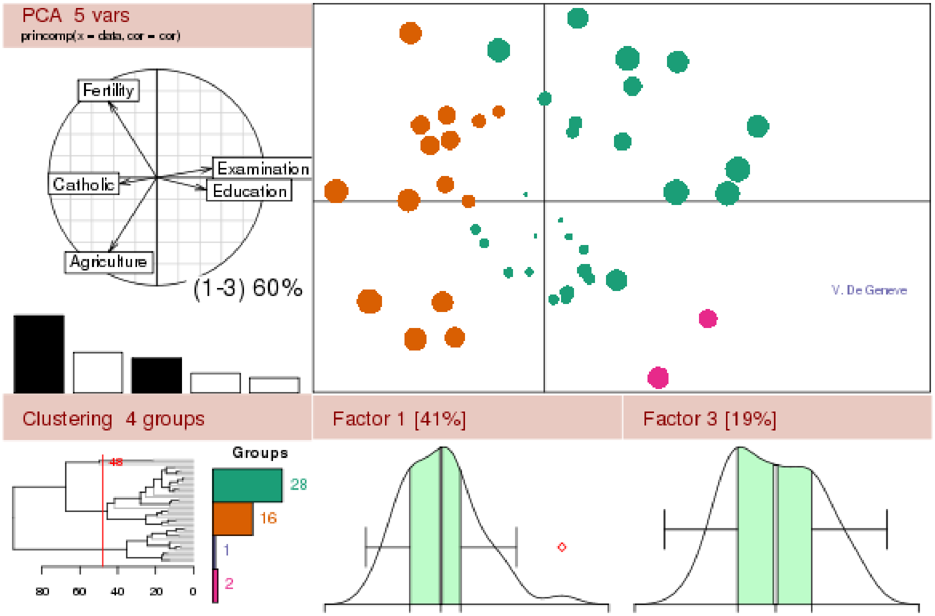
A short introduction to the R statistical programming language



Diamond Light Source - April 2013 – PART 4

James Foadi

MPL, Imperial College London - Diamond Light Source Ltd, Oxfordshire

R as a programming language

A computer program is simply a piece of code that carries to completion a series of tasks (calculations, printing, decisions, etc). An R program consists of a series of R commands. As we have been using several R commands up to now, we have in essence been using R as a programming language. But there are some important elements that need to be considered when writing any program. They are: 1) writing messages, 2) choosing among alternatives, 3) looping and 4) creating/writing to files.

R as a programming language

An R program can be executed from within an R interactive shell with the command “source”, as seen previously. The same program could be executed outside the R shell in batch mode, using the syntax:

```
R --vanilla --slave --quiet < program.R
R --vanilla --slave --quiet < program.R > out.txt
```

Consider a simple example, the program included in the file “prog1.R”. This is a simple program to output the message “Hello, world!”

R as a programming language

“prog1.R”

```
# First simple example of R program

msg <- "Hello, world!\n"
cat(msg)
print(msg)
```

msg is a character string

\n means “end line”

check out the difference between cat and print

Execution:

```
R --vanilla --slave --quiet < prog1.R
```

Output:

```
Hello, world!
[1] "Hello, world!\n"
```

R as a programming language

“prog2.R” is a little bit more involved. It takes two numbers and returns their sum.

```
# Simple R program. Takes in 2 numbers and returns their sum

# Extract input string
args <- commandArgs(trailingOnly=TRUE)

# Turn input into a vector of numbers
cln <- as.numeric(args)

# Output a first message
msg <- paste("You entered",length(cln),"numbers\n")
cat(msg)

# Output a second message
msg <- paste("The sum of the numbers you entered is",sum(cln),"\n")
cat(msg)
```

very useful command, to be used jointly with the –args option in the command line. Args is a character string

turn string into one or more numbers

paste allows concatenation of characters and numbers into one character string

R as a programming language

Execution:

```
R --vanilla --slave --quiet < prog2.R –args 1 10 2  4.5
```

Output:

```
You entered 4 numbers
The sum of the numbers you entered is 17.5
```

R as a programming language

To take decisions based on certain conditions, use the “if” form:

```
if (condition)
{
 execute this
}
```

Program “prog3.R” use the “if” form to control the accuracy of user's input.

R as a programming language

```
# Simple R program. Takes in 3 numbers, the a, b, c of second degree
# algebraic equation, and returns its discriminant, b^2-4*a*c

# Extract input string
args <- commandArgs(trailingOnly=TRUE)

# Turn input into a vector of numbers
cln <- as.numeric(args)

# Stop if less or more than 3 numbers
if (length(cln) != 3)
{
 stop("You need to provide exactly 3 numbers!")
}

# Calculate discriminant
delta <- cln[2]^2-4*cln[1]*cln[3]

# Output result
msg <- "The discriminant of the following equation:\n"
cat(msg)
msg <- paste(cln[1],"x^2 +",cln[2],"x +",cln[3],"= 0\n")
cat(msg)
msg <- paste("is",delta,"\n")
cat(msg)
```

If length of cln is not 3, what is included in curly brackets will be executed (i.e. the program will stop)

"==" is the equality condition
"!=" is the non-equality condition

R as a programming language

Execution1:

```
R --vanilla --slave --quiet < prog2.R –args 1 -5 6
```

Output:

```
The discriminant of the following equation:
1 x^2 + -5 x + 6 = 0
Is 1
```

Execution2:

```
R --vanilla --slave --quiet < prog2.R –args 1 -5
```

Output:

```
Error: You need to provide exactly 3 numbers!
Execution halted
```

R as a programming language

When the same action needs to be carried out several times, use the “for” loop:

```
for (i in vector)
{
 execute this
}
```

Program “prog4.R” use the “for” loop to describe a user's input.

R as a programming language

```
# Simple R program. Takes in many numbers and for each of them
# states if it is greater or smaller than 5.

# Extract input string
args <- commandArgs(trailingOnly=TRUE)

# Turn input into a vector of numbers
cln <- as.numeric(args)

# For loop (between 1 and length(cln))
for (i in 1:length(cln))
{
 n <- cln[i]
 if (n < 10)
 {
  msg <- paste("Number",cln[i],"is smaller than 10\n")
 }
 if (n > 10)
 {
  msg <- paste("Number",cln[i],"is greater than 10\n")
 }
 if (n == 10)
 {
  msg <- paste("Number",cln[i],"is exactly equal to 10\n")
 }
 cat(msg)
}
```

1:length(cln) is vector 1 2 3 …
i takes, in turn, the values of this vector

loop

R as a programming language

Execution:

```
R --vanilla --slave --quiet < prog4.R –args 1 10 2  45
```

Output:

```
Number 1 is smaller than 10
Number 10 is exactly equal to 10
Number 2 is smaller than 10
Number 45 is greater than 10
```

R as a programming language

To write output to a file we still use “cat”...

?

If you type “?cat” you discover that:

```
cat(... , file = "", sep = " ", fill = FALSE, labels = NULL,append = FALSE)
```

Thus you only need to specify a file name.

Try it yourself!

Exercise 1

Modify “prog3.R” so that its output consists of the equation's roots. Use the “a + i b” form if the roots are complex conjugate.

Exercise 2

Design a program to write the times table of side 12 in a file called “Ttable.txt”