4
Biên tập dữ liệu

Biên tập số liệu ở đây không có nghĩa là thay đổi số liệu gốc (vì đó là một tội lớn, một sự gian dối trong khoa học không thể chấp nhận được), mà chỉ có nghĩa tổ chức số liệu sao cho R có thể phân tích một cách hữu hiệu. Nhiều khi trong phân tích thống kê, chúng ta cần phải tập trung số liệu thành một nhóm, hay tách rời thành từng nhóm, hay thay thế từ kí tự (characters) sang số (numeric) cho tiện việc tính toán. Trong chương này, tôi sẽ bàn qua một số lệnh căn bản cho việc biên tập số liệu.

Chúng ta sẽ quay lại với dữ liệu `chol` trong ví dụ 1. Để tiện việc theo dõi và hiểu "câu chuyện", tôi xin nhắc lại rằng chúng ta đã nhập số liệu vào trong một dữ liệu R có tên là `chol` từ một text file có tên là `chol.txt`:

```
> setwd("c:/works/stats")
> chol <- read.table("chol.txt", header=TRUE)
> attach(chol)
```

4.1 Kiểm tra số liệu trống không (`missing value`)

Trong nghiên cứu, vì nhiều lí do số liệu không thể thu thập được cho tất cả đối tượng, hay không thể đo lường tất cả biến số cho một đối tượng. Trong trường hợp đó, số liệu trống được xem là "missing value" (mà tôi tạm dịch là số liệu trống không). R xem các số liệu trống không là `NA`. Có một số kiểm định thống kê đòi hỏi các số liệu trống không phải được loại ra (vì không thể tính toán được) trước khi phân tích. R có một lệnh rất có ích cho việc này: `na.omit`, và cách sử dụng như sau:

```
> chol.new <- na.omit(chol)
```

Trong lệnh trên, chúng ta yêu cầu R loại bỏ các số liệu trống không trong data.frame `chol` và đưa các số liệu không trống vào data.frame mới tên là `chol.new`. Chú ý lệnh trên chỉ là ví dụ, vì trong dữ liệu `chol` không có số liệu trống không.

4.2 Tách rời dữ liệu: `subset`

Nếu chúng ta, vì một lí do nào đó, chỉ 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
```

4.3 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`). Để ý từ lệnh `names(chol)` rằng biến số `id` là cột số 1, `age` là cột số 3, và biến số `tc` là cột số 7. 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)`.

4.4 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 `d2` gồm 3 cột: `id, sex, tg` như sau:

```
id 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").

4.5 Mã hóa 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
```

4.5.1 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)
```

4.5.2 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)
```

Dĩ nhiên, chúng ta có thể tính giá trị trung bình của `diagnosis`:

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

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

4.6 Chia nhóm bằng *cut*

Với một biến liên tục, chúng ta có thể chia thành nhiều nhóm bằng hàm `cut`. Ví dụ, chúng ta có biến `age` như sau:

```
> age <- c(17,19,22,43,14,8,12,19,20,51,8,12,27,31,44)
```

Độ tuổi thấp nhất là 8 và cao nhất là 51. Nếu chúng ta muốn chia thành 2 nhóm tuổi:

```
> cut(age, 2)

[1] (7.96,29.5] (7.96,29.5] (7.96,29.5] (29.5,51]   (7.96,29.5] (7.96,29.5]
(7.96,29.5] (7.96,29.5]

[9] (7.96,29.5] (29.5,51]   (7.96,29.5] (7.96,29.5] (7.96,29.5] (29.5,51]
(29.5,51]

Levels: (7.96,29.5] (29.5,51]
```

`cut` chia biến `age` thành 2 nhóm: nhóm 1 tuổi từ 7.96 đến 29.5; nhóm 2 từ 29.5 đến 51. Chúng ta có thể đếm số đối tượng trong từng nhóm tuổi bằng hàm `table` như sau:

```
> table(cut(age, 2))

(7.96,29.5]   (29.5,51]
         11           4
```

```
> ageg <- cut(age, 3, labels=c("low", "medium", "high"))
[1] low    low    low    high   low    low    low    low    low    high
low    low    medium medium
[15] high
Levels: low medium high
```

```
> ageg <- cut(age, 3, labels=c("low", "medium", "high"))
> table(ageg)
ageg
   low medium   high
    10      2      3
```

Tất nhiên, chúng ta cũng có thể chia age thành 4 nhóm (quartiles) bằng cách cho những thông số 0, 0.25, 0.50 và 0.75 như sau:

```
cut(age,
    breaks=quantiles(age, c(0, 0.25, 0.50, 0.75, 1)),
    labels=c("q1", "q2", "q3", "q4"),
```

```
    include.lowest=TRUE)


cut(age,
    breaks=quantiles(c(0, 0.25, 0.50, 0.75, 1)),
    labels=c("q1", "q2", "q3", "q4"),
    include.lowest=TRUE)
```

4.7. Tập hợp số liệu bằng *cut2* (Hmisc)

Hàm `cut` trên chia biến số theo giá trị của biến, chứ không dựa vào số mẫu, cho nên số lượng mẫu trong từng nhóm không bằng nhau. Tuy nhiên, trong phân tích thống kê, có khi chúng ta cần phải phân chia một biến số liên tục thành nhiều nhóm dựa vào phân phối của biến số nhưng số mẫu bằng hay tương đương nhau. Chẳng hạn như đối với biến số `bmd` chúng ta có thể "cắt" dãy số thành 3 nhóm với số mẫu tương đương nhau bằng cách dùng function `cut2` (trong thư viện `Hmisc`) như sau:

```
> # nhập thư viện Hmisc để có thể dùng function cut2

> library(Hmisc)

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

> # chia biến số bmd thành 2 nhóm và để trong đối tượng group

> group <- cut2(bmd, g=2)

> table(group)
group
[-3.21,-0.92) [-0.92, 2.12]
            5             5
```

Như thấy qua ví dụ trên, `g = 2` có nghĩa là chia thành 2 nhóm (g = group). R tự động chia thành nhóm 1 gồm giá trị `bmd` từ -3.21 đến -0.92, và nhóm 2 từ -0.92 đến 2.12. Mỗi nhóm gồm có 5 số.

Tất nhiên, chúng ta cũng có thể chia thành 3 nhóm bằng lệnh:

```
> group <- cut2(bmd, g=3)
```

Và với lệnh table chúng ta sẽ biết có 3 nhóm, nhóm 1 gồm 4 số, nhóm 2 và 3 mỗi nhóm có 3 số:

```
> table(group)
group
[-3.21,-1.80) [-1.80, 0.21) [ 0.21, 2.12]
            4             3             3
```