Trường Đại học Nông nghiệp Hà nội

Tài liệu tham khảo

Ngôn ngữ R và xử lý thống kê

Nguyễn đình Hiền

Hà nội 2011

Giới thiệu ngôn ngữ R

Năm 1996, trong một bài báo về tính toán thống kê, hai nhà thống kê học Ross Ihaka và Robert Gentleman thuộc Trường đại học Auckland, New Zealand phác hoạ một ngôn ngữ mới cho phân tích thống kê mà họ đặt tên là R . Sáng kiến này được rất nhiều nhà thống kê học trên thế giới tán thành và tham gia vào việc phát triển R.

Cho đến nay càng ngày càng có nhiều nhà thống kê học, toán học, nghiên cứu trong mọi lĩnh vực đã chuyển sang sử dụng R để phân tích dữ liệu khoa học.Trên toàn cầu đã có một mạng lưới hàng triệu người sử dụng R.

R là một phần mềm sử dụng cho phân tích thống kê và vẽ biểu đồ. Thật ra, về bản chất, R là ngôn ngữ máy tính đa năng, có thể sử dụng cho nhiều mục tiêu khác nhau, từ tính toán đơn giản, toán học giải trí, tính toán ma trận (matrix), đến các phân tích thống kê phức tạp. Vì là một ngôn ngữ, cho nên người ta có thể sử dụng R để phát triển thành các phần mềm chuyên môn cho một vấn đề tính toán cá biệt.

Cài đặt R

Để cài đặt R trong máy tính của mình phải truy nhập vào website "Comprehensive R Archive Network" (CRAN) sau đây:

http://cran.R-project.org.

sau đó chọn Cran mirrors thí dụ

http://cran.csiro.au/	CSIRO
http://cran.ms.unimelb.edu.au/	University of Melbourne

Dựa vào vào phiên bảnvà hệ điều hành để chọn tài liệu cần tải về.

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for MacOS X
- Download R for Windows

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2011-07-08): R-2.13.1.tar.gz (read what's new in the latest version).
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.

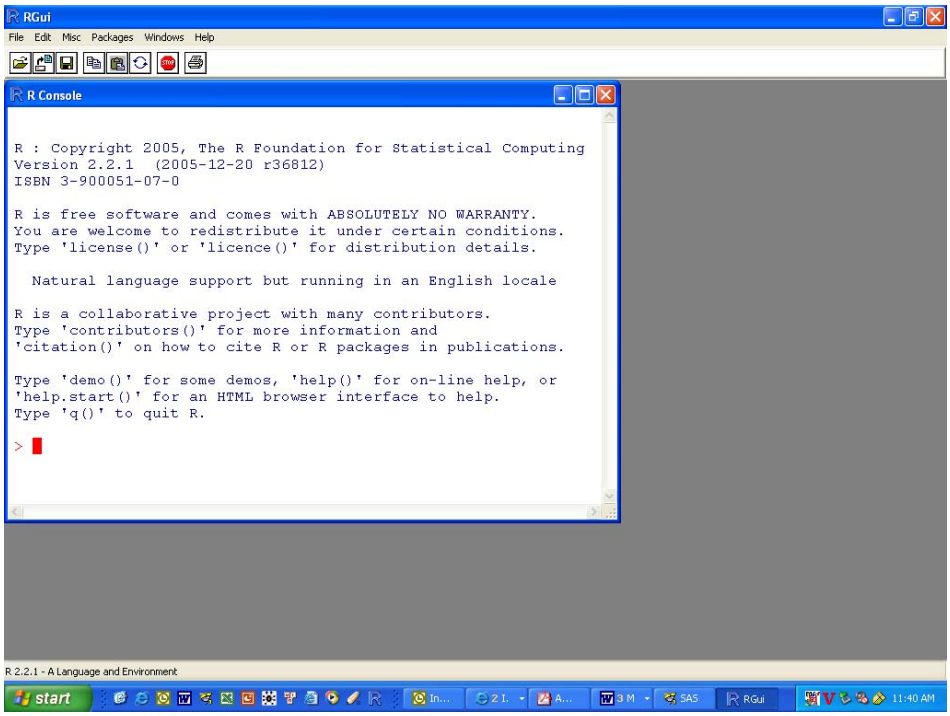
Chẳng hạn như phiên bản mới nhất dùng cho Windows

Tại các website này có thể tìm thấy rất nhiều tài liệu chỉ dẫn cách sử dụng R, đủ trình độ, từ đơn giản đến phức tạp.

Khi đã tải R xuống phải cài đặt vào máy tính. Để làm việc này cần nhấn chuột vào tài liệu trên và làm theo hướng dẫn cách cài đặt trên màn hình.

Sau khi cài đặt nhấp chuột vào biểu tượng R

sẽ có một cửa sổ như sau:



Dấu mời > báo hiệu máy đã sẵn sàng đợi lệnh để thực hiện.

2. Tải các package và cài đặt

R cung cấp một "ngôn ngữ" máy tính và một số *function* để làm các phân tích căn bản và đơn giản. Nếu muốn làm những phân tích phức tạp hơn cần phải tải về máy tính một số *package* khác. Package là một phần mềm nhỏ được các nhà thống kê phát triển để giải quyết một vấn đề cụ thể, và có thể chạy trong hệ thống R. Chẳng hạn như để phân tích hồi qui tuyến tính, R có function lm để sử dụng cho mục đích này, nhưng để làm các phân tích sâu hơn và phức tạp hơn cần đến các package như **lme4**. Các package này

cần tải về và cài đặt. Địa chỉ các package vẫn là: http://cran.r-project.org, bấm vào phần Packages"để tìm, kèm các trang web đẻ tải về. Một số package thường dùngtrong các phân tích thống kê là:

Tên package	**Chức năng**
`trellis`	Dùng để vẽ đồ thị và làm cho đồ thị đẹp hơn
`lattice`	Dùng để vẽ đồ thị và làm cho đồ thị đẹp hơn
agricolae	Statistical Procedures for agricultural Research
`Design`	Một số mô hình thiết kế nghiên cứu của F. Harrell
`Epi`	Dùng cho các phân tích dịch tễ học
`epitools`	Một package khác chuyên cho các phân tích dịch tễ học
`Foreign`	Dùng để nhập dữ liệu từ các phần mềm khác như SPSS, Stata, SAS, v.v…
lme4	Linear mixed effects models
Rcmdr	R commander
`pspearman`	Spearman' s rank correlation test
`survival`	Chuyên dùng cho phân tích theo mô hình Cox (Cox's proportional hazard model)
`Zelig`	Package dùng cho các phân tích thống kê trong lĩnh vực xã hội học
`Genetics`	Package dùng cho phân tích số liệu di truyền học
`BMA`	Bayesian Model Average

Các package này có thể cài đặt trực tuyến bằng cách chọn **Install packages** trong phần **packages** của R. Nếu package đã được tải xuống máy tính việc cài đặt có thể nhanh hơn bằng cách chọn **Install package(s) from local zip file** cũng trong phần **packages .**

3. Văn phạm R

R là một ngôn ngữ tương tác (interactive language), có nghĩa là khi chúng ta ra lệnh, và nếu lệnh đúng "văn phạm", R sẽ "đáp" lại bằng một kết quả. Và tương tác đó tiếp tục cho đến khi đạt được yêu cầu. "Văn phạm" chung của R là một lệnh (command) hay function ("hàm"). Mà đã là hàm thì phải có thông số; cho nên theo sau hàm là những thông số mà chúng ta phải cung cấp. Cú pháp chung của R như sau:

đối tượng <- hàm(thông số 1, thông số 2, …, thông số *n*)

Thí dụ

```
> reg <- lm(y ~ x)
```

thì `reg` là một đối tượng (object), còn `lm` là một hàm, và `y ~ x` là thông số của hàm. Hay:

```
> setwd("d:/nnR/thongke")
```

thì `setwd` là một hàm, còn `"d:/nnR/thongke"` là thông số của hàm.

Để biết một hàm cần có những thông số nào, chúng ta dùng lệnh `args(x)`, (args viết tắt chữ `arguments`) mà trong đó `x` là một hàm chúng ta cần biết:

```
> args(lm)

function (formula, data, subset, weights, na.action, method = "qr",
    model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
    contrasts = NULL, offset, ...)

NULL
```

R là một ngôn ngữ "đối tượng" (object oriented language). Điều này có nghĩa là các dữ liệu trong R được chứa trong object. Định hướng này ảnh hưởng đến cách viết của R. Chẳng hạn như thay vì viết `x = 5` như thông thường chúng ta vẫn viết, thì R yêu cầu viết là `x == 5`.

Đối với R, `x = 5` tương đương với `x <- 5`. Cách viết sau (dùng kí hiệu <−) được khuyến khích hơn là cách viết trước (=). Chẳng hạn như:

```
> x <- rnorm(10)
```

có nghĩa là mô phỏng 10 số liệu và chứa trong object `x`. Chúng ta cũng có thể viết `x = rnorm(10)`.

Một số kí hiệu hay dùng trong R là:

`x == 5`	x bằng 5
`x != 5`	x không bằng 5
`y < x`	y nhỏ hơn x
`x > y`	x lớn hơn y
`z <= 7`	z nhỏ hơn hoặc bằng 7
`p >= 1`	p lớn hơn hoặc bằng 1
`is.na(x)`	Có phải x là biến số trống không (missing value)
`A & B`	A và B (AND)
`A | B`	A hoặc B (OR)
`!`	Không là (NOT)

Với R, tất cả các câu chữ hay lệnh sau kí hiệu # đều không có hiệu ứng, vì # là kí hiệu dành cho người sử dụng thêm vào các ghi chú, ví dụ:

```
> # lệnh sau đây sẽ mô phỏng 10 giá trị normal
> x <- rnorm(10)
```

3.1 Cách đặt tên trong R

Việc đặt tên một đối tượng (object) hay một biến số (variable) trong R khá linh hoạt, vì R không có nhiều giới hạn như các phần mềm khác. Tên một object phải được viết liền nhau (tức không được cách rời bằng một khoảng trống). Chẳng hạn như R chấp nhận myobjectnhưng không chấp nhận my object.

```
> myobject <- rnorm(10)
> my object <- rnorm(10)
```

Error: syntax error in "my object"

Nhưng đôi khi tên myobjectkhó đọc, cho nên chúng ta nên tác rời bằng "." Như my.object.

> my.object <- rnorm(10)

Một điều quan trọng cần lưu ý là R phân biệt mẫu tự viết hoa và viết thường. Cho nên My.objectkhác với my.object. Ví dụ:

> My.object.u <- 15

> my.object.L <- 5

> My.object.u + my.object.L

[1] 20

Một vài điều cần lưu ý khi đặt tên trong R là:

– Không nên đặt tên một biến số hay variable bằng kí hiệu "_" (underscore) như `my_object` hay `my-object`.

– Không nên đặt tên một object giống như một biến số trong một dữ liệu. Ví dụ, nếu chúng ta có một `data.frame` (dữ liệu hay dataset) với biến số `age` trong đó, thì không nên có một object trùng tên `age`, tức là không nên viết: `age <- age`. Tuy nhiên, nếu data.frame tên là `data` thì chúng ta có thể đề cập đến biến số `age` với một kí tự `$` như sau: `data$age`.(Tức là biến số `age` trong data.frame `data`), và trong trường hợp đó, `age <- data$age` có thể chấp nhận được.

3.2 Hỗ trợ trong R

Ngoài lệnh `args()` R còn cung cấp lệnh `help()` để người sử dụng có thể hiểu "văn phạm" của từng hàm. Chẳng hạn như muốn biết hàm `lm` có những thông số (arguments) nào, chúng ta chỉ đơn giản lệnh:

```
> help(lm)
```

hay

```
> ?lm
```

Một cửa sổ sẽ hiện ra bên phải của màn hình chỉ rõ cách sử dụng ra sao và thậm chí có cả ví dụ. Bạn đọc có thể đơn giản copy và dán ví dụ vào R để xem cách vận hành.

Trước khi sử dụng R, ngoài sách này nếu cần bạn đọc có thể đọc qua phần chỉ dẫn có sẵn trong R bằng cách chọn mục `help` và sau đó chọn `Html help` để biết thêm chi tiết. Bạn đọc cũng có thể copy và dán các lệnh trong mục này vào R để xem cho biết cách vận hành của R.

4- Nhập dữ liệu trong R

Muốn phân tích dữ liệu bằng R, chúng ta phải có sẵn dữ liệu ở dạng mà R có thể hiểu được để xử lí. Dữ liệu mà R hiểu được phải là dữ liệu trong một data.frame. Có nhiều

cách để nhập số liệu vào một data.frame trong R, từ nhập trực tiếp đến nhập từ các nguồn khác nhau. Sau đây là những cách thông dụng nhất:

4.1 Nhập số liệu trực tiếp: c()

Ví dụ: chúng ta có số liệu về độ tuổi và insulin cho 10 bệnh nhân như sau, và muốn nhập vào R.

```
50      16.5
62      10.8
60      32.3
40      19.3
48      14.2
47      11.3
57      15.5
70      15.8
48      16.2
67      11.2
```

Chúng ta có thể sử dụng function có tên `c` như sau:

```
> age <- c(50,62, 60,40,48,47,57,70,48,67)
> insulin <- c(16.5,10.8,32.3,19.3,14.2,11.3,15.5,15.8,16.2,11.2)
```

Lệnh thứ nhất cho R biết rằng chúng ta muốn tạo một cột dữ liệu (*biến số, variable*) có tên là age, và lệnh thứ hai là tạo ra một cột khác có tên là insulin.

Chúng ta dùng function c (viết tắt của chữ concatenation – có nghĩa là "ghép") để nhập dữ liệu. Chú ý rằng mỗi số liệu cho mỗi bệnh nhân được cách nhau bằng một dấu phẩy.

Kí hiệu insulin <-(cũng có thể viết là insulin =) có nghĩa là các số liệu theo sau sẽ có nằm trong biến số insulin

R là một ngôn ngữ cấu trúc theo dạng đối tượng ("object-oriented language"), vì mỗi cột số liệu hay mỗi một data.frame là một đối tượng (object) đối với R. Vì thế, age và insulin là hai đối tượng riêng lẻ. Bây giờ chúng ta cần phải nhập hai đối tượng này thành một data.frame để sau này xử lý. Để làm việc này chúng ta cần đến function data.frame:

```
> bang <- data.frame(age, insulin)
```

Trong lệnh này, chúng ta muốn cho R biết rằng nhập hai cột (hay hai đối tượng) `age` và `insulin` vào một đối tượng có tên là `bang`.

Đến đây thì chúng ta đã có một đối tượng hoàn chỉnh để tiến hành phân tích thống kê.

Để kiểm tra xem trong `bang` có gì, chúng ta chỉ cần đơn giản gõ:

```
> bang
```

Và R sẽ cho kết quả:

```
   age insulin
1   50    16.5
2   62    10.8
3   60    32.3
4   40    19.3
5   48    14.2
6   47    11.3
7   57    15.5
8   70    15.8
9   48    16.2
10  67    11.2
```

Nếu chúng ta muốn lưu lại các số liệu này trong một file theo dạng R, chúng ta cần dùng lệnh save. Giả dụ như chúng ta muốn lưu số liệu trong thư mục có tên là "c:\nnR\thongke", chúng ta cần gõ như sau:

```
> setwd("d:/nnR/thongke")

> save(bang, file="bang.rda")
```

Lệnh đầu tiên (setwd– chữ *wd* có nghĩa là *working directory*) cho R biết chúng ta muốn lưu các số liệu trong thư mục có tên là "c:\nnR\thongke".

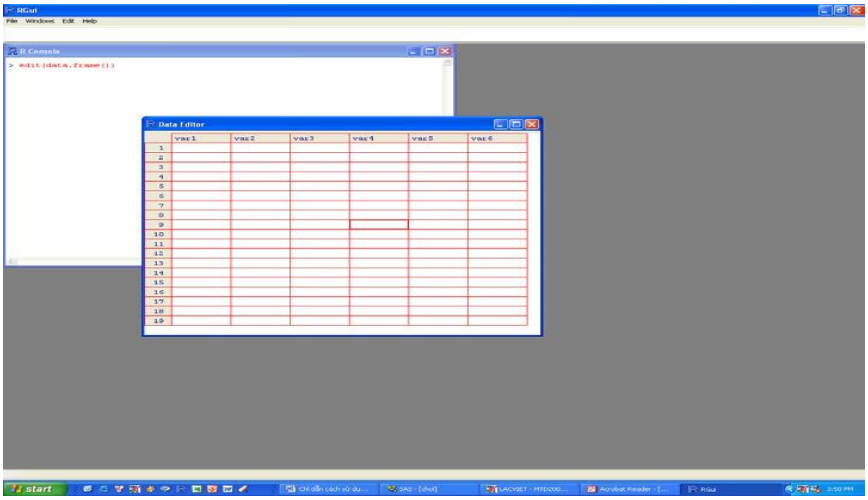
Lệnh thứ hai (save) cho R biết rằng các số liệu trong đối tượng bangsẽ lưu trong một tệp có tên "bang.rda". Sau khi gõ xong hai lệnh trên, một file có tên bang.rdasẽ có mặt trong thư mục "c:\nnR\thongke".

4.2 Nhập số liệu trực tiếp: edit(data.frame())

Ví dụ (tiếp tục): chúng ta có thể nhập số liệu về độ tuổi và insulin cho 10 bệnh nhân bằng một function rất có ích, đó là: edit(data.frame()). Với function này, R sẽ cung cấp cho chúng ta một window mới với một dãy cột và dòng giống như Excel, và chúng ta có thể nhập số liệu trong bảng đó. Ví dụ:

```
> ins <- edit(data.frame())
```

Chúng ta sẽ có một cửa sổ như sau:



Ở đây, R không biết chúng ta có biến số nào, cho nên R liệt kê các biến số var1, var2, v.v… Nhấp chuột vào cột var1và thay đổi bằng cách gõ vào đó age. Nhấp chuột vào cột var2và thay đổi bằng cách gõ vào đó insulin. Sau đó gõ số liệu cho

từng cột. Sau khi xong, bấm nút chéo X ở góc phải của spreadsheet, chúng ta sẽ có một data.frame tên insvới hai biến số agevà insulin.

4.3 Nhập số liệu từ một *text file:* read.table

Ví dụ: Chúng ta thu thập số liệu về độ tuổi và cholesterol từ một nghiên cứu ở

50 bệnh nhân mắc bệnh cao huyết áp. Các số liệu này được lưu trong một text file có tên là chol.txttại directory c:\nnR\thongke. Số liệu này như sau: cột 1 là mã số của bệnh nhân, cột 2 là giới tính, cột 3 là body mass index (bmi), cột 4 là HDL cholesterol (viết tắt là hdl), kế đến là LDL cholesterol, total cholesterol (tc) và triglycerides (tg).

id	sex	age	bmi	hdl	ldl	tc	tg
1	Nam	57	17	5.000	2.0	4.0	1.1
2	Nu	64	18	4.380	3.0	3.5	2.1
3	Nu	60	18	3.360	3.0	4.7	0.8
4	Nam	65	18	5.920	4.0	7.7	1.1
5	Nam	47	18	6.250	2.1	5.0	2.1
6	Nu	65	18	4.150	3.0	4.2	1.5
7	Nam	76	19	0.737	3.0	5.9	2.6
8	Nam	61	19	7.170	3.0	6.1	1.5
9	Nam	59	19	6.942	3.0	5.9	5.4
10	Nu	57	19	5.000	2.0	4.0	1.9
...							
46	Nu	52	24	3.360	2.0	3.7	1.2
47	Nam	64	24	7.170	1.0	6.1	1.9
48	Nam	45	24	7.880	4.0	6.7	3.3
49	Nu	64	25	7.360	4.6	8.1	4.0
50	Nu	62	25	7.750	4.0	6.2	2.5

Chúng ta muốn nhập các dữ liệu này vào R để tiện việc phân tích sau này. Dùng lệnh `read.table` như sau:

```
> setwd("d:/nnR/thongke")

> chol <- read.table("chol.txt", header=TRUE)
```

Lệnh thứ nhất đảm bảo R truy nhập đúng directory mà số liệu đang được lưu giữ. Lệnh thứ hai yêu cầu R nhập số liệu từ file có tên là "`chol.txt`" (trong directory c:\works\insulin và ghi vào bảng chol.Trong lệnh này, `header=TRUE` có nghĩa là dòng đầu tiên trong file là tên của các cột dữ liệu.

Chúng ta có thể kiểm tra xem R đã đọc đúngcác dữ liệu hay chưa bằng lệnh:

```
> chol
```

Hay

```
> names(chol)
```

R sẽ cho biết có các cột như sau trong dữ liệu (`names` là lệnh hỏi trong dữ liệu có những cột nào và tên gì):

```
 [1] "id" "sex" "age" "bmi" "hdl" "ldl" "tc"  "tg"
```

Bây giờ chúng ta có thể lưu dữ liệu dưới dạng R để xử lí sau này bằng cách ra lệnh:

```
> save(chol, file="chol.rda")
```

4.4 Nhập số liệu từ *Excel:* read.csv

Để nhập số liệu từ phần mềm Excel, chúng ta cần tiến hành 2 bước:

- Bước 1: Dùng lệnh "Save as" trong Excel và lưu số liệu dưới dạng "csv";
- Bước 2: Dùng R (lệnh read.csv) để nhập dữ liệu dạng csv.

Ví dụ: Một dữ liệu gồm các cột sau đây đang được lưu trong Excel, và chúng ta muốn chuyển vào R để phân tích. Dữ liệu này có tên là excel.xls.

ID	Age	Sex	Ethnicity	IGFI	IGFBP3	ALS	PINP	ICTP	P3NP
1	18	1	1	148.27	5.14	316.00	61.84	5.81	4.21
2	28	1	1	114.50	5.23	296.42	98.64	4.96	5.33
3	20	1	1	109.82	4.33	269.82	93.26	7.74	4.56
4	21	1	1	112.13	4.38	247.96	101.59	6.66	4.61
5	28	1	1	102.86	4.04	240.04	58.77	4.62	4.95
6	23	1	4	129.59	4.16	266.95	48.93	5.32	3.82
7	20	1	1	142.50	3.85	300.86	135.62	8.78	6.75
8	20	1	1	118.69	3.44	277.46	79.51	7.19	5.11
9	20	1	1	197.69	4.12	335.23	57.25	6.21	4.44
10	20	1	1	163.69	3.96	306.83	74.03	4.95	4.84
11	22	1	1	144.81	3.63	295.46	68.26	4.54	3.70
12	27	0	2	141.60	3.48	231.20	56.78	4.47	4.07
13	26	1	1	161.80	4.10	244.80	75.75	6.27	5.26
14	33	1	1	89.20	2.82	177.20	48.57	3.58	3.68
15	34	1	3	161.80	3.80	243.60	50.68	3.52	3.35
16	32	1	1	148.50	3.72	234.80	83.98	4.85	3.80
17	28	1	1	157.70	3.98	224.80	60.42	4.89	4.09
18	18	0	2	222.90	3.98	281.40	74.17	6.43	5.84
19	26	0	2	186.70	4.64	340.80	38.05	5.12	5.77
20	27	1	2	167.56	3.56	321.12	30.18	4.78	6.12

Vào Excel, chọn File Save as

Chọn Save as type“CSV (Comma delimited)”

Sau khi xong, chúng ta sẽ có một file với tên “excel.csv” trong directory “d:\nnR\thongke”.

Việc thứ hai là vào R và ra những lệnh sau đây:

> setwd(“d:/nnR/thongke”)

> gh <- read.csv ("excel.csv", header=TRUE)

Lệnh thứ hai read.csvyêu cầu R đọc số liệu từ “excel.csv”, dùng dòng thứ nhất là tên cột, và lưu các số liệu này trong một object có tên là gh.

Bây giờ chúng ta có thể lưu ghdưới dạng R để xử lí sau này bằng lệnh sau đây:

> save(gh, file="gh.rda")

4.5 Nhập số liệu từ một tệp trong *SPSS:* read.spss

Phần mềm thống kê SPSS lưu dữ liệu dưới dạng “sav”. Chẳng hạn như nếu chúng ta có một dữ liệu có tên là testo.sav trong directory c:\works\insulin, và muốn chuyển dữ liệu này sang dạng R thì phải sử dụng lệnh read.spss trong package có tên là foreign. Các lệnh sau đây sẽ hoàn tất dễ dàng việc này:

Việc đầu tiên chúng ta cho truy nhập foreign bằng lệnh library:

> library(foreign)

Việc thứ hai là lệnh `read.spss`:

> setwd(“d:/nnR/thongke”)

> testo <- read.spss(“testo.sav”, to.data.frame=TRUE)

Lệnh thứ hai `read.spss` yêu cầu R đọc số liệu từ “testo.sav”, và cho vào một `data.frame` có tên là testo.

Bây giờ chúng ta có thể lưu testo dưới dạng R để xử lí sau này bằng lệnh sau đây:

> save(testo, file="testo.rda")

4.6 Thông tin về dữ liệu

Giả dụ như chúng ta đã nhập số liệu vào một data.frame có tên là chol như trong ví dụ

1. Chúng ta có thể nhập chol vào R như sau:

> attach(chol)

Kiểm tra xem chol có phải là một data.frame không bằng lệnh is.data.frame(arg)với arglà tên của dữ liệu. Ví dụ:

> is.data.frame(chol) [1] TRUE

R cho biết cholquả là một data.frame.

Đẻ biết có bao nhiêu cột (hay *variable = biến số*) và bao nhiêu dòng số liệu (observations) trong dữ liệu này? Dùng lệnh dim(arg)với arglà tên của dữ liệu (dim viết tắt chữ dimension

```
 dim(chol) [1]
50      8
```

Như vậy, chúng ta có 50 dòng và 8 cột (hay biến số). Vậy những biến số này tên gì? Chúng ta dùng lệnh `names(arg)` với `arg` là tên của dữ liệu. Ví dụ:

```
> names(chol)
        [1] "id""sex" "age" "bmi" "hdl" "ldl" "tc"      "tg"
```

Trong biến số `sex` có bao nhiêu nam và nữ? Để trả lời câu hỏi này, chúng ta có thể dùng lệnh `table(arg)` với `arg` là tên của biến số. Ví dụ:

```
> table(sex)
sex
nam Nam    Nu
 1    21   28
```

Kết quả cho thấy dữ liệu này có 21 nam và 28 nữ.

5. Biên tập số liệu

5.1 Tách dữ liệu thành các nhóm con: `subset`

```
> setwd("d:/nnR/thongke")
> chol <- read.table("chol.txt", header=TRUE)
> attach(chol)
```

Nếu muốn phân tích riêng cho nam giới, chúng ta có thể tách `chol` ra thành hai data.frame, tạm gọi là `nam` và `nu`. Để làm chuyện này, chúng ta dùng lệnh `subset(data, cond)`, trong đó `data` là data.frame mà chúng ta muốn tách rời và `cond` là điều kiện. Ví dụ:

```
> nam <- subset(chol, sex=="Nam")
> nu <- subset(chol, sex=="Nu")
```

Sau khi ra hai lệnh này, chúng ta đã có 2 dữ liệu (hai data.frame) mới tên là `nam` và `nu`. Chú ý điều kiện `sex == "Nam"` và `sex == "Nu"` chúng ta dùng == thay vì = để chỉ điều kiện chính xác.

Tất nhiên, chúng ta cũng có thể tách dữ liệu thành nhiều data.frame khác nhau với những điều kiện dựa vào các biến số khác. Chẳng hạn như lệnh sau đây tạo ra một data.frame mới tên là `old` với những bệnh nhân trên 60 tuổi:

```
> old <- subset(chol, age>=60)
> dim(old)
  [1] 25  8
```

Hay một data.frame mới với những bệnh nhân trên 60 tuổi và nam giới:

```
> n60 <- subset(chol, age>=60 & sex=="Nam")
> dim(n60)
[1] 9  8
```

5.2 Chiết số liệu từ một data .frame

Trong `chol` có 8 biến số. Chúng ta có thể chiết dữ liệu `chol` và chỉ giữ lại những biến số cần thiết như mã số (`id`), độ tuổi (`age`) và total cholestrol (`tc`). Trong lệnh `names(chol)` biến số `id` là cột số 1, `age` là cột số 3, và biến số `tc` là cột số7. Chúng ta có thể dùng lệnh sau đây:

```
> data2 <- chol[, c(1,3,7)]
```

Ở đây, chúng ta lệnh cho R biết rằng chúng ta muốn chọn cột số 1, 3 và 7, và đưa tất cả số liệu của hai cột này vào data.frame mới có tên là `data2`. Chú ý chúng ta sử dụng ngoặc kép vuông [] chứ không phải ngoặc kép vòng (), vì `chol` không phải làm một function. Dấu phẩy phía trước `c`, có nghĩa là chúng ta chọn tất cả các dòng số liệu trong data.frame `chol`.

Nhưng nếu chúng ta chỉ muốn chọn 10 dòng số liệu đầu tiên, thì lệnh sẽ là:

```
> data3 <- chol[1:10, c(1,3,7)]
> print(data3)
   id sex  tc
1   1 Nam 4.0
2   2  Nu 3.5
3   3  Nu 4.7
4   4 Nam 7.7
5   5 Nam 5.0
6   6  Nu 4.2
7   7 Nam 5.9
8   8 Nam 6.1
9   9 Nam 5.9
10 10 Nu 4.0
```

Chú ý lệnh `print(arg)` đơn giản liệt kê tất cả số liệu trong data.frame `arg`. Thật ra, chúng ta chỉ cần đơn giản gõ `data3`, kết quả cũng giống y như `print(data3)`.

5.3 Nhập hai data.frame thành một: `merge`

Giả dụ như chúng ta có dữ liệu chứa trong hai data.frame. Dữ liệu thứ nhất tên là `d1` gồm 3 cột: `id, sex, tc` như sau:

id	sex	tc
1	Nam	4.0
2	Nu	3.5
3	Nu	4.7
4	Nam	7.7
5	Nam	5.0
6	Nu	4.2
7	Nam	5.9
8	Nam	6.1
9	Nam	5.9
10	Nu	4.0

Dữ liệu thứ hai tên là d2gồm 3 cột: id, sex, tg như sau:

	sex	tg
1	Nam	1.1

```
 2   Nu 2.1
 3   Nu 0.8
 4  Nam 1.1
 5  Nam 2.1
 6   Nu 1.5
 7  Nam 2.6
 8  Nam 1.5
 9  Nam 5.4
10   Nu 1.9
11   Nu 1.7
```

Hai dữ liệu này có chung hai biến số `id` và `sex`. Nhưng dữ liệu `d1` có 10 dòng, còn dữ liệu `d2` có 11 dòng. Chúng ta có thể nhập hai dữ liệu thành một data.frame bằng cách dùng lệnh `merge` như sau:

```
> d <- merge(d1, d2, by="id", all=TRUE)
> d
   id sex.x  tc sex.y  tg

1   1   Nam 4.0   Nam 1.1
2   2    Nu 3.5    Nu 2.1
3   3    Nu 4.7    Nu 0.8
4   4   Nam 7.7   Nam 1.1
5   5   Nam 5.0   Nam 2.1
6   6    Nu 4.2    Nu 1.5
7   7   Nam 5.9   Nam 2.6
8   8   Nam 6.1   Nam 1.5
9   9   Nam 5.9   Nam 5.4
10 10    Nu 4.0    Nu 1.9
11 11  <NA>  NA    Nu 1.7
```

Trong lệnh `merge`, chúng ta yêu cầu R nhập 2 dữ liệu `d1` và `d2` thành một và đưa vào data.frame mới tên là `d`, và dùng biến số `id` làm chuẩn. Chúng ta để ý thấy bệnh nhân số 11 không có số liệu cho `tc`, cho nên R cho là `NA` (một dạng "not available").

5.4 Biến đổi số liệu (data coding)

Trong việc xử lí số liệu dịch tễ học, nhiều khi chúng ta cần phải biến đổi số liệu từ biến liên tục sang biến mang tính cách phân loại. Chẳng hạn như trong chẩn đoán loãng xương, những phụ nữ có chỉ số T của mật độ chất khoáng trong xương (bone mineral density hay BMD) bằng hay thấp hơn -2.5 được xem là "loãng xương", những ai có BMD giữa -2.5 và -1.0 là "xốp xương" (osteopenia), và trên -1.0 là "bình thường". Ví dụ, chúng ta có số liệu BMD từ 10 bệnh nhân như sau:

```
-0.92, 0.21, 0.17, -3.21, -1.80, -2.60, -2.00, 1.71, 2.12, -2.11
```

Để nhập các số liệu này vào R chúng ta có thể sử dụng *function* `c` như sau:

```
bmd <- c(-0.92,0.21,0.17,-3.21,-1.80,-2.60,-2.00,1.71,2.12,-2.11)
```

Để phân loại 3 nhóm loãng xương, xốp xương, và bình thường, chúng ta có thể dùng mã số 1, 2 và 3. Nói cách khác, chúng ta muốn tạo nên một biến số khác (hãy gọi là `diagnosis`) gồm 3 giá trị trên dựa vào giá trị của `bmd`. Để làm việc này, chúng ta sử dụng lệnh:

```
# tạm thời cho biến số diagnosis bằng bmd
> diagnosis <- bmd
# biến đổi bmd thành diagnosis
> diagnosis[bmd <= -2.5] <- 1
> diagnosis[bmd > -2.5 & bmd <= 1.0] <- 2
> diagnosis[bmd > -1.0] <- 3
# tạo thành một data frame
> data <- data.frame(bmd, diagnosis)
# liệt kê để kiểm tra xem lệnh có hiệu quả không
> data
```

	Bmd	Diagnosis
1	-0.92	3
2	0.21	3
3	0.17	3
4	-3.21	1
5	-1.80	2
6	-2.60	1
7	-2.00	2
8	1.71	3
9	2.12	3
10	-2.11	2

5.5 Biến đổi số liệu bằng cách dùng *replace*

Một cách biến đổi số liệu khác là dùng `replace`, dù cách này có vẻ rườm rà chút ít. Tiếp tục ví dụ trên, chúng ta biến đổi từ `bmd` sang `diagnosis` như sau:

```
> diagnosis <- bmd
> diagnosis <- replace(diagnosis, bmd <= -2.5, 1)
> diagnosis <- replace(diagnosis, bmd > -2.5 & bmd <= 1.0, 2)
> diagnosis <- replace(diagnosis, bmd > -1.0, 3)
```

5.6 Biến đổi thành yếu tố (*factor*)

Trong phân tích thống kê, chúng ta phân biệt một biến số mang tính *yếu tố* (factor) và biến số liên tục bình thường. Biến số yếu tố không thể dùng để tính toán như cộng trừ nhân chia, nhưng biến số số học có thể sử dụng để tính toán. Chẳng hạn như trong ví dụ `bmd` và `diagnosis` trên, `diagnosis` là yếu tố vì giá trị trung bình giữa 1 và 2 chẳng có ý nghĩa thực tế gì cả; còn `bmd` là biến số số học.

Nhưng hiện nay, `diagnosis` được xem là một biến số số học. Để biến thành biến số yếu tố, chúng ta cần sử dụng *function* `factor` như sau:

```
> diag <- factor(diagnosis)
> diag
[1] 3 3 3 1 2 1 2 3 3 2
Levels: 1 2 3
```

Chú ý R bây giờ thông báo cho chúng ta biết diag có 3 bậc: 1, 2 và 3. Nếu chúng ta yêu cầu R tính số trung bình của diag, R sẽ không làm theo yêu cầu này, vì đó không phải là một biến số số học:

> mean(diag) [1]
NA
Warning message:
argument is not numeric or logical: returning NA in: mean.default(diag)
Trung bình của diagnosis:

```
> mean(diagnosis)
[1] 2.3
```

nhưng kết quả 2.3 này không có ý nghĩa gì trong thực tế cả.

6. Sử dụng R cho tính toán đơn giản

Một trong những lợi thế của R là có thể sử dụng như một … máy tính cầm tay. Thật ra, hơn thế nữa, R có thể sử dụng cho các phép tính ma trận và lập chương. Trong chương này tôi chỉ trình bày một số phép tính đơn giản mà học sinh hay sinh viên có thể sử dụng lập tức trong khi đọc những dòng chữ này.

6.1 Tính toán đơn giản

Cộng hai số hay nhiều số với nhau: > 15+2997 [1] 3012	Cộng và trừ: > 15+2997-9768 [1] -6756
Nhân và chia > -27*12/21 [1] -15.42857	Số lũy thừa: $(25 – 5)^3$ > (25 - 5)^3 [1] 8000
Căn số bậc hai: 10 > sqrt(10) [1] 3.162278	Số pi (π) > pi [1] 3.141593 > 2+3*pi [1] 11.42478
Logarit: $\log_e$ > log(10) [1] 2.302585	Logarit: $\log_{10}$ > log10(100) [1] 2
Số mũ: $e^{2.7689}$ > exp(2.7689) [1] 15.94109 > log10(2+3*pi) [1] 1.057848	Hàm số lượng giác > cos(pi) [1] -1
Vector > x <- c(2,3,1,5,4,6,7,6,8) > x [1] 2 3 1 5 4 6 7 6 8 > sum(x) [1] 42 > x*2 [1] 4 6 2 10 8 12 14 12 16	> exp(x/10) [1] 1.221403 1.349859 1.105171 1.648 1.491825 1.822119 2.013753 1.822119 2.225541 > exp(cos(x/10)) [1] 2.664634 2.599545 2.704736 2.405 2.511954 2.282647 2.148655 2.282647 2.007132

Tính tổng bình phương (sum of squares): $1^2 + 2^2 + 3^2 + 4^2 + 5^2 = ?$ > x <- c(1,2,3,4,5) > sum(x^2) [1] 55	Tính tổng bình phương điều chỉnh (adjusted sum of squares): $\sum_{i=1}^{n}(x_i - \bar{x})^2$ > x <- c(1,2,3,4,5) > sum((x-mean(x))^2) [1] 10 Trong công thức trên mean(x)là số trung bình của vector x
Tính sai số bình phương (mean square):	Tính phương sai (variance) và độ lệch chuẩn (standard deviation):