# 第 09 天

## 迴圈與流程控制

撰寫程式（任何程式，不限於 R 語言）相當重要的一個環節是迴圈與流程控制，對於從未接觸過程式的初學者可能會對於這個章節的標題感到陌生與擔心，其實完全不必，應當要將迴圈與流程控制視為幫助我們寫出更簡潔程式的利器！

### 迴圈

假設我們現在想做一件事情：將 R 語言中的一個內建向量 month.name 分別輸出在命令列。土法煉鋼的方法是像這樣，複製貼上程式碼，然後修改中括號 [] 裡面的索引值：

> month.name # 1-12 月的月份名稱  
 [1] "January" "February" "March" "April" "May"   
 [6] "June" "July" "August" "September" "October"   
[11] "November" "December"   
> month.name[1]  
[1] "January"  
> month.name[2]  
[1] "February"  
> month.name[3]  
[1] "March"  
> month.name[4]  
[1] "April"  
> month.name[5]  
[1] "May"  
> month.name[6]  
[1] "June"  
> month.name[7]  
[1] "July"  
> month.name[8]  
[1] "August"  
> month.name[9]  
[1] "September"  
> month.name[10]  
[1] "October"  
> month.name[11]  
[1] "November"  
> month.name[12]  
[1] "December"

當碰到這樣子需要大量手工複製貼上的事情時，就可以求助**迴圈**來幫助我們！

#### for 迴圈

一個 for 迴圈的外觀架構長得像這個樣子：

for (i in x){  
 # 每次迭代要執行的程式  
}

在第一次的迭代（iteration）中，i 是 x[1]；第二次的迭代中，i 是 x[2]；以此類推，然後在每一次迭代時，都會執行大括號 {} 裡面的程式。

我們開始撰寫多行且有縮排的程式，這時候使用左上角的來源（Source）區塊編寫就比在命令列（Console）撰寫來得更適合。我們首先新增一個 R 程式：

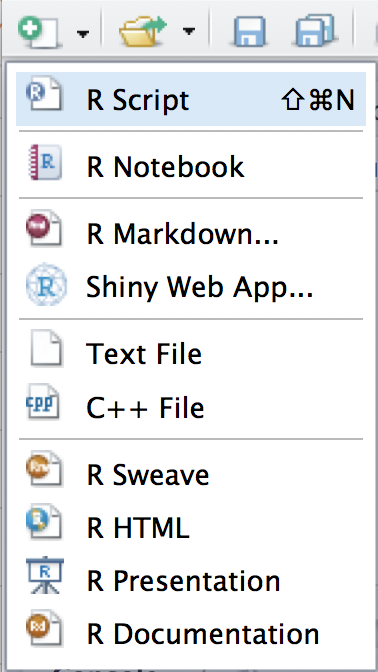


圖 9-1 新增 R 程式

如果我們想將內建向量 month.name 分別輸出，寫作成 for 迴圈的話，可以寫成這樣：

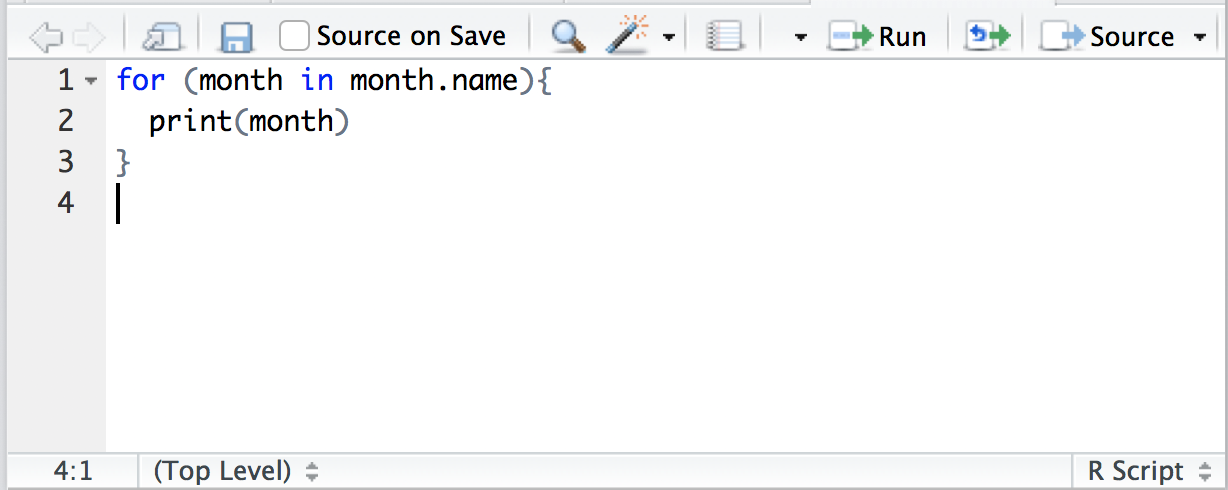


圖 9-2 編寫 for 迴圈

最後將這些程式選擇起來後點選 **Run**。

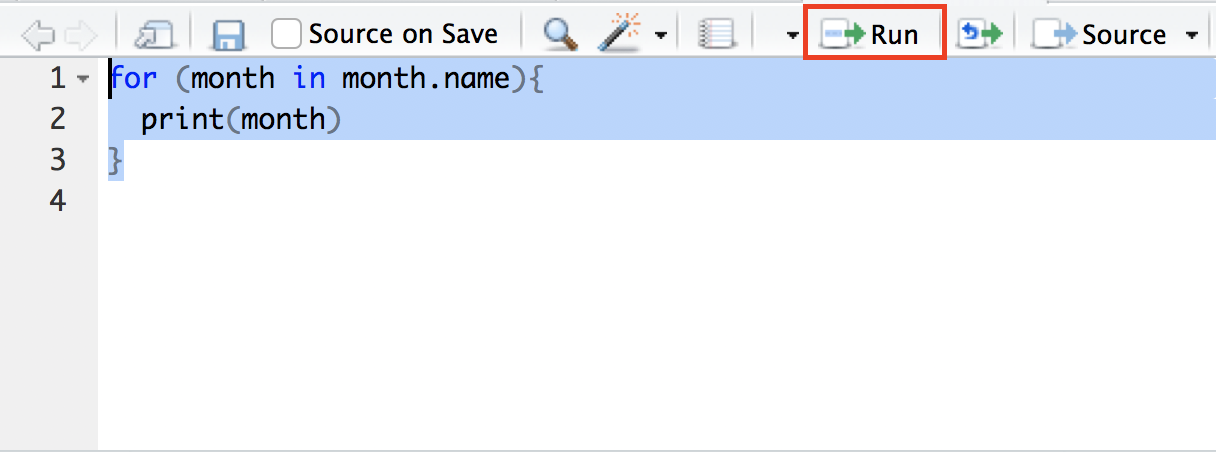


圖 9-3 執行 for 迴圈

這些程式就會在命令列被執行：

> for (month in month.name){  
+ print(month)  
+ }  
[1] "January"  
[1] "February"  
[1] "March"  
[1] "April"  
[1] "May"  
[1] "June"  
[1] "July"  
[1] "August"  
[1] "September"  
[1] "October"  
[1] "November"  
[1] "December"

#### while 迴圈

一個 while 迴圈的外觀架構長得像這樣子：

while (某種條件){  
 # 每次迭代要執行的程式  
}

在每一次的迭代之前，R 語言都會去檢查小括號 () 中的某種條件是否成立，判斷結果為邏輯值 TRUE 就會執行每次迭代要執行的程式；一但條件不成立，判斷結果為邏輯值 FALSE 就會離開迴圈。因此如果我們想將內建向量 month.name 分別輸出，寫作成 while 迴圈的話，可以寫成這樣：

> i <- 1  
> while (i < 13){  
+ print(month.name[i])  
+ i <- i + 1  
+ }  
[1] "January"  
[1] "February"  
[1] "March"  
[1] "April"  
[1] "May"  
[1] "June"  
[1] "July"  
[1] "August"  
[1] "September"  
[1] "October"  
[1] "November"  
[1] "December"

很重要的一行是 i <- i + 1，假如忘記寫這一行程式，我們的 while 迴圈會無限次數地一直輸出 “January”，原因是小括號 () 中條件（1 < 13）永遠都成立，所以不斷地執行 print(month.name[1])。

#### 兩種迴圈的運用時機

那麼在實際撰寫 R 語言程式的時候，我們何時應該運用 for 迴圈、何時應該運用 while 迴圈呢？一個簡單的判斷是：假如我們明確知道程式需要執行幾次（迭代次數），比如說我們知道 month.name 這個向量的長度為 12，就可以採用 for 迴圈或 while 迴圈，端看個人偏好；而在不知道迭代次數的情形下，我們就只能採用 while 迴圈了。

這個說法還是略嫌抽象，舉一個例子說明會比較好理解。假如我們現在投擲一枚公正的銅板，想知道總共需要投擲幾次才會出現三個正面，這就是一個不知道迭代次數的問題。運氣好一點也許投擲三次，三次都是正面；運氣差一點的也許要投擲十幾次才湊得齊三次正面。我們寫了一段簡單的程式來模擬這個過程，其中 sample() 就是從向量中隨機抽樣的函數，size 參數可以指定隨機抽樣的個數：

flip\_results <- c() # 建立一個空的向量來放置每一次投擲的結果  
coin <- c(TRUE, FALSE) # TRUE 代表正面，FALSE 代表反面  
n\_flips <- 1 # 從第一次投擲開始記錄  
  
while (sum(flip\_results) < 3){  
 flip\_results[n\_flips] <- sample(coin, size = 1) # 將每次投擲結果記錄起來  
 n\_flips <- n\_flips + 1 # 準備記錄下一次的投擲結果  
}  
flip\_results # 印出每次投擲的紀錄  
length(flip\_results) # 總共投擲了幾次

我們執行一輪試試看：

> flip\_results <- c() # 建立一個空的向量來放置每一次投擲的結果  
> coin <- c(TRUE, FALSE) # TRUE 代表正面，FALSE 代表反面  
> n\_flips <- 1 # 從第一次投擲開始記錄  
>   
> while (sum(flip\_results) < 3){  
+ flip\_results[n\_flips] <- sample(coin, size = 1) # 將每次投擲結果記錄起來  
+ n\_flips <- n\_flips + 1 # 準備記錄下一次的投擲結果  
+ }  
> flip\_results # 印出每次投擲的紀錄  
[1] TRUE TRUE TRUE  
> length(flip\_results) # 總共投擲了幾次  
[1] 3

哇，太幸運了！我們只投擲三次就投擲出了三次正面。再執行一輪試試看：

> flip\_results <- c() # 建立一個空的向量來放置每一次投擲的結果  
> coin <- c(TRUE, FALSE) # TRUE 代表正面，FALSE 代表反面  
> n\_flips <- 1 # 從第一次投擲開始記錄  
>   
> while (sum(flip\_results) < 3){  
+ flip\_results[n\_flips] <- sample(coin, size = 1) # 將每次投擲結果記錄起來  
+ n\_flips <- n\_flips + 1 # 準備記錄下一次的投擲結果  
+ }  
> flip\_results # 印出每次投擲的紀錄  
[1] TRUE FALSE FALSE TRUE FALSE TRUE  
> length(flip\_results) # 總共投擲了幾次  
[1] 6

這次雖然沒有上一輪那麼幸運，但也很不錯，只投擲了六次就投擲出了三次正面，每次投擲分別為正面、反面、反面、正面、反面與正面。您可以試著自行執行幾輪，看看每一輪投擲次數的差異。

### 流程控制

流程控制可以幫助我們在程式中撰寫不同的劇本（scenario），根據不同的判斷條件執行不同的程式。在 R 語言中我們使用 if、else if 與 else 建立出不同的分支，它的外觀架構長得像這樣子：

if (條件一) {  
  
 # 程式一  
  
} else if (條件二) {  
  
 # 程式二  
  
} else {  
  
 # 程式三  
  
}

如果條件一的判斷結果為 TRUE 就執行程式一，條件二的判斷結果為 TRUE 就執行程式二；假如條件一與條件二的判斷結果皆為 FALSE，就執行程式三。如果有更多的條件，只要增加 else if 的個數即可。

舉例來說，一個喜愛運動的人早上起床會看天氣決定當天的行程，在兩種天氣條件下，使用 if 跟 else 就可以建立出一個兩種劇本的行程：

weather <- sample(c("sunny", "rainy"), size = 1)  
weather  
if (weather == "sunny"){  
 print("Running outdoors!")  
} else {  
 print("Working out in the gym!")  
}

如果 weather 為 "sunny" 晴天，就在戶外跑步：

> weather <- sample(c("sunny", "rainy"), size = 1)  
> weather  
[1] "sunny"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else {  
+ print("Working out in the gym!")  
+ }  
[1] "Running outdoors!"

如果 weather 不是 "sunny" 晴天，就上健身房運動：

> weather <- sample(c("sunny", "rainy"), size = 1)  
> weather  
[1] "rainy"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else {  
+ print("Working out in the gym!")  
+ }  
[1] "Working out in the gym!"

在三種天氣條件下，使用 if、else if 與 else 可以建立出一個三種劇本的行程：

weather <- sample(c("sunny", "rainy", "cloudy"), size = 1)  
weather  
if (weather == "sunny"){  
 print("Running outdoors!")  
} else if (weather == "cloudy"){  
 print("Cycling!")  
} else {  
 print("Working out in the gym!")  
}

如果 weather 為 "sunny" 晴天，就在戶外跑步：

> weather <- sample(c("sunny", "rainy", "cloudy"), size = 1)  
> weather  
[1] "sunny"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else if (weather == "cloudy"){  
+ print("Cycling!")  
+ } else {  
+ print("Working out in the gym!")  
+ }  
[1] "Running outdoors!"

如果 weather 為 "cloudy" 陰天，就去騎單車：

> weather <- sample(c("sunny", "rainy", "cloudy"), size = 1)  
> weather  
[1] "cloudy"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else if (weather == "cloudy"){  
+ print("Cycling!")  
+ } else {  
+ print("Working out in the gym!")  
+ }  
[1] "Cycling!"

如果 weather 既不是 "sunny" 亦不是 "cloudy" 那就去健身房運動：

> weather <- sample(c("sunny", "rainy", "cloudy"), size = 1)  
> weather  
[1] "rainy"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else if (weather == "cloudy"){  
+ print("Cycling!")  
+ } else {  
+ print("Working out in the gym!")  
+ }  
[1] "Working out in the gym!"

在四種或更多個天氣條件下，依然使用 if、else if 與 else 建立出一個四種劇本的行程：

weather <- sample(c("sunny", "cloudy", "drizzle", "showers", "storm"), size = 1)  
weather  
if (weather == "sunny"){  
 print("Running outdoors!")  
} else if (weather == "cloudy"){  
 print("Cycling!")  
} else if (weather == "drizzle") {  
 print("Working out in the gym!")  
} else {  
 print("Couch potato.")  
}

如果 weather 為 "sunny" 晴天，就在戶外跑步：

> weather <- sample(c("sunny", "cloudy", "drizzle", "showers", "storm"), size = 1)  
> weather  
[1] "sunny"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else if (weather == "cloudy"){  
+ print("Cycling!")  
+ } else if (weather == "drizzle") {  
+ print("Working out in the gym!")  
+ } else {  
+ print("Couch potato.")  
+ }  
[1] "Running outdoors!"

如果 weather 為 "cloudy" 陰天，就去騎單車：

> weather <- sample(c("sunny", "cloudy", "drizzle", "showers", "storm"), size = 1)  
> weather  
[1] "cloudy"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else if (weather == "cloudy"){  
+ print("Cycling!")  
+ } else if (weather == "drizzle") {  
+ print("Working out in the gym!")  
+ } else {  
+ print("Couch potato.")  
+ }  
[1] "Cycling!"

如果 weather 為 "drizzle" 毛毛雨，就去健身房運動：

> weather <- sample(c("sunny", "cloudy", "drizzle", "showers", "storm"), size = 1)  
> weather  
[1] "drizzle"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else if (weather == "cloudy"){  
+ print("Cycling!")  
+ } else if (weather == "drizzle") {  
+ print("Working out in the gym!")  
+ } else {  
+ print("Couch potato.")  
+ }  
[1] "Working out in the gym!"

如果 weather 既不是 "sunny"、"cloudy" 亦不是 "drizzle" 就待在家裡看電視：

> weather <- sample(c("sunny", "cloudy", "drizzle", "showers", "storm"), size = 1)  
> weather  
[1] "storm"  
> if (weather == "sunny"){  
+ print("Running outdoors!")  
+ } else if (weather == "cloudy"){  
+ print("Cycling!")  
+ } else if (weather == "drizzle") {  
+ print("Working out in the gym!")  
+ } else {  
+ print("Couch potato.")  
+ }  
[1] "Couch potato."

### 結合迴圈與流程控制

加入流程控制讓使用迴圈時更有彈性。

#### break 敘述

在迴圈執行的過程中，滿足某種條件時候就離開迴圈，舉例來說，將內建向量 month.name 分別輸出時，假如碰到 "April" 就離開迴圈：

> for (month in month.name){  
+ if (month == "April"){  
+ break  
+ } else {  
+ print(month)  
+ }  
+ }  
[1] "January"  
[1] "February"  
[1] "March"

#### next 敘述

在迴圈執行的過程中，滿足某種條件時候略過那一次迭代，舉例來說，將內建向量 month.name 分別輸出時，假如碰到 "April" 就略過不輸出：

> for (month in month.name){  
+ if (month == "April"){  
+ next  
+ } else {  
+ print(month)  
+ }  
+ }  
[1] "January"  
[1] "February"  
[1] "March"  
[1] "May"  
[1] "June"  
[1] "July"  
[1] "August"  
[1] "September"  
[1] "October"  
[1] "November"  
[1] "December"

注意輸出 "March" 接著輸出的就是 "May"，而非 "April"。

### 小結

好啦！第九天的內容就是這麼多，我們練習了 R 語言的 for 迴圈以及 while 迴圈，還有懂得使用 if-else if-else 進行流程控制。

### 練習題

###### 1. 我們建立了一個 week 向量，裡面有一個星期中的七天名稱，請您使用 for 迴圈一一輸出每一天

week <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")

###### 2. 同樣的一個 week 向量，請您使用 while 迴圈一一輸出每一天

week <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")

###### 3. 同樣的一個 week 向量，請您在使用迴圈一一輸出每一天的時候略過週一到週五，只輸出我們最愛的週末兩天即可

week <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")