

Lesson 2

Jan Verbesselt

October 15, 2013

Abstract

An intro to reading in spatial data: raster and vector data with R.

1 Today's learning objectives

- Read, write, and visualize spatial data (vector/raster) using a script
- Find libraries which offer spatial data handling functions
- Learn to include functions from a library in your script

2 Set Your Working Directory and Load Your Libraries

2.1 Set the Working Directory

Let's do some basic set up first.

- Create a folder which will be your working directory e.g. *Lesson2*
- Create an R script within that folder
- Set your working directory to the *Lesson2* folder
- Create a *data* folder within your working directory

In the code block below type in the file path to where your data is being held and then (if you want) use the `setwd()` (set working directory) command to give R a default location to look for data files.

```
R> setwd("yourworkingdirectory")
R> # This sets the working directory (where R looks for files)
R>
R> getwd() # Double check your working directory
R> datdir <- "data/" #This is an example of a Mac file path
R> # datdir<- "/data/" #This is an example of a PC file path
```

2.2 Load Libraries

Next we will load a series of R packages that will give the functions we need to complete all the exercises in lesson 1 and 2. For this exercise all of the packages should (hopefully) be already installed on your machine (?). We will load them below using the `library()` command. I also included some comments describing how we use each of the packages in the exercises.

```
R> #----Packages for Reading/Writing/Manipulating Spatial Data---
R> library(rgdal) # reading shapefiles and raster data
R> library(rgeos)
R> library(maptools)
R> library(spdep) # useful spatial stat functions
R> library(spatstat) # functions for generating random points
R> library(raster)
R> #---Packages for Data Visualization and Manipulation---
R> library(ggplot2)
R> library(reshape2)
R> library(scales)
```

3 Read, plot, and explore spatial data

3.1 Read in a Shapefile

The most flexible way to read in a shapefile is by using the `readOGR` command. This is the only option that will also read in the .prj file associated with the shapefile. NCEAS has a useful summary of the various ways to read in a shapefile: <http://www.nceas.ucsb.edu/scicomp/usecases/ReadWriteESRIShapeFiles> I recommend always using `readOGR()`.

Read OGR can be used for almost any vector data format. To read in a shapefile, you enter two arguments:

- `dsn`: the directory containing the shapefile (even if this is already your working directory)
- `layer`: the name of the shapefile, without the file extension

```
R> download.file('http://rasta.r-forge.r-project.org/kenyashape.zip',  
+               'data/kenyashape.zip')  
R> unzip('data/kenyashape.zip', exdir = datdir)  
R> kenya <- readOGR(dsn = datdir, layer = 'kenya')
```

3.2 Plotting the Data

Plotting is easy, use the `plot()` command:

```
R> plot(kenya)
```

Obviously there are more options to dress up your plot and make a proper map/graphic. A common method is to use `splot()` from the `sp` package. However I prefer to use the functions available in the `ggplot2` package as I think they are more flexible and intuitive. We will address maps and graphics later in the in the class. For now, let us move onto reading in some tabular data and merging that data to our shapefile (similar to the join operation in ArcGIS).

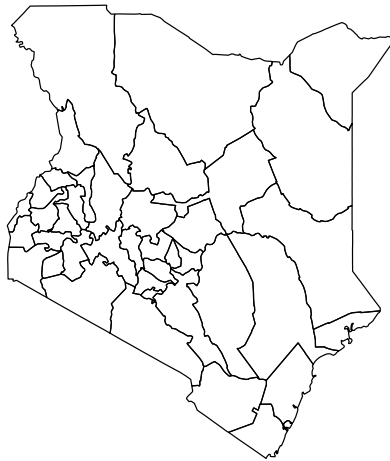


Figure 1: An example plot.

3.3 Exploring the Data within the vector file

We can explore some basic aspects of the data using `summary()` and `str()`. Summary works on almost all R objects but returns different results depending on the type of object. For example if the object is the result of a linear regression then summary will give you the coefficient estimates, standard errors, t-stats, R^2 , et cetera.

```
R> summary(kenya)
```

Object of class SpatialPolygonsDataFrame

Coordinates:

```
      min      max
x 33.908859 41.899078
y -4.678047  4.629333
Is projected: FALSE
proj4string : [+proj=longlat +ellps=clrk80 +no_defs]
```

Data attributes:

	ip89DId	ip89DName
Min.	:1010	Baringo : 1
1st Qu.:	:3050	Bugoma : 1
Median :	:5030	Busia : 1
Mean :	:5090	Elgeyo-Marakwet: 1
3rd Qu.:	:7060	Embu : 1
Max.	:8030	Garissa : 1
		(Other) :35

```
R> str(kenya,2)
```

Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots

```
..@ data      :'data.frame':      41 obs. of  2 variables:
..@ polygons  :List of 41
..@ plotOrder : int [1:41] 17 36 21 19 12 15 20 14 26 34 ...
..@ bbox      : num [1:2, 1:2] 33.91 -4.68 41.9 4.63
.. ..- attr(*, "dimnames")=List of 2
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
```

As mentioned above, the `summary()` command works on virtually all R objects. In this case it gives some basic information about the projection, coordinates, and data contained in our shapefile

The `str()` or structure command tells us how R is actually storing and organizing our shapefile. This is a useful way to explore complex objects in R. When we use `str()` on a spatial polygon object, it tells us the object has five 'slots':

1. *data*: This holds the data.frame
2. *polygons*: This holds the coordinates of the polygons
3. *plotOrder*: The order that the coordinates should be drawn
4. *bbox*: The coordinates of the bounding box (edges of the shape file)

5. *proj4string*: A character string describing the projection system

The only one we want to worry about is *data*, because this is where the `data.frame()` associated with our spatial object is stored. We access slots using the `@` sign.

```
R> #-----ACCESS THE SHAPEFILE DATA-----  
R> dsdat<-as(kenya, "data.frame") #extract the data into a regular data.frame  
R> head(dsdat)
```

	ip89DId	ip89DName
0	1010	Nairobi
1	2010	Kiambu
2	2020	Kirinyaga
3	2030	Muranga
4	2040	Nyandaura
5	2050	Nyeri

```
R> kenya$new<- 1:nrow(dsdat) #add a new column, just like adding data to a data.frame  
R> head(kenya@data)
```

	ip89DId	ip89DName	new
0	1010	Nairobi	1
1	2010	Kiambu	2
2	2020	Kirinyaga	3
3	2030	Muranga	4
4	2040	Nyandaura	5
5	2050	Nyeri	6