Lab 6.3: Introducing spatial data in R

In this Lab we will learn how to:

- Define spatial data
- Visualize spatial data

R 4.0.3 version has been used to run the following scripts. As usual we will start by defining our working directory and installing and loading the required packages.

```
# establishing the working directory

setwd("C:/datosR/GIS")

# installing different useful packages
install.packages(c("ggplot2", "devtools", "dplyr", "stringr"))

install.packages(c("maps", "mapdata", "ggmap", "mapproj"))

install.packages(c("sp", "raster", "rgdal", "rgeos", "spdep"))

# requiring the packages
library(ggplot2)
library(ggmap)
library(maps)
library(mapdata)
library(sp)
library(mapproj)

library(sp)  # vector data
library(raster)  # raster data
library(rgdal)  # input/output, projections
library(rgeos)  # geometry ops
library(spdep)  # spatial dependence
```

Types of spatial data in R

The basic spatial objects in R can be classified as points, lines or polygons. Each of these types must be addressed specifically in the different R packages when we want to deal with spatial data. Now we will concentrate in the use of the packages (already we have installed and required): sp, raster, rgdal, rgeos, spdep that deal with vector data, raster data, input and output, projections, geometry and spatial dependence.

- **Points:** *SpatialPoints* (ie, coordinates) and *SpatialPointsDataFrame* (ie, coordinates with data)
- **Lines:** *Line* (simple line strings), *SpatialLines* (lines with spatial nature), *SpatialLinesDataFrame* (spatial lines with data)
- **Polygons:** *Polygon* (rings or close figure), *SpatialPolygons* (polygons with spatial nature), *SpatialPolygonsDataFrame* (spatial polygons with data)

```
# Now we will start to work with spatial points

library(sp)
data(meuse)
 coords <- SpatialPoints(meuse[, c("x", "y")])
 summary(coords)
```

To obtain the following output (the data used are included in the sp package):

```
Object of class SpatialPoints
Coordinates:
        min    max
x    178605 181390
y    329714 333611
Is projected: NA
proj4string : [NA]
Number of points: 155
```

We can use the instruction coordnames( ) to know the name of the variables associated with the coordinates in the meuse dataset. As we can see there is no projection associated to this dataset (*Is projected: NA* and *proj4string: [NA]*) Later we'll see how to fix this.

```
# Now we will add the data frame to generate a SpatialPointsDataFrame
meuse1 <- SpatialPointsDataFrame(coords, meuse)
names(meuse1)
head(meuse1)
```

We will obtain the following:

```
[1] "x"      "y"      "cadmium" "copper"  "lead"    "zinc"    "elev"    "dist"    "om"      "ffreq"   "soil"
[12] "lime"    "landuse" "dist.m"
```

	x	y	cadmium	copper	lead	zinc	elev	dist	om	ffreq	soil	lime	landuse	dist.m
1	181072	333611	11.7	85	299	1022	7.909	0.00135803	13.6	1	1	1	Ah	50
2	181025	333558	8.6	81	277	1141	6.983	0.01222430	14.0	1	1	1	Ah	30
3	181165	333537	6.5	68	199	640	7.800	0.10302900	13.0	1	1	1	Ah	150
4	181298	333484	2.6	81	116	257	7.655	0.19009400	8.0	1	2	0	Ga	270
5	181307	333330	2.8	48	117	269	7.480	0.27709000	8.7	1	2	0	Ah	380
6	181390	333260	3.0	61	137	281	7.791	0.36406700	7.8	1	2	0	Ga	470

The line object is just a set of consecutive points joined by interpolation and can be defined by the sp package functions (see below) A polygon is just an area with a contour defined by a line with same initial and final point. With data from sp package we can delineated the river in the *meuse* dataset by using the following script:

```
# Generating a polygon
data(meuse.riv)
river_polygon <- Polygons(list(Polygon(meuse.riv)), ID = "meuse")

rivers <- SpatialPolygons(list(river_polygon))
summary(rivers)
```

There are a lot of different spatial classes provided by the package sp that allow building spatial objects:

data type	class	attributes	extends
points	SpatialPoints	none	Spatial
points	SpatialPointsDataFrame	data.frame	SpatialPoints
pixels	SpatialPixels	none	SpatialPoints
pixels	SpatialPixelsDataFrame	data.frame	SpatialPixels
			SpatialPointsDataFrame
full grid	SpatialGrid	none	SpatialPixels
full grid	SpatialGridDataFrame	data.frame	SpatialGrid
line	Line	none	
lines	Lines	none	Line list
lines	SpatialLines	none	Spatial, Lines list
lines	SpatialLinesDataFrame	data.frame	SpatialLines
polygon	Polygon	none	Line
polygons	Polygons	none	Polygon list
polygons	SpatialPolygons	none	Spatial, Polygons list
polygons	SpatialPolygonsDataFrame	data.frame	SpatialPolygons

Visualizing spatial data in R

We will start visualizing points from the meuse1 object we have defined previously.

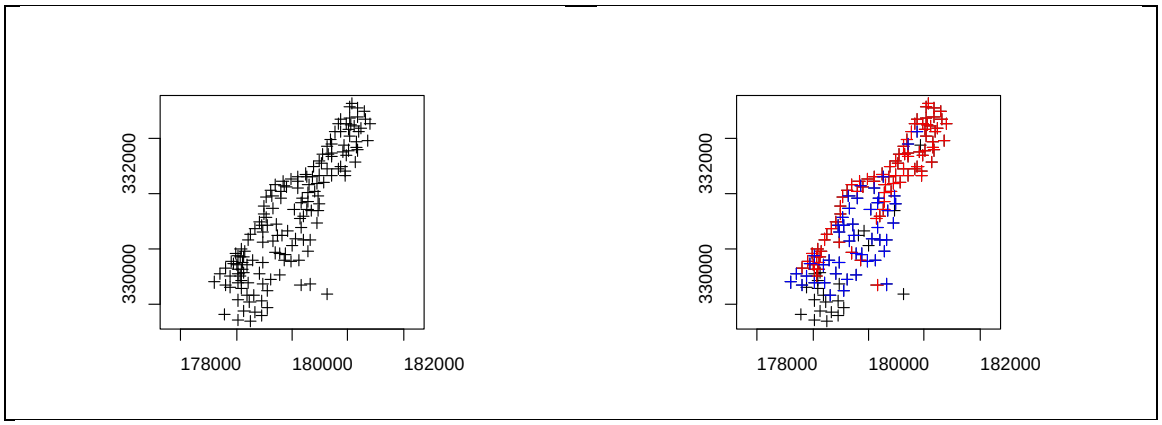
```
# Plotting the points generated in meuse1
plot(as(meuse1, "Spatial"), axes = TRUE)
plot(meuse1, add = TRUE)
```

see the left graph on the following box

```
# Now we can draw in different colors the previous graph for different ffreq values

plot(meuse1[meuse1$ffreq ==  1, ], col = "red", add = TRUE)
plot(meuse1[meuse1$ffreq ==  2, ], col = "blue", add = TRUE)
```

and obtain the graph on the right in the box (you can try to change the colors)



Now we are ready to plot a polygon. To do that we will use the object rivers we have defined previously by the function SpatialPolygons. We will use the following script:

```
# Plotting a polygon
plot(rivers, axes = TRUE, col = "grey", ylim = c(329400, 334000))
```

To obtain the plot on the left in the following box (see below). Now we are ready to overlap the points from the previous plot by coding as follow:

```
# Plotting a polygon and overlap the points
plot(rivers, axes = TRUE, col = "grey", ylim = c(329400, 334000))
+ plot(meuse1, add = TRUE) # pay attention to the + symbol
```

To obtain the figure on the right in the box.