# 第 17 天

## 實用 R 語言技巧彙整

在第 17 天與第 18 天的內容中我們彙整了一些解決常見問題的 R 語言技巧，提供給讀者參考，希望在您有需要的時候，這些函數或語法能夠幫上忙。

### 問題

我們希望隨機產出的結果可以固定。

#### 實用技巧

運用 set.seed() 函數，在同一個隨機種子的設定之下，會產出相同的隨機輸出，舉例來說我們希望利用 rnorm() 函數產出 5 個符合標準常態分佈的隨機數，在沒有設定隨機種子的情形之下，每次呼叫函數都會產出不同的 5 個隨機數。

> rnorm(5)  
[1] 1.2805549 -1.7272706 1.6901844 0.5038124 2.5283366  
> rnorm(5)  
[1] 0.5490967 0.2382129 -1.0488931 1.2947633 0.8255398  
> rnorm(5)  
[1] -0.05568601 -0.78438222 -0.73350322 -0.21586539 -0.33491276

如果我們指定一個隨機種子：123，我們可以確保每次呼叫函數產出的 5 個隨機數是相同的。

> set.seed(123)  
> rnorm(5)  
[1] -0.56047565 -0.23017749 1.55870831 0.07050839 0.12928774  
> set.seed(123)  
> rnorm(5)  
[1] -0.56047565 -0.23017749 1.55870831 0.07050839 0.12928774  
> set.seed(123)  
> rnorm(5)  
[1] -0.56047565 -0.23017749 1.55870831 0.07050839 0.12928774

### 問題

我們想要加總矩陣（matrix）或資料框（data.frame）的列資料或欄資料。

#### 實用技巧

運用 rowSums() 函數可以加總列資料，而 colSums() 函數可以加總欄資料。讓我們使用一個杜撰的資料 ice\_cream 來示範。

> set.seed(123)  
> ice\_cream <- matrix(round(runif(15) \* 100), nrow = 5)  
> colnames(ice\_cream) <- c("Vanilla", "Chocolate", "Strawberry")  
> rownames(ice\_cream) <- c("Mon", "Tue", "Wed", "Thu", "Fri")  
> ice\_cream  
 Vanilla Chocolate Strawberry  
Mon 29 5 96  
Tue 79 53 45  
Wed 41 89 68  
Thu 88 55 57  
Fri 94 46 10

我們可以利用 rowSums() 計算每天總共賣出多少冰淇淋，也可以利用 colSums() 計算每種口味的總銷量為何。

> set.seed(123)  
> ice\_cream <- matrix(round(runif(15) \* 100), nrow = 5)  
> colnames(ice\_cream) <- c("Vanilla", "Chocolate", "Strawberry")  
> rownames(ice\_cream) <- c("Mon", "Tue", "Wed", "Thu", "Fri")  
> rowSums(ice\_cream)  
Mon Tue Wed Thu Fri   
130 177 198 200 150   
> colSums(ice\_cream)  
 Vanilla Chocolate Strawberry   
 331 248 276

我們亦可以利用 cbind() 將列加總的結果加入原本的資料中。

> set.seed(123)  
> ice\_cream <- matrix(round(runif(15) \* 100), nrow = 5)  
> colnames(ice\_cream) <- c("Vanilla", "Chocolate", "Strawberry")  
> rownames(ice\_cream) <- c("Mon", "Tue", "Wed", "Thu", "Fri")  
> ice\_cream <- cbind(ice\_cream, Totals = rowSums(ice\_cream))  
> ice\_cream  
 Vanilla Chocolate Strawberry Totals  
Mon 29 5 96 130  
Tue 79 53 45 177  
Wed 41 89 68 198  
Thu 88 55 57 200  
Fri 94 46 10 150

接著利用 rbind() 將欄加總的結果也加入。

> set.seed(123)  
> ice\_cream <- matrix(round(runif(15) \* 100), nrow = 5)  
> colnames(ice\_cream) <- c("Vanilla", "Chocolate", "Strawberry")  
> rownames(ice\_cream) <- c("Mon", "Tue", "Wed", "Thu", "Fri")  
> ice\_cream <- cbind(ice\_cream, Totals = rowSums(ice\_cream))  
> ice\_cream <- rbind(ice\_cream, Totals = colSums(ice\_cream))  
> ice\_cream  
 Vanilla Chocolate Strawberry Totals  
Mon 29 5 96 130  
Tue 79 53 45 177  
Wed 41 89 68 198  
Thu 88 55 57 200  
Fri 94 46 10 150  
Totals 331 248 276 855

### 問題

我們知道某個特定值存在於一個向量中，想要尋找它的索引值為何。

#### 實用技巧

我們可以善用 match()、which()、which.min() 與 which.max() 這幾個函數請 R 語言回傳特定值的索引值。

match() 函數會回傳第一個吻合特定值的索引值。

> my\_vector <- 11:20  
> match(17, my\_vector)  
[1] 7

如果在向量中會有多個吻合特定值，而我們又希望可以全部取出來他們的索引值，這時我們可以使用邏輯值向量搭配 which() 函數。which() 函數可以幫助我們將 TRUE 的索引值取出來，舉例來說我們在前一個例子的向量後面多加上一個 17。

> my\_vector <- c(11:20, 17)  
> match(17, my\_vector)  
[1] 7

我們發現用 match() 函數的結果是相同的，因為它找到第一個 17 就停止了，於是改用邏輯值向量搭配 which() 函數。

> my\_vector <- c(11:20, 17)  
> which(my\_vector == 17)  
[1] 7 11

同樣的運用我們也可以來尋找 NA 的索引值，搭配 is.na() 函數進行。

> my\_vector <- c(11:20, NA, 24, 18, NA)  
> which(is.na(my\_vector))  
[1] 11 14

which.min() 與 which.max() 函數則能分別回傳最小值與最大值的索引值。

> my\_vector <- 11:20  
> which.min(my\_vector)  
[1] 1  
> which.max(my\_vector)  
[1] 10

如果在這個向量中，最小值或者最大值有超過一個，which.min() 與 which.max() 只會回傳第一個找到的最小值或最大值。

> my\_vector <- c(11:20, 11, 20)  
> which.min(my\_vector)  
[1] 1  
> which.max(my\_vector)  
[1] 10

假如我們希望的是把所有最小值或者最大值的索引值回傳，同樣可以利用邏輯值向量搭配 which() 函數。

> my\_vector <- c(11:20, 11, 20)  
> min\_logical <- my\_vector == min(my\_vector)  
> max\_logical <- my\_vector == max(my\_vector)  
> which(min\_logical)  
[1] 1 11  
> which(max\_logical)  
[1] 10 12

### 問題

我們希望排序向量與資料框。

#### 實用技巧

對於向量而言，我們使用 sort() 函數。

> my\_vector <- round(runif(10) \* 100)  
> my\_vector # 未排序  
 [1] 14 41 41 37 15 14 23 47 27 86  
> sort(my\_vector) # 預設遞增排序  
 [1] 14 14 15 23 27 37 41 41 47 86  
> sort(my\_vector, decreasing = TRUE) # 遞減排序  
 [1] 86 47 41 41 37 27 23 15 14 14

如果想針對資料框中的某一個欄位排序，不能夠使用 sort() 函數，要使用 order() 函數。我們使用內建資料 cars 來示範，cars 原本是依照 cars$speed 遞增排序：

> cars$speed  
 [1] 4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14  
[23] 14 15 15 15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22  
[45] 23 24 24 24 24 25

我們希望可以改爲以 cars$dist 遞增排序，首先看一下 order() 函數的輸出：

> order(cars$dist)  
 [1] 1 3 2 6 12 5 10 7 13 24 4 14 8 16 20 25 11 15 27 29 39 9  
[23] 17 18 21 36 28 30 32 19 37 40 31 41 26 45 33 42 22 43 44 38 46 34  
[45] 23 35 50 47 48 49

我們發現 order() 函數不是回傳排序後的資料，而是回傳排序後的列數索引值，因此我們要利用這些列數索引值來對 cars 排序：

> order\_by\_dist <- order(cars$dist)  
> reordered\_cars <- cars[order\_by\_dist, ]  
> reordered\_cars$dist  
 [1] 2 4 10 10 14 16 17 18 20 20 22 24 26 26 26 26  
[17] 28 28 32 32 32 34 34 34 36 36 40 40 42 46 46 48  
[33] 50 52 54 54 56 56 60 64 66 68 70 76 80 84 85 92  
[49] 93 120

### 問題

我們希望從向量中隨機抽樣。

#### 實用技巧

我們可以運用 sample() 函數協助我們從一個向量中抽樣，例如從 1 到 10 這 10 個數字中抽樣 5 個數字，我們可以將 1:10 作為 sample() 函數的輸入，並且指派參數 size = 5：

> sample(1:10, size = 5)  
[1] 1 4 7 10 9  
> sample(1:10, size = 5)  
[1] 10 5 6 9 1

如果我們希望抽樣的結果是一致的，就在抽樣前設定相同的隨機種子。

> set.seed(123)  
> sample(1:10, size = 5)  
[1] 3 8 4 7 6  
> set.seed(123)  
> sample(1:10, size = 5)  
[1] 3 8 4 7 6  
> set.seed(123)  
> sample(1:10, size = 5)  
[1] 3 8 4 7 6

sample() 函數預設不重複抽樣，如果我們希望可以重複抽樣，就在 sample() 函數中修改參數 replace = TRUE。由於每一次抽樣後都會將數字放回去，因此才會在第一次與第五次都同樣抽到了 3。

> set.seed(1)  
> sample(1:10, size = 5, replace = TRUE)  
[1] 3 4 6 10 3

我們可以用 sample() 函數模擬硬幣投擲的結果：

> coin\_sides <- c("Head", "Tail")  
> sample(coin\_sides, size = 1)  
[1] "Tail"

我們可以把投擲的次數也加進去，並且用一個向量儲存每一次投擲的結果。

# 自訂函數  
coin\_flips <- function(n){  
 coin\_sides <- c("Head", "Tail")  
 flip\_results <- rep(NA, times = n)  
 for (i in 1:n){  
 flip\_results[i] <- sample(coin\_sides, size = 1)  
 }  
 return(flip\_results)  
}

> coin\_flips(n = 5) # 投擲五次硬幣的結果  
[1] "Tail" "Head" "Head" "Head" "Head"  
> coin\_flips(n = 10) # 投擲十次硬幣的結果  
 [1] "Head" "Tail" "Head" "Head" "Tail" "Head" "Head" "Tail" "Tail"  
[10] "Tail"

同樣的設計，我們也可以設計一個投擲骰子的函數 dice\_rolls。

# 自訂函數  
dice\_rolls <- function(n){  
 dice\_sides <- 1:6  
 roll\_results <- rep(NA, times = n)  
 for (i in 1:n){  
 roll\_results[i] <- sample(dice\_sides, size = 1)  
 }  
 return(roll\_results)  
}

> dice\_rolls(n = 5) # 投擲五次骰子的結果  
[1] 1 5 3 5 4  
> dice\_rolls(n = 10) # 投擲十次骰子的結果  
 [1] 5 4 4 5 1 3 5 5 3 6

類似的設計，我們亦可以設計一個發撲克牌的函數 poker\_cards，撲克牌有十三個大小與四個花色。

poker\_cards <- function(n){  
 color <- c("spade", "heart", "diamond", "club")  
 number <- c("Ace", 2:10, "Jack", "Queen", "King")  
 colors <- rep(color, times = 13)  
 numbers <- rep(number, times = 4)  
 cards <- paste(numbers, colors)  
 return(sample(cards, size = n))  
}

> poker\_cards(n = 4) # 發 4 張牌  
[1] "Ace club" "5 spade" "5 club" "4 spade"   
> poker\_cards(n = 12) # 發 12 張牌  
 [1] "5 club" "5 heart" "4 spade" "Jack club"   
 [5] "4 diamond" "2 spade" "Queen diamond" "9 diamond"   
 [9] "9 club" "6 diamond" "4 heart" "Jack heart"