Bài 1.

Giới thiệu về R (phần 1)

1. R là gì?

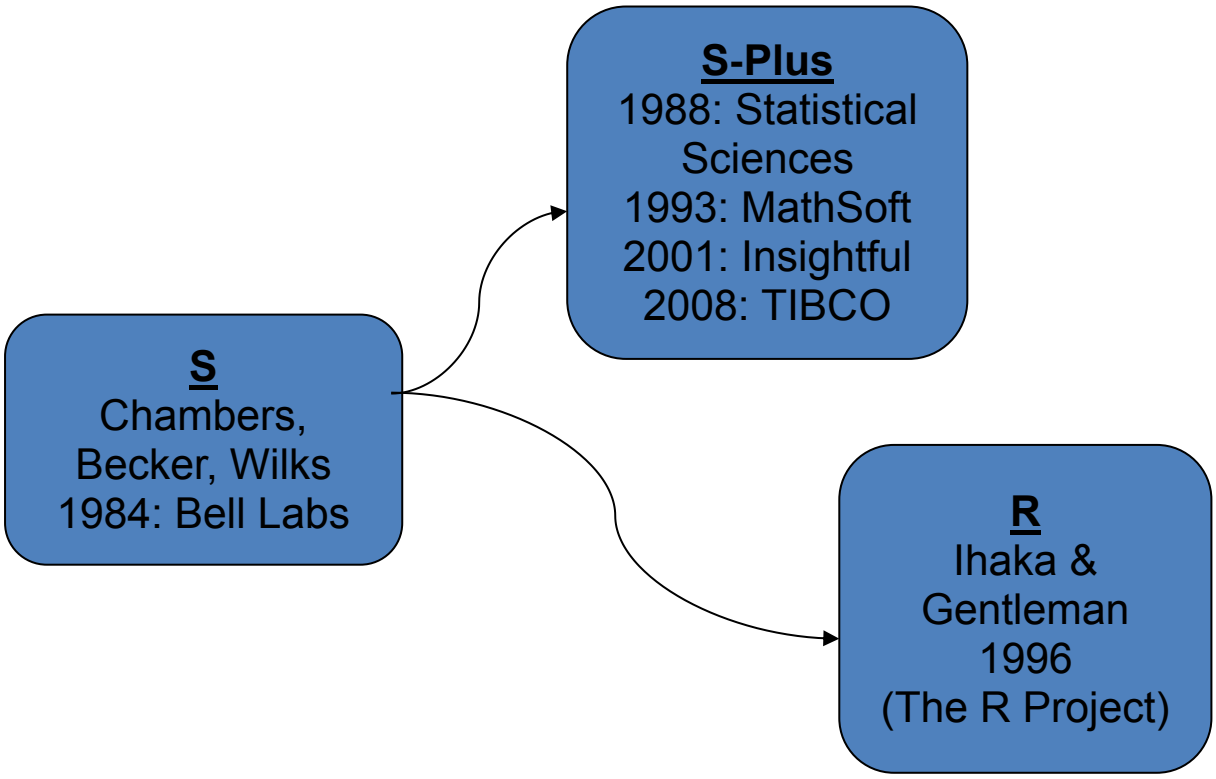
R là một phần mềm thống kê được sử dụng để phân tích dữ liệu. Nó chứa một lượng lớn các thủ tục thống kê chẳng hạn t-test, chi-square test, các mô hình hồi quy chuẩn, ước lượng các biến công cụ, hồi quy đa thức địa phương,… Nó cũng cung cấp khả năng đồ họa cấp cao. Có một số điểm tương đồng giữa ngôn ngữ lập trình C và R, nhưng cả hai đều có cách vận hành khác nhau.

2. Nguồn gốc lịch sử

R là một ngôn ngữ lập trình và môi trường phần mềm cho phân tích thống kê, trình bày đồ thị và viết báo cáo. R được tạo ra bởi Ross Ihaka và Robert Gentleman tại Đại Học Auckland, New Zealand, và hiện tại được phát triển bởi "R Development Core Team".

R xuất hiện lần đầu vào năm 1993. Khi đó một nhóm lớn các cá nhân đã đóng góp vào R bằng việc gửi code và báo cáo lỗi. Từ giữa 1997, có một nhóm trụ cột ("R Development Core Team" là những người có thể chỉnh sửa trên kho lưu trữ mã nguồn R.

Ngôn ngữ này được đặt tên **R**, dựa trên ký tự đầu của tên của hai tác giả R (Robert Gentleman và Ross Ihaka), và một phần theo cách đặt tên của Ngôn ngữ **S** của Bell Labs.



3. R làm được và không làm được gì?

3.1 Ưu điểm:
1. R là phần mềm miễn phí. R là một dự án GNU chính thức và được phân phối dưới "Free Software Foundation General Public License (GPL)"
2. R là một gói phân tích dữ liệu mạnh mẽ với nhiều chức năng thống kê tiên tiến và quy chuẩn. Xem the Comprehensive R Archive Network (CRAN)'s Task Views để thấy được những gì ta có thể làm với R.
3. R được sử dụng rộng rãi trong các nghành khoa học chính trị, thống kê, kinh tế lượng, khoa học bảo hiểm, xã hội học, tài chính, v.v…
4. R là một ngôn ngữ lập trình, vì vậy các khả năng của nó có thể được mở rộng một cách dễ dàng thông qua việc sử dụng các hàm do người dùng định nghĩa. Một tập hợp lớn các hàm và gói lệnh được đóng góp từ người dùng có thể tìm thấy trên CRAN's Contributed Packages.
5. Chạy trên mọi nền tảng máy tính - UNIX (Linux), Windows, và Macintosh,
6. R là ngôn ngữ hướng đối tượng. Hầu như bất cứ điều gì (chẳng hạn, các cấu trúc dữ liệu phức tạp) có thể được lưu giữ như một đối tượng R.
7. R là ngôn ngữ ma trận.
8. R tạo ra những biểu đồ với chất lượng cao (chuẩn bài báo khoa học).
9. R có khả năng lưu trữ và xử lý dữ liệu hiệu quả,
10. R cung cấp một bộ các phép toán để tính toán trên các mảng, danh sách, vector, và ma trận,

3.2 Khuyết điểm:
1. Không dễ để bắt đầu các phân tích với R – cần thiết phải học các kiến thức cơ bản về cú pháp và các lệnh cơ bản,
2. Hầu như không có giao diện đồ họa người dùng (GUI) mà chỉ có dòng lệnh,
3. Ta cần phải tìm kiếm các lệnh khả thi tốt nhất trong nhiều gói lệnh (package),
4. Mọi bảng và đồ thị phải được "lập trình" – không được tạo ra bằng cách click chuột,

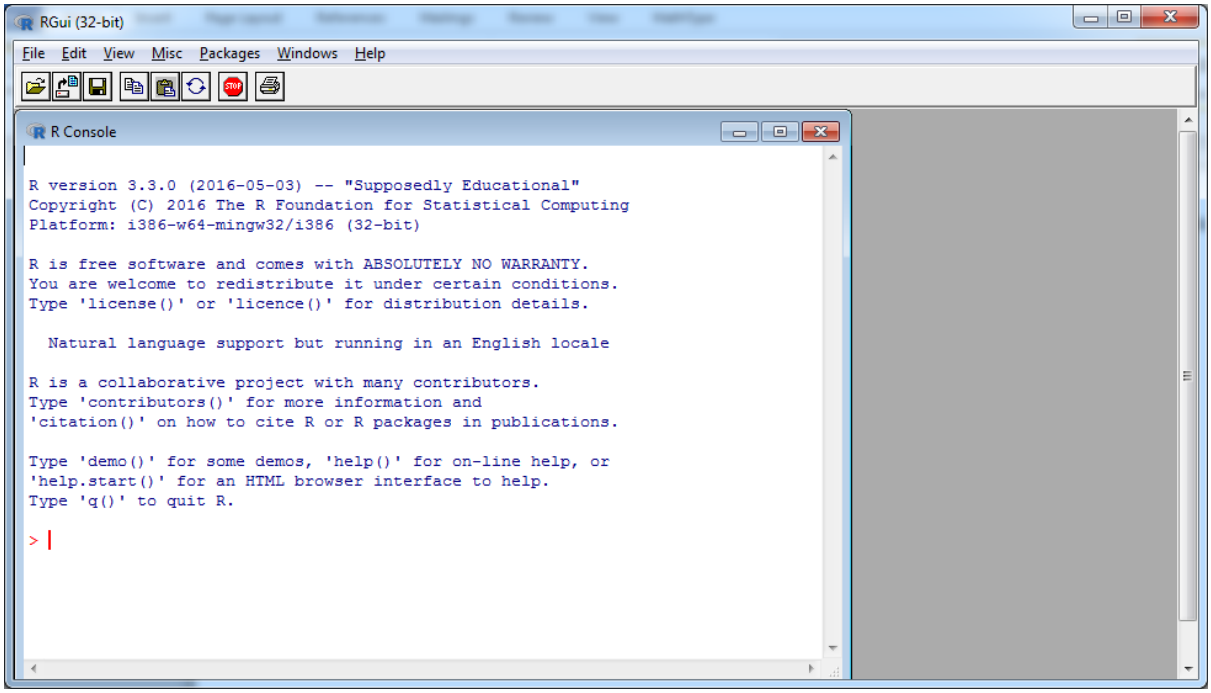
4. Cài đặt và làm quen với R

4.1 Cài đặt

Để cài đặt R trên máy tính của bạn, vào trang chủ của R: http://www.r-project.org/ và làm các bước sau (giả sử bạn dùng hệ điều hành Windows):
- Nhấp vào `download CRAN` ở thanh bên trái
- Chọn một download site
- Chọn Windows như là hệ điều hành mục tiêu
- Nhấp `base`
- Chọn `Download R 3.x.x for Windows` và chọn các câu trả lời mặc định cho mọi câu hỏi.

4.2 Chạy lần đầu

Cửa sổ chương trình R khi mở lên. Ta thấy nó xuất hiện cửa sổ chính là `RGUI (32bit)` và một cửa sổ con trong đó là `R Console` (Bảng điều khiển lệnh). Các lệnh trong R được thực hiện qua cửa sổ R Console này.



5. Trợ giúp trong R

- Lệnh `help(tên hàm)`, hoặc `?tên hàm` : xem trang thông tin về một hàm bất kỳ trong R
- Lệnh `help.search("chủ đề")` có thể giúp ích khi bạn không biết tên của một hàm.
- Lệnh `help.start()` sẽ mở một trình duyệt chỉ đến các thông tin được lưu trên máy về R.
- Lệnh `example(tên hàm)` : chạy minh họa các ví dụ của một hàm (nếu có)

Ví dụ:

Chạy thử các lệnh sau:

```
example(plot)
example(hist)
example(boxplot)
```

- Lệnh `demo()` : Minh họa tổng quan về các chức năng của R

6. Cấu trúc dữ liệu

Các kiểu đối tượng (các cấu trúc nguyên tử)

`integer`	các số nguyên (15, 23, 8, 42, 4, 16)
`numeric`	các số thực (double precision: 3.14, 0.0002, 6.022E23)
`character`	chuỗi ký tự (“Hello World”, “ROFLMAO”, “A”)
`logical`	TRUE/FALSE hoặc T/F

Các lớp đối tượng

`vector`	đối tượng với kiểu nguyên tử
`factor`	đối tượng vector với các nhóm rời rạc (có thứ tự/không thứ tự)
`array`	mảng nhiều chiều
`matrix`	mảng hai chiều
`list`	vector các thành phần

`data.frame` "tựa matrix" danh sách các biến với cùng số hàng

Các giá trị đặc biệt

`NULL` đối tượng có chiều dài 0, kiểm tra với lệnh `is.null(x)`
`NA` Not Available / giá trị khuyết, kiểm tra với lệnh `is.na(x)`
`NaN` Not a number, kiểm tra với lệnh `is.nan(x)` (e.g. `0/0`, `log(-1)`)
`Inf, -Inf` Positive/negative infinity, kiểm tra với lệnh `is.infinite(x)` (e.g. `1/0`)

Thông tin đối tượng

`summary(x)` tóm tắt chung về đối tượng
`str(x)` hiển thị cấu trúc đối tượng
`mode(x)` nhận hoặc thiết lập kiểu lưu trữ
`class(x)` tên của lớp đối tượng
`is.<mode>(x)` kiểm tra kiểu của đối tượng (is.numeric, is.logical, v.v.)
`attr(x, which)` nhận hoặc thiết lập thuộc tính của một đối tượng
`attributes(x)` nhận hoặc thiết lập mọi thuộc tính của một đối tượng

7. Vector, Ma trận, Data frame, Danh sách

7.1. Vector

Ta có thể tạo một vector bằng cách dùng lệnh `c()` để nối các phần tử lại với nhau. Ta có thể tạo một dãy bằng cách dùng lệnh `symbol` hoặc `seq()`. Ví dụ, 1:5 cho mọi số giữa 1 và 5. Lệnh `seq()` cho phép ta xác định khoảng giữa các số liên tiếp. Ta cũng có thể lặp lại một khuôn mẫu bằng cách dùng lệnh `rep()`. Ta cũng có thể tạo một vector số gồm các giá trị trống bằng cách dùng `numeric()`, một vector ký tự các giá trị trống bằng cách dùng `character()` và một vector logical các giá trị trống (tức là FALSE) bằng cách `logical()`

```
> c(1,2,3,4,5)
[1] 1 2 3 4 5
> c("a","b","c","d","e")
[1] "a" "b" "c" "d" "e"
> c(T,F,T,F)
[1]  TRUE FALSE  TRUE FALSE

> 1:5
[1] 1 2 3 4 5
> 5:1
[1] 5 4 3 2 1
> seq(1,5)
[1] 1 2 3 4 5
> seq(1,5,by=.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
> rep(1:2,5)
 [1] 1 2 1 2 1 2 1 2 1 2
> numeric(5)
[1] 0 0 0 0 0
> logical(5)
[1] FALSE FALSE FALSE FALSE FALSE
```

```
> character(5)
[1] "" "" "" "" ""
```
Lệnh `length()` tính chiều dài của vector. `last() (sfsmisc)` trả về phần tử của cùng của một vector nhưng điều này cũng có thể thực hiện một cách đơn giản mà không cần gói lệnh thêm nào.
```
x <- seq(1,5,by=.5)     # Tạo một dãy số
x                       # Hiển thị đối tượng này
 [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> length(x)             # Nhận chiều dài của đối tượng x
 [1] 9
> library(sfsmisc)
> last(x)               # Chọn phần tử cuối cùng của x
 [1] 5.0
> x[length(x)]          # Chọn phần tử cuối cùng mà không cần gói
lệnh thêm nào.
 [1] 5.0
```

7.2. Factor

`factor()` chuyển đổi một vector thành một factor. Một factor cũng có thể được sắp thứ tự với tùy chọn `ordered=T` hoặc lệnh `ordered()`. `levels()` trả về các mức của một factor. `gl()` sinh ra các factor. `n` là số mức, `k` là số lần lặp lại của mỗi factor và `length` là tổng chiều dài của factor. `labels` là tùy chọn và cho các nhãn của mỗi mức.
Các factor có thể được xem như các biến danh mục. Một hàm quan trọng cho phân tích factor là hàm `table()`, đưa ra một loại tổng hợp. Khi xét các loại dữ liệu thống kê (danh mục, thứ bậc, khoảng và tỷ lệ), các factor có thể là danh mục, thứ bậc, hoặc khoảng. Các factor danh mục là các tên danh mục, ví dụ trong số đó có thể là tên các quốc gia kết hợp với một số thông tin khác. Một ví dụ về factor thức bậc là tập các lần đua của một vận động viên nào đó kèm theo vị trí hoàn thành của vận động viên này (nhất, nhì, …). Cuối cùng, một ví dụ về các factor mức khoảng sẽ là các khoảng tuổi như "20 - 29", "30 - 39", v.v. Nói chung, R có thể thứ tự một cách tự động các số được lưu một cách phù hợp nhưng một lập trình viên có thể sử dụng cùng các kỹ thuật với loại dữ liệu này để đặt thứ bậc theo kiểu phù hợp nhất với áp dụng của họ.
Xem thêm `is.factor()`, `as.factor()`, `is.ordered()` và `as.ordered()`.

```
factor(c("yes","no","yes","maybe","maybe","no","maybe","no","no"))
[1] yes   no    yes   maybe maybe no    maybe no    no
Levels: maybe no yes

factor(c("yes","no","yes","maybe","maybe","no","maybe","no","no"),
ordered = T)
[1] yes   no    yes   maybe maybe no    maybe no    no
Levels: maybe < no < yes

ordered(c("yes","no","yes","maybe","maybe","no","maybe","no","no"))
[1] yes   no    yes   maybe maybe no    maybe no    no
Levels: maybe < no < yes

ordered(as.factor(c("First","Third","Second","Fifth","First","First"
,"Third")),levels = c("First","Second","Third","Fourth","Fifth"))
[1] First  Third  Second Fifth  First  First  Third
Levels: First < Second < Third < Fourth < Fifth
```

```
gl(n=2, k=2, length=10, labels = c("Male", "Female")) # generate
factor levels
[1] Male   Male   Female Female Male   Male   Female Female Male
Male
Levels: Male Female
```

7.3. Matrix

Nếu ta muốn tạo một ma trận mới, một cách là sử dụng hàm `matrix()`. Ta phải nhập một vector dữ liệu, số hàng và/hoặc cột và cuối cùng ta có thể xác định xem ta muốn R đọc vector theo hàng hay theo cột (tùy chọn mặc định). Dưới đây là hai ví dụ.

```
matrix(data = NA, nrow = 5, ncol = 5, byrow = T)
     [,1] [,2] [,3] [,4] [,5]
[1,]   NA   NA   NA   NA   NA
[2,]   NA   NA   NA   NA   NA
[3,]   NA   NA   NA   NA   NA
[4,]   NA   NA   NA   NA   NA
[5,]   NA   NA   NA   NA   NA

matrix(data = 1:15, nrow = 5, ncol = 5, byrow = T)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]    1    2    3    4    5
[5,]    6    7    8    9   10
```

Các hàm `cbind()` và `rbind()` kết hợp các vector thành ma trận theo cột hoặc theo hàng:

```
v1 <- 1:5
v2 <- 5:1
v2
[1] 5 4 3 2 1
cbind(v1,v2)
     v1 v2
[1,]  1  5
[2,]  2  4
[3,]  3  3
[4,]  4  2
[5,]  5  1

rbind(v1,v2)
   [,1] [,2] [,3] [,4] [,5]
v1    1    2    3    4    5
v2    5    4    3    2    1
```

Chiều của một ma trận có thể đạt được bằng cách dùng hàm `dim()`. Một cách khác là `nrow()` và `ncol()` trả về số hàng và cột của ma trận:

```
X<-matrix(data = 1:15, nrow = 5, ncol = 5, byrow = T)
dim(X)
[1] 5 5
nrow(X)
```

```
[1] 5
ncol(X)
[1] 5
```

Hàm `t()` chuyển vị một ma trận:

```
t(X)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11    1    6
[2,]    2    7   12    2    7
[3,]    3    8   13    3    8
[4,]    4    9   14    4    9
[5,]    5   10   15    5   10
```

Không giống data frame, các ma trận phải có dạng số hoặc ký tự:

```
a=matrix(2,2,2)
a
     [,1] [,2]
[1,]    2    2
[2,]    2    2
a = rbind(a,c("A","A"))
a
     [,1] [,2]
[1,] "2"  "2"
[2,] "2"  "2"
[3,] "A"  "A"
```

7.4. Mảng

Một mảng gồm n chiều với mỗi chiều là một vector về các đối tượng R cùng loại. Một mảng một chiều một phần tử có thể được xây dựng như sau.

```
x = array(c(T,F),dim=c(1))
print(x)
[1] TRUE
```

Mảng x được tạo với một chiều (dim=c(1)) được rút từ vector các giá trị có thể c(T,F). Một cách tương tự, y, có thể được tạo với một chiều và hai giá trị.

```
y = array(c(T,F),dim=c(2))
print(y)
[1]  TRUE FALSE
```

Một mảng ba chiều – 3x3x3 – có thể được tạo như sau.

```
z = array(1:27,dim=c(3,3,3))
dim(z)
[1] 3 3 3
print(z)
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

, , 2
```

```
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

, , 3

     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

Các mảng R được truy xuất theo cách tương tự các mảng trong các ngôn ngữ khác: theo chỉ số nguyên, bắt đầu là 1 (không phải 0). Đoạn mã sau đây cho ta thấy cách mà chiều thứ ba của mảng 3x3x3 có thể được truy xuất như thế nào.

```
z[,,3]
     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

Xác định cụ thể hai trong ba chiều sẽ trả về một mảng một chiều.

```
z[,3,3]
[1] 25 26 27
```

Xác định cụ thể ba trong ba chiều sẽ trả về một phần tử của mảng 3x3x3.

```
z[3,3,3]
[1] 27
```

Ta có thể phân hoạch mảng phức tạp hơn.

```
z[,c(2,3),c(2,3)]
, , 1

     [,1] [,2]
[1,]   13   16
[2,]   14   17
[3,]   15   18

, , 2

     [,1] [,2]
[1,]   22   25
[2,]   23   26
[3,]   24   27
```

Các mảng không cần đối xứng theo mọi chiều. Đoạn code sau tạo một cặp các mảng 3x3.

```
w = array(1:18,dim=c(3,3,2))
print(w)
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
```

```
[2,]    2    5    8
[3,]    3    6    9

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18
```

Các đối tượng của các vector tạo nên mảng phải cùng loại, nhưng chúng không cần phải là số.

```
u = array(c(T,F),dim=c(3,3,2))
print(u)
, , 1

      [,1]  [,2]  [,3]
[1,]  TRUE FALSE  TRUE
[2,] FALSE  TRUE FALSE
[3,]  TRUE FALSE  TRUE

, , 2

      [,1]  [,2]  [,3]
[1,] FALSE  TRUE FALSE
[2,]  TRUE FALSE  TRUE
[3,] FALSE  TRUE FALSE
```

7.5. Danh sách

Một danh sách là một tập các đối tượng R. `list()` tạo một danh sách. `unlist()` chuyển đổi một danh sách thành một vector. Các đối tượng trong một danh sách không phải cùng loại hoặc chiều dài.

```
x <- c(1:4)
y <- FALSE
z <- matrix(c(1:4),nrow=2,ncol=2)
myList <- list(x,y,z)
myList
 [[1]]
[1] 1 2 3 4

 [[2]]
[1] FALSE

 [[3]]
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Các danh sách có các phương pháp rất linh hoạt để tham chiếu
Theo chỉ số:

```
a = list()
a
```

```
list()
a[[1]] = "A"
a
[[1]]
[1] "A"

a[[2]]="B"
a
[[1]]
[1] "A"

[[2]]
[1] "B"
```
Theo tên:
```
a
list()
a$fruit = "Apple"
a
$fruit
[1] "Apple"

a$color = "green"
a
$fruit
[1] "Apple"

$color
[1] "green"
```
Điều này cũng có thể được đệ quy và kết hợp
```
a = list()
a[[1]] = "house"
a$park = "green's park"
a
[[1]]
[1] "house"

$park
[1] "green's park"


a$park = "green's park"
a[[1]]$address = "1 main st."

a
[[1]]
[[1]][[1]]
[1] "house"

[[1]]$address
[1] "1 main st."


$park
[1] "green's park"
```

Sử dụng các quy tắc phạm vi trong R người ta cũng có thể đặt tên động và tạo các phần tử danh sách

```
a = list()
n = 1:10
fruit = paste("number of coconuts in bin",n)
my.number = paste("I have",10:1,"coconuts")
for (i in 1:10)a[fruit[i]] = my.number[i]
a$'number of coconuts in bin 7'
[1] "I have 4 coconuts"
```

7.6. Data frame

Một data frame có thể được xem như "một danh sách các biến/vector có cùng chiều dài". Trong ví dụ sau, một data frame có hai vector được tạo, mỗi vector có năm phần tử. Vector đầu, v1, là một dãy các số nguyên từ 1 đến 5. Vector thứ hai, v2, là 5 giá trị logical được rút từ loại T và F. Data frame sau đó được tạo từ các vector này. Các cột của data frame có thể được truy xuất bằng cách dùng cách chỉ số nguyên hoặc tên cột và ký hiệu $.

```
v1 = 1:5
v2 = c(T,T,F,F,T)
df = data.frame(v1,v2)
print(df)
  v1    v2
1  1  TRUE
2  2  TRUE
3  3 FALSE
4  4 FALSE
5  5  TRUE
df[,1]
 [1] 1 2 3 4 5
df$v2
 [1] TRUE TRUE FALSE FALSE TRUE
```

Data frame có thể được tạo một cách trực tiếp. Trong đoạn code sau, data frame được tạo – đặt tên mỗi vector tạo nên data frame như là một phần của danh sách đối biến.

```
df = data.frame(foo=1:5,bar=c(T,T,F,F,T))
print(df)
  foo   bar
1   1  TRUE
2   2  TRUE
3   3 FALSE
4   4 FALSE
5   5  TRUE
```

8. Bài tập

Bài 1

Định nghĩa

```
> x<-c(4,2,6)
> y<-c(1,0,-1)
```

Đoán kết quả của các lệnh sau đây:

(a) `length(x)`

(b) `sum(x)`
(c) `sum(x^2)`
(d) `x+y`
(e) `x*y`
(f) `x-2`
(g) `x^2`
Dùng **R** để kiểm tra kết quả.

Bài 2

Đoán các dãy sau đây và dùng R để kiểm tra lại:
(a) 7:11
(b) seq(2,9)
(c) seq(4,10,by=2)
(d) seq(3,30,length=10)
(e) seq(6,-4,by=-2)

Bài 3

Xác định các kết quả của các biểu thức R sau đây, và sau đó sử dụng R để kiểm tra:
(a) rep(2,4)
(b) rep(c(1,2),4)
(c) rep(c(1,2),c(4,4))
(d) rep(1:4,4)
(e) rep(1:4,rep(3,4))

Bài 4

Sử dụng hàm `rep` để xác định các vector sau trong R.
(a) 6,6,6,6,6,6
(b) 5,8,5,8,5,8,5,8
(c) 5,5,5,5,8,8,8,8

Bài 5

Nếu `x<- c(5,9,2,3,4,6,7,0,8,12,2,9)` xác định biểu thức sau là gì và sử dụng **R** để kiểm tra lại:
(a) x[2]
(b) x[2:4]
(c) x[c(2,3,6)]
(d) x[c(1:5,10:12)]
(e) x[-(10:12)]

Bài 6

Dữ liệu `y<-c(33,44,29,16,25,45,33,19,54,22,21,49,11,24,56)` chứa doanh thu bán sữa theo lít trong 5 ngày trong ba cửa hàng khác nhau (3 giá trị đầu là cho cửa hàng 1, 2 và 3 vào Thứ Hai, v.v.) Tạo một tóm tắt thống kê về doanh thu cho từng ngày trong tuần và cũng vậy cho mỗi cửa hàng.

(HD: dùng lệnh `summary(y[1:3],...)`

Bài 7

Tạo trong **R** các ma trận

$$x = \begin{bmatrix} 3 & 2 \\ -1 & 1 \end{bmatrix}$$

và

$$y = \begin{bmatrix} 1 & 4 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

Tính biểu thức sau và kiểm tra kết quả trong R:
(a) `2*x`
(b) `x*x`
(c) `x%*%x`
(d) `x%*%y`
(e) `t(y)`
(f) `solve(x)`

Bài 8

(Sử dụng Bài 7)
Với `x` và `y` như trên, tính tác động của các phép toán subscript sau và kiểm tra trong R.
(a) `x[1,]`
(b) `x[2,]`
(c) `x[,2]`
(d) `y[1,2]`
(e) `y[,2:3]`

Bài 9

1. Attach dataset `quakes` và tạo một tóm tắt thống kê về các biến `depth` và `mag`.
2. Attach dataset `mtcars` và tìm trọng lượng trung bình và mức tiêu thụ nhiên liệu trung bình cho các xe trong dataset (gõ `help(mtcars)` để xem mô tả của các biến có sẵn).