16
Phụ lục 1: Lập trình và hàm với R

R được phát triển sao cho người sử dụng có thể phát triển những hàm thích hợp cho mục đích phân tích và tính toán của mình. Thật vậy, như đã đề cập trong phần đầu của sách, có thể xem R là một ngôn ngữ thống kê, và chúng ta có thể sử dụng ngôn ngữ để giải quyết các vấn đề không thường thấy trong sách giáo khoa. Trong phần này, tôi chỉ trình bày một vài hàm đơn giản để bạn đọc có thể hiểu cách vận hành của R và hi vọng giúp bạn đọc tự phát triển các hàm sau đó.

Hàm (hay có khi còn gọi là "macro" trong các phần mềm khác) thực chất là tập hợp một số lệnh được lưu trữ dưới một cái tên. Ở mức độ đơn giản nhất, hàm là "tốc kí" cho một nhóm lệnh.

Ví dụ 1. Trong các lệnh sau đây, chúng ta tạo hai dữ liệu (`data1` và `data2`). Mỗi dữ liệu có hai cột số liệu được tạo ra bằng mô phỏng từ phân phối chuẩn. Sau đó, vẽ biểu đồ cho hai dữ liệu với ghi chú.

```
data1 <- cbind(rnorm(100,1), rnorm(100,0))
data2 <- cbind(rnorm(100,-1), rnorm(100,0))
xr <- range(rbind(data1,data2)[,1])
yr <- range(rbind(data1,data2)[,2])
plot(data1, xlim=xr, ylim=yr, col=1, xlab="", ylab="")
par(new=T)
plot(data2, xlim=xr, ylim=yr, col=2, xlab="", ylab="")
title(main="My simulated data", xlab="Weight", ylab="Yield")
legend(-3.0, -1.5, c("Big", "Small"), col=1:2, pch=1)
```

Một cách để nhớ tất cả các lệnh này là lưu trữ chúng trong một text file chẳng hạn. Mỗi lần muốn sử dụng, chúng ta chỉ đơn giản cắt và dán các lệnh này vào R. Một cách khác tốt hơn là tạo ra một hàm gồm các lệnh trên để có thể sử dụng nhiều lần.

Mỗi hàm R phải có tên. Tất cả các lệnh được chứa trong khu vực được giới hạn bằng hai kí hiệu `{` và `}`. Kí hiệu `{` cho biết tất cả các lệnh sau đó là nằm trong hàm; và kí hiệu `}` cho biết chấm dứt hàm. Trong ví dụ trên, chúng ta gọi hàm là `plotfigure`:

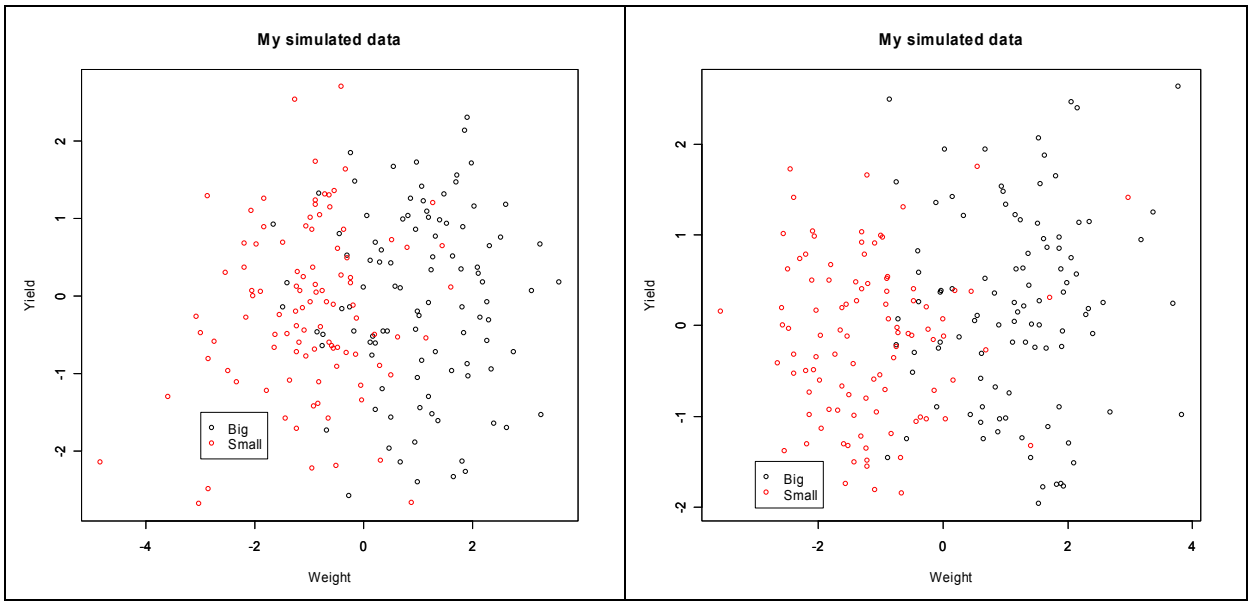
```
plotfigure <- function()
  {
     data1 <- cbind(rnorm(100,1), rnorm(100,0))
     data2 <- cbind(rnorm(100,-1), rnorm(100,0))
     xr <- range(rbind(data1,data2)[,1])
     yr <- range(rbind(data1,data2)[,2])
     plot(data1, xlim=xr, ylim=yr, col=1, xlab="", ylab="")
     par(new=T)
     plot(data2, xlim=xr, ylim=yr, col=2, xlab="", ylab="")
     title(main="My simulated data", xlab="Weight", ylab="Yield")
```

```
    legend(-3.0, -1.5, c("Big", "Small"), col=1:2, pch=1)
}
```

Sau khi đã cho vào R, chúng ta chỉ đơn giản gọi hàm nhiều lần như sau:

```
> plotfigure()
> plotfigure()
```

và kết quả sẽ như sau:



Trong hàm `plotfigure` trên, chúng ta mô phỏng 100 số liệu từ phân phối chuẩn. Và cứ mỗi lần ứng dụng, hàm chỉ tạo ra 100 số liệu, chứ chúng ta không thay đổi được (ngoại trừ phải thay đổi từ lúc biên tập, hay lập hàm). Nói cách khác, hàm trên không có thông số.

Khía cạnh tiện lợi của hàm là chúng ta có thể làm cho thông số thay đổi theo ý muốn của người sử dụng. Chẳng hạn như chúng ta muốn thay đổi số số liệu mô phỏng và trung bình từ luật phân phối chuẩn, chúng ta chỉ cần cho hai con số này là hai thông số (parameters) để người sử dụng có thể thay đổi. Tạm gọi đó là thông số **`n`**, **`mean1`**, và **`mean2`**, thì hàm sẽ như sau:

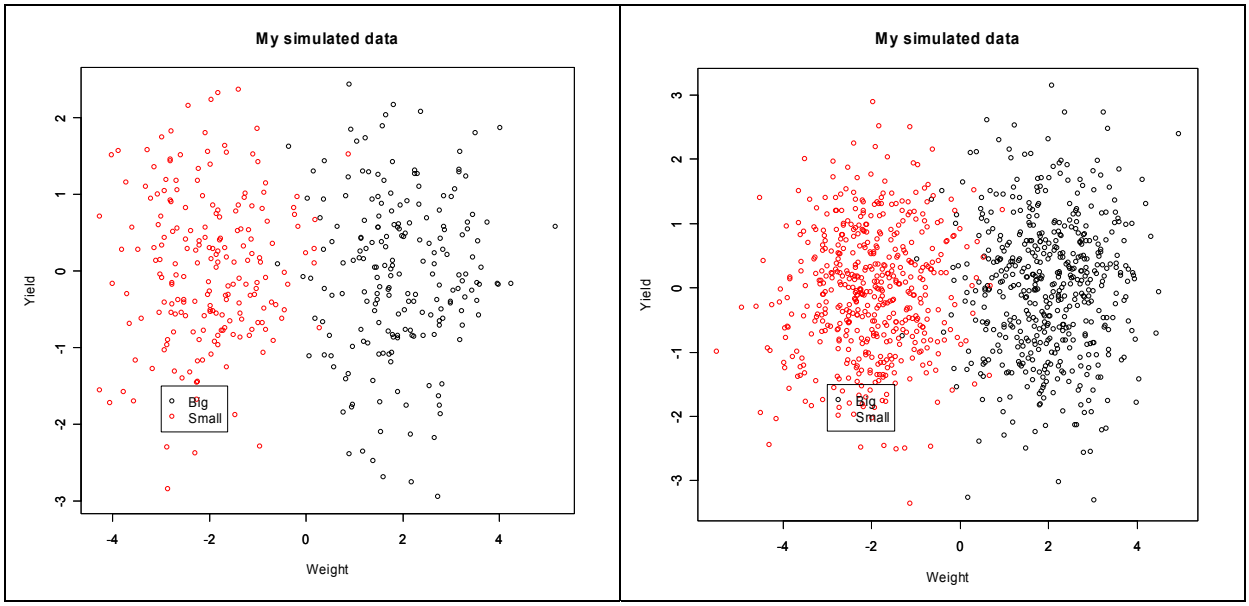
```
plotfigure <- function(n, mean1, mean2)
  {
    data1 <- cbind(rnorm(n,mean1), rnorm(n,0))
    data2 <- cbind(rnorm(n,mean2), rnorm(n,0))
    xr <- range(rbind(data1,data2)[,1])
    yr <- range(rbind(data1,data2)[,2])
    plot(data1, xlim=xr, ylim=yr, col=1, xlab="", ylab="")
    par(new=T)
    plot(data2, xlim=xr, ylim=yr, col=2, xlab="", ylab="")
```

```
    title(main="My simulated data", xlab="Weight", ylab="Yield")
    legend(-3.0, -1.5, c("Big", "Small"), col=1:2, pch=1)
}
```

Khi ứng dụng hàm, chúng ta chỉ đơn giản thay đổi `n` và `mean`. Trong hai lệnh sau đây, chúng ta đầu tiên vẽ một biểu đồ tán xạ với 200 số liệu, và số trung bình -2 và 2. Trong lệnh hai, chúng ta nâng số liệu lên 200, nhưng trung bình vẫn như lần mô phỏng trước:

```
> plotfigure(200, 2, -2)
> plotfigure(500, 2, -2)
```

Và kết quả sẽ khác trên:



Ví dụ 2. Chúng ta muốn viết một hàm để cộng hai số. (Tất nhiên R có khả năng làm "việc" này, nhưng vì lí do minh họa, tôi sẽ giả thiết đơn giản như thế). Gọi hàm đó là `add`. Hai thông số `a` và `b` là "arguments. Cách viết như sau:

```
add <- function(a, b)
 {
   sum = a+b
   ans <- "Answer = "
   cat(ans, sum, "\n")
 }
```

Thế là xong! Như thấy, bước đầu tiên, chúng ta cho tên hàm là `add` và định nghĩa thông số `a` và `b`. Một hàm phải được mở đầu bằng kí hiệu `{` và chấm dứt bằng `}`. `sum` là một biến số cộng `a` và `b`. `ans <- "Answer = "` định nghĩa trả lời (có thể không cần). `cat(ans, sum, "\n")` có chức năng thu thập số liệu và trình bày kết quả

cho người dử dụng hàm, trong đó "\" có nghĩa là sau khi trình bày, cho người sử dụng một prompt khác. Bạn đọc có thể dán các lệnh trên vào R và thử cho lệnh:

```
> add(3, 9)
Answer =  12

> add(sqrt(5), exp(10))
Answer =  22028.7
```

Ví dụ 3. Hàm sau đây tiến hành nhiều tính toán hơn hàm trong ví dụ 1. Nếu chúng ta có một biến số gồm n phần tử $x_1, x_2, x_3, ..., x_n$ tuân theo luật phân phối chuẩn với trung bình μ và phương sai σ^2. Viết theo kí hiệu toán:

$$x_i \sim N\left(\mu, \sigma^2\right)$$

Nếu chúng ta có thông tin trước cho biết μ có luật phân phối chuẩn với trung bình θ và phương sai τ^2, hay:

$$\mu \sim N\left(\theta, \tau^2\right)$$

Qua định lí Bayes, chúng ta có thể ước tính trung bình $\mu_p = \dfrac{\dfrac{\theta}{\tau^2} + \dfrac{\overline{nx}}{\sigma^2}}{\dfrac{1}{\tau^2} + \dfrac{n}{\sigma^2}}$ và phương sai $\sigma_p^2 = \left(\dfrac{1}{\tau^2} + \dfrac{n}{\sigma^2}\right)^{-1}$. Trong đó, $\bar{x}$ là số trung bình của mẫu *n*. μ_p và σ_p^2 được gọi là "posterior". Chúng ta có thể viết một hàm bằng R để tính hai số này như sau. Gọi tên hàm là bayes.

```
bayes <- function(x, prior.mean, prior.var)
 {
   n <- length(x)
   sample.mean <- mean(x)
   sample.var <- var(x)
   numerator <- (prior.mean/prior.var) + (n*sample.mean/sample.var)
   denominator <- 1/prior.var + n/sample.var
   posterior.mean = numerator/denominator
   posterior.var = 1/denominator
   a <- "Posterior mean = "
   b <- "Posterior variance = "
   cat("Sample size = ", n, "\n")
   cat("Sample mean = ", sample.mean, "\n")
   cat("Sample var = ", sample.var, "\n")
   cat("Prior mean = ", prior.mean, "\n")
   cat("Prior var = ", prior.var, "\n")
   cat(a, posterior.mean, "\n")
```

```
   cat(b, posterior.var, "\n")
}
```

Ví dụ 4. Mật độ chất khoáng trong xương (bone mineral density - `bmd`) trong một quần thể thường phân phối theo luật phân phối chuẩn, với giá trị trung bình khoảng 1.0 g/cm^2 và phương sai 0.0144 g/cm^4. Giả dụ chúng ta đo mật độ xương của một nhóm bệnh nhân như sau: `1.0, 1.5, 2.1, 1.7, 1.8, 0.9, 0.7.` Chúng ta muốn biết giá trị trung bình và phương sai của mẫu này sau khi "điều chỉnh" cho trung bình và phương sai đã biết trước. Trước hết, chúng ta gọi nhóm số liệu này là `bmd`:

```
> bmd  <- c(1.0, 1.5, 2.1, 1.7, 1.8, 0.9, 0.7)
```

và sau đó "gọi" hàm `bayes` như sau:

```
> bayes(bmd, 1.0, 0.0144)
Sample size =  7
Sample mean =  1.385714
Sample var =  0.2747619
Prior mean =  1
Prior var =  0.0144
Posterior mean =  1.103525
Posterior variance =  0.01053507
```

Trên đây chỉ là một vài hàng giới thiệu cách lập trình và viết hàm bằng ngôn ngữ R. Trong thực tế, tất cả các hàm như `survival`, `BMA`, `meta`, `Hmisc`, v.v… đều được phát triển bằng ngôn ngữ R. Bạn đọc có thể tham khảo tài liệu "Introduction to R" của W. Venables và B. Ripley (phần cuối của sách) để biết thêm chi tiết kĩ thuật.