第 6 章: 函式導論

6: Introduction to Function

R 語言有許多 函式 (function), 函式是一種物件, 是指令的集合, 執行特定功能或運算工作的指令, 資料整理, 資料分析等, 並且可以用在其他的運算式中. 透過函式, 擴展了 R 在程式語言的功能性與便利性. R 基本系統 (base) 提供了一部分常用函式, 也提供一些基本統計計算函式, 例如, `sum()`, `mean()`, `var()` 等等; 而更多不同類型的函式, 則由許多不同的學者貢獻到 R 系統 (contribution) 中, 這些函數都是用 R 程式語言寫成的.

6.1 函式語法

在 R 中資料物件的算數操作, 許多時候會透過 函式 (function) 執行, 函式內通常需輸入 引數 (argument) 執行特定功能需求. 例如, 統計常用函式 `mean()`, `var()`, `sd()`, `log()` 等. 一個函式 A 內可能還會用到其他函式 B, C, 等等.

多數函式對資料物件進行計算產生結果, 會回傳一個 R 的物件, 有些函式用來進行特殊繪圖與列印. 一個函式內通常需輸入 引數 (argument) 或是 統計公式, 統計模型, (formals). 引數可以是一個以上, 有些引數一定要輸入, 稱為 必要引數 (required argument), 有些引數必須存在函式中, 但不一定必須由使用者輸入, 稱為 自選引數

(optional argument), 另外一種引數則不一定須要存在函式中, 稱為 省略引數 (ellipsis argument) 這 3 種引可以同時存在一個函式內, 引數可以是數值, 文字, 資料框架 或 R 的任何物件.

```
> x.vec = c(1:5)
> x.vec           # show x.vec
[1] 1 2 3 4 5
> mean(x = x.vec) # function mean() calculate mean, return a scalar
[1] 3
> var(x = x.vec)  # function mean() calculate variance
[1] 2.5
> sd(x.vec)       # function mean() calculate standard deviation
[1] 1.581139
> summary(x.vec)  # summarized statistics
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       2       3       3       4       5
> log(x = x.vec)  # take log for all elements in vector x.vec
[1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379
```

一個 函式 通常是通過下面類似的語句形式定義:

```
> function.name(arg.1, arg.2, arg3 = value.3, ...)
```

函式中的引數分別為:

- `arg.1, arg.2` 為 必要引數 (required argument).
  為使用者必須輸入引數值.
- `arg.3 = value.3` 為 選擇引數 (optional argument).
  選擇引數 附有一個 = (等號), 例如, `arg.3 = value.3`. 表示必須使用的引數 `arg.3`, 但不一定必須輸入到函式 `function.name()` 之內, 若使用者沒有輸入引數值, 則函式會直接使用 R 的內部自動設定的引數值 `value.3`.
- `...` 為 省略引數 (ellipsis argument).
  不一定須要存在函式中, 通常是將省略引數傳送到 函式 `function.name()` 內, 作為其他函式的引數.
- 函式 `function.name()` 運算的最終結果 (數值, 物件), 是函式回傳的物件.
- 函式 `args(function.name)` 可以查看函式的必要引數以及引數內部自動設

定值. 引數並沒有特定的型態, 任何物件都可能成為一個引數, 必要引數名字應該在寫在內部自動設定值引數之前.

例如, 函式 log() 指令:

```
> log(x, base = exp(1))
```

log() 函式在 R 內設以自然數為底計算. 其中引數:

- x 為必備引數, 使用者必須自行輸入所要計算的數值作為引數值.
- base 為選擇引數, 必須在函式中存在, 但不一定必須使用者輸入到函式 log() 之內.
- base = exp(1) 表示若使用者沒有輸入引數值, 則函式 log() 內設以自然數 e 為底.
- 當然, 使用者可以更動選擇引數的設定值, 輸入 base = 2, 使用者更動底數的設定值, 例如更動為以 2 為底的對對數運算, 使用 log(x, base = 2).

```
> ## log()
> x.vec <- c(1, 2, 3, 4, 5)
> log(x = x.vec)
[1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379
> log(x = x.vec, base = 2)
[1] 0.000000 1.000000 1.584963 2.000000 2.321928
> args(log)
function (x, base = exp(1))
NULL
```

R 函式的原始碼定義分成 S3 classes 函式 與 S4 classes 函式, 許多時候, 可以在 R 輸入函式名稱, 檢視 S3 classes 的函式. 例如, function.name(), 查看函式的原始碼內容與計算過程. 有時必須使用 methods("function.name"), getAnywhere("function.n stats:::function.name 例如, 指令提示符號下輸入 sd, 查看函式 sd 的計算過程與引數.

```
> ## methods()
> sd
function (x, na.rm = FALSE)
sqrt(var(if (is.vector(x)  is.factor(x)) x else as.double(x),
```

```
    na.rm = na.rm))
<bytecode: 0x000000000c3d25a0>
<environment: namespace:stats>
> t
function (x)
UseMethod("t")
<bytecode: 0x00000000027282b0>
<environment: namespace:base>
> methods("t")
[1] t.data.frame t.default    t.ts*
see '?methods' for accessing help and source code
> methods(class="ts")
 [1] [             [ <-            aggregate     as.data.frame cbind
 [6] coerce        cycle         diff          diffinv       initialize
[11] kernapply     lines         Math          Math2         monthplot
[16] na.omit       Ops           plot          print         show
[21] slotsFromS3   t             time          window        window <-
see '?methods' for accessing help and source code
```

其中 t.ts* 的 "*" 符號表示無法直接探索套件內的原始碼, 可以改用getAnywhere("function.na stats:::function.name 檢視.

若要檢視 S4 classes 的函式, 可以使用 showClass("function.namme"), showMethods("fun getMethod("function.namme"), selectMethod(), existsMethod(), hasMethod(), removeClass(), removeMethod(), getClass(), getSlots(), slotNames(), slot(). 另外一種查看函式的原始碼內容與計算過程, 是下載原始碼使用文書處理軟體打開檔案檢視.

```
> download.packages(pkgs = "package.name",
                    destdir = "C:/RData",
                    type = "source")
```

6.2 數列函式

R 有幾個產生數列的基本函式, 包含 :, sequence(), rep() 等.

6.2.1 數列向量: seq() 與 sequence()

在統計運算中, 常需要產生數列向量, 例如產生 [1,2,3,4,5], [1,3,5,7,9] 等等, 可以使用 函式 : (冒號), seq() 或 sequence() 等產生數列向量.

```
> ## :
> 1:5
[1] 1 2 3 4 5
> 5:1
[1] 5 4 3 2 1
> -1:3
[1] -1  0  1  2  3
```

數列函式 seq() 或 sequence() 的指令, 不受限只有整數, 可以任產生何實數數列.

```
> seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
      length.out = NULL, along.with = NULL, ...)
> seq.int(from, to, by, length.out, along.with, ...)
> seq_along(along.with)
> seq_len(length.out)
```

其中引數:

- from = 1 數列起始值
- to = 1 數列結束值
- by 數列每次增加值
- length.out 數列長度 (元素數目)

```
> # seq()
> seq(from=1, to=5, by = 0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> seq(1, 5, 0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> seq(1, 5, length = 3)
[1] 1 3 5
> seq(from=0, to=1, by = 0.1)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(from=0, to=1, by = 0.33)
[1] 0.00 0.33 0.66 0.99
> z.vec <- c("a", "b", "c", "d", "e")
```

```
> seq(along = z.vec)
[1] 1 2 3 4 5
> ## the concatenated sequences 1:3, 1:4, 1:5.
> sequence(c(3, 4, 5))
 [1] 1 2 3 1 2 3 4 1 2 3 4 5
> help(seq)
```

6.2.2 重複元素函式: rep()

另一個與 seq() 類似的函式 rep(), 可以產生相同重複元素的向量.

```
> rep(x, ...)
> rep(x, times = 1, length.out = NA, each = 1)
> rep.int(x, times)
>rep_len(x, length.out)
```

其中引數:

- x 使用者想要相同重複元素的數值向量.
- times 數值向量 x 重複的次數.
- each 數值向量 x 的每個元素重複的次數.
- length.out = NA 數值向量 x 重複後的長度.

```
> ## rep()
> rep(0, times = 3)
[1] 0 0 0
> rep(1, 5)
[1] 1 1 1 1 1
> x.vec <- c(4, 5, 6)
> rep(x.vec, times = 2)
[1] 4 5 6 4 5 6
> rep(x.vec, each = 2)
[1] 4 4 5 5 6 6
> rep(x.vec, each = 2, times = 3)
 [1] 4 4 5 5 6 6 4 4 5 5 6 6 4 4 5 5 6 6
> rep(x.vec, times = c(2, 2, 2))
[1] 4 4 5 5 6 6
> rep(x.vec, times = c(1, 2, 3))
[1] 4 5 5 6 6 6
> rep(x.vec, each = 2, len = 4)    # first 4 only.
[1] 4 4 5 5
```

```
> #
> y.vec <- c("A", "B", "C")
> y.vec
[1] "A" "B" "C"
> rep(y.vec, times = 2)
[1] "A" "B" "C" "A" "B" "C"
> rep(y.vec, each = 2)
[1] "A" "A" "B" "B" "C" "C"
> rep(y.vec, each = 2, times = 3)
 [1] "A" "A" "B" "B" "C" "C" "A" "A" "B" "B" "C"
[12] "C" "A" "A" "B" "B" "C" "C"
> rep(y.vec, times = c(2, 2, 2))
[1] "A" "A" "B" "B" "C" "C"
> rep(y.vec, times = c(1, 2, 3))
[1] "A" "B" "B" "C" "C" "C"
> rep(y.vec, each = 2, len = 4)    # first 4 only.
[1] "A" "A" "B" "B"
> #
> z.vec <- factor(LETTERS[1:3])
> names(z.vec) <- letters[1:3]
> z.vec
a b c
A B C
Levels: A B C
> rep(z.vec, 2)
a b c a b c
A B C A B C
Levels: A B C
> rep(z.vec, each = 2)
a a b b c c
A A B B C C
Levels: A B C
> rep.int(z.vec, 2)  # no names
[1] A B C A B C
Levels: A B C
> rep_len(z.vec, 10)
 [1] A B C A B C A B C A
Levels: A B C
> help(rep)
```

6.3 算數函式 Arithmetic Computing Function

R 有許多內建的 算數函式 (arithmetic function), 包含指數, 對數, Gamma 函數, Beta 函數, 三角函數, 反三角函數, 雙曲函數, 反雙曲函數 等等. 詳見表 **??** – 表 6.2.

```
> ## rounding
> (x.vec <- 0.5 + -2:2)
[1] -1.5 -0.5  0.5  1.5  2.5
> round(x.vec) # IEEE rounding
[1] -2  0  0  2  2
> (y.vec <- seq(-2, 2, by = 0.5))
[1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0
> (y.round <- round(y.vec)) # IEEE rounding
[1] -2 -2 -1  0  0  0  1  2  2
> (y.trunc <- trunc(y.vec))
[1] -2 -1 -1  0  0  0  1  1  2
> (y.signif <- signif(y.vec))
[1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0
> (y.ceil <- ceiling(y.vec))
[1] -2 -1 -1  0  0  1  1  2  2
> (y.floor <- floor(y.vec))
[1] -2 -2 -1 -1  0  0  1  1  2
> cbind(y.vec, y.round, y.trunc, y.signif, y.ceil, y.floor)
      y.vec y.round y.trunc y.signif y.ceil y.floor
 [1,]  -2.0      -2      -2     -2.0     -2      -2
 [2,]  -1.5      -2      -1     -1.5     -1      -2
 [3,]  -1.0      -1      -1     -1.0     -1      -1
 [4,]  -0.5       0       0     -0.5      0      -1
 [5,]   0.0       0       0      0.0      0       0
 [6,]   0.5       0       0      0.5      1       0
 [7,]   1.0       1       1      1.0      1       1
 [8,]   1.5       2       1      1.5      2       1
 [9,]   2.0       2       2      2.0      2       2
> #
> (x.vec <- 0.5 + -2:3)
[1] -1.5 -0.5  0.5  1.5  2.5  3.5
> round(x.vec) # IEEE rounding
[1] -2  0  0  2  2  4
> (y.vec <- seq(-2, 3, by = 0.5))
 [1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
> (y.round <- round(y.vec)) # IEEE rounding
 [1] -2 -2 -1  0  0  0  1  2  2  2  3
> (y.trunc <- trunc(y.vec))
 [1] -2 -1 -1  0  0  0  1  1  2  2  3
> (y.signif <- signif(y.vec))
 [1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
> (y.ceil <- ceiling(y.vec))
 [1] -2 -1 -1  0  0  1  1  2  2  3  3
> (y.floor <- floor(y.vec))
 [1] -2 -2 -1 -1  0  0  1  1  2  2  3
> cbind(y.vec, y.round, y.trunc, y.signif, y.ceil, y.floor)
      y.vec y.round y.trunc y.signif y.ceil y.floor
 [1,]  -2.0      -2      -2     -2.0     -2      -2
 [2,]  -1.5      -2      -1     -1.5     -1      -2
 [3,]  -1.0      -1      -1     -1.0     -1      -1
```

```
 [4,]  -0.5       0       0     -0.5       0      -1
 [5,]   0.0       0       0      0.0       0       0
 [6,]   0.5       0       0      0.5       1       0
 [7,]   1.0       1       1      1.0       1       1
 [8,]   1.5       2       1      1.5       2       1
 [9,]   2.0       2       2      2.0       2       2
[10,]   2.5       2       2      2.5       3       2
[11,]   3.0       3       3      3.0       3       3
> #
> (y.vec <- seq(-2, 3, by = 0.5))
 [1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
> y.vec[trunc(y.vec) != floor(y.vec)]
[1] -1.5 -0.5
> y.vec[round(y.vec) != floor(y.vec + 0.5)]
[1] -1.5  0.5  2.5
> #
> (z.vec <- pi * 100^(-1:3))
[1]       0.03142       3.14159     314.15927   31415.92654 3141592.65359
> round(z.vec, 3)
[1]       0.031       3.142     314.159   31415.927 3141592.654
> signif(z.vec, 3)
[1]       0.0314       3.1400     314.0000   31400.0000 3140000.0000
> ## sign() abs()
> sign(pi) # == 1
[1] 1
> sign(-2:3)# -1 -1 0 1 1 1
[1] -1 -1  0  1  1  1
> abs(-2:3)
[1] 2 1 0 1 2 3
> #
> ## log(), exp() calculation
> (x.vec <- 1:3)
[1] 1 2 3
> log(exp(x.vec))
[1] 1 2 3
> (y.vec <- 10^(x.vec))
[1]   10  100 1000
> log10(y.vec)
[1] 1 2 3
> log10(1e7) # = 7
[1] 7
> ## options(digits, scipen)
> options(digits = 4, scipen = 0)
> z.vec <- pi*100^(-1:3)
> print(z.vec / 1000, digits = 4)
[1] 3.142e-05 3.142e-03 3.142e-01 3.142e+01 3.142e+03
> options(digits = 4, scipen = 100)
> print(z.vec/1000, digits = 4)
[1]    0.00003142    0.00314159    0.31415927   31.41592654 3141.59265359
> #
```

```
> options(digits = 4, scipen = 100)
> x.vec <- 100^-(1+2*1:3)
> cbind(x = x.vec,
       log1px = log(1+x.vec),
       log1p  = log1p(x.vec),
       exp    = exp(x.vec)-1,
       expm1  = expm1(x.vec))
                     x               log1px              log1p
[1,] 0.00000100000000 0.000000999999499918 0.00000099999950
[2,] 0.00000000010000 0.000000000100000008 0.00000000010000
[3,] 0.00000000000001 0.000000000000009992 0.00000000000001
                       exp             expm1
[1,] 0.000001000000499962 0.00000100000050
[2,] 0.000000000100000008 0.00000000010000
[3,] 0.000000000000009992 0.00000000000001
> #
> options(digits = 4, scipen = 0)
> x.vec <- 100^-(1+2*1:3)
> cbind(x = x.vec,
        log1px = log(1+x.vec),
        log1p  = log1p(x.vec),
        exp    = exp(x.vec)-1,
        expm1  = expm1(x.vec))
         x    log1px log1p       exp expm1
[1,] 1e-06 1.000e-06 1e-06 1.000e-06 1e-06
[2,] 1e-10 1.000e-10 1e-10 1.000e-10 1e-10
[3,] 1e-14 9.992e-15 1e-14 9.992e-15 1e-14
```

表 6.1: 常見數學函式 I

符號	定義
`-`	減法運算 (Substraction, can be unary or binary)
`+`	加法運算 (Addition, can be unary or binary)
`!`	否定運算 (Unary not)
`*`	乘法運算 (Multiplication, binary)
`/`	除法運算 (Division, binary)
`^`	指數乘冪運算 (Exponentiation, binary)
`%%`	整數除法的餘數 (Modulus, binary)
`%/%`	整數除法的商數 (Integer divide, binary)
`%*%`	矩陣內積乘法 (Matrix product, binary)
`%o%`	矩陣外積乘法 (Outer product, binary)
`%x%`	矩陣 Kronecker 乘法 (Kronecker product, binary)
`%in%`	配對運算 (Matching operator, binary)
	(In model formulae: nesting)
`round(x, digits = 0)`	設定小數位數 (數值會受到作業系統影響)
	(its first argument to the specified number of decimal places)
`signif(x, digits = 6)`	設定實數列印出的小數位數 (數值會受到作業系統影響)
	(rounds the values to the specified number of significant digits)
`trunc(x)`	將 `x` 的小數截斷, 向 0 靠近
	(the integers by truncating `'x'` toward '0')
`ceiling(x)`	大於 `x` 的最小整數
	(the smallest integers not less than `'x'`)
`floor(x)`	小於 `x` 的最大整數
	(the largest integers not greater than `'x'`)
`sign(x)`	求 `x` 的正負值, 回傳 1, 0, 或 -1.
	($sign(x)$, the sign of a real number is 1, 0, or -1
	if the number is positive, zero, or negative, respectively.)
`abs(x)`	求 `x` 的絕對值
	(|x|, absolute value of `x`)

表 6.2: 常見數學函式 II

函式	說明
`sqrt(x)`	$\sqrt{x}$
`exp(x)`	e^x
`expm1(x)`	當 $\|x\| << 1$, 計算 $e^x - 1$
	(computes $\exp(x) - 1$ accurately also for $\|x\| << 1$.)
`log(x)`	$\log(x)$
`log10(x)`	$log_{10}(x)$
`log2(x)`	$\log_2(x)$
`logb(x, base = z)`	$\log_z(x)$
`log1p(x)`	當 $\|x\| << 1$, 計算 $\log(1+x)$
	(computes $\log(1+x)$ accurately also for $\|x\| << 1$.)
`gamma(x)`	$\Gamma(x) = (x-1)! = \int_0^\infty t^{(x-1)} \exp(-t) dt$
`lgamma(x)`	$\log_e[\Gamma(x)]$
`beta(a, b)`	$B(a,b) = (\Gamma(a)\Gamma(b)) \,/\, (\Gamma(a+b))$
	$= \int_0^1 t^{(a-1)}(1-t)^{(b-1)} dt$
`lbeta(a, b)`	$\log_e[B(a,b)]$
`digamma(x)`	$\frac{d}{dx} \log_e[\Gamma(x)]$
`trigamma(x)`	$\frac{d^2}{dx^2} \log_e[\Gamma(x)]$
`psigamma(x, deriv = 0)`	$\frac{d^p}{dx^p} \log_e[\Gamma(x)]$
`sin(x) cos(x) tan(x)`	三角函數 (trigonometric functions)
`asin(x) acos(x) atan(x)`	反三角函數 (inverse functions)
`sinh(x) cosh(x) tanh(x)`	雙曲函數 (hyperbolic functionsx)
`asinh(x) acosh(x) atanh(x)`	反雙曲函數 (inverse hyperbolic functions)

測驗 **1.** R 的數學函式中包含 `zapsmall()`, 請說明其用法, 並舉例說明.

在數值得有效位數與科學符號上, R 會用到 `options(digits, scipen)` 指令, 請說明其用法, 並舉例說明.

```
> ## zapsmall()
> zapsmall(z.vec/1000, digits = 4)
[1]    0.0    0.0    0.3   31.4 3141.6
> zapsmall(exp(1i*0:4*pi/2))
```

```
[1]  1+0i  0+1i -1+0i  0-1i  1+0i
> #
> ## trigonometric function
> (x.tri <- c(0, 1/4, 1/3,     1/2, 1,
+              1+1/4, 1+1/3, 1+1/2, 2,
+              2+1/4, 2+1/3, 2+1/2, 3,
+              3+1/4, 3+1/3, 3+1/2, 4)*pi)
 [1]  0.0000000  0.7853982  1.0471976  1.5707963  3.1415927  3.9269908  4.1887902
    4.7123890
 [9]  6.2831853  7.0685835  7.3303829  7.8539816  9.4247780 10.2101761 10.4719755
    10.9955743
[17] 12.5663706
>
> sin(x.tri)
 [1]  0.000000e+00  7.071068e-01  8.660254e-01  1.000000e+00  1.224606e-16 -7.071068e-01
 [7] -8.660254e-01 -1.000000e+00 -2.449213e-16  7.071068e-01  8.660254e-01  1.000000e+00
[13]  3.673819e-16 -7.071068e-01 -8.660254e-01 -1.000000e+00 -4.898425e-16
> asin(sin(x.tri))
 [1]  0.000000e+00  7.853982e-01  1.047198e+00  1.570796e+00  1.224606e-16 -7.853982e-01
 [7] -1.047198e+00 -1.570796e+00 -2.449213e-16  7.853982e-01  1.047198e+00  1.570796e+00
[13]  3.673819e-16 -7.853982e-01 -1.047198e+00 -1.570796e+00 -4.898425e-16
> cos(x.tri)
 [1]  1.000000e+00  7.071068e-01  5.000000e-01  6.123032e-17 -1.000000e+00 -7.071068e-01
 [7] -5.000000e-01 -1.836910e-16  1.000000e+00  7.071068e-01  5.000000e-01  3.061516e-16
[13] -1.000000e+00 -7.071068e-01 -5.000000e-01 -4.286122e-16  1.000000e+00
> tan(x.tri)
 [1]  0.000000e+00  1.000000e+00  1.732051e+00  1.633124e+16 -1.224647e-16  1.000000e+00
 [7]  1.732051e+00  5.443746e+15 -2.449294e-16  1.000000e+00  1.732051e+00  3.266248e+15
[13] -3.673940e-16  1.000000e+00  1.732051e+00  2.333034e+15 -4.898587e-16
> cbind(x.tri, sinx = sin(x.tri), asin = asin(sin(x.tri)),
+       cosx = cos(x.tri), tan(x.tri))
           x.tri          sinx          asin          cosx
 [1,]  0.0000000  0.000000e+00  0.000000e+00  1.000000e+00  0.000000e+00
 [2,]  0.7853982  7.071068e-01  7.853982e-01  7.071068e-01  1.000000e+00
 [3,]  1.0471976  8.660254e-01  1.047198e+00  5.000000e-01  1.732051e+00
 [4,]  1.5707963  1.000000e+00  1.570796e+00  6.123032e-17  1.633124e+16
 [5,]  3.1415927  1.224606e-16  1.224606e-16 -1.000000e+00 -1.224647e-16
 [6,]  3.9269908 -7.071068e-01 -7.853982e-01 -7.071068e-01  1.000000e+00
 [7,]  4.1887902 -8.660254e-01 -1.047198e+00 -5.000000e-01  1.732051e+00
 [8,]  4.7123890 -1.000000e+00 -1.570796e+00 -1.836910e-16  5.443746e+15
 [9,]  6.2831853 -2.449213e-16 -2.449213e-16  1.000000e+00 -2.449294e-16
[10,]  7.0685835  7.071068e-01  7.853982e-01  7.071068e-01  1.000000e+00
[11,]  7.3303829  8.660254e-01  1.047198e+00  5.000000e-01  1.732051e+00
[12,]  7.8539816  1.000000e+00  1.570796e+00  3.061516e-16  3.266248e+15
[13,]  9.4247780  3.673819e-16  3.673819e-16 -1.000000e+00 -3.673940e-16
[14,] 10.2101761 -7.071068e-01 -7.853982e-01 -7.071068e-01  1.000000e+00
[15,] 10.4719755 -8.660254e-01 -1.047198e+00 -5.000000e-01  1.732051e+00
[16,] 10.9955743 -1.000000e+00 -1.570796e+00 -4.286122e-16  2.333034e+15
[17,] 12.5663706 -4.898425e-16 -4.898425e-16  1.000000e+00 -4.898587e-16
```

6.4 組合與階乘函式: choose() 與 factorial()

R 內建的組合與階乘函式 choose(),lchoose(),factorial(),lfactorial(), 其用法類似.

$$\texttt{choose(n, k)} = \binom{n}{k} \tag{6.4.1}$$

$$\texttt{fractorial(x)} = x!. \tag{6.4.2}$$

```
> choose(n, k)
> lchoose(n, k)
> factorial(x)
> lfactorial(x)
```

其中

- k 為正整數.
- x 與 n 為數值向量.
- choose() 與 factorial() 回傳原始結果. lchoose() 與 lfactorial() 回傳原始結果取對數值.

```
> ## combination
> ## choose()
> choose(n = 5, k = 2)
[1] 10
> log(choose(n = 5, k = 2))
[1] 2.303
> lchoose(n = 5, k = 2)
[1] 2.303
> for (n in 0:5) print(choose(n, k = 0:n))
[1] 1
[1] 1 1
[1] 1 2 1
[1] 1 3 3 1
[1] 1 4 6 4 1
[1]  1  5 10 10  5  1
>
> ## factorial
> factorial(x = 100)
[1] 9.333e+157
```

```
> log(factorial(x = 100))
[1] 363.7
> lfactorial(x = 100)
[1] 363.7
> lfactorial(x = 10000)
[1] 82109
> factorial(x = c(1, 3, 5))
[1]   1   6 120
```

表 6.3: 常見排列組合函式

函式	說明
`choose(n, k)`	$\frac{n!}{k!\,(n-k)!}$
`lchoose(n, k)`	$\log_e(\texttt{choose(n, k)})$
`factorial(x)`	$x! = \Gamma(x+1)$
`lfactorial(x)`	$log(x!) = \log_e[\Gamma(x)]$

6.5 選擇資料函式: all(), any(), which()

函式 `all(x)` 與 `any(x)` 可以用來查看特定向量物件 `obj.vec` 的元素是否符合某些特定條件, 回傳邏輯純量 `TRUE` 或 `FALSE`. 函式 `which()` 查看特定向量物件 `obj.vec` 的元素是否符合某些特定條件, 然後回傳一個下標向量, 紀錄向量物件符合某些特定條件的元素位置.

(A):

```
> all(..., na.rm = FALSE)
> any(..., na.rm = FALSE)
```

其中 `...` 為輸入向量物件的條件判斷. `all(x)` 與 `any(x)` 回傳一個純量 (scalar) 代表 `TRUE` 或 `FALSE`.

令一個 邏輯向量 **x** (x), 使用函式 `all(x)` 用來檢查 邏輯向量 **x** 的所有元素是否都是 `TRUE`? 另一方面, 使用函式 `any(x)` 用來檢查 邏輯向量 **x** 的任一元素是否為

TRUE?

(B):

```
> which(x, arr.ind = FALSE, useNames = TRUE)
> arrayInd(ind, .dim, .dimnames = NULL, useNames = FALSE)
```

令一個 邏輯向量 **x** (x), 使用函式指令 which(x) 則回傳一個向量, 是邏輯向量 **x** 中的元素為 TRUE 所在的下標位置 (index). 引數 arr.ind = TRUE 是當 **x** 為 array 時, 回傳 array 的下標位置.

(C):

```
> which.max(x)
> which.min(x)
```

函式指令 which.max(x) 與 which.min(x) 是函式指令 which(x) 的延伸. 令一個數值向量 **x** (x), which.max(x) 與 which.min(x) 回傳數值向量物件 **x** (x) 內元素的最大值 (最小值) 所在的下標位置 (index).

```
> ## all(), any(), which()
> (x.vec <- c(-1:2))
[1] -1  0  1  2
> all(x.vec > 0)
[1] FALSE
> any(x.vec > 0)
[1] TRUE
> which(x.vec > 0)
[1] 3 4
> which.max(x.vec)
[1] 4
> which.min(x.vec)
[1] 1
> #
> (x.mat <- matrix(c( 2, -1, -3,
                      -1,  2,  4,
                      -3,  4,  9),
                    nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    2   -1   -3
[2,]   -1    2    4
[3,]   -3    4    9
> all(x.mat > 0)
[1] FALSE
> any(x.mat > 0)
[1] TRUE
> which(x.mat > 0)
```

```
[1] 1 5 6 8 9
> #
> which(x.mat%%2 == 0)
[1] 1 5 6 8
> which(x.mat%%2 == 0, arr.ind = TRUE)
     row col
[1,]   1   1
[2,]   2   2
[3,]   3   2
[4,]   2   3
> rownames(x.mat) <- paste("Case", 1:3, sep = "_")
> which(x.mat%%2 == 0, arr.ind = TRUE)
       row col
Case_1   1   1
Case_2   2   2
Case_3   3   2
Case_2   2   3
```

6.6 排序函式 Ranking and Sorting

在 R 中有數個與排序相關的函式, 如 rev(), sort(), order() 與 rank(), 表 6.4 摘要常見的常見排序函式.

```
> rev(x)
> sort(x, decreasing = FALSE, na.last = NA, ...)
> rank(x, na.last = TRUE,
       ties.method = c("average", "first", "last", "random", "max", "min"))
> order(x, ..., na.last = TRUE, decreasing = FALSE,
        method = c("shell", "radix"))
> sort.int(x, partial = NULL, na.last = NA, decreasing = FALSE,
           method = c("shell", "quick", "radix"), index.return = FALSE)
```

這些函式的共用引數為:

- 引數 x 為向量 **x**.
- 引數 decreasing:
  - decreasing = FALSE 為 R 內設從小到大排序.
  - decreasing = TRUE 則從大到小排序.

- 引數 na.last:
  - na.last = TRUE 為 R 內設將 NA 排序在最大.
  - na.last = FALSE 為 R 內設將 NA 排序在最小.
  - na.last = NA 為 R 內設將 NA 排除.
- 函式 rev(x) 回傳一個向量 **z**, 是將向量 **x** 元素反轉.
- 函式 sort(x) 回傳一個向量 **z**, 是將向量 **x** 的元素從小到大排序的結果列出.
- 函式 rank(x) 回傳一個向量 **z**, 是將向量 **x** 每一個元素從小到大排序之後, **x** 元素之相對順序 (rank).
  - "average": 相同數值都是回傳平均排序值.
  - "first": 相同數值都是回傳依序由小到大不同排序值.
  - "random": 相同數值回傳隨機排序值.
  - "max": 相同數值回傳最大排序值.
  - "min": 相同數值回傳最小排序值.
- 函式 order(x) 回傳一個向量 **z**, 是將向量 **x** 從小到大排序後的向量之元素, 在原來向量 **x** 的原始位置.

```
> ## reverse, rank, sort and order
> ## rev(): reverse elements
> x.vec <- c(7, 7, 7, 6, 10, 9, 9, 9, NA, 8)
> rev(x.vec)
 [1]  8 NA  9  9  9 10  6  7  7  7
> #
> ## sort(): from the smallest to the largest
> sort(x.vec)
[1]  6  7  7  7  8  9  9  9 10
> #
> ## rank():
> rank(x.vec, na.last = TRUE)
 [1]  3  3  3  1  9  7  7  7 10  5
> rank(x.vec, na.last = FALSE)
 [1]  4  4  4  2 10  8  8  8  1  6
```

```
> set.seed(1)
> rank(x.vec, ties.method = "average")
 [1]  3  3  3  1  9  7  7  7 10  5
> rank(x.vec, ties.method = "first")
 [1]  2  3  4  1  9  6  7  8 10  5
> rank(x.vec, ties.method = "last")
 [1]  4  3  2  1  9  8  7  6 10  5
> rank(x.vec, ties.method = "random")
 [1]  2  3  4  1  9  7  8  6 10  5
> rank(x.vec, ties.method = "max")
 [1]  4  4  4  1  9  8  8  8 10  5
> rank(x.vec, ties.method = "min")
 [1]  2  2  2  1  9  6  6  6 10  5
> #
> ## order(): retrun index
> ## x.vec[] is the smallest one
> order(x.vec)
 [1]  4  1  2  3 10  6  7  8  5  9
> x.vec[order(x.vec)]
 [1]  6  7  7  7  8  9  9  9 10 NA
```

當向量內元數有相同的數值時, 在函式 `rank()` 內的引數 `ties.method`, 可以輸入各種相同的數值時排序的選項, 平均排序值可能出現分數或小數.

```
> ## rank(): ties.method = "average"
> x <- c(7, 9, 6, 7, 8, NA)
> sort(x, na.last = FALSE)
[1] NA  6  7  7  8  9
> rank(x, ties.method = "average", na.last = TRUE)
[1] 2.5 5.0 1.0 2.5 4.0 6.0
> (x.ord <- order(x, na.last = FALSE))
[1] 6 3 1 4 5 2
> x[x.ord] # = sort(x)
[1] NA  6  7  7  8  9
```

在統計分析中, 常須對矩陣或資料框架中的某些變數做排序, 可以利用 `order()`.

```
> ## matrix order()
> (x <- c(1, 1, 3:1, 1:4, 3))
 [1] 1 1 3 2 1 1 2 3 4 3
> (y <- c(9, 9:1))
 [1] 9 9 8 7 6 5 4 3 2 1
> (z <- c(2, 1:9))
 [1] 2 1 2 3 4 5 6 7 8 9
> id = 1:10
> (xyz.mat <- cbind(id, x, y, z))
      id x y z
 [1,]  1 1 9 2
 [2,]  2 1 9 1
```

```
 [3,]  3 3 8 2
 [4,]  4 2 7 3
 [5,]  5 1 6 4
 [6,]  6 1 5 5
 [7,]  7 2 4 6
 [8,]  8 3 3 7
 [9,]  9 4 2 8
[10,] 10 3 1 9
> #
> xyz.mat[order(x), ]    # sort by x
      id x y z
 [1,]  1 1 9 2
 [2,]  2 1 9 1
 [3,]  5 1 6 4
 [4,]  6 1 5 5
 [5,]  4 2 7 3
 [6,]  7 2 4 6
 [7,]  3 3 8 2
 [8,]  8 3 3 7
 [9,] 10 3 1 9
[10,]  9 4 2 8
> xyz.mat[order(x, y), ] # sort by x, y
      id x y z
 [1,]  6 1 5 5
 [2,]  5 1 6 4
 [3,]  1 1 9 2
 [4,]  2 1 9 1
 [5,]  7 2 4 6
 [6,]  4 2 7 3
 [7,] 10 3 1 9
 [8,]  8 3 3 7
 [9,]  3 3 8 2
[10,]  9 4 2 8
> (xyz.order <- order(x, y, z)) # sort by x, y, z sequentially
 [1]  6  5  2  1  7  4 10  8  3  9
> xyz.mat[xyz.order, ] # reordering (ties via 2nd & 3rd arg)
      id x y z
 [1,]  6 1 5 5
 [2,]  5 1 6 4
 [3,]  2 1 9 1
 [4,]  1 1 9 2
 [5,]  7 2 4 6
 [6,]  4 2 7 3
 [7,] 10 3 1 9
 [8,]  8 3 3 7
 [9,]  3 3 8 2
[10,]  9 4 2 8
> ## sort data frame
> (xyz.df <- data.frame(id, x, y, z))
   id x y z
```

```
1  1   1 1 9 2
2  2   2 1 9 1
3  3   3 3 8 2
4  4   4 2 7 3
5  5   5 1 6 4
6  6   6 1 5 5
7  7   7 2 4 6
8  8   8 3 3 7
9  9   9 4 2 8
10 10 10 3 1 9
> xyz.df[order(xyz.df$x, -xyz.df$y, xyz.df$z), ]
   id x y z
2   2 1 9 1
1   1 1 9 2
5   5 1 6 4
6   6 1 5 5
4   4 2 7 3
7   7 2 4 6
3   3 3 8 2
8   8 3 3 7
10 10 3 1 9
9   9 4 2 8
```

R 排序對缺失值 NA 處理要非常小心, 引數 `na.last = TRUE` 為 R 內設, 但在比較關係上, 對缺失值 NA 可有不同處理.

```
> ## tests of na.last
> x <- c(4, 3, 2, NA, 1)
> y <- c(8, NA, 7, 9, 6)
> id <- c(1:5)
> (z <- cbind(id, x, y))
     id  x  y
[1,]  1  4  8
[2,]  2  3 NA
[3,]  3  2  7
[4,]  4 NA  9
[5,]  5  1  6
> (o <- order(x, y))
[1] 5 3 2 1 4
> z[o, ]
     id  x  y
[1,]  5  1  6
[2,]  3  2  7
[3,]  2  3 NA
[4,]  1  4  8
[5,]  4 NA  9
> (o <- order(x, y, na.last = FALSE))
[1] 4 5 3 2 1
> z[o, ]
```

```
    id  x  y
[1,]  4 NA  9
[2,]  5  1  6
[3,]  3  2  7
[4,]  2  3 NA
[5,]  1  4  8
> (o <- order(x, y, na.last = NA))
[1] 5 3 1
> z[o, ]
    id x y
[1,]  5 1 6
[2,]  3 2 7
[3,]  1 4 8
> #
> ## rearrange matched vectors
> ## so that the first is in ascending order
> (x <- c(5:1, 6:8, 12:9))
 [1]  5  4  3  2  1  6  7  8 12 11 10  9
> (y <- (x-5)^2)
 [1]  0  1  4  9 16  1  4  9 49 36 25 16
> (z.o <- order(x))
 [1]  5  4  3  2  1  6  7  8 12 11 10  9
> rbind(x[z.o], y[z.o])
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    1    2    3    4    5    6    7    8    9    10    11    12
[2,]   16    9    4    1    0    1    4    9   16    25    36    49
> (z.mat <- rbind(x, y))
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
x    5    4    3    2    1    6    7    8   12    11    10     9
y    0    1    4    9   16    1    4    9   49    36    25    16
> z.mat[ , order(x, y)]
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
x    1    2    3    4    5    6    7    8    9    10    11    12
y   16    9    4    1    0    1    4    9   16    25    36    49
```

表 6.4: 常見之排序函式

函式	說明
`rev(x)`	將向量 **x** 元素反轉列出 (reverse order)
`rank(x)`	向量 **x** 元素的對應排序 (returns the sample ranks of the values)
	相同數值內設排序方式 `ties.method = "average"`
`sort(x)`	向量 **x** 從小到大列出 (sort a vector or factor, partially)
	into ascending or descending order).
`order(x)`	將向量 **x** 從小到大排序後的向量之元素
	(returns a permutation
	which rearranges its first argument into ascending
	or descending order, breaking ties by further arguments.)

6.7 物件查看與強制轉換函式

R 的許多函式型式為 `is.object()`, 例如函式 `is.na()`, `is.vector()` 等, 可以用來查看某一特定物件是否屬於某一類別.

R 的許多函式型式為 `as.object()`, 例如函式 `as.vector()`, `as.matrix(()` 等, 可以用來某強制轉換一特定物件到所指定的物件類別, 參見表 6.5, 並詳見輔助文件.

```
> ## is() and as()
> # vector
> x.vec <- c(1/1, 1/2, 1/3, 1/4, 1/5)
> x.vec
[1] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000
> is.vector(x.vec)
[1] TRUE
> is.character(x.vec)
[1] FALSE
> #
> y.vec <- c(1:5)
> y.vec
[1] 1 2 3 4 5
> is.vector(y.vec)
[1] TRUE
> is.character(y.vec)
[1] FALSE
```

```
> as.character(y.vec)
[1] "1" "2" "3" "4" "5"
> #
> x.vec <- c(1/1, 1/2, 1/3, 1/4, 1/5)
> x.vec <- as.character(x.vec)
> x.vec
[1] "1"                 "0.5"               "0.333333333333333"
[4] "0.25"              "0.2"
>
> ##
> b.df <- as.data.frame(matrix(c(1:24), nrow = 6, byrow = T))
> is.matrix(b.df)
[1] FALSE
> b.mat <- as.matrix(b.df)
> b.mat
     V1 V2 V3 V4
[1,]  1  2  3  4
[2,]  5  6  7  8
[3,]  9 10 11 12
[4,] 13 14 15 16
[5,] 17 18 19 20
[6,] 21 22 23 24
> b.mat <- as.vector(b.mat)
> b.mat
 [1]  1  5  9 13 17 21  2  6 10 14 18 22  3  7 11 15 19 23  4  8 12 16 20 24
```

表 6.5: 物件查看與強制轉換函式

物件查看函式	強制轉換函式
`is.na()`	`n/a`
`is.nan()`	`n/a`
`is.infinite()`	`n/a`
`is.null()`	`as.null()`
`is.numeric()`	`as.numeric()`
`is.integer()`	`as.integer()`
`is.character()`	`as.character()`
`is.logical()`	`as.logical()`
`is.complex()`	`as.complex()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.array()`	`as.array()`
`is.list()`	`as.list()`
`is.data.frame()`	`as.data.frame()`
`is.factor()`	`as.factor()`
`is.ordered()`	`as.ordered()`
`is.table()`	`as.table()`
`is.function()`	`as.function()`