Applied Geo-Scripting: Lesson 1

Jan Verbesselt, Sytze de Bruyn, Loic Dutrieux, Ben De Vries, ...

October 8, 2013

Geo-scripting learning objectives

- Handle spatial data using a scripting language
- Read, write, and visualize spatial data (vector/raster) using a script
- Know how to find help (on spatial data handling functions)
- Solve scripting problems (debug, reproducible example, writing functions)
- Find libraries which offer spatial data handling functions
- Learn to include functions from a library in your script
- Apply learned concepts in a case study: learning how to address a spatial/ecological/applied case (e.g. detect forest changes, ocean floor depth analysis, bear movement, etc.) with a raster and vector dataset.

Today's topics

- Intro to basic concepts of applied scripting for spatial data
- Why geo-scripting?
- Course planning and practical issues
- Getting Us up to speed with R and loading 'rasta' package

RASTA: Reproducible and Applied Spatial and Temporal Analysis
https://r-forge.r-project.org/R/?group_id=1743

Why geo-scripting?

- Reproducible: avoid clicking and you keep track of what you have done
- Efficient: you can write a script to do something for you e.g. multiple times e.g. automatically downloading data
- Enable collaboration: sharing scripts, functions, and packages
- Good for finding errors i.e. debugging e.g. this course is fully writing with scripting languages (i.e. R and Latex).

What is a scripting language?

A scripting language or script language is a programming language that supports the writing of scripts, programs written for a special runtime environment that can interpret and automate the execution of tasks which could alternatively be executed one-by-one by a human operator. Different

from compiled languages like C/C++/Fortran. A scripting language is the

glue, between different commands, functions, and objectives without the need to compile it for each OS/CPU Architecture.

Different scripting languages for geo-scripting

The main scripting languages for GIS and Remote sensing currently are: R, Python (stand-alone or integrated within ArcGIS, QGIS), GRASS.

Aldo can you help here

Python versus R

- Python is a general purpose programming language
- R is particularly strong in statistical computing and graphics
- Installing libraries in Python is sometimes challenging
- Syntactic differences can be confusing
- There are many R and Python packages for spatial analyses and for dealing with spatial data
- Scripts in both languages can be combined

Aldo can you help here

Course set-up

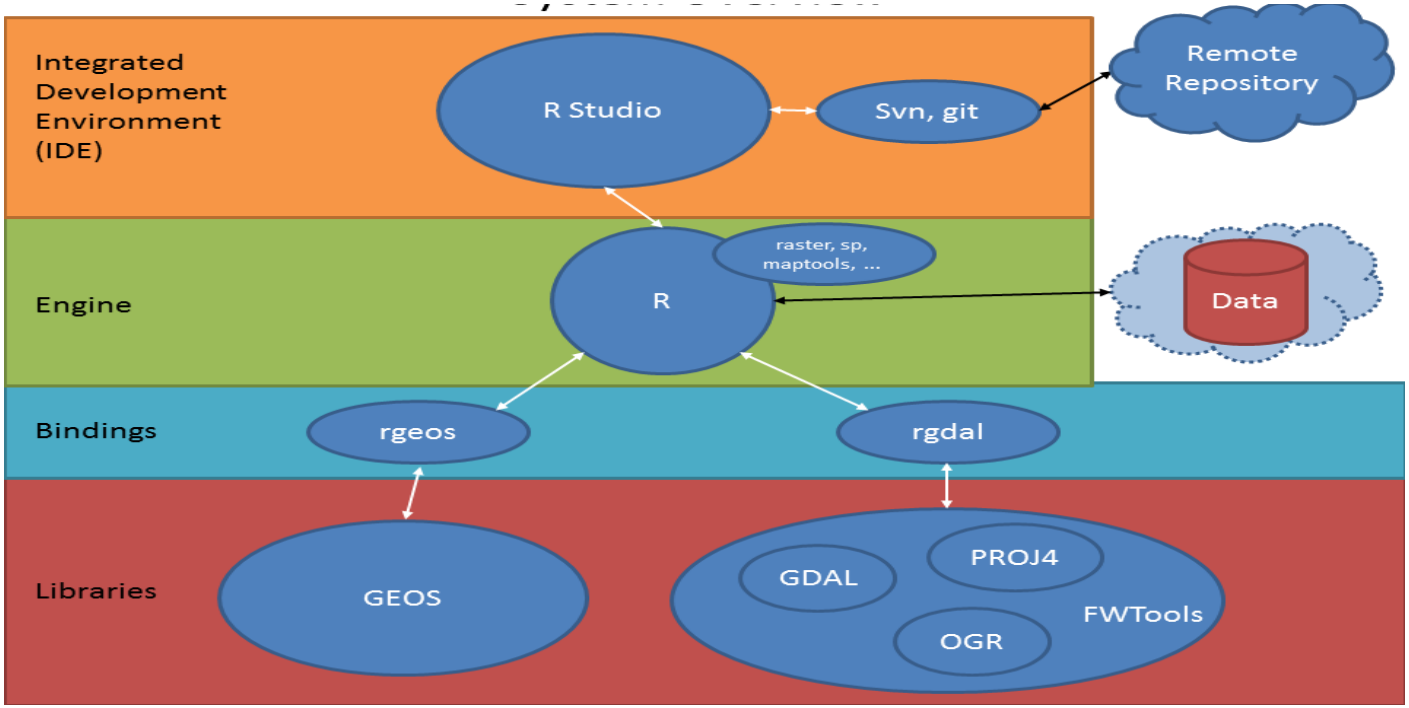

Figure : Course set-up

- SVN
- R libraries: rgeos, rgdal
- GDAL
- GEOS

Get Your R On

Getting started with Rstudio and the rasta package This preliminary section will cover some basic details about R. For this course we will use Rstudio as an IDE to write and run scripts. Open Rstudio! Now type the following script in the R console:

```
R> a <- 1
R> a

[1] 1
```

The first line you passed to the console created a new object named *a* in memory. The symbol '<-' is somewhat equivalent to an equal sign. In the second line you printed *a* to the console by simply typing it's name.

What is the class of this object?

Get Your R On

```
R> class(a)
[1] "numeric"
```

You now have requested the **class** attribute of *a* and the console has returned the attribute: **numeric**. R possesses a simple mechanism to support an object-oriented style of programming. All objects (*a* in this case) have a class attribute assigned to them. **R** is quite forgiving and will assign a class to an object even if you haven't specified one (as you didn't in this case). Classes are a very important feature of the **R** environment. Any function or method that is applied to an object takes into account its class and uses this information to determine the correct course of action.

Custom Functions

It is hard to unleash the full potential of R without writing your own functions. Luckily it's very easy to do. Here are some trivial examples:

```
R> add<-function(x){ #put the function arguments in () and the evaluatio
+   x + 1
+ }
R> add(3)
[1] 4
R> add(4)
[1] 5
R> # Set the default values for your function--
R> add<-function(x = 5){
+   x + 1
+ }
R> add()  #automatically evaluates x = 5
[1] 6
R> add(6) #but you can still change the defaults
[1] 7
```

That's about all there is too it. The function will generally return the result of the last line that was evaluated. However you can also use `return()` to specify exactly what the function will return.

Now, let's declares a new object, a new function, **newfunc** (this is just a name and if you like you can give this function another name). Appearing in the first set of brackets is an argument list that specifies (in this case) two names. The value of the function appears within the second set of brackets where the process applied to the named objects from the argument list is defined.

```
[1] 8
```

The rasta package

Now load the rasta package which you can download from:
https://r-forge.r-project.org/R/?group_id=1743

```
R> library(rasta) ## load the rasta library
R> #?rasta
R> #look at course content
```

Course overview and pdf's are available + Excercise.

Excercise needs to be handed in before 1030 the day after the lesson so that we can provide feedback on the excercise and discuss shortly.

Reading Data in and Out

The most common way to read in data is with the `read.csv()` command. Type ?read.table in your console for some other examples.

```
R> f <- system.file("extdata/kenpop89to99.csv", package="rasta")
R> mydat<-read.csv(f)
```

We can explore the data using the names(), summary(), head(), and tail() commands (we will use these frequently through out the exercise)

```
R> names(mydat)[1:3] #colunm names
```

```
[1] "ip89DId"   "ip89DName" "ADMIN3"
```

```
R> summary(mydat$Y89Pop)[1:3]
```

```
   Min. 1st Qu.  Median
  57960  222900  451500
```

```
R> head(mydat$Y89Births)[1:2]
```

```
[1] 42560 27720
```

What is the class of the *mydat*? We will go over ways to index and subscript data.frames later. Lets do a basic regression so you can see an example of a list.

Basic regression and example of a list

We use the `lm()` command to do a basic linear regression. The ~ symbol separates the left and right hand sides of the equation and we use '+' to separate terms and '*' to specify interactions. *Regress the Population in 1999 on the population and birthrate in 1989*

```
R> myreg<-lm(Y99Pop ~ Y89Births + Y89Brate, data = mydat)
R> myreg[c(1,8)]

$coefficients
(Intercept)   Y89Births    Y89Brate
502592.5928     38.0455 -14369.0931

$df.residual
[1] 45
```

Basic regression and example of a list

A regression object is an example of a list. We can use the `names()` command to see what the list contains. We can use the summary() command to get a standard regression output (coefficients, standard errors, et cetera) and we can also create a new object that contains all the elements of a regression summary.

```
R> names(myreg)[1:3]
[1] "coefficients" "residuals"    "effects"
R> myregsum <- summary(myreg)
R> names(myregsum)[1:2]
[1] "call"  "terms"
R> myregsum[['adj.r.squared']] #extract the adjusted r squared
[1] 0.8937546
R> myregsum$adj.r.squared # does the same thing
[1] 0.8937546
```

why is *myregsum* a list object? What is the advantage of a list?
That concludes our basic introduction to data.frames and lists. There is alot more material out on the web if you are interested. Later in the exercise we will look at data.frames in more detail.

Excercise Lesson 1: Write you own function

Write a function to visualise data and plots different variables
Submit a documented script to Using data with the 'rasta' package Submit your code?

This will be a test to see if your R scripting levels are ok to continue the course in the coming months.

More information

For more information about R please refer to the following links http://www.statmethods.net/index.html. This is a great website for learning R function, graphs, and stats. Also visit http://www.r-project.org/ and check out the Manuals i.e an introductions to R See also the book on Applied spatial Data analysis with R http://www.asdar-book.org/ (Bivand et al., 2013).

Bivand, R. S., Pebesma, E. J., & Rubio, V. G. (2013). Applied spatial data analysis with R, .