Spatial Plotting in R

Ani Ghosh (anighosh@ucdavis.edu) Robert Hijmans (rhijmans@ucdavis.edu)

20 January 2020

Introduction

One of the biggest advantages of working with spatial data is you can easily visualualize the spatial variation of the data. You do it by making maps. Maps are everywhere! In this session, we will focus on making maps using both `raster` and `spatail` class data. Like basic plots, for making maps there are number packages and functions available. However, with the following two functions from `sp` packages a large number of plotting can be done. 1. `plot` based on base R plotting system 2. `spplot` based on `grid` system

`plot` allows incremental addition of graphical elements in a single plotting device; whereas `spplot` does not allow such addition (similar to `lattice` or `ggplot2`). `spplot` makes maps with shared axes which makes visualization and comparison of multiple maps much easier.

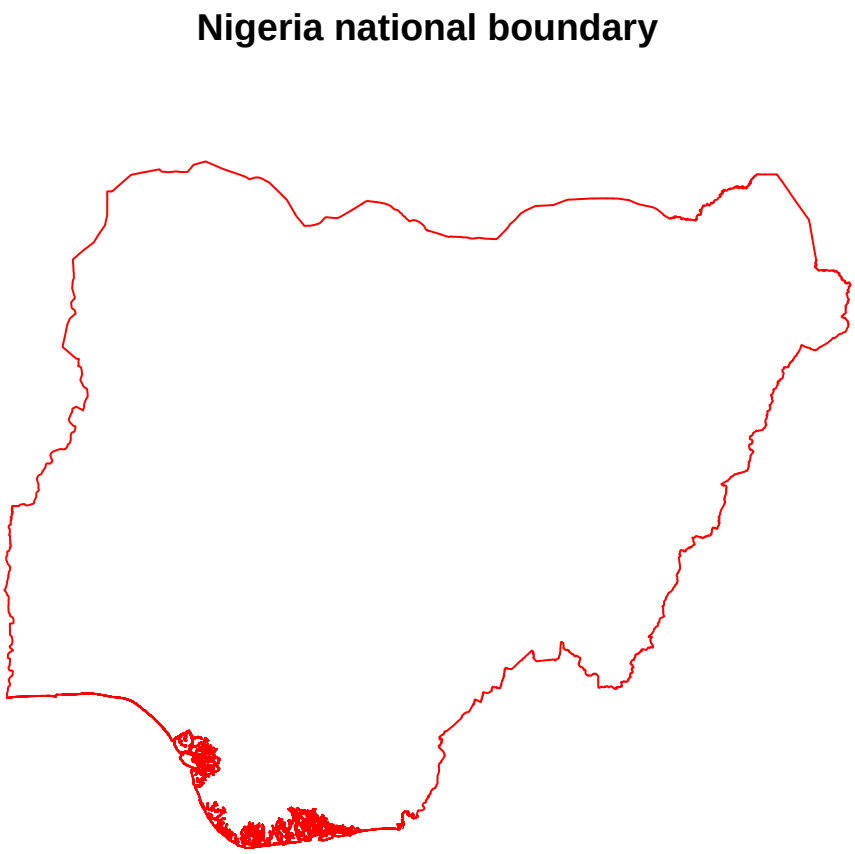
Plotting various spatial objects

We will plot various raster (elevation) and vector data (administrative boundaries, rivers, lakes and protected areas) of Tanzania. First we will use the `plot` function. Make sure your working directory is *data* directory for this specific excercise.

Important Some of the vector data is in `*.shp` file format while others is in `*.rds` R database format. You can save anykind of data in `*.rds` format. Use `readRDS` and `saveRDS` to read and save data in `*.rds` format respectively.

Using `plot()`

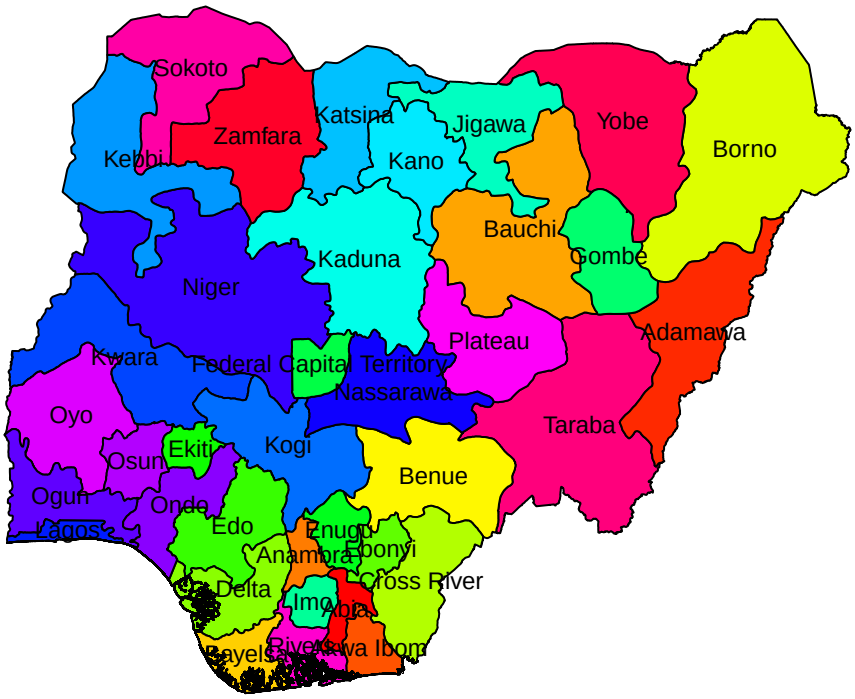
```
library(raster)
#setwd("...")
v0 <- getData("GADM", country = "NGA", level = 0)
plot(v0, border = "red")
title(main = "Nigeria national boundary", sub = "source GADM")
```



Ex 1. Fill the polygon using a different color and change increase the thickness of the border

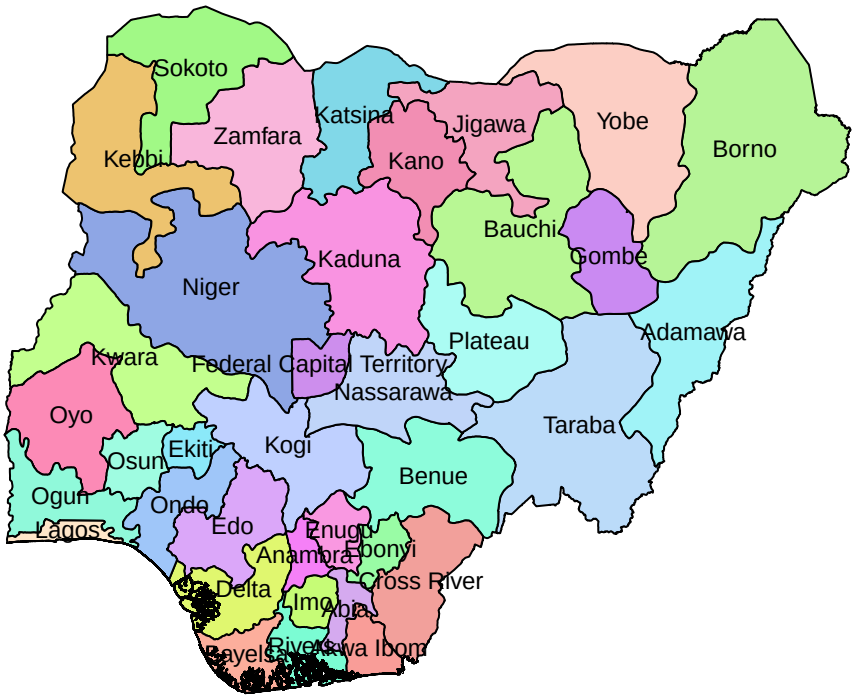
Color individual regions We will use the region boundaries from GADM database (level = 1)

```
v1 <- getData("GADM", country = "NGA", level = 1)

# Note: once you have downloaded the GADM boundaries, you can also read them using
# v1 <- readRDS("gadm36_NGA_1_sp.rds")

# Each region should have different color; we will use the color ramp
n <- length(v1$NAME_1)
plot(v1, col=rainbow(n), main = 'Administrative boundary: level 1')

# Now add name of the indiviudal regions to the map
text(v1, v1$NAME_1, cex=0.75)
```



I don't like the background colors. The name of the divisions are hardly visible. We can specify the colors we want or try to generate 'prettier' colors.

```
library("randomcoloR")
mycols <- randomColor(n, luminosity="light")
plot(v1, col=mycols, main = 'Administrative boundary: level 1')
text(v1, v1$NAME_1, cex=0.75)
```



Ex 2. Read and plot level 2 administrative boundaries of Tanzania. Give different colors to the regions.

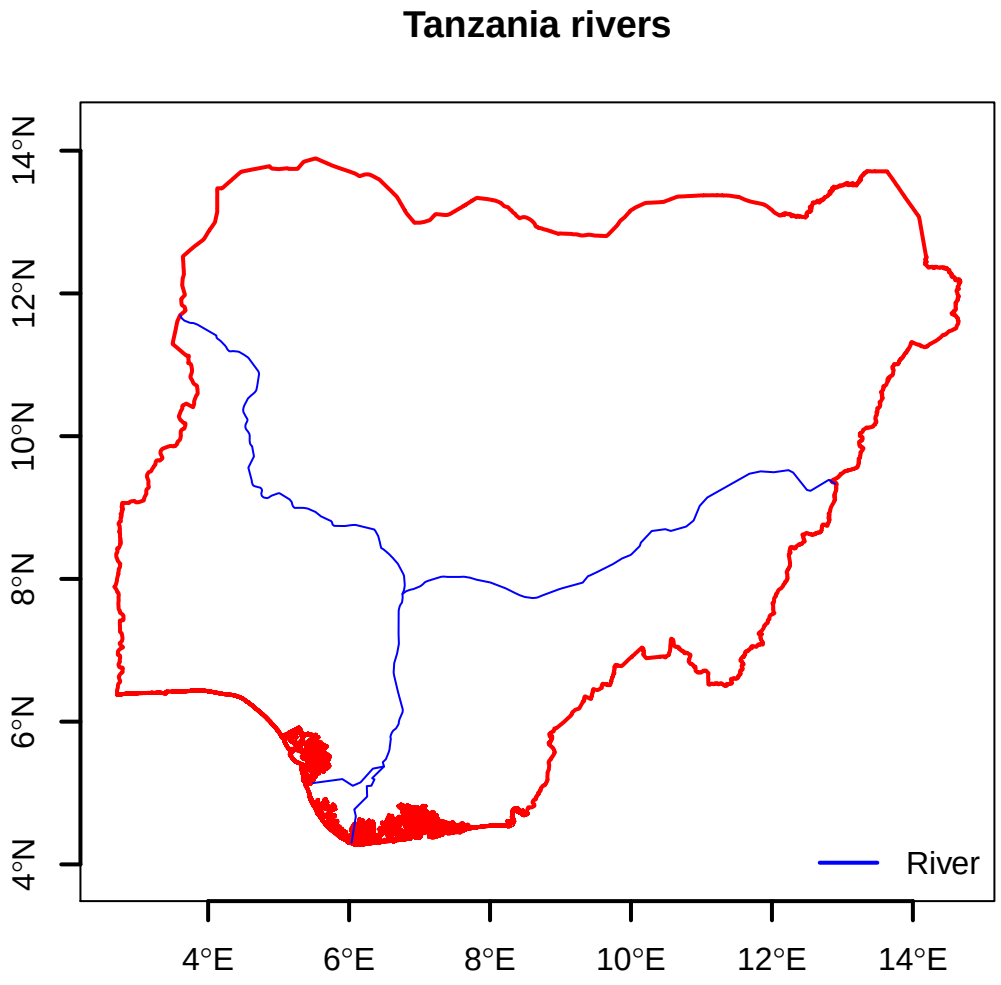
Add SpatialLines object to your plot Next we add new features to the boundary. I found the river boundaries from the Humdata website. I prefer to keep record of the source of the data and if possible, download the data with R. In this way, we can easily recreate the analysis and also share the analysis with our colleagues without the data (sometimes recommended to avoind the data volume).

First download the data. The download url can be found from this webpage and right-clicking to find the "Copy link address" against "NGA_river_line_esri.zip".

```
url <- "https://data.humdata.org/dataset/741c6f20-6956-420d-aae4-37015cdd1ad4/resource/0c6a29a7-0815-48
download.file(url, dest = 'nga_rivers.zip')
unzip('nga_rivers.zip')
```

Next inspect what happened in the 'data' folder.

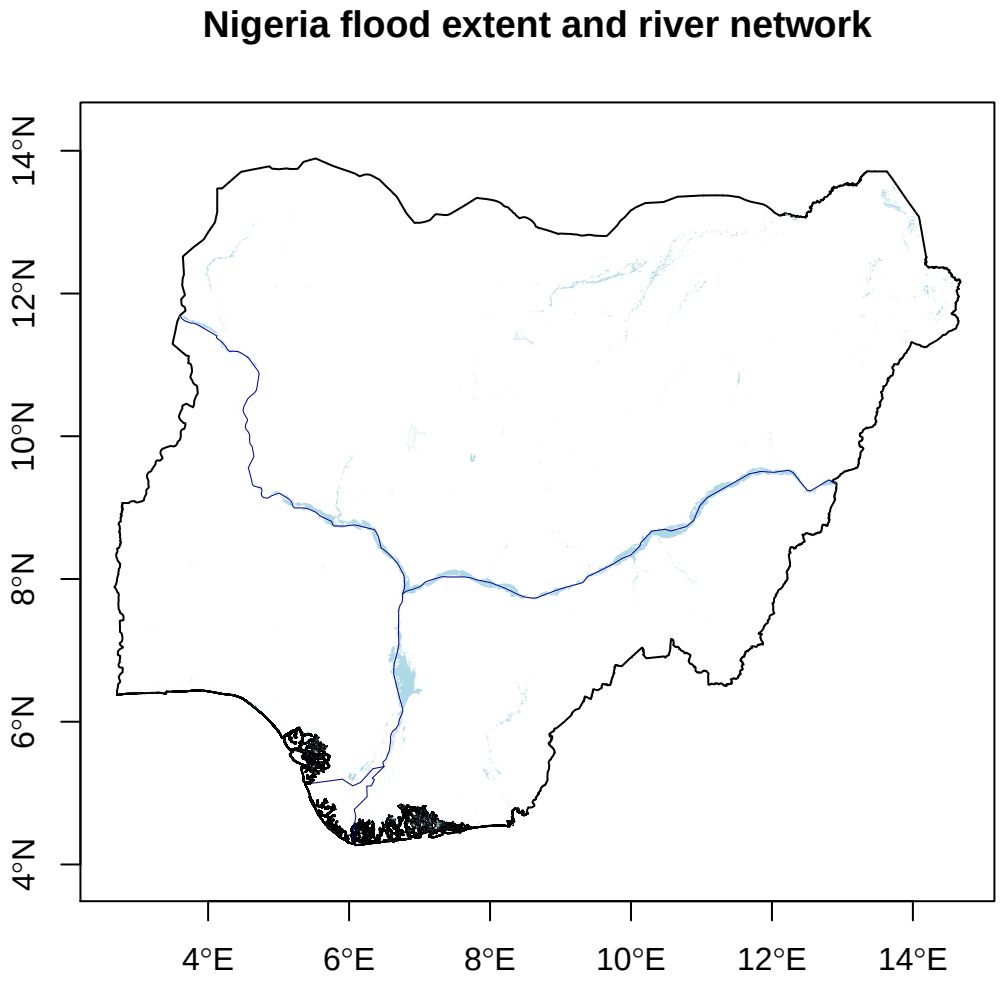
```r
# list the contents of the data folder.
ff <- list.files(path = ".", full.names = TRUE)

# we want to read the .shp file with for river outlines
rv <- shapefile('NGA_rvrsl_1m_esri.shp')

# Next plot rivers with the admin boundaries
plot(v0, border = "red", lwd = 2,  axes = TRUE)
plot(rv, col='blue', add = TRUE)
title(main = "Tanzania rivers")
# Add some more details
legend('bottomright', legend = 'River', lty = 1, lwd = 2, col = 'blue', bty = "n")
```



Ex 3. Create a similar plot with Nigeria waterways data found in https://data.humdata.org/dataset/nigeria-waterways.

Add additional SpatialPolygons We can include additional spatial polygons. One example is the flood extent in the year 2012 found in the same website (https://data.humdata.org/dataset/nigeria-flood-extents-nov-2012-fod).

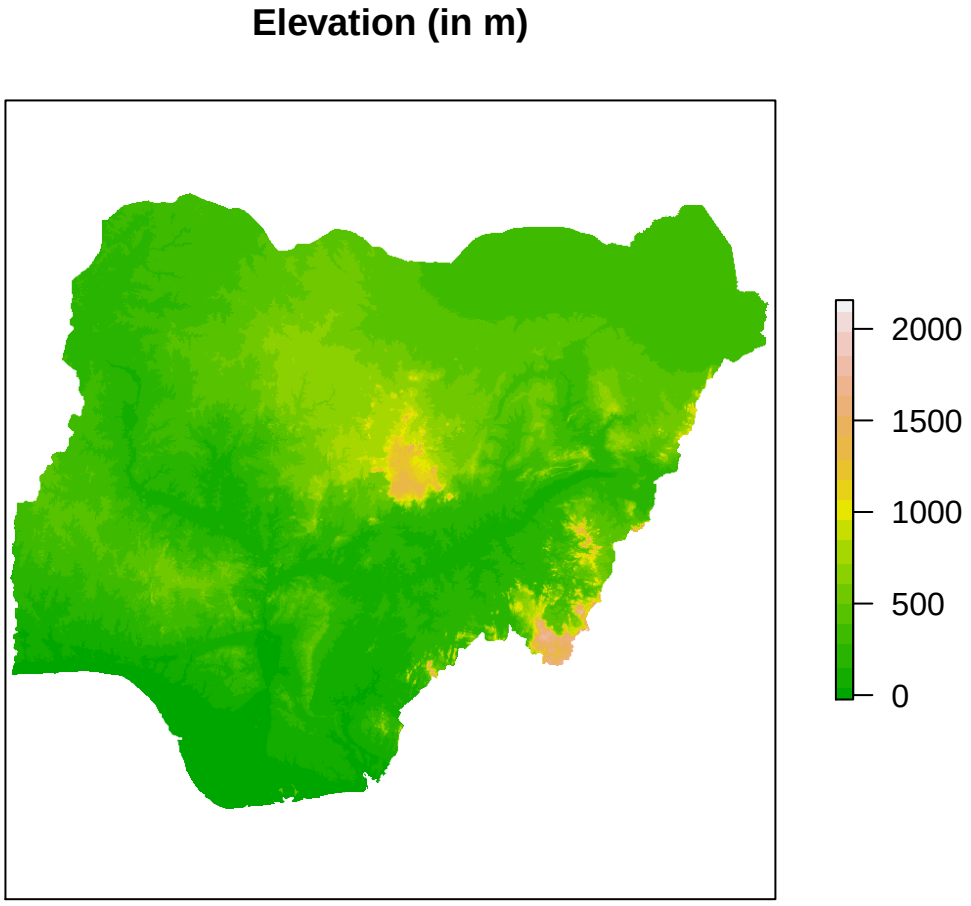
```
# data url
url <- "https://data.humdata.org/dataset/bd3b4ef6-0a96-4346-b229-2c410d9c6c6d/resource/3c6a77da-be1a-443

# download steps
download.file(url, dest = 'nga_flood.zip')
unzip('nga_flood.zip')

# read flood extent
fld <- shapefile("NGA_Flood_Modis_250_nasa.shp")

# plot
# boundary
plot(v0, lwd = 1,  axes = TRUE)
# flood extent
plot(fld, col='lightblue', border = 'transparent', add = TRUE)
# river
plot(rv, col='darkblue', add = TRUE, lwd = 0.5)
# title and legends
title(main = "Nigeria flood extent and river network")
```



Note that we can plot all the spatial layers together so far becuase because they have the same projection. Unlike desktop GIS software, R will not reproject the data on the fly. We need to manually reproject the data in a common projection system for any spatial analysis.

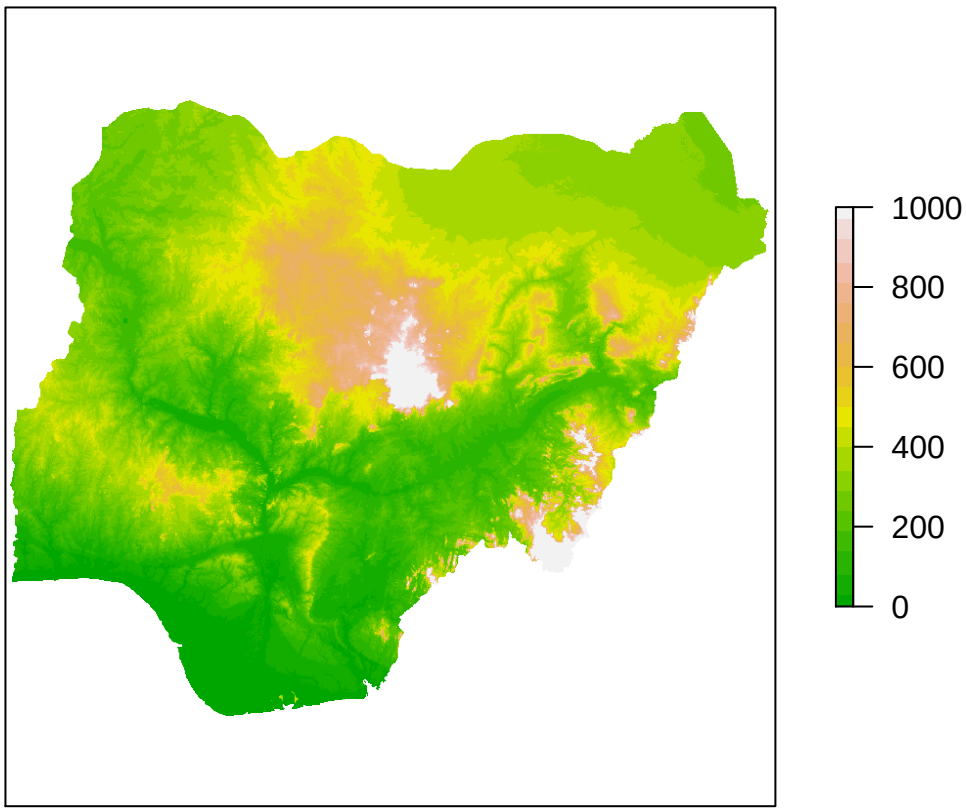
Plot Raster Plotting raster also follows the similar logic. Let's plot elevation of Nigeria. For some common geographical variables (elevation, climate), `getData` is a great source. First we will download the data.

```
alt <- getData('alt', country='NGA', mask=TRUE)
plot(alt, col = terrain.colors(20), axes=FALSE)
title(main = "Elevation (in m)")
```



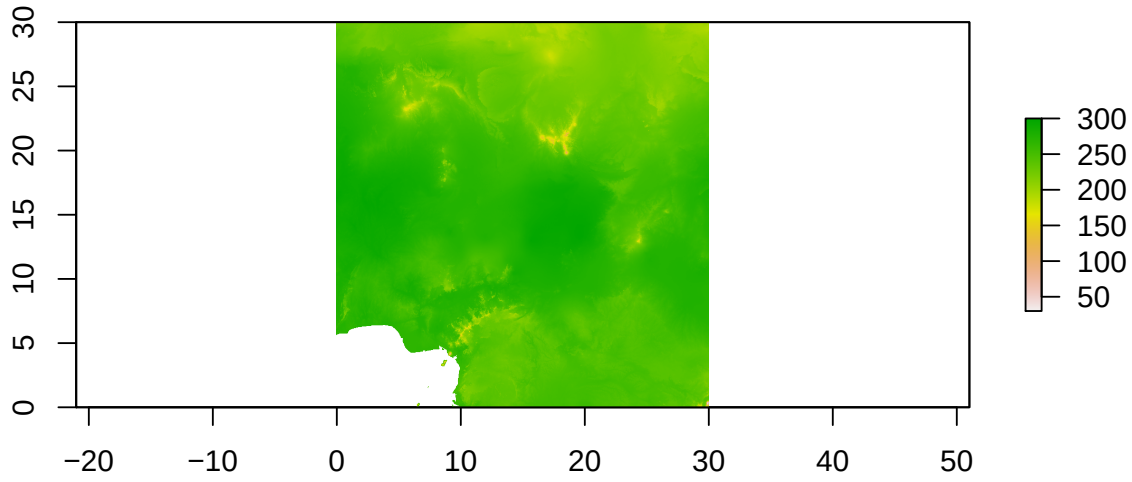
To improve the visualization, we can `limit` the higher and lower elevataion zones.

```
alt[alt < 0] = 0
alt[alt > 1000] = 1000
plot(alt, col = terrain.colors(20), axes=FALSE)
title(main = "Elevation (in m)")
```

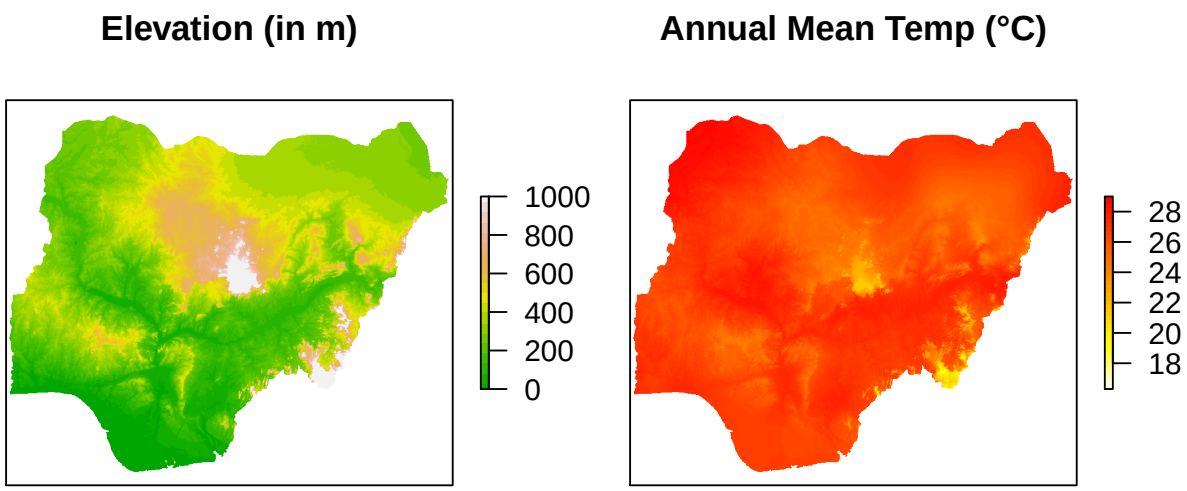


Plotting two raster layers We will show elevation and avergae annual temperature in the same plot.

```
# Download worldclim
wc <- getData('worldclim', var='bio', res=0.5, lon=8, lat=9)
# downloads all bioclim variables at 0.5 deg from https://www.worldclim.org/bioclim
# we are only interested in bio1 that is Annual Mean Temperature
b1 <- wc[[1]]
plot(b1)
```




```
# this is a tiled product, not masked to the country boundary.
# we can use the `crop` and `mask` function in the `raster` package.
b1 <- crop(b1, alt)
b1 <- mask(b1, alt)
b1 <- b1/10 # scale factor

# next plot elevation and annual mean temperature side-by-side
par(mfrow=c(1,2))
plot(alt, col = terrain.colors(20), axes=FALSE)
title(main = "Elevation (in m)")

plot(b1, col = rev(heat.colors(50)), axes=FALSE)
title(main ="Annual Mean Temp (°C)")
```



Ex 4. Rescale the temperature raster to remove the high (> 25) and low (< 15) values and then plot the raster again.

```
dev.off() # removes the mfrow settings
```

Adding Spatial *object to raster
Next we will add the rivers and administrative boundaries to the elevation raster plot

```
## null device
##           1
```

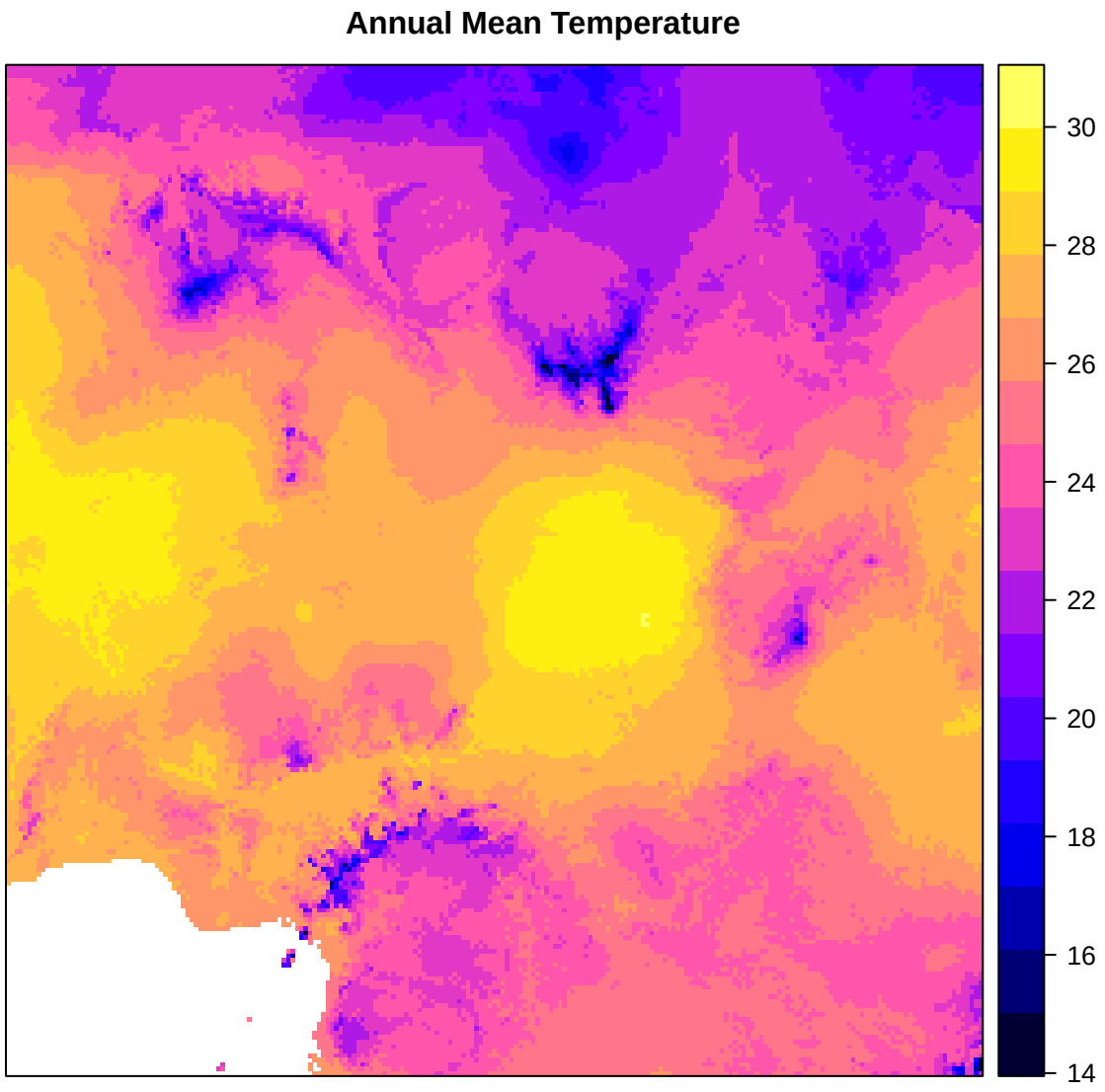
```
plot(alt, col = terrain.colors(20), legend = FALSE)
plot(v1, lwd = 0.5, border = "gray50", axes = TRUE, add = TRUE)
plot(rv, col='blue1', lwd = 2, add = TRUE)
title(main = "Rivers of Nigeria")
```

Using `spplot()`

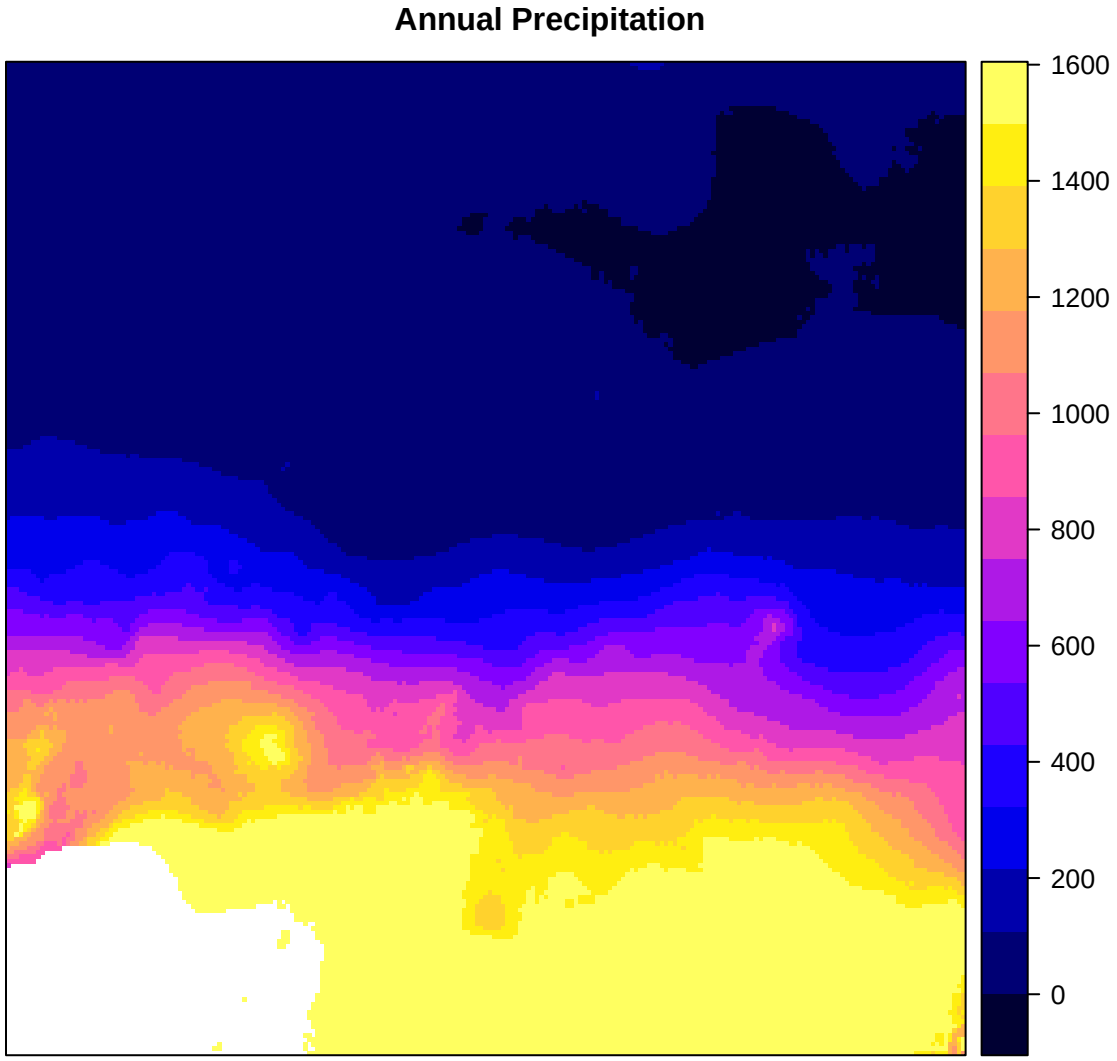
Like `raster::plot`, we can also use `sp::spplot` for different plotting applications.

```
# Use Annual Mean Temperature and Annual Precipitation
temp <- wc[["bio1_26"]] # band selection by name
prcp <- wc[["bio12_26"]]

# rescale values
# temperature
```

```
temp <- temp/10
temp[temp<15]<- 15
spplot(temp, main = list(label="Annual Mean Temperature",  cex = 1))
```

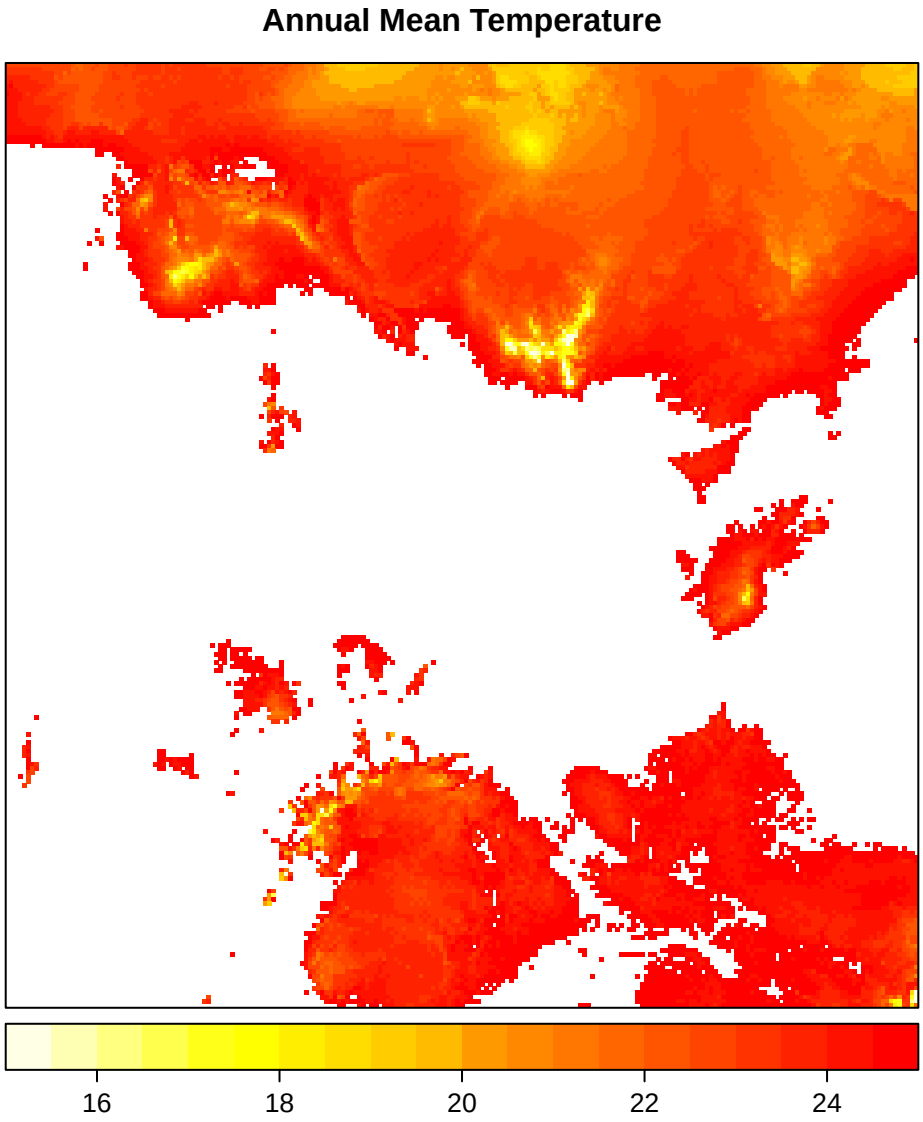



```
# precipitation
prcp[prcp > 1500] <- 1500
spplot(prcp, main = list(label="Annual Precipitation",  cex = 1))
```



Now to change the color ramp and change legend position:

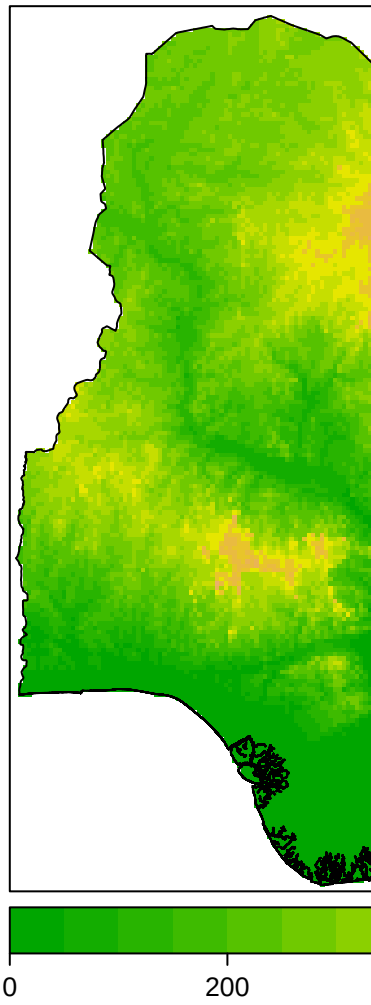
```
brks <- seq(15,25,0.5)
spplot(temp,
       at = round(brks, digits=2),
       col.regions = rev(heat.colors(length(brks))), colorkey = list(space = "bottom"),
       main = list(label="Annual Mean Temperature",  cex = 1))
```



Ex 5. Make similar changes to precipitation plot. Hint: the brks will take place at differnt sequence based on precipitation values

```
pols <- list("sp.lines", as(v0, 'SpatialLines'))
brks <- seq(0,1000,50)
spplot(alt,
       sp.layout=pols,
       at = round(brks, digits=2),
       col.regions = terrain.colors(length(brks)), colorkey = list(space = "bottom"),
       main = list(label="Elevation",  cex = 1))
```

Adding Spatial *object to raster

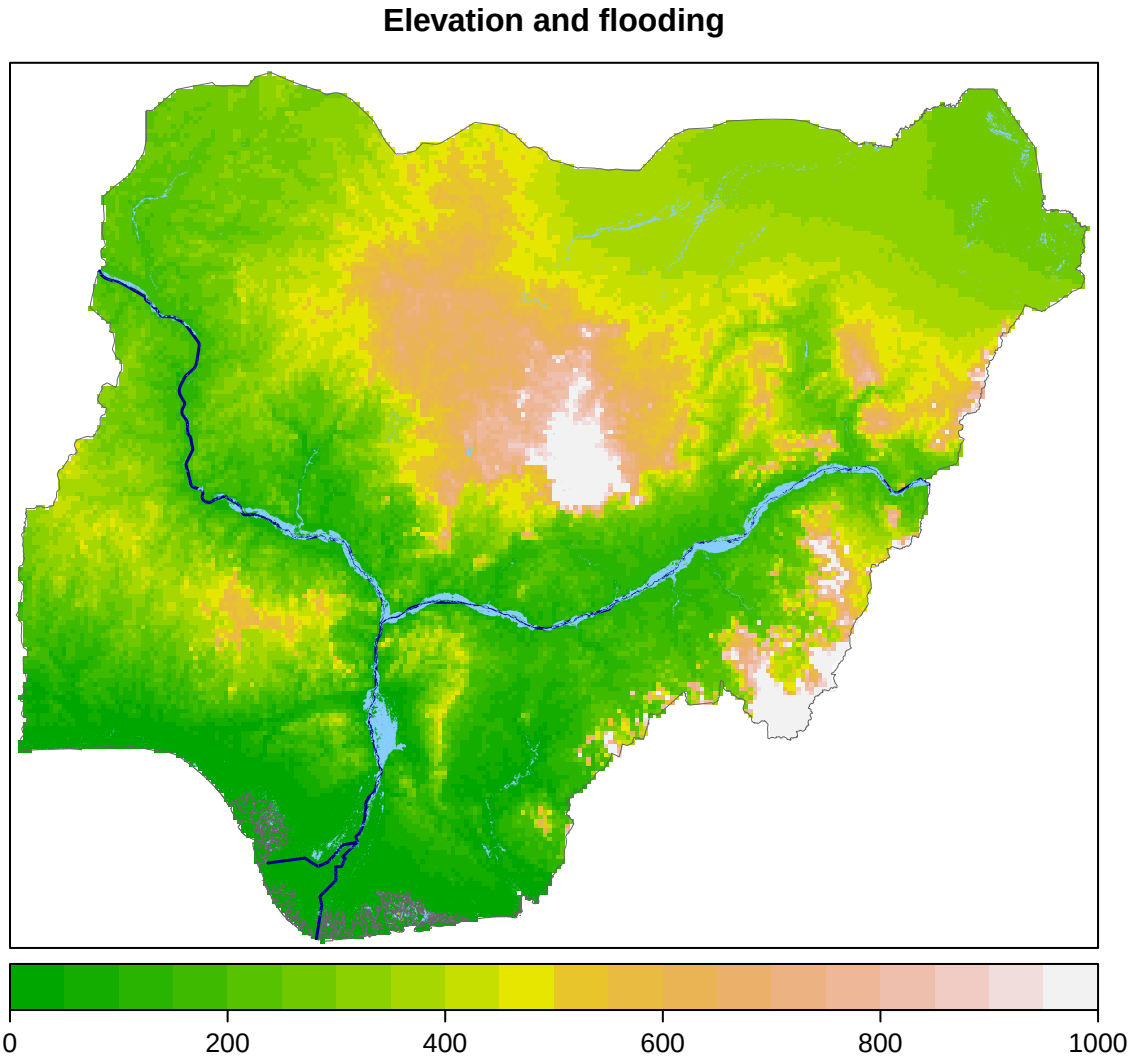


We will add the administrative boundary to the elevation raster plot in `spplot`.

See the difference with `plot` function. There is no use of `add=TRUE` argument.

You can also add multiple SpatialObjects to the plot.

```
pols1 <- list("sp.lines", as(v0, 'SpatialLines'), col = gray(0.4), lwd = 0.5)
pols2 <- list("sp.lines", as(rv, 'SpatialLines'), col = "darkblue", lwd = 1.5)
pols3 <- list("sp.polygons", as(fld, 'SpatialPolygons'),
              fill ='skyblue1',col="transparent", first = FALSE)

brks <- seq(0,1000,50)
spplot(alt,
       sp.layout=list(pols1, pols2, pols3),
       at = round(brks, digits=2),
       col.regions = terrain.colors(length(brks)), colorkey = list(space = "bottom"),
       main = list(label="Elevation and flooding",  cex = 1))
```



Ex 6. Save your favorite plot using the following: `png(filename = "your-favorite-plot", width = 250, height = 200, units = "mm", res = 300) plot... spplot... dev.off()`