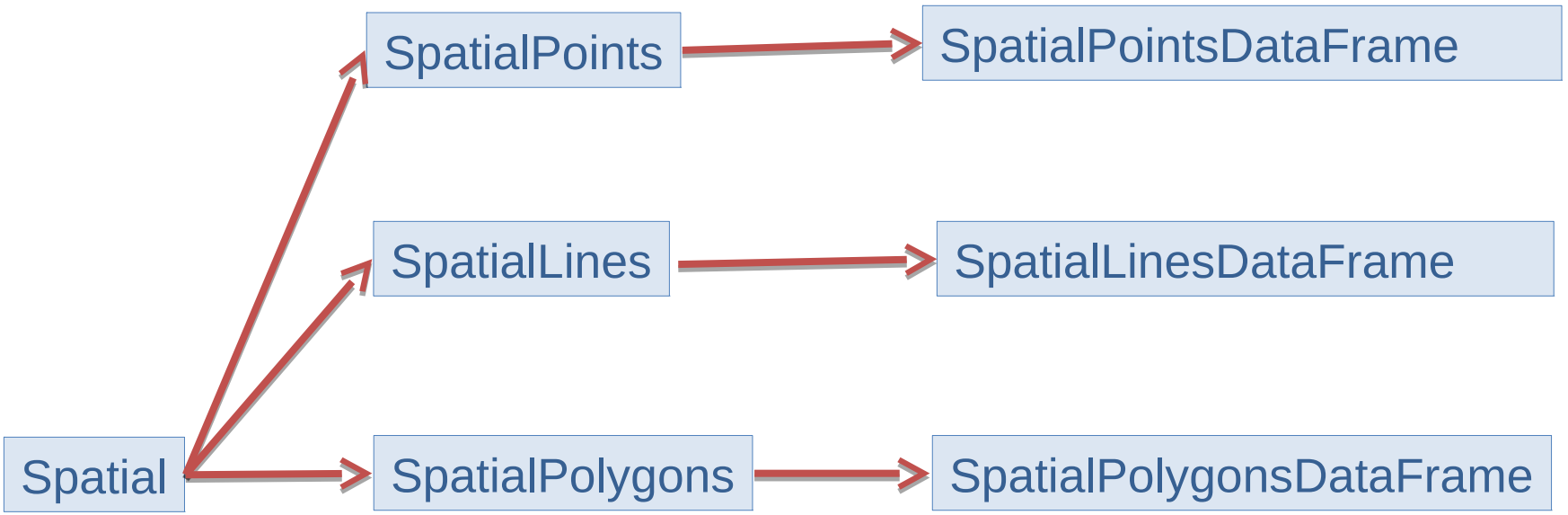
Spatial data analysis with

Spatial is special

- Complex: geometry and attributes
- The earth is not flat
- Size: lots and lots of it, multivariate, time series
- Special plots: maps
- First Law of Geography: nearby things are similar
  - Statistical assumptions: violated
  - Interpolation: possible

The ‘sp’ package defines *classes* to represent spatial data.



```
> str(new('SpatialPolygonsDataFrame'))

Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 0 obs. of  0 variables

Formal class 'data.frame' [package "methods"] with 4 slots
  .. .. ..@ .Data    : list()
  .. .. ..@ names    : chr(0)
  .. .. ..@ row.names: int(0)
  .. .. ..@ .S3Class : chr "data.frame"
  ..@ polygons   : list()
  ..@ plotOrder  : int(0)
  ..@ bbox       : num[0 , 0 ]
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr ""
>
```

sp is not designed to do much

To provide classes that can be used by other packages

Mostly house-keeping (creation and coercion of objects)

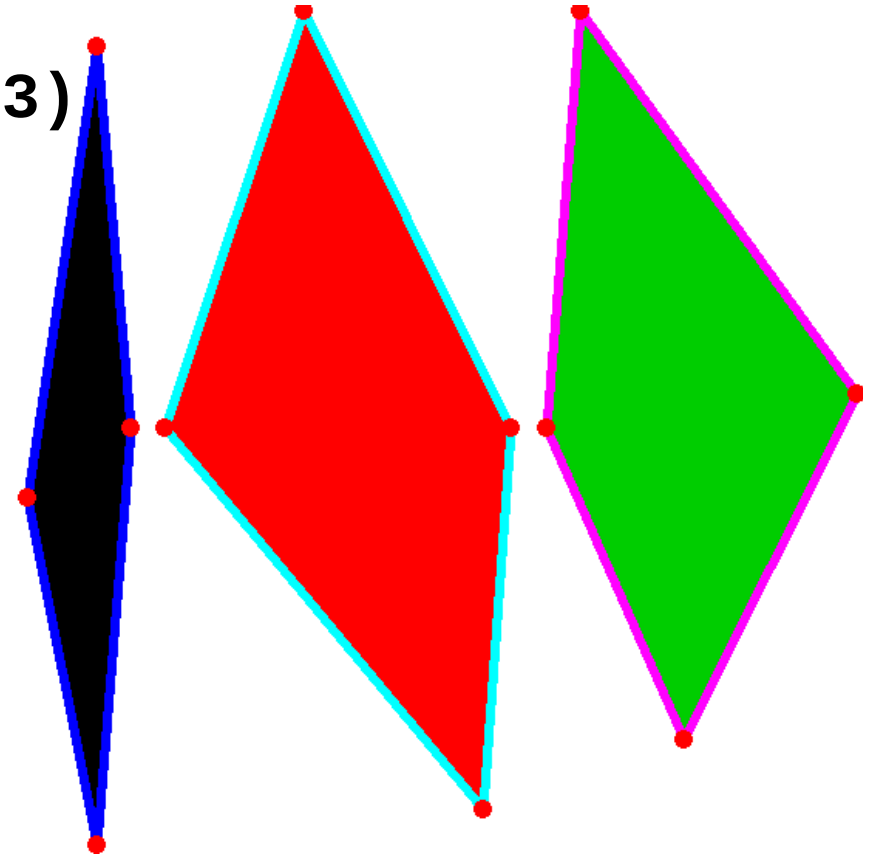
But it has functions for other things (overlay, sample, plot)

SpatialPoints

```
x <- c(4,7,3,8)
y <- c(9,6,12,11)
xy <- data.frame(x, y)

library(raster)
sp <- SpatialPoints(xy)
crs(sp) <- "+proj=longlat +datum=WGS84"

sp <- SpatialPoints(xy, proj4string=
            CRS("+proj=longlat +datum=WGS84"))

coordinates(xy) <- ~ x + y
```

```
sp
class       : SpatialPoints
features    : 4
extent      : 3, 8, 6, 12  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84


d <- data.frame(v1=1:4, v2=LETTERS[1:4])

SpatialPointsDataFrame(sp, d)

class       : SpatialPointsDataFrame
features    : 4
extent      : 3, 8, 6, 12  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84
variables   : 2
names       : v1, v2
min values  :  1,  A
max values  :  4,  D
```

```
x1 <- rbind(c(-80,-10), c(-70,55), c(-65, 0), c(-70,-60))
x2 <- rbind(c(-10,0), c(-40,60), c(-60,0), c(-14,-55))
x3 <- rbind(c(-5,0), c(0,60), c(40,5), c(15,-45))

p <- spPolygons(x1, x2, x3)
```



```
plot(p, col=1:3, border=4:6, lwd=5)
points(x1, col='red', pch=20, cex=2)
```

```
library(raster)
f <- "C:/data/lux.shp"
p <- shapefile(f)
p
```

```
class       : SpatialPolygonsDataFrame
features    : 12
extent      : 5.74, 6.53, 49.45, 50.18  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84
variables   : 5
names       : ID_1,     NAME_1, ID_2,   NAME_2, AREA
min values  :    1,   Diekirch,    1, Capellen,   76
max values  :    3, Luxembourg,   12,    Wiltz,  312
```

Equivalent to

```
library(rgdal)
readOGR(readOGR("C:/data", "lux"))
```

RasterLayer

```
> library(raster)
>
> x <- raster()
>
> x <- raster('volcano.tif')
>
> x
class       : RasterLayer
dimensions  : 87, 61, 5307  (nrow, ncol, ncell)
resolution  : 10, 10  (x, y)
extent      : 2667400, 2668010, 6478700, 6479570  (xmin,
coord. ref. : +proj=nzmg +lat_0=-41 +lon_0=173 +x_0=251
values      : d:\data\volcano.tif
min value   : 94
max value   : 195
```

RasterLayer

> str(x)

Formal class 'RasterLayer' [package "raster"] with 16 slots
..@ **file** :Formal class '.RasterFile' [package "raster"] with 9 slots
. . .. ..@ name : chr “d:\\data\\volcano.tif“
.. .. ..@ driver : chr "gdal"
..@ **data** :Formal class '.SingleLayerData' [package "raster"] with 11 slots
.. .. ..@ values : logi(0)
.. .. ..@ inmemory : logi FALSE
.. .. ..@ min : num 94
. . .. ..@ max : num 195
.. .. .. @ names: chr "volcano”
..@ **extent** :Formal class 'Extent' [package "raster"] with 4 slots
.. .. ..@ xmin: num 2667400
.. .. ..@ xmax: num 2668010
.. @ **rotation** :Formal class '.Rotation' [package "raster"] with 2 slots
.. .. ..@ geotrans: num(0)
.. .. ..@ transfun:function ()
..@ **ncols** : int 61
..@ **nrows** : int 87
..@ **crs** :Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr " +proj=nzmg +lat_0=-41 +lon_0=173 +x_0=2510000 +y_0=6023150

Multiple layers

RasterStack - many files

RasterBrick - single files

Basic functions

```
ncell(x)

xyFromCell(x, 10)

getValues(x, row)

writeRaster(x, filename, …)
```

Raster algebra

```
r <- raster(nc=10, nr=10)
values(r) <- 1:ncell(r)
q <- sqrt(r)
x <- (q + r) * 2

s <- stack(r, q, x)
ss <- s * r
```

Raster manipulation

merge, crop,

project, aggregate,

reclass, resample,

rasterize, …

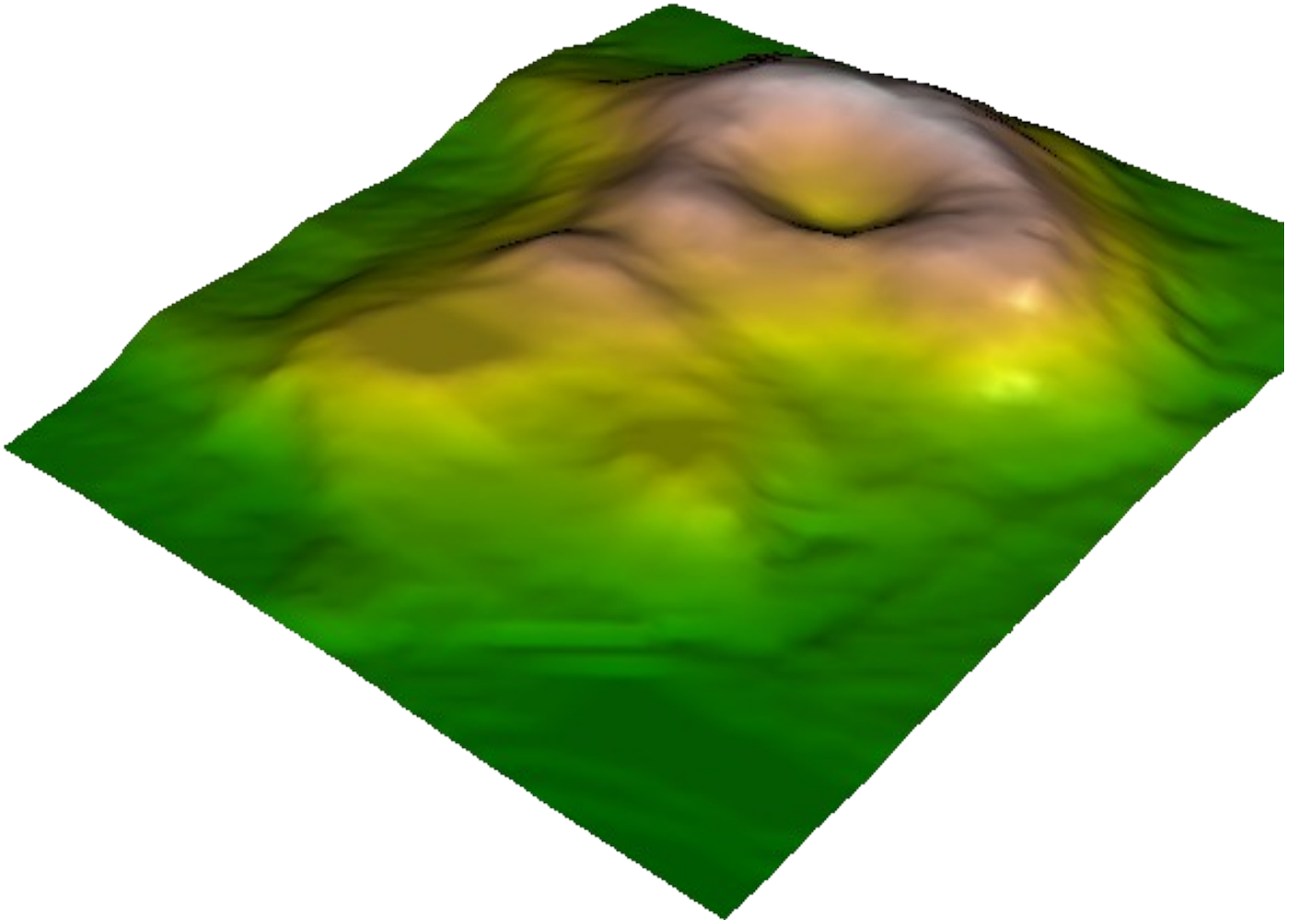
Raster analysis

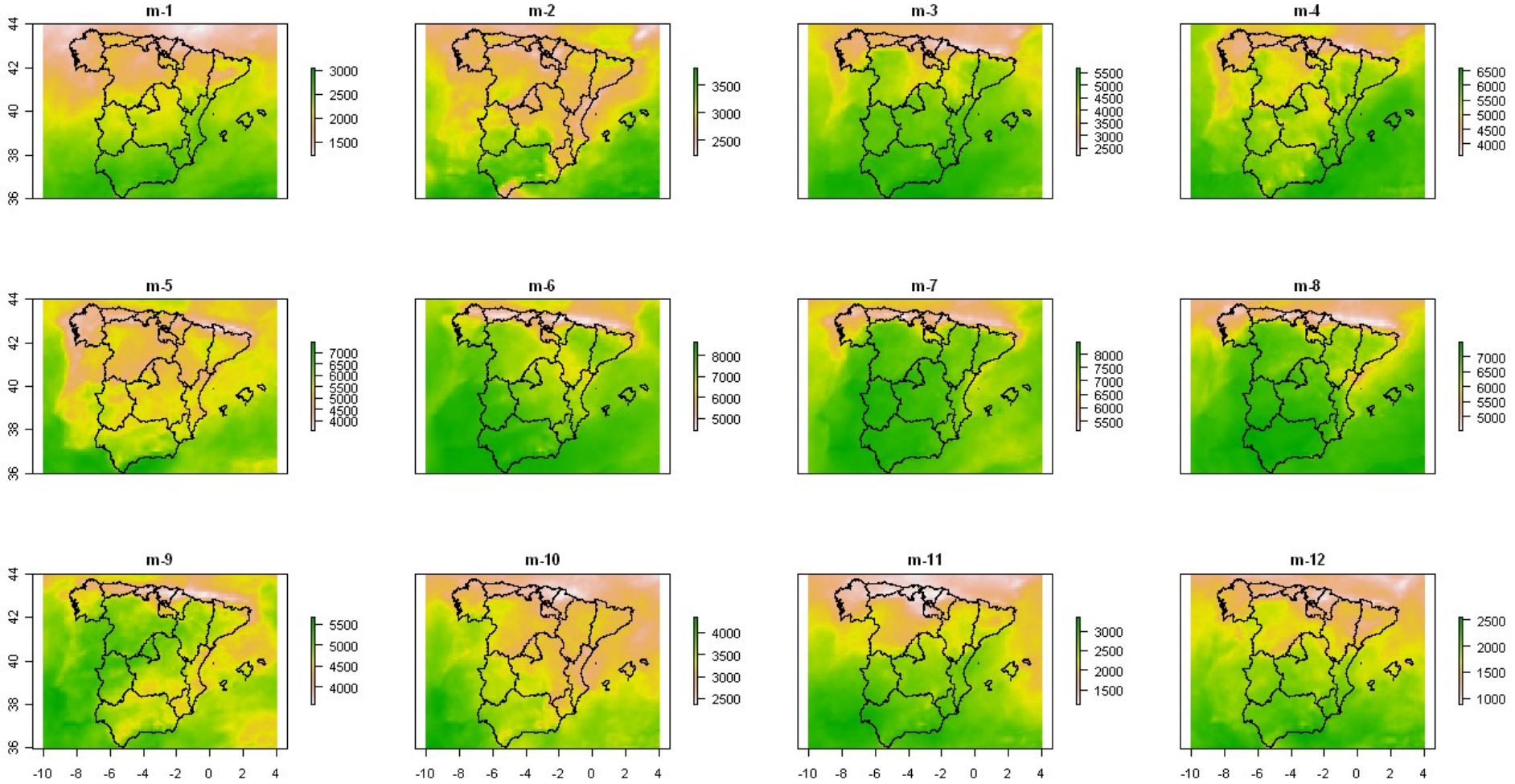
`distance`,
`focal`,
`predict`,
…

and other *R* functions
and external models

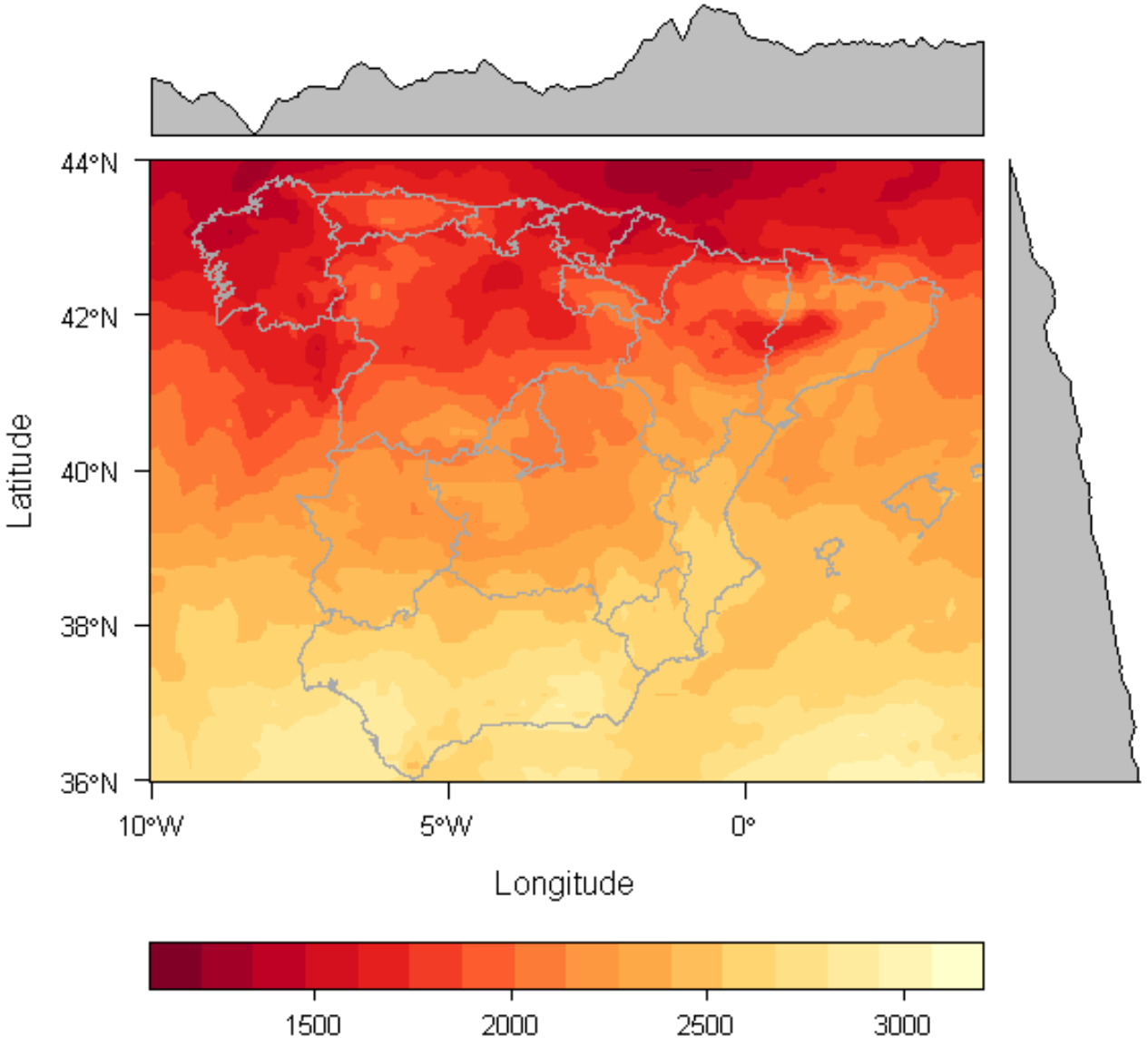
Visualization (rasterVis)

```
plot
plotRGB
contour
plot3D
…
```






```
plot(s, addfun=function()plot(esp, add=T))
```




```
p <- levelplot(x, layers=1, FUN.margin=median)
     p + layer(sp.lines(esp, lwd=0.8, col='darkgray'))
```

```
> library(dismo)
> g <- gmap('Albacete, Spain', scale=2, lonlat=TRUE)
> plot(g, interpolate=TRUE)
> xy <- geocode("Universidad de Castilla-La Mancha, Albacete, Spain")
> points(xy[,3:4], col='red', pch='*', cex=5)
```

Writing you own functions

```
AddA <- function(x, a, filename='', ...) {
    out <- raster(x)
    if (filename == '') {
        filename <- rasterTmpFile()
    }
    out <- writeStart(out, filename, ...)
    bs <- blockSize(x)
    for (i in 1:bs$n) {
        v <- getValues(x, row=bs$row[i], nrows=bs$nrows[i])
        v <- v + a
        out <- writeValues(out, v, bs$row[i])
    }
    out <- writeStop(out)
    return(out)
}
```

Speeding things up

Probably the major issue
(not necessarily *difficult* to fix --- just a lot of work)

Multi-core computing
Do you really need it?

Multi-core

```
library(raster)
r <- raster()
r[] <- 1:ncell(r)
s <- stack(r, r*2, r*3)
z1 <- max(s)

beginCluster()

# approach 1
fun1 <- function(x) calc(x, max)
z2 <- clusterR(s, fun1)

# approach 2
cl <- getCluster()
fun2 <- function(x){ parApply(cl, x, 1, max) }
z3 <- calc(s, fun2 )
returnCluster()

endCluster()
```