

Applied Geo-Scripting

Jan Verbesselt, et al.

September 25, 2013

1 Geo-scripting learning objectives

- Handle spatial data using a scripting language (e.g. R)
- Understand basic concepts of applied scripting for spatial data
- Learn to 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, flood mapping, ocean floor depth analysis, bear movement, etc.) with a raster and vector dataset.

2 Today's Learning objectives

- Understand basic concepts of applied scripting for spatial data
- Handle spatial data using a scripting language (e.g. R)

3 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 written with scripting languages (i.e. R and LaTeX).

4 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.

5 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), GRASS.

Sytze, Aldo, ... can you add more info here

6 Python versus R

Sytze can you help here...

7 Course set-up and planning

- R package: RASTA package https://r-forge.r-project.org/R/?group_id=1743
- Have look at the Reproducible and Applied Spatial and Temporal Analysis (RASTA) package (package content)
- Course set-up is that every lesson there will be a short introduction, followed by a tutorial and an exercise that needs to be handed in before the start of the next lesson.
- Course content and overview

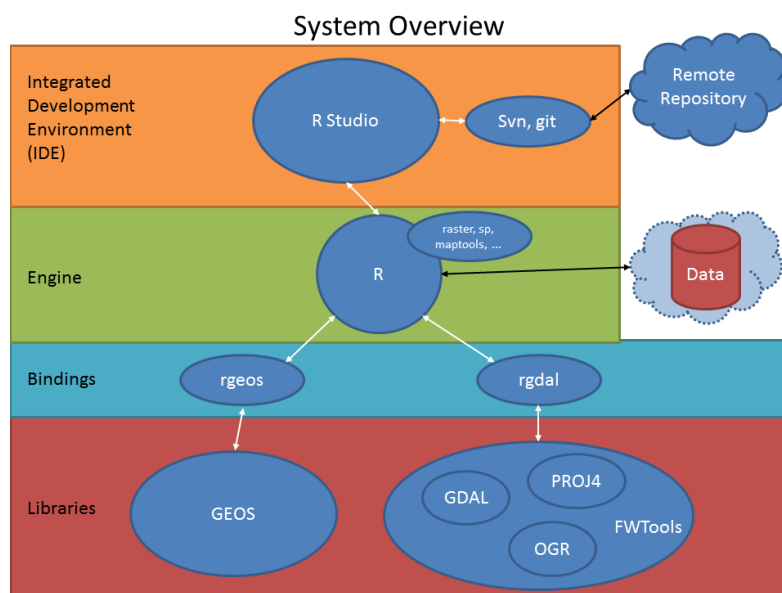


Figure 1: The graphical user interface to R

8 Get Your R On

This preliminary section will cover some basic details about R.

8.1 Data Structures

There are several ways that data are stored in R. Here are the main ones:

- **Data Frames** The most common format. Similar to a spread sheet. A `data.frame()` is indexed by rows and columns and store numeric and character data. The `data.frame` is typically what we use when we read in csv files, do regressions, et cetera.
- **Matrices and Arrays** Similar to `data.frames` but slightly faster computation wise while sacrificing some of the flexibility in terms of what information can be stored. In R a matrix object is a special case of an array that only has 2 dimensions. IE, an array is n-dimensional matrix while a matrix only has rows and columns (2 dimensions)
- **Lists** The most common and flexible type of R object. A list is simply a collection of other objects. For example a regression object is a list of: 1) Coefficient estimates 2) Standard Errors 3) The Variance/Covariance matrix 4) The design matrix (data) 5) Various measures of fit, et cetera.

We will look at examples of these objects in the next section

8.2 Reading Data in and Out

The most common way to read in data is with the `read.csv()` command. However you can read in virtually any type of text file. Type `?read.table` in your console for some examples. If you have really large binary data sets sometimes the `scan()` function is more efficient. Finally using the foreign package you can read in SPSS, STATA, Matlab, SAS, and a host of other data formats from other stat and math software.

Let's read in a basic csv file.

```
R> #-----READING DATA IN AND OUT-----  
R> mydat<-read.csv("data/kenpop89to99.csv")
```

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) #column names
```

```
[1] "ip89DId"      "ip89DName"    "ADMIN3"       "KEADMN3_ID"   "Y89Pop"  
[6] "Y89Births"    "Y89Brate"     "Y99Pop"       "Y99Births"    "Y99Brate"  
[11] "PopChg"       "BrateChg"
```

```
R> summary(mydat) #basic summary information
```

ip89DId	ip89DName	ADMIN3	KEADMN3_ID
Min. :1010	Kisii : 3	KISII : 2	Min. : 1.00
1st Qu.:3772	Kakamega : 2	BARINGO : 1	1st Qu.:12.75

Median :6010	Kericho : 2	BOMET : 1	Median :24.50
Mean :5207	Machakos : 2	BUNGOMA : 1	Mean :25.52
3rd Qu.:7052	Meru : 2	BUSIA : 1	3rd Qu.:35.25
Max. :8030	South Nyanza: 2	E. MARAKWET: 1	Max. :63.00
	(Other) :35	(Other) :41	

Y89Pop	Y89Births	Y89Brate	Y99Pop
Min. : 57960	Min. : 1680	Min. :22.64	Min. : 72380
1st Qu.: 222905	1st Qu.: 9350	1st Qu.:33.52	1st Qu.: 392545
Median : 451510	Median :18270	Median :37.38	Median : 629740
Mean : 619710	Mean :23719	Mean :37.03	Mean : 872928
3rd Qu.: 947500	3rd Qu.:39855	3rd Qu.:40.88	3rd Qu.:1384665
Max. :1476500	Max. :57460	Max. :51.01	Max. :2363120

Y99Births	Y99Brate	PopChg	BrateChg
Min. : 1760	Min. :19.03	Min. : -14.00	Min. : -38.00
1st Qu.:10870	1st Qu.:28.03	1st Qu.: 23.75	1st Qu.: -20.00
Median :21820	Median :31.01	Median : 33.50	Median : -14.00
Mean :27562	Mean :31.57	Mean : 47.73	Mean : -14.56
3rd Qu.:42140	3rd Qu.:36.36	3rd Qu.: 44.25	3rd Qu.: -6.75
Max. :69380	Max. :42.89	Max. :343.00	Max. : 0.00

R> head(mydat) #first 6 rows

ip89DId	ip89DName	ADMIN3	KEADMN3_ID	Y89Pop	Y89Births	Y89Brate	Y99Pop
1	1010 Nairobi	NAIROBI	41	1325620	42560	32.11	2085820
2	2010 Kiambu	KIAMBU	38	908120	27720	30.52	1383300
3	2020 Kirinyaga	KIRINYAGA	29	389440	10980	28.19	452180
4	2030 Muranga	MURANGA	36	862540	27940	32.39	737520
5	2040 Nyandaura	NYANDARUA	22	348520	12520	35.92	468300
6	2050 Nyeri	NYERI	26	607980	17540	28.85	644380
	Y99Births	Y99Brate	PopChg	BrateChg			
1	58700	28.14	57	-12			
2	36140	26.13	52	-14			
3	10840	23.97	16	-15			
4	16500	22.37	-14	-31			
5	13320	28.44	34	-21			
6	14340	22.25	6	-23			

R> tail(mydat) # last 6 rows

ip89DId	ip89DName	ADMIN3	KEADMN3_ID	Y89Pop	Y89Births	Y89Brate
43	7120 Uasin-Gishu	UASIN GISHU	13	443280	17900	40.38
44	7130 West-Pokot	WEST POKOT	5	224640	9440	42.02
45	8010 Bugoma	BUNGOMA	11	741940	34600	46.63
46	8020 Busia	BUSIA	16	425380	18640	43.82
47	8030 Kakamega	VIHIGA	21	1476500	57460	38.92
48	8030 Kakamega	KAKAMEGA	14	1476500	57460	38.92
	Y99Pop	Y99Births	Y99Brate	PopChg	BrateChg	

43	616240	22260	36.12	39	-11
44	309020	12940	41.87	38	0
45	1008080	43240	42.89	36	-8
46	547680	23440	42.80	29	-2
47	2011960	69380	34.48	36	-11
48	2011960	69380	34.48	36	-11

Write you own function to automatise a few tasks. E.g....