Spatial Data: a brief statistical introduction

OpenSpat – Class notes

Jorge Cadima

Instituto Superior de Agronomia – Universidade de Lisboa

May 30 and 31, 2019

Goals

We 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 $s_1, s_2, ..., s_n \in \mathcal{S}$, producing a (non-independent) sample of size n:

$$(Z(s_1), Z(s_2), Z(s_3), ..., Z(s_n))$$

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:

$$(Y_1, Y_2, Y_3, ..., Y_t)$$

  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 μ, when the population variance σ^2 is known:

$$\left] \overline{y} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \;,\; \overline{y} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \right[. \tag{1}$$

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:

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

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

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

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

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

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

Consequences of the independence model

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

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

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

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

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:

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

λ 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 ϵ_j $(j \leq i)$:

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

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

$$\begin{cases} Y_i = \mu + \sum_{j=1}^{i} \lambda^{i-j}\,\epsilon_j \\ \epsilon_i \sim \mathcal{N}(0, \sigma^2) \qquad (i.i.d.) \ . \end{cases} \tag{4}$$

Comparing the models

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

	Independence	AR(1)
$E[Y_i]$	μ	μ
$V[Y_i]$	σ^2	$\sigma^2 \left(\frac{1-\lambda^{2i}}{1-\lambda^2}\right) \rightarrow \frac{\sigma^2}{1-\lambda^2}$
$Cov[Y_i, Y_j]$ (for $i > j$)	0	$\lambda^{i-j} V[Y_j] \rightarrow \lambda^{i-j} \frac{\sigma^2}{1-\lambda^2}$
r_{ij} (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 $\overline{Y}$ with the AR(1) model

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

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

The sample mean can be re-written as:

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

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

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

If $0<\lambda<1$, for any n and t, $V[\overline{Y}] > \frac{\sigma^2}{n}$. The true sampling variance of $\overline{Y}$ 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:

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

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

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

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

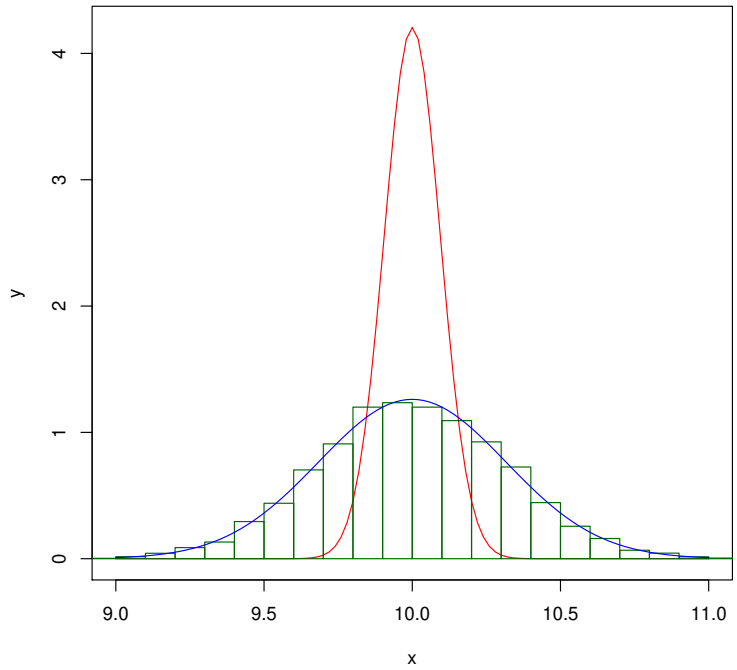


Figure: Histogram of the distribution of $\overline{y}$, for 10 000 repetitions of size $n = 1000$ samples, in an AR(1) model. Parameters: $\mu = 10$, $\sigma = 3$, $\lambda = 0.7$. Red curve: $\mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$ distribution of $\overline{Y}$ under independence. Blue curve: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

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

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 μ.

Effective sample size

A useful concept is that of effective sample size, defined as the value n_ϵ such that:

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

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	λ								
	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 σ^2

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

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

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

An (approximately) unbiased estimator of σ^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 $(1-\lambda^2)S^2$

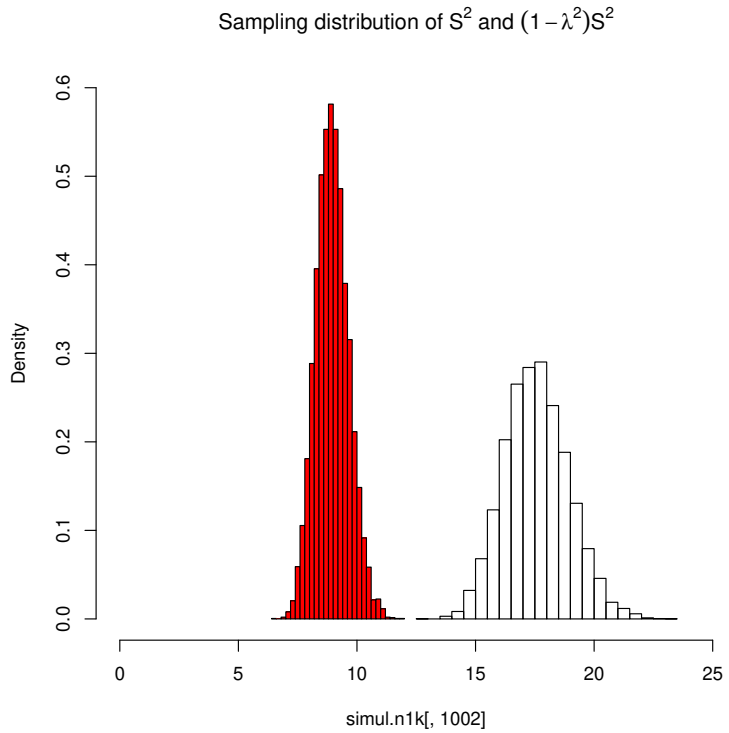


Figure: Histograms of the distributions of s^2 (in black and white) and $(1-\lambda^2)\,s^2$ (in red), for 10 000 repetitions of size $n=1000$ samples, in an AR(1) model. Parameters: $\mu=10$, $\sigma=3$, $\lambda=0.7$. The true variance is $\sigma^2=9$.

Confidence intervals for μ when σ^2 is unknown

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

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

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

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

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

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

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 μ when σ^2 is unknown

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

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

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 as Tinta Roriz or Tempranillo.

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

Goal: study the variety's yields.

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 are defined relative to an area (areal data) or plant. But we consider them as geostatistical data: we assume that yields vary continuously over the trial field.

The Aragonez trial field

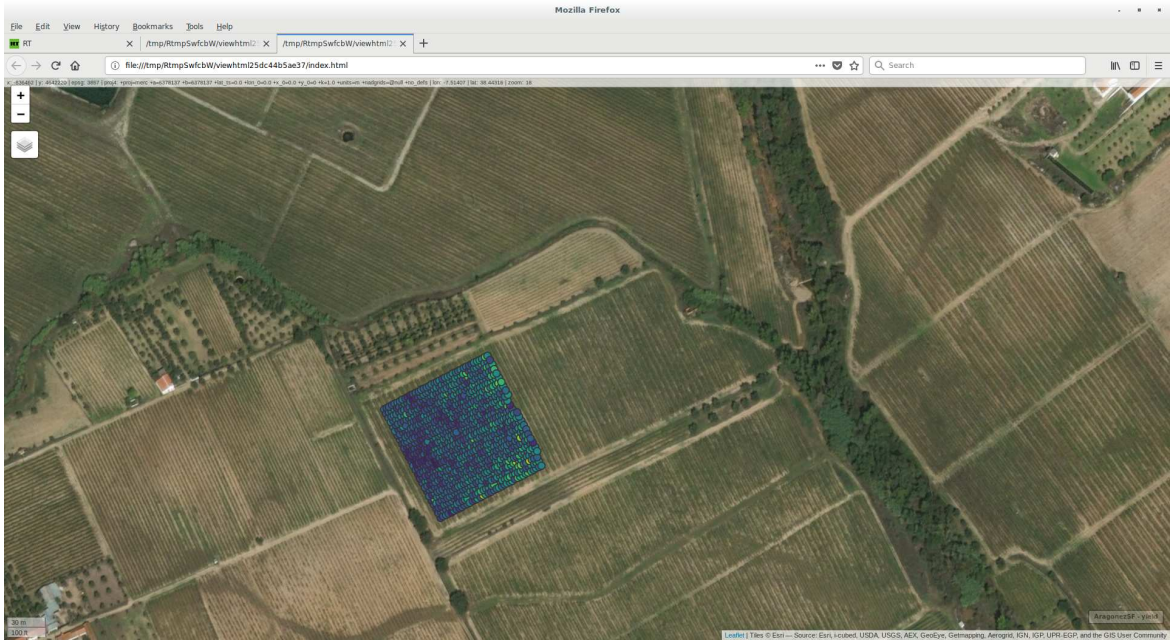


Figure: The Aragonez yields on the field. The southernmost corner has coordinates 38.4411239 N and 7.5159353 W.

The Aragonez data

The dataset was originally 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 several `sf` and `sp` objects were created.

Aragonez sf and sp objects

The sf object AragonezSF with point data.

```
> AragonezSF
Simple feature collection with 1019 features and 7 fields
geometry type:  POINT
dimension:      XY
bbox:           xmin: -7.516431 ymin: 38.44118 xmax: -7.515039 ymax: 38.4423
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
First 10 features:
genotype block col row colm  rowm yield                    geometry
1     RZ717    B1   4   2    0 93.75 2.417 POINT (-7.516431 38.44193)
2    RZ1158    B1   4   9    0 67.50 2.724 POINT (-7.516291 38.44172)
3    RZ1325    B1   4   6    0 78.75 2.647 POINT (-7.516351 38.44181)
4    RZ3313    B1   4   8    0 71.25 1.543 POINT (-7.516311 38.44175)
5    RZ3603    B1   4  12    0 56.25 0.865 POINT (-7.516231 38.44163)
6    RZ3604    B1   4   3    0 90.00 1.659  POINT (-7.516411 38.4419)
7    RZ3803    B1   4  13    0 52.50 0.481  POINT (-7.516211 38.4416)
8    RZ3902    B1   4  10    0 63.75 1.203 POINT (-7.516271 38.44169)
9    RZ6201    B1   4   4    0 86.25 2.108 POINT (-7.516391 38.44187)
10   RZ6204    B1   4   5    0 82.50 3.561 POINT (-7.516371 38.44184)
```

Aragonez sf and sp objects (cont.)

The sf object Aragonez3763Vor with polygon data (via Voronoi).

```
> Aragonez3763Vor
Simple feature collection with 1019 features and 7 fields
geometry type:  POLYGON
dimension:      XY
bbox:           xmin: 53834.87 ymin: -136048.6 xmax: 53962.69 ymax: -135918
epsg (SRID):    3763
proj4string:    +proj=tmerc +lat_0=39.66825833333333 +lon_0=-8.133108333333334 +k=1 +x_0=0 +y_0=0 +ellps=GRS80
First 10 features:
genotype block col row colm  rowm yield                          geometry
1     RZ717    B1   4    2    0 93.75 2.417 POLYGON ((53837.5 -135959.4...
2    RZ1158    B1   4    9    0 67.50 2.724 POLYGON ((53839.54 -135958....
3    RZ1325    B1   4    6    0 78.75 2.647 POLYGON ((53836.11 -135965....
4    RZ3313    B1   4    8    0 71.25 1.543 POLYGON ((53841.53 -135967,...
5    RZ3603    B1   4   12    0 56.25 0.865 POLYGON ((53841.58 -135957....
6    RZ3604    B1   4    3    0 90.00 1.659 POLYGON ((53837.88 -135969,...
7    RZ3803    B1   4   13    0 52.50 0.481 POLYGON ((53843.3 -135970.4...
8    RZ3902    B1   4   10    0 63.75 1.203 POLYGON ((53843.62 -135956....
9    RZ6201    B1   4    4    0 86.25 2.108 POLYGON ((53839.65 -135972....
10   RZ6204    B1   4    5    0 82.50 3.561 POLYGON ((53845.07 -135973....
```

Plotting the Aragonez cell yields

There is a `plot` method for `sf` objects, allowing us to view the yields:

```
> plot(Aragonez3763Vor[,"yield"], key.pos=4)
```

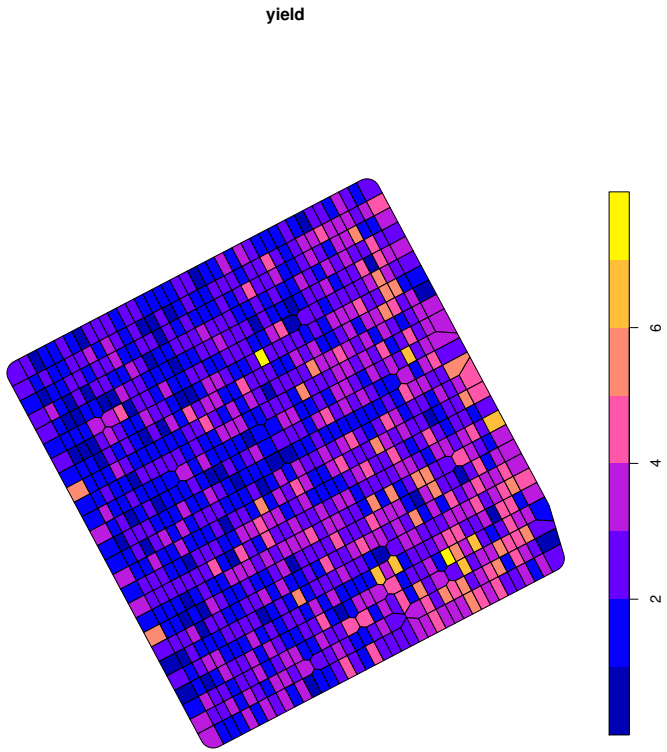


Figure: Yields tend to increase from left to right on the trial field. There seems to be an underlying (linear?) trend. The `key.pos` argument controlling where the legend is displayed.

Aragonez sf and sp objects (cont.)

A SpatialPointsDataFrame (sp) object is sometimes needed.

It can be created, as shown below (with CRS 3763).
AragonezPoints has 5 slots where the information is stored.

```
> Aragonez3763 <- st_transform(AragonezSF, crs=3763)
> AragonezPoints <- as_Spatial(Aragonez3763)
> 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] 53838 53850 53845 53849 53856 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
..@ bbox       : num [1:2, 1:2] 53838 -136046 53960 -135921
.. ..- 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=tmerc +lat_0=39.66825833333333 +lon_0=-8.133108333333334 +k=1 +x_0=0 +y_0=0 +ell
```

Plotting the Aragonez cell yields

For `sp` objects, package sp provides an `spplot` function:

```
> spplot(AragonezPoints ,zcol="yield", key.space="right")
```

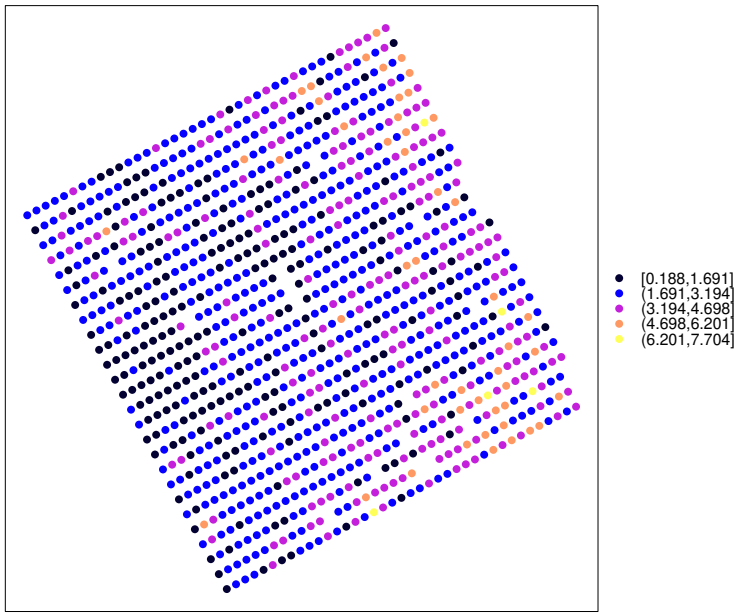


Figure: A similar view of the trend, based on a `SpatialPointsDataFrame` object. The `key.space` argument controls where the legend is displayed.

2-D Spatial autocorrelation and trends

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:

$$Z(x, y) \;=\; T(x, y) + \eta(x, y) + \epsilon(x, y) \;, \tag{12}$$

where:

- $T(x, y)$ is a deterministic (non-random) underlying spatial trend (sometimes further separated into $\mu + T(x, y)$, where μ 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:

$$z = \beta_0 + \beta_1 x + \beta_2 y \ . \tag{13}$$

- removing a quadratic trend on the geographical coordinates:

$$z = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 y^2 + \beta_5 xy \ . \tag{14}$$

Trends may also be defined by other (non-spatial) variables (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:

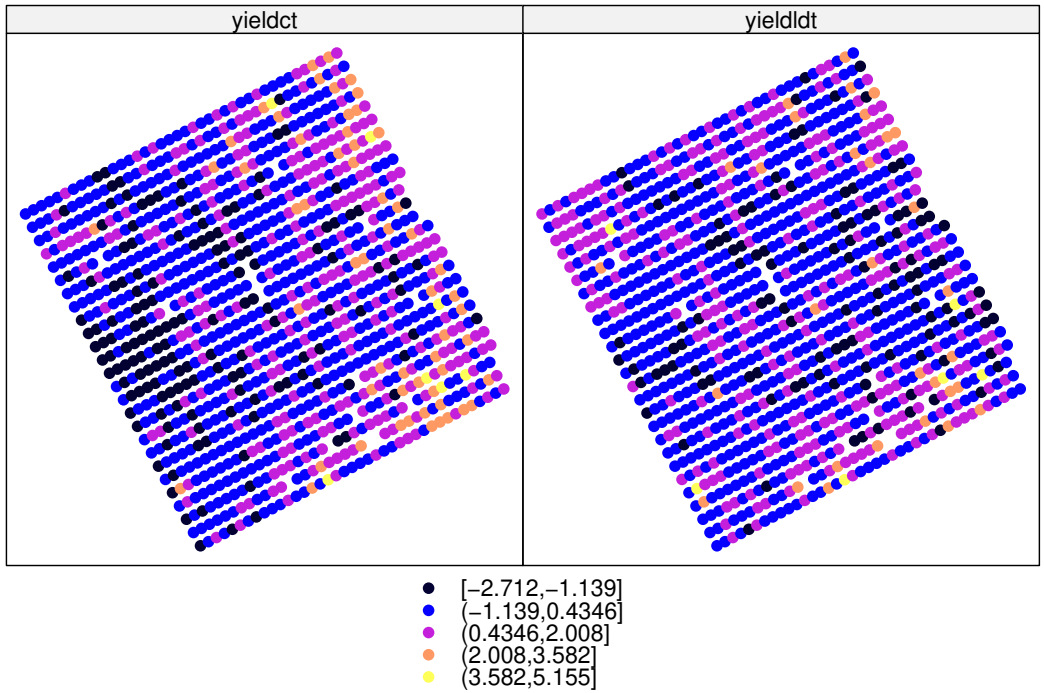
```
> head(AragonezPoints@data)
  genotype block col row colm  rowm yield     yieldct    yieldldt
1    RZ717    B1   4   2    0 93.75 2.417 -0.13168302  0.87742187
2   RZ1158    B1   4   9    0 67.50 2.724  0.17531698  1.08176580
3   RZ1325    B1   4   6    0 78.75 2.647  0.09831698  1.04876126
4   RZ3313    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
```

Viewing the detrended yields in the Aragonez data

The `sp::spplot` function allows us to view the detrended yields:

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

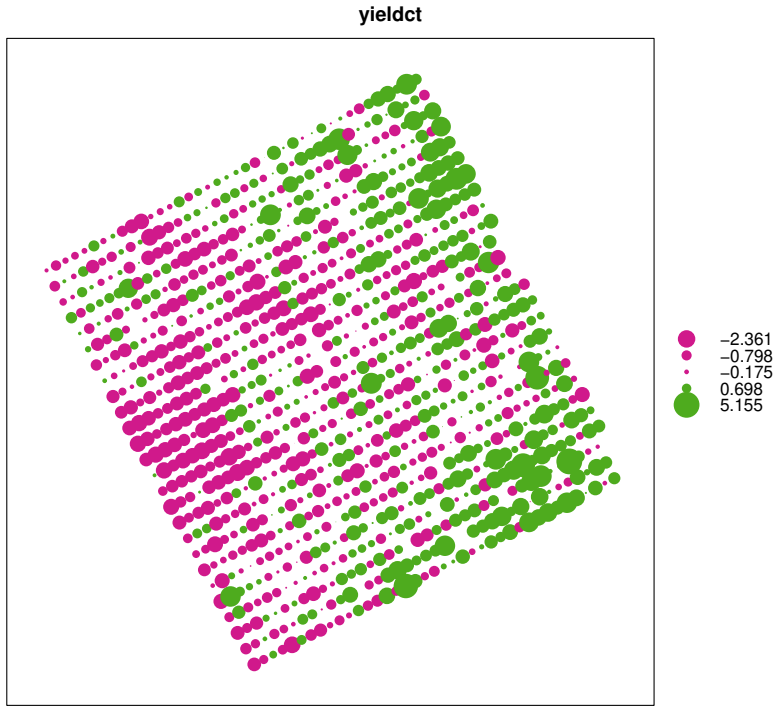
The `layout` option specifies the 'matrix' layout for multiple graphs (columns $\times$ rows).
Linear detrending seems to have broken down the original pattern.



Bubble plots

The `sp::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:

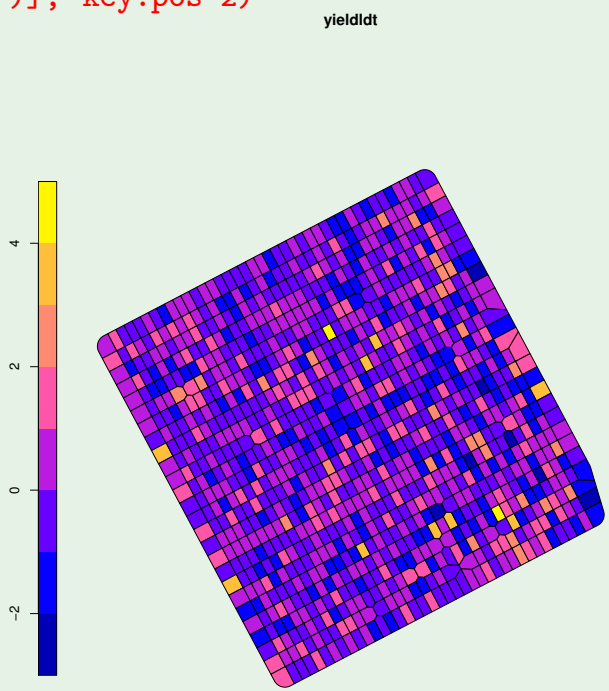
```
> bubble(AragonezPoints, zcol="yieldldt")
```

Areal data and polygons

Data of areal type are spatially described by polygons. But it is necessary to again create the two new columns for the detrended yields.

Here for the `sf` `Aragonez3763Vor` Voronoi polygons:

```
> Aragonez3763Vor$yieldct <- Aragonez3763Vor$yield-mean(Aragonez3763Vor$yield)
> Aragonez3763Vor$yielddt <- residuals(Arag.lm)
> plot(Aragonez3763Vor[,c("yielddt")], key.pos=2)
```

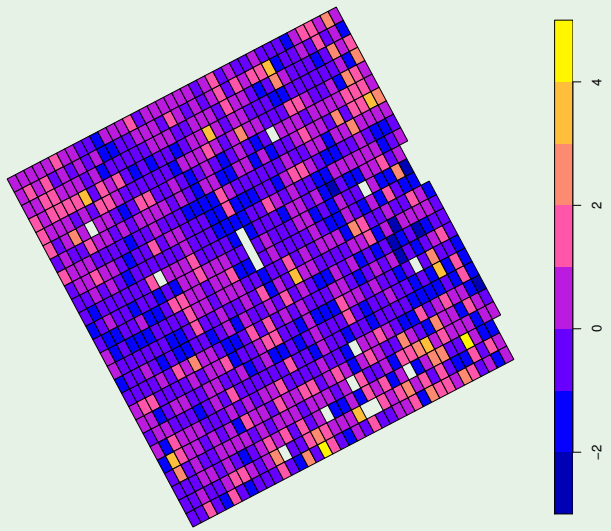


Areal data and polygons (cont.)

Now for the `sf` `Aragonez3763Grid` polygons.

Here, empty polygons are associated with missing values.

```
> Aragonez3763Grid$yieldct <- Aragonez3763Grid$yield-mean(Aragonez3763Grid$yield)
> Aragonez3763Grid$yielddt <- residuals(Arag.lm)
> plot(Aragonez3763Grid[,c("yielddt")], key.pos=4)
```



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 η at other points in some vicinity of (x, y).

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

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

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:

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

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:

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

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

$$w_{ij} = \frac{1}{d_{ij}^a} ; \tag{18}$$

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:

$$w_{ij} = \mathrm{e}^{-a\, d_{ij}} \ . \tag{19}$$

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 also 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 polygons to define pairs of neighbours, or via a distance-based criterion for neighbourhoods.

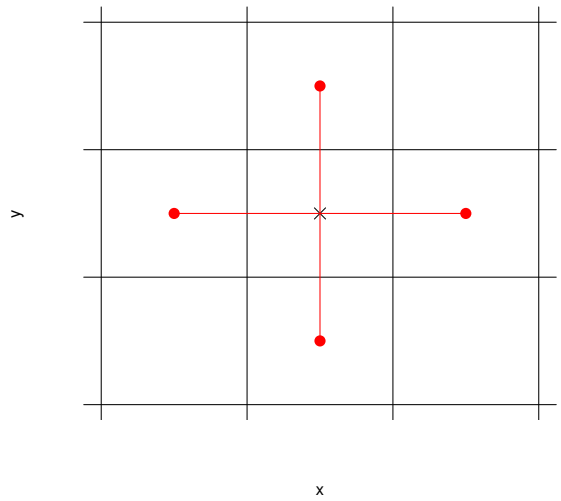
A standard convention in spatial statistics is that a polygon is not a neighbour of itself.

The rook's case

There are two famous conventions to define neighbourhoods for polygons: 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 dimension 1 (curve).

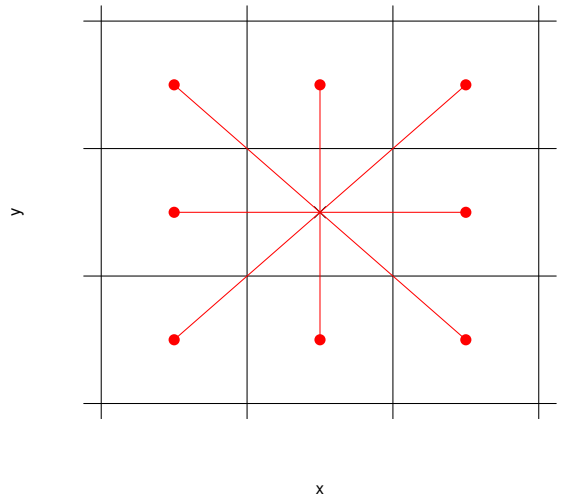
The name rook's case originates from the adjacent chessboard squares to which a rook can move, as illustrated below. The patchwork of cells does not have to be a rectangular grid for the definition to apply.



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.

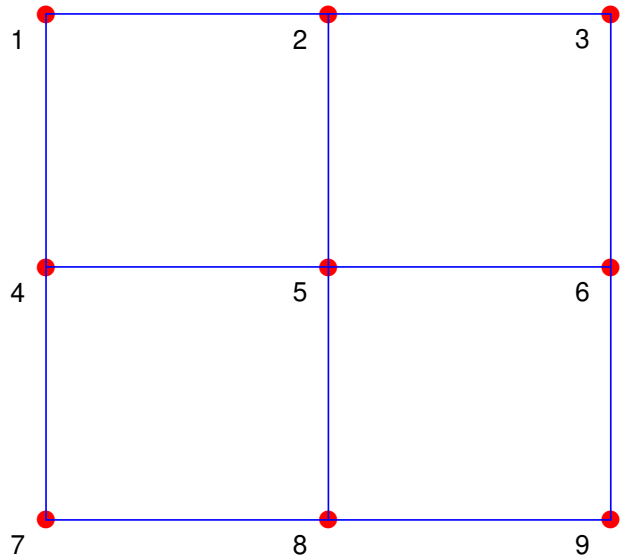
A graph is a pair of sets $G = (V, E)$, where:

- 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/polygons and edges the neighbour relations that may exist between different points/polygons.

A very simple graph

Here is the graph for a 3×3 grid of cells, with $n = 9$ 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 48 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 48:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{20}$$

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 48, 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 ($w_{ij} = w_{ji}, \ \forall\, i, j \quad \Leftrightarrow \quad \mathbf{W}^t = \mathbf{W}$).

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 48:

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

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 3×3 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 3×3 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:

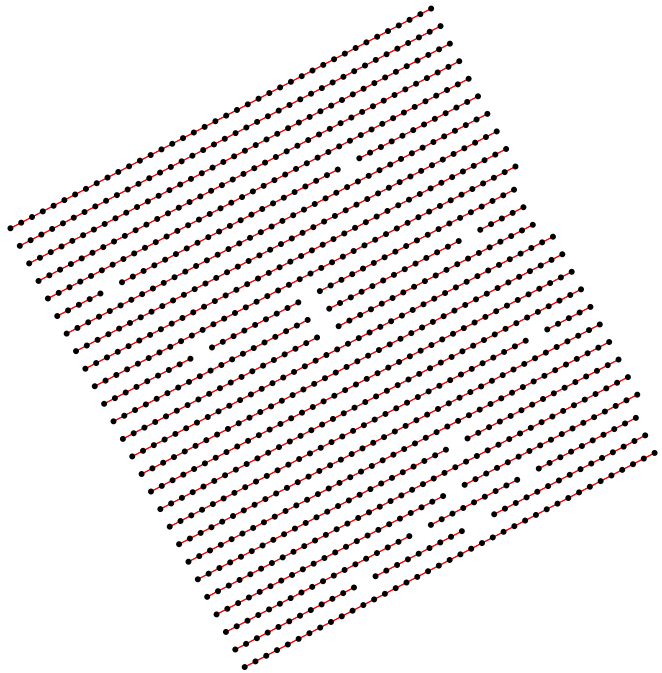
```
> dnearneigh(AragonezPoints, d1=0, d2=3)

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 ($d2=3$)

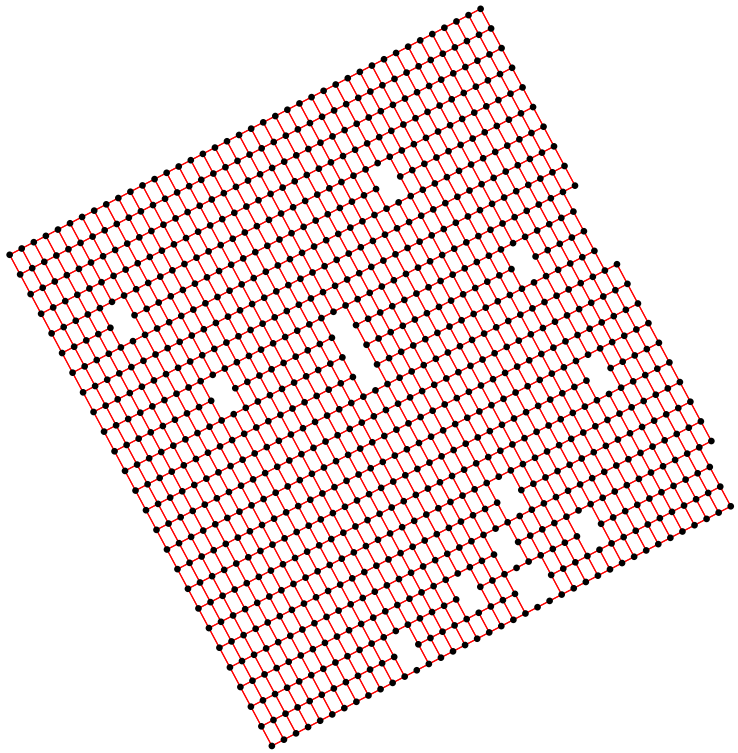
This is the resulting graph. Here the (same) neighbours were calculated from the `Aragonez3763` `sf` object:

```
> plot(dnearneigh(Aragonez3763, d1=0, d2=3),
+              coords=st_geometry(Aragonez3763), col="red", pch=16)
```



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 $4.5m$).

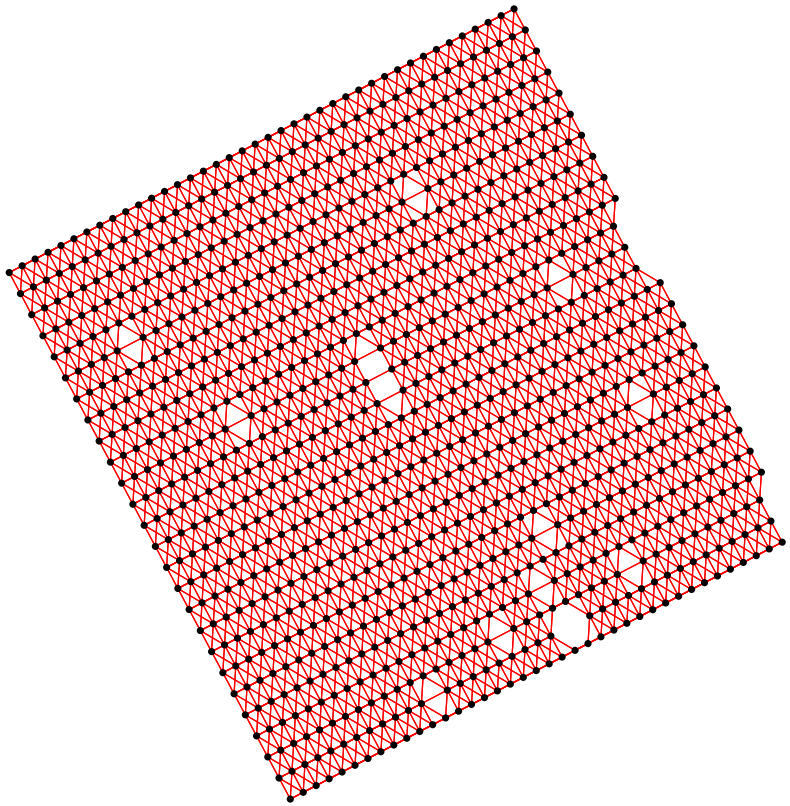
```
> dnearneigh(AragonezPoints, d1=0, d2=5)

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),
+ coord=coordinates(AragonezPoints), col="red")
```



knearneigh

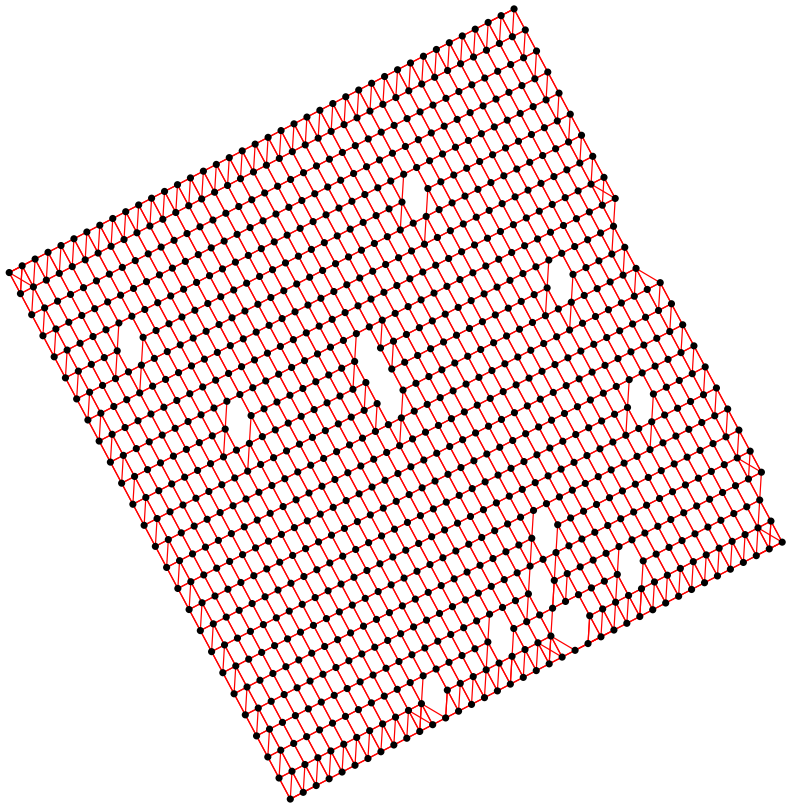
spdep::knearneigh searches k-nearest neighbours in a set of point coordinates. Output is of class knn (class package), not nb. The spdep knn2nb function converts knn objects to nb objects.

```
> knn2nb(knearneigh(Aragonez3763, 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(Aragonez3763, k=4)),
+                 st_geometry(Aragonez3763), col="red", pch=16)
```



poly2nb

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

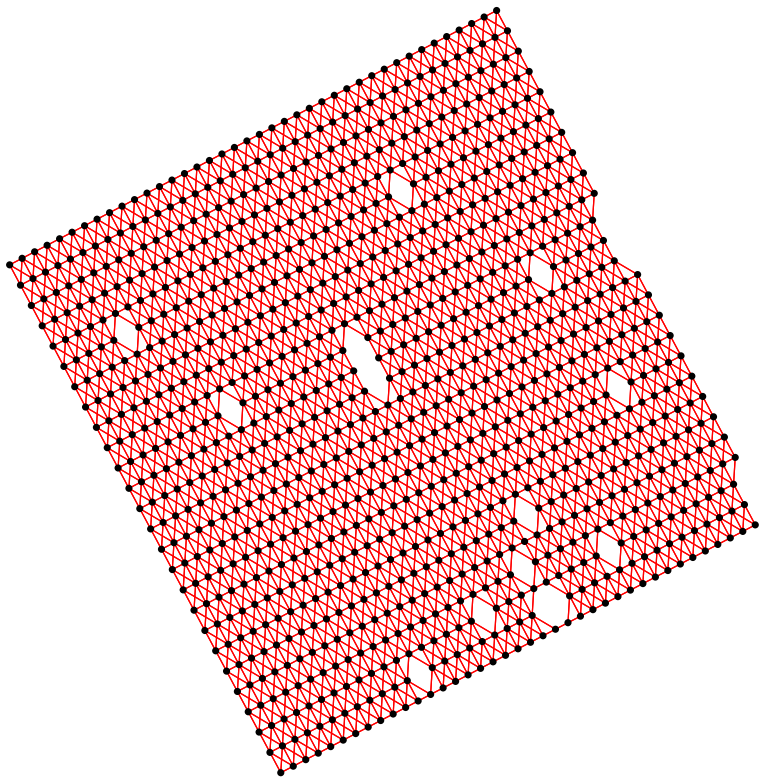
```
> poly2nb(Aragonez3763Grid)
```

```
Neighbour list object:
Number of regions: 1019
Number of nonzero links: 7646
Percentage nonzero weights: 0.7363528
Average number of links: 7.503435
```

For Aragonez3763Grid, on average, each grid polygon has almost 8 neighbours, as expected for the queen's case in a rectangular grid.

poly2nb

```
> plot(poly2nb(Aragonez3763Grid),
+      coords=st_geometry(Aragonez3763), col="red", pch=16)
```



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 3×3 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 3×3 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:

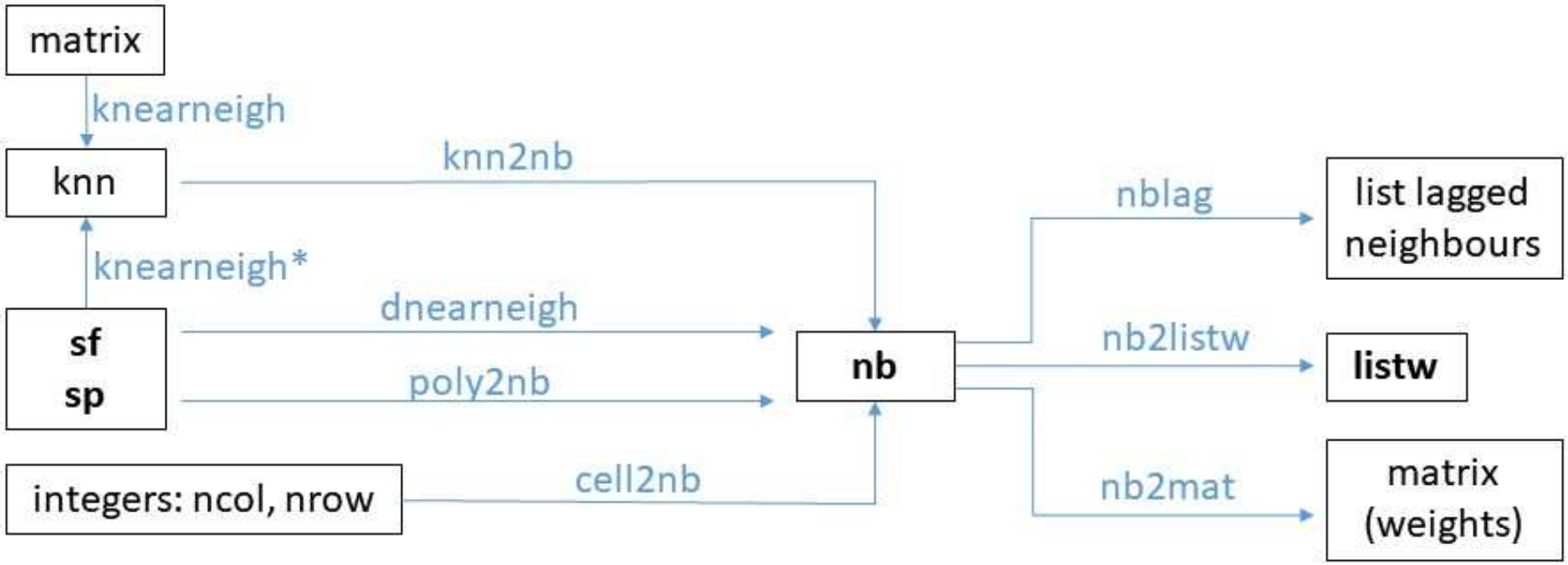
```
> str(nb2listw(cell2nb(3,3)))

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(Aragonez3763Grid), 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
```

Summary of spdep functions for neighbours and weights



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 40.

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

The 2-D autocorrelated model in vector notation

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

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

We have an alternative model formulation:

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

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

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

So the spatial autocorrelation model under consideration becomes:

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

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:

$$E[\vec{\mathbf{a}} + \mathbf{B}\vec{\mathbf{X}}] = \vec{\mathbf{a}} + \mathbf{B}\, E[\vec{\mathbf{X}}] \tag{24}$$

$$V[\vec{\mathbf{a}} + \mathbf{B}\vec{\mathbf{X}}] = \mathbf{B}\, V[\vec{\mathbf{X}}]\, \mathbf{B}^t \tag{25}$$

These properties give:

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

Therefore, under model (23), we have:

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

The distribution of $\overline{Z}$

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

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

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

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

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

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

The distribution of $\overline{Z}$

Since Normality is also preserved, we have:

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

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

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

Specific formulas for $V[\overline{Z}]$ depend on both λ 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 $(Z_1, Z_2, ..., Z_n)$.

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:

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

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

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:

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

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:

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

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

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. Both indicators have asymptotic Normal distribution, given H_0 (independence). But the variance ($V[I]$ or $V[c]$) can be computed in one of two ways:

- The standard assumption of a random sample: every new sample of size n will be a set of different values.
- The randomisation assumption: conditional on the observed values, we assume that the locations at which they observed are randomised.

The test statistics are $\frac{I-E[I]}{\sqrt{V[I]}}$ or $\frac{E[c]-c}{\sqrt{V[c]}}$, with asymptotic $\mathcal{N}(0,1)$ distribution, under H_0.

In both cases, large values of the statistic suggest rejection of H_0.

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, assuming asymptotic Normality. H_0 is the absence of spatial autocorrelation.

By default, both the `moran.test` and the `geary.test` functions will compute the variance assuming the randomisation option.

If the `randomisation` argument is set to the logical value `FALSE`, the variance is computed under the standard random sample assumption.

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 yields

We consider the 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$yield, listw=Wd3)

        Moran I test under randomisation

data:  AragonezPoints$yield
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 highly significant, indicating spatial autocorrelation.

But undetected spatial trends may be confused with spatial autocorrelation.

Spatial autocorrelation for Aragonez yields (cont.)

The test based on the alternative expression for $V[I]$ does not produce major differences.

Note: the values of I and $E[I]$ do not change with the type of test used.

```
> moran.test(AragonezPoints$yield, listw=Wd3, randomisation=FALSE)


Moran I test under normality

data:  AragonezPoints$yield
weights: Wd3

Moran I statistic standard deviate = 10.918, p-value < 2.2e-16
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
     0.3506489981     -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$yield, listw=Wd3)

        Geary C test under randomisation

data:  AragonezPoints$yield
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 yields, we must assume spatial autocorrelation.

Testing for spatial autocorrelation (cont.)

For smaller samples, where asymptotic Normality is doubtful, `moran.mc` carries out a permutation test on *I*.

The value of Moran's *I* 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 quantile of *I* or should not be extreme. If it is, this suggests the existence of spatial autocorrelation.

Spatial autocorrelation for Aragonez centred yields (cont.)

The permutation tests for the same setting:

```
> moran.mc(AragonezPoints$yield, listw=Wd3, nsim=10000)

Monte-Carlo simulation of Moran I

data:  AragonezPoints$yield
weights: Wd3
number of simulations + 1: 10001

statistic = 0.35065, observed rank = 10001, p-value = 9.999e-05
alternative hypothesis: greater
```

The value of Moran's I is, of course, the same. But its significance is assessed in a different way: its value is the most extreme, for all 10 001 permutations. Its empirical *p-value* is therefore $p = \frac{1}{10001} = 9.999 \times 10^{-5}$.

Testing the Aragonez linearly detrended yields

When examining residuals of a linear regression (such as `yielddt`), independence cannot be assumed (by design) as H_0.

The `lm.morantest` function should be used instead of `moran.test`.

The input argument must be an `lm` object, resulting from a linear regression, such as `Arag.lm` (slide 34), whose residuals are `yielddt`.

```
> lm.morantest(Arag.lm, listw=Wd3)

Global Moran I for regression residuals

data:
model: lm(formula = yield ~ rowm + colm, data = AragonezPoints)
weights: Wd3

Moran I statistic standard deviate = 6.1735, p-value = 3.34e-10
alternative hypothesis: greater
sample estimates:
Observed Moran I      Expectation         Variance
     0.195881656     -0.002945635      0.001037275
```

Moran's I is now noticeably smaller than for `yield`, but there is still strong indication of spatial autocorrelation ($p = 3.34 \times 10^{-10}$).

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 nblag

```
> nb.k4 <- knn2nb(knearneigh(AragonezPoints, k=4))
> nblag(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 `spdep::sp.correlogram` 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

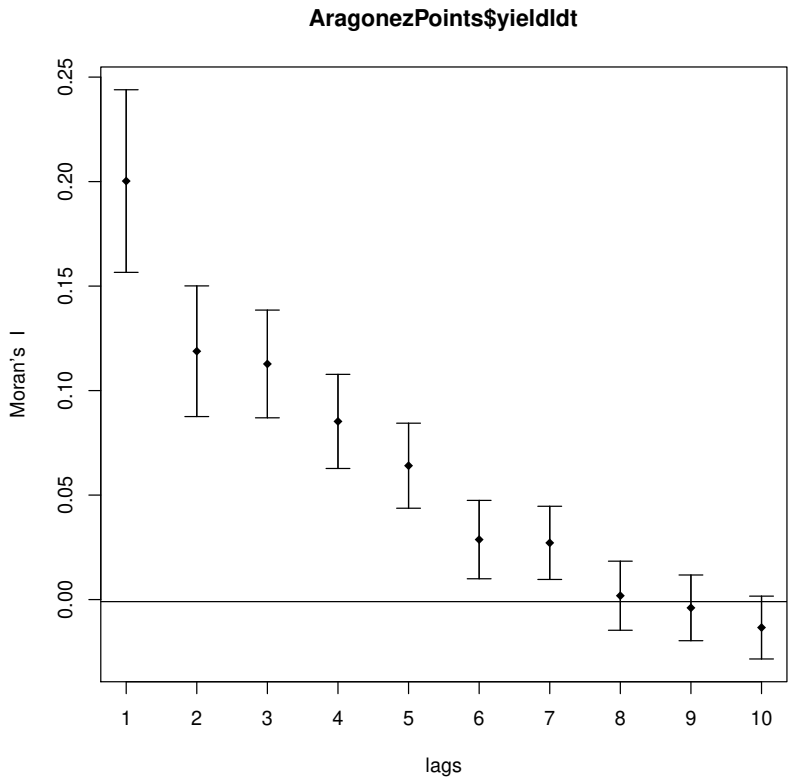


Figure: Moran's correlogram for the linearly detrended Aragonez yields, based on a $k = 4$ nearest neighbours list and a row-normalized weight matrix, with lags of up to 10. There is evidence of spatial autocorrelation, at least up to neighbours of lag $k = 5$ or $k = 7$.

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 μ_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 $Z(s_2)$:

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

- Spatial lag $\vec{\mathbf{d}} = s_1 - s_2$ 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 μ_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:

$$\begin{aligned} \mu_s &= \mu\,, \quad \forall\, s \in \mathcal{S} \qquad ; \qquad \text{and} \\ C(s_1, s_2) &= C_\ell(s_1 - s_2)\,, \qquad \forall\, s_1, s_2 \in \mathcal{S}. \end{aligned}$$

- 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 $\vec{\mathbf{d}} = s_1 - s_2$.

(Semi-)Variogram

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

- variogram is the function

$$2\gamma(s_1, s_2) \;=\; Var[Z(s_1) - Z(s_2)] \; . \tag{32}$$

- semi-variogram is the function

$$\gamma(s_1, s_2) \;=\; \frac{1}{2}\, Var[Z(s_1) - Z(s_2)] \; . \tag{33}$$

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

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

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

Semi-variogram

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

$$\gamma_\ell(\vec{\mathbf{d}}) \;=\; C_\ell(\vec{\mathbf{0}}) - C_\ell(\vec{\mathbf{d}}) \;.$$

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

$$\gamma_s(d) \;=\; C_s(0) - C_s(d) \;.$$

Properties of the semi-variogram

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

- nonnegative: $\gamma_s(d) \geq 0, \forall\, d$.
- $\gamma_s(0) = 0$.
- With no spatial autocorrelation, $C(d) = 0$, for $d \neq 0$, and so $\gamma_s(d) = C_s(0) = Var[Z(s)], \forall\, d \neq 0$.

  Without spatial autocorrelation, γ_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:

- $\lim_{d \to 0} \gamma_s(d) = c_0$ 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 $Z(s)$:

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

- the range of a spatial point s_1 is the value of d for which the semi-variogram is smaller than the sill, $\gamma(d) < C_s(0)$, 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 $Var[Z(s)]$ that is explained by the spatial autocorrelation process.

A typical semi-variogram

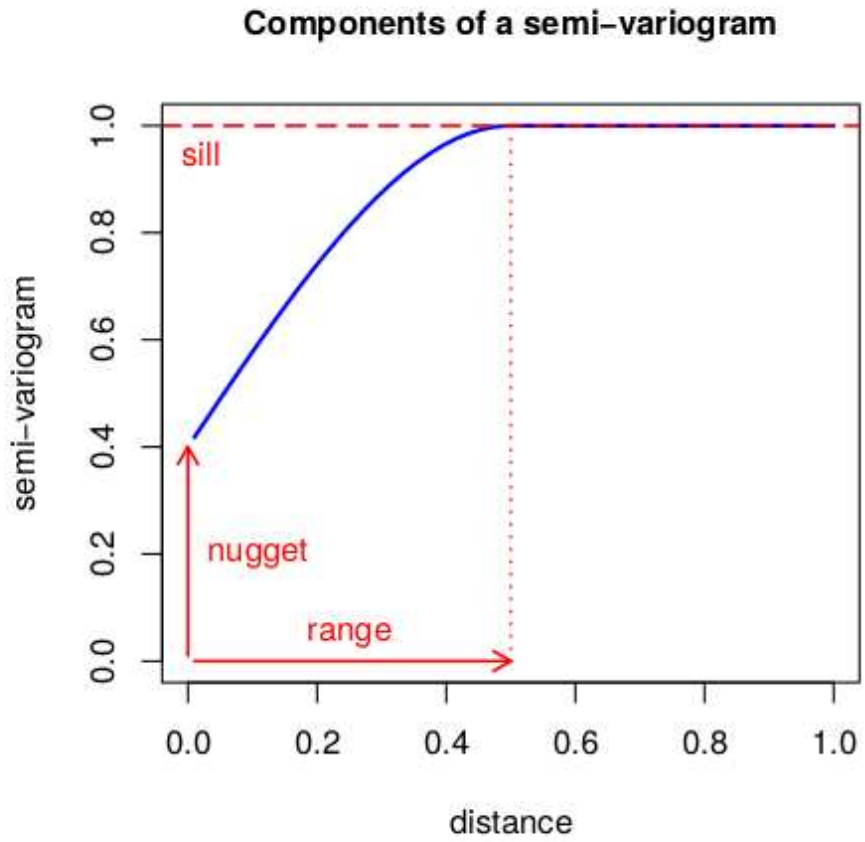


Figure: A typical variogram curve, with the nugget, sill and range highlighted.

Empirical semi-variograms

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

$$\hat{\gamma}(d) = \frac{1}{2} \frac{1}{|N(d)|} \sum_{i,j \in N(d)} (z(s_i) - z(s_j))^2 , \quad (34)$$

where, for any given distance $d = dist(s_1, s_2)$, $N(d)$ denotes the set of pairs of locations s_1, s_2 which are distance d, $|N(d)|$ 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, d is taken to be a small interval in order to ensure the existence of enough pairs $N(d)$ of observations, for any given d.

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 (use as_Spatial to convert an sf 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
	np	dist	gamma	dir.hor	dir.ver	id
1	1944	3.026404	0.8617851	0	0	var1
2	6513	5.666938	0.9560390	0	0	var1
3	13187	9.613532	0.9693001	0	0	var1
4	14887	13.512151	1.0027140	0	0	var1
5	20259	17.441649	1.0266800	0	0	var1
6	20529	21.302718	1.0582606	0	0	var1
7	24687	25.061267	1.0702769	0	0	var1
8	28165	29.142116	1.1049442	0	0	var1
9	26756	33.097528	1.1325510	0	0	var1
10	28621	36.892389	1.1600034	0	0	var1
11	29117	40.763906	1.1997004	0	0	var1
12	29146	44.621526	1.2327257	0	0	var1
13	28860	48.412351	1.2727498	0	0	var1
14	29956	52.323021	1.3368361	0	0	var1
15	27872	56.240912	1.3874178	0	0	var1

Column dist gives the values of d; column gamma gives the corresponding estimated value of the semi-variogram value $\gamma(d)$, 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.

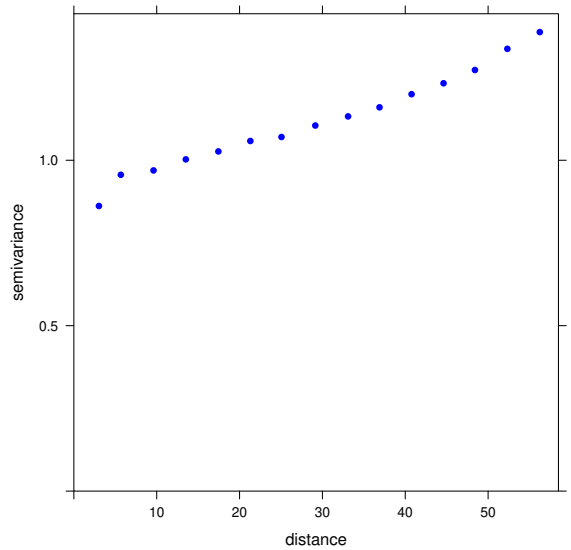


Figure: The empirical semi-variogram for the Aragonez yields, detrended by the (constant) mean, as produced by the `variogram` command in package `gstat`. The stabilization of the semi-variogram may have not been completed

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:

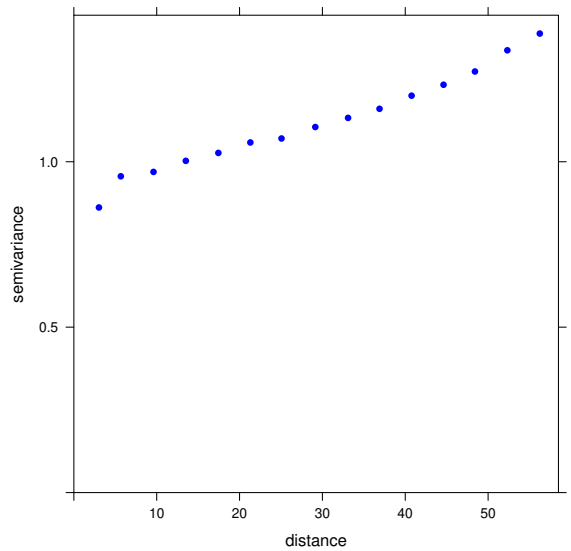
```
> variogram(yield ~ 1, data=AragonezPoints, cutoff=75)

      np      dist     gamma dir.hor dir.ver   id
1   4775  3.885159 0.9006839       0       0 var1
2  11662  8.016826 0.9640507       0       0 var1
3  17587 12.546920 0.9995182       0       0 var1
4  27227 17.609841 1.0286987       0       0 var1
5  27150 22.604166 1.0598021       0       0 var1
6  32791 27.481308 1.0921583       0       0 var1
7  35735 32.444811 1.1267466       0       0 var1
8  39019 37.578379 1.1697366       0       0 var1
9  33814 42.471293 1.2151499       0       0 var1
10 42088 47.444342 1.2590447       0       0 var1
11 34834 52.536738 1.3352649       0       0 var1
12 36149 57.389630 1.4167549       0       0 var1
13 35044 62.448067 1.4409542       0       0 var1
14 31021 67.475473 1.5638226       0       0 var1
15 26936 72.392091 1.6330283       0       0 var1
```

Increasing the `cutoff` argument has its limits: as d grows, values of γ will be estimated with a smaller numbers of points, becoming prone to erratic behaviour.

Extending the cutoff

The figure gives the new empirical semi-variogram, which continues to grow. This suggests a non-stationary variance in the process or an inappropriately removed underlying trend.

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



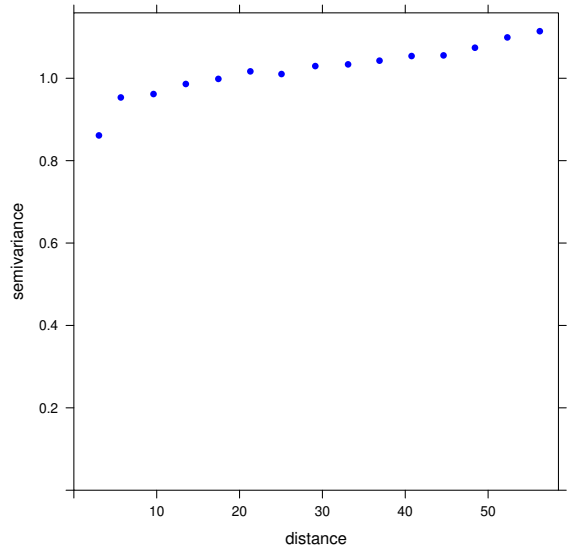
Variogram for the linearly detrended yields

Detrending with a linear regression on column and row distances (variables `colm` and `rowm`, respectively, in the `AragonezPoints` object) can be given directly 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   1944  3.026404 0.8612585       0       0 var1
2   6513  5.666938 0.9533967       0       0 var1
3  13187  9.613532 0.9616996       0       0 var1
4  14887 13.512151 0.9861200       0       0 var1
5  20259 17.441649 0.9984865       0       0 var1
6  20529 21.302718 1.0167690       0       0 var1
7  24687 25.061267 1.0102689       0       0 var1
8  28165 29.142116 1.0295995       0       0 var1
9  26756 33.097528 1.0337417       0       0 var1
10 28621 36.892389 1.0427308       0       0 var1
11 29117 40.763906 1.0539613       0       0 var1
12 29146 44.621526 1.0555425       0       0 var1
13 28860 48.412351 1.0741435       0       0 var1
14 29956 52.323021 1.0991958       0       0 var1
15 27872 56.240912 1.1142427       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]   4.977219  14.931657  24.886094  34.840532  44.794970  54.749408
 [7]  64.703845  74.658283  84.612721  94.567159 104.521596 114.476034
[13] 124.430472

$v
 [1] 0.9431928 0.9913348 1.0151768 1.0393587 1.0617242 1.1096586 1.1409804
 [8] 1.1868190 1.2239065 1.2387683 1.2562398 1.2209068 1.0539211

$n
 [1] 15572 44191 60139 73986 75903 71073 66636 51950 34997 16787  5712  1553
[13]   172
[...]
```

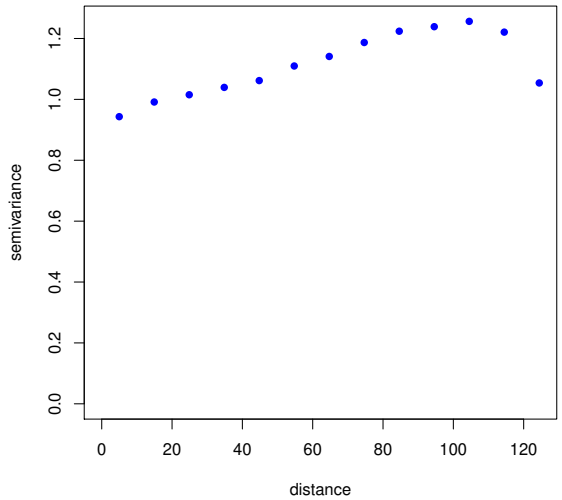
u gives the lags d; v the values $\gamma(d)$; n the points used to estimate $\gamma(h)$.

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

Plotting the variog results

Unlike variogram, the variog function uses, by default, all the spatial lags d (as center points in intervals, or bins) for which it finds pairs of points. A max.dist argument controls the maximum distance d 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 $d > 0$, the semi-variogram is given by:

$$\gamma(d) \;=\; c_0 + p\left[1 - \mathrm{e}^{-\frac{d}{r}}\right] \;,$$

  c_0 is the nugget, r the range and p the partial sill. The function grows to an asymptotic sill (given by $c_0 + p$), which is not attained.

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

$$\gamma(d) \;=\; c_0 + p\left[\frac{3}{2}\frac{d}{r} - \frac{1}{2}\left(\frac{d}{r}\right)^3\right] \quad \text{for } d < r \;,$$

  with $\gamma(d) = sill = c_0 + p$ for $d > r$.
  The model assumes that for $d > r$ there ceases to be spatial dependence and thereafter the semi-variogram $\gamma(d)$ is constant.

Variogram models

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

$$\gamma(d) \; = \; c_0 + p \left[1 - \mathrm{e}^{-\frac{d^2}{r^2}} \right] \; ,$$

  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

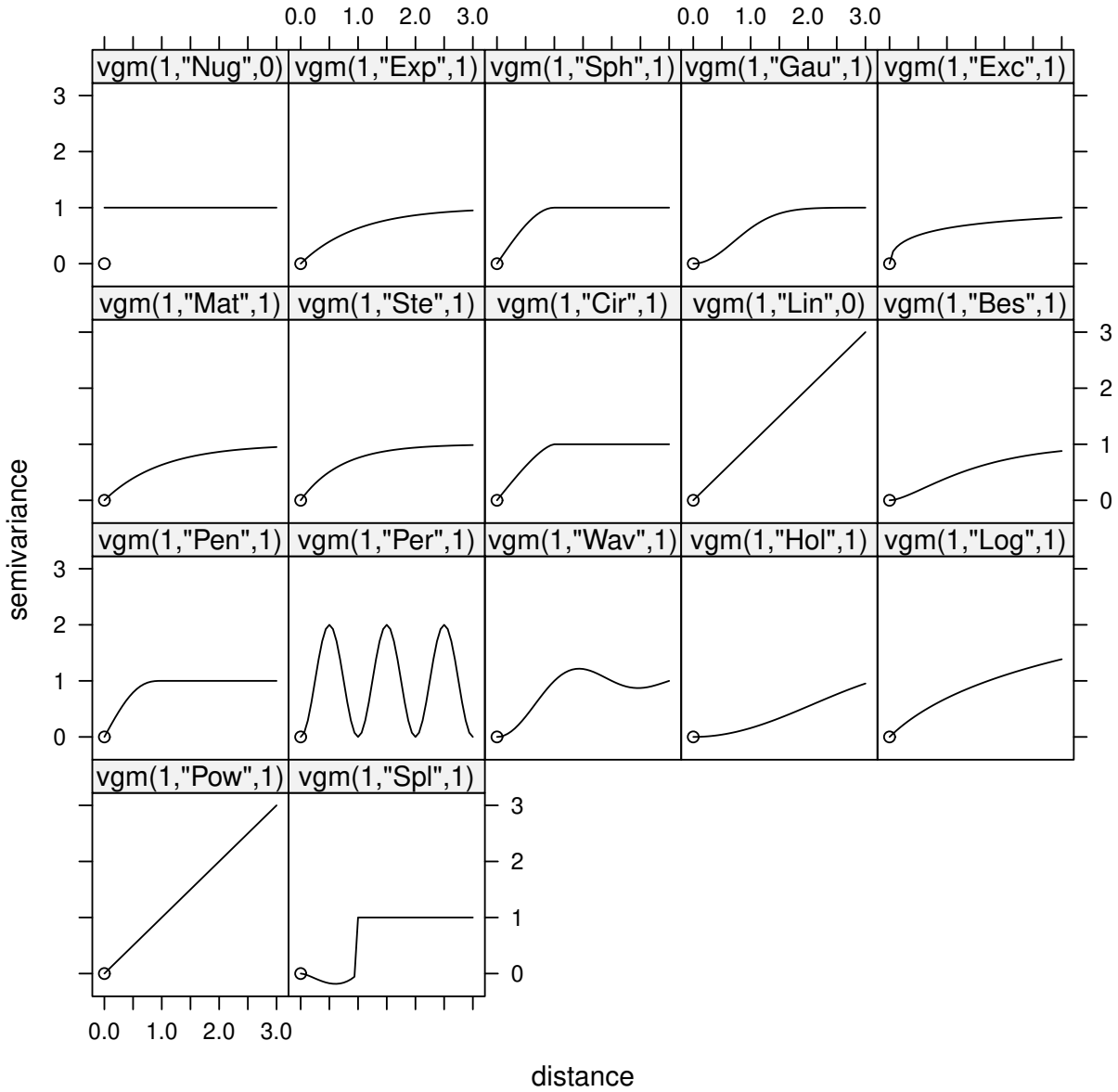
Model functions are specified with the vgm command in gstat, which requires as arguments:

- psill: an initial estimate of the partial sill (the difference between the sill and the nugget);
- nugget: an initial estimate of the nugget effect c_0;
- 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.

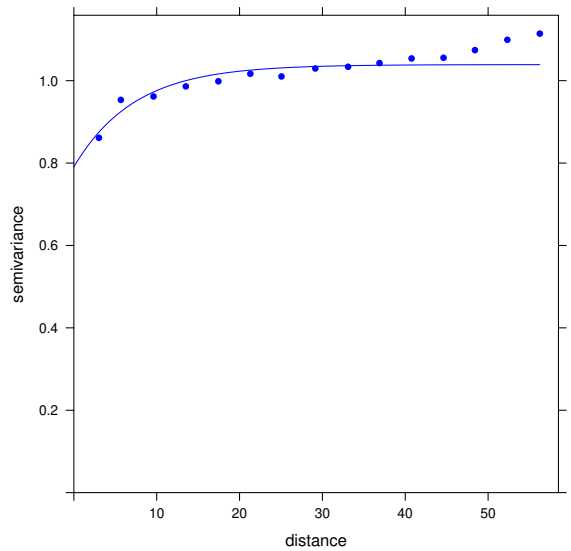
An exponential model fitted to the linearly detrended Aragonez yields:

```
> 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 0.7906716 0.000000
2   Exp 0.2483298 7.297103
```

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

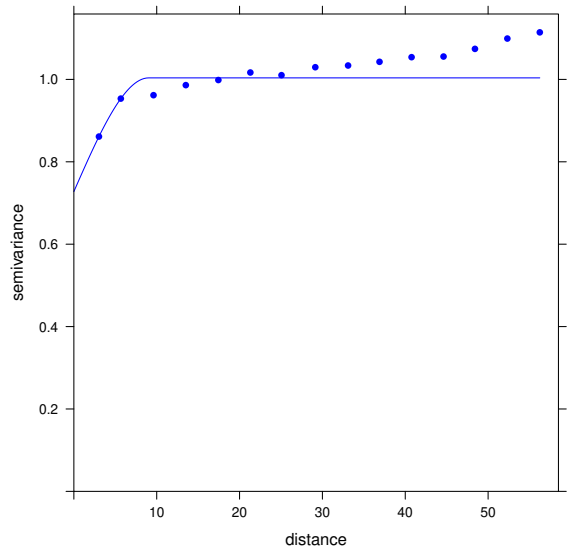
A spherical model fitted to the same empirical variogram:

```
> m2.fit <- fit.variogram(AragVarioLin,
+          model=vgm(psill=0.3,"Sph", range=7, nugget=0.8))

> m2.fit
  model     psill    range
1   Nug 0.7277332 0.000000
2   Sph 0.2759964 9.032198
```

While both models provide similar estimates for the partial sill and nugget, the estimates for the range differ a bit more. The exponential model is better suited in this case, since unlike the spherical model, it does not assume a constant $\gamma(d)$ 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.

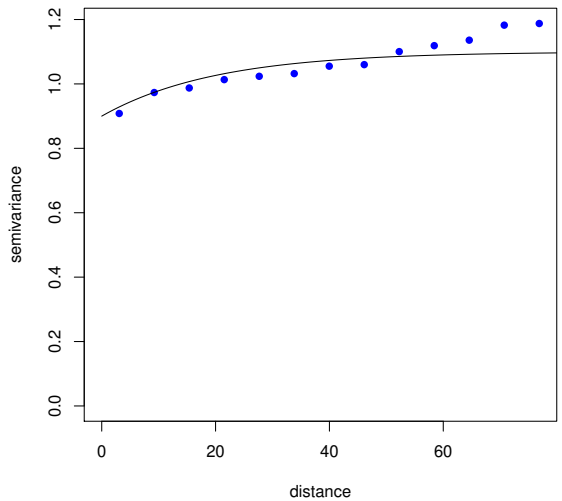
The lines.variomodel function arguments are:

- 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 $d = 80$), 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.

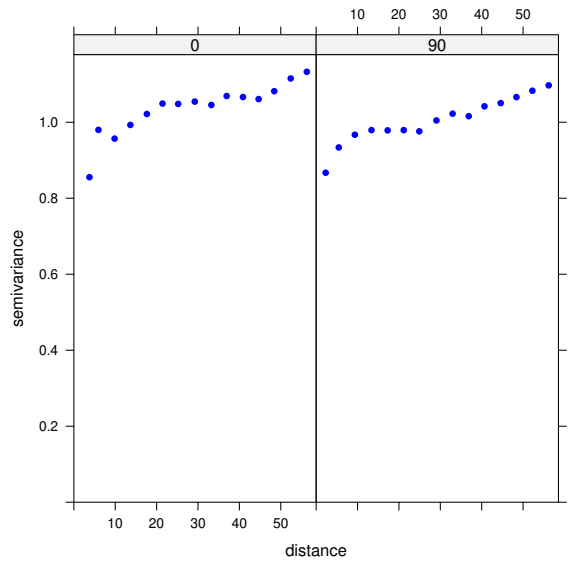
```
> variogram(yield ~ colm + rowm , data=AragonezPoints, alpha=c(0,90))
```

```
      np      dist     gamma dir.hor dir.ver   id
1    965  3.761890 0.8553530       0       0 var1
2   2782  5.937408 0.9799105       0       0 var1
3   7021  9.841740 0.9568937       0       0 var1
[...]
13 14835 48.420389 1.0816942       0       0 var1
14 15112 52.398543 1.1151889       0       0 var1
15 13430 56.296716 1.1327193       0       0 var1
16   979  2.301436 0.8670795      90       0 var1
17  3731  5.465264 0.9336268      90       0 var1
18  6166  9.353679 0.9671719      90       0 var1
[...]
29 14844 52.246134 1.0829139      90       0 var1
30 14442 56.189018 1.0970608      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:

$$\rho(d) = \frac{Cov[Z(s), Z(s+d)]}{\sqrt{Var[Z(s)]\, Var[Z(s+d)]}} = \frac{C_s(d)}{C_s(0)} . \tag{35}$$

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

$$\begin{aligned} & \gamma(d) = C_s(0) - C_s(d) = C_s(0) \left[1 - \frac{C_s(d)}{C_s(0)}\right] \\ \Leftrightarrow \quad & \gamma_s(d) = C_s(0)\,[1 - \rho(d)]. \end{aligned} \tag{36}$$

The intuitively obvious relation $\lim_{d \to +\infty} \rho(d) = 0$ is coherent with the idea that the sill is the asymptotic value of the semi-variogram as d 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)[1]. The data are reanalysis data (pre-processed in this case by the ERA-Interim data assimiliation 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)

[1] `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

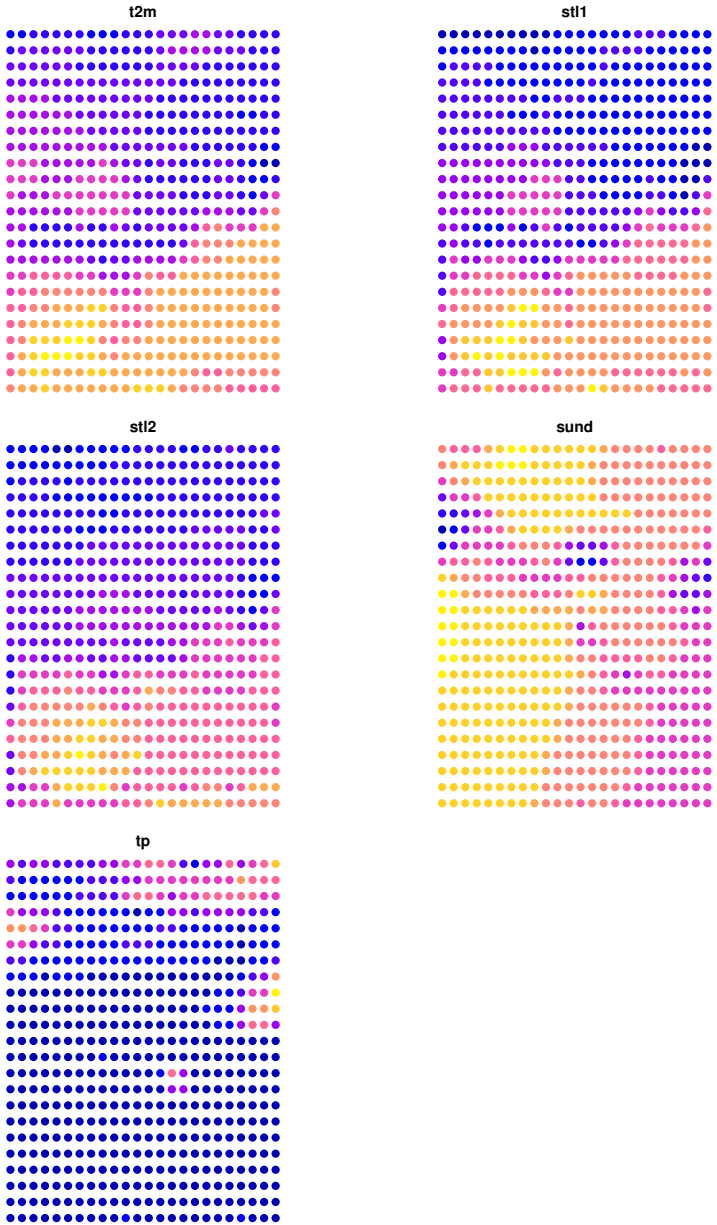
As before, we build objects of class sf and SpatialPointsDataFrame.

```
> meteo.sf <- st_as_sf(meteo, coords=c("lon","lat"), crs=4326)
> meteo.sp <- as_Spatial(meteo.sf)
```

The plot method for sf objects is used to preview the variables.

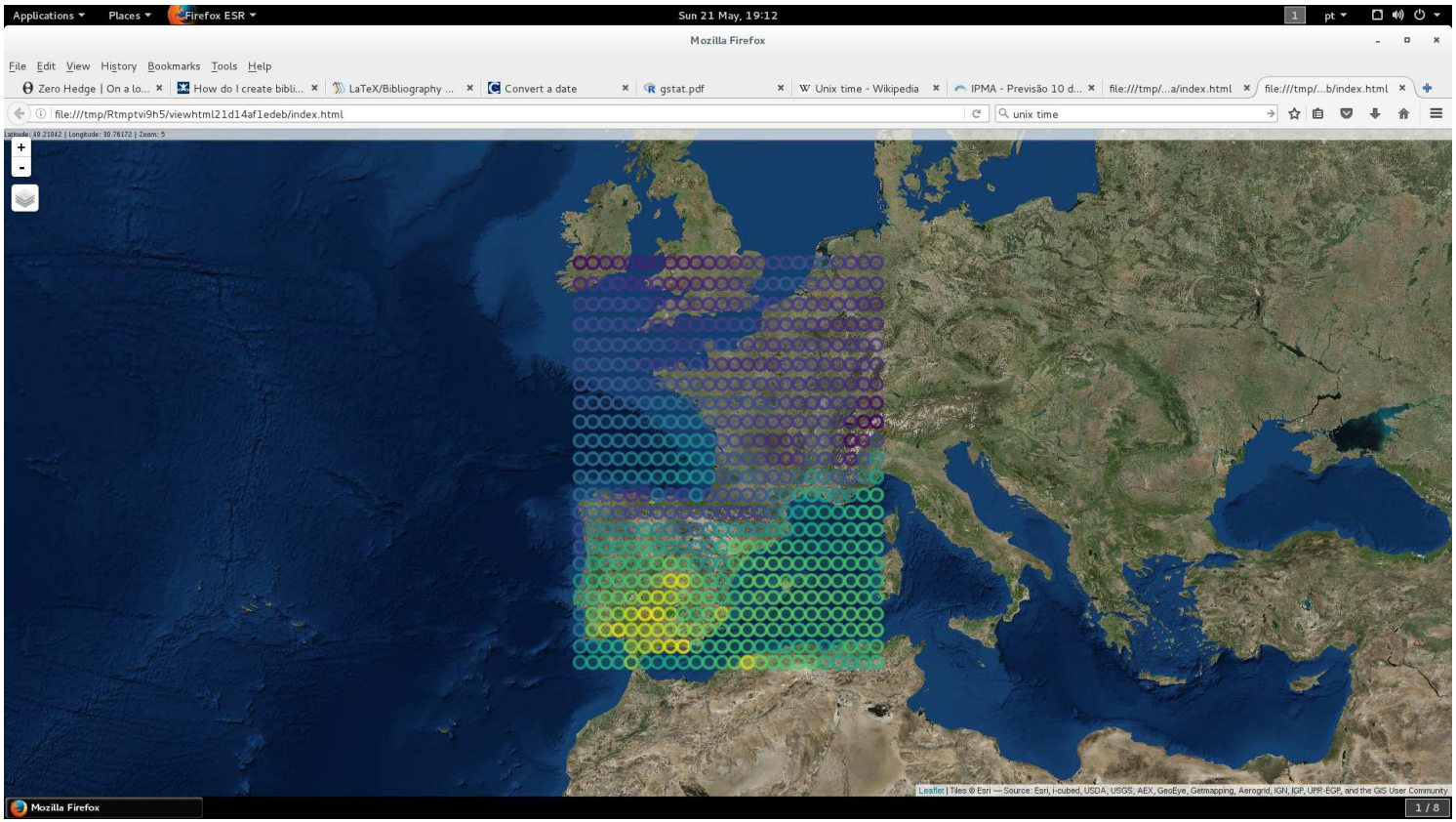
```
> plot(meteo.sf, pch=16)
```

N-S temperature gradient



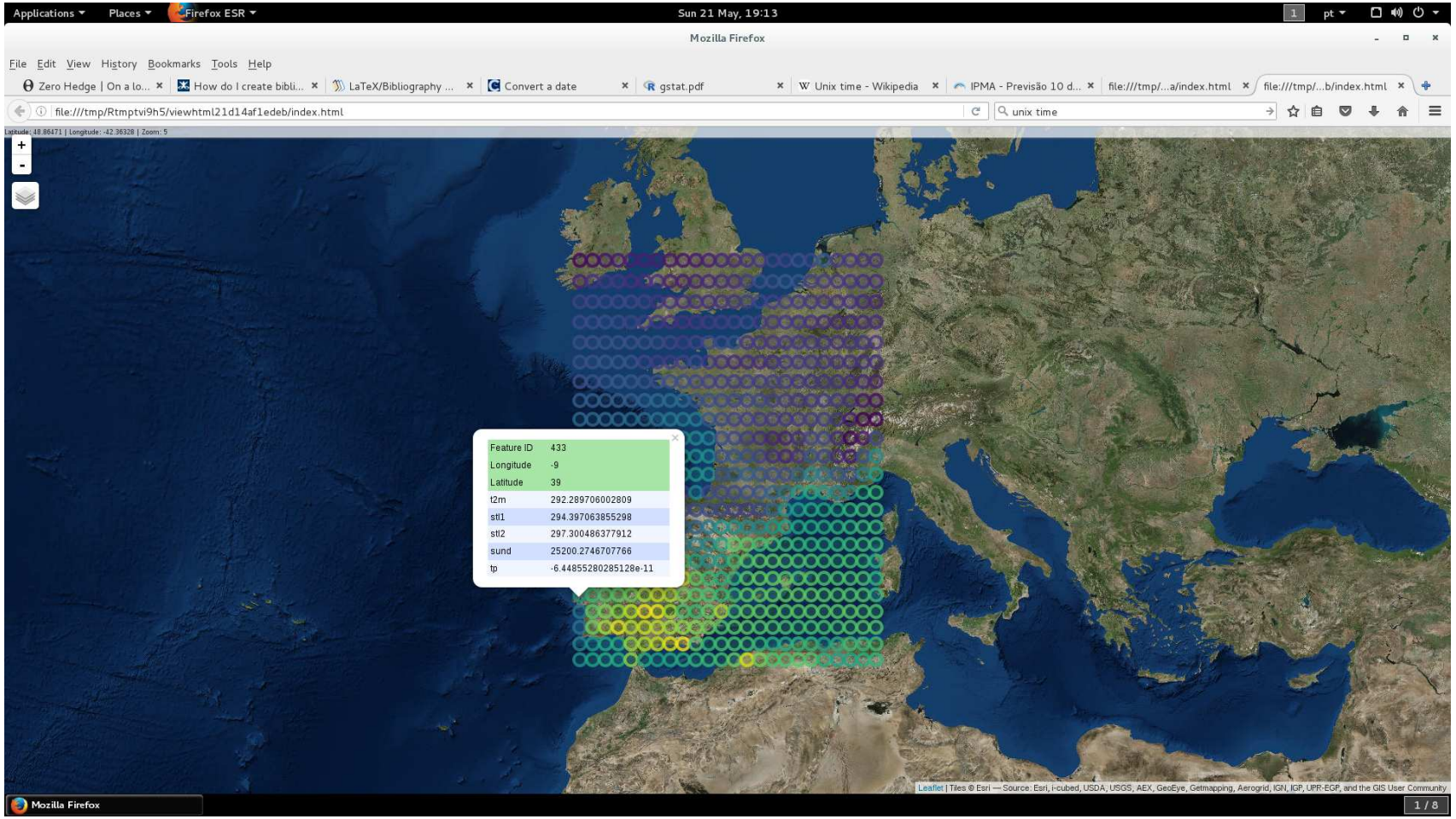
Meteorological data with mapview

```
> library(mapview)
> mapView(meteo.sf, 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

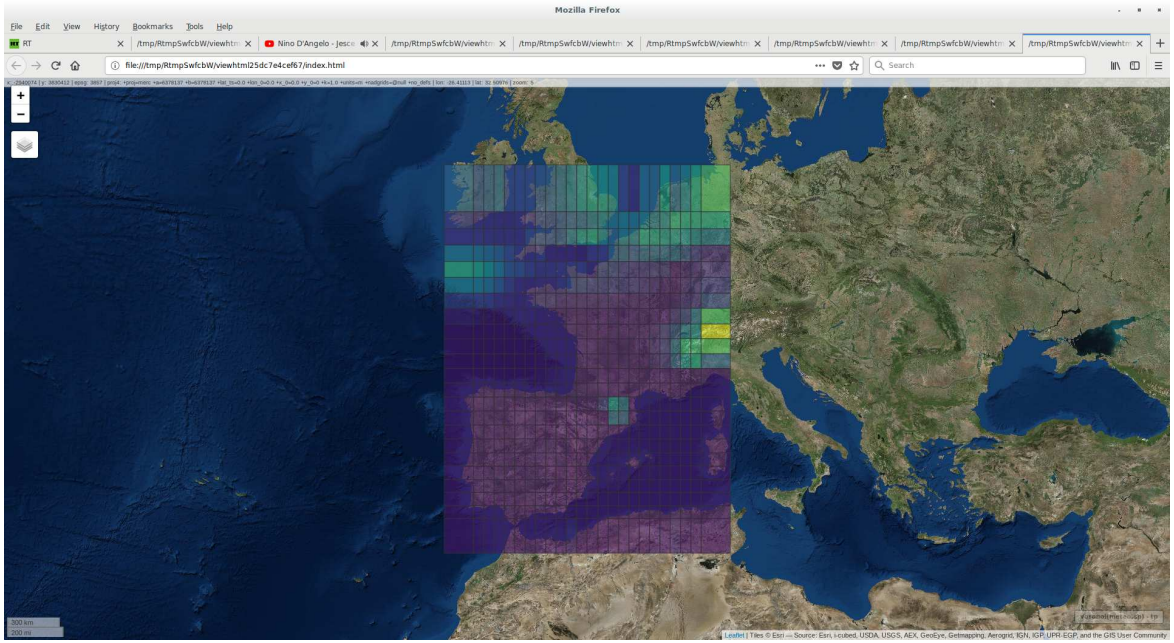


Clicking on any of the circles will open a window with the information for that location.

Several variables

The `tp` (total precipitation) variable can be seen after creating Voronoi polygons using the `voronoi` function from package `dismo`:

```
> library(dismo)
> meteo.voronoi <- voronoi(meteo.sp)
> mapView(meteo.voronoi, zcol="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 on slide (102) as:

$$2\gamma_{ii}(d) \;=\; Var\left(Z_{[i]}(s) - Z_{[i]}(s+d)\right)$$

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

$$2\gamma_{ij}(d) \;=\; Cov\left[\left(Z_{[i]}(s) - Z_{[i]}(s+d)\right), \left(Z_{[j]}(s) - Z_{[j]}(s+d)\right)\right]$$

Cressie gives an alternative definition:

$$2\gamma_{ij}(d) \;=\; Var\left(Z_{[i]}(s) - Z_{[j]}(s+d)\right)$$

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 ~ coords.x1 + coords.x2, meteo.sp)
> gobj <- gstat(gobj, "stl1" , stl1 ~ coords.x1 + coords.x2, meteo.sp)
> gobj <- gstat(gobj, "stl2" , stl2 ~ coords.x1 + coords.x2, meteo.sp)
> gobj <- gstat(gobj, "sund" , sund ~ coords.x1 + coords.x2, meteo.sp)
> gobj <- gstat(gobj, "tp" , tp ~ coords.x1 + coords.x2, meteo.sp)
```

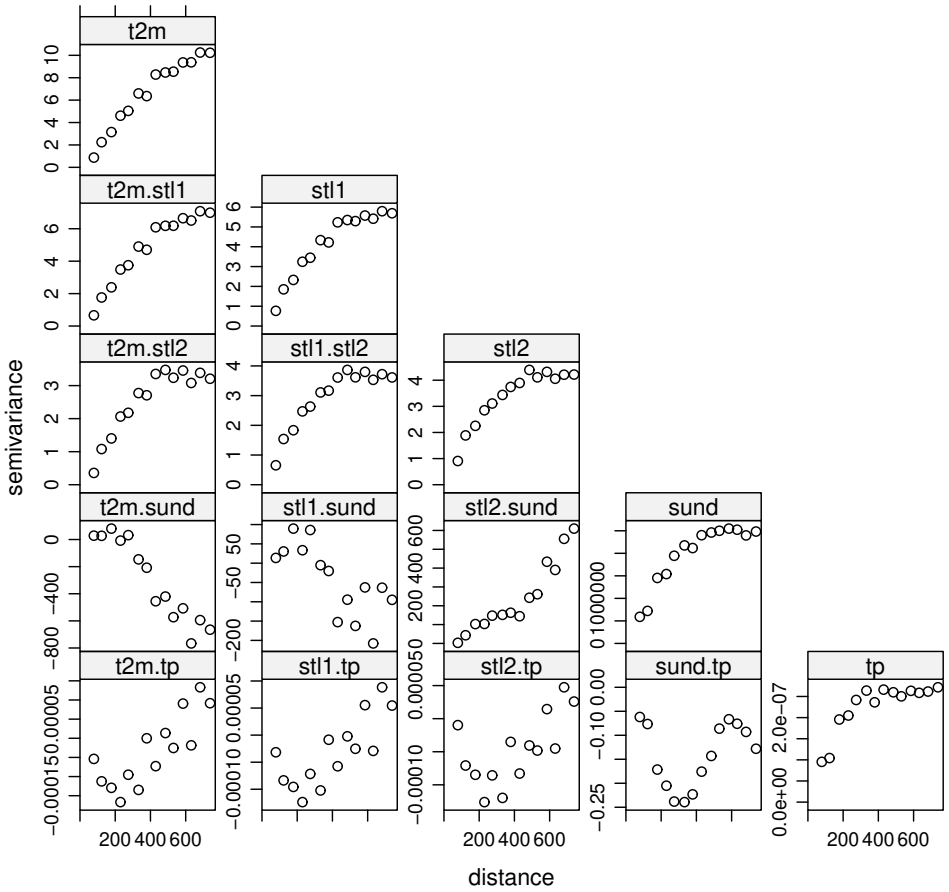
The variogram function for cross-variograms

```
> gobj
data:
t2m : formula = t2m ~ coords.x1 + coords.x2 ; data dim = 552 x 5
stl1 : formula = stl1 ~ coords.x1 + coords.x2 ; data dim = 552 x 5
stl2 : formula = stl2 ~ coords.x1 + coords.x2 ; data dim = 552 x 5
sund : formula = sund ~ coords.x1 + coords.x2 ; data dim = 552 x 5
tp : formula = tp ~ coords.x1 + coords.x2 ; 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.

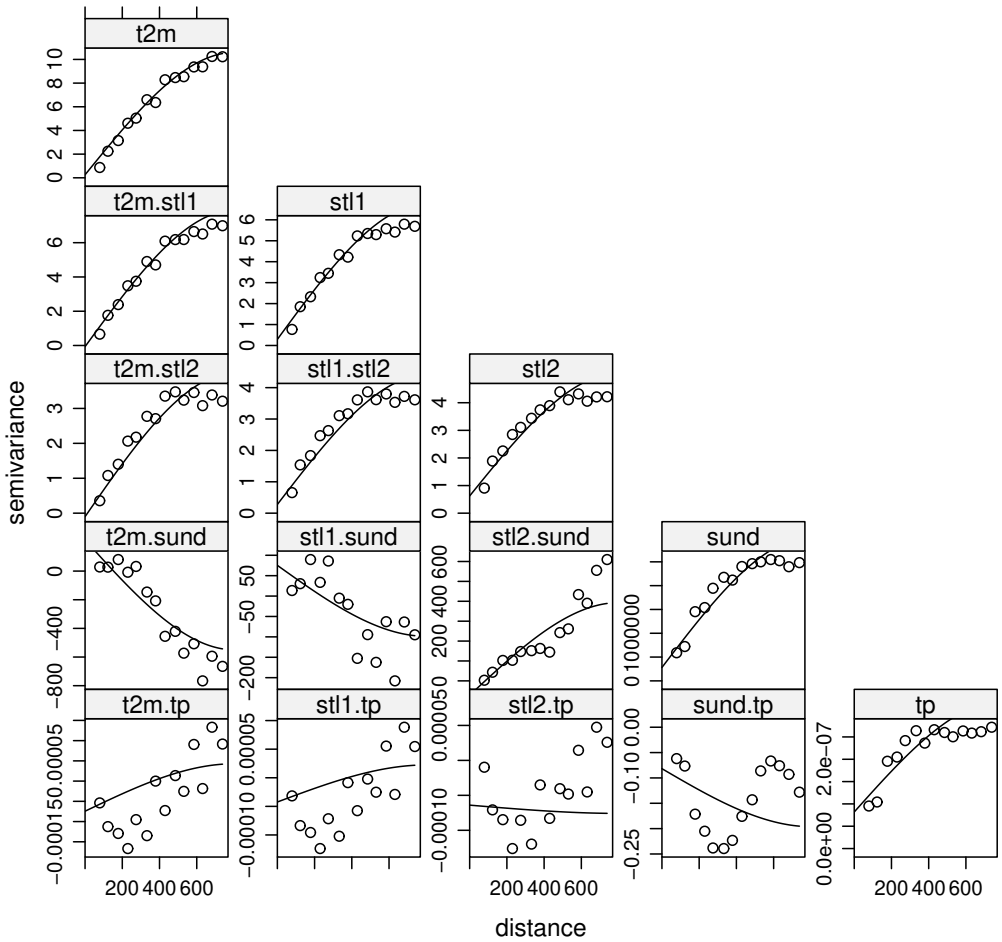
```
> vmeteo.fit <- fit.lmc(vario.meteo, gobj,
+              vgm(psill=1, "Sph", range=800, nugget=1))
```

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.