Introduction to R

Siria Angino
Federica Romei

24/09/2013

What is R

0.1 The R environment

[1]R is an integrated suite of software facilities for data manipulation, calculation and graphical display. Among other things it has an effective data handling and storage facility, a suite of operators for calculations on arrays, in particular matrices,a large, coherent, integrated collection of intermediate tools for data analysis, graphical facilities for data analysis and display either directly at the computer or on hardcopy, and a well developed, simple and effective programming language (called "S") which includes conditionals, loops, user defined recursive functions and input and output facilities. (Indeed most of the system supplied functions are themselves written in the S language.) The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software. R is very much a vehicle for newly developing methods of interactive data analysis. It has developed rapidly, and has been extended by a large collection of packages. However, most programs written in R are essentially ephemeral, written for a single piece of data analysis.

0.2 Where Can I get R?

The beauty of R is that it's shareware, so it's free to anyone. To obtain R for Windows (or Mac) go to The Comprehensive R Archive Network (CRAN) at http://www.r-project.org. Just download the executable file, and it will install itself.

0.3 Help

R has an inbuilt help facility similar to the man facility of UNIX. To get more information on any specific named function, for example solve, the command is

[1]Notes are taken from "An introduction to R" and "R intro"

```
> help(solve)
```

An alternative is

```
> ?solve
```

For a feature specified by special characters, the argument must be enclosed in double or single quotes, making it a "character string": This is also necessary for a few words with syntactic meaning including if, for and function.

```
> help("[[")
```

Either form of quote mark may be used to escape the other, as in the string "It's important". Our convention is to use double quote marks for preference. On most R installations help is available in HTML format by running

```
> help.start()
```

which will launch a Web browser that allows the help pages to be browsed with hyperlinks. On UNIX, subsequent help requests are sent to the HTML-based help system. The "Search Engine and Keywords" link in the page loaded by help.start() is particularly useful as it is contains a high-level concept list which searches though available functions. It can be a great way to get your bearings quickly and to understand the breadth of what R has to offer. The `help.search` command (alternatively ??) allows searching for help in various ways. For example,

```
> ??solve
```

0.4 R commands, case sensitivity

Technically R is an expression language with a very simple syntax. It is case sensitive as are most UNIX based packages, so A and a are different symbols and would refer to different variables. The set of symbols which can be used in R names depends on the operating system and country within which R is being run (technically on the locale in use). Normally all alphanumeric symbols are allowed (and in some countries this includes accented letters) plus "." and"_", with the restriction that a name must start with "." or a letter, and if it starts with "." the second character must not be a digit.

Elementary commands consist of either expressions or assignments. If an expression is given as a command, it is evaluated, printed (unless specifically made invisible), and the value is lost. An assignment also evaluates an expression and passes the value to a variable but the result is not automatically printed.

Commands are separated either by a semi-colon (";"), or by a newline. Elementary commands can be grouped together into one compound expression by braces (`'{' and '}'`). Comments can be put almost anywhere, starting with a hashmark ("#"), everything to the end of the line is a comment. If a command is not complete at the end of a line, R will give a different prompt, by default

```
+
```

on second and subsequent lines and continue to read input until the command is syntactically complete. This prompt may be changed by the user. We will generally omit the continuation prompt and indicate continuation by simple indenting. Command lines entered at the console are limited3 to about 4095 bytes (not characters).

0.5 Useful Command

R saves any object you create. To list the objects you have created in a session use either of the following commands:

```
> objects ()
> ls()
```

To remove all the objects in R type:

```
rm(list=ls(all=TRUE))
```

Quitting. To quit R type:

```
> q()
```

Packages. R has many useful add on components that are called packages. We will use a few packages in the practice session here. To load a package you simply type:

```
> library(packagename)
```

We will use this command shortly.

0.6 Vectors and Matrices

The way to assign values to a vector or matrix is to use the `<- command`. So to create a vector a with the specic values of 3, 4, and 5:

```
>a <- c(3,4,5)
```

Or to create a 10 by 1 vector of 1's:

```
> rep(1, 10)
```

To create a 4 by 4 matrix b with values of 3:

```
> b <- matrix(3, ncol=4,nrow=4)
```

To change specic values, you can index a vector a by the nth element, use a[n], to index a n by k matrix b for the value xij , where i represents the row and j represents the column, use the following syntax b[i,j]. If you want to select all values of the j th column of b you can use b[,j]. For example, to return the 3rd column of matrix b, type:

```
> b[,3]
```

To get the 2nd row of b:

```
> b[2,]
```

The assignment command is also used when performing basic vector and matrix operations. For example, to assign vector a the sum of two vectors b and c, we type:

```
> a <- b + c
```

All other arithmetic operations, such as +, -, *, / and log work similarly. Note that simple multiplication by * performs multiplication element by element, while multiplication. Some other useful matrix functions include: To take the transpose of b:

```
> t(b)
```

To take the inverse of b:

```
> solve(b)
```

To obtain the length of a vector a:

```
> length(a)
```

To obtain the dimension of a matrix:

```
dim(b)
```

To take the sum of a vector:

```
> sum(a)
```

To take the mean of a vector:

```
> mean(a)
```

To take the variance of a vector:

```
> var(a)
```

Or take the standard deviation with the sd command.

Practice 1

We clear the workspace:
`rm(list=ls())`
We upload the packages:
`library(foreign)`
We set the directory:
`setwd(" /Desktop/ProvaR")` If you'd like to know which is the directory you are working on you can type:
`getwd()`
Let's upload "Ceosal1" dataset:
`ceosal1=read.dta('ceosal1.dta')`
If you type
`summary(filename)`
R will show some statistics (such as the mean) of all the variables in the dataset. If you type
`name(filename)`
R will show you the name of all the variables in the dataset.

Variables

If you'd like to use a variable in the dataset you can type:
ceosal1$roe
A useful facility would be somehow to make the components of a list or data frame temporarily visible as variables under their component name, without the need to quote the list name explicitly each time. The attach() function takes a "database" such as a list or data frame as its argument. Thus suppose lentils is a data frame with three variables $lentils\$u$, $lentils\$v$, $lentils\$w$. The attach
`> attach(lentils)`
places the data frame in the search path at position 2, and provided there are no variables u, v or w in position 1, u, v and w are available as variables from the data frame in their own right. At this point an assignment such as
`> u <- v+w`
does not replace the component u of the data frame, but rather masks it with another variable u in the working directory at position 1 on the search path. To make a permanent change to the data frame itself, the simplest way is to resort once again to the $ notation:
$> lentils\$u < -v + w$
However the new value of component u is not visible until the data frame is detached and attached again. To detach a data frame, use the function
`> detach()`
More precisely, this statement detaches from the search path the entity currently at position 2.

Thus in the present context the variables u, v and w would be no longer visible, except under the list notation as *lentils$u* and so on. Entities at positions greater than 2 on the search path can be detached by giving their number to detach, but it is much safer to always use a name, for example by detach(lentils) or detach("lentils").

Hypothesis Test

A friend of yours is setting up a transport firm and he claims that transport firms have the same roe as the non transport one. You are convinced that on average their roe is less than non-transport ones. Can you test this hypothesis?

First of all you should notice testing this hypothesis is equivalent to test:

$$H_0 : \ \Delta \, roe = 0$$

$$H_0 : \ \Delta \, roe < 0$$

where $\Delta \, roe \equiv roe_{transport} - roe_{not_transport}$. Under regular conditions:

- Observation i.i.d;
- $E(\Delta \, roe) < \infty$
- $Var(\Delta \, roe) < \infty$

we can apply CLT, i.e.:

$$\Delta \, \hat{roe} \xrightarrow{D} N(0,1)$$

where $\Delta \, \hat{roe} = \frac{\hat{roe}_t - \hat{roe}_n t}{\sqrt{Se(roe_t)^2 + Se(roe_{nt})^2}}$.

In R you can type:

```
mean(roe[utility==1],na.rm="true")
[1] 11.40556
```

and you will have the mean conditioned on $utility = 1$. To perform the test we can write:

```
H= (mean(roe[utility==1]) -mean(roe[utility==0]))/
  (sqrt((sd(roe[utility==1]))^2/(length(roe[utility==1])) +sd(roe[utility==0])^2/(length(roe[uti
  >H
[1] -7.855254
```

Since $|H| > 1.96$ we can reject the null hypothesis. Transport firms have a lower roe than non transport ones.

Graph and Regression

A friends of yours has been hired in a company that has a very high roe as ceo. He is willing to know if this will have some consequences on his wage. Since he knows you are taking a course in Econometric he asks to you.

We can graph the ceo salary against the roe using the command either

```
plot(roe,salary)
```

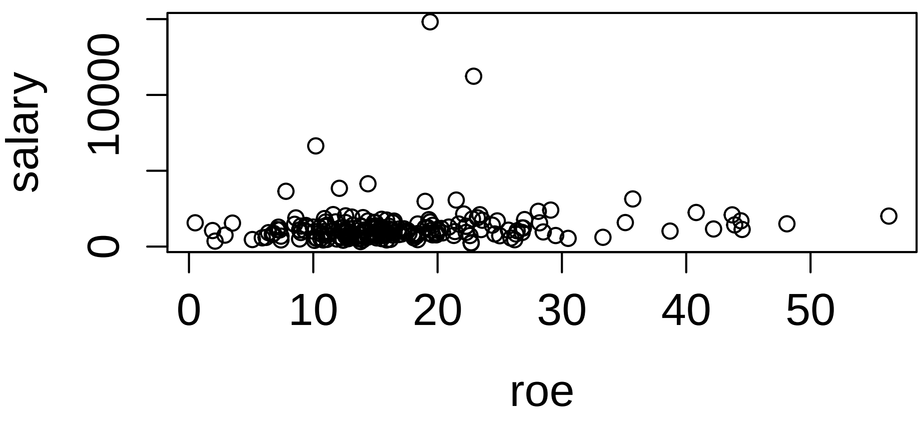

Figure 1: Figure with plot

or

```
abline(lm(salary,roe))
```

if we want the regression line plotted together with the data.

Effect seems to be positive. To be sure we can run a regression:

$$salary = \beta_0 + \beta_1 roe$$

If you are willing to see only the coefficients, β_0 and β_1, you can type:

```
> lm(salary~roe)

Call:
lm(formula = salary ~ roe)

Coefficients:
(Intercept)          roe
      963.2         18.5
```

Otherwise you can write:

```
> summary(lm(salary~roe))

Call:
lm(formula = salary ~ roe)

Residuals:
    Min      1Q  Median      3Q     Max
-1160.2  -526.0  -254.0   138.8 13499.9

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   963.19     213.24   4.517 1.05e-05 ***
roe            18.50      11.12   1.663   0.0978 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*'0.05 '.' 0.1  '' 1

Residual standard error: 1367 on 207 degrees of freedom
Multiple R-squared:  0.01319,Adjusted R-squared:  0.008421
F-statistic: 2.767 on 1 and 207 DF,  p-value: 0.09777
```

to display more informations on the regression.

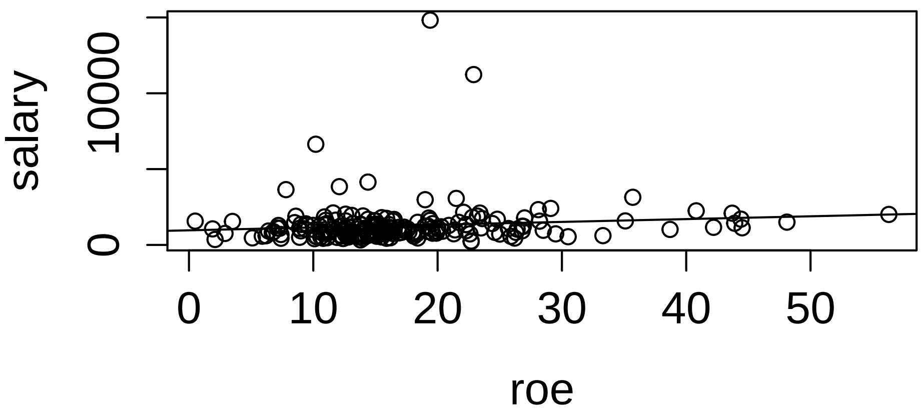

Figure 2: Figure with abline