STATISTICAL MODELLING Part IIC

Practical 2: More on the basics of R IAC/Lent 2011

Comments and corrections to ioana@statslab.cam.ac.uk

Login to the PWF, start R and change the current working directory (`U:\Rwork`) as we did last time.

When writing anything but a short algorithm, it is often easiest to edit the commands in the internal R editor. To start a new script in the R editor simply select `New Script` from the `File` menu. Once you have written the algorithm and saved the file as `Rubbish.R`, say, in the current working directory (`U:\Rwork`), you can execute the commands by typing

```
> source("Rubbish.R")
```

You can also source a script by selecting the corresponding option in the `File` menu, and take advantage of the Windows file browser. The R editor will also let you send individual lines, or whole hilighted selections, of code to the R console for execution using `Ctrl-R`.

If you are using the `Unix` version of R, e.g., on the Statslab machines, the same effect is achieved using `emacs`, `vim`, etc., to edit (and save) your files, and use the `source()` command in R to read in the contents of the file(s). Unfortuately the `Ctrl-R` feature is not supported.[1]

For example, you can use the R editor to write down the following program. It is good practice to start by clearing the existing workspace.

```
> rm(list = ls())
> n <- 50
> X <- runif(n)
> Y <- rnorm(n, mean = 0.5 * X^2 + 1, sd = 0.2)
> plot(X, Y, pch = 4, cex = 0.8, type = "p")
> title("Y~N(mu=0.5*X^2+1,sd=0.2)")
> x <- seq(0, 1, length = 101)
> lines(x, 0.5 * x^2 + 1)
```

See Figure 1 for the result of sourcing this code in R. Some questions to think about for the above code:

[1] However, there is something called `ESS` (`Emacs` Speaks Statistics) which enables similar integration of the `Emacs` editor and R.

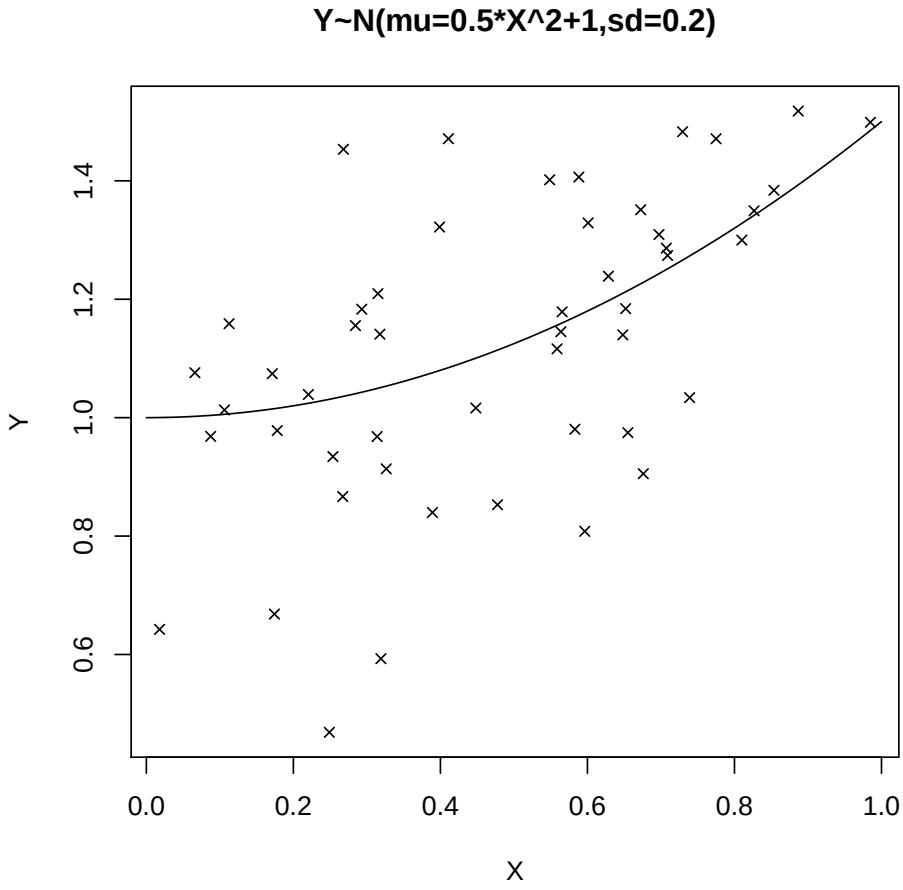

Figure 1: The figure obtained from the code written in the R editor.

- What is the mean Y_i given X_i?
- What does each argument of `plot()` do? What are the defaults? Use `?par` to explore the many graphical parameters.
- How would you write the `seq()` command using the `by` argument instead of `length`?
- `lines()` adds lines to an existing plot. What does `points()` do?
- See if you can replace the `seq()` and `lines()` commands by a single `curve()` or `plot()` command.

On these Windows machines you can print this graph by right-clicking in the graphics window. You are also given the option of saving the graph as a postscript file, which could

then be imported into a LaTeX document, for instance[2]. On Unix machines this option isn't available, but you can create a postscript file with `postscript(file='Rubbish.ps')` before the plot command (this writes output appearing subsequently to a postscript file) and `dev.off()` at the end (this stops writing output to the file). Similar commands `pdf()`, `png()`, etc., also exist.

For fun, you might like to look at

```
> demo(graphics)
```

Now return to the R command line. Although loops are often not necessary, here is an example of one:

```
> for (i in 1:10) {
+     cat("i =", i)
+     cat(", i! =", gamma(i + 1), "\n")
+ }
```

```
i = 1, i! = 1
i = 2, i! = 2
i = 3, i! = 6
i = 4, i! = 24
i = 5, i! = 120
i = 6, i! = 720
i = 7, i! = 5040
i = 8, i! = 40320
i = 9, i! = 362880
i = 10, i! = 3628800
```

The `cat()` function is more sophisticated than `print()`. It concatenates and prints to the screen.

Note that if you want to define a vector `x` component by component within a loop, you must first initialise the vector outside the loop with something like `x <- rep(0,10)` so that R knows the length of the vector.

When writing commands in the R editor or `emacs` and sourcing them, you have to explicitly write `print(x)` (or use `cat()`) for `x` to appear on the screen.

[2]These practical sheets are produced using LaTeX , but without explicitly including any figures, using a special library called `Sweave` which comes with R. Try `?Sweave` for more information.

Use `Esc` (`Ctrl-C` in Unix) to stop a running R command (or a procedure which is being executed).

You can write functions in R as follows:

```
> f <- function(x, y) {
+     z <- x^2 + y^2
+     return(c(cos(z), sin(z)))
+ }
```

These functions can be used in the same way as built-in functions. The return value for an R function is the last expression in the function. In this example, this expression is enclosed in `return()`. Make sure that the last instruction in your functions is not an assignment. Simply typing the function name will echo the code:

```
> f
```

```
function (x, y)
{
    z <- x^2 + y^2
    return(c(cos(z), sin(z)))
}
```

To evaluate the function you must supply (the correct) arguments.

```
> f(2, 3)
```

```
[1] 0.9074468 0.4201670
```

EXERCISES

1. Recall your simulation to estimate $\mathbb{E}(X^6)$ when $X \sim N(0,1)$. Check your answer by numerical integration (use the help!).

2. Suppose that $X_1, \ldots, X_n \overset{iid}{\sim} N(\theta, \sigma^2)$, and we wish to test $H_0 : \theta = 0$ against the one-sided alternative $H_1 : \theta > 0$. Write a simulation to plot the power functions of the t-tests when $n = 20$ and $\sigma = 0.5, 1, 2$ (`power.t.test` may help).

3. Let (E_n) be a sequence of independent Exp(λ) random variables (so each E_n has mean $1/\lambda$). Define a process $(X_t)_{t\geq 0}$ by

$$X_t = \max\left\{ N \in \mathbb{N}_0 : \sum_{n=1}^{N} E_n \leq t \right\}.$$

The process (X_t) is called a (homogeneous) Poisson process of rate λ, and is often used to model arrivals at queues. By definition, $X_0 = 0$. Plot a realisation of a Poisson process of rate 1 over the time interval $t \in [0, 20]$. Estimate by simulation the probability that no more than 15 arrivals have occurred by $t = 20$. Compute an expression for this probability, and use R to evaluate it.

4. With $n = 100$, generate independent pairs $(X_1, Y_1), \ldots, (X_n, Y_n)$ with $X_i \sim U(0, 1)$ and

$$Y_i = m(X_i) + \epsilon_i, \quad i = 1, \ldots, n,$$

where $m(x) = e^{-x}\sin(4\pi x)$ and $\epsilon_i \sim N(0, \sigma^2)$ with $\sigma^2 = 0.04$. Plot the data, as well as the true regression curve $m(x)$. Consider estimating $m(x)$ by the following weighted average of $Y_1, \ldots, Y_n$:

$$\hat{m}_h(x) = \frac{\sum_{i=1}^{n} \phi\left(\frac{X_i - x}{h}\right) Y_i}{\sum_{i=1}^{n} \phi\left(\frac{X_i - x}{h}\right)},$$

where $h > 0$ and ϕ is the standard normal density. What do you think would be a good choice of the bandwidth h? For a few sensible values of h, add the estimate to your plot with a dotted line (use `lines` with the `lty` argument – look this up with `?par`).