.060       0       0 var1
## 11 458 619.8400 3816.595       0       0 var1
## 12  90 670.8204 3345.739       0       0 var1
```

La formule `MG1 ~ 1` indique qu'aucun prédicteur linéaire n'est inclus dans ce modèle, tandis que l'argument `locations` indique quelles variables du tableau correspondent aux coordonnées spatiales.

Dans le tableau obtenu, `gamma` est la valeur du variogramme pour la classe de distance centrée sur `dist`, tandis que `np` est le nombre de paires de points dans cette classe. Ici, puisque les points sont situés sur une grille, nous obtenons des classes de distance régulières (ex.: 100 m pour les points voisins sur la grille, 141 m

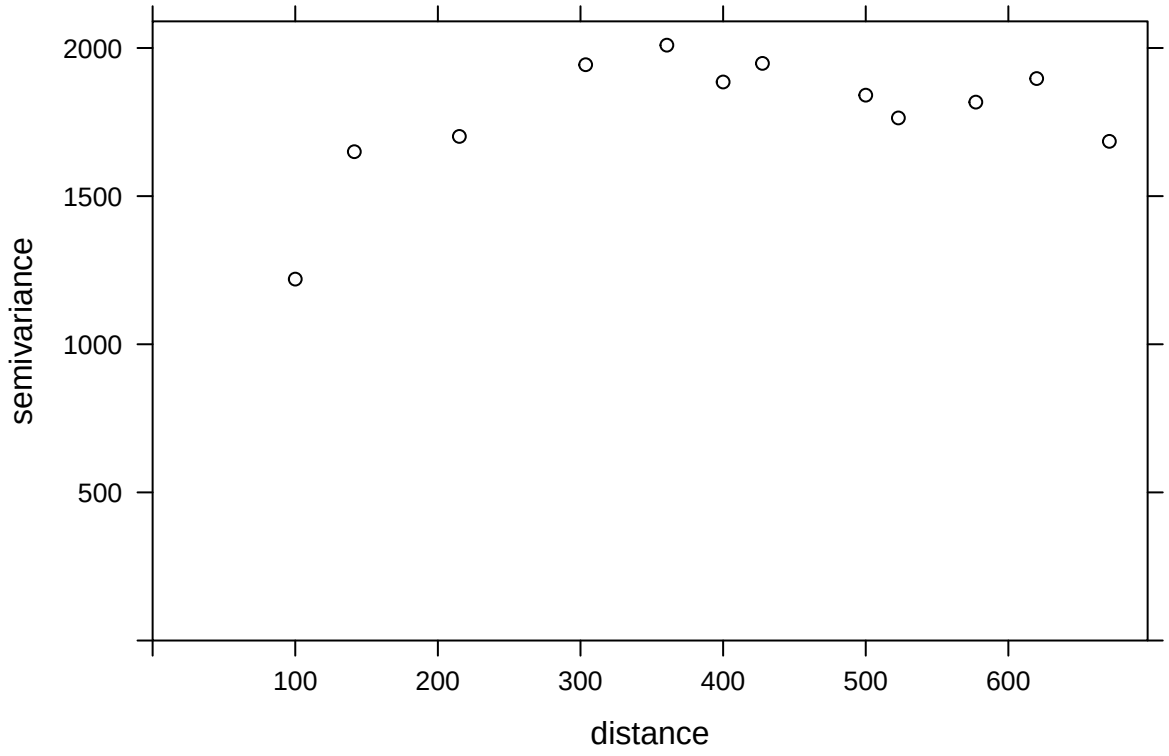
pour les voisins en diagonale, etc.) Nous pouvons illustrer le variogramme avec `plot`.

```
plot(var_mg, col = "black")
```



Si nous voulons estimer la corrélation spatiale résiduelle de `MG1` après avoir inclus l'effet de `PH1`, nous pouvons ajouter ce prédicteur à la formule.

```
var_mg <- variogram(MG1 ~ PH1, locations = ~ XCOORD + YCOORD, data = oxford)
plot(var_mg, col = "black")
```



En incluant l'effet du pH, la portée de la corrélation spatiale semble diminuer, alors que le plateau est atteint autour de 300 m. Il semble même que le variogramme diminue au-delà de 400 m. En général, nous supposons que la variance entre deux points ne diminue pas avec la distance, à moins d'avoir un patron spatial périodique.

La fonction `fit.variogram` accepte comme arguments un variogramme estimé à partir des données, ainsi qu'un modèle théorique décrit dans une fonction `vgm`, puis estime les paramètres de ce modèle en fonction des données. L'ajustement se fait par la méthode des moindres carrés.

Par exemple, `vgm("Exp")` indique d'ajuster un modèle exponentiel.

```
vfit <- fit.variogram(var_mg, vgm("Exp"))
vfit
```

```
##   model    psill    range
## 1   Nug    0.000  0.00000
## 2   Exp 1951.496 95.11235
```

Il n'y a aucun effet de pépite, car `psill = 0` pour la partie `Nug` (*nugget*) du modèle. La partie exponentielle a un palier à 1951 (correspondant à σ_z^2) et une portée de 95 m.

Pour comparer différents modèles, on peut donner un vecteur de noms de modèles à `vgm`. Dans l'exemple suivant, nous incluons les modèles exponentiel, gaussien ("Gau") et sphérique ("Sph").

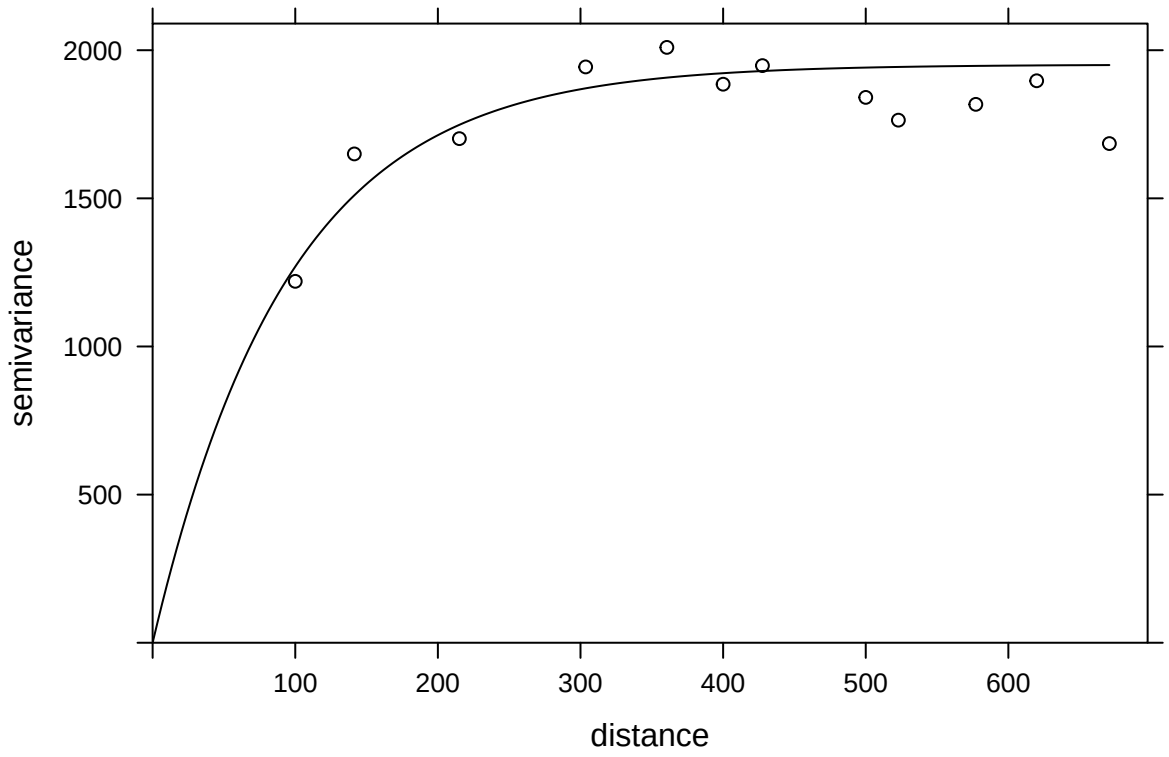
```
vfit <- fit.variogram(var_mg, vgm(c("Exp", "Gau", "Sph")))
vfit
```

```
##   model    psill    range
## 1   Nug    0.000  0.00000
## 2   Exp 1951.496 95.11235
```

Le modèle exponentiel demeure le mieux ajusté.

Finalement, nous pouvons superposer le modèle théorique et le variogramme empirique sur un même graphique.

```
plot(var_mg, vfit, col = "black")
```



Nota bene:
Le variogramme can provide information to help choose the distance at which to consider neighboring points in calculating the Moran's I statistic, based on the spatial correlation structure of the data. For example, in contexts where spatial correlation decreases slowly with distance or where complex spatial patterns are present, you may expect to need a larger k to adequately capture the spatial dependence structure of the data. However, the choice of k also depends on the sample size (larger k to account for a larger number of neighbors and obtain a more robust representation of spatial correlation); specific objectives of the analysis (e.g., global trends, patch structures, or spatial gradients), you may need a larger k to capture these patterns. If you are interested in a more local (small k) or global (large k) analysis of spatial dependence, this will influence the number of neighbors to consider.

Spatial autoregression models

Let us recall the formula for a linear regression with spatial dependence:

$$v = \beta_0 + \sum_i \beta_i u_i + z + \epsilon$$

,

where z is the portion of the residual variance that is spatially correlated.

There are two main types of autoregressive models to represent the spatial dependence of z: conditional autoregression (CAR) and simultaneous autoregressive (SAR).

Conditional autoregressive (CAR) model

In the conditional autoregressive model, the value of z_i for the region i follows a normal distribution: its mean depends on the value z_j of neighbouring regions, multiplied by the weight w_{ij} and a correlation coefficient ρ; its standard deviation σ_{z_i} may vary from one region to another.

$$z_i \sim \mathrm{N}\left(\sum_j \rho w_{ij} z_j, \sigma_{z_i}\right)$$

In this model, if w_{ij} is a binary matrix (0 for non-neighbours, 1 for neighbours), then ρ is the coefficient of partial correlation between neighbouring regions. This is similar to a first-order autoregressive model in the context of time series, where the autoregression coefficient indicates the partial correlation.

Simultaneous autoregressive (SAR) model

In the simultaneous autoregressive model, the value of z_i is given directly by the sum of contributions from neighbouring values z_j, multiplied by ρw_{ij}, with an independent residual ν_i of standard deviation σ_z.

$$z_i = \sum_j \rho w_{ij} z_j + \nu_i$$

At first glance, this looks like a temporal autoregressive model. However, there is an important conceptual difference. For temporal models, the causal influence is directed in only one direction: $v(t-2)$ affects $v(t-1)$ which then affects $v(t)$. For a spatial model, each z_j that affects z_i depends in turn on z_i. Thus, to determine the joint distribution of z, a system of equations must be solved simultaneously (hence the name of the model).

For this reason, although this model resembles the formula of CAR model, the solutions of the two models differ and in the case of SAR, the coefficient ρ is not directly equal to the partial correlation due to each neighbouring region.

For more details on the mathematical aspects of these models, see the article by Ver Hoef et al. (2018) suggested in reference.

For the moment, we will consider SAR and CAR as two types of possible models to represent a spatial correlation on a network. We can always fit several models and compare them with the AIC to choose the best form of correlation or the best weight matrix.

The CAR and SAR models share an advantage over geostatistical models in terms of efficiency. In a geostatistical model, spatial correlations are defined between each pair of points, although they become negligible as distance increases. For a CAR or SAR model, only neighbouring regions contribute and most weights are equal to 0, making these models faster to fit than a geostatistical model when the data are massive.

Finally, note that there is also a spatial equivalent of moving average (MA) models seen in a temporal context. However, since their application is rarer, we do not discuss them in this course.

Areal data in R

To illustrate the analysis of areal data in R, we load the packages *spData* (containing examples of spatial data), *spdep* (to define spatial networks and calculate the Moran index) and *spatialreg* (for SAR and CAR models).

```
library(spData)
library(spdep)
library(spatialreg)
```

As an example, we will use the `us_states` spatial dataset which contains polygons for 49 U.S. states (all states excluding Alaska and Hawaii, plus the District of Columbia).

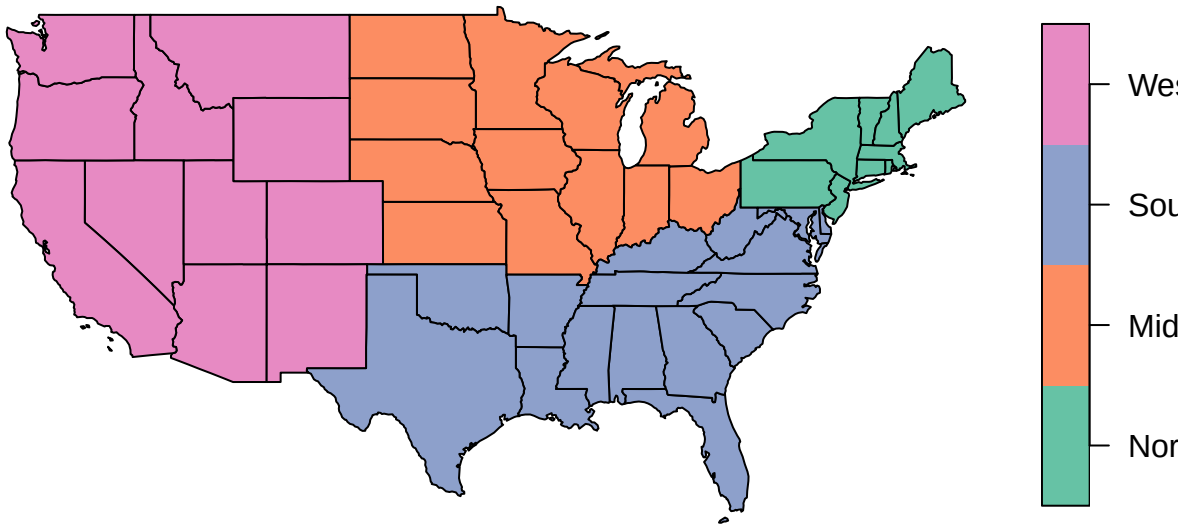
```
data(us_states)
head(us_states)
```

```
## Simple feature collection with 6 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -114.8136 ymin: 24.55868 xmax: -71.78699 ymax: 42.04964
## Geodetic CRS:  NAD83
##   GEOID        NAME   REGION             AREA total_pop_10 total_pop_15
## 1    01     Alabama    South 133709.27 [km^2]      4712651      4830620
## 2    04     Arizona     West 295281.25 [km^2]      6246816      6641928
## 3    08    Colorado     West 269573.06 [km^2]      4887061      5278906
## 4    09 Connecticut Norteast  12976.59 [km^2]      3545837      3593222
## 5    12     Florida    South 151052.01 [km^2]     18511620     19645772
## 6    13     Georgia    South 152725.21 [km^2]      9468815     10006693
##                          geometry
## 1 MULTIPOLYGON (((-88.20006 3...
## 2 MULTIPOLYGON (((-114.7196 3...
## 3 MULTIPOLYGON (((-109.0501 4...
## 4 MULTIPOLYGON (((-73.48731 4...
## 5 MULTIPOLYGON (((-81.81169 2...
## 6 MULTIPOLYGON (((-85.60516 3...
```

It is a spatial data frame where the last column defines the polygon corresponding to the state and the other columns define variables associated with it. We will not discuss this data structure in detail, but note that the *sf* packge allows you to import vector GIS files (shapefiles) in this data format for R.

To illustrate one of the variables on a map, we call the function `plot` with the column name in square brackets and quotes.

```
plot(us_states["REGION"])
```



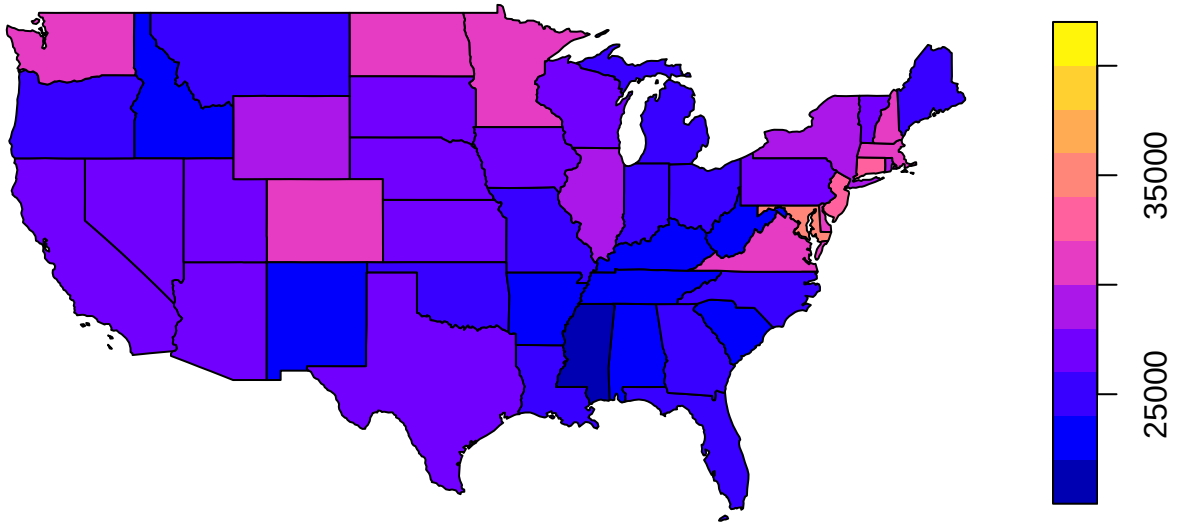
Here we want to model the median income in each state in 2015. This variable `median_income_15` is found in another dataset, `us_states_df`.

```
data(us_states_df)
head(us_states_df)
```

```
## # A tibble: 6 x 5
##   state      median_income_10 median_income_15 poverty_level_10 poverty_level_15
##   <chr>                 <dbl>            <dbl>            <dbl>            <dbl>
## 1 Alabama               21746            22890           786544           887260
## 2 Alaska                29509            31455            64245            72957
## 3 Arizona               26412            26156           933113          1180690
## 4 Arkansas              20881            22205           502684           553644
## 5 California            27207            27035          4919945          6135142
## 6 Colorado              29365            30752           584184           653969
```

We use the `inner_join` function of *dplyr* to join the two datasets, specifying with `by` that the *NAME* column of `us_states` matches the *state* column of `us_states_df`.

```
library(dplyr)
us_states <- inner_join(us_states, us_states_df,
                        by = c("NAME" = "state"))
plot(us_states["median_income_15"])
```



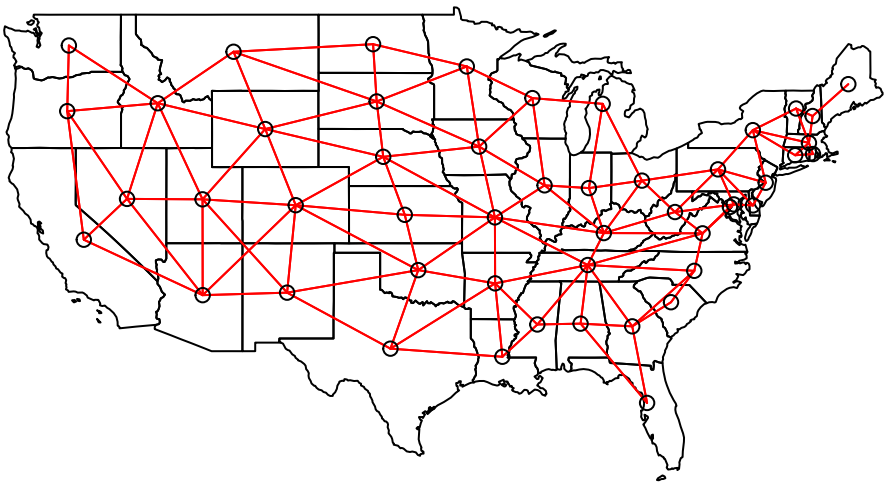
The `poly2nb` function of the *spdep* package defines a neighbourhood network from polygons. The result `vois` is a list of 49 elements where each element contains the indices of the neighbouring polygons of a given polygon.

```
vois <- poly2nb(us_states)
vois[[1]]
```

```
## [1]  5  6 36 44
```

We can illustrate this network by extracting the coordinates of the center of each state, creating a blank map with `plot(us_states["geometry"])`, then adding the network as an additional layer with `plot(vois, add = TRUE, coords = coords)`.

```
coords <- st_centroid(us_states) %>%
    st_coordinates()
plot(us_states["geometry"])
plot(vois, add = TRUE, col = "red", coords = coords)
```



We still have to add weights to each network link with the `nb2listw` function. The style of weight "B" corresponds to binary weights, i.e. 1 for the presence of a link and 0 for the absence of a link between two states.

Once these weights have been defined, we can verify if there is a significant autocorrelation of the median income between neighbouring states with the Moran test.

```
poids <- nb2listw(vois, style = "B")

moran.test(us_states$median_income_15, poids)
```

```
##
##  Moran I test under randomisation
##
## data:  us_states$median_income_15
## weights: poids
##
## Moran I statistic standard deviate = 4.127, p-value = 1.838e-05
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.342670652      -0.020833333       0.007758162
```

The value $I = 0.34$ is very significant judging by the p-value of the test.

Finally, we fit SAR and CAR models to these data with the 'spautolm' (*spatial autoregressive linear model*) function of *spatialreg*. Here is the code for a SAR model including the fixed effect of the region (west, mid-west, south or north-east).

```
modsp <- spautolm(median_income_15 ~ REGION, data = us_states,
                  listw = poids)
summary(modsp)
```

```
##
## Call: spautolm(formula = median_income_15 ~ REGION, data = us_states,
##     listw = poids)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5632.95 -2243.24  -856.84  1781.90 11770.13
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   28703.411   1607.825 17.8523   <2e-16
## REGIONMidwest    42.832   2157.389  0.0199   0.9842
## REGIONSouth   -1131.312   1904.512 -0.5940   0.5525
## REGIONWest     -815.428   2393.545 -0.3407   0.7333
##
## Lambda: 0.12915 LR test value: 7.7579 p-value: 0.0053479
## Numerical Hessian standard error of lambda: 0.033239
##
## Log likelihood: -466.35
## ML residual variance (sigma squared): 9804100, (sigma: 3131.2)
## Number of observations: 49
## Number of parameters estimated: 6
## AIC: 944.7
```

The value given by `Lambda` in the summary corresponds to the coefficient ρ in our description of the model. The likelihood ratio test (`LR test`) confirms that this residual spatial correlation (after accounting for the region effect) is significant.

To evaluate a CAR rather than SAR model, we must specify `family = "CAR"`.

```
modsp2 <- spautolm(median_income_15 ~ REGION, data = us_states,
                  listw = poids, family = "CAR")
summary(modsp2)
```

```
##
## Call: spautolm(formula = median_income_15 ~ REGION, data = us_states,
##     listw = poids, family = "CAR")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5709.20 -1999.90  -682.38  2072.01 11328.25
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    29022.2     1484.4 19.5519   <2e-16
## REGIONMidwest   -408.0     2049.4 -0.1991   0.8422
## REGIONSouth    -1436.2     1820.6 -0.7888   0.4302
## REGIONWest     -1378.0     2239.7 -0.6153   0.5384
##
## Lambda: 0.16539 LR test value: 5.8956 p-value: 0.015179
## Numerical Hessian standard error of lambda: 0.031414
```

```
##
## Log likelihood: -467.2811
## ML residual variance (sigma squared): 10168000, (sigma: 3188.7)
## Number of observations: 49
## Number of parameters estimated: 6
## AIC: 946.56
```

For a CAR model with binary weights, the value of `Lambda` (which we called ρ) directly gives the partial correlation coefficient between neighbouring states. Note that the AIC here is slightly higher than the SAR model, so the latter gave a better fit.

Spatial correlation in complex models

As for the class on time series, we conclude this class by presenting some avenues for integrating spatial correlations into more complex models.

Geostatistical models with *nlme*

In the last course, we saw that the `lme` function of the package *nlme* allowed us to include temporal correlations with terms of type `corARMA`. The same package contains spatial correlation functions, including exponential (`corExp`), Gaussian (`corGaus`) and spherical (`corSpher`) correlation.

Here is an example of a mixed linear model fitted with `lme`, where `v` is the response, `u` is a fixed effect and `group` is a random effect. The `correlation` argument indicates an exponential correlation as a function of the distance defined by the coordinates `x` and `y`, with a nugget effect `nugget = TRUE`.

```
library(nlme)
mod <- lme(v ~ u, data, random = list(groupe = ~1),
           correlation = corExp(form = ~ x + y, nugget = TRUE))
```

Note that the limitations of the *nlme* package mentioned in the last class still apply here: it is not possible to include several crossed (non-nested) effects and the package is not very efficient for estimating generalized models.

To add spatial correlation to a linear model, without random effects, we can replace `lme` by `gls`, for *generalized least squares*. This function is similar to `lm`, but allows correlations between model residuals.

```
library(nlme)
mod <- gls(v ~ u, data,
           correlation = corExp(form = ~ x + y, nugget = TRUE))
```

Finally, as we saw in the last course, the `gamm` function of the *mgcv* package combines the functionality of `lme` with the possibility to include additive effects (smoothing splines) for the predictors.

```
library(mgcv)
mod <- gamm(v ~ s(u), data, random = list(groupe = ~1),
            correlation = corExp(form = ~ x + y, nugget = TRUE))
```

Geostatistical models with *brms*

To include spatial correlation of the geostatistical type in a Bayesian model estimated with *brms*, we must specify a `gp` term, which describes a Gaussian process.

```
library(brms)
mod <- brm(v ~ u + gp(x, y, cov = "exp_quad"), data)
```

The term `gp` indicates the variables containing the spatial coordinates (`x`, `y`) as well as the form of the covariance. Currently, only the Gaussian correlation (`exp_quad`, for *exponential quadratic*) is available.*

* Gaussian processes with other correlation functions are possible, if they are manually coded with Stan. The term "Gaussian" in "Gaussian process" refers to the normal error distribution, not to the form of the spatial correlation.

Spatial autoregressive models with *brms*

On the other hand, the `brm` function allows us to specify a spatial autoregressive structure with `sar` and `car` terms, which is useful to combine a spatial autoregressive model with non-spatial random effects. The `sar` and `car` terms are only allowed in models where the response follows a normal or t distribution, so they cannot be combined with generalized models.

```
library(brms)
mod_sar <- brm(v ~ u + sar(W, type = "error"), data, data2 = list(W = W))
mod_car <- brm(v ~ u + car(W), data, data2 = list(W = W))
```

- `W` is the weight matrix. Since this matrix is not part of the `data`, it is given separately in the `data2` argument.
- The argument `type = "error"` in `sar` represents the type of SAR model seen in this course, where the unexplained portion of the response is autocorrelated. There are other types of SAR, including those where the value of the response itself is autocorrelated.

References

This course has been modified from Philippe Marchand's course, which was given on January 12, 2021. The original course is still available on the website https://bios2.github.io/posts/2021-01-12-4-day-training-in-spatial-statistics-with-philippe-marchand/#statistiques-spatiales-en-%C3%A9cologie

This course is only a brief introduction to the main spatial analysis techniques useful in environmental science. To go further in this field, Fortin and Dale's textbook provides a very comprehensive overview of these and other methods.

Fortin, M.-J. and Dale, M.R.T. (2005) *Spatial Analysis: A Guide for Ecologists*. Cambridge University Press: Cambridge, UK.

Ver Hoef, J.M., Peterson, E.E., Hooten, M.B., Hanks, E.M. and Fortin, M.-J. (2018) Spatial autoregressive models for statistical inference from ecological data. *Ecological Monographs* 88: 36-59.

Wiegand, T. and Moloney, K.A. (2013) *Handbook of Spatial Point-Pattern Analysis in Ecology*, CRC Press.

Ce cours a été modifié à partir du cours de Philippe Marchand qui a été donnée le 12 janvier 2021. Le cours originale est encore disponible dans le site https://bios2.github.io/posts/2021-01-12-4-day-training-in-spatial-statistics-with-philippe-marchand/#statistiques-spatiales-en-%C3%A9cologie

Ce cours ne constitue qu'une brève introduction aux principales techniques d'analyse spatiale utiles en sciences de l'environnement. Pour aller plus loin dans ce domaine, le manuel de Fortin et Dale donne un portrait très complet de ces méthodes et d'autres.

Fortin, M.-J. et Dale, M.R.T. (2005) *Spatial Analysis: A Guide for Ecologists*. Cambridge University Press: Cambridge, UK.

Ver Hoef, J.M., Peterson, E.E., Hooten, M.B., Hanks, E.M. et Fortin, M.-J. (2018) Spatial autoregressive models for statistical inference from ecological data. *Ecological Monographs* 88: 36-59.

Wiegand, T. et Moloney, K.A. (2013) *Handbook of Spatial Point-Pattern Analysis in Ecology*, CRC Press.