An introduction to R programming language

Hristos Tyralis, Civil Engineer,
Msc in Statistics and Operations Research, PhD candidate
Department of Water Resources and Environmental Engineering
School of Civil Engineering
National Technical University of Athens, Greece
(montchrister@gmail.com, itia.ntua.gr/en/)

Lecture given at the Naresuan University (NU)
Phitsanulok, 17 November 2011
Presentation outline

1. Introduction
2. Examples
3. Help
4. More examples
5. Objects
6. Operators
7. Vectors and arrays
8. Graphics
9. Last but not least
10. Bibliography

This presentation will be available online at
1.1 Introduction

- The presentation is about R under windows
- R is a system for statistical analyses and graphics.
- R is both a software and a language.
- Its development and distribution are carried out by several statisticians known as the R Development Core Team
1.2 Introduction

- R is freely distributed
- The files needed to install R are distributed from the internet site of the Comprehensive R Archive Network (CRAN).
- R journal is the refereed journal for the R project.
1.3 Introduction

- R has many functions for statistical analyses and graphics
- The R language allows the user to combine in a single program different statistical functions
- The results from a statistical analysis are displayed on the screen
- A prominent feature of R is its flexibility
- R stores these results in an object
- For example a function that computes the ratio \((\text{standard deviation})/\text{mean}\) of a sample

\[
\text{ratiosm <- function(x) }
\text{ \{ }
\text{ sd(x)/mean(x) }
\text{ \}}
\]

- \text{sd(x)} is a function, \text{mean(x)} is another function and \text{ratiosm(x)} is the new function
1.4 Introduction

- R is an interpreted language, not a compiled one, meaning that all commands typed on the keyboard are directly executed without requiring to build a complete program like in most computer languages (C, Fortran, Pascal, ...)
- R's syntax is very simple and intuitive
- When R is running, variables, data, functions, results, etc, are stored in the active memory of the computer in the form of objects which have a name
- The user can do actions on these objects with operators (arithmetic, logical, comparison, ...) and functions (which are themselves objects)
- Pay attention: Everything in R is an object. Even to define a vector we use an object
2.1 Examples

- assign operator: <-

```r
> n <- 10
> n
[1] 10
```
this is equivalent to the following:
```r
> n = 10
> 10 -> n
```

- R is case sensitive
```r
> a <- 5
> A <- 10
> a
[1] 5
> A
[1] 10
```
2.2 Examples

- sum of two numbers: +

```r
> n <- 10 + 2
> n
[1] 12
```

- just the sum of two numbers:

```r
> 5 + 3
[1] 8
```

- some more examples of assignment

```r
> name <- "Carmen"; n1 <- 3
> name
[1] "Carmen"
> n1
[1] 3
```
2.3 Examples

- How can I make a list of the workspace objects?: `ls`

  ```r
  > ls()
  [1] “n” “n1” “name”
  
  Be careful, `ls` is a function. Here it has not any arguments.
  ```

- Now we give it an argument:

  ```r
  > ls(pat = “m”)
  [1] “name”
  ```

- Another list function: `ls.str`

  ```r
  > ls.str()
  n : num 10
  n1 : num 3
  name : chr “Carmen”
  ```
2.4 Examples

- How to remove all objects?: `rm`

```r
> rm(list=ls())
```
3. Help

- R html-help
- It also works on the R environment
4.1 More examples

- Object attributes?: `mode`, `length`

```r
> x <- 1
> mode(x)
[1] "numeric"
> length(x)
[1] 1
```

- More attributes:
```r
> A <- "Gomp" ; compar <- TRUE ; z <- 1i
> mode(A)
[1] "character"
> mode(compar)
[1] "logical"
> mode(z)
[1] "complex"
```
4.2 More examples

- Big numbers:

```r
> N <- 2.1e23
> N
[1] “2.1e+23”
> x <- 5/0
> x
[1] “Inf”
> exp(-x)
[1] 0
```
5.1 Objects

- Objects representing the data:
  - vector
  - factor
  - array
  - matrix
  - data frame
  - ts
  - list

- In this lecture we work with vectors and matrices
5.2 Objects

- Vector example:

```r
> rm(list=ls())
> x <- 1:5
> x
[1] 1 2 3 4 5
> 1:10-1
[1] 0 1 2 3 4 5 6 7 8 9
> 1:(10-1)
[1] 1 2 3 4 5 6 7 8 9
> seq(1,5,0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> seq(length=9,from=1,to=5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> c(1 , 1.5 , 2 , 2.5 , 3 , 3.5 , 4 , 4.5 , 5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> rep(1,5)
[1] 1 1 1 1 1
> sequence(4:6)
[1] 1 2 3 4 1 2 3 4 5 1 2 3 4 5 6
```
5.3 Objects

- Normal distribution:

\[
> a <- \text{rnorm}(1000, \text{mean}=0, \text{sd}=1)
\]
Simulates 1000 numbers from a standard normal distribution
\[
> \text{hist}(a)
\]
See the histogram of a
\[
> a <- \text{rnorm}(1000)
\]
Simulates 1000 numbers from a standard normal distribution again. \text{mean} = 0 and \text{sd} = 1 are the default argument values of the function \text{rnorm}
\[
> \text{hist}(a)
\]
\[
> a <- \text{rnorm}(1000, \text{mean} = 10, \text{sd} = 5)
\]
Simulates from a normal distribution. See the histogram again.
\[
> \text{hist}(a)
\]

- In \text{rnorm} \text{r} denotes random numbers from a normal distribution. There are also the density function \text{dnorm}, the cumulative probability function \text{pnorm} and the value of the \text{q} quantile \text{qnorm}. 
5.4 Objects

- Probability distributions in R:
  - beta  beta
  - binomial  binom
  - Cauchy  caushy
  - chi-squared  chisq
  - exponential  exp
  - F  f
  - gamma  gamma
  - geometric  geom
  - hypergeometric  hyper
  - log-normal  lnorm
  - logistic  logis

and other distributions
6.1 Operators

- Arithmetic operators
  - +
  - -
  - *
  - /
  - ^

- Logical operators
  - <
  - >
  - <=
  - >=
  - == (for exact equality)
  - != (for inequality)
6.2 Operators

- Some operations

```r
> x <- 0.5 ; y <- 0.7
> 0 < x & x < 1
[1] TRUE
> 0 < x < 1  # wrong expression
Error: unexpected 'x' in "0 < x <"
> x <- 1:3 ; y <- 1:3
> x == y
[1] TRUE TRUE TRUE
> identical(x, y)
[1] TRUE
```
7.1 Vectors and arrays

- Some operations

```r
> x <- 1:5
> x[3]
[1] 3
> x[3] <- 20
> x
[1] 1 2 20 4 5
> i <- c(1,3)
> x[i]
[1] 1 20
> x <- matrix(1:6, 2, 3)
> x
[,1] [,2] [,3]
[1,]  1  3  5
[2,]  2  4  6
> x[,3] <- 21:22
> x
[,1] [,2] [,3]
[1,]  1  3  21
[2,]  2  4  22
```
7.2 Vectors and arrays

- Some operations

```r
> x <- 1:10
> x[x>=5] <- 20
> x
[1] 1 2 3 4 20 20 20 20 20 20
> x <- 1:4
> y <- rep(1,4)
> z <- x + y
> z
[1] 2 3 4 5
> x <- 1:4
> y <- 1:2
> z <- x + y  # y is repeated until it reaches the length of x
> z
[1] 2 4 4 6
> x <- 1:4
> a <- 10
> z <- a * x  # a is repeated until it reaches the length of x
> z
[1] 10 20 30 40
```
7.3 Vectors and arrays

- Some operations

```r
> x <- rnorm(1000)
> sum(x)
[1] -40.19532
> prod(x)
[1] -3.135068e-264
> mean(x)
[1] -0.04019532
> var(x)
[1] 1.015683
> m1 <- matrix(1,nr=2,nc=2)
> m2 <- matrix(2,nr=2,nc=2)
> cbind(m1,m2)
[1,]     1     1     2     2
[2,]     1     1     2     2
```
7.4 Vectors and arrays

- Some operations

```r
> rbind(m1, m2)  # matrix
[,1] [,2]
[1,]  1  1
[2,]  1  1
[3,]  2  2
[4,]  2  2
> rbind(m1, m2) %*% cbind(m1, m2)  # matrix product
[1,]  2  2  4  4
[2,]  2  2  4  4
[3,]  4  4  8  8
[4,]  4  4  8  8
> cbind(m1, m2) %*% rbind(m1, m2)  # matrix product
[,1] [,2]
[1,] 10  10
[2,] 10  10
```
7.5 Vectors and arrays

- Some operations

```r
> diag(m1)
[1] 1 1
> diag(m1) <- 10
> m1
[,1] [,2]
[1,] 10 1
[2,] 1 10
```
8. Graphics

- Some operations

```r
> x <- rnorm(10)
> y <- rnorm(10)
> plot(x, y, xlab = "Ten random values", ylab = "Ten random values")
```
9.1 Last but not least

- Programming. Here again is our first function, loaded in the workspace

```r
> ratiosm <- function(x) {
  sd(x)/mean(x)
}
Now see the use of this function
> ratiosm(x)
[1] 445.7792
```
Remember that mean(x) was equal to 0.00239885, and sd(x) was equal to 1.043072.
- Another option is using the usual if, for, while, etc. But try to avoid them. The program becomes very slow.
- The `apply` function can solve a lot of problems, when trying to avoid the usual syntax.
```r
> apply(m1,1,mean)
[1] 5.5 5.5
```
Again be careful. Try to use the available R functions, not loops and other programming syntax whenever possible. Your program will run faster.
- Another possibility is the ability to interact with C, Fortran Matlab and other programs.
9.2 Last but not least

- Literature for R
  There are a lot of manuals free to download. Some recommended for beginners
  - An introduction to R. (Installed with the software)
  - R for beginners (by Emmanuel Paradis)

The R-manuals

Contributed documentation
9.3 Last but not least

- **R-packages**
  Some basic packages are already installed. Some of them are loaded when the software starts. For the others you have to load them yourself if you need them.
  Besides the basic packages, the CRAN package repository additionally features approximately 3500 packages, sorted by name, task, date of release or etc. If you need a package, you can install it for free. Other repositories exist also.

**Contributed packages**

**CRAN Task Views**
10. Bibliography

- An introduction to R, (Installed with the software)
- R for beginners, by Emmanuel Paradis
- The friendly beginners’ R course, by Toby Marthews
Σας ευχαριστώ

ขอบคุณ