**3**

**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 thành nhiều nhóm, hay thay thế từ kí tự (characters) sang số (numeric) cho tiện việc tính toán. Chương này sẽ bàn qua một số lệnh căn bản cho việc biên tập số liệu.

**3.1 Phân biệt biến số**

Cũng như các chương trình phân tích dữ liệu khác (như SAS, SPSS, Stata), R phân biệt số liệu theo các thể loại. Có 2 thể loại chính:

**Biến liên tục** (continuous variable): Trong R, các biến này được gọi là *numeric*. Ví dụ như độ tuổi, chiều cao, cân nặng, v.v. được xem là biến liên tục. Chúng ta có thể tính các chỉ số thống kê như số trung bình, độ lệch chuẩn, bách phân vị trên các biến liên tục.

**Biến phân nhóm** (categorical variable): Trong R, các biến này được gọi là *factor*. Ví dụ tiêu biểu là giới tính có giá trị "Male" và "Female" được xem là biến định danh hay biến phân nhóm. Dĩ nhiên, chúng ta không thể tính trung bình và độ lệch chuẩn cho các biến này, mà chỉ có thể tính tỉ lệ (phần trăm).

Cần phải nói thêm rằng, các biến tuy có giá trị là số (như 1, 2, 3) nhưng cũng có thể là biến phân nhóm. Các biến này, vì có giá trị số, nên chúng ta có thể tính các chỉ số trung bình, độ lệch chuẩn, dù kết quả tính toán hoàn toàn không có ý nghĩa. Để báo cho R biết các biến này là *factor,* chúng ta cần phải dùng hàm as.factor*.* Ví dụ, chúng ta có biến education phản ảnh trình độ học vấn: 1 là tiểu học, 2 là trung học, và 3 là đại học; để hoán chuyển education sang biến factor, chúng ta có thể tạo ra một biến mới, tạm gọi là edu như sau:

edu = as.factor(education)

**3.2 Làm việc với biến số và dấu $**

Một cách ngắn gọn, mỗi data.frame trong R là một ma trận. Ma trận có dòng và cột. Do đó, một dataframe có nhiều dòng (observation hay row) và nhiều cột (variable hay column). Ví dụ như dataframe data có 4 dòng và 3 biến số:

age sex weight

1 18 M 60.3

2 21 F 48.5

3 35 M 62.0

4 50 F 47.2

Khi chúng ta gõ dim(data) sẽ có kết quả 4 dòng và 3 cột:

> dim(data)

[1] 4 3

Chúng ta có thể đề cập đến mỗi giá trị trong dataframe bằng công thức ma trận. Nên nhớ là trong ma trận, dòng được đề cập trước và cột sau. Chẳng hạn như (chú ý ngoặc vuông):

data[1, 3] là giá trị 60.3 (dòng 1, cột 3)

data[4, 2] là giá trị F (dòng 4, cột 2)

data[4, ] báo cáo số liệu dòng số 4

data[, 2] báo cáo số liệu của cột số 2

**Dấu dollar ($)**

Mỗi dataframe có nhiều cột, và dấu $ dùng để đề cập tên biến gắn liền với một dataframe. Chẳng hạn như khi chúng ta viết data$age, thì cú pháp này có nghĩa là age là tên biến số trong dataframe data.

Cách dùng $ rất quan trọng vì nếu có nhiều dataframe cùng được vận hành, thì dấu $ dùng để phân biệt biến số nào thuộc về dataframe nào. Ví dụ sau đây sẽ thấy hiệu quả của việc dùng dấu $. Trong ví dụ thứ nhất, chúng ta có 4 biến số (v1, v2, v3, v4) và tính tổng số (sum):

v1 = c(1,3,5)

v2 = c(3,4,7)

v3 = c(6,7,8)

v4 = c(7,9,0)

dat = data.frame(v1,v2,v3,v4)

attach(dat)

**dat$sum** = dat$v1 + dat$v3

**sum1 = v1 + v3**

dat

Kết quả là:

> dat

v1 v2 v3 v4 sum

1 1 3 6 7 7

2 3 4 7 9 10

3 5 7 8 0 13

Chú ý là dataframe dat không có biến sum1!

**3.3 Mã hoá (coding)**

Trong khi phân tích dữ liệu, chúng ta có khi phải mã hoá dữ liệu. Mã hoá là một hình thức hoán chuyển dữ liệu. Chẳng hạn như trong dataframe data trên, thay vì sex có giá trị là "F" và "M", chúng ta có thể mã hoá thành số 1 và 2 và đặt trong một biến mới có tên là gender.

Các hàm dưới đây tạo ra một dataframe có tên là data:

age = c(18, 21, 35, 50)

sex = c("M", "F", "M", "F")

weight = c(60.3, 48.5, 62.0, 47.2)

data = data.frame(age, sex, weight)

Chúng ta muốn tạo ra một biến mới có tên là gender (gắn liền với data), với điều kiện là nếu sex là "M" thì gender là 1, và sex là "F" thì gender là 2:

data$gender[sex=="M"] <- 1

data$gender[sex=="F"] <- 2

data

Trong ví dụ trên (biến có hai giá trị) chúng ta cũng có thể dùng hàm ifelse đơn giản hơn:

data$gender = ifelse(data$sex=="M", 1, 2)

Hàm trên có thể đọc là: "nếu sex có giá trị 'M' thì gender có giá trị 1; còn lại là gender có giá trị 2".

Kết quả là data có thêm một biến mới như sau:

> data

age sex weight gender

1 18 M 60.3 1

2 21 F 48.5 2

3 35 M 62.0 1

4 50 F 47.2 2

Tương tự, chúng ta có thể tạo ra biến mới gọi là agegroup (nhóm tuổi) theo điều kiện trên và dưới 20 tuổi như sau:

data$agegroup[age <= 20] <- "Under 20"

data$agegroup[age > 20] <- "Over 20"

Lệnh trên có nghĩa là biến agegroup là một biến phân nhóm có hai nhóm dưới 20 tuổi và trên 20 tuổi:

> data

age sex weight gender agegroup

1 18 M 60.3 1 Under 20

2 21 F 48.5 2 Over 20

3 35 M 62.0 1 Over 20

4 50 F 47.2 2 Over 20

Thỉnh thoảng, chúng ta cần phân chia một biến liên tục thành nhiều nhóm. Chẳng hạn như biến *body mass index* (bmi) là một biến liên tục, có thể chia thành 3 nhóm: thiếu cân, bình thường, quá cân, và béo phì theo tiêu chuẩn:

bmi < 18.5 là thiếu cân

bmi từ 18.5 đến 24.9 là bình thường

bmi từ 25.0 đến 29.9 là quá cân

bmi bằng hoặc trên 30 là béo phì

Trong R, chúng ta cũng có thể tạo ra một biến mới, tạm gọi là "obesity" như sau:

obesity[bmi < 18.5] = "Underweight"

obesity[bmi >= 18.5 & bmi <= 24.9] = "Normal"

obesity[bmi >= 25.0 & bmi <= 29.9] = "Overweight"

obesity[bmi >= 30.0] = "Obese"

Chú ý biến *obesity* là một *factor*.

**ifelse**

Đối với phân thành 2 nhóm, cú pháp đơn giản hơn. Chỉ cần ifelse là đủ. Ví dụ: chúng ta muốn chia bmi thành hai nhóm béo phì và không béo phì theo tiêu chuẩn như sau:

nếu bmi >= 30 thì obesity là béo phì

nếu bmi < 30 thì obesity là không béo phì

Cách triển khai trong R là:

obesity = iflese(bmi >= 30.0, "Obese", "Non-obese")

**3.4 Làm việc với data frame**

**3.4.1. Hàm subset**

Trong phân tích dữ liệu, có khi do nhu cầu khai thác, chúng ta chỉ làm việc với một phần của dữ liệu. Hàm subset trong R có thể giúp chúng ta chọn phần của dữ liệu để phân tích. Chẳng hạn như trong dataframe data trên, chúng ta muốn làm việc trên nhóm nữ và tạo ra một dataframe mới tạm gọi là fem:

fem = subset(data, sex=="F")

Hàm dưới đây tạo ra một dataframe mới với điều kiện là nam trên 20 tuổi:

male20 = subset(data, sex=="M" & age>20)

**3.4.2. Hàm merge**

Trong nhiều trường hợp chúng ta có hai hay nhiều hơn hai dataframe, và chúng ta muốn hợp nhất thành một dataframe để tiện phân tích. Dĩ nhiên, hai dữ liệu phải có chung một biến số để hợp nhất. Ví dụ như chúng ta có dat1 dưới đây gồm 4 dòng và 2 cột:

id = c(1,2,3,4)

sex = c("M","F","M","F")

dat1 = data.frame(id,sex)

và dat2 gồm có 5 dòng với 2 cột. Chú ý cả hai dataframe có cùng một cột chung là id.

id = c(1,2,3,4,5)

age = c(21,34,45,32,18)

dat2 = data.frame(id,age)

Để hợp nhất 2 dataframe, chúng ta dùng hàm merge như sau:

dat3 = merge(dat1, dat2, by="id")

dat4 = merge(dat1, dat2, by="id", all.x=T, all.y=T)

Kết quả của dat3. Chú ý dataframe này chỉ có 4 dòng, vì dòng 5 của dat2 không có id trong dat1:

> dat3

id sex age

1 1 M 21

2 2 F 34

3 3 M 45

4 4 F 32

Kết quả của dat4 sẽ có 5 dòng, nhưng biến sex có 1 giá trị trống (NA):

> dat4

id sex age

1 1 M 21

2 2 F 34

3 3 M 45

4 4 F 32

5 5 <NA> 18

**3.4.3 Hoán chuyển dataframe**

**melt: Hoán chuyển từ cột sang dòng**

Giả dụ chúng ta có dữ liệu dat của 4 đối tượng nghiên cứu, và mỗi đối tượng được đo lường 3 lần (day1, day2, day3) như sau:

> dat

id sex group day1 day2 day3

1 1 M 1 15 17 19

2 2 F 1 16 15 20

3 3 M 2 21 23 19

4 4 F 2 31 35 33

Nhưng để phân tích phương sai, chúng ta cần hoán chuyển sang cột như sau:

id sex group day value

1 M 1 day1 15

1 M 1 day2 16

1 M 1 day3 21

2 F 1 day1 16

2 F 1 day2 15

2 F 1 day3 20

Để đạt mục đích trên, chúng ta dùng hàm melt trong package reshape2 như sau:

require(reshape2)

d = melt(dat, id=c("id", "sex", "group"), measure.vars=c("day1", "day2", "day3"))

**cast: Hoán chuyển từ dòng sang cột**

Thỉnh thoảng, chúng ta cần chuyển dataframe từ dòng sang cột. Chẳng hạn như dữ liệu d trên đang ở dạng:

> d

id sex group variable value

1 1 M 1 day1 15

2 2 F 1 day1 16

3 3 M 2 day1 21

4 4 F 2 day1 31

5 1 M 1 day2 17

6 2 F 1 day2 15

7 3 M 2 day2 23

8 4 F 2 day2 35

9 1 M 1 day3 19

10 2 F 1 day3 20

11 3 M 2 day3 19

12 4 F 2 day3 33

Để chuyển sang 3 cột cho 3 ngày, chúng ta dùng hàm cast trong package reshape như sau:

library(reshape)

d2 = cast(d, id+sex+group ~ variable)

**3.5 Một số hàm có ích**

Trong việc sắp xếp dữ liệu và mô phỏng, R có một số hàm giúp chúng ta làm việc có năng suất cao. Những hàm này tự động hoá một số chức năng, và có thể ứng dụng trong nhiều tình huống, đặc biệt là mô phỏng.

**seq()** là hàm dùng để tạo ra một dãy số liệu. Ví dụ: x = seq(from=1, to=10, by=0.5) sẽ tạo ra một biến số *x* có giá trị từ 1 đến 10, với khoảng cách là 0.5. Nói cách khác, x là một vector: 1.0, 1.5, 2.0, 2.5, ..., 10.0.

**rep()** là hàm lặp lại một đối tượng nhiều lần. Chẳng hạn như chúng ta muốn tạo ra một biến số group với 10 giá trị "A", chúng ta có thể dùng rep như sau:

group = rep("A", 10)

Hàm dưới đây sẽ tạo ra biến số group với 10 giá trị "A" và 15 giá trị "B":

group = c(rep("A", 10), rep("B", 15))

Hàm dưới đây sẽ tạo ra biến số val với 10 giá trị 1,2 lặp lại 5 lần:

val = rep(c(1,2), 5)

> val

[1] 1 2 1 2 1 2 1 2 1 2

**cut()** là hàm dùng để chia một biến liên tục thành nhiều nhóm. Chẳng hạn như chúng ta có một biến liên tục age như sau:

x = c(17, 19, 22, 43, 14, 8, 12, 19, 20, 51, 8, 12, 27, 31)

và chúng ta muốn chia thành 3 nhóm và đặt kết quả trong biến y, hàm dưới đây sẽ đáp ứng nhu cầu đó:

y = cut(x, 3, labels=c("Low", "Medium", "High"))

cbind(x, y)

> cbind(x, y)

x y

[1,] 17 1

[2,] 19 1

[3,] 22 1

[4,] 43 3

[5,] 14 1

[6,] 8 1

[7,] 12 1

[8,] 19 1

[9,] 20 1

[10,] 51 3

[11,] 8 1

[12,] 12 1

[13,] 27 2

[14,] 31 2

Hàm cut() chia nhóm một cách "tự động", tức dựa vào bách phân vị. Một cách khác là dùng hàm cut2() trong Hmisc để chia nhóm theo một giá trị. Chẳng hạn như nếu chúng ta muốn chia biến x thành 3 nhóm: dưới 20, từ 20 đến 30, và trên 30:

library(Hmisc)

y1 = cut2(x, c(20, 30))

cbind(x, y1)

> cbind(x, y1)

x y1

[1,] 17 1

[2,] 19 1

[3,] 22 2

[4,] 43 3

[5,] 14 1

[6,] 8 1

[7,] 12 1

[8,] 19 1

[9,] 20 2

[10,] 51 3

[11,] 8 1

[12,] 12 1

[13,] 27 2

[14,] 31 3

**3.6 Biên tập dữ liệu với package "dplyr"**

Trong biên tập dữ liệu (data editing), chúng ta thường thực hiện một số thao tác như chọn dòng, chọn cột, thêm cột, sắp xếp thứ tự theo một hay nhiều biến số, tính toán đơn giản, lấy mẫu, v.v. Tất cả những thao tác này được triển khai trong package dplyr qua 4 hàm chính như sau: filter, arrange, select, mutate, và summrise.

Để hiểu tác dụng của các hàm này, chúng ta bắt đầu bằng dữ liệu birthwt như sau:

bw = read.csv("birthwt.csv")

head(bw)

id low age lwt race smoke ptl ht ui ftv bwt

1 85 0 19 182 2 0 0 0 1 0 2523

2 86 0 33 155 3 0 0 0 0 3 2551

3 87 0 20 105 1 1 0 0 0 1 2557

4 88 0 21 108 1 1 0 0 1 2 2594

5 89 0 18 107 1 1 0 0 1 0 2600

6 91 0 21 124 3 0 0 0 0 0 2622

**Hàm filter()** cho phép chúng ta chọn một phần dữ liệu trong một data.frame nhưng theo dòng. Chẳng hạn như trong dữ liệu trên, chúng ta muốn chọn những dữ liệu với điều kiện race là 1 và bwt dưới 2500, lệnh sau đây sẽ đáp ứng yêu cầu đó:

library(dplyr)

d1 = filter(bw, race==1, bwt<2500)

d1

Kết quả là:

id low age lwt race smoke ptl ht ui ftv bwt

1 10 1 29 130 1 0 0 0 1 2 1021

2 20 1 21 165 1 1 0 1 0 1 1790

3 22 1 32 105 1 1 0 0 0 0 1818

4 23 1 19 91 1 1 2 0 1 0 1885

5 26 1 25 92 1 1 0 0 0 0 1928

6 27 1 20 150 1 1 0 0 0 2 1928

7 29 1 24 155 1 1 1 0 0 0 1936

8 33 1 19 102 1 0 0 0 0 2 2082

9 34 1 19 112 1 1 0 0 1 0 2084

10 35 1 26 117 1 1 1 0 0 0 2084

11 36 1 24 138 1 0 0 0 0 0 2100

12 42 1 22 130 1 1 1 0 1 1 2187

13 45 1 17 110 1 1 0 0 0 0 2225

14 51 1 20 121 1 1 1 0 1 0 2296

15 56 1 31 102 1 1 1 0 0 1 2353

16 57 1 15 110 1 0 0 0 0 0 2353

17 65 1 30 142 1 1 1 0 0 0 2410

18 67 1 22 130 1 1 0 0 0 1 2410

19 68 1 17 120 1 1 0 0 0 3 2414

20 69 1 23 110 1 1 1 0 0 0 2424

21 77 1 26 190 1 1 0 0 0 0 2466

22 79 1 28 95 1 1 0 0 0 2 2466

23 84 1 21 130 1 1 0 1 0 3 2495

**Hàm select()** có thể dùng để chọn một phần dữ liệu trong một data.frame nhưng theo biến số. Chẳng hạn như trong dữ liệu trên, chúng ta muốn chọn ra 3 biến id, bwt và age:

d2 = select(bw, id, age, bwt)

head(d2)

id age bwt

1 85 19 2523

2 86 33 2551

3 87 20 2557

4 88 21 2594

5 89 18 2600

6 91 21 2622

Cũng có thể chọn nhiều biến bằng cách dùng ":" như ví dụ dưới đây chọn các biến số bắt đầu từ race đến biến bwt:

d2 = select(bw, race:bwt)

**Hàm mutate()** có thể dùng để tạo biến số mới dựa trên những biến số đang có. Chẳng hạn như biến lwt (cân nặng của mẹ được tính bằng pound) và chúng ta muốn tạo biến số bằng kilogram và gọi biến đó là mother.wt, và biến cân nặng của trẻ bằng kilogram (thay vì gram) và gọi tên biến mới là weight:

d3 = mutate(bw, mother.wt = lwt\*0.453592, weight = bwt/1000)

head(d3)

id low age lwt race smoke ptl ht ui ftv bwt mother.wt weight

1 85 0 19 182 2 0 0 0 1 0 2523 82.55374 2.523

2 86 0 33 155 3 0 0 0 0 3 2551 70.30676 2.551

3 87 0 20 105 1 1 0 0 0 1 2557 47.62716 2.557

4 88 0 21 108 1 1 0 0 1 2 2594 48.98794 2.594

5 89 0 18 107 1 1 0 0 1 0 2600 48.53434 2.600

6 91 0 21 124 3 0 0 0 0 0 2622 56.24541 2.622

**Hàm arrange()** có thể dùng để sắp xếp (sort) dữ liệu theo thứ tự của một hay nhiều biến số. Chẳng hạn như hàm dưới đây sẽ sắp xếp dữ liệu theo thứ tự của biến mother.wt và weight:

d4 = arrange(d3, mother.wt, weight)

head(d4)

id low age lwt race smoke ptl ht ui ftv bwt mother.wt weight

1 44 1 20 80 3 1 0 0 1 0 2211 36.28736 2.211

2 15 1 25 85 3 0 0 0 1 0 1474 38.55532 1.474

3 137 0 22 85 3 1 0 0 0 0 3090 38.55532 3.090

4 32 1 25 89 3 0 2 0 0 1 2055 40.36969 2.055

5 118 0 24 90 1 1 1 0 0 1 2948 40.82328 2.948

6 132 0 18 90 1 1 0 0 1 0 3062 40.82328 3.062

**Hàm sample\_n()** và **sample\_frac()**có thể dùng để chọn mẫu ngẫu nhiên theo dòng hoặc theo xác suất định trước. Lệnh dưới đây chọn 10 dòng ngẫu nhiên từ dữ liệu bw:

d5 = sample\_n(bw, 10)

d5

id low age lwt race smoke ptl ht ui ftv bwt

57 144 0 21 110 3 1 0 0 1 0 3203

159 43 1 27 130 2 0 0 0 1 0 2187

124 220 0 22 129 1 0 0 0 0 0 4111

93 187 0 19 235 1 1 0 1 0 0 3629

48 135 0 19 132 3 0 0 0 0 0 3090

73 164 0 23 115 3 1 0 0 0 1 3331

17 102 0 15 98 2 0 0 0 0 0 2778

154 35 1 26 117 1 1 1 0 0 0 2084

30 116 0 17 113 2 0 0 0 0 1 2920

79 172 0 20 121 2 1 0 0 0 0 3444

Lệnh dưới đây chọn ngẫu nhiên 5% từ dữ liệu bw:

d6 = sample\_frac(bw, 0.05)

d6

id low age lwt race smoke ptl ht ui ftv bwt

127 223 0 35 170 1 0 1 0 0 1 4174

125 221 0 25 130 1 0 0 0 0 2 4153

141 22 1 32 105 1 1 0 0 0 0 1818

146 27 1 20 150 1 1 0 0 0 2 1928

31 117 0 17 113 2 0 0 0 0 1 2920

48 135 0 19 132 3 0 0 0 0 0 3090

128 224 0 19 120 1 1 0 0 0 0 4238

116 212 0 28 134 3 0 0 0 0 1 3941

97 191 0 29 154 1 0 0 0 0 1 3651

**Hàm summarise()** và **group\_by()** rất có ích trong việc tính toán theo nhóm. Chẳng hạn như chúng ta muốn tính giá trị trung bình của biến age, lwt và bwt theo smoke, và tạo ra những biến mới có giá trị trung bình với tên là mean.age mean.lwt, và mean.bw:

**group** = group\_by(bw, smoke)

d7 = summarise(**group**,

count = n(),

mean.age = mean(age, na.rm=T),

mean.lwt = mean(lwt, na.rm=T),

mean.bw = mean(bwt, na.rm=T))

> d7

# A tibble: 2 x 5

smoke count mean.age mean.lwt mean.bw

<int> <int> <dbl> <dbl> <dbl>

1 0 115 23.42609 130.8957 3055.696

2 1 74 22.94595 128.1351 2771.919

Tất cả các hàm trong dplyr có thể thực hiện trong R (mà không cần dplyr), nhưng các hàm vừa mô tả trên thì dễ hiểu hơn và dễ sử dụng hơn. Dĩ nhiên, chúng ta cũng có thể kết hợp các hàm trong dplyr và các hàm cơ bản trong R để xử lí những dữ liệu lớn và phức tạp.

**3.7 Biến phụ thuộc và 1 biến tiên lượng**

Phần lớn các mô hình thống kê phân biệt hai loại biến số: biến phụ thuộc (dependent variables) và biến độc lập (independent variable), có khi còn gọi là biến *tiên lượng* (predictor variable). Biến phụ thuộc có khi còn gọi là *outcome*. Chẳng hạn như nếu chúng ta muốn so sánh độ tuổi giữa nam và nữ, thì độ tuổi là biến phụ thuộc, và giới tính là biến tiên lượng.

Cú pháp chuẩn của R trong việc mô tả mối liên quan giữa biến phụ thuộc (*y*) và biến tiên lượng (*x*) là:

*y* ~ *x*

trong đó, kí hiệu ~ (tilde) có thể đọc "hàm số của". Do đó, *y* ~ *x* có thể đọc là "*y* là hàm số của *x*", hay "*y* phụ thuộc vào *x.*" Điều đó cũng có nghĩa thực tế trong việc tổ chức dữ liệu cho phân tích. Bởi vì cú pháp trên phân biệt biệt *y* và *x* là hai biến, nên dữ liệu cũng phải sắp xếp thành hai biến riêng biệt.

Chẳng hạn như chúng ta có dữ liệu *y* cho 2 nhóm A và B như sau:

|  |  |
| --- | --- |
| **Nhóm A** | **Nhóm B** |
| 13.2 | 21.3 |
| 11.7 | 19.3 |
| 15.8 | 17.8 |
|  | 15.4 |

thì cách sắp xếp dữ liệu trong R phải theo hai biến. Biến thứ nhất là nhóm (tạm gọi là *group*) và biến thứ hai là *y*:

|  |  |
| --- | --- |
| *group* | *y* |
| A | 13.2 |
| A | 11.7 |
| A | 15.8 |
| B | 21.3 |
| B | 19.3 |
| B | 17.8 |
| B | 15.4 |

và khi nhập dữ liệu trong R:

group = c("A", "A", "A", "B", "B", "B", "B")

y = c(13.2, 11.7, 15.8, 21.3, 19.3, 17.8, 15.4)

dat = data.frame(group, y)

hoặc có thể làm gọn hơn bằng cách dùng hàm rep:

group = c(rep("A", 3), rep("B", 4))

y = c(13.2, 11.7, 15.8, 21.3, 19.3, 17.8, 15.4)

dat = data.frame(group, y)

Do đó, khi mô tả dữ liệu theo nhóm, ví dụ như biểu đồ hộp, chúng ta chỉ cần viết ngắn gọn:

boxplot(y ~ group)

hay kiểm định t:

t.test(y ~ group)

**Biến phụ thuộc và nhiều biến tiên lượng**

Cách tổ chức dữ liệu trên cũng được áp dụng cho nhiều biến tiên lượng. Chẳng hạn như một nghiên cứu theo dõi hai nhóm bệnh nhân, và mỗi bệnh nhân được đo lường ở 3 thời điểm như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Group** | **Time 1** | **Time 2** | **Time 3** |
| 1 | A | 1.10 | 1.12 | 1.11 |
| 2 | A | 0.80 | 0.85 | 0.83 |
| 3 | A | 0.76 | 0.75 | 0.77 |
| 4 | B | 0.90 | 0.92 | 0.95 |
| 5 | B | 0.95 | 0.97 | 1.00 |
| 6 | B | 1.02 | 1.07 | 1.10 |
| 7 | B | 0.89 | 0.87 | 0.92 |

Cách tổ chức dữ liệu như trên rất khó dùng cho phân tích chuyên sâu nhằm đánh giá thay đổi của bệnh nhân theo thời gian (ví dụ mô hình phân tích hỗn hợp). Bởi vì chúng ta muốn biết mức độ dao động giữa các thời điểm trong mỗi bệnh nhân, nên thời gian cần phải có một biến riêng. Cách tổ chức lí tưởng cho R là:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Group** | **Time** | **y** |
| 1 | A | 1 | 1.10 |
| 1 | A | 2 | 1.12 |
| 1 | A | 3 | 1.11 |
| 2 | A | 1 | 0.80 |
| 2 | A | 2 | 0.85 |
| 2 | A | 3 | 0.83 |
| 3 | A | 1 | 0.76 |
| 3 | A | 2 | 0.75 |
| 3 | A | 3 | 0.77 |
| 4 | B | 1 | 0.90 |
| 4 | B | 2 | 0.92 |
| 4 | B | 3 | 0.95 |
| 5 | B | 1 | 0.95 |
| 5 | B | 2 | 0.97 |
| 5 | B | 3 | 1.00 |
| 6 | B | 1 | 1.02 |
| 6 | B | 2 | 1.07 |
| 6 | B | 3 | 1.10 |
| 7 | B | 1 | 0.89 |
| 7 | B | 2 | 0.87 |
| 7 | B | 3 | 0.92 |

Cần chú ý rằng sắp xếp hay "tổ chức" dữ liệu là một bước rất quan trọng trong phân tích thống kê. Có thể nói rằng trong nhiều trường hợp, thời gian để sắp xếp dữ liệu chiếm đến 90%, còn thời gian phân tích chỉ chiếm khoảng 10% quĩ thời gian. Một khi bước sắp xếp dữ liệu hoàn chỉnh thì việc phân tích sẽ rất thuận tiện.