Spatial data visualization

Lecture 5

Joaquin Cavieres

1. Introduction

Spatial data visualization is beautiful. Maps are amongst the most compelling graphics, because the space they map is the space we think we live in, and maps may show things we cannot see otherwise.

2. Practical session

2.1. Plotting spatial data

In the following example session, we create points, lines, polygons, and grid object, from `data.frame` objects, retrieved from the "sp" package by function `data`, and plot them.

```
library(sp)
data(meuse)
```

The meuse data set comprising of four heavy metals measured in the top soil in a flood plain along the river Meuse, along with a handful of covariates. So, what are the variables in this data set?

```
head(meuse, 6)
```

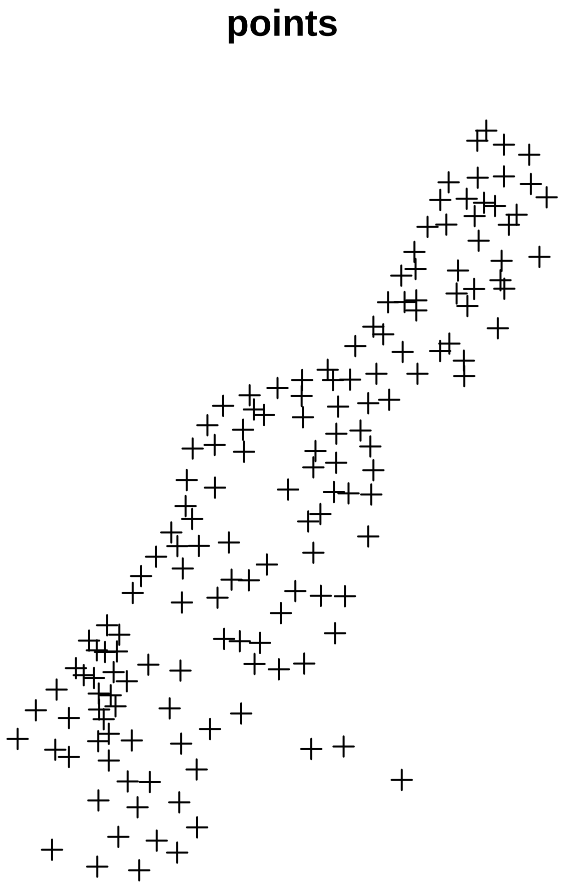
```
##        x      y cadmium copper lead zinc  elev       dist   om ffreq soil lime
## 1 181072 333611    11.7     85  299 1022 7.909 0.00135803 13.6     1    1    1
## 2 181025 333558     8.6     81  277 1141 6.983 0.01222430 14.0     1    1    1
## 3 181165 333537     6.5     68  199  640 7.800 0.10302900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

```
summary(meuse)
```

```
##        x                y             cadmium           copper
##  Min.   :178605   Min.   :329714   Min.   : 0.200   Min.   : 14.00
##  1st Qu.:179371   1st Qu.:330762   1st Qu.: 0.800   1st Qu.: 23.00
##  Median :179991   Median :331633   Median : 2.100   Median : 31.00
##  Mean   :180005   Mean   :331635   Mean   : 3.246   Mean   : 40.32
##  3rd Qu.:180630   3rd Qu.:332463   3rd Qu.: 3.850   3rd Qu.: 49.50
##  Max.   :181390   Max.   :333611   Max.   :18.100   Max.   :128.00
##
##       lead            zinc             elev             dist
##  Min.   : 37.0   Min.   : 113.0   Min.   : 5.180   Min.   :0.00000
##  1st Qu.: 72.5   1st Qu.: 198.0   1st Qu.: 7.546   1st Qu.:0.07569
##  Median :123.0   Median : 326.0   Median : 8.180   Median :0.21184
##  Mean   :153.4   Mean   : 469.7   Mean   : 8.165   Mean   :0.24002
##  3rd Qu.:207.0   3rd Qu.: 674.5   3rd Qu.: 8.955   3rd Qu.:0.36407
##  Max.   :654.0   Max.   :1839.0   Max.   :10.520   Max.   :0.88039
##
##        om         ffreq  soil   lime       landuse       dist.m
##  Min.   : 1.000   1:84   1:97   0:111   W      :50   Min.   :  10.0
##  1st Qu.: 5.300   2:48   2:46   1: 44   Ah     :39   1st Qu.:  80.0
##  Median : 6.900   3:23   3:12           Am     :22   Median : 270.0
##  Mean   : 7.478                         Fw     :10   Mean   : 290.3
##  3rd Qu.: 9.000                         Ab     : 8   3rd Qu.: 450.0
##  Max.   :17.000                         (Other):25   Max.   :1000.0
##  NA's   :2                              NA's   : 1
```

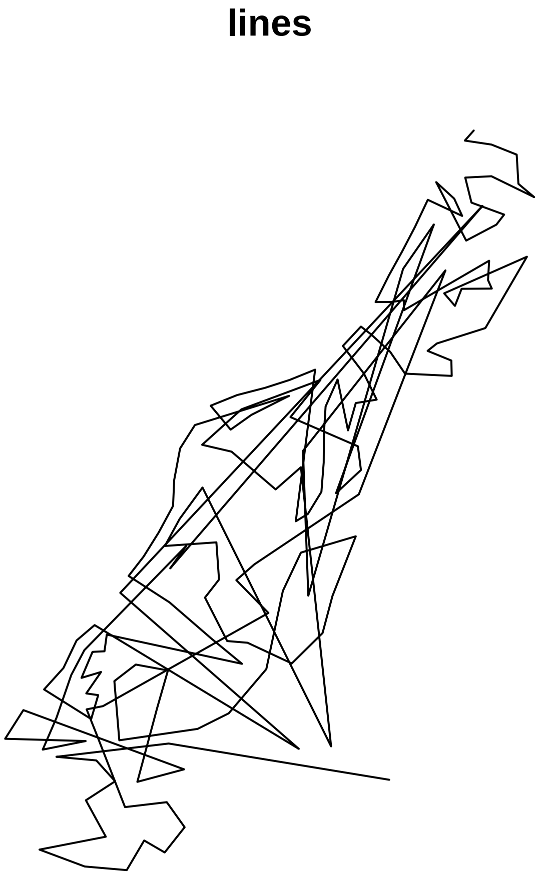
Now we can easily plot the “meuse” data as follows:

```
coordinates(meuse) <- c("x", "y")
plot(meuse)
title("points")
```



Here we have a `SpatialPointsDataFrame` object, and it was created from a `data.frame` provided with “sp” package. The `plot` function of the R base shows the points with the default symbol, but we can change this using other characteristic of the same function, for example:

```
loc <- coordinates(meuse)
line_sp <- SpatialLines(list(Lines(list(Line(loc)), "line1")))
plot(line_sp)
title("lines")
```

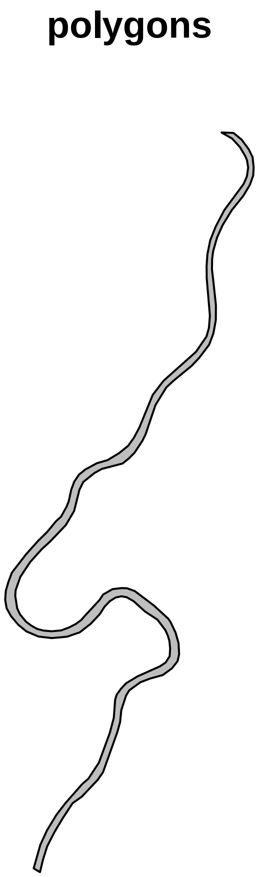


A SpatialLines object is made by joining up the points in sequence, and plot draws the resulting zig-zags. We can create a SpatialPolygons object from data provided with “sp” outlining the banks of the River Meuse.

```
data(meuse.riv)
summary(meuse.riv)
```

```
##        V1               V2
##  Min.   :178304   Min.   :325699
##  1st Qu.:179341   1st Qu.:329516
##  Median :180188   Median :330276
##  Mean   :180291   Mean   :331462
##  3rd Qu.:181174   3rd Qu.:333722
##  Max.   :182332   Max.   :337685
```

```
meuse.lst <- list(Polygons(list(Polygon(meuse.riv)), "meuse.riv"))
meuse.pol <- SpatialPolygons(meuse.lst)
plot(meuse.pol, col = "grey")
title("polygons")
```



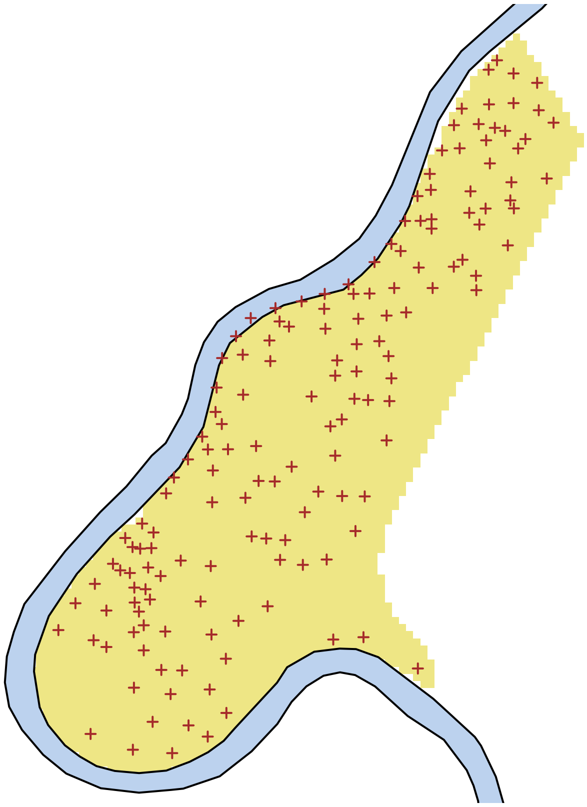
or, we als can convert grid data for the same Meuse bank study area into a ‘`SpatialPixels`’ object and display it using the image method, with all cells set to color ‘grey’.

```r
data(meuse.grid)
coordinates(meuse.grid) <- c("x", "y")
meuse.grid <- as(meuse.grid, "SpatialPixels")
image(meuse.grid, col = "grey")
title("grid")
```

From the previous maps, one unit in the x-direction equals one unit in the y-direction. This is the default when the coordinate reference system is not `longlat` or is unknown. For unprojected data in geographical coordinates (longitude/ latitude), the default aspect ratio depends on the (mean) latitude of the area plotted. We can change the default aspect by passing the `asp` argument.

Commonly, when we incorporate more elements in a map, it is more readable. Here we can use the function `add = TRUE` for this purpose:

```r
image(meuse.grid, col = "khaki2")
plot(meuse.pol, col = "lightsteelblue2", add = TRUE)
plot(meuse, add = TRUE, col = "brown", cex = .5)
```



2.2. Axes and layout elements

Often maps don't have axes. Since the projected coordinates are usually long, hard to read and geographical reference is much easier when recognisable features such as administrative boundaries, rivers, coast lines, etc. are present.

In the `plot`() function of R, the boolean argument `axes` can be set to control the shape of the axes.

```
layout(matrix(c(1, 2), 1, 2))
plot(meuse.pol, axes = TRUE)
plot(meuse.pol, axes = FALSE)
axis(1, at = c(178000 + 0:2 * 2000), cex.axis = 0.7)
axis(2, at = c(326000 + 0:3 * 4000), cex.axis = 0.7)
box()
```

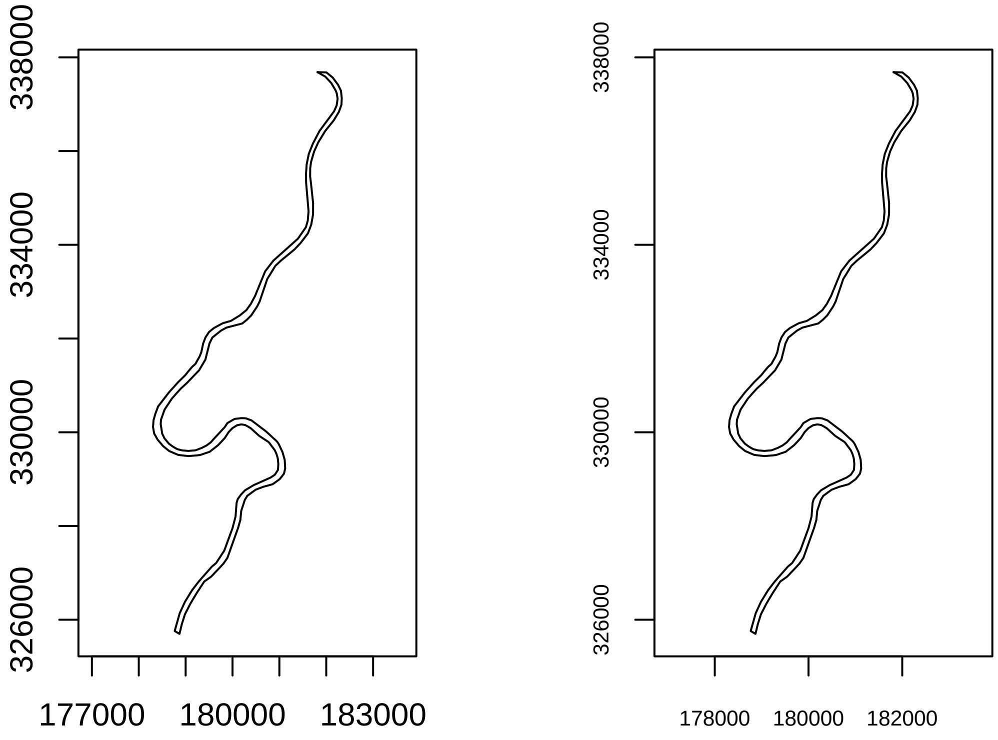
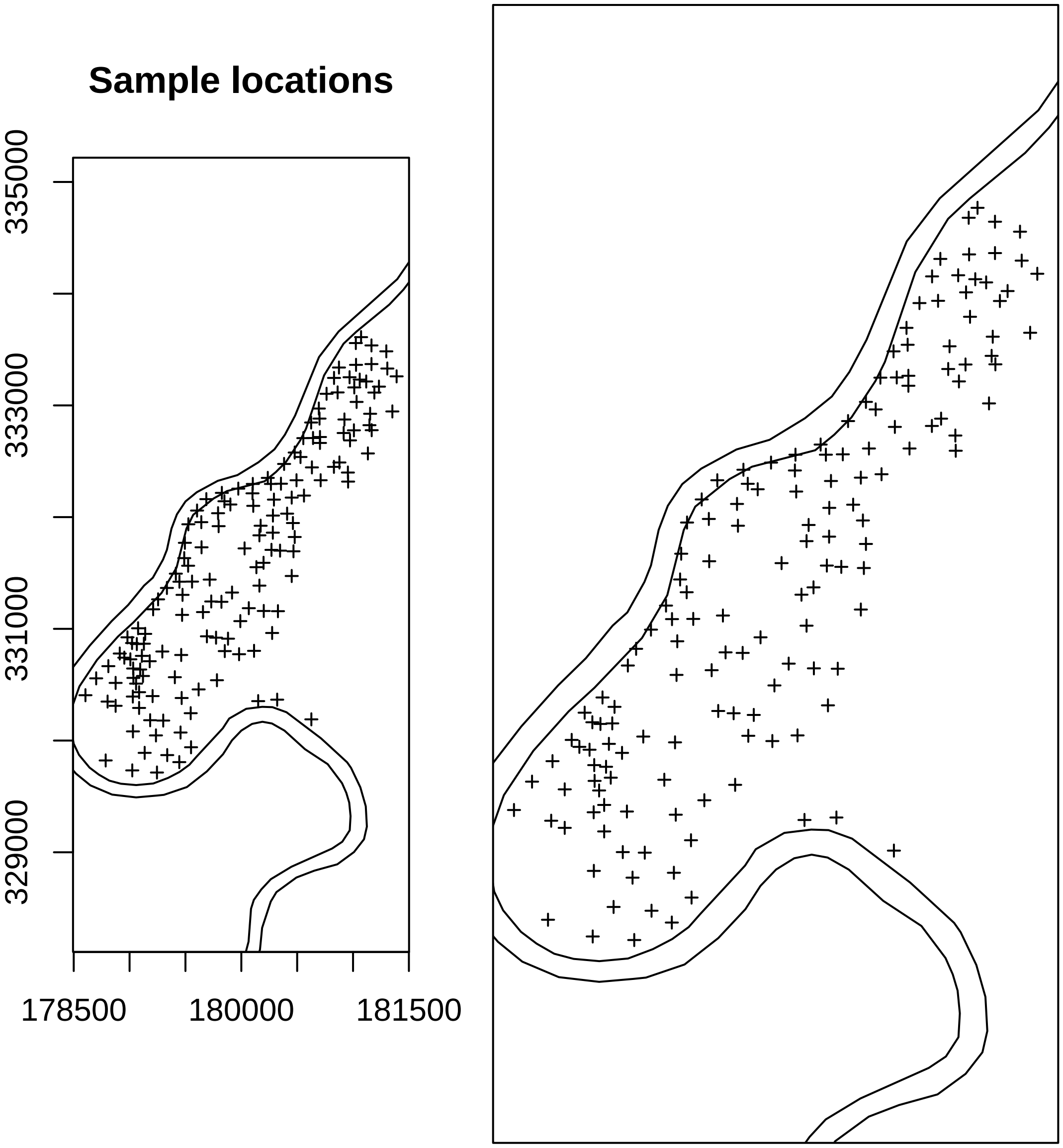


Figura 1: Default axes (left) and custom axes (right) for the meuse.riv data

R reserves the necessary space for adding axes and titles later on. However, we can instruct to R to not reserve this space by using function `par`, which is intended to have side effects on the next plot on the current device.

```r
oldpar = par(no.readonly = TRUE)
layout(matrix(c(1, 2), 1, 2))
plot(meuse, axes = TRUE, cex = 0.6)
plot(meuse.pol, add = TRUE)
title("Sample locations")
par(mar = c(0, 0, 0, 0) + 0.1)
plot(meuse, axes = FALSE, cex = 0.6)
plot(meuse.pol, add = TRUE)
box()
```



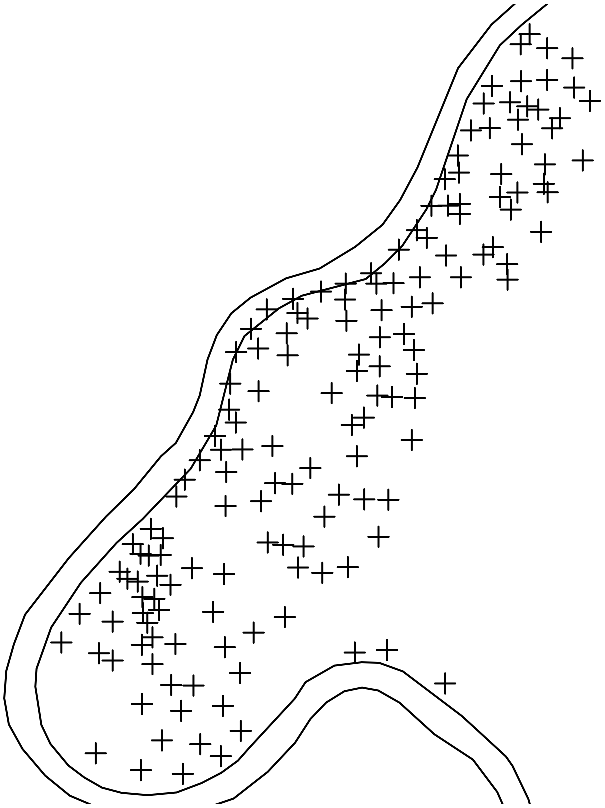
```
par(oldpar)
```

to return to the default version of plots in R using the line `par(oldpar)` declared previously in the same code.

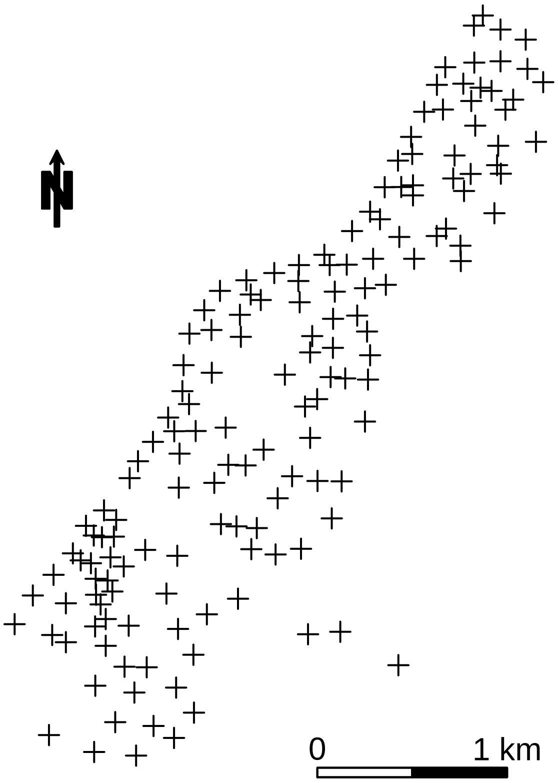
To modify the margins of the plot we have to set the `mar` argument in the `par` command. For example, doing the following `par(mar=c(3,3,2,1))`. To get more information visit the following link : https://bookdown.org/ndphillips/YaRrr/plot-margins.html

When we decide not to show the axes on a map, we can provide the reader of a map with a guidance for distance and direction by plotting a scale bar and a north arrow, which can be placed interactively using `locator` followed by a few well-chosen clicks in the map.

```
plot(meuse)
plot(meuse.pol, add=TRUE)
```



```
plot(meuse)
SpatialPolygonsRescale(layout.scale.bar(), offset = c(180200,329600),
                       scale = 1000, fill=c("transparent","black"), plot.grid = FALSE)
text(x = c(180200,181200), y = rep(329750, 2), c("0", "1 km"))
SpatialPolygonsRescale(layout.north.arrow(), offset = c(178750,332500),
                       scale = 400, plot.grid = FALSE)
```

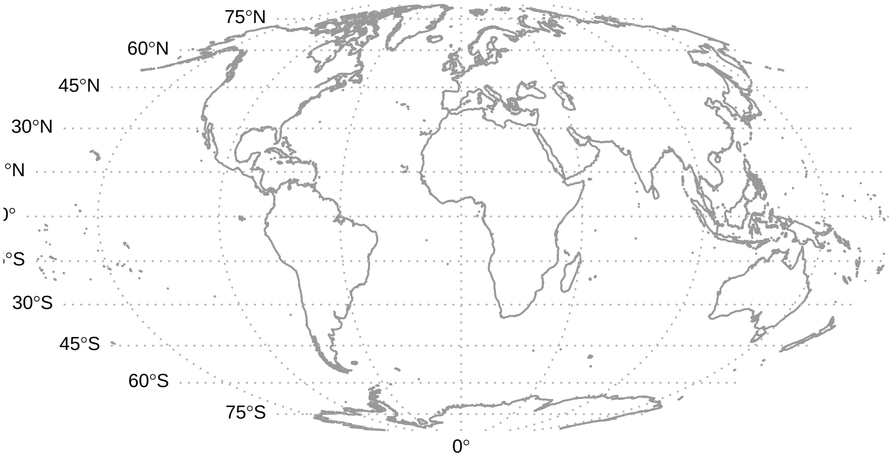


2.3. Degrees in axes labels and reference grid

Unprojected data have coordinates in latitude and longitude degrees, with negative degrees referring to degrees west (of the prime meridian) and south (of the Equator). When unprojected spatial data are plotted using the “sp” package (by `plot()` or `ssplot()` functions), the axis label marks will give units in decimal degrees N/S/E/W, for example 50,5° N.

```
library(maptools)
library(maps)
wrld <- map("world", interior = FALSE, xlim = c(-179, 179), ylim = c(-89, 89),
            plot = FALSE)
wrld_p <- pruneMap(wrld, xlim = c(-179, 179))
llCRS <- CRS("+proj=longlat +ellps=WGS84")
wrld_sp <- map2SpatialLines(wrld_p, proj4string = llCRS)
prj_new <- CRS("+proj=moll")

library(rgdal)
wrld_proj <- spTransform(wrld_sp, prj_new)
wrld_grd <- gridlines(wrld_sp, easts = c(-179, seq(-150, 150, 50), 179.5),
                      norths = seq(-75, 75, 15), ndiscr = 100)
wrld_grd_proj <- spTransform(wrld_grd, prj_new)
at_sp <- gridat(wrld_sp, easts = 0, norths = seq(-75, 75, 15), offset = 0.3)
at_proj <- spTransform(at_sp, prj_new)
plot(wrld_proj, col = "grey60")
plot(wrld_grd_proj, add = TRUE, lty = 3, col = "grey70")
text(coordinates(at_proj),pos = at_proj$pos, offset = at_proj$offset,
labels = parse(text = as.character(at_proj$labels)),
cex = 0.6)
```


75°N
60°N
45°N
30°N
°N
)°
°S
30°S
45°S
60°S
75°S
0°

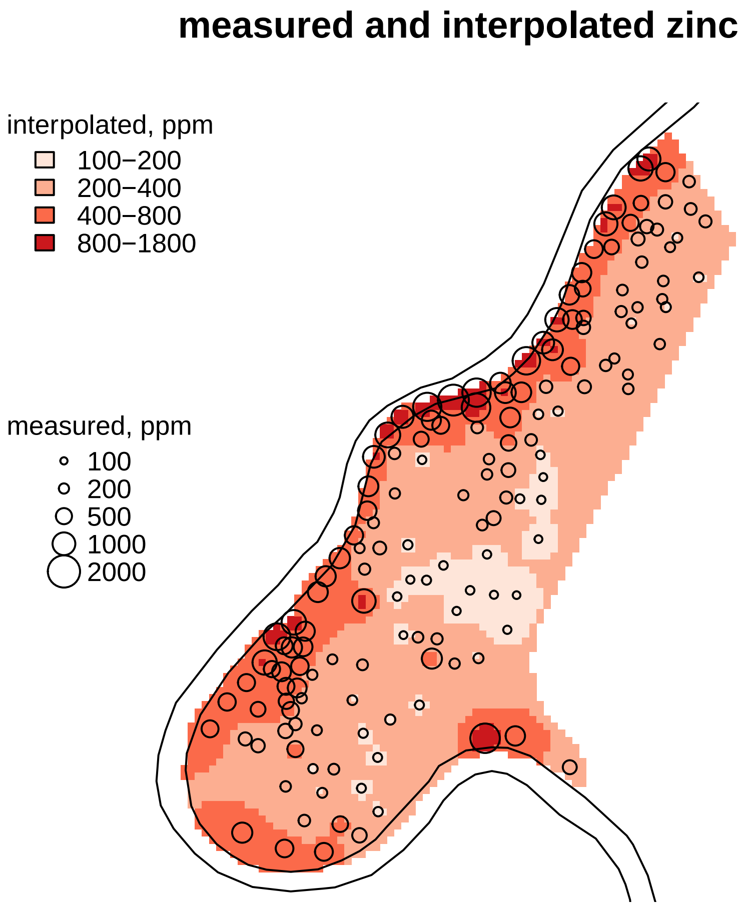
2.4. Plotting attributes and map legends

Up to now we have only plotted the geometry or topology of the spatial objects. If in addition we want to show feature characteristics or attributes of the objects, we need to use type, size, or colour of the symbols, lines, or polygons. There are graphic arguments that can be passed to the plot methods for the Spatial classes with attributes (https://cran.r-project.org/web/packages/sp/vignettes/intro_sp.pdf).

```
library(gstat)
data(meuse.grid)
coordinates(meuse.grid) <- c("x", "y")
gridded(meuse.grid) <- TRUE
zn.idw <- krige(log(zinc) ~ 1, meuse, meuse.grid)
```

```
## [inverse distance weighted interpolation]
```

```
library(RColorBrewer)
cols <- brewer.pal(4, "Reds")
image(zn.idw, col = cols, breaks=log(c(100,200,400,800,1800)))
plot(meuse.pol, add = TRUE)
plot(meuse, pch = 1, cex = sqrt(meuse$zinc)/20, add = TRUE)
legVals <- c(100, 200, 500, 1000, 2000)
legend("left", legend=legVals, pch = 1, pt.cex = sqrt(legVals)/20, bty = "n",
       title="measured, ppm", cex=0.8, y.inter=0.9)
legend("topleft", fill = cols,
       legend=c("100-200","200-400","400-800",
                "800-1800"), bty = "n", title = "interpolated, ppm",
       cex=0.8, y.inter=0.9)
title("measured and interpolated zinc")
```



3. Trellis/Lattice plots with `spplot()`

Aside from the traditional plot methods provided by the "sp" package, a second function, called `spplot()`, provides plotting of spatial data with attributes through the Trellis graphics system (Cleveland, 1993, 1994), which is for R provided (and extended) by package "lattice" (Sarkar, 2007).

Consider the plotting of two interpolation scenarios for the zinc variable in the "meuse" data set, obtained on the direct scale and on the log scale.

```
library(lattice)
data(meuse)
```

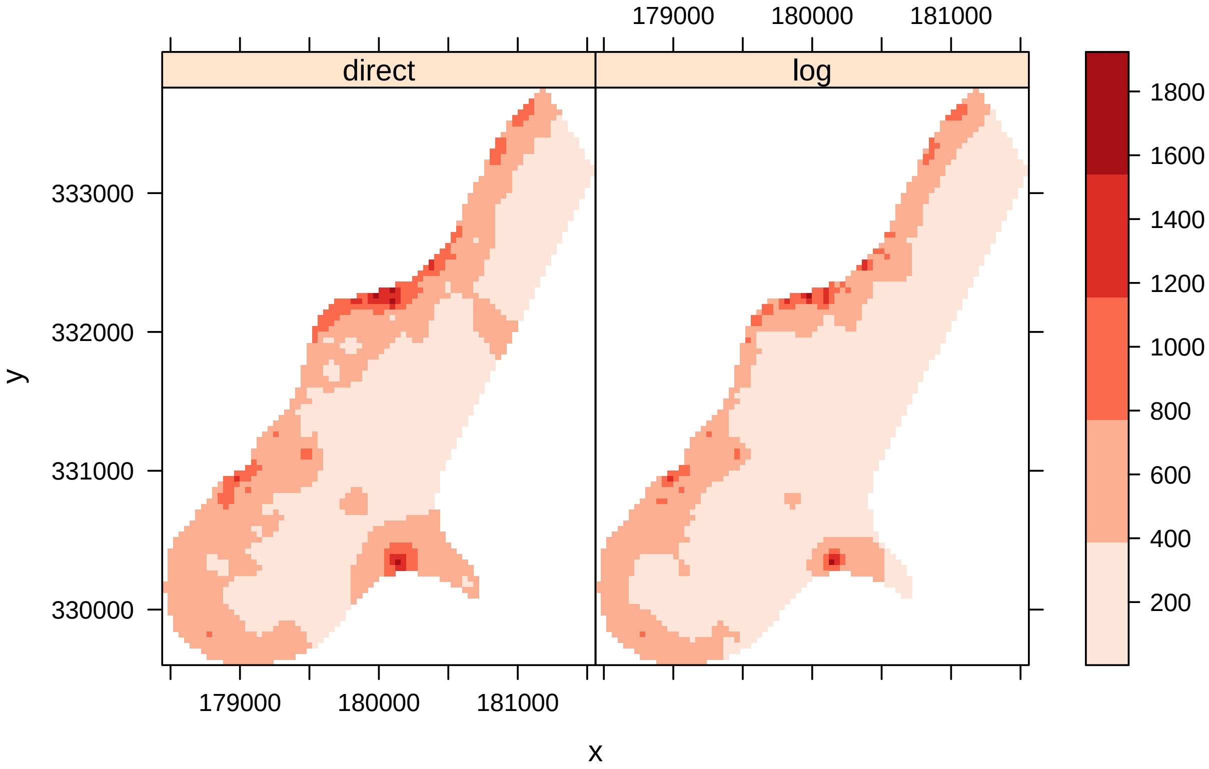
```
coordinates(meuse) <- ~x+y
data(meuse.grid)
coordinates(meuse.grid) <- ~x+y
gridded(meuse.grid) <- T
zn <- krige(zinc~1,meuse,meuse.grid)
```

```
## [inverse distance weighted interpolation]
```

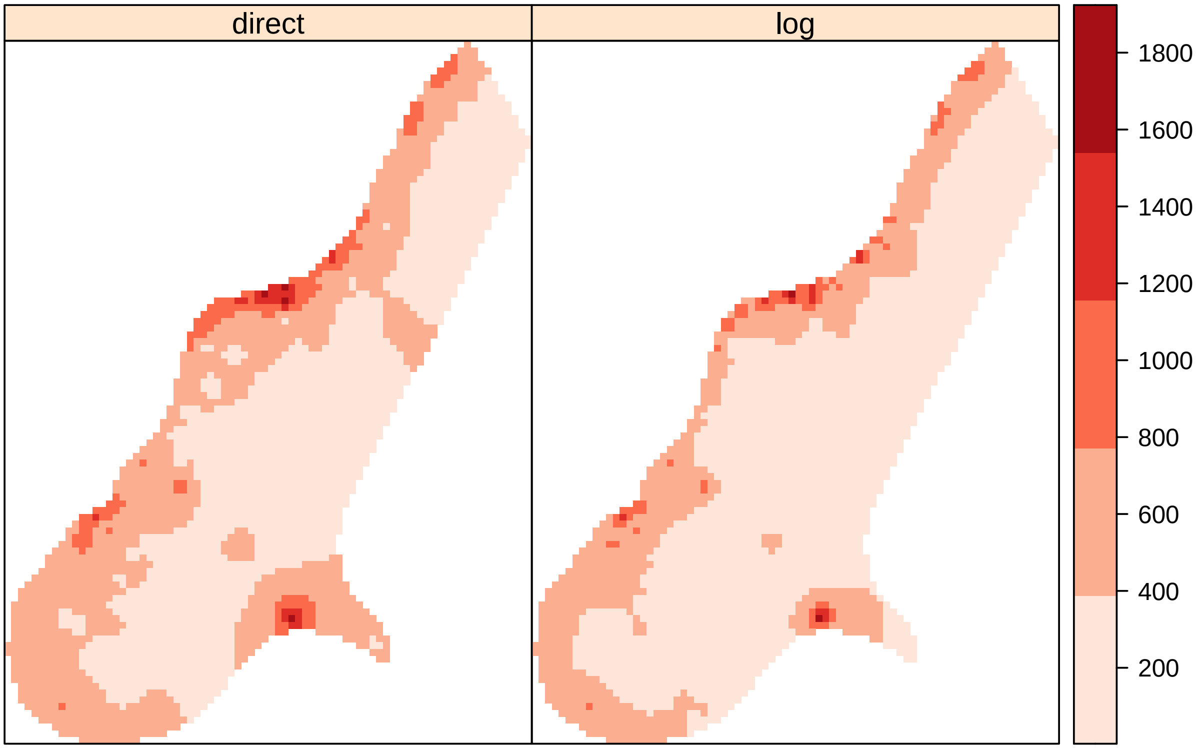
```
zn$direct <- zn$var1.pred
zn$log <- exp(krige(log(zinc)~1,meuse,meuse.grid)$var1.pred)
```

```
## [inverse distance weighted interpolation]
```

```
print(levelplot(z~x+y|name, spmap.to.lev(zn[c("direct", "log")]), asp = "iso",
                cuts=4, col.regions= brewer.pal(5, "Reds")))
```



```
print(spplot(zn[c("direct", "log")], cuts=4, asp = "iso",
             col.regions= brewer.pal(5, "Reds")))
```



The plot showed previously has four dimensions: the geographic space (x and y coordinates), the attribute values displayed in colour, and the panel identifier (the interpolation scenario) but which may be used to denote, for example attribute variable or time.

The `spplot` function does all this too, but hides many details. It provides a simple access to the functions provided by package “lattice” for plotting objects deriving from class `Spatial`, while retaining the flexibility offered by the “lattice” package.

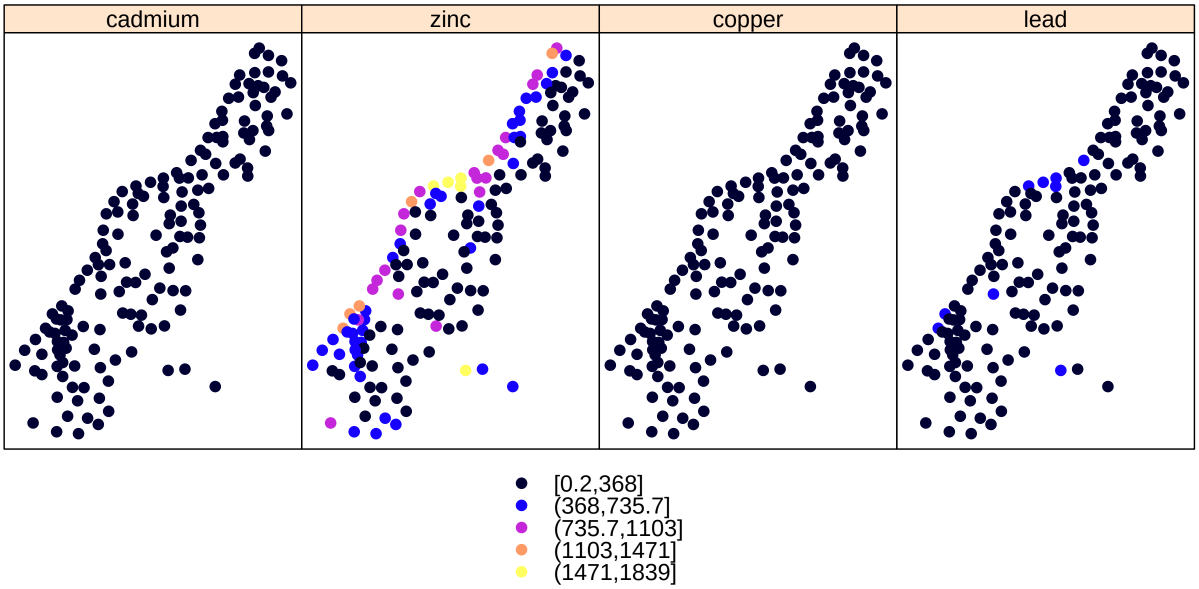
3.1. Plotting points, lines, polygons, and grids

The function `spplot()` plots spatial objects using colour (or grey tone) to denote attribute values. The first argument therefore has to be a spatial object with attributes.

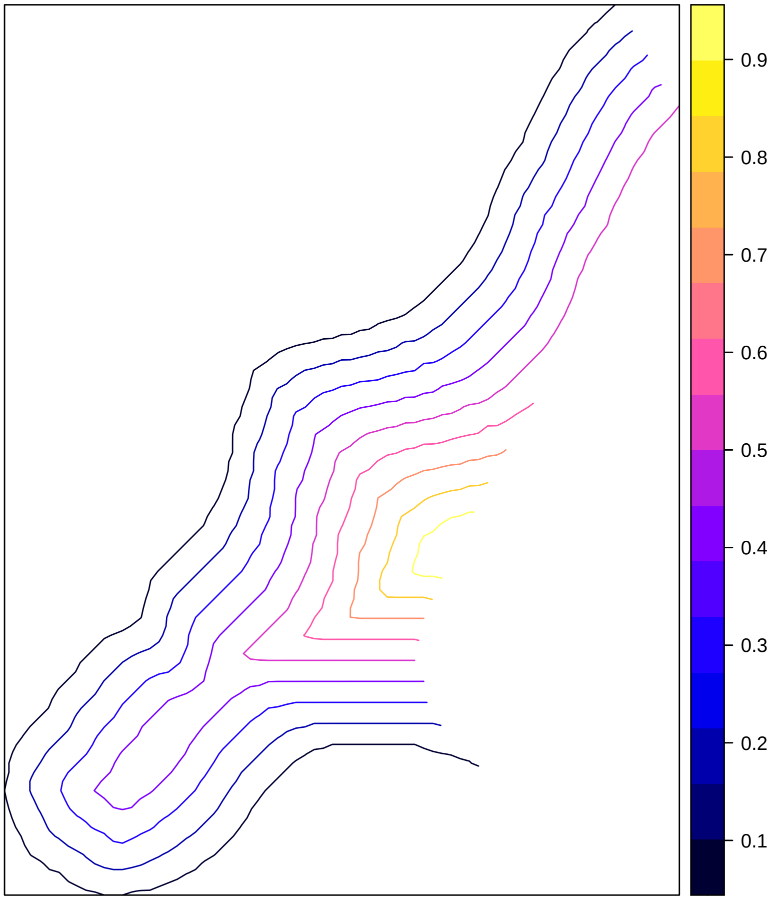
The basic arguments to use the `spplot()` function are:

- A `SpatialDataFrame` object with points, lines, polygons, or a grid.
- The attributes (column names or numbers) must be used; if omitted, all attributes are plotted. Further attributes control the plotting: colours, symbols, legend classes, size, axes, and geographical reference items to be added.

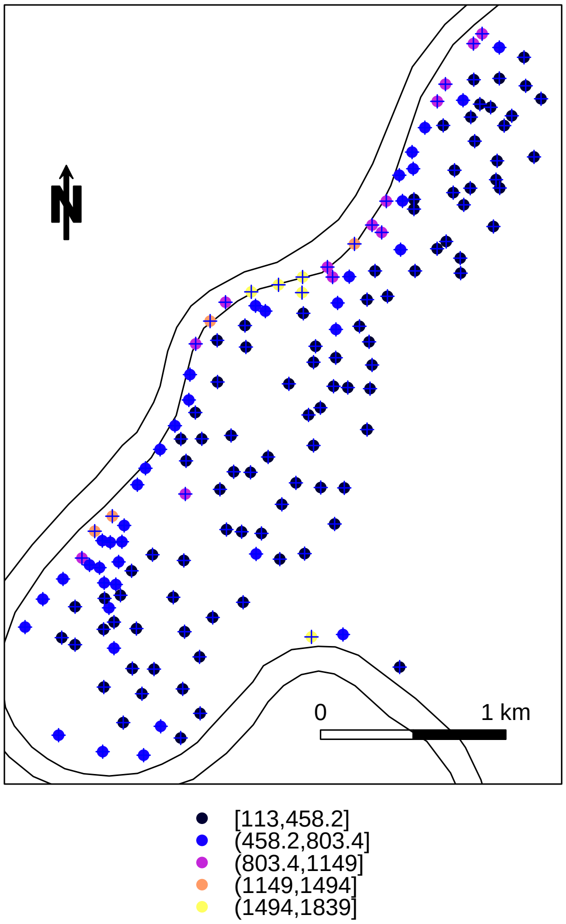
```
library(lattice)
spplot(meuse, c("cadmium", "zinc", "copper", "lead")) # easy way to plot side by side maps
```



```
library(maptools)
data(meuse.grid)
coordinates(meuse.grid) <- c("x", "y")
meuse.grid <- as(meuse.grid, "SpatialPixelsDataFrame")
im <- as.image.SpatialGridDataFrame(meuse.grid["dist"])
cl <- ContourLines2SLDF(contourLines(im))
spplot(cl)
```



```
river <- list("sp.polygons", meuse.pol)
north <- list("SpatialPolygonsRescale", layout.north.arrow(),
              offset = c(178750, 332500), scale = 400)
scale <- list("SpatialPolygonsRescale", layout.scale.bar(),
              offset = c(180200, 329800), scale = 1000,
              fill = c("transparent", "black"))
txt1 <- list("sp.text", c(180200, 329950), "0")
txt2 <- list("sp.text", c(181200, 329950), "1 km")
pts <- list("sp.points", meuse, pch = 3, col = "blue")
meuse.layout <- list(river, north, scale, txt1, txt2, pts)
zn <- meuse["zinc"]
spplot(zn, sp.layout = meuse.layout)
```



References

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

Pebesma, E., & Bivand, R. S. (2005). S classes and methods for spatial data: the sp package. R news, 5(2), 9-13.

Becker, R. A., Cleveland, W. S., Shyu, M. J., & Kaluzny, S. P. (1996). A tour of Trellis graphics. Murray Hill, NJ: AT & T Bell Laboratories, 44.

Sarkar, D., & Sarkar, M. D. (2007). The lattice package. Trellis Graphics for R.