

# Spatial Data: a brief introduction

OpenSpat – Class notes

Jorge Cadima

Instituto Superior de Agronomia – Universidade de Lisboa

June 7 and 8, 2018

# Goals

- Focus on the statistical processing of spatial data
- Understanding the effects of autocorrelation vs. independence
- Tools to assess and measure spatial autocorrelation
- Other tools for spatial statistical models

# Spatial Data

- **Spatial data** is data observed over some **spatial coordinate system** and with **autocorrelation** that cannot be ignored.
- A **random spatial process** generalizes the notion of a **random variable**  $Z$ : now  $Z$  is defined on a **space**  $\mathcal{S}$ :

$$\{Z(s) , s \in \mathcal{S}\} .$$

If  $\mathcal{S}$  is **one-dimensional**,  $Z(s)$  is also called a **time series**.

- $\mathcal{S}$  may be a finite or infinite (countable or uncountable) set.
- We **observe** the random process at  $n$  different locations  $s_1, s_2, \dots, s_n \in \mathcal{S}$ , producing a (**non-independent**) **sample of size**  $n$ :

$$(Z(s_1), Z(s_2), Z(s_3), \dots, Z(s_n))$$

# The nature of a spatial variable $Z$

As in standard statistical methods, a spatial variable  $Z$  may be of different types:

- numerical (e.g., air temperature);
- categorical (e.g., types of land use over a given region  $\mathcal{S}$ );
- ordinal (e.g., the intensity of some disease affecting crops in a region  $\mathcal{S}$ , with  $k$  ordered categories of observable effects);
- binary (e.g., absence/presence of some plant disease).

# Geostatistical data

A spatial variable may also be classified in a different way, taking into account its **spatial characteristics**:

## Geostatistical data:

- $s$  varies continuously in  $\mathcal{S}$ , and for any point  $s \in \mathcal{S}$  a value  $Z(s)$  exists, even if it is unknown.
- **Example**: air temperature over some region of earth.
- A common problem is **interpolation**: based on an available set of data  $\{Z(s_{ij})\}_{i,j}$ , obtain estimates for the values of  $Z$  in unobserved locations of  $\mathcal{S}$ .

We will focus on geostatistical data.

# Lattice (areal) and point data

## Lattice (areal) data:

- $Z$  only makes sense when  $\mathcal{S}$  is a collection of polygons or cells, distributed over space  $\mathcal{S}$
- **Example:** surface area of countries (or municipalities).
- The polygons may be represented by a label point or centroid, but this does not change the areal nature of variable  $Z$ .

## Point data:

- the location of points in space at which something happens.
- **Example:** location of cities in a region, or of trees in a wooded area.
- The main topic of interest is often the study of the point patterns defined by the data.
- Problems and methods for point data are somewhat different.

# Autocorrelation vs. independence

- Standard statistical methods assume **independence** of observations.
- **Independence is simpler. But it may be unrealistic.**
- Observing air temperature at a given location, every 10 minutes, generates a **sample of size  $t$** :

$$(Y_1, Y_2, Y_3, \dots, Y_t)$$

The observations are **not** independent. They have **one-dimensional autocorrelation (in time)**.

- Observing, at a given instant in time, air temperatures in a given rectangular grid of  $n_1 \times n_2$  points in space produces **two-dimensional (spatial) autocorrelation**.

# Autocorrelation

- **Positive autocorrelation** means that observations of a variable made at points that are close to each other (in time and/or space) will be more similar than observations at points that are further apart.
- **Negative autocorrelation** means that observations made at points that are closer will be more dissimilar than observations made further apart. It also exists, but is rarer and will not be considered further.
- The **absence of autocorrelation** means that the distance between points of observation has no bearing on whether values are similar or dissimilar. This is what assuming independence implies.



# Why worry about autocorrelation?

Even the simplest independence-based statistical methods give wrong results in the presence of autocorrelation in the sample.

Consider the standard  $(1 - \alpha) \times 100\%$  confidence interval for a population mean  $\mu$ , when the population variance  $\sigma^2$  is known:

$$\left[ \bar{y} - z_{\alpha/2} \frac{\sigma}{\sqrt{n}} , \bar{y} + z_{\alpha/2} \frac{\sigma}{\sqrt{n}} \right]. \quad (1)$$

This result is based on the assumption that the sample of size  $n$  has independent observations.

But what if there is autocorrelation in the sample? How does autocorrelation affect the confidence interval?

# The independence model

The standard confidence interval in the previous slide results from the assumptions of an **independence-based model**:

$$\begin{cases} Y_i = \mu + \epsilon_i \\ \epsilon_i \sim \mathcal{N}(0, \sigma^2) \end{cases} \quad (i.i.d.) , \quad (2)$$

where **i.i.d.** stands for independent and identically distributed: the random errors  $\epsilon_i$  are assumed to be independent.

Usual estimator of the population mean  $\mu$ : **sample mean**  $\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i$ .

With the independence model, the **sampling distribution of  $\bar{Y}$**  is:

$$\bar{Y} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right) .$$

Note:  $E[\bar{Y}] = \mu$  and  $V[\bar{Y}] = \frac{\sigma^2}{n}$ .

# Consequences of the independence model

The confidence interval flows directly from the result in the previous slide:

$$\frac{\bar{Y} - \mu}{\sqrt{\frac{\sigma^2}{n}}} \sim \mathcal{N}(0,1) \Rightarrow \left[ \bar{y} - z_{\alpha/2} \frac{\sigma}{\sqrt{n}} , \bar{y} + z_{\alpha/2} \frac{\sigma}{\sqrt{n}} \right].$$

Since the **population variance**  $\sigma^2$  is usually also **unknown**, it is **estimated** by the **sample variance**  $S^2 = \frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})^2$ . Corresponding results are:

$$\frac{\bar{Y} - \mu}{\sqrt{\frac{S^2}{n}}} \sim t_{n-1} \Rightarrow \left[ \bar{y} - t_{\alpha/2} \frac{s}{\sqrt{n}} , \bar{y} + t_{\alpha/2} \frac{s}{\sqrt{n}} \right].$$

# The AR(1) autocorrelation model

It is only possible to study the effects of autocorrelation in the sample if an alternative model is specified. We consider the simplest form of autocorrelation for the error terms: 1-D autocorrelation (in time).

This is the first order autoregressive, AR(1), model:

$$\begin{cases} Y_i = \mu + \eta_i \\ \eta_i = \lambda \eta_{i-1} + \epsilon_i \\ \epsilon_i \sim \mathcal{N}(0, \sigma^2) \end{cases}, \quad \eta_0 = 0 \quad (i.i.d.), \quad (3)$$

$\lambda$  gives the intensity and nature of the autocorrelation:

- $\lambda=0$ : the independence model (2).
- $\lambda>0$ : positive autocorrelation ( $\eta_{i-1} > 0$  makes  $\eta_i > 0$  more likely).
- $\lambda<0$ : negative autocorrelation ( $\eta_{i-1} > 0$  makes  $\eta_i < 0$  more likely).
- $|\lambda|>1$ : errors increase their effect over time (not usually realistic).

# The AR(1) autocorrelation model

We assume positive autocorrelation:  $0 < \lambda < 1$ .

Iterating the second equation of model (3) gives each observation of  $Y_i$  as a function only of the independent errors  $\epsilon_j$  ( $j \leq i$ ):

$$\begin{aligned} Y_i &= \mu + \lambda^{i-1} \epsilon_1 + \lambda^{i-2} \epsilon_2 + \lambda^{i-3} \epsilon_3 + \dots + \lambda^2 \epsilon_{i-2} + \lambda \epsilon_{i-1} + \epsilon_i \\ \Leftrightarrow Y_i &= \mu + \sum_{j=1}^i \lambda^{i-j} \epsilon_j . \end{aligned}$$

The AR(1) error model can be re-written as:

$$\left\{ \begin{array}{l} Y_i = \mu + \sum_{j=1}^i \lambda^{i-j} \epsilon_j \\ \epsilon_i \sim \mathcal{N}(0, \sigma^2) \quad (i.i.d.) . \end{array} \right. \quad (4)$$

## Comparing the models

AR(1) results are interesting for large  $i$  (after an initial transient period):

|                                   | Independence | AR(1)   |
|-----------------------------------|--------------|---|
| $E[Y_i]$                          | $\mu$        | $\mu$   |
| $V[Y_i]$                          | $\sigma^2$   | $\sigma^2 \left( \frac{1-\lambda^{2i}}{1-\lambda^2} \right) \rightarrow \frac{\sigma^2}{1-\lambda^2}$ |
| $Cov[Y_i, Y_j]$<br>(for $i > j$ ) | 0            | $\lambda^{i-j} V[Y_j] \rightarrow \lambda^{i-j} \frac{\sigma^2}{1-\lambda^2}$                         |
| $r_{ij}$<br>(for $i > j$ )        | 0            | $\lambda^{i-j} \sqrt{\frac{V[Y_j]}{V[Y_i]}} \rightarrow \lambda^{i-j}$                                |

AR(1) is stationary in the mean and (after transience) in the variance.

For  $0 < \lambda < 1$ , correlations decrease with the time lags  $i - j$ .

## Distribution of $\bar{Y}$ with the AR(1) model

How does the AR(1) model affect results for the sample mean  $\bar{Y}$ ?

Consider a random sample of size  $n$ , following model (3), but with  $t$  transient iterations:  $\vec{Y} = (Y_{t+1}, Y_{t+2}, Y_{t+3}, \dots, Y_{t+(n-1)}, Y_{t+n})^t$ .

The sample mean can be re-written as:

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_{t+i} = \mu + \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^{t+i} \lambda^{t+i-j} \epsilon_j .$$

The mean and variance of  $\bar{Y}$  are:

$$E[\bar{Y}] = \mu \quad , \quad V[\bar{Y}] = \frac{\sigma^2}{n^2(1-\lambda)^2} \left[ (1-\lambda^n)^2 \frac{\lambda^2}{1-\lambda^2} (1-\lambda^{2t}) + \sum_{i=1}^n (1-\lambda^i)^2 \right] .$$

If  $0 < \lambda < 1$ , for any  $n$  and  $t$ ,  $V[\bar{Y}] > \frac{\sigma^2}{n}$ . The true sampling variance of  $\bar{Y}$  is larger than with independence.

## Large sample AR(1) distribution of $\overline{Y}$

For large samples ( $n$  big), after large transient periods ( $t$  big), we have:

$$V[\overline{Y}] \approx \frac{\sigma^2}{n(1-\lambda)^2} \quad (5)$$

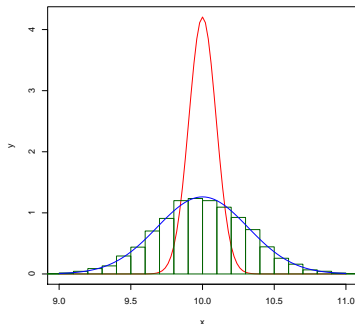
Normality of  $\overline{Y}$  also follows, and so, approximately:

$$\overline{Y} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{(1-\lambda)^2 n}\right) \quad (6)$$

We illustrate this result with a simulation of the sampling distribution of  $\overline{Y}$ : 10 000 samples of size  $n=1000$  were generated under model AR(1), with an initial transience of  $t=1000$  iterations. The population mean and standard deviation were chosen to be  $\mu=10$  and  $\sigma=3$ . The autocorrelation parameter was  $\lambda=0.7$ . The results are on the next slide.



# A simulation



**Figure:** Histogram of the distribution of  $\bar{y}$ , for 10 000 repetitions of size  $n=1000$  samples, in an AR(1) model. Parameters:  $\mu=10$ ,  $\sigma=3$ ,  $\lambda=0.7$ . Red curve:  $\mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$  distribution of  $\bar{Y}$  under independence. Blue curve: asymptotic equivalent under AR(1),  $\mathcal{N}\left(\mu, \frac{\sigma^2}{(1-\lambda)^2 n}\right)$ .

# The implications

The independence-based confidence interval formula will give shorter intervals, that do not ensure a true  $(1-\alpha) \times 100\%$  confidence level.

Of the 10 000 simulated samples, only 44.5% of the (nominally) 95% confidence intervals produced by formula (1) include  $\mu=10$ .

The appropriate (large-sample) confidence interval for AR(1) is:

$$\left[ \bar{y} - z_{\alpha/2} \frac{\sigma}{(1-\lambda)\sqrt{n}} , \bar{y} + z_{\alpha/2} \frac{\sigma}{(1-\lambda)\sqrt{n}} \right] . \quad (7)$$

94.95% of the 10 000 intervals given by the simulated samples contained the true population mean  $\mu=10$ .

There are similar implications for hypothesis testing on  $\mu$ .

## Effective sample size

A useful concept is that of **effective sample size**, defined as the value  $n_\epsilon$  such that:

$$V[\overline{Y}] = \frac{\sigma^2}{n_\epsilon} \quad \Leftrightarrow \quad n_\epsilon = \frac{\sigma^2}{V[\overline{Y}]} . \quad (8)$$

Effective sample size may be thought of as the **number of truly independent sources of information** in a sample of size  $n$ .

For AR(1) and the **large sample approximation**  $V[\overline{Y}] \approx \frac{\sigma^2}{(1-\lambda)^2 n}$ , we get:

$$n_\epsilon \approx n(1-\lambda)^2 .$$

| $n$   | $\lambda$ |         |         |         |         |         |        |        |       |
|-------|-----------|---------|---------|---------|---------|---------|--------|--------|-------|
|       | 0.1       | 0.2     | 0.3     | 0.4     | 0.5     | 0.6     | 0.7    | 0.8    | 0.9   |
| 10    | 8.27      | 6.66    | 5.20    | 3.87    | 2.68    | 1.63    | 0.78   | 0.23   | 0.03  |
| 50    | 40.66     | 32.26   | 24.78   | 18.25   | 12.67   | 8.03    | 4.35   | 1.67   | 0.21  |
| 100   | 81.16     | 64.25   | 49.28   | 36.25   | 25.17   | 16.03   | 8.85   | 3.64   | 0.60  |
| 1000  | 810.16    | 640.25  | 490.28  | 360.25  | 250.17  | 160.03  | 89.84  | 39.61  | 9.37  |
| 10000 | 8100.16   | 6400.25 | 4900.28 | 3600.25 | 2500.17 | 1600.03 | 899.84 | 399.61 | 99.33 |

## Estimating $\sigma^2$

The independence-based estimator of an unknown population variance  $\sigma^2$  becomes, in an AR(1) setting, a biased estimator, which overestimates  $\sigma^2$ .

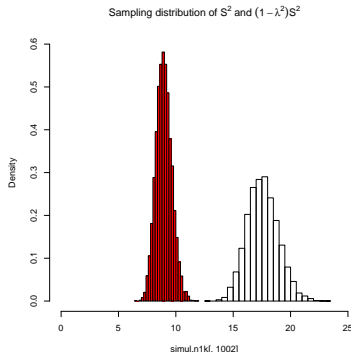
Assuming a large sample ( $n$  big) and a long transience ( $t$  big), the expected value of  $S^2 = \frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})^2$  is approximately:

$$E[S^2] \approx \frac{\sigma^2}{1 - \lambda^2} > \sigma^2. \quad (9)$$

An (approximately) unbiased estimator of  $\sigma^2$  is now  $\hat{\sigma}^2 = (1 - \lambda^2) S^2$ .

This is illustrated with the simulated samples in the next slide.

# A simulated sampling distribution of $S^2$ and $(1 - \lambda^2)S^2$



**Figure:** Histograms of the distributions of  $s^2$  (in black and white) and  $(1 - \lambda^2)s^2$  (in red), for 10 000 repetitions of size  $n=1000$  samples, in an AR(1) model. Parameters:  $\mu=10$ ,  $\sigma=3$ ,  $\lambda=0.7$ . The true variance is  $\sigma^2 = 9$ .

## Confidence intervals for $\mu$ when $\sigma^2$ is unknown

Using  $\frac{s^2}{n}$  instead of  $\frac{\sigma^2}{n}$  in the confidence interval for  $\mu$  gives a wider interval. But two wrongs do not make a right:

- The true (asymptotic) variance of  $\bar{Y}$  is  $\frac{\sigma^2}{(1-\lambda)^2 n}$ , not  $\frac{\sigma^2}{n}$ ;
- An unbiased estimator of  $\sigma^2$  is  $(1-\lambda^2) S^2$ , not  $S^2$ .

An unbiased estimator of the true (asymptotic) variance of  $\bar{Y}$  is:

$$\widehat{V[\bar{Y}]} = \frac{(1-\lambda^2) S^2}{(1-\lambda)^2 n} = \frac{1+\lambda}{1-\lambda} \frac{S^2}{n}. \quad (10)$$

The **standard** CIs for unknown  $\sigma^2$  (using the Normal distribution, since  $n$  is very large), are given by

$$\left[ \bar{y} - z_{\alpha/2} \frac{s}{\sqrt{n}}, \bar{y} + z_{\alpha/2} \frac{s}{\sqrt{n}} \right].$$

For the 10 000 simulated samples, only 58.77% of the nominally 95% confidence intervals contain the true population mean  $\mu=10$ .

## Confidence intervals for $\mu$ when $\sigma^2$ is unknown

Assuming Normality, the appropriate (large-sample) confidence interval for  $\mu$ , with AR(1), is:

$$\left[ \bar{y} - z_{\alpha/2} \sqrt{\frac{1+\lambda}{1-\lambda}} \frac{s}{\sqrt{n}}, \bar{y} + z_{\alpha/2} \sqrt{\frac{1+\lambda}{1-\lambda}} \frac{s}{\sqrt{n}} \right]. \quad (11)$$

This interval is wider than the standard CI by a factor of  $\sqrt{\frac{1+\lambda}{1-\lambda}}$ .

With these 95% (asymptotic) confidence intervals, the proportion of the 10 000 simulated AR(1) samples with confidence intervals containing  $\mu=10$  rises to 94.80%.

**Morale: autocorrelated data require specific methods.**

# Spatial data: the Aragonez data set

**Aragonez** is a Portuguese variety of grapes, also known by the name Tinta Roriz (and the Spanish name, Tempranillo).

A **field trial** was carried out in Reguengos de Monsaraz, in Southern Portugal.

**Goal:** measure the **yields** of this variety.

**Description:** A vineyard trellis was set up, with 40 wires (**columns**, numbered 4 to 43) running on an approximate N-S direction, and 2.25m apart. In each column, **groups of three plants** were taken to represent a cell, thereby **creating a rectangular grid with 40 columns and 26 rows**. The **rows are numbered 2 to 27** (bordering columns/rows were considered a 'transient' part, not included in the dataset). In each column, the centre of each grid cell (i.e, of the 'rows') are separated by 3.75m. The mean **yield in each grid cell (in kg of grapes/plant)** (with three plants in every cell) is the observation of interest.

**Warnings:**

- There are  $N=1019$  observations. Some of the 1040 cells have missing values.
- Yields make sense when defined relative to a given area (areal data). But we can use it as **geostatistical data: we assume that yields vary continuously over the trial field**.



# The Aragonez trial field

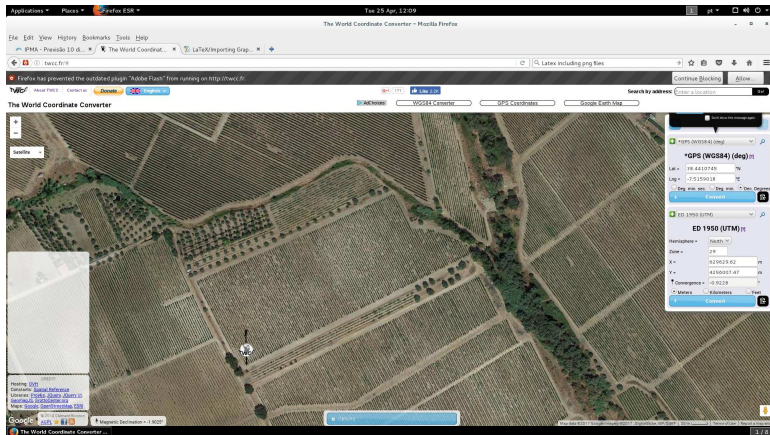


Figure: The region where the Aragonez dataset was collected. The southernmost corner has coordinates 38.4411239 N and 7.5159353 W.

# The Aragonez data

The dataset was provided as a **data frame**, whose six first lines and summary statistics are given below:

```
> head(Aragonez)
```

|   | genotype | block | col | row | colm | rowm  | yield |
|---|----------|-------|-----|-----|------|-------|-------|
| 1 | RZ717    | B1    | 4   | 2   | 0    | 0.00  | 2.417 |
| 2 | RZ1158   | B1    | 4   | 9   | 0    | 26.25 | 2.724 |
| 3 | RZ1325   | B1    | 4   | 6   | 0    | 15.00 | 2.647 |
| 4 | RZ3313   | B1    | 4   | 8   | 0    | 22.50 | 1.543 |
| 5 | RZ3603   | B1    | 4   | 12  | 0    | 37.50 | 0.865 |
| 6 | RZ3604   | B1    | 4   | 3   | 0    | 3.75  | 1.659 |

```
> summary(Aragonez)
```

|         | genotype | block  | col           | row           | colm          | rowm          | yield         |
|---------|----------|--------|---------------|---------------|---------------|---------------|---------------|
| RZ103   | : 4      | B1:255 | Min. : 4.00   | Min. : 2.00   | Min. : 0.00   | Min. : 0.00   | Min. : 0.188  |
| RZ107   | : 4      | B2:255 | 1st Qu.:13.00 | 1st Qu.: 8.00 | 1st Qu.:20.25 | 1st Qu.:22.50 | 1st Qu.:1.750 |
| RZ1103  | : 4      | B3:255 | Median :23.00 | Median :14.00 | Median :42.75 | Median :45.00 | Median :2.374 |
| RZ1110  | : 4      | B4:254 | Mean :23.36   | Mean :14.42   | Mean :43.57   | Mean :46.57   | Mean :2.549   |
| RZ1117  | : 4      |        | 3rd Qu.:33.00 | 3rd Qu.:21.00 | 3rd Qu.:65.25 | 3rd Qu.:71.25 | 3rd Qu.:3.247 |
| RZ1124  | : 4      |        | Max. :43.00   | Max. :27.00   | Max. :87.75   | Max. :93.75   | Max. :7.704   |
| (Other) | :995     |        |               |               |               |               |               |

Variables **genotype** and **block** are ignored.

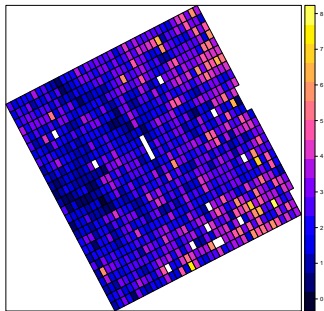
This data frame was geo-referenced in the initial class, and objects of both class **SpatialPointsDataFrame** and **SpatialPolygonsDataFrame** were created.

# The slots of a SpatialPointsDataFrame

Recall that a `SpatialPointsDataFrame` object has 5 **slots** where the information is stored. These five slots can be inspected using the general R function `str`:

```
> str(AragonezPoints)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data      : 'data.frame': 1019 obs. of  7 variables:
.. ..$ genotype: Factor w/ 255 levels "RZ103","RZ107",...: 193 11 36 81 95 96 106 112 158 160 ...
.. ..$ block   : Factor w/ 4 levels "B1","B2","B3",...: 1 1 1 1 1 1 1 1 1 1 ...
.. ..$ col     : int [1:1019] 4 4 4 4 4 4 4 4 4 4 ...
.. ..$ row     : int [1:1019] 2 9 6 8 12 3 13 10 4 5 ...
.. ..$ colm    : num [1:1019] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ rowm    : num [1:1019] 93.8 67.5 78.8 71.2 56.2 ...
.. ..$ yield   : num [1:1019] 2.417 2.724 2.647 1.543 0.865 ...
..@ coords.nrs : num(0)
..@ coords     : num [1:1019, 1:2] -43.7 -31.5 -36.7 -33.2 -26.2 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
..@ bbox       : num [1:2, 1:2] -4.37e+01 5.33e-15 7.58e+01 1.23e+02
.. ..- 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 slot
.. .. ..@ projargs: chr "+proj=aeqd +lat_0=38.4410745 +lon_0=-7.5159018 +datum=WGS84 +units=m +ellps=WGS84 +tow
```

## A first view of the Aragonez cell yields



**Figure:** The Aragonez yields for each cell in the rectangular grid of `AragonezPolygons` (scale on the right-hand side). Yields tend to increase from left to right on the trial field. There seems to be an underlying (linear?) trend.

## 2-D Spatial autocorrelation

We consider a generic two-dimensional spatial autocorrelation process  $Z$ :

- We assume that the (numerical) variable of interest  $Z$  depends on two spatial coordinates  $x$  and  $y$ ;
- $Z(x, y)$  represents the value of  $Z$  at location  $(x, y)$  on the  $x0y$  plane.
- In the spirit of the AR(1) model, we assume that  $Z(x, y)$  can be decomposed into three terms:

$$Z(x, y) = T(x, y) + \eta(x, y) + \epsilon(x, y) , \quad (12)$$

where:

- ▶  $T(x, y)$  is a **deterministic** (non-random) underlying spatial **trend** (sometimes further separated into  $\mu + T(x, y)$ , where  $\mu$  is the overall mean);
- ▶  $\eta(x, y)$  is a **spatially autocorrelated random process**, describing spatially correlated deviations from the underlying trend;
- ▶  $\epsilon(x, y)$  is an **uncorrelated random process**, i.e., **independent error terms**.

# Detrending the process

Detrending a spatial process separates explainable correlation between nearby values from unexplained spatial autocorrelation, in much the same way as standard regression methods remove explainable variability in a response variable, allowing us to focus on residual variability.

Several common ways of detrending a two-dimensional process:

- removing a constant (usually the mean)
- removing a linear trend on the geographical coordinates:

$$z = \beta_0 + \beta_1 x + \beta_2 y . \quad (13)$$

- removing a quadratic trend on the geographical coordinates:

$$z = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 y^2 + \beta_5 xy . \quad (14)$$

Other explanatory variables for trends will be considered next week.

# Detrending with least-squares fits

Standard least-squares (regression) fitting methods can be used to remove trends. In a strictly descriptive context, least-squares fitting does not need independent observations. Consider the `SpatialPointsDataFrame` object `AragonezPoints`. Create two new columns in the data frame, removing the overall mean yield (a constant) and a linear trend on the coordinates:

**Remark:** R accepts the shorthand `AragonezPoints$yield` for the complete `AragonezPoints@data$yield`.

```
> AragonezPoints$yieldct <- AragonezPoints$yield - mean(AragonezPoints$yield)
> Arag.lm <- lm(yield ~ rowm + colm, data=AragonezPoints)
> AragonezPoints$yieldldt <- AragonezPoints$yield - fitted(Arag.lm)
```

The resulting data frame:

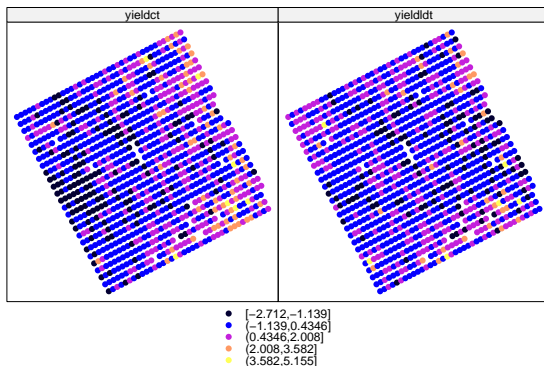
```
> head(AragonezPoints@data)
  genotype block col row colm rowm yield yieldct yieldldt
1  RZ717    B1  4  2    0 93.75 2.417 -0.13168302 0.87742187
2  RZ1158   B1  4  9    0 67.50 2.724 0.17531698 1.08176580
3  RZ1325   B1  4  6    0 78.75 2.647 0.09831698 1.04876126
4  RZ3313   B1  4  8    0 71.25 1.543 -1.00568302 -0.08456904
5  RZ3603   B1  4 12    0 56.25 0.865 -1.68368302 -0.82122965
6  RZ3604   B1  4  3    0 90.00 1.659 -0.88968302 0.10475672
```

## A second view of the Aragonez data

A function to plot spatial objects is the **spplot** function in package **sp**. It is a very powerful function, used here in the simplest possible way to view the detrended yields:

```
> spplot(AragonezPoints, layout=c(2,1), zcol=c("yieldct", "yieldldt"))
```

The **layout** option specifies the 'matrix' layout for multiple graphs (columns  $\times$  rows).

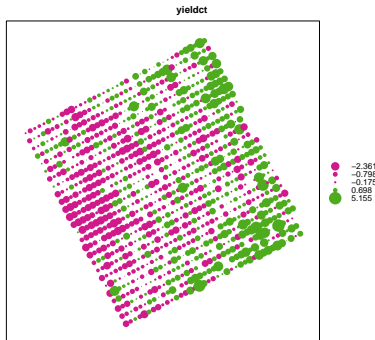




## Bubble plots

The `bubble` function (`sp` package) creates **bubble plots**, which are also useful to visualize spatial autocorrelation of (detrended) variables. Here is a bubble plot for the **centred yield** data, which still displays a **trend**:

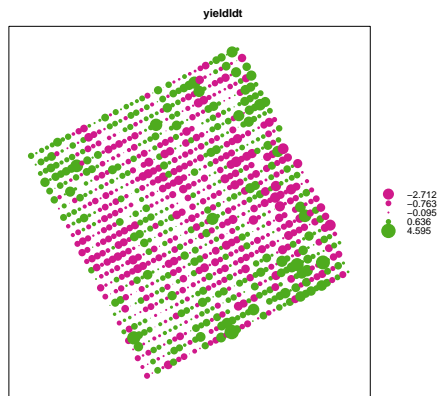
```
> bubble(AragonezPoints, zcol="yieldct")
```



## Another bubble plot

Here is a bubble plot for the **linearly detrended** yield data:

```
> bubble(AragonezPoints, zcol="yieldldt")
```



# Areal data and polygons

Data of areal type are spatially described by polygons, and the `sp` class `SpatialPolygonsDataFrame`.

The `AragonezPolygons`, of type `SpatialPolygonsDataFrame` has already been created. The plot on slide 28 was created with this object and the command:

```
> spplot(AragonezPolygons, zcol="yield")
```

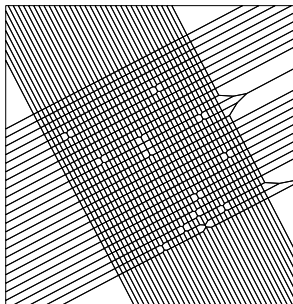
But it is necessary to again create the two new columns for the detrended yields, in its dataframe.

```
> AragonezPolygons$yieldct <- AragonezPolygons$yield-mean(AragonezPolygons$yield)
> AragonezPolygons$yieldldt <-
+   AragonezPolygons$yield - fitted(lm(yield ~ rowm + colm , data=AragonezPolygons))
```

## More on polygons

An object of class `SpatialPointsDataFrame` may also be converted to an object of class `SpatialPolygonsDataFrame`, using function `dismo::voronoi`, to create a tessellation of the space. Voronoi polygons partition a space into a set of polygons, given a set of coordinates, using a nearest centroid rule. No areas are left empty.

```
> library(dismo)
> AragVoronoi <- voronoi(AragonezPoints)
> plot(AragVoronoi)
```



# Autocorrelated spatial processes

To model two-dimensional autocorrelated processes, we assume that the values of the spatially autocorrelated process  $\eta(x, y)$  depend on the values of  $\eta$  at other points in some vicinity of  $(x, y)$ .

We assume that we have  $n$  observations of a spatial process  $Z(s)$ ,  $\{Z_i\}_{i=1}^n$ :

$$\begin{cases} Z_i = \mu + \eta_i \\ \eta_i = \lambda \left( \sum_{j=1}^n w_{ij} \eta_j \right) + \epsilon_i \\ \epsilon_i \sim \mathcal{N}(0, \sigma^2) \quad (i.i.d.) , \end{cases} \quad (15)$$

where  $w_{ij}$  is a constant measuring the influence of observation  $Z_j$  on observation  $Z_i$ . The  $n \times n$  matrix  $\mathbf{W}$ , whose  $(i, j)$ -th element is  $w_{ij}$ , is called a spatial weights matrix.

# Spatial weights matrices

Spatial weights matrices play a crucial role in the analysis of spatial data.

As a general rule,  $0 \leq w_{ij} \leq 1$ . But weights can be defined in several different ways.

For geostatistical data it is often appropriate to define weights  $w_{ij}$  as some non-increasing function,  $g$ , of the Euclidean distance  $d_{ij}$  between the coordinates of the points at which observations  $i$  and  $j$  were made:

$$w_{ij} = g(d_{ij}) , \quad (16)$$

Implicit in equation (16) is that weights depend only on the scalar distance  $d_{ij}$ , regardless of the direction which separates the points at which observations  $Z_i$  and  $Z_j$  were made. This is the isotropy assumption which may, or may not, be true in practice.

## Some distance-based weight functions

It may also be appropriate to define a range over which observation  $Z_i$  may be influenced by other observations, as well as the strength of that influence.

Some frequent choices for the distance functions  $g$  are:

- the **radial distance weight function**: observations made at a distance closer than some parameter  $d$  have a (common) weight 1, and observations made further apart have associated weight zero:

$$w_{ij} = \begin{cases} 1 & , \text{ if } 0 \leq d_{ij} \leq d \\ 0 & , \text{ if } d_{ij} > d \end{cases} \quad (17)$$

- the **power (inverse) distance weights function**: weights decrease with some power of the distance. For some positive constant  $a$ :

$$w_{ij} = \frac{1}{d_{ij}^a} ; \quad (18)$$

## More on distance functions

Another common (isotropic) distance function is:

- the **exponential distance weight function**: weights decrease exponentially with distance. For some positive constant  $a$ :

$$w_{ij} = e^{-a d_{ij}} . \quad (19)$$

For both the exponential and the power (inverse) weight functions, the larger the power  $a$ , the less influential will be distant points.

If appropriate, these weights can also be specified for **areal data**, defined on **polygons**. A **centroid, with coordinates**, is necessary for each polygon.

A more complex class of spatial weights assumes that, **for any given direction**, the weights decrease with distance, but in a way that differs, for different directions. This is the **anisotropy** assumption.



# Neighbourhoods

Other definitions of a spatial weights matrix **W** may be especially useful for areal data, where  $Z$  is observed on some regular grid, or some irregular arrangement of polygons. For each cell, neighbouring cells are specified and non-zero weights are then defined for each pair of neighbours.

Two different issues are at stake in neighbour-based weight matrices:

- the definition of pairs of neighbours, for which  $w_{ij} \neq 0$ ; and
- the precise way in which values are associated with the non-zero weights  $w_{ij}$ .

Neighbours can be defined when  $Z$  is observed only at points in space by creating a tessellation or grid of regions surrounding the points and using the resulting cells to define pairs of neighbours, or via a distance-based criterion for neighbourhoods.

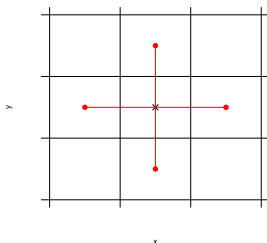
A standard convention in spatial statistics is that a cell is not a neighbour of itself.

# The rook's case

There are two famous conventions to define neighbourhoods: the **rook's case** and the **queen's case**.

**Rook's case:** pairs of cells are considered neighbours if they share a common border of length greater than zero.

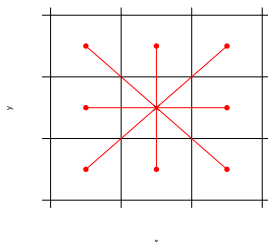
The name **rook's case** originates from the adjacent chessboard squares to which a rook can move, as illustrated below. The patchwork of cells does not have to be a rectangular grid for the definition to apply.



# The queen's case

**Queen's case:** pairs of cells are considered neighbours when they touch each other, even if only at a single point.

The name *queen's case* is again inspired by the possible movements of a queen on a chessboard, as illustrated below



Neighbours can also be specified in more general ways. The R package **spdep** provides ways to define neighbours.

# A touch of graph theory

Elementary concepts of **graph theory** are useful.

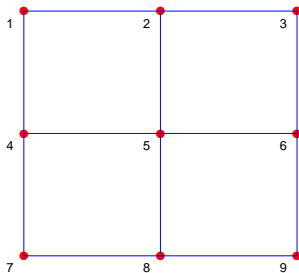
A **graph** is a pair of sets  $G = (V, E)$ , where:

- $V$  is a set of  $n$  **vertices** (or **points**, or **nodes**),  $\{v_i\}_{i=1}^n$ ;
- $E$  is a set of **edges** (or **lines**, or **arcs**) that may unite vertices:  
 $e_{ij} = (v_i, v_j)$ .

In our context, **vertices** will be **observation points/cells** and **edges** the **neighbour relations** that may exist between different points/cells.

## A very simple graph

Here is the graph for a  $3 \times 3$  grid of cells, using the rooks case criterion to define the edges:



There are  $n=9$  vertices and 12 edges.

## Some graph terminology

- The **order** of the graph is the **number of vertices**,  $|V|$ ;
- the **size** of the graph is the **number of edges**,  $|E|$ ;
- two vertices  $v_i$ ,  $v_j$  are **adjacent** if there is an edge  $e_{ij}$  between them;
- a given vertex  $v_i$  is **incident** with a given edge if that edge unites  $v_i$  with another vertex  $v_j$ ;
- the **degree** of a vertex is **the number of edges incident with that vertex**.

The central vertex in slide 45 is incident with 4 edges, thus, of degree 4. The four corner vertices (1, 3, 7, 9) are of degree 2 and all other vertices in that (very small) graph are of degree 3.

# Adjacency matrices

One way of fully specifying a graph is through its **adjacency matrix**  $A$ :

- both rows and columns are associated with the set of vertices;
- matrix element  $a_{ij}$  can take two values:
  - ▶  $a_{ij}=1$  if  $v_i$  and  $v_j$  are adjacent (edge  $e_{ij}$  exists);
  - ▶  $a_{ij}=0$  if  $v_i$  and  $v_j$  are not adjacent (edge  $e_{ij}$  does not exist).

Here is the adjacency matrix for the graph in slide 45:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (20)$$

## Adjacency matrices (cont.)

- Convention in spatial statistics: a vertex is not adjacent to itself; the diagonal elements in the adjacency matrix are all zero.
- The row sums of an adjacency matrix are the degree of each vertex.
- adjacency matrices are **symmetric** (except for directed graphs – see later):  $a_{ij} = a_{ji}$  for any  $i, j$ , or equivalently,  $A^t = A$ .
- In graphs of very high order ( $|V|$  very big) adjacency matrices are very large and require a lot of memory. It is more efficient to define a **list of  $n$  vectors** indicating, for each vertex, the adjacent vertices. In the above example, the list would have 9 objects, the first of which gives the vertices adjacent to vertex 1: (2, 4); the second the vector for vertex 2: (1, 3, 5); and so on.



# Weighted and directed graphs

- A **weighted graph** is a graph in which **edges have weights**, giving different strengths to the connections between vertices. In our context, **weighted graphs may be used to represent the distance, or some function  $g$  of the distance, between the observations/vertices.**
- Graphs may be **directed**, if an edge from an initial vertex  $v_i$  to a terminal vertex  $v_j$  is not the same thing as an edge from vertex  $v_j$  to vertex  $v_i$  (which may even not exist). Directed graphs are also called **digraphs**.
- The **adjacency matrix of a directed graph** is, in general, **not symmetric**.
- For directed graphs,
  - ▶ the **in-degree** of vertex  $v_i$  is the **number of edges that end at  $v_i$** ;
  - ▶ the **out-degree** of vertex  $v_i$  is the **number of edges that begin at  $v_i$** .

## Binary weight matrices

Once neighbours are defined, the specific weights  $w_{ij}$  must be specified. A few common options are:

**Binary weights matrix:** it is the **adjacency matrix** of the graph of neighbours:  $w_{ij} = 1$  if cells/points  $i, j$  are neighbours, and  $w_{ij} = 0$  otherwise (similar to a radial distance weight function, except for the fact that the neighbours may be defined in ways that are not simply functions of a distance).

For the graph on slide 45, we get:

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

For undirected neighbour graphs, a binary weights matrix is **symmetric** ( $w_{ij} = w_{ji}, \forall i, j \Leftrightarrow \mathbf{W}^t = \mathbf{W}$ ).

# Row-normalized weight matrices

**Row-normalized weights matrix:** all non-zero weights  $w_{ij}$  in a given row are equal and add to 1:  $\sum_{j=1}^n w_{ij} = 1$ , for any row  $i$ . The weights are given by  $w_{ij} = \frac{1}{d_i}$ , where  $d_i$  is the degree of vertex  $i$  in the graph of neighbours.

For the example on slide 45:

$$\mathbf{W} = \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

This is **not** a symmetric matrix ( $\mathbf{W} \neq \mathbf{W}^t$ ). But its use is fairly common.

## Standardized by mean number of links weight matrix

Another frequent types of weight matrices is the **globally standardized by the mean number of links weight matrix**. It is the neighbourhood graph adjacency (binary weights) matrix divided by  $\frac{|E|}{|V|}$  where  $|E|$  is the total number of neighbour links that were established between the  $n=|V|$  observations.

The non-zero weights (at the same positions as in the binary weights matrix) have value  $\frac{|V|}{|E|}$ , and add up to  $|V|$ , the number of observations.

This is a **symmetric** weight matrix.

## Standardized by the total number of links weight matrix

The globally standardized by the total number of links weight matrix is the neighbourhood graph adjacency (binary weights) matrix divided by the graph size  $|E|$ , that is, the total number of neighbour links that were specified.

All non-zero elements of  $\mathbf{W}$  are  $\frac{1}{|E|}$ , and their sum total is 1.

This is a symmetric weights matrix.

# Defining neighbours with R packages

Package `spdep` provides a class called `nb` for **neighbour lists**.

Details about this class can be found in the `nb vignette`, which can be invoked (after loading the `spdep` package) with the command:

```
> vignette("nb")
```

Objects of class `nb` store, in a compact way, the information about which pairs of objects are to be considered neighbours.

The following commands create `nb` objects.

## cell2nb

`spdep::cell2nb` assumes a rectangular grid with `nrow` rows and `ncol` columns, which must be specified as arguments to the command.

By default the command uses the rook's case to create neighbours, but the argument `type="queen"` uses the queen's case instead.

The appropriate command for the rook's case  $3 \times 3$  example is:

```
> cell2nb(3,3)
```

The results are on the next slide.

## cell2nb (example)

```
> cell2nb(3,3)
```

Neighbour list object:

Number of regions: 9

Number of nonzero links: 24

Percentage nonzero weights: 29.62963

Average number of links: 2.666667

There are  $|V|=9$  cells in the  $3 \times 3$  grid, for a maximum of  $9^2=81$  possible links between pairs of neighbours (counting links between each cell and itself, and different permutations, as in a directed graph).

Of these, only  $2 \times |E|=24$  are pairs of neighbours (for the rook's case), a percentage of  $\frac{24}{81} \times 100\% = 29.62963\%$ . The average number of links/cell is  $\frac{2|E|}{|V|} = \frac{24}{9} = 2.666667$ . In graph terminology this is the **mean degree per vertex**.

Since there are missing values in the Aragonez rectangular grid, the command `cell2nb(nrow=26, ncol=40)` does **not** solve our problem.



## `dnearneigh`

`spdep::dnearneigh` creates a list of neighbours with the distance between the (supplied) point coordinates that are between `d1` (usually zero) and `d2`.

For the `Aragonez` data set, with  $d1 = 0$  and  $d2 = 3$ , only points in the same row of adjacent columns (separated by  $2.25m$ ) will be neighbours:

```
> dnearneigh(AragonezPoints, d1=0, d2=3)
```

Neighbour list object:

Number of regions: 1019

Number of nonzero links: 1958

Percentage nonzero weights: 0.1885664

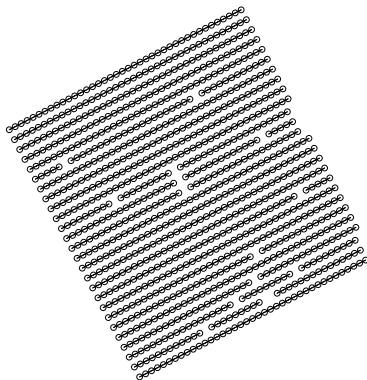
Average number of links: 1.921492

The number of non-zero links (1958) is almost twice the number of points (1019): most points have 2 neighbours (there are border points, but also missing values).

## `dnearneigh (d2=3)`

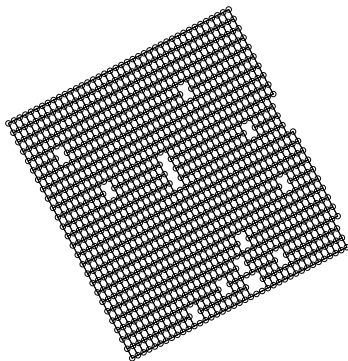
This is the resulting graph:

```
> plot(dnearneigh(AragonezPoints, d1=0, d2=3),  
+      coord=coordinates(AragonezPoints))
```



## `dnearneigh` ( $d2=4$ )

Different upper distance bounds give different sets of neighbours. For example,  $d2 = 4$  connects adjacent points in a way similar to the rook's case: for most points, the neighbours are the four cells immediately above, to the right, below, and to the left:



## `dnearneigh` ( $d2=5$ )

$d2=5$  is a rule similar to the queen's case, but with an extra-long horizontal connection (diagonally adjacent points are at distance  $4.37m$ ; points on the same row are neighbours of points two columns away, separated by  $4.5m$ ).

```
> dnearneigh(AragonezPoints, d1=0, d2=5)
```

Neighbour list object:

Number of regions: 1019

Number of nonzero links: 9550

Percentage nonzero weights: 0.9197187

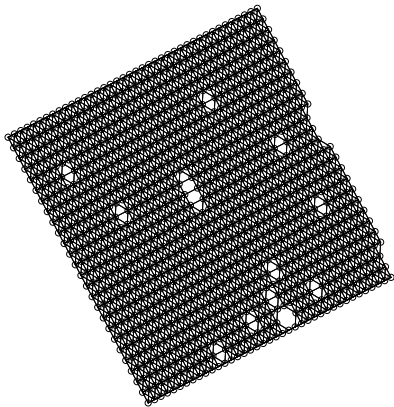
Average number of links: 9.371933

Most grid points have 10 neighbours: 1 above; 2 to the right; 1 below; 2 to the left; 4 in diagonal directions.

This choice allows for spatial dependence over gaps in the data.

`dnearneigh(d2=5)`

```
> plot(dnearneigh(AragonezPoints, d1=0, d2=5))
```



## knearneigh

`spdep::knearneigh` searches  $k$ -nearest neighbours in a set of point coordinates. Output is of class `knn` (class package), not `nb`. The `spdep.knn2nb` function converts `knn` objects to `nb` objects.

```
> knn2nb(knearneigh(AragonezPoints, k=4))
```

Neighbour list object:

Number of regions: 1019

Number of nonzero links: 4076

Percentage nonzero weights: 0.3925417

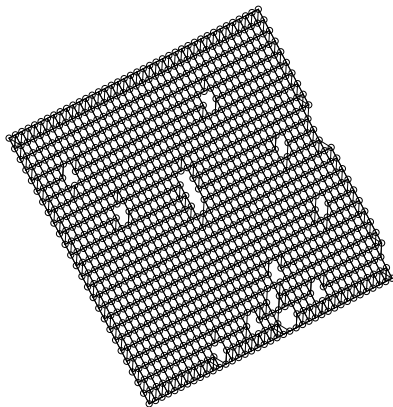
Average number of links: 4

Non-symmetric neighbours list

The average number of links is exactly 4 (by design), usually the adjacent cells in the `rook's case` sense.

## knearneigh

```
> plot(knn2nb(knearneigh(AragonezPoints, k=4)),  
+       coordinates(AragonezPoints))
```



## poly2nb

`spdep::poly2nb` accepts input of class `SpatialPolygonsDataFrame` and creates a neighbour list (of class `nb`) using the queen's case rule, by default.

```
> poly2nb(AragonezPolygons)
```

Neighbour list object:

Number of regions: 1019

Number of nonzero links: 7646

Percentage nonzero weights: 0.7363528

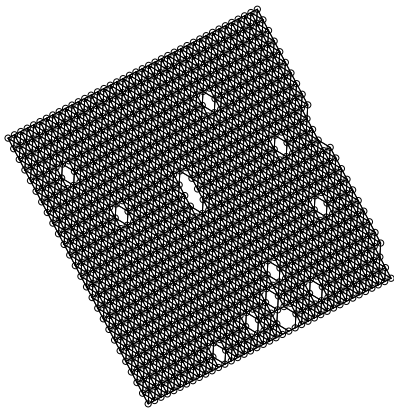
Average number of links: 7.503435

For `Aragonez`, on average, each grid point has almost 8 neighbours, as expected for the queen's case in a rectangular grid.



## poly2nb

```
> plot(poly2nb(AragonezPolygons),  
+       coords=coordinates(AragonezPolygons))
```



# Weights matrices

Once a neighbour list has been created, the `weights` matrix results from assigning weights to each pair of neighbours.

`spdep::nb2mat` accepts as input an `nb` object and creates a spatial weights matrix with, by default, the row-normalized weights criterion.

This is illustrated with the  $3 \times 3$  rectangular grid and a `rook's` case neighbour pattern (the default in the `cell2nb` function):

```
> nb2mat(cell2nb(3,3))
```

# Weights matrices

```
> nb2mat(cell2nb(3,3))
```

|     | [,1]      | [,2] | [,3]      | [,4] | [,5]      | [,6] | [,7]      | [,8] | [,9]      |
|-----|-----------|------|-----------|------|-----------|------|-----------|------|-----------|
| 1:1 | 0.0000000 | 0.50 | 0.0000000 | 0.50 | 0.0000000 | 0.00 | 0.0000000 | 0.00 | 0.0000000 |
| 2:1 | 0.3333333 | 0.00 | 0.3333333 | 0.00 | 0.3333333 | 0.00 | 0.0000000 | 0.00 | 0.0000000 |
| 3:1 | 0.0000000 | 0.50 | 0.0000000 | 0.00 | 0.0000000 | 0.50 | 0.0000000 | 0.00 | 0.0000000 |
| 1:2 | 0.3333333 | 0.00 | 0.0000000 | 0.00 | 0.3333333 | 0.00 | 0.3333333 | 0.00 | 0.0000000 |
| 2:2 | 0.0000000 | 0.25 | 0.0000000 | 0.25 | 0.0000000 | 0.25 | 0.0000000 | 0.25 | 0.0000000 |
| 3:2 | 0.0000000 | 0.00 | 0.3333333 | 0.00 | 0.3333333 | 0.00 | 0.0000000 | 0.00 | 0.3333333 |
| 1:3 | 0.0000000 | 0.00 | 0.0000000 | 0.50 | 0.0000000 | 0.00 | 0.0000000 | 0.50 | 0.0000000 |
| 2:3 | 0.0000000 | 0.00 | 0.0000000 | 0.00 | 0.3333333 | 0.00 | 0.3333333 | 0.00 | 0.3333333 |
| 3:3 | 0.0000000 | 0.00 | 0.0000000 | 0.00 | 0.0000000 | 0.50 | 0.0000000 | 0.50 | 0.0000000 |

But spatial weights matrices are usually very large, and tend to be sparse.

It is advisable to avoid creating the  $n \times n$  weights matrices.

# Objects of class `listw`

`spdep::nb2listw` is a function that creates objects of class `listw`, which efficiently store (sparse) spatial weights matrices.

The class `listw` is a list, with 3 components:

- `style` records the style of weights used, with row-sum normalized (`W`) as the default;
- `neighbours` is the `nb` object;
- `weights` is a list of numeric vectors giving the values of spatial weights for each pair  $(i, j)$  of neighbours.

# Objects of class listw

This is the **visible** output for the  $3 \times 3$  cell grid:

```
> nb2listw(cell2nb(3,3))
```

Characteristics of weights list object:

Neighbour list object:

Number of regions: 9

Number of nonzero links: 24

Percentage nonzero weights: 29.62963

Average number of links: 2.666667

Weights style: W

Weights constants summary:

|   | n | nn | S0 | S1       | S2       |
|---|---|----|----|----------|----------|
| W | 9 | 81 | 9  | 6.916667 | 36.80556 |

## Styles for weight matrices

The argument **style** exists in the output of both `nb2mat` and `nb2listw`. It controls the type of weights assigned to the neighbour pairs, with the following conventions:

- W** (default) a **row-normalized weights matrix** (the weights of each row add to 1).
- B** a **binary weights matrix** (all links have weight 1).
- C** the **globally standardized by the mean number of links** weight matrix (all weights add to  $n$ ).
- U** the **globally standardized by the total number of links** weight matrix (elements add to 1).

## A listw object

The five **weights constants summary** values in the output are:

$n$  - the number  $n$  of observations (sample size);

$nn$  - the number  $n^2$  of elements in the  $n \times n$  weight matrix;

$S0$  - the sum of all the weights in the weights matrix:  $S_0 = \sum_{i=1}^n \sum_{j=1}^n w_{ij}$ .

$S1$  - twice the sum of squares of all elements in the **symmetric part** of matrix  $\mathbf{W}$ , defined as  $\frac{\mathbf{W} + \mathbf{W}^t}{2}$ :

$$S_1 = 2 \sum_{i=1}^n \sum_{j=1}^n \left( \frac{w_{ij} + w_{ji}}{2} \right)^2 = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (w_{ij} + w_{ji})^2.$$

$S2$  - If  $w_{i.} = \sum_{j=1}^n w_{ij}$  is the **sum of  $\mathbf{W}$ 's  $i$ -th row**, and  $w_{.i} = \sum_{j=1}^n w_{ji}$  is the **sum of  $\mathbf{W}$ 's  $i$ -th column**:

$$S_2 = \sum_{i=1}^n (w_{i.} + w_{.i})^2.$$

# The structure of a listw object

This is the **actual** (partial) output of a `nb2listw` command:

```
> str(nb2listw(cell2nb(3,3)))
```

```
List of 3
 $ style      : chr "W"
 $ neighbours:List of 9
 ..$ : int [1:2] 2 4
 ..$ : int [1:3] 1 3 5
 ..$ : int [1:2] 2 6
 ..$ : int [1:3] 1 5 7
 ..$ : int [1:4] 2 4 6 8
 ..$ : int [1:3] 3 5 9
 ..$ : int [1:2] 4 8
 ..$ : int [1:3] 5 7 9
 ..$ : int [1:2] 6 8
 [...]
 $ weights    :List of 9
 ..$ : num [1:2] 0.5 0.5
 ..$ : num [1:3] 0.333 0.333 0.333
 ..$ : num [1:2] 0.5 0.5
 ..$ : num [1:3] 0.333 0.333 0.333
 ..$ : num [1:4] 0.25 0.25 0.25 0.25
 ..$ : num [1:3] 0.333 0.333 0.333
 ..$ : num [1:2] 0.5 0.5
 ..$ : num [1:3] 0.333 0.333 0.333
 ..$ : num [1:2] 0.5 0.5
 [...]
```



# A listw object

The `nb2listw` for the Aragonez polygons, with style "C":

```
> nb2listw(poly2nb(AragonezPolygons), style="C")
```

Characteristics of weights list object:

Neighbour list object:

Number of regions: 1019

Number of nonzero links: 7646

Percentage nonzero weights: 0.7363528

Average number of links: 7.503435

Weights style: C

Weights constants summary:

|   | n    | nn      | S0   | S1       | S2       |
|---|------|---------|------|----------|----------|
| C | 1019 | 1038361 | 1019 | 271.6089 | 4159.318 |

# Effects of two-dimensional autocorrelation

The effects of 2-D autocorrelation on standard statistical methods are similar to those for 1-D autocorrelation discussed previously.

Autocorrelation decreases the effective sample size, as there are no longer *n independent sources of information*. Standard independence-based statistical techniques provide mistaken significance levels and *p*-values, as well as mistaken confidence levels for confidence intervals.

This can be seen by again assuming the *extension of the AR(1) autocorrelation process*, which was already introduced in slide 37.

We re-write the model using the *random vector  $\vec{Z}$  of  $n$  observations of a temporal process*.

## The 2-D autocorrelated model in vector notation

Assuming independent Normal errors, with  $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$  for all  $i$ , is equivalent to assuming that the random error vector  $\vec{\epsilon}$  has a Multinormal distribution, with mean vector  $E[\vec{\epsilon}] = \vec{0}$  and variance-covariance matrix  $V[\vec{\epsilon}] = \sigma^2 \mathbf{I}_n$ , where  $\mathbf{I}_n$  is the  $n \times n$  identity matrix:

$$\vec{\epsilon} \sim \mathcal{N}_n(\vec{0}, \sigma^2 \mathbf{I}_n)$$

We have an alternative model formulation:

$$\begin{cases} \vec{Z} = \mu \vec{1}_n + \vec{\eta} \\ \vec{\eta} = \lambda \mathbf{W} \vec{\eta} + \vec{\epsilon} \\ \vec{\epsilon} \sim \mathcal{N}_n(\vec{0}, \sigma^2 \mathbf{I}_n) \end{cases}, \quad (21)$$

With a transient period of length  $t$ ,  $\mathbf{W}$  is  $(t+n) \times (t+n)$ . Only the post-transient part of  $\vec{Z}$  is of interest. Post-transience, matrix  $\mathbf{W}$  is  $n \times n$ .

## The 2-D autocorrelated error model in vector notation

The second equation in model (21) can be re-written (assuming the matrix inverse exists) as:

$$(\mathbf{I}_n - \lambda \mathbf{W}) \vec{\eta} = \vec{\epsilon} \quad \Leftrightarrow \quad \vec{\eta} = (\mathbf{I}_n - \lambda \mathbf{W})^{-1} \vec{\epsilon}. \quad (22)$$

So the spatial autocorrelation model under consideration becomes:

$$\begin{cases} \vec{\mathbf{Z}} = \mu \vec{\mathbf{1}}_n + (\mathbf{I}_n - \lambda \mathbf{W})^{-1} \vec{\epsilon} \\ \vec{\epsilon} \cap \mathcal{N}_n(\vec{\mathbf{0}}, \sigma^2 \mathbf{I}_n) \end{cases}. \quad (23)$$

## The distribution of $\vec{\mathbf{Z}}$

A linear (affine) transformation of a Multinormal vector, as in the first equation in model (23), preserves Multinormality.

The expected vector and (co-)variance matrix of  $\vec{\mathbf{Z}}$  can be derived by their general properties for linear transformations:

$$E[\vec{\mathbf{a}} + \mathbf{B}\vec{\mathbf{X}}] = \vec{\mathbf{a}} + \mathbf{B} E[\vec{\mathbf{X}}] \quad (24)$$

$$V[\vec{\mathbf{a}} + \mathbf{B}\vec{\mathbf{X}}] = \mathbf{B} V[\vec{\mathbf{X}}] \mathbf{B}^t \quad (25)$$

These properties give:

$$E[\vec{\mathbf{Z}}] = \mu \vec{\mathbf{1}}_n, \quad V[\vec{\mathbf{Z}}] = \sigma^2 [(\mathbf{I}_n - \lambda \mathbf{W})^t (\mathbf{I}_n - \lambda \mathbf{W})]^{-1}.$$

Therefore, under model (23), we have:

$$\vec{\mathbf{Z}} \cap \mathcal{N}_n \left( \mu \vec{\mathbf{1}}_n, \sigma^2 [\mathbf{I}_n - \lambda (\mathbf{W} + \mathbf{W}^t) + \lambda^2 \mathbf{W}^t \mathbf{W}]^{-1} \right) \quad (26)$$

# The distribution of $\bar{\mathbf{Z}}$

Note the role of both the spatial weights matrix  $\mathbf{W}$  and the overall autocorrelation parameter  $\lambda$  in the distribution of  $\bar{\mathbf{Z}}$ .

Using this vector/matrix notation, the sample mean is:  $\bar{\mathbf{Z}} = \frac{1}{n} \mathbf{1}_n^t \bar{\mathbf{Z}}$  (the inner product  $\mathbf{1}_n^t \bar{\mathbf{Z}}$  gives the sum of elements of  $\bar{\mathbf{Z}}$ ).

The general properties (24) for expected values and variances give:

$$E[\bar{\mathbf{Z}}] = \frac{1}{n} \mathbf{1}_n^t \cdot \mu \mathbf{1}_n = \mu \frac{1}{n} \mathbf{1}_n^t \mathbf{1}_n = \mu \quad (27)$$

$$V[\bar{\mathbf{Z}}] = \frac{1}{n^2} \mathbf{1}_n^t V[\bar{\mathbf{Z}}] \mathbf{1}_n. \quad (28)$$

**Remark:** For any matrix  $\mathbf{B}$ ,  $\mathbf{1}_n^t \mathbf{B} \mathbf{1}_n$  gives the sum of elements in  $\mathbf{B}$ .

# The distribution of $\bar{\mathbf{Z}}$

Since Normality is also preserved, we have:

$$\bar{\mathbf{Z}} \cap \mathcal{N} \left( \mu, \frac{\text{sum}(V[\bar{\mathbf{Z}}])}{n^2} \right)$$

with  $V[\bar{\mathbf{Z}}] = \sigma^2 [\mathbf{I}_n - \lambda(\mathbf{W} + \mathbf{W}^t) + \lambda^2 \mathbf{W}^t \mathbf{W}]^{-1}$ .

For  $\lambda = 0$  (no spatial autocorrelation)  $V[\bar{\mathbf{Z}}]$  reverts back to  $\frac{\sigma^2}{n}$ , the result for independent samples.

Specific formulas for  $V[\bar{\mathbf{Z}}]$  depend on both  $\lambda$  and the weights matrix  $\mathbf{W}$ , but are in general different from the variance for independent samples.

# Measuring spatial autocorrelation

The most frequent measure of spatial autocorrelation is **Moran's I** indicator, originally developed to test the null hypothesis of zero autocorrelation for a (fully numerical) spatial random process  $Z$ , given a random sample  $(Z_1, Z_2, \dots, Z_n)$ .

We start with the following expression, which resembles a weighted covariance, not between different variables measured at corresponding points, but between the same variables (the sample values  $Z_i$ ), measured at all possible pairs of points:

$$\frac{\sum_{i=1}^n \sum_{j=1}^n w_{ij} (Z_i - \bar{Z})(Z_j - \bar{Z})}{\sum_{i=1}^n \sum_{j=1}^n w_{ij}} \quad (29)$$

The weights  $w_{ij}$  are the elements of a spatial weights matrix **W**.

The sum of the weights in the denominator is  $S_0$ .



# Moran's $I$

Moran's  $I$  indicator compares this 'Moran covariance' with the value that would result if the spatial weights matrix were an identity matrix ( $\mathbf{W} = \mathbf{I}$ ), which is the assumption of independence.

Moran's  $I$  is measuring how well the spatial weights  $w_{ij}$  applied to neighbouring values  $Z_j$  are capable of reconstituting the observed values  $Z_i$ :

$$I = \frac{\frac{\sum_{i=1}^n \sum_{j=1}^n w_{ij} (Z_i - \bar{Z})(Z_j - \bar{Z})}{\sum_{i=1}^n \sum_{j=1}^n w_{ij}}}{\frac{\sum_{i=1}^n (Z_i - \bar{Z})^2}{n}} = \frac{n}{S_0} \cdot \frac{\sum_{i=1}^n \sum_{j=1}^n w_{ij} (Z_i - \bar{Z})(Z_j - \bar{Z})}{\sum_{i=1}^n (Z_i - \bar{Z})^2}. \quad (30)$$

More positive (negative) values of  $I$  tend to be associated with more intense positive (negative) autocorrelation.

The expected value of  $I$  in the absence of spatial autocorrelation is  $E[I] = -\frac{1}{n-1}$ .

## Geary's $c$

Geary's  $c$  is a related indicator, which instead of using 'Moran's covariance', uses a weighted sum of the squared distances between the observed variable values, at all possible pairs of observed points:

$$c = \frac{n-1}{2S_0} \cdot \frac{\sum_{i=1}^n \sum_{j=1}^n w_{ij} (Z_i - Z_j)^2}{\sum_{i=1}^n (Z_i - \bar{Z})^2} . \quad (31)$$

The expected value of Geary's  $c$ , with no spatial autocorrelation, is 1.

Smaller values (necessarily non-negative) of  $c$  indicate positive autocorrelation, and values  $c > 1$  indicate negative autocorrelation.

# Testing for spatial autocorrelation

Both Moran's  $I$  and Geary's  $c$  are used to test the null hypothesis of no spatial autocorrelation, in one of two ways:

- both indicators have asymptotic Normal distribution, given the null hypothesis.
- permutation tests can also be used: the value of the indicators (Moran or Geary) is computed for a large number of permutations of the variable values along the spatial distribution, and the empirical quantile of our true indicator value is registered. In the absence of spatial autocorrelation, the empirical quantiles of  $I$  or  $c$  should not be extreme. If they are, this suggests the existence of spatial autocorrelation.

# Testing for spatial autocorrelation in R

The functions `spdep::moran` and `spdep::geary`, compute the value of each indicator.

The functions `spdep::moran.test` and `spdep::geary.test` test for spatial autocorrelation.  $H_0$  is the absence of spatial autocorrelation.

By default, both the `moran.test` and the `geary.test` functions will carry out a permutations-based test.

If the `randomisation` argument is set to the logical value `FALSE`, Normality-based tests are carried out.

Since the values of Moran's  $I$  and of Geary's  $c$  are also displayed when using the test functions, we focus on these `*.test` functions, with the Aragonez dataset, for various weights matrices.

# Spatial autocorrelation for the Aragonez centred yields

We first consider the centred yields, with a row-normalized weight matrix and neighbours defined by the maximum distance of  $3m$ :

```
> Wd3 <- nb2listw(dnearneigh(AragonezPoints, d1=0, d2=3))  
> moran.test(AragonezPoints$yieldct, listw=Wd3)
```

Moran I test under randomisation

data: AragonezPoints\$yieldct  
weights: Wd3

Moran I statistic standard deviate = 10.924, p-value < 2.2e-16  
alternative hypothesis: greater  
sample estimates:

| Moran I statistic | Expectation   | Variance     |
|-------------------|---------------|--------------|
| 0.3506489981      | -0.0009823183 | 0.0010361789 |

$I = 0.3506489981$  is considered highly significant with a permutation test. But an undetected spatial trend may be confused with spatial autocorrelation.

# Testing the Aragonez linearly detrended yields

With other aspects equal, we now test the linearly detrended yields:

```
> moran.test(AragonezPoints$yieldldt, listw=Wd3)
```

Moran I test under randomisation

data: AragonezPoints\$yieldldt

weights: Wd3

Moran I statistic standard deviate = 6.1155, p-value = 4.812e-10

alternative hypothesis: greater

sample estimates:

| Moran I statistic | Expectation   | Variance     |
|-------------------|---------------|--------------|
| 0.1958816555      | -0.0009823183 | 0.0010362521 |

Moran's  $I$  is now noticeably smaller than before, but there is still strong indication of spatial autocorrelation.

# Normality-based tests

Normality-based tests do not produce major differences, as illustrated below. Note that the value of  $I$  does not change with the type of test used.

```
> moran.test(AragonezPoints$yieldldt, listw=Wd3, randomisation=F)
```

Moran I test under normality

data: AragonezPoints\$yieldldt

weights: Wd3

Moran I statistic standard deviate = 6.1128, p-value = 4.895e-10

alternative hypothesis: greater

sample estimates:

| Moran I statistic | Expectation   | Variance     |
|-------------------|---------------|--------------|
| 0.1958816555      | -0.0009823183 | 0.0010371728 |

# Geary's $c$ with Aragonez

The use of Geary's  $c$  gives similar results (keeping in mind that the absence of spatial autocorrelation is indicated by the value  $c = 1$ ):

```
> geary.test(AragonezPoints$yieldct, listw=Wd3)
```

```
Geary C test under randomisation
```

```
data: AragonezPoints$yieldct  
weights: Wd3
```

```
Geary C statistic standard deviate = 10.561, p-value < 2.2e-16  
alternative hypothesis: Expectation greater than statistic  
sample estimates:
```

| Geary C statistic | Expectation | Variance    |
|-------------------|-------------|-------------|
| 0.655693718       | 1.000000000 | 0.001062794 |

The tests depend on the spatial weights matrix used.

For the Aragonez detrended yields, spatial autocorrelation seems to exist.



## K-th order neighbours

Given a list of neighbours of each observation, **second-order neighbours** are the neighbours of neighbours.

**Third-order neighbours** include the neighbours of second-order neighbours. Neighbours of order  $k$ , for any natural number  $k$ , are defined in a similar fashion.

The `spdep` function `nblag`, given a neighbour's list, computes neighbours of successive order (up to a value  $k$  given by the `maxlag` argument).

The **output** is a list of length `maxlag`, with the summary characteristics of the neighbour's list for each lag:

- the first object in the output list summarizes the initial neighbours list;
- the second list object summarizes the neighbours of order 2;
- and so on.

## K-th order neighbours and nb1ag

```
> nb.k4 <- knn2nb(knearneigh(AragonezPoints, k=4))  
> nb1ag(nb.k4, maxlag=3)
```

```
[[1]]
```

```
Neighbour list object:  
Number of regions: 1019  
Number of nonzero links: 4076  
Percentage nonzero weights: 0.3925417  
Average number of links: 4  
Non-symmetric neighbours list
```

```
[[2]]
```

```
Neighbour list object:  
Number of regions: 1019  
Number of nonzero links: 7762  
Percentage nonzero weights: 0.7475242  
Average number of links: 7.617272  
Non-symmetric neighbours list
```

```
[[3]]
```

```
Neighbour list object:  
Number of regions: 1019  
Number of nonzero links: 11253  
Percentage nonzero weights: 1.083727  
Average number of links: 11.04318  
Non-symmetric neighbours list
```

# Moran's correlogram

K-th order neighbours are useful to see how indicators such as Moran's  $I$  vary as successive orders of neighbours are considered.

Function `sp.correlogram` (package `spdep`) computes Moran's  $I$  for the neighbours of each successive order.

Arguments which must be specified are:

- the original `neighbours` list (an object of class `nb`);
- the `method`, giving the type of indicator (`"I"` or `"C"`);
- the `variable` of interest (argument `var`);
- the maximum `order` for neighbours (argument `order`);
- The `style` of the weight matrix (by default it is `style="W"`, a row-normalized weights matrix).

# Moran's correlogram for the Aragonez data

```
> sp.correlogram(nb.k4, var=AragonezPoints$yieldldt, method="I", order=3)
```

Spatial correlogram for AragonezPoints\$yieldldt

method: Moran's I

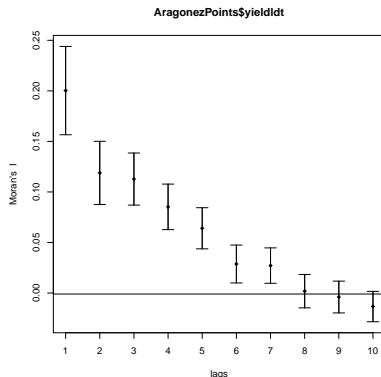
|          | estimate   | expectation | variance   | standard deviate | Pr(I)     | two sided |
|----------|------------|-------------|------------|------------------|-----------|-----------|
| 1 (1019) | 0.20026079 | -0.00098232 | 0.00047723 | 9.2121           | < 2.2e-16 | ***       |
| 2 (1019) | 0.11882755 | -0.00098232 | 0.00024437 | 7.6643           | 1.798e-14 | ***       |
| 3 (1019) | 0.11274136 | -0.00098232 | 0.00016604 | 8.8255           | < 2.2e-16 | ***       |

Row  $k$  is the output of a `moran.test` function with neighbours of order  $k$ . As would be expected, Moran's  $I$  decreases as the order  $k$  of neighbours grows: spatial correlation tends to decrease with increasing spatial lags.

Plotting the values of Moran's  $I$  against the order  $k$  of the neighbours gives a **Moran's correlogram**. The following command produces a Moran's correlogram up to order 10, which is shown in the next slide.

```
> plot(sp.correlogram(nb.k4, var=AragonezPoints$yieldldt, method="I", order=10))
```

# Moran's correlogram for Aragonez



**Figure:** Moran's correlogram for the linearly detrended Aragonez yields, based on a  $k = 4$  nearest neighbours list and a row-normalized weight matrix, with lags of up to 10. There is evidence of spatial autocorrelation, at least up to neighbours of lag  $k = 5$  or  $k = 7$ .

# Some concepts of random processes of geostatistical data

We assume that  $Z(s)$  is a **random spatial process** where  $s \in \mathcal{S}$  ( $s$  is a **vector** of coordinates).

- **Mean function  $\mu_s$**  is the function that, for each location  $s \in \mathcal{S}$  gives the expected value  $\mu_s = E[Z(s)]$ .
- **Covariogram  $C(s_1, s_2)$** , or **auto-covariance function**, is the function that, for any pair of locations  $s_1, s_2 \in \mathcal{S}$ , gives the covariance between  $Z(s_1)$  and  $Z(s_2)$ :

$$C(s_1, s_2) = \text{Cov}[Z(s_1), Z(s_2)] = E[(Z(s_1) - \mu_{s_1})(Z(s_2) - \mu_{s_2})] . \quad (32)$$

- **Spatial lag  $s_1 - s_2$**  is the difference between two locations  $s_1$  and  $s_2$  where  $Z(s)$  is observed. It is a 2-D vector.

## Second-order (weak) stationarity

We say that the process  $Z(s)$  is:

- **second-order (or weakly) stationary** if  $\mu_s$  does not depend on the location  $s$  (is constant over  $\mathcal{S}$ ) and  $C(s_1, s_2)$  depends only on the spatial lag:

$$\mu_s = \mu, \quad \forall s \in \mathcal{S} \quad ; \quad \text{and} \quad (33)$$

$$C(s_1, s_2) = C_\ell(s_1 - s_2), \quad \forall s_1, s_2 \in \mathcal{S}. \quad (34)$$

- **isotropic** when the covariogram  $C(s_1, s_2)$  depends only on the (scalar) distance between the points  $s_1$  and  $s_2$ :  $C(s_1, s_2) = C_s(d(s_1, s_2))$ .
- **Anisotropic** if it is a second-order stationary process but **not** isotropic, that is,  $C(s_1, s_2)$  depends on the spatial lag, but in ways that vary according to the direction of the spatial lag vector  $s_1 - s_2$ .

## (Semi-)Variogram

A crucial concept is the (semi-)variogram function:

- variogram is the function

$$2\gamma(s_1, s_2) = \text{Var}[Z(s_1) - Z(s_2)] . \quad (35)$$

- semi-variogram is the function

$$\gamma(s_1, s_2) = \frac{1}{2} \text{Var}[Z(s_1) - Z(s_2)] . \quad (36)$$

Confusingly, the semi-variogram is often just called a variogram.

It is straightforward to see that, for weakly stationary processes:

$$2\gamma(s_1, s_2) = C(s_1, s_1) + C(s_2, s_2) - 2 C(s_1, s_2) = 2C_\ell(\vec{0}) - 2C_\ell(\vec{h}) .$$



# Semi-variogram

So, for weakly stationary processes, the **semi-variogram** is:

$$\gamma_{\ell}(\vec{h}) = C_{\ell}(\vec{0}) - C_{\ell}(\vec{h}) . \quad (37)$$

With the further assumption of **isotropy**, the semi-variogram becomes a function of a single real variable, the distance  $h = d(s_1 - s_2)$  associated with the spatial lag:

$$\gamma_s(h) = C_s(0) - C_s(h) . \quad (38)$$

# Properties of the semi-variogram

With **isotropy**, the **semi-variogram** of a spatial process  $Z(s)$  is:

- **nonnegative**:  $\gamma_s(h) \geq 0, \forall h$ .
- an **even** function:  $\gamma_s(h) = \gamma_s(-h)$ .
- $\gamma_s(0) = 0$ .
- With no spatial autocorrelation,  $C(h) = 0$ , for  $h \neq 0$ , and so  $\gamma_s(h) = C_s(0) = \text{Var}[Z(s)], \forall h \neq 0$ .

Without spatial autocorrelation,  $\gamma_s$  is discontinuous at the origin.

- Even with spatial autocorrelation, the variogram is not usually continuous at the origin. This may be thought of as a feature of the semi-variogram itself, or as a consequence of the necessary discretization that any measurement of the covariances implies.

# Nugget, sill, range and partial sill

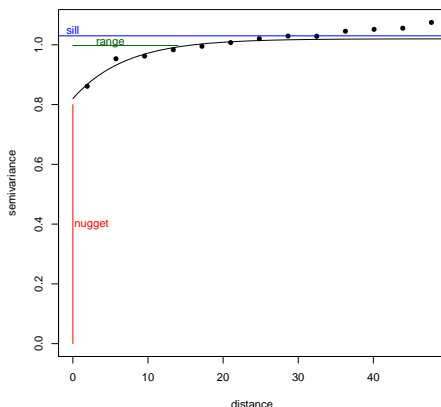
For **isotropic processes**, we define:

- $\lim_{h \rightarrow 0} \gamma_s(h) = b$  is called the **nugget effect**. The nugget effect can be viewed as the part of the variance of the random process  $Z(s)$  that has **not** been explained by the spatial autocorrelation process.
- The **sill** is the (constant) variance of the stationary process  $Z(s)$ :

$$\text{sill} = \lim_{h \rightarrow +\infty} \gamma_s(h) = C_s(0) - \lim_{h \rightarrow +\infty} C_s(h) = C_s(0) = \text{Var}[Z(s)] .$$

- the **range** of a spatial point  $s_1$  is the value of  $h$  for which the semi-variogram is smaller than the sill,  $\gamma(h) < C(0)$ , or, if the sill is an asymptotic value, for which the semi-variogram becomes some proportion, very close to 1 (say 95%) of the sill.
- The **partial sill** is the difference between the sill and the nugget. It can be viewed as that part of  $\text{Var}[Z(s)]$  that **is** explained by the spatial autocorrelation process.

# A typical semi-variogram



**Figure:** A typical variogram curve, with the nugget, sill and range (95% of sill) highlighted.

## Empirical semi-variograms

Assuming isotropy, the semi-variogram is estimated by the **empirical semi-variogram**, from the sample  $(z(s_1), z(s_2), \dots, z(s_n))$ :

$$\hat{\gamma}(h) = \frac{1}{2} \frac{1}{|N(h)|} \sum_{i,j \in N(h)} (z(s_i) - z(s_j))^2, \quad (39)$$

where, for any given distance  $h = \text{dist}(s_1, s_2)$ ,  $N(h)$  denotes the set of pairs of locations  $s_1, s_2$  which are distance  $h$ ,  $|N(h)|$  is the cardinality (size) of this set, and the summation is over all pairs of locations  $s_i, s_j$  at that given distance.

Usually,  $h$  is taken to be a small interval in order to ensure the existence of enough pairs  $N(h)$  of observations, for any given  $h$ .

To interpret the empirical semi-variogram, we must consider the properties of the semi-variogram which it is estimating.

# Empirical semi-variograms in R

The `gstat` package has a `variogram` function that computes the empirical semi-variogram.

Input is:

- a formula to detrend the variable (similar to the R formulas for linear regression);
- a `SpatialPointsDataFrame` object.

Alternatively, the latter argument may be replaced by the name of a data frame containing the variable and a list of coordinates for each observed point.

# The variogram function

We compute the empirical semi-variogram of the Aragonez variable `yield`, detrended by just subtracting a constant (the mean):

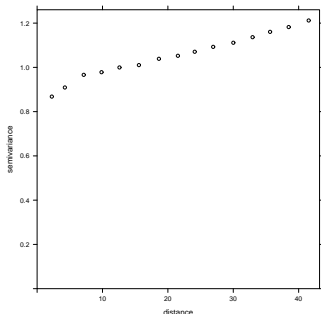
```
> variogram(yield ~ 1, data=AragonezPoints)
```

|    | np    | dist      | gamma     | dir.hor | dir.ver | id   |
|----|-------|-----------|-----------|---------|---------|------|
| 1  | 979   | 2.250000  | 0.8679609 | 0       | 0       | var1 |
| 2  | 3796  | 4.246580  | 0.9091233 | 0       | 0       | var1 |
| 3  | 7270  | 7.127722  | 0.9662692 | 0       | 0       | var1 |
| 4  | 7864  | 9.849207  | 0.9782520 | 0       | 0       | var1 |
| 5  | 12517 | 12.594300 | 0.9996118 | 0       | 0       | var1 |
| 6  | 12825 | 15.568268 | 1.0102484 | 0       | 0       | var1 |
| 7  | 16747 | 18.627263 | 1.0387344 | 0       | 0       | var1 |
| 8  | 15290 | 21.530216 | 1.0523777 | 0       | 0       | var1 |
| 9  | 17219 | 24.104900 | 1.0706070 | 0       | 0       | var1 |
| 10 | 18576 | 26.924936 | 1.0927295 | 0       | 0       | var1 |
| 11 | 22484 | 30.008119 | 1.1112754 | 0       | 0       | var1 |
| 12 | 19601 | 32.950881 | 1.1364426 | 0       | 0       | var1 |
| 13 | 20476 | 35.629361 | 1.1605164 | 0       | 0       | var1 |
| 14 | 22326 | 38.471422 | 1.1822151 | 0       | 0       | var1 |
| 15 | 23360 | 41.534207 | 1.2118956 | 0       | 0       | var1 |

Column `dist` gives the values of  $h$ ; column `gamma` gives the corresponding estimated value of the semi-variogram value  $\gamma(h)$ , computed from the `np` available points.

## An empirical semi-variogram

An empirical semi-variogram can be plotted using the appropriate `plot` method for objects of class `gstatVariogram`, in other words, by enclosing the previous command inside a `plot()` call.



**Figure:** The empirical semi-variogram for the Aragonez yields, detrended by the (constant) mean, as produced by the `variogram` command in package `gstat`. The stabilization of the semi-variogram may have not been completed



## Extending the cutoff point

To check whether the curve is approaching a horizontal asymptote at about 1.2, the `cutoff` argument in the function `variogram` will be set to a larger value:

```
> variogram(yield ~ 1, data=AragonezPoints, cutoff=75)
```

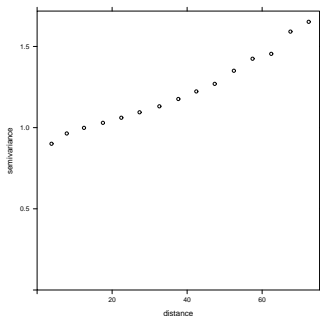
|    | np    | dist      | gamma     | dir.hor | dir.ver | id   |
|----|-------|-----------|-----------|---------|---------|------|
| 1  | 4775  | 3.837229  | 0.9006839 | 0       | 0       | var1 |
| 2  | 11662 | 7.905869  | 0.9640507 | 0       | 0       | var1 |
| 3  | 18434 | 12.501070 | 0.9986043 | 0       | 0       | var1 |
| 4  | 27127 | 17.542904 | 1.0296377 | 0       | 0       | var1 |
| 5  | 27127 | 22.449346 | 1.0609890 | 0       | 0       | var1 |
| 6  | 34050 | 27.313614 | 1.0945563 | 0       | 0       | var1 |
| 7  | 39916 | 32.596498 | 1.1313532 | 0       | 0       | var1 |
| 8  | 34879 | 37.653478 | 1.1764153 | 0       | 0       | var1 |
| 9  | 37274 | 42.447983 | 1.2230035 | 0       | 0       | var1 |
| 10 | 39118 | 47.351256 | 1.2696245 | 0       | 0       | var1 |
| 11 | 38784 | 52.461356 | 1.3503768 | 0       | 0       | var1 |
| 12 | 34371 | 57.443675 | 1.4244315 | 0       | 0       | var1 |
| 13 | 36161 | 62.426631 | 1.4545192 | 0       | 0       | var1 |
| 14 | 30510 | 67.543776 | 1.5915366 | 0       | 0       | var1 |
| 15 | 26498 | 72.413497 | 1.6518498 | 0       | 0       | var1 |

Increasing the `cutoff` argument has its limits: as the distance  $h$  grows, the values of  $\gamma$  will be estimated with a substantially smaller numbers of points, becoming prone to erratic behaviour.

## Extending the cutoff

The figure gives the new empirical semi-variogram. It is still unlikely that an asymptotic stabilization has been achieved. This suggests a non-stationary variance in the process or an inappropriately removed underlying trend.

```
> variogram(yield ~ 1, data=AragonezPoints, cutoff=75)
```



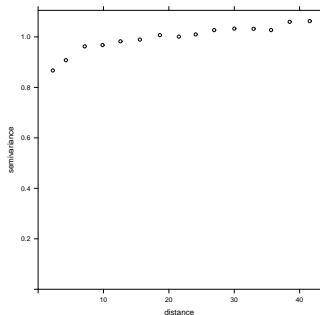
## Variogram for the linearly detrended yields

Detrending with a linear regression on column and row distances (variables `colm` and `rowm`, respectively, in the `AragonezPoints` object) can be given in the formula argument of the command, as illustrated below:

```
> variogram(yield ~ colm + rowm, data=AragonezPoints)
```

|    | np    | dist      | gamma     | dir.hor | dir.ver | id   |
|----|-------|-----------|-----------|---------|---------|------|
| 1  | 979   | 2.250000  | 0.8670795 | 0       | 0       | var1 |
| 2  | 3796  | 4.246580  | 0.9079705 | 0       | 0       | var1 |
| 3  | 7270  | 7.127722  | 0.9625727 | 0       | 0       | var1 |
| 4  | 7864  | 9.849207  | 0.9676781 | 0       | 0       | var1 |
| 5  | 12517 | 12.594300 | 0.9825738 | 0       | 0       | var1 |
| 6  | 12825 | 15.568268 | 0.9894840 | 0       | 0       | var1 |
| 7  | 16747 | 18.627263 | 1.0070784 | 0       | 0       | var1 |
| 8  | 15290 | 21.530216 | 1.0010836 | 0       | 0       | var1 |
| 9  | 17219 | 24.104900 | 1.0098093 | 0       | 0       | var1 |
| 10 | 18576 | 26.924936 | 1.0270918 | 0       | 0       | var1 |
| 11 | 22484 | 30.008119 | 1.0328343 | 0       | 0       | var1 |
| 12 | 19601 | 32.950881 | 1.0322501 | 0       | 0       | var1 |
| 13 | 20476 | 35.629361 | 1.0271511 | 0       | 0       | var1 |
| 14 | 22326 | 38.471422 | 1.0599523 | 0       | 0       | var1 |
| 15 | 23360 | 41.534207 | 1.0627653 | 0       | 0       | var1 |

# Variogram for the linearly detrended yields



The semi-variogram clearly **flattens out**, suggesting that the **linear detrending has been more successful** than detrending by just a constant value.

The sill appears to be at approximately 1 and the nugget value at 0.8.

# The variog function

An alternative function to compute empirical variograms is the **variog** function in package **geoR**.

```
> library(geoR)
> variog(coords=coordinates(AragonezPoints), data=AragonezPoints$yieldldt)

$u
[1] 4.938846 14.816537 24.694228 34.571920 44.449611 54.327303
[7] 64.204994 74.082685 83.960377 93.838068 103.715759 113.593451
[13] 123.471142

$v
[1] 0.9410075 0.9930845 1.0147577 1.0401008 1.0638983 1.1092289 1.1442758
[8] 1.1868496 1.2252729 1.2358345 1.2514819 1.2040493 1.0811141

$n
[1] 16437 44064 61984 73429 75419 71508 65236 52279 34855 16563 5346 1407
[13] 143
[...]
```

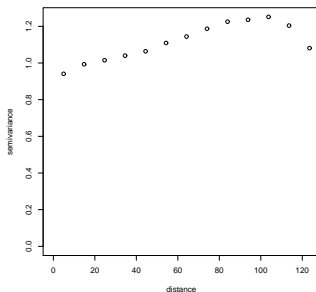
**u** gives the lags  $h$ ; **v** the values  $\gamma(h)$ ; **n** the points used to estimate  $\gamma(h)$ .

**n** decreases substantially for large  $h$ , making the estimates of  $\gamma(h)$  fall for large  $h$ .

## Plotting the variog results

Unlike variogram, the `variog` function uses, by default, all the spatial lags  $h$  (as center points in intervals, or `bins`) for which it finds pairs of points. A `max.dist` argument controls the maximum distance  $h$  used.

```
> plot(variog(coords=coordinates(AragonezPoints),  
+         data=AragonezPoints$yieldldt))
```



# Variogram models

Several classes of functions have been proposed for **smooth semi-variogram curves**. Among them:

- **exponential model**: for  $h > 0$ , the semi-variogram is given by:

$$\gamma(h) = b + psill \left[ 1 - e^{-\frac{h}{r}} \right] , \quad (40)$$

where  $b$  is the nugget,  $r$  the range and  $psill$  the partial sill. This function grows towards an **asymptotic sill** (given by  $b + psill$ ), which is not attained.

- **spherical model**: for  $0 < h < r$ , the semi-variogram is given by:

$$\gamma(h) = b + psill \left[ \frac{3}{2} \frac{h}{r} - \frac{1}{2} \left( \frac{h}{r} \right)^3 \right] \quad \text{for } h < r , \quad (41)$$

with  $\gamma(h) = sill = b + psill$  for  $h > r$ . This model assumes that for  $h > r$  there ceases to be spatial dependence and thereafter the semi-variogram  $\gamma(h)$  is constant.

# Variogram models

- **gaussian model**: for  $h > 0$ , the semi-variogram is given by:

$$\gamma(h) = b + psill \left[ 1 - e^{-\frac{h^2}{r^2}} \right], \quad (42)$$

with constants defined as above.

The **gstat** package provides tools to

- fit such models;
- plot smooth curves on the empirical semi-variogram; and
- estimate the sill, the nugget effect and the range.



# The vgm function

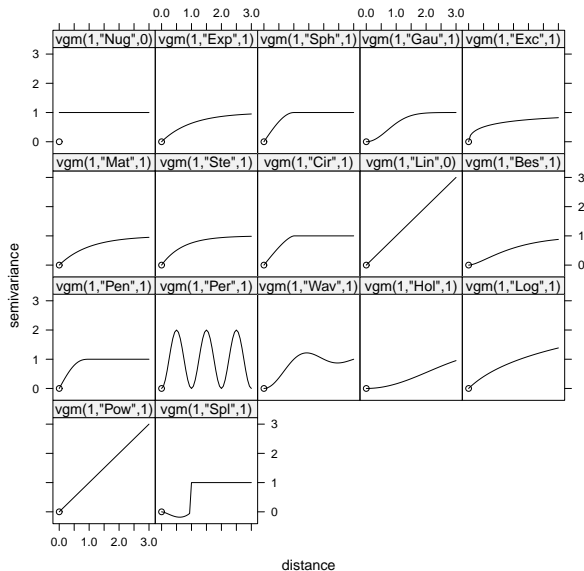
Model functions are specified with the `vgm` command in `gstat`, which requires as arguments:

- `psill`: an initial estimate of the partial sill (the difference between the sill and the nugget);
- `nugget`: an initial estimate of the nugget effect  $b$ ;
- `range`: an initial estimate of the range  $r$ ;
- `model`: the class of model function.

The shapes of `available variogram models` can be viewed by typing the following command, with results in the next slide:

```
> show.vgms()
```

# Variogram models available with vgm



# The `fit.variogram` function

To fit a variogram model, package `gstat` has the `fit.variogram` function, with two main arguments:

- an empirical semi-variogram; and
- a model class of functions.

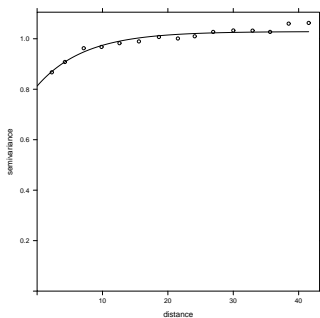
An exponential model fitted to the linearly detrended Aragonez yields:

```
> AragVarioLin <- variogram(yield ~ colm + rowm,  
+                           data=AragonezPoints, locations=coordinates(AragonezPoints))  
> m.fit <- fit.variogram(AragVarioLin,  
+                         model=vgm(psill=0.3,"Exp", range=7, nugget=0.8))  
  
> m.fit  
  model      psill      range  
1  Nug 0.8125333 0.000000  
2  Exp 0.2160780 7.264675
```

## Plotting a variogram model

The empirical semi-variogram and fitted (exponential) variogram model from the previous slide can be jointly plotted with the following command and the results shown below:

```
> plot(AragVarioLin, m.fit)
```



# A spherical model

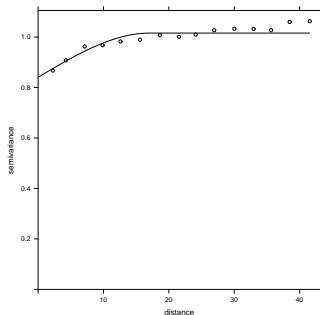
A spherical model fitted to the same empirical variogram:

```
> m2.fit <- fit.variogram(AragVarioLin,  
+                          model=vgm(psill=0.3,"Sph", range=7, nugget=0.8))  
  
> m2.fit  
  model      psill    range  
1  Nug 0.8407014 0.000000  
2  Sph 0.1750312 17.06707
```

While both models provide similar estimates for the partial sill and nugget, the estimates for the range differ substantially. The exponential model is probably better suited in this case, since unlike the spherical model, it does not assume a constant  $\gamma(h)$  from some point onwards.

# The plotted spherical model

```
> plot(AragVarioLin, m2.fit)
```



## Variogram models in package geoR

The `geoR` package also has functionalities to fit variogram models for the empirical variograms estimated with its `variog` function. They allow us to try on various models 'by hand'.

After plotting an empirical variogram, we can use the function `lines.variomodel` to fit a given model function, with parameters provided by the user.

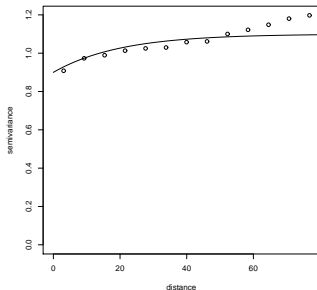
The `lines.variomodel` function arguments are:

- `cov.model` specifies the type of model (see details in the helpfile for `cov.spatial`);
- `cov.pars`, a `vector` with the values for the `partial sill` and `range`;
- `nugget`, the value for the nugget effect.

## Using the geoR functions

For the linearly detrended Aragonez yields, the commands to compute an empirical variogram (with a maximum lag  $h = 80$ ), plot it, and try out an exponential model:

```
> AragVariog <- variog(coords=coordinates(AragonezPoints),  
+                       data=AragonezPoints$yieldldt, max.dist=80)  
> plot(AragVariog)  
> lines.variomodel(cov.model="exp", cov.pars=c(0.2, 20), nugget=0.9)
```





# Anisotropy

**Anisotropy** is harder to identify and to work with.

The authors of the `gstat` package provide an argument **alpha** for the `variogram` function, which allows the user to define a vector of angles giving the main directions along which to inspect if the resulting semi-variograms are similar.

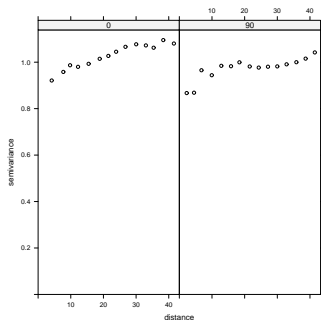
```
> variogram(yield ~ colm + rowm , data=AragonezPoints, alpha=c(0,90))
```

|       | np    | dist      | gamma     | dir.hor | dir.ver | id   |
|-------|-------|-----------|-----------|---------|---------|------|
| 1     | 2844  | 4.161751  | 0.9211549 | 0       | 0       | var1 |
| 2     | 2736  | 7.718343  | 0.9580814 | 0       | 0       | var1 |
| 3     | 4359  | 9.783072  | 0.9868073 | 0       | 0       | var1 |
| [...] |       |           |           |         |         |      |
| 13    | 12495 | 38.320527 | 1.0951004 | 0       | 0       | var1 |
| 14    | 12660 | 41.576874 | 1.0803228 | 0       | 0       | var1 |
| 15    | 979   | 2.250000  | 0.8670795 | 90      | 0       | var1 |
| 16    | 952   | 4.500000  | 0.8685836 | 90      | 0       | var1 |
| 17    | 4534  | 6.771318  | 0.9652830 | 90      | 0       | var1 |
| [...] |       |           |           |         |         |      |
| 28    | 9831  | 38.663207 | 1.0152797 | 90      | 0       | var1 |
| 29    | 10700 | 41.483723 | 1.0419915 | 90      | 0       | var1 |

# Variograms for different directions

Requesting a `plot` of the previous command gives the **two empirical semi-variograms** which, in relation to the each observation, are **in the angular sectors** defined by the two main bisecting lines.

With anisotropy, we expect to see differences in the semi-variograms for points on the **vertical (0 degrees)** and **horizontal (90 degrees)** directions.



## Correlograms

For isotropic models, the **correlogram**, or **autocorrelation function**, may be easier to interpret. It basically considers the **correlation coefficient** between observations that are separated by a spatial lag  $h$ :

$$\rho(h) = \frac{\text{Cov}[Z(s), Z(s+h)]}{\sqrt{\text{Var}[Z(s)] \text{Var}[Z(s+h)]}} = \frac{C_s(h)}{C_s(0)}. \quad (43)$$

The **semi-variogram**  $\gamma(h)$  and the correlogram  $\rho(h)$  are related:

$$\begin{aligned} \gamma(h) &= C_s(0) - C_s(h) = C_s(0) \left[ 1 - \frac{C_s(h)}{C_s(0)} \right] \\ \Leftrightarrow \gamma_s(h) &= C_s(0) [1 - \rho(h)]. \end{aligned} \quad (44)$$

The intuitively obvious relation  $\lim_{h \rightarrow +\infty} \rho(h) = 0$  is coherent with the idea that the sill is the asymptotic value of the semi-variogram as  $h$  tends to infinity. It is also natural that  $\gamma_s(0) = 0$ , since  $\rho(0) = 1$

## A meteorological dataset

Downloaded from the website of the European Centre for Medium-Range Weather Forecasts (ECMWF)<sup>1</sup>. The data are **reanalysis** data (pre-processed in this case by the ERA-Interim data assimilation system).

For a given hour of June 18, 2016, reanalysis values were obtained, relative to a **rectangular grid covering 24 longitudes from 9W to 8E and 23 latitudes from 36N to 52N**.

The variables in the dataset are:

| Short name | Long name                          | Units |
|------------|------------------------------------|-------|
| t2m        | temperature at 2 meters            | °K    |
| stl1       | soil temperature level 1 (surface) | °K    |
| stl2       | soil temperature level 2           | °K    |
| sund       | sunshine duration                  | (s)   |
| tp         | total precipitation                | (m)   |

---

<sup>1</sup>[apps.ecmwf.int/datasets/interim-full-daily](https://apps.ecmwf.int/datasets/interim-full-daily)

## Some data values

The data format in the ERA-Interim website is [NCDF](#) and, with the R packages [ncdf4](#) and [raster](#), the dataset was transformed into an R data frame, called [meteo](#), whose first six lines are shown below:

```
> head(meteo)
```

|   | lon   | lat  | t2m      | stl1     | stl2     | sund     | tp           |
|---|-------|------|----------|----------|----------|----------|--------------|
| 1 | -9.00 | 52.5 | 283.7192 | 284.7060 | 286.6936 | 23399.97 | 0.0012258549 |
| 2 | -8.25 | 52.5 | 283.6690 | 284.8637 | 286.8675 | 22049.91 | 0.0009417163 |
| 3 | -7.50 | 52.5 | 284.0929 | 285.2671 | 287.2538 | 22219.33 | 0.0010797265 |
| 4 | -6.75 | 52.5 | 284.6273 | 285.6325 | 287.4712 | 22949.73 | 0.0012786235 |
| 5 | -6.00 | 52.5 | 285.9954 | 285.8502 | 285.8498 | 24637.31 | 0.0007062872 |
| 6 | -5.25 | 52.5 | 285.9974 | 285.9481 | 285.9473 | 25986.71 | 0.0007062872 |

The [longitudes](#) had to be converted to the range  $-9$  to  $8$ , to ensure contiguous plotting of results.

# Correlations

The **standard linear correlation coefficients** (rounded to two decimal places) are given below. Unsurprisingly, they reveal **strong positive correlations between the three temperature variables and negative correlations between rainfall and the temperature variables**. Sunshine duration is almost uncorrelated with most variables, with a small negative correlation with total precipitation.

```
> round(cor(meteo[,3:7]),d=2)
```

|      | t2m   | stl1  | stl2  | sund  | tp    |
|------|-------|-------|-------|-------|-------|
| t2m  | 1.00  | 0.97  | 0.86  | -0.01 | -0.47 |
| stl1 | 0.97  | 1.00  | 0.95  | 0.02  | -0.50 |
| stl2 | 0.86  | 0.95  | 1.00  | 0.05  | -0.50 |
| sund | -0.01 | 0.02  | 0.05  | 1.00  | -0.24 |
| tp   | -0.47 | -0.50 | -0.50 | -0.24 | 1.00  |

# Spatial objects

As before, we build objects of class `SpatialPoints` and `SpatialPointsDataFrame`.

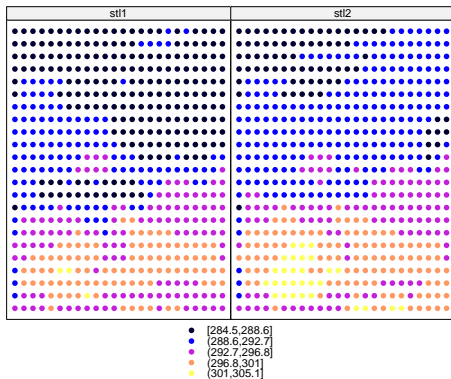
The `proj4str` argument with the geographical coordinate reference system is specified, so that the data may be geo-referenced.

```
> meteo.SP <- SpatialPoints(coord=meteo[,c("lon","lat")],  
+                           proj4string=CRS("+proj=longlat +datum=WGS84"))  
> meteo.data <- SpatialPointsDataFrame(coords=meteo.SP,  
+                                     data=meteo[,c("t2m","stl1","stl2","sund","tp")])
```

The `spplot` is used to preview variables with similar values.

```
> spplot(meteo.data, zcol=c("stl1","stl2"))
```

# N-S gradient

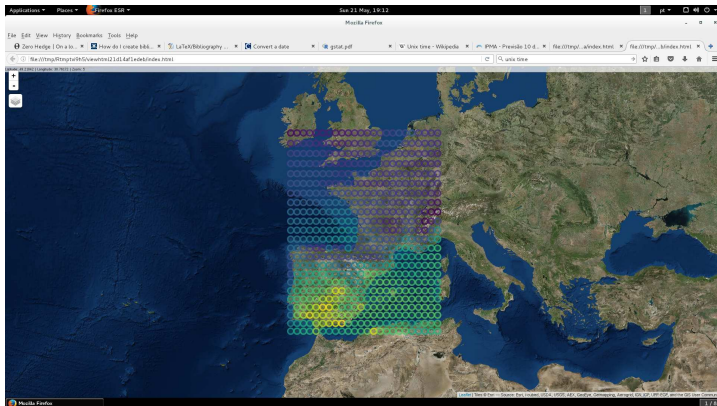


A North-South soil temperature gradient is clearly visible for temperatures `st1` (surface temperature level 1) and `st2` (surface temperature level 2).



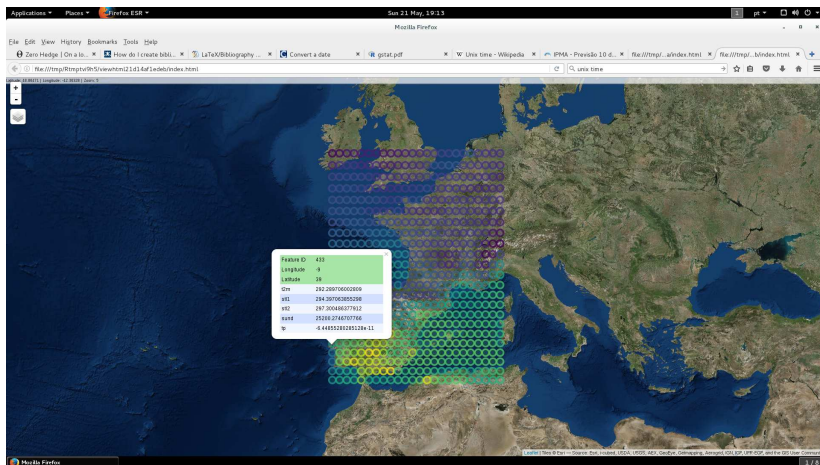
# Meteorological data with mapview

```
> library(mapview)
> mapView(meteo.data, zcol="stl1")
```



A small [dialogue window](#) on the left of the browser window will allow you to [select](#) different types of maps. This is the “ESRI.WorldImagery” map option.

# Interactive information



Clicking on any of the circles will open a window with the information for that location.

## Several variables

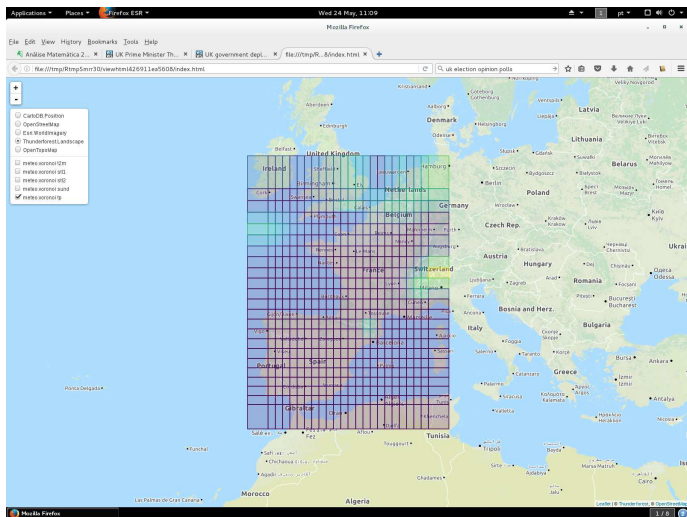
Several variables can be made available, using the `mapView` function, simply by providing a `vector` with their names to the `zcol` argument.

But first, the object of class `SpatialPointsDataFrame` object, `meteo.data`, will be converted into an object of class `SpatialPolygonsDataFrame` object using the `voronoi` function from package `dismo`:

```
> library(dismo)
> meteo.voronoi <- voronoi(meteo.data)
> mapView(meteo.voronoi, zcol=colnames(meteo.voronoi@data))
```

# Interactive visualisation of different variables

All five observed variables are available, and can be selected using the dialogue window on the left of the screen. The screenshot displays variable **tp**.



# Relating different variables

We will briefly consider some concepts relating to **spatial correlation** between **different** variables.

The study of relations between different variables is frequent in standard statistics. If spatial autocorrelation and cross-correlation between different variables exists, it should be taken into account.

The variables may, or may not, be **collocated** (*co-located*), that is, if they are observed at the same set of locations. In what follows, we assume that different variables **are** collocated.

If the observed variables are **not** collocated, we should **interpolate** in order to obtain a collocated set of data (an issue for next week!).

# The cross-variogram

The **variogram** for a **single** variable  $Z_{[i]}$  was defined in equation (35) as:

$$2\gamma_{ii}(h) = \text{Var} (Z_{[i]}(s) - Z_{[i]}(s + h)) \quad (45)$$

The usual extension to a pair of different variables,  $Z_{[i]}$  and  $Z_{[j]}$ , is:

$$2\gamma_{ij}(h) = \text{Cov} [(Z_{[i]}(s) - Z_{[i]}(s + h)), (Z_{[j]}(s) - Z_{[j]}(s + h))] \quad (46)$$

Cressie gives an alternative definition:

$$2\gamma_{ij}(h) = \text{Var} (Z_{[i]}(s) - Z_{[j]}(s + h)) \quad (47)$$

Both extensions give the standard variogram when  $i=j$ .

# Cross-variograms in R

The `gstat` package produces cross-variograms.

We define an object of class `gstat` which collects and detrends variables. Objects of class `gstat` may be attached to each other.

Each variable in the `meteo` dataset, will be detrended using a linear trend on the geographical coordinates:

```
> gobj <- gstat(NULL, "t2m", t2m ~ lon + lat, meteo.data)
> gobj <- gstat(gobj, "stl1" , stl1 ~ lon + lat, meteo.data)
> gobj <- gstat(gobj, "stl2" , stl2 ~ lon + lat, meteo.data)
> gobj <- gstat(gobj, "sund" , sund ~ lon + lat, meteo.data)
> gobj <- gstat(gobj, "tp" , tp ~ lon + lat, meteo.data)
```

# The variogram function for cross-variograms

```
> gobj
```

```
data:
```

```
t2m : formula = t2m ~ lon + lat ; data dim = 552 x 5
```

```
stl1 : formula = stl1 ~ lon + lat ; data dim = 552 x 5
```

```
stl2 : formula = stl2 ~ lon + lat ; data dim = 552 x 5
```

```
sund : formula = sund ~ lon + lat ; data dim = 552 x 5
```

```
tp : formula = tp ~ lon + lat ; data dim = 552 x 5
```

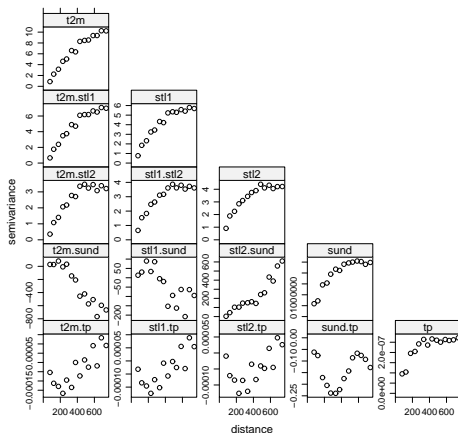
A call to `gstat::variogram` function will compute both the empirical variograms and the empirical cross-variograms:

```
> vario.meteo <- variogram(gobj)
```

```
> plot(vario.meteo)
```



# Plotted empirical cross-variograms



The variables whose cross-variograms have a clearer pattern are best suited for subsequent use in spatial models that use information from multiple variables.

# Fitting (cross-)variogram models

Variogram models may be fitted to the empirical variograms and cross-variograms, using the `fit.lmc` function, as shown below (to fit a spherical model in all cases)

The numerical estimates of the ranges, nuggets and partial sills can be viewed by just writing the name of the object that results from invoking the `fit.lmc` function.

```
> vmeteo.fit <- fit.lmc(vario.meteo, gobj,  
+                       vgm(psill=1, "Sph", range=800, nugget=1))
```

# Fitting (cross-)variogram models (cont.)

```
> vmeteo.fit
[...]
```

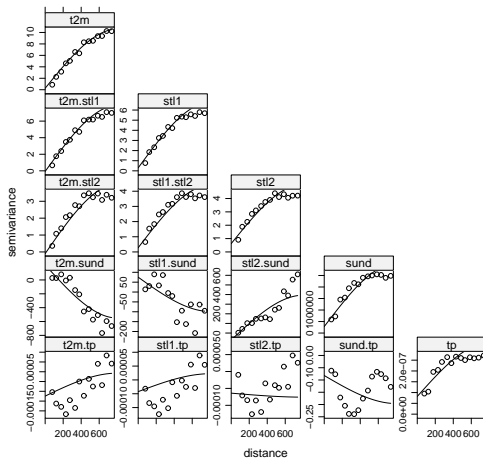
variograms:

|              | model | psill         | range |
|--------------|-------|---------------|-------|
| t2m[1]       | Nug   | 2.558868e-01  | 0     |
| t2m[2]       | Sph   | 1.032814e+01  | 800   |
| stl1[1]      | Nug   | 2.932069e-01  | 0     |
| stl1[2]      | Sph   | 6.462121e+00  | 800   |
| stl2[1]      | Nug   | 6.235986e-01  | 0     |
| stl2[2]      | Sph   | 4.442875e+00  | 800   |
| sund[1]      | Nug   | 2.818659e+05  | 0     |
| sund[2]      | Sph   | 2.730965e+06  | 800   |
| tp[1]        | Nug   | 8.193747e-08  | 0     |
| tp[2]        | Sph   | 2.485045e-07  | 800   |
| t2m.stl1[1]  | Nug   | -6.684264e-02 | 0     |
| t2m.stl1[2]  | Sph   | 7.955068e+00  | 800   |
| t2m.stl2[1]  | Nug   | -9.927164e-02 | 0     |
| t2m.stl2[2]  | Sph   | 4.107862e+00  | 800   |
| stl1.stl2[1] | Nug   | 2.779786e-01  | 0     |
| stl1.stl2[2] | Sph   | 4.119670e+00  | 800   |
| t2m.sund[1]  | Nug   | 2.177144e+02  | 0     |
| t2m.sund[2]  | Sph   | -7.672389e+02 | 800   |
| stl1.sund[1] | Nug   | 7.526683e+01  | 0     |
| stl1.sund[2] | Sph   | -1.738549e+02 | 800   |
| stl2.sund[1] | Nug   | -7.237660e+01 | 0     |
| stl2.sund[2] | Sph   | 4.655195e+02  | 800   |
| t2m.tp[1]    | Nug   | -7.439963e-05 | 0     |
| t2m.tp[2]    | Sph   | 1.172327e-04  | 800   |

```
[...]
```

# Plotted (cross-)variogram model fits

```
> plot(vario.meteo, vmeteo.fit)
```



Next week you will use the concepts covered so far to specify models, similar to linear regressions, but with spatial autocorrelation.

Good luck for the rest of the course.