

# R 程式設計

*認識向量*

郭耀仁 [yaojenkuo@datainpoint.com](mailto:yaojenkuo@datainpoint.com) (<mailto:yaojenkuo@datainpoint.com>) 來自數據交點  
(<https://www.datainpoint.com/>)

*There are only two hard things in Computer Science: cache invalidation and naming things.*

*Phil Karlton*

# 數值運算

## 7 個基本數值運算符

- + 加號
- - 減號
- \* 乘號
- / 除號
- \*\* 或 ^ 次方
- %% 餘數
- %/% 商數

# 數值運算的先後順序

- 小括號 ( ) 優先
- $**$  或  $^$  次方其次
- $*$  乘號與  $/$  除號次之
- $+$  加號與  $-$  減號
- 運算順序若是相同，則由左至右依序運算

In [1]:  $0 \cdot 9/5 + 32$

32

In [2]:  $(32 - 32) \cdot 5/9$

0

# 賦値運算符

## 推薦使用 <- 作為賦值運算符

- = 也可以進行賦值，但是絕大部分的 R 使用者習慣 <-
- 在 RStudio 可以按 Alt + = 來獲得 <-

```
In [3]: lucky_number <- 5566  
lucky_number  
lucky_number = 5566  
lucky_number
```

5566

5566



# 使用 # 作為註解

單行註解或者行末註解

```
In [4]: # 5566 is my lucky number  
lucky_number <- 5566 # 5566 is my lucky number
```

# 該如何寫出與多數使用者相符的 R 程式？

參考 R 的風格指南：<https://style.tidyverse.org/>(<https://style.tidyverse.org/>)

# 馬上就能派上用場的內建函式

# 以 `rm()` 移除物件

```
In [5]: rm(lucky_number)  
lucky_number
```

```
Error in eval(expr, envir, enclos): object 'lucky_number' not found  
Traceback:
```

# 以 `help()` 查詢函式或資料的說明文件

```
In [6]: help(rm) # ?rm will do
```

```
In [7]: help(cars) # ?cars will do
```

以 `q()` 離開 RStudio, 不需要儲存工作空間圖案

# 使用 `class()` 得知向量的類型

```
In [8]: movie_title <- "The Dark Knight"  
movie_rating <- 9.0  
is_a_good_movie <- TRUE  
class(movie_title)  
class(movie_rating)  
class(is_a_good_movie)
```

'character'

'numeric'

'logical'

# numeric 是主要的數值向量

```
In [9]: class(5566)
class(55.66)
class(0)
class(-5566)
```

'numeric'

'numeric'

'numeric'

'numeric'



R 有非常豐富的內建函式，但這些函式不可能滿足我們所有需求

# 這時候我們會求助於「自訂函式」

顧名思義是由使用者定義的函式。

```
function_name <- function(INPUTS) {  
  # 函式主體的程式敘述...  
  # 將「輸入」轉換為「輸出」的規則以 R 程式表達...  
  return(OUTPUTS)  
}
```

# 自訂函式的組成：自由發揮的部分

- `function_name`: 自訂函式的名稱，由使用者自行取名，建議採用動詞。
- `INPUTS`: 輸入的變數名稱。
- `OUTPUTS`: 輸出的變數名稱。

## 自訂函式的組成：規範的部分

- `function`: 保留字，告知 R 這個物件是一個函式。
- `{}`: 定義「程式區塊」的範圍，區隔出輸入與輸出的可作用範圍。
- `return()`: 告知 R 這個函式的輸出變數為何。

# 能夠將公里轉換為英里函式：convert\_km\_to\_mile

$$\text{Miles} = \text{Kilometers} \times 0.62137$$

```
In [10]: convert_km_to_mile <- function(km) {  
  mile <- km * 0.62137  
  return(mile)  
}
```

# 定義完畢以後必須要「呼叫」才能產生作用

```
In [11]: convert_km_to_mile(42.195)
```

26.21870715

隨堂練習：林書豪（191cm / 91kg）的 BMI

$$BMI = \frac{weight_{kg}}{height_m^2}$$

# 文字向量



# 使用 ' ' 或 " " 宣告文字向量

```
In [12]: movie_title <- "Avengers: Endgame"  
class(movie_title)
```

'character'

```
In [13]: movie_title <- 'Avengers: Endgame'  
class(movie_title)
```

'character'

## 多數的時候使用 ' ' 或 " " 都沒有差異，不過...

```
In [14]: #shaq <- 'Shaquille O'Neal' # error  
shaq <- 'Shaquille O\'Neal' # \ is the escape symbol  
shaq <- "Shaquille O'Neal"
```

## 隨堂練習： What did Ross Geller say?

Let's put aside the fact that you "accidentally" pick up my grand mother's ring.

# 使用 `sprintf()` 函式進行在文字向量中嵌入變數

```
In [15]: user_name <- "Jovyan"  
         sprintf("Hello, %s!", user_name)
```

'Hello, Jovyan!'

# 邏輯運算符與邏輯值向量

## 邏輯運算符與邏輯值向量

# 邏輯運算符

- > 大於
- >= 大於等於
- < 小於
- <= 小於等於
- == 等於
- != 不等於
- %in% 屬於
- ! 非
- & 交集
- | 聯集

In [16]:

```
8 > 7  
8 >= 7  
8 < 7  
8 <= 7  
8 == 7  
8 !=7  
!(8 != 7)
```

TRUE

TRUE

FALSE

FALSE

FALSE

TRUE

FALSE

# R 程式設計對大小寫敏感 (case-sensitive)

In [17]:

```
class(TRUE)
class(FALSE)
#class(True) # error
#class(False) # error
#class(true) # error
#class(false) # error
```

'logical'

'logical'



# 我們將在未來應用邏輯值向量於三個用途

- 條件判斷
- while 迴圈
- 篩選資料