Introduction to ‘R’

Vincent Rouilly
Imperial College London
Oct 2009

Objectives

- Introduction to a computational tool
- Practice some mathematical modelling
- Support your future MRes projects

About Computational Biology

- What is your experience ?

- What do you expect from this tutorial ?

About Computational Biology

- "All models are wrong, but some of them are useful"
  - G. Box.

- "Models are always as good as their assumptions"
  - unknown.

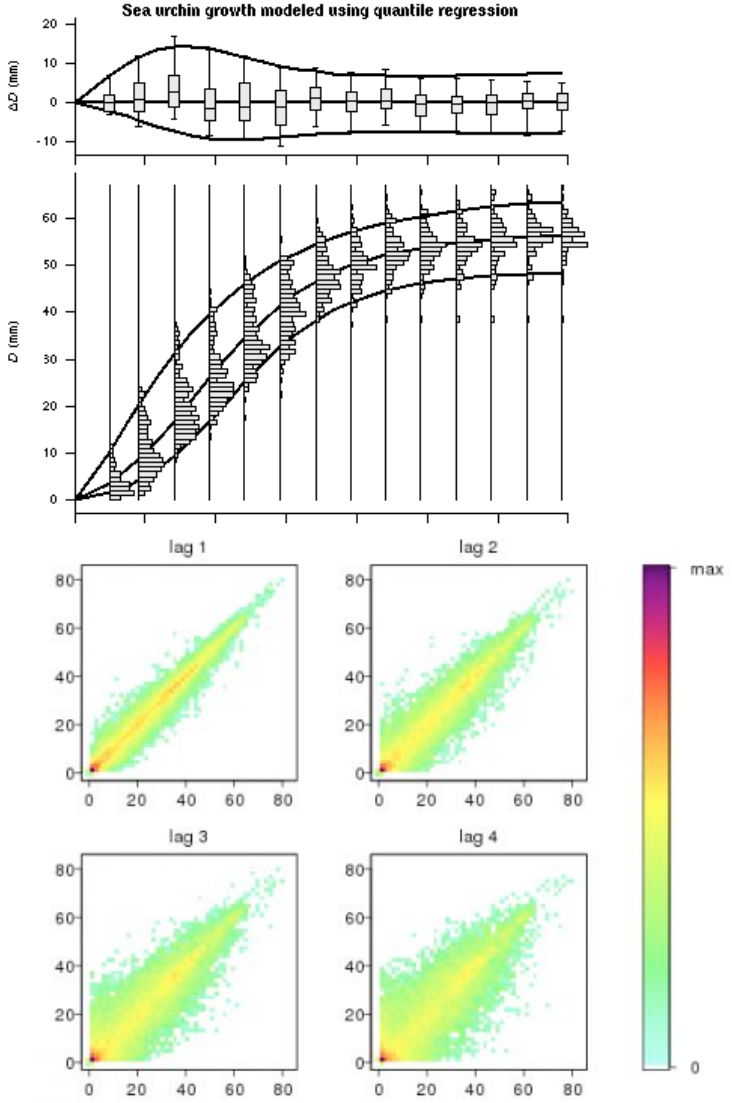
What is ‘R’ ?

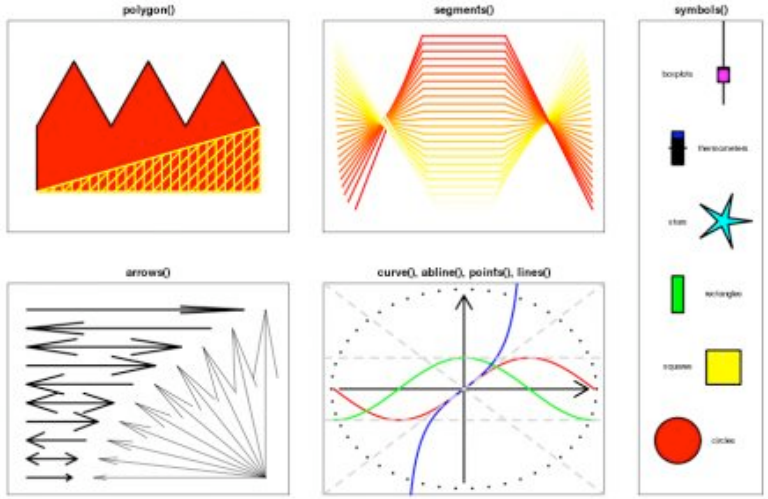
- Computing environment, similar to matlab.
- Very popular in many areas of statistics, computational biology.
- Writing your own functions.
- Approach: command-line for one-liners; write scripts/functions for larger work (edit/run cycle).
- Open Source Software

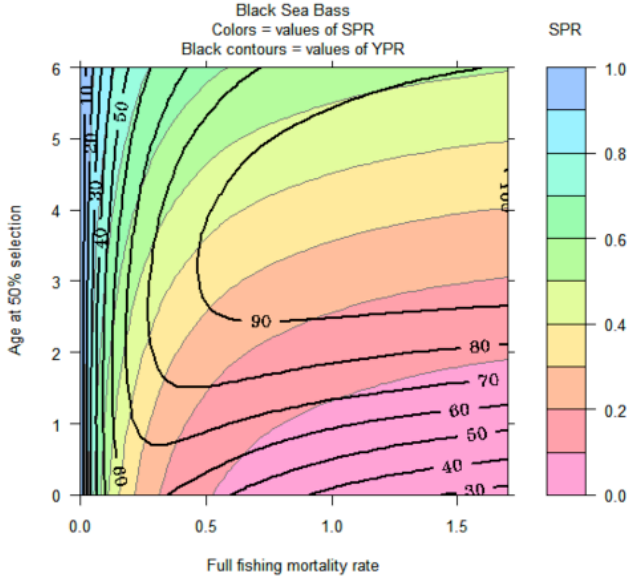
Strength of ‘R’

- GPL’d, available on many platforms.
- Good collection of numerical/statistical routines.
- Comprehensive R Archive Network (CRAN) ~ 1550 packages + BioconductoR project
- On-line doc, with examples
- High-quality graphics (pdf, postscript, quartz, x11, bitmaps). Often used just for plotting . . .

Graphics examples







My very first ‘R’ session

```
x < − rnorm ( 5 0 , mean =4)
x
mean(x)
range(x)
hist(x)
?hist   #check help for ‘hist’ function
hist( x , main=”my first plot ” )
q()
```

Data Types

- integer
- numeric
- character
- complex
- logical

Data Structures

- Vectors
- Lists
- Factors
- Matrices
- Data Frames

Vectors

- Vectors are a fundamental object for R. Scalars are treated as vector of 1

```
y <- c( 10 , 20 , 40 )  # create new vector
y[ 2 ]                  # return 2nd value from vector
length( y )     # return size of vector
x <- 5
length( x )
```

Generating Vectors

- Many short hand methods for regular sequences; c() for irregular.

```
x <- seq ( from=1 , to=9 , by=2)
y <- seq ( from =2 , by =7 , length=3)
z <- 4:8
a <- seq.int( 5 )
b <- c( 3 , 9 , 2 )
d <- c( a , 1 0 , b)
e <- rep( c( 1 , 2 ) , 3 )
```

Vectors: Accessing Elements

- x <− seq(from=100 , by =1 , length=20)

```
x[3]   # return element 3
x[ c(12,14) ]  # return elements 12 and 14
x[1:5]   # return elements from 1 to 5
bad <- 1:4
x[-bad]  # exclude elements 1 to 4
```

Vectors: Setting Elements

- Elements can be set in several ways:

```
x <– rep( 0 , 1 0 )
x[1 : 3] <–2
x[5:6] <– c( –5 , NA)
x[7:10] <– c( 1 , 9 )     # recycling
```

Vectors: Common functions

- length()
- rev()
- sum(), cumsum(), prod(), cumprod()
- mean(), sd(), var(), median()
- min(), max(), range(), summary()
- exp(), log(), sin(), cos(), tan() [radians, not degrees]
- round(), ceil(), floor(), signif()
- sort(), order(), rank()
- which(), which.max()
- any(), all()

Getting help: key commands

- help(hist) # to see help file (or ? hist).
- args(hist) # to see arguments of a function.
- example(boxplot) # run examples in help page.
- help . start () # starts web-browser for help/ on-line docs.
- help . search ("histogram")
- demo() # to list all demos, e.g. demo(graphics)
- **What you can expect to find:**
  - Description
  - one line summary
  - Usage
  - formal arguments
  - Arguments – interpretation of arguments
  - Details – what the function does

Inspecting environment

- objects() # list all variable in memory
- ls() # shorthand for objects
- rm(c(x, b, z)) # delete x, b, z from memory
- rm(list=ls()) # clear all memory

Scripts

- ‘R’ commands can be stored in a text file
- Scripts Use:
  - source (’ trig . R’ , echo=T) # to see commands and output.
- Use print(x) to print an object within a script
- Use “.R” or “.r” as the filename extension. Avoid any temptation to put spaces (although R does not mind) in your filenames!
- On windows, the initial directory might be “My Documents”. You may need to change directory (setwd) first.
- Comment your Code ! Using #...

Data Frames

- Data frame is a special kind of list; all elements are vectors of same length.
- But each column can be of a different type.
- Create a data frame:

```
names <− c( ”joe” , ”fred” , ”harry” )
a <− c( 24 , 19 , 30 )   # age
ht <- c(1.7, 1.85, 1.65)    # height
s <- c(T, F, F)   # are they students ?
d <- data.frame(name=n ames , age=a , height=ht, student=s)
```

- Accessing data.frames

```
 d$age
names(d)
d[2,]  # second row
```

Reading / Writing data to file

- If data are tabular, read.table or read.csv is often useful. It returns a data.frame object

```
x < − read.table ( ’ . . /data/players.dat’ , sep= ’\ t ’, header=T )
names(x)
x
x[2,]
x$age
is.data.frame(x)
```

- Write to file:

```
write.csv ( x , ’ /tmp/players.csv ’ , row.names=F )
```

- See ?read.table and ?write.csv for more info.

Writing functions

- How to define a new function:

```
my.fun <– function ( arg1 , arg2 , . . . ) {

## Doc string here.
X<– arg1 * 2
Z < – sqrt (arg2) + 5
Z < – X * Y
## last value is the return value of the function. ## Use a data
    structure to return several items.
Z
}
```

Writing function: example

$$std.dev = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \quad \text{where} \quad \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

- Compute standard deviation:

```
std.dev <- function(x) {

# return std dev of x
n <- length(x)
xbar <- sum(x)/n
diff <- x – xbar
sum.sq <- sum(diff^2)
var <- sum.sq / (n-1)

sqrt(var)
}
```

Control and Loops

- If
- for
- while

Control and Loops: if

if (expression) { command1 } else { command2 }

```
x <- 8
if (x > 10) {
        cat("x is bigger than 10\n")
} else {
        cat("x is 10 or less \n")
}
```

Always use braces to make your code readable.

Control and Loops: while

while (condition) { command }

```
n1 <- 0 ; n2 <- 1
while ( n2 < 100) {
        print(n2)
        old <- n2
        n2 <- n2 + n1
        n1 <- old
}
```

Always use braces to make your code readable.

Control and Loops: for

for (var in seq) { command }

```
x <- 6
for (i in 1:10) {
    result <- x * i
    cat ( x , ” * ” , i , ”=” , result , ” \ n ” )
}
```

Always use braces to make your code readable.

Basic Plotting

- First example:

```
x < − seq ( from=0 , to =2* pi , len=1 0 0 0 )
y < − cos ( 2 * x )
plot ( x , y )
```

Basic Plotting

Expanding on previous plot . . .

```
plot( x , y , main= ’ cos(2x) ’ , type= ’ l ’ , lty=1 , bty= ’ n ’ )

y2 < − sin( 2∗x ) lines( x , y 2 , main= ’ sin(2x ) ’ , type= ’ l ’ ,
    lty=2)

same <−which(abs (y−y2) < 0 . 0 1 )

points( x[same] , y[same] , pch =19 , col = ’ re d ’ , cex =3)
    legend ( ’ bottom right ’ , c ( ” cos ( 2x ) ” , ” sin( 2x ) ” ) ,
    lty=c ( 1 , 2 ) )
```

Thank you