Introduction to the analysis of spatial data using R

Ana I. Moreno-Monroy

13-16 June 2017

Chapter 1: Introduction

- The **R**evolution
- R and R Studio
- Getting familiar with R studio
- Basic inspection and statistics
- Subsetting, selecting and creating variables

The Revolution

The R project was conceived in 1992 by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. Since the realese of the first beta version in 2000, it has grown exponentially

The Revolution

See the Google Trends website for more details

The Revolution: Advantages

- R is Open Source = R is not property of a corporation but of its users
- R is a highly valued skill (the pain pays off!)
- It is flexible. It is highly extensively (allows for user generated packages) and can be linked to other languages (e.g. C++, Fortran)
- It is far better are handling spatial information than Stata
- Reproducibility: Easy and reproducible reporting of statistical analysis (with R Markdown)

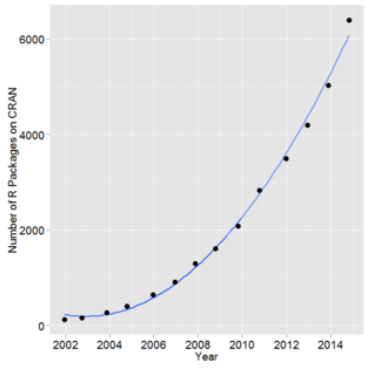


Figure 1: Number of contributed packages in CRAN

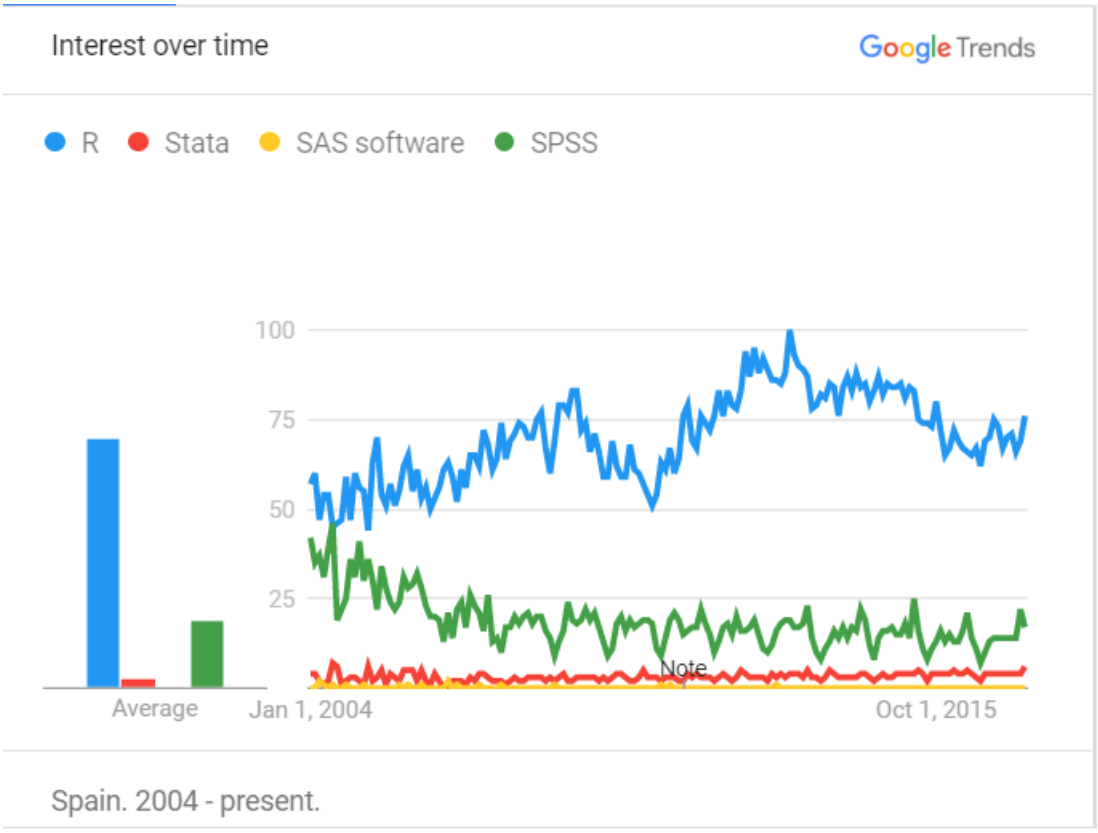


Figure 2: Google trends comparison of different statistical software

The Revolution: Some possibilities

- Automated FTP data downloading
- Getting and handling geolocation, commuting information and routing from Google's APIs (Application Programming Interfaces)
- Making (these!) slides and publishing them online in one click
- Having text and code for books and reports in one file (R Markdown) using the formatting possibilities of LaTex
- Making interactive maps and high-quality (3-D) graphs, and publishing them online
- Handling big data
- Constructing research inputs based on spatial variables (e.g. distance to nearest metro station) and outputs (spatial density functions, segregation indices, geographical weighted regressions, spatial statistics. . .)
- Developing web apps

Software

- R is a "free software environment for statistical analysis". You can get the latest version of R (3.4.0 - You Stupid Darkness) here
- As R works through the command line, it can be a bit intimidating
- R Studio is an Integrated Development Environment that facilitates the use of R by providing extra functionality
- Click here to download the latest version (RStudio Desktop 1.0.143). Note: Check R is installed (and running properly) before installing R Studio
- R is flexible for writing new functions based on existing ones
- Packages: An R package contains functions, compiled code, data and documentation. R incorporates base packages, which can be complemented with contributed packages. All available packages are listed in the Comprehensive R Archive Network (CRAN)

R Studio

- 4 panels: source editor and data viewer; command history and workspace browser; R Console; and file/help/package/plots
- Types of files: .R (similar to .do file); .RData (default when you press Save, will save all objects in the Global Environment); .Rmd (R Markdown – this presentation)
- Load projects, data or code using File options
- Some window options (e.g. Tools/install packages, Debug)

Getting help

- In the commmand line in the console, type ?+your query (similar to **help** in Stata)
- Googling question usually works
- Useful net resources: (GIS) StackExchange, StackOverFlow and R-Bloggers
- Check the R Journal for new developments

Getting familiar with R studio

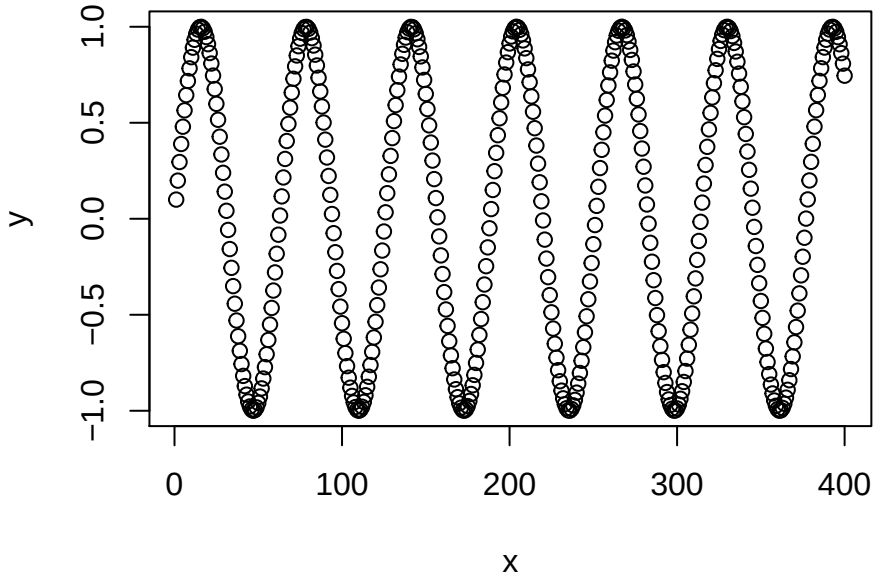
- Type **2 + 4** in the command line
- Type **library(ggmap)** in the command line. If you get an error it means the package **ggmap** is not intalled. Install it by clicking on Tools/Install Packages on R studio, or run

```
pkgs<-c("ggmap")
install.packages(pkgs)
```

- It is recommended to update packages regularly using the **update.packages** function
- Type **sessionInfo()** to get general information about versions and packages installed

Assigning and plotting

```
x <- 1:400
y <- sin(x/10)
plot (x,y)
```



Getting familiar with R studio

- Create a new R Script (File/New File/R Script). Write a short description after #
- Write a line of code to assign an object called "data" the value of the built-in data frame `USAarrests` and verify its class

```
data = USArrests
class(data)
```

```
## [1] "data.frame"
```

- From the .R document, press CTRL + Enter at the end of each line to execute it. Select all and press the "Run" button to execute all lines
- Type "USArrest" in the command line and press Enter. What happens?
- Try using <- instead of =
- Type "data = US" and immediately press the tab key after the "S". What happens?

- Get help: Type "?USArrests" in the command line

Basic inspection and statistics

- Try clicking on table on the right-hand side of the Environment tab
- View the first part of an object

```
head(data)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

Structure of object

- Compact display of an object's basic structure

```
str(data)
```

```
## 'data.frame':    50 obs. of  4 variables:
##  $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
##  $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
##  $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
##  $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

Summary statistics

- Basic summary statistics (Stata's **summarize**)

```
summary(data)
```

```
##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

Basic inspection and statistics

- Column total sum

```
colSums(data)
```

```
##   Murder  Assault UrbanPop     Rape
##    389.4   8538.0   3277.0   1061.6
```

- Mean of a variable

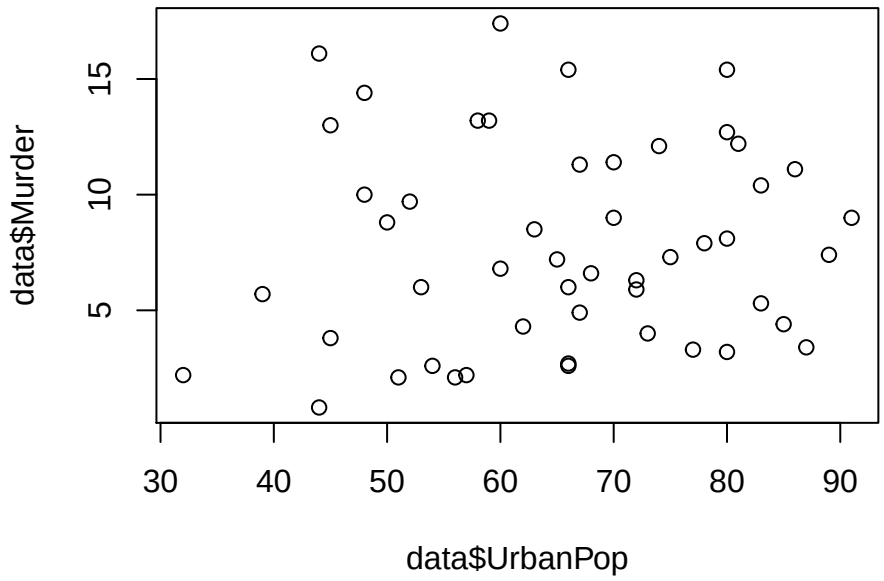
```
mean(data$UrbanPop)
```

```
## [1] 65.54
```

- Type ?sd in the command line. Calculate the standard deviation of the first variable and compare with corresponding value in the summary statistics table above

Simple plots

```
plot(data$UrbanPop, data$Murder)
```



Subset data

- By rows

```
data1<-data[1:20,]
```

- By columns

```
data2<-data[,3:4]
```

- By observation

```
data2<-data[1,1]
```

Removing unwanted elements and cleaning workspace

- Remove elements from workspace.

```
data2<-data1
rm(data1, data2)
```

- To remove all, type rm(list = ls()).
- Note that R (unlike Stata) automatically overwrites an existing object if it is given an the same name
- To save space and gain in reproducibility, do not save objects that can be quickly generated with a line of code from a parent object (save the code leading to the new object instead!)

Selecting and creating variables

- In console start with database name, add $ and press Tab key

```
var<-data$Murder
```

- Creating a new variable and adding it to the data frame

```
data$all_crimes<-data$Murder + data$Assault + data$Rape
```

- To create a character vector, name the variable and write words in parenthesis inside c()

```
answer<-c("FALSE", "FALSE", "TRUE")
names<-c("UrbanPop", "Murder")
```

- R can sum strings satisfying a condition without having to convert them

```
sum(answer=="FALSE")
```

```
## [1] 2
```

Subset data satisfying a condition

- Above/below value of variable: first create TRUE/FALSE variable using the condition, then grab only observations satisfying the condition (TRUE) using brackets[]

```
sel<-data$UrbanPop>mean(data$UrbanPop)
table(sel)
```

```
## sel
## FALSE  TRUE
##    22    28
```

```
data1<-data[sel,]
```

- Another way: using the **subset** base function (all in one line)

```
data2<-subset(data, UrbanPop>mean(UrbanPop))
dim(data2)
```

```
## [1] 28  5
```

Subset data satisfying a condition

- Subset by column names

```
keep<-c("UrbanPop", "Murder")
data3<-data[,colnames(data) %in% keep]
```

- Equivalently, drop unwanted columns using ! (IS NOT)

```
drop<-c("Rape", "Assault")
data4<-data[,!colnames(data) %in% drop]
```

Handling data using the dplyr package

dplyr is a good package for handling tabulated data. We can do the same operations as above using the **mutate**, **filter** and **select** functions

```
library (dplyr)
data<-USArrests
data<-mutate(data, all_crimes=(Murder + Assault + Rape))
data_2 <- filter(data, UrbanPop>mean(UrbanPop))
dim(data_2)
```

```
## [1] 28  5
```

```
data_3 <- select(data, UrbanPop, Murder)
dim(data_3)
```

```
## [1] 50  2
```

```
data_4 <- select(data, -Rape, -Assault)
dim(data_4)
```

```
## [1] 50  3
```

Downloading and unzipping files

Files can be downloaded from the internet into a local folder using the **download.file** and **unzip** functions of the **utils** package.

The first step is to verify the current working directory, set a new working directory where the files are going to be saved (if necessary) and creating a new folder where the files will be saved:

```
getwd()
setwd("C:/Your_local_path")
if(!dir.exists("data")){dir.create("data")}
```

After identifying the website containing the path to the ".zip" file, we can access it, download it, unzip it and save it in our local directory using the following lines:

```
library(utils)
download.file(url = "http://www.rigis.org/geodata/bnd/muni97d.zip",
              destfile = "data/muni97d.zip")
unzip(zipfile = "data/muni97d.zip", exdir = "data")
```

The unzipped files should be now in the local directory!